Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Reporte de Proyecto Tecnológico

Asistente para la generación de código para procedimientos almacenados en un manejador de bases de datos

Versión 1.0

Modalidad: Proyecto Tecnológico

Alumno:

Edgar Ivan Hernández Pérez
209200073

Trimestre 14 Primavera
29 agosto de 2014

Asesor:

Oscar Herrera Alcántara
Profesor Asociado
Departamento de Sistemas

Yo, Oscar Herrera Alcántara, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Oscar Herrera Alcántara
Profesor Asociado
Departamento de Sistemas

Yo, Edgar Ivan Hernández Pérez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Edgar Ivan Hernández Pérez
209200073

El usuario se compromete a utilizar el contenido del presente Proyecto de Integración de manera acorde al uso para el cual fue diseñado.

El usuario queda obligado a no utilizar el contenido del presente Proyecto de Integración (reporte y código fuente), con fines o efectos ilícitos o lesivos de derechos y/o intereses del alumno y el asesor o de terceros que, de cualquier forma, puedan dañar el normal funcionamiento del producto generado del presente Proyecto de Integración.

El uso del contenido del presente proyecto de integración se realizará bajo la única y exclusiva responsabilidad del usuario, quedando exonerado, expresamente, el alumno y el asesor de los daños y/o perjuicios que en dado caso ocurrieran al usuario o a terceros por dicho uso en contra de la finalidad del presente Proyecto de Integración.

Respecto a los contenidos del presente Proyecto de Integración (información, textos, gráficos, imágenes, diseños, archivos de código fuente, etc.) se prohíbe:

- Su reproducción, copia, distribución, difusión, comunicación pública, transformación o modificación, a menos que se cuente con la autorización del alumno o del asesor (de preferencia ambos en acuerdo) o resulte legalmente permitido (como en la presente).
- Cualquier vulneración de los derechos del alumno o asesor del presente Proyecto de Integración.
- Su utilización para todo tipo de fines comerciales o publicitarios, distintos de los estrictamente permitidos.

Las marcas, íconos, imágenes y logos mencionados son propiedad de sus respectivos dueños.

Las marcas y productos mencionados son propiedad de sus respectivos dueños. El alumno y el asesor del presente Proyecto de Integración, no ofrecen soporte técnico de programas de terceros ni se hace responsable por el daño que la instalación de dicho software pueda ocasionar.

Resumen

En el presente Proyecto de Integración, se propuso diseñar e implementar una herramienta CASE (de las siglas en inglés de Computer Aided Software Engineering), para la generación de código SQL asociado a la creación de procedimientos almacenados en un determinado Sistema Gestor de Bases de Datos (SGBD, también conocido como manejador de bases de datos), en este caso específico del SGBD MySQL. Cumpliendo principalmente con las siguientes funciones: Generar procedimientos almacenados con los parámetros, y ajustes proporcionados por el usuario mediante la GUI (de las siglas en inglés de Graphic User Interface) de la herramienta CASE; así como poder visualizar el anidamiento[1] relacionado con las invocaciones dentro de un procedimiento almacenado. Generar automáticamente procedimientos almacenados que implementen las funciones básicas en bases de datos (Altas, Bajas, Cambios y Consultas de registros) para una determinada entidad.

La herramienta CASE obtenida, permite generar los procedimientos almacenados descritos anteriormente y se ejecutan de manera satisfactoria en el SGBD MySQL.

[1] anidamiento (llamado *nesting* en inglés) es la práctica de incorporar llamadas (*calls*) a funciones o procedimientos (unas) dentro de otras. [1]

Agradecimientos

Quisiera comenzar agradeciendo a todos los que hacen posible que nuestra casa de estudios funcione, desde el personal, hasta los alumnos que conformamos la comunidad estudiantil, ya que todos somos UAM y aportamos un granito de arena para tener una mejor universidad.

De manera muy especial agradezco al Dr. Oscar Herrera Alcántara, por compartir sus conocimientos y visiones, necesarios para poder traer a la realidad este proyecto.

Al Centro de Consulta de Ingeniería Física (CCIF) y a sus fundadores, ya que hicieron posible contar con un espacio, para poder continuar con nuestras jornadas de estudio.

A mi familia, por aquellos momentos que nos privamos para estar juntos, con el objetivo de poder concluir con mis estudios.

A mis amigos, con los cuales no pude estar presente en algunos momentos, por alcanzar este fin.

Y por último a aquellas personas que se cruzaron en mi camino y que de alguna manera formaron parte de este proyecto.

Dedicatoria

> *"Las demás personas no son tus enemigos, son competidores que al igual que tú, tienen sus propios motivos y razones por las cuales están presentes en esta carrera de la vida. Hay que aprender esa lección y saber aceptar con humildad, tanto la derrota como la victoria."*[2]

Dedico este proyecto a mi familia y a mis amigos. Por todos los esfuerzos que hicieron por mí, y por esas veces que dejaron de suplirse sus propias necesidades por suplir las mías. A mis hermanos, que forman parte de mi vida y que son mi motor para seguir adelante. A mi madre, por qué no fue solo eso, sino también supo ser padre en esos momentos donde él estuvo lejos. Por aquellos que creyeron en mí y depositaron su confianza. Y por último de manera muy especial a la memoria de mi padre, que aunque Dios quiso llevárselo antes de que pudiera estar presente en otros de mis logros, en honor a él, he duplicado mis esfuerzos, para que nuestros sueños y metas que teníamos planeadas en familia se hagan realidad.

[2] inspirada en la frase original "En una competición deportiva no hay enemigos solo contrincantes, tienes que aprender esa lección y aceptar la derrota y eso es lo más difícil" - Águila roja, Primer temporada, Capitulo 13

Índice general

Índice de figuras

Índice de tablas

Índice de espacios de nombres

Paquetes

Índice jerárquico

Jerarquía de clases

Índice de clases

Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Índice de archivos

Lista de archivos

Lista de todos los archivos con descripciones breves:

Introducción

En las últimas dos décadas los avances en la investigación en bases de datos han permitido el desarrollo de la industria de servicios de información [2]. Tal ha sido la importancia de las bases de datos que hoy en día forman parte de nuestra vida cotidiana, aunque a veces no somos conscientes de su uso y aplicación.

Un Sistema Gestor de Bases de Datos (SGBD) administra y controla el acceso a la base de datos a través de un conjunto de programas. Una manera de modelar los datos en un SGBD es mediante el así llamado Modelo Relacional [3] que tiene un fundamento matemático robusto, lo cual le ha permitido ser el dominante no solo en la parte científica sino también en aplicaciones tecnológicas, a diferencia de otros modelos como el jerárquico y el de red que rápidamente perdieron popularidad. Para acceder a una base de datos relacional se cuenta con un lenguaje de definición de datos (LDD) para especificar el esquema (diseño) de la base de datos, y un lenguaje de manipulación de datos (LMD) para expresar consultas y modificaciones a los datos. Ambos lenguajes están incluidos en el lenguaje estructurado de consultas SQL (por las siglas en inglés de Structured Query Language [2]).

La gran mayoría de los SGBD incluyen el soporte de SQL con el cual se pueden desarrollar aplicaciones, reducir la dependencia física de los datos (cómo y en donde se almacenan los datos) y depender al mínimo de los diferentes perfiles de los programadores que suelen definir estructuras de datos poco compatibles.

Los procedimientos almacenados (PA) son un concepto fundamental en los sistemas con acceso a bases de datos, toda vez que permiten incluir instrucciones de control de flujo (tales como for, while e if-then-else) e instrucciones SQL compuestas, en forma de unidades de compilación, almacenadas y ejecutadas dentro del SGBD, lo que tiene varias ventajas que incluyen el tiempo de ejecución y la seguridad del sistema.

Relacionado con los sistemas con acceso a bases de datos están los sistemas cliente-servidor. Los sistemas cliente-servidor son un caso especial de sistemas distribuidos [4] en el cual:

- Algunos sitios son sitios clientes y algunos son servidores.
- Todos los datos residen en los sitios servidores.
- Todas las aplicaciones son ejecutadas en los sitios clientes.

En un sistema con acceso a bases de datos el término cliente-servidor se refiere principalmente a una arquitectura o división lógica de responsabilidades, en donde el cliente es la aplicación (conocido también como interfaz o parte frontal) y el servidor es un manejador de bases de datos (a quien también se conoce como servicios de fondo o parte dorsal) (ver Figura 1).

Debido a que el sistema cliente-servidor puede ser dividido en dos partes, existe la posibilidad de ejecutarlas en computadoras diferentes, lo cual se conoce como sistema de dos capas.

Un enfoque cliente-servidor tiene determinadas implicaciones para la programación de aplicaciones, como es el procesamiento de consultas distribuidas. En un sistema cliente-servidor con acceso a bases de datos, el programador de aplicaciones no debe usar al servidor sólo como un método de acceso para escribir a nivel de registros individuales. En su lugar debe procurar integrar la mayor cantidad posible de solicitudes a nivel de conjunto de registros; de no hacerlo, disminuirá el rendimiento debido a la cantidad de mensajes (peticiones) involucrados.

Esto es un factor importante a considerar toda vez que un objetivo de un SGBD es proporcionar un entorno que sea tanto práctico como eficiente en la recuperación y el almacenamiento de la información de la base de datos.

La cantidad de mensajes entre clientes y servidores puede ser reducida todavía más si el sistema proporciona algún tipo de mecanismo de procedimientos almacenados (ya abreviados como PA) [4]. Los PA son procedimientos precompilados que están almacenados en el sitio del servidor. Pueden ser llamados desde programas de aplicación por el cliente mediante una llamada a procedimiento remoto (RPC, por las siglas en inglés de Remote Procedure Call) con lo cual, la penalización en el rendimiento asociado con el almacenamiento a nivel de registro puede reducirse considerablemente.

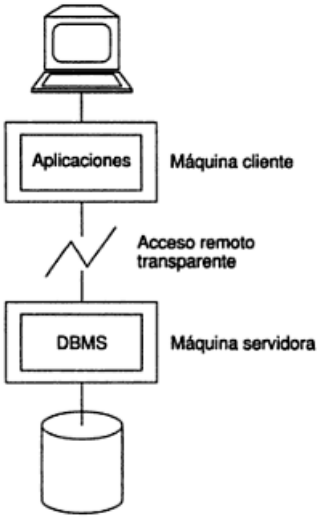


Figura 1: Un sistema cliente-servidor. Tomado de [4], pág. 576.

Cabe mencionar que la mejora en el rendimiento, no es la única ventaja de usar procedimientos almacenados. Otras incluyen:

- Ocultar al usuario diversos detalles específicos del manejador de la base de datos, o de la misma base de datos.
- Un procedimiento almacenado puede ser compartido por varios clientes.
- Los procedimientos almacenados pueden proporcionar mayor seguridad. Por ejemplo en un escenario donde un cierto usuario puede estar autorizado para llamar a un procedimiento dado, pero no para operar directamente sobre los datos a los cuales accede ese procedimiento.

Antecedentes y justificación

La gran mayoría de los SGBD soportan PA, desafortunadamente no se tiene la misma suerte que con los editores de código de lenguajes de programación, para los cuales, se cuenta con herramientas integradas de desarrollo (IDE, por las siglas en inglés de Integrated Development Environment) y herramientas generadoras de código como en el caso de Java, C++, y Python, por mencionar algunos, y que son parte de las así llamadas herramientas CASE.

Lo anterior pone de manifiesto la necesidad de contar con herramientas para el desarrollo de aplicaciones que incluyan el uso de PA, y preferentemente que hagan uso de interfaces gráficas de usuario (GUI). La finalidad de este proyecto de integración es diseñar e implementar un asistente con GUI que genere código para procedimientos almacenados.

En la actualidad, el uso de los PA en un sistema cliente-servidor (sistema de información) no es muy común; y se debe a diversas razones, tales como:

1. La dependencia de cada manejador de base de datos.
2. El desconocimiento de SQL y sus extensiones en un manejador de base de datos.
3. La especialización en solo lenguajes de alto nivel.
4. La consecuente creación de muchos PA en la base de datos que lo convierte en un “servidor gordo”.

Las razones descritas anteriormente ilustran solo algunos motivos por lo que no es muy común el uso de PA. La mayoría de esas razones depende solamente de la perspectiva del diseñador de la base de datos, es decir, tener iniciativa propia para conocer los alcances de las tecnologías que se están utilizando.

Pero la última razón listada es un poco peculiar y no sería una razón válida, ya que el manejador de bases de datos no tiene problema alguno con la creación de objetos. El problema más bien, es respecto a la ardua tarea de crear los PA al no contar con herramientas CASE suficientes, útiles o alcanzables. Esta es una motivación para proponer el presente proyecto.

Hasta la fecha hay muchas opciones a las cuales recurrir, desde generadores de procedimientos almacenados básicos CRUD[3] (por las siglas en ingles Create, Read, Update y Delete) hasta especializados. Pero la gran mayoría de estos generadores son para manejadores de base de datos privativas, y en contra parte hay muy pocas opciones para manejadores de bases de datos libres.

El asistente propuesto para la generación de procedimientos almacenados, generara procedimientos almacenados básicos CRUD y procedimientos almacenados con los parámetros y ajustes proporcionados por el usuario. En otros generadores no incluyen estos procedimientos, solamente los básicos [5]. La mayoría de los generadores investigados, solo crean los procedimientos básicos a partir de una tabla y esta tiene que ser un catálogo. El asistente para la generación de procedimientos almacenados propuesto, aparte de realizar procedimientos almacenados para catálogos, también tendrá la capacidad de poder crear procedimientos almacenados para entidades. Mientras que para las relaciones, el asistente generador proporcionara un acercamiento de procedimiento CRUD considerando la relación como si fuera una entidad. Por consecuente el usuario podrá modificar el comportamiento del PA propuesto para cumplir su cometido.

[3] Los procedimientos almacenados básicos CRUD, se entiende que son procedimientos almacenados que implementan alguna función básica en bases de datos, ya sea Alta (Insert), Baja (Delete), Cambio (Update) o Consulta (Read) de registros.

Entonces, para hacer una aportación en el desarrollo científico y tecnológico en bases de datos, se pretende contribuir con el desarrollo de este proyecto de integración para el SGBD MySQL, ya que cuenta con muy pocos generadores de procedimientos almacenados. Aunque podríamos mencionar a la herramienta denominada MySQL Workbench [6], la cual es una herramienta visual de administración y diseño de bases de datos del SGBD MySQL, que integra creación y mantenimiento de sistemas de bases de datos. Esta herramienta contiene un editor de PA, pero solo proporciona el código básico asociado a la creación de un PA, es decir solo proporciona las líneas de código pertinentes para establecer el nombre del PA, parámetros y el cuerpo de la rutina. Otra herramienta que genera PA, es la denominada phpMyAdmin [7], en sus últimas versiones, presentan un generador para procedimientos almacenados y funciones, que utiliza los parámetros y ajustes proporcionados por el usuario mediante el uso de una interfaz web.

Por último, lo que hace diferente a este asistente para la generación de procedimientos almacenados, frente a los anteriormente mencionados; es su forma de poder visualizar en una estructura de tipo árbol, el anidamiento asociado a las invocaciones dentro de un procedimiento. Ya que actualmente, solo existen herramientas para crear y editar procedimientos almacenados, y ninguna que realice esta función. Además cabe mencionar que en el SGBD solamente se pueden apreciar los procedimientos almacenados de forma plana, es decir, no podemos visualizar el anidamiento asociado a las invocaciones dentro de los procedimientos, pudiendo ser de mucha ayuda en el momento de analizar el rendimiento de nuestro SGBD, como en el caso de la identificación de cuellos de botella o de concurrencia para mejorar el desempeño de nuestro sistema relacional de bases de datos.

Objetivos

Objetivo general

Diseñar e implementar un programa generador de código para procedimientos almacenados en un manejador de bases de datos.

Objetivos específicos

- Diseñar e implementar una aplicación de escritorio con interfaz gráfica de usuario que facilite la generación de código de procedimientos almacenados para un manejador de bases de datos.
- Diseñar e implementar un módulo de configuración de parámetros requeridos en la generación de código de procedimientos almacenados.
- Diseñar e implementar un módulo de generación de código de procedimientos almacenados.

Marco teórico

Para la realización de este proyecto se recurrió a la investigación de diversos temas específicos.

Estructura de un Procedimiento Almacenado (PA) [8]

Dentro de la descripción de la sintaxis para las declaraciones SQL soportadas por el manejador de bases de datos MySQL; las declaraciones de definición de datos son de suma importancia para este proyecto, ya que aquí, se encuentran algunas de las declaraciones que se usan para administrar un Procedimiento Almacenado, tales como:

- **CREATE PROCEDURE**
- **DROP PROCEDURE**
- **ALTER PROCEDURE**

En los siguientes subsecciones se describen los elementos involucrados en las sentencias descritas.

La sentencia CREATE PROCEDURE

La sentencia **CREATE PROCEDURE** crea un procedimiento almacenado en el Sistema Gestor de Bases de Datos MySQL, para cumplir con su cometido se debe especificar ya sea de manera explícita o implícita algunas cláusulas. La sintaxis de la sentencia **CREATE PROCEDURE** se muestra de manera general en la Figura 2.

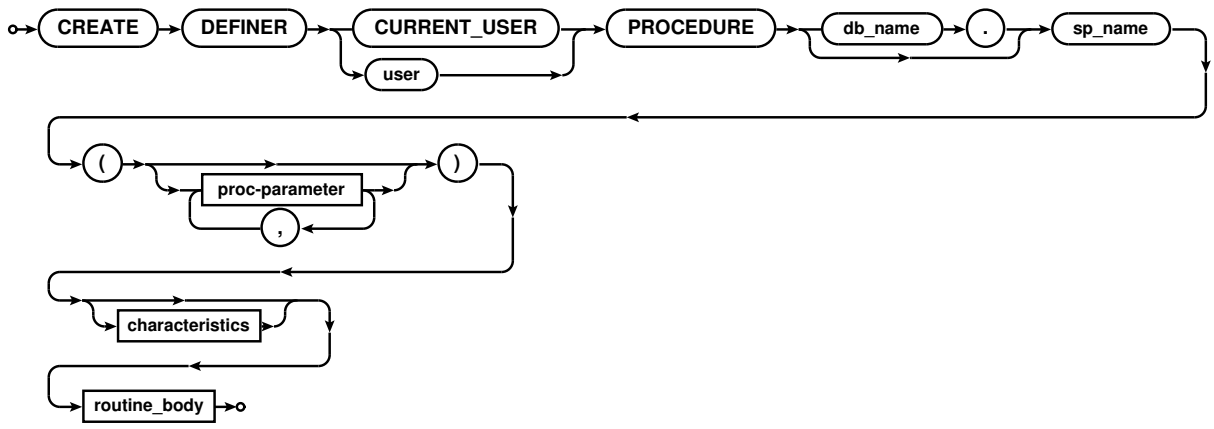


Figura 2: Estructura sintáctica para la creación de un procedimiento almacenado.

La clausula DEFINER

La cláusula **DEFINER** especifica la cuenta de MySQL para ser usada cuando se verifiquen los privilegios de acceso en el tiempo de ejecución de la rutina; solamente para aquellas rutinas que tienen la característica **SQL SECURITY DEFINER**.

Si un valor **user** es proporcionado por la cláusula **DEFINER**, este debería ser una cuenta MySQL especificada como **'user_name'@'host_name'**, **CURRENT_USER** o **CURRENT_USER()**. El valor por defecto de la cláusula **DEFINER** es el usuario quien ejecuta la declaración **CREATE PROCEDURE**. Es lo mismo como especificar **DEFINER = CURRENT_USER** explícitamente. Si se especifica la cláusula **DEFINER**, las siguientes reglas determinan los valores de usuarios válidos para la cláusula:

- Si no se cuenta con privilegios de superusuario, el único valor autorizado es el de su propia cuenta, ya sea que se especifique literalmente o usando **CURRENT_USER**. No se podrá establecer la definición para alguna otra cuenta.
- Si se cuenta con privilegios de superusuario, se puede especificar cualquier nombre de cuenta valida sintácticamente. Si la cuenta no existiera actualmente, una advertencia es generada.
- Aunque es posible crear una rutina con una cuenta no existente (especificada explícitamente en la cláusula **DEFINER**), un error ocurre en tiempo de ejecución de la rutina, si el valor de la característica **SQL SECURITY** es **DEFINER** y la cuenta definida aún no existe. Recuerde crear el usuario después de crear la rutina asociada o antes de ejecutarla.

Nombre del procedimiento [9]

Ciertos objetos dentro de MySQL, incluyendo los nombres de: bases de datos, tablas, índices, columnas, alias, vistas, procedimientos almacenados, particiones y otros nombres de objetos son conocidos como identificadores. Un identificador puede estar citado o no. Si un identificador contiene caracteres especiales o es una palabra reservada, el identificador debería citarse, siempre que se haga referencia a este. Los identificadores son convertidos a Unicode [10] internamente. Podrían contener los siguientes caracteres:

- Caracteres permitidos en identificadores no citados:
  - ASCII: [0-9,a-z,A-Z$_] (Letras latinas básicas, dígitos del 0 al 9, signo de dólar y guion bajo)
  - Extendido: U+0080 ... U+FFFF
- Caracteres permitidos en identificadores citados, incluye el completo Unicode BMP (por sus siglas en inglés Basic Multilingual Plane), excepto U+0000:
  - ASCII: U+0001 ... U+007F
  - Extendido: U+0080 ... U+FFFF
- ASCII NUL (U+0000) y caracteres suplementarios (U+10000 y los más altos) no son permitidos ya sea que los identificadores se encuentre citados o no.
- Los identificadores pueden comenzar por un dígito, pero a menos que se encuentren citados no podrán consistir solamente de dígitos.

El carácter delimitador usado al momento de citar un identificador es el acento grave (" ` ", *backtick*).

Una variable de usuario no puede ser usada directamente en una declaración SQL como un identificador o como parte de uno. En la tabla 1, se describe la longitud máxima para cada tipo de identificador.

Identificador	Longitud máxima (en caracteres)
Database	64
Table	64
Column	64
Index	64
Constraint	64
Stored Program	64
View	64
Tablespace	64
Server	64
Log File Group	64
Alias	256
Compound Statement Label	16

Tabla 1: Longitud máxima por tipo de identificador

Por último, al crear una rutina, esta es asociada a la base de datos que se encuentre por defecto. Para poder asociarla a una base de datos dada, se especifica de manera explícita en la parte del nombre del procedimiento, cuando es creado:

db_name.sp_name

Lista de parámetros

La lista de parámetros de un procedimiento almacenado se encuentran encerrados entre paréntesis (“ ( ) ”). En dado caso en que el procedimiento almacenado no requiera de parámetro alguno, la lista de parámetros debería estar vacía, es decir, que solamente deberíamos apreciar los paréntesis. Lo descrito anteriormente se ilustra en la Figura 2.

Cada parámetro es por defecto un parámetro de “entrada” (la palabra reservada **IN** indica dicho tipo de entrada de manera explícita). Para especificar otro tipo de entrada, se usa la palabra reservada **OUT** o **INOUT** antes del nombre del parámetro; que indican el tipo de entrada como de: “salida” y “entrada-salida” respectivamente (vea la Figura 3).

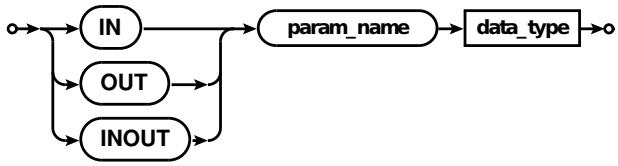


Figura 3: Tipos de entrada de los parámetros en los procedimientos almacenados.

Un parámetro de “entrada” pasa un valor al interior de un procedimiento. El procedimiento podría modificar el valor, pero la modificación no es visible para quien invoca el procedimiento. Un parámetro de “salida” pasa un valor desde el interior hacia el exterior de un procedimiento. Su valor inicial dentro del procedimiento, es un valor nulo (**NULL**), y su valor es visible para quien invoca al procedimiento. Un paramento de “entrada-salida” es inicializado por quien realiza la invocación, el valor podría ser modificado por el procedimiento y ningún cambio realizado antes de que el procedimiento retorne es visible para quien realiza la invocación.

Para cada parámetro de “salida” o ”entrada-salida“, se pasa una variable definida por el usuario en la declaración **CALL** que invoca al procedimiento a modo de que se puedan obtener sus valores cuando este retorne. Si se invoca al procedimiento desde el interior de otro o de alguna función, se puede pasar un parámetro o una variable local de la rutina como un parámetro de “entrada” o “entrada-salida”.

Por último, los parámetros de la rutina no pueden ser referenciados en declaraciones preparadas (*statements prepared*) dentro de la rutina.

Los valores de la cláusula CHARACTERISTICS

La cláusula **CHARACTERISTICS** describe de manera general al procedimiento almacenado, es decir, proporciona información importante para el DBA (por las siglas en inglés de Data Base Administrator), tal como: los datos que maneja el procedimiento almacenado, el contexto de seguridad y los comentarios. Los valores que pueden tomar la cláusula **CHARACTERISTICS** son los que se muestran en la Figura 4.

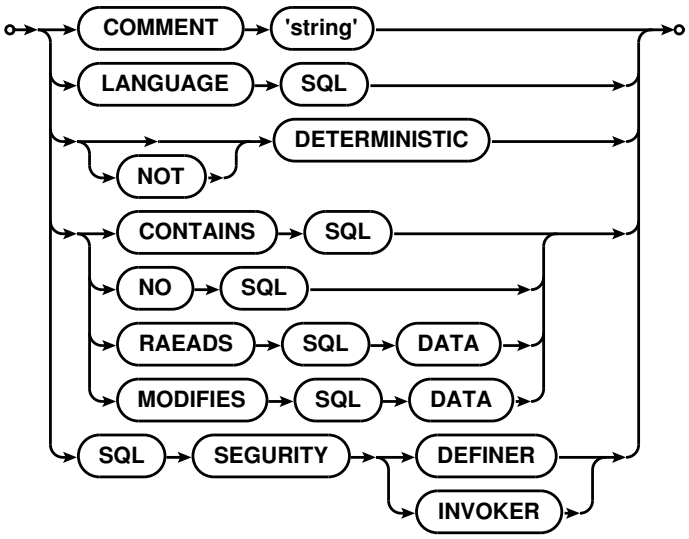


Figura 4: Valores de la cláusula **CHARACTERISTICS** de un procedimiento almacenado.

A continuación se describen cada uno de los valores de la cláusula **CHARACTERISTICS**:

- **COMMENT**: la característica **COMMENT** es una extensión de MySQL, y puede ser usada para describir la rutina almacenada. Esta información puede ser visualizada en el contenido del retorno al invocar la declaración **SHOW CREATE PROCEDURE**.
- **LANGUAGE**: La cláusula **LANGUAGE** indica el lenguaje en el cual la rutina está escrita. El servidor ignora esta característica porque solamente soporta rutinas SQL.
- **DETERMINISTIC/NO DETERMINISTIC**: Una rutina es considerada “determinista” (se usa la palabra reservada **DETERMINISTIC**) si esta siempre produce el mismo resultado para los parámetros de entrada, y “no determinista” (se usa la palabra reservada **NO DETERMINISTIC**) si es de otra forma. Si no se proporciona ninguna de las palabras reservadas, ya sea **DETERMINISTIC** o **NO DETERMINISTIC**, por defecto toma la palabra reservada **NO DETERMINISTIC**. Para declarar que la rutina almacenada es “ determinista”, se debería especificar de manera explícita.
- Varias características proveen información acerca de la naturaleza de los datos usados por una rutina. En MySQL, dichas características son solamente de consulta. El servidor no las utiliza para limitar que tipo de declaraciones en una rutina podrían ser permitidas para ejecutarse.
    - **CONTAINS SQL**: indica que la rutina no contiene declaraciones que leen o escriben datos. Esta palabra reservada es el valor por defecto si ninguna de las siguientes características son proporcionadas de manera explícita.

  - **NO SQL**: indica que la rutina no contiene declaraciones SQL.
  - **READS SQL DATA**: indica que la rutina contiene declaraciones que solamente leen datos.
  - **MODIFIES SQL DATA**: indica que la rutina contiene declaraciones que pueden escribir datos.

- **SQL SECURITY**: la característica **SQL SECURITY** le puede seguir la palabra reservada **DEFINER** o **INVOQUER** para especificar el contexto de seguridad; esto es, si se ejecuta la rutina usando los privilegios de la cuenta especificada en la cláusula de la rutina **DEFINER** o el usuario quien invoca el procedimiento. El valor por defecto es **DEFINER** (**SQL SECURITY DEFINER**).

El cuerpo de la rutina

EL cuerpo de la rutina consiste de una declaración de rutina SQL valida. Esta puede ser una simple declaración tal como un **SELECT** o un **INSERT**, o una sentencia compuesta utilizando **BEGIN** y **END**. Las sentencias compuestas pueden contener desde declaraciones, ciclos, y otros tipos de estructuras de control.

Algunas consideraciones que deben ser tomadas en cuenta al trabajar con procedimientos almacenados son:

- Para invocar un procedimiento almacenado, se usa la declaración **CALL**.
- la declaración **CREATE PROCEDURE** requiere privilegios sobre creación de rutinas, también podría requerir privilegios de superusuario, dependiendo del valor de **DEFINER**. Por defecto, MySQL automáticamente otorga privilegios ejecución y modificación para las rutinas creadas. Este comportamiento puede ser cambiando al deshabilitar la variable del sistema denominada **automatic_sp_privileges**.
- Las clausulas **DEFINER** y **SQL SEGURITY** especifican el contexto de seguridad para ser usadas cuando se verifican los permisos de acceso en tiempo de ejecución de la rutina.
- Si el nombre de la rutina es el mismo nombre de una función interna de SQL, ocurre un error de sintaxis a menos que se use un espacio en blanco entre el nombre y los paréntesis cuando se defina la rutina o cuando esta sea invocada. Por esta razón, se evita usar los nombres de funciones internas de SQL en los nombres de procedimientos almacenados a diseñar.
- El modo SQL **IGNORE SPACE** aplica a las funciones, mas no a las rutinas almacenadas. Esto siempre permite tener espacios después del nombre de una rutina almacenada, independientemente si el modo **IGNORE SPACE** está habilitado.

La sentencia DROP PROCEDURE

La sentencia **DROP PROCEDURE** es usada para dar de baja un procedimiento almacenado. Esto es, la rutina especificada en la sentencia es removida del servidor. La sintaxis de la sentencia **DROP PROCEDURE** se muestra de manera general en la Figura 5.

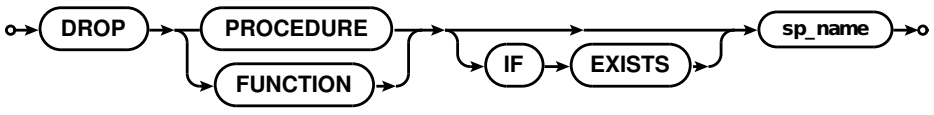


Figura 5: La sentencia **DROP PROCEDURE**.

La cláusula **IF EXISTS** es una extensión. Impide que se produzca un error si el procedimiento o función no existe.

Usaremos la declaración **DROP PROCEDURE** solamente en el dado caso en el que el usuario use nuestra aplicación para depurar su PA, es decir, que se encuentre en un estado de realización de cambios constantes en la creación

del PA y necesite ver los cambios reflejados que provocan dichas ediciones (antes de llegar a un PA final). Con dicha instrucción el usuario solamente tendría que probar su PA y automáticamente si existiese una versión previa del PA en su servidor al que se encuentra conectado (para probar el PA, se debe establecer una conexión con el servidor del usuario previamente) esta se borrará y podrá cargar el nuevo código de PA a probar, sin la necesidad de que el usuario borre directamente en el manejador la versión del PA anterior.

Desarrollo

Para la generación de código SQL asociado a la creación de procedimientos almacenados en el SGBD MySQL, resulta tedioso hasta el punto de cometer errores sintácticos en el diseño, cuando se crean demasiados procedimientos. Algunas veces la lógica que implementan los procedimientos almacenados suelen ser la misma para algunas entidades, catálogos o relaciones, por lo que resulta conveniente hacer algunas modificaciones (como en los parámetros de entrada) a procedimientos previamente diseñados. Inclusive al diseñar un procedimiento almacenado, este puede que necesite invocar a otros procedimientos que aún no existen para poder implementar su función a desempeñar. Llevar ese control de anidamiento resulta extenuante para el usuario. Para ello se propone el presente proyecto de integración, que tiene como base proporcionar una herramienta CASE con GUI con el fin de que el usuario solo se enfoque en construir el cuerpo de la rutina asociado al procedimiento almacenado, mientras que la herramienta mantendrá el control sobre los demás componentes del procedimiento (nombre del procedimiento, parámetros de entrada, contexto de seguridad, etc.) y el anidamiento.

En las siguientes secciones se describen de manera general las etapas que se tomaron en consideración para el desarrollo del presente proyecto de integración.

Diseño de una interfaz gráfica de usuario

El diseño de la interfaz gráfica de usuario se realizó principalmente para atender la necesidad de poder mostrar al usuario las funcionalidades como: crear un procedimiento almacenado, crear un nuevo procedimiento anidado, establecer una conexión con una base de datos, crear un procedimiento que implemente alguna función básica en bases de datos y visualizar el código SQL generado para la creación de un procedimiento almacenado. Se optó por usar el conjunto de componentes Java Swing [11] necesarios para el diseño. En los siguientes apartados se describen las ventanas diseñadas de la aplicación que satisfacen las funcionalidades mencionadas.

Ventana principal

La ventana principal, está compuesta por tres apartados visuales (ver Figura 6):

1. **Árbol de Procedimientos**, en este apartado se visualizan los procedimientos almacenados que han sido creados por el usuario y las invocaciones a otros procedimientos.

2. **Procedimiento Almacenado**, aquí se presenta al usuario los componentes necesarios para la creación de un procedimiento almacenado.

3. **Árbol de Tablas**, se presenta las tablas que contiene una base de datos proporcionada por el usuario (mediante una conexión establecida previamente).

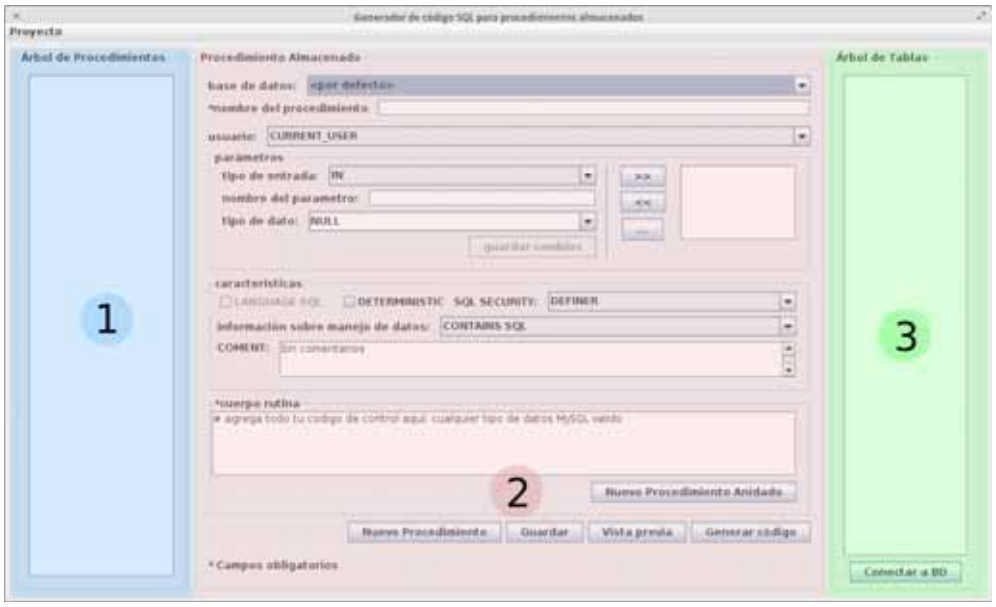


Figura 6: Ventana Principal del Generador de Procedimientos.

Ventana "Generador de código para un nuevo procedimiento almacenado"

La ventana "Generador de código para un nuevo procedimiento almacenado", está compuesta por solo dos apartados visuales (Procedimiento Almacenado y Árbol de Tablas); similares al de la ventana principal, solo con una diferencia en el apartado denominado "Árbol de tablas", en este, no se cuenta con el botón que conecta/desconecta a una base de datos (ver Figura 7). A esta ventana se accede si se desea crear un nuevo procedimiento almacenado anidado, a causa de una invocación dentro de algún procedimiento previo.

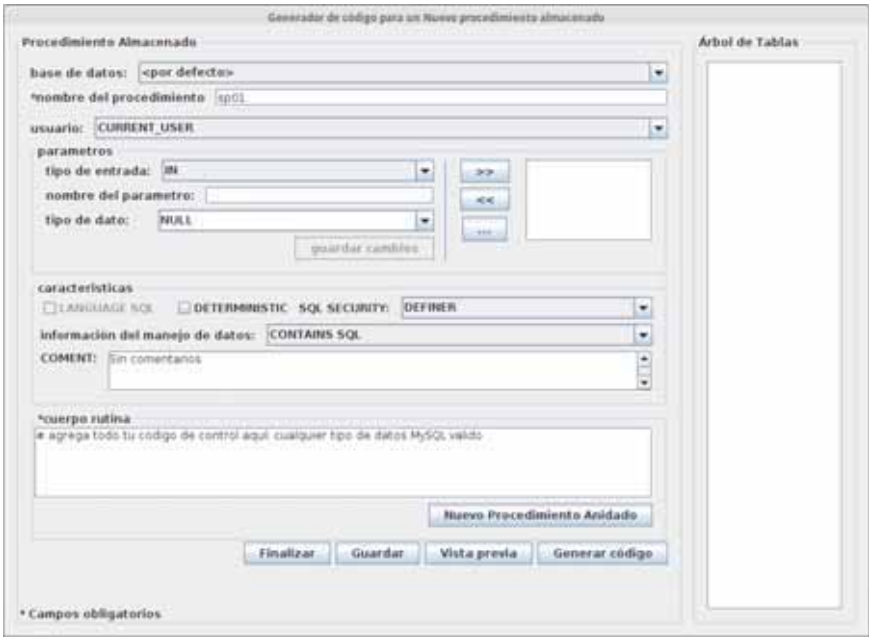


Figura 7: Ventana "Generador de código para un nuevo procedimiento almacenado".

Ventana “Generador de código para un procedimiento almacenado CRUD”

La ventana “Generador de código para un procedimiento almacenado CRUD”, está compuesta por dos apartados visuales (Procedimiento Almacenado CRUD y Árbol de Tablas); similares al de la ventana principal, solo con algunas diferencias, como en el apartado denominado “Procedimiento Almacenado CRUD”, este contiene una parte denominada “Funciones básicas” en la cual el usuario podrá elegir la función básica en bases de datos que desea implementar en el procedimiento, y por otra parte no cuenta con los botones asociados a las funciones que crean nuevos procedimientos. En el apartado “Árbol de tablas”, no cuenta con el botón que conecta/desconecta a una base de datos (ver Figura 8). A esta ventana se accede si se desea crear un procedimiento que implemente alguna función CRUD para una tabla específica, proporcionada previamente en una selección en la ventana principal.

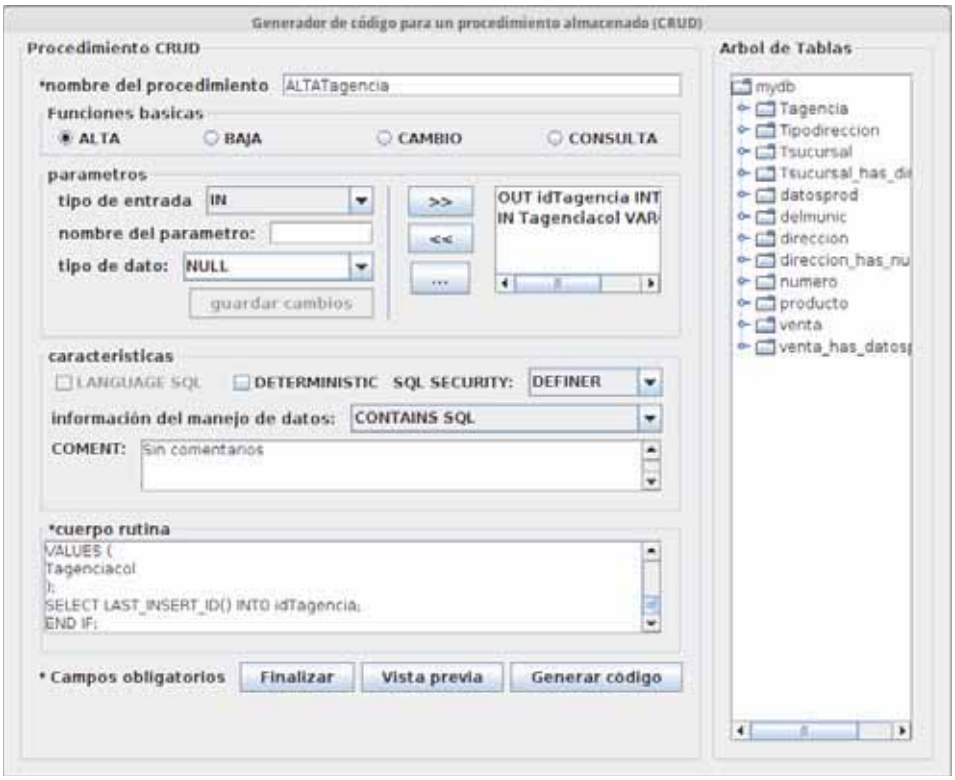


Figura 8: Ventana “Generador de código para un procedimiento almacenado CRUD”

Ventana “Conexión a una base de datos”

La ventana “Conexión a una base de datos”, es la encargada de obtener del usuario los parámetros necesarios para poder establecer una conexión a una base de datos, como: nombre de la base de datos, el nombre del servidor (o dirección IP, del inglés Internet Protocol), puerto, nombre de usuario y contraseña (ver Figura 9).

Ventana “Administración del procedimiento generado”

La ventana “Administración del procedimiento generado”, es la encargada de administrar las funciones que se realizarán al obtener el código de un procedimiento construido previamente, como: almacenar el código generado en un archivo y/o enviar el código generado a una base de datos (ver Figura 10). La opción denominada “remplazar existente”, al estar habilitada se agrega la sentencia **DROP** para el procedimiento del código generado, ya sea en el momento de ser almacenado en un archivo o enviado a una base de datos.

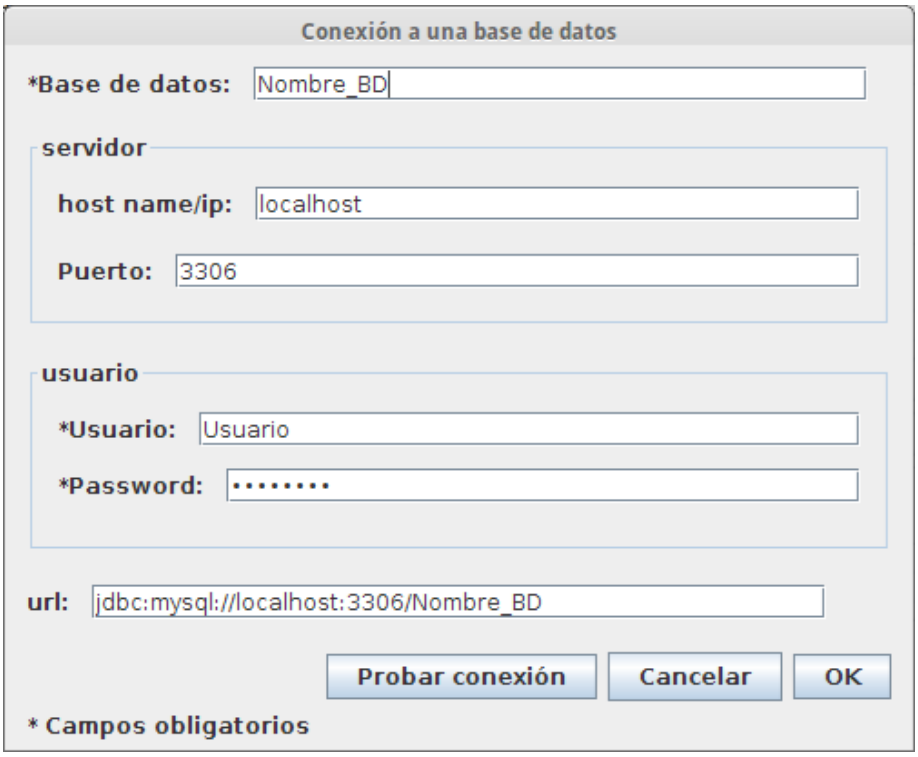


Figura 9: Ventana “Conexión a una base de datos”.

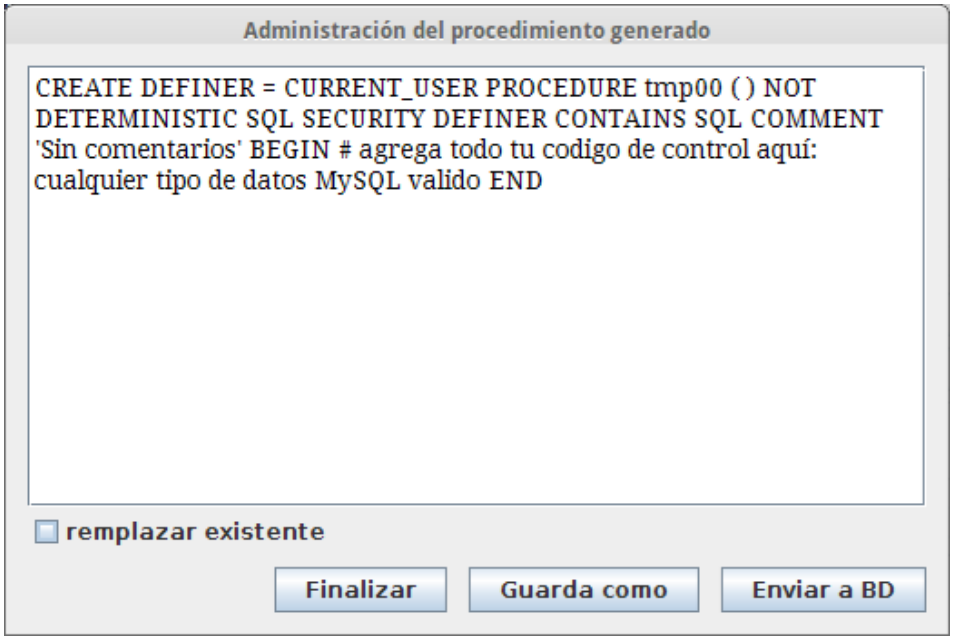


Figura 10: Ventana “Administración del procedimiento generado”.

Conexión a una base de datos

La conexión a una base de datos se estable mediante tres objetos: el primer objeto es de la clase ConexionBD, dicha clase hace uso de la tecnología JAVA DATABASE CONECTIVITY (JDBC) [12] mediante el Driver JDBC proporcionado por MySQL [13] para establecer la conexión. EL segundo Objeto es de la clase VantanaConexion, la cual implementa una ventana de dialogo (JDialog) y diversos elementos más, para poder obtener del usuario los parámetros necesarios para establecer la conexión. El tercer Objeto es de la clase controlConexion, la cual se encarga de establecer la comunicación entre la vista y el modelo. Para poder ver un bosquejo general ver Figura 11. Para más detalle sobre la implementación ir al Apéndice en la sección B.9.

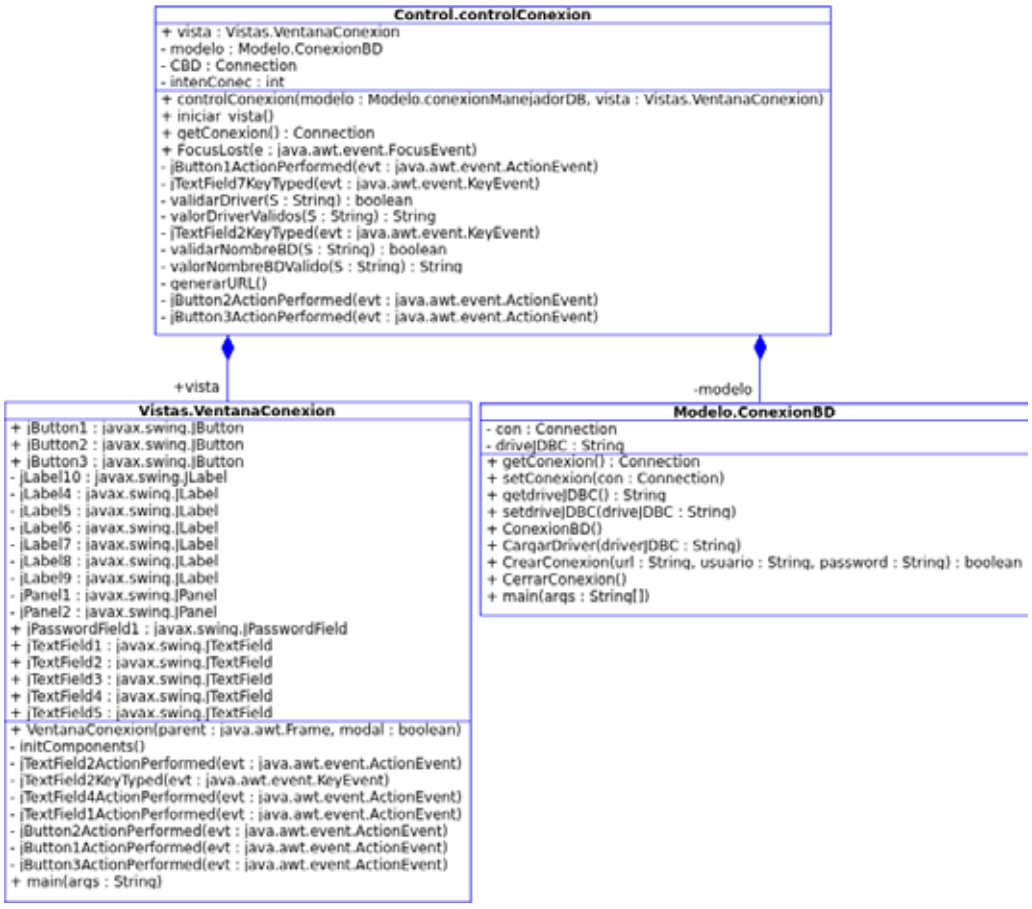


Figura 11: Diagrama de clases Conexion a una base de datos.

Para poder establecer una conexión a una base de datos el usuario debe seguir la siguiente secuencia de pasos:

1. En la ventana principal dirigirse al botón “Conectar a BD”, ver Figura 12.

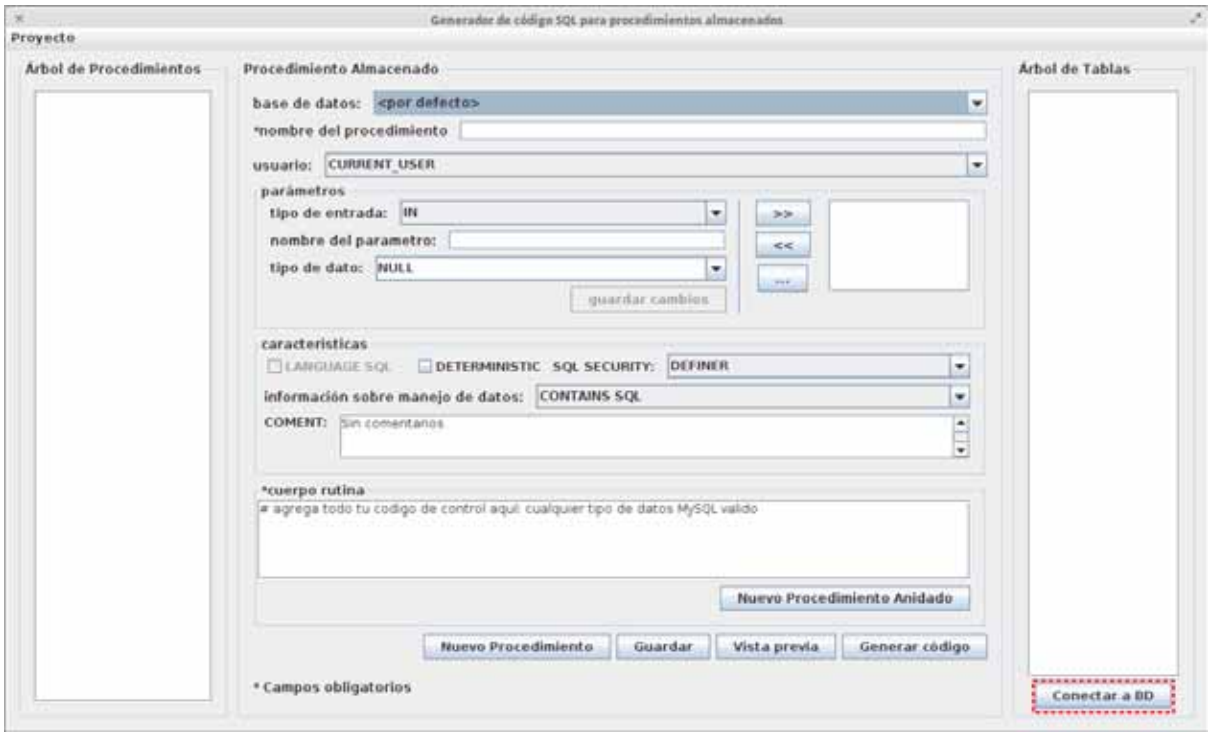


Figura 12: Establecer conexión a una base de datos paso 1.

2. Proporcionar los parámetros indicados como obligatorios y presionar el botón “Probar conexión”. Si se establece la conexión sin errores, deberá presionar el botón “OK”, para poder regresar a la ventana principal (ver Figura 13).

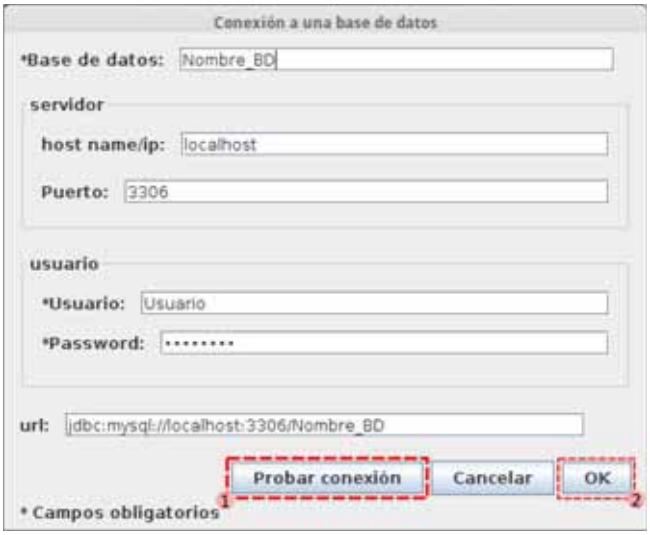


Figura 13: Establecer conexión a una base de datos paso 2.

Extracción de metadatos

La extracción de metadatos sobre el esquema de una base de datos, se realizó mediante el uso de las clase DataBaseMetaData [14]. La cual fue usada en el método denominado MVETDB de la clase controlProcedimiento (ver Apéndice en la sección B.11.3.43). En la Figura 14 se aprecia la extracción de los metadatos reflejados en el apartado visual "Árbol de tablas" al establecer una conexión a una base de datos, en este caso a la base de datos "mydb" (ejemplo de base de datos proporcionada por el SGBD MySQL).

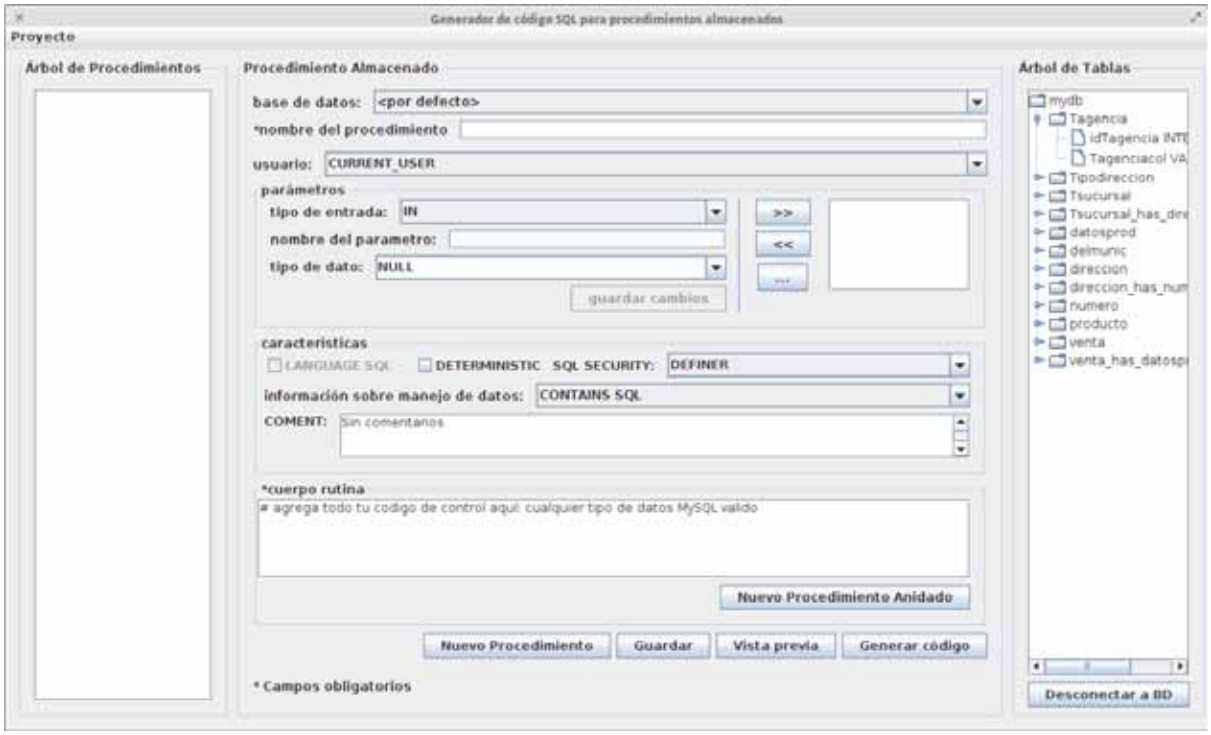


Figura 14: Extracción de metadatos.

Módulo entrada de datos

Este módulo es el encargado de establecer los atributos del procedimiento a partir de los datos proporcionados por el usuario en el apartado visual "Procedimiento almacenado". El método que implementa este módulo es el denominado ModuloEntradaDatos de la clase controlProcedimiento (ver Figura 15), para más detalle ver Apéndice en la sección B.11.3.42.

En la Figura 16 se puede apreciar el diseño de un procedimiento almacenado. Para ver en acción el módulo entrada de datos, se debe guardar, visualizar o generar el código SQL asociado al procedimiento almacenado. En este caso, se optó por guardar el procedimiento y nos percatamos que en el apartado visual "Árbol de procedimientos" ya se encuentra nuestro procedimiento.

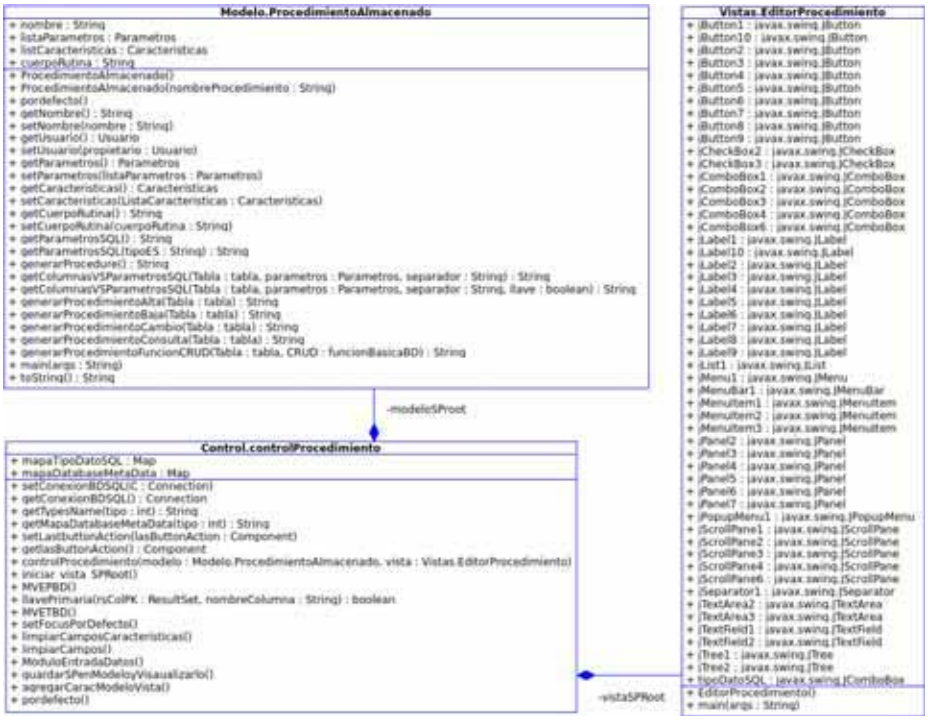


Figura 15: Diagrama de clases sobre el control del procedimiento.

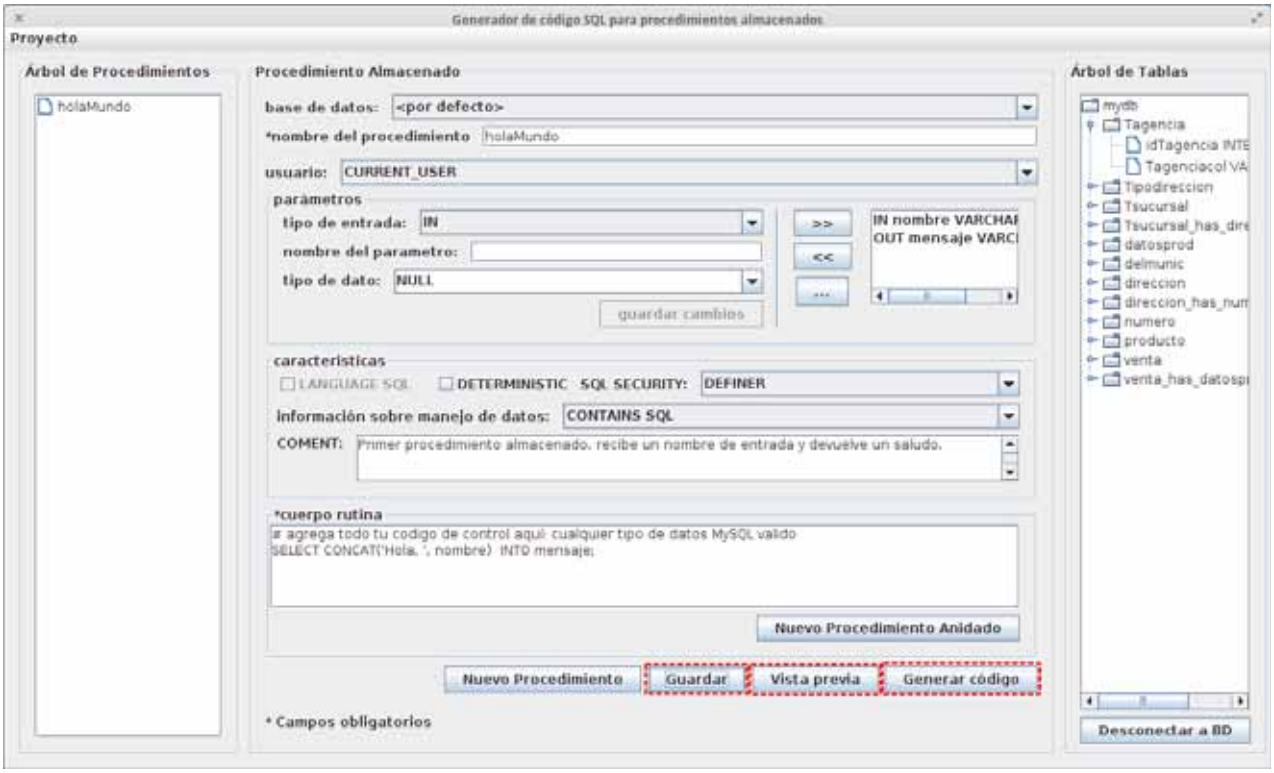


Figura 16: Creación de un procedimiento almacenado.

Módulo de configuración de operaciones básicas

El módulo de configuración de operaciones básicas, permite al usuario poder generar de manera automática un procedimiento almacenando en el cual se implemente alguna función básica. Este módulo lo implementa el método getFuncionCRUD de la clase contolProcedimientoCRUD ver Figura 17 para más detalle ir al Apéndice en la sección B.12.3.8.

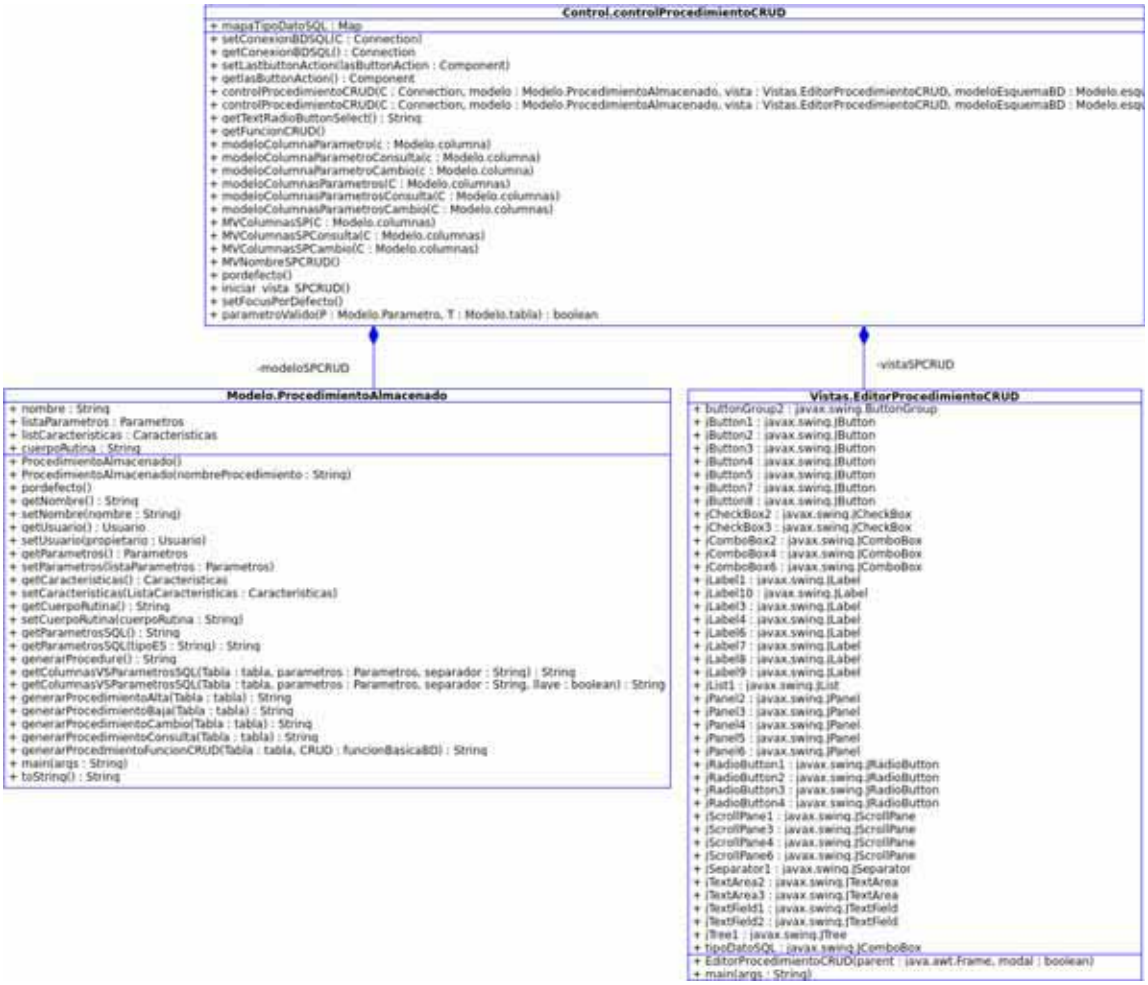


Figura 17: Diagrama de clases Módulo de configuración de operaciones básicas.

Para generar un procedimiento almacenado que implemente alguna función básica en bases de datos, partiendo de la premisa de que ya se ha establecido una conexión a una base de datos previamente, se debe de seguir la siguiente secuencia de pasos:

1. Seleccionar una tabla del árbol y con un clic derecho sobre la selección, elegir la opción “Crear SP CRUD” del menú desplegable (ver Figura 18).

2. Aparecerá una ventana denominada “Generador de código para un procedimiento almacenado (CRUD) ”, la cual presenta configuraciones y parámetros para la generación de código SQL asociado a un procedimiento que implemente alguna función básica en bases de datos, por defecto es la función Alta (puede ser cambiada por alguna otra función). Seleccionar el botón “Vista previa” o “Generar código” para ver el código SQL generado (ver Figura 19).

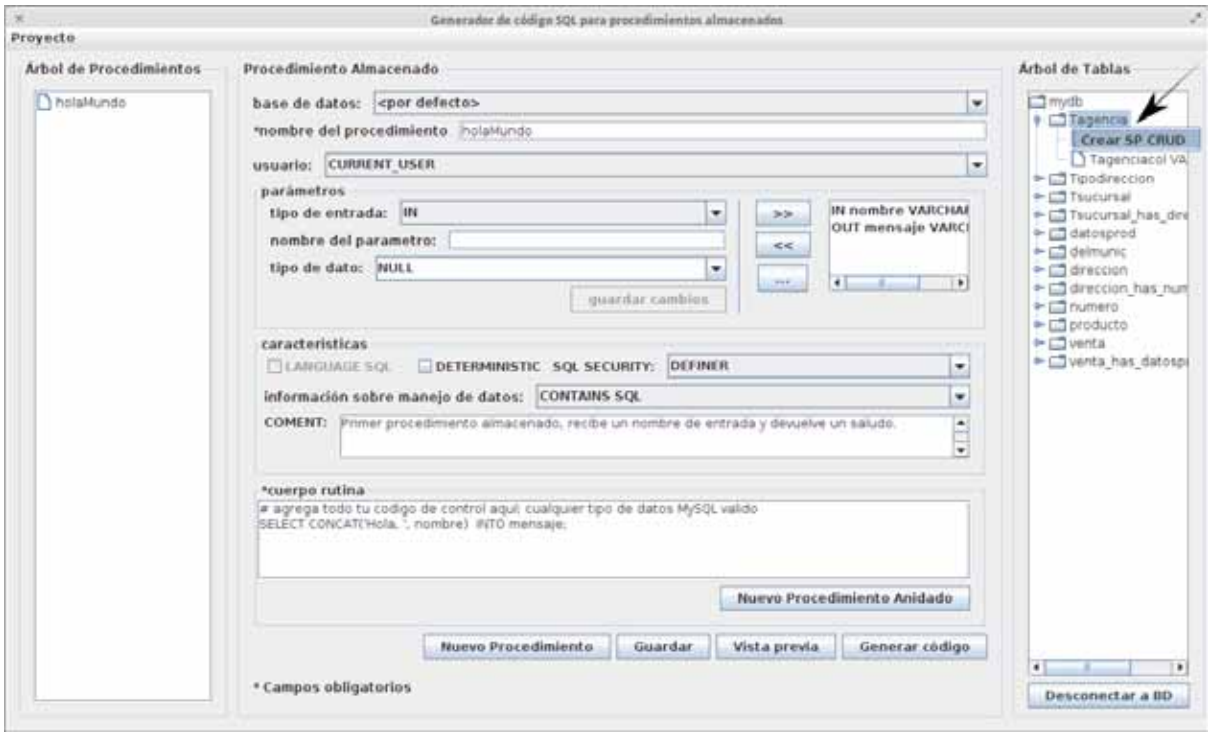


Figura 18: Creación de un procedimiento CRUD, paso 1.

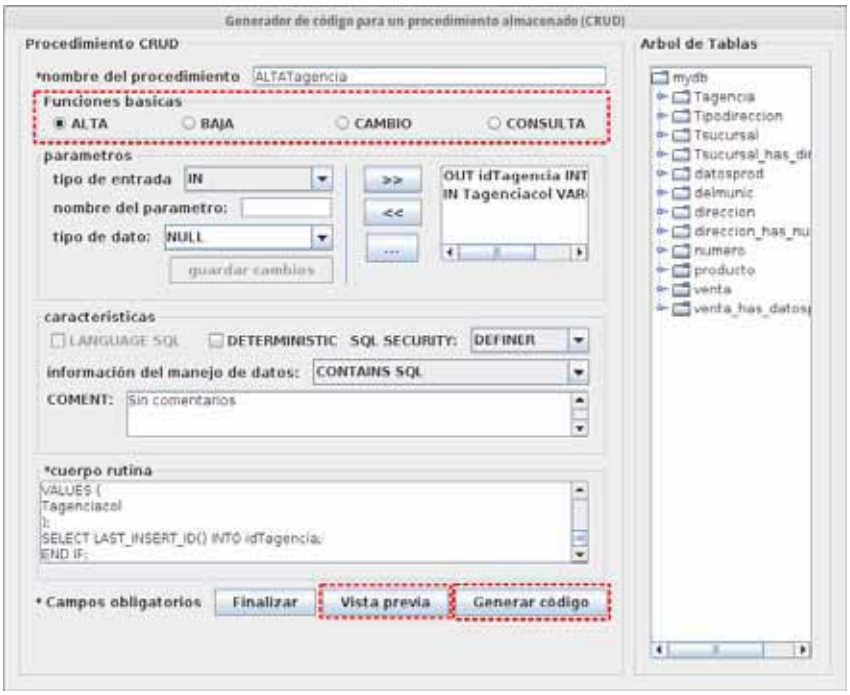


Figura 19: Creación de un procedimiento CRUD, paso 2.

Módulo seleccionador de tablas

Este módulo permite determinar si la selección de un elemento en el apartado visual “Árbol de tablas” es precisamente una tabla. De ser cierto permitirá, ya sea acceder a la ventana “Generador de código para un procedimiento CRUD” desde la ventana principal (como se vio en la sección anterior) o cargar los parámetros pertinentes de una tabla seleccionada, para la creación de un procedimiento almacenado que implemente alguna función básica en la ventana “Generador de código para un procedimiento CRUD”.

Este módulo es implementado por el método jMenuItem1ActionPerformed de la clase controlprocedimiento (ver Figura 22, para mas detalles dirigirse al Apéndice en la sección B.11.3.32) y el método seleccionadorTablas de la clase controlProcedimientoCRUD (ver Figura 17, para mas detalles dirigirse al Apéndice en la sección B.12.3.44)

Solo se describirá el segundo caso en la siguiente secuencia de pasos, como una continuación de la sección anterior:

1. Seleccionar otra función básica en bases de datos (ver Figura 20).

2. Seleccionar otra tabla del árbol (ver Figura 21).

Se puede apreciar que automáticamente se establecen los parámetros, nombre del procedimiento y cuerpo de la rutina, etc. dependiendo de la selección de la tabla y la función básica.

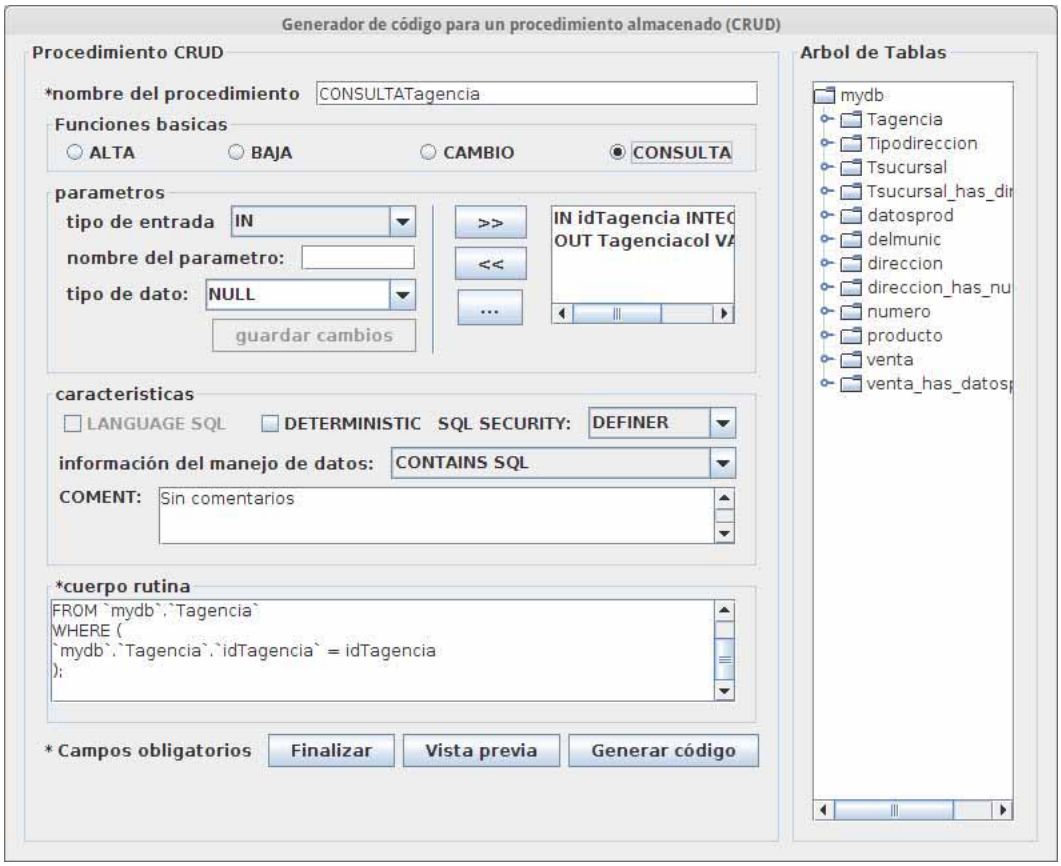


Figura 20: Creación de un procedimiento CRUD, selección de una función básica.

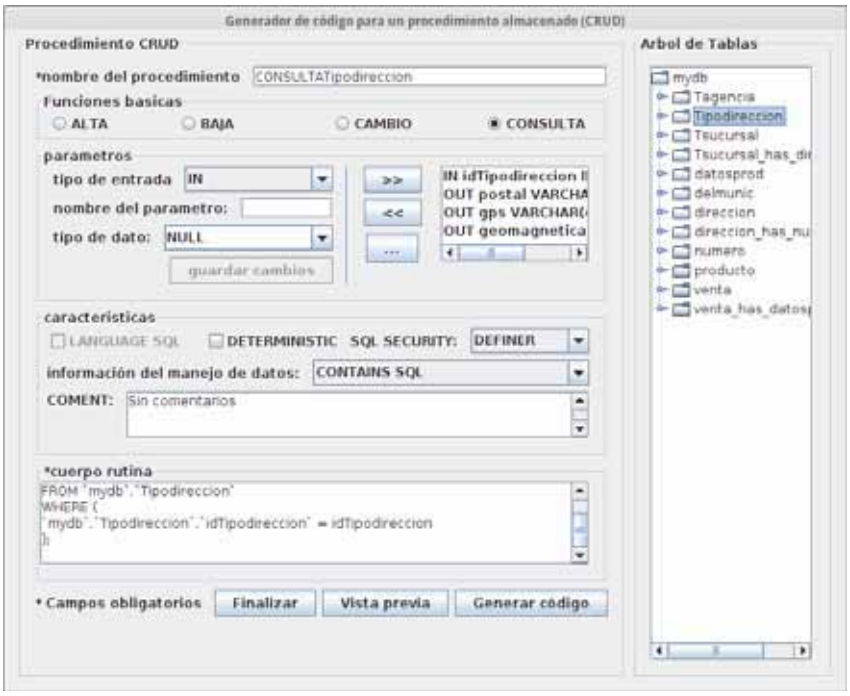


Figura 21: Creación de un procedimiento CRUD, selección de una tabla.

Módulo del control de flujo de un procedimiento

El módulo de control de flujo, permite establecer el flujo dentro de la rutina de un procedimiento. De manera particular, el generador de procedimientos, genera una rutina especial para el procedimiento que implementa la función ALTA. Esta rutina tiene la característica de verificar si el registro a insertar a la base de datos existe, en caso verdadero retorna su llave primaria asociada, de lo contrario inserta el registro y retorna la nueva llave.

Para implementar lo descrito anteriormente, dentro de la clase ProcedimientoAlmacenado (ver Figura 22) el método generarProcedimientoAlta invoca el método ColtroldeFlujoIF, el cual implementa el módulo de control de flujo con una estructura de control IF, permitiendo generar la rutina del procedimiento almacenado para la función ALTA. Para ver más detalles ir al Apéndice en la secciones B.24.3.4 y B.24.3.1.

Figura 22: Diagrama de clase del procedimiento almacenado.

Módulo de configuración de parámetros

Este módulo es el encargado de administrar los parámetros para un procedimiento almacenado, es decir, debe tener la capacidad de gestionar los parámetros. Gestionar un parámetro, significa que se puedan realizar las siguientes funciones:

- Agregar parámetro.
- Editar atributos de un parámetro:
  - Editar el tipo de entrada.
  - Editar el nombre de un parámetro.
  - Editar el tipo de dato.
- Remover un parámetro.

En la vista de la edición de un procedimiento almacenado, el módulo descrito es la sección denominada “parámetros” y las funciones mencionadas las implementan los métodos:

- agregarParametroVista
- editarParametroVistaModelo
- borrarParametroVistaModelo

De las clases:

- controlProcedimiento (ver Figura 22, para más detalle ir al Apéndice en la sección B.11)
- controlProcedimientoCRUD (ver Figura 17, para mas detalle ir al Apéndice en la sección B.12)
- controlNuevoProcedimiento (ver Figura 23, para mas detalle ir al Apéndice en la sección B.10)

Figura 23: Creación de un procedimiento CRUD, selección de una tabla.

En la siguiente secuencia de pasos se puede apreciar en acción el módulo de configuración de parámetros:

- Agregar un parámetro.
  - Establecer los valores de los campos en el apartado denominado “parámetros” y dar clic en el botón “agregar” (ver Figura 24).

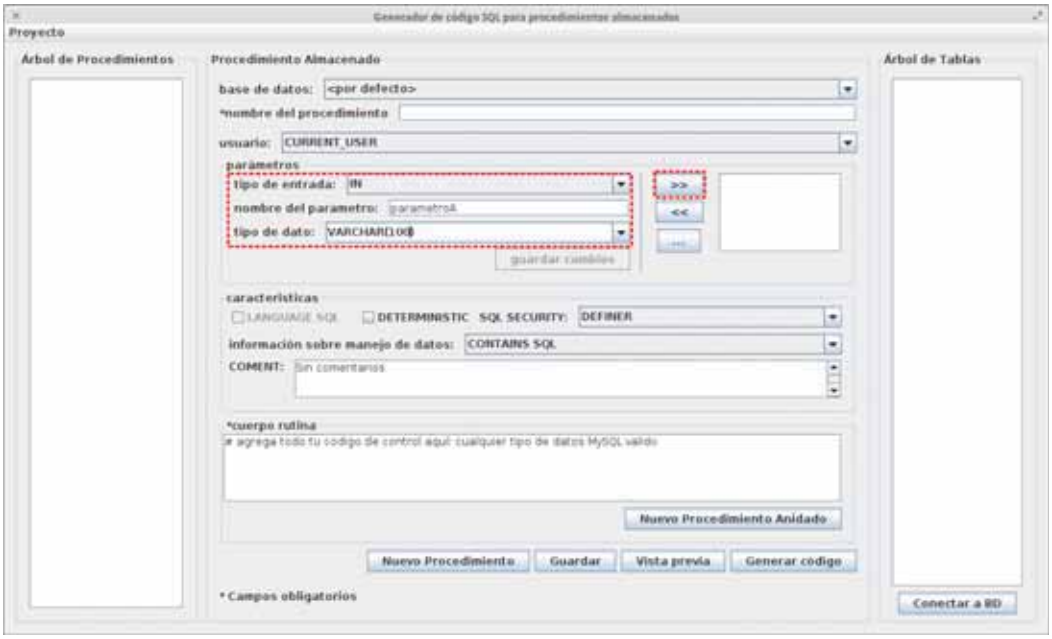


Figura 24: Creación de un procedimiento almacenado, agregar un parámetro.

Se verá reflejado el parámetro agregado como en la Figura 25.

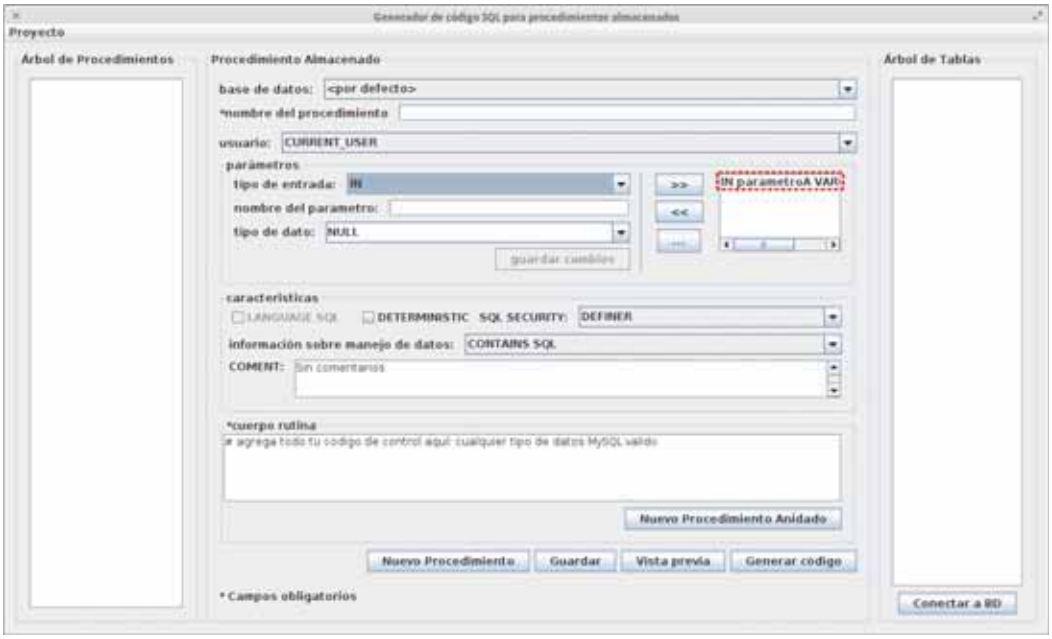


Figura 25: Creación de un procedimiento almacenado, parámetro agregado.

- Editar un parámetro.

  1. Seleccionar el parámetro y hacer clic en el botón “editar” (ver Figura 26).

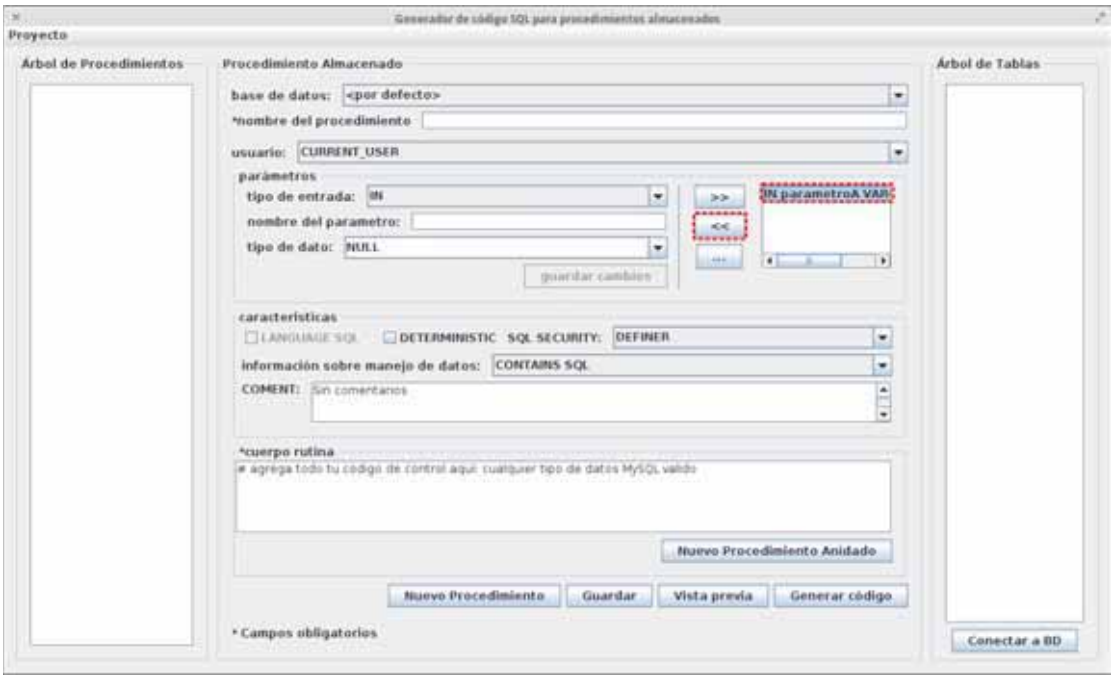


Figura 26: Creación de un procedimiento almacenado, editar parámetro.

  2. Editar el tipo de entrada, el nombre y el tipo de dato de un parámetro y guardar cambios (ver Figura 27).

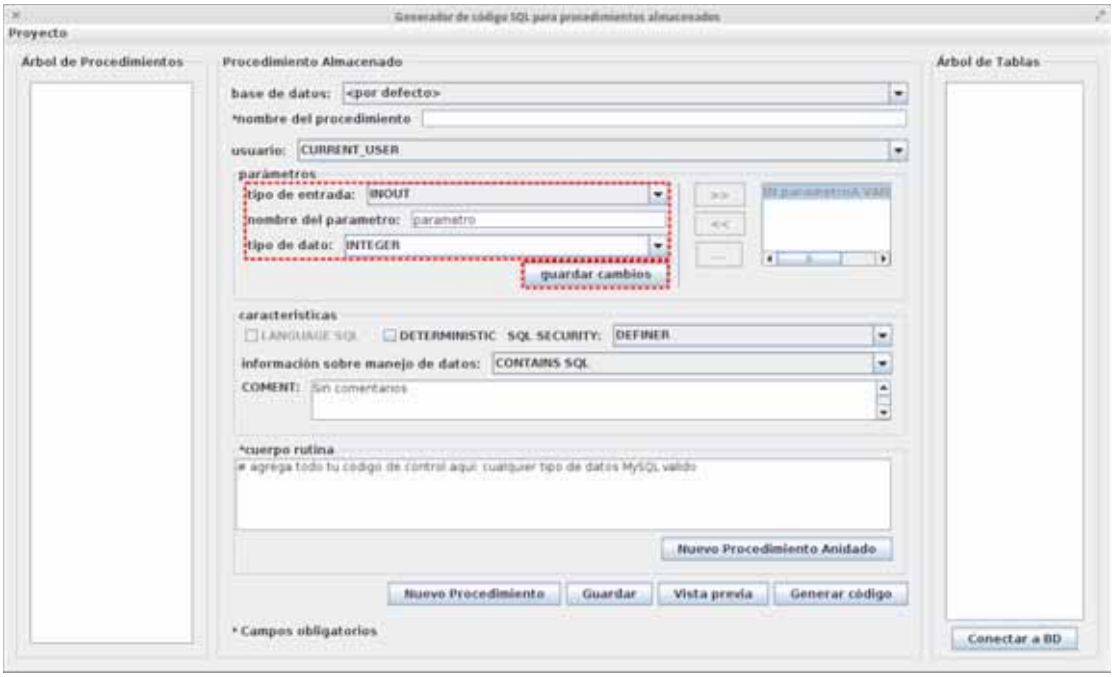


Figura 27: Creación de un procedimiento almacenado, guardar cambios en parámetro.

Se verá reflejado el parámetro agregado como en la Figura 28.

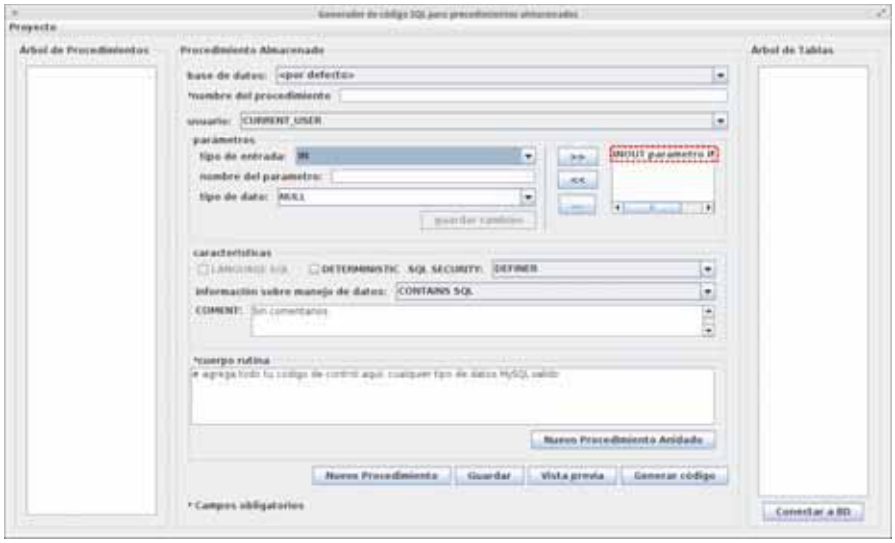


Figura 28: Creación de un procedimiento almacenado, parámetro editado.

Módulo generador de la estructura básica de un procedimiento almacenado

Este módulo genera la estructura básica de un procedimiento almacenado, es decir proporciona el código necesario para crear un procedimiento. El método generarProcedimientoSQL de la clase ProcedimientoAlmacenado (ver Figura 22, para más información dirigirse al Apéndice en la sección B.24.3.8) se encarga de generar la estructura final a partir de los siguientes atributos de un procedimiento: nombre del procedimiento, lista de parámetros, lista de características y cuerpo de la rutina.

La estructura resultante se visualiza al usuario para sus fines convenientes. En la Figura 29 se ilustra el código SQL asociado a la creación del procedimiento holaMundo, como resultado de presionar el botón “vista previa”.

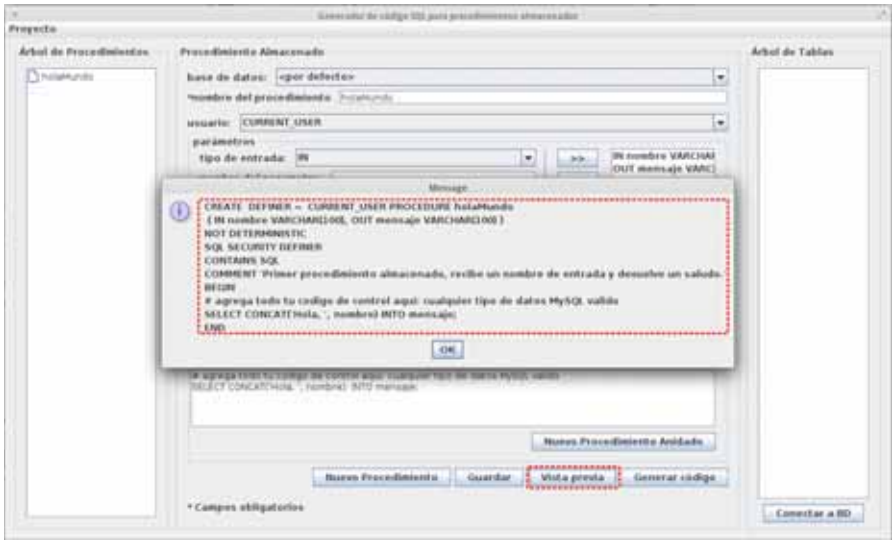


Figura 29: Creación de un procedimiento almacenado, parámetro editado.

Módulo visualizador de código de un procedimiento almacenado

El módulo visualizador de código muestra al usuario el código SQL que se generó, ya sea a partir de los parámetros y ajustes proporcionados por el usuario o por la generación automática de alguna de las funciones básicas en bases de datos. La clase control controlCodigoProcedimientoGenerado es la encargada de implementar este módulo (ver Figura 30, para más detalles ir al Apéndice en la sección B.8).

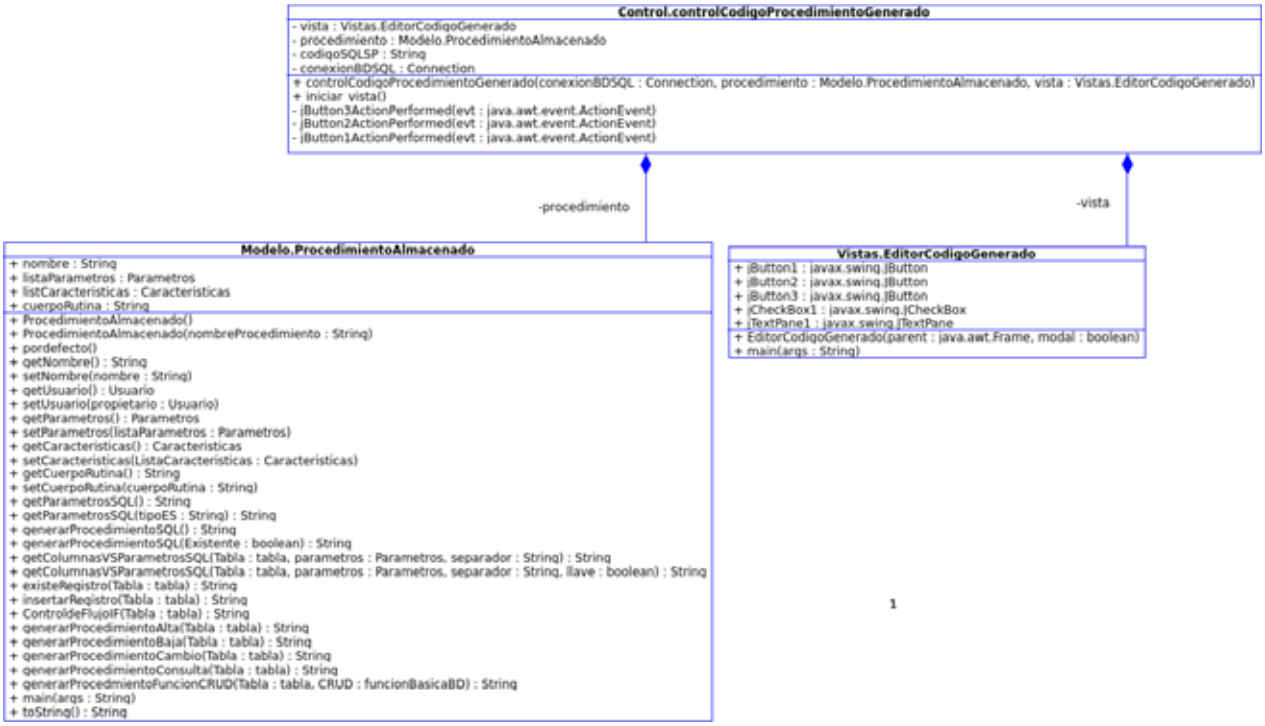


Figura 30: Diagrama de clase del visualizador de código de un procedimiento almacenado.

En la Figura 31 se aprecia el visualizador de código que muestra el código SQL asociado al procedimiento almacenado holaMundo.

Módulo formateador de código de un procedimiento almacenado

Este módulo, se enfoca en el cuerpo de la rutina de un procedimiento, trata de formatear el código para tener una mejor visualización del mismo. El método denominado formatearCuerpoRutina de la clase ProcedimientoAlmacenado (ver Figura 22) implementa el método descrito. Para más información ir al Apéndice en la sección B.24.3.3. En la Figura 31 se visualiza el código SQL asociado al procedimiento holaMundo, se aprecia que esta formateado y con un resaltado que permiten una mejor legibilidad del código.

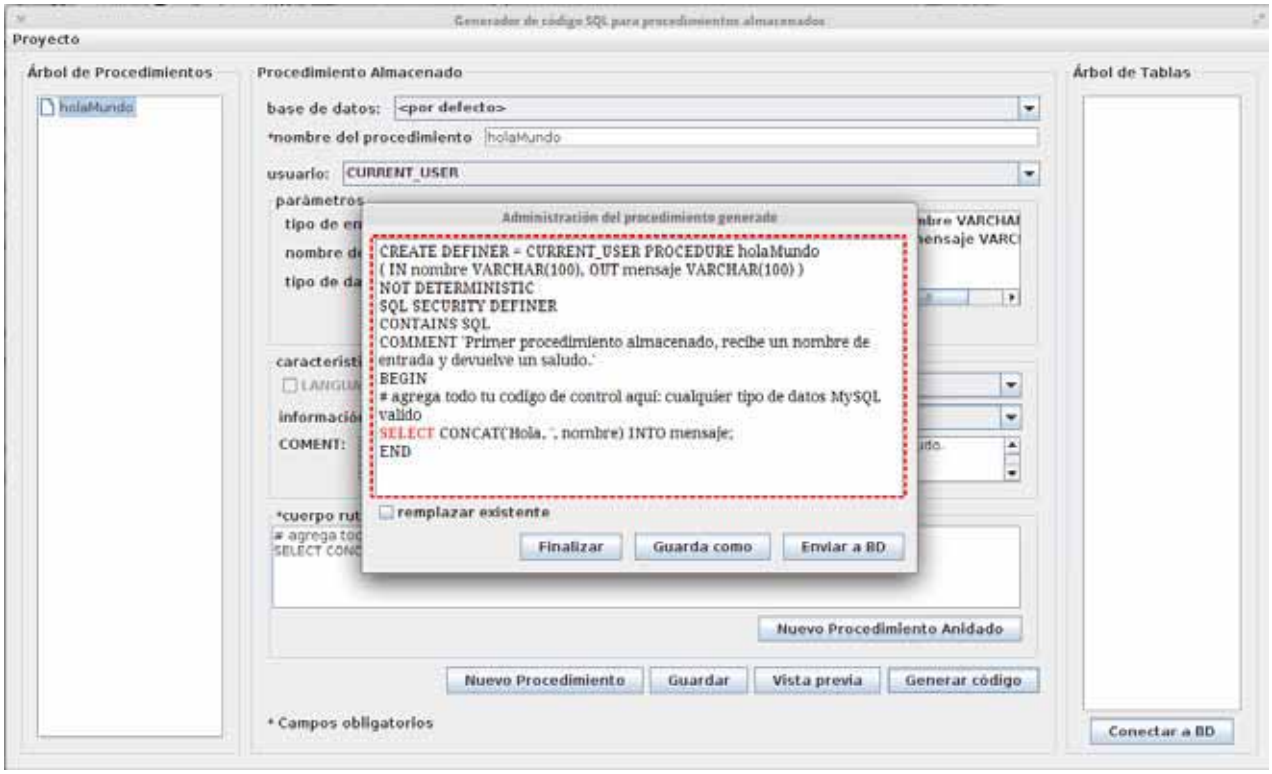


Figura 31: Ventana “Administración del procedimiento generado”.

Módulo generador de código de procedimientos anidados

Este módulo permite generar el código del procedimiento actual y el de los procedimientos que han sido invocados en su interior, desde la ventana principal de la herramienta. El módulo esta implementado por el método generadorCodigoSQLProcedimientosAnidados de la clase controlProcedimiento ver Figura 22, para más detalles ver el Apéndice en la sección B.11.3.7.

En la siguiente secuencia de pasos se puede visualizar un ejemplo que ilustra el anidamiento de los procedimientos en un caso especial, así como la generación de del código SQL asociado:

En un caso hipotético, le solicitan una consulta del personal que cumple años en el mes actual. Tiene un presentimiento de que no será la última vez que le soliciten esto, por lo que ha optado diseñar un procedimiento almacenado que realice esta consulta por usted, el cual podrá invocar en cualquier momento que se lo soliciten de nuevo. Por suerte, en la base de datos de la empresa, cuenta con una tabla que contiene algunos campos que le proporciona los nombres del personal, así como la fecha de nacimiento.

1. Iniciar el “Asistente para la generación de código para procedimientos almacenados” (ver Figura 32).

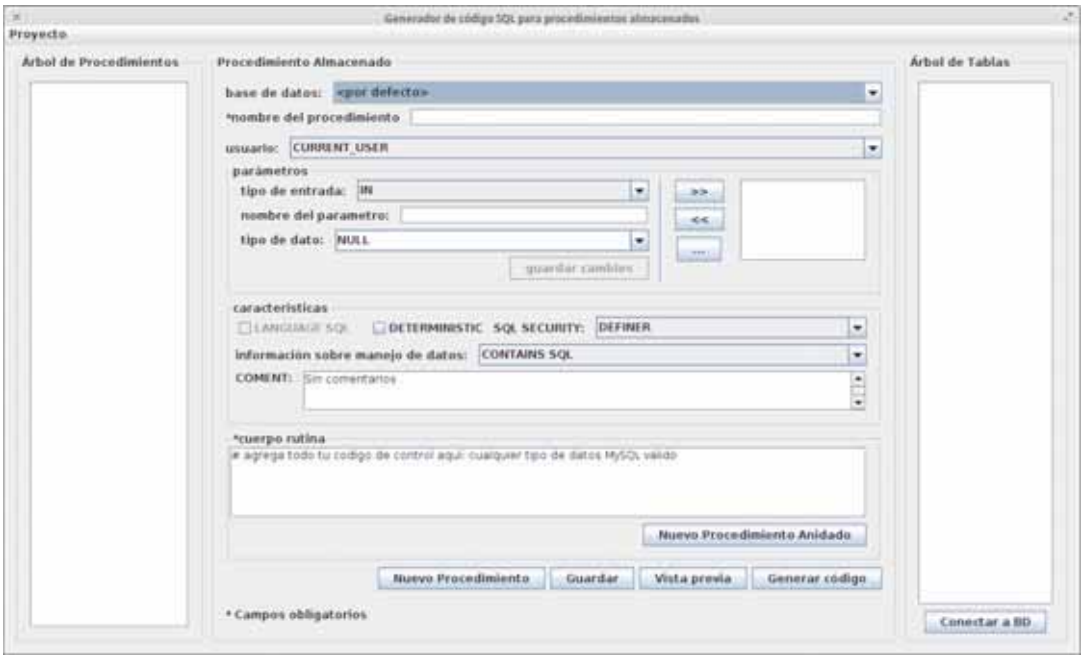


Figura 32: Inicio del “Asistente para la generación de código para procedimientos almacenados”.

2. Establecer una conexión con la base de datos de la empresa, para poder guiarse en la construcción del procedimiento almacenado.

   a) En la ventana principal dirigirse al botón “Conectar a BD”, ver Figura 34.

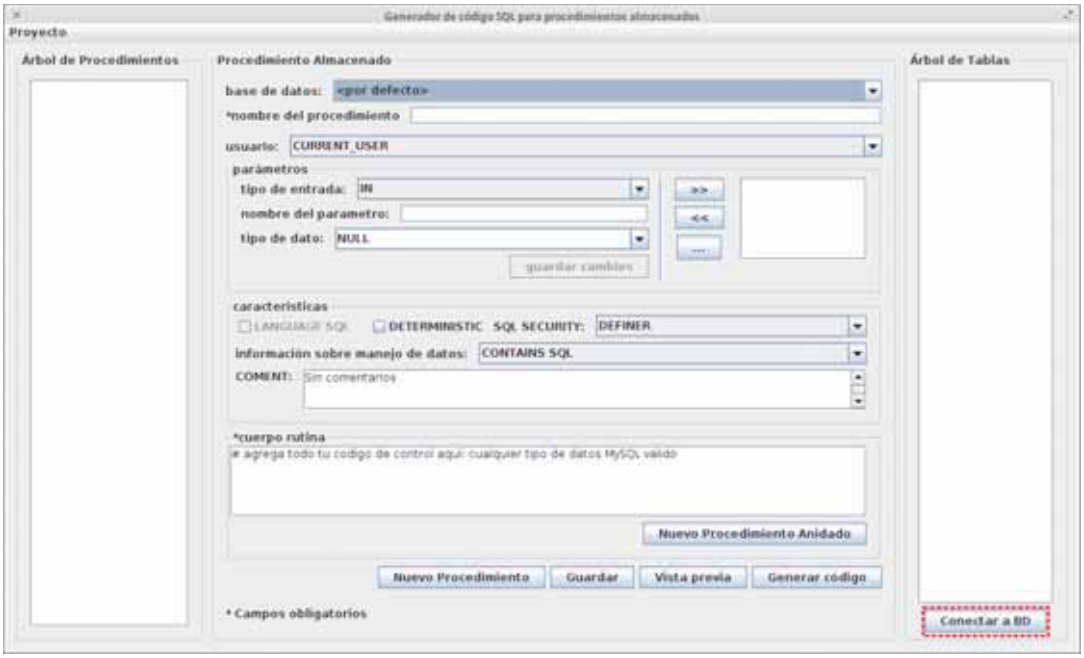


Figura 33: Acceder a la ventana “Conexión a una base de datos”.

b) Proporcionar los parámetros indicados como obligatorios y presionar el botón “Probar conexión”. Si se establece la conexión sin errores, deberá presionar el botón “OK”, para poder regresar a la ventana principal (ver Figura 34).

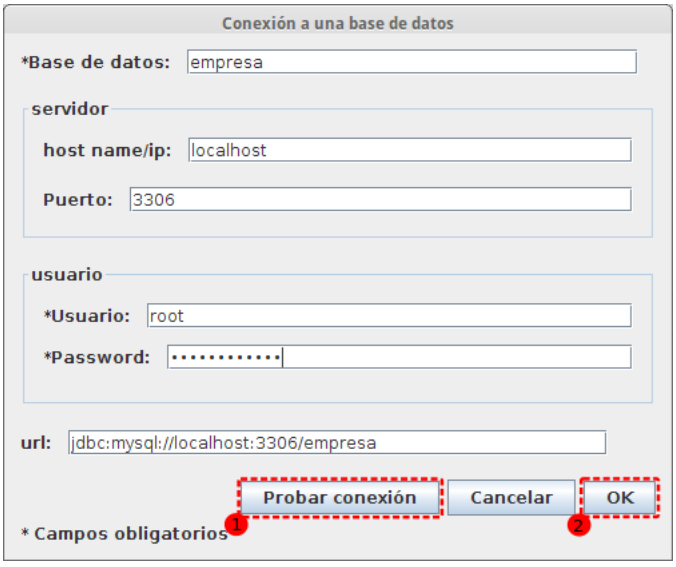


Figura 34: Establecer conexión a una base de datos

En la Figura 35 se muestra la ventana principal resultante.

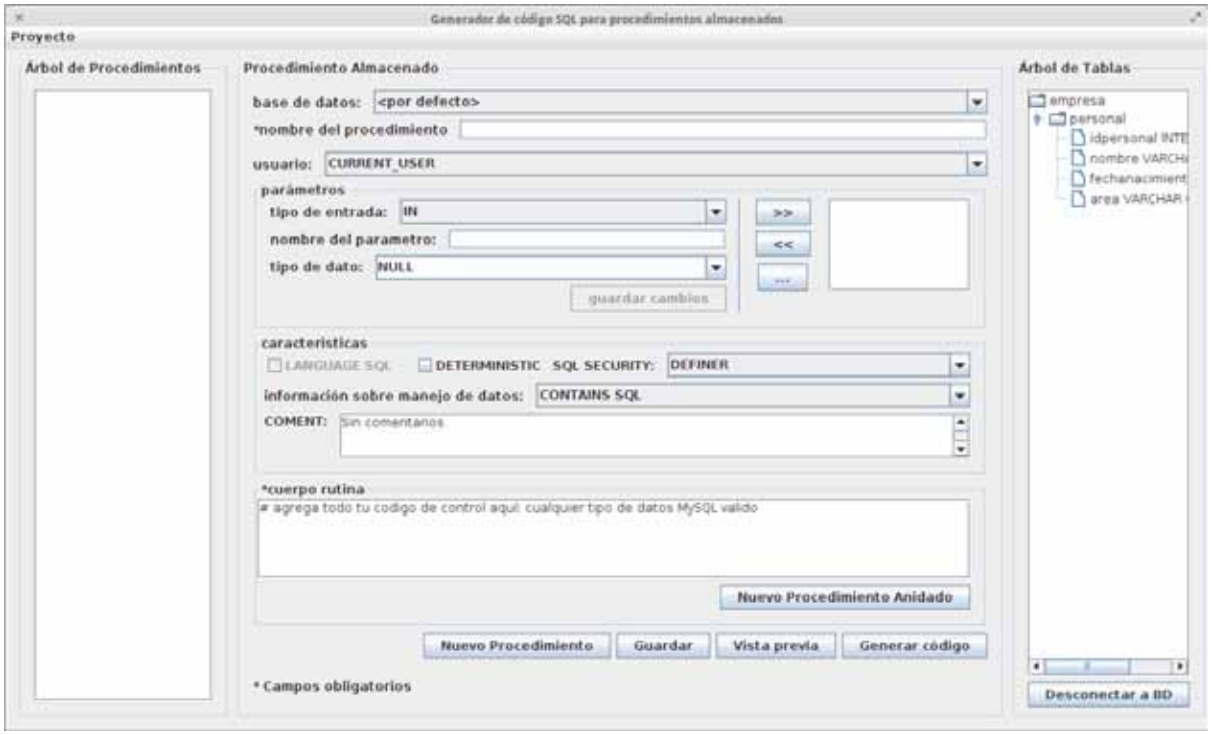


Figura 35: Ventana principal al establecer una conexión a una base de datos.

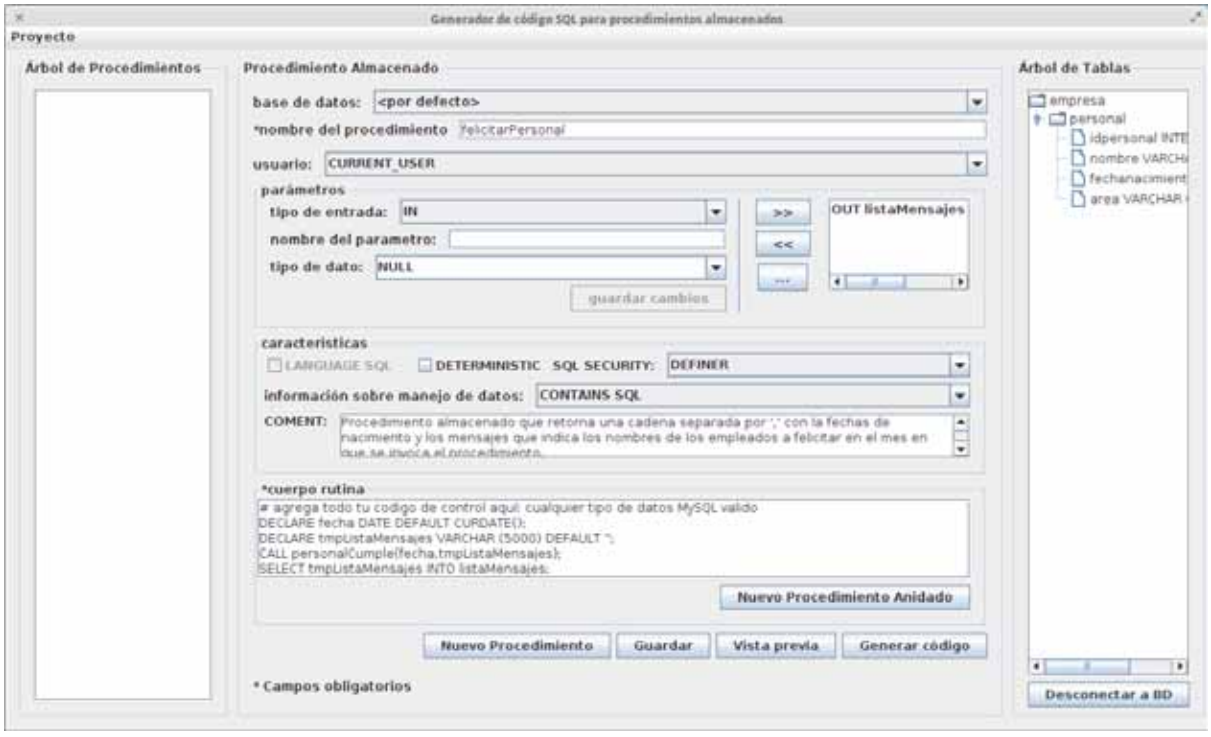
3. Diseñar el procedimiento almacenado llamado felicitarPersonal, con un parámetro de salida que retorna una cadena separada por ',' con la fechas de nacimiento y los mensajes que indica los nombres de los empleados a felicitar.

    a) Necesita obtener la fecha en el momento en que es invocado el procedimiento, para ello usa la función CURDATE().

    b) Necesita construir una sentencia que invoque a un procedimiento almacenado de nombre personalCumple, con los siguientes parámetros: un parámetro de entrada por el cual proporcionara la fecha obtenida, un parámetro de salida que retorna una cadena separada por ',' con las fechas de nacimiento y los mensajes que indica los nombres de los empleados a felicitar.

    c) La lista de mensajes que retorna el procedimiento personalCumple la enviamos en el parámetro de salida del procedimiento felicitarPersonal.

    Lo descrito anteriormente se ilustra en la Figura 36.

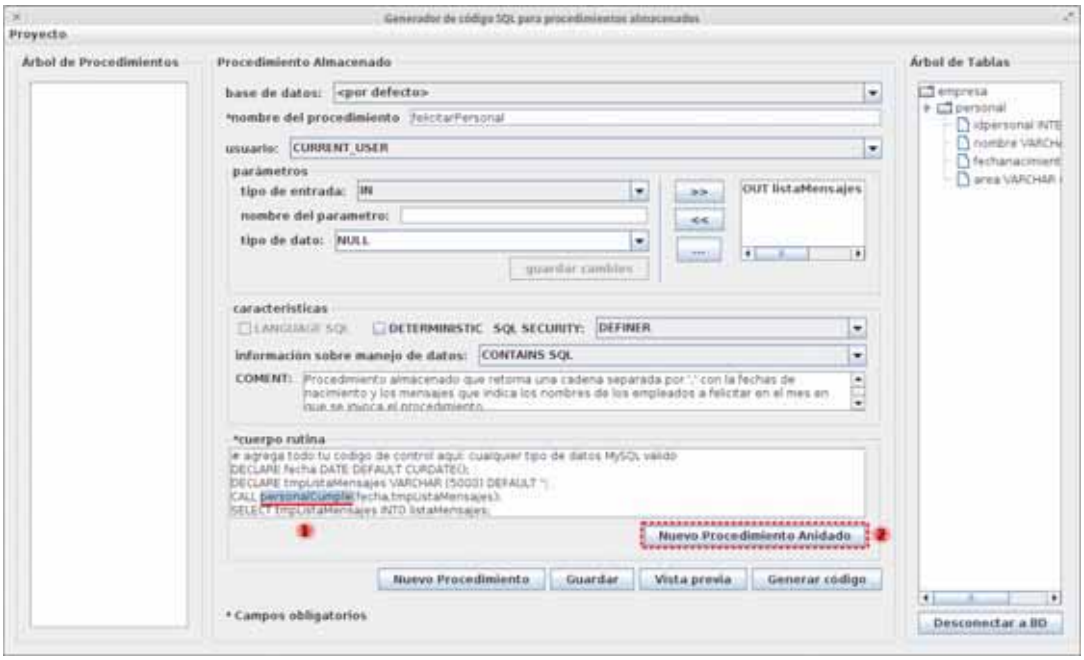


'

Figura 36: Diseño del procedimiento almacenado "felicitarPersonal".

4. Diseñar el procedimiento almacenado llamado personalCumple, con un parámetro de entrada que recibe una fecha y un parámetro de salida que retorna una cadena separada por ',' con la fechas de nacimiento y los mensajes que indica los nombres de los empleados a felicitar, filtrador por la fecha proporcionada.

    a) Seleccionamos únicamente el nombre del procedimiento almacenado y damos clic en el botón "Nuevo Procedimiento Anidado" (ver Figura 37).



‘

Figura 37: Establecer el nuevo procedimiento “personalCumple” anidado.

b) Se necesitan declarar las variables que almacenarán los saludos, un indicador de fin de registros, variable temporal para almacenar el nombre del empleado, fecha de nacimiento, variable temporal que almacenara el saludo, declarar un cursor que recorra el resultado de la consulta, declarar un manejador que nos alerte cuando se llegue al último registro, un ciclo que nos permita iterar a través del conjunto de resultados devuelto por la consulta, y un procedimiento almacenado que asocie el nombre y un saludo. (Ver Figura 38).

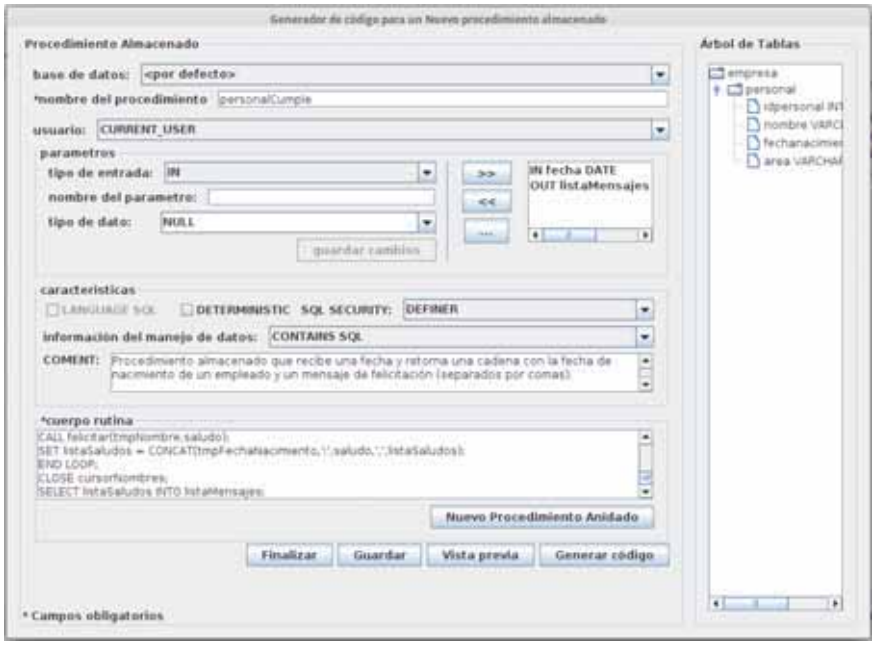


Figura 38: Diseño del procedimiento “personalCumple” anidado.

5. Diseñar el procedimiento almacenado llamado felicitar, con un parámetro de entrada que recibe el nombre del empleado y un parámetro de salida que retorna el saludo asociado al nombre.

   a) Seleccionamos únicamente el nombre del procedimiento almacenado y damos clic en el botón “Nuevo Procedimiento Anidado” (ver Figura 39).

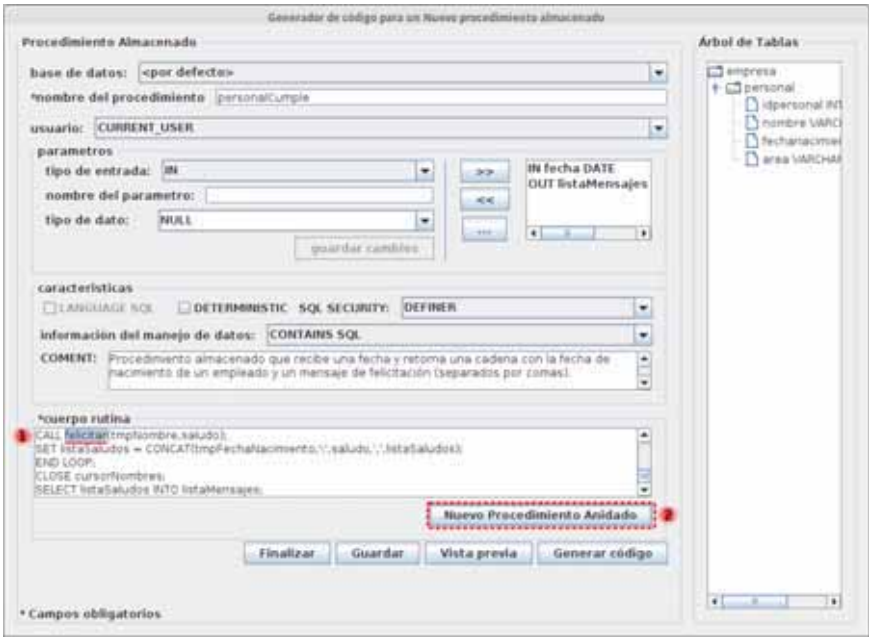


Figura 39: Establecer el nuevo procedimiento “felicitar” anidado.

   b) construir la sentencia que asocie el nombre del empleado con un saludo especifico y se retorne por el parámetro de salida del procedimiento almacenado felicitar. (Ver Figura 40).

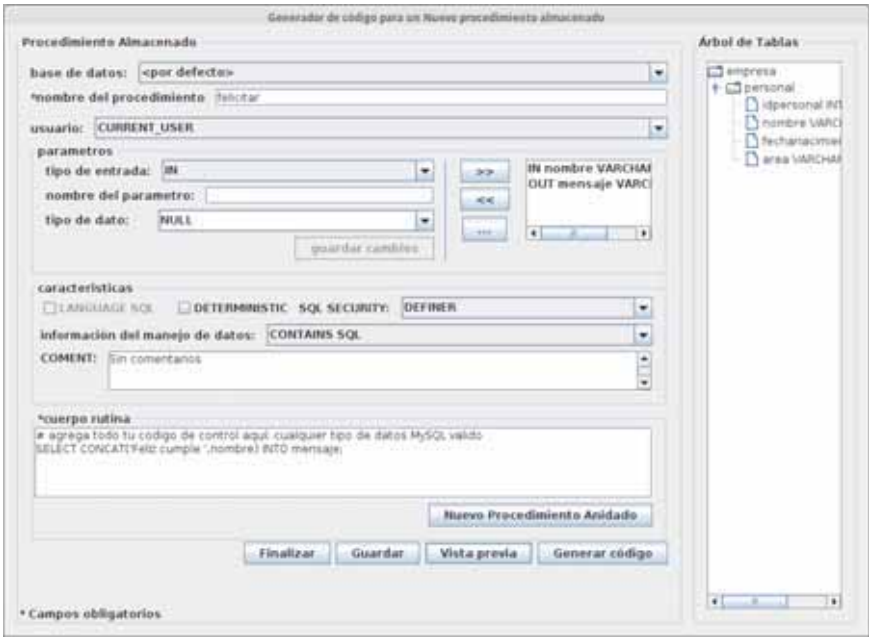


Figura 40: Diseño del procedimiento “felicitar” anidado.

6. guardar los procedimientos diseñados

   a) guardar y finalizar el procedimiento almacenado “felicitar” (ver Figura 41).

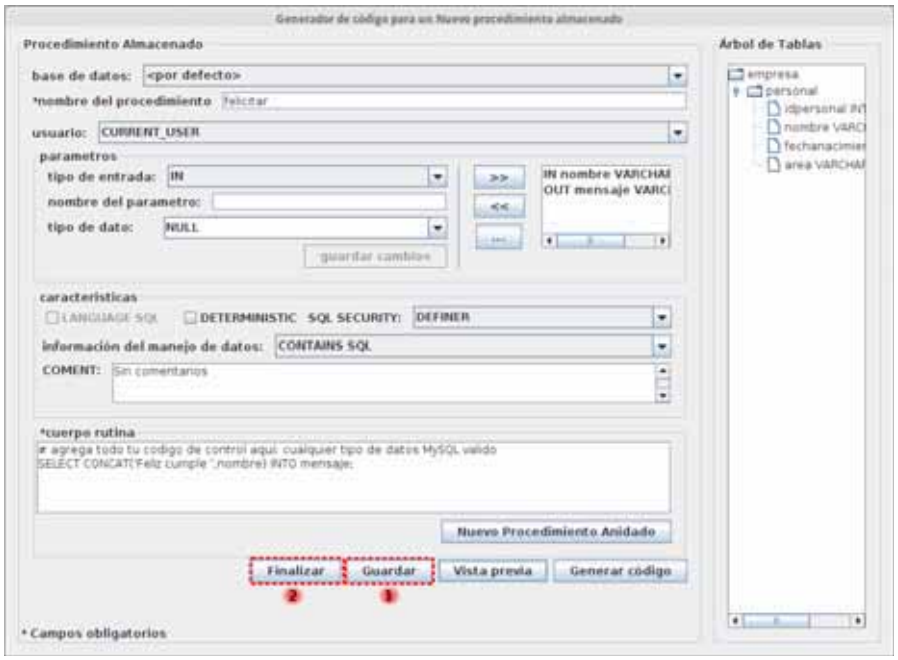


Figura 41: Guardar el procedimiento “felicitar” anidado.

   b) guardar y finalizar el procedimiento almacenado “personalCumple” (ver Figura 38).

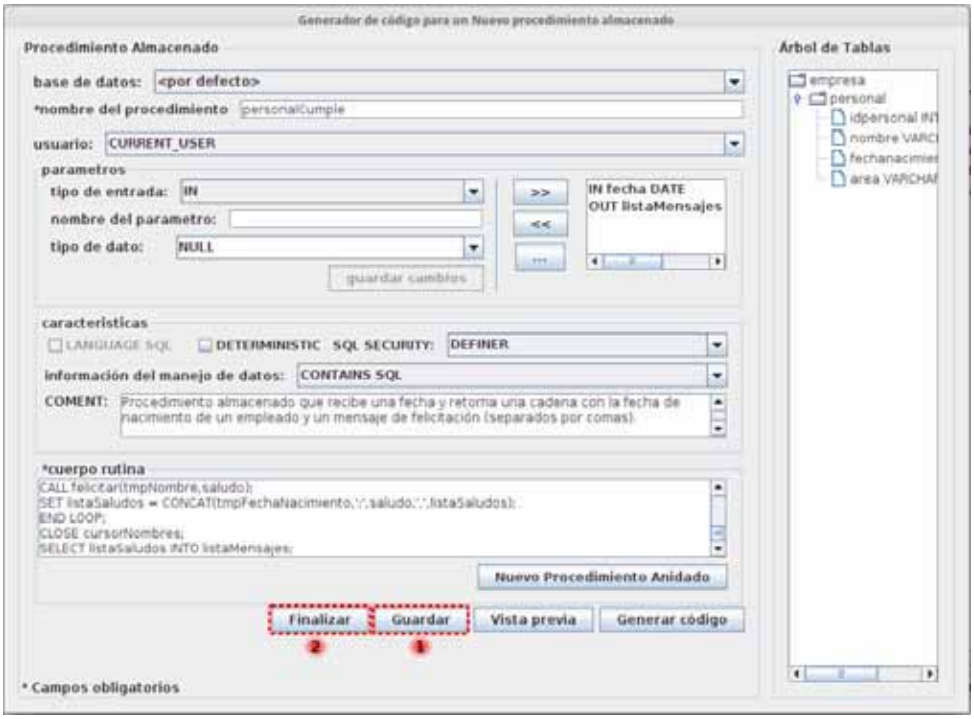


Figura 42: Guardar el procedimiento “personalCumple” anidado.

c) guardar el procedimiento almacenado “felicitarPersonal” (ver Figura 43).

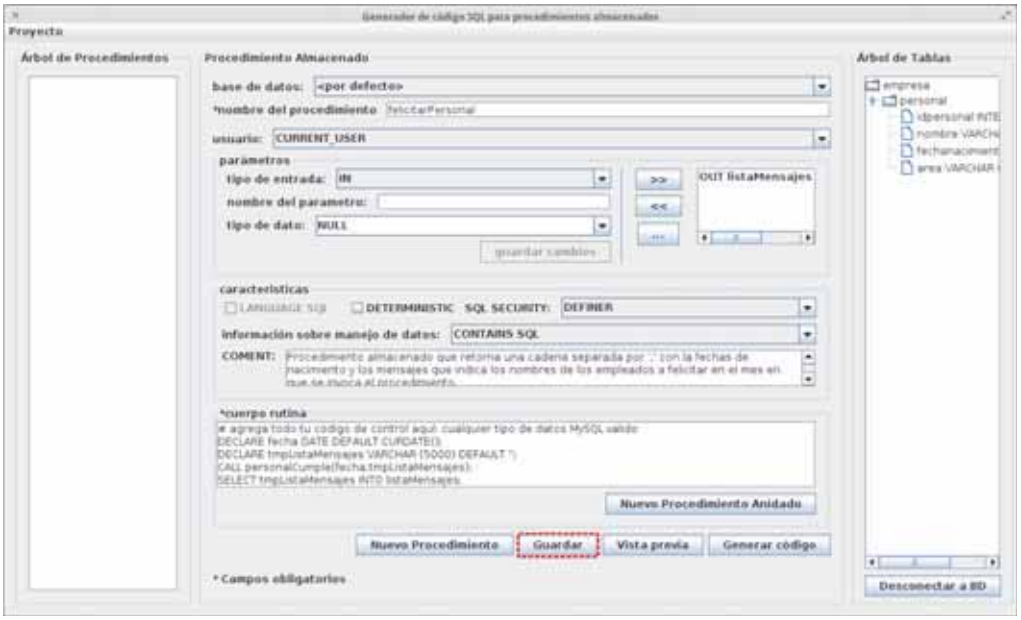


Figura 43: Guardar el procedimiento “felicitarPersonal”.

7. generar código SQL asociado, podemos apreciar que en el apartado visual “Árbol de procedimientos” se visualiza el anidamiento de procedimientos necesarios para el procedimiento “felicitarPersonal”, debemos situarnos en el procedimiento principal quien invoca. Al dar clic en el botón “Generar código” se notificara si existe invocación de procedimientos y si desea generar su código SQL asociado (presionamos si-*yes*). Podremos visualizar el código SQL (Script) asociado al procedimiento “felicitarPersonal” y los procedimientos que invoca, pudiendo guardar el código en un archivo para cargarlo desde la consola del manejador (ver Figura 44).

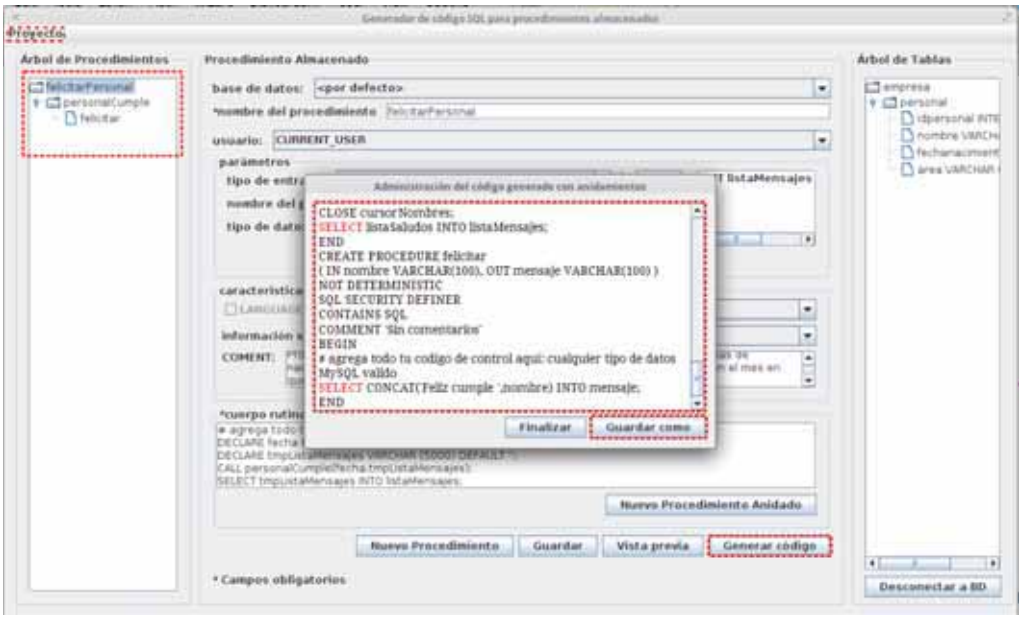


Figura 44: código SQL asociado al procedimiento “felicitarPersonal” y sus invocaciones a otros procedimientos.

Puede administrar el código SLQ asociado a los procedimientos por separado si así lo requiere, tendría que seleccionar la opción “No” cuando da clic en el botón “Generar código”. Como opción adicional puede dirigirse al “Proyecto” para poder guardar/abrir el proyecto, recuerde guardar los cambios que realice y guardar el proyecto para poder garantizar que se guardaron los cambios realizados.

El ejemplo anterior ilustra al procedimiento “felicitarPersonal” de nivel 2 (considerando que el procedimiento que no realiza ninguna invocación es de nivel 0), ya que invoca a otro procedimiento denominado “felicitarPersonal” de nivel 1 (considerando que el procedimiento invoca a “felicitar”) para su construcción. La herramienta permite construir procedimientos con “n” nivel de anidamiento que necesite el usuario.

Resultados

Se desarrolló e implementó una herramienta CASE con interfaz gráfica de usuario para la generación de procedimientos almacenados, para el manejador de bases de datos MySQL. De la cual se pudieron obtener:

- procedimientos almacenados con parámetros y configuraciones proporcionados por el usuario.
- procedimientos almacenados que implementan alguna función básica en bases de datos, como: Altas[4], Bajas, Cambios y Consultas.
- Visualización de invocaciones a otros procedimientos almacenados dentro la rutina de un procedimiento.

[4] Para los procedimientos almacenados que implementan la función Alta de una cierta tabla, esta debe de tener la propiedad autoincrementable en la columna que contiene las llaves primarias; para tener una mejor experiencia de uso

Análisis y discusión de resultados

Al proponer este proyecto de integración, se tenía planeado incluir un nivel de anidamiento en las configuraciones para la generación de un procedimiento. Esto con el fin de poder marcar un control sobre la creación de procedimientos. Al diseñar la interfaz de usuario, se determinó que no era necesario incluir este nivel de manera explícita y fatigar al usuario con ajustes y configuraciones.

Cuando se diseña un procedimiento almacenado, la mayoría de las veces no se sabe si se necesita la invocación a otro, por lo que sería tedioso configurar un proyecto en el generador para ciertos niveles de anidamientos que se tenían pensados y en realidad estos no fueron ya sea los necesarios o los suficientes.

Por lo que al final se decidió optar por implementar el nivel de anidamiento como una función implícita, es decir, no solicitando al usuario establecer un nivel de anidamiento, sino, que estés se establezca cuando se requiera, permitiendo así una mayor flexibilidad en el diseño de procedimientos que incluyen invocaciones a otros.

Conclusiones

- Se diseñó e implementó una aplicación de escritorio con interfaz gráfica de usuario, que facilitó la generación de código de procedimientos almacenados para el manejador de bases de datos MySQL.
- Se diseñó e implementó un módulo de configuración de parámetros requeridos en la generación de código de procedimientos almacenados para el manejador de bases de datos MySQL.
- Se diseñó e implementó un módulo de generación de código de procedimientos almacenados para el manejador de bases de datos MySQL.

Apéndice A

Documentación de espacios de nombres

A.1. Paquetes Control

Clases

- class controlCodigoAnidadoGenerado
- class controlCodigoProcedimientoGenerado
- class controlConexion
- class controlNuevoProcedimiento
- class controlProcedimiento
- class controlProcedimientoCRUD
- class MiFocusTraversalPolicy

A.2. Paquetes Modelo

Clases

- class Caracteristica
- class Caracteristicas
- class columna
- class columnas
- class ConexionBD
- class conexionManejadorBD
- class esquemaBD
- enum funcionBasicaBD
- class Parametro
- class Parametros
- class ProcedimientoAlmacenado
- class procedimientos
- class tabla
- class tablas
- enum tipoTabla
- class Usuario

A.3. Paquetes Programa

Clases

- class GeneradorProcedimientosAlmacenados

A.4. Paquetes Vistas

Clases

- class EditorCodigoAnidadoGenerado
- class EditorCodigoGenerado
- class EditorProcedimiento
- class EditorProcedimientoCRUD
- class nuevoProcedimiento
- class VentanaConexion

Apéndice B

Documentación de las clases

B.1. Referencia de la Clase Modelo.Caracteristica

Diagrama de herencias de Modelo.Caracteristica

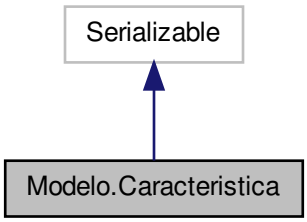
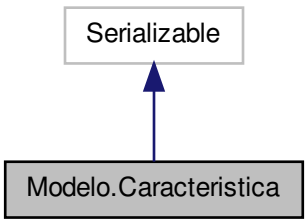


Diagrama de colaboración para Modelo.Caracteristica:



Métodos públicos

- void setComment (String comment)
- String getComment ()
- void setValue (String value)
- String getValue ()
- Caracteristica ()
- Caracteristica (String value)
- Caracteristica (String value, String comment)
- String generarSQL ()

Atributos públicos

- String value
- String comment

B.1.1. Descripción detallada

Clase que implementa la característica de un procedimiento almacenado

Autor

ivan

Definición en la línea 15 del archivo Caracteristica.java.

B.1.2. Documentación del constructor y destructor

B.1.2.1. Modelo.Caracteristica.Caracteristica ( )

Constructor de la clase característica

Definición en la línea 65 del archivo Caracteristica.java.

```
65                                {
66     }
```

B.1.2.2. Modelo.Caracteristica.Caracteristica (String *value*)

Sobrecarga del constructor de la característica, establece el valor de una característica

Parámetros

value	Valor de una característica

Definición en la línea 74 del archivo Caracteristica.java.

```
74                                        {
75         this.setValue(value);
76     }
```

B.1.2.3. Modelo.Caracteristica.Caracteristica (String *value,* String *comment*)

Sobrecarga del constructor de la característica, establece el valor y el comentario de una caracteristica

Parámetros

value	Valor de la característica
comment	Comentario de la característica

Definición en la línea 85 del archivo Caracteristica.java.

```
85                                                      {
86         if (value.equals("COMMENT")) {
87             this.value = value;
88             this.comment = comment;
89         } else {
90             this.value = value;
91             this.comment = comment;
92         }
93     }
```

B.1.3. Documentación de las funciones miembro

B.1.3.1. String Modelo.Caracteristica.generarSQL ( )

Método que genera el código SQL de una característica

Devuelve

String Código SQL generado.

Definición en la línea 100 del archivo Caracteristica.java.

```
100                                 {
101         if (this.comment == null) {
102             return this.value;
103         } else {
104             if (value.equals("COMMENT")) {
105                 return this.value + " " + "'" + this.comment + "'";
106             } else {
107                 return this.value + " " + this.comment;
108             }
109         }
110     }
```

B.1.3.2. String Modelo.Caracteristica.getComment ( )

Método que obtienen el comentario de una característica

Devuelve

String Comentario de la característica.

Definición en la línea 40 del archivo Caracteristica.java.

```
40                                  {
41         return this.comment;
42     }
```

B.1.3.3. String Modelo.Caracteristica.getValue ( )

Método que obtiene el valor de una característica

Devuelve

String Valor de la característica.

Definición en la línea 58 del archivo Caracteristica.java.

```
58                              {
59          return this.value;
60      }
```

B.1.3.4. void Modelo.Caracteristica.setComment (String *comment*)

Método que establece el comentario a una característica

Parámetros

comment	Comentario de una característica.

Definición en la línea 31 del archivo Caracteristica.java.

```
31                                          {
32          this.comment = comment;
33      }
```

B.1.3.5. void Modelo.Caracteristica.setValue (String *value*)

Método que establece el valor de una característica

Parámetros

value	Valor de la característica

Definición en la línea 49 del archivo Caracteristica.java.

```
49                                        {
50          this.value = value;
51      }
```

B.1.4. Documentación de los datos miembro

B.1.4.1. String Modelo.Caracteristica.comment

Comentario de la característica

Definición en la línea 24 del archivo Caracteristica.java.

B.1.4.2. String Modelo.Caracteristica.value

Valor/Nombre de la característica

Definición en la línea 20 del archivo Caracteristica.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/Caracteristica.java

B.2. Referencia de la Clase Modelo.Caracteristicas

Diagrama de herencias de Modelo.Caracteristicas

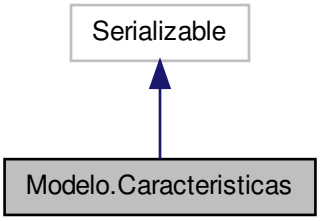
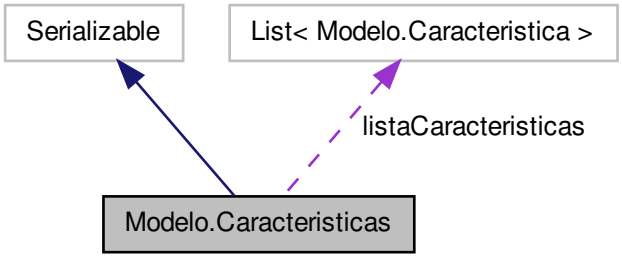


Diagrama de colaboración para Modelo.Caracteristicas:



Métodos públicos

- List getListCarac ()
- void setListCarac (List listaCaracteristicas)
- Caracteristicas ()
- boolean agregarCaracteristica (Caracteristica C)
- boolean quitarcaracteristica (Caracteristica C)
- int numCaracteristicas ()
- boolean listaCaracVacia ()
- Caracteristica obtenerCaracteristica (int indice)
- String generarSQL ()

Atributos privados

- List< Caracteristica > listaCaracteristicas

B.2.1. Descripción detallada

Clase que implementa una lista de características

Autor

ivan

Definición en la línea 17 del archivo Caracteristicas.java.

B.2.2. Documentación del constructor y destructor

B.2.2.1. Modelo.Caracteristicas.Caracteristicas ( )

Constructor de la clase por defecto

Definición en la línea 45 del archivo Caracteristicas.java.

```
45                              {
46         this.listaCaracteristicas = new ArrayList();
47         this.agregarCaracteristica(new Caracteristica("NOT DETERMINISTIC"));
48         this.agregarCaracteristica(new Caracteristica("SQL SECURITY", "DEFINER"));
49         this.agregarCaracteristica(new Caracteristica("CONTAINS SQL"));
50         this.agregarCaracteristica(new Caracteristica("COMMENT", "sin comentarios"));
51     }
```

B.2.3. Documentación de las funciones miembro

B.2.3.1. boolean Modelo.Caracteristicas.agregarCaracteristica (Caracteristica *C*)

Método que agrega una característica a la lista.

Parámetros

C	Característica

Devuelve

boolean

- true La característica se agregó a la lista
- false La característica no se se argegó a la lista

Definición en la línea 63 del archivo Caracteristicas.java.

```
63                                                      {
64         return this.listaCaracteristicas.add(C);
65     }
```

B.2.3.2. String Modelo.Caracteristicas.generarSQL ( )

Método que genera código SQL de la lista de características y lo retorna

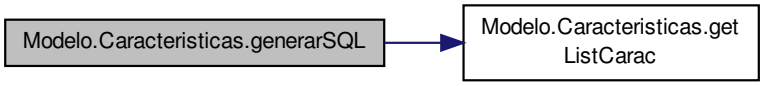
Devuelve

String Código SQL asociado a la lista de características

Definición en la línea 118 del archivo Caracteristicas.java.

```
118                                {
119         Caracteristica item;
120         String SQL = "";
121         if (!this.listaCaracteristicas.isEmpty()) {
122             for (int i = 0; i < this.getListCarac().size() - 1; i++) {
123                 item = (Caracteristica) this.getListCarac().get(i);
124                 SQL = SQL + item.generarSQL() + "\n";
125             }
126             item = (Caracteristica) this.getListCarac().get(this.getListCarac().size() - 1);
127             SQL = SQL + item.generarSQL();
128         }
129         return SQL;
130     }
```

Gráfico de llamadas para esta función:



B.2.3.3. List Modelo.Caracteristicas.getListCarac ( )

Método que obtiene la lista de características

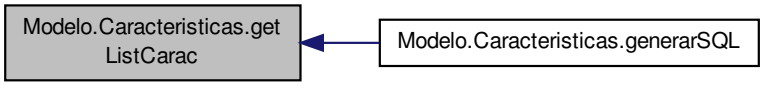
Devuelve

List Lista de características

Definición en la línea 29 del archivo Caracteristicas.java.

```
29                                {
30         return this.listaCaracteristicas;
31     }
```

Gráfico de llamadas a esta función:



B.2.3.4. boolean Modelo.Caracteristicas.listaCaracVacia ( )

Método que verifica si la lista está vacía

Devuelve

boolean

- true la lista está vacía
- false la lista contiene almenos una característica

Definición en la línea 99 del archivo Caracteristicas.java.

```
99                                      {
100         return this.listaCaracteristicas.isEmpty();
101     }
```

B.2.3.5. int Modelo.Caracteristicas.numCaracteristicas ( )

Método que retorna el número de de características en la lista

Devuelve

int Número de características

Definición en la línea 86 del archivo Caracteristicas.java.

```
86                                      {
87         return this.listaCaracteristicas.size();
88     }
```

B.2.3.6. Caracteristica Modelo.Caracteristicas.obtenerCaracteristica (int *indice*)

Método que obtiene una característica a partir del indice proporcionado

Parámetros

indice	Indice

Devuelve

Caracteristica Característica

Definición en la línea 109 del archivo Caracteristicas.java.

```
109                                                     {
110         return this.listaCaracteristicas.get(indice);
111     }
```

B.2.3.7. boolean Modelo.Caracteristicas.quitarcaracteristica (Caracteristica *C*)

Método que remueve una característica de la lista.

Parámetros

C	Característica

Devuelve

boolean

- true La característica se removió de la lista
- false La característica no se se removió de la lista

Definición en la línea 77 del archivo Caracteristicas.java.

```
77                                                  {
78         return this.listaCaracteristicas.remove(C);
79     }
```

B.2.3.8. void Modelo.Caracteristicas.setListCarac (List *listaCaracteristicas*)

Método que establece una lista de características

Parámetros

lista-Caracteristicas	lista de carcaterísticas

Definición en la línea 38 del archivo Caracteristicas.java.

```
38                                                  {
39         this.listaCaracteristicas = listaCaracteristicas;
40     }
```

B.2.4. Documentación de los datos miembro

B.2.4.1. List<Caracteristica> Modelo.Caracteristicas.listaCaracteristicas [private]

Lista de características

Definición en la línea 22 del archivo Caracteristicas.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/Caracteristicas.java

B.3. Referencia de la Clase Modelo.columna

Métodos públicos

- columna ()
- columna (String nombreColumna, String tipoDato)
- columna (Boolean llavePrimaria, String nombreColumna, String tipoDato)
- columna (Boolean llavePrimaria, String nombreColumna, String tipoDato, int longitudColumna)
- void setLlavePrimaria (boolean llavePrimaria)
- boolean getLlavePrimaria ()

- void setCandidatoSP (boolean candidatoSP)
- boolean getCandidatoSP ()
- void setNombreColumna (String nombreColumna)
- String getNombreColumna ()
- void setTipoDato (String tipoDato)
- String getTipoDato ()
- void setLongitudColumna (int longitudColumna)
- int getLongitudColumna ()

Atributos públicos

- String nombreColumna
- String tipoDato
- boolean candidatoSP
- boolean llavePrimaria

Atributos privados

- int longitudColumna

B.3.1. Descripción detallada

Clase que implementa una columna de una tabla

Autor

ivan

Definición en la línea 13 del archivo columna.java.

B.3.2. Documentación del constructor y destructor

B.3.2.1. Modelo.columna.columna ( )

Constructor por defecto de la clase columna

Definición en la línea 48 del archivo columna.java.

```
48                          {
49     }
```

B.3.2.2. Modelo.columna.columna (String *nombreColumna,* String *tipoDato*)

Sobrecarga del constructor de la clase columna, establece el nombre de la columna y el tipo de dato

Parámetros

nombreColumna	Nombre de la columna
tipoDato	Tipo de dato

Definición en la línea 58 del archivo columna.java.

```
58                                                      {
59         this.setNombreColumna(nombreColumna);
60         this.setTipoDato(tipoDato);
61         this.setCandidatoSP(true);
62     }
```

B.3.2.3. Modelo.columna.columna (Boolean *llavePrimaria,* String *nombreColumna,* String *tipoDato*)

Sobrecarga del constructor de la clase columna, establece la propiedad llave, el nombre y el tipo de dato de una columna

Parámetros

llavePrimaria	propiedad de la columna que determina si es llave primaria ■ true es llave primaria ■ false no es llave primaria
nombreColumna	Nombre de la columna
tipoDato	Tipo de dato de la columna

Definición en la línea 77 del archivo columna.java.

```
77                                                                      {
78         this.setNombreColumna(nombreColumna);
79         this.setTipoDato(tipoDato);
80         this.setCandidatoSP(true);
81         this.setLlavePrimaria(llavePrimaria);
82     }
```

B.3.2.4. Modelo.columna.columna (Boolean *llavePrimaria,* String *nombreColumna,* String *tipoDato,* int *longitudColumna*)

Sobrecarga del constructor de la clase columna, establece la propiedad llave, el nombre, el tipo de dato y la precisión/longitud del tipo de dato de una columna

Parámetros

llavePrimaria	propiedad de la columna que determina si es llave primaria ■ true es llave primaria ■ false no es llave primaria

nombreColumna	Nombre de la columna
tipoDato	Tipo de dato de la columna
longitudColumna	Precisión del tipo de dato de la columna

Definición en la línea 99 del archivo columna.java.

```
99                                                                                  {
100         this.setNombreColumna(nombreColumna);
101         this.setTipoDato(tipoDato);
102         this.setLongitudColumna(longitudColumna);
103         this.setCandidatoSP(true);
104         this.setLlavePrimaria(llavePrimaria);
105     }
```

B.3.3. Documentación de las funciones miembro

B.3.3.1. boolean Modelo.columna.getCandidatoSP ( )

Método que proporciona si la columna es candidata a ser parámetro en un procedimiento almacenado.

Devuelve

boolean

- true es candidata a ser parámetro en un procedimiento almacenado
- false no es candidata a ser parámetro en un procedimiento almacenado

Definición en la línea 158 del archivo columna.java.

```
158                                     {
159         return this.candidatoSP;
160     }
```

B.3.3.2. boolean Modelo.columna.getLlavePrimaria ( )

Método que obtiene la propiedad llave primaria de la columna

Devuelve

boolean

- true es llave primaria
- false no es llave primaria

Definición en la línea 130 del archivo columna.java.

```
130                                     {
131         return this.candidatoSP;
132     }
```

B.3.3.3. int Modelo.columna.getLongitudColumna ( )

Método que obtiene la precisión del tipo de dato de la columna

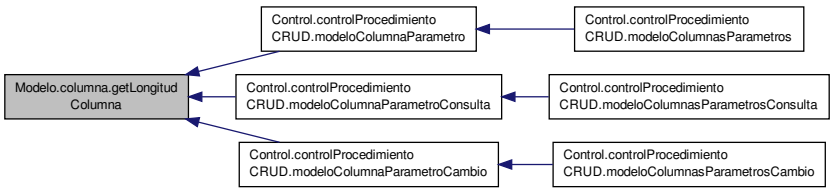
Devuelve

int Precisión del tipo de dato

Definición en la línea 212 del archivo columna.java.

```
212                                     {
213         return this.longitudColumna;
214     }
```

Gráfico de llamadas a esta función:



B.3.3.4. String Modelo.columna.getNombreColumna ( )

Método que obtiene el nombre de la columna

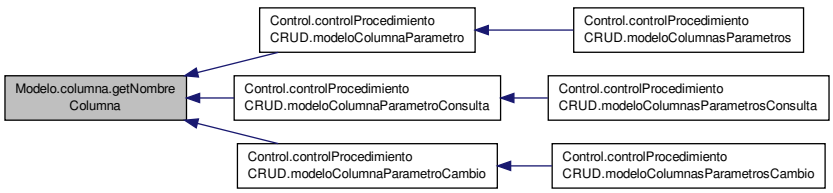
Devuelve

String Nombre de la columna

Definición en la línea 176 del archivo columna.java.

```
176                                     {
177         return this.nombreColumna;
178     }
```

Gráfico de llamadas a esta función:



B.3.3.5. String Modelo.columna.getTipoDato ( )

Método que obtiene el tipo de dato de la columna

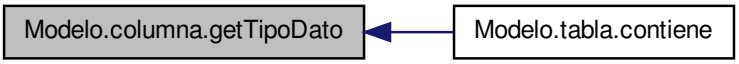
Devuelve

String Tipo de dato de la columna

Definición en la línea 194 del archivo columna.java.

```
194                                 {
195         return this.tipoDato;
196     }
```

Gráfico de llamadas a esta función:



B.3.3.6. void Modelo.columna.setCandidatoSP (boolean *candidatoSP*)

Método que establece si la columna es candidata a ser parámetro en un procedimiento almacenado

Parámetros

candidatoSP	■ true es candidata a ser parámetro en un procedimiento almacenado ■ false no es candidata a ser parámetro en un procedimiento almacenado

Definición en la línea 144 del archivo columna.java.

```
144                                                 {
145         this.candidatoSP = candidatoSP;
146     }
```

B.3.3.7. void Modelo.columna.setLlavePrimaria (boolean *llavePrimaria*)

Método que establece la propiedad llave primaria de la columna

Parámetros

llavePrimaria	propiedad de la columna que determina si es llave primaria ■ true es llave primaria ■ false no es llave primaria

Definición en la línea 117 del archivo columna.java.

```
117                                                 {
118         this.llavePrimaria = llavePrimaria;
119     }
```

B.3.3.8. void Modelo.columna.setLongitudColumna (int *longitudColumna*)

Método que establece la precisión del tipo de dato de la columna

Parámetros

longitudColumna	precisión del tipo de dato

Definición en la línea 203 del archivo columna.java.

```
203                                                 {
204         this.longitudColumna = longitudColumna;
205     }
```

B.3.3.9. void Modelo.columna.setNombreColumna (String *nombreColumna*)

Método que establece el nombre de la columna

Parámetros

nombreColumna	Nombre de la columna

Definición en la línea 167 del archivo columna.java.

```
167                                                 {
168         this.nombreColumna = nombreColumna;
169     }
```

B.3.3.10. void Modelo.columna.setTipoDato (String *tipoDato*)

Método que establece el tipo de dato de la columna

Parámetros

tipoDato	Tipo de dato de la columna

Definición en la línea 185 del archivo columna.java.

```
185                                         {
186         this.tipoDato = tipoDato;
187     }
```

B.3.4. Documentación de los datos miembro

B.3.4.1. boolean Modelo.columna.candidatoSP

propiedad de la columna que determina si es candidata a ser parámetro en un procedimiento almacenado

- true es candidata a ser parámetro en un procedimiento almacenado
- false no es candidata a ser parámetro en un procedimiento almacenado

Definición en la línea 35 del archivo columna.java.

B.3.4.2. boolean Modelo.columna.llavePrimaria

propiedad de la columna que determina si es llave primaria

- true es llave primaria
- false no es llave primaria

Definición en la línea 43 del archivo columna.java.

B.3.4.3. int Modelo.columna.longitudColumna [private]

Precisión del tipo de dato de la columna

Definición en la línea 26 del archivo columna.java.

B.3.4.4. String Modelo.columna.nombreColumna

Nombre de la columna

Definición en la línea 18 del archivo columna.java.

B.3.4.5. String Modelo.columna.tipoDato

Tipo de dato de la columna

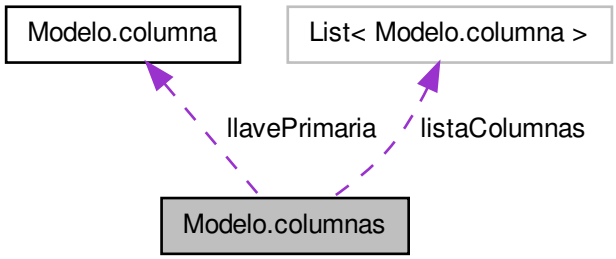
Definición en la línea 22 del archivo columna.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/columna.java

B.4. Referencia de la Clase Modelo.columnas

Diagrama de colaboración para Modelo.columnas:



Métodos públicos

- columnas ()
- void setLLavePrimaria (columna llavePrimaria)
- columna getLlavePrimaria ()
- void setListaColumnas (List listaColumnas)
- List getListaColumnas ()
- boolean agregarColumna (columna C)
- int numeroColumnas ()
- boolean listaColumnasVacia ()
- columna getColumna (int indice)
- columna getColumna (String nombreColumna)
- boolean llavePrimaria ()
- columnas getColumnasSP ()
- void limpiar ()

Atributos públicos

- List< columna > listaColumnas
- columna llavePrimaria

B.4.1. Descripción detallada

Clase que implementa una lista de columnas

Autor

ivan

Definición en la línea 16 del archivo columnas.java.

B.4.2. Documentación del constructor y destructor

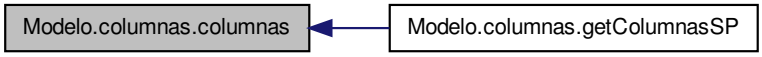
B.4.2.1. Modelo.columnas.columnas ( )

Constructor por defecto de la clase

Definición en la línea 30 del archivo columnas.java.

```
30                      {
31         this.setListaColumnas(new ArrayList());
32     }
```

Gráfico de llamadas a esta función:



B.4.3. Documentación de las funciones miembro

B.4.3.1. boolean Modelo.columnas.agregarColumna (columna *C*)

Metodo que agrega una columna a la lista

Parámetros

C	Columna

Devuelve

boolean

- true la columna se agrego a la lista
- false la columna bo se agrego a la lista

Definición en la línea 80 del archivo columnas.java.

```
80                                          {
81          return this.listaColumnas.add(C);
82      }
```

B.4.3.2. columna Modelo.columnas.getColumna (int *indice*)

Método que obtiene la columna asociada al indice proporcionado

Parámetros

indice	indice asociado a una columna

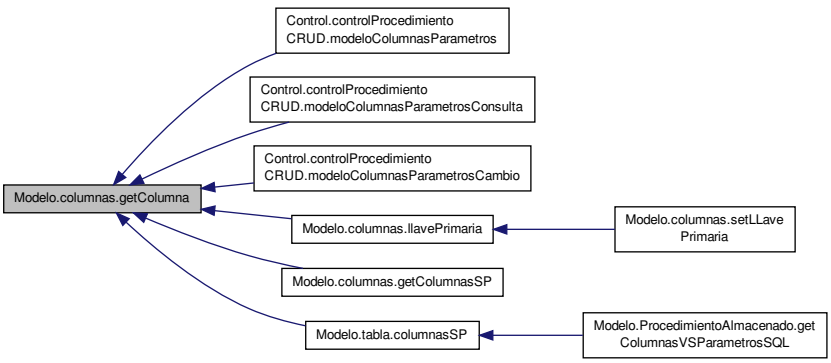
Devuelve

columna Columna

Definición en la línea 112 del archivo columnas.java.

```
112                                         {
113         return this.listaColumnas.get(indice);
114     }
```

Gráfico de llamadas a esta función:



B.4.3.3. columna Modelo.columnas.getColumna (String *nombreColumna*)

Sobrecarga del método getColumna, obtiene la columna asociada al nombre proporcionado

Parámetros

nombreColumna	Nombre de la columna

Devuelve

columna Columna

Definición en la línea 123 del archivo columnas.java.

```
123                                                 {
124         columna tmpColumna = null;
125         for (columna tmp : this.listaColumnas) {
126             if (tmp.getNombreColumna().equals(nombreColumna)) {
127                 tmpColumna = tmp;
128                 break;
129             }
130         }
131         return tmpColumna;
132     }
```

B.4.3.4. columnas Modelo.columnas.getColumnasSP ( )

Método que obtiene una lista de columnas que son candiatas a ser parámetros en un procedimiento almacenado

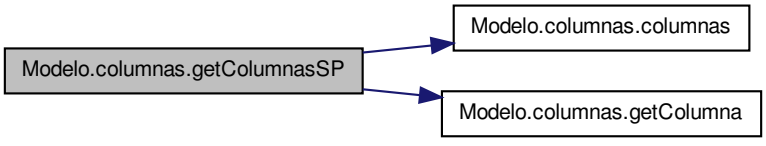
Devuelve

columnas Lista de columnas

Definición en la línea 161 del archivo columnas.java.

```
161                                     {
162         columnas tmp = new columnas();
163         for (int i = 0; i < this.numeroColumnas(); i++) {
164             if (this.getColumna(i).candidatoSP) {
165                 tmp.agregarColumna(this.getColumna(i));
166             }
167         }
168         return tmp;
169     }
```

Gráfico de llamadas para esta función:



B.4.3.5. List Modelo.columnas.getListaColumnas ( )

Método que obtiene la lista de columnas

Devuelve

List Lista de columnas

Definición en la línea 66 del archivo columnas.java.

```
66                                      {
67          return this.listaColumnas;
68      }
```

B.4.3.6. columna Modelo.columnas.getLlavePrimaria ( )

Método que obtiene la columna llave primaria

Devuelve

columna Columna que es llave primaria

Definición en la línea 48 del archivo columnas.java.

```
48                                      {
49          return this.llavePrimaria;
50      }
```

B.4.3.7. void Modelo.columnas.limpiar ( )

Método que limpia la lista de columnas

Definición en la línea 174 del archivo columnas.java.

```
174                             {
175          this.listaColumnas.clear();
176          this.llavePrimaria = null;
177      }
```

B.4.3.8. boolean Modelo.columnas.listaColumnasVacia ( )

Método que verifica si la lista de columnas se encuentra vacía

Devuelve

boolean

- true la lista de columnas está vacía
- false la lista contiene almenos una columna

Definición en la línea 102 del archivo columnas.java.

```
102                                     {
103          return this.listaColumnas.isEmpty();
104      }
```

B.4.3.9. boolean Modelo.columnas.llavePrimaria ( )

Método que verifica si la lista de columnas contiene una columna que es llave primaria

Devuelve

boolean

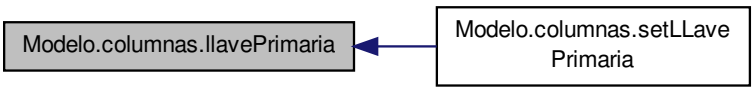
- true la lista contiene una columna que es llave primaria
- false la lista no contiene ninguna columna que sea llave primaria

Definición en la línea 144 del archivo columnas.java.

```
144                                     {
145         boolean tmp = false;
146         for (int i = 0; i < this.numeroColumnas(); i++) {
147             if (this.getColumna(i).llavePrimaria) {
148                 this.setLLavePrimaria(this.getColumna(i));
149                 tmp = true;
150             }
151         }
152         return tmp;
153     }
```

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:



B.4.3.10. int Modelo.columnas.numeroColumnas ( )

Método que proporciona el número de columnas en la lista

Devuelve

int Número de columnas en la lista

Definición en la línea 89 del archivo columnas.java.

```
89                                  {
90          return this.listaColumnas.size();
91      }
```

B.4.3.11. void Modelo.columnas.setListaColumnas (List *listaColumnas*)

Método que estable la lista de columnas

Parámetros

listaColumnas	Lista de columnas

Definición en la línea 57 del archivo columnas.java.

```
57                                                  {
58          this.listaColumnas = listaColumnas;
59      }
```

B.4.3.12. void Modelo.columnas.setLLavePrimaria (columna *llavePrimaria*)

Método que establece la columna llave primaria

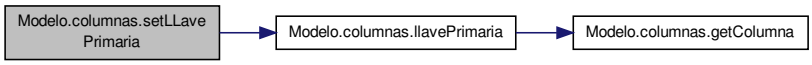
Parámetros

llavePrimaria	Columna que es llave primaria

Definición en la línea 39 del archivo columnas.java.

```
39                                                  {
40          this.llavePrimaria = llavePrimaria;
41      }
```

Gráfico de llamadas para esta función:



B.4.4. Documentación de los datos miembro

B.4.4.1. List<columna> Modelo.columnas.listaColumnas

Lista de columnas

Definición en la línea 21 del archivo columnas.java.

B.4.4.2. columna Modelo.columnas.llavePrimaria

Columna que es llave primaria

Definición en la línea 25 del archivo columnas.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/columnas.java

B.5. Referencia de la Clase Modelo.ConexionBD

Diagrama de herencias de Modelo.ConexionBD

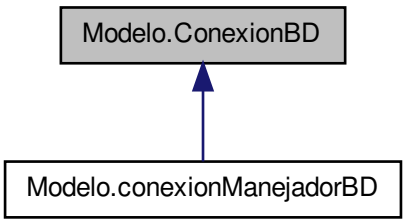
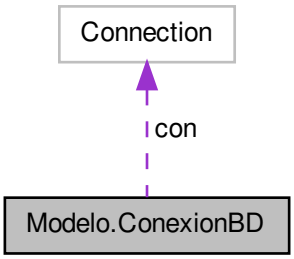


Diagrama de colaboración para Modelo.ConexionBD:



Métodos públicos

- Connection getConexion ()
- void setConexion (Connection con)

- String getdriveJDBC ()
- void setdriveJDBC (String driveJDBC)
- ConexionBD ()
- void CargarDriver (String driverJDBC)
- boolean CrearConexion (String url, String usuario, String password)
- void CerrarConexion ()

Atributos privados

- Connection con
- String driveJDBC

B.5.1. Descripción detallada

Clase que implementa la conexión a una base de datos

Autor

ivan

Definición en la línea 15 del archivo ConexionBD.java.

B.5.2. Documentación del constructor y destructor

B.5.2.1. Modelo.ConexionBD.ConexionBD ( )

Constructor por defecto de la clase

Definición en la línea 67 del archivo ConexionBD.java.

```
67                          {
68         this.driveJDBC = "jdbc";
69     }
```

B.5.3. Documentación de las funciones miembro

B.5.3.1. void Modelo.ConexionBD.CargarDriver (String *driverJDBC*)

Método que implementa la operación "Carga del driver"

Parámetros

driverJDBC	Clase del driver JDBC

Definición en la línea 76 del archivo ConexionBD.java.

```
76                                              {
77         try {
78             Class.forName(driverJDBC);
79         } catch (ClassNotFoundException e) {
80             e.printStackTrace();
81         }
82     }
```

B.5.3.2. void Modelo.ConexionBD.CerrarConexion ( )

Método que implementa la operación "Cierre de la conexión"

Definición en la línea 109 del archivo ConexionBD.java.

```
109                                 {
110         try {
111             this.con.close();
112         } catch (SQLException e) {
113             e.printStackTrace();
114         }
115     }
```

B.5.3.3. boolean Modelo.ConexionBD.CrearConexion (String *url,* String *usuario,* String *password*)

Método que implementa la operación "Creación de la conexión" a partir de los parametros proporcionados: URL, usuario y contraseña.

Parámetros

url	URL
usuario	usuario
password	Constraseña

Devuelve

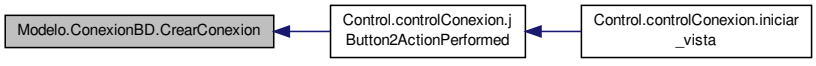
boolean

- true Se creó la conexión a la base de datos
- false No se creó la conexión a la base de datos

Definición en la línea 97 del archivo ConexionBD.java.

```
97                                                                                  {
98         try {
99             this.con = DriverManager.getConnection(url, usuario, password);
100         } catch (SQLException e) {
101             e.printStackTrace();
102         }
103         return (this.con != null);
104     }
```

Gráfico de llamadas a esta función:



B.5.3.4. Connection Modelo.ConexionBD.getConexion ( )

Método que obtiene la variable que contiene la conexión a una base de datos

Devuelve

Connection Conexión a una base de datos

Definición en la línea 32 del archivo ConexionBD.java.

```
32                                      {
33         return this.con;
34     }
```

B.5.3.5. String Modelo.ConexionBD.getdriveJDBC ( )

Método que obtiene la clase del driver

Devuelve

String Clase del driver JDBC

Definición en la línea 51 del archivo ConexionBD.java.

```
51                                     {
52         return this.driveJDBC;
53     }
```

B.5.3.6. void Modelo.ConexionBD.setConexion (Connection *con*)

Método que establece el valor a la variable que contiene la conexión a una base de datos

Parámetros

con	Conexión a una base de datos

Definición en la línea 42 del archivo ConexionBD.java.

```
42                                          {
43         this.con = con;
44     }
```

B.5.3.7. void Modelo.ConexionBD.setdriveJDBC (String *driveJDBC*)

Método que establece la clase del driver

Parámetros

driveJDBC	Clase del driver JDBC

Definición en la línea 60 del archivo ConexionBD.java.

```
60                                            {
61         this.driveJDBC = driveJDBC;
62     }
```

B.5.4. Documentación de los datos miembro

B.5.4.1. Connection Modelo.ConexionBD.con [private]

Variable que representa la conexión a una base de datos

Definición en la línea 20 del archivo ConexionBD.java.

B.5.4.2. String Modelo.ConexionBD.driveJDBC [private]

Variable que contiene la clase del driver a cargar

Definición en la línea 24 del archivo ConexionBD.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/ConexionBD.java

B.6. Referencia de la Clase Modelo.conexionManejadorBD

Diagrama de herencias de Modelo.conexionManejadorBD

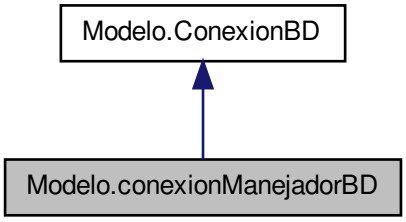
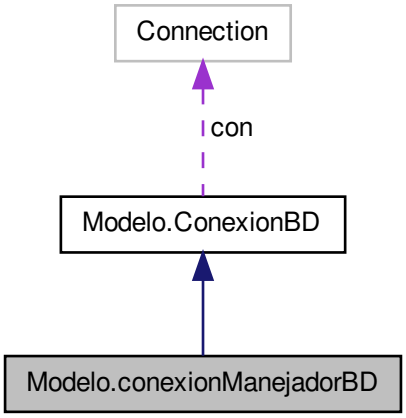


Diagrama de colaboración para Modelo.conexionManejadorBD:



Métodos públicos

- String getdriverJDBC ()
- void setdriverJDBC (String driverJDBC)
- String getsubProtocol ()
- void setsubProtocol (String subProtocol)
- String gethostName_ip ()
- void sethostName_ip (String hostName_ip)
- String getpuerto ()
- void setpuerto (String puerto)
- String getbase_datos ()
- void setbase_datos (String base_datos)
- String getusuario ()
- void setusuario (String usuario)
- String getpassword ()
- void setpassword (String password)
- conexionManejadorBD ()
- String generarURL ()
- void conectar (String url)

Atributos privados

- String driverJDBC
- String subProtocol
- String hostName_ip
- String puerto
- String base_datos
- String usuario
- String password

B.6.1. Descripción detallada

Clase que implementa una conexión a una base de datos del sistema gestor de bases de datos MySQL.

Autor

ivan

Definición en la línea 14 del archivo conexionManejadorBD.java.

B.6.2. Documentación del constructor y destructor

B.6.2.1. Modelo.conexionManejadorBD.conexionManejadorBD ( )

Constructor por defecto de la clase

Definición en la línea 174 del archivo conexionManejadorBD.java.

```
174                                 {
175         super();
176     }
```

B.6.3. Documentación de las funciones miembro

B.6.3.1. void Modelo.conexionManejadorBD.conectar (String *url*)

Método que establece la conexión a una base de datos con los parámetros proporcionados.

Parámetros

url	URL

Definición en la línea 194 del archivo conexionManejadorBD.java.

```
194                                     {
195         this.CargarDriver(driverJDBC);
196         this.CrearConexion(this.generarURL(), usuario, password);
197     }
```

B.6.3.2. String Modelo.conexionManejadorBD.generarURL ( )

Método que genera y retorna la URL para establecer conexión a una base de datos

Devuelve

String URL

Definición en la línea 184 del archivo conexionManejadorBD.java.

```
184                                    {
185         return subProtocol + ":" + "//" + hostName_ip + ":" +
      puerto + "/" + base_datos;
186     }
```

B.6.3.3. String Modelo.conexionManejadorBD.getbase_datos ( )

Método que obtiene el nombre de la base de datos

Devuelve

String Nombre de la base de datos

Definición en la línea 122 del archivo conexionManejadorBD.java.

```
122                                     {
123         return this.base_datos;
124     }
```

B.6.3.4. String Modelo.conexionManejadorBD.getdriverJDBC ( )

Método que obtiene el driver JDBC

Devuelve

String Driver JDBC

Definición en la línea 50 del archivo conexionManejadorBD.java.

```
50                                     {
51         return this.driverJDBC;
52     }
```

B.6.3.5. String Modelo.conexionManejadorBD.gethostName_ip ( )

Método que obtiene el nombre/dirección ip del host

Devuelve

String Nombre/dirección ip del host

Definición en la línea 86 del archivo conexionManejadorBD.java.

```
86                                  {
87         return this.hostName_ip;
88     }
```

B.6.3.6. String Modelo.conexionManejadorBD.getpassword ( )

Método que obtiene la contraseña asociada al usuario

Devuelve

String Contraseña

Definición en la línea 158 del archivo conexionManejadorBD.java.

```
158                                 {
159         return this.password;
160     }
```

B.6.3.7. String Modelo.conexionManejadorBD.getpuerto ( )

Método que obtiene el puerto de servicio del SGBD

Devuelve

String puerto de servicio del SGBD

Definición en la línea 104 del archivo conexionManejadorBD.java.

```
104                                 {
105         return this.puerto;
106     }
```

B.6.3.8. String Modelo.conexionManejadorBD.getsubProtocol ( )

Método que obtiene el subprotocolo osociado al SGBD

Devuelve

String Subprotocolo

Definición en la línea 68 del archivo conexionManejadorBD.java.

```
68                                  {
69         return this.subProtocol;
70     }
```

B.6.3.9. String Modelo.conexionManejadorBD.getusuario ( )

Método que obtiene el usuario

Devuelve

String usuario

Definición en la línea 140 del archivo conexionManejadorBD.java.

```
140                                {
141         return this.usuario;
142     }
```

B.6.3.10. void Modelo.conexionManejadorBD.setbase_datos (String *base_datos*)

Método que establece el nombre de la base de datos

Parámetros

base_datos	Nombre de la base de datos

Definición en la línea 131 del archivo conexionManejadorBD.java.

```
131                                                {
132         this.base_datos = base_datos;
133     }
```

B.6.3.11. void Modelo.conexionManejadorBD.setdriverJDBC (String *driverJDBC*)

Método que establece el driver JDBC

Parámetros

driverJDBC	Driver JDBC

Definición en la línea 59 del archivo conexionManejadorBD.java.

```
59                                                 {
60         this.driverJDBC = driverJDBC;
61     }
```

B.6.3.12. void Modelo.conexionManejadorBD.sethostName_ip (String *hostName_ip*)

Método que establece el/la nombre/dirección ip del host

Parámetros

hostName_ip	Nombre/dirección ip del host

Definición en la línea 95 del archivo conexionManejadorBD.java.

```
95                                                  {
96         this.hostName_ip = hostName_ip;
97     }
```

B.6.3.13. void Modelo.conexionManejadorBD.setpassword (String *password*)

Método que establece la contraseña asociada al usuario

Parámetros

password	Contraseña

Definición en la línea 167 del archivo conexionManejadorBD.java.

```
167                                             {
168         this.password = password;
169     }
```

B.6.3.14. void Modelo.conexionManejadorBD.setpuerto (String *puerto*)

Método que establece el puerto de servicio del SGBD

Parámetros

puerto	Puerto de servicio del SGBD

Definición en la línea 113 del archivo conexionManejadorBD.java.

```
113                                         {
114         this.puerto = puerto;
115     }
```

B.6.3.15. void Modelo.conexionManejadorBD.setsubProtocol (String *subProtocol*)

Método que establece el subprotocolo asociado al SGBD

Parámetros

subProtocol	Subprotocolo

Definición en la línea 77 del archivo conexionManejadorBD.java.

```
77                                              {
78         this.driverJDBC = subProtocol;
79     }
```

B.6.3.16. void Modelo.conexionManejadorBD.setusuario (String *usuario*)

Método que establece el usuario

Parámetros

usuario	usuario

Definición en la línea 149 del archivo conexionManejadorBD.java.

```
149                                         {
150         this.usuario = usuario;
151     }
```

B.6.4. Documentación de los datos miembro

B.6.4.1. String Modelo.conexionManejadorBD.base_datos [private]

Nombre de la base de datos

Definición en la línea 35 del archivo conexionManejadorBD.java.

B.6.4.2. String Modelo.conexionManejadorBD.driverJDBC [private]

Driver JDBC proporcionado por el SGBD a cargar

Definición en la línea 19 del archivo conexionManejadorBD.java.

B.6.4.3. String Modelo.conexionManejadorBD.hostName_ip [private]

Nombre/dirección ip del Host

Definición en la línea 27 del archivo conexionManejadorBD.java.

B.6.4.4. String Modelo.conexionManejadorBD.password [private]

Contraseña aosciada al usuario

Definición en la línea 43 del archivo conexionManejadorBD.java.

B.6.4.5. String Modelo.conexionManejadorBD.puerto [private]

Puerto de servicio del SGBD

Definición en la línea 31 del archivo conexionManejadorBD.java.

B.6.4.6. String Modelo.conexionManejadorBD.subProtocol [private]

Subporotocolo asociado al SGBD

Definición en la línea 23 del archivo conexionManejadorBD.java.

B.6.4.7. String Modelo.conexionManejadorBD.usuario [private]

Nombre del usuario

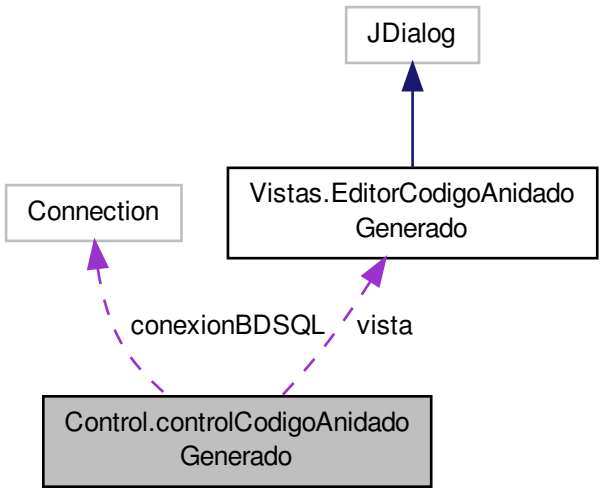
Definición en la línea 39 del archivo conexionManejadorBD.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/conexionManejadorBD.java

B.7. Referencia de la Clase Control.controlCodigoAnidadoGenerado

Diagrama de colaboración para Control.controlCodigoAnidadoGenerado:



Métodos públicos

- controlCodigoAnidadoGenerado (Connection conexionBDSQL, String codigoSQLSP, String codigoSQLSP-Visualizador, EditorCodigoAnidadoGenerado vista)
- void iniciar_vista ()

Métodos privados

- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)

Atributos privados

- EditorCodigoAnidadoGenerado vista
- String codigoSQLSP = ""
- String codigoSQLSPVisualizador = ""
- Connection conexionBDSQL

B.7.1. Descripción detallada

Clase que implementa el control entre el modelo y la vista del código anidado generado de un procedimiento almacenado

Autor

ivan

Definición en la línea 22 del archivo controlCodigoAnidadoGenerado.java.

B.7.2. Documentación del constructor y destructor

B.7.2.1. Control.controlCodigoAnidadoGenerado.controlCodigoAnidadoGenerado (Connection *conexionBDSQL,* String *codigoSQLSP,* String *codigoSQLSPVisualizador,* EditorCodigoAnidadoGenerado *vista*)

Constructor por defecto de la clase, establece la conexión con la base de datos, el código anidado generado y la vista del editor del código

Parámetros

conexionBDSQL	Conexión establecida con alguna base de datos
codigoSQLSP	Código SQL anidado
codigoSQLSP-Visualizador	Código SQL anidado y formateado para el visualizador
vista	Vista: Editor del código anidado generado

Definición en la línea 53 del archivo controlCodigoAnidadoGenerado.java.

```
53
                                                {
54          this.conexionBDSQL = conexionBDSQL;
55          this.codigoSQLSP = codigoSQLSP;
56          this.codigoSQLSPVisualizador = codigoSQLSPVisualizador.replaceAll("\n", "<br>");
57          this.vista = vista;
58      }
```

B.7.3. Documentación de las funciones miembro

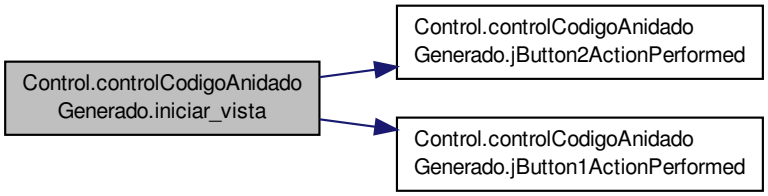
B.7.3.1. void Control.controlCodigoAnidadoGenerado.iniciar_vista ( )

Método que implementa el valor por defecto de la Vista

Definición en la línea 63 del archivo controlCodigoAnidadoGenerado.java.

```
63                                  {
64          this.vista.visualizadorSQL.setText(codigoSQLSPVisualizador);
65          this.vista.setTitle("Administración del código generado con anidamientos");
66          this.vista.setLocationRelativeTo(null);
67          this.vista.jButton2.addActionListener(new java.awt.event.ActionListener() {
68              public void actionPerformed(java.awt.event.ActionEvent evt) {
69                  jButton2ActionPerformed(evt);
70              }
71          });
72
73          this.vista.jButton1.addActionListener(new java.awt.event.ActionListener() {
74              public void actionPerformed(java.awt.event.ActionEvent evt) {
75                  jButton1ActionPerformed(evt);
76              }
77          });
78
79          this.vista.setVisible(true);
80      }
```

Gráfico de llamadas para esta función:



B.7.3.2. void Control.controlCodigoAnidadoGenerado.jButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método que implementa la acción que cierra la vista

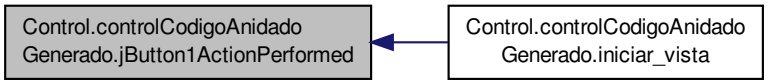
Parámetros

evt	Evento

Definición en la línea 119 del archivo controlCodigoAnidadoGenerado.java.

```
119                                                                     {
120         // TODO add your handling code here:
121         this.vista.dispose();
122     }
```

Gráfico de llamadas a esta función:



B.7.3.3. void Control.controlCodigoAnidadoGenerado.jButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método que implementa la acción de guarda el código SQL generado

Parámetros

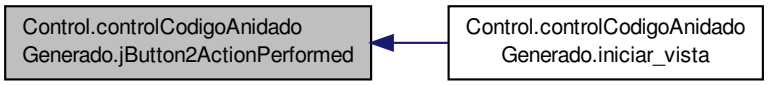
evt	Evento

Definición en la línea 87 del archivo controlCodigoAnidadoGenerado.java.

```
87                                                                      {
88         // TODO add your handling code here:
89         JFileChooser jF1 = new JFileChooser();
```

```
90          String ruta = "";
91          try {
92              if (jF1.showSaveDialog(null) == jF1.getApproveButtonMnemonic()) {
93                  ruta = jF1.getSelectedFile().getAbsolutePath();
94                  if (new File(ruta).exists()) {
95                      if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, "El scrip existe\n
        desea reemplazarlo", "Guardar Scrip", JOptionPane.YES_NO_OPTION)) {
96                          PrintWriter out = new PrintWriter(new FileWriter(new File(ruta), false));
97                          out.print(this.codigoSQLSP);
98                          out.flush();
99                          out.close();
100                     }
101
102                 } else {
103                     PrintWriter out = new PrintWriter(new FileWriter(new File(ruta)));
104                     out.print(this.codigoSQLSP);
105                     out.flush();
106                     out.close();
107                 }
108             }
109         } catch (Exception e) {
110             e.printStackTrace();
111         }
112     }
```

Gráfico de llamadas a esta función:



B.7.4. Documentación de los datos miembro

B.7.4.1. String Control.controlCodigoAnidadoGenerado.codigoSQLSP = "" [private]

Código SQL anidado generado para un procedimiento almacenado e invocaciones

Definición en la línea 32 del archivo controlCodigoAnidadoGenerado.java.

B.7.4.2. String Control.controlCodigoAnidadoGenerado.codigoSQLSPVisualizador = "" [private]

Código SQL anidado generado para un procedimiento almacenado e invocaciones(formateado para el visualizador)

Definición en la línea 37 del archivo controlCodigoAnidadoGenerado.java.

B.7.4.3. Connection Control.controlCodigoAnidadoGenerado.conexionBDSQL [private]

Conexión establecida con alguna base de datos

Definición en la línea 41 del archivo controlCodigoAnidadoGenerado.java.

B.7.4.4. EditorCodigoAnidadoGenerado Control.controlCodigoAnidadoGenerado.vista [private]

Vista: Editor del código anidado generado

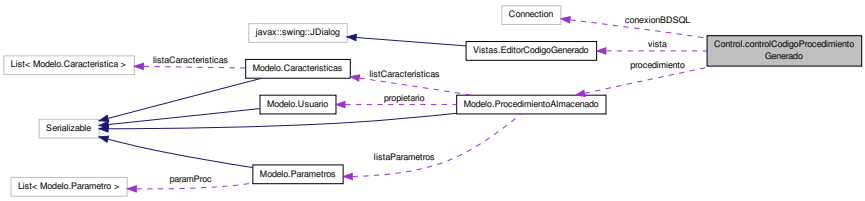
Definición en la línea 27 del archivo controlCodigoAnidadoGenerado.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/controlCodigoAnidadoGenerado.java

B.8. Referencia de la Clase Control.controlCodigoProcedimientoGenerado

Diagrama de colaboración para Control.controlCodigoProcedimientoGenerado:



Métodos públicos

- controlCodigoProcedimientoGenerado (Connection conexionBDSQL, ProcedimientoAlmacenado procedimiento, EditorCodigoGenerado vista)
- void iniciar_vista ()

Métodos privados

- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)

Atributos privados

- EditorCodigoGenerado vista
- ProcedimientoAlmacenado procedimiento
- String codigoSQLSP = ""
- Connection conexionBDSQL

B.8.1. Descripción detallada

Clase que implementa el control entre el modelo y la vista del código generado de un procedimiento almacenado

Autor

ivan

Definición en la línea 25 del archivo controlCodigoProcedimientoGenerado.java.

B.8.2. Documentación del constructor y destructor

B.8.2.1. Control.controlCodigoProcedimientoGenerado.controlCodigoProcedimientoGenerado (Connection *conexionBDSQL,* ProcedimientoAlmacenado *procedimiento,* EditorCodigoGenerado *vista*)

Constructor por defecto de la clase, establece la variable de conexión con alguna base de datos, el modelo y la vista.

Parámetros

conexionBDSQL	Conexión establecida con alguna base de datos
procedimiento	Modelo: Objeto de la clase que implementa un procedimiento almacenado
vista	Vista: Editor del código generado de un procedimiento almacenado

Definición en la línea 54 del archivo controlCodigoProcedimientoGenerado.java.

```
54
                                    {
55          this.conexionBDSQL = conexionBDSQL;
56          this.procedimiento = procedimiento;
57          this.vista = vista;
58      }
```

B.8.3. Documentación de las funciones miembro

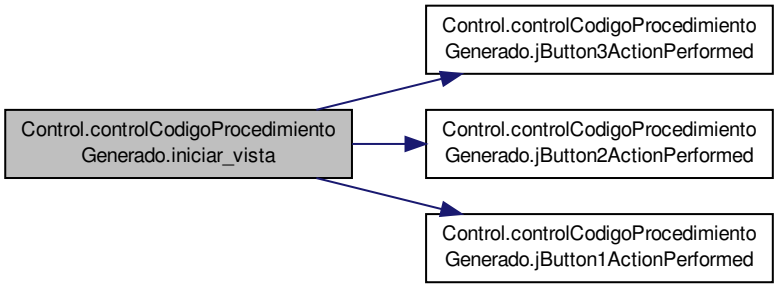
B.8.3.1. void Control.controlCodigoProcedimientoGenerado.iniciar_vista ( )

Método que implementa el valor por defecto de la vista

Definición en la línea 63 del archivo controlCodigoProcedimientoGenerado.java.

```
63                                  {
64          this.codigoSQLSP = this.procedimiento.generarProcedimientoSQL(true).replaceAll("\n", "<br>");
65          this.vista.Visualizador.setText(this.codigoSQLSP);
66          this.vista.setTitle("Administración del procedimiento generado");
67          this.vista.setLocationRelativeTo(null);
68          this.vista.jButton3.addActionListener(new java.awt.event.ActionListener() {
69              public void actionPerformed(java.awt.event.ActionEvent evt) {
70                  jButton3ActionPerformed(evt);
71              }
72          });
73          this.vista.jButton2.addActionListener(new java.awt.event.ActionListener() {
74              public void actionPerformed(java.awt.event.ActionEvent evt) {
75                  jButton2ActionPerformed(evt);
76              }
77          });
78          this.vista.jButton1.addActionListener(new java.awt.event.ActionListener() {
79              public void actionPerformed(java.awt.event.ActionEvent evt) {
80                  jButton1ActionPerformed(evt);
81              }
82          });
83          this.vista.setVisible(true);
84      }
```

Gráfico de llamadas para esta función:



B.8.3.2. void Control.controlCodigoProcedimientoGenerado.jButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método que implementa la acción de enviar el código SQL generado a la base de datos

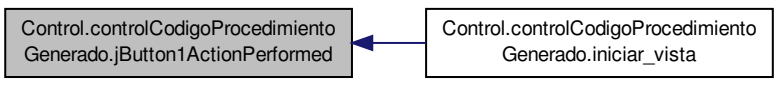
Parámetros

evt	Evento

Definición en la línea 135 del archivo controlCodigoProcedimientoGenerado.java.

```
135                                                                     {
136         // TODO add your handling code here:
137         if (this.conexionBDSQL != null) {
138             Statement stmt = null;
139             try {
140                 stmt = this.conexionBDSQL.createStatement();
141                 if (this.vista.jCheckBox1.isSelected()) {
142                     stmt.execute("DROP PROCEDURE IF EXISTS " + this.procedimiento.getNombre() + ";");
143                 }
144                 stmt.execute(this.procedimiento.generarProcedimientoSQL(false));
145             } catch (SQLException SQLe) {
146                 JOptionPane.showMessageDialog(null, SQLe.getMessage());
147             }
148         } else {
149             JOptionPane.showMessageDialog(null, "Para realizar esta funcion\n debe establecer una conexion
      con alguna base de datos");
150         }
151     }
```

Gráfico de llamadas a esta función:



B.8.3.3. void Control.controlCodigoProcedimientoGenerado.jButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método que implementa la acción que guarda el código SQL generado

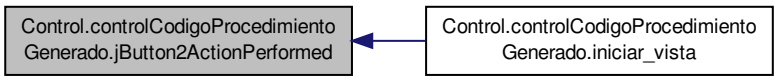
Parámetros

evt	Evento

Definición en la línea 101 del archivo controlCodigoProcedimientoGenerado.java.

```
101                                                                         {
102         // TODO add your handling code here:
103         JFileChooser jF1 = new JFileChooser();
104         String ruta = "";
105         try {
106             if (jF1.showSaveDialog(null) == jF1.getApproveButtonMnemonic()) {
107                 ruta = jF1.getSelectedFile().getAbsolutePath();
108                 if (new File(ruta).exists()) {
109                     if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, "El scrip existe\n
     desea reemplazarlo", "Guardar Scrip", JOptionPane.YES_NO_OPTION)) {
110                         PrintWriter out = new PrintWriter(new FileWriter(new File(ruta), false));
111                         out.print(((this.vista.jCheckBox1.isSelected()) ? "DROP PROCEDURE IF EXIST " +
    this.procedimiento.getNombre() + ";\n" : "")
112                                 + this.procedimiento.generarProcedimientoSQL(false));
113                         out.flush();
114                         out.close();
115                     }
116                 } else {
117                     PrintWriter out = new PrintWriter(new FileWriter(new File(ruta)));
118                     out.print(((this.vista.jCheckBox1.isSelected()) ? "DROP PROCEDURE IF EXIST " +
    this.procedimiento.getNombre() + ";\n" : "")
119                             + this.procedimiento.generarProcedimientoSQL(false));
120                     out.flush();
121                     out.close();
122                 }
123             }
124         } catch (Exception e) {
125             e.printStackTrace();
126         }
127     }
```

Gráfico de llamadas a esta función:



B.8.3.4. void Control.controlCodigoProcedimientoGenerado.jButton3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método que implementa la acción que cierra la vista

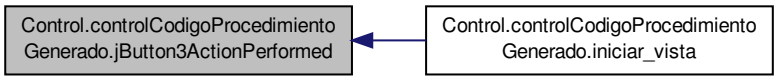
Parámetros

evt	Evento

Definición en la línea 91 del archivo controlCodigoProcedimientoGenerado.java.

```
91                                                                          {
92         // TODO add your handling code here:
93         this.vista.dispose();
94     }
```

Gráfico de llamadas a esta función:



B.8.4. Documentación de los datos miembro

B.8.4.1. String Control.controlCodigoProcedimientoGenerado.codigoSQLSP = "" [private]

Código anidado generado para un procedimiento almacenado

Definición en la línea 38 del archivo controlCodigoProcedimientoGenerado.java.

B.8.4.2. Connection Control.controlCodigoProcedimientoGenerado.conexionBDSQL [private]

Conexión establecida con alguna base de datos

Definición en la línea 42 del archivo controlCodigoProcedimientoGenerado.java.

B.8.4.3. ProcedimientoAlmacenado Control.controlCodigoProcedimientoGenerado.procedimiento [private]

Modelo: Objeto de la clase que implementa un procedimiento almacenado

Definición en la línea 34 del archivo controlCodigoProcedimientoGenerado.java.

B.8.4.4. EditorCodigoGenerado Control.controlCodigoProcedimientoGenerado.vista [private]

Vista: Editor del código generado de un procedimiento almacenado

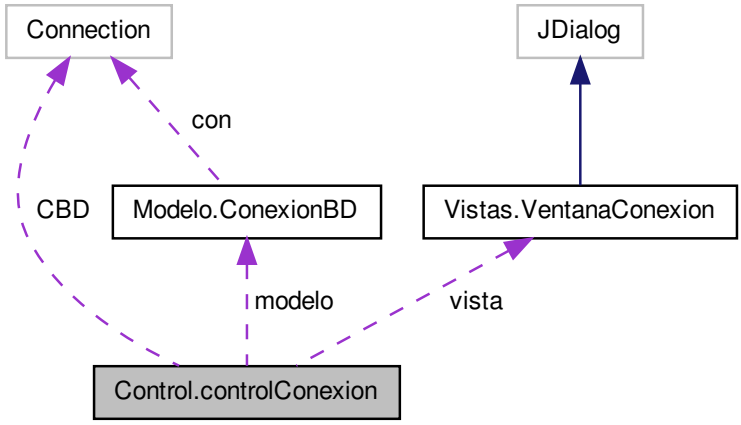
Definición en la línea 30 del archivo controlCodigoProcedimientoGenerado.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/controlCodigoProcedimientoGenerado.java

B.9. Referencia de la Clase Control.controlConexion

Diagrama de colaboración para Control.controlConexion:



Métodos públicos

- controlConexion (conexionManejadorBD modelo, VentanaConexion vista)
- void iniciar_vista ()
- Connection getConexion ()
- void FocusLost (java.awt.event.FocusEvent e)

Atributos públicos

- VentanaConexion vista

Métodos privados

- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
- void jTextField2KeyTyped (java.awt.event.KeyEvent evt)
- boolean validarNombreBD (String S)
- String valorNombreBDValido (String S)
- void generarURL ()
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)

Atributos privados

- ConexionBD modelo
- Connection CBD

B.9.1. Descripción detallada

Clase que implementa el control entre el modelo y la vista de una conexión a la base de datos

Autor

ivan

Definición en la línea 19 del archivo controlConexion.java.

B.9.2. Documentación del constructor y destructor

B.9.2.1. Control.controlConexion.controlConexion (conexionManejadorBD *modelo,* VentanaConexion *vista*)

Constructor por defecto de la clase controlConexion, establece el modelo y la vista

Parámetros

modelo	Modelo: Objeto de la clase que implementa una conexión a una base de datos
vista	Vista: Establecer conexión con una base de datos

Definición en la línea 43 del archivo controlConexion.java.

```
43                                                                                  {
44         this.vista = vista;
45         this.modelo = modelo;
46     }
```

B.9.3. Documentación de las funciones miembro

B.9.3.1. void Control.controlConexion.FocusLost (java.awt.event.FocusEvent *e*)

verificación de que el campo no este vacío

Definición en la línea 120 del archivo controlConexion.java.

```
120                                                 {
121         if (e.getSource().equals(vista.jTextField2)) {
122             if (vista.jTextField2.getText().equals("")) {
123                 JOptionPane.showMessageDialog(null, "Este campo \"Base de datos\"no puede estar vacío\n
      recuerde que es obligatorio");
124                 vista.jTextField2.requestFocus();
125             } else if (!this.validarNombreBD(vista.
     jTextField2.getText())) {
126                 JOptionPane.showMessageDialog(null, "El nombre de la base de datos no es valido");
127                 vista.jTextField2.setText(this.valorNombreBDValido(vista.jTextField2.getText()));
128                 vista.jTextField2.requestFocus();
129 
130             }
131             this.generarURL();
132         } else if (e.getSource().equals(vista.jTextField5)) {
133             if (vista.jTextField5.getText().equals("")) {
134                 JOptionPane.showMessageDialog(null, "El campo \"usuario\" no puede estar vacío\n recuerde
      que es obligatorio");
135                 vista.jTextField5.requestFocus();
136             }
137             this.generarURL();
138         } else if (e.getSource().equals(vista.jPasswordField1)) {
139 
140             if (vista.jPasswordField1.getPassword().length == 0) {
141                 JOptionPane.showMessageDialog(null, "el campo \"password\" no puede estar vacío\n recuerde
      que es obligatorio");
142                 vista.jPasswordField1.requestFocus();
143             }
```

```
144             this.generarURL();
145         } else if (e.getSource().equals(vista.jTextField4)) {
146             if (!"".equals(vista.jTextField4.getText())) {
147                 String port = vista.jTextField4.getText();
148                 int puerto = Integer.parseInt(port);
149                 if (!(puerto >= 0 && puerto <= 65535)) {
150                     JOptionPane.showMessageDialog(null, "puerto: " + puerto + " Invalido\n" + "
      http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml");
151                     vista.jTextField4.setText("");
152                     vista.jTextField4.requestFocus();
153                 }
154             }
155             this.generarURL();
156         }
157     }
```

Gráfico de llamadas para esta función:

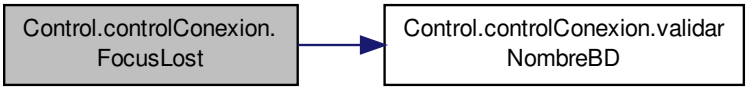
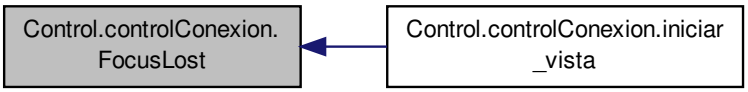


Gráfico de llamadas a esta función:



B.9.3.2. void Control.controlConexion.generarURL ( ) [private]

Método que genera la URL para establecer una conexión a la base de datos, a partir de los parámetros proporcionado por el usuario en la interfaz gráfica.

Definición en la línea 223 del archivo controlConexion.java.

```
223                             {
224         String URL = "jdbc:" + "mysql" + ":" + "//"
225                 + vista.jTextField3.getText() + ":"
226                 + vista.jTextField4.getText() + "/"
227                 + vista.jTextField2.getText();
228         vista.jTextField1.setText(URL);
229     }
```

B.9.3.3. Connection Control.controlConexion.getConexion ( )

Definición en la línea 113 del archivo controlConexion.java.

```
113                                     {
114         return CBD;
115     }
```

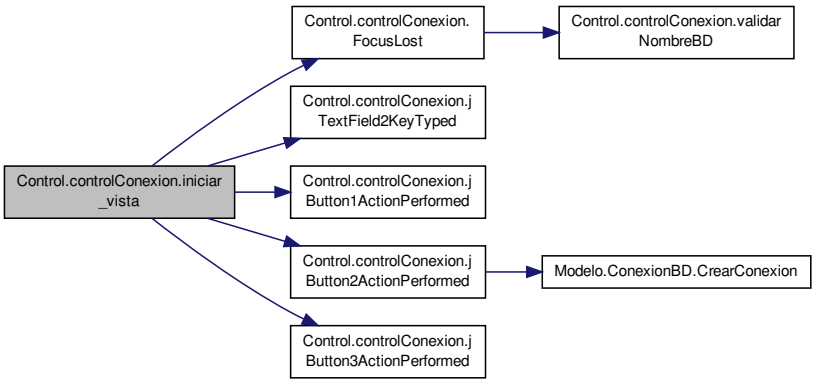
B.9.3.4. void Control.controlConexion.iniciar_vista ( )

Método que implementa el valor por defecto de la vista

Definición en la línea 51 del archivo controlConexion.java.

```
51                                  {
52         vista.setTitle("Conexión a una base de datos");
53         vista.setLocationRelativeTo(null);
54         vista.jTextField1.setText(String.valueOf(modelo.getdriveJDBC()));
55         vista.jTextField2.addFocusListener(new java.awt.event.FocusAdapter() {
56             @Override
57             public void focusLost(java.awt.event.FocusEvent evt) {
58                 FocusLost(evt);
59             }
60         });
61         vista.jTextField2.addKeyListener(new java.awt.event.KeyAdapter() {
62             @Override
63             public void keyTyped(java.awt.event.KeyEvent evt) {
64                 jTextField2KeyTyped(evt);
65             }
66         });
67         vista.jTextField3.addFocusListener(
68                 new java.awt.event.FocusAdapter() {
69                     @Override
70                     public void focusLost(java.awt.event.FocusEvent evt) {
71                         FocusLost(evt);
72                     }
73                 });
74         vista.jTextField4.addFocusListener(new java.awt.event.FocusAdapter() {
75             @Override
76             public void focusLost(java.awt.event.FocusEvent evt) {
77                 FocusLost(evt);
78             }
79         });
80         vista.jTextField5.addFocusListener(new java.awt.event.FocusAdapter() {
81             @Override
82             public void focusLost(java.awt.event.FocusEvent evt) {
83                 FocusLost(evt);
84             }
85         });
86         vista.jPasswordField1.addFocusListener(new java.awt.event.FocusAdapter() {
87             @Override
88             public void focusLost(java.awt.event.FocusEvent evt) {
89                 FocusLost(evt);
90             }
91         });
92         vista.jButton1.addActionListener(new java.awt.event.ActionListener() {
93             @Override
94             public void actionPerformed(java.awt.event.ActionEvent evt) {
95                 jButton1ActionPerformed(evt);
96             }
97         });
98         vista.jButton2.addActionListener(new java.awt.event.ActionListener() {
99             @Override
100             public void actionPerformed(java.awt.event.ActionEvent evt) {
101                 jButton2ActionPerformed(evt);
102             }
103         });
104         this.vista.jButton3.addActionListener(new java.awt.event.ActionListener() {
105             @Override
106             public void actionPerformed(java.awt.event.ActionEvent evt) {
107                 jButton3ActionPerformed(evt);
108             }
109         });
110         this.vista.setVisible(true);
111     }
```

Gráfico de llamadas para esta función:



B.9.3.5. void Control.controlConexion.jButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita cerrarla

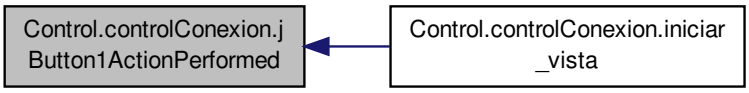
Parámetros

evt	Evento

Definición en la línea 165 del archivo controlConexion.java.

```
165                                                                 {
166         vista.dispose();
167     }
```

Gráfico de llamadas a esta función:



B.9.3.6. void Control.controlConexion.jButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer conexión con una base de datos

Parámetros

evt	Evento

Definición en la línea 237 del archivo controlConexion.java.

```
237                                                                     {
238         String tmptxt = this.vista.jButton2.getText();
239         String mensaje = null;
240         ConexionBD conMySQL = new ConexionBD();
241         conMySQL.CargarDriver("com.mysql.jdbc.Driver");
242         if (vista.jPasswordField1.getPassword().length > 0
243                 && vista.jTextField2.getText().compareTo("") != 0
244                 && vista.jTextField5.getText().compareTo("") != 0) {
245             if (tmptxt.compareTo("Terminar conexión") == 0) {
246                 vista.jPasswordField1.setEnabled(true);
247                 vista.jPasswordField1.setText("1234");
248                 vista.jTextField1.setEnabled(true);
249                 vista.jTextField2.setEnabled(true);
250                 vista.jTextField2.setText("nombre_BD");
251                 vista.jTextField3.setEnabled(true);
252                 vista.jTextField4.setEnabled(true);
253                 vista.jTextField5.setEnabled(true);
254                 vista.jTextField5.setText("Usuario");
255                 tmptxt = "Probar conexión";
256                 vista.jTextField2.requestFocus();
257                 this.CBD = null;
258                 mensaje = "Conexión Terminada";
259             } else if (tmptxt.compareTo("Probar conexión") == 0) {
260                 if (!conMySQL.CrearConexion(vista.jTextField1.getText(),
    vista.jTextField5.getText(), String.valueOf(vista.jPasswordField1.getPassword()))) {
261                     tmptxt = "Probar conexión";
262                     mensaje = "Error, en tratar de establecer conexion a la base de datos.\n"
263                             + "Verifique los campos obligatorios";
264                     vista.jTextField2.requestFocus();
265                 } else {
266                     CBD = conMySQL.getConexion();
267                     vista.jPasswordField1.setEnabled(false);
268                     vista.jTextField1.setEnabled(false);
269                     vista.jTextField2.setEnabled(false);
270                     vista.jTextField3.setEnabled(false);
271                     vista.jTextField4.setEnabled(false);
272                     vista.jTextField5.setEnabled(false);
273                     tmptxt = "Terminar conexión";
274                     mensaje = "Conexión satisfactoria";
275                 }
276 
277             }
278             this.vista.jButton2.setText(tmptxt);
279             JOptionPane.showMessageDialog(null, mensaje);
280         }
281     }
```

Gráfico de llamadas para esta función:

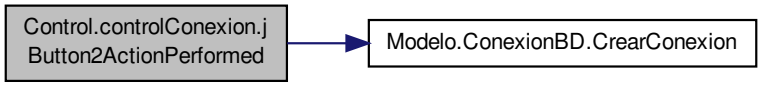
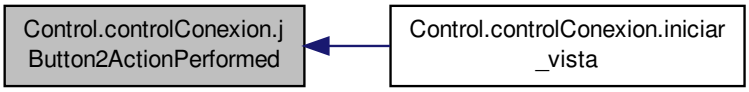


Gráfico de llamadas a esta función:



B.9.3.7. void Control.controlConexion.jButton3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita cerrarla.

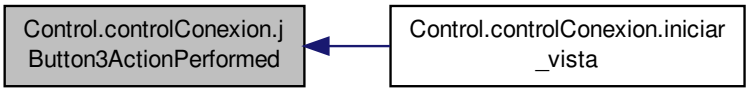
Parámetros

evt	Evento

Definición en la línea 289 del archivo controlConexion.java.

```
289                                                                 {
290         // TODO add your handling code here:
291         this.vista.dispose();
292     }
```

Gráfico de llamadas a esta función:



B.9.3.8. void Control.controlConexion.jTextField2KeyTyped (java.awt.event.KeyEvent *evt*) [private]

Método que verifica que el nombre de la base de datos no sobrepasé su longitud máxima

Parámetros

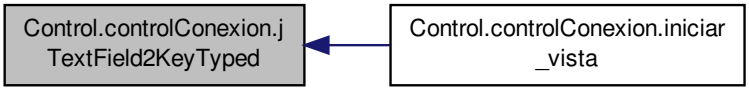
evt	Evento

Definición en la línea 175 del archivo controlConexion.java.

```
175                                                                 {
176         char caracter = evt.getKeyChar();
177         if (caracter == 32) {
178             evt.consume();
179         }
180         if (vista.jTextField2.getText().length() >= 64) {
```

```
181             evt.consume();
182         }
183     }
```

Gráfico de llamadas a esta función:



B.9.3.9. boolean Control.controlConexion.validarNombreBD (String *S*) [private]

Método que valida la longitud del nombre de la base de datos proporcionada

Parámetros

S	Nombre de la base de datos

Devuelve

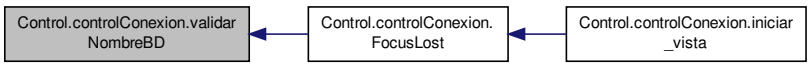
boolean

- true El nombre de la base de datos es valido
- false El nombre de la base de datos no es valido

Definición en la línea 196 del archivo controlConexion.java.

```
196                                         {
197         return S.indexOf(' ') < 0 && S.length() <= 64;
198     }
```

Gráfico de llamadas a esta función:



B.9.3.10. String Control.controlConexion.valorNombreBDValido (String *S*) [private]

Método que retorna el nombre de la base de datos valido, es decir, si sobrepasa la longitud aceptable este método lo recorta.

Parámetros

S	Nombre de la base de datos

Devuelve

String Nombre de la base de datos valido

Definición en la línea 207 del archivo controlConexion.java.

```
207                                                 {
208         String patron = "(\\.*)(\\s+)(\\.*)";
209         S = S.replaceAll(patron, "$1$3");
210         if (S.length() > 64) {
211             return S.substring(0, 64);
212         } else {
213             return S;
214         }
215 
216     }
```

B.9.4. Documentación de los datos miembro

B.9.4.1. Connection Control.controlConexion.CBD [private]

Conexión establecida con alguna base de datos

Definición en la línea 33 del archivo controlConexion.java.

B.9.4.2. ConexionBD Control.controlConexion.modelo [private]

Modelo: Objeto de la clase que implementa una conexión a una base de datos

Definición en la línea 29 del archivo controlConexion.java.

B.9.4.3. VentanaConexion Control.controlConexion.vista

Vista: Establecer conexión con una base de datos

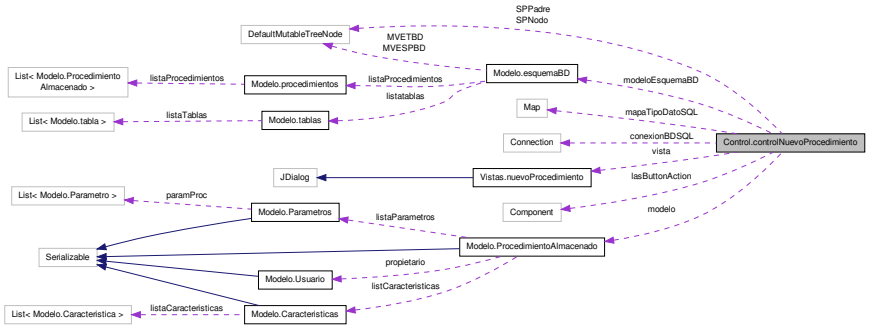
Definición en la línea 24 del archivo controlConexion.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/controlConexion.java

B.10. Referencia de la Clase Control.controlNuevoProcedimiento

Diagrama de colaboración para Control.controlNuevoProcedimiento:



Métodos públicos

- void setConexionBDSQL (Connection C)
- Connection getConexionBDSQL ()
- void setLastbuttonAction (Component lasButtonAction)
- Component getlasButtonAction ()
- controlNuevoProcedimiento (Connection C, ProcedimientoAlmacenado modelo, nuevoProcedimiento vista, esquemaBD modeloEsquemaBD, String nombreProcedimiento, DefaultMutableTreeNode Nodo)
- void setFocusPorDefecto ()
- void MVNombreProcedimiento ()
- void pordefecto ()
- void iniciar_Vista_NuevoProcedimiento ()
- void guardarSPenModeloyVisaualizarlo ()
- void limpiarCampos ()
- void limpiarCamposCaracteristicas ()

Atributos públicos

- Map mapaTipoDatoSQL

Métodos privados

- void makeTypesName ()
- void inicializarTipoDatoSQLModeloVista ()
- void agregarParamModeloVista ()
- void limpiarCamposParametros ()
- void agregarParametroVista ()
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void editarParametroVistaModelo ()
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
- void borrarParametroVistaModelo ()
- void jButton13ActionPerformed (java.awt.event.ActionEvent evt)

- void guardarCambiosParametroVistaModelo ()
- void jButton14ActionPerformed (java.awt.event.ActionEvent evt)
- void jCheckBox3ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox2ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox6ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton12ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton5ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton15ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton4ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)

Atributos privados

- nuevoProcedimiento vista
- ProcedimientoAlmacenado modelo
- Connection conexionBDSQL
- esquemaBD modeloEsquemaBD
- DefaultMutableTreeNode SPNodo
- DefaultMutableTreeNode SPPadre
- Component lasButtonAction

B.10.1. Descripción detallada

Clase que implementa el control entre el modelo y la vista de un nuevo procedimiento almacenado

Autor

ivan

Definición en la línea 34 del archivo controlNuevoProcedimiento.java.

B.10.2. Documentación del constructor y destructor

B.10.2.1. Control.controlNuevoProcedimiento.controlNuevoProcedimiento (Connection *C,* ProcedimientoAlmacenado *modelo,* nuevoProcedimiento *vista,* esquemaBD *modeloEsquemaBD,* String *nombreProcedimiento,* DefaultMutableTreeNode *Nodo*)

Constructor por defecto de la clase, establece la variable que contiene una conexión establecida con alguna base de datos, el modelo, la vista, el esquema de una base de datos, el nombre del procedimiento, el padre del nodo asociado al procedimiento actual

Parámetros

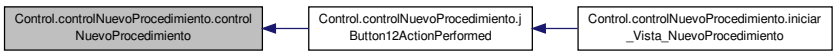
C	Conexión establecida con alguna base de datos
modelo	Modelo: Objeto de la clase que implementa un procedimiento almacenado
vista	Vista: Editor de un nuevo procedimiento almacenado

modeloEsquema-BD	Modelo: Objeto de la clase que implementa un esquema de una base de datos
nombre-Procedimiento	Nombre del procedimiento almacenado
Nodo	Nodo asociado al procedimiento actual

Definición en la línea 122 del archivo controlNuevoProcedimiento.java.

```
122                                                                                                    {
123         this.setConexionBDSQL(C);
124         this.modelo = modelo;
125         this.vista = vista;
126         this.modeloEsquemaBD = modeloEsquemaBD;
127         this.modelo.setNombre(nombreProcedimiento);
128         this.SPPadre = Nodo;
129     }
```

Gráfico de llamadas a esta función:



B.10.3. Documentación de las funciones miembro

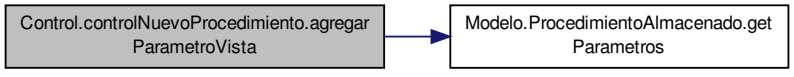
B.10.3.1. void Control.controlNuevoProcedimiento.agregarParametroVista ( ) [private]

Método que implementa la función agregar parámetro de la vista al modelo

Definición en la línea 316 del archivo controlNuevoProcedimiento.java.

```
316                                         {
317         if (this.vista.jTextField2.getText().isEmpty()) {
318             JOptionPane.showMessageDialog(null, "nombre/parametro vacio\n");
319             this.vista.jComboBox4.requestFocus();
320         } else {
321             if (this.modelo.getParametros() == null) {
322                 this.modelo.setParametros(new Parametros());
323             }
324             this.modelo.getParametros().agregarParametro(new Parametro(
325                     (String) this.vista.jComboBox4.getSelectedItem().toString(),
326                     this.vista.jTextField2.getText(),
327                     this.vista.tipoDatoSQL.getSelectedItem().toString()
328             ));
329             this.agregarParamModeloVista();
330             this.limpiarCamposParametros();
331         }
332     }
```

Gráfico de llamadas para esta función:



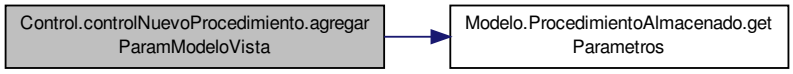
B.10.3.2. void Control.controlNuevoProcedimiento.agregarParamModeloVista ( ) [private]

Método que implementa la función agregar parámetro del modelo a la vista

Definición en la línea 281 del archivo controlNuevoProcedimiento.java.

```
281                                         {
282         if (this.modelo.getParametros() != null) {
283             Parametro item;
284             ArrayList<String> P = new ArrayList<>();
285             if (!this.modelo.getParametros().listaParamVacia()) {
286                 for (int i = 0; i < this.modelo.getParametros().numElementos() - 1; i++) {
287                     item = (Parametro) this.modelo.getParametros().obtenerParametro(i);
288                     P.add(item.generarSQL());
289                 }
290                 item = (Parametro) this.modelo.getParametros().getParamProc().get(
291                         this.modelo.getParametros().numElementos() - 1);
292                 P.add(item.generarSQL());
293             }
294             DefaultListModel<String> model = new DefaultListModel<>();
295             for (String s : P) {
296                 model.addElement(s);
297             }
298             this.vista.jList1.setModel(model);
299         }
300     }
```

Gráfico de llamadas para esta función:



B.10.3.3. void Control.controlNuevoProcedimiento.borrarParametroVistaModelo ( ) [private]

Método que implementa la función borrar parámetro, garantiza que los cambios generados al ejecutar la función se vean reflejados en la vista y en el modelo

Definición en la línea 413 del archivo controlNuevoProcedimiento.java.

```
413                                         {
414         if (this.vista.jList1.getModel().getSize() > 0) {
415             if (this.vista.jList1.isSelectionEmpty()) {
416                 JOptionPane.showMessageDialog(null, "Seleleccione algun parametro");
417                 this.vista.jList1.setSelectedIndex(0);
418                 if (this.vista.jList1.getModel().getSize() > 1) {
419                     this.vista.jList1.requestFocus();
420 
421                 } else {
422                     this.vista.jButton13.requestFocus();
423                 }
424             } else {
425                 this.modelo.getParametros().getParamProc().remove(this.vista.jList1.getSelectedIndex());
426                 this.setLastbuttonAction(null);
427                 this.agregarParamModeloVista();
428                 this.vista.jComboBox4.requestFocus();
429             }
430         } else {
431             JOptionPane.showMessageDialog(null, "Agrege almenos un Parametro");
432             this.vista.jComboBox4.requestFocus();
433         }
434     }
```

B.10.3.4. void Control.controlNuevoProcedimiento.editarParametroVistaModelo ( ) [private]

Método que implementa la función editar parámetro, garantiza que los cambios en la vista se vean reflejados en el modelo

Definición en la línea 348 del archivo controlNuevoProcedimiento.java.

```
348                                             {
349         if (this.vista.jList1.getModel().getSize() > 0) {
350             if (this.vista.jList1.isSelectionEmpty()) {
351                 JOptionPane.showMessageDialog(null, "Seleccione algun parametro");
352                 this.vista.jList1.setSelectedIndex(0);
353                 if (this.vista.jList1.getModel().getSize() > 1) {
354                     this.vista.jList1.requestFocus();
355                 } else {
356                     this.vista.jButton1.requestFocus();
357                 }
358             } else {
364                 this.vista.jComboBox4.setSelectedItem(
365                         this.modelo.getParametros().obtenerParametro(this.vista.
    jList1.getSelectedIndex()).getTipoES()
366                 );
367                 this.vista.jTextField2.setText(
368                         this.modelo.getParametros().obtenerParametro(this.vista.
    jList1.getSelectedIndex()).getNombre()
369                 );
370                 this.vista.tipoDatoSQL.setSelectedItem(
371                         this.modelo.getParametros().obtenerParametro(this.vista.
    jList1.getSelectedIndex()).getTipoDato()
372                 );
373                 this.vista.jButton1.setEnabled(false);
374                 this.vista.jButton2.setEnabled(false);
375                 this.vista.jButton13.setEnabled(false);
376                 this.vista.jList1.setEnabled(false);
377                 this.vista.jButton14.setEnabled(true);
381                 ArrayList tmpOrden = new ArrayList();
382                 tmpOrden.add(this.vista.jComboBox4);
383                 tmpOrden.add(this.vista.jTextField2);
384                 tmpOrden.add(this.vista.tipoDatoSQL);
385                 tmpOrden.add(this.vista.jButton14);
386                 MiFocusTraversalPolicy newTMPPolicy = new MiFocusTraversalPolicy(tmpOrden);
387                 vista.setFocusTraversalPolicy(newTMPPolicy);
388                 this.vista.jComboBox4.requestFocus();
389                 this.modelo.getParametros().modificarParametro(this.vista.
    jList1.getSelectedIndex());
390             }
391         } else {
392             JOptionPane.showMessageDialog(null, "Agrege almenos un Parametro");
393             this.vista.jComboBox4.requestFocus();
394         }
395     }
```

B.10.3.5. Connection Control.controlNuevoProcedimiento.getConexionBDSQL ( )

Método que obtiene la variable que contiene la conexión establecida con alguna base de datos

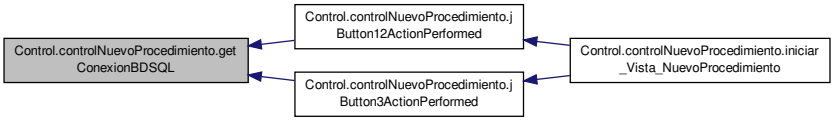
Devuelve

Connection conexión establecida con alguna base de datos

Definición en la línea 85 del archivo controlNuevoProcedimiento.java.

```
85                                          {
86         return this.conexionBDSQL;
87     }
```

Gráfico de llamadas a esta función:



B.10.3.6. Component Control.controlNuevoProcedimiento.getlasButtonAction ( )

Método que obtiene el ultimo componente que provocó algún evento

Devuelve

Component Componente

Definición en la línea 103 del archivo controlNuevoProcedimiento.java.

```
103                                     {
104         return this.lasButtonAction;
105     }
```

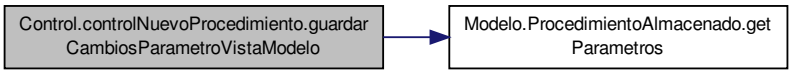
B.10.3.7. void Control.controlNuevoProcedimiento.guardarCambiosParametroVistaModelo ( ) [private]

Método que implementa la función guardar los cambios en el parámetro, garantiza que los cambios generados al ejecutar la función se vean reflejados en la vista y en el modelo.

Definición en la línea 452 del archivo controlNuevoProcedimiento.java.

```
452                                                 {
453         if (this.modelo.getParametros().getModParam() >= 0) {
454             this.modelo.getParametros().obtenerParametro(
455                     this.modelo.getParametros().getModParam()
456             ).setTipoES((String) this.vista.jComboBox4.getSelectedItem().toString());
457             this.modelo.getParametros().obtenerParametro(
458                     this.modelo.getParametros().getModParam()
459             ).setNombre(this.vista.jTextField2.getText());
460             this.modelo.getParametros().obtenerParametro(
461                     this.modelo.getParametros().getModParam()
462             ).setTipoDato(this.vista.tipoDatoSQL.getSelectedItem().toString());
463             this.agregarParamModeloVista();
464             this.limpiarCamposParametros();
465             this.setFocusPorDefecto();
466             this.setLastbuttonAction(null);
467             this.vista.jButton14.setEnabled(false);
468             this.vista.jButton1.setEnabled(true);
469             this.vista.jButton2.setEnabled(true);
470             this.vista.jButton13.setEnabled(true);
471             this.vista.jList1.setEnabled(true);
472         }
473     }
```

Gráfico de llamadas para esta función:



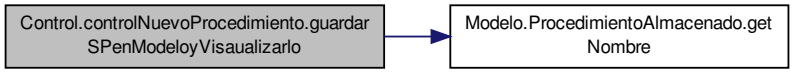
B.10.3.8. void Control.controlNuevoProcedimiento.guardarSPenModeloyVisaualizarlo ( )

Método que implementa la función guardar procedimiento alctual

Definición en la línea 558 del archivo controlNuevoProcedimiento.java.

```
558                                                 {
559         if (this.vista.jTextField1.getText().isEmpty()) {
560             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
    (*campo obligatorio)");
561             this.vista.jTextField1.requestFocus();
562         } else if (this.vista.jTextArea8.getText().isEmpty()) {
563             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
     cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
564         } else {
565             if (this.modelo == null) {
566                 this.modelo.setUsuario(new Usuario());
567             }
568             this.modelo.setNombre(this.vista.jTextField1.getText());
569             this.modelo.getCaracteristicas().obtenerCaracteristica(3).setComment(
570                     this.vista.jTextArea9.getText()
571             );
572             this.modelo.setCuerpoRutina(this.vista.jTextArea8.getText());
573             if (this.SPNodo == null) {
574                 this.SPNodo = new DefaultMutableTreeNode(this.modelo);
575             } else {
576                 this.SPNodo.setUserObject(this.modelo);
577             }
578             if (this.SPPadre.getChildCount() > 0) {
579                 int i;
580                 boolean existe = false;
581                 DefaultMutableTreeNode tmpDMTN;
582                 ProcedimientoAlmacenado tmpP;
583                 for (i = 0; i < this.SPPadre.getChildCount(); i++) {
584                     tmpDMTN = (DefaultMutableTreeNode) this.SPPadre.getChildAt(i);
585                     tmpP = (ProcedimientoAlmacenado) tmpDMTN.getUserObject();
586                     if (tmpP.getNombre().equals(this.modelo.getNombre())) {
587                         existe = true;
588                         break;
589                     }
590                 }
591                 if (existe) {
592                     if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, "Ya has salvaguardado
     un Procedimiento con este mismo nombre\n deseas reemplazarlo", "Guardar Procedimiento", JOptionPane.
    YES_NO_OPTION)) {
593                         this.SPPadre.insert(this.SPNodo, i);
594                         JOptionPane.showMessageDialog(null, "La operación se realizo de manera correcta");
595                     }
596                 } else {
597                     this.SPPadre.add(this.SPNodo);
598                     JOptionPane.showMessageDialog(null, "La operación se realizo de manera correcta");
599                 }
600             } else {
601                 this.SPPadre.add(this.SPNodo);
602                 JOptionPane.showMessageDialog(null, "La operación se realizo de manera correcta");
603             }
604         }
605     }
```

Gráfico de llamadas para esta función:



B.10.3.9. void Control.controlNuevoProcedimiento.inicializarTipoDatoSQLModeloVista ( ) [private]

Método que inicializa la vista que muestra los tipos de datos en SQL

Definición en la línea 149 del archivo controlNuevoProcedimiento.java.

```
149                                                 {
150         DefaultComboBoxModel tmpTipoDatoSQL = new DefaultComboBoxModel();
151         for (Iterator it = this.mapaTipoDatoSQL.entrySet().iterator(); it.hasNext();) {
152             Map.Entry<Integer, String> e = (Map.Entry<Integer, String>) it.next();
153             tmpTipoDatoSQL.addElement(e.getValue());
154         }
155         this.vista.tipoDatoSQL.setModel(tmpTipoDatoSQL);
156     }
```

B.10.3.10. void Control.controlNuevoProcedimiento.iniciar_Vista_NuevoProcedimiento ( )

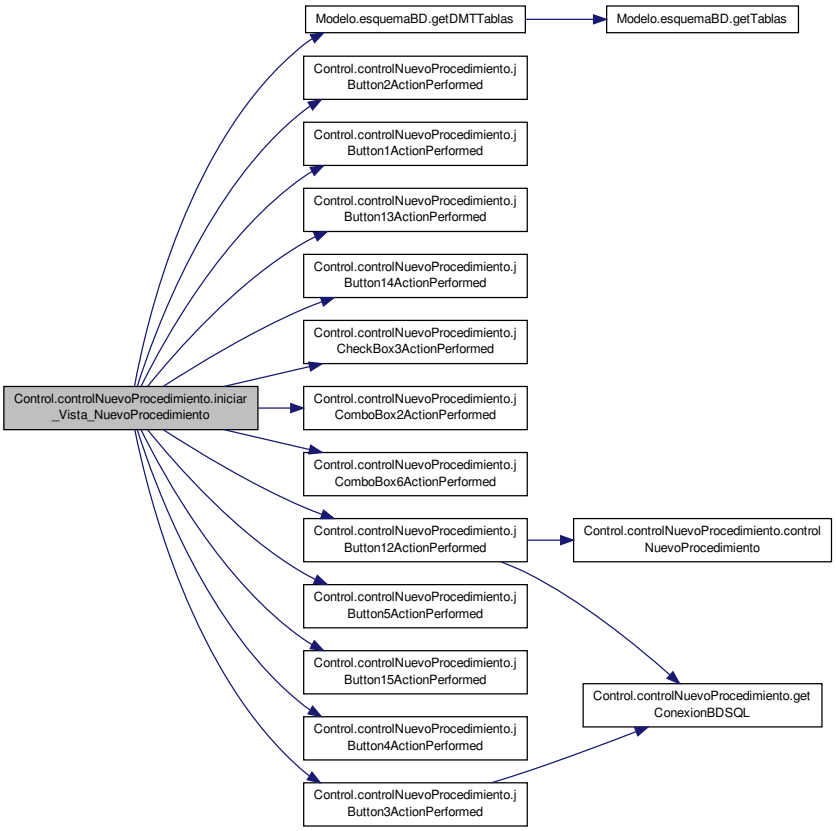
Método que inicializa la vista del editor de un nuevo procedimiento almacenado

Definición en la línea 202 del archivo controlNuevoProcedimiento.java.

```
202                                                 {
203         this.makeTypesName();
204         this.inicializarTipoDatoSQLModeloVista();
205         this.vista.tipoDatoSQL.setEditable(true);
206         this.vista.setTitle("Generador de código para un Nuevo procedimiento almacenado");
207         this.vista.setLocationRelativeTo(null);
208         this.setFocusPorDefecto();
209         this.vista.jTree1.setModel(new DefaultTreeModel(
210                 this.modeloEsquemaBD.getDMTTablas()
211         )
212         );
213         this.pordefecto();
214         this.vista.jButton2.addActionListener(new java.awt.event.ActionListener() {
215             public void actionPerformed(java.awt.event.ActionEvent evt) {
216                 jButton2ActionPerformed(evt);
217             }
218         });
219         this.vista.jButton1.addActionListener(new java.awt.event.ActionListener() {
220             public void actionPerformed(java.awt.event.ActionEvent evt) {
221                 jButton1ActionPerformed(evt);
222             }
223         });
224         this.vista.jButton13.addActionListener(new java.awt.event.ActionListener() {
225             public void actionPerformed(java.awt.event.ActionEvent evt) {
226                 jButton13ActionPerformed(evt);
227             }
228         });
229 
230         this.vista.jButton14.addActionListener(new java.awt.event.ActionListener() {
231             public void actionPerformed(java.awt.event.ActionEvent evt) {
232                 jButton14ActionPerformed(evt);
233             }
234         });
235         this.vista.jCheckBox3.addActionListener(new java.awt.event.ActionListener() {
```

```
236             public void actionPerformed(java.awt.event.ActionEvent evt) {
237                 jCheckBox3ActionPerformed(evt);
238             }
239         });
240         this.vista.jComboBox2.addActionListener(new java.awt.event.ActionListener() {
241             public void actionPerformed(java.awt.event.ActionEvent evt) {
242                 jComboBox2ActionPerformed(evt);
243             }
244         });
245         this.vista.jComboBox6.addActionListener(new java.awt.event.ActionListener() {
246             public void actionPerformed(java.awt.event.ActionEvent evt) {
247                 jComboBox6ActionPerformed(evt);
248             }
249         });
250         this.vista.jButton12.addActionListener(new java.awt.event.ActionListener() {
251             public void actionPerformed(java.awt.event.ActionEvent evt) {
252                 jButton12ActionPerformed(evt);
253             }
254         });
255         this.vista.jButton5.addActionListener(new java.awt.event.ActionListener() {
256             public void actionPerformed(java.awt.event.ActionEvent evt) {
257                 jButton5ActionPerformed(evt);
258             }
259         });
260         this.vista.jButton15.addActionListener(new java.awt.event.ActionListener() {
261             public void actionPerformed(java.awt.event.ActionEvent evt) {
262                 jButton15ActionPerformed(evt);
263             }
264         });
265         this.vista.jButton4.addActionListener(new java.awt.event.ActionListener() {
266             public void actionPerformed(java.awt.event.ActionEvent evt) {
267                 jButton4ActionPerformed(evt);
268             }
269         });
270         this.vista.jButton3.addActionListener(new java.awt.event.ActionListener() {
271             public void actionPerformed(java.awt.event.ActionEvent evt) {
272                 jButton3ActionPerformed(evt);
273             }
274         });
275         this.vista.setVisible(true);
276     }
```

Gráfico de llamadas para esta función:



B.10.3.11. void Control.controlNuevoProcedimiento.jButton12ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita crear un nuevo procedimiento a efecto de la invocacion del mismo en el cuerpo de la rutina del procedimiento actual.

Parámetros

evt	Evento

Definición en la línea 533 del archivo controlNuevoProcedimiento.java.

```
533                                                                     {
534         // TODO add your handling code here:
535         if (this.vista.jTextArea8.getSelectedText() != null) {
536             ProcedimientoAlmacenado modeloNuevoProcedimiento = new ProcedimientoAlmacenado();
537             if (this.SPNodo == null) {
538                 this.SPNodo = new DefaultMutableTreeNode();
539             }
540             nuevoProcedimiento vistaNuevoProcedimiento = new nuevoProcedimiento(new javax.swing.JFrame(),
    true);
541             controlNuevoProcedimiento ControlNP = new
    controlNuevoProcedimiento(
542                     this.getConexionBDSQL(),
543                     modeloNuevoProcedimiento,
544                     vistaNuevoProcedimiento,
```

```
545                     this.modeloEsquemaBD,
546                     this.vista.jTextArea8.getSelectedText(),
547                     this.SPNodo
548             );
549             ControlNP.iniciar_Vista_NuevoProcedimiento();
550         } else {
551             JOptionPane.showMessageDialog(null,"Debe seleccionar unicamente el nombre del procedimiento que
    desea crear");
552         }
553     }
```

Gráfico de llamadas para esta función:

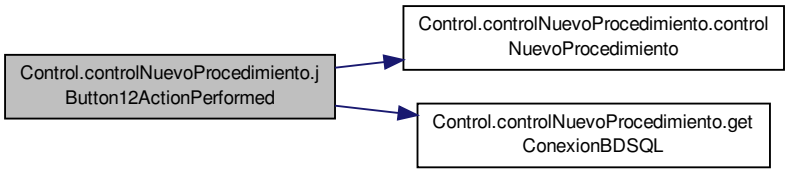
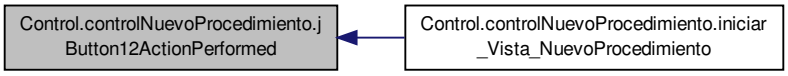


Gráfico de llamadas a esta función:



B.10.3.12. void Control.controlNuevoProcedimiento.jButton13ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita borrar un parámetro de un procedimiento almacenado.

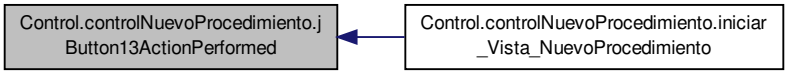
Parámetros

evt	Evento

Definición en la línea 442 del archivo controlNuevoProcedimiento.java.

```
442                                                                 {
443         this.setLastbuttonAction(this.vista.jButton13);
444         this.borrarParametroVistaModelo();
445     }
```

Gráfico de llamadas a esta función:



B.10.3.13. void Control.controlNuevoProcedimiento.jButton14ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita guardar los cambios en un parámetro de un procedimiento almacenado.

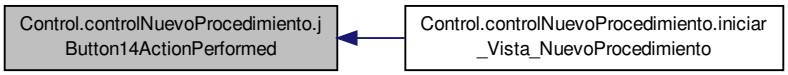
Parámetros

evt	Evento

Definición en la línea 482 del archivo controlNuevoProcedimiento.java.

```
482                                                                     {
483         this.guardarCambiosParametroVistaModelo();
484     }
```

Gráfico de llamadas a esta función:



B.10.3.14. void Control.controlNuevoProcedimiento.jButton15ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita guardar el procedimeinto actual

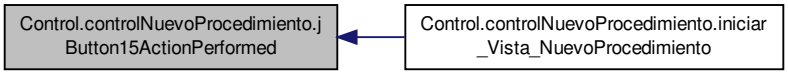
Parámetros

evt	Evento

Definición en la línea 646 del archivo controlNuevoProcedimiento.java.

```
646                                                                     {
647         // TODO add your handling code here:
648         this.guardarSPenModeloyVisaualizarlo();
649     }
```

Gráfico de llamadas a esta función:



B.10.3.15. void Control.controlNuevoProcedimiento.jButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita editar un parámetro de un procedimiento almacenado.

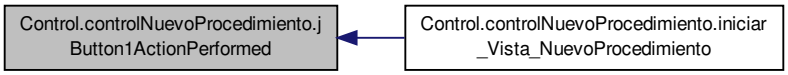
Parámetros

evt	Evento

Definición en la línea 403 del archivo controlNuevoProcedimiento.java.

```
403                                                                 {
404         this.editarParametroVistaModelo();
405         this.setLastbuttonAction(this.vista.jButton1);
406     }
```

Gráfico de llamadas a esta función:



B.10.3.16. void Control.controlNuevoProcedimiento.jButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita agregar un parámetro de un procedimiento almacenado.

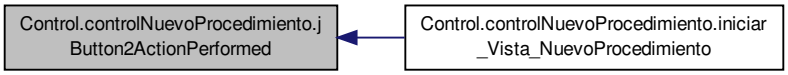
Parámetros

evt	Evento

Definición en la línea 340 del archivo controlNuevoProcedimiento.java.

```
340                                                                 {
341         this.agregarParametroVista();
342     }
```

Gráfico de llamadas a esta función:



B.10.3.17. void Control.controlNuevoProcedimiento.jButton3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita generar código SQL asociado al procedimiento actual

Parámetros

evt	Evento

Definición en la línea 682 del archivo controlNuevoProcedimiento.java.

```
682                                                                         {
683         // TODO add your handling code here:
684         if (this.vista.jTextField1.getText().isEmpty()) {
685             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
    (*campo obligatorio)");
686             this.vista.jTextField1.requestFocus();
687         } else if (this.vista.jTextArea8.getText().isEmpty()) {
688             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
     cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
689         } else {
690             this.modelo.setUsuario(new Usuario());
691             this.modelo.setNombre(this.vista.jTextField1.getText());
692             this.modelo.getCaracteristicas().obtenerCaracteristica(3).setComment(
693                     this.vista.jTextArea9.getText()
694             );
695             this.modelo.setCuerpoRutina(this.vista.jTextArea8.getText());
696             EditorCodigoGenerado vistaPGen = new EditorCodigoGenerado(new javax.swing.JFrame(), true);
697             controlCodigoProcedimientoGenerado controlSPCodGen = new controlCodigoProcedimientoGenerado(
698                     this.getConexionBDSQL(),
699                     this.modelo,
700                     vistaPGen
701             );
702             controlSPCodGen.iniciar_vista();
703         }
704     }
```

Gráfico de llamadas para esta función:

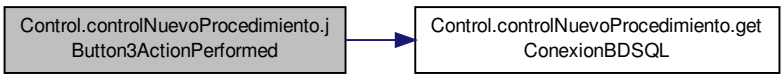
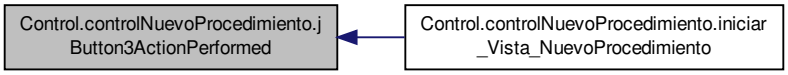


Gráfico de llamadas a esta función:



B.10.3.18. void Control.controlNuevoProcedimiento.jButton4ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita una vista previa del código SQL asociado al procedimiento actual.

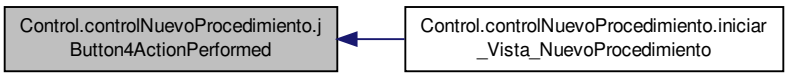
Parámetros

evt	Evento

Definición en la línea 658 del archivo controlNuevoProcedimiento.java.

```
658                                                                     {
659         if (this.vista.jTextField1.getText().isEmpty()) {
660             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
      (*campo obligatorio)");
661             this.vista.jTextField1.requestFocus();
662         } else if (this.vista.jTextArea8.getText().isEmpty()) {
663             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
       cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
664 
665         } else {
666             this.modelo.setUsuario(new Usuario());
667             this.modelo.setNombre(this.vista.jTextField1.getText());
668             this.modelo.getCaracteristicas().obtenerCaracteristica(3).setComment(
669                     this.vista.jTextArea9.getText()
670             );
671             this.modelo.setCuerpoRutina(this.vista.jTextArea8.getText());
672             JOptionPane.showMessageDialog(null, this.modelo.generarProcedimientoSQL(false));
673         }
674     }
```

Gráfico de llamadas a esta función:



B.10.3.19. void Control.controlNuevoProcedimiento.jButton5ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita cerrar la vista del editor de un nuevo procedimiento almacenado

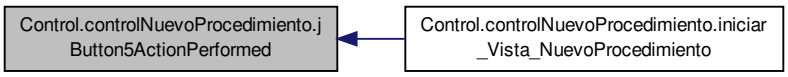
Parámetros

evt	Evento

Definición en la línea 635 del archivo controlNuevoProcedimiento.java.

```
635                                                                         {
636         // TODO add your handling code here:
637         this.vista.dispose();
638     }
```

Gráfico de llamadas a esta función:



B.10.3.20. void Control.controlNuevoProcedimiento.jCheckBox3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer si un procedimiento es determinista.

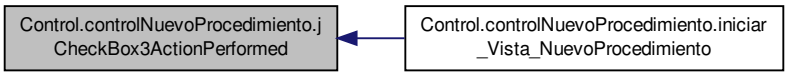
Parámetros

evt	Evento

Definición en la línea 492 del archivo controlNuevoProcedimiento.java.

```
492                                                                         {
493         if (this.vista.jCheckBox3.getModel().isSelected()) {
494             this.modelo.getCaracteristicas().obtenerCaracteristica(0).setValue("DETERMINISTIC");
495         } else {
496             this.modelo.getCaracteristicas().obtenerCaracteristica(0).setValue("NOT DETERMINISTIC");
497         }
498     }
```

Gráfico de llamadas a esta función:



B.10.3.21. void Control.controlNuevoProcedimiento.jComboBox2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta, al evento generado por un componente de la vista que solicita establecer el contexto de seguridad en un procedimiento almacenado.

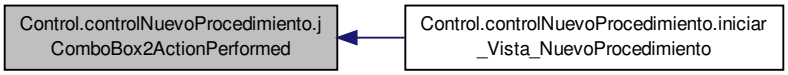
Parámetros

evt	Evento

Definición en la línea 507 del archivo controlNuevoProcedimiento.java.

```
507                                                                         {
508         this.modelo.getCaracteristicas().obtenerCaracteristica(1).setComment(
509                 this.vista.jComboBox2.getSelectedItem().toString()
510         );
511     }
```

Gráfico de llamadas a esta función:



B.10.3.22. void Control.controlNuevoProcedimiento.jComboBox6ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta, al evento generado por un componente de la vista que solicita establecer la naturaleza de los datos usados por un procedimiento almacenado.

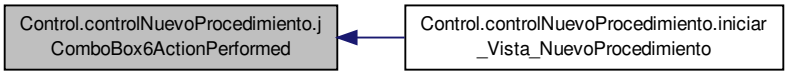
Parámetros

evt	Evento

Definición en la línea 520 del archivo controlNuevoProcedimiento.java.

```
520                                                                         {
521         this.modelo.getCaracteristicas().obtenerCaracteristica(2).setValue(
522                 this.vista.jComboBox6.getSelectedItem().toString()
523         );
524     }
```

Gráfico de llamadas a esta función:



B.10.3.23. void Control.controlNuevoProcedimiento.limpiarCampos ( )

Método que limpia la vista del editor de un nuevo procedimiento almacenado, es decir, establece el estado por defecto de la vista

Definición en la línea 611 del archivo controlNuevoProcedimiento.java.

```
611                                 {
612         this.vista.jTextField1.setText("");
613         this.limpiarCamposParametros();
614         this.limpiarCamposCaracteristicas();
615         this.vista.jTextArea8.setText("# agrega todo tu codigo de control aquí: cualquier tipo de datos
        MySQL valido\n");
616     }
```

B.10.3.24. void Control.controlNuevoProcedimiento.limpiarCamposCaracteristicas ( )

Método que establece el estado por defecto de las características de un procedimiento almacenado tanto en la vista como en el modelo

Definición en la línea 622 del archivo controlNuevoProcedimiento.java.

```
622                                             {
623         this.vista.jCheckBox3.setSelected(false);
624         this.vista.jComboBox2.setSelectedIndex(0);
625         this.vista.jComboBox6.setSelectedIndex(0);
626         this.vista.jTextArea9.setText("Sin comentarios");
627     }
```

B.10.3.25. void Control.controlNuevoProcedimiento.limpiarCamposParametros ( ) [private]

Método que limpia los campos asociados a los parametros de un procedimiento en la vista

Definición en la línea 306 del archivo controlNuevoProcedimiento.java.

```
306                                         {
307         this.vista.jTextField2.setText(null);
308         this.vista.tipoDatoSQL.setSelectedIndex(0);
309         this.vista.jComboBox4.setSelectedIndex(0);
310         this.vista.jComboBox4.requestFocus();
311     }
```

B.10.3.26. void Control.controlNuevoProcedimiento.makeTypesName ( ) [private]

Método que construye el mapa de los tipos de datos en SQL

Definición en la línea 134 del archivo controlNuevoProcedimiento.java.

```
134                                 {
135         this.mapaTipoDatoSQL = new HashMap();
136         Field[] fields = java.sql.Types.class.getFields();
137         for (int i = 0; i < fields.length; i++) {
138             try {
139                 this.mapaTipoDatoSQL.put((Integer) fields[i].get(null), fields[i].getName());
140             } catch (IllegalAccessException e) {
141                 e.printStackTrace();
142             }
143         }
144     }
```

B.10.3.27. void Control.controlNuevoProcedimiento.MVNombreProcedimiento ( )

Definición en la línea 184 del archivo controlNuevoProcedimiento.java.

```
184                                         {
185         this.vista.jTextField1.setText(
186                 this.modelo.getNombre()
187         );
188     }
```

B.10.3.28. void Control.controlNuevoProcedimiento.pordefecto ( )

Método que establece el estado por defecto del procedimiento almacenado actual

Definición en la línea 194 del archivo controlNuevoProcedimiento.java.

```
194                             {
195         this.MVNombreProcedimiento();
196     }
```

B.10.3.29. void Control.controlNuevoProcedimiento.setConexionBDSQL (Connection *C*)

Método que establece la variable que contiene la conexión establecida con alguna base de datos

Parámetros

C	conexión establecida con alguna base de datos

Definición en la línea 75 del archivo controlNuevoProcedimiento.java.

```
75                                              {
76          this.conexionBDSQL = C;
77     }
```

B.10.3.30. void Control.controlNuevoProcedimiento.setFocusPorDefecto ( )

Método que establece el orden de los elementos de la interfaz para la política de recorrimiento del focus

Definición en la línea 162 del archivo controlNuevoProcedimiento.java.

```
162                                     {
163         ArrayList orden = new ArrayList();
164         orden.add(this.vista.jTextField1);
165         orden.add(this.vista.jComboBox4);
166         orden.add(this.vista.jTextField2);
167         orden.add(this.vista.tipoDatoSQL);
168         orden.add(this.vista.jButton2);
169         orden.add(this.vista.jButton1);
170         orden.add(this.vista.jButton13);
171         orden.add(this.vista.jList1);
172         orden.add(this.vista.jCheckBox3);
173         orden.add(this.vista.jComboBox2);
174         orden.add(this.vista.jComboBox6);
175         orden.add(this.vista.jTextArea9);
176         orden.add(this.vista.jTextArea8);
177         orden.add(this.vista.jTree1);
178         orden.add(this.vista.jButton4);
179         orden.add(this.vista.jButton3);
180         MiFocusTraversalPolicy newPolicy = new MiFocusTraversalPolicy(orden);
181         this.vista.setFocusTraversalPolicy(newPolicy);
182     }
```

B.10.3.31. void Control.controlNuevoProcedimiento.setLastbuttonAction (Component *lasButtonAction*)

Método que establece el ultimo componente que provocó algún evento

Parámetros

lasButtonAction	Componente

Definición en la línea 94 del archivo controlNuevoProcedimiento.java.

```
94                                                          {
95         this.lasButtonAction = lasButtonAction;
96     }
```

B.10.4. Documentación de los datos miembro

B.10.4.1. Connection Control.controlNuevoProcedimiento.conexionBDSQL [private]

Conexión establecida con alguna base de datos

Definición en la línea 47 del archivo controlNuevoProcedimiento.java.

B.10.4.2. Component Control.controlNuevoProcedimiento.lasButtonAction [private]

Componente de la interfaz gráfica

Definición en la línea 63 del archivo controlNuevoProcedimiento.java.

B.10.4.3. Map Control.controlNuevoProcedimiento.mapaTipoDatoSQL

Mapa de los tipos de datos en SQL

Definición en la línea 67 del archivo controlNuevoProcedimiento.java.

B.10.4.4. ProcedimientoAlmacenado Control.controlNuevoProcedimiento.modelo [private]

Modelo: Objeto de la clase que implementa un procedimiento almacenado

Definición en la línea 43 del archivo controlNuevoProcedimiento.java.

B.10.4.5. esquemaBD Control.controlNuevoProcedimiento.modeloEsquemaBD [private]

Modelo: Objeto de la clase que implementa un esquema de una base de datos

Definición en la línea 51 del archivo controlNuevoProcedimiento.java.

B.10.4.6. DefaultMutableTreeNode Control.controlNuevoProcedimiento.SPNodo [private]

Nodo asociado al procedimiento actual

Definición en la línea 55 del archivo controlNuevoProcedimiento.java.

B.10.4.7. DefaultMutableTreeNode Control.controlNuevoProcedimiento.SPPadre [private]

Nodo padre asociado al nodo actual

Definición en la línea 59 del archivo controlNuevoProcedimiento.java.

B.10.4.8. nuevoProcedimiento Control.controlNuevoProcedimiento.vista [private]

Vista: Editor de un nuevo procedimiento almacenado

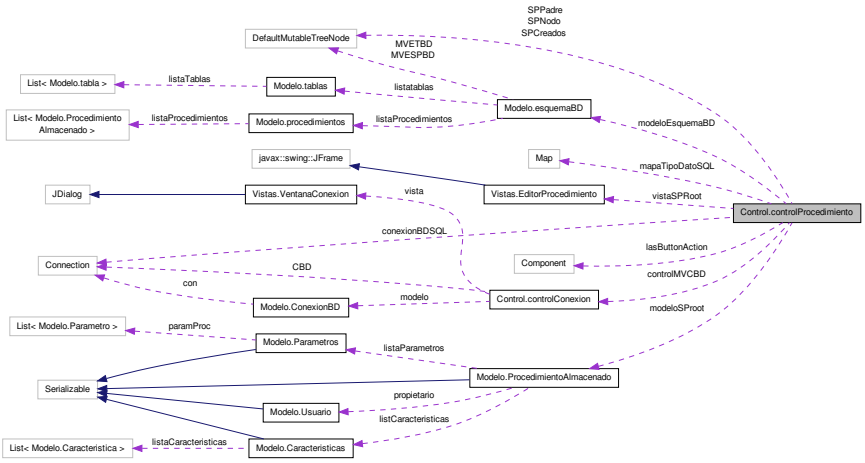
Definición en la línea 39 del archivo controlNuevoProcedimiento.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/controlNuevoProcedimiento.java

B.11. Referencia de la Clase Control.controlProcedimiento

Diagrama de colaboración para Control.controlProcedimiento:



Métodos públicos

- void setConexionBDSQL (Connection C)
- Connection getConexionBDSQL ()
- String getTypesName (int tipo)
- void setLastbuttonAction (Component lasButtonAction)
- Component getlasButtonAction ()
- controlProcedimiento (ProcedimientoAlmacenado modelo, EditorProcedimiento vista)
- void iniciar_vista_SPRoot ()
- boolean llavePrimaria (ResultSet rsColPK, String nombreColumna)
- void MVETBD ()
- void setFocusPorDefecto ()
- void limpiarCamposCaracteristicas ()
- void limpiarCampos ()
- void ModuloEntradaDatos ()
- void guardarSPenModeloyVisaualizarlo ()
- void agregarCaracModeloVista ()
- void pordefecto ()

Atributos públicos

- Map mapaTipoDatoSQL

Métodos privados

- void makeTypesName ()
- void guardarCambiosParametroVistaModelo ()
- void agregarParametroVista ()
- void borrarParametroVistaModelo ()
- void editarParametroVistaModelo ()
- void agregarParamModeloVista ()
- void inicializarTipoDatoSQLModeloVista ()
- void limpiarCamposParametros ()
- void jButton4ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton7ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton8ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox1ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox2ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox6ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton5ActionPerformed (java.awt.event.ActionEvent evt)
- void jCheckBox3ActionPerformed (java.awt.event.ActionEvent evt)
- void jList1MouseClicked (java.awt.event.MouseEvent evt)
- void jList1KeyPressed (java.awt.event.KeyEvent evt)
- String getProcedimientoSQLNodo (DefaultMutableTreeNode Nodo, boolean visualizador)
- String generadorCodigoSQLProcedimientosAnidados (boolean visualizador)
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)
- boolean esTabla (javax.swing.tree.TreeModel VETBD, Object seleccion)
- void jMenuItem1ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton9ActionPerformed (java.awt.event.ActionEvent evt)
- void jTree2ValueChanged (javax.swing.event.TreeSelectionEvent evt)
- void jButton6ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton10ActionPerformed (java.awt.event.ActionEvent evt)
- void jTree2KeyPressed (java.awt.event.KeyEvent evt)
- void jMenuItem2ActionPerformed (java.awt.event.ActionEvent evt)
- void jMenuItem3ActionPerformed (java.awt.event.ActionEvent evt)

Atributos privados

- EditorProcedimiento vistaSPRoot
- ProcedimientoAlmacenado modeloSProot
- Component lasButtonAction
- Connection conexionBDSQL
- controlConexion controlMVCBD
- esquemaBD modeloEsquemaBD = new esquemaBD()
- DefaultMutableTreeNode SPCreados = new DefaultMutableTreeNode("Procedimientos Creados")
- DefaultMutableTreeNode SPNodo
- DefaultMutableTreeNode SPPadre

B.11.1. Descripción detallada

Clase que implementa el control entre el modelo y la vista de un procedimiento almacenado

Autor

ivan

Definición en la línea 53 del archivo controlProcedimiento.java.

B.11.2. Documentación del constructor y destructor

B.11.2.1. Control.controlProcedimiento.controlProcedimiento (ProcedimientoAlmacenado *modelo,* EditorProcedimiento *vista*)

Constructor por defecto de la clase, establece el modelo y la vista del editor de un procedimiento almacenado

Parámetros

modelo	Modelo: Objeto de la clase que implementa un procedimiento almacenado
vista	Vista: Editor de un procedimiento almacenado

Definición en la línea 174 del archivo controlProcedimiento.java.

```
174                                                                                 {
175         this.vistaSPRoot = vista;
176         this.modeloSProot = modelo;
177     }
```

B.11.3. Documentación de las funciones miembro

B.11.3.1. void Control.controlProcedimiento.agregarCaracModeloVista ( )

Método que establece las carcaterísticas de un procedimiento almacenado del modelo a la vista

Definición en la línea 1197 del archivo controlProcedimiento.java.

```
1197                                        {
1198         this.vistaSPRoot.jCheckBox3.setSelected(
1199                 this.modeloSProot.getCaracteristicas().obtenerCaracteristica(0).getValue().equals(
1200                         this.vistaSPRoot.jCheckBox3.getText()
1201                 )
1202         );
1203         this.vistaSPRoot.jComboBox2.setSelectedItem(this.modeloSProot.getCaracteristicas().
     obtenerCaracteristica(1).getValue());
1204         this.vistaSPRoot.jComboBox6.setSelectedItem(this.modeloSProot.getCaracteristicas().
     obtenerCaracteristica(2).getValue());
1205         this.vistaSPRoot.jTextArea3.setText(this.modeloSProot.getCaracteristicas().obtenerCaracteristica(3)
     .getComment());
1206     }
```

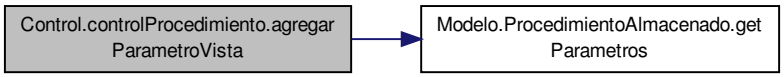
B.11.3.2. void Control.controlProcedimiento.agregarParametroVista ( ) [private]

Método que implementa la función agregar parámetro de la vista al modelo

Definición en la línea 520 del archivo controlProcedimiento.java.

```
520                                         {
521         if (this.vistaSPRoot.jTextField2.getText().isEmpty()) {
522             JOptionPane.showMessageDialog(null, "nombre/parametro vacio\n");
523             this.vistaSPRoot.jComboBox4.requestFocus();
524         } else {
525             if (this.modeloSProot.getParametros() == null) {
526                 this.modeloSProot.setParametros(new Parametros());
527             }
528             this.modeloSProot.getParametros().agregarParametro(new Parametro(
529                     (String) this.vistaSPRoot.jComboBox4.getSelectedItem().toString(),
530                     this.vistaSPRoot.jTextField2.getText(),
531                     this.vistaSPRoot.tipoDatoSQL.getSelectedItem().toString()
532             ));
533             this.agregarParamModeloVista();
534             this.limpiarCamposParametros();
535         }
536     }
```

Gráfico de llamadas para esta función:



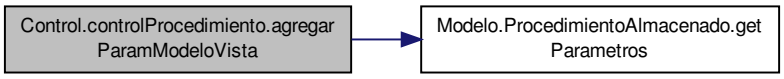
B.11.3.3. void Control.controlProcedimiento.agregarParamModeloVista ( ) [private]

Método que implementa la función agregar parámetro del modelo a la vista

Definición en la línea 614 del archivo controlProcedimiento.java.

```
614                                         {
615         if (this.modeloSProot.getParametros() != null) {
616             Parametro item;
617             ArrayList<String> P = new ArrayList<>();
621             if (!this.modeloSProot.getParametros().listaParamVacia()) {
622                 for (int i = 0; i < this.modeloSProot.getParametros().numElementos() - 1; i++) {
623                     item = (Parametro) this.modeloSProot.
      getParametros().obtenerParametro(i);
624                     P.add(item.generarSQL());
625                 }
626                 item = (Parametro) this.modeloSProot.getParametros().getParamProc(
      ).get(
627                         this.modeloSProot.getParametros().numElementos() - 1);
628                 P.add(item.generarSQL());
629             }
633             DefaultListModel<String> model = new DefaultListModel<>();
634             for (String s : P) {
635                 model.addElement(s);
636             }
637             this.vistaSPRoot.jList1.setModel(model);
638         }
639     }
```

Gráfico de llamadas para esta función:



B.11.3.4. void Control.controlProcedimiento.borrarParametroVistaModelo ( ) [private]

Método que implementa la función borrar parámetro, garantiza que los cambios generados al ejecutar la función se vean reflejados en la vista y en el modelo

Definición en la línea 543 del archivo controlProcedimiento.java.

```
543                                             {
544         if (this.vistaSPRoot.jList1.getModel().getSize() > 0) {
545             if (this.vistaSPRoot.jList1.isSelectionEmpty()) {
546                 JOptionPane.showMessageDialog(null, "Seleleccione algun parametro");
547                 this.vistaSPRoot.jList1.setSelectedIndex(0);
548                 if (this.vistaSPRoot.jList1.getModel().getSize() > 1) {
549                     this.vistaSPRoot.jList1.requestFocus();
550 
551                 } else {
552                     this.vistaSPRoot.jButton7.requestFocus();
553                 }
554             } else {
555                 this.modeloSProot.getParametros().getParamProc().remove(
    this.vistaSPRoot.jList1.getSelectedIndex());
556                 this.setLastbuttonAction(null);
557                 this.agregarParamModeloVista();
558                 this.vistaSPRoot.jComboBox4.requestFocus();
559             }
560         } else {
561             JOptionPane.showMessageDialog(null, "Agrege almenos un Parametro");
562             this.vistaSPRoot.jComboBox4.requestFocus();
563         }
564     }
```

B.11.3.5. void Control.controlProcedimiento.editarParametroVistaModelo ( ) [private]

Método que implementa la función editar parámetro, garantiza que los cambios en la vista se vean reflejados en el modelo

Definición en la línea 570 del archivo controlProcedimiento.java.

```
570                                             {
571         if (this.vistaSPRoot.jList1.getModel().getSize() > 0) {
572             if (this.vistaSPRoot.jList1.isSelectionEmpty()) {
573                 JOptionPane.showMessageDialog(null, "Seleccione algun parametro");
574                 this.vistaSPRoot.jList1.setSelectedIndex(0);
575                 if (this.vistaSPRoot.jList1.getModel().getSize() > 1) {
576                     this.vistaSPRoot.jList1.requestFocus();
577                 } else {
578                     this.vistaSPRoot.jButton1.requestFocus();
579                 }
580             } else {
581                 this.vistaSPRoot.jComboBox4.setSelectedItem(
582                         this.modeloSProot.getParametros().obtenerParametro(this.
    vistaSPRoot.jList1.getSelectedIndex()).getTipoES()
583                 );
```

```
584                 this.vistaSPRoot.jTextField2.setText(
585                         this.modeloSProot.getParametros().obtenerParametro(this.
    vistaSPRoot.jList1.getSelectedIndex()).getNombre()
586                 );
587                 this.vistaSPRoot.tipoDatoSQL.setSelectedItem(
588                         this.modeloSProot.getParametros().obtenerParametro(this.
    vistaSPRoot.jList1.getSelectedIndex()).getTipoDato()
589                 );
590                 this.vistaSPRoot.jButton1.setEnabled(false);
591                 this.vistaSPRoot.jButton2.setEnabled(false);
592                 this.vistaSPRoot.jButton7.setEnabled(false);
593                 this.vistaSPRoot.jList1.setEnabled(false);
594                 this.vistaSPRoot.jButton8.setEnabled(true);
595                 ArrayList tmpOrden = new ArrayList();
596                 tmpOrden.add(this.vistaSPRoot.jComboBox4);
597                 tmpOrden.add(this.vistaSPRoot.jTextField2);
598                 tmpOrden.add(this.vistaSPRoot.tipoDatoSQL);
599                 tmpOrden.add(this.vistaSPRoot.jButton8);
600                 MiFocusTraversalPolicy newTMPPolicy = new MiFocusTraversalPolicy(tmpOrden);
601                 vistaSPRoot.setFocusTraversalPolicy(newTMPPolicy);
602                 this.vistaSPRoot.jComboBox4.requestFocus();
603                 this.modeloSProot.getParametros().modificarParametro(this.
    vistaSPRoot.jList1.getSelectedIndex());
604             }
605         } else {
606             JOptionPane.showMessageDialog(null, "Agrege almenos un Parametro");
607             this.vistaSPRoot.jComboBox4.requestFocus();
608         }
609     }
```

B.11.3.6. boolean Control.controlProcedimiento.esTabla (javax.swing.tree.TreeModel *VETBD,* Object *seleccion*) [private]

Método que identifica si la selección en el árbol de tablas es precisamente una tabla

Parámetros

VETBD	Árbol de tablas
seleccion	Objeto seleccionado en el árbol de tablas.

Devuelve

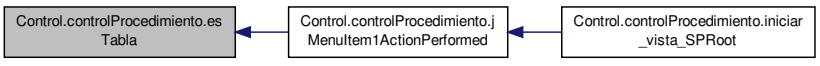
boolean

- true indica que el elemento de la seleción en el árbol de tablas, es una tabla
- false indica que el elemento de la seleción en el árbol de tablas, no es una tabla

Definición en la línea 1023 del archivo controlProcedimiento.java.

```
1023                                                                                {
1024
1025         return (!VETBD.getRoot().equals(seleccion) && !VETBD.isLeaf(seleccion));
1026
1027     }
```

Gráfico de llamadas a esta función:



B.11.3.7. String Control.controlProcedimiento.generadorCodigoSQLProcedimientosAnidados (boolean *visualizador*) [private]

Método que genera y retorna el código SQL asociado al procedimiento actual y a los invocaciones a otros, si se diera el caso.

Parámetros

visualizador	▪ true indica que el código SQL asociado al procedimiento contendra etiquetas html para el módulo "visualizador de código". ▪ false indica que el código SQL aosicado al procedimiento no contendra etiquetas html y podra ser usado ya sea para enviar directamenta a la base de datos o en un archivo.

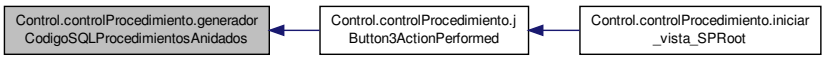
Devuelve

String Código SQL

Definición en la línea 936 del archivo controlProcedimiento.java.

```
936                                                                                 {
937         ProcedimientoAlmacenado tmpP = (ProcedimientoAlmacenado) this.SPNodo.getUserObject();
938         String SQL = tmpP.generarProcedimientoSQL(visualizador);
939         for (int j = 0; j < this.SPNodo.getChildCount(); j++) {
940             SQL = SQL + this.getProcedimientoSQLNodo((DefaultMutableTreeNode) this.
      SPNodo.getChildAt(j), visualizador);
941         }
942         return SQL;
943     }
```

Gráfico de llamadas a esta función:



B.11.3.8. Connection Control.controlProcedimiento.getConexionBDSQL ( )

Método que obtiene la variable que contiene la conexión establecida con alguna base de datos

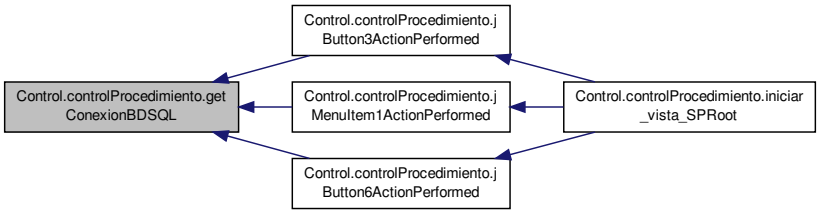
Devuelve

Connection conexión establecida con alguna base de datos

Definición en la línea 116 del archivo controlProcedimiento.java.

```
116                                     {
117         return this.conexionBDSQL;
118     }
```

Gráfico de llamadas a esta función:



B.11.3.9. Component Control.controlProcedimiento.getlasButtonAction ( )

Método que obtiene el ultimo componente que provocó algún evento

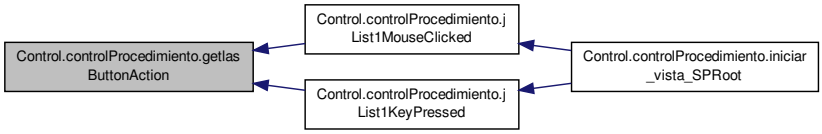
Devuelve

Component Componente

Definición en la línea 162 del archivo controlProcedimiento.java.

```
162                                     {
163         return this.lasButtonAction;
164     }
```

Gráfico de llamadas a esta función:



B.11.3.10. String Control.controlProcedimiento.getProcedimientoSQLNodo (DefaultMutableTreeNode *Nodo,* boolean *visualizador*) [private]

Método que genera y retorna el código SQL del procedimiento asociado al nodo proporcionado.

Parámetros

Nodo	Nodo
visualizador	■ true indica que el código SQL asociado al procedimiento contendra etiquetas html para el módulo "visualizador de código". ■ false indica que el código SQL aosicado al procedimiento no contendra etiquetas html y podra ser usado ya sea para enviar directamenta a la base de datos o en un archivo.

Devuelve

String Código SQL

Definición en la línea 908 del archivo controlProcedimiento.java.

```
908                                                                                 {
909         String SQL = "";
910         ProcedimientoAlmacenado tmpProcedimiento = (ProcedimientoAlmacenado) Nodo.getUserObject();
911         if (Nodo.getChildCount() > 0) {
912             SQL = SQL + tmpProcedimiento.generarProcedimientoSQL(visualizador);
913             for (int i = 0; i < Nodo.getChildCount(); i++) {
914                 SQL = SQL + this.getProcedimientoSQLNodo((DefaultMutableTreeNode) Nodo.getChildAt(i),
    visualizador);
915             }
916         } else {
917             SQL = SQL + tmpProcedimiento.generarProcedimientoSQL(visualizador);
918         }
919         return SQL;
920     }
```

B.11.3.11. String Control.controlProcedimiento.getTypesName (int *tipo*)

Método que obtiene el tipo de dato en SQL asociado al idenficador numérico proporcionado

Parámetros

tipo	Idenficador numérico asociado a un tipo de dato en SQL

Devuelve

String Nombre del tipo de dato en SQL

Definición en la línea 127 del archivo controlProcedimiento.java.

```
127                                         {
128
129         return this.mapaTipoDatoSQL.get(tipo).toString();
130
131     }
```

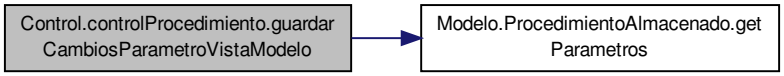
B.11.3.12. void Control.controlProcedimiento.guardarCambiosParametroVistaModelo ( ) [private]

Método que implementa la función guardar los cambios en el parámetro, garantiza que los cambios generados al ejecutar la función se vean reflejados en la vista y en el modelo

Definición en la línea 494 del archivo controlProcedimiento.java.

```
494                                                 {
495         if (this.modeloSProot.getParametros().getModParam() >= 0) {
496             this.modeloSProot.getParametros().obtenerParametro(
497                     this.modeloSProot.getParametros().getModParam()
498             ).setTipoES((String) this.vistaSPRoot.jComboBox4.getSelectedItem()
    .toString());
499             this.modeloSProot.getParametros().obtenerParametro(
500                     this.modeloSProot.getParametros().getModParam()
501             ).setNombre(this.vistaSPRoot.jTextField2.getText());
502             this.modeloSProot.getParametros().obtenerParametro(
503                     this.modeloSProot.getParametros().getModParam()
504             ).setTipoDato(this.vistaSPRoot.tipoDatoSQL.getSelectedItem().toString());
505             this.agregarParamModeloVista();
506             this.limpiarCamposParametros();
507             this.setFocusPorDefecto();
508             this.setLastbuttonAction(null);
509             this.vistaSPRoot.jButton8.setEnabled(false);
510             this.vistaSPRoot.jButton1.setEnabled(true);
511             this.vistaSPRoot.jButton2.setEnabled(true);
512             this.vistaSPRoot.jButton7.setEnabled(true);
513             this.vistaSPRoot.jList1.setEnabled(true);
514         }
515     }
```

Gráfico de llamadas para esta función:



B.11.3.13. void Control.controlProcedimiento.guardarSPenModeloyVisaualizarlo ( )

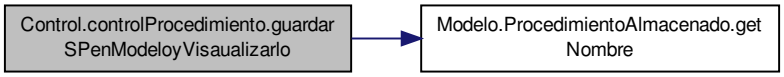
Método que implementa la función guardar procedimiento alctual

Definición en la línea 1115 del archivo controlProcedimiento.java.

```
1115                                                 {
1116         if (this.vistaSPRoot.jTextField1.getText().isEmpty()) {
1117             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
    (*campo obligatorio)");
1118             this.vistaSPRoot.jTextField1.requestFocus();
1119         } else if (this.vistaSPRoot.jTextArea2.getText().isEmpty()) {
1120             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
     cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
1121         } else {
1122             ProcedimientoAlmacenado tmpP;
1123             this.ModuloEntradaDatos();
1124             if (this.SPNodo == null) {
1125                 this.SPNodo = new DefaultMutableTreeNode(this.modeloSProot);
1126                 this.SPPadre = this.SPCreados;
1127             } else {
1128                 this.SPNodo.setUserObject(this.modeloSProot);
1129                 if (this.SPNodo.getParent() != null) {
1130                     this.SPPadre = (DefaultMutableTreeNode) this.SPNodo.getParent();
1131                 } else {
1132                     this.SPPadre = this.SPCreados;
1133                 }
1134             }
1135             if (this.SPPadre.getChildCount() > 0) {
1136                 int i;
1137                 boolean existe = false;
1138                 DefaultMutableTreeNode tmpDMTN;
1139 
1140                 for (i = 0; i < this.SPPadre.getChildCount(); i++) {
```

```
1141                    tmpDMTN = (DefaultMutableTreeNode) this.SPPadre.getChildAt(i);
1142                    tmpP = (ProcedimientoAlmacenado) tmpDMTN.getUserObject();
1143                    if (tmpP.getNombre().equals(this.modeloSProot.getNombre())) {
1144                        existe = true;
1145                        break;
1146                    }
1147                }
1148                if (existe) {
1149                    if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, "Ya has salvaguardado
       un Procedimiento con este mismo nombre\n deseas reemplazarlo", "Guardar Procedimiento", JOptionPane.
      YES_NO_OPTION)) {
1150
1151                        this.SPPadre.insert(this.SPNodo, i);
1152                        this.vistaSPRoot.jTree2.setModel(
1153                                new DefaultTreeModel(
1154                                        this.SPCreados
1155                                )
1156                        );
1157                    }
1158                } else {
1159                    this.SPCreados.add(this.SPNodo);
1160                    this.vistaSPRoot.jTree2.setModel(
1161                            new DefaultTreeModel(
1162                                    this.SPCreados
1163                            )
1164                    );
1165                }
1166            } else {
1167                if (this.SPNodo == null) {
1168                    this.SPNodo = new DefaultMutableTreeNode(this.modeloSProot);
1169                } else {
1170                    this.SPNodo.setUserObject(this.modeloSProot);
1171                }
1172                this.SPPadre.add(this.SPNodo);
1173                this.vistaSPRoot.jTree2.setModel(
1174                        new DefaultTreeModel(
1175                                this.SPCreados
1176                        )
1177                );
1178            }
1179        }
1180    }
```

Gráfico de llamadas para esta función:



B.11.3.14. void Control.controlProcedimiento.inicializarTipoDatoSQLModeloVista ( ) [private]

Método que inicializa la vista que muestra los tipos de datos en SQL

Definición en la línea 644 del archivo controlProcedimiento.java.

```
644                                                 {
645        DefaultComboBoxModel tmpTipoDatoSQL = new DefaultComboBoxModel();
646        for (Iterator it = this.mapaTipoDatoSQL.entrySet().iterator(); it.hasNext();) {
647            Entry<Integer, String> e = (Entry<Integer, String>) it.next();
648            tmpTipoDatoSQL.addElement(e.getValue());
649        }
650        this.vistaSPRoot.tipoDatoSQL.setModel(tmpTipoDatoSQL);
651    }
```

B.11.3.15. void Control.controlProcedimiento.iniciar_vista_SPRoot ( )

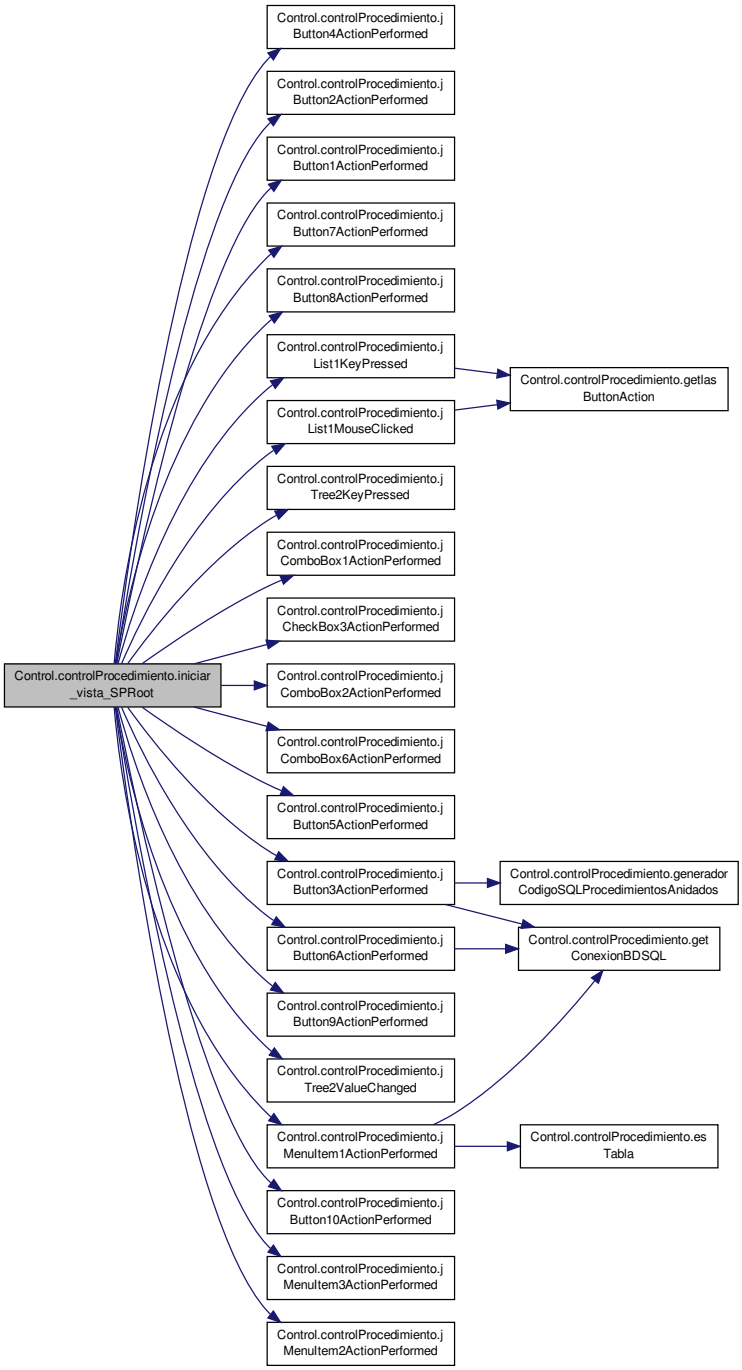
Método que inicializa la vista del editor de un procedimiento almacenado

Definición en la línea 182 del archivo controlProcedimiento.java.

```
182                                     {
183         this.makeTypesName();
184         this.inicializarTipoDatoSQLModeloVista();
185         javax.swing.tree.DefaultMutableTreeNode treeNode1 = null;
186         this.vistaSPRoot.jTree1.setModel(new javax.swing.tree.DefaultTreeModel(treeNode1));
187         this.vistaSPRoot.jTree2.setModel(new DefaultTreeModel(this.SPCreados));
188         this.vistaSPRoot.tipoDatoSQL.setEditable(true);
189         vistaSPRoot.setTitle("Generador de código SQL para procedimientos almacenados");
190         vistaSPRoot.setLocationRelativeTo(null);
191         this.setFocusPorDefecto();
192         this.vistaSPRoot.setVisible(true);
193         this.vistaSPRoot.jButton4.addActionListener(new java.awt.event.ActionListener() {
194             public void actionPerformed(java.awt.event.ActionEvent evt) {
195                 jButton4ActionPerformed(evt);
196             }
197         });
198         this.vistaSPRoot.jButton2.addActionListener(new java.awt.event.ActionListener() {
199             public void actionPerformed(java.awt.event.ActionEvent evt) {
200                 jButton2ActionPerformed(evt);
201             }
202         });
203         this.vistaSPRoot.jButton1.addActionListener(new java.awt.event.ActionListener() {
204             public void actionPerformed(java.awt.event.ActionEvent evt) {
205                 jButton1ActionPerformed(evt);
206             }
207         });
208         this.vistaSPRoot.jButton7.addActionListener(new java.awt.event.ActionListener() {
209             public void actionPerformed(java.awt.event.ActionEvent evt) {
210                 jButton7ActionPerformed(evt);
211             }
212         });
213         this.vistaSPRoot.jButton8.addActionListener(new java.awt.event.ActionListener() {
214             public void actionPerformed(java.awt.event.ActionEvent evt) {
215                 jButton8ActionPerformed(evt);
216             }
217         });
218         this.vistaSPRoot.jList1.addMouseListener(new java.awt.event.MouseAdapter() {
219             public void mouseClicked(java.awt.event.MouseEvent evt) {
220                 jList1MouseClicked(evt);
221             }
222         });
226         this.vistaSPRoot.jList1.addKeyListener(new java.awt.event.KeyAdapter() {
227             public void keyPressed(java.awt.event.KeyEvent evt) {
228                 jList1KeyPressed(evt);
229             }
230         });
231         this.vistaSPRoot.tipoDatoSQL.addKeyListener(new java.awt.event.KeyAdapter() {
232             public void keyPressed(java.awt.event.KeyEvent evt) {
233                 jList1KeyPressed(evt);
234             }
235         });
236         this.vistaSPRoot.jButton2.addKeyListener(new java.awt.event.KeyAdapter() {
237             public void keyPressed(java.awt.event.KeyEvent evt) {
238                 jList1KeyPressed(evt);
239             }
240         });
241         this.vistaSPRoot.jButton1.addKeyListener(new java.awt.event.KeyAdapter() {
242             public void keyPressed(java.awt.event.KeyEvent evt) {
243                 jList1KeyPressed(evt);
244             }
245         });
246         this.vistaSPRoot.jButton7.addKeyListener(new java.awt.event.KeyAdapter() {
247             public void keyPressed(java.awt.event.KeyEvent evt) {
248                 jList1KeyPressed(evt);
249             }
250         });
251         this.vistaSPRoot.jButton8.addKeyListener(new java.awt.event.KeyAdapter() {
252             public void keyPressed(java.awt.event.KeyEvent evt) {
253                 jList1KeyPressed(evt);
254             }
255         });
256         this.vistaSPRoot.jTree2.addKeyListener(new java.awt.event.KeyAdapter() {
257             public void keyPressed(java.awt.event.KeyEvent evt) {
```

```
258                 jTree2KeyPressed(evt);
259             }
260         });
261 
262         this.vistaSPRoot.jComboBox1.addActionListener(new java.awt.event.ActionListener() {
263             public void actionPerformed(java.awt.event.ActionEvent evt) {
264                 jComboBox1ActionPerformed(evt);
265             }
266         });
267         this.vistaSPRoot.jCheckBox3.addActionListener(new java.awt.event.ActionListener() {
268             public void actionPerformed(java.awt.event.ActionEvent evt) {
269                 jCheckBox3ActionPerformed(evt);
270             }
271         });
272         this.vistaSPRoot.jComboBox2.addActionListener(new java.awt.event.ActionListener() {
273             public void actionPerformed(java.awt.event.ActionEvent evt) {
274                 jComboBox2ActionPerformed(evt);
275             }
276         });
277         this.vistaSPRoot.jComboBox6.addActionListener(new java.awt.event.ActionListener() {
278             public void actionPerformed(java.awt.event.ActionEvent evt) {
279                 jComboBox6ActionPerformed(evt);
280             }
281         });
282         this.vistaSPRoot.jButton5.addActionListener(new java.awt.event.ActionListener() {
283             public void actionPerformed(java.awt.event.ActionEvent evt) {
284                 jButton5ActionPerformed(evt);
285             }
286         });
287         this.vistaSPRoot.jButton3.addActionListener(new java.awt.event.ActionListener() {
288             public void actionPerformed(java.awt.event.ActionEvent evt) {
289                 jButton3ActionPerformed(evt);
290             }
291         });
292         this.vistaSPRoot.jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
293             public void actionPerformed(java.awt.event.ActionEvent evt) {
294                 jMenuItem1ActionPerformed(evt);
295             }
296         });
297         this.vistaSPRoot.jButton9.addActionListener(new java.awt.event.ActionListener() {
298             public void actionPerformed(java.awt.event.ActionEvent evt) {
299                 jButton9ActionPerformed(evt);
300             }
301         });
302         this.vistaSPRoot.jTree2.addTreeSelectionListener(new javax.swing.event.TreeSelectionListener() {
303             public void valueChanged(javax.swing.event.TreeSelectionEvent evt) {
304                 jTree2ValueChanged(evt);
305             }
306         });
307         this.vistaSPRoot.jButton6.addActionListener(new java.awt.event.ActionListener() {
308             public void actionPerformed(java.awt.event.ActionEvent evt) {
309                 jButton6ActionPerformed(evt);
310             }
311         });
312         this.vistaSPRoot.jButton10.addActionListener(new java.awt.event.ActionListener() {
313             public void actionPerformed(java.awt.event.ActionEvent evt) {
314                 jButton10ActionPerformed(evt);
315             }
316         });
317         this.vistaSPRoot.jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
318             public void actionPerformed(java.awt.event.ActionEvent evt) {
319                 jMenuItem3ActionPerformed(evt);
320             }
321         });
322         this.vistaSPRoot.jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
323             public void actionPerformed(java.awt.event.ActionEvent evt) {
324                 jMenuItem2ActionPerformed(evt);
325             }
326         });
327     }
```

Gráfico de llamadas para esta función:



B.11.3.16. void Control.controlProcedimiento.jButton10ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita crear un nuevo procedimiento al nivel actual de anidamiento.

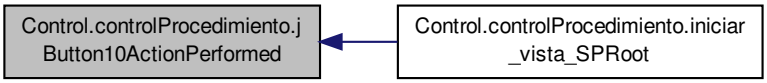
Parámetros

evt	Evento

Definición en la línea 1270 del archivo controlProcedimiento.java.

```
1270                                                                    {
1271         // TODO add your handling code here:
1272         this.modeloSProot = new ProcedimientoAlmacenado();
1273         this.limpiarCampos();
1274         this.SPNodo = null;
1275         this.vistaSPRoot.jTree2.removeSelectionPath(this.vistaSPRoot.jTree2.getSelectionPath());
1276     }
```

Gráfico de llamadas a esta función:



B.11.3.17. void Control.controlProcedimiento.jButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita editar un parámetro de un procedimiento almacenado.

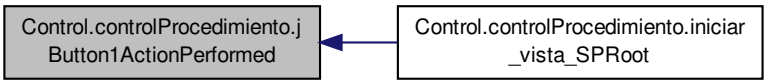
Parámetros

evt	Evento

Definición en la línea 705 del archivo controlProcedimiento.java.

```
705                                                                 {
706         this.editarParametroVistaModelo();
707         this.setLastbuttonAction(this.vistaSPRoot.jButton1);
708     }
```

Gráfico de llamadas a esta función:



B.11.3.18. void Control.controlProcedimiento.jButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita agregar un parámetro de un procedimiento almacenado.

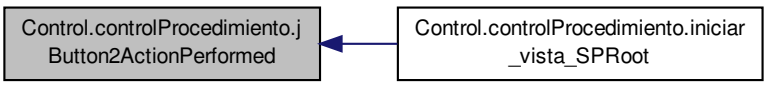
Parámetros

evt	Evento

Definición en la línea 695 del archivo controlProcedimiento.java.

```
695                                                                     {
696         this.agregarParametroVista();
697     }
```

Gráfico de llamadas a esta función:



B.11.3.19. void Control.controlProcedimiento.jButton3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita generar código SQL asociado al procedimiento actual

Parámetros

evt	Evento

Definición en la línea 951 del archivo controlProcedimiento.java.

```
951                                                                     {
952         // TODO add your handling code here:
953         if (this.vistaSPRoot.jTextField1.getText().isEmpty()) {
954             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
    (*campo obligatorio)");
955             this.vistaSPRoot.jTextField1.requestFocus();
956         } else if (this.vistaSPRoot.jTextArea2.getText().isEmpty()) {
957             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
     cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
958         } else {
959             this.modeloSProot.setUsuario(new Usuario());
960             this.modeloSProot.setNombre(this.vistaSPRoot.jTextField1.getText());
961             this.modeloSProot.getCaracteristicas().obtenerCaracteristica(3).setComment(
962                     this.vistaSPRoot.jTextArea3.getText()
963             );
964             this.modeloSProot.setCuerpoRutina(this.vistaSPRoot.jTextArea2.getText());
965             if (this.SPNodo == null) {
966                 EditorCodigoGenerado vistaPGen = new EditorCodigoGenerado(new javax.swing.JFrame(), true);
967                 controlCodigoProcedimientoGenerado controlSPCodGen = new controlCodigoProcedimientoGenerado
    (
968                         this.getConexionBDSQL(),
969                         this.modeloSProot,
970                         vistaPGen
971                 );
972                 controlSPCodGen.iniciar_vista();
973             } else {
974                 if (this.SPNodo.getChildCount() > 0) {
975                     if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, "El procedimiento
     almacenado invoca a otro(s) procedimiento(s)\n deseas tambien generar su(s) codigo(s)", "Invocaciones presentes
    ", JOptionPane.YES_NO_OPTION)) {
976                         String SQL = "DELIMITER //\n";
977                         EditorCodigoAnidadoGenerado VistaProcedimientoAnidadoGenerado = new
    EditorCodigoAnidadoGenerado(new javax.swing.JFrame(), true);
```

```
978                         controlCodigoAnidadoGenerado controlCodProcAniGen = new
    controlCodigoAnidadoGenerado(
979                                 this.getConexionBDSQL(),
980                                 SQL + this.
    generadorCodigoSQLProcedimientosAnidados(false),
981                                 SQL + this.
    generadorCodigoSQLProcedimientosAnidados(true),
982                                 VistaProcedimientoAnidadoGenerado
983                         );
984                         controlCodProcAniGen.iniciar_vista();
985
986                     } else {
987                         EditorCodigoGenerado vistaPGen = new EditorCodigoGenerado(new javax.swing.JFrame(),
     true);
988                         controlCodigoProcedimientoGenerado controlSPCodGen = new
    controlCodigoProcedimientoGenerado(
989                                 this.getConexionBDSQL(),
990                                 this.modeloSProot,
991                                 vistaPGen
992                         );
993                         controlSPCodGen.iniciar_vista();
994                     }
995
996                 } else {
997                     EditorCodigoGenerado vistaPGen = new EditorCodigoGenerado(new javax.swing.JFrame(),
    true);
998                     controlCodigoProcedimientoGenerado controlSPCodGen = new
    controlCodigoProcedimientoGenerado(
999                             this.getConexionBDSQL(),
1000                             this.modeloSProot,
1001                             vistaPGen
1002                     );
1003                     controlSPCodGen.iniciar_vista();
1004                 }
1005             }
1006         }
1007     }
```

Gráfico de llamadas para esta función:

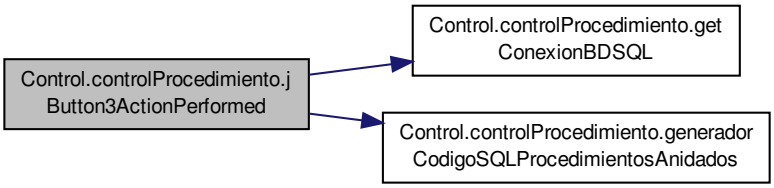
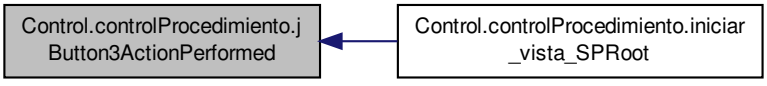


Gráfico de llamadas a esta función:



B.11.3.20. void Control.controlProcedimiento.jButton4ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita una vista previa del código SQL asociado al procedimiento actual.

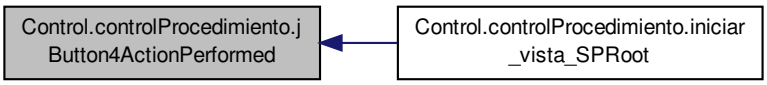
Parámetros

evt	Evento

Definición en la línea 671 del archivo controlProcedimiento.java.

```
671                                                                     {
672         if (this.vistaSPRoot.jTextField1.getText().isEmpty()) {
673             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
      (*campo obligatorio)");
674             this.vistaSPRoot.jTextField1.requestFocus();
675         } else if (this.vistaSPRoot.jTextArea2.getText().isEmpty()) {
676             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
       cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
677 
678         } else {
679             this.modeloSProot.setUsuario(new Usuario());
680             this.modeloSProot.setNombre(this.vistaSPRoot.jTextField1.getText());
681             this.modeloSProot.getCaracteristicas().obtenerCaracteristica(3).setComment(
682                     this.vistaSPRoot.jTextArea3.getText()
683             );
684             this.modeloSProot.setCuerpoRutina(this.vistaSPRoot.jTextArea2.getText());
685             JOptionPane.showMessageDialog(null, this.modeloSProot.generarProcedimientoSQL(false));
686         }
687     }
```

Gráfico de llamadas a esta función:



B.11.3.21. void Control.controlProcedimiento.jButton5ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer conexión con a una base de datos.

Parámetros

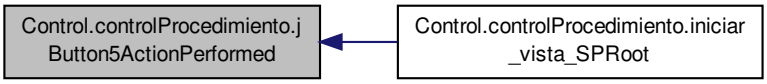
evt	Evento

Definición en la línea 780 del archivo controlProcedimiento.java.

```
780                                                                     {
781         if (this.vistaSPRoot.jButton5.getText().compareTo("Conectar a BD") == 0) {
782             conexionManejadorBD conexionMySQL = new conexionManejadorBD();
783             VentanaConexion ventanaconexionMySQL = new VentanaConexion(new javax.swing.JFrame(), true);
784             controlConexion control = new controlConexion(conexionMySQL, ventanaconexionMySQL);
785             control.iniciar_vista();
786             this.controlMVCBD = control;
787             if (this.controlMVCBD != null) {
788                 this.conexionBDSQL = this.controlMVCBD.getConexion();
789                 if (this.conexionBDSQL != null) {
790                     this.vistaSPRoot.jButton5.setText("Desconectar a BD");
791                 } else {
792                     this.vistaSPRoot.jButton5.setText("Conectar a BD");
793                 }
794             }
795         } else if (this.controlMVCBD != null && this.conexionBDSQL != null &&
      this.vistaSPRoot.jButton5.getText().compareTo("Desconectar a BD") == 0) {
796             this.vistaSPRoot.jButton5.setText("Conectar a BD");
```

```
797             this.controlMVCBD = null;
798             this.conexionBDSQL = null;
799             JOptionPane.showMessageDialog(null, "Conexion Terminada");
800         }
801         this.MVETBD();
802     }
```

Gráfico de llamadas a esta función:



B.11.3.22. void Control.controlProcedimiento.jButton6ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita crear un nuevo procedimiento a efecto de la invocacion del mismo en el cuerpo de la rutina del procedimiento actual.

Parámetros

evt	Evento

Definición en la línea 1241 del archivo controlProcedimiento.java.

```
1241                                                                    {
1242         // TODO add your handling code here:
1243         if (this.vistaSPRoot.jTextArea2.getSelectedText() != null) {
1244             ProcedimientoAlmacenado modeloNuevoProcedimiento = new ProcedimientoAlmacenado();
1245             if (this.SPNodo == null) {
1246                 this.SPNodo = new DefaultMutableTreeNode();
1247             }
1248             nuevoProcedimiento vistaNuevoProcedimiento = new nuevoProcedimiento(new javax.swing.JFrame(),
     true);
1249             controlNuevoProcedimiento ControlNP = new controlNuevoProcedimiento(
1250                     this.getConexionBDSQL(),
1251                     modeloNuevoProcedimiento,
1252                     vistaNuevoProcedimiento,
1253                     this.modeloEsquemaBD,
1254                     this.vistaSPRoot.jTextArea2.getSelectedText(),
1255                     this.SPNodo
1256             );
1257             ControlNP.iniciar_Vista_NuevoProcedimiento();
1258 
1259         } else {
1260             JOptionPane.showMessageDialog(null, "Debe seleccionar unicamente el nombre del procedimiento
      que desea crear");
1261         }
1262     }
```

Gráfico de llamadas para esta función:

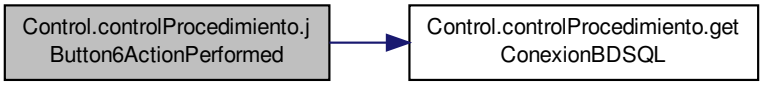
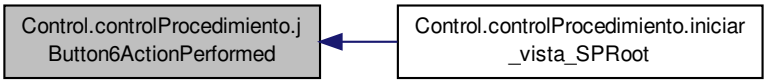


Gráfico de llamadas a esta función:



B.11.3.23. void Control.controlProcedimiento.jButton7ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita borrar un parámetro de un procedimiento almacenado.

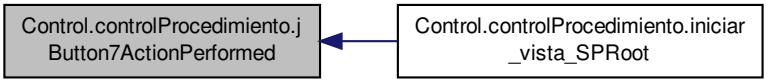
Parámetros

evt	Evento

Definición en la línea 716 del archivo controlProcedimiento.java.

```
716                                                                 {
717         this.setLastbuttonAction(this.vistaSPRoot.jButton7);
718         this.borrarParametroVistaModelo();
719     }
```

Gráfico de llamadas a esta función:



B.11.3.24. void Control.controlProcedimiento.jButton8ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita guardar los cambios en un parámetro de un procedimiento almacenado.

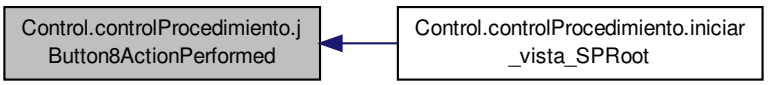
Parámetros

evt	Evento

Definición en la línea 728 del archivo controlProcedimiento.java.

```
728                                                                    {
729         this.guardarCambiosParametroVistaModelo();
730     }
```

Gráfico de llamadas a esta función:



B.11.3.25. void Control.controlProcedimiento.jButton9ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita guardar el procedimeinto actual

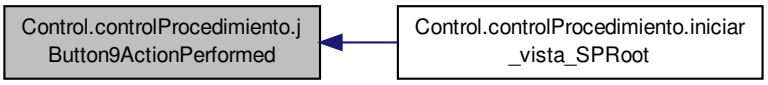
Parámetros

evt	Evento

Definición en la línea 1188 del archivo controlProcedimiento.java.

```
1188                                                                    {
1189         // TODO add your handling code here:
1190         this.guardarSPenModeloyVisaualizarlo();
1191     }
```

Gráfico de llamadas a esta función:



B.11.3.26. void Control.controlProcedimiento.jCheckBox3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer si un procedimiento es determinista.

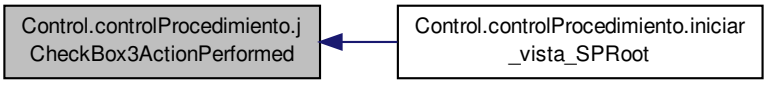
Parámetros

evt	Evento

Definición en la línea 810 del archivo controlProcedimiento.java.

```
810                                                                     {
811         if (this.vistaSPRoot.jCheckBox3.getModel().isSelected()) {
812             this.modeloSProot.getCaracteristicas().obtenerCaracteristica(0).setValue("DETERMINISTIC");
813         } else {
814             this.modeloSProot.getCaracteristicas().obtenerCaracteristica(0).setValue("NOT DETERMINISTIC");
815         }
816     }
```

Gráfico de llamadas a esta función:



B.11.3.27. void Control.controlProcedimiento.jComboBox1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer la cuenta de MySQL para ser usada cuando se verifica los privilegios de acceso en el tiempo de ejecución de un procedimiento almacenado.

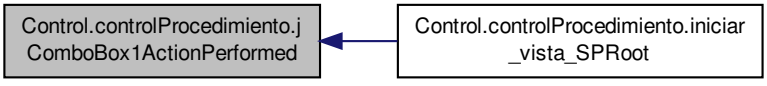
Parámetros

evt	Evento

Definición en la línea 740 del archivo controlProcedimiento.java.

```
740                                                                     {
741         if (this.vistaSPRoot.jComboBox1.getSelectedIndex() > 0) {
742             JOptionPane.showMessageDialog(null, "estamos trabajando en esta area.");
743         } else {
744             this.modeloSProot.getUsuario().resetDefaultUser();
745         }
746     }
```

Gráfico de llamadas a esta función:



B.11.3.28. void Control.controlProcedimiento.jComboBox2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta, al evento generado por un componente de la vista que solicita establecer el contexto de seguridad en un procedimiento almacenado.

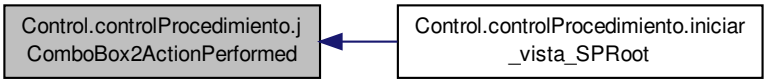
Parámetros

evt	Evento

Definición en la línea 755 del archivo controlProcedimiento.java.

```
755                                                                         {
756         this.modeloSProot.getCaracteristicas().obtenerCaracteristica(1).setComment(
757                 this.vistaSPRoot.jComboBox2.getSelectedItem().toString()
758         );
759     }
```

Gráfico de llamadas a esta función:



B.11.3.29. void Control.controlProcedimiento.jComboBox6ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta, al evento generado por un componente de la vista que solicita establecer la naturaleza de los datos usados por un procedimiento almacenado.

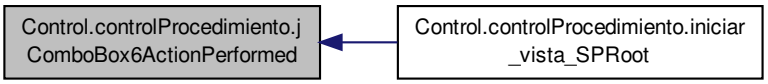
Parámetros

evt	Evento

Definición en la línea 768 del archivo controlProcedimiento.java.

```
768                                                                         {
769         this.modeloSProot.getCaracteristicas().obtenerCaracteristica(2).setValue(
770                 this.vistaSPRoot.jComboBox6.getSelectedItem().toString()
771         );
772     }
```

Gráfico de llamadas a esta función:



B.11.3.30. void Control.controlProcedimiento.jList1KeyPressed (java.awt.event.KeyEvent *evt*) [private]

Método de respuesta al evento generado por presionar una tecla asociada a un componente de la vista

Parámetros

evt	Evento

Definición en la línea 846 del archivo controlProcedimiento.java.

```
846                                                         {
847         if (evt.getComponent().equals(this.vistaSPRoot.tipoDatoSQL)) {
848             if (evt.getKeyCode() == 10) {
849                 if (this.vistaSPRoot.jButton8.isEnabled() == false) {
850                     this.agregarParametroVista();
851                 } else {
852                     this.guardarCambiosParametroVistaModelo();
853                 }
854             }
855         } else if (evt.getComponent().equals(this.vistaSPRoot.jList1)) {
856             if (evt.getKeyCode() == 127) {
857                 this.borrarParametroVistaModelo();
858             } else if (evt.getKeyCode() == 10) {
859                 //tratar de ubicar el ultimo evento o evento anterior
860                 if (this.getlasButtonAction() != null) {
861                     if (this.getlasButtonAction().equals(this.vistaSPRoot.jButton1)) {
862                         this.editarParametroVistaModelo();
863                         this.setLastbuttonAction(null);
864                     } else if (this.getlasButtonAction().equals(this.vistaSPRoot.jButton7
    )) {
865                         this.setLastbuttonAction(null);
866                         this.borrarParametroVistaModelo();
867                     }
868                 } else {
869                     this.editarParametroVistaModelo();
870                 }
871             }
872         } else if (evt.getComponent().equals(this.vistaSPRoot.jButton2)) {
873             if (evt.getKeyCode() == 10) {
874                 this.agregarParametroVista();
875             }
876         } else if (evt.getComponent().equals(this.vistaSPRoot.jButton1)) {
877             if (evt.getKeyCode() == 10) {
878                 this.editarParametroVistaModelo();
879                 this.setLastbuttonAction(this.vistaSPRoot.jButton1);
880             }
881         } else if (evt.getComponent().equals(this.vistaSPRoot.jButton7)) {
882             if (evt.getKeyCode() == 10) {
883                 this.borrarParametroVistaModelo();
884                 this.setLastbuttonAction(this.vistaSPRoot.jButton7);
885             }
886         } else if (evt.getComponent().equals(this.vistaSPRoot.jButton8)) {
887             if (evt.getKeyCode() == 10) {
888                 this.guardarCambiosParametroVistaModelo();
889             }
890         }
891     }
```

Gráfico de llamadas para esta función:

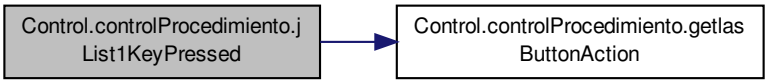
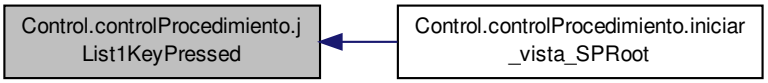


Gráfico de llamadas a esta función:



B.11.3.31. void Control.controlProcedimiento.jList1MouseClicked (java.awt.event.MouseEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita seleccionar un parámetro de un procedimiento almacenado en la vista

Parámetros

evt	Evento

Definición en la línea 825 del archivo controlProcedimiento.java.

```
825                                                                 {
826         if (this.getlasButtonAction() != null) {
827             if (this.getlasButtonAction().equals(this.
    vistaSPRoot.jButton1)) {
828                 this.editarParametroVistaModelo();
829                 this.setLastbuttonAction(null);
830             } else if (this.getlasButtonAction().equals(this.vistaSPRoot.jButton7)) {
831                 this.setLastbuttonAction(null);
832                 this.borrarParametroVistaModelo();
833             }
834         }
835         if (evt.getClickCount() == 2) {
836             this.editarParametroVistaModelo();
837         }
838     }
```

Gráfico de llamadas para esta función:

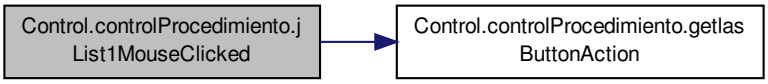
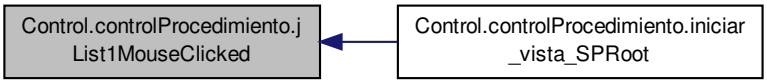


Gráfico de llamadas a esta función:



B.11.3.32. void Control.controlProcedimiento.jMenuItem1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita generar código SQL para un procedimiento que implemente alguna función básica en bases de datos para una tabla seleccionada en el árbol

Parámetros

evt	Evento

Definición en la línea 1036 del archivo controlProcedimiento.java.

```
1036                                                                        {
1037         if (!this.vistaSPRoot.jTree1.isSelectionEmpty()) {
1038             if (this.getConexionBDSQL() != null) {
1039                 if (this.esTabla(this.vistaSPRoot.jTree1.getModel(),
     this.vistaSPRoot.jTree1.getLastSelectedPathComponent())) {
1040                     ProcedimientoAlmacenado modeloPCRUD = new ProcedimientoAlmacenado();
1041                     EditorProcedimientoCRUD vistaPCRUD = new EditorProcedimientoCRUD(new javax.swing.JFrame
     (), true);
1042                     controlProcedimientoCRUD controlSPCRUD = new controlProcedimientoCRUD(
1043                             this.getConexionBDSQL(),
1044                             modeloPCRUD,
1045                             vistaPCRUD,
1046                             this.modeloEsquemaBD,
1047                             this.vistaSPRoot.jTree1.getLastSelectedPathComponent()
     .toString()
1048                     );
1049                     controlSPCRUD.iniciar_vista_SPCRUD();
1050                 } else {
1051                     JOptionPane.showMessageDialog(null, "El elemento que selecciono \n" +
     this.vistaSPRoot.jTree1.getLastSelectedPathComponent() + "\nNo es una tabla");
1052                 }
1053                 this.vistaSPRoot.jTree1.setSelectionPath(null);
1054             } else {
1055                 JOptionPane.showMessageDialog(null, "Debe establecer una conexion a una BD");
1056                 this.vistaSPRoot.jButton5.requestFocus();
1057             }
1058         } else {
1059             JOptionPane.showMessageDialog(null, "Debe seleccionar algun elemento");
1060         }
1061     }
```

Gráfico de llamadas para esta función:

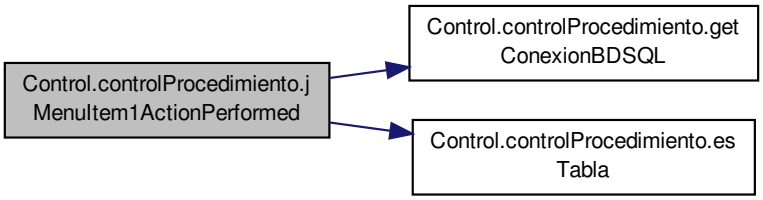
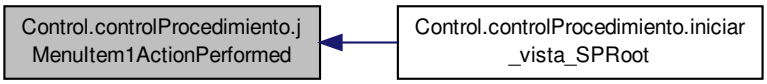


Gráfico de llamadas a esta función:



B.11.3.33. void Control.controlProcedimiento.jMenuItem2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita abrir un proyecto existente.

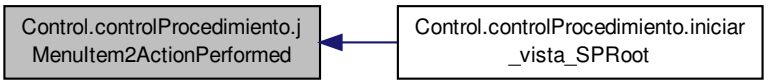
Parámetros

evt	Evento

Definición en la línea 1302 del archivo controlProcedimiento.java.

```
1302                                                                    {
1303         // TODO add your handling code here:
1304         JFileChooser jF1 = new JFileChooser();
1305         String ruta = "";
1306         try {
1307             if (jF1.showOpenDialog(null) == jF1.getApproveButtonMnemonic()) {
1308                 ruta = jF1.getSelectedFile().getAbsolutePath();
1309                 FileInputStream fs = new FileInputStream(ruta);
1310                 ObjectInputStream os = new ObjectInputStream(fs);
1311                 this.SPCreados = (DefaultMutableTreeNode) os.readObject();
1312                 os.close();
1313                 this.vistaSPRoot.jTree2.setModel(
1314                         new DefaultTreeModel(this.SPCreados)
1315                 );
1316             }
1317         } catch (Exception e) {
1318             e.printStackTrace();
1319         }
1320     }
```

Gráfico de llamadas a esta función:



B.11.3.34. void Control.controlProcedimiento.jMenuItem3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita guardar el proyecto actual.

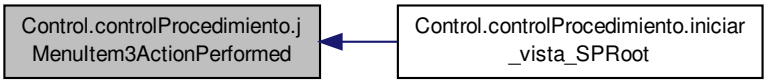
Parámetros

evt	Evento

Definición en la línea 1328 del archivo controlProcedimiento.java.

```
1328                                                                                {
1329         // TODO add your handling code here:
1330         if (this.SPCreados != null) {
1331             JFileChooser jF1 = new JFileChooser();
1332             String ruta = "";
1333             try {
1334                 if (jF1.showSaveDialog(null) == jF1.getApproveButtonMnemonic()) {
1335                     ruta = jF1.getSelectedFile().getAbsolutePath();
1336                 }
1337                 if (new File(ruta).exists()) {
1338                     if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, "El nombre del
      proyecto existe\n desea reemplazarlo", "Guardar Proyecto", JOptionPane.YES_NO_OPTION)) {
1339                         FileOutputStream fs = new FileOutputStream(ruta);
1340                         ObjectOutputStream os = new ObjectOutputStream(fs);
1341                         os.writeObject(this.SPCreados);
1342                         os.close();
1343                     }
1344                 } else {
1345                     FileOutputStream fs = new FileOutputStream(ruta);
1346                     ObjectOutputStream os = new ObjectOutputStream(fs);
1347                     os.writeObject(this.SPCreados);
1348                     os.close();
1349                 }
1350             } catch (Exception e) {
1351                 e.printStackTrace();
1352             }
1353         } else {
1354             JOptionPane.showMessageDialog(null, "Debe almenos crear y guardar un procedimiento");
1355         }
1356     }
```

Gráfico de llamadas a esta función:



B.11.3.35. void Control.controlProcedimiento.jTree2KeyPressed (java.awt.event.KeyEvent *evt*) [private]

Método de respuesta al evento generado por presionar la tecla "delete"

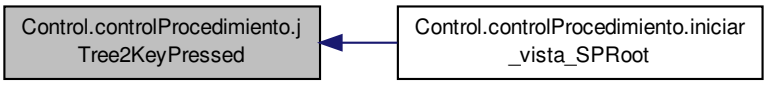
Parámetros

evt	Evento

Definición en la línea 1283 del archivo controlProcedimiento.java.

```
1283                                                        {
1284         // TODO add your handling code here:
1285         if (evt.getKeyCode() == 127) {
1286             JOptionPane.showMessageDialog(null, "Vamos a eliminar " +
     this.vistaSPRoot.jTree2.getLeadSelectionRow());
1287             if (!this.vistaSPRoot.jTree2.isSelectionEmpty()) {
1288                 DefaultMutableTreeNode tmpDMTN = (DefaultMutableTreeNode) this.
     vistaSPRoot.jTree2.getLastSelectedPathComponent();
1289                 DefaultMutableTreeNode tmpPadre = (DefaultMutableTreeNode) tmpDMTN.getParent();
1290                 tmpPadre.remove(tmpDMTN);
1291                 this.vistaSPRoot.jTree2.setModel(new DefaultTreeModel(this.
     SPCreados));
1292             }
1293         }
1294     }
```

Gráfico de llamadas a esta función:



B.11.3.36. void Control.controlProcedimiento.jTree2ValueChanged (javax.swing.event.TreeSelectionEvent *evt*) [private]

Método que controla la selección de procedimientos en el árbol. Mantiene el control entre el modelo y la vista

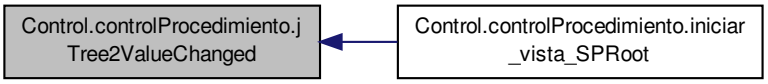
Parámetros

evt	Evento

Definición en la línea 1225 del archivo controlProcedimiento.java.

```
1225                                                                    {
1226         // TODO add your handling code here:
1227         if (!this.vistaSPRoot.jTree2.isSelectionEmpty()) {
1228             this.SPNodo = (DefaultMutableTreeNode) this.vistaSPRoot.
     jTree2.getLastSelectedPathComponent();
1229             this.modeloSProot = (ProcedimientoAlmacenado) this.SPNodo.getUserObject();
1230             this.pordefecto();
1231         }
1232     }
```

Gráfico de llamadas a esta función:



B.11.3.37. void Control.controlProcedimiento.limpiarCampos ( )

Método que limpia la vista del editor de un nuevo procedimiento almacenado, es decir, establece el estado por defecto de la vista

Definición en la línea 1078 del archivo controlProcedimiento.java.

```
1078                                {
1079         this.vistaSPRoot.jTextField1.setText("");
1080         this.limpiarCamposParametros();
1081         this.limpiarCamposCaracteristicas();
1082         this.vistaSPRoot.jTextArea2.setText("# agrega todo tu codigo de control aquí: cualquier tipo de
       datos MySQL valido\n");
1083     }
```

B.11.3.38. void Control.controlProcedimiento.limpiarCamposCaracteristicas ( )

Método que establece el estado por defecto de las características de un procedimiento almacenado tanto en la vista como en el modelo

Definición en la línea 1067 del archivo controlProcedimiento.java.

```
1067                                              {
1068         this.vistaSPRoot.jCheckBox3.setSelected(false);
1069         this.vistaSPRoot.jComboBox2.setSelectedIndex(0);
1070         this.vistaSPRoot.jComboBox6.setSelectedIndex(0);
1071         this.vistaSPRoot.jTextArea3.setText("Sin comentarios");
1072     }
```

B.11.3.39. void Control.controlProcedimiento.limpiarCamposParametros ( ) [private]

Método que limpia los campos asociados a los parametros de un procedimiento en la vista

Definición en la línea 657 del archivo controlProcedimiento.java.

```
657                                          {
658         this.vistaSPRoot.jTextField2.setText(null);
659         this.vistaSPRoot.tipoDatoSQL.setSelectedIndex(0);
660         this.vistaSPRoot.jComboBox4.setSelectedIndex(0);
661         this.vistaSPRoot.jComboBox4.requestFocus();
662     }
```

B.11.3.40. boolean Control.controlProcedimiento.llavePrimaria (ResultSet *rsColPK,* String *nombreColumna*)

Método que determina si una columna es llave primaria

Parámetros

rsColPK	Conjuto de resultados de una consulta
nombreColumna	Nombre de la columna

Devuelve

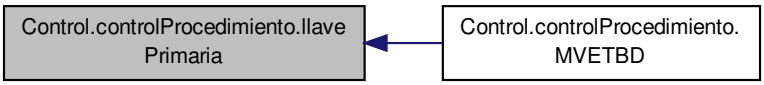
boolean

- true La columna proporcionada es una llave primaria
- false La columna proporcionada no es una llave primaria

Definición en la línea 340 del archivo controlProcedimiento.java.

```
340                                                                                 {
341         boolean tmp = false;
342         try {
343             while (rsColPK.next()) {
344                 if (rsColPK.getString(4).equals(nombreColumna)) {
345                     tmp = rsColPK.getString(4).equals(nombreColumna);
346                     break;
347                 }
348             }
349         } catch (SQLException e) {
350             e.printStackTrace();
351         }
352         return tmp;
353     }
```

Gráfico de llamadas a esta función:



B.11.3.41. void Control.controlProcedimiento.makeTypesName ( ) [private]

Método que construye el mapa de los tipos de datos en SQL

Definición en la línea 136 del archivo controlProcedimiento.java.

```
136                                 {
137         this.mapaTipoDatoSQL = new HashMap();
138         Field[] fields = java.sql.Types.class.getFields();
139         for (int i = 0; i < fields.length; i++) {
140             try {
141                 this.mapaTipoDatoSQL.put((Integer) fields[i].get(null), fields[i].getName());
142             } catch (IllegalAccessException e) {
143                 e.printStackTrace();
144             }
145         }
146     }
```

B.11.3.42. void Control.controlProcedimiento.ModuloEntradaDatos ( )

Método que establece los datos proporcionado por el usuario en la vista al modelo

Definición en la línea 1089 del archivo controlProcedimiento.java.

```
1089                                    {
1093         if (this.modeloSProot == null) {
1094             this.modeloSProot.setUsuario(new Usuario());
1095         }
1099         this.modeloSProot.setNombre(this.vistaSPRoot.jTextField1.getText());
1103         this.modeloSProot.getCaracteristicas().obtenerCaracteristica(3).setComment(
1104                 this.vistaSPRoot.jTextArea3.getText()
1105         );
1109         this.modeloSProot.setCuerpoRutina(this.vistaSPRoot.jTextArea2.getText());
1110     }
```

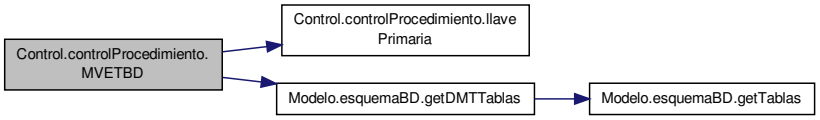
B.11.3.43. void Control.controlProcedimiento.MVETBD ( )

Método MVETBD (Modelo a Vista del Esquema de las Tablas de una Base de Datos), que visualiza el esquema de una base de datos en la vista

Definición en la línea 359 del archivo controlProcedimiento.java.

```
359                         {
364         if (this.conexionBDSQL != null) {
371             ResultSet rsTab, rsCol, rsColPK;
376             tablas tmpTab = new tablas();
377             try {
382                 DatabaseMetaData metaDatos = this.conexionBDSQL.getMetaData();
386                 rsTab = metaDatos.getTables(null, null, "%", null);
390                 while (rsTab.next()) {
395                     columnas tmpCol = new columnas();
399                     rsCol = metaDatos.getColumns(null, null, rsTab.getString(3), null);
404                     rsColPK = metaDatos.getPrimaryKeys(null, null, rsTab.getString(3));
408                     while (rsCol.next()) {
415                         tmpCol.agregarColumna(new columna(this.llavePrimaria(rsColPK, rsCol.
    getString(4)), rsCol.getString(4), this.getTypesName(rsCol.getInt(5)), rsCol.getInt(7)));
416                     }
423                     tmpTab.agregarTabla(new tabla(rsTab.getString(1), rsTab.getString(3), tmpCol));
424                 }
429                 this.modeloEsquemaBD.setCatalogo(tmpTab.getCatalogo());
434                 this.modeloEsquemaBD.setTablas(tmpTab);
439                 this.vistaSPRoot.jTree1.setModel(
440                         new DefaultTreeModel(
441                                 this.modeloEsquemaBD.getDMTTablas()
442                         )
443                 );
444             } catch (SQLException e) {
445                 e.printStackTrace();
446             }
447         } else {
451             this.vistaSPRoot.jTree1.setModel(null);
456             this.modeloEsquemaBD = new esquemaBD();
457         }
458     }
```

Gráfico de llamadas para esta función:



B.11.3.44. void Control.controlProcedimiento.pordefecto ( )

Método que establece el estado por defecto de un procedimiento almacenado del modelo hacia la vista

Definición en la línea 1212 del archivo controlProcedimiento.java.

```
1212                            {
1213         this.vistaSPRoot.jTextField1.setText(this.modeloSProot.getNombre());
1214         this.agregarParamModeloVista();
1215         this.agregarCaracModeloVista();
1216         this.vistaSPRoot.jTextArea2.setText(this.modeloSProot.getCuerpoRutina(false));
1217     }
```

B.11.3.45. void Control.controlProcedimiento.setConexionBDSQL (Connection *C*)

Método que establece la variable que contiene la conexión establecida con alguna base de datos

Parámetros

C	Conexión establecida con alguna base de datos

Definición en la línea 106 del archivo controlProcedimiento.java.

```
106                                             {
107         this.conexionBDSQL = C;
108     }
```

B.11.3.46. void Control.controlProcedimiento.setFocusPorDefecto ( )

Método que establece el orden de los elementos de la interfaz para la política de recorrimiento del focus

Definición en la línea 464 del archivo controlProcedimiento.java.

```
464                                     {
465         ArrayList orden = new ArrayList();
466         orden.add(this.vistaSPRoot.jComboBox3);
467         orden.add(this.vistaSPRoot.jTextField1);
468         orden.add(this.vistaSPRoot.jComboBox1);
469         orden.add(this.vistaSPRoot.jComboBox4);
470         orden.add(this.vistaSPRoot.jTextField2);
471         orden.add(this.vistaSPRoot.tipoDatoSQL);
472         orden.add(this.vistaSPRoot.jButton2);
473         orden.add(this.vistaSPRoot.jButton1);
474         orden.add(this.vistaSPRoot.jButton7);
475         orden.add(this.vistaSPRoot.jList1);
476         orden.add(this.vistaSPRoot.jCheckBox3);
477         orden.add(this.vistaSPRoot.jComboBox2);
478         orden.add(this.vistaSPRoot.jComboBox6);
479         orden.add(this.vistaSPRoot.jTextArea3);
480         orden.add(this.vistaSPRoot.jTextArea2);
481         orden.add(this.vistaSPRoot.jButton6);
482         orden.add(this.vistaSPRoot.jButton4);
483         orden.add(this.vistaSPRoot.jButton3);
484         orden.add(this.vistaSPRoot.jButton5);
485         MiFocusTraversalPolicy newPolicy = new MiFocusTraversalPolicy(orden);
486         vistaSPRoot.setFocusTraversalPolicy(newPolicy);
487     }
```

B.11.3.47. void Control.controlProcedimiento.setLastbuttonAction (Component *lasButtonAction*)

Método que establece el ultimo componente que provocó algún evento

Parámetros

lasButtonAction	Componente

Definición en la línea 153 del archivo controlProcedimiento.java.

```
153                                                     {
154         this.lasButtonAction = lasButtonAction;
155     }
```

B.11.4. Documentación de los datos miembro

B.11.4.1. Connection Control.controlProcedimiento.conexionBDSQL [private]

Conexión establecida con alguna base de datos

Definición en la línea 70 del archivo controlProcedimiento.java.

B.11.4.2. controlConexion Control.controlProcedimiento.controlMVCBD [private]

Control: Objeto de la clase que implementa el control entre el modelo y la vista de una conexión a una base de datos

Definición en la línea 75 del archivo controlProcedimiento.java.

B.11.4.3. Component Control.controlProcedimiento.lasButtonAction [private]

Componente de la interfaz gráfica

Definición en la línea 66 del archivo controlProcedimiento.java.

B.11.4.4. Map Control.controlProcedimiento.mapaTipoDatoSQL

Mapa de los tipos de datos en SQL

Definición en la línea 83 del archivo controlProcedimiento.java.

B.11.4.5. esquemaBD Control.controlProcedimiento.modeloEsquemaBD = new esquemaBD() [private]

Modelo: Objeto de la clase que implementa un esquema de una base de datos

Definición en la línea 79 del archivo controlProcedimiento.java.

B.11.4.6. ProcedimientoAlmacenado Control.controlProcedimiento.modeloSProot [private]

Modelo: Objeto de la clase que implementa un procedimiento almacenado

Definición en la línea 62 del archivo controlProcedimiento.java.

B.11.4.7. DefaultMutableTreeNode Control.controlProcedimiento.SPCreados = new DefaultMutableTreeNode("Procedimientos Creados") [private]

estructura donde se salvarán los procedimientos generados Árbol que contendrá los procedimientos creados por el usuario

Definición en la línea 90 del archivo controlProcedimiento.java.

B.11.4.8. DefaultMutableTreeNode Control.controlProcedimiento.SPNodo [private]

Nodo asociado al procedimiento actual

Definición en la línea 94 del archivo controlProcedimiento.java.

B.11.4.9. DefaultMutableTreeNode Control.controlProcedimiento.SPPadre [private]

Nodo padre asociado al nodo actual

Definición en la línea 98 del archivo controlProcedimiento.java.

B.11.4.10. EditorProcedimiento Control.controlProcedimiento.vistaSPRoot [private]

Vista: Editor de un procedimiento almacenado

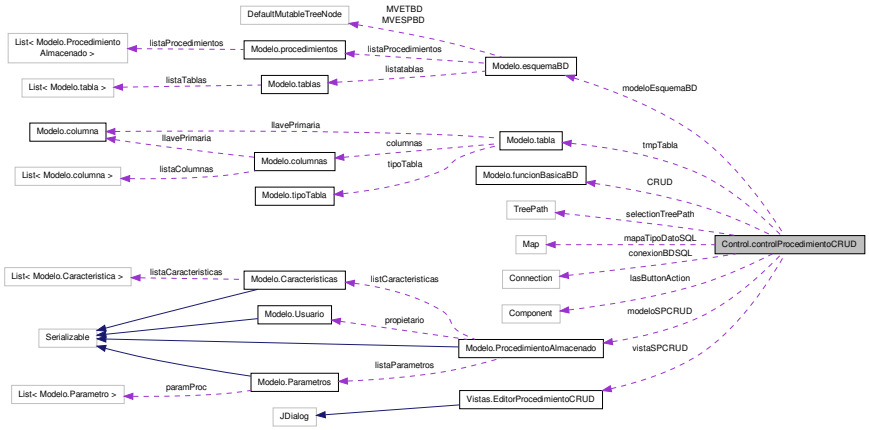
Definición en la línea 58 del archivo controlProcedimiento.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/controlProcedimiento.java

B.12. Referencia de la Clase Control.controlProcedimientoCRUD

Diagrama de colaboración para Control.controlProcedimientoCRUD:



Métodos públicos

- void setConexionBDSQL (Connection C)
- Connection getConexionBDSQL ()
- void setLastbuttonAction (Component lasButtonAction)
- Component getlasButtonAction ()

- controlProcedimientoCRUD (Connection C, ProcedimientoAlmacenado modelo, EditorProcedimientoCRUD vista, esquemaBD modeloEsquemaBD)
- controlProcedimientoCRUD (Connection C, ProcedimientoAlmacenado modelo, EditorProcedimientoCRUD vista, esquemaBD modeloEsquemaBD, String nombreTabla)
- String getTextRadioButtonSelect ()
- void getFuncionCRUD ()
- void modeloColumnaParametro (columna c)
- void modeloColumnaParametroConsulta (columna c)
- void modeloColumnaParametroCambio (columna c)
- void modeloColumnasParametros (columnas C)
- void modeloColumnasParametrosConsulta (columnas C)
- void modeloColumnasParametrosCambio (columnas C)
- void MVColumnasSP (columnas C)
- void MVColumnasSPConsulta (columnas C)
- void MVColumnasSPCambio (columnas C)
- void MVNombreSPCRUD ()
- void pordefecto ()
- void iniciar_vista_SPCRUD ()
- void setFocusPorDefecto ()
- boolean parametroValido (Parametro P, tabla T)
- void seleccionadorTablas (javax.swing.event.TreeSelectionEvent evt)

Atributos públicos

- Map mapaTipoDatoSQL

Métodos privados

- void agregarParamModeloVista ()
- void makeTypesName ()
- void inicializarTipoDatoSQLModeloVista ()
- void limpiarCamposParametros ()
- void agregarParametroVista ()
- void editarParametroVistaModelo ()
- void guardarCambiosParametroVistaModelo ()
- void borrarParametroVistaModelo ()
- boolean esTabla (javax.swing.tree.TreeModel VETBD, Object seleccion)
- boolean esBD (javax.swing.tree.TreeModel VETBD, Object seleccion)
- void jRadioButton1ActionPerformed (java.awt.event.ActionEvent evt)
- void jRadioButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void jRadioButton3ActionPerformed (java.awt.event.ActionEvent evt)
- void jRadioButton4ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton7ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton8ActionPerformed (java.awt.event.ActionEvent evt)
- void jCheckBox3ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox2ActionPerformed (java.awt.event.ActionEvent evt)
- void jComboBox6ActionPerformed (java.awt.event.ActionEvent evt)
- void jList1KeyPressed (java.awt.event.KeyEvent evt)

- void jTree1ValueChanged (javax.swing.event.TreeSelectionEvent evt)
- void jButton5ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton4ActionPerformed (java.awt.event.ActionEvent evt)
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)

Atributos privados

- EditorProcedimientoCRUD vistaSPCRUD
- ProcedimientoAlmacenado modeloSPCRUD
- tabla tmpTabla
- esquemaBD modeloEsquemaBD
- funcionBasicaBD CRUD
- Connection conexionBDSQL
- Component lasButtonAction
- int SelectionCount
- TreePath selectionTreePath

B.12.1. Descripción detallada

Clase que implementa el control entre el modelo y la vista de un procedimiento almacenado que implementa alguna función básica en base de datos: ALTA, BAJA, CAMBIO y CONSULTA. Procedimiento almacenado CRUD (de las siglas CREATE, READ, UPDATE y DELETE)

Autor

ivan

Definición en la línea 32 del archivo controlProcedimientoCRUD.java.

B.12.2. Documentación del constructor y destructor

B.12.2.1. Control.controlProcedimientoCRUD.controlProcedimientoCRUD (Connection *C,* ProcedimientoAlmacenado *modelo,* EditorProcedimientoCRUD *vista,* esquemaBD *modeloEsquemaBD*)

Constructor por defecto de la clase, establece la variable que contiene una conexión establecida con alguna base de datos, el modelo, la vista y el esquema de una base de datos

Parámetros

C	Conexión establecida con alguna base de datos
modelo	Modelo: Objeto de la clase que implementa un procedimiento almacenado
vista	Vista: Editor de un nuevo procedimiento almacenado
modeloEsquema-BD	Modelo: Objeto de la clase que implementa un esquema de una base de datos

Definición en la línea 125 del archivo controlProcedimientoCRUD.java.

```
125
                                    {
126         this.setConexionBDSQL(C);
127         this.modeloSPCRUD = modelo;
128         this.vistaSPCRUD = vista;
129         this.modeloEsquemaBD = modeloEsquemaBD;
130     }
```

B.12.2.2. Control.controlProcedimientoCRUD.controlProcedimientoCRUD (Connection *C,* ProcedimientoAlmacenado *modelo,* EditorProcedimientoCRUD *vista,* esquemaBD *modeloEsquemaBD,* String *nombreTabla*)

Sobrecarga del constructor de la clase, establece la variable que contiene una conexión establecida con alguna base de datos, el modelo, la vista, el esquema de una base de datos y el nombre de una tabla.

Parámetros

C	Conexión establecida con alguna base de datos
modelo	Modelo: Objeto de la clase que implementa un procedimiento almacenado
vista	Vista: Editor de un nuevo procedimiento almacenado
modeloEsquema-BD	Modelo: Objeto de la clase que implementa un esquema de una base de datos
nombreTabla	Nombre de una tabla

Definición en la línea 145 del archivo controlProcedimientoCRUD.java.

```
145                                                                 {
146         this.setConexionBDSQL(C);
147         this.modeloSPCRUD = modelo;
148         this.vistaSPCRUD = vista;
149         this.modeloEsquemaBD = modeloEsquemaBD;
150         this.tmpTabla = this.modeloEsquemaBD.getTablas().getTabla(nombreTabla);
151     }
```

B.12.3. Documentación de las funciones miembro

B.12.3.1. void Control.controlProcedimientoCRUD.agregarParametroVista ( ) [private]

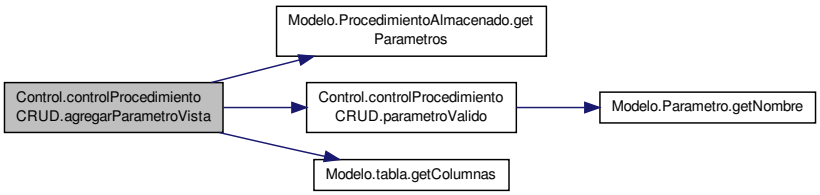
Método que implementa la función agregar parámetro de la vista al modelo

Definición en la línea 698 del archivo controlProcedimientoCRUD.java.

```
698                                         {
699         if (this.vistaSPCRUD.jTextField2.getText().isEmpty()) {
700             JOptionPane.showMessageDialog(null, "nombre/parametro vacio\n");
701             this.vistaSPCRUD.jComboBox4.requestFocus();
702         } else {
703             if (this.modeloSPCRUD.getParametros() == null) {
704                 this.modeloSPCRUD.setParametros(new Parametros());
705             }
706             if (this.parametroValido(
707                     new Parametro(
708                             (String) this.vistaSPCRUD.jComboBox4.getSelectedItem()
    .toString(),
709                             this.vistaSPCRUD.jTextField2.getText(),
710                             this.vistaSPCRUD.tipoDatoSQL.getSelectedItem().toString()
711                     ),
712                     this.tmpTabla)) {
713                 if (this.tmpTabla.getColumnas().getColumna(
    this.vistaSPCRUD.jTextField2.getText()).candidatoSP) {
714                     JOptionPane.showMessageDialog(null, "El parametro que desas agregar, ya existe\n porque
     no intentas editar el parametro");
715                 } else {
716                     this.tmpTabla.getColumnas().getColumna(this.vistaSPCRUD.
    jTextField2.getText()).setCandidatoSP(true);
717                     if (this.modeloSPCRUD.getParametros().contiene(
718                             new Parametro(
719                                     (String) this.vistaSPCRUD.
    jComboBox4.getSelectedItem().toString(),
720                                     this.vistaSPCRUD.jTextField2.getText(),
721                                     this.vistaSPCRUD.tipoDatoSQL.getSelectedItem().toString()
722                             ))) {
723                         JOptionPane.showMessageDialog(null, "El parametro que desas agregar, ya existe\n
     porque no intentas editar el parametro");
724                     } else {
725                         this.modeloSPCRUD.getParametros().agregarParametro(new
    Parametro(
726                                 (String) this.vistaSPCRUD.jComboBox4.getSelectedItem().toString(
    ),
727                                 this.vistaSPCRUD.jTextField2.getText(),
728                                 this.vistaSPCRUD.tipoDatoSQL.getSelectedItem().toString()
729                         ));
730                         this.agregarParamModeloVista();
731                         this.limpiarCamposParametros();
732                     }
```

```
733                 }
734             } else {
735                 JOptionPane.showMessageDialog(null, "El parametro que desas agregar\n"
736                         + (String) this.vistaSPCRUD.jComboBox4.getSelectedItem()
    .toString() + " "
737                         + this.vistaSPCRUD.jTextField2.getText() + " "
738                         + this.vistaSPCRUD.tipoDatoSQL.getSelectedItem().toString() + " "
739                         + "\n no es un parametro valido!"
740                         + "Recuerde que un parametro valido:\n Es alguna columna de la tabla que se
     encuentra trabajando");
741             }
742         }
743     }
```

Gráfico de llamadas para esta función:



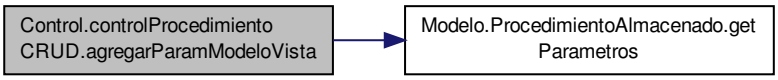
B.12.3.2. void Control.controlProcedimientoCRUD.agregarParamModeloVista ( ) [private]

Método que implementa la función agregar parámetro del modelo a la vista

Definición en la línea 394 del archivo controlProcedimientoCRUD.java.

```
394                                         {
395         Parametro item;
396         ArrayList<String> P = new ArrayList<>();
400         if (!this.modeloSPCRUD.getParametros().listaParamVacia()) {
401             for (int i = 0; i < this.modeloSPCRUD.getParametros().numElementos() - 1; i++) {
402                 item = (Parametro) this.modeloSPCRUD.
    getParametros().obtenerParametro(i);
403                 P.add(item.generarSQL());
404             }
405             item = (Parametro) this.modeloSPCRUD.
    getParametros().getParamProc().get(
406                     this.modeloSPCRUD.getParametros().numElementos() - 1);
407             P.add(item.generarSQL());
408         }
412         DefaultListModel<String> model = new DefaultListModel<>();
413         for (String s : P) {
414             model.addElement(s);
415         }
416         this.vistaSPCRUD.jList1.setModel(model);
417     }
```

Gráfico de llamadas para esta función:



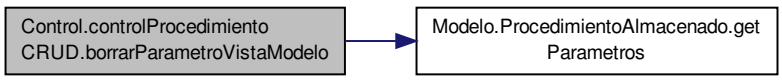
B.12.3.3. void Control.controlProcedimientoCRUD.borrarParametroVistaModelo ( ) [private]

Método que implementa la función borrar parámetro, garantiza que los cambios generados al ejecutar la función se vean reflejados en la vista y en el modelo

Definición en la línea 837 del archivo controlProcedimientoCRUD.java.

```
837                                         {
838         if (this.vistaSPCRUD.jList1.getModel().getSize() > 0) {
839             if (this.vistaSPCRUD.jList1.isSelectionEmpty()) {
840                 JOptionPane.showMessageDialog(null, "Seleleccione algun parametro");
841                 this.vistaSPCRUD.jList1.setSelectedIndex(0);
842                 if (this.vistaSPCRUD.jList1.getModel().getSize() > 1) {
843                     this.vistaSPCRUD.jList1.requestFocus();
844 
845                 } else {
846                     this.vistaSPCRUD.jButton7.requestFocus();
847                 }
848             } else {
849                 this.tmpTabla.getColumnas().getColumna(
850                         this.modeloSPCRUD.getParametros().obtenerParametro(
    this.vistaSPCRUD.jList1.getSelectedIndex()).getNombre()
851                 ).setCandidatoSP(false);
852                 this.modeloSPCRUD.getParametros().getParamProc().remove(
    this.vistaSPCRUD.jList1.getSelectedIndex());
853                 this.setLastbuttonAction(null);
854                 this.agregarParamModeloVista();
855                 this.vistaSPCRUD.jComboBox4.requestFocus();
856                 this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
    tmpTabla, CRUD));
857             }
858         } else {
859             JOptionPane.showMessageDialog(null, "Agrege almenos un Parametro");
860             this.vistaSPCRUD.jComboBox4.requestFocus();
861         }
862     }
```

Gráfico de llamadas para esta función:



B.12.3.4. void Control.controlProcedimientoCRUD.editarParametroVistaModelo ( ) [private]

Método que implementa la función editar parámetro, garantiza que los cambios en la vista se vean reflejados en el modelo

Definición en la línea 749 del archivo controlProcedimientoCRUD.java.

```
749                                         {
750         if (this.vistaSPCRUD.jList1.getModel().getSize() > 0) {
751             if (this.vistaSPCRUD.jList1.isSelectionEmpty()) {
752                 JOptionPane.showMessageDialog(null, "Seleleccione algun parametro");
754                 this.vistaSPCRUD.jList1.setSelectedIndex(0);
755                 if (this.vistaSPCRUD.jList1.getModel().getSize() > 1) {
756                     this.vistaSPCRUD.jList1.requestFocus();
757                 } else {
758                     this.vistaSPCRUD.jButton1.requestFocus();
759                 }
760             } else {
```

```
766                 this.vistaSPCRUD.jComboBox4.setSelectedItem(
767                         this.modeloSPCRUD.getParametros().obtenerParametro(this.
    vistaSPCRUD.jList1.getSelectedIndex()).getTipoES()
768                 );
769                 this.vistaSPCRUD.jTextField2.setText(
770                         this.modeloSPCRUD.getParametros().obtenerParametro(this.
    vistaSPCRUD.jList1.getSelectedIndex()).getNombre()
771                 );
772                 this.vistaSPCRUD.tipoDatoSQL.setSelectedItem(
773                         this.modeloSPCRUD.getParametros().obtenerParametro(this.
    vistaSPCRUD.jList1.getSelectedIndex()).getTipoDato()
774                 );
775                 this.vistaSPCRUD.jButton1.setEnabled(false);
776                 this.vistaSPCRUD.jButton2.setEnabled(false);
777                 this.vistaSPCRUD.jButton7.setEnabled(false);
778                 this.vistaSPCRUD.jTextField2.setEditable(false);
779                 this.vistaSPCRUD.tipoDatoSQL.setEnabled(false);
780                 this.vistaSPCRUD.jList1.setEnabled(false);
781                 this.vistaSPCRUD.jButton8.setEnabled(true);
785                 ArrayList tmpOrden = new ArrayList();
786                 tmpOrden.add(this.vistaSPCRUD.jComboBox4);
787                 tmpOrden.add(this.vistaSPCRUD.jButton8);
788                 MiFocusTraversalPolicy newTMPPolicy = new MiFocusTraversalPolicy(tmpOrden);
789                 this.vistaSPCRUD.setFocusTraversalPolicy(newTMPPolicy);
790                 this.vistaSPCRUD.jComboBox4.requestFocus();
791                 this.modeloSPCRUD.getParametros().modificarParametro(this.
    vistaSPCRUD.jList1.getSelectedIndex());
792                 this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
    tmpTabla, CRUD));
793             }
794         } else {
795             JOptionPane.showMessageDialog(null, "Agrege almenos un Parametro");
796             this.vistaSPCRUD.jComboBox4.requestFocus();
797         }
798     }
```

B.12.3.5. boolean Control.controlProcedimientoCRUD.esBD (javax.swing.tree.TreeModel *VETBD,* Object *seleccion*) [private]

Método que identifica si la selección en el árbol de tablas es la base de datos (el nodo raíz del árbol de tablas, es la base de datos).

Parámetros

VETBD	Árbol de tablas
seleccion	Objeto seleccionado en el árbol de tablas.

Devuelve

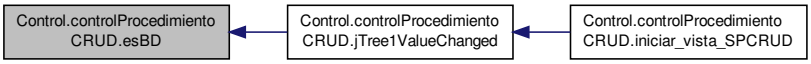
boolean

- true indica que el elemento de la seleción en el árbol de tablas, es la base de datos
- false indica que el elemento de la seleción en el árbol de tablas, no es la base de datos

Definición en la línea 898 del archivo controlProcedimientoCRUD.java.

```
898                                                                     {
899
900         return VETBD.getRoot().equals(seleccion);
901
902     }
```

Gráfico de llamadas a esta función:



B.12.3.6. boolean Control.controlProcedimientoCRUD.esTabla (javax.swing.tree.TreeModel *VETBD,* Object *seleccion*) [private]

Método que identifica si la selección en el árbol de tablas es precisamente una tabla

Parámetros

VETBD	Árbol de tablas
seleccion	Objeto seleccionado en el árbol de tablas.

Devuelve

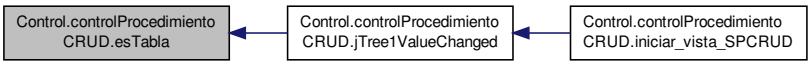
boolean

- true indica que el elemento de la seleción en el árbol de tablas, es una tabla
- false indica que el elemento de la seleción en el árbol de tablas, no es una tabla

Definición en la línea 878 del archivo controlProcedimientoCRUD.java.

```
878                                                                                 {
879
880         return (!VETBD.getRoot().equals(seleccion) && !VETBD.isLeaf(seleccion));
881
882     }
```

Gráfico de llamadas a esta función:



B.12.3.7. Connection Control.controlProcedimientoCRUD.getConexionBDSQL ( )

Método que obtiene la variable que contiene la conexión establecida con alguna base de datos

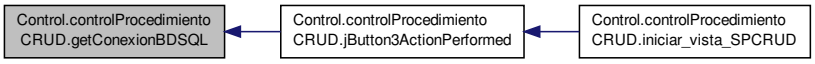
Devuelve

Connection conexión establecida con alguna base de datos

Definición en la línea 91 del archivo controlProcedimientoCRUD.java.

```
91                                          {
92          return this.conexionBDSQL;
93      }
```

Gráfico de llamadas a esta función:



B.12.3.8. void Control.controlProcedimientoCRUD.getFuncionCRUD ( )

Método que invoca a la función básica en bases de datos, seleccionada en la vista

Definición en la línea 174 del archivo controlProcedimientoCRUD.java.

```
174                                 {
175         if (this.vistaSPCRUD.jRadioButton1.isSelected()) {
176             this.jRadioButton1ActionPerformed(null);
177         } else if (this.vistaSPCRUD.jRadioButton2.isSelected()) {
178             this.jRadioButton2ActionPerformed(null);
179         } else if (this.vistaSPCRUD.jRadioButton3.isSelected()) {
180             this.jRadioButton3ActionPerformed(null);
181         } else if (this.vistaSPCRUD.jRadioButton4.isSelected()) {
182             this.jRadioButton4ActionPerformed(null);
183         }
184     }
```

B.12.3.9. Component Control.controlProcedimientoCRUD.getlasButtonAction ( )

Método que obtiene el ultimo componente que provocó algún evento

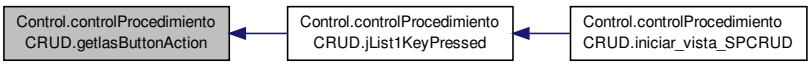
Devuelve

Component Componente

Definición en la línea 109 del archivo controlProcedimientoCRUD.java.

```
109                                         {
110         return this.lasButtonAction;
111     }
```

Gráfico de llamadas a esta función:



B.12.3.10. String Control.controlProcedimientoCRUD.getTextRadioButtonSelect ( )

Método que obtiene el nombre de la función básica en bases de datos seleccionada en la vista

Devuelve

String Nombre de la función básica en una base de datos.

Definición en la línea 159 del archivo controlProcedimientoCRUD.java.

```
159                                         {
160         return (this.vistaSPCRUD.jRadioButton1.isSelected())
161                 ? (this.vistaSPCRUD.jRadioButton1.getText())
162                 : (this.vistaSPCRUD.jRadioButton2.isSelected())
163                 ? this.vistaSPCRUD.jRadioButton2.getText()
164                 : (this.vistaSPCRUD.jRadioButton3.isSelected())
165                 ? this.vistaSPCRUD.jRadioButton3.getText()
166                 : (this.vistaSPCRUD.jRadioButton4.isSelected())
167                 ? this.vistaSPCRUD.jRadioButton4.getText() : "";
168     }
```

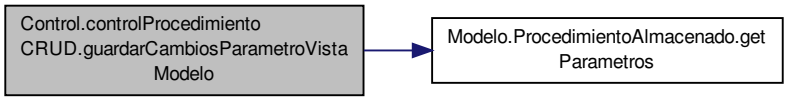
B.12.3.11. void Control.controlProcedimientoCRUD.guardarCambiosParametroVistaModelo ( ) [private]

Método que implementa la función guardar cambios en parámetro, garantiza que los cambios generados al ejecutar la función se vean reflejados en la vista y en el modelo

Definición en la línea 805 del archivo controlProcedimientoCRUD.java.

```
805                                                 {
806         if (this.modeloSPCRUD.getParametros().getModParam() >= 0) {
807             this.modeloSPCRUD.getParametros().obtenerParametro(
808                     this.modeloSPCRUD.getParametros().getModParam()
809             ).setTipoES((String) this.vistaSPCRUD.jComboBox4.getSelectedItem()
    .toString());
810             this.modeloSPCRUD.getParametros().obtenerParametro(
811                     this.modeloSPCRUD.getParametros().getModParam()
812             ).setNombre(this.vistaSPCRUD.jTextField2.getText());
813             this.modeloSPCRUD.getParametros().obtenerParametro(
814                     this.modeloSPCRUD.getParametros().getModParam()
815             ).setTipoDato(this.vistaSPCRUD.tipoDatoSQL.getSelectedItem().toString());
816             JOptionPane.showMessageDialog(null, this.modeloSPCRUD.getParametros().obtenerParametro(this.
    modeloSPCRUD.getParametros().getModParam()).generarSQL());
817             this.agregarParamModeloVista();
818             this.limpiarCamposParametros();
819             this.setFocusPorDefecto();
820             this.setLastbuttonAction(null);
821             this.vistaSPCRUD.jButton8.setEnabled(false);
822             this.vistaSPCRUD.jTextField2.setEditable(true);
823             this.vistaSPCRUD.tipoDatoSQL.setEnabled(true);
824             this.vistaSPCRUD.jButton1.setEnabled(true);
825             this.vistaSPCRUD.jButton2.setEnabled(true);
826             this.vistaSPCRUD.jButton7.setEnabled(true);
827             this.vistaSPCRUD.jList1.setEnabled(true);
828             this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
    tmpTabla, CRUD));
829         }
830     }
```

Gráfico de llamadas para esta función:



B.12.3.12. void Control.controlProcedimientoCRUD.inicializarTipoDatoSQLModeloVista ( ) [private]

Método que inicializa la vista que muestra los tipos de datos en SQL

Definición en la línea 660 del archivo controlProcedimientoCRUD.java.

```
660                                                 {
661         DefaultComboBoxModel tmpTipoDatoSQL = new DefaultComboBoxModel();
662         for (Iterator it = this.mapaTipoDatoSQL.entrySet().iterator(); it.hasNext();) {
663             Map.Entry<Integer, String> e = (Map.Entry<Integer, String>) it.next();
664             tmpTipoDatoSQL.addElement(e.getValue());
665         }
666         this.vistaSPCRUD.tipoDatoSQL.setModel(tmpTipoDatoSQL);
667     }
```

B.12.3.13. void Control.controlProcedimientoCRUD.iniciar_vista_SPCRUD ( )

Método implementa la función de inicializar la vista: editor de un procedimiento almacenado CRUD

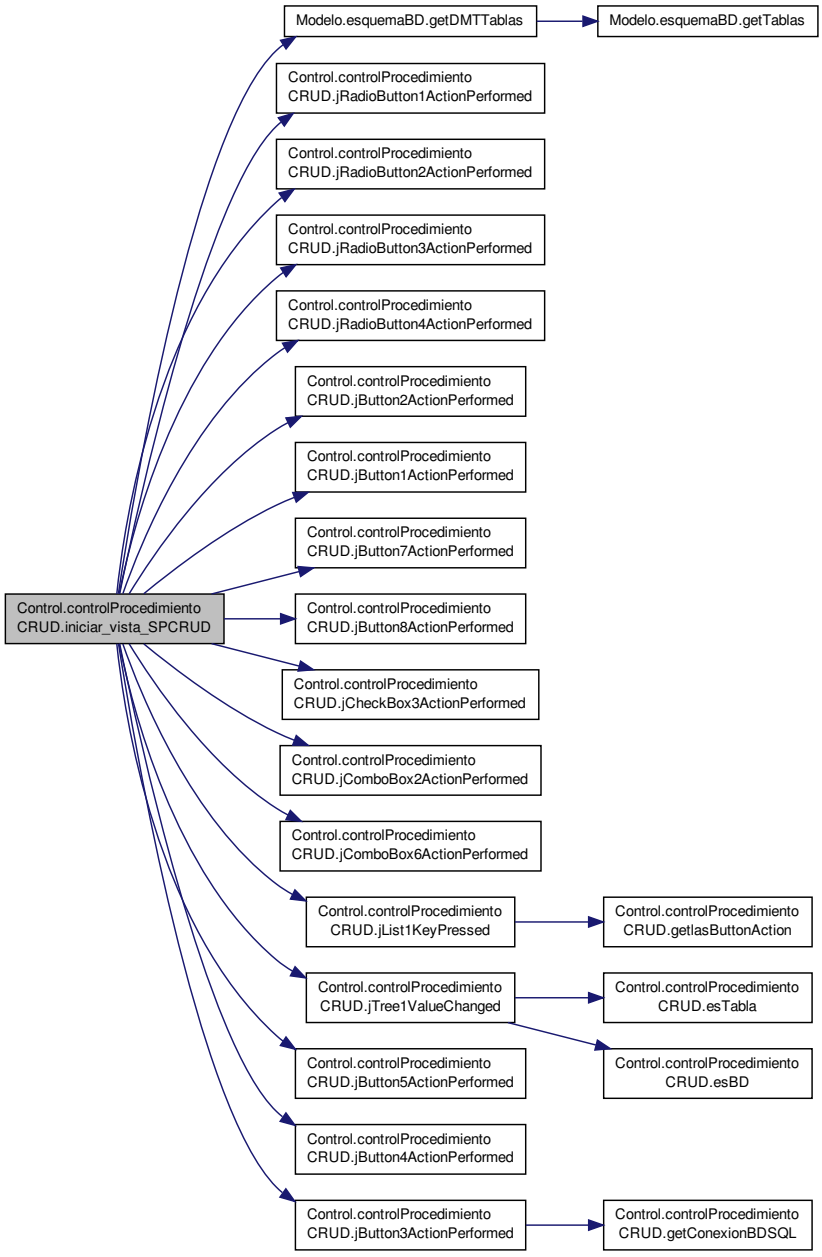
Definición en la línea 477 del archivo controlProcedimientoCRUD.java.

```
477                                     {
478         this.makeTypesName();
479         this.inicializarTipoDatoSQLModeloVista();
480         this.vistaSPCRUD.tipoDatoSQL.setEditable(true);
481         this.vistaSPCRUD.setTitle("Generador de código para un procedimiento almacenado (CRUD)");
482         this.vistaSPCRUD.setLocationRelativeTo(null);
486         this.setFocusPorDefecto();
491         this.vistaSPCRUD.jTree1.setModel(new DefaultTreeModel(
492                 this.modeloEsquemaBD.getDMTTablas()
493         )
494         );
495         this.pordefecto();
499         this.vistaSPCRUD.jRadioButton1.addActionListener(new java.awt.event.ActionListener() {
500             public void actionPerformed(java.awt.event.ActionEvent evt) {
501                 jRadioButton1ActionPerformed(evt);
502             }
503         });
504         this.vistaSPCRUD.jRadioButton2.addActionListener(new java.awt.event.ActionListener() {
505             public void actionPerformed(java.awt.event.ActionEvent evt) {
506                 jRadioButton2ActionPerformed(evt);
507             }
508         });
509         this.vistaSPCRUD.jRadioButton3.addActionListener(new java.awt.event.ActionListener() {
510             public void actionPerformed(java.awt.event.ActionEvent evt) {
511                 jRadioButton3ActionPerformed(evt);
512             }
513         });
514         this.vistaSPCRUD.jRadioButton4.addActionListener(new java.awt.event.ActionListener() {
515             public void actionPerformed(java.awt.event.ActionEvent evt) {
516                 jRadioButton4ActionPerformed(evt);
517             }
518         });
```

```
519         this.vistaSPCRUD.jButton2.addActionListener(new java.awt.event.ActionListener() {
520             public void actionPerformed(java.awt.event.ActionEvent evt) {
521                 jButton2ActionPerformed(evt);
522             }
523         });
524         this.vistaSPCRUD.jButton1.addActionListener(new java.awt.event.ActionListener() {
525             public void actionPerformed(java.awt.event.ActionEvent evt) {
526                 jButton1ActionPerformed(evt);
527             }
528         });
529         this.vistaSPCRUD.jButton7.addActionListener(new java.awt.event.ActionListener() {
530             public void actionPerformed(java.awt.event.ActionEvent evt) {
531                 jButton7ActionPerformed(evt);
532             }
533         });
534 
535         this.vistaSPCRUD.jButton8.addActionListener(new java.awt.event.ActionListener() {
536             public void actionPerformed(java.awt.event.ActionEvent evt) {
537                 jButton8ActionPerformed(evt);
538             }
539         });
540         this.vistaSPCRUD.jCheckBox3.addActionListener(new java.awt.event.ActionListener() {
541             public void actionPerformed(java.awt.event.ActionEvent evt) {
542                 jCheckBox3ActionPerformed(evt);
543             }
544         });
545         this.vistaSPCRUD.jComboBox2.addActionListener(new java.awt.event.ActionListener() {
546             public void actionPerformed(java.awt.event.ActionEvent evt) {
547                 jComboBox2ActionPerformed(evt);
548             }
549         });
550         this.vistaSPCRUD.jComboBox6.addActionListener(new java.awt.event.ActionListener() {
551             public void actionPerformed(java.awt.event.ActionEvent evt) {
552                 jComboBox6ActionPerformed(evt);
553             }
554         });
558         this.vistaSPCRUD.jList1.addKeyListener(new java.awt.event.KeyAdapter() {
559             public void keyPressed(java.awt.event.KeyEvent evt) {
560                 jList1KeyPressed(evt);
561             }
562         });
563         this.vistaSPCRUD.tipoDatoSQL.addKeyListener(new java.awt.event.KeyAdapter() {
564             public void keyPressed(java.awt.event.KeyEvent evt) {
565                 jList1KeyPressed(evt);
566             }
567         });
568         this.vistaSPCRUD.jButton2.addKeyListener(new java.awt.event.KeyAdapter() {
569             public void keyPressed(java.awt.event.KeyEvent evt) {
570                 jList1KeyPressed(evt);
571             }
572         });
573         this.vistaSPCRUD.jButton1.addKeyListener(new java.awt.event.KeyAdapter() {
574             public void keyPressed(java.awt.event.KeyEvent evt) {
575                 jList1KeyPressed(evt);
576             }
577         });
578         this.vistaSPCRUD.jButton7.addKeyListener(new java.awt.event.KeyAdapter() {
579             public void keyPressed(java.awt.event.KeyEvent evt) {
580                 jList1KeyPressed(evt);
581             }
582         });
583         this.vistaSPCRUD.jButton8.addKeyListener(new java.awt.event.KeyAdapter() {
584             public void keyPressed(java.awt.event.KeyEvent evt) {
585                 jList1KeyPressed(evt);
586             }
587         });
588         this.vistaSPCRUD.jTree1.addTreeSelectionListener(new javax.swing.event.TreeSelectionListener() {
589             public void valueChanged(javax.swing.event.TreeSelectionEvent evt) {
590                 jTree1ValueChanged(evt);
591             }
592         });
593         this.vistaSPCRUD.jButton5.addActionListener(new java.awt.event.ActionListener() {
594             public void actionPerformed(java.awt.event.ActionEvent evt) {
595                 jButton5ActionPerformed(evt);
596             }
597         });
598         this.vistaSPCRUD.jButton4.addActionListener(new java.awt.event.ActionListener() {
599             public void actionPerformed(java.awt.event.ActionEvent evt) {
600                 jButton4ActionPerformed(evt);
601             }
602         });
```

```
603         this.vistaSPCRUD.jButton3.addActionListener(new java.awt.event.ActionListener() {
604             public void actionPerformed(java.awt.event.ActionEvent evt) {
605                 jButton3ActionPerformed(evt);
606             }
607         });
608 
609         this.vistaSPCRUD.setVisible(true);
610     }
```

Gráfico de llamadas para esta función:



B.12.3.14. void Control.controlProcedimientoCRUD.jButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita editar un parámetro de un procedimiento almacenado.

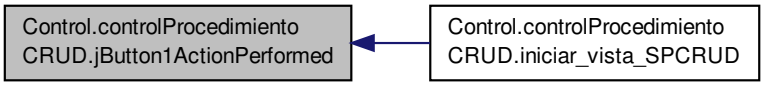
Parámetros

evt	Evento

Definición en la línea 986 del archivo controlProcedimientoCRUD.java.

```
986                                                                     {
987         this.editarParametroVistaModelo();
988         this.setLastbuttonAction(this.vistaSPCRUD.jButton1);
989     }
```

Gráfico de llamadas a esta función:



B.12.3.15. void Control.controlProcedimientoCRUD.jButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita agregar un parámetro de un procedimiento almacenado.

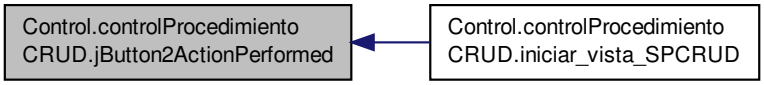
Parámetros

evt	Evento

Definición en la línea 976 del archivo controlProcedimientoCRUD.java.

```
976                                                                     {
977         this.agregarParametroVista();
978     }
```

Gráfico de llamadas a esta función:



B.12.3.16. void Control.controlProcedimientoCRUD.jButton3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita generar código SQL asociado al procedimiento CRUD actual

Parámetros

evt	Evento

Definición en la línea 1199 del archivo controlProcedimientoCRUD.java.

```
1199                                                                    {
1200         // TODO add your handling code here:
1201         if (this.vistaSPCRUD.jTextField1.getText().isEmpty()) {
1202             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
     (*campo obligatorio)");
1203             this.vistaSPCRUD.jTextField1.requestFocus();
1204         } else if (this.vistaSPCRUD.jTextArea2.getText().isEmpty()) {
1205             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
      cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
1206 
1207         } else {
1208             this.modeloSPCRUD.setUsuario(new Usuario());
1209             this.modeloSPCRUD.setNombre(this.vistaSPCRUD.jTextField1.getText());
1210             this.modeloSPCRUD.getCaracteristicas().obtenerCaracteristica(3).setComment(
1211                     this.vistaSPCRUD.jTextArea3.getText()
1212             );
1213             this.modeloSPCRUD.setCuerpoRutina(this.vistaSPCRUD.jTextArea2.getText());
1214             EditorCodigoGenerado vistaPGen = new EditorCodigoGenerado(new javax.swing.JFrame(), true);
1215             controlCodigoProcedimientoGenerado controlSPCodGen = new controlCodigoProcedimientoGenerado(
1216                     this.getConexionBDSQL(),
1217                     this.modeloSPCRUD,
1218                     vistaPGen
1219             );
1220             controlSPCodGen.iniciar_vista();
1221         }
1222     }
```

Gráfico de llamadas para esta función:

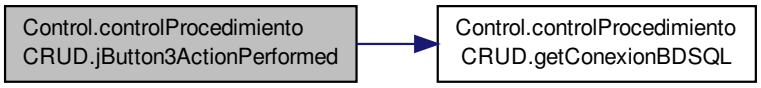
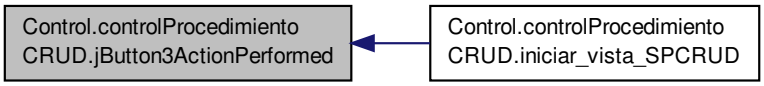


Gráfico de llamadas a esta función:



B.12.3.17. void Control.controlProcedimientoCRUD.jButton4ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita una vista previa del código SQL asociado al procedimiento actual.

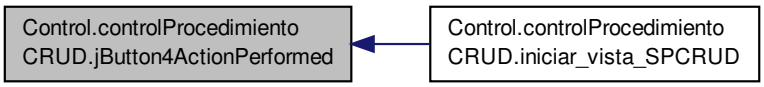
Parámetros

evt	Evento

Definición en la línea 1174 del archivo controlProcedimientoCRUD.java.

```
1174                                                                    {
1175         // TODO add your handling code here:
1176         if (this.vistaSPCRUD.jTextField1.getText().isEmpty()) {
1177             JOptionPane.showMessageDialog(null, "Favor de ingresar un nombre al Procedimiento Almacenado\n
     (*campo obligatorio)");
1178             this.vistaSPCRUD.jTextField1.requestFocus();
1179         } else if (this.vistaSPCRUD.jTextArea2.getText().isEmpty()) {
1180             JOptionPane.showMessageDialog(null, "Favor de agrega codigo de control ... \n recuerde que
      cualquier tipo de datos MySQL valido\n(*campo obligatorio)");
1181 
1182         } else {
1183             this.modeloSPCRUD.setUsuario(new Usuario());
1184             this.modeloSPCRUD.setNombre(this.vistaSPCRUD.jTextField1.getText());
1185             this.modeloSPCRUD.getCaracteristicas().obtenerCaracteristica(3).setComment(
1186                     this.vistaSPCRUD.jTextArea3.getText()
1187             );
1188             this.modeloSPCRUD.setCuerpoRutina(this.vistaSPCRUD.jTextArea2.getText());
1189             JOptionPane.showMessageDialog(null, this.modeloSPCRUD.generarProcedimientoSQL(false));
1190         }
1191     }
```

Gráfico de llamadas a esta función:



B.12.3.18. void Control.controlProcedimientoCRUD.jButton5ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita cerrar la vista del editor de un procedimiento CRUD

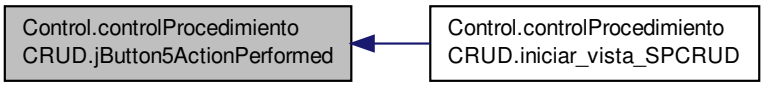
Parámetros

evt	Evento

Definición en la línea 1162 del archivo controlProcedimientoCRUD.java.

```
1162                                                                    {
1163         // TODO add your handling code here:
1164         this.vistaSPCRUD.dispose();
1165     }
```

Gráfico de llamadas a esta función:



B.12.3.19. void Control.controlProcedimientoCRUD.jButton7ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita borrar un parámetro de un procedimiento almacenado.

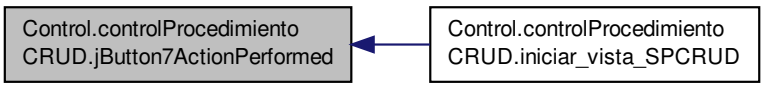
Parámetros

evt	Evento

Definición en la línea 997 del archivo controlProcedimientoCRUD.java.

```
997                                                                 {
998         this.setLastbuttonAction(this.vistaSPCRUD.jButton7);
999         this.borrarParametroVistaModelo();
1000     }
```

Gráfico de llamadas a esta función:



B.12.3.20. void Control.controlProcedimientoCRUD.jButton8ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita guardar los cambios en un parámetro de un procedimiento almacenado.

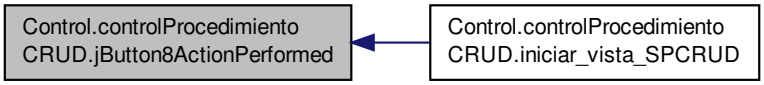
Parámetros

evt	Evento

Definición en la línea 1009 del archivo controlProcedimientoCRUD.java.

```
1009                                                                {
1010         this.guardarCambiosParametroVistaModelo();
1011     }
```

Gráfico de llamadas a esta función:



B.12.3.21. void Control.controlProcedimientoCRUD.jCheckBox3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer si un procedimiento es determinista.

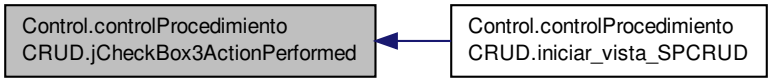
Parámetros

evt	Evento

Definición en la línea 1019 del archivo controlProcedimientoCRUD.java.

```
1019                                                                    {
1020         if (this.vistaSPCRUD.jCheckBox3.getModel().isSelected()) {
1021             this.modeloSPCRUD.getCaracteristicas().obtenerCaracteristica(0).setValue("DETERMINISTIC");
1022         } else {
1023             this.modeloSPCRUD.getCaracteristicas().obtenerCaracteristica(0).setValue("NOT DETERMINISTIC");
1024         }
1025     }
```

Gráfico de llamadas a esta función:



B.12.3.22. void Control.controlProcedimientoCRUD.jComboBox2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta, al evento generado por un componente de la vista que solicita establecer el contexto de seguridad en un procedimiento almacenado.

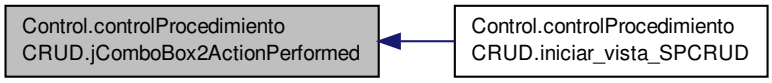
Parámetros

evt	Evento

Definición en la línea 1034 del archivo controlProcedimientoCRUD.java.

```
1034                                                                    {
1035         this.modeloSPCRUD.getCaracteristicas().obtenerCaracteristica(1).setComment(
1036                 this.vistaSPCRUD.jComboBox2.getSelectedItem().toString()
1037         );
1038     }
```

Gráfico de llamadas a esta función:



B.12.3.23. void Control.controlProcedimientoCRUD.jComboBox6ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta, al evento generado por un componente de la vista que solicita establecer la naturaleza de los datos usados por un procedimiento almacenado.

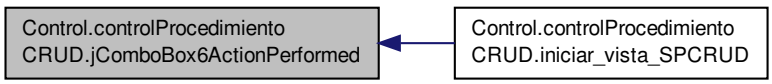
Parámetros

evt	Evento

Definición en la línea 1047 del archivo controlProcedimientoCRUD.java.

```
1047                                                                    {
1048         this.modeloSPCRUD.getCaracteristicas().obtenerCaracteristica(2).setValue(
1049                 this.vistaSPCRUD.jComboBox6.getSelectedItem().toString()
1050         );
1051
1052     }
```

Gráfico de llamadas a esta función:



B.12.3.24. void Control.controlProcedimientoCRUD.jList1KeyPressed (java.awt.event.KeyEvent *evt*) [private]

KeyPressed Método de respuesta al evento de presionar alguna tecla asociada algún componente de la vista

Parámetros

evt	Evento

Definición en la línea 1063 del archivo controlProcedimientoCRUD.java.

```
1063                                                                {
1064         if (evt.getComponent().equals(this.vistaSPCRUD.tipoDatoSQL)) {
1065             if (evt.getKeyCode() == 10) {
1066                 if (this.vistaSPCRUD.jButton8.isEnabled() == false) {
1067                     this.agregarParametroVista();
1068                 } else {
1069                     this.guardarCambiosParametroVistaModelo();
1070                 }
1071             }
1072         } else if (evt.getComponent().equals(this.vistaSPCRUD.jList1)) {
1073             if (evt.getKeyCode() == 127) {
1074                 this.borrarParametroVistaModelo();
1075             } else if (evt.getKeyCode() == 10) {
1079                 if (this.getlasButtonAction() != null) {
1080                     if (this.getlasButtonAction().equals(this.vistaSPCRUD.jButton1)) {
1081                         this.editarParametroVistaModelo();
1082                         this.setLastbuttonAction(null);
1083                     } else if (this.getlasButtonAction().equals(this.vistaSPCRUD.jButton7
     )) {
1084                         this.setLastbuttonAction(null);
1085                         this.borrarParametroVistaModelo();
1086                     }
1087                 } else {
1088                     this.editarParametroVistaModelo();
1089                 }
1090             }
1091         } else if (evt.getComponent().equals(this.vistaSPCRUD.jButton2)) {
1092             if (evt.getKeyCode() == 10) {
1093                 this.agregarParametroVista();
1094             }
1095         } else if (evt.getComponent().equals(this.vistaSPCRUD.jButton1)) {
1096             if (evt.getKeyCode() == 10) {
1097                 this.editarParametroVistaModelo();
1098                 this.setLastbuttonAction(this.vistaSPCRUD.jButton1);
1099             }
1100         } else if (evt.getComponent().equals(this.vistaSPCRUD.jButton7)) {
1101             if (evt.getKeyCode() == 10) {
1102                 this.borrarParametroVistaModelo();
1103                 this.setLastbuttonAction(this.vistaSPCRUD.jButton7);
1104             }
1105         } else if (evt.getComponent().equals(this.vistaSPCRUD.jButton8)) {
1106             if (evt.getKeyCode() == 10) {
1107                 this.guardarCambiosParametroVistaModelo();
1108             }
1109         }
1110     }
```

Gráfico de llamadas para esta función:

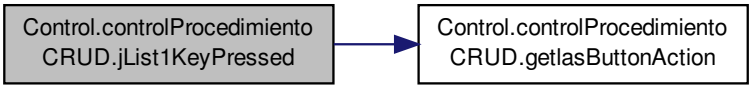
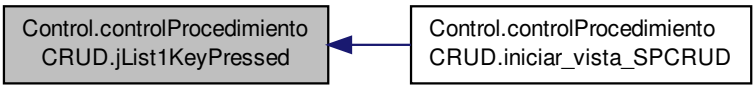


Gráfico de llamadas a esta función:



B.12.3.25. void Control.controlProcedimientoCRUD.jRadioButton1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer la función básica en bases de datos ALTA como la función que implementará el procedimiento almacenado

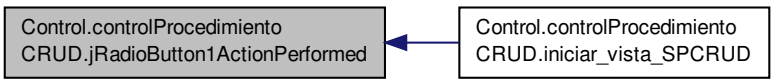
Parámetros

evt	Evento

Definición en la línea 912 del archivo controlProcedimientoCRUD.java.

```
912                                                                         {
913         // TODO add your handling code here:
914         this.CRUD = funcionBasicaBD.ALTA;
915         this.MVNombreSPCRUD();
916         this.modeloSPCRUD.pordefecto();
917         this.MVColumnasSP(this.tmpTabla.getColumnas());
918         this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
      tmpTabla, CRUD));
919     }
```

Gráfico de llamadas a esta función:



B.12.3.26. void Control.controlProcedimientoCRUD.jRadioButton2ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer la función básica en bases de datos BAJA como la función que implementará el procedimiento almacenado.

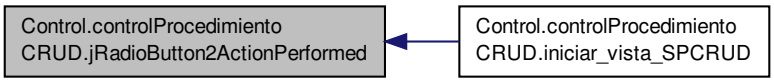
Parámetros

evt	Evento

Definición en la línea 928 del archivo controlProcedimientoCRUD.java.

```
928                                                                         {
929         // TODO add your handling code here:
930         this.CRUD = funcionBasicaBD.BAJA;
931         this.MVNombreSPCRUD();
932         this.tmpTabla.getLlavePrimaria().setCandidatoSP(false);
933         this.modeloSPCRUD.pordefecto();
934         this.MVColumnasSP(this.tmpTabla.getColumnas().getColumnasSP());
935         this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
      tmpTabla, CRUD));
936     }
```

Gráfico de llamadas a esta función:



B.12.3.27. void Control.controlProcedimientoCRUD.jRadioButton3ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer la función básica en bases de datos CAMBIO como la función que implementará el procedimiento almacenado

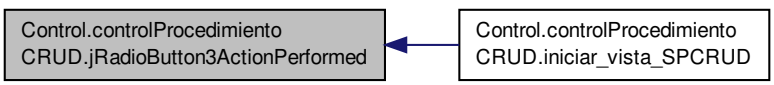
Parámetros

evt	Evento

Definición en la línea 945 del archivo controlProcedimientoCRUD.java.

```
945                                                                         {
946         // TODO add your handling code here:
947         this.CRUD = funcionBasicaBD.CAMBIO;
948         this.MVNombreSPCRUD();
949         this.modeloSPCRUD.pordefecto();
950         this.MVColumnasSPCambio(this.tmpTabla.getColumnas());
951         this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
      tmpTabla, CRUD));
952     }
```

Gráfico de llamadas a esta función:



B.12.3.28. void Control.controlProcedimientoCRUD.jRadioButton4ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Método de respuesta al evento generado por un componente de la vista que solicita establecer la función básica en bases de datos CONSULTA como la función que implementará el procedimiento almacenado

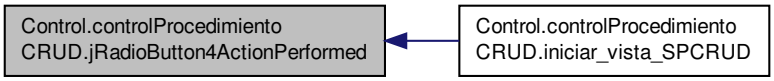
Parámetros

evt	Evento

Definición en la línea 961 del archivo controlProcedimientoCRUD.java.

```
961                                                                         {
962         // TODO add your handling code here:
963         this.CRUD = funcionBasicaBD.CONSULTA;
964         this.MVNombreSPCRUD();
965         this.modeloSPCRUD.pordefecto();
966         this.MVColumnasSPConsulta(this.tmpTabla.getColumnas());
967         this.vistaSPCRUD.jTextArea2.setText(this.modeloSPCRUD.generarProcedmientoFuncionCRUD(
      tmpTabla, CRUD));
968     }
```

Gráfico de llamadas a esta función:



B.12.3.29. void Control.controlProcedimientoCRUD.jTree1ValueChanged (javax.swing.event.TreeSelectionEvent *evt*) [private]

Método que controla la selección de tablas en el árbol. Mantiene el control entre el modelo y la vista

Parámetros

evt	Evento

Definición en la línea 1142 del archivo controlProcedimientoCRUD.java.

```
1142                                                                         {
1143         // TODO add your handling code here:
1144         if (this.esTabla(this.vistaSPCRUD.jTree1.getModel(),
      evt.getNewLeadSelectionPath().getLastPathComponent())) {
1145             this.seleccionadorTablas(evt);
1146         } else {
1147             if (!this.esBD(this.vistaSPCRUD.jTree1.getModel(),
      evt.getNewLeadSelectionPath().getLastPathComponent()))//{
1148             {
1149                 this.vistaSPCRUD.jTree1.setSelectionPath(this.selectionTreePath);
1150             } else {
1151                 this.vistaSPCRUD.jTree1.setSelectionPath(this.selectionTreePath);
1152             }
1153         }
1154     }
```

Gráfico de llamadas para esta función:

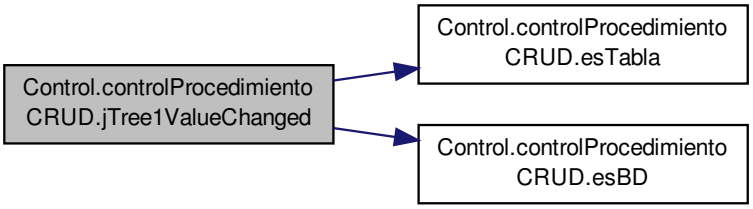
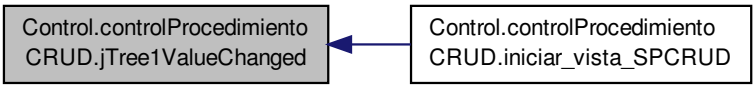


Gráfico de llamadas a esta función:



B.12.3.30. void Control.controlProcedimientoCRUD.limpiarCamposParametros ( ) [private]

Método que limpia los campos asociados a los parametros de un procedimiento en la vista

Definición en la línea 673 del archivo controlProcedimientoCRUD.java.

```
673                                         {
674         this.vistaSPCRUD.jTextField2.setText(null);
675         this.vistaSPCRUD.tipoDatoSQL.setSelectedIndex(0);
676         this.vistaSPCRUD.jComboBox4.setSelectedIndex(0);
677         this.vistaSPCRUD.jComboBox4.requestFocus();
678     }
```

B.12.3.31. void Control.controlProcedimientoCRUD.makeTypesName ( ) [private]

Método que cosntruye el mapa de los tipos de datos en SQL

Definición en la línea 645 del archivo controlProcedimientoCRUD.java.

```
645                                     {
646         this.mapaTipoDatoSQL = new HashMap();
647         Field[] fields = java.sql.Types.class.getFields();
648         for (int i = 0; i < fields.length; i++) {
649             try {
650                 this.mapaTipoDatoSQL.put((Integer) fields[i].get(null), fields[i].getName());
651             } catch (IllegalAccessException e) {
652                 e.printStackTrace();
653             }
654         }
655     }
```

B.12.3.32. void Control.controlProcedimientoCRUD.modeloColumnaParametro (columna *c*)

Método que inicializa un parámetro del procedimiento almacenado en el modelo, a partir de una columna proporcionada.

Parámetros

c	Columna

Definición en la línea 192 del archivo controlProcedimientoCRUD.java.

```
192                                                 {
193         if (this.modeloSPCRUD.listaParametros == null) {
194             this.modeloSPCRUD.setParametros(new Parametros());
195             if (c.llavePrimaria) {
196                 this.modeloSPCRUD.listaParametros.agregarParametro(
197                         new Parametro(
198                                 "OUT",
199                                 c.getNombreColumna(),
200                                 c.getTipoDato(),
201                                 c.getLongitudColumna(),
202                                 c.getLlavePrimaria()
203                         )
204                 );
205             } else {
206                 this.modeloSPCRUD.getParametros().agregarParametro(
207                         new Parametro(
208                                 "IN",
209                                 c.getNombreColumna(),
210                                 c.getTipoDato(),
211                                 c.getLongitudColumna(),
212                                 c.getLlavePrimaria()
213                         )
214                 );
215             }
216         } else {
217             if (c.llavePrimaria) {
218                 this.modeloSPCRUD.getParametros().agregarParametro(
219                         new Parametro(
220                                 "OUT",
221                                 c.getNombreColumna(),
222                                 c.getTipoDato(),
223                                 c.getLongitudColumna(),
224                                 c.getLlavePrimaria()
225                         )
226                 );
227             } else {
228                 this.modeloSPCRUD.getParametros().agregarParametro(
229                         new Parametro(
230                                 "IN",
231                                 c.getNombreColumna(),
232                                 c.getTipoDato(),
233                                 c.getLongitudColumna(),
234                                 c.getLlavePrimaria()
235                         )
236                 );
237             }
238         }
239     }
```

Gráfico de llamadas para esta función:

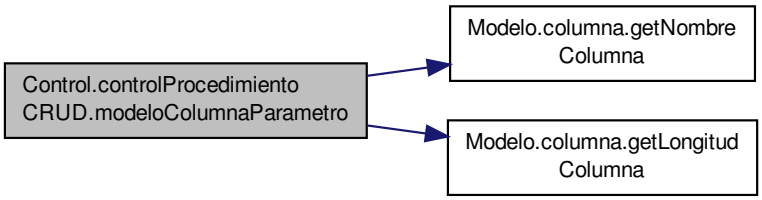
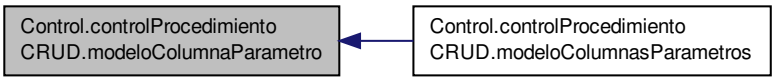


Gráfico de llamadas a esta función:



B.12.3.33. void Control.controlProcedimientoCRUD.modeloColumnaParametroCambio (columna *c*)

Método que inicializa un parámetro del procedimiento almacenado que implementa la función básica CAMBIO en el modelo, a partir de una columna proporcionada.

Parámetros

c	Columna

Definición en la línea 304 del archivo controlProcedimientoCRUD.java.

```
304                                                 {
305         if (this.modeloSPCRUD.listaParametros == null) {
306             this.modeloSPCRUD.setParametros(new Parametros());
307             if (c.llavePrimaria) {
308                 this.modeloSPCRUD.listaParametros.agregarParametro(
309                         new Parametro(
310                                 "IN",
311                                 c.getNombreColumna(),
312                                 c.getTipoDato(),
313                                 c.getLongitudColumna(),
314                                 c.getLlavePrimaria()
315                         )
316                 );
317             } else {
318                 this.modeloSPCRUD.getParametros().agregarParametro(
319                         new Parametro(
320                                 "IN",
321                                 c.getNombreColumna(),
322                                 c.getTipoDato(),
323                                 c.getLongitudColumna(),
324                                 c.getLlavePrimaria()
325                         )
326                 );
```

```
327             }
328         } else {
329             if (c.llavePrimaria) {
330                 this.modeloSPCRUD.getParametros().agregarParametro(
331                         new Parametro(
332                                 "IN",
333                                 c.getNombreColumna(),
334                                 c.getTipoDato(),
335                                 c.getLongitudColumna(),
336                                 c.getLlavePrimaria()
337                         )
338                 );
339             } else {
340                 this.modeloSPCRUD.getParametros().agregarParametro(
341                         new Parametro(
342                                 "IN",
343                                 c.getNombreColumna(),
344                                 c.getTipoDato(),
345                                 c.getLongitudColumna(),
346                                 c.getLlavePrimaria()
347                         )
348                 );
349             }
350         }
351     }
```

Gráfico de llamadas para esta función:

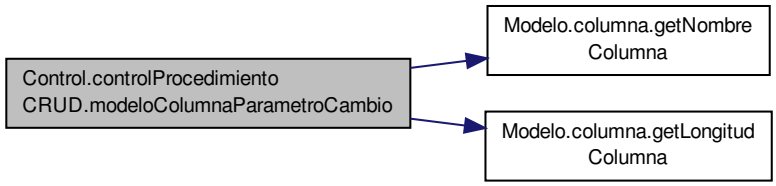
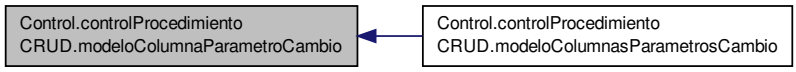


Gráfico de llamadas a esta función:



B.12.3.34. void Control.controlProcedimientoCRUD.modeloColumnaParametroConsulta (columna *c*)

Método que inicializa un parámetro del procedimiento almacenado que implementa la función básica CONSULTA en el modelo, a partir de una columna proporcionada.

Parámetros

c	Columna

Definición en la línea 248 del archivo controlProcedimientoCRUD.java.

```
248                                                     {
249         if (this.modeloSPCRUD.listaParametros == null) {
250             this.modeloSPCRUD.setParametros(new Parametros());
251             if (c.llavePrimaria) {
252                 this.modeloSPCRUD.listaParametros.agregarParametro(
253                         new Parametro(
254                                 "IN",
255                                 c.getNombreColumna(),
256                                 c.getTipoDato(),
257                                 c.getLongitudColumna(),
258                                 c.getLlavePrimaria()
259                         )
260                 );
261             } else {
262                 this.modeloSPCRUD.getParametros().agregarParametro(
263                         new Parametro(
264                                 "OUT",
265                                 c.getNombreColumna(),
266                                 c.getTipoDato(),
267                                 c.getLongitudColumna(),
268                                 c.getLlavePrimaria()
269                         )
270                 );
271             }
272         } else {
273             if (c.llavePrimaria) {
274                 this.modeloSPCRUD.getParametros().agregarParametro(
275                         new Parametro(
276                                 "IN",
277                                 c.getNombreColumna(),
278                                 c.getTipoDato(),
279                                 c.getLongitudColumna(),
280                                 c.getLlavePrimaria()
281                         )
282                 );
283             } else {
284                 this.modeloSPCRUD.getParametros().agregarParametro(
285                         new Parametro(
286                                 "OUT",
287                                 c.getNombreColumna(),
288                                 c.getTipoDato(),
289                                 c.getLongitudColumna(),
290                                 c.getLlavePrimaria()
291                         )
292                 );
293             }
294         }
295     }
```

Gráfico de llamadas para esta función:

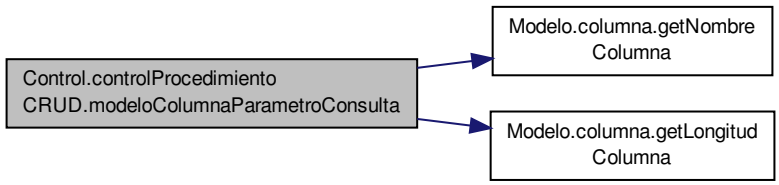
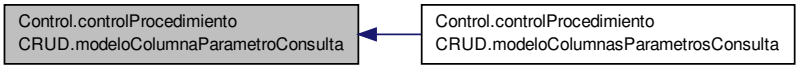


Gráfico de llamadas a esta función:



B.12.3.35. void Control.controlProcedimientoCRUD.modeloColumnasParametros (columnas *C*)

Método que inicializa los parámetro del procedimiento almacenado en el modelo, a partir de una lista de columnas proporcionada.

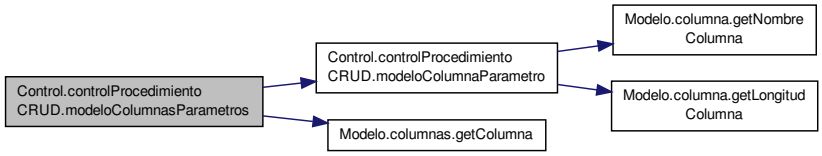
Parámetros

C	Lista de columnas

Definición en la línea 359 del archivo controlProcedimientoCRUD.java.

```
359                                                 {
360         for (int i = 0; i < C.numeroColumnas(); i++) {
361             modeloColumnaParametro(C.getColumna(i));
362         }
363     }
```

Gráfico de llamadas para esta función:



B.12.3.36. void Control.controlProcedimientoCRUD.modeloColumnasParametrosCambio (columnas *C*)

Método que inicializa los parámetros del procedimiento almacenado que implementa la función básica CAMBIO en el modelo, a partir de una lista de columnas proporcionada.

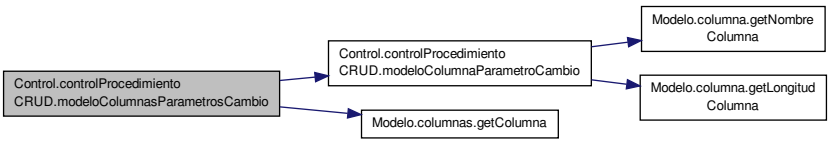
Parámetros

C	Lista de columnas

Definición en la línea 385 del archivo controlProcedimientoCRUD.java.

```
385                                                       {
386         for (int i = 0; i < C.numeroColumnas(); i++) {
387             modeloColumnaParametroCambio(C.
      getColumna(i));
388         }
389     }
```

Gráfico de llamadas para esta función:



B.12.3.37. void Control.controlProcedimientoCRUD.modeloColumnasParametrosConsulta (columnas *C*)

Método que inicializa los parámetros del procedimiento almacenado que implementa la función básica CONSULTA en el modelo, a partir de una lista de columnas proporcionada.

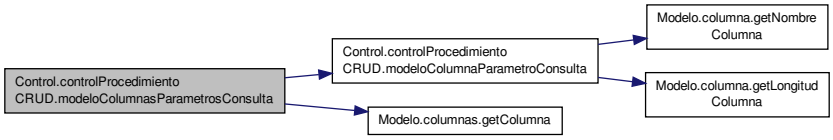
Parámetros

C	Lista de columnas

Definición en la línea 372 del archivo controlProcedimientoCRUD.java.

```
372                                                         {
373         for (int i = 0; i < C.numeroColumnas(); i++) {
374             modeloColumnaParametroConsulta(C.
      getColumna(i));
375         }
376     }
```

Gráfico de llamadas para esta función:



B.12.3.38. void Control.controlProcedimientoCRUD.MVColumnasSP (columnas *C*)

Método que se encarga de establecer los parámetros sugeridos para el procedimiento a partir de una lista de columnas. Mantiene el control entre el modelo y la vista.

Parámetros

C	Lista de columnas

Definición en la línea 426 del archivo controlProcedimientoCRUD.java.

```
426                                     {
427         this.modeloColumnasParametros(C);
428         this.agregarParamModeloVista();
429     }
```

B.12.3.39. void Control.controlProcedimientoCRUD.MVColumnasSPCambio (columnas *C*)

Método que se encarga de establecer los parámetros sugeridos para el procedimiento que implementa la función básica CAMBIO, a partir de una lista de columnas. Mantiene el control entre el modelo y la vista.

Parámetros

C	Lista de columnas

Definición en la línea 450 del archivo controlProcedimientoCRUD.java.

```
450                                             {
451         this.modeloColumnasParametrosCambio(C);
452         this.agregarParamModeloVista();
453     }
```

B.12.3.40. void Control.controlProcedimientoCRUD.MVColumnasSPConsulta (columnas *C*)

Método que se encarga de establecer los parámetros sugeridos para el procedimiento que implementa la función básica CONSULTA, a partir de una lista de columnas. Mantiene el control entre el modelo y la vista.

Parámetros

C	Lista de columnas

Definición en la línea 438 del archivo controlProcedimientoCRUD.java.

```
438                                             {
439         this.modeloColumnasParametrosConsulta(C);
440         this.agregarParamModeloVista();
441     }
```

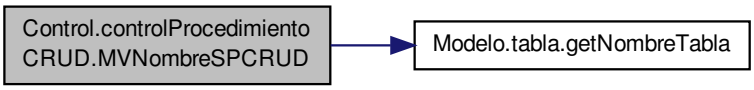
B.12.3.41. void Control.controlProcedimientoCRUD.MVNombreSPCRUD ( )

Método que establece el nombre del procedimiento almacenado que implementa la función CRUD

Definición en la línea 459 del archivo controlProcedimientoCRUD.java.

```
459                                 {
460         this.vistaSPCRUD.jTextField1.setText(
461                 this.getTextRadioButtonSelect() + this.tmpTabla.
    getNombreTabla()
462         );
463     }
```

Gráfico de llamadas para esta función:



B.12.3.42. boolean Control.controlProcedimientoCRUD.parametroValido (Parametro *P*, tabla *T*)

Método que determina si un parámetro es valido

Parámetros

P	Parámetro
T	Tabla

Devuelve

boolean

- true indica que el parámetro es valido
- false indica que el parámetro no es valido

Definición en la línea 691 del archivo controlProcedimientoCRUD.java.

```
691                                                     {
692         return T.contiene(new columna(P.getNombre(), P.getTipoDato()));
693     }
```

Gráfico de llamadas para esta función:

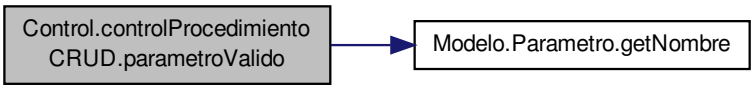
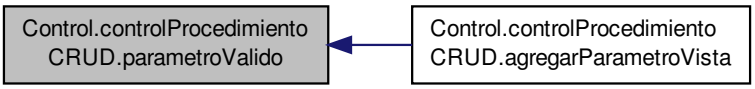


Gráfico de llamadas a esta función:



B.12.3.43. void Control.controlProcedimientoCRUD.pordefecto ( )

Método que establece la vista por defecto

Definición en la línea 468 del archivo controlProcedimientoCRUD.java.

```
468                             {
469         this.MVColumnasSP(this.tmpTabla.getColumnas().getColumnasSP());
470         this.getFuncionCRUD();
471     }
```

B.12.3.44. void Control.controlProcedimientoCRUD.seleccionadorTablas (javax.swing.event.TreeSelectionEvent *evt*)

MouseClick Método que obtiene los parámtros y configuraciones del procedimiento asociado a la tabla seleccionada en el árbol.

Parámetros

evt	Evento

Definición en la línea 1121 del archivo controlProcedimientoCRUD.java.

```
1121                                                                        {
1122         if (!this.tmpTabla.equals(this.modeloEsquemaBD.getTablas().getTabla(
      evt.getNewLeadSelectionPath().getLastPathComponent().toString()))) {
1123             this.selectionTreePath = evt.getNewLeadSelectionPath();
1124             this.tmpTabla = this.modeloEsquemaBD.getTablas().getTabla(evt.getNewLeadSelectionPath()
      .getLastPathComponent().toString());
1128             this.modeloSPCRUD.pordefecto();
1132             this.pordefecto();
1133         }
1134     }
```

B.12.3.45. void Control.controlProcedimientoCRUD.setConexionBDSQL (Connection *C*)

Método que establece la variable que contiene la conexión establecida con alguna base de datos

Parámetros

C	Conexión establecida con alguna base de datos

Definición en la línea 81 del archivo controlProcedimientoCRUD.java.

```
81                                              {
82         this.conexionBDSQL = C;
83     }
```

B.12.3.46. void Control.controlProcedimientoCRUD.setFocusPorDefecto ( )

Método que establece el orden de los elementos de la interfaz para la política de recorrimiento del focus

Definición en la línea 616 del archivo controlProcedimientoCRUD.java.

```
616                                     {
617         ArrayList orden = new ArrayList();
618         orden.add(this.vistaSPCRUD.jTextField1);
619         orden.add(this.vistaSPCRUD.jRadioButton1);
620         orden.add(this.vistaSPCRUD.jRadioButton2);
621         orden.add(this.vistaSPCRUD.jRadioButton3);
622         orden.add(this.vistaSPCRUD.jRadioButton4);
623         orden.add(this.vistaSPCRUD.jComboBox4);
624         orden.add(this.vistaSPCRUD.jTextField2);
625         orden.add(this.vistaSPCRUD.tipoDatoSQL);
626         orden.add(this.vistaSPCRUD.jButton2);
627         orden.add(this.vistaSPCRUD.jButton1);
628         orden.add(this.vistaSPCRUD.jButton7);
629         orden.add(this.vistaSPCRUD.jList1);
630         orden.add(this.vistaSPCRUD.jCheckBox3);
631         orden.add(this.vistaSPCRUD.jComboBox2);
632         orden.add(this.vistaSPCRUD.jComboBox6);
633         orden.add(this.vistaSPCRUD.jTextArea3);
634         orden.add(this.vistaSPCRUD.jTextArea2);
635         orden.add(this.vistaSPCRUD.jTree1);
636         orden.add(this.vistaSPCRUD.jButton4);
637         orden.add(this.vistaSPCRUD.jButton3);
638         MiFocusTraversalPolicy newPolicy = new MiFocusTraversalPolicy(orden);
639         this.vistaSPCRUD.setFocusTraversalPolicy(newPolicy);
640     }
```

B.12.3.47. void Control.controlProcedimientoCRUD.setLastbuttonAction (Component *lasButtonAction*)

Método que establece el ultimo componente que provocó algún evento

Parámetros

lasButtonAction	Componente

Definición en la línea 100 del archivo controlProcedimientoCRUD.java.

```
100                                                     {
101         this.lasButtonAction = lasButtonAction;
102     }
```

B.12.4. Documentación de los datos miembro

B.12.4.1. Connection Control.controlProcedimientoCRUD.conexionBDSQL [private]

Conexión establecida con alguna base de datos

Definición en la línea 57 del archivo controlProcedimientoCRUD.java.

B.12.4.2. funcionBasicaBD Control.controlProcedimientoCRUD.CRUD [private]

Función básica CRUD

Definición en la línea 53 del archivo controlProcedimientoCRUD.java.

B.12.4.3. Component Control.controlProcedimientoCRUD.lasButtonAction [private]

Componente de la interfaz de usuario

Definición en la línea 61 del archivo controlProcedimientoCRUD.java.

B.12.4.4. Map Control.controlProcedimientoCRUD.mapaTipoDatoSQL

Mapa de los tipos de datos en SQL

Definición en la línea 65 del archivo controlProcedimientoCRUD.java.

B.12.4.5. esquemaBD Control.controlProcedimientoCRUD.modeloEsquemaBD [private]

Esquema de la base de datos

Definición en la línea 49 del archivo controlProcedimientoCRUD.java.

B.12.4.6. ProcedimientoAlmacenado Control.controlProcedimientoCRUD.modeloSPCRUD [private]

Modelo: Objeto de la clase que implementa un procedimiento almacenado

Definición en la línea 41 del archivo controlProcedimientoCRUD.java.

B.12.4.7. int Control.controlProcedimientoCRUD.SelectionCount [private]

Número de fila asociado al elemento seleccionado en el árbol de tablas

Definición en la línea 69 del archivo controlProcedimientoCRUD.java.

B.12.4.8. TreePath Control.controlProcedimientoCRUD.selectionTreePath [private]

Trayecto en el árbol de tablas, asociado al elemento seleccionado

Definición en la línea 73 del archivo controlProcedimientoCRUD.java.

B.12.4.9. tabla Control.controlProcedimientoCRUD.tmpTabla [private]

Tabla

Definición en la línea 45 del archivo controlProcedimientoCRUD.java.

B.12.4.10. EditorProcedimientoCRUD Control.controlProcedimientoCRUD.vistaSPCRUD [private]

Vista: Editor de un procedimiento almacenado CRUD

Definición en la línea 37 del archivo controlProcedimientoCRUD.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/controlProcedimientoCRUD.java

B.13. Referencia de la Clase Vistas.EditorCodigoAnidadoGenerado

Diagrama de herencias de Vistas.EditorCodigoAnidadoGenerado

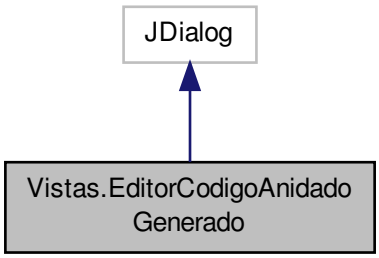
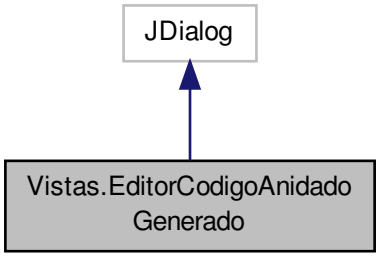


Diagrama de colaboración para Vistas.EditorCodigoAnidadoGenerado:



Métodos públicos

- EditorCodigoAnidadoGenerado (java.awt.Frame parent, boolean modal)

Métodos públicos estáticos

- static void main (String args[])

Atributos públicos

- javax.swing.JButton jButton1
- javax.swing.JButton jButton2
- javax.swing.JEditorPane visualizadorSQL

Métodos privados

- void initComponents ()

Atributos privados

- javax.swing.JScrollPane jScrollPane2

B.13.1. Descripción detallada

Autor

ivan

Definición en la línea 13 del archivo EditorCodigoAnidadoGenerado.java.

B.13.2. Documentación del constructor y destructor

B.13.2.1. Vistas.EditorCodigoAnidadoGenerado.EditorCodigoAnidadoGenerado (java.awt.Frame *parent,* boolean *modal*)

Creates new form EditarCodigoAnidadoGenerado

Definición en la línea 18 del archivo EditorCodigoAnidadoGenerado.java.

```
18                                                                          {
19         super(parent, modal);
20         initComponents();
21     }
```

Gráfico de llamadas para esta función:

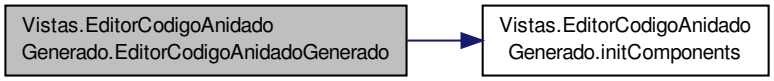
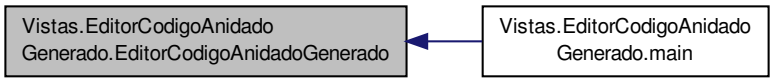


Gráfico de llamadas a esta función:



B.13.3. Documentación de las funciones miembro

B.13.3.1. void Vistas.EditorCodigoAnidadoGenerado.initComponents ( ) [private]

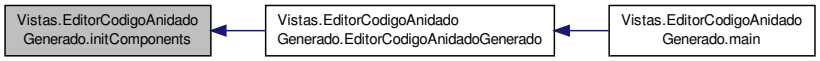
This method is called from within the constructor to initialize the form. WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

Definición en la línea 30 del archivo EditorCodigoAnidadoGenerado.java.

```
30                                {
31
32         jButton1 = new javax.swing.JButton();
33         jButton2 = new javax.swing.JButton();
34         jScrollPane2 = new javax.swing.JScrollPane();
35         visualizadorSQL = new javax.swing.JEditorPane();
36
37         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
38
39         jButton1.setText("Finalizar");
40
41         jButton2.setText("Guardar como");
42
43         visualizadorSQL.setEditable(false);
```

```
44          visualizadorSQL.setContentType("text/html");
45          jScrollPane2.setViewportView(visualizadorSQL);
46
47          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
48          getContentPane().setLayout(layout);
49          layout.setHorizontalGroup(
50              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
51              .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
52                  .addContainerGap()
53                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
54                      .addComponent(jScrollPane2)
55                      .addGroup(layout.createSequentialGroup()
56                          .addGap(0, 222, Short.MAX_VALUE)
57                          .addComponent(jButton1)
58                          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
59                          .addComponent(jButton2)))
60                  .addContainerGap())
61          );
62          layout.setVerticalGroup(
63              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
64              .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
65                  .addContainerGap()
66                  .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 243, Short.MAX_VALUE)
67                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
68                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
69                      .addComponent(jButton1)
70                      .addComponent(jButton2))
71                  .addContainerGap())
72          );
73
74          pack();
75      }// </editor-fold>//GEN-END:initComponents
```

Gráfico de llamadas a esta función:



B.13.3.2. static void Vistas.EditorCodigoAnidadoGenerado.main (String *args[]*) [static]

Parámetros

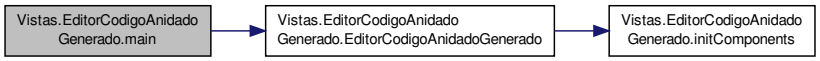
args	the command line arguments

Definición en la línea 80 del archivo EditorCodigoAnidadoGenerado.java.

```
80                                      {
81          /* Set the Nimbus look and feel */
82          //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
83          /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
84           * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
85           */
86          try {
87              for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
        getInstalledLookAndFeels()) {
88                  if ("Nimbus".equals(info.getName())) {
89                      javax.swing.UIManager.setLookAndFeel(info.getClassName());
90                      break;
91                  }
92              }
93          } catch (ClassNotFoundException ex) {
94              java.util.logging.Logger.getLogger(EditorCodigoAnidadoGenerado.class.getName()).log(
        java.util.logging.Level.SEVERE, null, ex);
95          } catch (InstantiationException ex) {
96              java.util.logging.Logger.getLogger(EditorCodigoAnidadoGenerado.class.getName()).log(
```

```
      java.util.logging.Level.SEVERE, null, ex);
97          } catch (IllegalAccessException ex) {
98              java.util.logging.Logger.getLogger(EditorCodigoAnidadoGenerado.class.getName()).log(
      java.util.logging.Level.SEVERE, null, ex);
99          } catch (javax.swing.UnsupportedLookAndFeelException ex) {
100             java.util.logging.Logger.getLogger(EditorCodigoAnidadoGenerado.class.getName()).log(
      java.util.logging.Level.SEVERE, null, ex);
101         }
102         //</editor-fold>
103
104         /* Create and display the dialog */
105         java.awt.EventQueue.invokeLater(new Runnable() {
106             public void run() {
107                 EditorCodigoAnidadoGenerado dialog = new
      EditorCodigoAnidadoGenerado(new javax.swing.JFrame(), true);
108                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
109                     @Override
110                     public void windowClosing(java.awt.event.WindowEvent e) {
111                         System.exit(0);
112                     }
113                 });
114                 dialog.setVisible(true);
115             }
116         });
117     }
```

Gráfico de llamadas para esta función:



B.13.4. Documentación de los datos miembro

B.13.4.1. javax.swing.JButton Vistas.EditorCodigoAnidadoGenerado.jButton1

Definición en la línea 120 del archivo EditorCodigoAnidadoGenerado.java.

B.13.4.2. javax.swing.JButton Vistas.EditorCodigoAnidadoGenerado.jButton2

Definición en la línea 121 del archivo EditorCodigoAnidadoGenerado.java.

B.13.4.3. javax.swing.JScrollPane Vistas.EditorCodigoAnidadoGenerado.jScrollPane2 [private]

Definición en la línea 122 del archivo EditorCodigoAnidadoGenerado.java.

B.13.4.4. javax.swing.JEditorPane Vistas.EditorCodigoAnidadoGenerado.visualizadorSQL

Definición en la línea 123 del archivo EditorCodigoAnidadoGenerado.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vistas/EditorCodigoAnidadoGenerado.java

B.14. Referencia de la Clase Vistas.EditorCodigoGenerado

Diagrama de herencias de Vistas.EditorCodigoGenerado

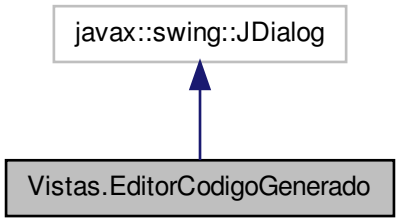
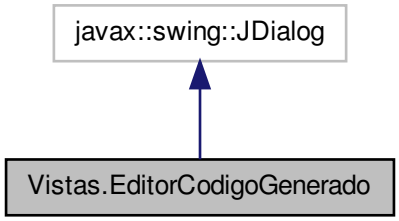


Diagrama de colaboración para Vistas.EditorCodigoGenerado:



Métodos públicos

- EditorCodigoGenerado (java.awt.Frame parent, boolean modal)

Métodos públicos estáticos

- static void main (String args[])

Atributos públicos

- javax.swing.JEditorPane Visualizador
- javax.swing.JButton jButton1
- javax.swing.JButton jButton2
- javax.swing.JButton jButton3
- javax.swing.JCheckBox jCheckBox1

Métodos privados

- void initComponents ()

Atributos privados

- javax.swing.JScrollPane jScrollPane2

B.14.1. Descripción detallada

Autor

ivan

Definición en la línea 13 del archivo EditorCodigoGenerado.java.

B.14.2. Documentación del constructor y destructor

B.14.2.1. Vistas.EditorCodigoGenerado.EditorCodigoGenerado (java.awt.Frame *parent*, boolean *modal*)

Creates new form SPEditorCodigoGenerado

Definición en la línea 18 del archivo EditorCodigoGenerado.java.

```
18                                                                      {
19         super(parent, modal);
20         initComponents();
21     }
```

Gráfico de llamadas para esta función:

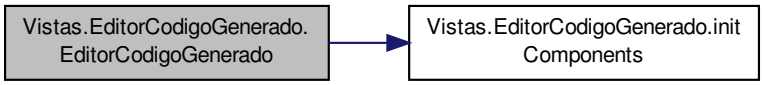
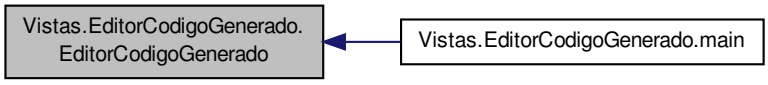


Gráfico de llamadas a esta función:



B.14.3. Documentación de las funciones miembro

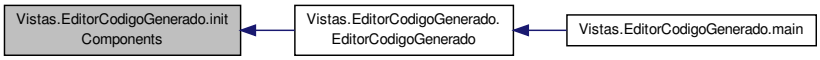
B.14.3.1. void Vistas.EditorCodigoGenerado.initComponents ( ) [private]

This method is called from within the constructor to initialize the form. WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

Definición en la línea 30 del archivo EditorCodigoGenerado.java.

```
30                                {
31
32          jCheckBox1 = new javax.swing.JCheckBox();
33          jButton1 = new javax.swing.JButton();
34          jButton2 = new javax.swing.JButton();
35          jButton3 = new javax.swing.JButton();
36          jScrollPane2 = new javax.swing.JScrollPane();
37          Visualizador = new javax.swing.JEditorPane();
38
39          setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
40
41          jCheckBox1.setText("remplazar existente");
42
43          jButton1.setText("Enviar a BD");
44
45          jButton2.setText("Guarda como");
46
47          jButton3.setText("Finalizar");
48
49          Visualizador.setEditable(false);
50          Visualizador.setContentType("text/html");
51          jScrollPane2.setViewportView(Visualizador);
52
53          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
54          getContentPane().setLayout(layout);
55          layout.setHorizontalGroup(
56              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
57              .addGroup(layout.createSequentialGroup()
58                  .addContainerGap()
59                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
60                      .addComponent(jScrollPane2)
61                      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
62                          .addGap(0, 136, Short.MAX_VALUE)
63                          .addComponent(jButton3)
64                          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
65                          .addComponent(jButton2)
66                          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
67                          .addComponent(jButton1))
68                      .addGroup(layout.createSequentialGroup()
69                          .addComponent(jCheckBox1)
70                          .addGap(0, 0, Short.MAX_VALUE)))
71                  .addContainerGap())
72          );
73          layout.setVerticalGroup(
74              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
75              .addGroup(layout.createSequentialGroup()
76                  .addContainerGap()
77                  .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 243, javax.swing.
    GroupLayout.PREFERRED_SIZE)
78                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
79                  .addComponent(jCheckBox1)
80                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
81                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
82                      .addComponent(jButton1)
83                      .addComponent(jButton2)
84                      .addComponent(jButton3))
85                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
86          );
87
88          pack();
89      }// </editor-fold>//GEN-END:initComponents
```

Gráfico de llamadas a esta función:



B.14.3.2. static void Vistas.EditorCodigoGenerado.main (String *args[]*) [static]

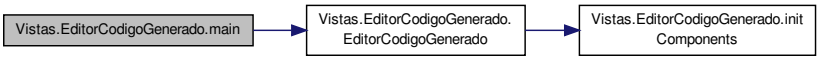
Parámetros

args	the command line arguments

Definición en la línea 94 del archivo EditorCodigoGenerado.java.

```
94                                      {
95         /* Set the Nimbus look and feel */
96         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
97         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
98          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
99          */
100        try {
101            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
    getInstalledLookAndFeels()) {
102                if ("Nimbus".equals(info.getName())) {
103                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
104                    break;
105                }
106            }
107        } catch (ClassNotFoundException ex) {
108            java.util.logging.Logger.getLogger(EditorCodigoGenerado.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
109        } catch (InstantiationException ex) {
110            java.util.logging.Logger.getLogger(EditorCodigoGenerado.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
111        } catch (IllegalAccessException ex) {
112            java.util.logging.Logger.getLogger(EditorCodigoGenerado.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
113        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
114            java.util.logging.Logger.getLogger(EditorCodigoGenerado.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
115        }
116        //</editor-fold>
117
118        /* Create and display the dialog */
119        java.awt.EventQueue.invokeLater(new Runnable() {
120            public void run() {
121                EditorCodigoGenerado dialog = new
    EditorCodigoGenerado(new javax.swing.JFrame(), true);
122                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
123                    @Override
124                    public void windowClosing(java.awt.event.WindowEvent e) {
125                        System.exit(0);
126                    }
127                });
128                dialog.setVisible(true);
129            }
130        });
131    }
```

Gráfico de llamadas para esta función:



B.14.4. Documentación de los datos miembro

B.14.4.1. javax.swing.JButton Vistas.EditorCodigoGenerado.jButton1

Definición en la línea 135 del archivo EditorCodigoGenerado.java.

B.14.4.2. javax.swing.JButton Vistas.EditorCodigoGenerado.jButton2

Definición en la línea 136 del archivo EditorCodigoGenerado.java.

B.14.4.3. javax.swing.JButton Vistas.EditorCodigoGenerado.jButton3

Definición en la línea 137 del archivo EditorCodigoGenerado.java.

B.14.4.4. javax.swing.JCheckBox Vistas.EditorCodigoGenerado.jCheckBox1

Definición en la línea 138 del archivo EditorCodigoGenerado.java.

B.14.4.5. javax.swing.JScrollPane Vistas.EditorCodigoGenerado.jScrollPane2 [private]

Definición en la línea 139 del archivo EditorCodigoGenerado.java.

B.14.4.6. javax.swing.JEditorPane Vistas.EditorCodigoGenerado.Visualizador

Definición en la línea 134 del archivo EditorCodigoGenerado.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vistas/EditorCodigoGenerado.java

B.15. Referencia de la Clase Vistas.EditorProcedimiento

Diagrama de herencias de Vistas.EditorProcedimiento

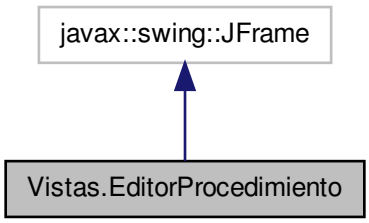
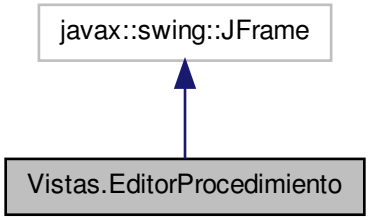


Diagrama de colaboración para Vistas.EditorProcedimiento:



Métodos públicos

- EditorProcedimiento ()

Métodos públicos estáticos

- static void main (String args[])

Atributos públicos

- javax.swing.JButton jButton1
- javax.swing.JButton jButton10
- javax.swing.JButton jButton2
- javax.swing.JButton jButton3

- javax.swing.JButton jButton4
- javax.swing.JButton jButton5
- javax.swing.JButton jButton6
- javax.swing.JButton jButton7
- javax.swing.JButton jButton8
- javax.swing.JButton jButton9
- javax.swing.JCheckBox jCheckBox2
- javax.swing.JCheckBox jCheckBox3
- javax.swing.JComboBox jComboBox1
- javax.swing.JComboBox jComboBox2
- javax.swing.JComboBox jComboBox3
- javax.swing.JComboBox jComboBox4
- javax.swing.JComboBox jComboBox6
- javax.swing.JLabel jLabel1
- javax.swing.JLabel jLabel10
- javax.swing.JLabel jLabel2
- javax.swing.JLabel jLabel3
- javax.swing.JLabel jLabel4
- javax.swing.JLabel jLabel5
- javax.swing.JLabel jLabel6
- javax.swing.JLabel jLabel7
- javax.swing.JLabel jLabel8
- javax.swing.JLabel jLabel9
- javax.swing.JList jList1
- javax.swing.JMenu jMenu1
- javax.swing.JMenuBar jMenuBar1
- javax.swing.JMenuItem jMenuItem1
- javax.swing.JMenuItem jMenuItem2
- javax.swing.JMenuItem jMenuItem3
- javax.swing.JPanel jPanel2
- javax.swing.JPanel jPanel3
- javax.swing.JPanel jPanel4
- javax.swing.JPanel jPanel5
- javax.swing.JPanel jPanel6
- javax.swing.JPanel jPanel7
- javax.swing.JPopupMenu jPopupMenu1
- javax.swing.JScrollPane jScrollPane1
- javax.swing.JScrollPane jScrollPane2
- javax.swing.JScrollPane jScrollPane3
- javax.swing.JScrollPane jScrollPane4
- javax.swing.JScrollPane jScrollPane6
- javax.swing.JSeparator jSeparator1
- javax.swing.JTextArea jTextArea2
- javax.swing.JTextArea jTextArea3
- javax.swing.JTextField jTextField1
- javax.swing.JTextField jTextField2
- javax.swing.JTree jTree1
- javax.swing.JTree jTree2
- javax.swing.JComboBox tipoDatoSQL

Métodos privados

- void initComponents ()
- void jMenuItem1ActionPerformed (java.awt.event.ActionEvent evt)

B.15.1. Descripción detallada

Autor

ivan

Definición en la línea 13 del archivo EditorProcedimiento.java.

B.15.2. Documentación del constructor y destructor

B.15.2.1. Vistas.EditorProcedimiento.EditorProcedimiento ( )

Creates new form SPEditor

Definición en la línea 18 del archivo EditorProcedimiento.java.

```
18                                  {
19         initComponents();
20     }
```

Gráfico de llamadas para esta función:

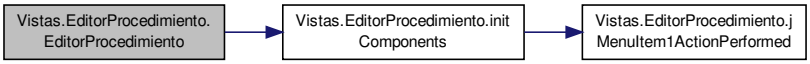
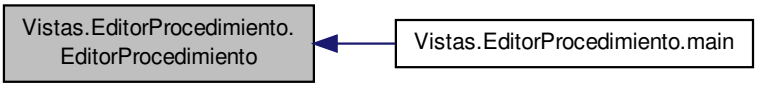


Gráfico de llamadas a esta función:



B.15.3. Documentación de las funciones miembro

B.15.3.1. void Vistas.EditorProcedimiento.initComponents ( ) [private]

This method is called from within the constructor to initialize the form. WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

Definición en la línea 29 del archivo EditorProcedimiento.java.

```
29                                  {
30
31          jPopupMenu1 = new javax.swing.JPopupMenu();
32          jMenuItem1 = new javax.swing.JMenuItem();
33          jPanel4 = new javax.swing.JPanel();
34          jPanel3 = new javax.swing.JPanel();
35          jScrollPane3 = new javax.swing.JScrollPane();
36          jTextArea2 = new javax.swing.JTextArea();
37          jButton6 = new javax.swing.JButton();
38          jPanel2 = new javax.swing.JPanel();
39          jLabel3 = new javax.swing.JLabel();
40          jTextField2 = new javax.swing.JTextField();
41          jLabel4 = new javax.swing.JLabel();
42          jButton2 = new javax.swing.JButton();
43          jButton1 = new javax.swing.JButton();
44          jLabel6 = new javax.swing.JLabel();
45          jComboBox4 = new javax.swing.JComboBox();
46          jSeparator1 = new javax.swing.JSeparator();
47          jScrollPane1 = new javax.swing.JScrollPane();
48          jList1 = new javax.swing.JList();
49          jButton7 = new javax.swing.JButton();
50          jButton8 = new javax.swing.JButton();
51          tipoDatoSQL = new javax.swing.JComboBox();
52          jLabel1 = new javax.swing.JLabel();
53          jTextField1 = new javax.swing.JTextField();
54          jButton4 = new javax.swing.JButton();
55          jButton3 = new javax.swing.JButton();
56          jLabel10 = new javax.swing.JLabel();
57          jLabel2 = new javax.swing.JLabel();
58          jComboBox1 = new javax.swing.JComboBox();
59          jLabel5 = new javax.swing.JLabel();
60          jComboBox3 = new javax.swing.JComboBox();
61          jPanel6 = new javax.swing.JPanel();
62          jCheckBox2 = new javax.swing.JCheckBox();
63          jCheckBox3 = new javax.swing.JCheckBox();
64          jComboBox2 = new javax.swing.JComboBox();
65          jComboBox6 = new javax.swing.JComboBox();
66          jScrollPane6 = new javax.swing.JScrollPane();
67          jTextArea3 = new javax.swing.JTextArea();
68          jLabel7 = new javax.swing.JLabel();
69          jLabel8 = new javax.swing.JLabel();
70          jLabel9 = new javax.swing.JLabel();
71          jButton9 = new javax.swing.JButton();
72          jButton10 = new javax.swing.JButton();
73          jPanel5 = new javax.swing.JPanel();
74          jScrollPane4 = new javax.swing.JScrollPane();
75          jTree1 = new javax.swing.JTree();
76          jButton5 = new javax.swing.JButton();
77          jPanel7 = new javax.swing.JPanel();
78          jScrollPane2 = new javax.swing.JScrollPane();
79          jTree2 = new javax.swing.JTree();
80          jMenuBar1 = new javax.swing.JMenuBar();
81          jMenu1 = new javax.swing.JMenu();
82          jMenuItem2 = new javax.swing.JMenuItem();
83          jMenuItem3 = new javax.swing.JMenuItem();
84
85          jMenuItem1.setText("Crear SP CRUD");
86          jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
87              public void actionPerformed(java.awt.event.ActionEvent evt) {
88                  jMenuItem1ActionPerformed(evt);
89              }
90          });
91          jPopupMenu1.add(jMenuItem1);
92
93          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
94
95          jPanel4.setBorder(javax.swing.BorderFactory.createTitledBorder("Procedimiento Almacenado"));
96
97          jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder("*cuerpo rutina"));
98
99          jTextArea2.setColumns(20);
100          jTextArea2.setRows(5);
101          jTextArea2.setText("# agrega todo tu codigo de control aquí: cualquier tipo de datos MySQL valido")
     ;
102          jScrollPane3.setViewportView(jTextArea2);
103
104          jButton6.setText("Nuevo Procedimiento Anidado");
105
106          javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(
```

```
    jPanel3);
107         jPanel3.setLayout(jPanel3Layout);
108         jPanel3Layout.setHorizontalGroup(
109             jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
110             .addGroup(jPanel3Layout.createSequentialGroup()
111                 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
112                     .addComponent(jScrollPane3)
113                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
    jPanel3Layout.createSequentialGroup()
114                         .addGap(0, 0, Short.MAX_VALUE)
115                         .addComponent(jButton6)))
116                 .addContainerGap())
117         );
118         jPanel3Layout.setVerticalGroup(
119             jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
120             .addGroup(jPanel3Layout.createSequentialGroup()
121                 .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
122                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
123                 .addComponent(jButton6)
124                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
125         );
126 
127         jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("parámetros"));
128 
129         jLabel3.setText("nombre del parametro:");
130 
131         jLabel4.setText("tipo de dato:");
132 
133         jButton2.setText("»");
134 
135         jButton1.setText("«");
136 
137         jLabel6.setText("tipo de entrada:");
138 
139         jComboBox4.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "IN", "OUT", "INOUT" }));
140 
141         jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);
142 
143         jScrollPane1.setViewportView(jList1);
144 
145         jButton7.setText("borrar");
146         jButton7.setMaximumSize(new java.awt.Dimension(26, 27));
147         jButton7.setMinimumSize(new java.awt.Dimension(26, 27));
148         jButton7.setPreferredSize(new java.awt.Dimension(26, 27));
149 
150         jButton8.setText("guardar cambios");
151         jButton8.setEnabled(false);
152 
153         javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(
    jPanel2);
154         jPanel2.setLayout(jPanel2Layout);
155         jPanel2Layout.setHorizontalGroup(
156             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
157             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
158                 .addContainerGap()
159                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
160                     .addGroup(jPanel2Layout.createSequentialGroup()
161                         .addComponent(jLabel3)
162                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
163                         .addComponent(jTextField2))
164                     .addGroup(jPanel2Layout.createSequentialGroup()
165                         .addComponent(jLabel4)
166                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
167                         .addComponent(tipoDatoSQL, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
168                     .addGroup(jPanel2Layout.createSequentialGroup()
169                         .addComponent(jLabel6)
170                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
171                         .addComponent(jComboBox4, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE))
172                     .addGroup(jPanel2Layout.createSequentialGroup()
173                         .addGap(0, 0, Short.MAX_VALUE)
174                         .addComponent(jButton8)))
175                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
176                 .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 11, javax.
    swing.GroupLayout.PREFERRED_SIZE)
177                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
178                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
179                     .addComponent(jButton1)
180                     .addComponent(jButton2)
```

```
181                     .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
    javax.swing.GroupLayout.PREFERRED_SIZE))
182                 .addGap(18, 18, 18)
183                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 139, javax.
    swing.GroupLayout.PREFERRED_SIZE)
184                 .addContainerGap())
185         );
186         jPanel2Layout.setVerticalGroup(
187             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
188             .addGroup(jPanel2Layout.createSequentialGroup()
189                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190                     .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 111,
    javax.swing.GroupLayout.PREFERRED_SIZE)
191                     .addGroup(jPanel2Layout.createSequentialGroup()
192                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
193                             .addComponent(jLabel6)
194                             .addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
195                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
196                         .addGroup(jPanel2Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.BASELINE)
197                             .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
198                             .addComponent(jLabel3))
199                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
200                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
201                             .addComponent(jLabel4)
202                             .addComponent(tipoDatoSQL, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
203                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
204                         .addComponent(jButton8))
205                     .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
     false)
206                         .addGroup(jPanel2Layout.createSequentialGroup()
207                             .addComponent(jButton2)
208                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
209                             .addComponent(jButton1)
210                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.
    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
211                             .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
212                         .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
    javax.swing.GroupLayout.PREFERRED_SIZE, 90, javax.swing.GroupLayout.PREFERRED_SIZE)))
213                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
214         );
215 
216         jLabel1.setText("*nombre del procedimiento");
217 
218         jButton4.setText("Vista previa");
219 
220         jButton3.setText("Generar código");
221 
222         jLabel10.setText("* Campos obligatorios");
223 
224         jLabel2.setText("usuario:");
225 
226         jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "CURRENT_USER" }));
227 
228         jLabel5.setText("base de datos:");
229 
230         jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "<por defecto>" }));
231 
232         jPanel6.setBorder(javax.swing.BorderFactory.createTitledBorder(
    javax.swing.BorderFactory.createTitledBorder("caracteristicas")));
233 
234         jCheckBox2.setText("LANGUAGE SQL");
235         jCheckBox2.setEnabled(false);
236 
237         jCheckBox3.setText("DETERMINISTIC");
238 
239         jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "DEFINER", "INVOKER" }));
240 
241         jComboBox6.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "CONTAINS SQL", "NO SQL", "
    READS SQL DATA", "MODIFIES SQL DATA" }));
242 
243         jTextArea3.setColumns(20);
244         jTextArea3.setRows(5);
245         jTextArea3.setText("Sin comentarios");
246         jScrollPane6.setViewportView(jTextArea3);
```

```
247
248         jLabel7.setText("SQL SECURITY:");
249
250         jLabel8.setText("información sobre manejo de datos:");
251
252         jLabel9.setText("COMENT:");
253
254         javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(
    jPanel6);
255         jPanel6.setLayout(jPanel6Layout);
256         jPanel6Layout.setHorizontalGroup(
257             jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
258             .addGroup(jPanel6Layout.createSequentialGroup()
259                 .addGap(6, 6, 6)
260                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
261                     .addGroup(jPanel6Layout.createSequentialGroup()
262                         .addComponent(jCheckBox2)
263                         .addGap(18, 18, 18)
264                         .addComponent(jCheckBox3)
265                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
266                         .addComponent(jLabel7)
267                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
268                         .addComponent(jComboBox2, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE))
269                     .addGroup(jPanel6Layout.createSequentialGroup()
270                         .addComponent(jLabel9)
271                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
272                         .addComponent(jScrollPane6))
273                     .addGroup(jPanel6Layout.createSequentialGroup()
274                         .addComponent(jLabel8)
275                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
276                         .addComponent(jComboBox6, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE)))
277                 .addContainerGap())
278         );
279         jPanel6Layout.setVerticalGroup(
280             jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
281             .addGroup(jPanel6Layout.createSequentialGroup()
282                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
283                     .addComponent(jCheckBox2)
284                     .addComponent(jCheckBox3)
285                     .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing
    .GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
286                     .addComponent(jLabel7))
287                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
288                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
289                     .addComponent(jComboBox6, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
290                     .addComponent(jLabel8))
291                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
292                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
293                     .addGroup(jPanel6Layout.createSequentialGroup()
294                         .addComponent(jLabel9)
295                         .addGap(0, 29, Short.MAX_VALUE))
296                     .addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
    Short.MAX_VALUE))
297                 .addContainerGap())
298         );
299
300         jButton9.setText("Guardar");
301
302         jButton10.setText("Nuevo Procedimiento");
303
304         javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(
    jPanel4);
305         jPanel4.setLayout(jPanel4Layout);
306         jPanel4Layout.setHorizontalGroup(
307             jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
308             .addGroup(jPanel4Layout.createSequentialGroup()
309                 .addContainerGap()
310                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
311                     .addComponent(jPanel3, javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
312                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel4Layout.
    createSequentialGroup()
313                         .addComponent(jLabel2)
314                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
315                         .addComponent(jComboBox1, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
316                     .addGroup(jPanel4Layout.createSequentialGroup()
317                         .addComponent(jLabel1)
```

```
318                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
319                         .addComponent(jTextField1))
320                     .addGroup(jPanel4Layout.createSequentialGroup()
321                         .addComponent(jLabel5)
322                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
323                         .addComponent(jComboBox3, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
324                     .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
325                     .addComponent(jPanel6, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
326                     .addGroup(jPanel4Layout.createSequentialGroup()
327                         .addComponent(jLabel10)
328                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
329                         .addComponent(jButton10)
330                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
331                         .addComponent(jButton9)
332                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
333                         .addComponent(jButton4)
334                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
335                         .addComponent(jButton3)))
336                 .addContainerGap())
337         );
338         jPanel4Layout.setVerticalGroup(
339             jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
340             .addGroup(jPanel4Layout.createSequentialGroup()
341                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
342                     .addGroup(jPanel4Layout.createSequentialGroup()
343                         .addContainerGap()
344                         .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
345                             .addComponent(jLabel5)
346                             .addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
347                         .addGap(6, 6, 6)
348                         .addGroup(jPanel4Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.BASELINE)
349                             .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
350                             .addComponent(jLabel1))
351                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
352                         .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
353                             .addComponent(jLabel2)
354                             .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
355                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
356                         .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
357                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
358                         .addComponent(jPanel6, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
359                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
360                         .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
361                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
362                         .addGroup(jPanel4Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.CENTER)
363                             .addComponent(jButton10)
364                             .addComponent(jButton9)
365                             .addComponent(jButton4)
366                             .addComponent(jButton3)))
367                     .addGroup(jPanel4Layout.createSequentialGroup()
368                         .addGap(592, 592, 592)
369                         .addComponent(jLabel10)))
370                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
371         );
372 
373         jPanel5.setBorder(javax.swing.BorderFactory.createTitledBorder("Árbol de Tablas"));
374         jPanel5.setPreferredSize(new java.awt.Dimension(180, 424));
375 
376         javax.swing.tree.DefaultMutableTreeNode treeNode1 = new javax.swing.tree.DefaultMutableTreeNode("
    root");
377         jTree1.setModel(new javax.swing.tree.DefaultTreeModel(treeNode1));
378         jTree1.setAutoscrolls(true);
379         jTree1.setComponentPopupMenu(jPopupMenu1);
380         jTree1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
381         jTree1.setInheritsPopupMenu(true);
382         jScrollPane4.setViewportView(jTree1);
383 
```

```
384         jButton5.setText("Conectar a BD");
385 
386         javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(
    jPanel5);
387         jPanel5.setLayout(jPanel5Layout);
388         jPanel5Layout.setHorizontalGroup(
389             jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
390             .addGroup(jPanel5Layout.createSequentialGroup()
391                 .addContainerGap()
392                 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
393                     .addComponent(jButton5)
394                     .addComponent(jScrollPane4, javax.swing.GroupLayout.DEFAULT_SIZE, 147,
    Short.MAX_VALUE))
395                 .addContainerGap())
396         );
397         jPanel5Layout.setVerticalGroup(
398             jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
399             .addGroup(jPanel5Layout.createSequentialGroup()
400                 .addContainerGap()
401                 .addComponent(jScrollPane4, javax.swing.GroupLayout.DEFAULT_SIZE, 574, Short.
    MAX_VALUE)
402                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
403                 .addComponent(jButton5)
404                 .addGap(6, 6, 6))
405         );
406 
407         jPanel7.setBorder(javax.swing.BorderFactory.createTitledBorder("Árbol de Procedimientos"));
408         jPanel7.setPreferredSize(new java.awt.Dimension(180, 424));
409 
410         treeNode1 = new javax.swing.tree.DefaultMutableTreeNode("vacia");
411         jTree2.setModel(new javax.swing.tree.DefaultTreeModel(treeNode1));
412         jTree2.setRootVisible(false);
413         jScrollPane2.setViewportView(jTree2);
414 
415         javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(
    jPanel7);
416         jPanel7.setLayout(jPanel7Layout);
417         jPanel7Layout.setHorizontalGroup(
418             jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
419             .addGroup(jPanel7Layout.createSequentialGroup()
420                 .addContainerGap()
421                 .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 173, Short.
    MAX_VALUE)
422                 .addContainerGap())
423         );
424         jPanel7Layout.setVerticalGroup(
425             jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
426             .addGroup(jPanel7Layout.createSequentialGroup()
427                 .addContainerGap()
428                 .addComponent(jScrollPane2)
429                 .addContainerGap())
430         );
431 
432         jMenu1.setText("Proyecto");
433 
434         jMenuItem2.setText("Abrir");
435         jMenu1.add(jMenuItem2);
436 
437         jMenuItem3.setText("Guardar");
438         jMenu1.add(jMenuItem3);
439 
440         jMenuBar1.add(jMenu1);
441 
442         setJMenuBar(jMenuBar1);
443 
444         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
445         getContentPane().setLayout(layout);
446         layout.setHorizontalGroup(
447             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
448             .addGroup(layout.createSequentialGroup()
449                 .addContainerGap()
450                 .addComponent(jPanel7, javax.swing.GroupLayout.PREFERRED_SIZE, 209, javax.swing.
    GroupLayout.PREFERRED_SIZE)
451                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
452                 .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout
    .DEFAULT_SIZE, Short.MAX_VALUE)
453                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
454                 .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE, 183, Short.MAX_VALUE)
455                 .addContainerGap())
456         );
457         layout.setVerticalGroup(
```

```
458             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
459             .addGroup(layout.createSequentialGroup()
460                 .addContainerGap()
461                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
462                     .addComponent(jPanel7, javax.swing.GroupLayout.DEFAULT_SIZE, 648,
    Short.MAX_VALUE)
463                     .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
464                     .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE, 648,
    Short.MAX_VALUE))
465                 .addContainerGap())
466         );
467 
468         pack();
469     }// </editor-fold>//GEN-END:initComponents
```

Gráfico de llamadas para esta función:

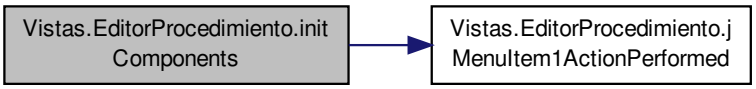
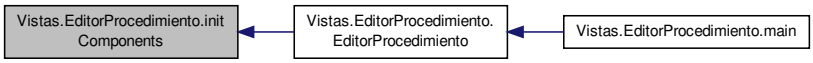


Gráfico de llamadas a esta función:

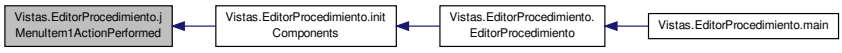


B.15.3.2. void Vistas.EditorProcedimiento.jMenuItem1ActionPerformed (java.awt.event.ActionEvent *evt*) [private]

Definición en la línea 471 del archivo EditorProcedimiento.java.

```
471                                                                         {//
    GEN-FIRST:event_jMenuItem1ActionPerformed
472         // TODO add your handling code here:
473     }//GEN-LAST:event_jMenuItem1ActionPerformed
```

Gráfico de llamadas a esta función:



B.15.3.3. static void Vistas.EditorProcedimiento.main (String *args[]*) [static]

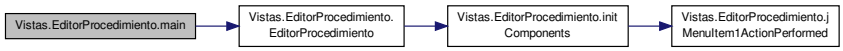
Parámetros

args	the command line arguments

Definición en la línea 478 del archivo EditorProcedimiento.java.

```
478                                         {
479         /* Set the Nimbus look and feel */
480         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
481         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
482          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
483          */
484         try {
485             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
    getInstalledLookAndFeels()) {
486                 if ("Nimbus".equals(info.getName())) {
487                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
488                     break;
489                 }
490             }
491         } catch (ClassNotFoundException ex) {
492             java.util.logging.Logger.getLogger(EditorProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
493         } catch (InstantiationException ex) {
494             java.util.logging.Logger.getLogger(EditorProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
495         } catch (IllegalAccessException ex) {
496             java.util.logging.Logger.getLogger(EditorProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
497         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
498             java.util.logging.Logger.getLogger(EditorProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
499         }
500         //</editor-fold>
501 
502         /* Create and display the form */
503         java.awt.EventQueue.invokeLater(new Runnable() {
504             public void run() {
505                 new EditorProcedimiento().setVisible(true);
506             }
507         });
508     }
```

Gráfico de llamadas para esta función:



B.15.4. Documentación de los datos miembro

B.15.4.1. javax.swing.JButton Vistas.EditorProcedimiento.jButton1

Definición en la línea 511 del archivo EditorProcedimiento.java.

B.15.4.2. javax.swing.JButton Vistas.EditorProcedimiento.jButton10

Definición en la línea 512 del archivo EditorProcedimiento.java.

B.15.4.3. javax.swing.JButton Vistas.EditorProcedimiento.jButton2

Definición en la línea 513 del archivo EditorProcedimiento.java.

B.15.4.4. javax.swing.JButton Vistas.EditorProcedimiento.jButton3

Definición en la línea 514 del archivo EditorProcedimiento.java.

B.15.4.5. javax.swing.JButton Vistas.EditorProcedimiento.jButton4

Definición en la línea 515 del archivo EditorProcedimiento.java.

B.15.4.6. javax.swing.JButton Vistas.EditorProcedimiento.jButton5

Definición en la línea 516 del archivo EditorProcedimiento.java.

B.15.4.7. javax.swing.JButton Vistas.EditorProcedimiento.jButton6

Definición en la línea 517 del archivo EditorProcedimiento.java.

B.15.4.8. javax.swing.JButton Vistas.EditorProcedimiento.jButton7

Definición en la línea 518 del archivo EditorProcedimiento.java.

B.15.4.9. javax.swing.JButton Vistas.EditorProcedimiento.jButton8

Definición en la línea 519 del archivo EditorProcedimiento.java.

B.15.4.10. javax.swing.JButton Vistas.EditorProcedimiento.jButton9

Definición en la línea 520 del archivo EditorProcedimiento.java.

B.15.4.11. javax.swing.JCheckBox Vistas.EditorProcedimiento.jCheckBox2

Definición en la línea 521 del archivo EditorProcedimiento.java.

B.15.4.12. javax.swing.JCheckBox Vistas.EditorProcedimiento.jCheckBox3

Definición en la línea 522 del archivo EditorProcedimiento.java.

B.15.4.13. javax.swing.JComboBox Vistas.EditorProcedimiento.jComboBox1

Definición en la línea 523 del archivo EditorProcedimiento.java.

B.15.4.14. javax.swing.JComboBox Vistas.EditorProcedimiento.jComboBox2

Definición en la línea 524 del archivo EditorProcedimiento.java.

B.15.4.15. javax.swing.JComboBox Vistas.EditorProcedimiento.jComboBox3

Definición en la línea 525 del archivo EditorProcedimiento.java.

B.15.4.16. javax.swing.JComboBox Vistas.EditorProcedimiento.jComboBox4

Definición en la línea 526 del archivo EditorProcedimiento.java.

B.15.4.17. javax.swing.JComboBox Vistas.EditorProcedimiento.jComboBox6

Definición en la línea 527 del archivo EditorProcedimiento.java.

B.15.4.18. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel1

Definición en la línea 528 del archivo EditorProcedimiento.java.

B.15.4.19. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel10

Definición en la línea 529 del archivo EditorProcedimiento.java.

B.15.4.20. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel2

Definición en la línea 530 del archivo EditorProcedimiento.java.

B.15.4.21. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel3

Definición en la línea 531 del archivo EditorProcedimiento.java.

B.15.4.22. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel4

Definición en la línea 532 del archivo EditorProcedimiento.java.

B.15.4.23. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel5

Definición en la línea 533 del archivo EditorProcedimiento.java.

B.15.4.24. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel6

Definición en la línea 534 del archivo EditorProcedimiento.java.

B.15.4.25. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel7

Definición en la línea 535 del archivo EditorProcedimiento.java.

B.15.4.26. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel8

Definición en la línea 536 del archivo EditorProcedimiento.java.

B.15.4.27. javax.swing.JLabel Vistas.EditorProcedimiento.jLabel9

Definición en la línea 537 del archivo EditorProcedimiento.java.

B.15.4.28. javax.swing.JList Vistas.EditorProcedimiento.jList1

Definición en la línea 538 del archivo EditorProcedimiento.java.

B.15.4.29. javax.swing.JMenu Vistas.EditorProcedimiento.jMenu1

Definición en la línea 539 del archivo EditorProcedimiento.java.

B.15.4.30. javax.swing.JMenuBar Vistas.EditorProcedimiento.jMenuBar1

Definición en la línea 540 del archivo EditorProcedimiento.java.

B.15.4.31. javax.swing.JMenuItem Vistas.EditorProcedimiento.jMenuItem1

Definición en la línea 541 del archivo EditorProcedimiento.java.

B.15.4.32. javax.swing.JMenuItem Vistas.EditorProcedimiento.jMenuItem2

Definición en la línea 542 del archivo EditorProcedimiento.java.

B.15.4.33. javax.swing.JMenuItem Vistas.EditorProcedimiento.jMenuItem3

Definición en la línea 543 del archivo EditorProcedimiento.java.

B.15.4.34. javax.swing.JPanel Vistas.EditorProcedimiento.jPanel2

Definición en la línea 544 del archivo EditorProcedimiento.java.

B.15.4.35. javax.swing.JPanel Vistas.EditorProcedimiento.jPanel3

Definición en la línea 545 del archivo EditorProcedimiento.java.

B.15.4.36. javax.swing.JPanel Vistas.EditorProcedimiento.jPanel4

Definición en la línea 546 del archivo EditorProcedimiento.java.

B.15.4.37. javax.swing.JPanel Vistas.EditorProcedimiento.jPanel5

Definición en la línea 547 del archivo EditorProcedimiento.java.

B.15.4.38. javax.swing.JPanel Vistas.EditorProcedimiento.jPanel6

Definición en la línea 548 del archivo EditorProcedimiento.java.

B.15.4.39. javax.swing.JPanel Vistas.EditorProcedimiento.jPanel7

Definición en la línea 549 del archivo EditorProcedimiento.java.

B.15.4.40. javax.swing.JPopupMenu Vistas.EditorProcedimiento.jPopupMenu1

Definición en la línea 550 del archivo EditorProcedimiento.java.

B.15.4.41. javax.swing.JScrollPane Vistas.EditorProcedimiento.jScrollPane1

Definición en la línea 551 del archivo EditorProcedimiento.java.

B.15.4.42. javax.swing.JScrollPane Vistas.EditorProcedimiento.jScrollPane2

Definición en la línea 552 del archivo EditorProcedimiento.java.

B.15.4.43. javax.swing.JScrollPane Vistas.EditorProcedimiento.jScrollPane3

Definición en la línea 553 del archivo EditorProcedimiento.java.

B.15.4.44. javax.swing.JScrollPane Vistas.EditorProcedimiento.jScrollPane4

Definición en la línea 554 del archivo EditorProcedimiento.java.

B.15.4.45. javax.swing.JScrollPane Vistas.EditorProcedimiento.jScrollPane6

Definición en la línea 555 del archivo EditorProcedimiento.java.

B.15.4.46. javax.swing.JSeparator Vistas.EditorProcedimiento.jSeparator1

Definición en la línea 556 del archivo EditorProcedimiento.java.

B.15.4.47. javax.swing.JTextArea Vistas.EditorProcedimiento.jTextArea2

Definición en la línea 557 del archivo EditorProcedimiento.java.

B.15.4.48. javax.swing.JTextArea Vistas.EditorProcedimiento.jTextArea3

Definición en la línea 558 del archivo EditorProcedimiento.java.

B.15.4.49. javax.swing.JTextField Vistas.EditorProcedimiento.jTextField1

Definición en la línea 559 del archivo EditorProcedimiento.java.

B.15.4.50. javax.swing.JTextField Vistas.EditorProcedimiento.jTextField2

Definición en la línea 560 del archivo EditorProcedimiento.java.

B.15.4.51. javax.swing.JTree Vistas.EditorProcedimiento.jTree1

Definición en la línea 561 del archivo EditorProcedimiento.java.

B.15.4.52. javax.swing.JTree Vistas.EditorProcedimiento.jTree2

Definición en la línea 562 del archivo EditorProcedimiento.java.

B.15.4.53. javax.swing.JComboBox Vistas.EditorProcedimiento.tipoDatoSQL

Definición en la línea 563 del archivo EditorProcedimiento.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vistas/EditorProcedimiento.java

B.16. Referencia de la Clase Vistas.EditorProcedimientoCRUD

Diagrama de herencias de Vistas.EditorProcedimientoCRUD

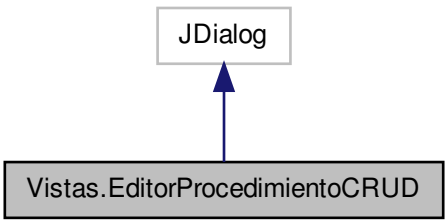
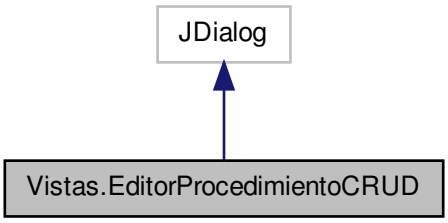


Diagrama de colaboración para Vistas.EditorProcedimientoCRUD:



Métodos públicos

- EditorProcedimientoCRUD (java.awt.Frame parent, boolean modal)

Métodos públicos estáticos

- static void main (String args[])

Atributos públicos

- javax.swing.ButtonGroup buttonGroup2
- javax.swing.JButton jButton1
- javax.swing.JButton jButton2
- javax.swing.JButton jButton3
- javax.swing.JButton jButton4
- javax.swing.JButton jButton5
- javax.swing.JButton jButton7
- javax.swing.JButton jButton8
- javax.swing.JCheckBox jCheckBox2
- javax.swing.JCheckBox jCheckBox3
- javax.swing.JComboBox jComboBox2
- javax.swing.JComboBox jComboBox4
- javax.swing.JComboBox jComboBox6
- javax.swing.JLabel jLabel1
- javax.swing.JLabel jLabel10
- javax.swing.JLabel jLabel3
- javax.swing.JLabel jLabel4
- javax.swing.JLabel jLabel6
- javax.swing.JLabel jLabel7
- javax.swing.JLabel jLabel8
- javax.swing.JLabel jLabel9
- javax.swing.JList jList1
- javax.swing.JPanel jPanel2

- javax.swing.JPanel jPanel3
- javax.swing.JPanel jPanel4
- javax.swing.JPanel jPanel5
- javax.swing.JPanel jPanel6
- javax.swing.JRadioButton jRadioButton1
- javax.swing.JRadioButton jRadioButton2
- javax.swing.JRadioButton jRadioButton3
- javax.swing.JRadioButton jRadioButton4
- javax.swing.JScrollPane jScrollPane1
- javax.swing.JScrollPane jScrollPane3
- javax.swing.JScrollPane jScrollPane4
- javax.swing.JScrollPane jScrollPane6
- javax.swing.JSeparator jSeparator1
- javax.swing.JTextArea jTextArea2
- javax.swing.JTextArea jTextArea3
- javax.swing.JTextField jTextField1
- javax.swing.JTextField jTextField2
- javax.swing.JTree jTree1
- javax.swing.JComboBox tipoDatoSQL

Métodos privados

- void initComponents ()

Atributos privados

- javax.swing.JPanel jPanel1

B.16.1. Descripción detallada

Autor

ivan

Definición en la línea 13 del archivo EditorProcedimientoCRUD.java.

B.16.2. Documentación del constructor y destructor

B.16.2.1. Vistas.EditorProcedimientoCRUD.EditorProcedimientoCRUD (java.awt.Frame *parent,* boolean *modal*)

Creates new form SPEditorFuncionCRUD

Definición en la línea 18 del archivo EditorProcedimientoCRUD.java.

```
18                                                                      {
19         super(parent, modal);
20         initComponents();
21     }
```

Gráfico de llamadas para esta función:

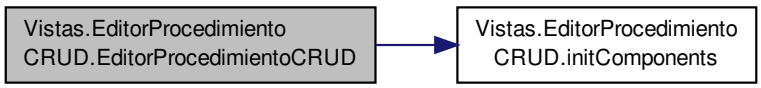
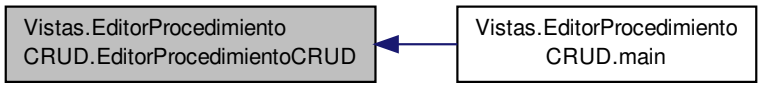


Gráfico de llamadas a esta función:



B.16.3. Documentación de las funciones miembro

B.16.3.1. void Vistas.EditorProcedimientoCRUD.initComponents ( ) [private]

This method is called from within the constructor to initialize the form. WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

Definición en la línea 30 del archivo EditorProcedimientoCRUD.java.

```
30                                {
31
32         buttonGroup2 = new javax.swing.ButtonGroup();
33         jPanel4 = new javax.swing.JPanel();
34         jPanel3 = new javax.swing.JPanel();
35         jScrollPane3 = new javax.swing.JScrollPane();
36         jTextArea2 = new javax.swing.JTextArea();
37         jPanel2 = new javax.swing.JPanel();
38         jLabel3 = new javax.swing.JLabel();
39         jTextField2 = new javax.swing.JTextField();
40         jLabel4 = new javax.swing.JLabel();
41         jButton2 = new javax.swing.JButton();
42         jButton1 = new javax.swing.JButton();
43         jLabel6 = new javax.swing.JLabel();
44         jComboBox4 = new javax.swing.JComboBox();
45         jSeparator1 = new javax.swing.JSeparator();
46         jScrollPane1 = new javax.swing.JScrollPane();
47         jList1 = new javax.swing.JList();
48         jButton7 = new javax.swing.JButton();
49         jButton8 = new javax.swing.JButton();
50         tipoDatoSQL = new javax.swing.JComboBox();
51         jLabel1 = new javax.swing.JLabel();
52         jTextField1 = new javax.swing.JTextField();
53         jButton4 = new javax.swing.JButton();
54         jButton3 = new javax.swing.JButton();
55         jLabel10 = new javax.swing.JLabel();
56         jPanel6 = new javax.swing.JPanel();
57         jCheckBox2 = new javax.swing.JCheckBox();
58         jCheckBox3 = new javax.swing.JCheckBox();
59         jComboBox2 = new javax.swing.JComboBox();
60         jComboBox6 = new javax.swing.JComboBox();
```

```
61          jScrollPane6 = new javax.swing.JScrollPane();
62          jTextArea3 = new javax.swing.JTextArea();
63          jLabel7 = new javax.swing.JLabel();
64          jLabel8 = new javax.swing.JLabel();
65          jLabel9 = new javax.swing.JLabel();
66          jPanel1 = new javax.swing.JPanel();
67          jRadioButton1 = new javax.swing.JRadioButton();
68          jRadioButton2 = new javax.swing.JRadioButton();
69          jRadioButton3 = new javax.swing.JRadioButton();
70          jRadioButton4 = new javax.swing.JRadioButton();
71          jButton5 = new javax.swing.JButton();
72          jPanel5 = new javax.swing.JPanel();
73          jScrollPane4 = new javax.swing.JScrollPane();
74          jTree1 = new javax.swing.JTree();
75
76          setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
77          setFocusTraversalPolicyProvider(true);
78
79          jPanel4.setBorder(javax.swing.BorderFactory.createTitledBorder("Procedimiento CRUD"));
80
81          jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder("*cuerpo rutina"));
82
83          jTextArea2.setColumns(20);
84          jTextArea2.setRows(5);
85          jTextArea2.setText("# agrega todo tu codigo de control aquí: cualquier tipo de datos MySQL valido")
    ;
86          jScrollPane3.setViewportView(jTextArea2);
87
88          javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(
    jPanel3);
89          jPanel3.setLayout(jPanel3Layout);
90          jPanel3Layout.setHorizontalGroup(
91              jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
92              .addGroup(jPanel3Layout.createSequentialGroup()
93                  .addComponent(jScrollPane3)
94                  .addContainerGap())
95          );
96          jPanel3Layout.setVerticalGroup(
97              jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
98              .addGroup(jPanel3Layout.createSequentialGroup()
99                  .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
100                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
101         );
102
103         jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("parametros"));
104
105         jLabel3.setText("nombre del parametro:");
106
107         jLabel4.setText("tipo de dato:");
108
109         jButton2.setText("»");
110
111         jButton1.setText("«");
112
113         jLabel6.setText("tipo de entrada");
114
115         jComboBox4.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "IN", "OUT", "INOUT" }));
116
117         jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);
118
119         jScrollPane1.setViewportView(jList1);
120
121         jButton7.setText("borrar");
122         jButton7.setMaximumSize(new java.awt.Dimension(26, 27));
123         jButton7.setMinimumSize(new java.awt.Dimension(26, 27));
124         jButton7.setPreferredSize(new java.awt.Dimension(26, 27));
125
126         jButton8.setText("guardar cambios");
127         jButton8.setEnabled(false);
128
129         javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(
    jPanel2);
130         jPanel2.setLayout(jPanel2Layout);
131         jPanel2Layout.setHorizontalGroup(
132             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
133             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
134                 .addContainerGap()
135                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
136                     .addGroup(jPanel2Layout.createSequentialGroup()
137                         .addComponent(jLabel3)
```

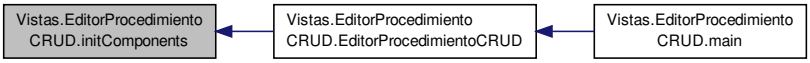
```
138                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
139                         .addComponent(jTextField2))
140                     .addGroup(jPanel2Layout.createSequentialGroup()
141                         .addComponent(jLabel4)
142                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
143                         .addComponent(tipoDatoSQL, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
144                     .addGroup(jPanel2Layout.createSequentialGroup()
145                         .addComponent(jLabel6)
146                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
147                         .addComponent(jComboBox4, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE))
148                     .addGroup(jPanel2Layout.createSequentialGroup()
149                         .addGap(0, 0, Short.MAX_VALUE)
150                         .addComponent(jButton8)))
151                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
152                 .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 11, javax.
    swing.GroupLayout.PREFERRED_SIZE)
153                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
154                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
155                     .addComponent(jButton1)
156                     .addComponent(jButton2)
157                     .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
    javax.swing.GroupLayout.PREFERRED_SIZE))
158                 .addGap(18, 18, 18)
159                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 139, javax.
    swing.GroupLayout.PREFERRED_SIZE)
160                 .addContainerGap())
161         );
162         jPanel2Layout.setVerticalGroup(
163             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
164             .addGroup(jPanel2Layout.createSequentialGroup()
165                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
166                     .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 111,
    javax.swing.GroupLayout.PREFERRED_SIZE)
167                     .addGroup(jPanel2Layout.createSequentialGroup()
168                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
169                             .addComponent(jLabel6)
170                             .addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
171                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
172                         .addGroup(jPanel2Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.BASELINE)
173                             .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
174                             .addComponent(jLabel3))
175                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
176                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
177                             .addComponent(jLabel4)
178                             .addComponent(tipoDatoSQL, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
179                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
180                         .addComponent(jButton8))
181                     .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
     false)
182                         .addGroup(jPanel2Layout.createSequentialGroup()
183                             .addComponent(jButton2)
184                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
185                             .addComponent(jButton1)
186                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.
    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
187                             .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
188                         .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
    javax.swing.GroupLayout.PREFERRED_SIZE, 90, javax.swing.GroupLayout.PREFERRED_SIZE)))
189                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
190         );
191 
192         jLabel1.setText("*nombre del procedimiento");
193 
194         jButton4.setText("Vista previa");
195 
196         jButton3.setText("Generar código");
197 
198         jLabel10.setText("* Campos obligatorios");
199 
200         jPanel6.setBorder(javax.swing.BorderFactory.createTitledBorder(
    javax.swing.BorderFactory.createTitledBorder("caracteristicas")));
201 
```

```
202         jCheckBox2.setText("LANGUAGE SQL");
203         jCheckBox2.setEnabled(false);
204 
205         jCheckBox3.setText("DETERMINISTIC");
206 
207         jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "DEFINER", "INVOKER" }));
208 
209         jComboBox6.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "CONTAINS SQL", "NO SQL", "
    READS SQL DATA", "MODIFIES SQL DATA" }));
210 
211         jTextArea3.setColumns(20);
212         jTextArea3.setRows(5);
213         jTextArea3.setText("Sin comentarios");
214         jScrollPane6.setViewportView(jTextArea3);
215 
216         jLabel7.setText("SQL SECURITY:");
217 
218         jLabel8.setText("información del manejo de datos:");
219 
220         jLabel9.setText("COMENT:");
221 
222         javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(
    jPanel6);
223         jPanel6.setLayout(jPanel6Layout);
224         jPanel6Layout.setHorizontalGroup(
225             jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
226             .addGroup(jPanel6Layout.createSequentialGroup()
227                 .addGap(6, 6, 6)
228                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
229                     .addGroup(jPanel6Layout.createSequentialGroup()
230                         .addComponent(jCheckBox2)
231                         .addGap(18, 18, 18)
232                         .addComponent(jCheckBox3)
233                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
234                         .addComponent(jLabel7)
235                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
236                         .addComponent(jComboBox2, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE))
237                     .addGroup(jPanel6Layout.createSequentialGroup()
238                         .addComponent(jLabel9)
239                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
240                         .addComponent(jScrollPane6))
241                     .addGroup(jPanel6Layout.createSequentialGroup()
242                         .addComponent(jLabel8)
243                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
244                         .addComponent(jComboBox6, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE)))
245                 .addContainerGap())
246         );
247         jPanel6Layout.setVerticalGroup(
248             jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
249             .addGroup(jPanel6Layout.createSequentialGroup()
250                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
251                     .addComponent(jCheckBox2)
252                     .addComponent(jCheckBox3)
253                     .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing
    .GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
254                     .addComponent(jLabel7))
255                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
256                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
257                     .addComponent(jComboBox6, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
258                     .addComponent(jLabel8))
259                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
260                 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
261                     .addComponent(jLabel9)
262                     .addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
    javax.swing.GroupLayout.PREFERRED_SIZE))
263                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
264         );
265 
266         jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Funciones basicas"));
267         jPanel1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
268 
269         buttonGroup2.add(jRadioButton1);
270         jRadioButton1.setSelected(true);
271         jRadioButton1.setText("ALTA");
272 
273         buttonGroup2.add(jRadioButton2);
274         jRadioButton2.setText("BAJA");
275 
```

```
276         buttonGroup2.add(jRadioButton3);
277         jRadioButton3.setText("CAMBIO");
278 
279         buttonGroup2.add(jRadioButton4);
280         jRadioButton4.setText("CONSULTA");
281 
282         javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(
    jPanel1);
283         jPanel1.setLayout(jPanel1Layout);
284         jPanel1Layout.setHorizontalGroup(
285             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
286             .addGroup(jPanel1Layout.createSequentialGroup()
287                 .addContainerGap()
288                 .addComponent(jRadioButton1, javax.swing.GroupLayout.DEFAULT_SIZE, 54, Short.
    MAX_VALUE)
289                 .addGap(58, 58, 58)
290                 .addComponent(jRadioButton2, javax.swing.GroupLayout.DEFAULT_SIZE, 54, Short.
    MAX_VALUE)
291                 .addGap(83, 83, 83)
292                 .addComponent(jRadioButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 72, Short.
    MAX_VALUE)
293                 .addGap(59, 59, 59)
294                 .addComponent(jRadioButton4, javax.swing.GroupLayout.DEFAULT_SIZE, 87, Short.
    MAX_VALUE)
295                 .addContainerGap())
296         );
297         jPanel1Layout.setVerticalGroup(
298             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
299             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
300                 .addGap(0, 0, 0)
301                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
302                     .addComponent(jRadioButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
303                     .addComponent(jRadioButton3, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
304                     .addComponent(jRadioButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
305                     .addComponent(jRadioButton1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
306         );
307 
308         jButton5.setText("Finalizar");
309 
310         javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(
    jPanel4);
311         jPanel4.setLayout(jPanel4Layout);
312         jPanel4Layout.setHorizontalGroup(
313             jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
314             .addGroup(jPanel4Layout.createSequentialGroup()
315                 .addContainerGap()
316                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
317                     .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
318                     .addComponent(jPanel3, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.
    GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
319                     .addGroup(jPanel4Layout.createSequentialGroup()
320                         .addComponent(jLabel1)
321                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
322                         .addComponent(jTextField1))
323                     .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
324                     .addComponent(jPanel6, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
325                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel4Layout.
    createSequentialGroup()
326                         .addComponent(jLabel10)
327                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
328                         .addComponent(jButton5)
329                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
330                         .addComponent(jButton4)
331                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
332                         .addComponent(jButton3)))
333                 .addContainerGap())
334         );
335         jPanel4Layout.setVerticalGroup(
336             jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
337             .addGroup(jPanel4Layout.createSequentialGroup()
338                 .addContainerGap()
339                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
340                     .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
341                     .addComponent(jLabel1))
342                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
343                 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
344                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
345                 .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
346                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
347                 .addComponent(jPanel6, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
348                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
349                 .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
350                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
351                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
352                     .addComponent(jButton3)
353                     .addComponent(jButton4)
354                     .addComponent(jLabel10)
355                     .addComponent(jButton5))
356                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
357         );
358 
359         jPanel5.setBorder(javax.swing.BorderFactory.createTitledBorder("Arbol de Tablas"));
360         jPanel5.setPreferredSize(new java.awt.Dimension(180, 518));
361 
362         javax.swing.tree.DefaultMutableTreeNode treeNode1 = new javax.swing.tree.DefaultMutableTreeNode("
    vacía");
363         jTree1.setModel(new javax.swing.tree.DefaultTreeModel(treeNode1));
364         jScrollPane4.setViewportView(jTree1);
365 
366         javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(
    jPanel5);
367         jPanel5.setLayout(jPanel5Layout);
368         jPanel5Layout.setHorizontalGroup(
369             jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
370             .addGroup(jPanel5Layout.createSequentialGroup()
371                 .addContainerGap()
372                 .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 152, javax.
    swing.GroupLayout.PREFERRED_SIZE)
373                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
374         );
375         jPanel5Layout.setVerticalGroup(
376             jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
377             .addGroup(jPanel5Layout.createSequentialGroup()
378                 .addContainerGap()
379                 .addComponent(jScrollPane4)
380                 .addContainerGap())
381         );
382 
383         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
384         getContentPane().setLayout(layout);
385         layout.setHorizontalGroup(
386             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
387             .addGroup(layout.createSequentialGroup()
388                 .addContainerGap()
389                 .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
390                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
391                 .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE, 188, javax.swing.
    GroupLayout.PREFERRED_SIZE)
392                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
393         );
394         layout.setVerticalGroup(
395             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
396             .addGroup(layout.createSequentialGroup()
397                 .addGap(6, 6, 6)
398                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
399                     .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
400                     .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE, 601, Short.
    MAX_VALUE))
401                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
402         );
403 
404         pack();
405     }// </editor-fold>//GEN-END:initComponents
```

Gráfico de llamadas a esta función:



B.16.3.2. static void Vistas.EditorProcedimientoCRUD.main (String *args[]*) [static]

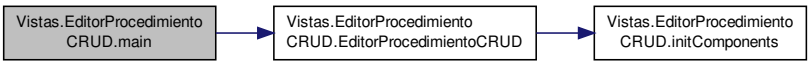
Parámetros

args	the command line arguments

Definición en la línea 410 del archivo EditorProcedimientoCRUD.java.

```
410                                         {
411         /* Set the Nimbus look and feel */
412         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
413         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
414          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
415          */
416         try {
417             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
    getInstalledLookAndFeels()) {
418                 if ("Nimbus".equals(info.getName())) {
419                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
420                     break;
421                 }
422             }
423         } catch (ClassNotFoundException ex) {
424             java.util.logging.Logger.getLogger(EditorProcedimientoCRUD.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
425         } catch (InstantiationException ex) {
426             java.util.logging.Logger.getLogger(EditorProcedimientoCRUD.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
427         } catch (IllegalAccessException ex) {
428             java.util.logging.Logger.getLogger(EditorProcedimientoCRUD.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
429         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
430             java.util.logging.Logger.getLogger(EditorProcedimientoCRUD.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
431         }
432         //</editor-fold>
433 
434         /* Create and display the dialog */
435         java.awt.EventQueue.invokeLater(new Runnable() {
436             public void run() {
437                 EditorProcedimientoCRUD dialog = new
    EditorProcedimientoCRUD(new javax.swing.JFrame(), true);
438                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
439                     @Override
440                     public void windowClosing(java.awt.event.WindowEvent e) {
441                         System.exit(0);
442                     }
443                 });
444                 dialog.setVisible(true);
445             }
446         });
447     }
```

Gráfico de llamadas para esta función:



B.16.4. Documentación de los datos miembro

B.16.4.1. javax.swing.ButtonGroup Vistas.EditorProcedimientoCRUD.buttonGroup2

Definición en la línea 450 del archivo EditorProcedimientoCRUD.java.

B.16.4.2. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton1

Definición en la línea 451 del archivo EditorProcedimientoCRUD.java.

B.16.4.3. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton2

Definición en la línea 452 del archivo EditorProcedimientoCRUD.java.

B.16.4.4. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton3

Definición en la línea 453 del archivo EditorProcedimientoCRUD.java.

B.16.4.5. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton4

Definición en la línea 454 del archivo EditorProcedimientoCRUD.java.

B.16.4.6. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton5

Definición en la línea 455 del archivo EditorProcedimientoCRUD.java.

B.16.4.7. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton7

Definición en la línea 456 del archivo EditorProcedimientoCRUD.java.

B.16.4.8. javax.swing.JButton Vistas.EditorProcedimientoCRUD.jButton8

Definición en la línea 457 del archivo EditorProcedimientoCRUD.java.

B.16.4.9. javax.swing.JCheckBox Vistas.EditorProcedimientoCRUD.jCheckBox2

Definición en la línea 458 del archivo EditorProcedimientoCRUD.java.

B.16.4.10. javax.swing.JCheckBox Vistas.EditorProcedimientoCRUD.jCheckBox3

Definición en la línea 459 del archivo EditorProcedimientoCRUD.java.

B.16.4.11. javax.swing.JComboBox Vistas.EditorProcedimientoCRUD.jComboBox2

Definición en la línea 460 del archivo EditorProcedimientoCRUD.java.

B.16.4.12. javax.swing.JComboBox Vistas.EditorProcedimientoCRUD.jComboBox4

Definición en la línea 461 del archivo EditorProcedimientoCRUD.java.

B.16.4.13. javax.swing.JComboBox Vistas.EditorProcedimientoCRUD.jComboBox6

Definición en la línea 462 del archivo EditorProcedimientoCRUD.java.

B.16.4.14. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel1

Definición en la línea 463 del archivo EditorProcedimientoCRUD.java.

B.16.4.15. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel10

Definición en la línea 464 del archivo EditorProcedimientoCRUD.java.

B.16.4.16. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel3

Definición en la línea 465 del archivo EditorProcedimientoCRUD.java.

B.16.4.17. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel4

Definición en la línea 466 del archivo EditorProcedimientoCRUD.java.

B.16.4.18. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel6

Definición en la línea 467 del archivo EditorProcedimientoCRUD.java.

B.16.4.19. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel7

Definición en la línea 468 del archivo EditorProcedimientoCRUD.java.

B.16.4.20. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel8

Definición en la línea 469 del archivo EditorProcedimientoCRUD.java.

B.16.4.21. javax.swing.JLabel Vistas.EditorProcedimientoCRUD.jLabel9

Definición en la línea 470 del archivo EditorProcedimientoCRUD.java.

B.16.4.22. javax.swing.JList Vistas.EditorProcedimientoCRUD.jList1

Definición en la línea 471 del archivo EditorProcedimientoCRUD.java.

B.16.4.23. javax.swing.JPanel Vistas.EditorProcedimientoCRUD.jPanel1 [private]

Definición en la línea 472 del archivo EditorProcedimientoCRUD.java.

B.16.4.24. javax.swing.JPanel Vistas.EditorProcedimientoCRUD.jPanel2

Definición en la línea 473 del archivo EditorProcedimientoCRUD.java.

B.16.4.25. javax.swing.JPanel Vistas.EditorProcedimientoCRUD.jPanel3

Definición en la línea 474 del archivo EditorProcedimientoCRUD.java.

B.16.4.26. javax.swing.JPanel Vistas.EditorProcedimientoCRUD.jPanel4

Definición en la línea 475 del archivo EditorProcedimientoCRUD.java.

B.16.4.27. javax.swing.JPanel Vistas.EditorProcedimientoCRUD.jPanel5

Definición en la línea 476 del archivo EditorProcedimientoCRUD.java.

B.16.4.28. javax.swing.JPanel Vistas.EditorProcedimientoCRUD.jPanel6

Definición en la línea 477 del archivo EditorProcedimientoCRUD.java.

B.16.4.29. javax.swing.JRadioButton Vistas.EditorProcedimientoCRUD.jRadioButton1

Definición en la línea 478 del archivo EditorProcedimientoCRUD.java.

B.16.4.30. javax.swing.JRadioButton Vistas.EditorProcedimientoCRUD.jRadioButton2

Definición en la línea 479 del archivo EditorProcedimientoCRUD.java.

B.16.4.31. javax.swing.JRadioButton Vistas.EditorProcedimientoCRUD.jRadioButton3

Definición en la línea 480 del archivo EditorProcedimientoCRUD.java.

B.16.4.32. javax.swing.JRadioButton Vistas.EditorProcedimientoCRUD.jRadioButton4

Definición en la línea 481 del archivo EditorProcedimientoCRUD.java.

B.16.4.33. javax.swing.JScrollPane Vistas.EditorProcedimientoCRUD.jScrollPane1

Definición en la línea 482 del archivo EditorProcedimientoCRUD.java.

B.16.4.34. javax.swing.JScrollPane Vistas.EditorProcedimientoCRUD.jScrollPane3

Definición en la línea 483 del archivo EditorProcedimientoCRUD.java.

B.16.4.35. javax.swing.JScrollPane Vistas.EditorProcedimientoCRUD.jScrollPane4

Definición en la línea 484 del archivo EditorProcedimientoCRUD.java.

B.16.4.36. javax.swing.JScrollPane Vistas.EditorProcedimientoCRUD.jScrollPane6

Definición en la línea 485 del archivo EditorProcedimientoCRUD.java.

B.16.4.37. javax.swing.JSeparator Vistas.EditorProcedimientoCRUD.jSeparator1

Definición en la línea 486 del archivo EditorProcedimientoCRUD.java.

B.16.4.38. javax.swing.JTextArea Vistas.EditorProcedimientoCRUD.jTextArea2

Definición en la línea 487 del archivo EditorProcedimientoCRUD.java.

B.16.4.39. javax.swing.JTextArea Vistas.EditorProcedimientoCRUD.jTextArea3

Definición en la línea 488 del archivo EditorProcedimientoCRUD.java.

B.16.4.40. javax.swing.JTextField Vistas.EditorProcedimientoCRUD.jTextField1

Definición en la línea 489 del archivo EditorProcedimientoCRUD.java.

B.16.4.41. javax.swing.JTextField Vistas.EditorProcedimientoCRUD.jTextField2

Definición en la línea 490 del archivo EditorProcedimientoCRUD.java.

B.16.4.42. javax.swing.JTree Vistas.EditorProcedimientoCRUD.jTree1

Definición en la línea 491 del archivo EditorProcedimientoCRUD.java.

B.16.4.43. javax.swing.JComboBox Vistas.EditorProcedimientoCRUD.tipoDatoSQL

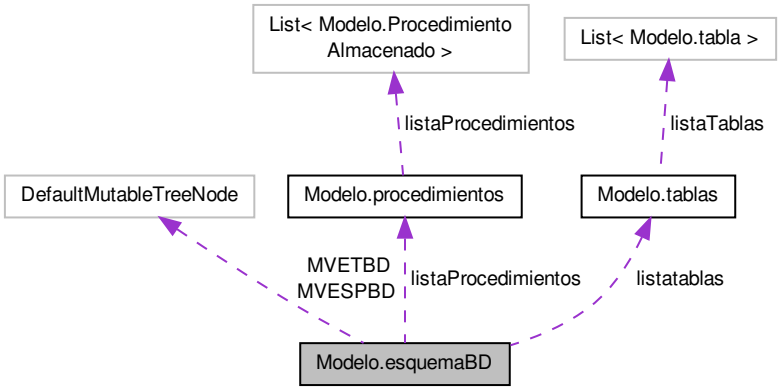
Definición en la línea 492 del archivo EditorProcedimientoCRUD.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vistas/EditorProcedimientoCRUD.java

B.17. Referencia de la Clase Modelo.esquemaBD

Diagrama de colaboración para Modelo.esquemaBD:



Métodos públicos

- esquemaBD ()
- esquemaBD (String nombreDB)
- esquemaBD (String nombreBD, tablas listaTablas)
- void setCatalogo (String catalogo)
- String getCatalgo ()
- void setTablas (tablas Ts)
- tablas getTablas ()
- DefaultMutableTreeNode getDMTTablas ()
- DefaultMutableTreeNode getMVETBD ()
- void setMVETBD (String nombreBD)
- void setMVETBD (DefaultMutableTreeNode arbolTablas)
- void vaciarMVETBD ()
- boolean MVETBDvacia ()
- void agregarTabla (tabla T)

Atributos privados

- String catalogo
- tablas listatablas
- procedimientos listaProcedimientos
- DefaultMutableTreeNode MVETBD

B.17.1. Descripción detallada

La clase esquemaBD define a un esquema de base de datos de forma básica.

Autor

ivan

Definición en la línea 15 del archivo esquemaBD.java.

B.17.2. Documentación del constructor y destructor

B.17.2.1. Modelo.esquemaBD.esquemaBD ( )

Constructor del esquema de una base de datos

Definición en la línea 37 del archivo esquemaBD.java.

```
37                          {
38     }
```

B.17.2.2. Modelo.esquemaBD.esquemaBD (String *nombreDB*)

Sobrecarga del constructor del esquema de una base de datos, inicializa al árbol de tablas con el nombre de la base de datos como nodo raíz.

Parámetros

nombreBD	nombre de la base de datos.

Definición en la línea 46 del archivo esquemaBD.java.

```
46                                      {
47         this.setMVETBD(nombreDB);
48     }
```

B.17.2.3. Modelo.esquemaBD.esquemaBD (String *nombreBD,* tablas *listaTablas*)

Sobrecarga del constructor del esquema de una base de datos, inicializa al árbol de tablas con el nombre de la base de datos como nodo raíz. Inicializa el nombre de la base de datos del esquema. Inicializa la lista de las tablas.

Parámetros

nombreBD	nombre de la base de datos
listaTablas	lista de tablas

Definición en la línea 59 del archivo esquemaBD.java.

```
59                                                          {
60         this.setMVETBD(nombreBD);
61         this.setCatalogo(nombreBD);
62         this.setTablas(listaTablas);
63     }
```

B.17.3. Documentación de las funciones miembro

B.17.3.1. void Modelo.esquemaBD.agregarTabla (tabla *T*)

Método que agrega una tabla al arbol de tablas

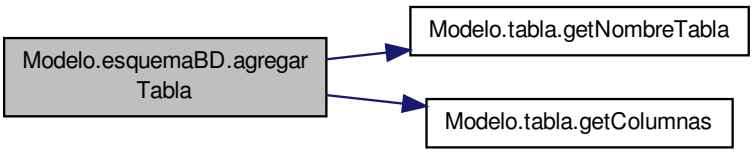
Parámetros

T	Tabla

Definición en la línea 172 del archivo esquemaBD.java.

```
172                                     {
173         DefaultMutableTreeNode tmpTabla = new DefaultMutableTreeNode(T.getNombreTabla());
174         DefaultMutableTreeNode tmpNombreColumna;
175         for (int x = 0; x < T.getColumnas().numeroColumnas(); x++) {
176             tmpNombreColumna = new DefaultMutableTreeNode(T.getColumnas().getColumna(x).getNombreColumna()
    + " "
177                     + T.getColumnas().getColumna(x).getTipoDato()
178                     + ((T.getColumnas().getColumna(x).getLongitudColumna() != 0) ? (" (" + T.getColumnas()
    .getColumna(x).getLongitudColumna() + ")") : " ")
179             );
180             tmpTabla.add(tmpNombreColumna);
181         }
182         this.MVETBD.add(tmpTabla);
183     }
```

Gráfico de llamadas para esta función:



B.17.3.2. String Modelo.esquemaBD.getCatalgo ( )

Método que recupera el nombre del catálogo de un esquema de base de datos

Devuelve

String nombre del catalgo

Definición en la línea 80 del archivo esquemaBD.java.

```
80                                {
81         return this.catalogo;
82     }
```

B.17.3.3. DefaultMutableTreeNode Modelo.esquemaBD.getDMTTablas ( )

Método que construye el árbol de tablas y lo retorna.

Devuelve

DefaultMutableTreeNode Árbol de tablas

Definición en la línea 107 del archivo esquemaBD.java.

```
107                                                {
108
109         if (this.getTablas() != null) {
110             this.setMVETBD(this.getCatalgo());
111             for (int i = 0; i < this.getTablas().numTablas(); i++) {
112                 this.agregarTabla(this.getTablas().obtenerTabla(i));
113             }
114             return this.getMVETBD();
115         } else {
116             return null;
117         }
118
119     }
```

Gráfico de llamadas para esta función:

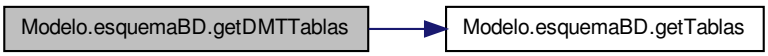
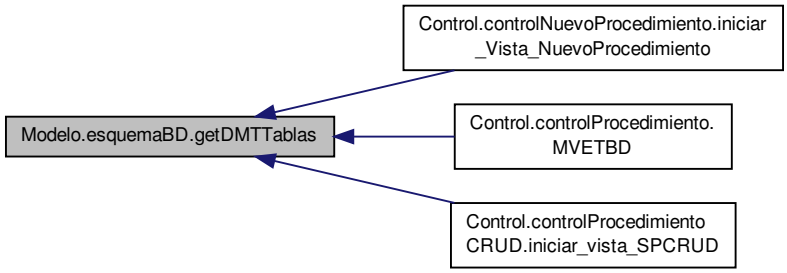


Gráfico de llamadas a esta función:



B.17.3.4. DefaultMutableTreeNode Modelo.esquemaBD.getMVETBD ( )

Método que recupera el árbol que contiene las tablas de un esquema de base de datos.

Devuelve

DefaultMutableTreeNode árbol de las tablas de un esquema de base de datos

Definición en la línea 126 del archivo esquemaBD.java.

```
126                                             {
127         return this.MVETBD;
128     }
```

B.17.3.5. tablas Modelo.esquemaBD.getTablas ( )

Método que recupera la lista de tablas de un esquema de base de datos

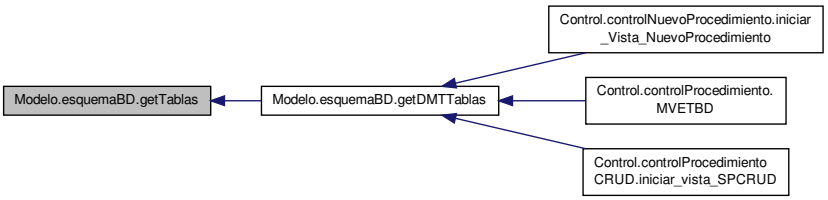
Devuelve

tablas lista de tablas

Definición en la línea 98 del archivo esquemaBD.java.

```
98                                {
99         return this.listatablas;
100     }
```

Gráfico de llamadas a esta función:



B.17.3.6. boolean Modelo.esquemaBD.MVETBDvacia ( )

Método que verifica si el árbol esta vacío. return

- true: si el árbol de tablas se encuentra vacío.
- false: si el árbol de tablas contiene almenos un elemento.

Definición en la línea 163 del archivo esquemaBD.java.

```
163                                {
164         return this.MVETBD == null;
165     }
```

B.17.3.7. void Modelo.esquemaBD.setCatalogo (String *catalogo*)

Método que establece el nombre del catálogo en un esquema de base de datos

Parámetros

catalogo	nombre del catalogo

Definición en la línea 71 del archivo esquemaBD.java.

```
71                                          {
72          this.catalogo = catalogo;
73      }
```

B.17.3.8. void Modelo.esquemaBD.setMVETBD (String *nombreBD*)

Método que establece el nombre de la base de datos como raiz del arbol de tablas.

Parámetros

nombreBD	nombre de la base de datos

Definición en la línea 136 del archivo esquemaBD.java.

```
136                                         {
137         this.MVETBD = new DefaultMutableTreeNode(nombreBD);
138     }
```

B.17.3.9. void Modelo.esquemaBD.setMVETBD (DefaultMutableTreeNode *arbolTablas*)

Método que establece el árbol de tablas en una base de datos.

Parámetros

arbolTablas	árbol de tablas

Definición en la línea 145 del archivo esquemaBD.java.

```
145                                                     {
146         this.MVETBD = arbolTablas;
147     }
```

B.17.3.10. void Modelo.esquemaBD.setTablas (tablas *Ts*)

Método que establece la lista de tablas en un esquema de base de datos

Parámetros

Ts	lista de tablas

Definición en la línea 89 del archivo esquemaBD.java.

```
89                                      {
90          this.listatablas = Ts;
91      }
```

B.17.3.11. void Modelo.esquemaBD.vaciarMVETBD ( )

Método que vacía al árbol de las tablas.

Definición en la línea 152 del archivo esquemaBD.java.

```
152                                 {
153         this.MVETBD.removeAllChildren();
154         this.MVETBD.removeFromParent();
155     }
```

B.17.4. Documentación de los datos miembro

B.17.4.1. String Modelo.esquemaBD.catalogo [private]

Nombre del catálogo

Definición en la línea 20 del archivo esquemaBD.java.

B.17.4.2. procedimientos Modelo.esquemaBD.listaProcedimientos [private]

Lista de procedimientos

Definición en la línea 28 del archivo esquemaBD.java.

B.17.4.3. tablas Modelo.esquemaBD.listatablas [private]

Lista de tablas

Definición en la línea 24 del archivo esquemaBD.java.

B.17.4.4. DefaultMutableTreeNode Modelo.esquemaBD.MVETBD [private]

Árbol de tablas a visualizar

Definición en la línea 32 del archivo esquemaBD.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/esquemaBD.java

B.18. Referencia del enum Modelo.funcionBasicaBD

Métodos públicos

- funcionBasicaBD (String funcion)
- String funcion ()
- String toString ()

Atributos públicos

- ALTA =("alta")
- BAJA =("baja")
- CAMBIO =("cambio")
- CONSULTA =("consulta")

Atributos privados

- String funcion

B.18.1. Descripción detallada

Enumeración de los tipos de funciones básicas en bases de datos

Autor

ivan

Definición en la línea 13 del archivo funcionBasicaBD.java.

B.18.2. Documentación del constructor y destructor

B.18.2.1. Modelo.funcionBasicaBD.funcionBasicaBD (String *funcion*)

Constructor por defecto de la clase, establece la etiqueta de la función básica en bases de datos

Parámetros

funcion	etiqueta de la función básica en bases de datos

Definición en la línea 42 del archivo funcionBasicaBD.java.

```
42                                      {
43         this.funcion = funcion;
44     }
```

B.18.3. Documentación de las funciones miembro

B.18.3.1. String Modelo.funcionBasicaBD.funcion ( )

Método que obtiene la etiqueta de la función básica en bases de datos.

Devuelve

String Etiqueta de la función básica en bases de datos.

Definición en la línea 51 del archivo funcionBasicaBD.java.

```
51                              {
52         return this.funcion;
53     }
```

B.18.3.2. String Modelo.funcionBasicaBD.toString ( )

Sobreescritura del método toString.

Devuelve

String Etiqueta de la función básica en bases de datos.

Definición en la línea 60 del archivo funcionBasicaBD.java.

```
60                              {
61          return this.funcion();
62      }
```

B.18.4. Documentación de los datos miembro

B.18.4.1. Modelo.funcionBasicaBD.ALTA =("alta")

Función que da de alta un registro en la base de datos

Definición en la línea 18 del archivo funcionBasicaBD.java.

B.18.4.2. Modelo.funcionBasicaBD.BAJA =("baja")

Función que da de baja un registro en la base de datos

Definición en la línea 22 del archivo funcionBasicaBD.java.

B.18.4.3. Modelo.funcionBasicaBD.CAMBIO =("cambio")

Función que realiza algún cambio en un registro en la base de datos

Definición en la línea 26 del archivo funcionBasicaBD.java.

B.18.4.4. Modelo.funcionBasicaBD.CONSULTA =("consulta")

Función que realiza una consulta sobre un registro en la base de datos

Definición en la línea 30 del archivo funcionBasicaBD.java.

B.18.4.5. String Modelo.funcionBasicaBD.funcion [private]

Etiqueta de la función básica en bases de datos

Definición en la línea 34 del archivo funcionBasicaBD.java.

La documentación para este enum ha sido generada a partir del siguiente fichero:

- Modelo/funcionBasicaBD.java

B.19. Referencia de la Clase Programa.GeneradorProcedimientosAlmacenados

Métodos públicos estáticos

- static void main (String[] args)

B.19.1. Descripción detallada

Clase Principal que implementa el asistente para la generación de código para procedimientos almacenados en un manejador de bases de datos

Autor

ivan

Definición en la línea 18 del archivo GeneradorProcedimientosAlmacenados.java.

B.19.2. Documentación de las funciones miembro

B.19.2.1. static void Programa.GeneradorProcedimientosAlmacenados.main (String[] *args*) [static]

Método estático que invoca al modelo vista controlador del asistente para la generación de código para procedimientos almacenados en un manejador de bases de datos

Parámetros

args	Argumentos (en este proyecto no se recurrio a usarlos)

Definición en la línea 27 del archivo GeneradorProcedimientosAlmacenados.java.

```
27                                          {
28         java.awt.EventQueue.invokeLater(new Runnable() {
29             @Override
30             public void run() {
31                 ProcedimientoAlmacenado PARoot = new
       ProcedimientoAlmacenado();
32                 EditorProcedimiento ventanaSPEditoRoot = new
       EditorProcedimiento();
33                 controlProcedimiento control = new
       controlProcedimiento(PARoot, ventanaSPEditoRoot);
34                 control.iniciar_vista_SPRoot();
35             }
36         });
37     }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Programa/GeneradorProcedimientosAlmacenados.java

B.20. Referencia de la Clase Control.MiFocusTraversalPolicy

Diagrama de herencias de Control.MiFocusTraversalPolicy

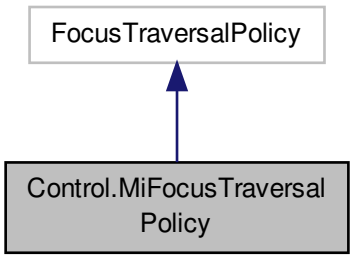
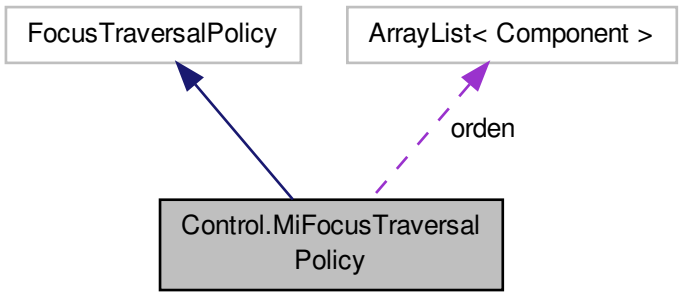


Diagrama de colaboración para Control.MiFocusTraversalPolicy:



Métodos públicos

- MiFocusTraversalPolicy (ArrayList< Component > orden)
- Component getComponentAfter (Container aContainer, Component aComponent)
- Component getComponentBefore (Container aContainer, Component aComponent)
- Component getFirstComponent (Container aContainer)
- Component getLastComponent (Container aContainer)
- Component getDefaultComponent (Container aContainer)

Atributos privados

- final ArrayList< Component > orden

B.20.1. Descripción detallada

Clase que implementa una política de recorrido del foco

Autor

ivan

Definición en la línea 18 del archivo MiFocusTraversalPolicy.java.

B.20.2. Documentación del constructor y destructor

B.20.2.1. Control.MiFocusTraversalPolicy.MiFocusTraversalPolicy (ArrayList< Component > *orden*)

Constructor por defecto de las clase, establece la lista de componentes

Parámetros

orden	Lista de componentes

Definición en la línea 30 del archivo MiFocusTraversalPolicy.java.

```
30                                                          {
31         this.orden = new ArrayList<>(orden);
32         this.orden.addAll(orden);
33     }
```

B.20.3. Documentación de las funciones miembro

B.20.3.1. Component Control.MiFocusTraversalPolicy.getComponentAfter (Container *aContainer,* Component *aComponent*)

Sobreescritura del método getComponentAfter

Parámetros

aContainer	Contenedor
aComponent	Componente

Devuelve

Component Componente

Definición en la línea 43 del archivo MiFocusTraversalPolicy.java.

```
43                                                                                  {
44         int idx;
45         idx = (orden.indexOf(aComponent) + 1) % orden.size();
46         return orden.get(idx);
47 
48     }
```

B.20.3.2. Component Control.MiFocusTraversalPolicy.getComponentBefore (Container *aContainer,* Component *aComponent*)

Sobreescritura del método getComponentBefore

Parámetros

aContainer	Contenedor
aComponent	Componente

Devuelve

Component Componente

Definición en la línea 58 del archivo MiFocusTraversalPolicy.java.

```
58                                                                                  {
59         int idx = orden.indexOf(aComponent) - 1;
60         if (idx < 0) {
61             idx = orden.size() - 1;
62         }
63         return orden.get(idx);
64     }
```

B.20.3.3. Component Control.MiFocusTraversalPolicy.getDefaultComponent (Container *aContainer*)

Sobreescritura del método getDefaultComponent

Parámetros

aContainer	Contenedor

Devuelve

Component Componente

Definición en la línea 95 del archivo MiFocusTraversalPolicy.java.

```
95                                                      {
96         return orden.get(0);
97     }
```

B.20.3.4. Component Control.MiFocusTraversalPolicy.getFirstComponent (Container *aContainer*)

Sobreescritura del método getFirtsComponent

Parámetros

aContainer	Contenedor

Devuelve

Component Componente

Definición en la línea 73 del archivo MiFocusTraversalPolicy.java.

```
73                                                     {
74         return orden.get(0);
75     }
```

B.20.3.5. Component Control.MiFocusTraversalPolicy.getLastComponent (Container *aContainer*)

Sobreescritura del método getLastComponent

Parámetros

aContainer	Contenedor

Devuelve

Component Componente

Definición en la línea 84 del archivo MiFocusTraversalPolicy.java.

```
84                                                      {
85         return orden.get(orden.size() - 1);
86     }
```

B.20.4. Documentación de los datos miembro

B.20.4.1. final ArrayList<Component> Control.MiFocusTraversalPolicy.orden [private]

Lista de componentes

Definición en la línea 23 del archivo MiFocusTraversalPolicy.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Control/MiFocusTraversalPolicy.java

B.21. Referencia de la Clase Vistas.nuevoProcedimiento

Diagrama de herencias de Vistas.nuevoProcedimiento

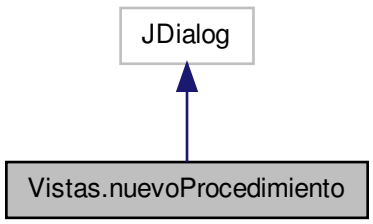
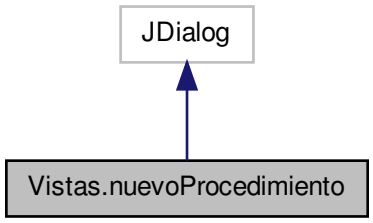


Diagrama de colaboración para Vistas.nuevoProcedimiento:



Métodos públicos

- nuevoProcedimiento (java.awt.Frame parent, boolean modal)

Métodos públicos estáticos

- static void main (String args[])

Atributos públicos

- javax.swing.JButton jButton1
- javax.swing.JButton jButton12
- javax.swing.JButton jButton13
- javax.swing.JButton jButton14
- javax.swing.JButton jButton15
- javax.swing.JButton jButton2
- javax.swing.JButton jButton3
- javax.swing.JButton jButton4
- javax.swing.JButton jButton5
- javax.swing.JCheckBox jCheckBox2
- javax.swing.JCheckBox jCheckBox3
- javax.swing.JComboBox jComboBox1
- javax.swing.JComboBox jComboBox2
- javax.swing.JComboBox jComboBox3
- javax.swing.JComboBox jComboBox4
- javax.swing.JComboBox jComboBox6
- javax.swing.JLabel jLabel1
- javax.swing.JLabel jLabel10
- javax.swing.JLabel jLabel2
- javax.swing.JLabel jLabel3
- javax.swing.JLabel jLabel4
- javax.swing.JLabel jLabel5
- javax.swing.JLabel jLabel6

- javax.swing.JLabel jLabel7
- javax.swing.JLabel jLabel8
- javax.swing.JLabel jLabel9
- javax.swing.JList jList1
- javax.swing.JPanel jPanel15
- javax.swing.JPanel jPanel16
- javax.swing.JPanel jPanel17
- javax.swing.JPanel jPanel18
- javax.swing.JPanel jPanel2
- javax.swing.JScrollPane jScrollPane1
- javax.swing.JScrollPane jScrollPane10
- javax.swing.JScrollPane jScrollPane11
- javax.swing.JScrollPane jScrollPane9
- javax.swing.JSeparator jSeparator1
- javax.swing.JTextArea jTextArea8
- javax.swing.JTextArea jTextArea9
- javax.swing.JTextField jTextField1
- javax.swing.JTextField jTextField2
- javax.swing.JTree jTree1
- javax.swing.JComboBox tipoDatoSQL

Métodos privados

- void initComponents ()

B.21.1. Descripción detallada

Autor

ivan

Definición en la línea 13 del archivo nuevoProcedimiento.java.

B.21.2. Documentación del constructor y destructor

B.21.2.1. Vistas.nuevoProcedimiento.nuevoProcedimiento (java.awt.Frame *parent,* boolean *modal*)

Creates new form nuevoProcedimiento

Definición en la línea 18 del archivo nuevoProcedimiento.java.

```
18                                                                  {
19         super(parent, modal);
20         initComponents();
21     }
```

Gráfico de llamadas para esta función:

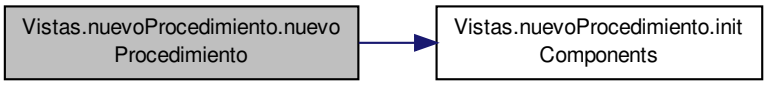
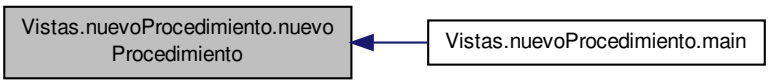


Gráfico de llamadas a esta función:



B.21.3. Documentación de las funciones miembro

B.21.3.1. void Vistas.nuevoProcedimiento.initComponents ( ) [private]

This method is called from within the constructor to initialize the form. WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

Definición en la línea 30 del archivo nuevoProcedimiento.java.

```
30                                {
31
32          jPanel15 = new javax.swing.JPanel();
33          jPanel16 = new javax.swing.JPanel();
34          jScrollPane9 = new javax.swing.JScrollPane();
35          jTextArea8 = new javax.swing.JTextArea();
36          jButton12 = new javax.swing.JButton();
37          jPanel2 = new javax.swing.JPanel();
38          jLabel3 = new javax.swing.JLabel();
39          jTextField2 = new javax.swing.JTextField();
40          jLabel4 = new javax.swing.JLabel();
41          jButton2 = new javax.swing.JButton();
42          jButton1 = new javax.swing.JButton();
43          jLabel6 = new javax.swing.JLabel();
44          jComboBox4 = new javax.swing.JComboBox();
45          jSeparator1 = new javax.swing.JSeparator();
46          jScrollPane1 = new javax.swing.JScrollPane();
47          jList1 = new javax.swing.JList();
48          jButton13 = new javax.swing.JButton();
49          jButton14 = new javax.swing.JButton();
50          tipoDatoSQL = new javax.swing.JComboBox();
51          jLabel1 = new javax.swing.JLabel();
52          jTextField1 = new javax.swing.JTextField();
53          jButton4 = new javax.swing.JButton();
54          jButton3 = new javax.swing.JButton();
55          jLabel10 = new javax.swing.JLabel();
56          jLabel2 = new javax.swing.JLabel();
57          jComboBox1 = new javax.swing.JComboBox();
58          jLabel5 = new javax.swing.JLabel();
59          jComboBox3 = new javax.swing.JComboBox();
60          jPanel17 = new javax.swing.JPanel();
61          jCheckBox2 = new javax.swing.JCheckBox();
```

```
62          jCheckBox3 = new javax.swing.JCheckBox();
63          jComboBox2 = new javax.swing.JComboBox();
64          jComboBox6 = new javax.swing.JComboBox();
65          jScrollPane10 = new javax.swing.JScrollPane();
66          jTextArea9 = new javax.swing.JTextArea();
67          jLabel7 = new javax.swing.JLabel();
68          jLabel8 = new javax.swing.JLabel();
69          jLabel9 = new javax.swing.JLabel();
70          jButton15 = new javax.swing.JButton();
71          jButton5 = new javax.swing.JButton();
72          jPanel18 = new javax.swing.JPanel();
73          jScrollPane11 = new javax.swing.JScrollPane();
74          jTree1 = new javax.swing.JTree();
75
76          setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
77
78          jPanel15.setBorder(javax.swing.BorderFactory.createTitledBorder("Procedimiento Almacenado"));
79
80          jPanel16.setBorder(javax.swing.BorderFactory.createTitledBorder("*cuerpo rutina"));
81
82          jTextArea8.setColumns(20);
83          jTextArea8.setRows(5);
84          jTextArea8.setText("# agrega todo tu codigo de control aquí: cualquier tipo de datos MySQL valido")
    ;
85          jScrollPane9.setViewportView(jTextArea8);
86
87          jButton12.setText("Nuevo Procedimiento Anidado");
88
89          javax.swing.GroupLayout jPanel16Layout = new javax.swing.GroupLayout(
    jPanel16);
90          jPanel16.setLayout(jPanel16Layout);
91          jPanel16Layout.setHorizontalGroup(
92              jPanel16Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
93              .addGroup(jPanel16Layout.createSequentialGroup()
94                  .addGroup(jPanel16Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
95                      .addComponent(jScrollPane9)
96                      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
    jPanel16Layout.createSequentialGroup()
97                          .addGap(0, 0, Short.MAX_VALUE)
98                          .addComponent(jButton12)))
99                  .addContainerGap())
100         );
101         jPanel16Layout.setVerticalGroup(
102             jPanel16Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
103             .addGroup(jPanel16Layout.createSequentialGroup()
104                 .addComponent(jScrollPane9, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
105                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
106                 .addComponent(jButton12)
107                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
108         );
109
110         jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("parametros"));
111
112         jLabel3.setText("nombre del parametro:");
113
114         jLabel4.setText("tipo de dato:");
115
116         jButton2.setText("»");
117
118         jButton1.setText("«");
119
120         jLabel6.setText("tipo de entrada:");
121
122         jComboBox4.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "IN", "OUT", "INOUT" }));
123
124         jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);
125
126         jScrollPane1.setViewportView(jList1);
127
128         jButton13.setText("borrar");
129         jButton13.setMaximumSize(new java.awt.Dimension(26, 27));
130         jButton13.setMinimumSize(new java.awt.Dimension(26, 27));
131         jButton13.setPreferredSize(new java.awt.Dimension(26, 27));
132
133         jButton14.setText("guardar cambios");
134         jButton14.setEnabled(false);
135
136         javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(
    jPanel2);
137         jPanel2.setLayout(jPanel2Layout);
```

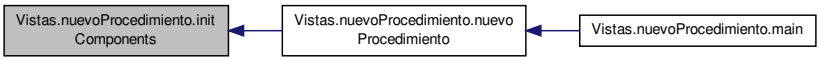
```
138         jPanel2Layout.setHorizontalGroup(
139             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
140             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
141                 .addContainerGap()
142                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
143                     .addGroup(jPanel2Layout.createSequentialGroup()
144                         .addComponent(jLabel3)
145                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
146                         .addComponent(jTextField2))
147                     .addGroup(jPanel2Layout.createSequentialGroup()
148                         .addComponent(jLabel4)
149                         .addGap(32, 32, 32)
150                         .addComponent(tipoDatoSQL, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
151                     .addGroup(jPanel2Layout.createSequentialGroup()
152                         .addComponent(jLabel6)
153                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
154                         .addComponent(jComboBox4, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE))
155                     .addGroup(jPanel2Layout.createSequentialGroup()
156                         .addGap(0, 0, Short.MAX_VALUE)
157                         .addComponent(jButton14)))
158                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
159                 .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 11, javax.
    swing.GroupLayout.PREFERRED_SIZE)
160                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
161                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
162                     .addComponent(jButton1)
163                     .addComponent(jButton2)
164                     .addComponent(jButton13, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
    javax.swing.GroupLayout.PREFERRED_SIZE))
165                 .addGap(18, 18, 18)
166                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 139, javax.
    swing.GroupLayout.PREFERRED_SIZE)
167                 .addContainerGap())
168         );
169         jPanel2Layout.setVerticalGroup(
170             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
171             .addGroup(jPanel2Layout.createSequentialGroup()
172                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
173                     .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 111,
    javax.swing.GroupLayout.PREFERRED_SIZE)
174                     .addGroup(jPanel2Layout.createSequentialGroup()
175                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
176                             .addComponent(jLabel6)
177                             .addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
178                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
179                         .addGroup(jPanel2Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.BASELINE)
180                             .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
181                             .addComponent(jLabel3))
182                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
183                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
184                             .addComponent(jLabel4)
185                             .addComponent(tipoDatoSQL, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
186                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
187                         .addComponent(jButton14))
188                     .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
     false)
189                         .addGroup(jPanel2Layout.createSequentialGroup()
190                             .addComponent(jButton2)
191                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
192                             .addComponent(jButton1)
193                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.
    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
194                             .addComponent(jButton13, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
195                         .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
    javax.swing.GroupLayout.PREFERRED_SIZE, 90, javax.swing.GroupLayout.PREFERRED_SIZE)))
196                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
197         );
198 
199         jLabel1.setText("*nombre del procedimiento");
200 
201         jButton4.setText("Vista previa");
202 
```

```
203         jButton3.setText("Generar código");
204 
205         jLabel10.setText("* Campos obligatorios");
206 
207         jLabel2.setText("usuario:");
208 
209         jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "CURRENT_USER" }));
210 
211         jLabel5.setText("base de datos:");
212 
213         jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "<por defecto>" }));
214 
215         jPanel17.setBorder(javax.swing.BorderFactory.createTitledBorder(
    javax.swing.BorderFactory.createTitledBorder("caracteristicas")));
216 
217         jCheckBox2.setText("LANGUAGE SQL");
218         jCheckBox2.setEnabled(false);
219 
220         jCheckBox3.setText("DETERMINISTIC");
221 
222         jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "DEFINER", "INVOKER" }));
223 
224         jComboBox6.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "CONTAINS SQL", "NO SQL", "
    READS SQL DATA", "MODIFIES SQL DATA" }));
225 
226         jTextArea9.setColumns(20);
227         jTextArea9.setRows(5);
228         jTextArea9.setText("Sin comentarios");
229         jScrollPane10.setViewportView(jTextArea9);
230 
231         jLabel7.setText("SQL SECURITY:");
232 
233         jLabel8.setText("información del manejo de datos:");
234 
235         jLabel9.setText("COMENT:");
236 
237         javax.swing.GroupLayout jPanel17Layout = new javax.swing.GroupLayout(
    jPanel17);
238         jPanel17.setLayout(jPanel17Layout);
239         jPanel17Layout.setHorizontalGroup(
240             jPanel17Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
241             .addGroup(jPanel17Layout.createSequentialGroup()
242                 .addGap(6, 6, 6)
243                 .addGroup(jPanel17Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
244                     .addGroup(jPanel17Layout.createSequentialGroup()
245                         .addComponent(jCheckBox2)
246                         .addGap(18, 18, 18)
247                         .addComponent(jCheckBox3)
248                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
249                         .addComponent(jLabel7)
250                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
251                         .addComponent(jComboBox2, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE))
252                     .addGroup(jPanel17Layout.createSequentialGroup()
253                         .addComponent(jLabel9)
254                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
255                         .addComponent(jScrollPane10))
256                     .addGroup(jPanel17Layout.createSequentialGroup()
257                         .addComponent(jLabel8)
258                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
259                         .addComponent(jComboBox6, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
    MAX_VALUE)))
260                 .addContainerGap())
261         );
262         jPanel17Layout.setVerticalGroup(
263             jPanel17Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
264             .addGroup(jPanel17Layout.createSequentialGroup()
265                 .addGroup(jPanel17Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
266                     .addComponent(jCheckBox2)
267                     .addComponent(jCheckBox3)
268                     .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing
    .GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
269                     .addComponent(jLabel7))
270                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
271                 .addGroup(jPanel17Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
272                     .addComponent(jComboBox6, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
273                     .addComponent(jLabel8))
274                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
275                 .addGroup(jPanel17Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
276                     .addComponent(jLabel9)
```

```
277                     .addComponent(jScrollPane10, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
    javax.swing.GroupLayout.PREFERRED_SIZE))
278                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
279         );
280 
281         jButton15.setText("Guardar");
282 
283         jButton5.setText("Finalizar");
284 
285         javax.swing.GroupLayout jPanel15Layout = new javax.swing.GroupLayout(
    jPanel15);
286         jPanel15.setLayout(jPanel15Layout);
287         jPanel15Layout.setHorizontalGroup(
288             jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
289             .addGroup(jPanel15Layout.createSequentialGroup()
290                 .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
291                     .addGroup(jPanel15Layout.createSequentialGroup()
292                         .addContainerGap()
293                         .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
294                             .addComponent(jPanel16, javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
295                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel15Layout.
    createSequentialGroup()
296                                 .addComponent(jLabel2)
297                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
298                                 .addComponent(jComboBox1, 0, javax.swing.GroupLayout.DEFAULT_SIZE
    , Short.MAX_VALUE))
299                             .addGroup(jPanel15Layout.createSequentialGroup()
300                                 .addComponent(jLabel1)
301                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
302                                 .addComponent(jTextField1))
303                             .addGroup(jPanel15Layout.createSequentialGroup()
304                                 .addComponent(jLabel5)
305                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
306                                 .addComponent(jComboBox3, 0, javax.swing.GroupLayout.DEFAULT_SIZE
    , Short.MAX_VALUE))
307                             .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
308                             .addComponent(jPanel17, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
309                     .addGroup(jPanel15Layout.createSequentialGroup()
310                         .addComponent(jLabel10)
311                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 91,
    Short.MAX_VALUE)
312                         .addComponent(jButton5)
313                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
314                         .addComponent(jButton15)
315                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
316                         .addComponent(jButton4)
317                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
318                         .addComponent(jButton3)))
319                 .addContainerGap())
320         );
321         jPanel15Layout.setVerticalGroup(
322             jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
323             .addGroup(jPanel15Layout.createSequentialGroup()
324                 .addContainerGap()
325                 .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
326                     .addComponent(jLabel5)
327                     .addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing
    .GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
328                 .addGap(6, 6, 6)
329                 .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
330                     .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
331                     .addComponent(jLabel1))
332                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
333                 .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
334                     .addComponent(jLabel2)
335                     .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing
    .GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
336                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
337                 .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
338                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
339                 .addComponent(jPanel17, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
340                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
341                 .addComponent(jPanel16, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
342                 .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
343                     .addGroup(jPanel15Layout.createSequentialGroup()
344                         .addGroup(jPanel15Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
345                             .addComponent(jButton3)
346                             .addComponent(jButton4)
347                             .addComponent(jButton15)
348                             .addComponent(jButton5))
349                         .addContainerGap())
350                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel15Layout.
    createSequentialGroup()
351                         .addGap(0, 0, Short.MAX_VALUE)
352                         .addComponent(jLabel10))))
353         );
354 
355         jPanel18.setBorder(javax.swing.BorderFactory.createTitledBorder("Árbol de Tablas"));
356         jPanel18.setPreferredSize(new java.awt.Dimension(180, 424));
357 
358         javax.swing.tree.DefaultMutableTreeNode treeNode1 = new javax.swing.tree.DefaultMutableTreeNode("
    vacía");
359         jTree1.setModel(new javax.swing.tree.DefaultTreeModel(treeNode1));
360         jTree1.setAutoscrolls(true);
361         jTree1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
362         jTree1.setInheritsPopupMenu(true);
363         jScrollPane11.setViewportView(jTree1);
364 
365         javax.swing.GroupLayout jPanel18Layout = new javax.swing.GroupLayout(
    jPanel18);
366         jPanel18.setLayout(jPanel18Layout);
367         jPanel18Layout.setHorizontalGroup(
368             jPanel18Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
369             .addGroup(jPanel18Layout.createSequentialGroup()
370                 .addContainerGap()
371                 .addComponent(jScrollPane11, javax.swing.GroupLayout.DEFAULT_SIZE, 147, Short.
    MAX_VALUE)
372                 .addContainerGap())
373         );
374         jPanel18Layout.setVerticalGroup(
375             jPanel18Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
376             .addGroup(jPanel18Layout.createSequentialGroup()
377                 .addContainerGap()
378                 .addComponent(jScrollPane11, javax.swing.GroupLayout.DEFAULT_SIZE, 595, Short.
    MAX_VALUE)
379                 .addContainerGap())
380         );
381 
382         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
383         getContentPane().setLayout(layout);
384         layout.setHorizontalGroup(
385             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
386             .addGroup(layout.createSequentialGroup()
387                 .addContainerGap()
388                 .addComponent(jPanel15, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
389                 .addContainerGap(204, Short.MAX_VALUE))
390             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
391                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
392                     .addContainerGap(560, Short.MAX_VALUE)
393                     .addComponent(jPanel18, javax.swing.GroupLayout.PREFERRED_SIZE, 183, javax.
    swing.GroupLayout.PREFERRED_SIZE)
394                     .addContainerGap())))
395         );
396         layout.setVerticalGroup(
397             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
398             .addGroup(layout.createSequentialGroup()
399                 .addContainerGap()
400                 .addComponent(jPanel15, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
401                 .addContainerGap())
402             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
403                 .addGroup(layout.createSequentialGroup()
404                     .addContainerGap()
405                     .addComponent(jPanel18, javax.swing.GroupLayout.DEFAULT_SIZE, 650, Short.
    MAX_VALUE)
406                     .addContainerGap()))
407         );
408 
409         pack();
410     }// </editor-fold>//GEN-END:initComponents
```

Gráfico de llamadas a esta función:



B.21.3.2. static void Vistas.nuevoProcedimiento.main (String *args[]*) [static]

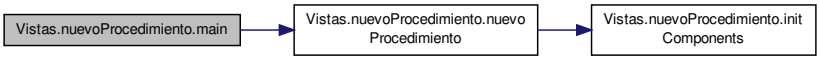
Parámetros

args	the command line arguments

Definición en la línea 415 del archivo nuevoProcedimiento.java.

```
415                                         {
416         /* Set the Nimbus look and feel */
417         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
418         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
419          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
420          */
421         try {
422             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
    getInstalledLookAndFeels()) {
423                 if ("Nimbus".equals(info.getName())) {
424                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
425                     break;
426                 }
427             }
428         } catch (ClassNotFoundException ex) {
429             java.util.logging.Logger.getLogger(nuevoProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
430         } catch (InstantiationException ex) {
431             java.util.logging.Logger.getLogger(nuevoProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
432         } catch (IllegalAccessException ex) {
433             java.util.logging.Logger.getLogger(nuevoProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
434         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
435             java.util.logging.Logger.getLogger(nuevoProcedimiento.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
436         }
437         //</editor-fold>
438 
439         /* Create and display the dialog */
440         java.awt.EventQueue.invokeLater(new Runnable() {
441             public void run() {
442                 nuevoProcedimiento dialog = new
    nuevoProcedimiento(new javax.swing.JFrame(), true);
443                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
444                     @Override
445                     public void windowClosing(java.awt.event.WindowEvent e) {
446                         System.exit(0);
447                     }
448                 });
449                 dialog.setVisible(true);
450             }
451         });
452     }
```

Gráfico de llamadas para esta función:



B.21.4. Documentación de los datos miembro

B.21.4.1. javax.swing.JButton Vistas.nuevoProcedimiento.jButton1

Definición en la línea 455 del archivo nuevoProcedimiento.java.

B.21.4.2. javax.swing.JButton Vistas.nuevoProcedimiento.jButton12

Definición en la línea 456 del archivo nuevoProcedimiento.java.

B.21.4.3. javax.swing.JButton Vistas.nuevoProcedimiento.jButton13

Definición en la línea 457 del archivo nuevoProcedimiento.java.

B.21.4.4. javax.swing.JButton Vistas.nuevoProcedimiento.jButton14

Definición en la línea 458 del archivo nuevoProcedimiento.java.

B.21.4.5. javax.swing.JButton Vistas.nuevoProcedimiento.jButton15

Definición en la línea 459 del archivo nuevoProcedimiento.java.

B.21.4.6. javax.swing.JButton Vistas.nuevoProcedimiento.jButton2

Definición en la línea 460 del archivo nuevoProcedimiento.java.

B.21.4.7. javax.swing.JButton Vistas.nuevoProcedimiento.jButton3

Definición en la línea 461 del archivo nuevoProcedimiento.java.

B.21.4.8. javax.swing.JButton Vistas.nuevoProcedimiento.jButton4

Definición en la línea 462 del archivo nuevoProcedimiento.java.

B.21.4.9. javax.swing.JButton Vistas.nuevoProcedimiento.jButton5

Definición en la línea 463 del archivo nuevoProcedimiento.java.

B.21.4.10. javax.swing.JCheckBox Vistas.nuevoProcedimiento.jCheckBox2

Definición en la línea 464 del archivo nuevoProcedimiento.java.

B.21.4.11. javax.swing.JCheckBox Vistas.nuevoProcedimiento.jCheckBox3

Definición en la línea 465 del archivo nuevoProcedimiento.java.

B.21.4.12. javax.swing.JComboBox Vistas.nuevoProcedimiento.jComboBox1

Definición en la línea 466 del archivo nuevoProcedimiento.java.

B.21.4.13. javax.swing.JComboBox Vistas.nuevoProcedimiento.jComboBox2

Definición en la línea 467 del archivo nuevoProcedimiento.java.

B.21.4.14. javax.swing.JComboBox Vistas.nuevoProcedimiento.jComboBox3

Definición en la línea 468 del archivo nuevoProcedimiento.java.

B.21.4.15. javax.swing.JComboBox Vistas.nuevoProcedimiento.jComboBox4

Definición en la línea 469 del archivo nuevoProcedimiento.java.

B.21.4.16. javax.swing.JComboBox Vistas.nuevoProcedimiento.jComboBox6

Definición en la línea 470 del archivo nuevoProcedimiento.java.

B.21.4.17. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel1

Definición en la línea 471 del archivo nuevoProcedimiento.java.

B.21.4.18. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel10

Definición en la línea 472 del archivo nuevoProcedimiento.java.

B.21.4.19. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel2

Definición en la línea 473 del archivo nuevoProcedimiento.java.

B.21.4.20. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel3

Definición en la línea 474 del archivo nuevoProcedimiento.java.

B.21.4.21. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel4

Definición en la línea 475 del archivo nuevoProcedimiento.java.

B.21.4.22. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel5

Definición en la línea 476 del archivo nuevoProcedimiento.java.

B.21.4.23. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel6

Definición en la línea 477 del archivo nuevoProcedimiento.java.

B.21.4.24. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel7

Definición en la línea 478 del archivo nuevoProcedimiento.java.

B.21.4.25. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel8

Definición en la línea 479 del archivo nuevoProcedimiento.java.

B.21.4.26. javax.swing.JLabel Vistas.nuevoProcedimiento.jLabel9

Definición en la línea 480 del archivo nuevoProcedimiento.java.

B.21.4.27. javax.swing.JList Vistas.nuevoProcedimiento.jList1

Definición en la línea 481 del archivo nuevoProcedimiento.java.

B.21.4.28. javax.swing.JPanel Vistas.nuevoProcedimiento.jPanel15

Definición en la línea 482 del archivo nuevoProcedimiento.java.

B.21.4.29. javax.swing.JPanel Vistas.nuevoProcedimiento.jPanel16

Definición en la línea 483 del archivo nuevoProcedimiento.java.

B.21.4.30. javax.swing.JPanel Vistas.nuevoProcedimiento.jPanel17

Definición en la línea 484 del archivo nuevoProcedimiento.java.

B.21.4.31. javax.swing.JPanel Vistas.nuevoProcedimiento.jPanel18

Definición en la línea 485 del archivo nuevoProcedimiento.java.

B.21.4.32. javax.swing.JPanel Vistas.nuevoProcedimiento.jPanel2

Definición en la línea 486 del archivo nuevoProcedimiento.java.

B.21.4.33. javax.swing.JScrollPane Vistas.nuevoProcedimiento.jScrollPane1

Definición en la línea 487 del archivo nuevoProcedimiento.java.

B.21.4.34. javax.swing.JScrollPane Vistas.nuevoProcedimiento.jScrollPane10

Definición en la línea 488 del archivo nuevoProcedimiento.java.

B.21.4.35. javax.swing.JScrollPane Vistas.nuevoProcedimiento.jScrollPane11

Definición en la línea 489 del archivo nuevoProcedimiento.java.

B.21.4.36. javax.swing.JScrollPane Vistas.nuevoProcedimiento.jScrollPane9

Definición en la línea 490 del archivo nuevoProcedimiento.java.

B.21.4.37. javax.swing.JSeparator Vistas.nuevoProcedimiento.jSeparator1

Definición en la línea 491 del archivo nuevoProcedimiento.java.

B.21.4.38. javax.swing.JTextArea Vistas.nuevoProcedimiento.jTextArea8

Definición en la línea 492 del archivo nuevoProcedimiento.java.

B.21.4.39. javax.swing.JTextArea Vistas.nuevoProcedimiento.jTextArea9

Definición en la línea 493 del archivo nuevoProcedimiento.java.

B.21.4.40. javax.swing.JTextField Vistas.nuevoProcedimiento.jTextField1

Definición en la línea 494 del archivo nuevoProcedimiento.java.

B.21.4.41. javax.swing.JTextField Vistas.nuevoProcedimiento.jTextField2

Definición en la línea 495 del archivo nuevoProcedimiento.java.

B.21.4.42. javax.swing.JTree Vistas.nuevoProcedimiento.jTree1

Definición en la línea 496 del archivo nuevoProcedimiento.java.

B.21.4.43. javax.swing.JComboBox Vistas.nuevoProcedimiento.tipoDatoSQL

Definición en la línea 497 del archivo nuevoProcedimiento.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vistas/nuevoProcedimiento.java

B.22. Referencia de la Clase Modelo.Parametro

Diagrama de herencias de Modelo.Parametro

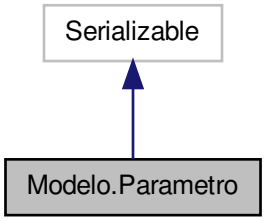
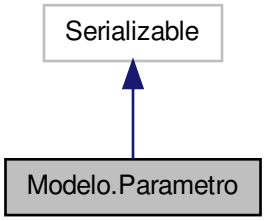


Diagrama de colaboración para Modelo.Parametro:



Métodos públicos

- boolean getLlave ()
- void setLlave (boolean llave)
- int getPrecision ()
- void setPrecision (int precision)
- String getTipoES ()

- void setTipoES (String tipoES)
- String getNombre ()
- void setNombre (String nombre)
- String getTipoDato ()
- void setTipoDato (String tipoDato)
- Parametro ()
- Parametro (String tipoES, String nombre, String tipoDato)
- Parametro (String tipoES, String nombre, String tipoDato, int precision)
- Parametro (String tipoES, String nombre, String tipoDato, int precision, boolean llave)
- Parametro (String nombre, String tipoDato)
- String generarSQL ()

Atributos privados

- String tipoES
- String nombre
- String tipoDato
- int precision
- boolean llave

B.22.1. Descripción detallada

Clase que implementa el parámetro de un procedimiento almacenado

Autor

ivan

Definición en la línea 17 del archivo Parametro.java.

B.22.2. Documentación del constructor y destructor

B.22.2.1. Modelo.Parametro.Parametro ( )

Constructor por defecto de la clase

Definición en la línea 147 del archivo Parametro.java.

```
147                         {
148     }
```

B.22.2.2. Modelo.Parametro.Parametro (String *tipoES,* String *nombre,* String *tipoDato*)

Sobrecarga del constructor de la clase, establece el tipo de entrada, el nombre y el tipo de dato de un parámetro

Parámetros

tipoES	tipo de entrada de un parámetro
nombre	nombre de un parámetro
tipoDato	tipo de dato de un parámetro

Definición en la línea 158 del archivo Parametro.java.

```
158                                                                 {
159         this.setTipoES(tipoES);
160         this.setNombre(nombre);
161         this.setTipoDato(tipoDato);
162     }
```

B.22.2.3. Modelo.Parametro.Parametro (String *tipoES,* String *nombre,* String *tipoDato,* int *precision*)

Sobrecarga del constructor de la clase, establece el tipo de entrada, el nombre, el tipo de dato y la precisión del tipo de dato de un parámetro

Parámetros

tipoES	tipo de entrada de un parámetro
nombre	nombre de un parámetro
tipoDato	tipo de dato de un parámetro
precision	precisión del tipo de dato de un parámetro

Definición en la línea 173 del archivo Parametro.java.

```
173                                                                         {
174         this.setTipoES(tipoES);
175         this.setNombre(nombre);
176         this.setTipoDato(tipoDato);
177         this.setPrecision(precision);
178     }
```

B.22.2.4. Modelo.Parametro.Parametro (String *tipoES,* String *nombre,* String *tipoDato,* int *precision,* boolean *llave*)

Sobrecarga del constructor de la clase, establece el tipo de entrada, el nombre, el tipo de dato, la precisión del tipo de dato y la propiedad llave de un parámetro

Parámetros

tipoES	tipo de entrada de un parámetro
nombre	nombre de un parámetro
tipoDato	tipo de dato de un parámetro
precision	precisión del tipo de dato de un parámetro
llave	propiedad llave de un parámetro

Definición en la línea 191 del archivo Parametro.java.

```
191                                                                                 {
192         this.setTipoES(tipoES);
193         this.setNombre(nombre);
194         this.setTipoDato(tipoDato);
195         this.setPrecision(precision);
196         this.setLlave(llave);
197     }
```

B.22.2.5. Modelo.Parametro.Parametro (String *nombre,* String *tipoDato*)

Sobrecarga del constructor de la clase, establece el nombre y el tipo de dato de un parámetro

Parámetros

nombre	Nombre de un parámetro
tipoDato	Tipo de dato de un parámetro

Definición en la línea 206 del archivo Parametro.java.

```
206                                                 {
207         this.setNombre(nombre);
208         this.setTipoDato(tipoDato);
209     }
```

B.22.3. Documentación de las funciones miembro

B.22.3.1. String Modelo.Parametro.generarSQL ( )

Método que genera y retorna el código SQL de un parámetro

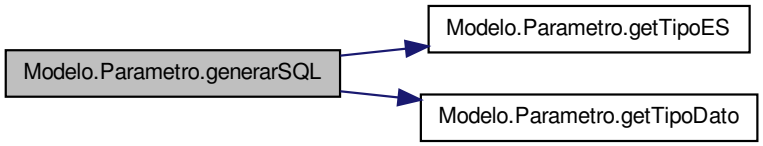
Devuelve

String Código SQL

Definición en la línea 216 del archivo Parametro.java.

```
216                                 {
217         String SQL = "";
218         if (this.getTipoES() != null) {
219             SQL = SQL + this.getTipoES() + " ";
220         }
221         return SQL + this.getNombre() + " " + this.getTipoDato();
222     }
```

Gráfico de llamadas para esta función:



B.22.3.2. boolean Modelo.Parametro.getLlave ( )

Método que obtiene la propiedad llave del parámetro

Devuelve

boolean

- true indica que el parámetro representa a una columna que es una llave
- false indica que el parámetro no representa a una columna que es una llave

Definición en la línea 57 del archivo Parametro.java.

```
57                              {
58          return this.llave;
59      }
```

B.22.3.3. String Modelo.Parametro.getNombre ( )

Método que obtiene el nombre de un parámetro

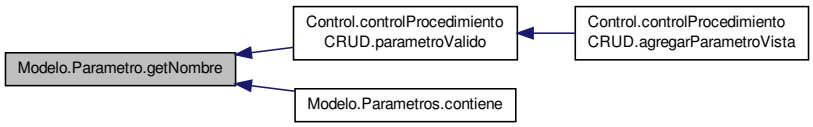
Devuelve

String Nombre del parámetro

Definición en la línea 111 del archivo Parametro.java.

```
111                              {
112          return this.nombre;
113      }
```

Gráfico de llamadas a esta función:



B.22.3.4. int Modelo.Parametro.getPrecision ( )

Método que obtiene la precisión del tipo de dato de un parámetro

Devuelve

int Precisión del tipo de dato de un parámetro

Definición en la línea 75 del archivo Parametro.java.

```
75                              {
76          return this.precision;
77      }
```

B.22.3.5. String Modelo.Parametro.getTipoDato ( )

Método que obtiene el tipo de dato de un parámetro

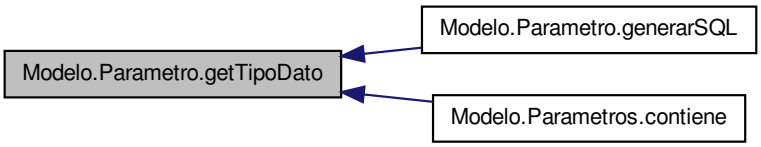
Devuelve

String Tipo de dato de un parámetro

Definición en la línea 129 del archivo Parametro.java.

```
129                                 {
130         Pattern p = Pattern.compile("\\D*\\S*[(]\\d*[)]\\.*");
131         Matcher m = p.matcher(this.tipoDato);
132         return (m.matches()) ? (this.tipoDato) : ((this.getPrecision() != 0) ? this.tipoDato + "("
      + this.precision + ")" : this.tipoDato);
133     }
```

Gráfico de llamadas a esta función:



B.22.3.6. String Modelo.Parametro.getTipoES ( )

Método que obtiene el tipo de entrada de un parámetro

Devuelve

String Tipo de entrada de un parámetro

Definición en la línea 93 del archivo Parametro.java.

```
93                                 {
94         return this.tipoES;
95     }
```

Gráfico de llamadas a esta función:

B.22.3.7. void Modelo.Parametro.setLlave (boolean *llave*)

Método que establece la propiedad llave de un parámetro

Parámetros

llave	propiedad llave de un parámetro

Definición en la línea 66 del archivo Parametro.java.

```
66                                          {
67          this.llave = llave;
68      }
```

B.22.3.8. void Modelo.Parametro.setNombre (String *nombre*)

Método que establece el nombre de un parámetro

Parámetros

nombre	Nombre del parámetro

Definición en la línea 120 del archivo Parametro.java.

```
120                                         {
121         this.nombre = nombre;
122     }
```

B.22.3.9. void Modelo.Parametro.setPrecision (int *precision*)

Método que establece la precisión del tipo de dato de un parámetro

Parámetros

precision	Precisión del tipo de dato de un parámetro

Definición en la línea 84 del archivo Parametro.java.

```
84                                          {
85          this.precision = precision;
86      }
```

B.22.3.10. void Modelo.Parametro.setTipoDato (String *tipoDato*)

Método que establece el tipo de dato de un parámetro

Parámetros

tipoDato	Tipo de dato de un parámetro

Definición en la línea 140 del archivo Parametro.java.

```
140                                         {
141         this.tipoDato = tipoDato;
142     }
```

B.22.3.11. void Modelo.Parametro.setTipoES (String *tipoES*)

Método que establece el tipo de entrada de un parámetro

Parámetros

tipoES	Tipo de entrada de un parámetro

Definición en la línea 102 del archivo Parametro.java.

```
102                                         {
103         this.tipoES = tipoES;
104     }
```

B.22.4. Documentación de los datos miembro

B.22.4.1. boolean Modelo.Parametro.llave [private]

propiedad Llave

- true indica que el parámetro representa a una columna que es una llave
- false indica que el parámetro no representa a una columna que es una llave

Definición en la línea 44 del archivo Parametro.java.

B.22.4.2. String Modelo.Parametro.nombre [private]

Nombre del parámetro

Definición en la línea 26 del archivo Parametro.java.

B.22.4.3. int Modelo.Parametro.precision [private]

Precisión del tipo de dato de un parámetro

Definición en la línea 34 del archivo Parametro.java.

B.22.4.4. String Modelo.Parametro.tipoDato [private]

Tipo de dato de un parámetro

Definición en la línea 30 del archivo Parametro.java.

B.22.4.5. String Modelo.Parametro.tipoES [private]

Tipo de entrada de un parámetro

Definición en la línea 22 del archivo Parametro.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/Parametro.java

B.23. Referencia de la Clase Modelo.Parametros

Diagrama de herencias de Modelo.Parametros

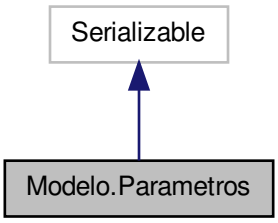
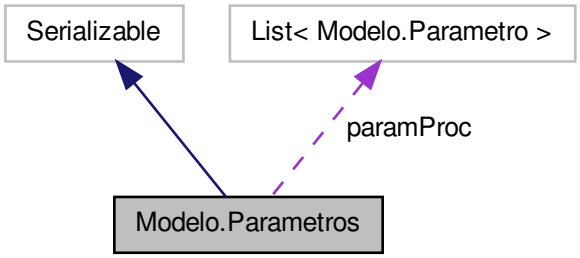


Diagrama de colaboración para Modelo.Parametros:



Métodos públicos

- int getModParam ()
- List getParamProc ()
- void setParamProc (List paramProc)
- Parametros ()
- boolean agregarParametro (Parametro P)
- boolean quitarParametro (Parametro P)
- int numElementos ()
- boolean listaParamVacia ()
- Parametro obtenerParametro (int indice)
- void modificarParametro (int indice)
- String getNombreParametrosSQL ()
- Parametros getParametros (String TipoES)

- Parametros getParametros (String nombreLlave, boolean llave)
- String generarSQL ()
- void vaciar ()
- boolean contiene (String nombreParametro)
- boolean contiene (Parametro p)

Atributos privados

- List< Parametro > paramProc
- int modificarParametro

B.23.1. Descripción detallada

Clase que implementa una lista de parámetros

Autor

ivan

Definición en la línea 17 del archivo Parametros.java.

B.23.2. Documentación del constructor y destructor

B.23.2.1. Modelo.Parametros.Parametros ( )

Constructor por defecto de la clase

Definición en la línea 58 del archivo Parametros.java.

```
58                          {
59          this.setParamProc(new ArrayList());
60      }
```

Gráfico de llamadas a esta función:

B.23.3. Documentación de las funciones miembro

B.23.3.1. boolean Modelo.Parametros.agregarParametro (Parametro *P*)

Método que agrega un parámetro a la lista

Parámetros

P	Parámetro

Devuelve

boolean

- true indica que se agrego el parámetro a la lista
- false indica que no se agrego el parámetro a la lista

Definición en la línea 71 del archivo Parametros.java.

```
71                                                  {
72          return this.paramProc.add(P);
73      }
```

B.23.3.2. boolean Modelo.Parametros.contiene (String *nombreParametro*)

Método que verifica en la lista de parámetros, si contiene el parámetro asociado al nombre proporcionado

Parámetros

nombre-Parametro	Nombre de un parámetro

Devuelve

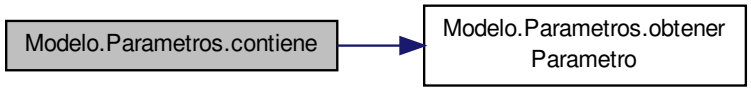
boolean

- true el nombre del parámetro porporcionado coincide con algún parámetro en la lista
- false el nombre del parámetro proporcionado no coincide con ningún parámetro en la lista

Definición en la línea 223 del archivo Parametros.java.

```
223                                                     {
224         boolean tmp = false;
225         for (int i = 0; i < this.getParamProc().size(); i++) {
226             if (this.obtenerParametro(i).getNombre().equals(nombreParametro)) {
227                 tmp = true;
228                 break;
229             }
230         }
231         return tmp;
232
233     }
```

Gráfico de llamadas para esta función:



B.23.3.3. boolean Modelo.Parametros.contiene (Parametro *p*)

Método que verifica en la lista de parámetros, si contiene el parámetro proporcionado

Parámetros

Parametro	Parámetro proporcionado

Devuelve

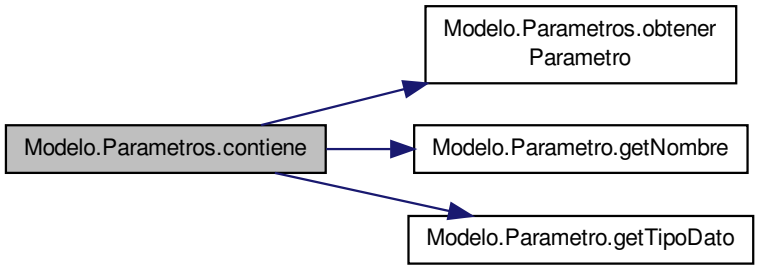
boolean

- true el parámetro porporcionado coincide con algún parámetro en la lista
- false el parámetro proporcionado no coincide con ningún parámetro en la lista

Definición en la línea 247 del archivo Parametros.java.

```
247                                         {
248         boolean tmp = false;
249         for (int i = 0; i < this.getParamProc().size(); i++) {
250             if (this.obtenerParametro(i).getNombre().equals(p.getNombre())
251                     && this.obtenerParametro(i).getTipoDato().equals(p.getTipoDato())) {
252                 tmp = true;
253                 break;
254             }
255         }
256         return tmp;
257     }
```

Gráfico de llamadas para esta función:



B.23.3.4. String Modelo.Parametros.generarSQL ( )

Método que genera y retorna el código SQL asociado a la lista de parámetros

Devuelve

String código SQL asociado a la lista de parámetros

Definición en la línea 190 del archivo Parametros.java.

```
190                                 {
191         Parametro item;
192         String SQL = "";
193         if (!this.paramProc.isEmpty()) {
194             for (int i = 0; i < this.getParamProc().size() - 1; i++) {
```

```
195                 item = (Parametro) this.getParamProc().get(i);
196                 SQL = SQL + item.generarSQL() + ", ";
197             }
198             item = (Parametro) this.getParamProc().get(this.getParamProc().size() - 1);
199             SQL = SQL + item.generarSQL();
200         }
201         return SQL;
202     }
```

Gráfico de llamadas para esta función:

B.23.3.5. int Modelo.Parametros.getModParam ( )

Método que obtiene el parámetro candidato a ser modificado

Devuelve

int indice del parámetro candidato a ser modificado

Definición en la línea 33 del archivo Parametros.java.

```
33                              {
34         return this.modificarParametro;
35     }
```

B.23.3.6. String Modelo.Parametros.getNombreParametrosSQL ( )

Método que genera y retorna el código SQL asociado al nombre del parámetro

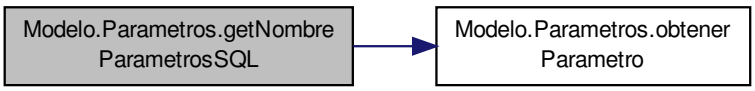
Devuelve

String código SQL asociado al nombre del parámetro

Definición en la línea 135 del archivo Parametros.java.

```
135                                             {
136         String SQL = "";
137         for (int i = 0; i < this.getParamProc().size(); i++) {
138             SQL = (i == 0) ? SQL + "`" + this.obtenerParametro(i).getNombre() + "`\n"
139                     : SQL + ",`" + this.obtenerParametro(i).getNombre() + "`\n";
140         }
141         return SQL;
142     }
```

Gráfico de llamadas para esta función:



B.23.3.7. Parametros Modelo.Parametros.getParametros (String *TipoES*)

Método que obtiene una lista de parámetros filtrados de acuerdo al tipo de entrada

Parámetros

TipoES	tipo de entrada de un parámetro

Devuelve

Parametros Lista de parámetros

Definición en la línea 151 del archivo Parametros.java.

```
151                                                 {
152         Parametros tmp = new Parametros();
153         for (int i = 0; i < this.getParamProc().size(); i++) {
154             if (this.obtenerParametro(i).getTipoES().equals(TipoES)) {
155                 tmp.agregarParametro(this.obtenerParametro(i));
156             }
157         }
158         return tmp;
159     }
```

Gráfico de llamadas para esta función:

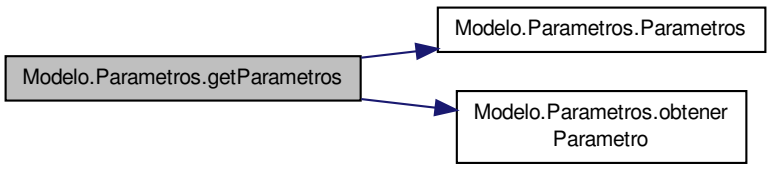
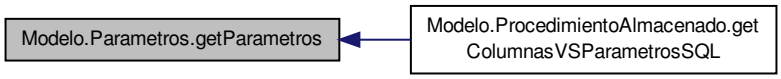


Gráfico de llamadas a esta función:



B.23.3.8. Parametros Modelo.Parametros.getParametros (String *nombreLlave,* boolean *llave*)

Método que obtiene una lista de parámetros filtrados de acuerdo al nombre y al atributo llave de un parámetro

Parámetros

nombreLlave	nombre de la columna que es llave y se desea buscar en la lista de parametros
llave	atributo llave de un parámetro

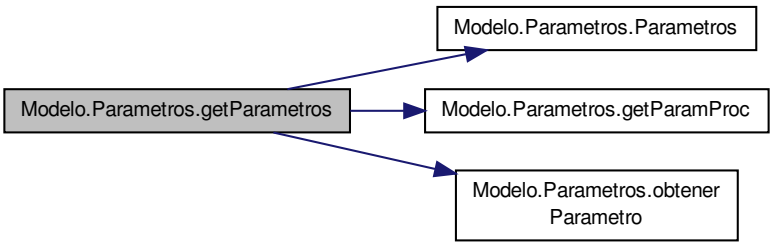
Devuelve

Parametros Lista de parámetros

Definición en la línea 170 del archivo Parametros.java.

```
170                                                                     {
171         Parametros tmp = new Parametros();
172         if (llave) {
173             tmp = (Parametros) this.getParamProc();
174         } else {
175             for (int i = 0; i < this.getParamProc().size(); i++) {
176                 if (!this.obtenerParametro(i).getNombre().equals(nombreLlave)) {
177                     tmp.agregarParametro(this.obtenerParametro(i));
178                 }
179             }
180         }
181         return tmp;
182     }
```

Gráfico de llamadas para esta función:



B.23.3.9. List Modelo.Parametros.getParamProc ( )

Método que obtiene la lista de parámetros

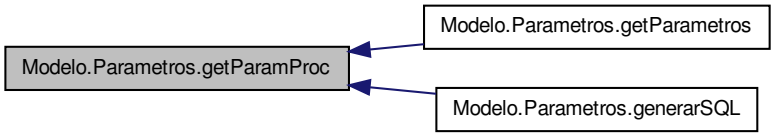
Devuelve

List Lista de parametros

Definición en la línea 42 del archivo Parametros.java.

```
42                                {
43          return this.paramProc;
44      }
```

Gráfico de llamadas a esta función:



B.23.3.10. boolean Modelo.Parametros.listaParamVacia ( )

Método que idica si la lista de parámetros está vaciá

Devuelve

boolean

- true la lista de parámetros está vaciá
- false la lista contiene al menos un parámetro

Definición en la línea 105 del archivo Parametros.java.

```
105                                    {
106         return this.paramProc.isEmpty();
107     }
```

B.23.3.11. void Modelo.Parametros.modificarParametro (int *indice*)

Método que establece el parámetro candidato a ser modificado

Parámetros

indice	indice del parámetro candidato a ser modificado

Definición en la línea 125 del archivo Parametros.java.

```
125                                             {
126         this.modificarParametro = indice;
127     }
```

B.23.3.12. int Modelo.Parametros.numElementos ( )

Método que proporciona el número de parámetros de la lista

Devuelve

int número de parámetros de la lista

Definición en la línea 93 del archivo Parametros.java.

```
93                              {
94         return this.paramProc.size();
95     }
```

B.23.3.13. Parametro Modelo.Parametros.obtenerParametro (int *indice*)

Método que obtiene un parámetro de la lista, a partir de un indice proporcionado

Parámetros

indice	indice asociado al parámetro a obtener

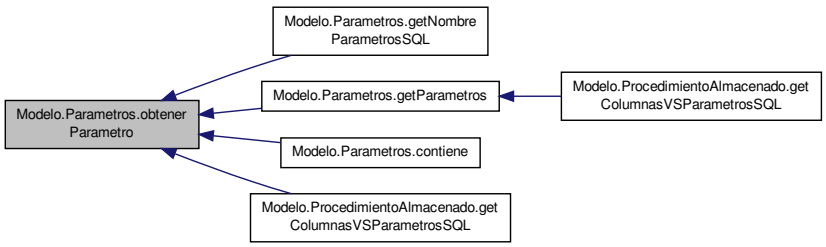
Devuelve

Parametro Parámetro obtenido

Definición en la línea 116 del archivo Parametros.java.

```
116                                             {
117         return this.paramProc.get(indice);
118     }
```

Gráfico de llamadas a esta función:



B.23.3.14. boolean Modelo.Parametros.quitarParametro (Parametro *P*)

Método que remueve un parámetro de la lista

Parámetros

P	Parámetro

Devuelve

boolean

- true indica que se removió el parámetro de la lista
- false indica que no se removió el parámetro de la lista

Definición en la línea 84 del archivo Parametros.java.

```
84                                              {
85         return this.paramProc.remove(P);
86     }
```

B.23.3.15. void Modelo.Parametros.setParamProc (List *paramProc*)

Método que establece la lista de parámetros

Parámetros

paramProc	Lista de parámetros

Definición en la línea 51 del archivo Parametros.java.

```
51                                             {
52         this.paramProc = paramProc;
53     }
```

B.23.3.16. void Modelo.Parametros.vaciar ( )

Método que vacía la lista de parámetros

Definición en la línea 207 del archivo Parametros.java.

```
207                           {
208         this.paramProc.clear();
209     }
```

B.23.4. Documentación de los datos miembro

B.23.4.1. int Modelo.Parametros.modificarParametro [private]

Indice que indica el parámetro candidato a ser modificado

Definición en la línea 26 del archivo Parametros.java.

B.23.4.2. List<Parametro> Modelo.Parametros.paramProc [private]

Lista de parámetros

Definición en la línea 22 del archivo Parametros.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/Parametros.java

B.24. Referencia de la Clase Modelo.ProcedimientoAlmacenado

Diagrama de herencias de Modelo.ProcedimientoAlmacenado

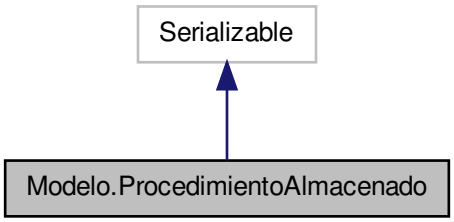
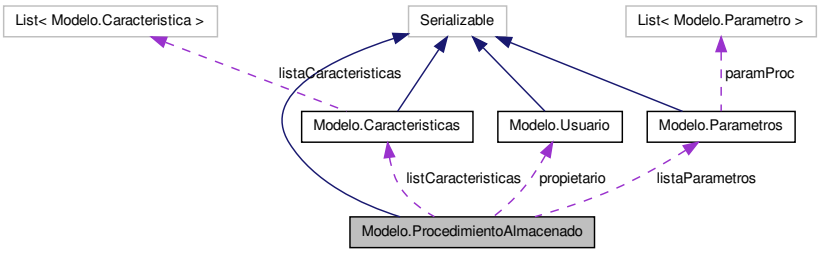


Diagrama de colaboración para Modelo.ProcedimientoAlmacenado:



Métodos públicos

- ProcedimientoAlmacenado ()
- ProcedimientoAlmacenado (String nombreProcedimiento)
- void pordefecto ()
- String getNombre ()
- void setNombre (String nombre)
- Usuario getUsuario ()
- void setUsuario (Usuario propietario)
- Parametros getParametros ()
- void setParametros (Parametros listaParametros)
- Caracteristicas getCaracteristicas ()
- void setCaracteristicas (Caracteristicas ListaCaracteristicas)
- String getCuerpoRutina (boolean visualizador)
- void setCuerpoRutina (String cuerpoRutina)
- String getParametrosSQL ()
- String getParametrosSQL (String tipoES)

- String generarProcedimientoSQL (boolean visualizador)
- String getColumnasVSParametrosSQL (tabla Tabla, Parametros parametros, String separador)
- String getColumnasVSParametrosSQL (tabla Tabla, Parametros parametros, String separador, boolean llave)
- String existeRegistro (tabla Tabla)
- String insertarRegistro (tabla Tabla)
- String ControldeFlujoIF (tabla Tabla)
- String generarProcedimientoAlta (tabla Tabla)
- String generarProcedimientoBaja (tabla Tabla)
- String generarProcedimientoCambio (tabla Tabla)
- String generarProcedimientoConsulta (tabla Tabla)
- String generarProcedmientoFuncionCRUD (tabla Tabla, funcionBasicaBD CRUD)
- String formatearCuerpoRutina (String SQL, boolean visualizador)
- String toString ()

Atributos públicos

- String nombre
- Parametros listaParametros
- Caracteristicas listCaracteristicas
- String cuerpoRutina

Atributos privados

- Usuario propietario

B.24.1. Descripción detallada

Clase que implementa un procedimiento almacenado

Autor

ivan

Definición en la línea 16 del archivo ProcedimientoAlmacenado.java.

B.24.2. Documentación del constructor y destructor

B.24.2.1. Modelo.ProcedimientoAlmacenado.ProcedimientoAlmacenado ( )

Constructor por defecto de la clase

Definición en la línea 43 del archivo ProcedimientoAlmacenado.java.

```
43                                      {
44         this.setCaracteristicas(new Caracteristicas());
45     }
```

B.24.2.2. Modelo.ProcedimientoAlmacenado.ProcedimientoAlmacenado (String *nombreProcedimiento*)

Sobrecarga del constructor de la clase, establece el nombre del procedimiento almacenado

Parámetros

nombre-Procedimiento	Nombre del procedimiento almacenado

Definición en la línea 53 del archivo ProcedimientoAlmacenado.java.

```
53                                                          {
54         this.setNombre(nombreProcedimiento);
55     }
```

B.24.3. Documentación de las funciones miembro

B.24.3.1. String Modelo.ProcedimientoAlmacenado.ControldeFlujoIF (tabla *Tabla*)

Método que genera y retorna código SQL que implementa el módulo "Control de flujo"

Parámetros

Tabla	Tabla

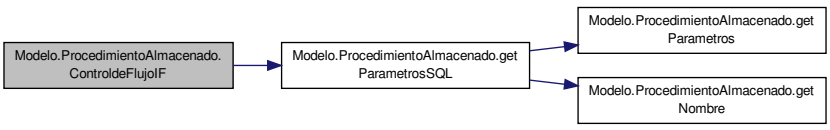
Devuelve

String Código SQL

Definición en la línea 334 del archivo ProcedimientoAlmacenado.java.

```
334                                                 {
335         String SQL = "IF EXISTS(\n";
336         SQL = SQL + this.existeRegistro(Tabla);
337         SQL = SQL + ")\n";
338         SQL = SQL + "THEN\n";
339         SQL = SQL + this.existeRegistro(Tabla);
340         SQL = (this.getParametrosSQL("OUT").equals("")) ? SQL + " " : SQL + "INTO " + this.
      getParametrosSQL("OUT") + ";\n";
341         SQL = SQL + "ELSE \n";
342         SQL = SQL + this.insertarRegistro(Tabla);
343         SQL = (this.getParametrosSQL("OUT").equals("")) ? SQL + " " : SQL + "SELECT LAST_INSERT_ID() INTO "
       + this.getParametrosSQL("OUT") + ";\n";
344         SQL = SQL + "END IF;";
345         return SQL;
346     }
```

Gráfico de llamadas para esta función:



B.24.3.2. String Modelo.ProcedimientoAlmacenado.existeRegistro (tabla *Tabla*)

Método que genera y retorna código SQL que implementa la operación "Exite registro"

Parámetros

Tabla	Tabla

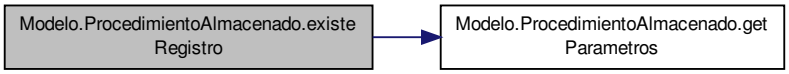
Devuelve

String Código SQL

Definición en la línea 302 del archivo ProcedimientoAlmacenado.java.

```
302                                                 {
303         String SQL = "SELECT " + Tabla.getLlavePrimariaSQL() + "\n";
304         SQL = SQL + "FROM " + Tabla.getNombreTablaSQL() + "\n";
305         SQL = SQL + "WHERE \n";
306         SQL = SQL + this.getColumnasVSParametrosSQL(Tabla, this.getParametros().
      getParametros("IN"), "AND");
307         return SQL;
308     }
```

Gráfico de llamadas para esta función:



B.24.3.3. String Modelo.ProcedimientoAlmacenado.formatearCuerpoRutina (String *SQL*, boolean *visualizador*)

Método que implementa el módulo "Formatear código SQL del cuerpo de la rutina del procedimiento almacenado"

Parámetros

SQL	Código SQL del cuerpo de la rutina del procedimiento almacenado
visualizador	■ true indica que el código SQL aparte de ser formateado contendrá etiquetas html para el módulo "visualizador de código". ■ false indica que el código SQL solo será formateado, no contendra etiquetas html y podra ser usado ya sea para enviar directamenta a la base de datos o en un archivo.

Devuelve

String Código SQL formateado del cuerpo de la rutina del procedimiento almacenado

Definición en la línea 460 del archivo ProcedimientoAlmacenado.java.

```
460                                                                 {
461         ArrayList<String> palabrasreservadas = new ArrayList();
462         palabrasreservadas.add("ALTER");
463         palabrasreservadas.add("CALL");
464         palabrasreservadas.add("CREATE");
465         palabrasreservadas.add("DELETE");
466         palabrasreservadas.add("DROP");
```

```
467         palabrasreservadas.add("ELSE");
468         palabrasreservadas.add("ELSEIF");
469         palabrasreservadas.add("FOR");
470         palabrasreservadas.add("FROM");
471         palabrasreservadas.add("GROUP");
472         palabrasreservadas.add("HAVING");
473         palabrasreservadas.add("IF");
474         palabrasreservadas.add("INSERT ");
475         palabrasreservadas.add("SELECT");
476         palabrasreservadas.add("THEN");
477         palabrasreservadas.add("UPDATE");
478         palabrasreservadas.add("WHERE");
479         palabrasreservadas.add("WHILE");
480         for (int i = 0; i < palabrasreservadas.size(); i++) {
481             SQL = SQL.replaceAll(palabrasreservadas.get(i), ((visualizador) ? ("\n" + "<font color=\"red\">
      " + palabrasreservadas.get(i) + "</font>") : ("\n" + palabrasreservadas.get(i))));
482         }
483         SQL = SQL.replaceAll(" +", " ");
484         SQL = SQL.replaceAll("\n+", ((visualizador) ? ("<br>") : ("\n")));
485         return SQL;
486     }
```

B.24.3.4. String Modelo.ProcedimientoAlmacenado.generarProcedimientoAlta (tabla *Tabla*)

Método que genera y retorna código SQL del cuerpo de una rutina que implementa la función básica "ALTA" en bases de datos

Parámetros

Tabla	Tabla

Devuelve

String Código SQL

Definición en la línea 355 del archivo ProcedimientoAlmacenado.java.

```
355                                                 {
356         return this.ControldeFlujoIF(Tabla);
357     }
```

B.24.3.5. String Modelo.ProcedimientoAlmacenado.generarProcedimientoBaja (tabla *Tabla*)

Método que genera y retorna código SQL del cuerpo de una rutina que implementa la función básica "BAJA" en bases de datos

Parámetros

Tabla	Tabla

Devuelve

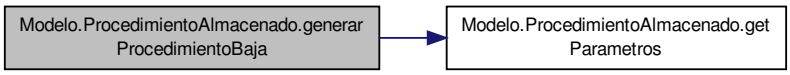
String Código SQL

Definición en la línea 366 del archivo ProcedimientoAlmacenado.java.

```
366                                                 {
367         String SQL = "";
368         SQL = "DELETE\n";
369         SQL = SQL + "FROM " + Tabla.getNombreTablaSQL() + "\n";
370         SQL = SQL + "WHERE (\n";
371         SQL = SQL + this.getColumnasVSParametrosSQL(Tabla, this.getParametros().
```

```
      getParametros("IN"), "AND");
372         SQL = SQL + ");\n";
373         return SQL;
374     }
```

Gráfico de llamadas para esta función:



B.24.3.6. String Modelo.ProcedimientoAlmacenado.generarProcedimientoCambio (tabla *Tabla*)

Método que genera y retorna código SQL del cuerpo de una rutina que implementa la función básica "CAMBIO" en bases de datos

Parámetros

Tabla	Tabla

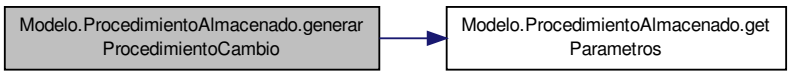
Devuelve

String Código SQL

Definición en la línea 383 del archivo ProcedimientoAlmacenado.java.

```
383                                                     {
384         String SQL = "";
385         SQL = "UPDATE\n";
386         SQL = SQL + Tabla.getNombreTablaSQL() + "\n";
387         SQL = SQL + "SET\n";
388         SQL = SQL + this.getColumnasVSParametrosSQL(Tabla, this.getParametros().
      getParametros("IN"), ",", false);
389         SQL = SQL + "WHERE (\n";
390         SQL = (this.getParametros().contiene(Tabla.getLlavePrimaria().getNombreColumna())) ? (SQL + Tabla.
      getLlavePrimariaSQL() + " = " + Tabla.getLlavePrimaria().getNombreColumna() + "\n") : (SQL + "la columna id
       de la tabla debe ser un parametro\n");
391         SQL = SQL + ");";
392         return SQL;
393     }
```

Gráfico de llamadas para esta función:



B.24.3.7. String Modelo.ProcedimientoAlmacenado.generarProcedimientoConsulta (tabla *Tabla*)

Método que genera y retorna código SQL del cuerpo de una rutina que implementa la función básica "CONSULTA" en bases de datos

Parámetros

Tabla	Tabla

Devuelve

String Código SQL

Definición en la línea 402 del archivo ProcedimientoAlmacenado.java.

```
402                                                     {
403         String SQL = "";
404         SQL = "SELECT\n";
405         SQL = SQL + this.getParametrosSQL("OUT") + "\n";
406         SQL = SQL + "FROM " + Tabla.getNombreTablaSQL() + "\n";
407         SQL = SQL + "WHERE (\n";
408         SQL = SQL + Tabla.getLlavePrimariaSQL() + " = " + this.getParametrosSQL("IN") + "\n
    ";
409 //         SQL = SQL + this.getColumnasVSParametrosSQL(Tabla, this.getParametros().getParametros("IN"),
     "AND");
410         SQL = SQL + ");\n";
411         return SQL;
412     }
```

Gráfico de llamadas para esta función:

B.24.3.8. String Modelo.ProcedimientoAlmacenado.generarProcedimientoSQL (boolean *visualizador*)

Método que genera y retorna código SQL asociado al procedimiento almacenado, ya sea con formato para la visualizador de codigo o no.

Parámetros

visualizador	■ true indica que el código SQL asociado al procedimiento contendra etiquetas html para el módulo "visualizador de código". ■ false indica que el código SQL aosicado al procedimiento no contendra etiquetas html y podra ser usado ya sea para enviar directamenta a la base de datos o en un archivo.

Devuelve

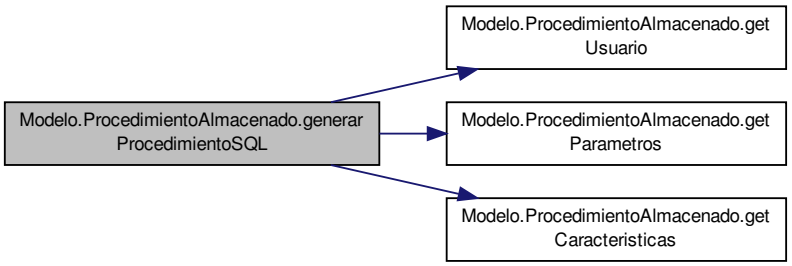
String código SQL asociado al procedimiento almacenado

Definición en la línea 226 del archivo ProcedimientoAlmacenado.java.

```
226                                                     {
227         String SQL = "CREATE ";
```

```
228
229         if (this.getUsuario() != null) {
230             SQL = SQL + this.getUsuario().generarSQL();
231         }
232
233         SQL = SQL + " PROCEDURE " + this.getNombre();
234
235         if (this.getParametros() == null) {
236             SQL = SQL + " ( )\n";
237         } else {
238             SQL = SQL + "\n ( " + this.getParametros().generarSQL() + " )\n";
239         }
240
241         if (this.getCaracteristicas() != null) {
242             SQL = SQL + this.getCaracteristicas().generarSQL() + "\n";
243         }
244
245         SQL = SQL + "BEGIN\n" + this.getCuerpoRutina(visualizador) + "\nEND\n";
246         return SQL;
247     }
```

Gráfico de llamadas para esta función:



B.24.3.9. String Modelo.ProcedimientoAlmacenado.generarProcedmientoFuncionCRUD (tabla *Tabla,* funcionBasicaBD *CRUD*)

Método que retorna código SQL del cuerpo de una rutina que implementa una función básica en base de datos asociada a una tabla

Parámetros

Tabla	Tabla
CRUD	Función básica en base de datos

Devuelve

String Código SQL

Definición en la línea 422 del archivo ProcedimientoAlmacenado.java.

```
422                                                                                 {
423         String SQLPCRUD = "";
424         switch (CRUD) {
425             case ALTA:
426                 SQLPCRUD = this.generarProcedimientoAlta(Tabla);
427                 break;
428             case BAJA:
```

```
429                 SQLPCRUD = this.generarProcedimientoBaja(Tabla);
430                 break;
431             case CAMBIO:
432                 SQLPCRUD = this.generarProcedimientoCambio(Tabla);
433                 break;
434             case CONSULTA:
435                 SQLPCRUD = this.generarProcedimientoConsulta(Tabla);
436                 break;
437             default:
438                 System.out.println("Error, no es un operacion CRUD");
439         }
440         return SQLPCRUD;
441     }
```

B.24.3.10. Caracteristicas Modelo.ProcedimientoAlmacenado.getCaracteristicas ( )

Método que obtiene las características del procedimiento almacenado

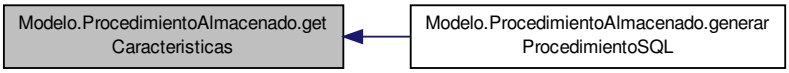
Devuelve

Caracteristicas Lista de características del procedimiento almacenado

Definición en la línea 132 del archivo ProcedimientoAlmacenado.java.

```
132                                             {
133         return this.listCaracteristicas;
134     }
```

Gráfico de llamadas a esta función:



B.24.3.11. String Modelo.ProcedimientoAlmacenado.getColumnasVSParametrosSQL (tabla *Tabla,* Parametros *parametros,* String *separador*)

Método que retorna el código SQL asociado a los parámetros que hacen referencias a columnas separados por un delimitador proporcionado por el usuario

Parámetros

Tabla	Tabla
parametros	Lista de Parámetros
separador	Separador para delimitar los parametros

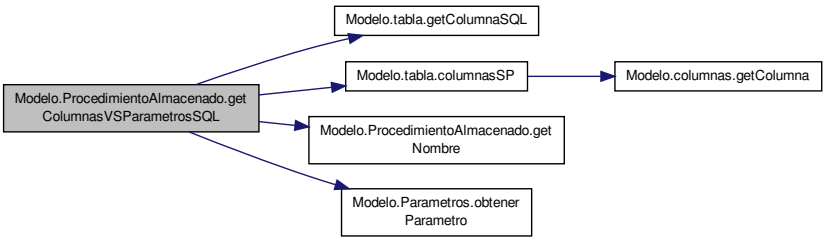
Devuelve

String código SQL

Definición en la línea 259 del archivo ProcedimientoAlmacenado.java.

```
259                                                                                                                 {
260         String SQL = "";
261         for (int i = 0; i < Tabla.columnasSP().numeroColumnas(); i++) {
262             SQL = (i == 0) ? SQL + Tabla.getColumnaSQL(Tabla.columnasSP().getColumna(i), true) + " = " +
    parametros.obtenerParametro(i).getNombre() + "\n"
263                     : SQL + separador + " " + Tabla.getColumnaSQL(Tabla.columnasSP().getColumna(i), true) +
     " = " + parametros.obtenerParametro(i).getNombre() + "\n";
264         }
265         return SQL;
266     }
```

Gráfico de llamadas para esta función:



B.24.3.12. String Modelo.ProcedimientoAlmacenado.getColumnasVSParametrosSQL (tabla *Tabla,* Parametros *parametros,* String *separador,* boolean *llave*)

Método que retorna el código SQL asociado a los parámetros que hacen referencias a columnas filtradas por el atributo llave.

Parámetros

Tabla	Tabla
parametros	Lista de Parámetros
separador	Separador para delimitar los parametros
llave	Propiedad llave

Devuelve

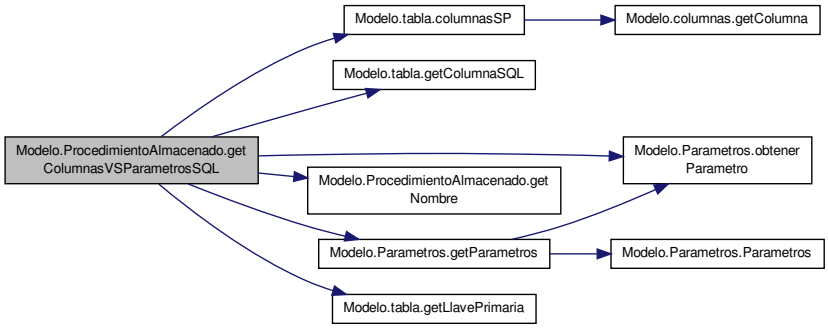
String código SQL

Definición en la línea 278 del archivo ProcedimientoAlmacenado.java.

```
278                                                                                                                 {
279         String SQL = "";
280         if (llave) {
281             for (int i = 0; i < Tabla.columnasSP().numeroColumnas(); i++) {
282                 SQL = (SQL.equals("")) ? SQL + Tabla.getColumnaSQL(Tabla.columnasSP().getColumna(i), true)
    + " = " + parametros.obtenerParametro(i).getNombre() + "\n"
283                         : SQL + separador + " " + Tabla.getColumnaSQL(Tabla.columnasSP().getColumna(i),
    true) + " = " + parametros.obtenerParametro(i).getNombre() + "\n";
284             }
285         } else {
286
287             for (int i = 0; i < Tabla.columnasSP().numeroColumnas(); i++) {
288                 SQL = (SQL.equals("")) ? SQL + Tabla.getColumnaSQL(Tabla.columnasSP().getColumna(i), true)
    + " = " + parametros.getParametros(Tabla.getLlavePrimaria().getNombreColumna(), false).obtenerParametro(i).
    getNombre() + "\n"
```

```
289                         : SQL + separador + " " + Tabla.getColumnaSQL(Tabla.columnasSP().getColumna(i),
      true) + " = " + parametros.getParametros(Tabla.getLlavePrimaria().getNombreColumna(), false).obtenerParametro(i
      ).getNombre() + "\n";
290             }
291         }
292         return SQL;
293     }
```

Gráfico de llamadas para esta función:



B.24.3.13. String Modelo.ProcedimientoAlmacenado.getCuerpoRutina (boolean *visualizador*)

Método que obtiene el cuerpo de la rutina de un procedimiento almacenado, con o sin etiquetas html.

Parámetros

visualizador	■ true indica que el código SQL asociado al cuerpo de la rutina contendra etiquetas html. ■ false indica que el código SQL aosicado al cuerpo de la rutina no contendra etiquetas html.

Devuelve

String Cuerpo de la rutina de un procedimiento almacenado

Definición en la línea 159 del archivo ProcedimientoAlmacenado.java.

```
159                                                 {
160         return this.formatearCuerpoRutina(this.cuerpoRutina, visualizador);
161     }
```

B.24.3.14. String Modelo.ProcedimientoAlmacenado.getNombre ( )

Método que obtiene el nombre del procedimiento almacenado

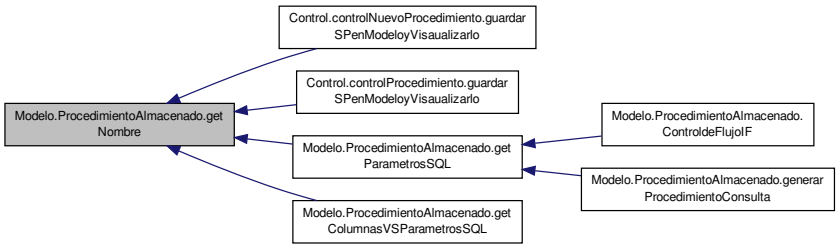
Devuelve

String Nombre del procedimiento almacenado

Definición en la línea 73 del archivo ProcedimientoAlmacenado.java.

```
73                                  {
74          return this.nombre;
75      }
```

Gráfico de llamadas a esta función:



B.24.3.15. Parametros Modelo.ProcedimientoAlmacenado.getParametros ( )

Método que obtiene la lista de parámetros del procedimiento almacenado

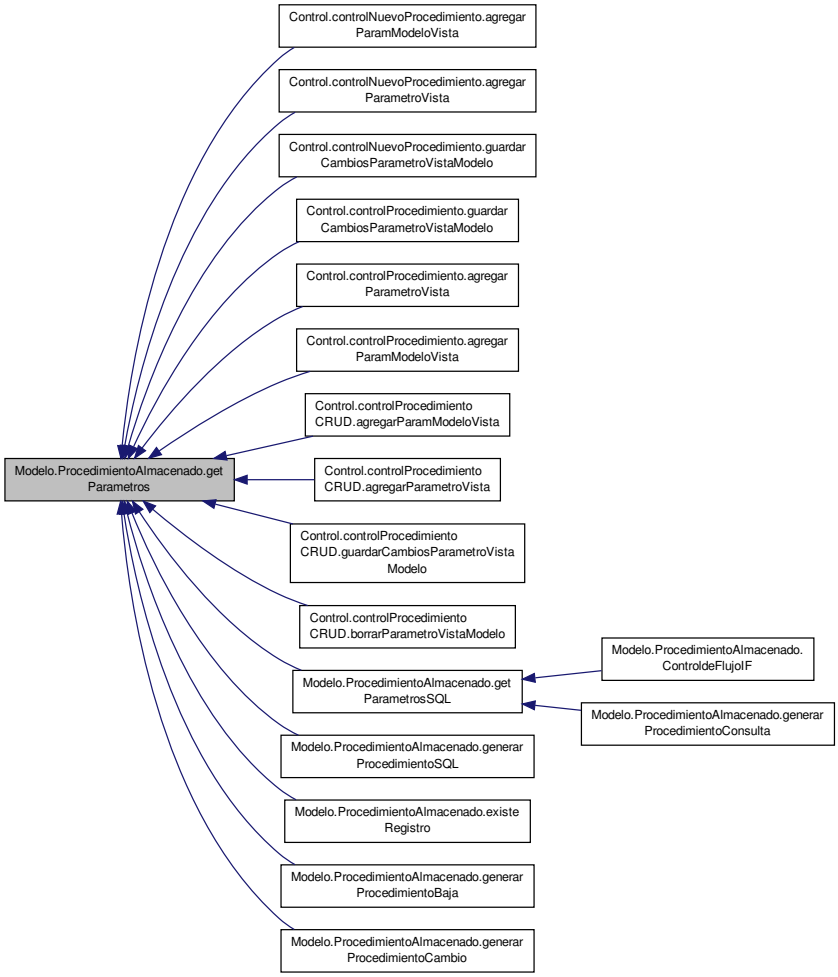
Devuelve

Parametros Lista de parámetros de un procedimiento almacenado

Definición en la línea 113 del archivo ProcedimientoAlmacenado.java.

```
113                                     {
114         return this.listaParametros;
115     }
```

Gráfico de llamadas a esta función:



B.24.3.16. String Modelo.ProcedimientoAlmacenado.getParametrosSQL ( )

Método que genera y retorna el código SQL asociado a la lista de parámetros de un procedimiento almacenado

Devuelve

String Código SQL asociado a la lista de parámetros de un procedimiento almacenado

Definición en la línea 180 del archivo ProcedimientoAlmacenado.java.

```
180                                     {
181         String SQL = "";
182         for (int i = 0; i < this.getParametros().numElementos(); i++) {
183             if (this.getParametros().obtenerParametro(i).getTipoES().equals("IN")) {
184                 SQL = (SQL.equals("")) ? SQL + this.getParametros().obtenerParametro(i).getNombre() + "\n"
      : SQL + "," + this.getParametros().obtenerParametro(i).getNombre() + "\n";
185             }
186         }
```

```
187         return SQL;
188     }
```

Gráfico de llamadas para esta función:

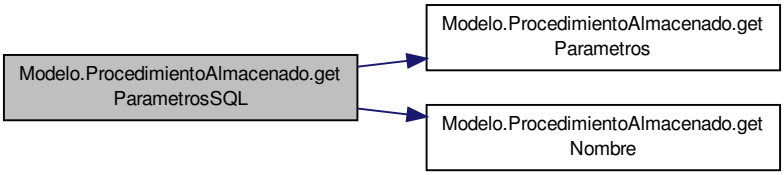
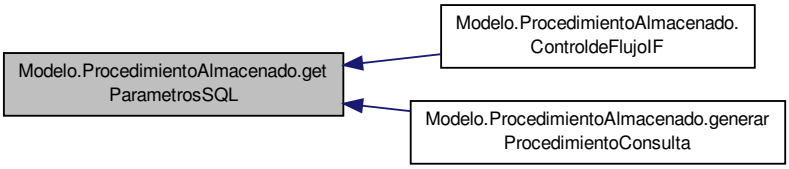


Gráfico de llamadas a esta función:



B.24.3.17. String Modelo.ProcedimientoAlmacenado.getParametrosSQL (String *tipoES*)

Método que genera y retorna el código SQL asociado a la lista de parámetros filtrados por el tipo de entrada

Parámetros

tipoES	Tipo de entrada de un parametro

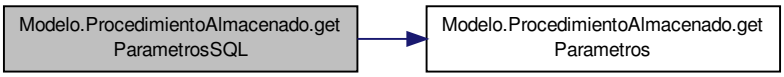
Devuelve

String Código SQL asociado a la lista de parámetros de un procedimiento almacenado

Definición en la línea 198 del archivo ProcedimientoAlmacenado.java.

```
198                                                 {
199         String SQL = "";
200         for (int i = 0; i < this.getParametros().numElementos(); i++) {
201             if (this.getParametros().obtenerParametro(i).getTipoES().equals(tipoES)) {
202                 if (SQL.equals("")) {
203                     SQL = SQL + this.getParametros().obtenerParametro(i).getNombre();
204                 } else {
205                     SQL = SQL + ", " + this.getParametros().obtenerParametro(i).getNombre();
206                 }
207             }
208         }
209         return SQL;
210     }
```

Gráfico de llamadas para esta función:



B.24.3.18. Usuario Modelo.ProcedimientoAlmacenado.getUsuario ( )

Método que obtiene el usuario del cual se usaran sus privilegios al ejecutar el procedimiento almacenado

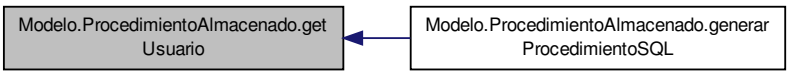
Devuelve

Usuario usuario del cual se usaran sus privilegios al ejecutar el procedimiento almacenado

Definición en la línea 93 del archivo ProcedimientoAlmacenado.java.

```
93                                  {
94          return this.propietario;
95      }
```

Gráfico de llamadas a esta función:



B.24.3.19. String Modelo.ProcedimientoAlmacenado.insertarRegistro (tabla *Tabla*)

Método que genera y retorna código SQL que implementa la operación "Insertar registro"

Parámetros

Tabla	Tabla

Devuelve

String Código SQL

Definición en la línea 317 del archivo ProcedimientoAlmacenado.java.

```
317                                             {
318         String SQL = "INSERT INTO " + Tabla.getNombreTablaSQL() + " (\n";
319         SQL = SQL + Tabla.getColumnasSQL(false);
320         SQL = SQL + ")\n";
```

```
321         SQL = SQL + "VALUES (\n";
322         SQL = SQL + this.getParametrosSQL();
323         SQL = SQL + ");\n";
324         return SQL;
325     }
```

B.24.3.20. void Modelo.ProcedimientoAlmacenado.pordefecto ( )

Método que establece el estado por defecto de un procedimiento almacenado

Definición en la línea 60 del archivo ProcedimientoAlmacenado.java.

```
60                              {
61         this.setNombre(null);
62         this.setUsuario(null);
63         this.setParametros(new Parametros());
64         this.setCaracteristicas(new Caracteristicas());
65         this.setCuerpoRutina(null);
66     }
```

B.24.3.21. void Modelo.ProcedimientoAlmacenado.setCaracteristicas (Caracteristicas *ListaCaracteristicas*)

Método que establece las características del procedimiento almacenado

Parámetros

Lista-Caracteristicas	Lista de características del procedimiento almacenado

Definición en la línea 142 del archivo ProcedimientoAlmacenado.java.

```
142                                                                 {
143         this.listCaracteristicas = ListaCaracteristicas;
144     }
```

B.24.3.22. void Modelo.ProcedimientoAlmacenado.setCuerpoRutina (String *cuerpoRutina*)

Método que establece el cuerpo de la rutina de un procedimiento almacenado

Parámetros

cuerpoRutina	Cuerpo de la rutina de un procedimiento almacenado

Definición en la línea 169 del archivo ProcedimientoAlmacenado.java.

```
169                                             {
170         this.cuerpoRutina = cuerpoRutina;
171     }
```

B.24.3.23. void Modelo.ProcedimientoAlmacenado.setNombre (String *nombre*)

Método que establece el nombre del procedimiento almacenado

Parámetros

nombre	Nombre del procedimiento almacenado

Definición en la línea 82 del archivo ProcedimientoAlmacenado.java.

```
82                                          {
83         this.nombre = nombre;
84     }
```

B.24.3.24. void Modelo.ProcedimientoAlmacenado.setParametros (Parametros *listaParametros*)

Método que establece la lista de parámetros del procedimiento almacenado

Parámetros

listaParametros	Lista de parámetros del procedimiento almacenado

Definición en la línea 122 del archivo ProcedimientoAlmacenado.java.

```
122                                                     {
123         this.listaParametros = listaParametros;
124     }
```

B.24.3.25. void Modelo.ProcedimientoAlmacenado.setUsuario (Usuario *propietario*)

Método que establece el usuario del cual se usaran sus privilegios al ejecutar el procedimiento almacenado

Parámetros

Usuario	usuario del cual se usaran sus privilegios al ejecutar el procedimiento almacenado

Definición en la línea 104 del archivo ProcedimientoAlmacenado.java.

```
104                                             {
105         this.propietario = propietario;
106     }
```

B.24.3.26. String Modelo.ProcedimientoAlmacenado.toString ( )

Sobreescritura del método toString

Devuelve

Nombre del procedimiento almacenado

Definición en la línea 493 del archivo ProcedimientoAlmacenado.java.

```
493                                 {
494         return this.getNombre();
495     }
```

B.24.4. Documentación de los datos miembro

B.24.4.1. String Modelo.ProcedimientoAlmacenado.cuerpoRutina

Cuerpo de la rutina de un procedimiento almacenado

Definición en la línea 38 del archivo ProcedimientoAlmacenado.java.

B.24.4.2. Parametros Modelo.ProcedimientoAlmacenado.listaParametros

Lista de parámetros de un procedimiento almacenado

Definición en la línea 30 del archivo ProcedimientoAlmacenado.java.

B.24.4.3. Caracteristicas Modelo.ProcedimientoAlmacenado.listCaracteristicas

Lista de características de un procedimiento almacenado

Definición en la línea 34 del archivo ProcedimientoAlmacenado.java.

B.24.4.4. String Modelo.ProcedimientoAlmacenado.nombre

nombre del procedimiento almacenado

Definición en la línea 21 del archivo ProcedimientoAlmacenado.java.

B.24.4.5. Usuario Modelo.ProcedimientoAlmacenado.propietario [private]

Usuario del cual se usaran sus privilegios al ejecutar el procedimiento almacenado

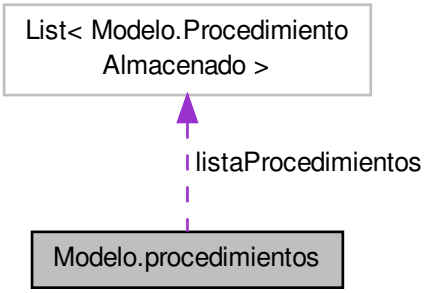
Definición en la línea 26 del archivo ProcedimientoAlmacenado.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/ProcedimientoAlmacenado.java

B.25. Referencia de la Clase Modelo.procedimientos

Diagrama de colaboración para Modelo.procedimientos:



Métodos públicos

- procedimientos ()

- void setListaProcedimientos (List listaProcedimientos)
- List getListaProcedimientos ()
- void agregarProcedimiento (ProcedimientoAlmacenado P)
- ProcedimientoAlmacenado obtenerProcedimiento (int indice)
- int numProcedimientos ()
- boolean vacia ()
- void limpiar ()

Atributos privados

- List< ProcedimientoAlmacenado > listaProcedimientos

B.25.1. Descripción detallada

Clase que implementa una lista de procedimientos almacenados

Autor

ivan

Definición en la línea 16 del archivo procedimientos.java.

B.25.2. Documentación del constructor y destructor

B.25.2.1. Modelo.procedimientos.procedimientos ( )

Constructor por defecto de la clase

Definición en la línea 26 del archivo procedimientos.java.

```
26                          {
27         this.setListaProcedimientos(new ArrayList());
28     }
```

B.25.3. Documentación de las funciones miembro

B.25.3.1. void Modelo.procedimientos.agregarProcedimiento (ProcedimientoAlmacenado *P*)

Método que agrega un procedimiento almacenado a la lista

Parámetros

P	Procedimiento almacenado

Definición en la línea 53 del archivo procedimientos.java.

```
53                                                      {
54         this.listaProcedimientos.add(P);
55     }
```

B.25.3.2. List Modelo.procedimientos.getListaProcedimientos ( )

Método que obtiene la lista de procedimientos almacenados

Devuelve

List lista de procedimientos almacenados

Definición en la línea 44 del archivo procedimientos.java.

```
44                                          {
45          return this.listaProcedimientos;
46      }
```

B.25.3.3. void Modelo.procedimientos.limpiar ( )

Método que vacía la lista de procedimientos almacenados

Definición en la línea 93 del archivo procedimientos.java.

```
93                          {
94          this.listaProcedimientos.clear();
95      }
```

B.25.3.4. int Modelo.procedimientos.numProcedimientos ( )

Método que obtiene el número de procedimientos almacenados de la lista

Devuelve

int Numero de procedimientos almacenados de la lista

Definición en la línea 73 del archivo procedimientos.java.

```
73                                  {
74          return this.listaProcedimientos.size();
75      }
```

B.25.3.5. ProcedimientoAlmacenado Modelo.procedimientos.obtenerProcedimiento (int *indice*)

Método que obtiene un procedimiento almacenado de la lista a partir del indice proporcionado.

Parámetros

indice	Indice

Devuelve

ProcedimientoAlmacenado Procedimiento almacenado

Definición en la línea 64 del archivo procedimientos.java.

```
64                                                                  {
65          return this.listaProcedimientos.get(indice);
66      }
```

B.25.3.6. void Modelo.procedimientos.setListaProcedimientos (List *listaProcedimientos*)

Método que establece la lista de procedimientos almacenados

Parámetros

listaProcedimientos	lista de procedimientos almacenados

Definición en la línea 35 del archivo procedimientos.java.

```
35                                                          {
36         this.listaProcedimientos = listaProcedimientos;
37     }
```

B.25.3.7. boolean Modelo.procedimientos.vacia ( )

Método que verifica si la lista de procedimientos almacenados está vacía

Devuelve

boolean

- true La lista de procedimientos almacenados está vacía.
- false La lista contiene al menos un procedimiento almacenado.

Definición en la línea 86 del archivo procedimientos.java.

```
86                          {
87         return this.listaProcedimientos.isEmpty();
88     }
```

B.25.4. Documentación de los datos miembro

B.25.4.1. List<ProcedimientoAlmacenado> Modelo.procedimientos.listaProcedimientos [private]

Lista de procedimeintos almacenados

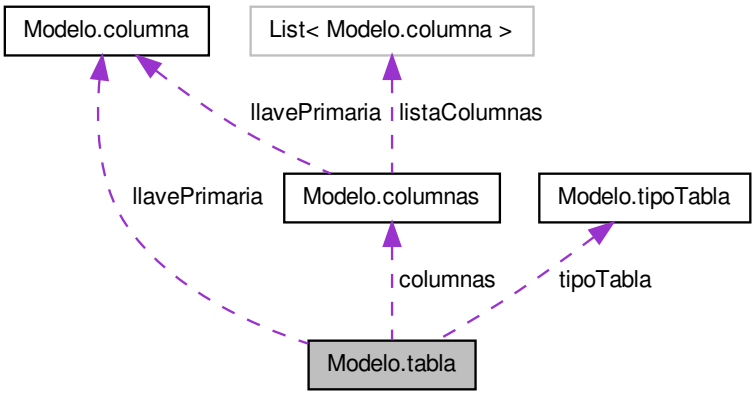
Definición en la línea 21 del archivo procedimientos.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/procedimientos.java

B.26. Referencia de la Clase Modelo.tabla

Diagrama de colaboración para Modelo.tabla:



Métodos públicos

- tabla ()
- tabla (String catalogo, String nombreTabla, columnas listaColumnas)
- void setLlavePrimaria (columna llavePrimaria)
- columna getLlavePrimaria ()
- void setCatalogo (String catalogo)
- String getCatalogo ()
- void setNombreTabla (String nombreTabla)
- String getNombreTabla ()
- void setColumnas (columnas col)
- columnas getColumnas ()
- tipoTabla getTipoTabla ()
- void setTipoTabla (tipoTabla tipoTabla)
- columnas columnasSP ()
- String getColumnaSQL (columna c, boolean SQL)
- String getTablaSQL ()
- boolean contiene (columna c)
- String getLlavePrimariaSQL ()
- String getNombreTablaSQL ()
- String getColumnasSQL (boolean sql)

Atributos públicos

- String nombreTabla
- columnas columnas
- columna llavePrimaria

- tipoTabla tipoTabla
- String catalogo

B.26.1. Descripción detallada

Clase que implementa una tabla de una base de datos

Autor

ivan

Definición en la línea 13 del archivo tabla.java.

B.26.2. Documentación del constructor y destructor

B.26.2.1. Modelo.tabla.tabla ( )

Constructor por defecto de la clase tabla

Definición en la línea 39 del archivo tabla.java.

```
39                      {
40
41      }
```

B.26.2.2. Modelo.tabla.tabla (String *catalogo,* String *nombreTabla,* columnas *listaColumnas*)

Sobrecargar del constructor, establece el nombre del catálogo, nombre de la tabla y la lista de columnas

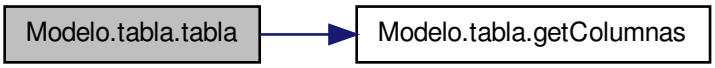
Parámetros

catalogo	Nombre del catálogo
nombreTabla	Nombre de la tabla
listaColumnas	Lista de columnas

Definición en la línea 51 del archivo tabla.java.

```
51                                                                                  {
52          this.setCatalogo(catalogo);
53          this.setNombreTabla(nombreTabla);
54          this.setColumnas(listaColumnas);
55
56          for (int i = 0; i < this.getColumnas().numeroColumnas(); i++) {
57              if (this.getColumnas().getColumna(i).llavePrimaria) {
58                  this.setLlavePrimaria(this.getColumnas().getColumna(i));
59                  break;
60              }
61          }
62
63      }
```

Gráfico de llamadas para esta función:



B.26.3. Documentación de las funciones miembro

B.26.3.1. columnas Modelo.tabla.columnasSP ( )

Método que obtiene una lista de columnas que son candidatas a ser parámetros en un procedimiento almacenado

Devuelve

columnas Lista de columnas

Definición en la línea 161 del archivo tabla.java.

```
161                                 {
162         columnas col = new columnas();
163         for (int i = 0; i < this.columnas.numeroColumnas(); i++) {
164             if (this.columnas.getColumna(i).getCandidatoSP() && !this.
      columnas.getColumna(i).llavePrimaria) {
165                 col.agregarColumna(this.columnas.getColumna(i));
166             }
167         }
168         return col;
169     }
```

Gráfico de llamadas para esta función:

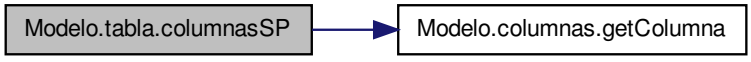
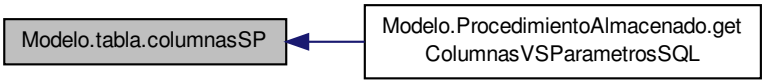


Gráfico de llamadas a esta función:



B.26.3.2. boolean Modelo.tabla.contiene (columna *c*)

Método que verifica si en la lista de columnas contiene la columna proporcionada

Parámetros

c	columna

Devuelve

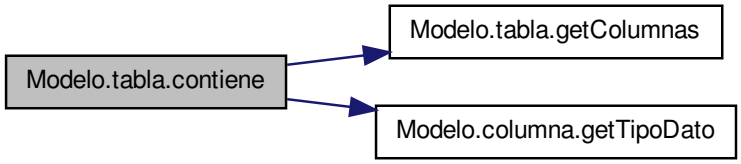
boolean

- true la columna proporcionada está en la lista de columnas
- false la columna proporcionada no está en la lista de columnas

Definición en la línea 212 del archivo tabla.java.

```
212                                     {
213         boolean tmp = false;
214         for (int i = 0; i < this.getColumnas().numeroColumnas(); i++) {
215             if (this.getColumnas().getColumna(i).getNombreColumna().equals(c.getNombreColumna())
216                     && this.getColumnas().getColumna(i).getTipoDato().equals(c.getTipoDato())) {
217                 tmp = true;
218                 break;
219             }
220         }
221         return tmp;
222     }
```

Gráfico de llamadas para esta función:



B.26.3.3. String Modelo.tabla.getCatalogo ( )

Método que obtiene el nombre del catálogo

Devuelve

String Nombre del catálogo

Definición en la línea 97 del archivo tabla.java.

```
97                                  {
98         return this.catalogo;
99     }
```

B.26.3.4. columnas Modelo.tabla.getColumnas ( )

Método que obtiene la lista de columnas

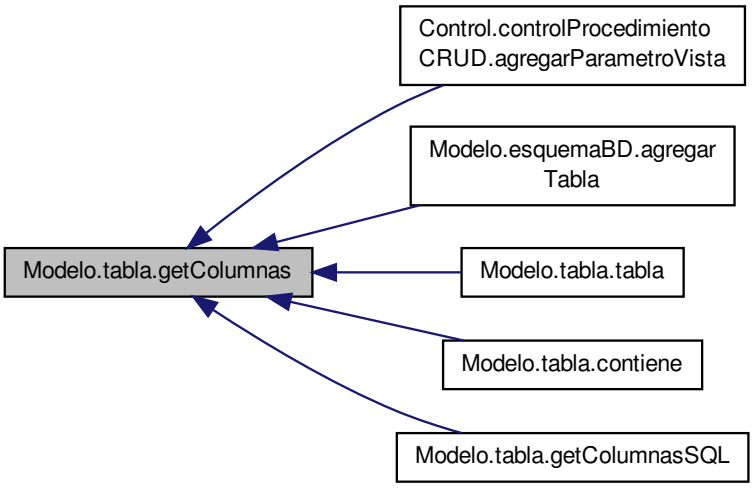
Devuelve

columnas Lista de columnas

Definición en la línea 133 del archivo tabla.java.

```
133                                 {
134         return this.columnas;
135     }
```

Gráfico de llamadas a esta función:



B.26.3.5. String Modelo.tabla.getColumnaSQL (columna *c*, boolean *SQL*)

Método que genera y retorna código SQL asociado a la columna proporcionada, ya sea solamene el nombre de la columna o el nombre completo incluyendo el del catálogo y la tabla.

Parámetros

c	Columna
SQL	boolean ▪ true Indica que incluirá el nombre del catálogo y la tabla ▪ false Indica que solamente será el nombre de la columna

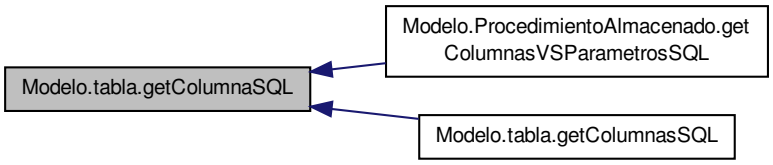
Devuelve

String código SQL asociado

Definición en la línea 184 del archivo tabla.java.

```
184                                                     {
185         if (SQL) {
186             return "`" + this.catalogo + "`" + "." + "`" + this.nombreTabla + "`" + "." + "`" +
      c.getNombreColumna() + "`";
187         } else {
188             return "`" + c.getNombreColumna() + "`";
189         }
190     }
```

Gráfico de llamadas a esta función:



B.26.3.6. String Modelo.tabla.getColumnasSQL (boolean *sql*)

Método que genera y retorna código SQL asociado a la lista de columnas del procedimiento almacenado

Parámetros

sql	boolean ■ true significa que el separador de columnas es la palabra "AND" ■ false significa que el separador de columnas es ","

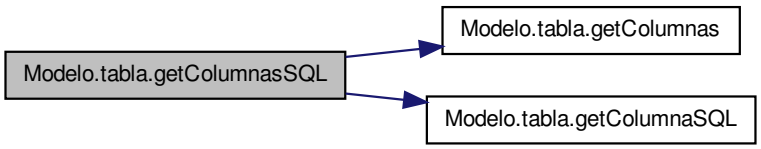
Devuelve

Definición en la línea 254 del archivo tabla.java.

```
254                                                     {
255
256         String SQL = "";
257         if (sql) {
258             for (int i = 0; i < this.getColumnas().numeroColumnas(); i++) {
259                 if (!this.getColumnas().getColumna(i).llavePrimaria && this.
      getColumnas().getColumna(i).candidatoSP) {
260                     SQL = (SQL.equals("")) ? SQL + this.getColumnaSQL(this.getColumnas().getColumna(i), sql
      ) + "\n"
261                             : SQL + "AND " + this.getColumnaSQL(this.
      getColumnas().getColumna(i), sql) + "\n";
262                 }
263             }
```

```
264         } else {
265             for (int i = 0; i < this.getColumnas().numeroColumnas(); i++) {
266                 if (!this.getColumnas().getColumna(i).llavePrimaria && this.
    getColumnas().getColumna(i).candidatoSP) {
267                     SQL = (SQL.equals("")) ? SQL + this.getColumnaSQL(this.getColumnas().getColumna(i), sql
    ) + "\n"
268                             : SQL + ", " + this.getColumnaSQL(this.
    getColumnas().getColumna(i), sql) + "\n";
269                 }
270             }
271         }
272         return SQL;
273     }
```

Gráfico de llamadas para esta función:



B.26.3.7. columna Modelo.tabla.getLlavePrimaria ( )

Méto que obtiene la llave primaria

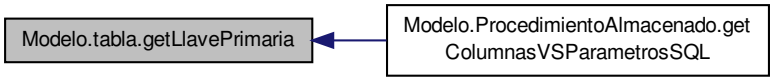
Devuelve

Llave primaria

Definición en la línea 79 del archivo tabla.java.

```
79                                      {
80         return this.llavePrimaria;
81     }
```

Gráfico de llamadas a esta función:



B.26.3.8. String Modelo.tabla.getLlavePrimariaSQL ( )

Método que proporciona el código SQL asociado a la llave primaria

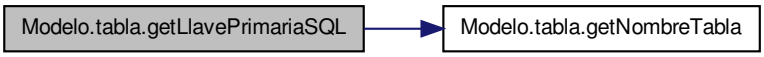
Devuelve

String código SQL asociado

Definición en la línea 229 del archivo tabla.java.

```
229                                     {
230         return "`" + this.getCatalogo() + "`" + "." + "`" + this.getNombreTabla() + "`" + "."
      + "`" + this.llavePrimaria.getNombreColumna() + "`";
231     }
```

Gráfico de llamadas para esta función:



B.26.3.9. String Modelo.tabla.getNombreTabla ( )

Método que obtiene el nombre de la tabla

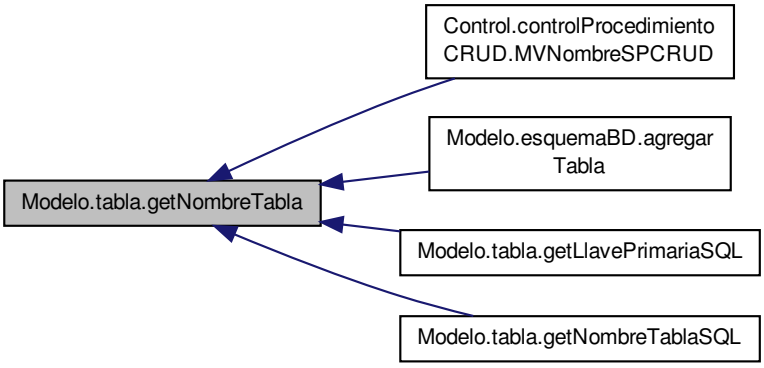
Devuelve

String Nombre de la tabla

Definición en la línea 115 del archivo tabla.java.

```
115                                 {
116         return this.nombreTabla;
117     }
```

Gráfico de llamadas a esta función:



B.26.3.10. String Modelo.tabla.getNombreTablaSQL ( )

Método que genera y retorna código SQL asociado al nombre completo de la tabla, es decir, se incluye el nombre del catálogo.

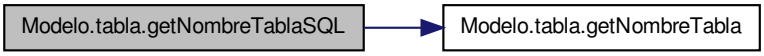
Devuelve

String código SQL asociado

Definición en la línea 239 del archivo tabla.java.

```
239                                 {
240         return "`" + this.getCatalogo() + "`" + "." + "`" + this.getNombreTabla() + "`";
241     }
```

Gráfico de llamadas para esta función:



B.26.3.11. String Modelo.tabla.getTablaSQL ( )

Método que genera y retorna el código SQL asociado al nombre de la tabla

Devuelve

String código SQL asociado

Definición en la línea 197 del archivo tabla.java.

```
197                                 {
198         return "`" + this.catalogo + "`" + "." + "`" + this.nombreTabla + "`";
199     }
```

B.26.3.12. tipoTabla Modelo.tabla.getTipoTabla ( )

Método que obtiene el tipo de tabla

Devuelve

tipoTabla Tipo de tabla

Definición en la línea 142 del archivo tabla.java.

```
142                                 {
143         return this.tipoTabla;
144     }
```

B.26.3.13. void Modelo.tabla.setCatalogo (String *catalogo*)

Método que establece el nombre del catálogo

Parámetros

catalogo	Nombre del catálogo

Definición en la línea 88 del archivo tabla.java.

```
88                                          {
89          this.catalogo = catalogo;
90      }
```

B.26.3.14. void Modelo.tabla.setColumnas (columnas *col*)

Método que establece la lista de columnas

Parámetros

col	Lista de columnas

Definición en la línea 124 del archivo tabla.java.

```
124                                         {
125          this.columnas = col;
126      }
```

B.26.3.15. void Modelo.tabla.setLlavePrimaria (columna *llavePrimaria*)

Método que establece la llave primaria

Parámetros

llavePrimaria	Llave primaria

Definición en la línea 70 del archivo tabla.java.

```
70                                                  {
71          this.llavePrimaria = llavePrimaria;
72      }
```

B.26.3.16. void Modelo.tabla.setNombreTabla (String *nombreTabla*)

Método que establece el nombre de la tabla

Parámetros

nombreTabla	Nombre de la tabla

Definición en la línea 106 del archivo tabla.java.

```
106                                             {
107          this.nombreTabla = nombreTabla;
108      }
```

B.26.3.17. void Modelo.tabla.setTipoTabla (tipoTabla *tipoTabla*)

Método que establece el tipo de tabla

Parámetros

tipoTabla	Tipo de tabla

Definición en la línea 151 del archivo tabla.java.

```
151                                                 {
152         this.tipoTabla = tipoTabla;
153     }
```

B.26.4. Documentación de los datos miembro

B.26.4.1. String Modelo.tabla.catalogo

Nombre del catálogo al que pertenece la tabla

Definición en la línea 34 del archivo tabla.java.

B.26.4.2. columnas Modelo.tabla.columnas

Lista de columnas

Definición en la línea 22 del archivo tabla.java.

B.26.4.3. columna Modelo.tabla.llavePrimaria

Llave primaria de la tabla

Definición en la línea 26 del archivo tabla.java.

B.26.4.4. String Modelo.tabla.nombreTabla

Nombre de la tabla

Definición en la línea 18 del archivo tabla.java.

B.26.4.5. tipoTabla Modelo.tabla.tipoTabla

Tipo de tabla

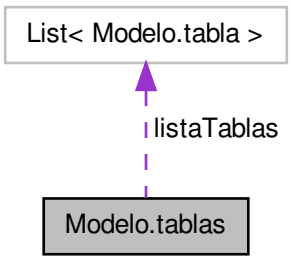
Definición en la línea 30 del archivo tabla.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/tabla.java

B.27. Referencia de la Clase Modelo.tablas

Diagrama de colaboración para Modelo.tablas:



Métodos públicos

- tablas ()
- void setCatalogo (String catalogo)
- String getCatalogo ()
- void setListaTablas (List listaTablas)
- List getListaTablas ()
- boolean agregarTabla (tabla T)
- int numTablas ()
- boolean listaTablasVacia ()
- tabla obtenerTabla (int indice)
- tabla getTabla (String nombreTabla)
- boolean vacia ()

Atributos públicos

- String catalogo

Atributos privados

- List< tabla > listaTablas

B.27.1. Descripción detallada

Clase que implementa una lista de tablas

Autor

ivan

Definición en la línea 16 del archivo tablas.java.

B.27.2. Documentación del constructor y destructor

B.27.2.1. Modelo.tablas.tablas ( )

Constructor por defecto de la clase

Definición en la línea 30 del archivo tablas.java.

```
30                      {
31          this.setListaTablas(new ArrayList());
32      }
```

B.27.3. Documentación de las funciones miembro

B.27.3.1. boolean Modelo.tablas.agregarTabla (tabla *T*)

Método que agrega una tabla a la lista

Parámetros

T	Tabla

Devuelve

boolean

- true la tabla se agregó a la lista
- false la tabla no se agregó a la lista

Definición en la línea 80 del archivo tablas.java.

```
80                                      {
81          this.setCatalogo(T.getCatalogo());
82          return this.listaTablas.add(T);
83      }
```

B.27.3.2. String Modelo.tablas.getCatalogo ( )

Método que obtiene el nombre del catálogo de la lista de tablas

Devuelve

String Nombre del catálogo

Definición en la línea 48 del archivo tablas.java.

```
48                                  {
49          return this.catalogo;
50      }
```

B.27.3.3. List Modelo.tablas.getListaTablas ( )

Método que obtiene la lista de tablas

Devuelve

List Lista de tablas

Definición en la línea 66 del archivo tablas.java.

```
66                                  {
67         return this.listaTablas;
68     }
```

B.27.3.4. tabla Modelo.tablas.getTabla (String *nombreTabla*)

Método que obtiene la tabla asociada al nombre de una tabla

Parámetros

nombreTabla	Nombre de la tabla

Devuelve

tabla Tabla

Definición en la línea 123 del archivo tablas.java.

```
123                                             {
124         tabla tmpTabla = null;
125         for (tabla tmp : listaTablas) {
126             if (tmp.getNombreTabla().equals(nombreTabla)) {
127                 tmpTabla = tmp;
128                 break;
129             }
130         }
131         return tmpTabla;
132     }
```

B.27.3.5. boolean Modelo.tablas.listaTablasVacia ( )

Método que verifica si la lista de tablas está vacía

Devuelve

bolean

- true, la lista está vacía
- false, la lista no está vacía

Definición en la línea 103 del archivo tablas.java.

```
103                                     {
104         return this.listaTablas.isEmpty();
105     }
```

B.27.3.6. int Modelo.tablas.numTablas ( )

Método que obtiene el número de tablas en la lista

Devuelve

int Número de tablas

Definición en la línea 90 del archivo tablas.java.

```
90                              {
91          return this.listaTablas.size();
92      }
```

B.27.3.7. tabla Modelo.tablas.obtenerTabla (int *indice*)

Método que obtiene la tabla asociada al indice proporcionado

Parámetros

indice	Indice

Devuelve

tabla Tabla

Definición en la línea 113 del archivo tablas.java.

```
113                                         {
114          return this.listaTablas.get(indice);
115      }
```

B.27.3.8. void Modelo.tablas.setCatalogo (String *catalogo*)

Método que establece el nombre del catálogo de la lista de tablas

Parámetros

catalogo	Nombre del catálogo

Definición en la línea 39 del archivo tablas.java.

```
39                                          {
40          this.catalogo = catalogo;
41      }
```

B.27.3.9. void Modelo.tablas.setListaTablas (List *listaTablas*)

Método que establece la lista de tablas

Parámetros

listaTablas	Lista de tablas

Definición en la línea 57 del archivo tablas.java.

```
57                                              {
58          this.listaTablas = listaTablas;
59      }
```

B.27.3.10. boolean Modelo.tablas.vacia ( )

Método que verifica si un la lista de tablas está vacía

Devuelve

bolean

- true, la lista está vacía
- false, la lista no está vacía

Definición en la línea 143 del archivo tablas.java.

```
143                             {//borrar esta repetida
144         return this.listaTablas.isEmpty();
145     }
```

B.27.4. Documentación de los datos miembro

B.27.4.1. String Modelo.tablas.catalogo

Nombre del catálogo a que pertenece la lista

Definición en la línea 25 del archivo tablas.java.

B.27.4.2. List<tabla> Modelo.tablas.listaTablas [private]

Lista de tablas

Definición en la línea 21 del archivo tablas.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/tablas.java

B.28. Referencia del enum Modelo.tipoTabla

Métodos públicos

- tipoTabla (String tipoTabla)
- String tipoTabla ()
- String toString ()

Atributos públicos

- TABLE =("tabla")
- VIEW =("vista")
- SYSTEM_TABLE =("tabla de sistema")
- GLOBAL_TEMPORARY =("temporal global")
- LOCAL_TEMPORARY =("temporal local")
- ALIAS =("alias")
- SYNONYM =("sinonima")
- String tipoTabla

B.28.1. Descripción detallada

Enumeración de los tipos de tablas

Autor

ivan

Definición en la línea 13 del archivo tipoTabla.java.

B.28.2. Documentación del constructor y destructor

B.28.2.1. Modelo.tipoTabla.tipoTabla (String *tipoTabla*)

Constructor por defecto de la clase

Parámetros

tipoTabla	Etiqueta del tipo de tabla

Definición en la línea 53 del archivo tipoTabla.java.

```
53                                  {
54          this.tipoTabla = tipoTabla;
55      }
```

B.28.2.2. String Modelo.tipoTabla.tipoTabla ( )

Método que obtiene la etiqueta del tipo de tabla

Devuelve

String Etiqueta del tipo de tabla

Definición en la línea 62 del archivo tipoTabla.java.

```
62                                  {
63          return this.tipoTabla;
64      }
```

B.28.3. Documentación de las funciones miembro

B.28.3.1. String Modelo.tipoTabla.toString ( )

Sobreescritura del método toString

Devuelve

String Etiqueta del tipo de tabla

Definición en la línea 71 del archivo tipoTabla.java.

```
71                                  {
72          return this.tipoTabla();
73      }
```

B.28.4. Documentación de los datos miembro

B.28.4.1. Modelo.tipoTabla.ALIAS =("alias")

Tipo de tabla: Alias

Definición en la línea 38 del archivo tipoTabla.java.

B.28.4.2. Modelo.tipoTabla.GLOBAL_TEMPORARY =("temporal global")

Tipo de tabla: Temporal global

Definición en la línea 30 del archivo tipoTabla.java.

B.28.4.3. Modelo.tipoTabla.LOCAL_TEMPORARY =("temporal local")

Tipo de tabla: Temporal local

Definición en la línea 34 del archivo tipoTabla.java.

B.28.4.4. Modelo.tipoTabla.SYNONYM =("sinonima")

Tipo de tabla: Sinónima

Definición en la línea 42 del archivo tipoTabla.java.

B.28.4.5. Modelo.tipoTabla.SYSTEM_TABLE =("tabla de sistema")

Tipo de tabla: Tabla de sistema

Definición en la línea 26 del archivo tipoTabla.java.

B.28.4.6. Modelo.tipoTabla.TABLE =("tabla")

Tipo de tabla: Tabla

Definición en la línea 18 del archivo tipoTabla.java.

B.28.4.7. String Modelo.tipoTabla.tipoTabla

Etiqueta del tipo de tabla

Definición en la línea 46 del archivo tipoTabla.java.

B.28.4.8. Modelo.tipoTabla.VIEW =("vista")

Tipo de tabla: Vista

Definición en la línea 22 del archivo tipoTabla.java.

La documentación para este enum ha sido generada a partir del siguiente fichero:

- Modelo/tipoTabla.java

B.29. Referencia de la Clase Modelo.Usuario

Diagrama de herencias de Modelo.Usuario

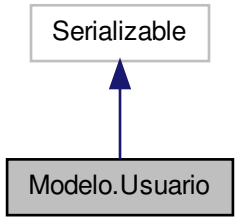
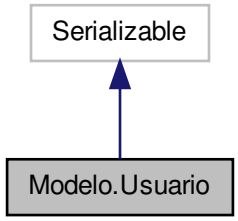


Diagrama de colaboración para Modelo.Usuario:



Métodos públicos

- String getValor ()
- void setValor (String valor)
- void resetDefaultUser ()
- String getNUsuario ()
- void setNUsuario (String nombreUsuario)
- String getNHost ()
- void setNHost (String nombreHost)
- Usuario ()
- Usuario (String valor)
- Usuario (String usuario, String host)
- String generarSQL ()

Atributos privados

- String valor
- String nombreUsuario
- String nombreHost

B.29.1. Descripción detallada

Clase que implementa el usuario del cual se usaran sus privilegios para ejecutar el procedimiento almacenado

Autor

ivan

Definición en la línea 16 del archivo Usuario.java.

B.29.2. Documentación del constructor y destructor

B.29.2.1. Modelo.Usuario.Usuario ( )

Constructor por defecto de la clase

Definición en la línea 99 del archivo Usuario.java.

```
99                      {
100         this.setValor("CURRENT_USER");
101     }
```

B.29.2.2. Modelo.Usuario.Usuario (String *valor*)

Sobrecargar del constructor de la clase, establece el valor del usuario del procedimiento almacenado

Parámetros

valor	Valor del usuario

Definición en la línea 109 del archivo Usuario.java.

```
109                                 {
110         this.setValor(valor);
111     }
```

B.29.2.3. Modelo.Usuario.Usuario (String *usuario,* String *host*)

Sobrecarga del constructor de la clase, establece el nombre del usuario y el nombre del host asociado al procedimiento almacenado

Parámetros

usuario	Nombre del usuario
host	Nombre del host

Definición en la línea 120 del archivo Usuario.java.

```
120                                             {
121         this.setNUsuario(usuario);
122         this.setNHost(host);
123     }
```

B.29.3. Documentación de las funciones miembro

B.29.3.1. String Modelo.Usuario.generarSQL ( )

Método que genera y retorna el código SQL asociado al usuario del cual se usaran sus privilegios para ejecutar el procedimiento almacenado

Devuelve

String Código SQL

Definición en la línea 131 del archivo Usuario.java.

```
131                                 {
132         String SQL = " DEFINER = ";
133         if (this.valor == null) {
134             SQL = SQL + " " + "'" + this.nombreUsuario + "'@'" + this.nombreHost + "'";
135         } else {
136             SQL = SQL + " " + this.valor;
137         }
138         return SQL;
139     }
```

B.29.3.2. String Modelo.Usuario.getNHost ( )

Método que obtiene el nombre del host asociado al usuario de un procedimiento almacenado

Devuelve

String Nombre del host

Definición en la línea 82 del archivo Usuario.java.

```
82                                  {
83         return this.nombreHost;
84     }
```

B.29.3.3. String Modelo.Usuario.getNUsuario ( )

Método que obtiene el nombre del usuario de un procedimiento almacenado

Devuelve

String nombre del usuario

Definición en la línea 63 del archivo Usuario.java.

```
63                                  {
64         return this.nombreUsuario;
65     }
```

B.29.3.4. String Modelo.Usuario.getValor ( )

Método que obtiene el valor del usuario de un procedimiento almacenado

Devuelve

String valor del usuario de un procedimiento almacenado

Definición en la línea 36 del archivo Usuario.java.

```
36                              {
37         return this.valor;
38     }
```

B.29.3.5. void Modelo.Usuario.resetDefaultUser ( )

Método que establece el estado por defecto de la clase

Definición en la línea 52 del archivo Usuario.java.

```
52                                  {
53         this.valor = "CURRENT_USER";
54         this.nombreUsuario = null;
55         this.nombreHost = null;
56     }
```

B.29.3.6. void Modelo.Usuario.setNHost (String *nombreHost*)

Método que establece el nombre del host asociado al usuario de un procedimiento almacenado

Parámetros

nombreHost	nombre del host

Definición en la línea 92 del archivo Usuario.java.

```
92                                          {
93         this.nombreHost = nombreHost;
94     }
```

B.29.3.7. void Modelo.Usuario.setNUsuario (String *nombreUsuario*)

Método que establece el nombre del usuario de un procedimiento almacenado

Parámetros

nombreUsuario	nombre del usuario

Definición en la línea 72 del archivo Usuario.java.

```
72                                              {
73         this.nombreUsuario = nombreUsuario;
74     }
```

B.29.3.8. void Modelo.Usuario.setValor (String *valor*)

Método que establece el valor del usuario de un procedimiento almacenado

Parámetros

valor	valor de usuario

Definición en la línea 45 del archivo Usuario.java.

```
45                                      {
46         this.valor = valor;
47     }
```

B.29.4. Documentación de los datos miembro

B.29.4.1. String Modelo.Usuario.nombreHost [private]

Nombre del host asociado al usuario de un procedimiento almacenado

Definición en la línea 29 del archivo Usuario.java.

B.29.4.2. String Modelo.Usuario.nombreUsuario [private]

Nombre del usuario de un procedimiento almacenado

Definición en la línea 25 del archivo Usuario.java.

B.29.4.3. String Modelo.Usuario.valor [private]

Valor del usuario de un procedimiento almacenado

Definición en la línea 21 del archivo Usuario.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Modelo/Usuario.java

B.30. Referencia de la Clase Vistas.VentanaConexion

Diagrama de herencias de Vistas.VentanaConexion

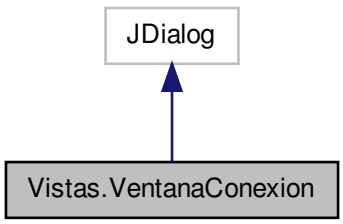
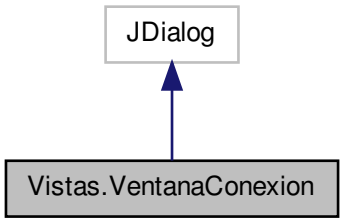


Diagrama de colaboración para Vistas.VentanaConexion:



Métodos públicos

- VentanaConexion (java.awt.Frame parent, boolean modal)

Métodos públicos estáticos

- static void main (String args[])

Atributos públicos

- javax.swing.JButton jButton1
- javax.swing.JButton jButton2
- javax.swing.JButton jButton3
- javax.swing.JPasswordField jPasswordField1
- javax.swing.JTextField jTextField1
- javax.swing.JTextField jTextField2
- javax.swing.JTextField jTextField3
- javax.swing.JTextField jTextField4
- javax.swing.JTextField jTextField5

Métodos privados

- void initComponents ()

Atributos privados

- javax.swing.JLabel jLabel10
- javax.swing.JLabel jLabel4
- javax.swing.JLabel jLabel5
- javax.swing.JLabel jLabel6
- javax.swing.JLabel jLabel7
- javax.swing.JLabel jLabel8

- javax.swing.JLabel jLabel9
- javax.swing.JPanel jPanel1
- javax.swing.JPanel jPanel2

B.30.1. Descripción detallada

Autor

ivan

Definición en la línea 13 del archivo VentanaConexion.java.

B.30.2. Documentación del constructor y destructor

B.30.2.1. Vistas.VentanaConexion.VentanaConexion (java.awt.Frame *parent,* boolean *modal*)

Creates new form tmp

Definición en la línea 18 del archivo VentanaConexion.java.

```
18                                                                  {
19         super(parent, modal);
20         initComponents();
21     }
```

Gráfico de llamadas para esta función:

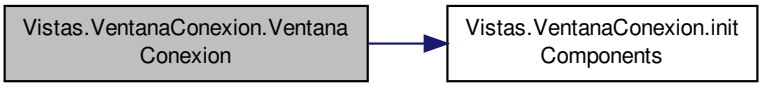
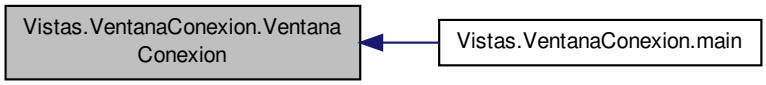


Gráfico de llamadas a esta función:



B.30.3. Documentación de las funciones miembro

B.30.3.1. void Vistas.VentanaConexion.initComponents ( ) [private]

This method is called from within the constructor to initialize the form. WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

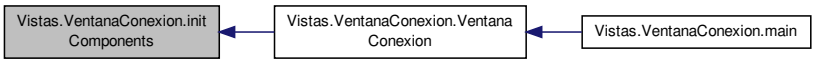
Definición en la línea 30 del archivo VentanaConexion.java.

```
30                                {
31
32          jLabel5 = new javax.swing.JLabel();
33          jTextField2 = new javax.swing.JTextField();
34          jPanel1 = new javax.swing.JPanel();
35          jLabel6 = new javax.swing.JLabel();
36          jTextField3 = new javax.swing.JTextField();
37          jLabel7 = new javax.swing.JLabel();
38          jTextField4 = new javax.swing.JTextField();
39          jPanel2 = new javax.swing.JPanel();
40          jLabel8 = new javax.swing.JLabel();
41          jTextField5 = new javax.swing.JTextField();
42          jLabel9 = new javax.swing.JLabel();
43          jPasswordField1 = new javax.swing.JPasswordField();
44          jTextField1 = new javax.swing.JTextField();
45          jLabel4 = new javax.swing.JLabel();
46          jLabel10 = new javax.swing.JLabel();
47          jButton2 = new javax.swing.JButton();
48          jButton3 = new javax.swing.JButton();
49          jButton1 = new javax.swing.JButton();
50
51          setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
52
53          jLabel5.setText("*Base de datos:");
54
55          jTextField2.setText("Nombre_BD");
56          jTextField2.setToolTipText("");
57
58          jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("servidor"));
59
60          jLabel6.setText("host name/ip:");
61
62          jTextField3.setText("localhost");
63          jTextField3.setToolTipText("");
64
65          jLabel7.setText("Puerto:");
66
67          jTextField4.setText("3306");
68          jTextField4.setToolTipText("");
69
70          javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(
    jPanel1);
71          jPanel1.setLayout(jPanel1Layout);
72          jPanel1Layout.setHorizontalGroup(
73              jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
74              .addGroup(jPanel1Layout.createSequentialGroup()
75                  .addContainerGap()
76                  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
77                      .addGroup(jPanel1Layout.createSequentialGroup()
78                          .addComponent(jLabel6)
79                          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
80                          .addComponent(jTextField3))
81                      .addGroup(jPanel1Layout.createSequentialGroup()
82                          .addComponent(jLabel7)
83                          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
84                          .addComponent(jTextField4)))
85                  .addContainerGap())
86          );
87          jPanel1Layout.setVerticalGroup(
88              jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
89              .addGroup(jPanel1Layout.createSequentialGroup()
90                  .addContainerGap()
91                  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
92                      .addComponent(jLabel6)
93                      .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
94                  .addGap(18, 18, 18)
95                  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
96                      .addComponent(jLabel7)
97                      .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
98                  .addContainerGap(16, Short.MAX_VALUE))
```

```
99          );
100
101         jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("usuario"));
102
103         jLabel8.setText("*Usuario:");
104
105         jTextField5.setText("Usuario");
106         jTextField5.setToolTipText("");
107
108         jLabel9.setText("*Password:");
109
110         jPasswordField1.setText("12345678");
111         jPasswordField1.setToolTipText("");
112
113         javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(
    jPanel2);
114         jPanel2.setLayout(jPanel2Layout);
115         jPanel2Layout.setHorizontalGroup(
116             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
117             .addGroup(jPanel2Layout.createSequentialGroup()
118                 .addContainerGap()
119                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
120                     .addGroup(jPanel2Layout.createSequentialGroup()
121                         .addComponent(jLabel8)
122                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
123                         .addComponent(jTextField5))
124                     .addGroup(jPanel2Layout.createSequentialGroup()
125                         .addComponent(jLabel9)
126                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
127                         .addComponent(jPasswordField1, javax.swing.GroupLayout.DEFAULT_SIZE,
     349, Short.MAX_VALUE)))
128                 .addContainerGap())
129         );
130         jPanel2Layout.setVerticalGroup(
131             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
132             .addGroup(jPanel2Layout.createSequentialGroup()
133                 .addContainerGap()
134                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
135                     .addComponent(jLabel8)
136                     .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
137                 .addGap(12, 12, 12)
138                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
139                     .addComponent(jLabel9)
140                     .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
141                 .addContainerGap(22, Short.MAX_VALUE))
142         );
143
144         jTextField1.setText("jdbc:");
145         jTextField1.setToolTipText("");
146
147         jLabel4.setText("url:");
148
149         jLabel10.setText("* Campos obligatorios");
150
151         jButton2.setText("Probar conexión");
152         jButton2.setToolTipText("");
153
154         jButton3.setText("Cancelar");
155         jButton3.setToolTipText("");
156
157         jButton1.setText("OK");
158         jButton1.setToolTipText("");
159
160         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
161         getContentPane().setLayout(layout);
162         layout.setHorizontalGroup(
163             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
164             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
165                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
166                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
167                     .addComponent(jLabel10)
168                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
169                         .addGroup(layout.createSequentialGroup()
170                             .addComponent(jButton2)
171                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
172                             .addComponent(jButton3)
173                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
174                             .addComponent(jButton1))
175                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
```

```
    false)
176                             .addGroup(layout.createSequentialGroup()
177                                 .addComponent(jLabel5)
178                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
179                                 .addComponent(jTextField2, javax.swing.GroupLayout.
    PREFERRED_SIZE, 338, javax.swing.GroupLayout.PREFERRED_SIZE))
180                             .addGroup(layout.createSequentialGroup()
181                                 .addComponent(jLabel4)
182                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
183                                 .addComponent(jTextField1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 404, javax.swing.GroupLayout.PREFERRED_SIZE))
184                             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
185                             .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))))
186                 .addContainerGap())
187         );
188         layout.setVerticalGroup(
189             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190             .addGroup(layout.createSequentialGroup()
191                 .addContainerGap()
192                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
193                     .addComponent(jLabel5)
194                     .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
195                 .addGap(18, 18, 18)
196                 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
197                 .addGap(18, 18, 18)
198                 .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
199                 .addGap(18, 18, 18)
200                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
201                     .addComponent(jLabel4)
202                     .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
203                 .addGap(18, 18, 18)
204                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
205                     .addComponent(jButton1)
206                     .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing
    .GroupLayout.PREFERRED_SIZE)
207                     .addComponent(jButton2))
208                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
209                 .addComponent(jLabel10)
210                 .addContainerGap())
211         );
212 
213         pack();
214     }// </editor-fold>//GEN-END:initComponents
```

Gráfico de llamadas a esta función:



B.30.3.2. static void Vistas.VentanaConexion.main (String *args[]*) [static]

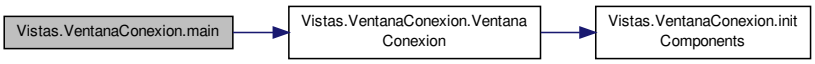
Parámetros

args	the command line arguments

Definición en la línea 219 del archivo VentanaConexion.java.

```
219                                         {
220         /* Set the Nimbus look and feel */
221         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
222         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
223          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
224          */
225         try {
226             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
    getInstalledLookAndFeels()) {
227                 if ("Nimbus".equals(info.getName())) {
228                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
229                     break;
230                 }
231             }
232         } catch (ClassNotFoundException ex) {
233             java.util.logging.Logger.getLogger(VentanaConexion.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
234         } catch (InstantiationException ex) {
235             java.util.logging.Logger.getLogger(VentanaConexion.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
236         } catch (IllegalAccessException ex) {
237             java.util.logging.Logger.getLogger(VentanaConexion.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
238         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
239             java.util.logging.Logger.getLogger(VentanaConexion.class.getName()).log(
    java.util.logging.Level.SEVERE, null, ex);
240         }
241         //</editor-fold>
242 
243         /* Create and display the dialog */
244         java.awt.EventQueue.invokeLater(new Runnable() {
245             public void run() {
246                 VentanaConexion dialog = new VentanaConexion(new javax.swing.
    JFrame(), true);
247                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
248                     @Override
249                     public void windowClosing(java.awt.event.WindowEvent e) {
250                         System.exit(0);
251                     }
252                 });
253                 dialog.setVisible(true);
254             }
255         });
256     }
```

Gráfico de llamadas para esta función:



B.30.4. Documentación de los datos miembro

B.30.4.1. javax.swing.JButton Vistas.VentanaConexion.jButton1

Definición en la línea 259 del archivo VentanaConexion.java.

B.30.4.2. javax.swing.JButton Vistas.VentanaConexion.jButton2

Definición en la línea 260 del archivo VentanaConexion.java.

B.30.4.3. javax.swing.JButton Vistas.VentanaConexion.jButton3

Definición en la línea 261 del archivo VentanaConexion.java.

B.30.4.4. javax.swing.JLabel Vistas.VentanaConexion.jLabel10 [private]

Definición en la línea 262 del archivo VentanaConexion.java.

B.30.4.5. javax.swing.JLabel Vistas.VentanaConexion.jLabel4 [private]

Definición en la línea 263 del archivo VentanaConexion.java.

B.30.4.6. javax.swing.JLabel Vistas.VentanaConexion.jLabel5 [private]

Definición en la línea 264 del archivo VentanaConexion.java.

B.30.4.7. javax.swing.JLabel Vistas.VentanaConexion.jLabel6 [private]

Definición en la línea 265 del archivo VentanaConexion.java.

B.30.4.8. javax.swing.JLabel Vistas.VentanaConexion.jLabel7 [private]

Definición en la línea 266 del archivo VentanaConexion.java.

B.30.4.9. javax.swing.JLabel Vistas.VentanaConexion.jLabel8 [private]

Definición en la línea 267 del archivo VentanaConexion.java.

B.30.4.10. javax.swing.JLabel Vistas.VentanaConexion.jLabel9 [private]

Definición en la línea 268 del archivo VentanaConexion.java.

B.30.4.11. javax.swing.JPanel Vistas.VentanaConexion.jPanel1 [private]

Definición en la línea 269 del archivo VentanaConexion.java.

B.30.4.12. javax.swing.JPanel Vistas.VentanaConexion.jPanel2 [private]

Definición en la línea 270 del archivo VentanaConexion.java.

B.30.4.13. javax.swing.JPasswordField Vistas.VentanaConexion.jPasswordField1

Definición en la línea 271 del archivo VentanaConexion.java.

B.30.4.14. javax.swing.JTextField Vistas.VentanaConexion.jTextField1

Definición en la línea 272 del archivo VentanaConexion.java.

B.30.4.15. javax.swing.JTextField Vistas.VentanaConexion.jTextField2

Definición en la línea 273 del archivo VentanaConexion.java.

B.30.4.16. javax.swing.JTextField Vistas.VentanaConexion.jTextField3

Definición en la línea 274 del archivo VentanaConexion.java.

B.30.4.17. javax.swing.JTextField Vistas.VentanaConexion.jTextField4

Definición en la línea 275 del archivo VentanaConexion.java.

B.30.4.18. javax.swing.JTextField Vistas.VentanaConexion.jTextField5

Definición en la línea 276 del archivo VentanaConexion.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vistas/VentanaConexion.java

Apéndice C

Documentación de archivos

C.1. Referencia del Archivo Control/controlCodigoAnidadoGenerado.java

Clases

- class Control.controlCodigoAnidadoGenerado

Paquetes

- package Control

C.2. Referencia del Archivo Control/controlCodigoProcedimientoGenerado.java

Clases

- class Control.controlCodigoProcedimientoGenerado

Paquetes

- package Control

C.3. Referencia del Archivo Control/controlConexion.java

Clases

- class Control.controlConexion

Paquetes

- package Control

C.4. Referencia del Archivo Control/controlNuevoProcedimiento.java

Clases

- class Control.controlNuevoProcedimiento

Paquetes

- package Control

C.5. Referencia del Archivo Control/controlProcedimiento.java

Clases

- class Control.controlProcedimiento

Paquetes

- package Control

C.6. Referencia del Archivo Control/controlProcedimientoCRUD.java

Clases

- class Control.controlProcedimientoCRUD

Paquetes

- package Control

C.7. Referencia del Archivo Control/MiFocusTraversalPolicy.java

Clases

- class Control.MiFocusTraversalPolicy

Paquetes

- package Control

C.8. Referencia del Archivo Modelo/Caracteristica.java

Clases

- class Modelo.Caracteristica

Paquetes

- package Modelo

C.9. Referencia del Archivo Modelo/Caracteristicas.java

Clases

- class Modelo.Caracteristicas

Paquetes

- package Modelo

C.10. Referencia del Archivo Modelo/columna.java

Clases

- class Modelo.columna

Paquetes

- package Modelo

C.11. Referencia del Archivo Modelo/columnas.java

Clases

- class Modelo.columnas

Paquetes

- package Modelo

C.12. Referencia del Archivo Modelo/ConexionBD.java

Clases

- class Modelo.ConexionBD

Paquetes

- package Modelo

C.13. Referencia del Archivo Modelo/conexionManejadorBD.java

Clases

- class Modelo.conexionManejadorBD

Paquetes

- package Modelo

C.14. Referencia del Archivo Modelo/esquemaBD.java

Clases

- class Modelo.esquemaBD

Paquetes

- package Modelo

C.15. Referencia del Archivo Modelo/funcionBasicaBD.java

Clases

- enum Modelo.funcionBasicaBD

Paquetes

- package Modelo

C.16. Referencia del Archivo Modelo/Parametro.java

Clases

- class Modelo.Parametro

Paquetes

- package Modelo

C.17. Referencia del Archivo Modelo/Parametros.java

Clases

- class Modelo.Parametros

Paquetes

- package Modelo

C.18. Referencia del Archivo Modelo/ProcedimientoAlmacenado.java

Clases

- class Modelo.ProcedimientoAlmacenado

Paquetes

- package Modelo

C.19. Referencia del Archivo Modelo/procedimientos.java

Clases

- class Modelo.procedimientos

Paquetes

- package Modelo

C.20. Referencia del Archivo Modelo/tabla.java

Clases

- class Modelo.tabla

Paquetes

- package Modelo

C.21. Referencia del Archivo Modelo/tablas.java

Clases

- class Modelo.tablas

Paquetes

- package Modelo

C.22. Referencia del Archivo Modelo/tipoTabla.java

Clases

- enum Modelo.tipoTabla

Paquetes

- package Modelo

C.23. Referencia del Archivo Modelo/Usuario.java

Clases

- class Modelo.Usuario

Paquetes

- package Modelo

C.24. Referencia del Archivo Programa/GeneradorProcedimientosAlmacenados.java

Clases

- class Programa.GeneradorProcedimientosAlmacenados

Paquetes

- package Programa

C.25. Referencia del Archivo Vistas/EditorCodigoAnidadoGenerado.java

Clases

- class Vistas.EditorCodigoAnidadoGenerado

Paquetes

- package Vistas

C.26. Referencia del Archivo Vistas/EditorCodigoGenerado.java

Clases

- class Vistas.EditorCodigoGenerado

Paquetes

- package Vistas

C.27. Referencia del Archivo Vistas/EditorProcedimiento.java

Clases

- class Vistas.EditorProcedimiento

Paquetes

- package Vistas

C.28. Referencia del Archivo Vistas/EditorProcedimientoCRUD.java

Clases

- class Vistas.EditorProcedimientoCRUD

Paquetes

- package Vistas

C.29. Referencia del Archivo Vistas/nuevoProcedimiento.java

Clases

- class Vistas.nuevoProcedimiento

Paquetes

- package Vistas

C.30. Referencia del Archivo Vistas/VentanaConexion.java

Clases

- class Vistas.VentanaConexion

Paquetes

- package Vistas

Bibliografía

[1] Wikipedia. (2014, Julio) Anidamiento (informática) — wikipedia, la enciclopedia libre. [en línea]. Disponible en: http://es.wikipedia.org/wiki/Anidamiento_(inform%C3%A1tica) III

[2] A. Silberschatz, H. Korth, and S. Sudarshan, *Database System Concepts*, ser. McGraw-Hill Series in Computer Science. McGraw-Hill, 1999. 13

[3] S. Sumathi and S. Esakkirajan, *Fundamentals of Relational Database Management Systems*, ser. Studies in Computational Intelligence. Springer, 2007. 13

[4] C. Date and S. Faudón, *Introducción a los sistemas de bases de datos*. Pearson Educación, 2001. 1, 13, 14

[5] N. H. Loli. (2013, Septiembre) generador-procedimientos-almacenados-mysql. [en línea]. Disponible en: https://code.google.com/p/generador-procedimientos-almacenados-mysql/ 15

[6] MySQL. (2014, Enero) MySQL Workbench. [en línea]. Disponible en: http://www.mysql.com/products/workbench/ 16

[7] phpMyAdmin contributors. (2014, Junio) phpMyAdmin. [en línea]. Disponible en: http://www.phpmyadmin.net/home_page/downloads.php 16

[8] MySQL. (2014, mayo) 13.1 Data Definition Statements. [en línea]. Disponible en: http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-data-definition.html 19

[9] ——. (2014, mayo) 9.2 Schema Object Names. [en línea]. Disponible en: http://dev.mysql.com/doc/refman/5.7/en/identifiers.html 20

[10] Wikipedia. (2014, mayo) Unicode — wikipedia, the free encyclopedia. [en línea]. Disponible en: http://en.wikipedia.org/w/index.php?title=Unicode&oldid=617496029 20

[11] J. Zukowski, *The Definitive Guide to Java Swing*, ser. Books for professionals by professionals. Apress, 2005. 25

[12] A. Sierra, *Programador Java 2 certificado: curso práctico*. Alfaomega, 2007. 29

[13] MySQL. (2014, mayo) Mysql connector/j. [en línea]. Disponible en: http://dev.mysql.com/downloads/connector/j/5.1.html 29

[14] oracle. (2014, junio) Interface databasemetadata. [en línea]. Disponible en: http://docs.oracle.com/javase/7/docs/api/java/sql/DatabaseMetaData.html 31

Índice alfabético