Universidad Autónoma Metropolitana Unidad Azcapotzalco División de Ciencias Básicas e Ingeniería

Reporte final del proyecto de integración

Licenciatura en ingeniería en Computación

Modalidad de Proyecto tecnológico

UML4TeX v2: Integración de diagramas de estado, de actividad y de paquetes en UML4TeX

Diego Vázquez Alvarez

210203597

Dra. Beatriz Adriana González Beltrán

Profesor asociado

bgonzalez@azc.uam.mx

Departamento de Sistemas

División de Ciencias Básicas e Ingeniería

Yo, Beatriz Adriana González Beltrán, *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.

Firma de la asesora

Yo, Diego Vázquez Alvarez, 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.

Firma del alumno

RESUMEN

Este proyecto presenta la nueva versión de UML4TeX, aplicación desarrollada en Java que nace como consecuencia de la necesidad de desarrollar un editor gráfico de diagramas UML.

En su primera versión UML4TeX era capaz de editar y generar tres tipos de diagramas UML como son los diagramas de clases, de casos de uso y diagramas de secuencia, a su vez la aplicación permite traducir estos diagramas a macros en PSTricks e incluir estos macros en documentos LaTeX.

Con esta nueva versión de UML4TeX se mantiene completamente la funcionalidad de su primera versión añadiendo esta vez tres nuevos tipos de diagramas editables, los cuales incluyen los diagramas de actividad, los diagramas de estado, los diagramas de paquetes UML.

Tanto en la primera como esta segunda versión de UML4TeX la aplicación fue construida utilizando el Proceso Rational Unificado (RUP). Así mismo la aplicación continúa utilizando el Modelo Vista-Controlador para su diseño y el lenguaje de programación Java, dichas características le permiten a la aplicación tener un desarrollo modular, escalable y además ser una aplicación multiplataforma.

Esta versión de UML4TeX incluye finalmente seis tipos de diagramas UML, tres de los cuales son completamente nuevos dentro de la aplicación y como ya se mencionó estos son los diagramas de actividad, de estados y de paquetes, además de que tal como sucedió con este proyecto pueden seguirse desarrollando otros diagramas en el futuro.

TABLA DE CONTENIDO

INDICE DE FIGURAS Y TABLAS

INDICE DE DIAGRAMAS

INDICE DE APÉNDICE B

1. Introducción

Unified Modeling Langage (UML) significa Lenguaje Unificado de Modelado. Este lenguaje se utiliza principalmente en la construcción y especificación de un sistema de manera conceptual a través de diversos diagramas con los cuales se pueden visualizar diversas propiedades de lo que en ellos se representa [1].

Dentro de los diagramas que este lenguaje permite representar, están los diagramas de estado, diagramas de actividades y diagrama de paquetes.

Los diagramas de estado son utilizados para representar la vida de un objeto dentro de un sistema; es decir, muestran la manera en que un objeto cambia de un estado a otro, así como las acciones que ocasionan que este objeto cambie de estado a lo largo de su "vida" [1].

Los diagramas de actividad son usados para definir las acciones que se llevan a cabo dentro de determinado sistema, describen la funcionalidad del sistema en cuestión. En estos diagramas se plantea ¿qué acciones llevará a cabo el sistema?, ¿cómo, cuándo, dónde, por qué y por quién son ejecutadas las acciones del sistema? [1].

Los diagramas de paquetes permiten agrupar de manera lógica los elementos de un sistema, así como mostrar las dependencias que dichas agrupaciones tienen entre sí, lo que permite tener un panorama jerárquico del sistema [1].

Hoy en día hay distintos paquetes de software como Rational Software Architect de IBM [2], Umbrello [3] o UML4TeX [4] que permiten la creación y manipulación de distintos tipos de diagramas UML.

UML4TeX es una herramienta desarrollada en Java que permite crear, editar y almacenar diagramas UML y a su vez exportar dichos diagramas a distintos formatos como lo es XML o PSTricks para así poder integrar dichos diagramas en documentos de LaTeX.

En la actualidad UML4TeX es capaz de manejar dos tipos de diagramas UML como son, los diagramas de clases, diagramas de casos de uso.

Es por eso que, en este proyecto se realizará una expansión a la funcionalidad de UML4TeX, la cual permitirá el manejo de diagramas de estado, de actividad y de paquetes dentro de la aplicación.

2. Antecedentes

2.1 Referencias internas

2.1.1 Proyectos de integración o terminales

Editor gráfico de diagramas UML con la capacidad de generar macros PSTricks

Este proyecto es un editor gráfico de diagramas UML desarrollado en Java con la capacidad crear diagramas de casos de uso y diagramas de clases, además es capaz de traducir dichos diagramas a macros de PSTricks. Dado que está desarrollado en Java tiene la cualidad de ser multiplataforma [4].

Es sobre este editor gráfico que se va a realizar una expansión de su funcionalidad durante el proyecto de integración, para que sea capaz de editar diagramas de estado, de actividad y de paquetes.

Dibujo con Touch Screen transmitido por "PC" a un dispositivo con láser

Este proyecto es capaz de dibujar figuras por medio de un láser, dichas figuras son previamente diseñadas mediante una pantalla táctil.

Dentro de los módulos que componen este proyecto se encuentra un módulo correspondiente al software que permite dibujar las figuras [5].

Se ha elegido este proyecto como antecedente, ya que el módulo correspondiente al editor de imágenes tiene básicamente las mismas funcionalidades que tiene UML4TeX, las cuales son agregar o quitar componentes de un gráfico.

A diferencia de UML4TeX cuya principal función es la manipulación de diagramas UML, este dibujador láser solo se enfoca en el trazado de figuras geométricas.

Implementación de una interfaz gráfica interactiva para algoritmos de ordenamiento y estructuras tipo árbol

Este proyecto implementa una interfaz gráfica donde se pueden modelar y visualizar estructuras de tipo árbol, teniendo la opción de agregar o quitar elementos de la estructura, así como seleccionar entre varios tipos de estructuras, como árboles roji-negros, AVL, árboles binarios y árboles 2, 3,4 [6].

La relación que guarda con UML4TeX es que maneja varios tipos de diagramas (estructuras de árbol) por medio de una interfaz gráfica.

2.2 Referencias externas

2.2.1 Software

Umbrello [3]**.** Es un editor gráfico de diagramas UML el cual está desarrollado bajo la tecnología KDE. Umbrello permite actualmente crear nueve tipos de diagramas UML, así como generar diagramas a partir de código en determinados lenguajes como C++ o Java. A su vez permite también la exportación de los diagramas creados a distintos formatos como son: DocBook y XHTML.

La diferencia con UML4TeX es que Umbrello al estar desarrollado bajo la tecnología KDE se reduce su portabilidad, además de que Umbrello no incluye un formato de exportación compatible con LaTeX.

MacA&D [7]. Este software permite la creación de varios tipos de diagramas UML e incluye funciones para la generación de código en distintos lenguajes como C++, Java y PHP.Esta herramienta está específicamente orientada a ambientes de desarrollo Mac lo cual limita su portabilidad y ejecución en otras plataformas.

Uno de los puntos destacables de UML4TeX es que al estar desarrollado completamente en Java le brinda la posibilidad a la aplicación de ser multiplataforma, pudiendo ejecutarse prácticamente en cualquier sistema operativo.

2.2.2 Artículos

UML-to-Java transformation in IBM Rational Software Architect editions and related software

En este artículo se habla del software "Rational Software Architect" el cual permite la creación de distintos diagramas UML. El artículo se enfoca e ilustra la capacidad que tiene Rational Software Architect de generar a partir de un diagrama UML su correspondiente código en lenguaje Java [8].

Una de las principales desventajas de Rational Software Architect es que al ser propiedad de IBM se requiere la adquisición de una licencia para poder utilizarlo.

3. Justificación

Como ya se ha mencionado anteriormente, existen actualmente diversos editores gráficos de diagramas UML. Sin embargo, su funcionalidad en muchas ocasiones se ve limitada debido principalmente a la portabilidad de dichas aplicaciones, ya que muchas de estas herramientas generalmente son desarrolladas para ejecutarse dentro de plataformas muy específicas o no cuentan con un soporte integral para el desarrollo de diagramas UML.

En este proyecto se dará continuidad al desarrollo de UML4TeX, extendiendo su funcionalidad. Dicha extensión es posible debido a que UML4TeX está desarrollada en Java bajo el patrón de arquitectura de software Modelo-Vista-Controlador. El primer

aspecto permite a la aplicación tener la capacidad de ser portable (ser multiplataforma) y el segundo aspecto permite ser extensible.

4. Objetivos

4.1 Objetivo general

Extender la funcionalidad de UML4TeX permitiendo la edición de los diagramas de estado, diagramas de actividades y diagramas de paquetes UML.

4.2 Objetivos específicos

- Integrar los diagramas de estado, de actividad y de paquetes UML en UML4TeX dentro del módulo que permite la representación lógica de los diagramas.
- Integrar los diagramas de estado, de actividad y de paquetes UML en UML4TeX dentro del módulo que permite la visualización de los diagramas.
- Integrar los diagramas de estado, de actividad y de paquetes UML en UML4TeX dentro del módulo que permite la manipulación gráfica (edición) de los diagramas.
- Integrar elementos gráficos correspondientes a cada uno de los diagramas: de estado, de actividad y de paquetes dentro del módulo de la Interfaz Gráfica de Usuario de UML4TeX.

5. Marco teórico

El presente reporte corresponde a los resultados obtenidos al realizar una extensión de un proyecto que ya existía con anterioridad, Dicho proyecto lleva por nombre: “Editor gráfico de diagramas UML con la capacidad de generar macros PSTricks” la aplicación que deriva de la extensión de este proyecto a su vez lleva el nombre de: “UML4TeX v2”.

5.1 UML

Conocido comúnmente como Lenguaje Unificado de Modelado o por sus siglas en inglés UML (Unified Modeling Language). Este leguaje es utilizado principalmente en el desarrollo de un sistema para efectos de la conceptualización, construcción y especificación del mismo.
Todo esto a través de los 13 distintos diagramas con los que cuenta dicho lenguaje y los cuales pone a disposición del usuario para ayudarlo a modelar y especificar las distintas funcionalidades con las que finalmente contará el sistema que se desee implementar [1].

5.2 UML4TeX

Es un editor gráfico de diagramas UML, su desarrollo nace a partir de la necesidad de contar con una herramienta de modelado UML que así mismo permitiera traducir los diagramas generados en dicha aplicación a macros de PSTricks para poder integrarlos posteriormente en documentos de LaTeX o exportarlos como imagen con formato JPEG o PNG [4].

UML4TeX v2 cuenta con un total de 6 diagramas editables los cuales son los diagramas de clases, diagramas de secuencia, diagramas de casos de uso, diagramas de actividad, diagramas de estado y diagramas de paquetes.
Los diagramas agregados en esta segunda versión y sobre los cuales se enfoca el presente reporte, fueron los diagramas de actividad, de estado y de paquetes.

A continuación, se explica detalladamente el propósito de cada uno de estos diagramas, así como la funcionalidad de cada uno de sus elementos

5.3 Diagrama de estado

Estos diagramas se emplean como un medio para explicar el comportamiento que puede tener un objeto dentro de un sistema, ya que estos diagramas están hechos para ilustrar los posibles "estados" por los que puede pasar una instancia y de qué manera se modifica dicha instancia debido a los eventos que inciden sobre ésta.
En el diagrama de estados se pueden identificar principalmente cuatro tipos de componentes: nodo inicial, estados, transiciones y un nodo final.
Además de los componentes mencionados anteriormente, un diagrama de estado puede contener también otros tipos de nodos además de los nodos iniciales, finales y los estados, como son: nodos de objetos, nodos de decisión, nodos de mezcla, nodos de bifurcación (decisión, merge, join y fork)
Fuera del contexto de UML un diagrama de estado puede ser fácilmente comparado con lo que en el ámbito computacional se conoce como "autómata finito", ya que tanto el autómata finito como el diagrama de estado en algún punto de su progresión alcanzan un estado final de aceptación y terminan [9].

5.3.1 Nodo inicial

Este nodo sirve para indicar en que momento inicia la ejecución de la transacción o transacciones representadas en un diagrama de estado [10].

Figura 1 Representación de un Nodo de estado inicial en UML4TeX.

5.3.2 Estado

Es la unidad básica de la que se compone un diagrama de estado y que representa el momento en el que se encuentra una transacción.
Para efectos de este proyecto se implementaron dentro del diagrama de estado, dos posibles representaciones de un estado, una representación simple y una representación compuesta [10].

5.3.3 Estado simple

Contiene una descripción simple de la situación que representa dicho estado, sin especificar bajo qué condiciones se puede acceder o salir de dicho estado [10].

Figura 2. Representación de un estado simple en UML4TeX.

5.3.4 Estado compuesto

Esta representación de un estado está constituido por dos secciones, la primera sección es un compartimiento para la asignación de un nombre al estado y la segunda sección contiene la descripción de cómo se accede o se sale de dicho estado, es decir, las acciones y actividades que se llevan a cabo dentro del estado [10].

Figura 3. Representación de un estado compuesto en UML4TeX.

5.3.4.1 Acciones y actividades

Los componentes que conforman la caja de acciones y actividades del estado compuesto se dividen de la siguiente manera [9].

ACCIONES Y ACTIVIDADES	SIGNIFICADO
Entry	Acción que se realiza cuando se entra al estado
Exit	Acción que se realiza cuando se sale del estado-
Do	Actividad que se realiza mientras se está dentro del estado.
Otro	Puede referirse a cualquier otro evento dentro del estado y que produce un efecto o acción.

Tabla 1. Descripción de las acciones y actividades que conforman un estado compuesto.

Dentro de UML4TeX los componentes de la caja de acciones y actividades de un estado compuesto deben cumplir cierto formato para poder ser incluidos dentro de esta sección del estado compuesto. El formato para dichos componentes es el siguiente:

1. Visibilidad del componente: la cual está determinada de la siguiente manera.

   - + Publica
   - # Protegida
   - - Privada
   - ~ Paquete
   - No mostrar: Este tipo de visibilidad permite no mostrar el componente de la caja de acciones y actividades a la que se le aplique dicha visibilidad.

2. Modificadores de comportamiento: Esta parte del formato permite decidir si se quiere que el comportamiento en cuestión sea un comportamiento estático o no.

3. Descripción del comportamiento: Esta sección del formato del componente está divida a su vez en:

   - Tipo de comportamiento: Los cuales ya hemos descrito anteriormente en la Tabla1.
   - Nombre del comportamiento
   - Tipo de retorno del comportamiento, los cuales pueden ser:
     - String
     - int
     - Integer
     - char
     - Character
     - Object

- Float
- Double
- Date
- Otro

- Argumentos del componente

5.3.4.2 Evento

Motivo, causa o suceso que promueve el cambio entre un estado y otro [9].

5.3.4.3 Transiciones

Es el paso que se lleva a cabo al pasar de un estado a otro por causa de un evento, dichas transiciones se representan en el diagrama de estado por medio de flechas, las cuales conectan a un estado con el siguiente [9].

Figura 4a

Figura 4b

Figura 4. Tipos de Transiciones en un diagrama de estado en UML4TeX.

5.3.5 Flujo de objeto

El flujo de objeto (Figura 4a) se encarga de conectar a los nodos de tipo objeto con otros nodos tipo objeto y nodos de tipo estado, así como con el resto de nodos de control como son nodos Merge, Decision, Fork, Join, Flow Final, Estado inicial y estado final [10].

5.3.6 Flujo de control

El flujo de control (Figura 4b) se encarga de conectar los nodos de tipo estado con el resto de los nodos del diagrama como son: los nodos de control y los nodos de tipo objeto [10].

5.3.6.1 Multiplicidad

Cabe mencionar que las transiciones de tipo flujo de control y flujo de objeto cuentan con un cierto formato el cual los distingue uno de otro, este formato se distingue principalmente por la etiqueta que acompaña al conector y por las figuras desde las que pueden tener origen estos conectores.

Además, ambos conectores cuentan con etiquetas para indicar el tipo de multiplicidad con la que se asocian a cada nodo. Dicha multiplicidad se explica en la siguiente tabla [9].

MULTIPLICIDAD	SIGNIFICADO
1	Uno y solo uno (uno a uno).
0…1	Cero o uno.
X…Y	Desde X hasta Y(muchos a muchos)
*	Cero o varios
1…*	Uno o varios (uno a muchos)

Tabla 2. Multiplicidad de las transiciones dentro de UML4TeX.

5.3.7 Nodo de Estado final

Este nodo sirve para indicar en que momento termina la ejecución de las transacciones representadas en un diagrama de estado [10].

Figura 5. Representación de un Nodo de estado final en UML4TeX.

5.3.8 Otros nodos de un diagrama de estado UML4TeX

5.3.9 Fork

Este nodo es un nodo de control el cual sirve para dividir un flujo dentro del diagrama de estado en múltiples flujos que saldrán de este nodo *fork*. Es importante mencionar que este nodo únicamente contará con un flujo entrante pero más de un flujo saliente [10].

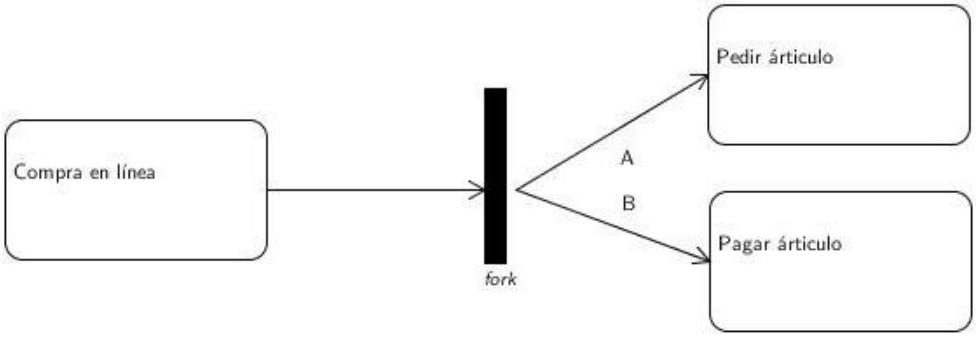


Figura 6. Representación de un Nodo Fork en UML4TeX

5.3.10 Join

El nodo Join sirve para unificar los flujos entrantes hacia este nodo en un único flujo. El nodo Join por tanto, puede tener múltiples flujos entrantes pero un solo flujo saliente [10].

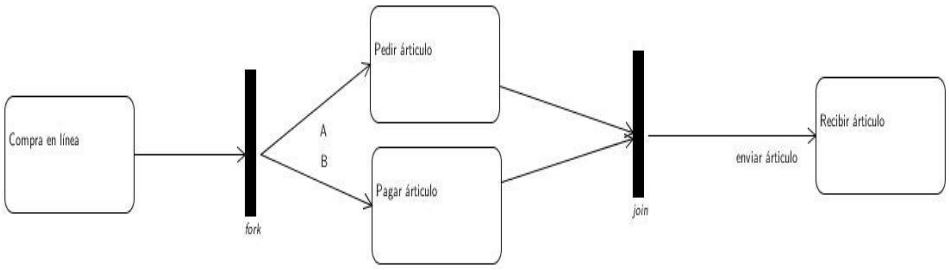


Figura 7. Representación de un Nodo Join en UML4TeX.

5.3.11 Decision Node

Este nodo es considerado también un nodo de control que como su nombre lo indica sirve para las operaciones relacionadas con la toma de decisiones. Dichas decisiones pueden ser tomadas por el usuario directamente o debido al cumplimiento o incumplimiento de una condición previamente establecida.
Un nodo de decisión recibirá un solo flujo de entrada y tendrá múltiples flujos de salida [10].

Figura 8. Representación de un Nodo Decision en UML4TeX.

5.3.12 Merge Node

Un nodo Merge tiene la función de recibir distintos flujos entrantes los cuales desembocan en una misma acción o estado. Es decir que el primer nodo que termine su ejecución activa automáticamente la ejecución del estado o acción saliente del nodo Merge [10].

Figura 9. Representación de un Nodo Merge en UML4TeX.

5.3.13 Flow Final Node

Este nodo sirve en un diagrama de estado UML para indicar el final de un flujo de dicho estado. El nodo de final de flujo solo indica la terminación del flujo al que se asocia dicho estado, permitiendo que el resto de los flujos del diagrama continúen en ejecución [10].

Figura 10. Representación de un Nodo Flow Final en UML4TeX.

5.3.14 Objetos

Los objetos dentro de un diagrama de estado suelen producirse como resultado de la ejecución de uno o varios estos dentro del diagrama. Es decir, los objetos representan instancias de los estados a los que se asocia dicho objeto [10].

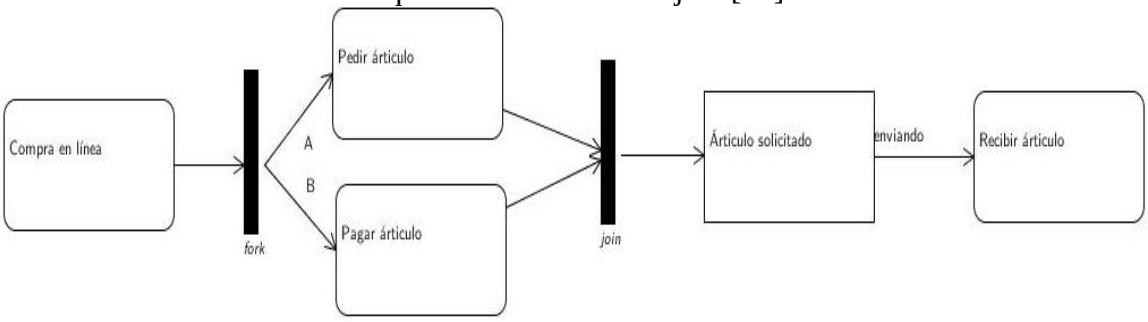


Figura 11. Representación de un Objeto en UML4TeX.

5.4 Diagrama de actividad

Los diagramas de actividad, representan un tipo de diagrama de estados en los que las transiciones entre una actividad y otra no se producen por eventos externos a dicha actividad. En este tipo de diagrama al igual que en el diagrama de estados, las actividades suceden secuencial o concurrentemente comenzando en un estado inicial y desplazándose de actividad en actividad hasta un único estado final.
Un diagrama de actividad consta principalmente de tres tipos de nodos, los nodos de acción y/o actividad, los nodos de control, y los nodos de objeto [9].

5.4.1 Action Node

Este nodo representa una acción la cual se lleva a cabo de manera atómica, por lo que dicha actividad es indivisible en otras actividades [10].

Figura 12. Representación de un nodo de actividad en UML4TeX.

5.4.2 Nodos de control

Los nodos de control en un diagrama de actividad, son aquellos nodos que permiten alterar el flujo de las actividades del diagrama en cuestión por medio de bifurcaciones o concurrencias hacia otros flujos.
Los nodos de control de un diagrama de actividad son los mismos nodos de control de un diagrama de estados, tales como son: nodos Fork y Join, nodos Merge y Decision y los nodos de estado inicial, final y de final de flujo.
Todos estos nodos cuentan con la misma representación gráfica, así como con la misma funcionalidad que tienen en un diagrama de estado [10].

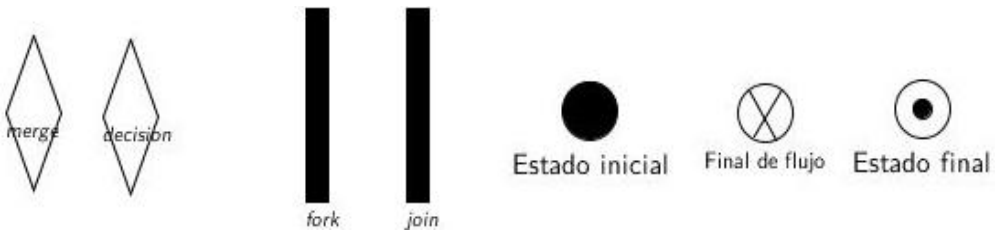


Figura 13. Representación de los nodos de control en un diagrama de actividad en UML4TeX.

5.4.3 Objetos

Los nodos de objeto representan instancias usadas y/o generadas por una o actividad. Los objetos tienen la misma representación y funcionalidad que en un diagrama de estado [10].

Figura 14. Representación de un objeto en un diagrama de actividad en UML4TeX.

5.4.4 Transiciones

Las transiciones en un diagrama de actividad representan el paso que se da entre un nodo de actividad y otro, pero a diferencia de las transiciones de un diagrama de estados en este caso las transiciones de un diagrama de actividad no se producen por eventos externos.
Las transiciones que un diagrama de actividad puede contener son como en un diagrama de estado, Control Flow y Object Flow (Figura 4 a y Figura 4 b), teniendo esta la misma funcionalidad y formato. Además el diagrama de actividad cuenta con un tercer tipo de conector, el cual representa una relación de excepción y el cual es representado por una flecha en "Z" [9].

5.4.4.1 Relación de excepción

Este tipo de transición sirve para indicar una excepción, es decir alguna acción o acontecimiento que debido a su ocurrencia cambie repentinamente el flujo normal del diagrama [10].

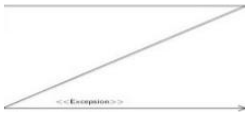


Figura 15. Representación de una Relación de excepción en UML4TeX.

5.5 Diagrama de paquetes

Los diagramas de paquetes son los diagramas UML que nos ayudan a representar la organización y agrupación de los elementos de un modelo en paquetes, así como las dependencias e importaciones y relaciones que existen entre los distintos elementos que conforman un diagrama de paquetes.
Estos diagramas también proporcionan una visualización de los espacios de nombres correspondientes.
Un diagrama de paquetes se conforma de nodos de paquetes, clases, interfaces de clase y las respectivas transiciones que se asocian a cada uno de los nodos que conforman el diagrama.

Para efectos de este proyecto se incluyen dos tipos de paquetes los cuales sirven para representar distintos tipos de vistas de los elementos que se agrupan dentro de dichos paquetes [9].

5.5.1 Class Package

Este tipo de paquete es la representación más simple que incluye el diagrama de paquetes debido a que únicamente incluye una etiqueta para descripción o espacio para un solo nombre [11].

Figura 16. Representación de un Class Package en UML4TeX.

5.5.2 Package Composite

Un paquete es un espacio de nombres, así como un elemento que puede estar contenido en los espacios de nombres de otros paquetes. Un paquete puede poseer o combinarse con otros paquetes y sus elementos se pueden importar en el espacio de nombres de un paquete [11].

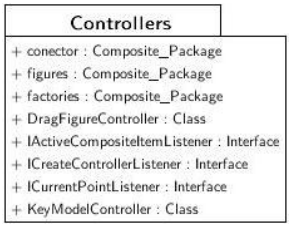


Figura 17. Representación de un Package Composite en UML4TeX.

6. Desarrollo del proyecto

En la sección 6.1 se describe la metodología empleada en el desarrollo del proyecto. En la sección 6.2 se explica el diseño del sistema y en la sección 6.3 se menciona el hardware y software necesario para el desarrollo del proyecto.

Cabe señalar que el apéndice A se presenta el manual de usuario y en el apéndice B se presenta el API de este proyecto.

6.1. Metodología empleada en el desarrollo del proyecto

El modelo utilizado en el desarrollo de la aplicación fue el Proceso Racional Unificado (Rational Unified Process en inglés, habitualmente resumido como RUP) [12] el cual es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML [1], constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
Cabe señalar que el RUP es un proceso iterativo. La aplicación se fue desarrollando en distintas iteraciones, concretamente, una iteración por cada uno de los distintos módulos sobre los cuales la aplicación está construida y posteriormente por cada uno de los diagramas UML que soporta la aplicación.

6.2 Diseño del sistema

Debido a que este proyecto es una extensión en la funcionalidad de un proyecto ya existente por lo que lo abordado en esta sección se enfocará principalmente en documentar como se realizó dicha extensión, más no como se construyó la aplicación desde cero.

En la sección 6.2.1 se describen los casos de uso.

6.2.1 Casos de uso

Tomando en cuenta la funcionalidad previa de UML4TeX la cual se basa principalmente en cinco casos de uso los cuales son: crear un diagrama, editar un diagrama, recuperar un diagrama, guardar un diagrama y exportar un diagrama, dichos casos de uso se pueden ver representados en el Diagrama 1.

Esta extensión estuvo enfocada en los dos primeros casos de uso los cuales son crear y editar un diagrama. En el Diagrama 2 se ilustra el diagrama de dichos casos de uso.

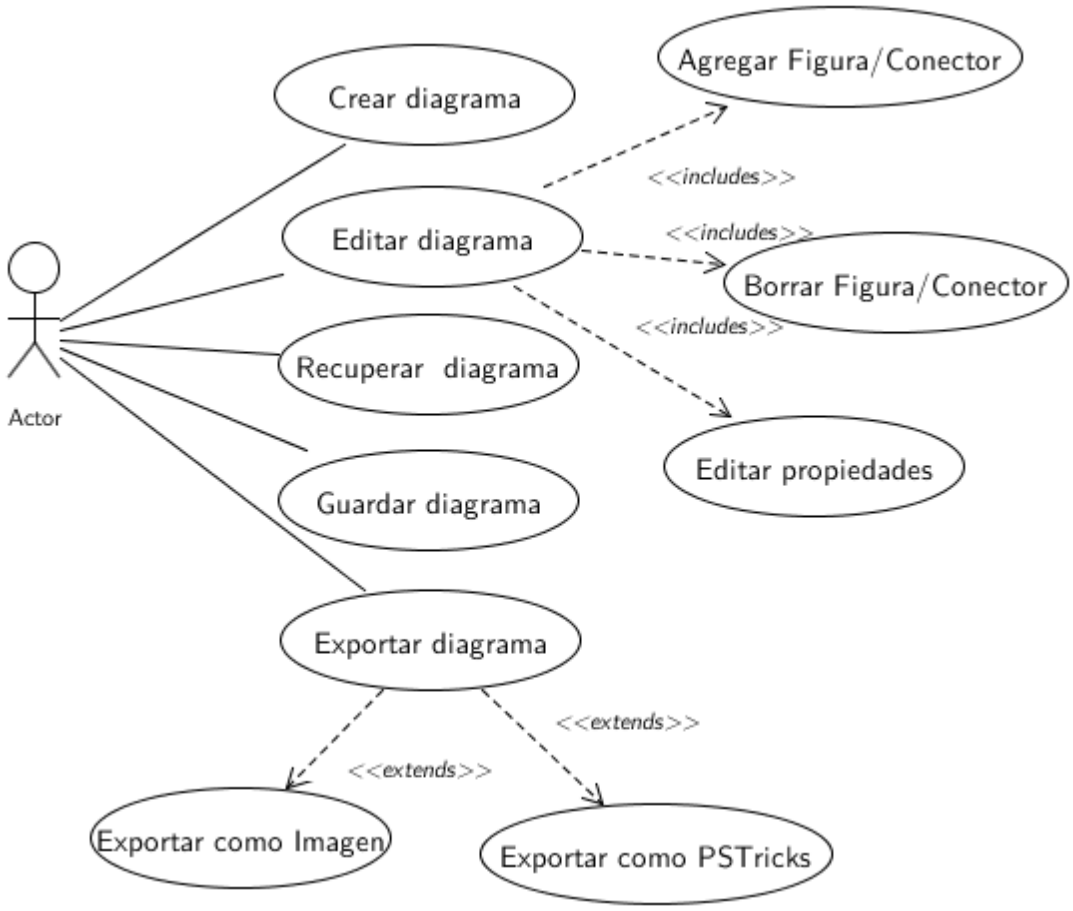


Diagrama 1. Diagrama de Casos de uso de la aplicación UML4TeX.

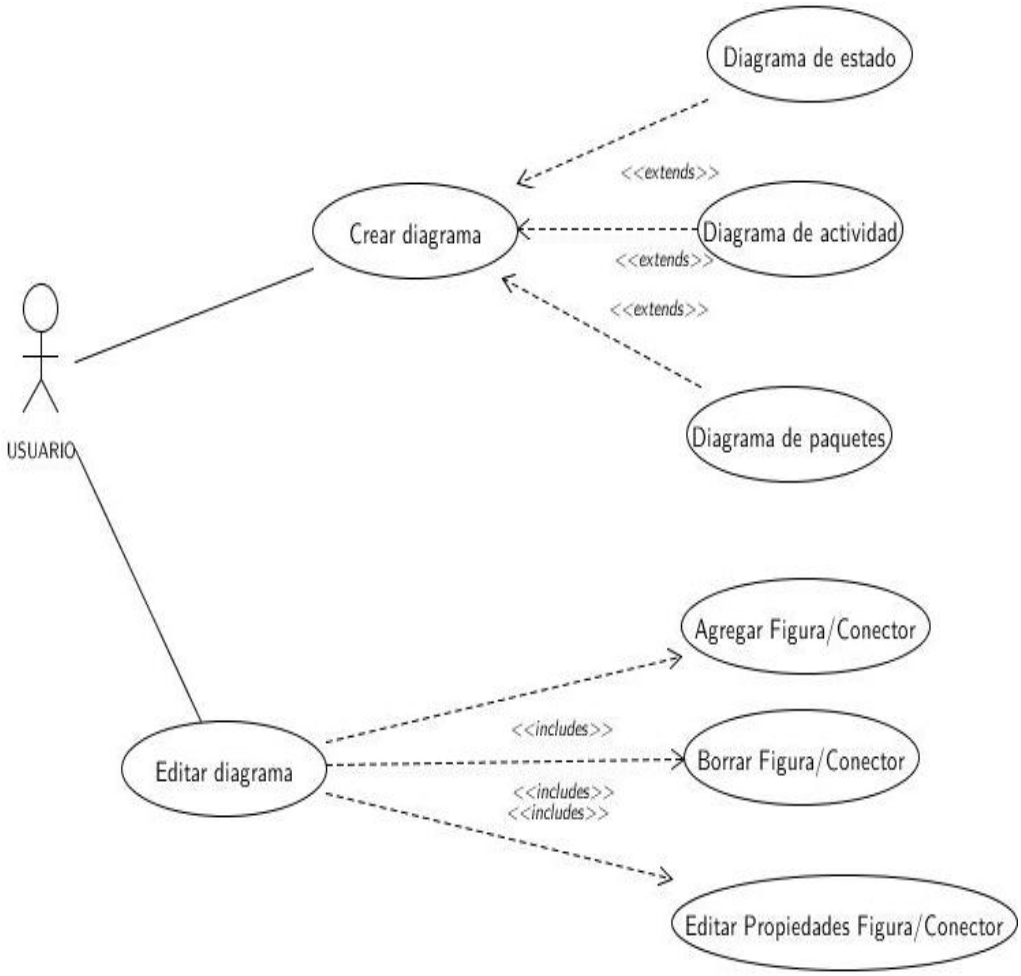


Diagrama 2. Casos de uso considerados para la extensión de UML4TeX.

A continuación se documentan los principales casos de uso mostrados en el Diagrama 2.

6.2.1.1 Caso de uso “Crear Diagrama”

El propósito de este caso de uso es proporcionar al usuario la opción para generar o crear un nuevo diagrama UML de acuerdo a su necesidad.

Caso de uso 1	**Crear Diagrama**
Actores	Usuario
Tipo	Básico
Resumen	El usuario solicita a la aplicación crear un nuevo diagrama UML para su edición. Los diagramas que puede elegir son los siguientes: Diagramas de estado, diagramas de actividad y diagramas de paquetes.
Pre condiciones	Tener la aplicación abierta.
Post condiciones	La aplicación muestra un panel de trabajo con un diagrama vacío y una barra de herramientas con los artefactos UML correspondientes al tipo de diagrama elegido.
Flujo principal	1. El usuario solicita a la aplicación la realización de la operación de creación de nuevo diagrama y especifica el tipo de diagrama. 2. Si hay un diagrama abierto ir a excepción <E1 Diagrama Existente>. Si no hay diagrama continuar en 3). 3. La aplicación crea una nueva área de trabajo 4. La aplicación obtiene la barra de herramientas con los artefactos UML correspondientes al tipo de diagrama elegido. 5. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código).
Sub flujo	Ninguno.
Excepciones	**<E1 Diagrama Existente**> 1. El sistema notificará al usuario mediante un cuadro de diálogo si desea descartar el diagrama actual con las opciones SI y NO. 2. Si la respuesta el SI. Regresar al flujo principal en el punto **3)** 3. Si la respuesta es NO. Finalizar caso de uso (No crea diagrama).
Frecuencia	Alta.

6.2.1.2 Caso de uso “Crear Diagrama de estado”

El propósito de este caso de uso es brindarle a usuario la opción para crear un diagrama de estado UML desde cero, proporcionando así los artefactos para que esta tarea pueda llevarse a cabo.

Caso de uso 2	**Crear Diagrama de estado**
Actores	Usuario
Tipo	Básico
Resumen	El usuario solicita a la aplicación crear un nuevo diagrama de estado UML para su manipulación.
Pre condiciones	Tener la aplicación abierta.
Post condiciones	La aplicación muestra un panel de trabajo con un diagrama vacío y una barra de herramientas con los artefactos UML correspondientes a un diagrama de estado.
Flujo principal	1. El usuario solicita a la aplicación la realización de la operación de creación de nuevo diagrama y especifica el tipo de diagrama. 2. Si hay un diagrama abierto ir a excepción <E1 Diagrama Existente>. Si no hay diagrama continuar en 3). 3. La aplicación crea una nueva área de trabajo 4. La aplicación obtiene la barra de herramientas con los artefactos UML correspondientes a un diagrama de estado. 5. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código).
Sub flujo	Ninguno.
Excepciones	**<E1 Diagrama Existente**> 1. El sistema notificará al usuario mediante un cuadro de diálogo si desea descartar el diagrama actual con las opciones SI y NO. 2. Si la respuesta el SI. Regresar al flujo principal en el punto **3)** 3. Si la respuesta es NO. Finalizar caso de uso (No crea diagrama).
Frecuencia	Alta.

6.2.1.3 Caso de uso "Crear Diagrama de actividad"

El propósito de este caso de uso es brindarle a usuario la opción para crear un diagrama de actividad UML desde cero, proporcionando así los artefactos para que esta tarea pueda llevarse a cabo.

Caso de uso 3	**Crear Diagrama de actividad**
Actores	Usuario
Tipo:	Básico
Resumen:	El usuario solicita a la aplicación crear un nuevo diagrama de actividad UML para su manipulación.
Pre condiciones	Tener la aplicación abierta.
Post condiciones	La aplicación muestra un panel de trabajo con un diagrama vacío y una barra de herramientas con los artefactos UML correspondientes a un diagrama de actividad.
Flujo principal	1. El usuario solicita a la aplicación la realización de la operación de creación de nuevo diagrama y especifica el tipo de diagrama. 2. Si hay un diagrama abierto ir a excepción <E1 Diagrama Existente>. Si no hay diagrama continuar en 3). 3. La aplicación crea una nueva área de trabajo 4. La aplicación obtiene la barra de herramientas con los artefactos UML correspondientes a un diagrama de actividad. 5. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código).
Sub flujo	Ninguno.
Excepciones	**<E1 Diagrama Existente**> 1. El sistema notificará al usuario mediante un cuadro de diálogo si desea descartar el diagrama actual con las opciones SI y NO. 2. Si la respuesta el SI. Regresar al flujo principal en el punto **3)** 3. Si la respuesta es NO. Finalizar caso de uso (No crea diagrama).
Frecuencia	Alta.

6.2.1.4 Caso de uso “Crear Diagrama de paquetes”

El propósito de este caso de uso es brindarle a usuario la opción para crear un diagrama de paquetes UML desde cero, proporcionando así los artefactos para que esta tarea pueda llevarse a cabo.

Caso de uso 4	**Crear Diagrama de paquetes**
Actores	Usuario
Tipo	Básico
Resumen	El usuario solicita a la aplicación crear un nuevo diagrama de paquetes UML para su manipulación.
Pre condiciones	Tener la aplicación abierta.
Post condiciones	La aplicación muestra un panel de trabajo con un diagrama vacío y una barra de herramientas con los artefactos UML correspondientes a un diagrama de paquetes.
Flujo principal	1. El usuario solicita a la aplicación la realización de la operación de creación de nuevo diagrama y especifica el tipo de diagrama. 2. Si hay un diagrama abierto ir a excepción <E1 Diagrama Existente>. Si no hay diagrama continuar en 3). 3. La aplicación crea una nueva área de trabajo 4. La aplicación obtiene la barra de herramientas con los artefactos UML correspondientes a un diagrama de paquetes. 5. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código).
Sub flujo	Ninguno.
Excepciones	**<E1 Diagrama Existente**> 1. El sistema notificará al usuario mediante un cuadro de diálogo si desea descartar el diagrama actual con las opciones SI y NO. 2. Si la respuesta el SI. Regresar al flujo principal en el punto **3)** 3. Si la respuesta es NO. Finalizar caso de uso (No crea diagrama).
Frecuencia	Alta.

6.2.1.5 Caso de uso “Editar diagrama”

El propósito de este caso de uso es proporcionar al usuario las funciones necesarias para que este sea capaz de editar un diagrama UML dentro de UML4TeX.

Caso de uso 5	**Editar diagrama**
Actores	Usuario
Tipo	Básico
Resumen	El usuario puede realizar una serie de operaciones para editar gráficamente el modelo de un diagrama. Las operaciones que puede hacer son: - Añadir un artefacto UML al diagrama (figura/conector) - Eliminar un artefacto UML del diagrama (figura/conector) - Editar las propiedades de un artefacto UML (figura/conector)
Pre condiciones	El usuario ha creado o recuperado un diagrama.
Post condiciones	Se cambia la estructura del modelo del diagrama y se actualizan las vistas del diagrama.
Flujo principal	1. El usuario desea agregar un artefacto UML al diagrama. Ejecuta el **caso de uso 6 Agregar Figura/Conector.** 2. El usuario desea editar un artefacto UML. Ejecuta el **caso de uso 7 Editar Propiedades** 3. El usuario desea borrar un artefacto UML del diagrama. **Ejecuta el caso de uso 8 Borrar Figura/Conector.**
Sub flujo	Ninguno
Excepciones	Ninguno.
Frecuencia	Alta

.

6.2.1.6 Caso de uso “Agregar Figura/Conector”

El propósito de este caso de uso es proporcionar al usuario un método para poder agregar figuras y conectores al diagrama en cuestión, así como de la misma manera poder eliminar dichos componentes a voluntad.

Caso de uso 6	**Agregar Figura/Conector**
Actores	Usuario
Tipo	Inclusión
Resumen	Agrega un nuevo artefacto UML (figura o conector) de algún tipo de diagrama al área de trabajo, actualiza el modelo del diagrama y su vista en la aplicación.
Pre condiciones	La aplicación tiene un diagrama abierto para su edición.
Post condiciones	Actualiza el modelo y la vista del diagrama editado.
Flujo principal	1. El usuario elige de la barra de herramientas un artefacto UML (figura o conector). 2. El usuario se posiciona en el área de edición y coloca el artefacto UML previamente elegido. 3. La aplicación agrega el nuevo componente al modelo del diagrama. 4. La aplicación actualiza el modelo y las vistas asociadas al diagrama editado.
Sub flujo	Ninguno
Excepciones	Ninguno
Frecuencia	Alta

6.2.1.7 Caso de uso “Editar propiedades Figura/Conector”

Proporciona al usuario un formulario con las características editables propias de la figura o conector del cual se quieran modificar las características iniciales de dicho componente.

Caso de uso 7	**Editar propiedades.**
Actores	Usuario
Tipo:	Inclusión
Resumen	Muestra pantallas y diálogos donde se pueden editar las propiedades alusivas a un componente (figura o conector) del diagrama editado y actualiza su modelo.
Pre condiciones	Existe un diagrama a editar y se ha seleccionado un componente para modificar propiedades.
Post condiciones	Actualiza el modelo lógico y las vistas del componente modificado.
Flujo principal	1. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona. 2. El usuario pide a la aplicación Editar el artefacto UML seleccionado. 3. La aplicación muestra el formulario relacionado con el artefacto UML a editar. 4. El usuario edita las propiedades y acepta los cambios. Si el usuario decide no aceptar los cambios ejecutar ***<E1 No aceptar cambios>*** 5. Se actualiza el modelo y la vista del diagrama editado.
Sub flujo	Ninguno.
Excepciones	***<E1 No aceptar cambios>*** 1. La aplicación no actualiza el diagrama. 2. Fin del caso de uso.
Frecuencia	Alta

6.2.1.8 Caso de uso “Borrar Figura/Conector”

Proporciona al usuario los métodos necesarios para eliminar una figura o conector del diagrama.

Caso de uso 8	**Borrar Figura/Conector**
Actores	Usuario
Tipo	Inclusión
Resumen	Elimina un artefacto UML (figura o conector) del diagrama, actualiza el modelo del diagrama y su vista en la aplicación.
Pre condiciones	La aplicación tiene un diagrama abierto para su edición.
Post condiciones	Actualiza el modelo y la vista del diagrama editado.
Flujo principal	1. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona. 2. El usuario pide a la aplicación borrar el artefacto UML seleccionado. 3. La aplicación elimina el nuevo componente al modelo del diagrama. 4. La aplicación actualiza el modelo y las vistas asociadas al diagrama editado.
Sub flujo	Ninguno
Excepciones	Ninguno
Frecuencia	Alta

6.2.2 Diagrama de clases

Se utilizó el Modelo Vista-Controlador [13] como patrón arquitectural.

6.2.2.1 Modelo de dominio

El modelo de dominio inicial de la aplicación está basado en las clases identificadas dentro del diagrama de clases y debido a que en general un diagrama se compone de figuras y conectores se tienen identificadas tres clases principales sobre las cuales se basa el modelo inicial de la aplicación y sobre las cuales está centrada la extensión de este proyecto. Dichas clases son:

- Clase Diagrama
- Clase Figura
- Clase conector

Cada diagrama UML tiene asociadas a su estructura cierta cantidad y tipo de figuras y conectores determinados y perfectamente definidos y delimitados en sus características y funciones que cada uno desempeña dentro de un determinado tipo de diagrama.

Puesto en claro que un diagrama está compuesto por una colección de figuras y conectores, se debe tomar en cuenta que dicha colección de figuras y conectores deben tener una coherencia para poder ser interpretados y que de esta manera representen alguna estructura del mundo real. En este caso la coherencia entre figuras y conectores se logra cuando dos figuras se unen por medio de un conector.

Por lo que en esta ocasión al igual que en su primera versión, los diagramas que se agregaron a esta nueva versión deben ser capaces de agregar y eliminar tanto figuras como conectores pertenecientes a su estructura, así como también tener la capacidad de conectar dos figuras de su las contenidas en su colección

A continuación, se presenta el diagrama de clases desarrollado inicialmente para este proyecto y sobre el cual se realizaron distintas iteraciones para lograr extender la funcionalidad del mismo.

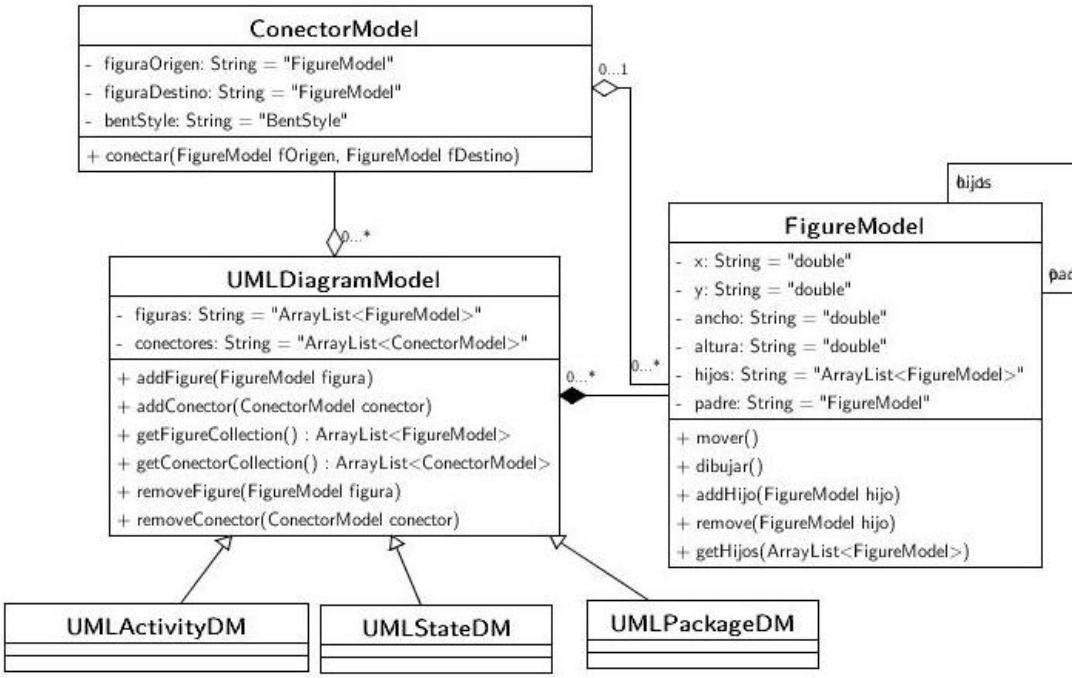


Diagrama 3. Diagrama de clases utilizado en el modelado del problema

El Diagrama 3 muestra la interacción que tienen las clases identificadas en el punto 6.2.2.1. En este diagrama se observa que la clase `UMLActivityDM`, `UMLStateDM` y `UMLPackageDM` extienden a la clase `UMLDiagramModel`.

Para propósito de este proyecto se realizaron iteraciones sobre las clases y paquetes que conforman el proyecto, por lo cual se realizaron implementaciones concretas de las clases mostradas en el diagrama anterior, las cuales están representadas en el Diagrama 4.

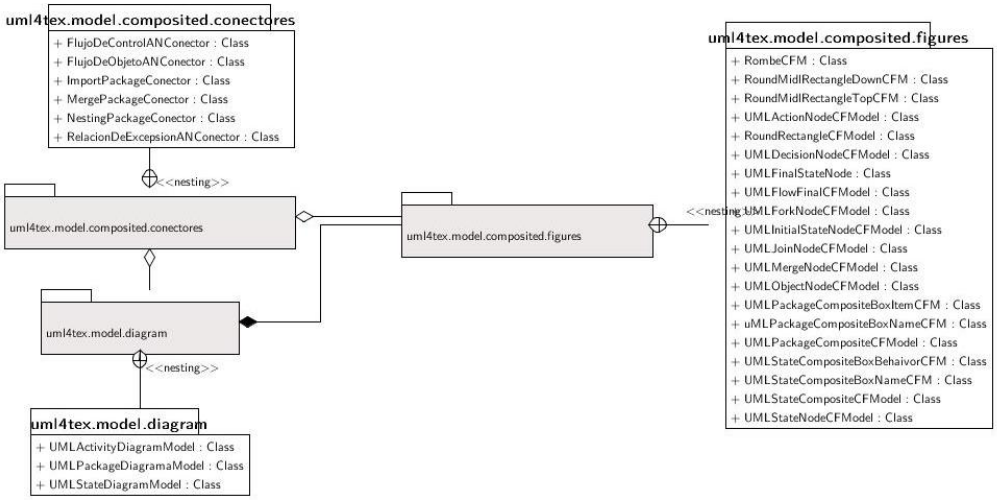


Diagrama 4. Diagrama de paquetes utilizado para la iteración sobre el modelo del problema

6.2.2.2 Vistas

La aplicación contiene tres vistas básicas, las cuales son la vista de la representación gráfica, la vista, las vista de la información de los elementos del modelo y la vista que representa el modelo en código *PSTricks.*

Para generar estas tres vistas se utilizó el patrón de diseño *Observer* [14] el cual se implementó dentro de la interfaz `Observable` la cual se encarga de monitorear los cambios ocurridos en el modelo. El modelo representado por la clase `UMLDiagramModel` se encarga de notificar a todas las vistas si existe algún cambio.

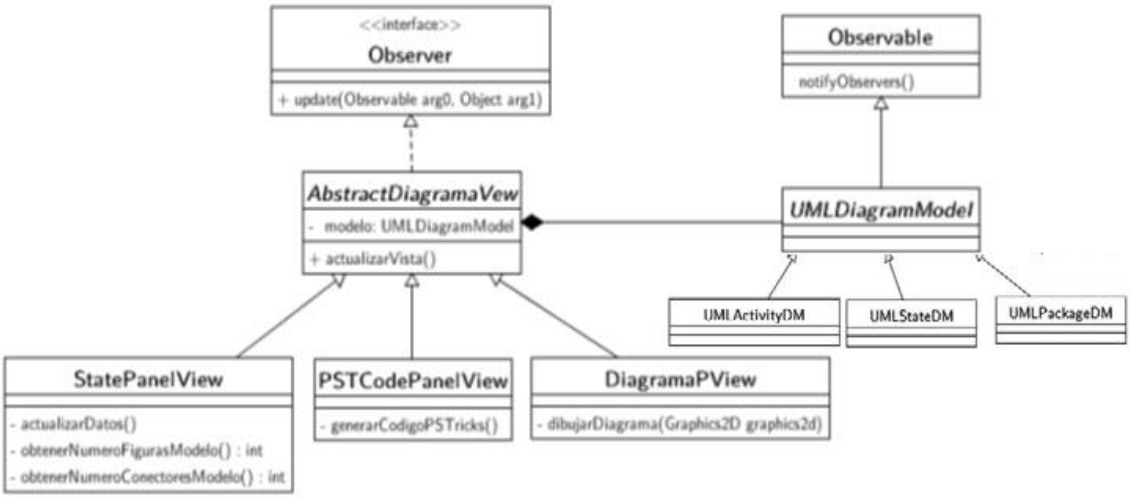


Diagrama 5. Diagrama general de las vistas de la aplicación

6.2.3 Interfaz de usuario

La interfaz de usuario está compuesta por las tres vistas mencionadas en el apartado 6.2.2.2, así mismo integra una **barra de menús** y una **barra de herramientas** Además la interfaz está diseñada de tal manera que sea fácil de entender y manejar para el usuario.

Dicha interfaz provee al usuario de un área de trabajo y las herramientas necesarias para la creación y edición de los diagramas con los que puede trabajar dentro de la aplicación (ver Figura 18).

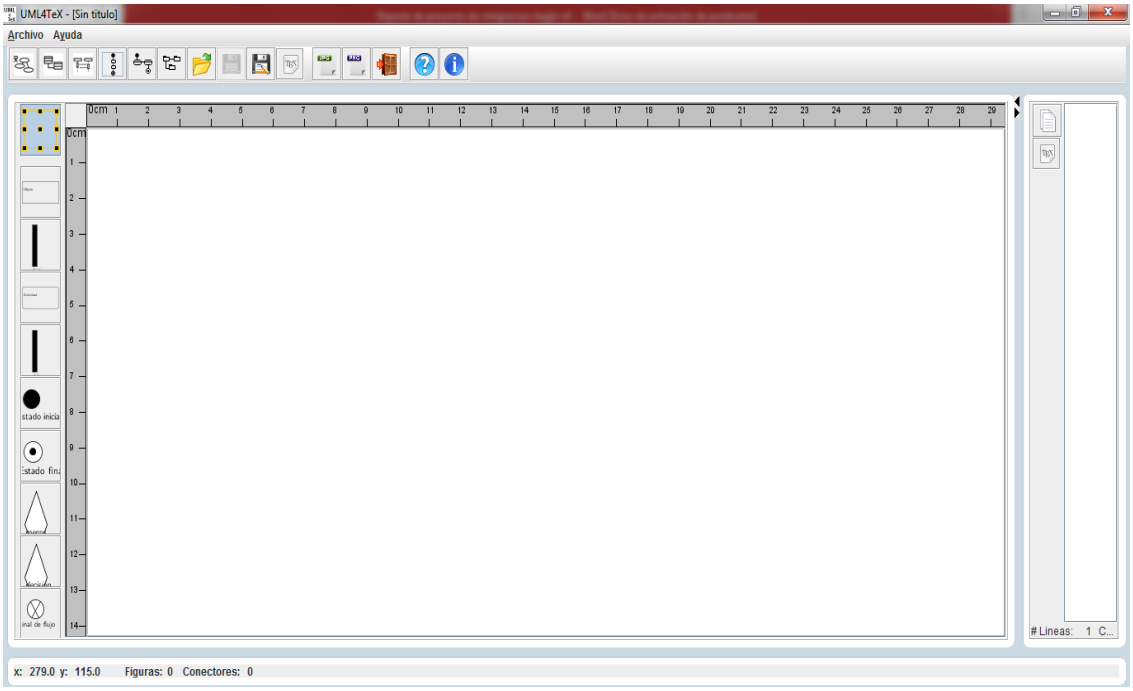


Figura 18 Interfaz Gráfica de Usuario

Barra de herramientas: Contiene los botones de las herramientas principales de la aplicación donde se realizarán las principales acciones que permitirá la aplicación como abrir un archivo, guardarlo, crear diagramas o exportarlos.

Barra de menús: Contiene comandos y menús para la edición de propiedades. O diferentes comandos útiles en la aplicación.

Área de trabajo: Área donde se recibirán los eventos de entrada del usuario además de servir como uno de los contenedores de una de las vistas del modelo como es la representación gráfica del diagrama.

6.2.3.1 Clase Área de trabajo

Para efectos de este proyecto que resulto en UML4TeX v2 se trabajó principalmente con los aspectos que estaban relacionados con el área de trabajo por lo cual se tomó en cuenta el siguiente diagrama de clases.

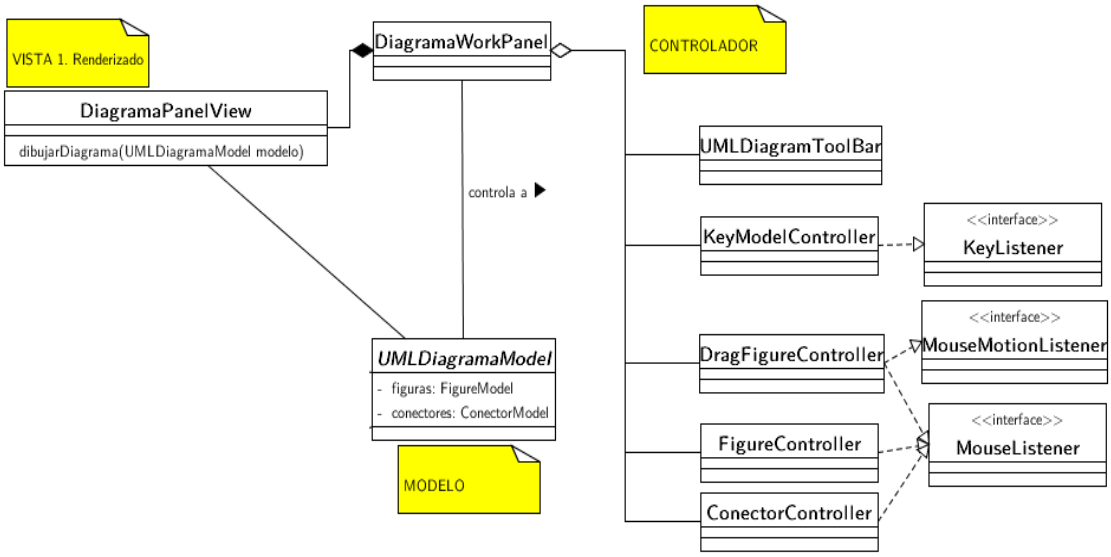


Diagrama 6. Diagrama de clases del área de trabajo

Donde la clase `DiagramaWorkPanel` actua como controlador central de la plicación ya que se encarga de recibir interpretar todos los eventos ocurridos dentro de la aplicación como lo son la creacion de nuevas instancias de las clases `UMLDiagramaModel`, `UMLDiagramToolBar`, `FigureController`, `ConectorController` y `DragFigureController`.

6.2.3.2 Clase UMLDiagramToolBar

La aplicación cuenta con una barra de herramientas para cada uno de los diagramas que puede editar.
La clase `UMLDiagramToolBar` cumple con esta tarea ya que para cada diagrama soportado de la aplicación se creará una instancia y agregará a cada botón que la compone un icono representando la figura o conector que quiere representar. Cabe mencionar que dichas barras de herramientas no son estáticas como la barra de herramientas general de la aplicación; es decir que las barras de herramientas de cada uno de los diagramas se construyen cada vez que se abre o crea un tipo de diagrama UML de los 6 que soporta UML4TeX v2
La clase `DiagramaWorkPanel` debe implementar la interfaz `ICreateControllerListener` y registrarlo a la barra de herramientas de artefactos UML para que pueda escuchar la acción que debe hacer la barra de herramientas al presionar un botón.
Cuando se dispara un evento de la barra de herramientas, este es capturado por `DiagramaWorkPanel` y dependiendo del botón elegido creará una instancia del controlador de figuras (`FigureController`) o del controlador de conectores (`ConectorController`), controladores que deben ser creados por la clase de utilería `AFigureAndConectorControllerFactory` que es una fábrica abstracta para instanciar el controlador de figura o conector adecuado para poder dibujarlo en el área de trabajo. A continuación se muestra el diagrama de clases considerado para la creación de las distintas barras de herramientas de los diagramas.

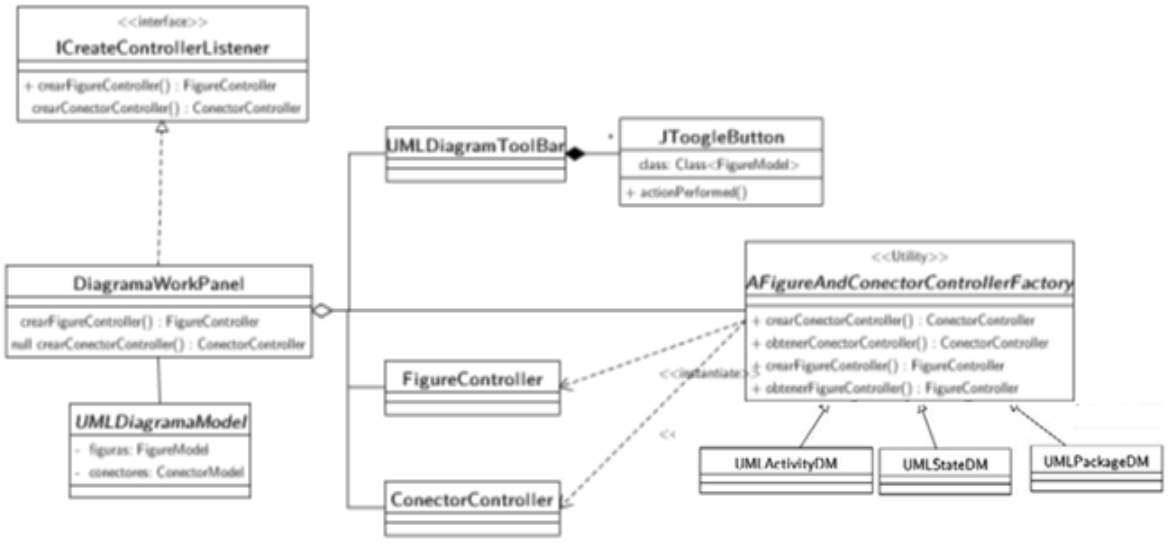


Diagrama 7. Diagrama de clases de las barras de herramientas

6.2.3.3 Controladores

Los controladores son los encargados de capturar los eventos de entrada de la aplicación y manejarlos de tal manera que se modifique el modelo, también debe notificar de los cambios del modelo a las vistas para que estas puedan actualizar su presentación.

La aplicación cuenta con tres tipos importantes de controladores los cuales son `FigureController, ConectorController y DragFigureController` este último como controlador para arrastrar las figuras y recibir cualquier evento relacionado con el mouse de ordenador.

6.2.3.4 Clase controlador de figuras

Este tipo de controlador se encarga de recibir y procesar todos los eventos que estén relacionado con un artefacto UML que represente una Figura. A continuación se presenta el diagrama de clases donde se representa todas las relaciones que tiene la clase `FigureController` y la manera en la que interactúa con la colección de figuras y el diagrama con el que están asociadas dichas figuras. Todos los controladores de figuras `ConcreteFigureController` se obtienen de la fábrica que hereda de `AfigureAndConectorControllerFactory` (Ver Diagrama 8).

Los controladores de figuras tienen las facultades de crear, agregar y eliminar una figura en el modelo, además también tienen la facultad para llamar a los respectivos paneles de edición de las figuras.

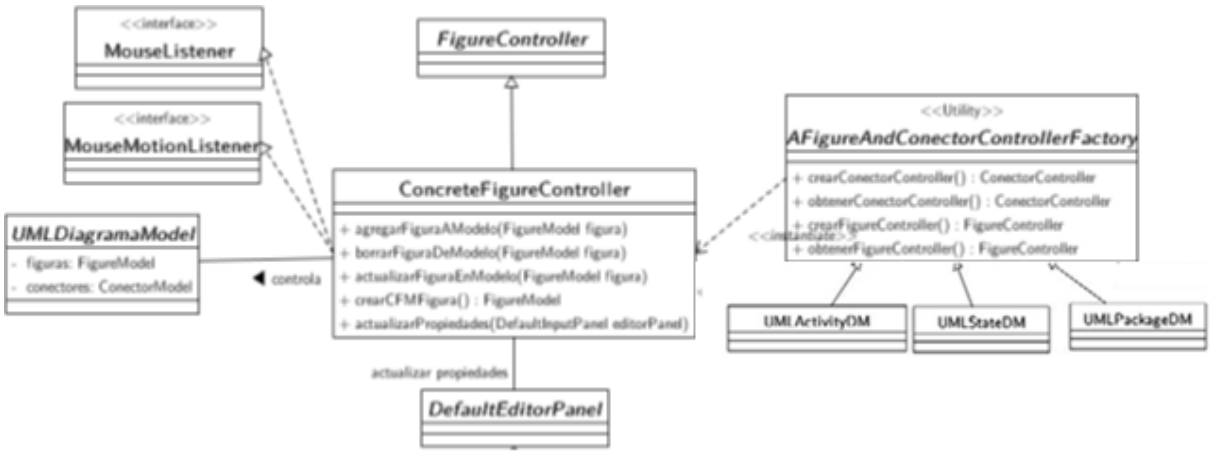


Diagrama 8. Diagrama de clases de los controladores de figuras

Por cada figura concreta se codificó un controlador el cual se encuentra dentro del paquete que se muestra en el siguiente diagrama (Ver Diagrama 9).

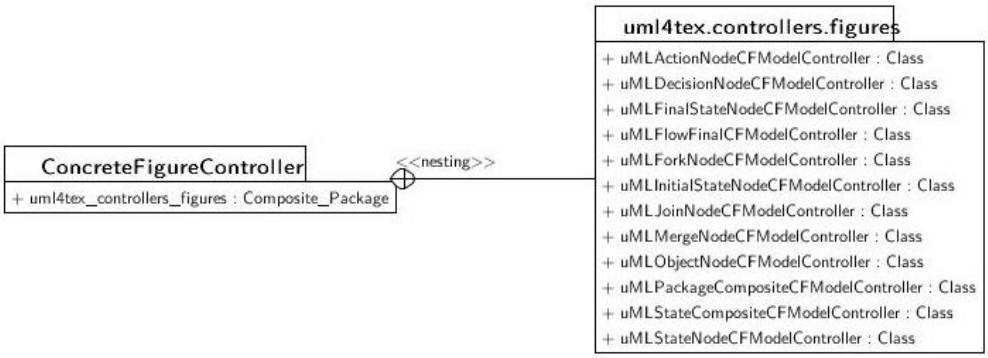


Diagrama 9. Implementaciones concretas de controladores de UML4TeX v2

Para las figuras que así lo requieren se implementaron igualmente los paneles de edición necesarios para poder controlar las características de cada figura para lo cual se realizaron las siguientes implementaciones concretas que heredan de la clase `DefaultEditorPanel` (Ver Diagrama 10).

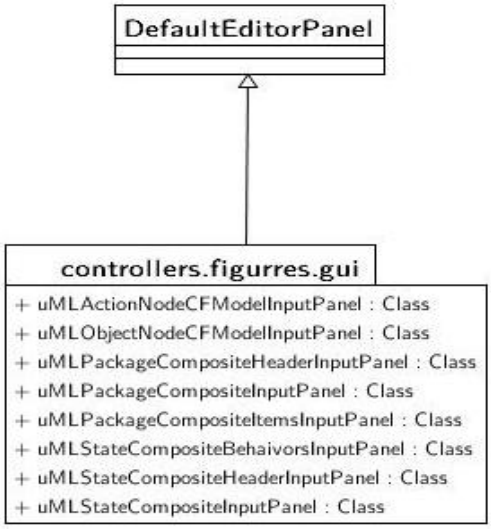


Diagrama 10. Implementaciones concretas de paneles de edición de figuras de UML4TeX v2

6.2.3.5 Clase Controlador de Conectores

Este controlador se encargará de interceptar y procesar los eventos relacionados con un artefacto UML que represente un conector.

A continuación, se presenta el diagrama de clases para la clase Controlador de Conectores.

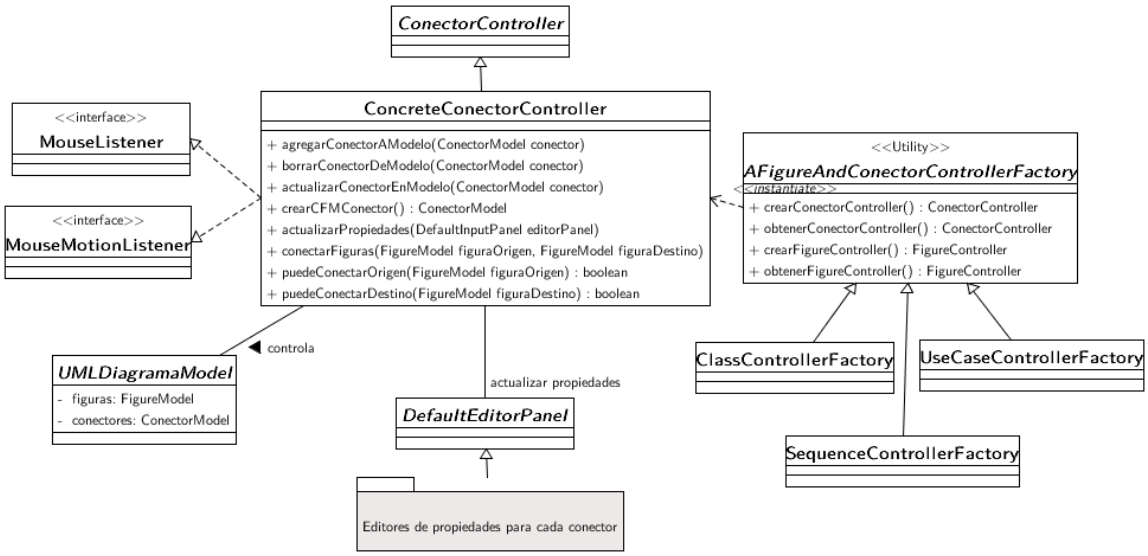


Diagrama 11. Diagrama de clases de los controladores de conectores

Al igual que el controlador de figuras todos los controladores de conectores `ConcreteConectorController` se obtienen de la fábrica `AFigureAndConectorControllerFactory` y todos los controladores concretos de los conectores heredan de la clase `ConectorController` (Ver Diagrama 11).

Para cada uno de los conectores que se incluyeron en este proyecto se realizó una implementación concreta,así mismo para cada panel de edición de los conectores implementados se realizó una implementación concreta la cual hereda de la clase `DefaultEditorPanel`

6.2.3.6 Clase para mover y arrastrar figuras

La clase `DragFigureController` es el controlador del modelo para interceptar los eventos del mouse que se presentan en el área de dibujo del diagrama.
El diagrama de clases muestra la relación que tiene con el modelo y los interceptores de eventos del mouse, llamados *listeners*, los cuales deben asociar una acción para cada evento recibido del mouse. Por ejemplo, seleccionar una figura del diagrama cuando se haga click sobre ella, o arrastrar la figura cuando se mantenga el botón izquierdo del mouse presionado (ver Diagrama 14).

De esta manera cada que se agrega un nuevo tipo de diagrama en la aplicación y por consiguiente una colección distinta de figuras y conectores lo único que se debe hacer es agregar el nuevo tipo de diagrama en la clase `DragFigureController`

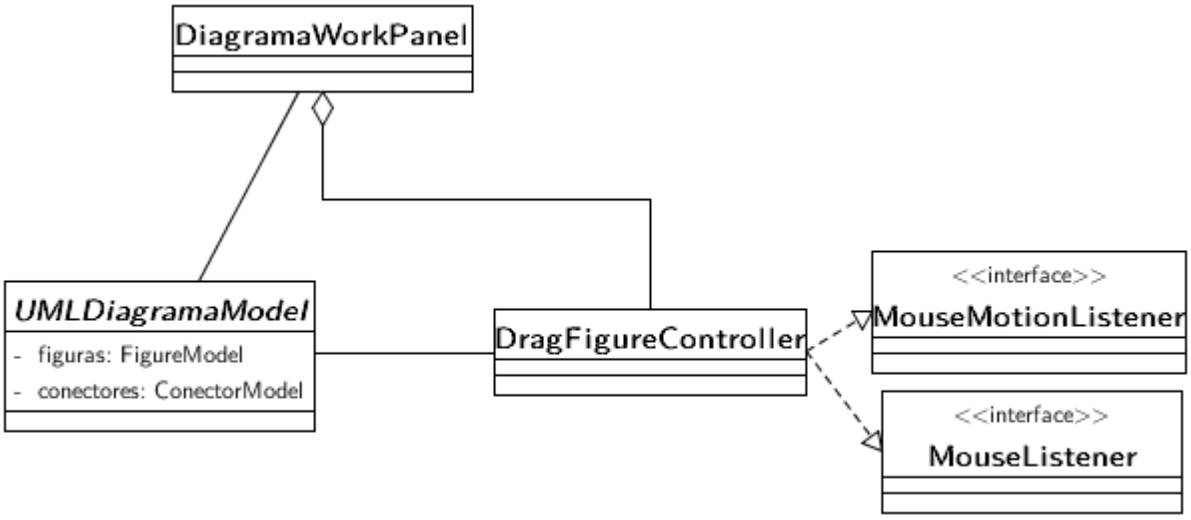
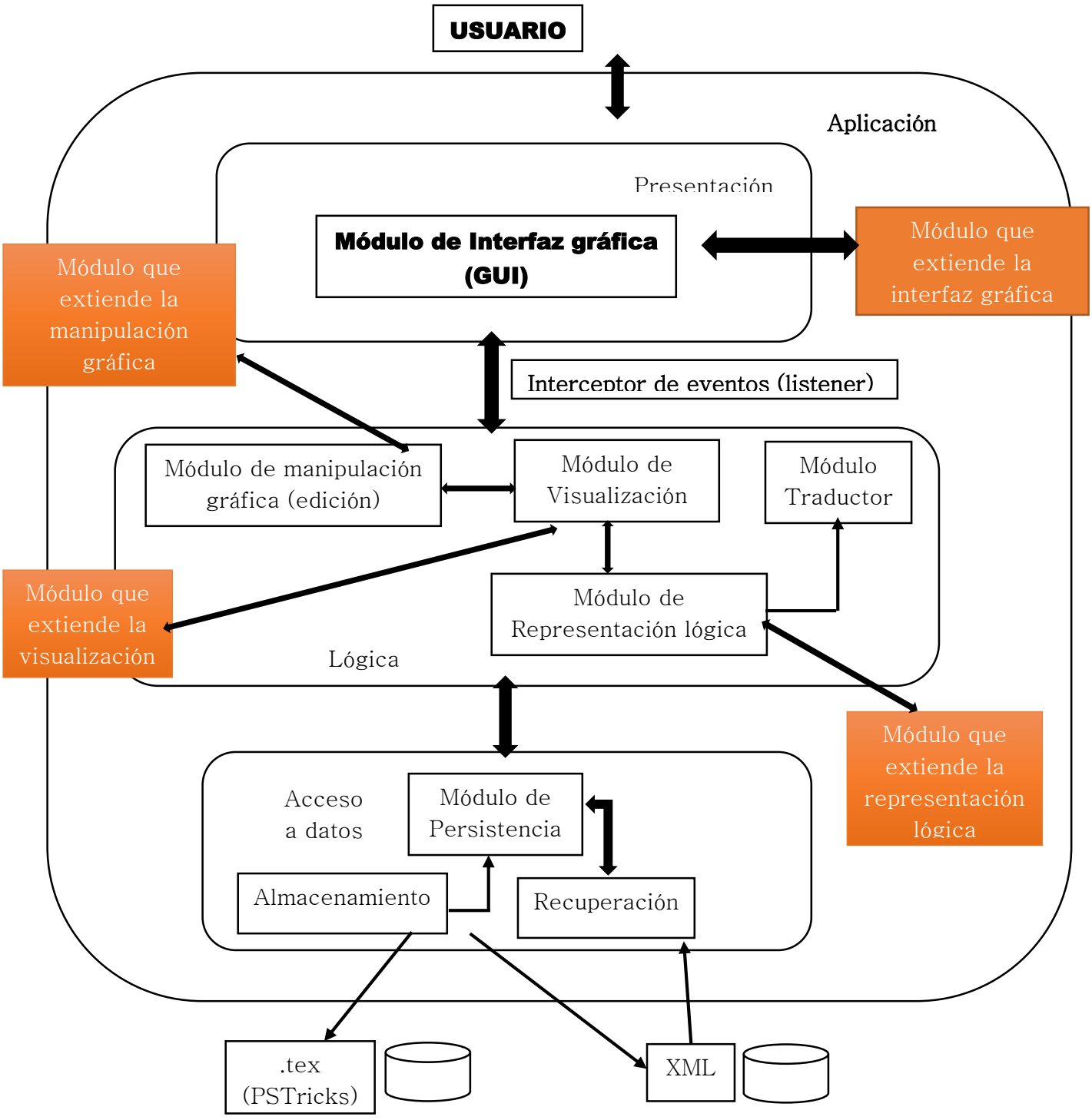


Diagrama 12. Diagrama de clases del controlador para arrastrar las figuras

6.2.4 Arquitectura del sistema

Como ya se ha mencionado anteriormente en este reporte, el patrón arquitectural sobre el que está construida esta aplicación es el Modelo-Vista-Controlador [13]. En el siguiente diagrama se muestra la forma que está dividida la aplicación y la manera en que se comunican cada uno de los bloques que la componen

Cabe hacer mención que los módulos de color naranja son los módulos que se implementaron en UML4TeX v2 para llevar a cabo la extensión de funcionalidad.



La aplicación se construyó siguiendo una arquitectura de diseño en tres capas, la cual se basa en a) capa de acceso a datos, b) capa de lógica de negocio y c) capa de presentación o aplicación.

Dado que la aplicación está construida en tres capas la adición de módulos que permitieron su extensión para dar paso a UML4TeX v2 fue relativamente sencilla debido a que no se tuvo que modificar la estructura de los módulos ya existentes, sino que simplemente se iban añadiendo los elementos necesarios para la extensión del módulo con el que se estuviera trabajando.

6.2.5 Comunicación entre bloques

El diseño de la aplicación está basado en la arquitectura de tres capas y se desarrolló implementando el patrón MVC (Modelo-Vista-Controlador) [13].

El intercambio de información entre los diferentes bloques de cada capa se da por medio del módulo interceptor de eventos que implementa el patrón de diseño *observador*.

El formato de entrada y salida que son utilizados en los módulos de persistencia y de representación lógica de los diagramas UML diseñados en la aplicación, está basado en el estándar XML, usando una extensión de aplicación propietaria de la aplicación *. UMLaTeX.

La aplicación es capaz de exportar los diagramas en el formato de salida. .tex, que contiene las macros de PSTricks generadas por el módulo traductor.

6.3 Hardware y software necesario

6.3.1 Tecnología para el desarrollo de la aplicación

Se tuvieron en disposición los siguientes recursos, los cuales fueron suficientes para la realización del proyecto:

Hardware:
- Computadora Personal con un procesador Intel Pentium ™ CPU G3250 ™ a 3.20GHz, con memoria RAM de 8GB.

Software:
- Entorno de desarrollo integrado (IDE) NetBeans 8.1 [15]
- JDK (Java Development Kit) 1.8 [16]

Se eligió como plataforma de desarrollo Java ya que desde la concepción de la aplicación se tenía en mente que fuera portable.
El software necesario para el desarrollo de la aplicación, es de distribución libre y no requiere de ninguna licencia.

7. Resultados

En la realización de este proyecto se tuvieron los siguientes resultados de acuerdo con los objetivos planteados en el punto **4.2**

- Se crearon e integraron las clases y métodos que conforman el módulo que extiende la representación Lógica de los diagramas de estado, de actividad y de paquetes UML en UML4TeX v2.
- Se crearon e integraron las clases y métodos que conforman el módulo que extiende la visualización de los diagramas de estado, de actividad y de paquetes UML en UML4TeX v2.
- Se crearon e integraron las clases y métodos que conforman el módulo que extiende manipulación gráfica (edición) de los diagramas de estado, de actividad y de paquetes UML en UML4TeX v2.
- Se crearon e integraron las clases y métodos que añaden los elementos gráficos correspondientes a cada uno de los diagramas: de estado, de actividad y de paquetes y que da como resultado la extensión del módulo de la interfaz gráfica de usuario para UML4TeX v2.

Cabe hacer mención que a pesar de que se logró integrar de manera correcta todos y cada uno de los diagramas antes mencionados, no se pudo incluir dentro del diagrama de paquetes una figura que represente un paquete y que este a su vez contenga explícitamente otras figuras correspondientes al diagrama de paquetes.

8. Análisis y discusión de resultados

En esta sección se analizarán y discutirán los resultados obtenidos al finalizar este proyecto que dio como resultado la aplicación UML4TeX v2.

Dado que este proyecto estuvo basado en la integración de tres nuevos tipos de diagramas en la aplicación que lleva por nombre UML4TeX v2 lo que se buscaba era que dicha aplicación fuera capaz de crear, editar y manipular los diagramas de estado, de actividad y de paquetes UML

Para esto es de vital importancia que se pueda crear un diagrama nuevo, insertar tantas figuras y conectores al diagrama como el usuario requiera, mover dichas figuras y conectores a voluntad sobre el área de trabajo, editar las propiedades de las figuras y los conectores asociados al diagrama y por último borrar/eliminar las figuras y conectores del diagrama siempre que así se desee.

8.1 Módulo de manipulación gráfica del diagrama. Creación de un diagrama

La aplicación cuenta con un controlador el cual se encarga de administrar que tipos de diagramas son los que se pueden crear y manipular en la aplicación. Dicho controlador cuenta con un vector en donde se encuentran registrados los tipos de diagramas editables, de esta manera agregar otro tipo de diagrama en un futuro a la aplicación es relativamente sencillo.

Gráficamente el vector con el que cuenta la aplicación se representa por medio de un listado de botones que se presentan en la pantalla principal de la aplicación al ejecutarse por primera vez o en la barra de herramientas de la interfaz gráfica (Ver Figura 19).

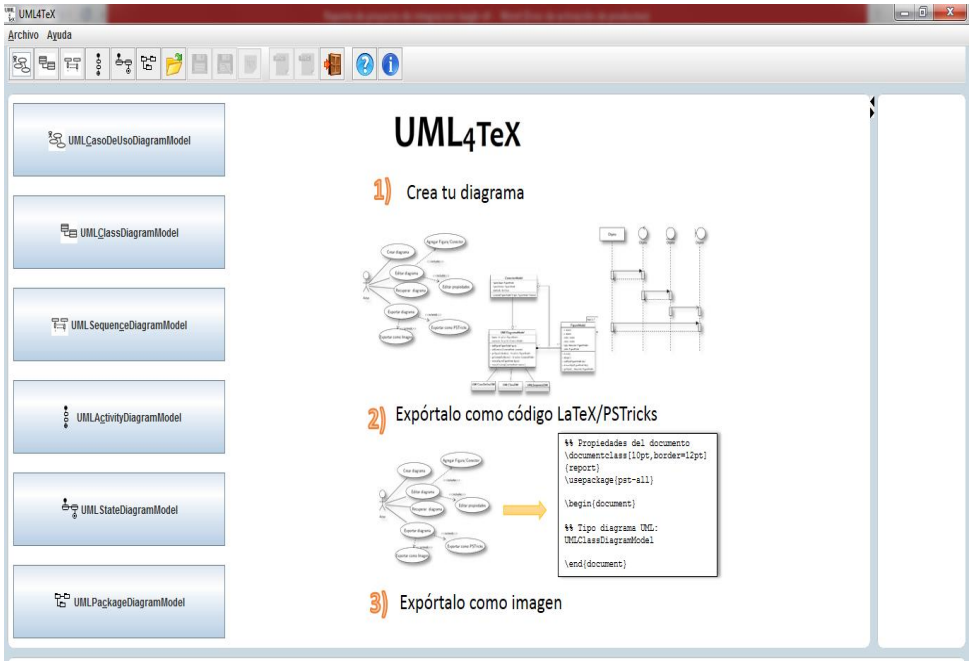


Figura 19 Creación de un nuevo diagrama

8.1.1 Módulo de manipulación gráfica del diagrama. Barra de herramientas de cada uno de los diagramas

Para lograr crear y editar un diagrama UML es necesario contar con las herramientas y elementos necesarios y suficientes para conformar un diagrama UML el cual tenga coherencia, para ello cada uno de los diagramas cuenta con sus elementos propios que lo identifican y diferencian de otros diagramas.

Con el fin de poder integrar nuevas figuras y conectores a la colección de cada uno de los nuevos diagramas integrados fue necesario crear para cada una de las nuevas figuras integradas una clase que modelara gráficamente dichas figuras, así como el respectivo controlador a cada una de estas figuras y después añadir cada una de estas figuras y sus respectivos controladores a las fabricas correspondientes según el diagrama al que pertenecieran (Ver Figura 20).

De la misma manera para cada una de las figuras creadas se incluyó una tablilla con la cual se pudieran editar las características correspondientes a cada una de las figuras y conectores creados.

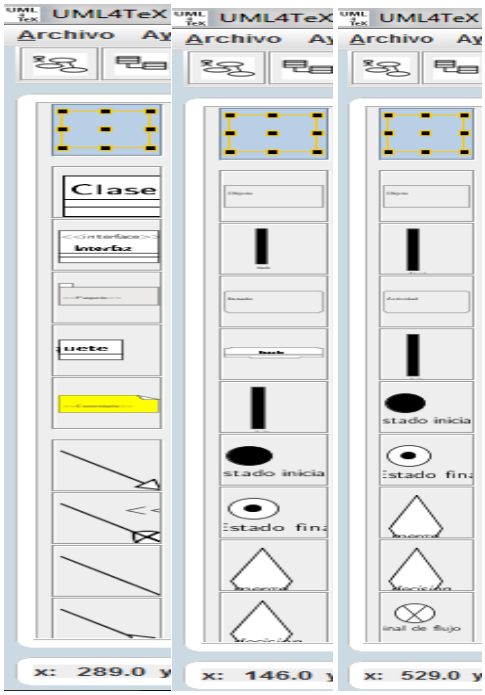


Figura 20 Barras de herramientas de los diferentes diagramas

8.1.2 Módulo de manipulación gráfica del diagrama. Movimiento

El controlador asociado al movimiento de las figuras permitirá saber los eventos de entrada del mouse dentro de la aplicación y permite seleccionar y arrastrar los diferentes artefactos de UML que se encuentran en el diagrama, así como de actualizar su posición dentro del diagrama.

Este controlador de movimiento ya existía, de manera que solo se amplió su funcionalidad, primero creando los métodos fábrica de los elementos que debía contener cada diagrama y después añadiendo dichas fábricas al controlador de movimiento (Ver Figura 21).

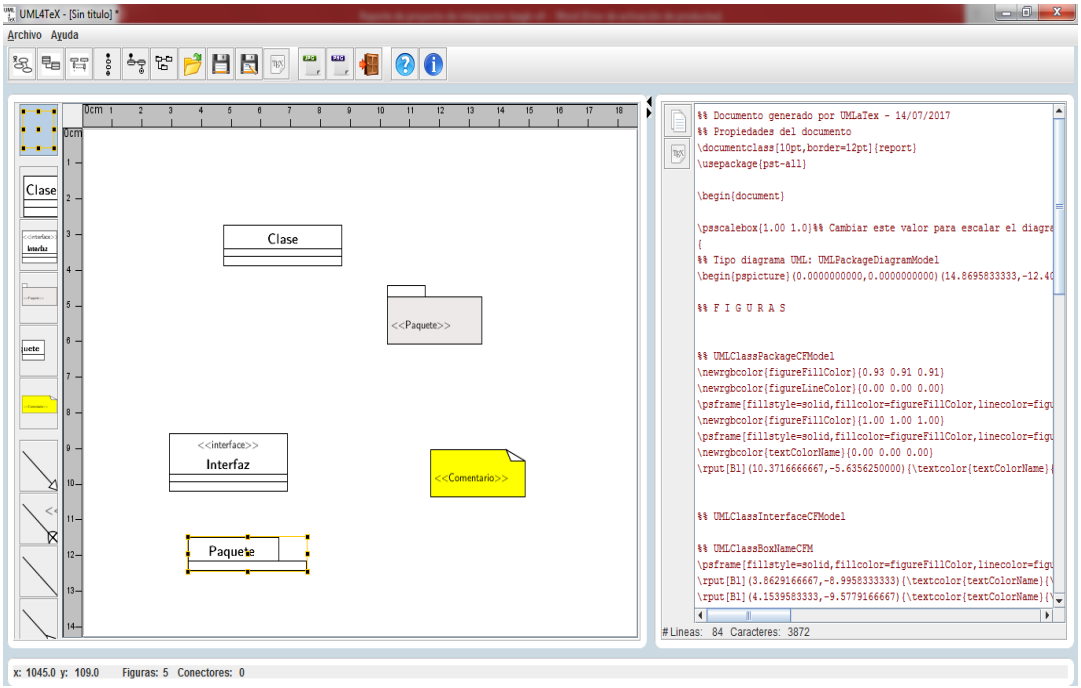


Figura 21 Movimiento de los elementos de un diagrama a través del área de trabajo

8.1.2.1 Inserción de elementos al modelo

La barra de herramientas asociada a cada diagrama dispara los eventos que obtienen el controlador asociado al artefacto UML que se va a insertar en el área de trabajo de la aplicación. Permitiendo así modificar el modelo para agregar una nueva figura o conector (Ver Figura 22).

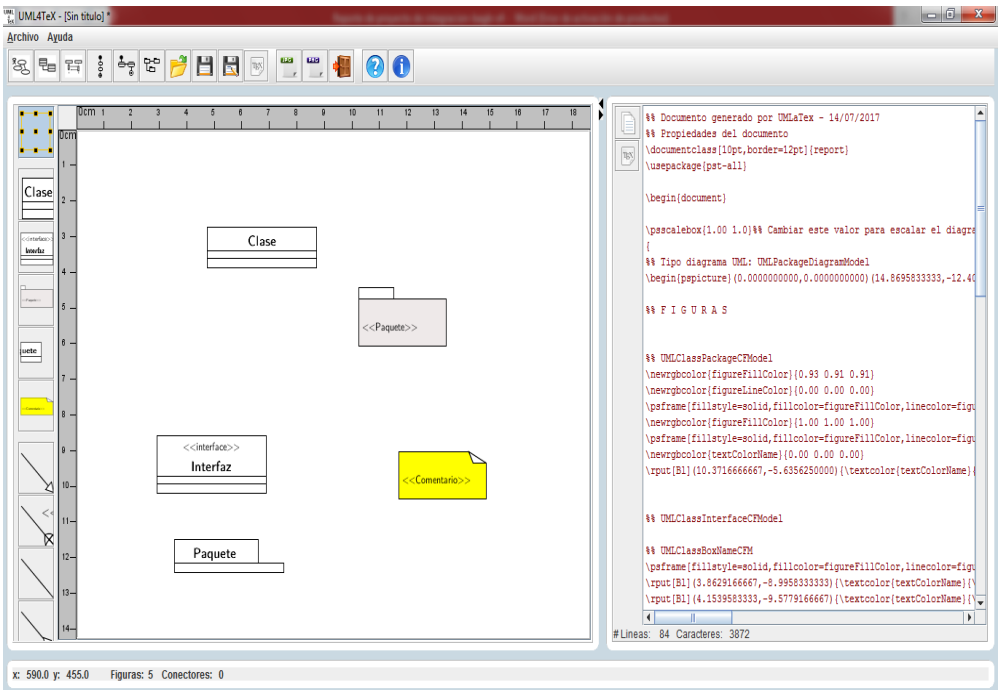


Figura 22 Inserción de elementos al modelo

8.1.2.2 Módulo de manipulación gráfica del diagrama. Edición de los elementos del diagrama

Cada modelo de diagrama UML cuenta con una serie de controladores para recibir y tratar los elementos relacionados a la adición, cambio de propiedades, incluso eliminación de una figura o conector utilizados en el diagrama.

El controlador asociado al elemento UML que se encuentra dentro del modelo del diagrama es capaz de saber cuál acción a tomar cuando se va a editar alguna de sus propiedades y actualiza el modelo cuando identifica que alguna de las propiedades ha cambiado. Esto se representa mediante la inclusión de distintos paneles que apoyan al usuario en la tarea de actualizar los datos de una figura o conector dentro del diagrama (Ver Figura 23).

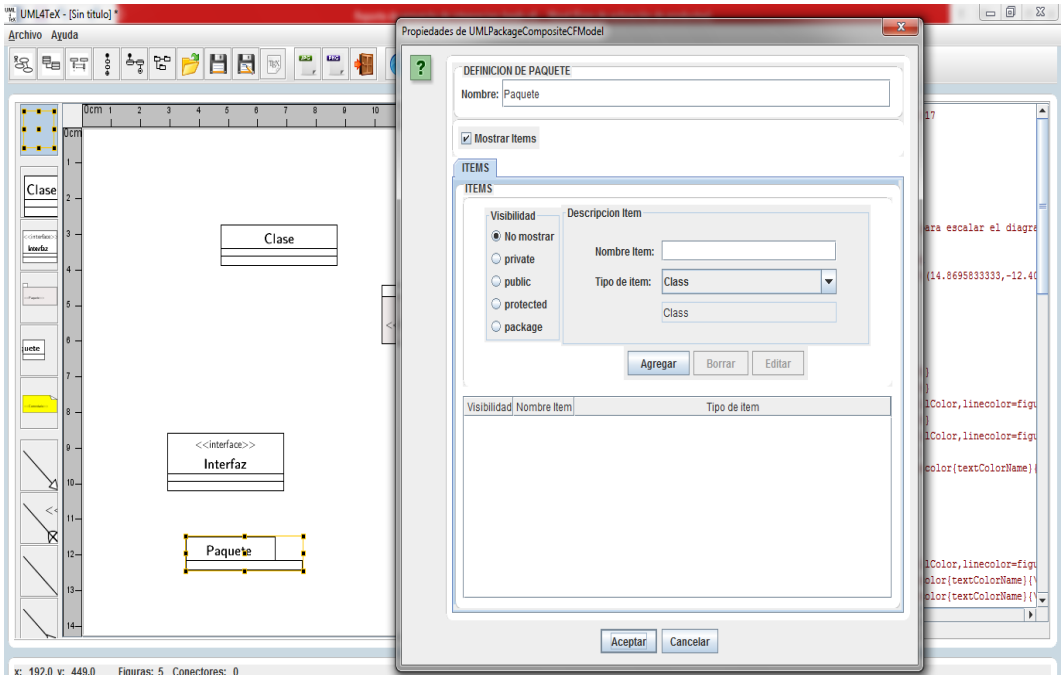


Figura 23 Edición de las propiedades de un elemento del modelo

8.1.2.3 Módulo de manipulación gráfica del diagrama. Eliminar un elemento

El controlador asociado al modelo es capaz de identificar el evento de entrada del usuario, por ejemplo, al presionar una tecla y si este evento corresponde una acción, entonces la realiza. Para el caso de eliminar la figura basta con oprimir la tecla suprimir para que se dispare el evento y el controlador elimine del modelo la figura o conector asociado (Ver Figura 23 y Figura 24).

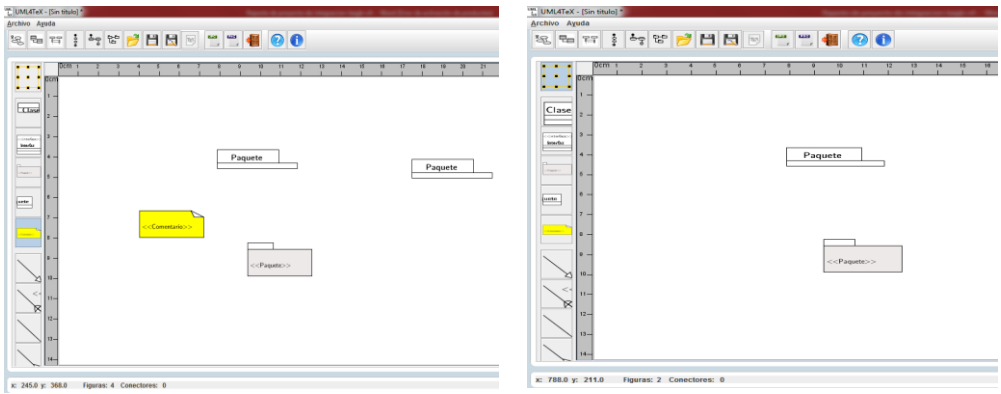


Figura 24 Eliminación de un elemento del modelo

9. Conclusiones

UML4TeX v2 cumple con los objetivos establecidos en el alcance de este proyecto. Esto puede concluirse con base en que se construyeron e integraron exitosamente los módulos y paquetes que permitieron la extensión la funcionalidad de la aplicación dando como resultado la inclusión de tres nuevos tipos de diagramas (diagramas de actividad, diagramas de estado y diagramas de paquetes) dentro de UML4TeX los cuales se suman a los tres tipos de diagramas con los que previamente contaba la aplicación.

En UML4TeX v2 falta agregar dentro del diagrama de paquetes una figura que represente un paquete en el cual se puedan colocar explícitamente otras figuras dentro de este paquete.

Dado que el desarrollo de la aplicación continúa basado en el patrón arquitectural Modelo-Vista-Controlador. Se utiliza el RUP como metodología de desarrollo de software y es un proceso iterativo e incremental. El lenguaje de programación Java. Estos tres aspectos contribuyen a que la aplicación pueda seguir desarrollándose más en un futuro haciendo que su expansión sea sencilla.

10. Perspectivas de proyecto

Debido a que este proyecto fue construido desde su primera versión utilizando el Modelo-Vista-Controlador, así como el modelo RUP, es relativamente sencillo hacer una extensión de su funcionalidad, ya que el diseño arquitectónico del proyecto está muy bien organizado y estructurado por lo que es sencillo seguir apoyándose del modelado en espiral para seguir aplicando más iteraciones en los paquetes y las clases para extender la funcionalidad de la aplicación fácilmente.

Se tiene previsto que a futuro la aplicación cuente con funciones que le permitan

1. Realizar lo que se conoce como ingeniería inversa.
2. Generación de código fuente en distintos lenguajes de programación con base en los diagramas generados por la aplicación.
3. Integrar más tipos de diagramas UML para que puedan ser creados y editados dentro de la aplicación (la perspectiva para este punto es incluir los trece diagramas que maneja el lenguaje UML).
4. Integrar otros formatos de exportación de los diagramas generados dentro de la aplicación, tales como son el .pdf o incluso el .docx.

11. Referencias bibliográficas

[1] C.Larman, *Applying UML and patterns*. Upper Saddle River, N.J.: Prentice Hall PTR, 2005.

[2] Ibm.com, "IBM developerWorks : Download : IBM Rational Software Architect", 2016. [Online]. Available: https://www.ibm.com/developerworks/downloads/r/architect/.

[3] Umbrello.kde.org, "Umbrello Project - Welcome to Umbrello - The UML Modeller", 2016. [Online]. Available: https://umbrello.kde.org/.

[4] F. Villalobos Martínez, *Editor gráfico de diagramas UML con la capacidad de generar macros PSTricks*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[5] R. Cortés Díaz, C. Linares Martínez, Dibujo *en touch screen transmitido por PC a un dispositivo con láser*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2006.

[6] J.C. Mares Martínez, *Implementacion de una interfaz gráfica interactiva para algoritmos de ordenamiento y estructuras de árbol*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[7] Excelsoftware.com, "MacA&D Products - Requirements Management & Software Design Tools", 2016. [Online]. Available: http://www.excelsoftware.com/maca&dproducts.html.

[8] Ibm.com, "UML-to-Java transformation in IBM Rational Software Architect editions and related software", 2016. [Online]. Available: http://www.ibm.com/developerworks/rational/library/08/1202_berfeld/.

[9]C. Jiménez de Parga and M. Arias Calleja, *UML*. Madrid: Ra-Ma, 2015.

[10]"UML 2.5", *Omg.org*, 2017. [Online]. Available: http://www.omg.org/spec/UML/2.5/PDF/.

[11]"Package Diagram | Enterprise Architect User Guide", *Sparxsystems.com.au*, 2017. [Online]. Available: http://www.sparxsystems.com.au/enterprise_architect_user_guide/13.0/model_domains/packagediagram.html.

[12] G. Booch, J. Rumbaugh, I. Jacobson., *El Lenguaje Unificado de Modelado*, Addison Wesley Iberoamericana, 1999.

[13] E. Gamma et al., *Patrones de diseño. Elementos de software orientado a elementos reutilizable*, 1a ed. Núñez de Balboa, Madrid: PEARSON EDUCACIÓN SA, 2003

[14]C. Larman, *Applying UML and patterns*. Upper Saddle River (New Jersey): Prentice-Hall, 2005.

[15] *Netbeans.org*, 2017. [Online]. Available: https://netbeans.org/downloads/8.1/?pagelang=pt_BR

[16]"Java SE Development Kit 8 - Downloads", *Oracle.com*, 2017. [Online]. Available: http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html

12. APÉNDICE A. API del código desarrollado a lo largo del proyecto

El API del código que se desarrolló para este proyecto se encuentra disponible en la siguiente liga:

..\PTUML4TeX_v2_Diego_Vazquez_Alvarez_210203597.zip\PTUML4TeX_v2_DVA\dist\javadoc

13. APÉNDICE B. Entregable: Manual del usuario

1. Requerimientos de la aplicación.

Para ejecutar la aplicación se requieren de los siguientes elementos.
a) Ambiente de ejecución JAVA 6 o mayor. Se puede obtener en la siguiente liga https://www.java.com/

b) Se recomienda utilizar equipos con memoria RAM de 1 GB.

2. Ejecución de la aplicación.

La aplicación UML4Tex ya viene empaquetado en un paquete .JAR listo para su ejecución. Para ejecutar la aplicación se deben realizar los siguientes pasos.

2.1. Desde interfaz gráfica del sistema:

a) Hacer doble clic en el archivo UML4Tex_vx.x.JAR, o bien, posicionarse en el icono de la aplicación y oprimir el botón derecho para que aparezca la opción Abrir (ver Figura 25).

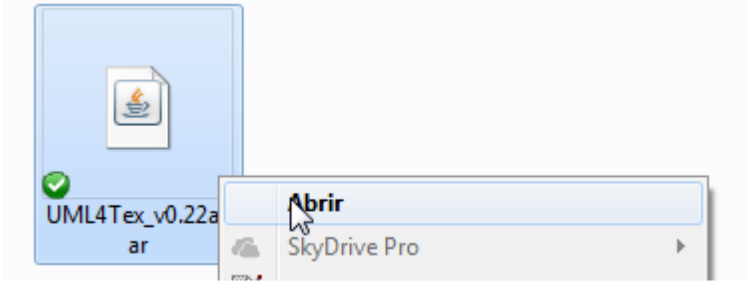


Figura 25. Se puede ejecutar la aplicación con el menú abrir.

2.2. Desde línea de comandos:

a) Abrir una consola
b) Posicionarse en el directorio del archivo JAR de la aplicación
c) Escribir en la línea de comando: java -jar UML4Tex_vx.x.JAR

3. Interfaz gráfica

La aplicación tiene la siguiente interfaz gráfica (Ver Figura 26).

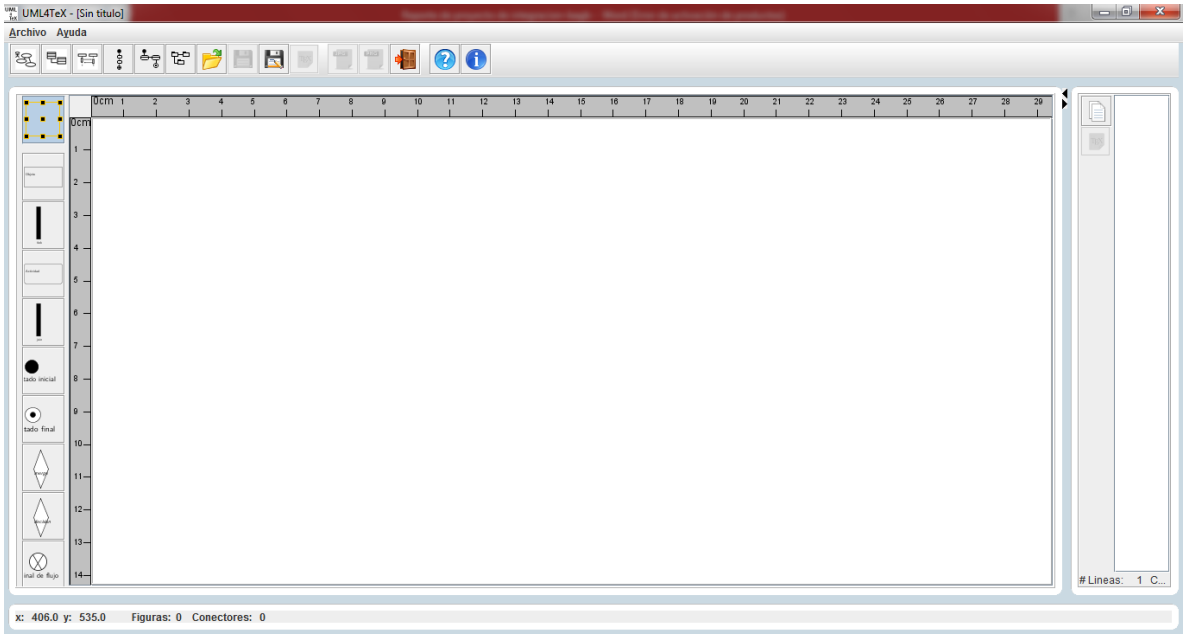


Figura 26. Interfaz gráfica de UML4TeX

3.1. Componentes de la interfaz gráfica

3.1.1. Menús

La interfaz gráfica dispone de menús desplegables a los que se accede mediante la barra de menús situada en la parte superior de la ventana de la interfaz (ver Figura 27).

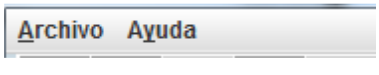


Figura 27. Barra de menús.

3.1.1.1. Para utilizar un menú.

a) En la barra de menús, hacer clic, con el botón izquierdo del ratón, en el nombre de un menú para visualizar la lista de opciones.

b) Pulse una opción o bien utilice flecha, para desplazarse por la lista y, a continuación, pulse la tecla Intro-enter.

3.1.1.2. Menú Archivo.

Este menú incluye las siguientes opciones, las cuales son las acciones principales que puede ejecutar la aplicación:

Icono	Nombre	Descripción
	Nuevo	
	(Nuevo) Diagrama de caso de uso	Crea un nuevo diagrama UML de Casos de Uso
	(Nuevo) Diagrama de clases	Crea un nuevo diagrama UML de Clases
	(Nuevo) Diagrama de secuencia	Crea un nuevo diagrama UML de Secuencia
	(Nuevo) Diagrama de Actividad	Crea un nuevo diagrama UML de Actividad
	(Nuevo) Diagrama de Estado	Crea un nuevo diagrama UML de Estado
	(Nuevo) Diagrama de Paquetes	Crea un nuevo diagrama UML de Paquetes
	Abrir	Abre un diagrama UML (existente)
	Guardar	Guarda el diagrama actual UML
	Guardar como…	Guarda el diagrama con otro nombre
	Exportar como código PST	Exporta el diagrama a un archivo .tex como macros PSTricks
	Exportar diagrama como	
	Exportar a imagen PNG	Exporta el diagrama como una imagen formato PNG
	Exportar a imagen JPG	Exporta el diagrama como una imagen formato JPG
	Salir	Sale de la aplicación

3.1.1.3. Menú Ayuda.

Este menú incluye las siguientes opciones, las cuales muestran información general de la aplicación e incluye un panel de ayuda con la descripción de uso de las acciones que puede realizar la aplicación.

Icono	Nombre	Descripción
	Acerca de…	Muestra una ventana con la información principal de la aplicación.
	Mostrar ayuda	Muestra un panel de ayuda con la descripción de uso de las principales acciones que puede realizar la aplicación.

3.1.2 Barra de herramientas

Esta barra de herramientas agrupa los botones que representan las acciones que puede realizar la aplicación. Estas acciones son las mismas que se describen en los puntos **3.1.1.2** y **3.1.1.3** de esta sección (Ver Figura 28).

Figura 28 Barra de herramientas de UML4TeX v2

3.1.3. Panel de dibujo.

En esta área de la aplicación el usuario puede dibujar los diagramas UML (ver Figura 29). El panel de trabajo se compone de una barra de herramientas para elegir los artefactos UML a insertar y esos dependen del tipo de diagrama elegido.

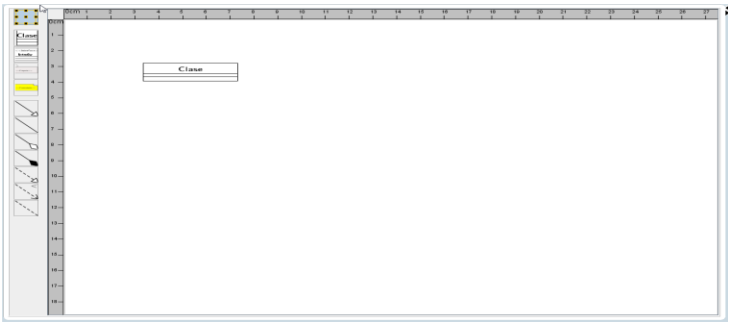
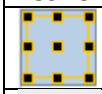
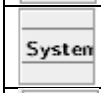


Figura 29. Área de Trabajo de la aplicación

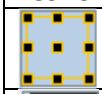
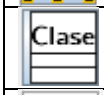
3.1.3.1. Barra de herramientas de Diagrama de Caso de Uso

Las figuras y relaciones soportadas por esta barra de herramientas son las siguientes:

FIGURAS Y CONECTORES		
Icono	**Nombre**	**Descripción**
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Actor	Dibuja una figura representando el Artefacto UML Actor.
	Dibujar Caso de Uso	Dibuja una figura representando el artefacto UML Caso de Uso
	Dibujar Borde de Sistema	Dibuja una figura representando el artefacto UML los límites del sistema.
	Dibujar Nota	Dibuja una figura representando una Nota
	Dibujar Relación de Comunicación	Dibuja un conector que representa una relación de comunicación.
	Dibujar Relación de Extensión	Dibuja un conector que representa una relación de extensión
	Dibujar Relación de Inclusión	Dibuja un conector que representa una relación de inclusión.
	Dibujar Relación de Generalización	Dibuja un conector que representa una relación de generalización.
	Dibujar Conector de Nota	Dibuja un conector de figuras de Nota

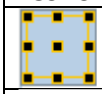
3.1.3.2. Barra de herramientas de diagrama de Clases

Las figuras y relaciones soportadas por esta barra de herramientas son las siguientes.

FIGURAS Y CONECTORES		
Icono	**Nombre**	**Descripción**
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Clase	Dibuja una figura representando el Artefacto UML Clase.
	Dibujar Interfaz	Dibuja una figura representando el artefacto UML Interfaz
	Dibujar Paquete	Dibuja una figura representando el artefacto UML Paquete.
	Dibujar Nota	Dibuja una figura representando una Nota
	Dibujar Relación de Generalización	Dibuja un conector que representa una relación de generalización de clases (herencia)
	Dibujar Relación de Asociación de Clases	Dibuja un conector que representa una relación de asociación de clases.
	Dibujar Relación de agregación	Dibuja un conector que representa una relación de agregación entre clases.
	Dibujar Relación de composición	Dibuja un conector que representa una relación de composición entre clases.
	Dibuja una Relación de Realización de interfaz	Dibuja un conector que representa una relación de realización de interfaz.
	Dibuja una Relación de dependencia.	Dibuja un conector que representa una relación de dependencia entre clases.
	Dibujar conector de Nota	Dibuja un conector de figuras de Nota

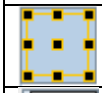
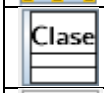
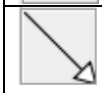
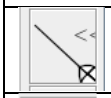
3.1.3.3. Barra de herramientas de diagrama de Secuencia

Las figuras y relaciones soportadas por esta barra de herramientas son las siguientes.

FIGURAS Y CONECTORES		
Icono	**Nombre**	**Descripción**
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Objeto	Dibuja una figura representando el Artefacto UML Objeto
	Dibujar Nota	Dibuja una figura representando una Nota
	Dibuja una mensaje petición	Dibuja un conector que representa un mensaje de petición
	Dibuja un mensaje de respuesta	Dibuja un conector que representa un mensaje de respuesta.
	Dibujar conector de Nota	Dibuja un conector de figuras de Nota

3.1.3.3. Barra de herramientas de diagrama de Paquetes

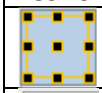
Las figuras y relaciones soportadas por esta barra de herramientas son las siguientes.

FIGURAS Y CONECTORES		
Icono	**Nombre**	**Descripción**
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Clase	Dibuja una figura representando el Artefacto UML Clase.
	Dibujar Interfaz	Dibuja una figura representando el artefacto UML Interfaz
	Dibujar Paquete	Dibuja una figura representando el artefacto UML Paquete.
	Dibujar Paquete Compuesto	Dibuja una figura representando el artefacto UML Paquete Compuesto
	Dibujar Nota	Dibuja una figura representando una Nota
	Dibujar Relación de Generalización	Dibuja un conector que representa una relación de generalización de clases (herencia)
	Dibujar un Conector Nesting	Dibuja un conector Nesting que representa anidamientos entre paquetes
	Dibujar Relación de Asociación de Clases	Dibuja un conector que representa una relación de asociación de clases.
	Dibujar Relación de agregación	Dibuja un conector que representa una relación de agregación entre clases.
	Dibujar Relación de composición	Dibuja un conector que representa una relación de composición entre clases.
	Dibuja una Relación de Realización de interfaz	Dibuja un conector que representa una relación de realización de interfaz.
	Dibuja una Relación de dependencia.	Dibuja un conector que representa una relación de dependencia entre clases.
	Dibuja una relación de importación	Dibuja un conector que representa una importación entre paquetes
	Dibuja una relación de mezcla (Merge)	Dibuja una relación que representa la mezcla de paquetes

	Dibujar conector de Nota	Dibuja un conector de figuras de Nota

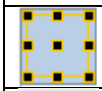
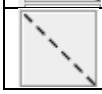
3.1.3.4. Barra de herramientas de diagrama de Actividad

Las figuras y relaciones soportadas por esta barra de herramientas son las siguientes.

FIGURAS Y CONECTORES		
Icono	**Nombre**	**Descripción**
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Objeto	Dibuja una figura representando el artefacto UML Objeto
	Dibujar Fork	Dibuja una figura representando el artefacto UML Fork
	Dibjar Actividad	Dibuja una figura representando el artefacto UML de Actividad
	Dibujar Join	Dibuja una figura representando el artefacto UML Interfaz
	Dibujar Estado Inicial	Dibuja una figura representando el artefacto UML Estado Inicial
	Dibujar Estado Final	Dibuja una figura representando el artefacto UML Estado Final
	Dibujar Merge	Dibuja una figura representando el artefacto UML Merge (mezcla)
	Dibujar Decision	Dibuja una figura representando el artefacto UML Decisión
	Dibujar final de flujo	Dibuja una figura representando el artefacto UML Final de Flujo
	Dibujar Nota	Dibuja una figura representando el artefacto UML Nota
	Dibujar Flujo de Control	Dibuja un conector que representa un Flujo de Control
	Dibujar Flujo de Objeto	Dibuja un conector que representa un Flujo de Objeto
	Dibujar Relación de Excepción	Dibuja un conector que representa una Relación de Excepción
	Dibujar conector de Nota	Dibuja un conector de figuras de Nota

3.1.3.5. Barra de herramientas de diagrama de Estado

Las figuras y relaciones soportadas por esta barra de herramientas son las siguientes.

FIGURAS Y CONECTORES		
Icono	**Nombre**	**Descripción**
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Objeto	Dibuja una figura representando el artefacto UML Objeto
	Dibujar Fork	Dibuja una figura representando el artefacto UML Fork
	Dibujar Estado	Dibuja una figura representando el artefacto UML de Estado
	Dibujar Estado Compuesto	Dibuja una figura representando el artefacto UML de Estado Compuesto
	Dibujar Join	Dibuja una figura representando el artefacto UML Interfaz
	Dibujar Estado Inicial	Dibuja una figura representando el artefacto UML Estado Inicial
		Dibuja una figura representando el artefacto UML Estado Final
	Dibujar Merge	Dibuja una figura representando el artefacto UML Merge (mezcla)
	Dibujar Decision	Dibuja una figura representando el artefacto UML Decisión
	Dibujar Nota	Dibuja una figura representando el artefacto UML Nota
	Dibujar Flujo de Control	Dibuja un conector que representa un Flujo de Control
	Dibujar Flujo de Objeto	Dibuja un conector que representa un Flujo de Objeto
	Dibujar conector de Nota	Dibuja un conector de figuras de Nota

Uso del sistema

1. Crear diagrama

1. El usuario solicita a la aplicación la realización de una operación de creación de nuevo diagrama.
    a. (En la pantalla inicial **[Elegir el diagrama que se desea crear]**) (Ver Figura 30).
    b. (Elegir menú **Archivo → Nuevo... → [Elegir el diagrama que se desea crear]**) (Ver Figura 31).
    c. (De la barra de herramientas **[Elegir el diagrama que se desea crear]**) (Ver Figura 32).

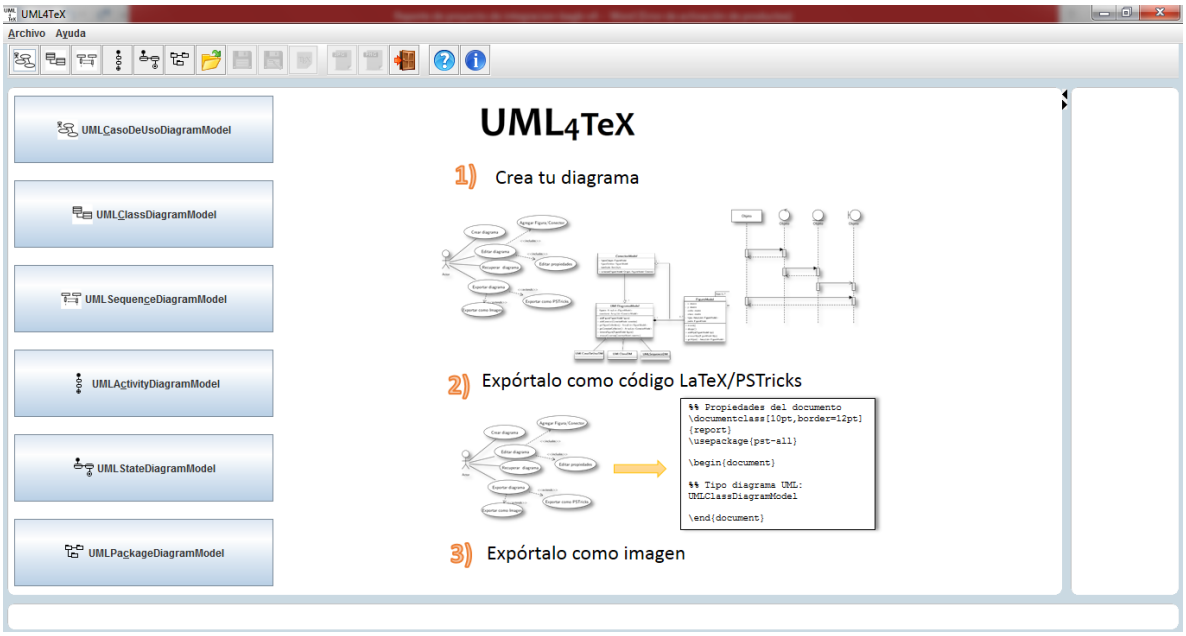


Figura 30. Crear un nuevo diagrama según la forma 1-a

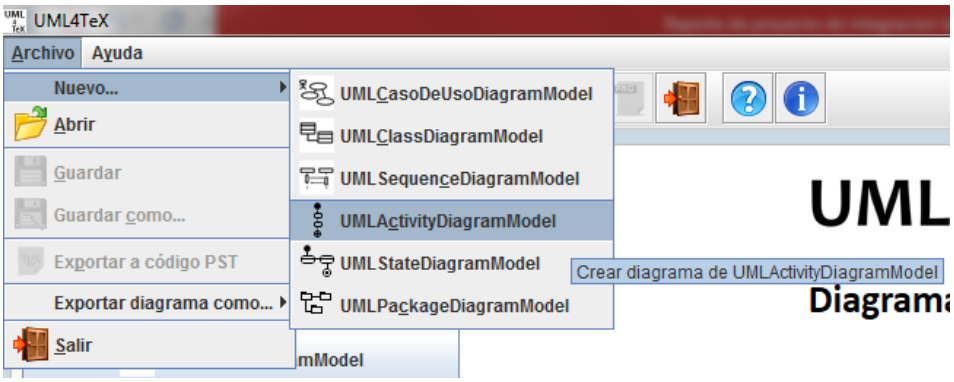


Figura 31. Crear un nuevo diagrama según la forma 1-b

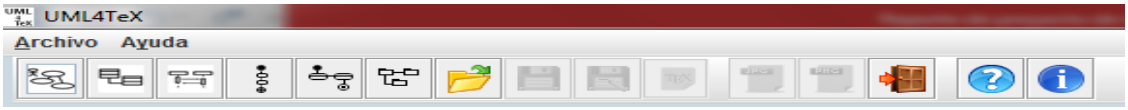


Figura 32. Crear un nuevo diagrama según la forma 1-c

2. La aplicación muestra las diferentes vistas del modelo (Área de trabajo, panel de código PSTricks, Barra de estado) (ver Figura 33).

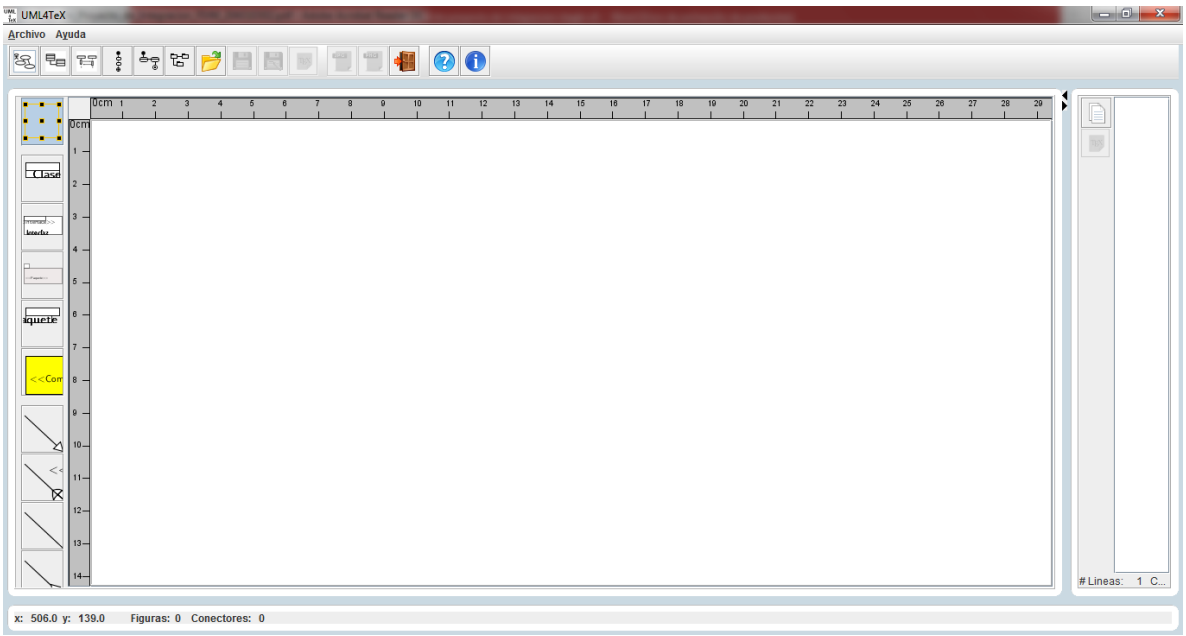


Figura 33. Vistas de la Interfaz de UML4TeX v2

2. Recuperar diagrama

1. El usuario solicita a la aplicación recuperar (abrir) un diagrama previamente guardado por la aplicación
    a. (Elegir menú **Archivo → Abrir → [Elegir el diagrama que se desea abrir]** (Ver Figura 34 y Figura 36)
    b. (De la barra de herramientas seleccionar el botón **Abrir**) (Ver Figura 34)

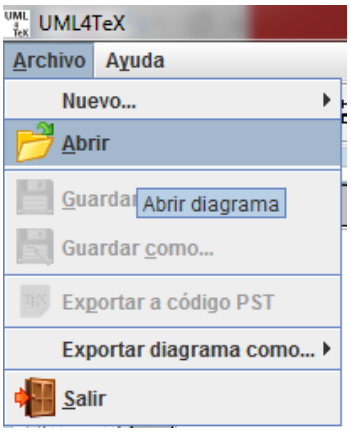


Figura 34. Recuperar un diagrama según la forma 1a.

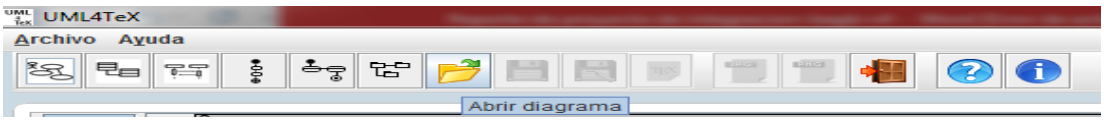


Figura 35. Recuperar un diagrama según la forma 1b.

2. La aplicación muestra un cuadro de dialogo con un explorador de archivos para que el usuario pueda elegir el archivo que desea abrir (ver Figura 36).

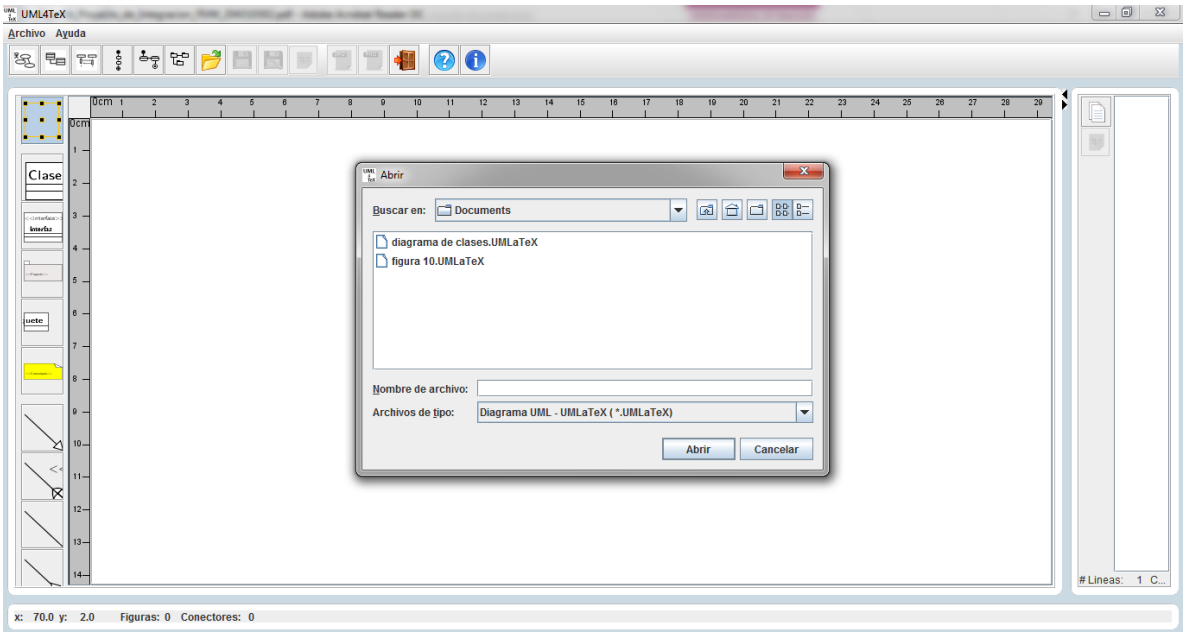


Figura 36. Recuperar un diagrama

La aplicación muestra el archivo recuperado en el área de trabajo.

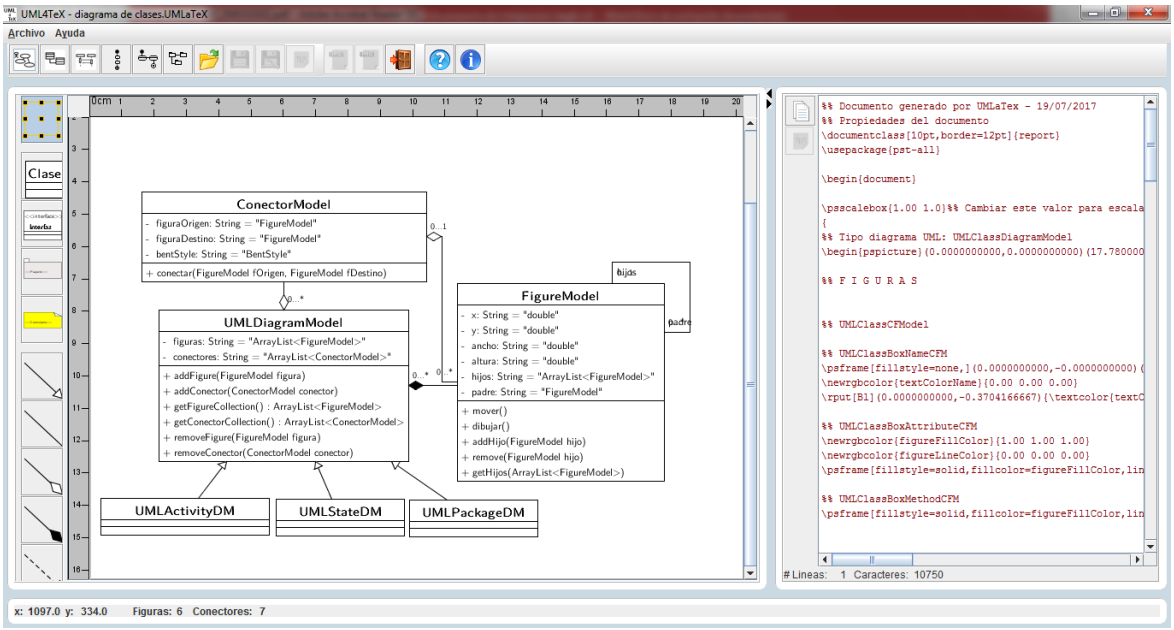


Figura 37. Diagrama recuperado

3. Editar diagrama: Mover Figuras

1. El usuario elige el artefacto para seleccionar figuras UML (figura o conector) (ver Figura 38).
2. El usuario selecciona la figura que desea arrastrar (ver Figura 39)
3. El usuario mueve la figura libremente a través del área de trabajo

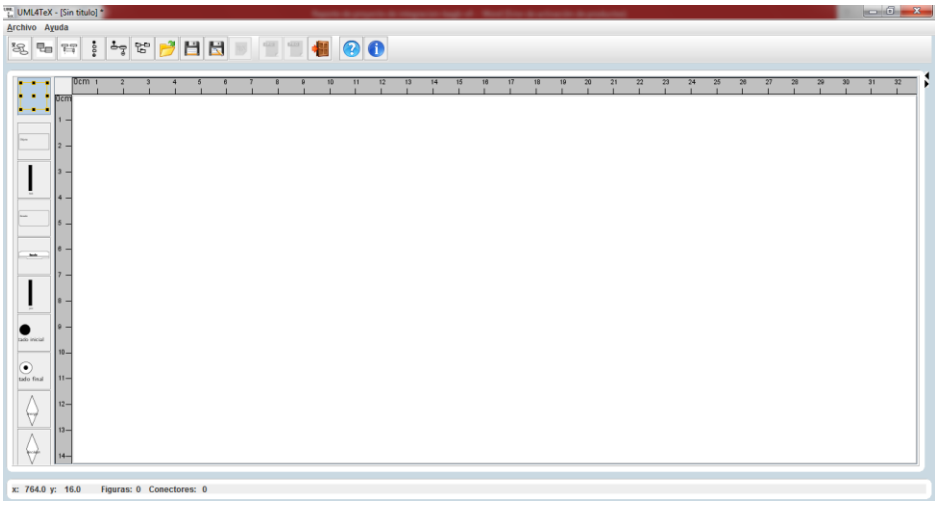


Figura 38. Mover figuras

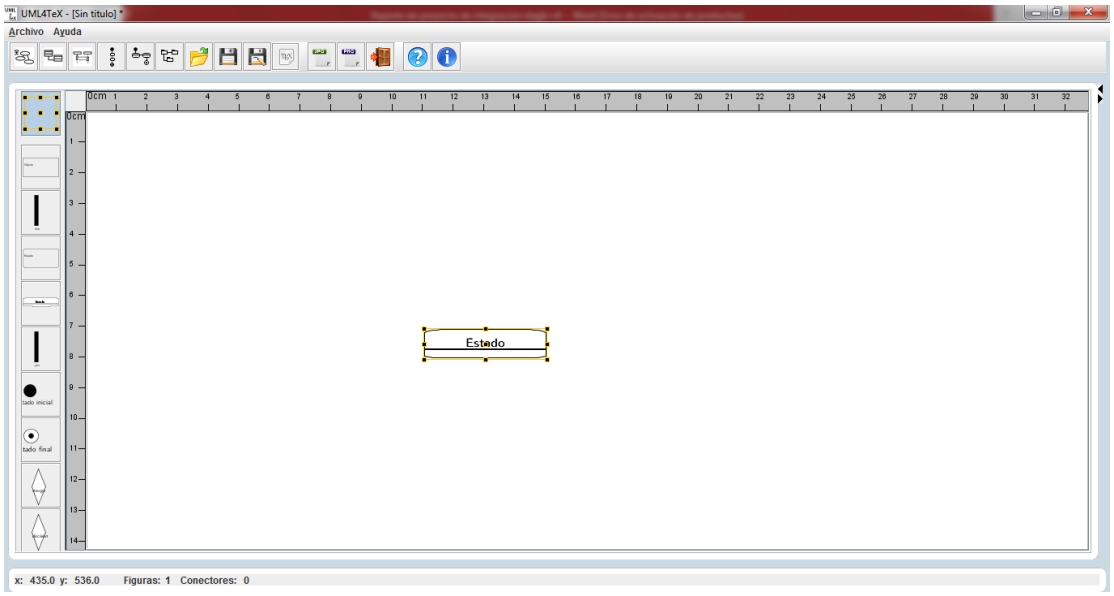


Figura 39. Mover figuras

4. Agregar Figura/Conector

1. El usuario elige de la barra de herramientas un artefacto UML (figura/conector).
2. El usuario se posiciona en el área de trabajo y presiona el botón izquierdo del mouse y la figura se agregará al área de trabajo

Para agregar un conector es necesario que existan al menos dos figuras en el área de trabajo

1. El usuario selecciona el conector que desea agregar
2. El usuario posiciona el cursor en la figura de la que desee que salga el conector
3. El usuario arrastra el conector hasta la otra figura que desee conectar(ver Figura 40)

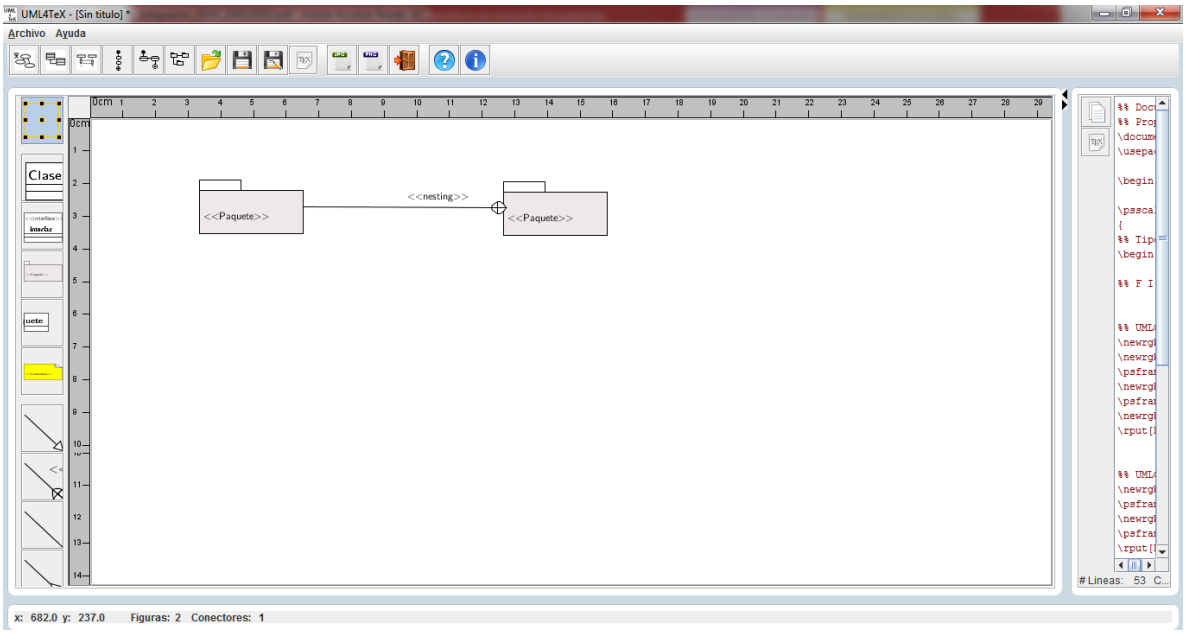


Figura 40. Agregar figuras y conectores

5. Editar propiedades

1. El usuario elige un artefacto UML que se encuentra en el área de trabajo y lo selecciona (ver Figura 41).
2. Dar doble clic sobre la figura seleccionada.
3. La aplicación muestra el formulario asociado a dicha figura para editar sus propiedades.
4. El usuario edita las propiedades que desee y acepta los cambios, y estos se ven en la figura.

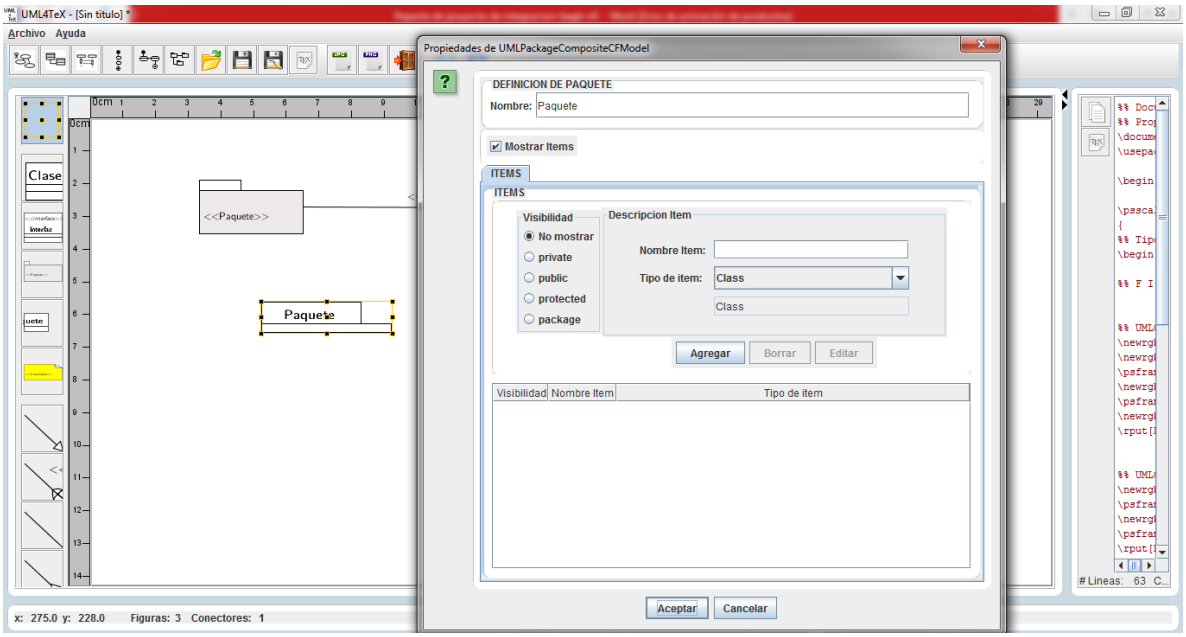


Figura 41. Editar propiedades

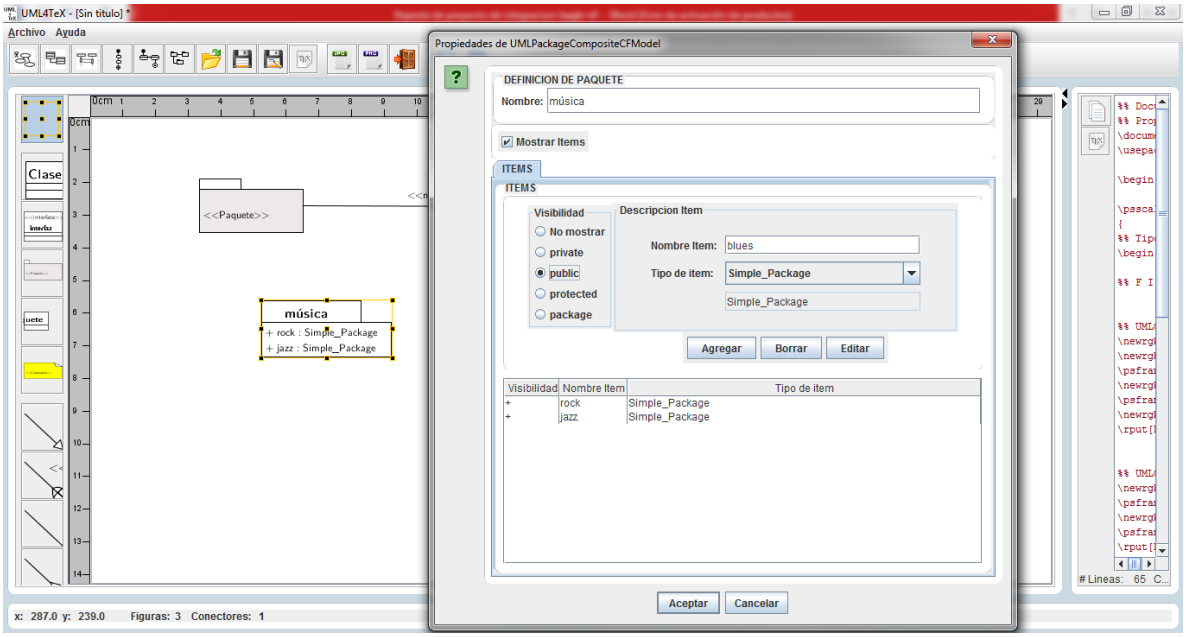


Figura 42. Editar propiedades

6. Borrar Figura/Conector

1. El usuario elige un artefacto UML que se encuentra en el área de trabajo y lo selecciona.
2. El usuario pide a la aplicación borrar el artefacto UML seleccionado. Esto se logra oprimiendo la tecla **SUPRIMIR.**

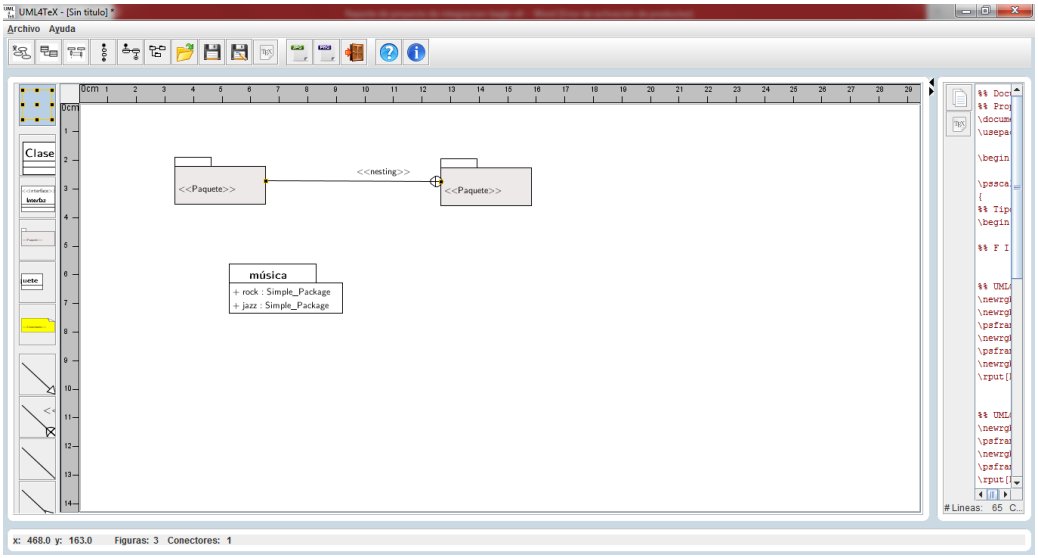


Figura 43. Eliminar un elemento (figura/conector)

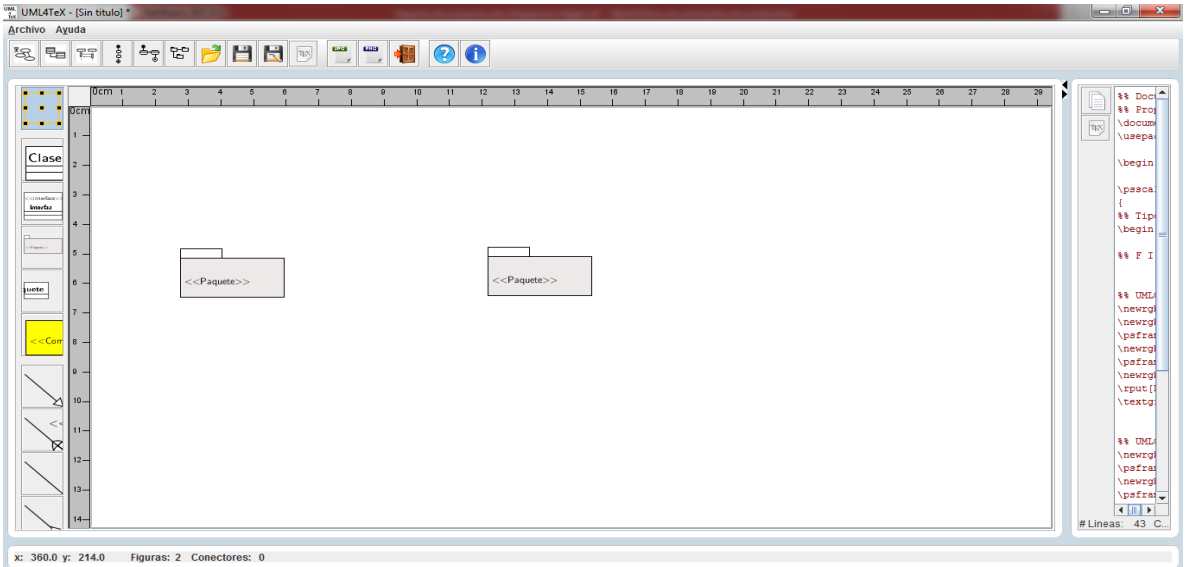


Figura 44. Eliminar un elemento (figura/conector)

7. Guardar Diagrama

1. El usuario solicita a la aplicación que se ejecute una operación de salvado.
   a. Esto lo puede hacer presionando el botón de la barra de herramientas Guardar o Guardar Como…(ver Figura 45)
   b. También pude hacerlo seleccionando el menú **Archivo □□Guardar/Guardar Como…** (ver Figura 46)
2. La aplicación mostrará un cuadro de dialogo con un sistema de archivos en donde el usuario podrá seleccionar la ubicación y el nombre con el que se guardará el archivo.(ver Figura 47).

Figura 45. Guardar un diagrama

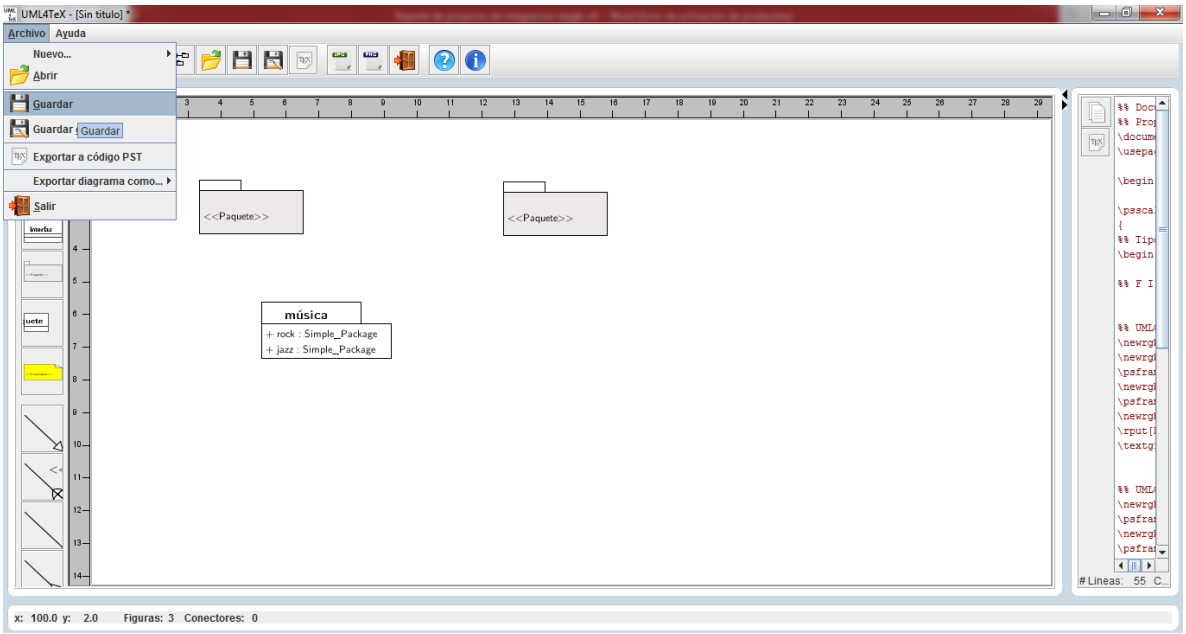


Figura 46. Guardar un diagrama

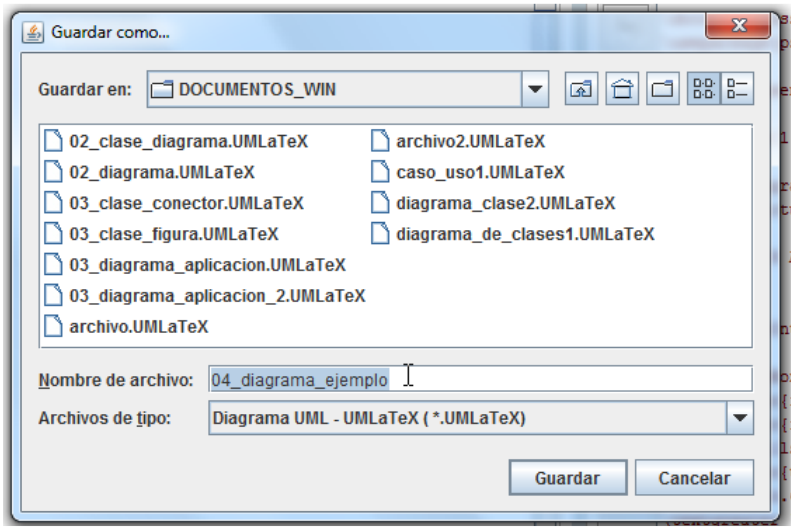


Figura 47. Guardar un diagrama

8. Exportar Diagrama como PSTricks

1. El usuario solicita a la aplicación que se ejecute la exportación como macros de PSTricks. Esto se logra de las siguientes maneras:
    a. Presionando el Botón correspondiente de la barra de herramientas (ver Figura 48).
    b. Accediendo al menú **Archivo → Exportar a código PST** (ver Figura 49)
    c. En el panel de vista de código PST también está habilitada la el botón para exportar el diagrama (ver Figura 50)
    d. La aplicación abrirá un diálogo con sistema de archivos donde elegirá la ubicación del archivo *.tex de salida (ver Figura 51).
2. La aplicación avisará el éxito de la operación (ver Figura 52)

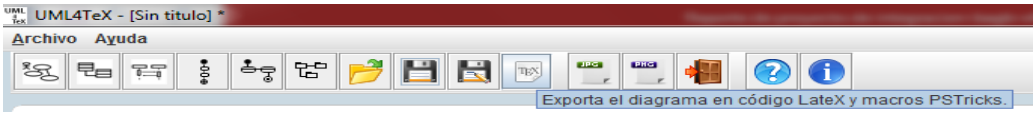


Figura 48. Exportar un diagrama a PSTricks

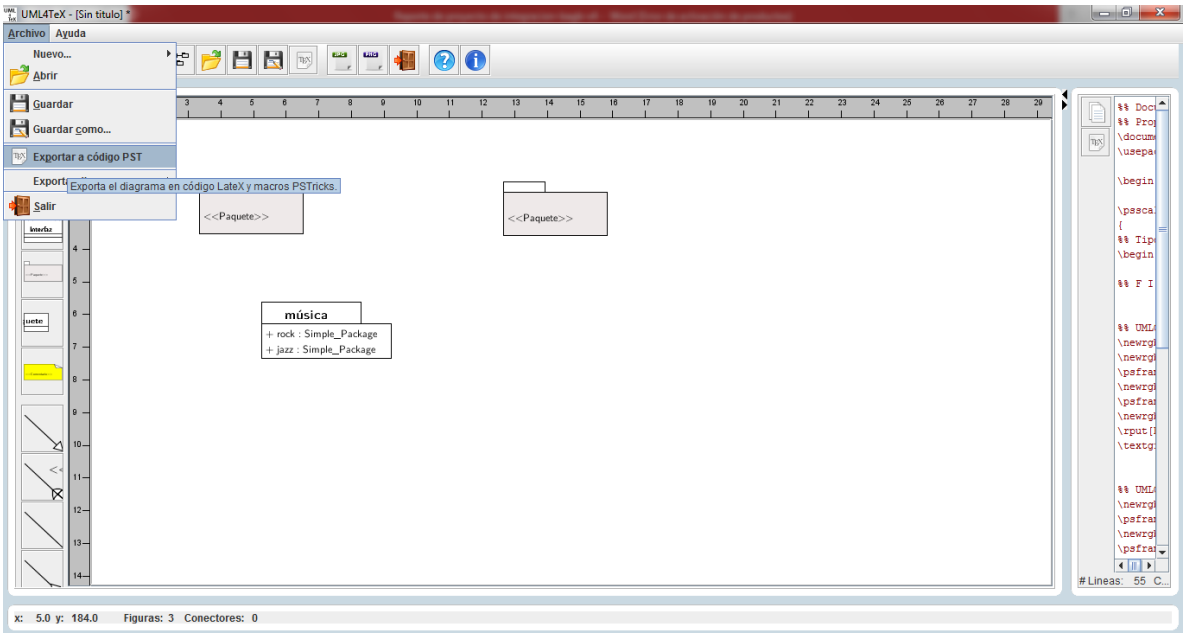


Figura 49. Exportar un diagrama a PSTricks

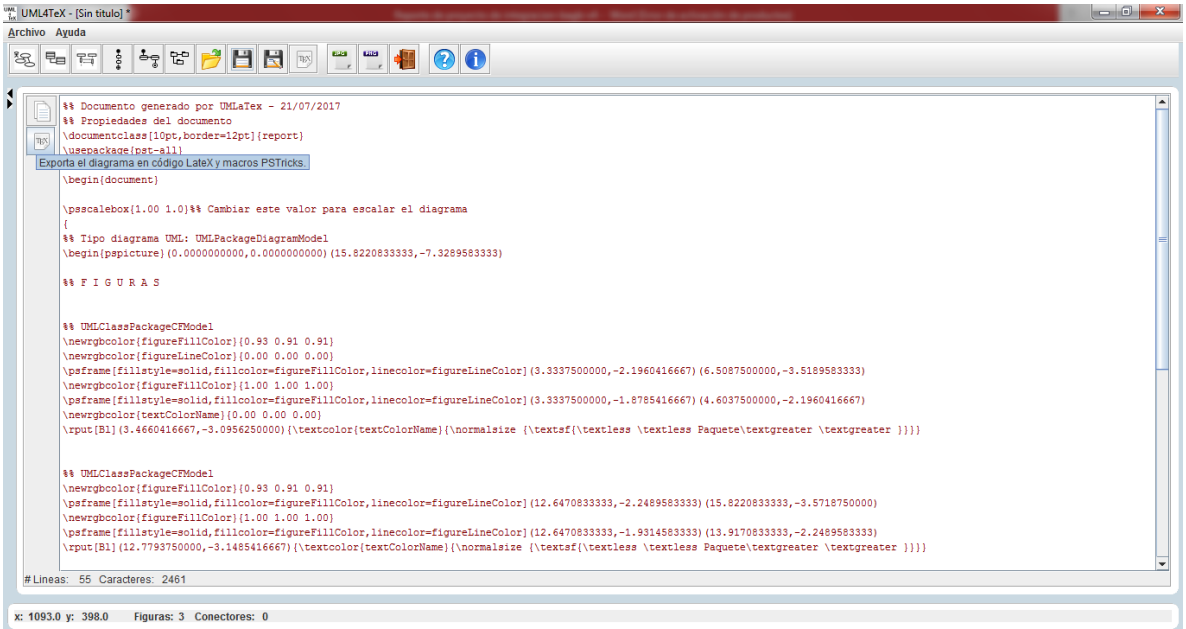


Figura 50. Exportar un diagrama a PSTricks

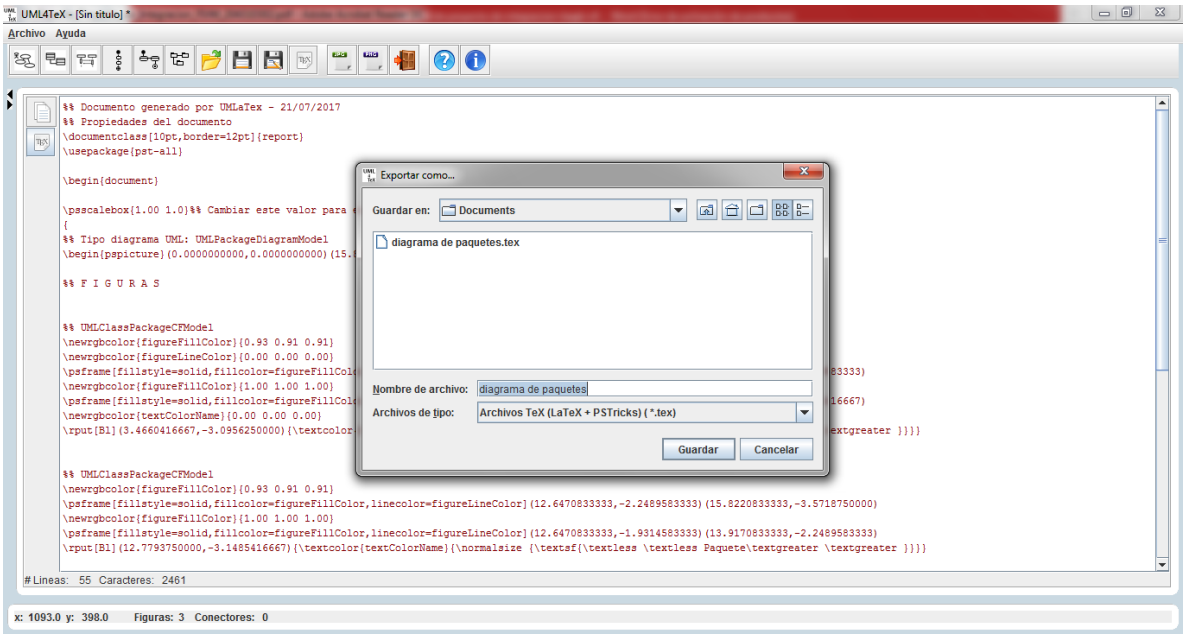


Figura 51. Exportar un diagrama a PSTricks

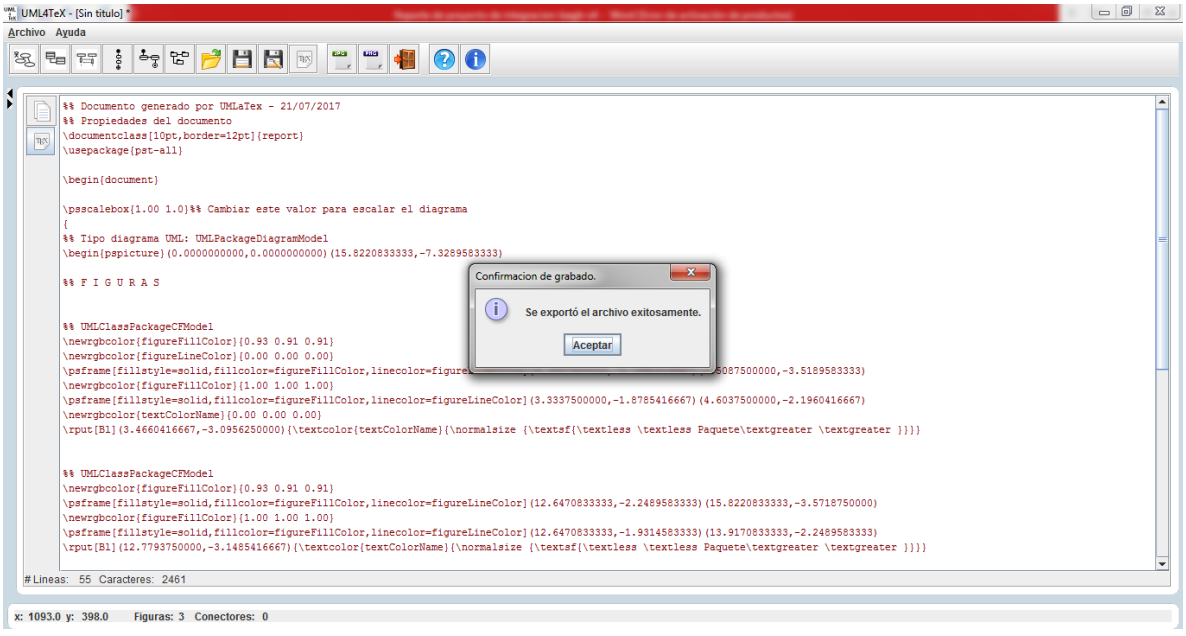


Figura 52. Exportar un diagrama a PSTricks

9. Exportar Diagramas en formato Imagen (PNG/JPG)

1. El usuario solicita a la aplicación que se ejecute una operación de exportar como imagen (ver Figura 53). El usuario puede elegir cualquiera de estas opciones de la siguiente manera:

a. Accediendo a las opciones de la barra de herramientas de la aplicación
   I. para exportar el diagrama en formato JPG
   II. para exportar el diagrama en formato PNG

b. Eligiendo las opciones del menú de la aplicación:
   I. **Archivo → Exportar diagrama como… → PNG**
   II. **Archivo → Exportar diagrama como… → JPG**

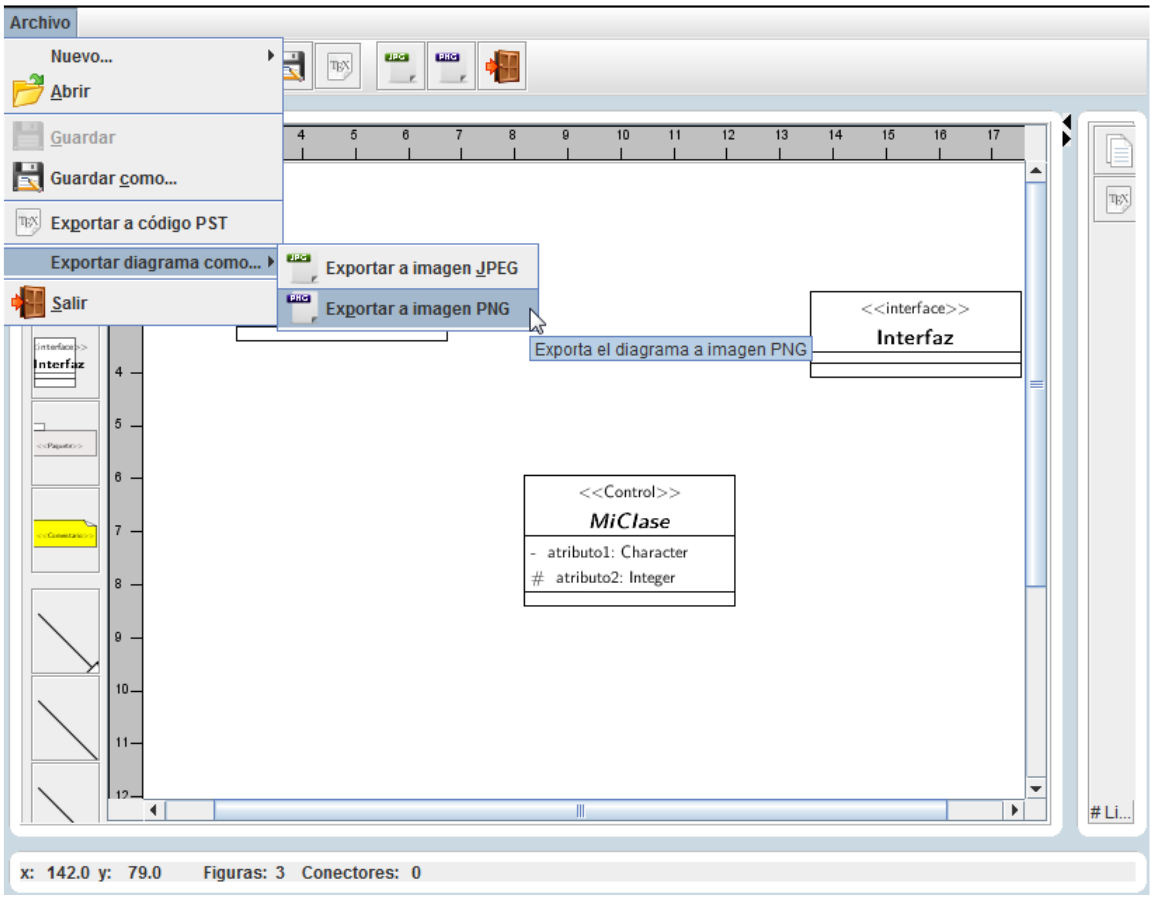


Figura 53. Exportar un diagrama a PNG

La aplicación pregunta al cliente dónde guardar el archivo de imagen de salida (ver Figura 54)

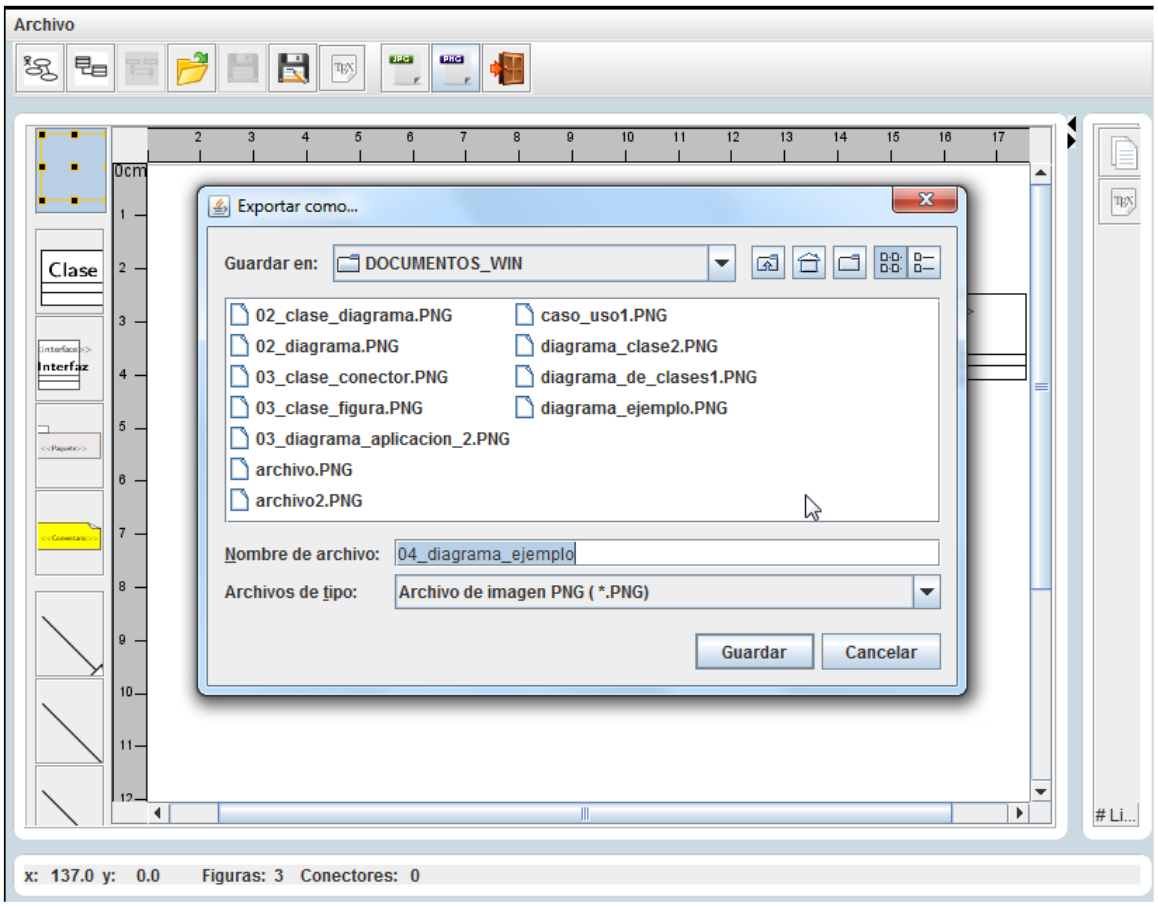


Figura 54. Exportar un diagrama a PNG

10. Mostrar ayuda.

Para mostrar la ayuda de la aplicación seguir los siguientes pasos.
1. Ir al menú y seleccionar los elementos **Ayuda → Mostrar Ayuda** (ver Figura 55)**.**
Alternativamente puede presionar el botón de la barra de herramientas.

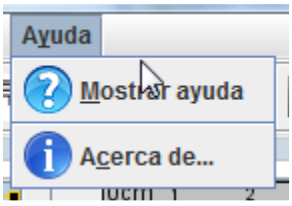


Figura 55. Mostrar ayuda

2. La aplicación abrirá una nueva ventana con el contenido de la ayuda de la aplicación (ver Figura 56).

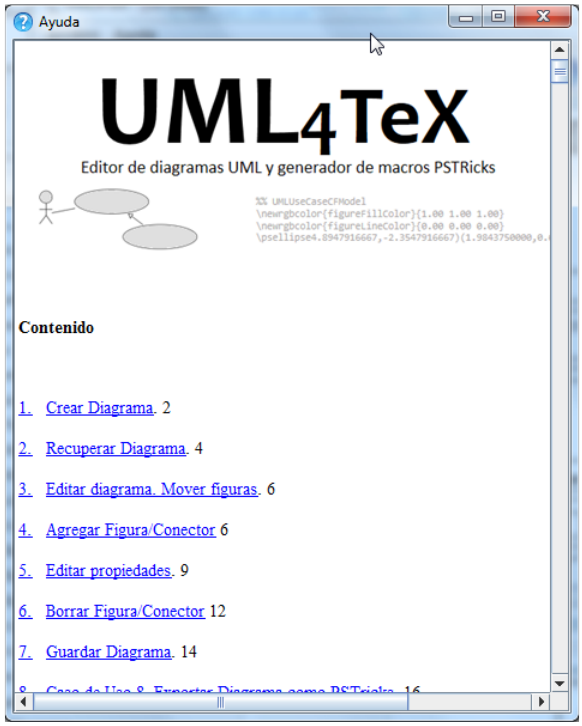


Figura 55. Panel de ayuda