Documentación Del Sistema.

Contenido

1. Propuesta del proyecto

1.1 Objetivo.

Diseñar e implementar un sistema que genere la redacción de ejercicios de SQL con sus respectivas respuestas a partir de una base de datos indicada, para ayudar a un docente en la elaboración y calificación de tareas y exámenes.

1.2 Justificación

En cualquier curso impartido por un docente o incluidos los cursos en línea se tiene que generar ejercicios y exámenes para poder evaluar, lo cual consume tiempo para su elaboración y tiene cierta dificultad ya que además de tener que generar los ejercicios de tal forma que un alumno entienda su enunciado, posteriormente se tiene que resolver para calificar las respuestas de los alumnos.

1.3 Problemática a resolver

El sistema propuesto será de gran ayuda en la generación de problemas para poder evaluar alumnos de cursos de bases de datos o relacionados. También servirá de apoyo al calificar los mismos ya que dará el código correcto del ejercicio generado. Los resultados del sistema proporcionarán un gran ahorro de tiempo a un docente.
En particular al generar ejercicios a partir de una base de datos indicada. Los docentes que imparten esta materia tienen la necesidad de buscar ejemplos y problemas para que sus alumnos practiquen y sean evaluados. Al dar como ejemplo una base de datos el docente tiene que estudiarla para dominar los problemas que pueden solicitar a los alumnos.

1.4 Recursos disponibles

Descripción de los recursos a utilizar.
Características de la computadora empleada:
Laptop: 64 bits, procesador Pentium Dual-Core y 3GB de memoria RAM, Sistema operativo Ubuntu 11 64 bits.
Software: PostgreSQL *open source*, Java *free software,* Apache Tomcat *free software.*
El software necesario es *open source*, se encuentra disponible en internet por lo que no se requieren licencias.

2 Diagnostico y visión

2.1 Modelo de casos de uso

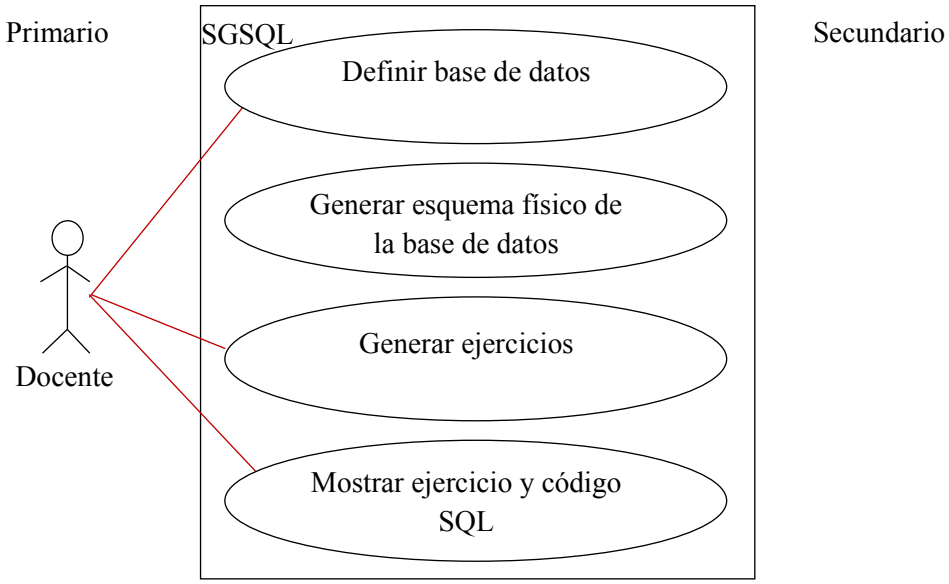


Diagrama de casos de uso

Módulo definir bases de datos: El docente indica al sistema cuál será la base de datos a la cual se deberá conectarse. Conexión a la base de datos.

Módulo generar esquema físico de la base de datos: El sistema se conecta a la base de datos indicada por el docente. Analiza la base de datos e identifica las relaciones, tablas, campos, tipos de datos, llaves y registros.

Módulo generar ejercicios: El sistema recibe parámetros para generar el código SQL tales como: Tipos de acción (SELECT, INSERT, DELETE, UPDATE, JOIN, subconsultas), tablas, número de ejercicios en base a estos parámetros el sistema genera el código SQL y los ejercicios.

Módulo mostrar ejercicio y código SQL: El sistema mostrará los ejercicios con su respectivo código SQL.

2.2 Descripción (alto nivel o extendidas) de todos los Casos de uso.

CASO DE USO	Definir base de datos
ACTORES	Docente.
PRE-CONDICIONES	El docente debe tener una base desarrollada.
EVENTOS	Direccionar la base de datos al sistema.
POST-CONDICIONES	Se tiene definida la base de datos para obtener los ejercicios.
CAMINOS ALTERNATIVOS	Ninguno.

CASO DE USO	Generar esquema físico de la base de datos
ACTORES	Sistema.
PRE-CONDICIONES	Tener direccionada la base de datos.
EVENTOS	Obtener el esquema físico de la base de datos indicada.
POST-CONDICIONES	Esquema físico de la base de datos en un archivo plano.
CAMINOS ALTERNATIVOS	Ninguno.

CASO DE USO	Generar ejercicios
ACTORES	Sistema.
PRE-CONDICIONES	Tener el esquema físico de la base de datos.
EVENTOS	Generar código SQL y ejercicios en lenguaje natural.
POST-CONDICIONES	Ejercicios elaborados con apoyo de plantillas y el codigo SQL.
CAMINOS ALTERNATIVOS	Ninguno

CASO DE USO	Mostrar ejercicio y código SQL
ACTORES	Sistema
PRE-CONDICIONES	Ejercicio y código SQL.
EVENTOS	Mostrar el ejercicio con su respectiva respuesta.
POST-CONDICIONES	Ejercicio y respuesta (codigo SQL) mostrado en pantalla y guardado en archivo, en una ruta, carpeta determinada.
CAMINOS ALTERNATIVOS	Solo guardar el ejercicio en archivo sin mostrarlo en pantalla.

3. Especificación de requerimientos del software

3.1 Matriz de rastreabilidad de requerimientos

	Reducir tiempo de captura	Reducir tiempo para generar ejercicios	Reducir tiempo para generar código SQL	Reducir tiempo para calificar ejercicios
El sistema debe de ser personalizado				✅
Almacenamiento únicamente de la información necesaria	✅	✅	✅	
Encapsulamiento de información	✅	✅	✅	

3.2 Objetivos específicos.

- Diseñar e implementar el módulo para analizar las relaciones y la estructura en una base de datos indicada.
- Diseñar e implementar el módulo para generar el esquema físico de la base de datos.
- Analizar e implementar módulos del sistema para generar código SQL a partir de la base de datos.
- Diseñar e implementar el módulo para generar ejercicios a partir de un código SQL.

3.3 Estimación inicial de costo y tiempo

El sistema será desarrollado por una persona.
Se utilizara software libre con licencias GPL, GNU y licencias gratuitas.
El tiempo estimado para el desarrollo estimado del sistema, será de siete meses utilizando los siguientes periodos de tiempo para su realización:
Inicio del proyecto: Enero del 2012.
Fin del proyecto: Julio del 2012.

3.4 Descripción de las fases del modelo de desarrollo a utilizar

Modelo de desarrollo a utilizar: Modelo RUP
El RUP es un conjunto de metodologías adaptables al contexto y necesidades de cada organización, por lo cual, las fases presentadas en este documento, podrían variar en la aplicación del modelo en otro contexto.

Análisis y Diseño:

- ∴ Identifica las entidades externas o actores con las que se trata.
- ∴ Identifica los casos de uso.
- ∴ Esquema Conceptual.
- ∴ Definición de métricas.
- ∴ Análisis de fuentes de datos.
- ∴ Identificación de riesgos.
- ∴ Elaboración de diagramas.

Desarrollo e implementación:

- ∴ Desarrollo de la aplicación.
- ∴ Creación y modelado de la base de datos.
- ∴ Creación de interfaces graficas.

Fase de aceptación:

- ∴ Validación de funcionalidad.
- ∴ Validación de diseño.
- ∴ Revisión y aprobación de los entregables.
- ∴ Pruebas y modificaciones finales.

Finalización y liberación del proyecto

- ∴ Entrega de documentación y entregables impresos
- ∴ Versión final del sistema.

4. Planeación y calendarización
   Presentación de los riesgos más comunes.

   4.1 Tabla de riesgos con descripción, probabilidad, impacto y estrategia

Riesgo	**Descripción**
Complejidad	Sistema con módulos o procesos muy complejos o robustos que requieran más tiempo.
Seguridad	Perdida de información por fallas tecnológicos/humanos (Fallas de hardware/software, modificación, alteración).

Análisis de riesgos. Probabilidad de impacto.

Riesgo	**Probabilidad de ocurrencia**	**Impacto**
Complejidad	Moderado	Serio
Seguridad	Moderado	Tolerable

Gestión de riesgos. Estrategia

Riesgo	**Estrategia**
Complejidad	Contemplar posibles retrasos ante la complejidad de los módulos.
Seguridad	Implementar seguridad, estrategias preventivas ante perdida de datos, realizar respaldos continuos a todo el sistema y las bases de datos.

5. Estimación de costos
   5.1 Listado de puntos de función

   De Usuarios
   1. Usuario
   2. Password
   3. Activo/Inactivo

   De Catalogo de la base de datos
   4. idCatalogo
   5. Nombre Base de datos
   6. Fecha de Ingreso
   7. Usuario que la ingreso

   De las plantillas
   8. idPlantilla
   9. Plantillas enunciado
   10. Plantillas Resultado

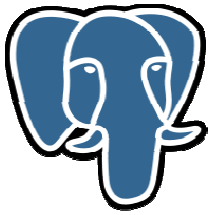
Del ejercicio

11. idEjercicio
12. Fecha de creación
13. Enunciado final
14. Codigo SQL
15. Base de datos de donde se obtuvo
16. Plantilla correspondiente

Interfaces

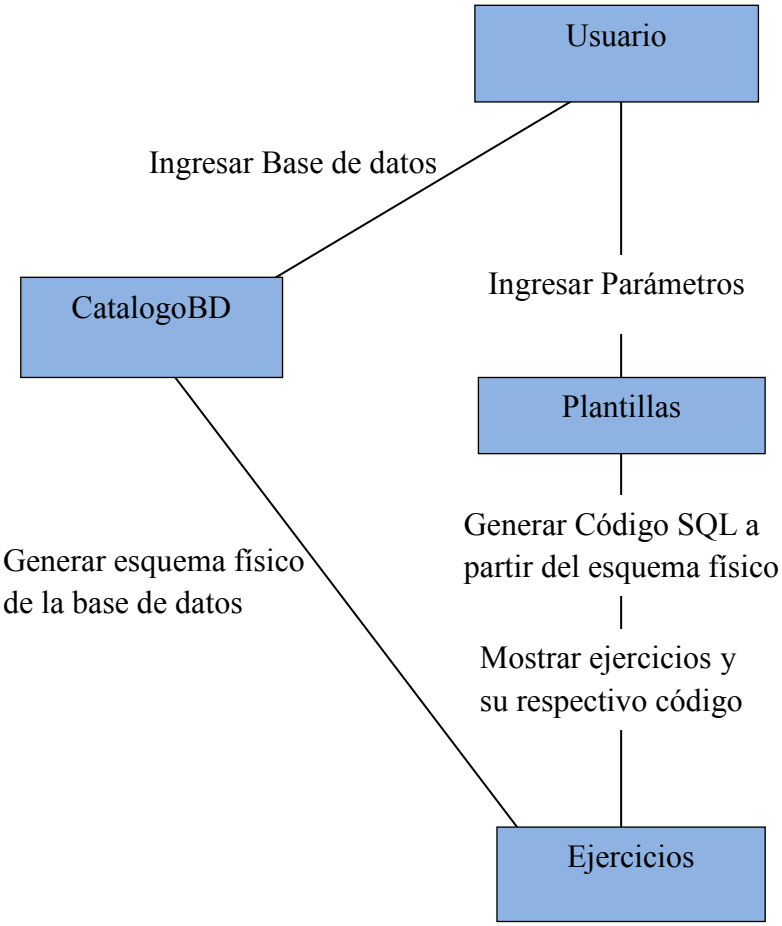
17. Bienvenida
18. Autentificación
19. Conexión
20. Ingresar base de datos
21. Ingresar parámetros
22. Ejercicios

5.2 Calculo de TDE y costo

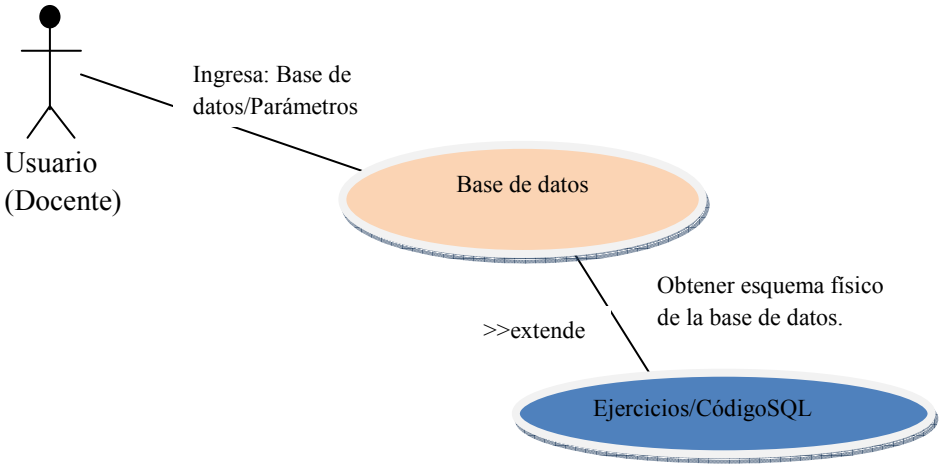
Software				
Item	Descripción	Cantidad	Precio Unitario	Total
1	Gestor de base de datos **PostgreSQL** 9.1 sin soporte 	1	$0.00	$0.00
2	Servidor WEB **Apache Tomcat** 	1	$0.00	$0.00
3	**Java** Lenguaje de programación. 	1	$0.00	$0.00

6. Metodología de Jacobson

6.1 Diagrama Contextual

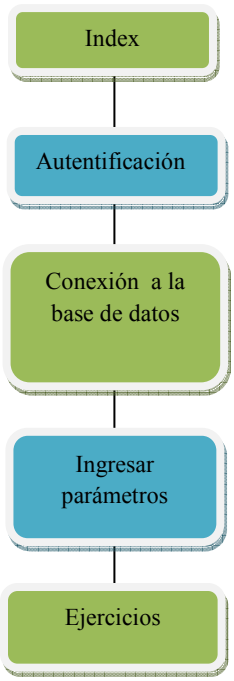


6.2 DCU con relaciones usa y extiende

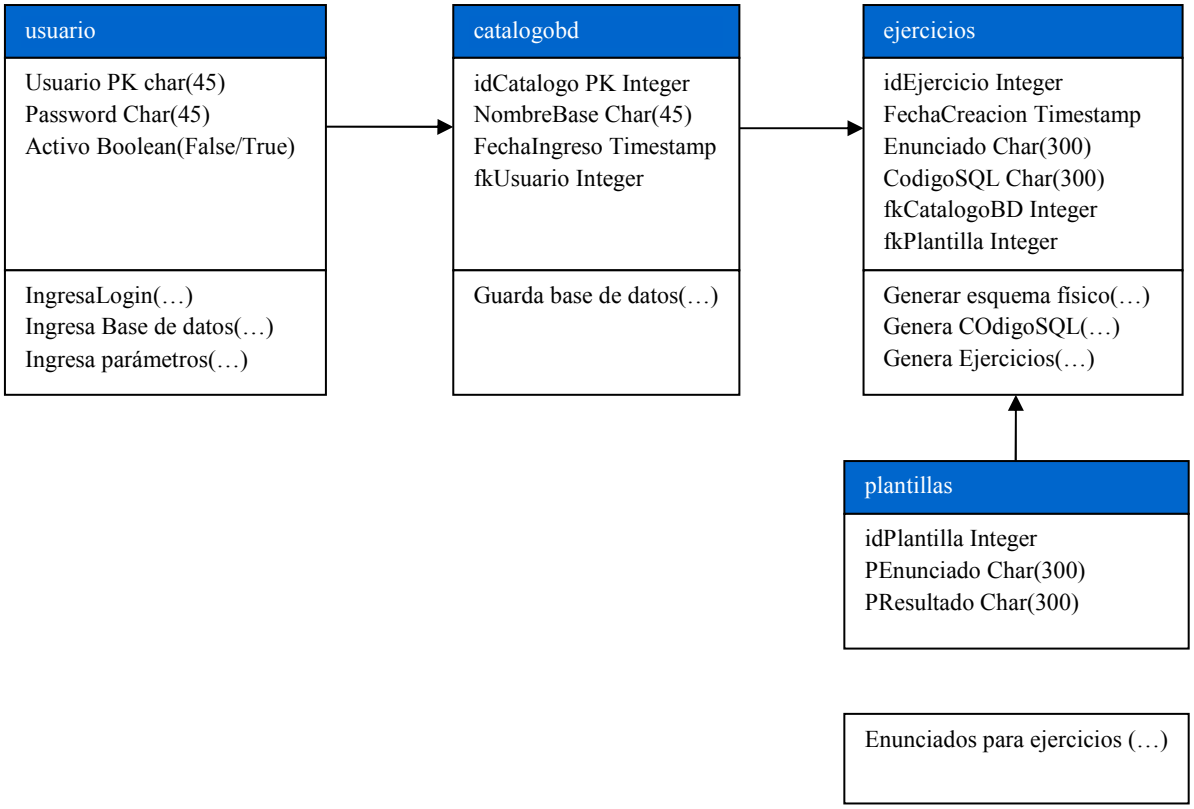


6.3 Interfaces y firmas de operación

6.4 Diagrama de navegación



6.5 Diagrama general de clases



6.6 Diagrama de secuencias

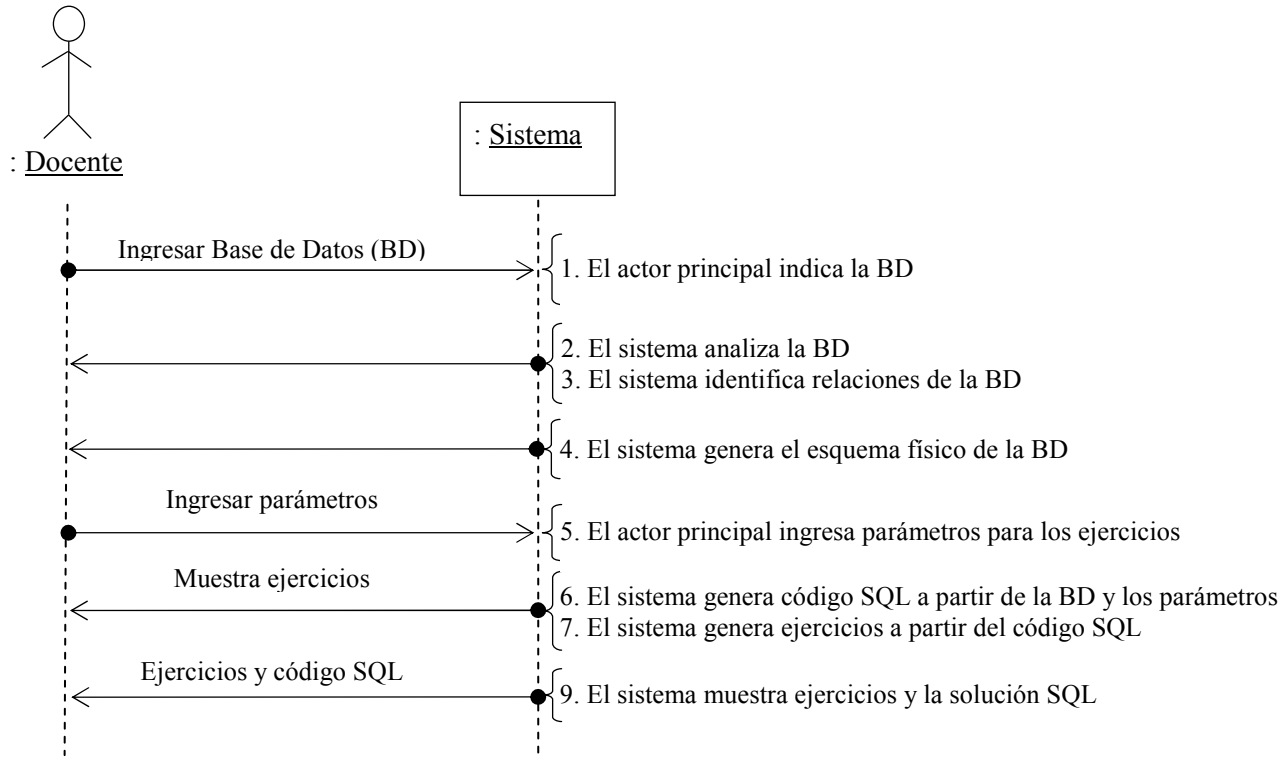


Diagrama de secuencias

6.7 Código del cliente

Clases con atributos y métodos propios

class **usuario** {
private char Usuario;
private char Password;
private boolean}

class **catalogobd** {
private int idCatalogo;
private char NombreBase;
private date FechaIngreso
private char fkUsuario}

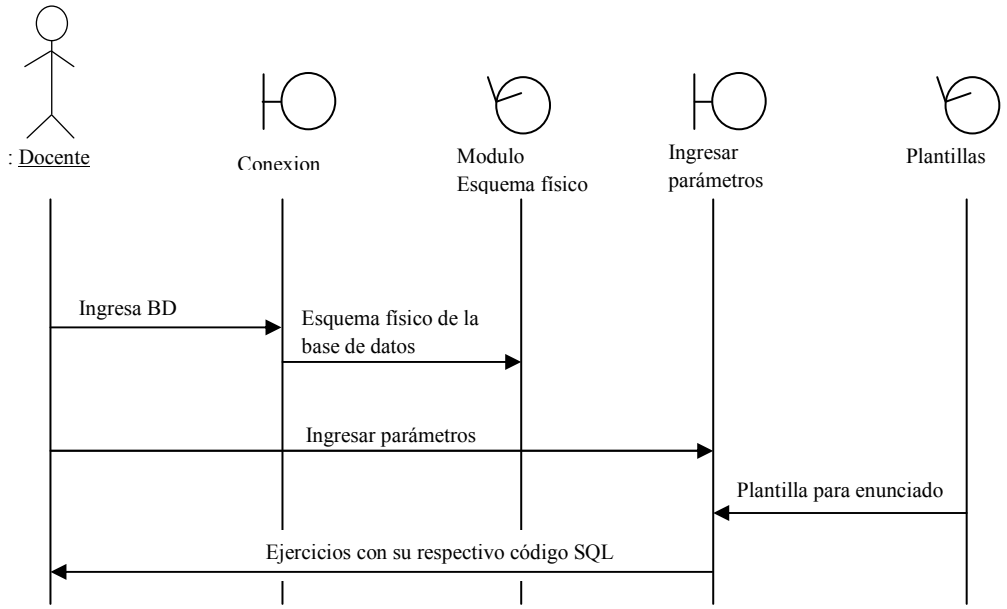
```
class plantillas {
private int idPlantillas;
private char PEnunciado;
private char PResultado}

class ejercicios {
private int idEjercicio;
private date FechaCreacion;
private char Enunciado;
private char códigoSQL;
private int fkCatalogoBD;
private int fkPlantilla;
private char fkUsuario}

public class Jacobson{
public static void main(String[]argc){
Boolean ingresa.BaseDatos;
Usuarios Usuario[];
Usuarios = new Usuario[45];
CatalogoBD = new BaseDatos;
Plantillas = new Plantilla[300];
Ejercicios = new Ejercicio[300];
Usuarios Usuario[];
Usuario = new BaseDatos[];
Usuarios Usuario = new catalogoBD[];
Usuario = new Ejercicio[300];
Usuario[0] = new Ejercicio();
Usuario[0].ingresaParametros();
EsquemaFisico();
CodigoSQL();
Plantilla.EnunciadoEjercicios();
Ejercicio();
```

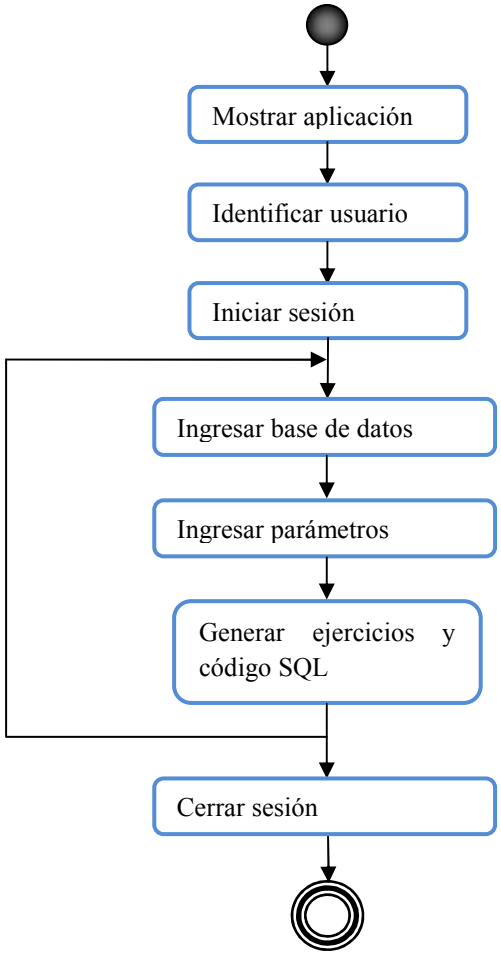
7. Metodología OMT

7.1 Diagramas de actividades del modelo dinámico

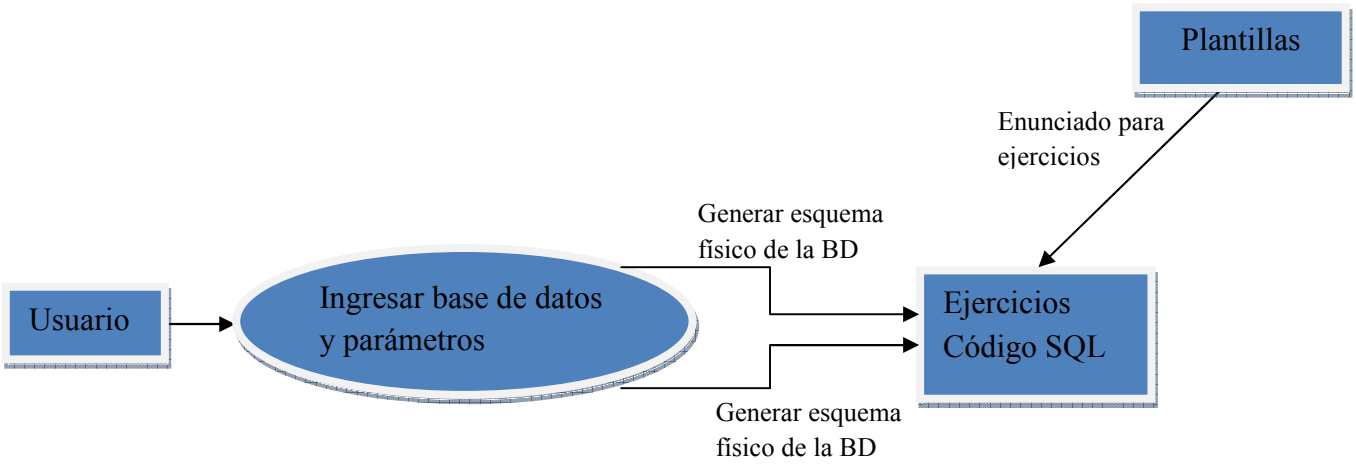


El docente indica al sistema cuál será la base de datos a la cual se deberá conectarse. Conexión a la base de datos.
Módulo generar esquema físico de la base de datos: El sistema se conecta a la base de datos indicada por el docente. Analiza la base de datos e identifica las relaciones, tablas, campos, tipos de datos, llaves y registros.
El sistema recibe parámetros para generar el código SQL tales como: Tipos de acción (SELECT, INSERT, DELETE, UPDATE, JOIN, subconsultas), tablas, número de ejercicios en base a estos parámetros el sistema genera el código SQL y los ejercicios.
El sistema mostrará los ejercicios con su respectivo código SQL.

7.2 Diagrama de actividades general



7.3 Diagrama de flujo de datos



7.4 Optimizar acceso a datos

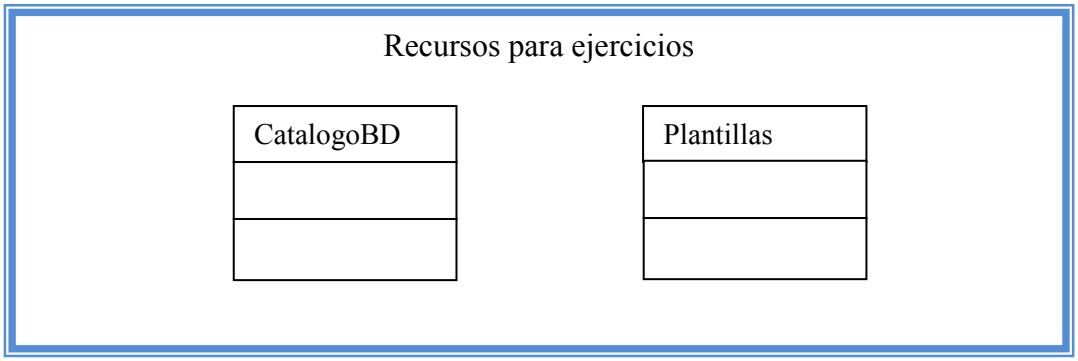
Además de mostrar la información en pantalla, se guardara en una ruta definida.

7.5 Implementar el control de software

7.5.1 Implementar herencia

No es posible implementar herencia en las clases ya que todas son totalmente diferentes, ninguna tiene atributos o métodos que se puedan heredar.

7.5.2 Definir paquetes



8. Pruebas

8.1 Grafo de la función interna

8.1.1Ciclomática

Validar: Usuario y password correcto

{/*1*/ Ingresar valor (Usuario)

/*2*/ Inicialización de la variable (Guardar valor en una variable)

/*3*/ Evaluación del valor Condicion1 (Usuario registrado)

/*4*/ (Condicion1){(Usuario=Usuarios)

/*6*/ if (Condición1)

/*7*/ Ingresar password

/*8*/ Evaluación del valor condición2 (Password correcto)

/*9*/ (Condición2) {(Password = Password)

/*10*/ if(Condición2)

/*11*/ Ingresa a la siguiente pantalla(Conexion)

else /*12*/ Notifica “Password invalido”;

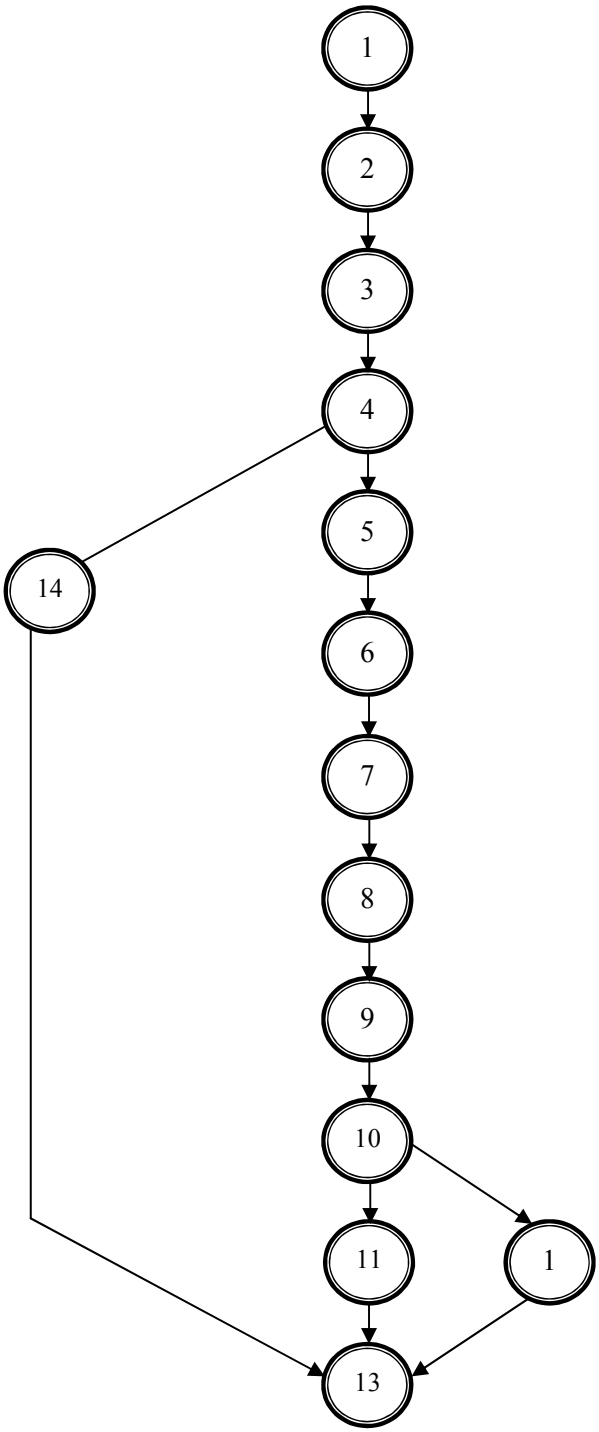
/*13*/ Terminar

else /*14*/ Notifica “Usuario invalido”;

/*13*/ Terminar

/*14*/ Terminar

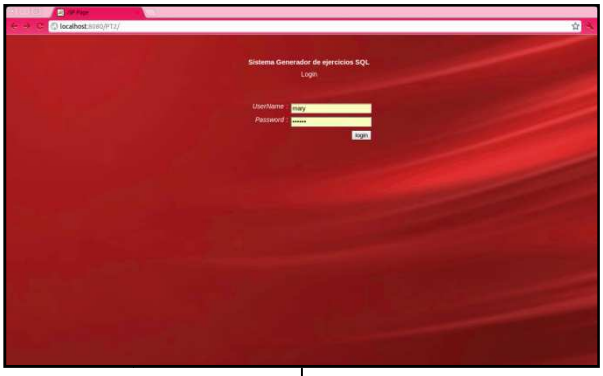
Grafo de flujo



// validar que la base contenga registros
// validar que la base de datos se encuentre direccionada (Ingresada) por el docente
// validar que el docente haya ingresado los parámetros para generar ejercicios.

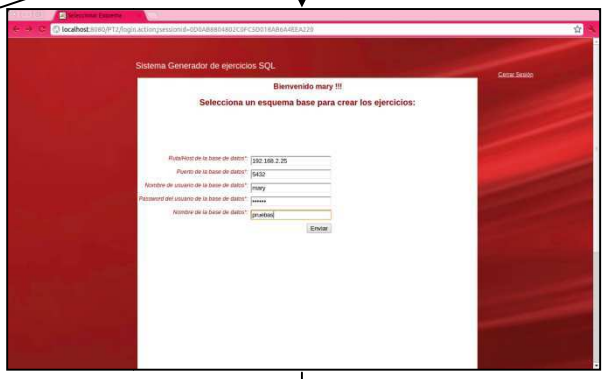
Diagrama de funcionamiento con pantallas

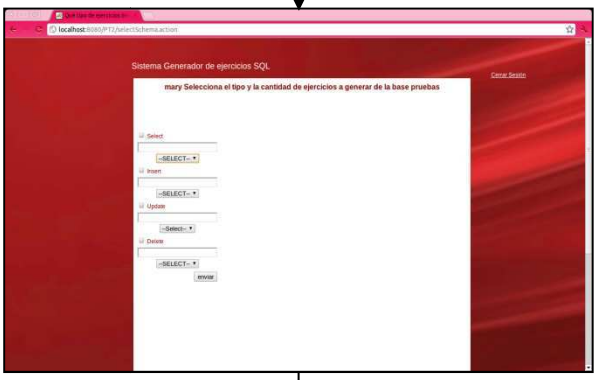
Navegabilidad



Credenciales para ingresar al sistema

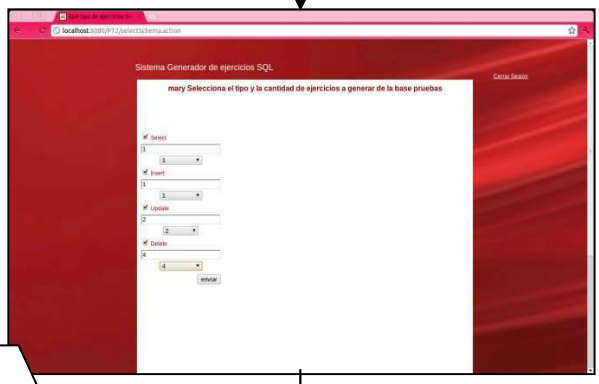
Definir base de datos

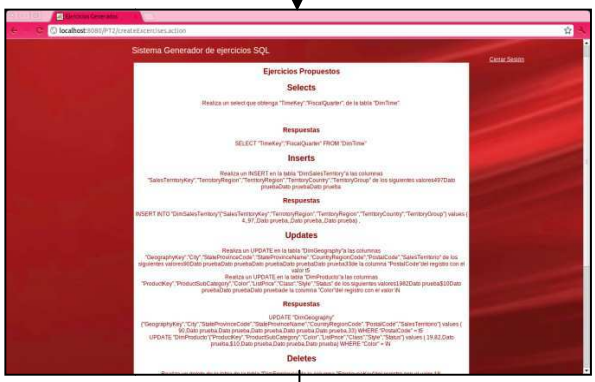




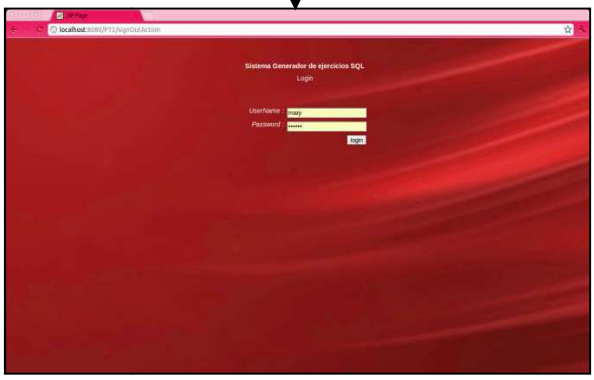
Selección de sentencias SQL

Selección de sentencias SQL



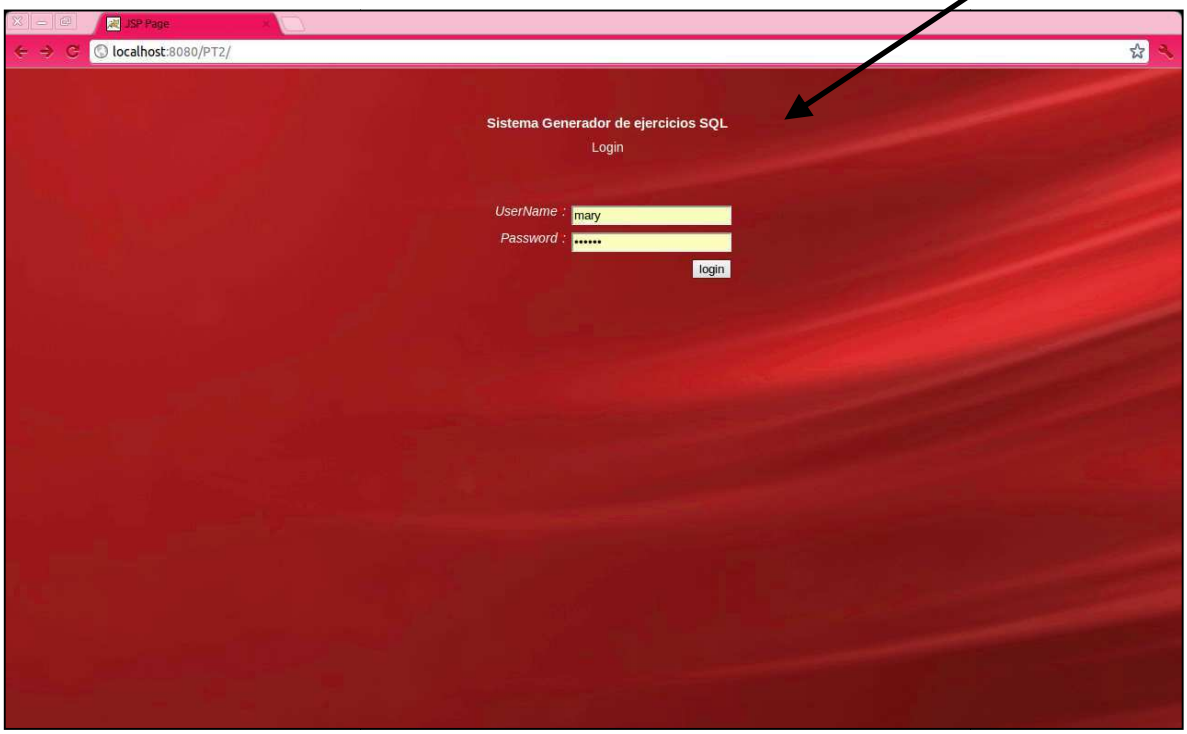


Mostrar ejercicios y código SQL

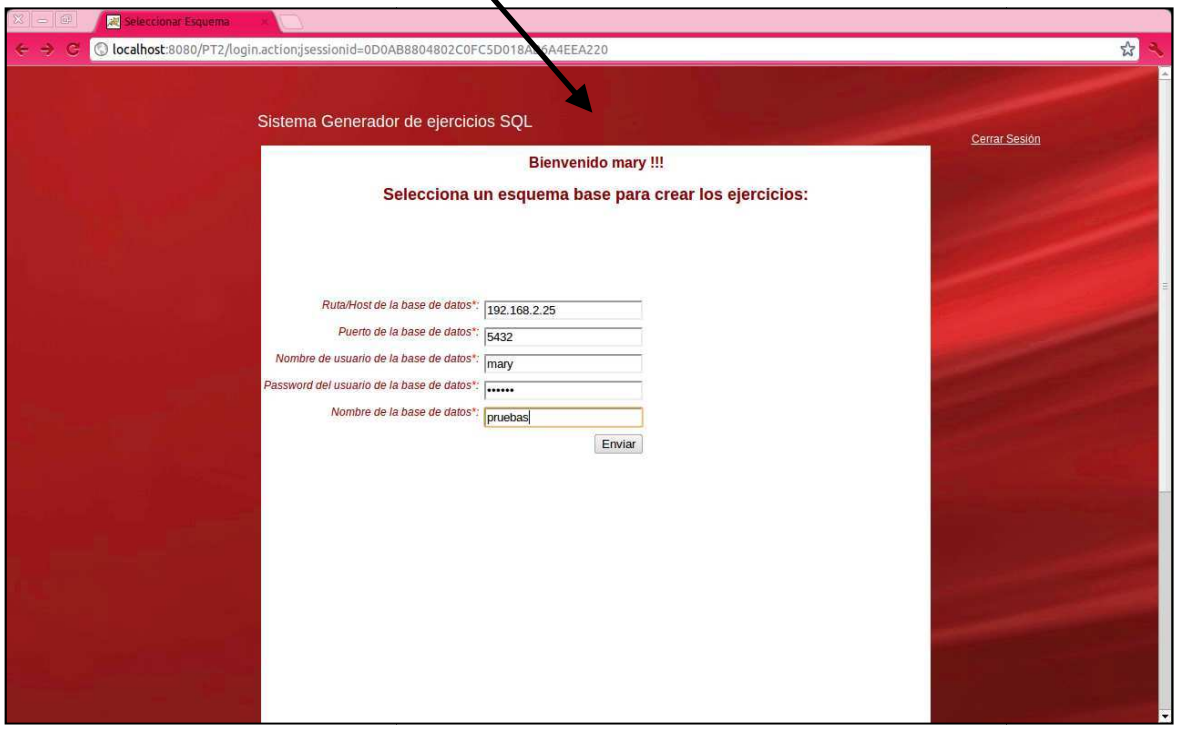


Pantalla de inicio de sesión

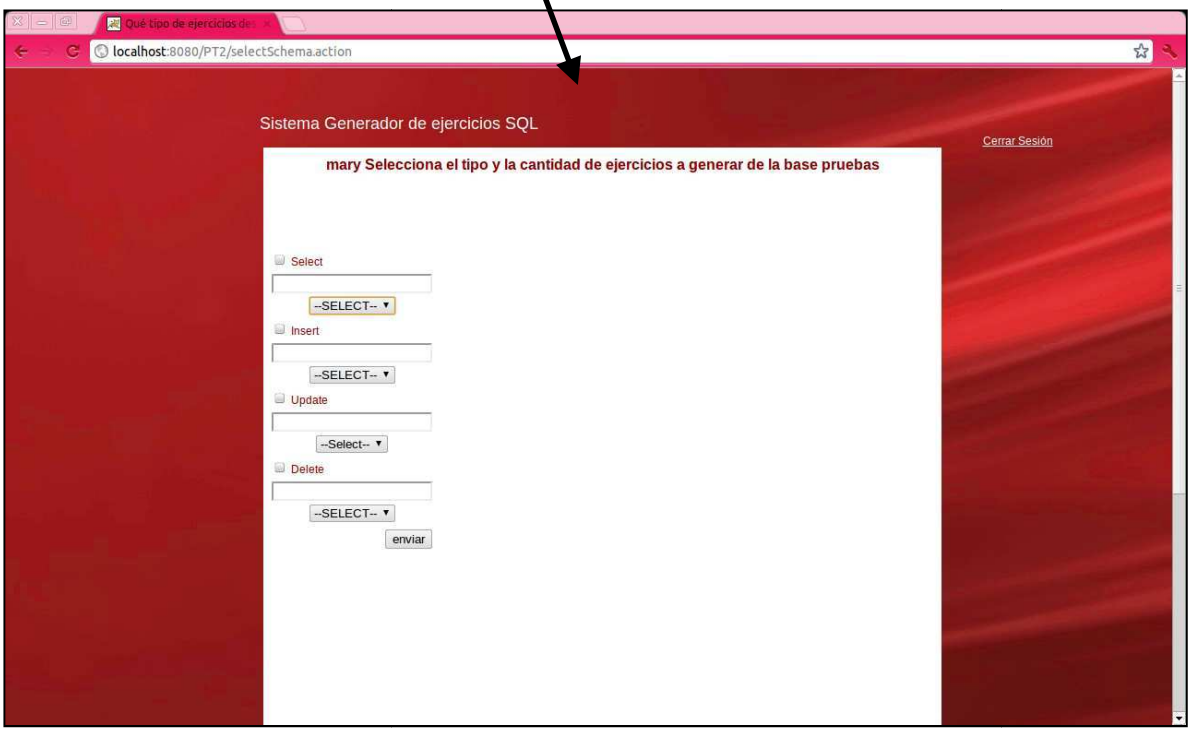
Ingresar las credenciales correspondientes para entrar al sistema (Credenciales actuales Usuario:mary Password: mary00, Usuario: hugoleyva Password: hugoleyva).

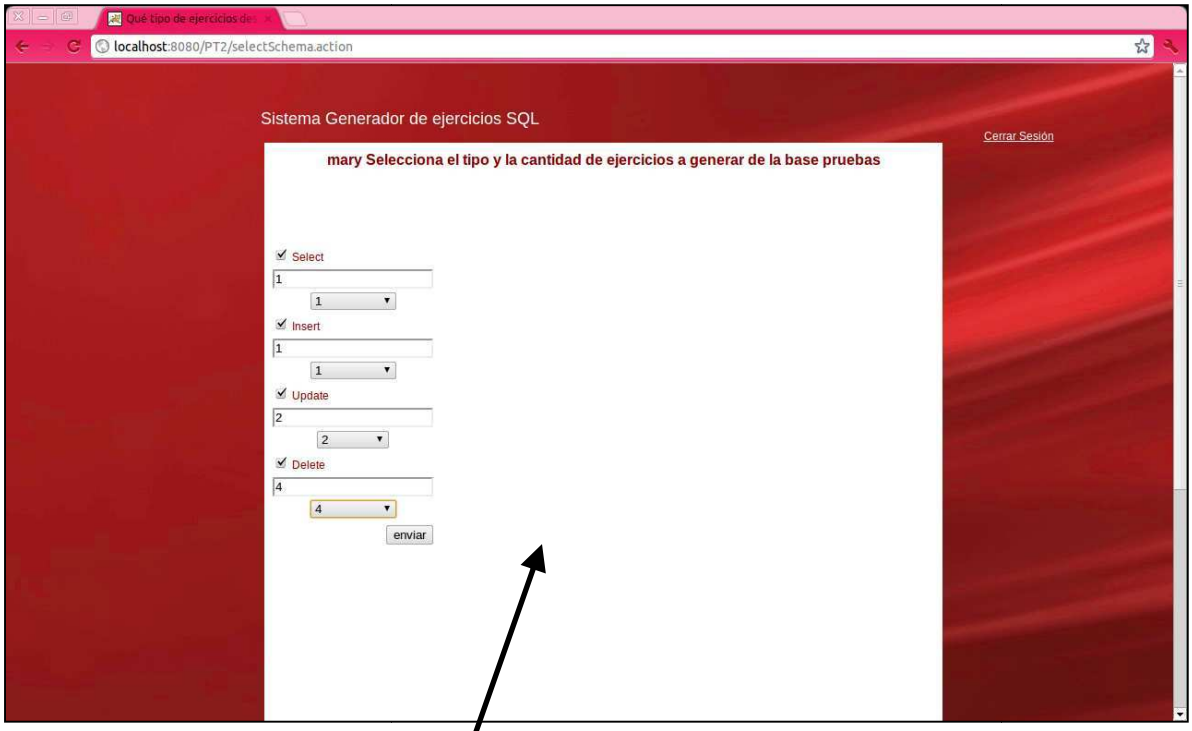


Ingresar datos correspondientes de la base de datos a explotar. En el caso del ejemplo mostrado la ip es 192.168.2.25, el puerto para postgresql es 5432, el usuario:mary, password:mary00, nombre de la base de datos: pruebas



Pantalla para seleccionar el tipo de sentencia SQL deseado y el número de ejercicios.

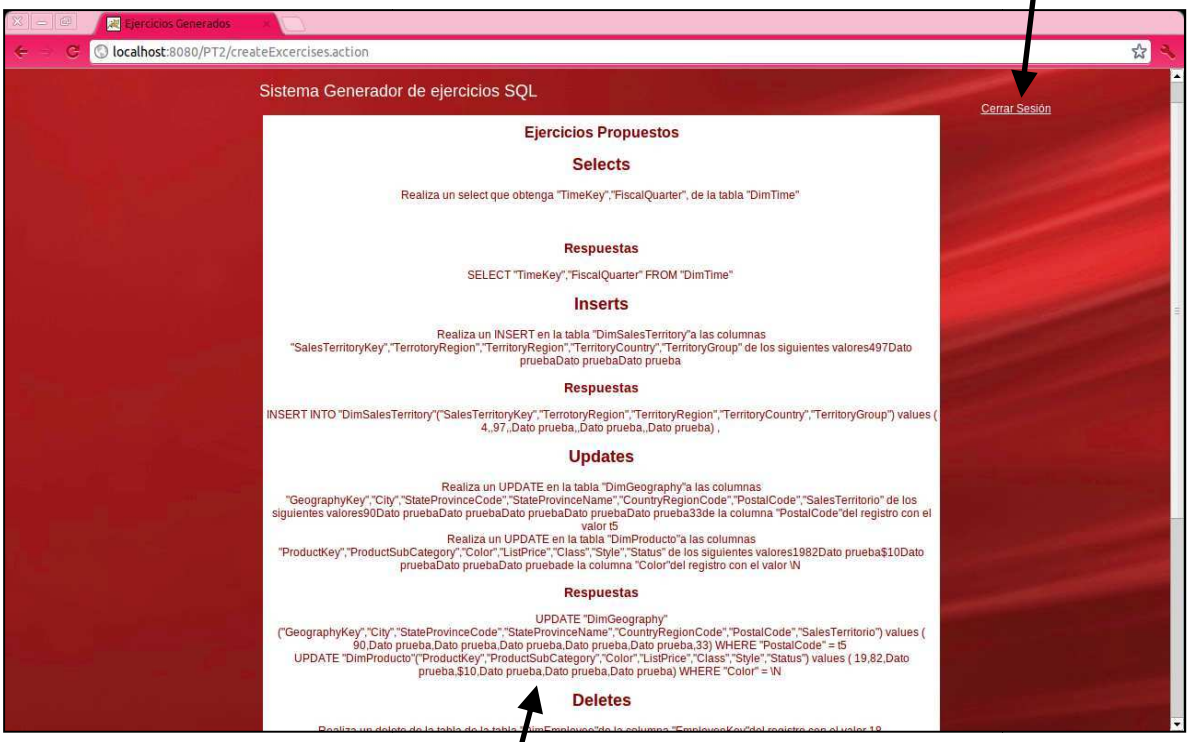




Tipo de sentencia SQL : SELECT, UPDATE, INSERT, DELETE.

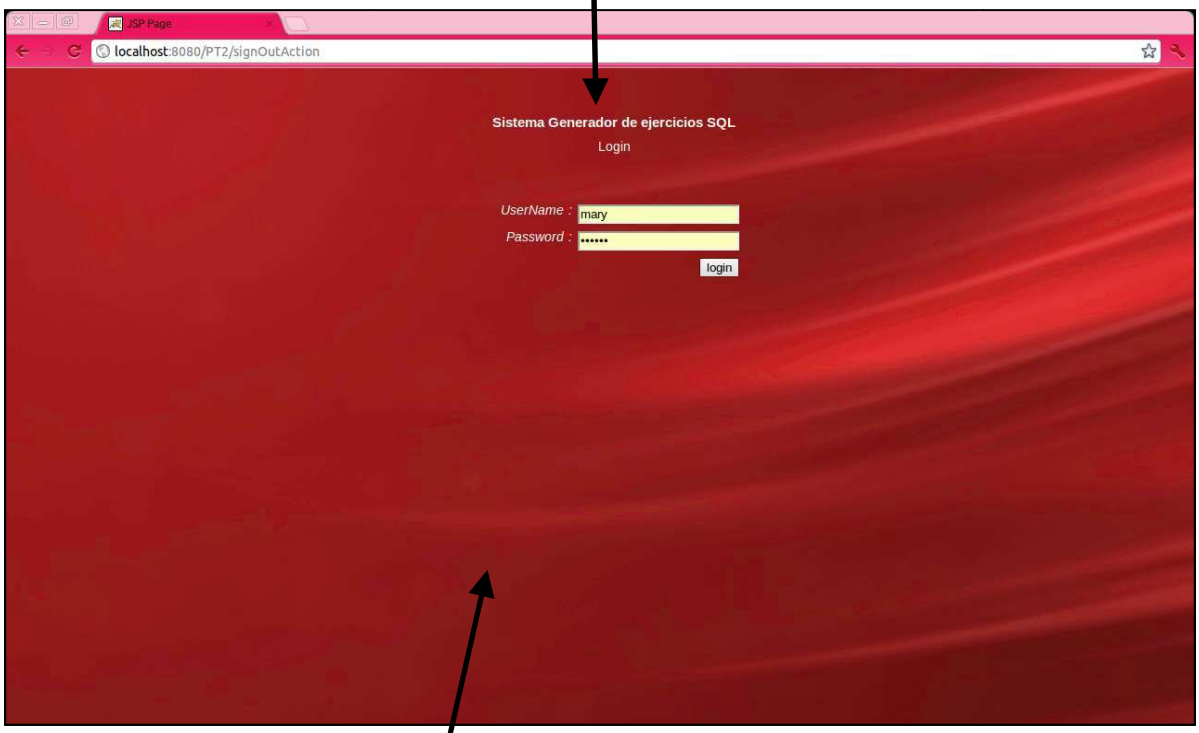
Número en cada sentencia: 1-5

Cerrar sesión



Muestra el resultado de los ejercicios seleccionados en la pantalla anterior con su respectiva respuesta (Código SQL)

Al cerrar sesión, la aplicación regresa a la pantalla inicial.



Página de inicio de sesión para la aplicación (Usuarios registrados Usuario:mary Password:mary00, Usuario:hugoleyva Password:hugoleyva

Manual de instalación

1. Configurar el firewall para abrir el puerto 5432.
   1.2 Ejecutar el siguiente comando en la terminal:
   sudo ufw allow 5432

   1.3 Para comprobar si el Puerto se encuentra escuchando ejecutar en la terminal el commando:
   telnet localhost 5432
   La respuesta debe ser character scape.

2. Instalación de jdk 1.7

   2.1 Descargar jdk de la pagina oficial:
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
   (Aceptar los términos de la licencia para poder iniciar la descarga)
   2.2 Mover de carpeta el paquete con los siguientes comandos:
   cd Descargas
   sudo mv jdk-7u4-linux-x64.tar.gz /usr/home
   cd /usr/home
   sudo tar –zxvf jdk-7u4-linux-x64.tar.gz
   sudo mv jdk1.7.0_04 / jdk1.7

   Con esto tendremos los archivos binarios y el resto de los archivos en /usr/home/jdk1.7

   2.3 Agregar la ruta de java al PATH de la terminal con el comando:
   sudo gedit /etc/bash.bashrc
   2.4 Agregar las siguientes líneas al final del archivo.
   #PATH DE JAVA
   export JAVA_HOME=/home/user//jdk1.7
   export PATH=$JAVA_HOME/bin:$PATH
   2.5 Comprobar la instalación con el comando: which java

3. Instalar el servidor web Apache Tomcat.
   2.1 Descargar el servidor Apahce Tomcat, disponible en la página.
   http://tomcat.apache.org/
   2.2 Descomprimir el archivo con el comando: tar xvzf apache-tomcat-7.0.29.tar.gz
   2.3 Mover el archivo a la dirección /usr/share/tomcat6 con el siguiente comando: sudo mv apahce-tomcat-7.0.29 /usr/share/tomcat6
   2.4 Editar el archivo .bashrc con el comando: sudo /usr/share/tomcat6/bin/startup.sh.
   2.5 Permisos de ejecución con el comando: sudo chmod 755 /etc/init.d/tomcat

4. Base de datos de usuario
   4.1 Ejecutar el scrip backup_ptdb anexo en el CD, con el comando pg_dump del manejador PostgreSQL

5. Instalación de PostgreSQL 9.1 (postgres-9.1 y postgresql-client-9.1)
   5.1 Para la instalación de los 2 paquetes ejecutar el comando:
   Sudo aptitude install postgresql-9.1 postgresql-client-9.1

6. Configurar del manejador PostgreSQL para permitir conexiones remotas.
   6.2 Editar el archivo /etc/sysconfig/iptables
   6.3 Agregar la siguiente línea: A INPUT -m state --state NEW,ESTABLISHED -m tcp -p tcp --dport 5432 -j ACCEPT
   6.4 En el archivo /var/lib/pgsql/data/postgresql.conf ubicar la siguiente línea: #listen_address='localhost'
   6.5 descomentar y reemplazar por lo siguiente: listen_address='*'
   6.6 Agregar la siguiente línea al final del archivo pg_hba: host all all 0.0.0.0 0.0.0.0 md5

7. Archivos necesarios
   El archivo necesario en la maquina en donde se ejecutara el sistema son los siguientes:

   - pg_dump_sinpass.sh

   Deben estar guardados en la ruta /home/usr/
   Se encuentran anexos en el CD de instalación.

8. Requisitos del sistema

   9.1 Sistema operativo

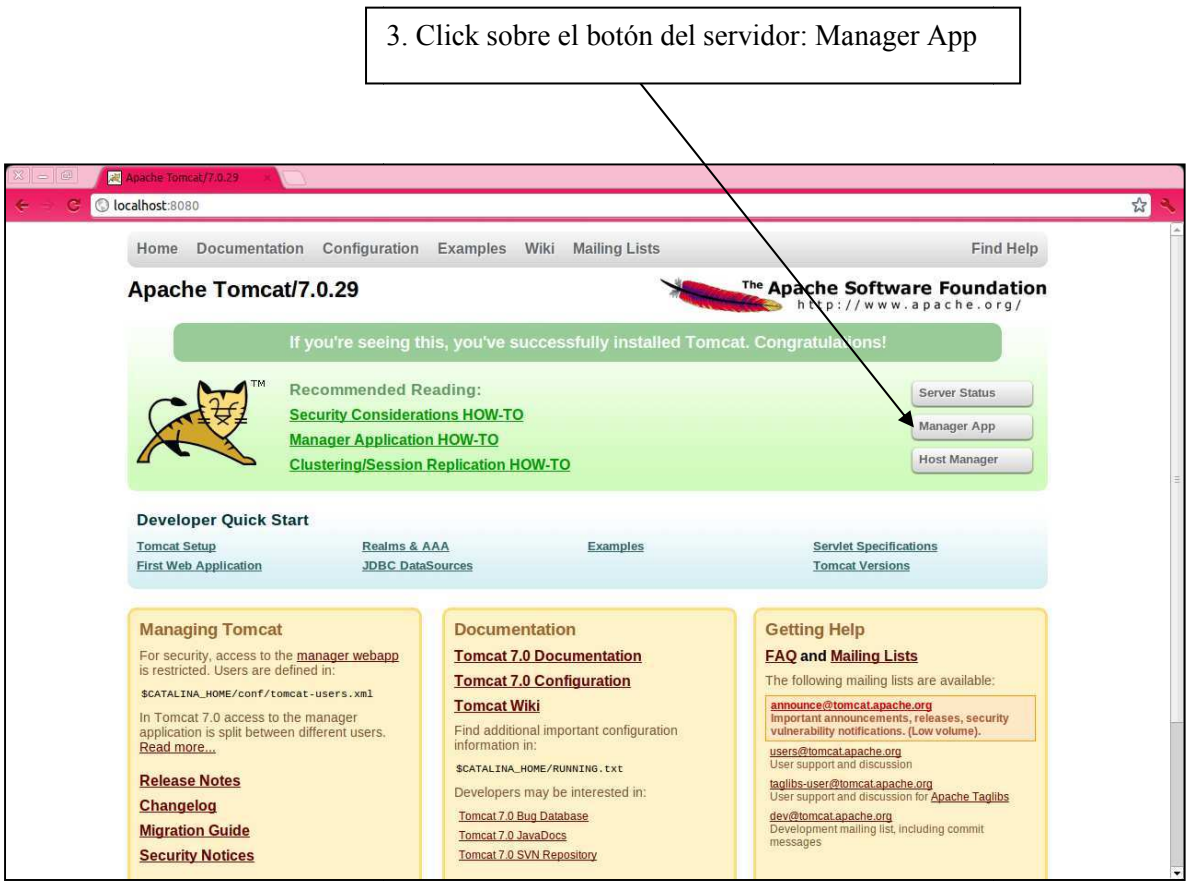
   Sistema operativo: Ubuntu o similar (Linux).

   8.2 Interprete de comandos
   Tener instalado en la maquina en donde se ejecutara el proyecto un interprete de comandos, necesario para interpretar el comando para obtener el esquema físico de la base de datos y para la configuración de la instalación del proyecto.

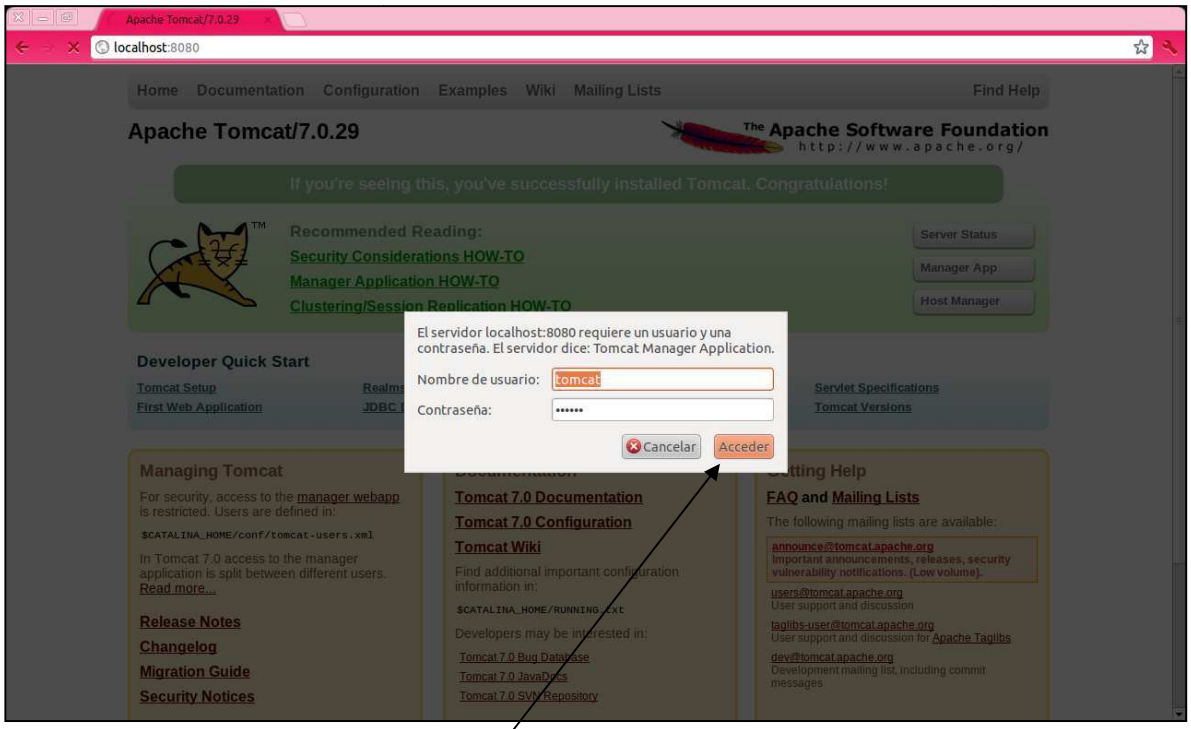
Manual de usuario

Desplegar el proyecto en tomcat.

1. Iniciar el servidor web desde la terminal con el comando:
2. Si el proyecto se encuentra en local colocar la url: localhost:8080 , de lo contrario colocar la ip correspondiente de la maquina en donde se encuentra el proyecto en cualquier explorador.

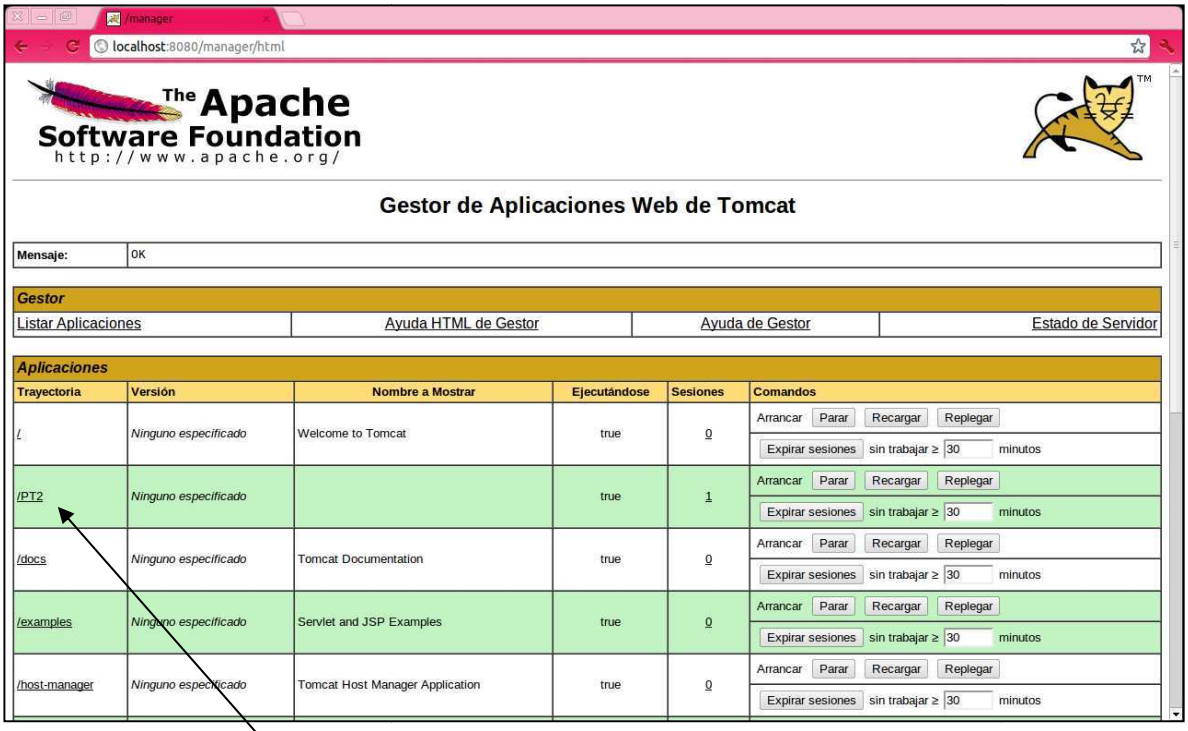


4. Ingresar las credenciales correspondientes del servidor Apache Tomcat.



5. Click en el botón: Acceder

6. Desplegar el archivo *.war que genere el código de la aplicación o desplegar el archivo PT2.war anexo en el CD (El funcionamiento dependerá de la configuración del manual de instalación y de que el PATH del código no cambie en la maquina en donde se ejecuta el proyecto).



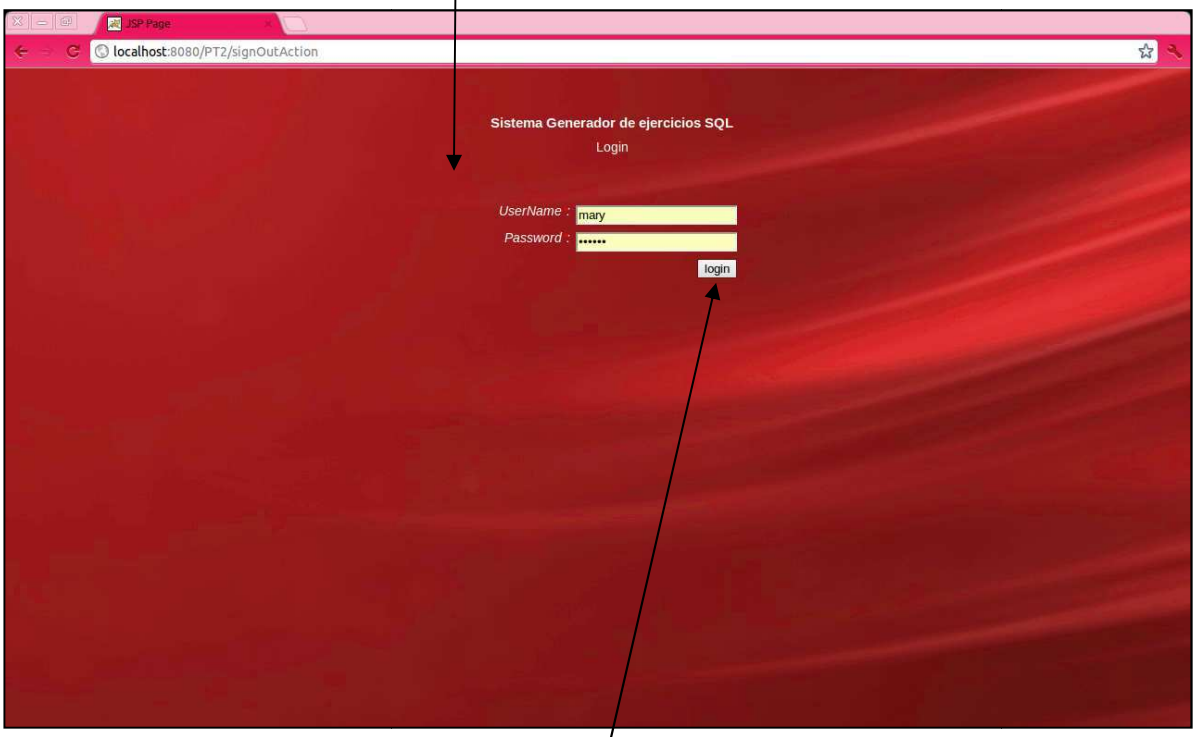
7. Click sobre el nombre del archivo PT2.war

8. Al concluir los pasos anteriores se mostrara en el explorador la aplicación.

9. Pantalla inicial (login).

Ingresar las credenciales del sistema. Los usuarios registrados son los siguientes:

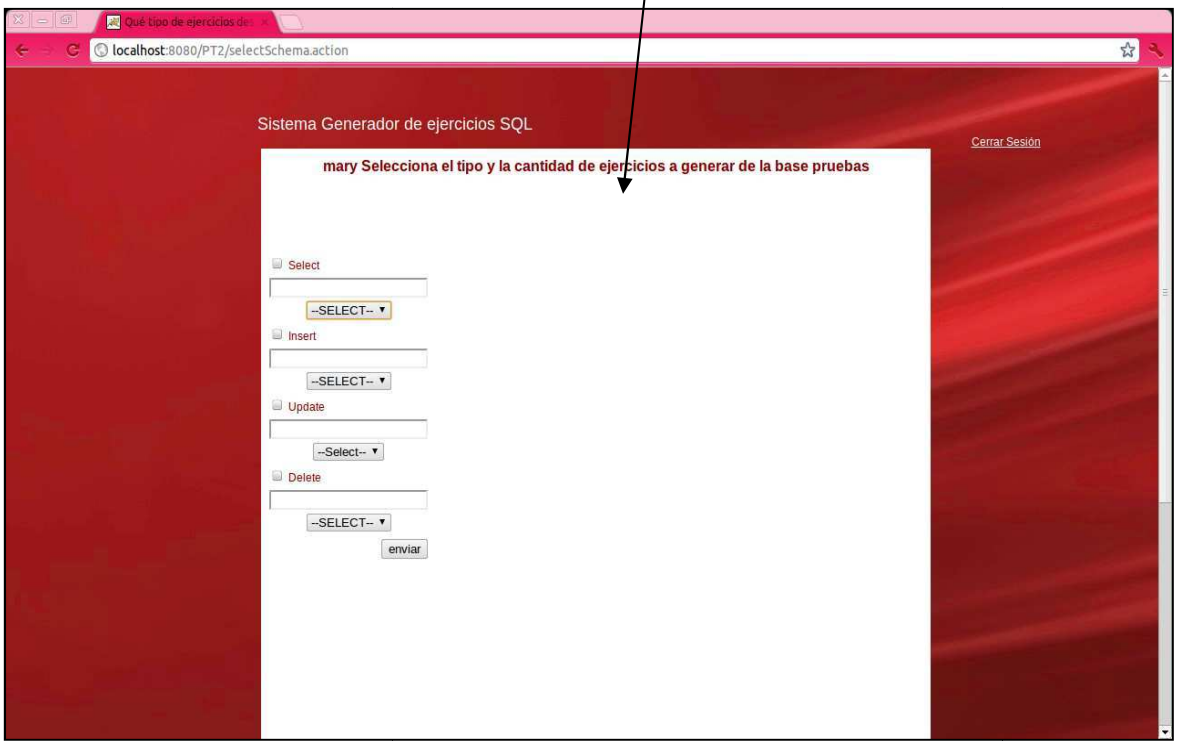
Usuario: mary Password: mary00
Usuario: hugoleyva Password: hugoleyva

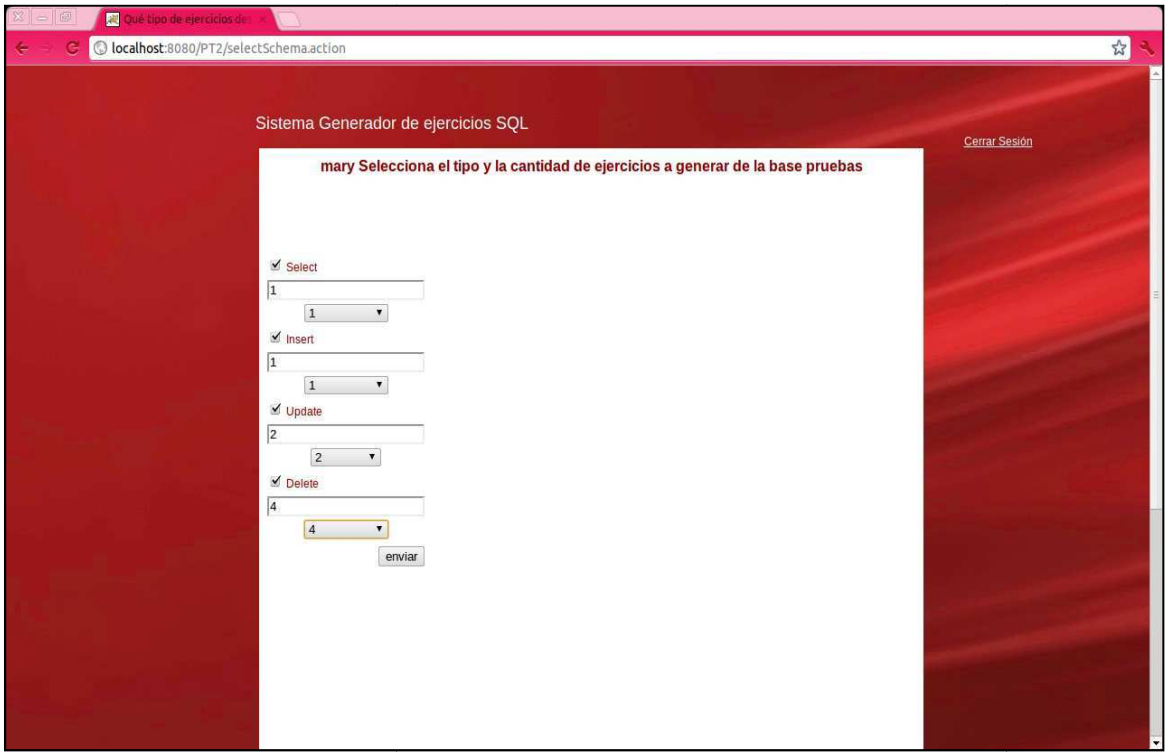


10. Al terminar de ingresar las credenciales, click en el botón “login”

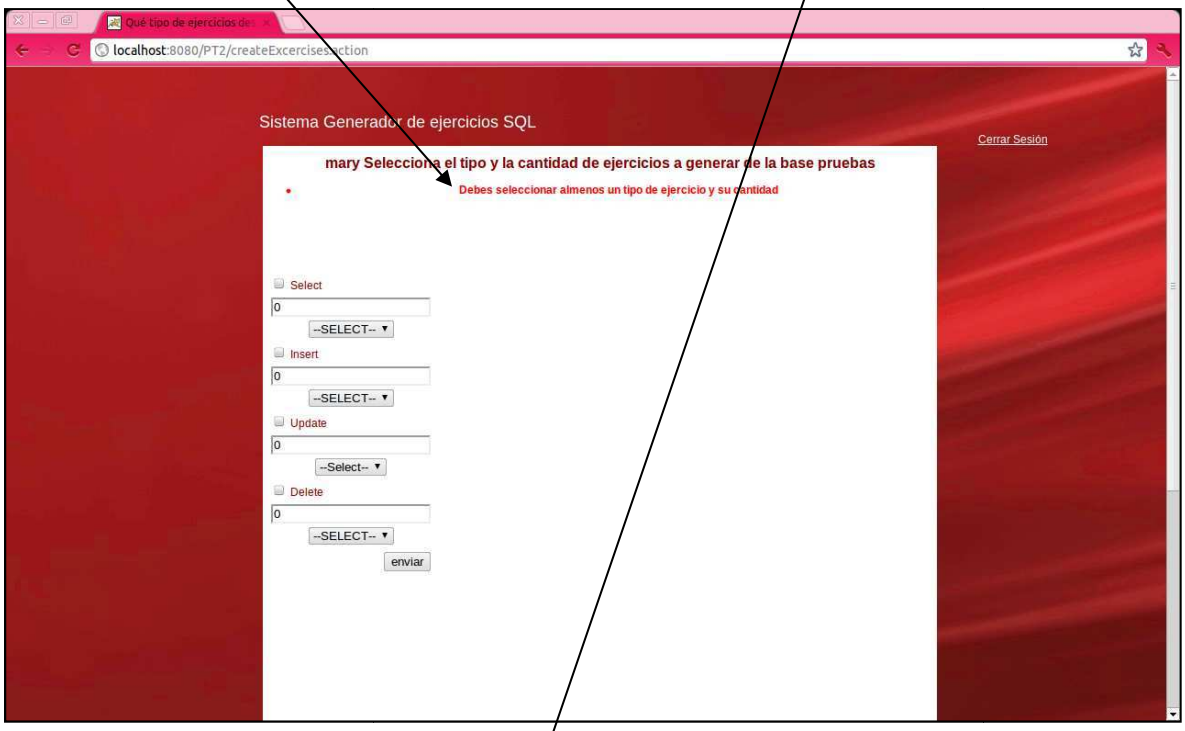
Nota: La aplicación valida los usuarios y password correctos en la base de datos de usuarios.

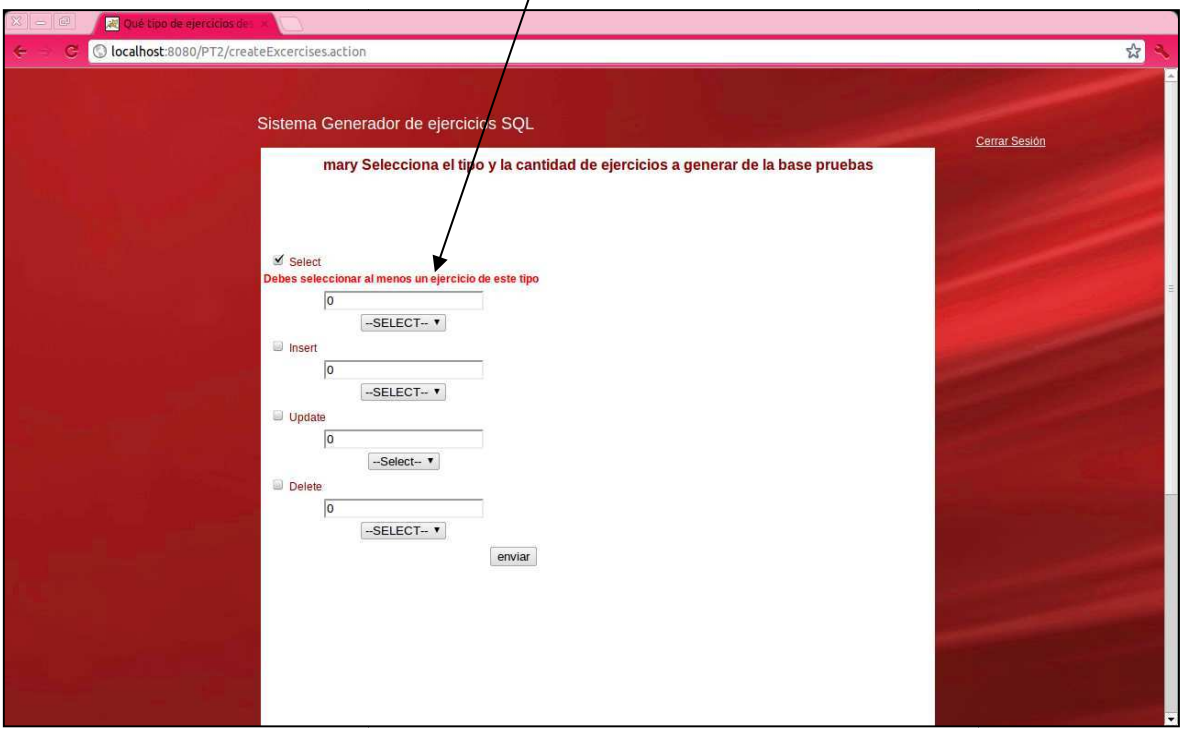
10. En la siguiente pantalla seleccionar la sentencia deseada y el número de ejercicios que se desea para cada sentencia.





Nota: La aplicación valida que sea seleccionada al menos una instrucción SQL.





11. Ingresar los datos correspondientes para la conexión a la base de datos.

1. Ruta / Host de la maquina en donde está ubicada la base de datos a explotar.

Ejemplo: ip = 192.168.2.25

2. Puerto de la base de datos (Para el manejador PostgreSQL puerto = 5432).

3. Nombre del usuario de la base de datos a explotar.

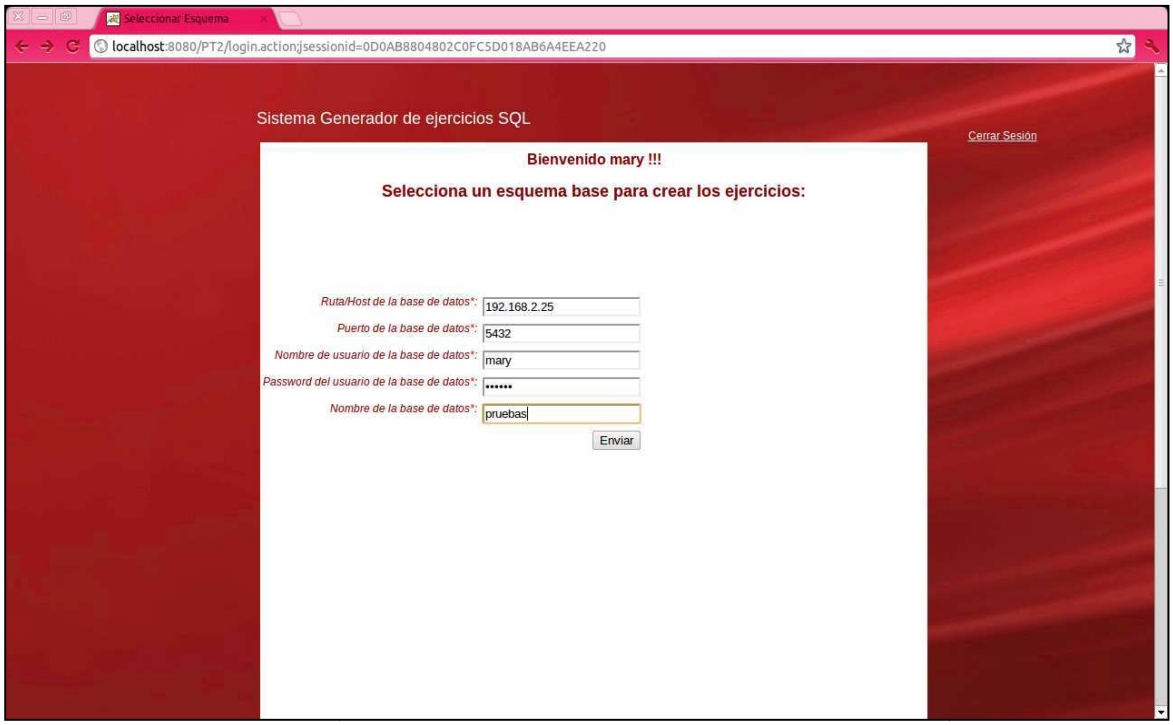
Usuario de la base de datos pruebas = mary

4. Password del usuario de la base de datos.

Password del usuario mary = mary00

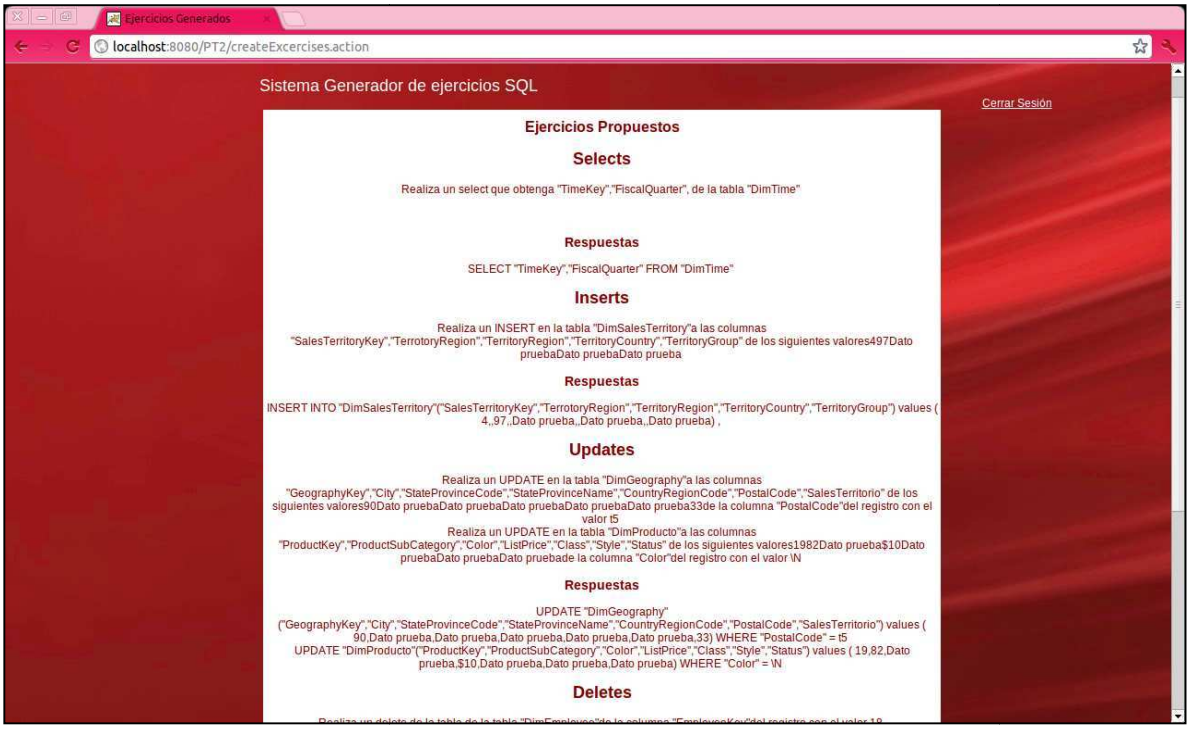
5. Nombre de la base de datos.

El nombre de la base de datos anexa en el CD es: pruebas

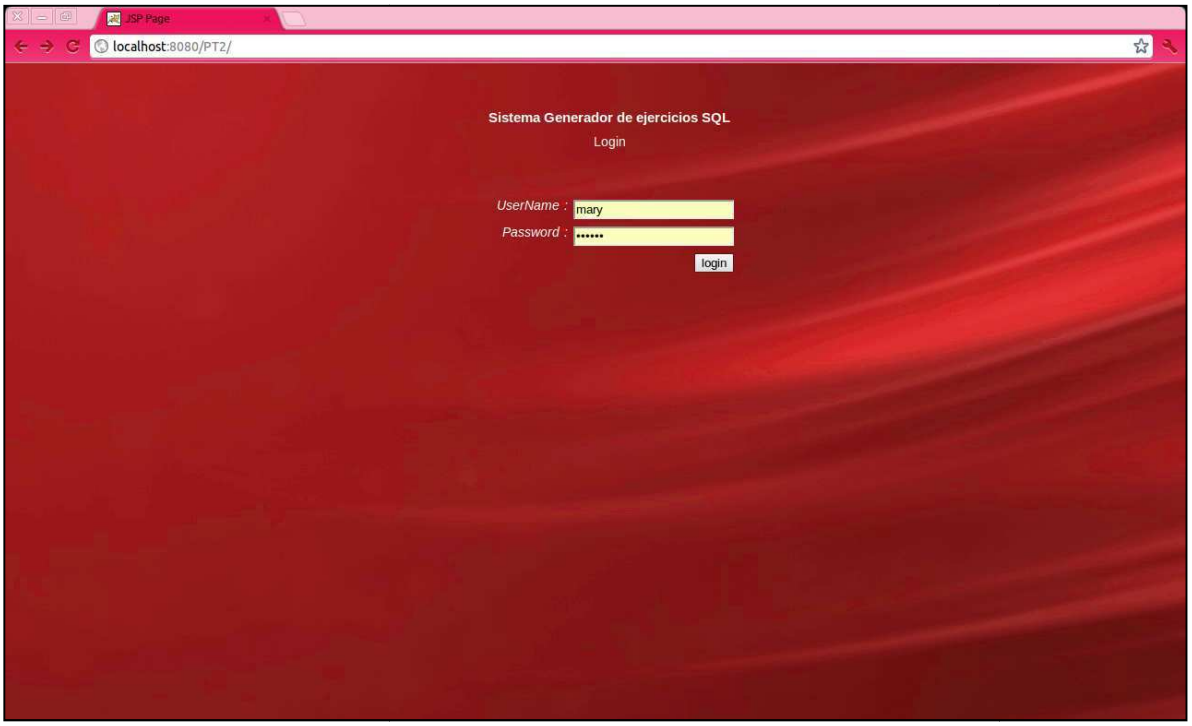


12. Al ingresar los datos correctamente la aplicación mostrara las sentencias seleccionadas anteriormente y el número de ejercicio deseado, con su respectiva respuesta en código SQL para cada una en la siguiente pantalla.

Como se muestra en el ejemplo:



13. Al terminar cerrar sesión, Click en el menú “Cerrar sesión”. El sistema mostrara la pantalla inicial “Login”.



Universidad Autónoma Metropolitana

Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto Terminal

Sistema generador de ejercicios del uso de SQL para explotar la información de una base de datos

María Del Carmen Carreón Huerta - 207206219

Fecha de entrega: 10-Diciembre-2012

Asesor:

M. en C. Hugo Pablo Leyva

Profesor titular

Departamento: Sistemas

Índice

1. Introducción

1.1 Planteamiento de la problemática que se aborda con el proyecto terminal

El sistema propuesto será de gran ayuda en la generación de problemas para poder evaluar alumnos de cursos de bases de datos o relacionados. También servirá de apoyo al calificar los mismos ya que dará el código correcto del ejercicio generado. Los resultados del sistema proporcionarán un gran ahorro de tiempo a un docente.
En particular al generar ejercicios a partir de una base de datos indicada. Los docentes que imparten esta materia tienen la necesidad de buscar ejemplos y problemas para que sus alumnos practiquen y sean evaluados. Al dar como ejemplo una base de datos el docente tiene que estudiarla para dominar los problemas que pueden solicitar a los alumnos.

1.2 Marco teórico

1.2.1 Código SQL

SQL (Lenguaje de Consulta Estructurado), es un lenguaje declarativo de acceso a bases de datos relacionales que permiten especificar diversos tipos de operaciones en ellas. Permiten efectuar consultas por medio del manejo de álgebra y el cálculo relacional con el fin de recuperar de forma sencilla información en las bases de datos, así como hacer cambios en ellas.

Su origen están ligados a las bases de datos relacionales con la contribución de las investigaciones de los laboratorios IBM que definen el lenguaje SEQUEL (Structured English Query Language) que más tarde seria implementado por el SGBD (Sistema de gestión de bases de datos) desarrollado también por IBM.
En 1979 Oracle fue quien lo introdujo por primera vez en un programa comercial.
SQL es el estándar de facto de la inmensa mayoría de los SGBD comerciales.

SQL es un lenguaje de acceso a bases de datos que explotan la flexibilidad y potencia de los sistemas relacionales y permiten una gran variedad de operaciones. Es un lenguaje declarativo de “alto nivel” o “de no procesamiento” con una fuerte base teórica y orientación al manejo de conjuntos de registros, permiten una alta productividad en codificación y la orientación a objetos. De esta forma una sola sentencia puede equivaler a uno o más programas de bajo nivel orientado a registros.

Características:
Lenguaje de definición de datos (LDD). El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación
Lenguaje interactivo de manipulación de datos. El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.
Integridad. El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.
Definición de Vistas. El LDD incluye comandos para definir las vistas
Control de transacciones. SQL tiene comandos para especificar el comienzo y el final de una transacción.
SQL Incorporado y dinámico. Puede incorporar instrucciones de SQL en lenguajes de programación como c++, c, java, Cobol, Pascal y Fortran
Autorización. El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas

Optimización.
SQL es un lenguaje declarativo, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.
El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a cabo una optimización antes de su ejecución. El uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos. Dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.
Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa. Este lenguaje ha sido implementado a nivel experimental y está evolucionando rápidamente.
Lenguaje de definición de datos (DDL): El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

1.2.2 Esquema físico de una base de datos

El esquema de una base de datos (Database Schema) describe la estructura de una Base de datos, en un lenguaje formal soportado por un Sistema administrador de Base de datos (DBMS). En una Base de datos Relacional, el Esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla.

El esquema es generalmente almacenado en un Diccionario de Datos. El esquema es definido en un lenguaje de Base de datos, el término se usa para referirse a una representación gráfica de la estructura de base de datos.

Niveles de Esquema de Base de datos

- Esquema Conceptual, un mapa de conceptos y sus relaciones.
- Esquema Lógico, un mapa de las entidades y sus atributos y las relaciones.
- Esquema Físico, una aplicación de un esquema lógico.
- Esquema Objeto, Base de datos Oracle Objeto.

Para la obtención del esquema físico de la base de datos se utilizo el comando del manejador postgreSQL pg_dump

pg_dump es un utilitario para volcar una base de datos Postgres en un fichero de script conteniendo comandos de consulta. Los ficheros de script son en formato de texto y pueden ser usados para reconstruir la base de datos, incluso en otras máquinas y con otras arquitecturas. pg_dump producirá las consultas necesarias para regenerar todos los tipos definidos por el usuario, funciones, tablas, índices, agregados, y operadores. Adicionalmente, toda la data es copiada en formato de texto el cual puede ser nuevamente copiado, también puede ser importado a herramientas para su edición.

pg_dump es útil para verter el contenido de una base de datos que se vaya a mudar de una instalación de Postgres a otra. Después de ejecutar pg_dump, se debe examinar el script de salida a ver si contiene alguna advertencia, especialmente a la luz de las limitaciones citadas en la parte inferior.

Entradas y salidas:

Entrada: `pg_dump` accepta los siguientes argumentos de la línea de comando	
base_de_datos	Epecifica el nombre de la base de datos que se va a extraer. *`base_de_datos`* tiene como estándar el valor de la variable de entorno `USER`
-a	Vuelca sólo los datos, no el esquema (las definiciones).
-c	Limpia el esquema antes de crearlo.
-d	Vuelca la data como propios insertos de cadenas.
-D	Vuelca la data como insertos con nombres de atributos

-n	Suprime las dobles comillas de los identificadores, a menos que sean absolutamente necesarias. Esto puede causar problemas al cargar la misma si esta data volcada contiene palabras reservadas usadas por los identificadores. Esta era la conducta estándar en `pg_dump` pre-v6.4.
-N	Incluye comillas dobles en los identificadores. Este es el estándar.
-o	Vuelca los identificadores de objetos (OIDs) para cada tabla.
-s	Vuelca solo el esquema (las definiciones), no la data.
-t *tabla*	Vuelca la data para la *tabla* únicamente.
-u	Usa autenticación por medio de clave de acceso. Pide un nombre de usuario y clave de acceso.
-v	Epecifica el modo verbose(parlanchín)
-x	Evita el volcado de ACLs (comandos grant/revoke) y la información de propiedad de la tabla.
conexión	
-h *huésped*	Especifica el nombre del huésped de la máquina en la cual se está ejecutando el `postmaster`. El estándar es usar un socket de dominio local Unix en vez de una conexión IP.
-p *puerto*	Especifica el puerto de Internet TCP/IP o extensión de archivo socket de dominio local Unix en el cual `postmaster` está esperando que se efectúen conexiones. En número estándar de puerto es 5432, o el valor de la variable de ambiente `PGPORT` (si está establecida).
-u	Usa autenticación con clave de acceso. Pide *nombre_de_usuario* y *clave_de_acceso*.
Salida: `pg_dump` creará un fichero o escribirá a **`stdout`.**	
La conexión con la base de datos 'template1' falló. connectDB() falló: ¿Está el postmaster ejecutándose y acceptando conexiones en el 'Socket de UNIX' en el puerto '*puerto*'	pg_dump no pudo unirse al proceso postmaster en el huésped y puerto especificados. Si ve usted este mensaje, verifique que postmaster se este ejecutando en el huésped indicado, y que usted especificó el puerto correcto. Si su site usa algún sistema de autenticación, verifique que usted tiene las credenciales de autenticación requeridas.
La conexión con la base de datos '*base_de_datos*' falló. FATAL 1: SetUserId: el usuario '*nombre_de_usuario*' no está en 'pg_shadow'	Usted no posee una entrada válida en la relación pg_shadow y no le será permitido tener acceso a Postgres. Contacte a su administrador de Postgres.
dumpSequence(*tabla*): SELECT falló	Usted carece del permiso para leer la base de datos. Contacte a su administrador de site Postgres.

1.2.3 Sentencias SELECT, UPDATE, INSERT, DELETE y JOIN

Lenguaje de manipulación de datos DML(Data Manipulation Language)
[editar] Definición
Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.
El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional.

La sentencia SELECT
La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.

El formato de la sentencia select es:

```
SELECT [ALL | DISTINCT ]
<nombre_campo> [{,<nombre_campo>}]
FROM <nombre_tabla>|<nombre_vista>
[{,<nombre_tabla>|<nombre_vista>}]
[WHERE <condicion> [{ AND|OR <condicion>}]]
[GROUP BY <nombre_campo> [{,<nombre_campo >}]]
[HAVING <condicion>[{ AND|OR <condicion>}]]
[ORDER BY <nombre_campo>|<indice_campo> [ASC | DESC]
[{,<nombre_campo>|<indice_campo> [ASC | DESC ]}]]
```

Ejemplo:

```
SELECT * FROM tCoches ORDER BY marca,modelo;
```

UPDATE
Una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.

Ejemplo

```
UPDATE My_table SET field1 = 'updated value asd' WHERE field2 = 'N';
```

INSERT

Una sentencia INSERT de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional.

Forma básica

```
 INSERT INTO 'tabla' ('columna1',['columna2,... '])
VALUES ('valor1', ['valor2,...'])
```

Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo

```
 INSERT INTO agenda_telefonica (nombre, numero)
VALUES ('Roberto Jeldrez', 4886850);
```

Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada:

```
 INSERT INTO nombreTabla VALUES ('valor1', ['valor2,...'])
```

Ejemplo (asumiendo que 'nombre' y 'número' son las únicas columnas de la tabla 'agenda_telefonica'):

```
 INSERT INTO agenda_telefonica
VALUES ('Jhonny Aguiar', 080473968);
```

Formas avanzadas

Una característica de SQL (desde SQL-92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

```
INSERT INTO ''tabla'' (''columna1'', [''columna2,... ''])
VALUES (''valor1a'', [''valor1b,...'']),
(''value2a'', [''value2b,...'']),...;
```

DELETE

Una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.

Forma básica

```
SQL="DELETE FROM ''tabla'' WHERE ''columna1'' = ''valor1''"
```

Ejemplo

```
DELETE FROM My_table WHERE field2 = 'N';
```

1.3 Trabajos relacionados

Existen las siguientes propuestas de proyectos terminales en la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, que tienen alguna relación con el proyecto propuesto pero no el mismo objetivo:

"Programa de migración de manejadores de bases de datos relacionales" [1]. Programa que realiza la migración de una base de datos entre 4 manejadores (MySQL, PostgreSQL*, DB2*, e Informix), Se obtienen respaldos y ajustes para lograr una migración de una base de datos a diferentes manejadores, pero no resuelve los problemas del proyecto propuesto.

"Herramienta para crear exámenes con distintos formatos de pregunta/respuesta" [2]. Este sistema permite al usuario adicionar preguntas a un curso y dinámicamente posibles respuestas a cada pregunta. El usuario crea, modifica y elimina preguntas. A diferencia del propuesto se obtienen exámenes y no ejercicios. No está orientado a SQL, además, de que el sistema no genera las preguntas. El usuario debe ingresar previamente la pregunta indicando cuál de las respuestas es la correcta en determinado curso. En resumen, el sistema administra preguntas/respuestas y es de gran ayuda para elaborar los exámenes, pero no genera los ejercicios.

"Desarrollo de reactivos y sus respectivas soluciones, para la elaboración de exámenes, a partir de una base de datos" [3] El sistema genera una base de datos para formular automáticamente preguntas con sus respectivas respuestas para exámenes de métodos numéricos. Después, aplica las respuestas a la base de datos. Tiene como limitante que los exámenes están programados en Java. La diferencia es que no genera problemas de la base de datos, ni las respuestas son en código de SQL. El proyecto se enfoca a la materia de Métodos Numéricos, no en SQL.

"Sistema de gestión de calificaciones para los cursos impartidos por un profesor" [4] Sistema que permite a un profesor administrar las calificaciones de los alumnos. Administra: Materias, grupos, evaluaciones, registra y visualiza los cursos. Al igual que el propuesto, ayuda a calificar a los alumnos pero no genera problemas. No está enfocado a un curso/materia específico.

Referencias externas:

"Sistema evaluador de desarrollo de productos de software bajo la norma SPICE" [5] Es un sistema evaluador que permite al usuario llevar a cabo una mejor toma de decisiones sobre el desarrollo de software bajo la norma SPICE[7]. Sólo orienta al usuario sobre el desarrollo del software, pero no genera ejercicios a partir de una base de datos.

"Suit ESCOM" [6]. Sistema que permite dar servicios y ofrecer aplicaciones de red haciendo uso de los conocimientos de programación. Con base en la programación se obtienen recomendaciones y aplicaciones para dar servicios, pero no tiene la finalidad del proyecto propuesto.

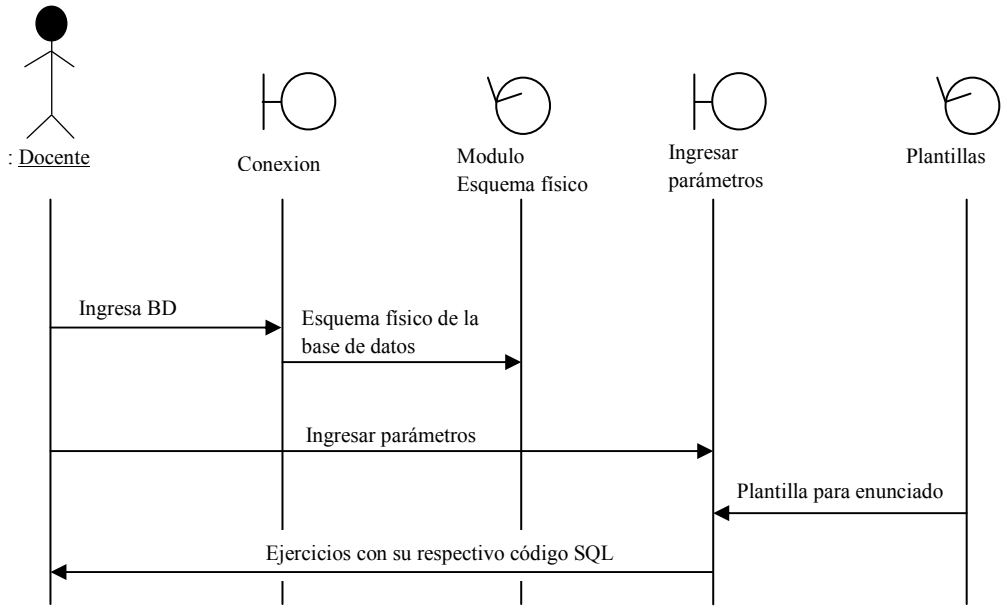
1.3.1 Tabla comparativa

Titulo	**Genera problemas**	**Características**	**Tipo**	**Orientada a SQL**	**Ayuda a evaluar**	**Base de datos**
Herramienta para crear exámenes con distintos formatos de pregunta/respuesta	no	Genera exámenes con preguntas y respuestas	Internos	no	si	si
Sistema de gestión de calificaciones para los cursos impartidos por un profesor	no	Sistema que permita a un profesor administrar las calificaciones de sus alumnos en los diferentes cursos	Internos	no	si	si
Desarrollo de reactivos y sus respectivas soluciones, para la elaboración de exámenes, a partir de una base de datos.	no		Internos		si	si
Programa de migración de manejadores de bases de datos relacionales	no		Internos		no	si
Sistema evaluador de desarrollo de productos de software bajo la norma SPICE	si	Evaluador que permita llevar a cabo una mejor toma de decisiones o estrategias sobre el desarrollo de un producto de software	Externos	no	no	si
Suit ESCOM	si	Sistema da servicios y ofrece aplicaciones de red haciendo uso de los conocimientos de programación con que se cuentan	Externos	no	si	si
Esta propuesta	Si	Si	Interno	Si	si	si

2. Desarrollo funcional

2.1 Diseño

2.1.1 Diagrama de actividades



2.1.2 Casos de uso

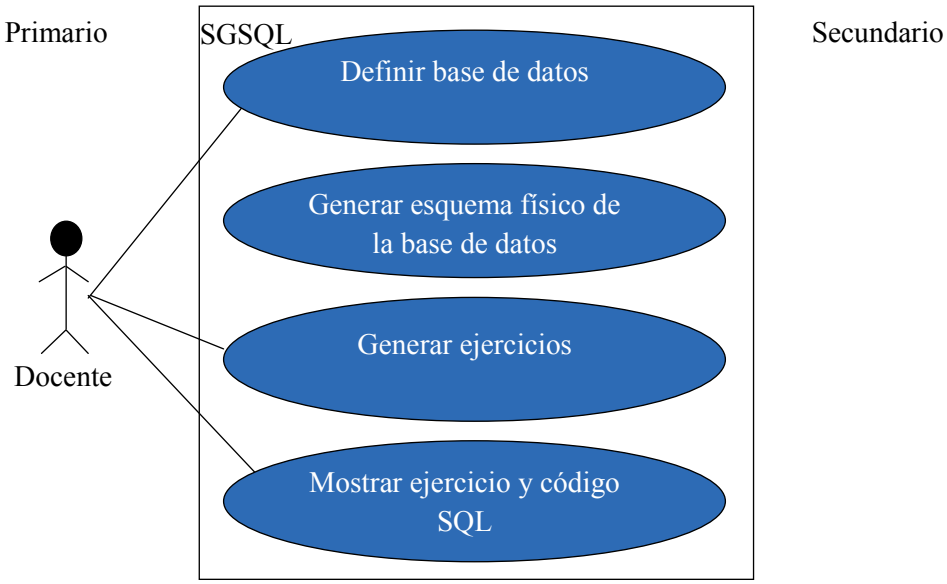


Diagrama de casos de uso

2.2 Implementación

El sistema fue desarrollado, en la plataforma java, con la IDE NetBeans.

2.2.1 API´s (Java Platform SE 7)

Para el desarrollo de la interfaz se utilizo la API (Application Programming Interfaces) de java

Tipos de lenguaje de programación Java Interfaces de programación de aplicaciones (API) :

- el funcionario núcleo Java API, contenido en el JDK o JRE , de una de las ediciones de la plataforma Java.
- API oficial la especificación de estas API se definen de acuerdo con un Java Specification Request (JSR).
- API no oficial, desarrollado por terceros, pero no está relacionado con ningún JSRs.

Lista parcial de las Interfaces de programación de aplicaciones (API) para el lenguaje de programación Java .

Oficial APIs

Nombre	Siglas	Descripción e Historia de las versiones
Java Advanced Imaging	JAI	Un conjunto de interfaces que soportan un alto nivel de modelo de programación que permite manipular las imágenes con facilidad.
Java Data Objects	JDO	Una especificación de objeto Java persistencia .
JavaHelp		A todas las funciones, sistema de ayuda ampliable que le permite incorporar la ayuda en línea en sus applets, componentes, aplicaciones, sistemas operativos y dispositivos.
Java Media Framework	JMF	Una API que permite a audio, vídeo y otros medios basados en el tiempo que se añade a las aplicaciones Java y applets.
Java Naming and Directory Interface	JNDI	Una API para servicios de directorio.
Java Speech API	JSAPI	Esta API permite la síntesis de voz y reconocimiento de voz.

Java 3D	J3D	Un escenario gráfico basado 3D API.
Java OpenGL	JOGL	Una envoltura biblioteca OpenGL .
Java Mail	(Ninguno)	Un marco para crear aplicaciones de correo y mensajería
Java USB para Windows	(Ninguno)	Una comunicación USB de aplicaciones Java

APIs (parte de la descarga estándar)

Nombre	Siglas	Java paquete (s) que contienen el API
Java Message Service	JMS	
JavaServer Faces	JSF	javax.faces

Opcional APIs (descarga por separado)

Nombre	Siglas
Java API para XML basado en RPC	JAX-RPC
XQuery API para Java	XQJ

Java Platform, Micro Edition (Java ME)

Nombre	Siglas
Configuración Limitada de Dispositivos Conectados	CLDC

2.2.2 Struts 2:

Struts 2 fue utilizado como framework. Struts 2 es un framework para el desarrollo de aplicaciones web, al cual ayudo mucho en la implementación del sistema ya que con la ayuda del mismo la implementación fue flexible y sencilla.

Struts 2: Framework de presentación, dentro de las capas en las que se divide una aplicación en la arquitectura JEE, el cual implementa el controlador del patrón de diseño MVC (Modelo Vista Controlador) proporciona algunos componentes para la capa de vista y también proporciona una integración perfecta con otros frameworks para implementar la capa del modelo (como Hibernate).

Para hacer más fácil presentar datos dinámicos, el framework incluye una biblioteca de etiquetas web. Las etiquetas interactúan con las validaciones y las características de internacionalización del framework, para asegurar que las entradas son válidas, y las salidas están localizadas. La biblioteca de etiquetas puede ser usada con JSP, FreeMarker, o Velocity; también pueden ser usadas otras bibliotecas de etiquetas como JSTL y soporta el uso de componentes JSF.

Permite agregarle funcionalidades, mediante el uso de plugins, de forma transparente, ya que los plugins no tienen que ser declarados ni configurados de ninguna forma. Basta con agregar al classpath el jar que contiene al plugin, y eso es todo.
Struts 2 cuenta con características que permiten reducir la configuración gracias a que proporciona un conjunto inteligente de valores por default. Además hace uso de anotaciones y proporciona una forma de hacer la configuración de manera automática.

Componentes de Struts 2

El corazón de Struts 2 es un filtro, conocido como el "FilterDispatcher". Este es el punto de entrada del framework. A partir de él se lanza la ejecución de todas las peticiones que involucran al framework.

Las principales responsabilidades del "FilterDispatcher" son:

- Ejecutar los Actions, que son los manejadores de las peticiones.
- Comenzar la ejecución de la cadena de interceptores.
- Limpiar el "ActionContext", para evitar fugas de memoria.

Struts 2 procesa las peticiones usando tres elementos principales:

- Interceptores
- Acciones
- Resultados

Interceptores

Los interceptores son clases que siguen el patrón interceptor. Estos permiten que se implementen funcionalidades cruzadas o comunes para todos los Actions, pero que se ejecuten fuera del Action. Los interceptores realizan tareas antes y después de la ejecución de un Action y también pueden evitar que un Action se ejecute (por ejemplo si estamos haciendo alguna validación que no se ha cumplido).

Acciones

Las acciones o Actions son clases encargadas de realizar la lógica para servir una petición. Cada URL es mapeada a una acción específica, la cual proporciona la lógica necesaria para servir a cada petición hecha por el usuario.

Results

Después que un Action ha sido procesado se debe enviar la respuesta de regreso al usuario, esto se realiza usando results. Este proceso tiene dos componentes, el tipo del result y el result mismo.
El tipo del result indica cómo debe ser tratado el resultado que se le regresará al cliente. Un Action puede tener más de un result asociado. Esto nos permitirá enviar al usuario a una vista distinta dependiendo del resultado de la ejecución del Action. Por ejemplo en caso de que todo salga bien, enviaremos al usuario al result "sucess", si algo sale mal lo enviaremos al result "error", o si no tiene permisos lo enviaremos al result "denied".

Funcionamiento de Struts 2

Los pasos que sigue una petición son:

1 - El navegador web hace una petición para un recurso de la aplicación (index.action, reporte.pdf, etc.). El filtro deStruts (al cual llamamos FilterDispatcher aunque esto no es del todo correcto, retomaré esto un poco más adelante) revisa la petición y determina el Action apropiado para servirla.
2 - Se aplican los interceptores, los cuales realizan algunas funciones como validaciones, flujos de trabajo, manejo de la subida de archivos, etc.

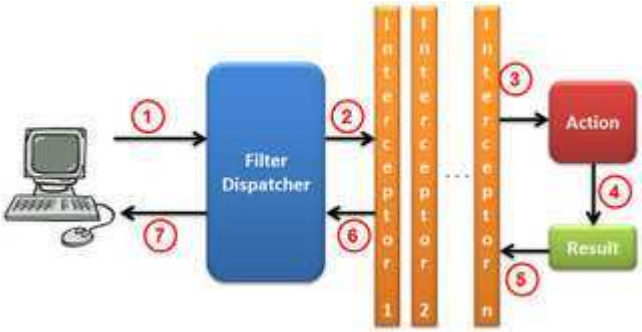
3 - Se ejecuta el método adecuado del Action (por default el método "execute"), este método usualmente almacena y/o regresa alguna información referente al proceso.

4 - El Action indica cuál result debe ser aplicado. El result genera la salida apropiada dependiendo del resultado del proceso.

5 - Se aplican al resultado los mismos interceptores que se aplicaron a la petición, pero en orden inverso.

6 - El resultado vuelve a pasar por el FilterDispatcher aunque este ya no hace ningún proceso sobre el resultado (por definición de la especificación de Servlets, si una petición pasa por un filtro, su respuesta asociada pasa también por el mismo filtro).

7 - El resultado es enviado al usuario y este lo visualiza.



2.2.3 Hibernate

Se utilizo Hibernate para el desarrollo del sistema, que es una una herramienta de mapeo de objeto-relacional (ORM) para la plataforma java, ayuda a el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, esto mediante archivos declarativos (XML) o anotaciones de beans en las entidades que permiten establecer relaciones. Ayudo en problemas de diferencias entre los modelos de datos coexistentes en la aplicación, como: uso de las bases de datos. Fue de gran ayuda en el desarrollo del sistema ya que permite de manera automática manipular los datos de las bases de datos y todas sus características, Hibernate permitió a la aplicación interactuara con tipos de datos de java y SQL. Tambien fue utilizado el lenguaje HQL (*Hibernate Query Languaje*) se utilizo para realizar consultas programáticamente.

2.2.4 Apache Tomcat

Apache Tomcat fue utilizado como servidor HTTP y contenedor de servlets. Fue de gran ayuda por ser un servidor web con soporte de servlets y JSPs, al realizar el desarrollo en java y utilizar Tomcat como servidor todo el sistema fue compatible sin ningún inconveniente y la programación fue de fácil intuición, cubriendo con todos los requerimientos necesarios para llevar a cabo el desarrollo del sistema con características de aplicación web segura y persistente.

ApacheTomcat: Es un código abierto de servidor web desarrollado por laApache Software Foundation (ASF). Tomcat implementa el servlet de Java y las JavaServer Pages (JSP) de Oracle Corporation , y proporciona HTTP web server.

Componentes:

Catalina

Es el nombre del contenedor de servlets del Jakarta Tomcat desde la versión 4x. Fue desarrollado bajo el Proyecto Jakarta de la Apache Software Foundation. Tomcat implementa las especificaciones de Sun Microsystems para servlets y Java Server Pages (JSP), las cuales son importantes tecnologías web basadas en Java.

El contenedor de servlets de Tomcat fue rediseñado como Catalina en la versión 4x de Tomcat. El arquitecto de Catalina fue Craig McClanahan

Coyote

Coyote es un componente Tomcat HTTP conector que soporta el protocolo HTTP 1.1 para el servidor web o contenedor de aplicaciones. Coyote escucha las conexiones entrantes en un determinado TCP puerto en el servidor y envía la solicitud al motor Tomcat para procesar la solicitud y enviar una respuesta al cliente solicitante.

Jasper

Jasper es el motor de Tomcat JSP. Tomcat 5.x utiliza Jasper 2, que es una implementación de la Sun Microsystems s JavaServer Pages 2.0. Jasper analiza los archivos JSP para compilarlos en código Java como servlets (que puede ser manejado por Catalina). En tiempo de ejecución, Jasper detecta cambios en los archivos JSP y recompila ellos.

De Jasper Jasper a 2, se añadieron características importantes:

- JSP Tag biblioteca pooling - Cada etiqueta de marcado en el archivo JSP está a cargo de una clase de controlador de etiqueta. Tag objetos de clase de controlador se unen y se reutiliza en el conjunto de servlets JSP.
- Antecedentes compilación JSP - Mientras que recompilar el código modificado JSP Java, la versión anterior todavía está disponible para solicitudes del servidor. Cuanto

más viejo servlet JSP se elimina una vez que el nuevo servlet JSP ha terminado de ser recompilado.

- Recompile JSP cuando se incluyen cambios de la página - Las páginas se pueden insertar e incluido en una JSP en tiempo de ejecución. El JSP no sólo se volverá a compilar con cambios en los archivos JSP, sino también con los cambios de página incluidos.
- JDT Java compiler - Jasper 2 puede utilizar el JDT Eclipse (Java Development Tools).

Cluster

Este componente ha sido añadido a administrar grandes aplicaciones. Se utiliza para el balanceo de carga.

Alta disponibilidad

En tomcat, característica de alta disponibilidad se añade que es útil para la programación planificada de antemano el rendimiento del sistema a la carta base por usuario. Característica Tomcat apoya a responder a las peticiones de usuario para manejar las operaciones de las aplicaciones web.

Aplicaciones Web

Asimismo, ha añadido el usuario, así como el sistema basado en la mejora de aplicaciones web para añadir soporte para el despliegue en toda la variedad de environments. También trata de gestionar la sesión, así como aplicaciones en toda la red.

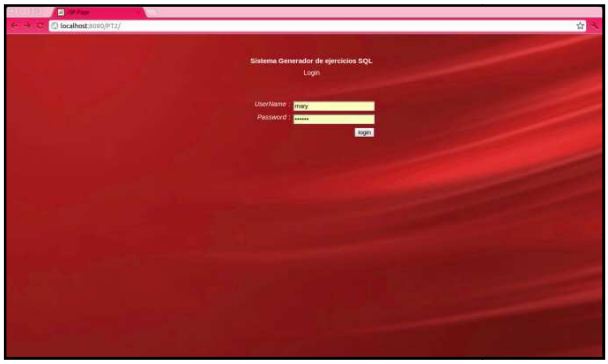
2.3 Pruebas o evaluaciones del sistema

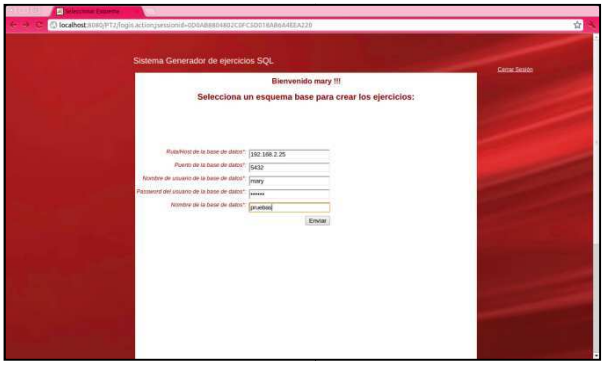
Objetivo 1: Interfaz web
Diseñar e implementar el sistema, interfaz web y base de datos.

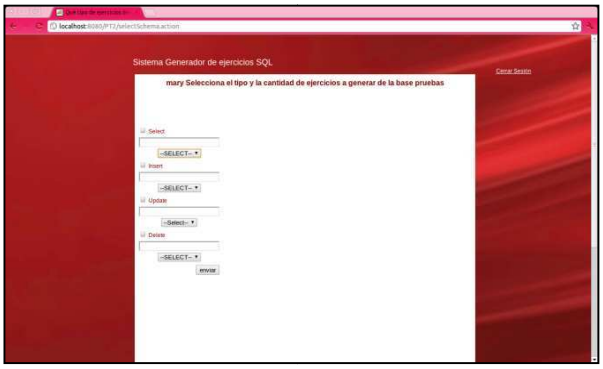
Nombre: Interfaz web

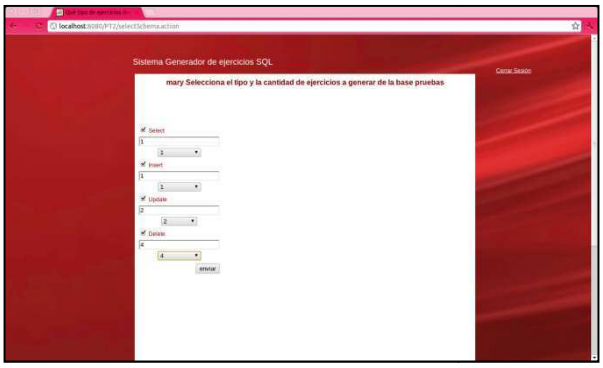
Procedimiento: Se muestra el sistema con interfaz web.

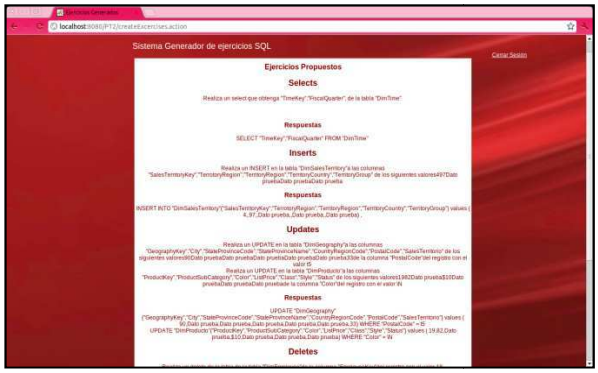
Resultado:

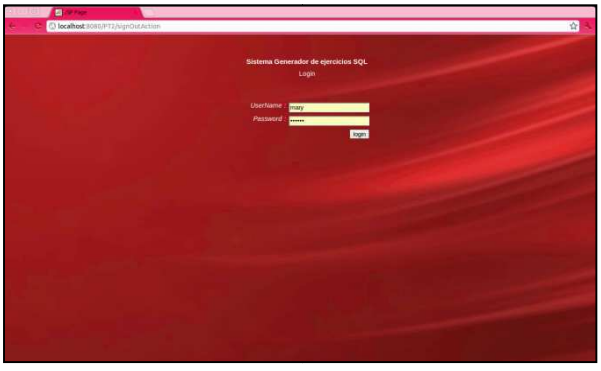

Sistema Generador de ejercicios SQL
Login
UserName :
Password :
login


Sistema Generador de ejercicios SQL
Cerrar Sesión
Bienvenido mary !!!
Selecciona un esquema base para crear los ejercicios:
Envíar


Sistema Generador de ejercicios SQL
Cerrar Sesión
mary Selecciona el tipo y la cantidad de ejercicios a generar de la base pruebas
Select
Insert
Update
Delete
envíar


Sistema Generador de ejercicios SQL
Cerrar Sesión
mary Selecciona el tipo y la cantidad de ejercicios a generar de la base pruebas
Select
Insert
Update
Delete
envíar


Sistema Generador de ejercicios SQL
Cerrar Sesión
Ejercicios Propuestos
Selects
Respuestas
Inserts
Respuestas
Updates
Respuestas
Deletes


Sistema Generador de ejercicios SQL
Login
UserName :
Password :
login

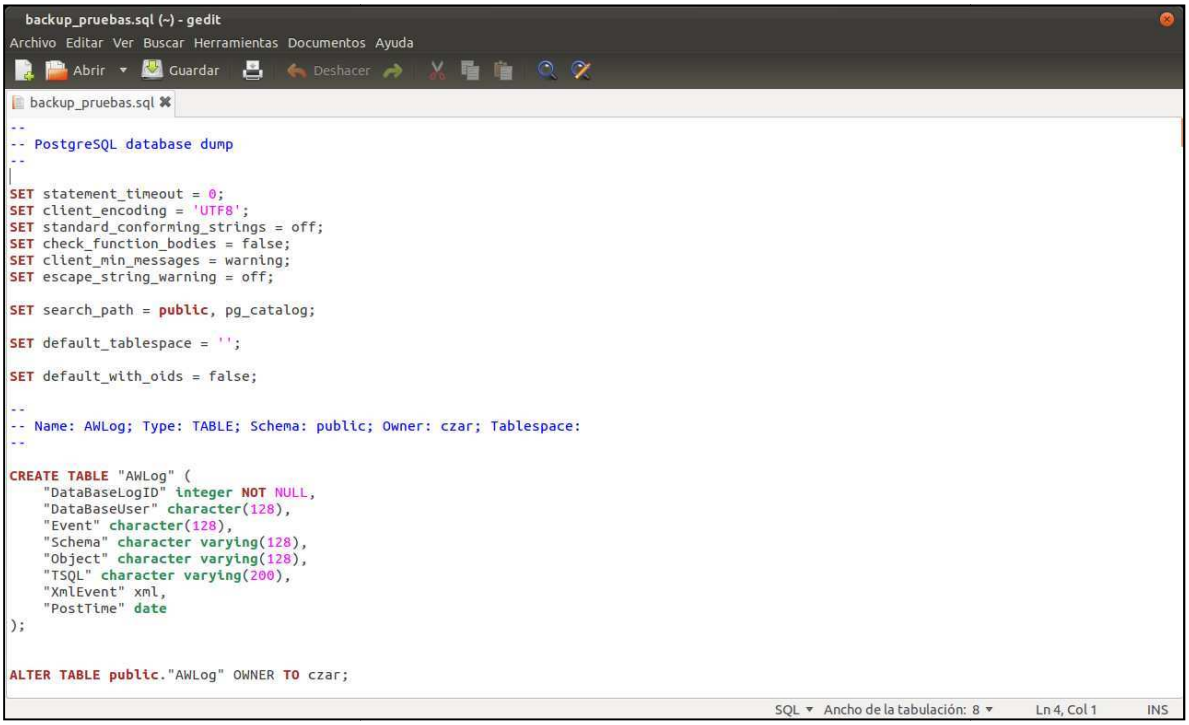
Objetivo 2: Generar el esquema físico de la base de datos.

Nombre: Generar el esquema físico de una base de datos indicada.

Procedimiento: El sistema obtiene el esquema físico de cualquier base de datos realizada con el manejador PostgreSQL con el comando pg_dump y la guarda en un archivo de texto, con un nombre relacionado al nombre de la base de datos.

Resultado:

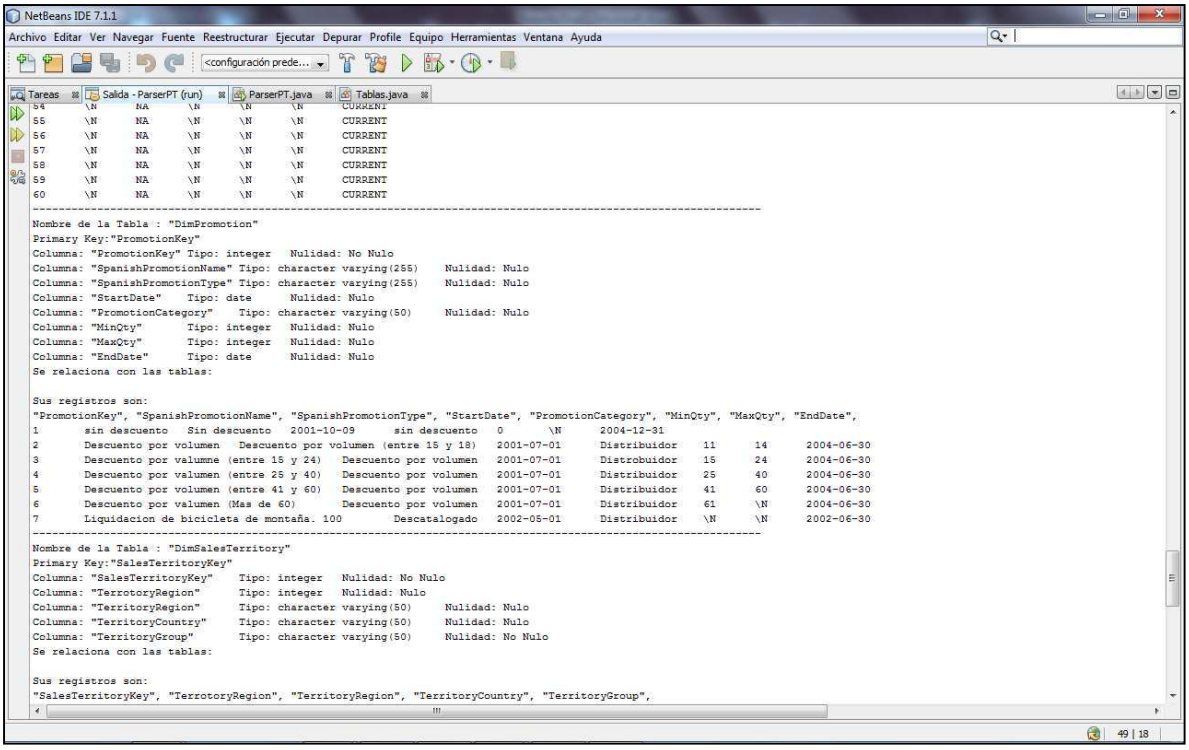
backup_pruebas.sql



Objetivo 3: Analizar las relaciones y la estructura en una base de datos indicada.

Nombre: Analizar el esquema físico de la base de datos.

Procedimiento: El sistema genera el esquema físico de cualquier base de datos realizada con un manejador PostgreSQL, se realiza en un archivo .txt y se analiza con un parser.

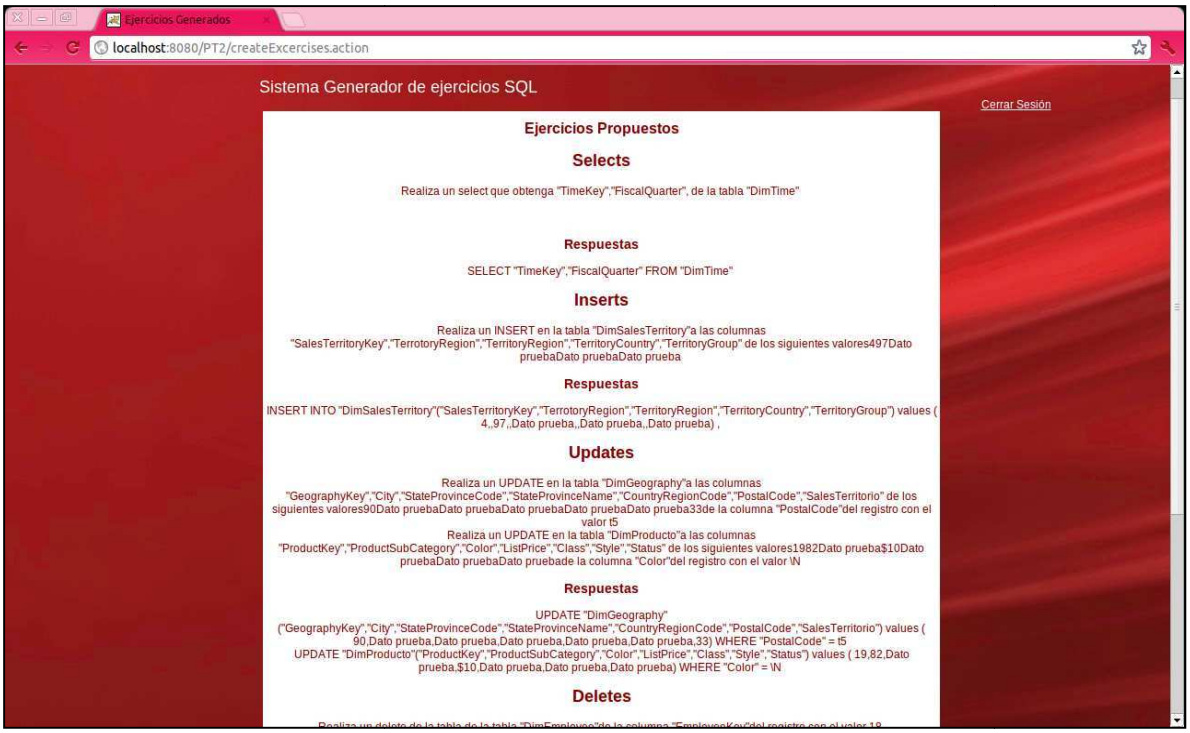


Objetivo 4: Diseñar e implementar el módulo para generar ejercicios a partir de un código SQL.

Nombre: Obtener ejercicios y su resultado (Código SQL) a partir de una base de datos indicada.

Procedimiento: Se diseño, programo e implemento el modulo para obtener ejercicios de sentencias SQL a partir de el esquema físico de una base de datos indicada y sus respuestas en código SQL.

Resultado:



Objetivo 5: Realizar pruebas de funcionalidad

Nombre: Realizar pruebas de funcionalidad

Resultado: Se cumplieron con todas las pruebas de funcionamiento, en diferentes escenarios.

3. Resultados

En el proyecto terminal se obtuvieron los siguientes resultados:

- Conexiones remotas de cualquier ip en el puerto 5432 del manejador PostgreSQL.
- Se realizo la conexión correctamente a todas las base de datos en las que se probo el sistema.
- Se obtuvo el esquema físico en todas las bases de datos que se probó el sistema.
- Se obtienen instrucciones de ejercicios en lenguaje natural en la base de datos de prueba.
- Se obtienen sentencias SQL de la base de datos en la base de datos de prueba.
- Funcionalidad del sistema en bases de datos locales (localhost).
- Funcionamiento del sistema en bases de datos remotas (Teniendo conexión a internet).
- El sistema hace validación de credenciales en el sistema y en las bases de datos.
- Validación de selección en las sentencias SQL.
- Variación en tablas, columnas, registros para los ejercicios y para el resultado (Codigo SQL).
- Con la configuración indicada en el manual de instalación es posible obtener la conexión en cualquier base de datos del manejador PostgreSQL de cualquier ip.

4. Trabajos futuros

La posible continuación del proyecto es extenderlo para varios manejadores (Informix, DB2, MySQL, Microsoft SQL Server, DB2, Firebird, SQLite, etc.).

5. Bibliografía

[1] J. F. Rojas Sánchez "Programa de migración de manejadores de bases de datos relacionales," Propuesta de proyecto terminal Ing. en Computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2009.

[2] L. O. Vázquez Mondragón, "Herramienta para crear exámenes con distintos formatos de pregunta/respuesta," Proyecto terminal Ing. en Computación, Dept. Sist, UAM-Azc., Ciudad de Méx., D.F., Méx., 2010.

[3] J. Mendoza Barrón "Desarrollo de reactivos y sus respectivas soluciones, para la elaboración de exámenes, a partir de una base de datos," Propuesta de proyecto terminal Ing. en Computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2010.

[4] I. Y. Olmos Aquino, "Sistema de gestión de calificaciones para los cursos impartidos por un profesor," Propuesta de proyecto terminal Ing. en computación, Dept. Sist, UAM-Azc., Ciudad de Méx., D.F., Méx., 2011.

[5] "Sistema evaluador de desarrollo de productos de software bajo la norma SPICE," Tesis Ing. en Sistemas Computacionales, Dept. Sist., IPN, Ciudad de Méx., D.F., Méx. 1999.

[6] "Suit ESCOM," Tesis Ing. en Sistemas Computacionales, Dept. Sist., IPN, Ciudad de Méx., D.F., Méx., 2001.

[7] (Agosto 2012). Tomcat [En línea]. Disponible : http://clean-9/install-tomcat-7-on-ubuntu/ .

[8] (Agosto 2012). Java. [En línea]. Disponible: http://www.javatutoriales.com/2011/06/struts-2-parte-1-configuracion.html .

[9] (Agosto 2012) Struts 2. [En línea]. Disponible: http://www.javatutoriales.com/2011/06/struts-2-parte-1-configuracion.html .

[10] (Agosto 2012) Tomcat. [En línea]. Disponible: http://clean-clouds.com/2011/10/19/install-tomcat-7-on-ubuntu/ .

[11] (Noviembre 2012) SQL [En línea]. Disponible: http://es.wikipedia.org/wiki/SQL .

[12] (Noviembre 2012) PostgrSQL. [En línea]. Disponible: http://manuales.gfc.edu.co/postgres/es/app-pgdump.html .

[13] (Noviembre 2012) APIs Java. [En línea]. Disponible: http://en.wikipedia.org/wiki/List_of_Java_APIs .

[14] (Noviembre 2012) Apache Tomcat. [En línea]. Disponible: http://en.wikipedia.org/wiki/Apache_Tomcat .