

*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*

**“Editor gráfico de diagramas UML con la capacidad de generar macros  
PSTricks”**

Villalobos Martínez Fidel - 204310302

Asesora: Dra. González Beltrán Beatriz Adriana

Trimestre 14-I

Fecha de entrega: 25 de marzo de 2014

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.

Yo, **Fidel Villalobos Martínez**, 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.

## **RESUMEN**

Este proyecto presenta UML4TeX, una aplicación desarrollada en Java que nace de la necesidad de crear un editor gráfico que permita generar y editar diagramas UML y que a su vez los traduzca en macros PSTricks, las cuales se pueden integrar en documentos LaTeX. De esta forma, la aplicación apoya al usuario a convertir los diagramas UML en macros y así mitigar la complejidad de aprender a codificar figuras complejas en el lenguaje LaTeX.

La aplicación fue desarrollada utilizando como metodología de desarrollo de software el Proceso Racional Unificado (RUP). Además, la aplicación fue diseñada implementando el patrón Modelo-Vista-Controlador (MVC) permitiendo así un desarrollo modular y extensible. Adicionalmente, UML4Tex está construido en el lenguaje de programación Java lo cual le permite la particularidad de ser portable, es decir, permite que sea multiplataforma.

UML4TeX implementó tres de los trece diagramas UML, por lo que pueden desarrollarse más diagramas en el futuro.

## TABLA DE CONTENIDO

RESUMEN.....	2
TABLA DE CONTENIDO .....	3
INDICE DE FIGURAS .....	8
INDICE DE DIAGRAMAS .....	11
1. Introducción.....	12
2. Antecedentes.....	13
2.1. Referencias internas.....	13
2.2. Referencias Externas .....	13
3. Justificación.....	15
4. Objetivos.....	16
4.1. Objetivo general .....	16
4.2. Objetivos específicos.....	16
5. Marco teórico.....	17
5.1. Editor de diagramas .....	17
5.1.1. Editor de diagramas UML .....	18
5.2. UML .....	18
5.2.1. Diagrama de clases .....	19
5.2.1.1. Clase.....	19
5.2.1.1.1. Atributos .....	20
5.2.1.1.2. Métodos u operaciones .....	20
5.2.1.1.3. Visibilidad de los atributos y los métodos de la clase .....	21
5.2.1.2. Relaciones .....	21
5.2.1.2.1. Herencia (Especialización/Generalización).....	21
5.2.1.2.2. Composición .....	22
5.2.1.2.3. Agregación.....	22

5.2.1.2.4.	Dependencia o Instanciación .....	23
5.2.1.2.5.	Asociación .....	23
5.2.2.	Diagrama de caso de uso .....	24
5.2.2.1.	Actor.....	24
5.2.2.2.	Caso de uso.....	25
5.2.2.3.	Relaciones .....	25
5.2.2.3.1.	Comunicación .....	25
5.2.2.3.2.	Inclusión.....	25
5.2.2.3.3.	Extensión .....	26
5.2.2.3.4.	Generalización .....	26
5.2.3.	Diagrama de secuencia .....	27
5.2.3.1.	Tipos de mensajes .....	27
5.2.3.1.1.	Mensajes síncronos .....	27
5.2.3.1.2.	Mensajes asíncronos .....	27
5.3.	PSTricks .....	28
6.	Desarrollo del proyecto .....	29
6.1.	Metodología empleada en el desarrollo del proyecto.....	29
6.2.	Diseño del sistema.....	29
6.2.1.	Documentación de Actores.....	29
6.2.2.	Casos de uso .....	30
6.2.2.1.	CU-01 Crear Diagrama .....	31
6.2.2.2.	CU-02 Recuperar Diagrama.....	32
6.2.2.3.	CU-03 Editar diagrama .....	33
6.2.2.4.	CU-04 Agregar Figura/Conector.....	34
6.2.2.5.	CU-05 Editar propiedades.....	35
6.2.2.6.	CU-06 Borrar Figura/Conector .....	36
6.2.2.5.	Editar propiedades.....	35

6.2.2.7.	CU-07 Guardar Diagrama .....	37
6.2.2.8.	CU-08 Exportar Diagrama .....	38
6.2.2.9.	CU-09 Exportar Diagrama como PSTricks.....	39
6.2.2.10.	CU-10 Exportar Diagrama como imagen .....	40
6.2.3.	Diagrama de clases .....	41
6.2.3.1.	Modelo del dominio del problema .....	41
6.2.3.1.1.	Clase Diagrama.....	42
6.2.3.1.2.	Clase Figura .....	43
6.2.3.1.3.	Clase Conector .....	44
6.2.3.1.4.	Identificación de relaciones .....	45
6.2.3.1.5.	Identificación de multiplicidad .....	45
6.2.3.2.	Vistas.....	47
6.2.3.2.1.	Vista 1. Representación gráfica del modelo .....	47
6.2.3.2.2.	Vista 2. Representación en forma de información de los elementos del modelo	49
6.2.3.2.3.	Vista 3. Representación en código PSTricks del modelo .....	50
6.2.3.3.	Módulo traductor a código PSTricks .....	50
6.2.3.4.	Módulo para implementar la persistencia .....	53
6.2.3.5.	Módulo para exportar archivos .....	53
6.2.3.6.	Interfaz de usuario.....	54
6.2.3.6.1.	Clase de la aplicación principal .....	56
6.2.3.6.2.	Clase Barra de Menú.....	56
6.2.3.6.3.	Clase Barra de Herramientas .....	57
6.2.3.6.4.	Diseño de las acciones realizadas por la aplicación.....	58
6.2.3.6.5.	Clase Área de Trabajo.....	59
6.2.3.7.	Controladores .....	62

6.2.3.7.1.	Clase UMLDiagramToolBar .....	62
6.2.3.7.2.	Clase Controlador de Figuras .....	63
6.2.3.7.3.	Clase Controlador de Conectores .....	64
6.2.3.7.4.	Clase para mover y arrastrar figuras .....	66
6.2.3.7.5.	Clase KeyModelController .....	66
6.2.4.	Arquitectura del sistema .....	68
6.2.5.	Comunicación entre bloques .....	70
6.2.6.	Uso del sistema.....	71
6.2.6.1.	Crear Diagrama .....	71
6.2.6.2.	Recuperar Diagrama.....	73
6.2.6.3.	Editar diagrama. Mover figuras .....	75
6.2.6.4.	Agregar Figura/Conector.....	75
6.2.6.5.	Editar propiedades.....	78
6.2.6.6.	Borrar Figura/Conector .....	81
6.2.6.7.	Guardar Diagrama .....	83
6.2.6.8.	Exportar Diagrama como PSTricks.....	85
6.2.6.9.	Exportar Diagramas en formato Imagen (PNG/JPG).....	87
6.3.	Hardware y software necesario.....	89
6.3.1.	Tecnología para el desarrollo de la aplicación .....	89
6.3.2.	Tecnología para la instalación y puesta en marcha de la aplicación .....	89
7.	Resultados.....	90
8.	Análisis y discusión de resultados .....	91
8.1.	Módulos de estructura lógica y de visualización de los diagramas UML .....	91
8.2.	Integración de los módulos de edición (manipulación gráfica) de los diagramas UML	92
8.2.1.	Módulo de manipulación gráfica del diagrama. Movimiento .....	93
8.2.2.	Módulo de manipulación gráfica del diagrama. Edición de los elementos del diagrama .....	93

8.2.2.1.	Inserción de elementos al modelo .....	93
8.2.2.2.	Edición de un elemento .....	94
8.2.2.3.	Eliminar un elemento .....	95
8.3.	Módulo de interfaz gráfica de usuario .....	96
8.4.	Módulo de persistencia de los diagramas .....	98
8.5.	Módulo de traducción de estructura lógica del diagrama UML cómo macros PSTricks .....	101
8.6.	Integración de los módulos de la aplicación.....	103
9.	Conclusiones.....	104
10.	Perspectivas del proyecto .....	105
11.	Referencias bibliográficas .....	106
APÉNDICE A.	Entregable: Listado del API del código fuente desarrollado .....	108
APÉNDICE B.	Entregable: Manual del usuario .....	116



## INDICE DE FIGURAS

Figura 1: Estructura de la organización de los distintos diagramas UML.....	19
Figura 2: Ejemplo de una clase en UML.....	20
Figura 3: Representación de las propiedades de un objeto en una clase UML. ....	20
Figura 4: Representación de los métodos u operaciones de un objeto en una clase UML...	21
Figura 5: Relación de Especialización/Generalización en UML.....	21
Figura 6: Relación de Composición en UML.....	22
Figura 7. Relación de Agregación en UML .....	22
Figura 8: Relación de Dependencia o Instanciación en UML.....	23
Figura 9: Relación de Asociación en UML.....	23
Figura 10. Multiplicidad de una asociación UML.....	24
Figura 11: Representación de un Actor en UML .....	24
Figura 12: Representación de un caso de uso en UML .....	25
Figura 13: Representación de una relación de comunicación en UML.....	25
Figura 14: Representación de la relación de inclusión de casos de uso en UML.....	26
Figura 15: Representación de la relación de extensión de casos de uso en UML.....	26
Figura 16: Representación de herencia de casos de uso/actores en UML.....	27
Figura 17: Ejemplo de codificación de macros PSTricks y la figura resultante para dibujar una un circulo. ....	28
Figura 18: Diagrama de Casos de uso para la aplicación UML4TeX.....	30
Figura 19: Ejemplos de artefactos UML. ....	41
Figura 20: Clase Diagrama.....	42
Figura 21: Clase Figura .....	43
Figura 22: Clase Conector .....	44
Figura 23: Características de un conector. ....	44
Figura 24: Ejemplo de la vista del modelo del diagrama en la aplicación. ....	48
Figura 25: Ejemplo de la vista que muestra la información básica del modelo de diagrama. ....	49
Figura 26: Interfaz de usuario para la aplicación. ....	55
Figura 27: Ejemplo de barra de menú Archivo.....	57
Figura 28. Ejemplo de un diseño de barra de herramientas de la aplicación. ....	58
Figura 29: Ejemplo del área de trabajo de la aplicación.....	60
Figura 30: Diagrama de la arquitectura funcional de la aplicación.....	68
Figura 31: UML4TeX, elegir nuevo diagrama.....	71
Figura 32: UML4TeX, área de trabajo mostrada tras crear un nuevo diagrama. A la izquierda, la barra de herramientas correspondiente al diagrama UML elegido. Al centro el área de trabajo con la Vista del diagrama. A la derecha la Vista de código PSTricks.....	72

Figura 33: UML4TeX. Abrir un diagrama .....	73
Figura 34:UML4TeX. Cuadro de diálogo para abrir un diagrama *.UMLaTeX .....	74
Figura 35: UML4TeX. Muestra diagrama recuperado tras abrir el archivo.....	74
Figura 36:UML4TeX. La herramienta de selección de figura siempre estará en la parte superior de la barra de herramientas.....	75
Figura 37: UML4TeX. El usuario puede elegir de la barra de herramientas el artefacto UML a agregar al diagrama.....	76
Figura 38:UML4TeX. La aplicación actualiza el modelo y las vistas al agregar una nueva figura.....	76
Figura 39: UML4TeX. El usuario puede elegir de la barra de herramientas el artefacto UML a agregar al diagrama.....	77
Figura 40:UML4TeX. La aplicación actualiza el modelo y las vistas al conectar dos figuras. ....	78
Figura 41. UML4TeX. Al elegir una figura se resaltarán sus bordes en color naranja. ....	79
Figura 42:UML4TeX. Cada artefacto UML (Figura/Conector) tiene un formulario destinado para editar sus propiedades. ....	80
Figura 43:UML4TeX. Formulario de propiedades para el artefacto UML Clase. ....	80
Figura 44:UML4TeX. Se actualiza el diagrama tras la edición de las propiedades de un artefacto UML. ....	80
Figura 45:UML4TeX. La selección de figuras/conectores se realiza con la herramienta de selección. ....	81
Figura 46: Se pueden eliminar los artefactos UML oprimiendo la tecla SUPRIMIR.....	82
Figura 47: UML4TeX. Las opciones de Guardar/Guardar Como... están disponibles en la barra de herramientas principal de la aplicación o en el Menú. ....	83
Figura 48: UML4TeX. Los diagramas generados por la aplicación son guardados en formato XML y con la extensión *.UMLaTeX.....	84
Figura 49: UML4TeX. La aplicación ofrece varios métodos para exportar los diagramas a macros PSTricks. ....	85
Figura 50: UML4TeX. Los diagramas pueden ser convertidos a macros PSTricks, que son integrables a documentos LaTeX .....	86
Figura 51:UML4TeX. La aplicación genera el archivo *.tex con las macros PSTricks del diagrama actual.....	86
Figura 52:UML4TeX. La aplicación ofrece la posibilidad de exportar el diagrama generado en formato de imagen JPG y PNG.....	87
Figura 53: UML4TeX. La aplicación realiza la operación de exportación en el formato indicado (PNG/JPG).....	88
Figura 54: UML4TeX. La aplicación integrando los módulos de estructura lógica (modelo) y de visualización (vistas) de los diagramas UML. La primera vista (panel central) es el renderizador del diagrama, mostrando la representación gráfica de éste. La segunda vista (panel de la derecha) es el diagrama como representación en macros PSTricks. Una tercera	

vista (panel de abajo) muestra una representación del modelo en función del número de figuras y conectores asociados a él.....	92
Figura 55. UML4TeX. El controlador de movimiento de figuras actualiza la posición de un artefacto UML dentro del diagrama. ....	93
Figura 56. UML4TeX. Selección de figura para agregar, la interfaz recibe los eventos relacionados para obtener el controlador asociado de la figura a insertar.....	94
Figura 57: UML4TeX. Edición de las propiedades de un elemento del diagrama. ....	95
Figura 58: UML4TeX. 1) Se selecciona el elemento Clase del diagrama para eliminar. 2) El controlador detecta que el usuario oprimió la tecla "Suprimir" y lo traduce en la eliminación de la figura del modelo. ....	96
Figura 59: Interfaz gráfica de usuario de UML4TeX. Cuenta con los paneles y menús y barras de herramientas a utilizar.....	97
Figura 60: Imagen de uno del Menú Archivo, que utiliza la aplicación. ....	98
Figura 61: Barra de herramientas de la aplicación. Cada icono describe la acción que puede realizar. ....	98
Figura 62. Diagrama de Caso de Uso a ser guardado con el módulo de persistencia. ....	99
Figura 63. La aplicación pregunta al usuario donde desea guardar el diagrama.....	99
Figura 64. Archivo *.UML4TeX generado por la aplicación .....	100
Figura 65: Contenido del archivo XML *.UMLaTeX .....	100
Figura 66. Representación gráfica del Diagrama de Caso de Uso. ....	101
Figura 67. Código LateX con macros PSTricks generado por la aplicación.....	102

## INDICE DE DIAGRAMAS

Diagrama 1: Diagrama de clases en alto nivel para modelo del problema.....	46
Diagrama 2: Diagrama de clases para diseño de las vistas de la aplicación. ....	47
Diagrama 3: Diagrama de clases del diseño de la vista de representación gráfica del modelo. ....	48
Diagrama 4: Diagrama de clases para vista de información general del modelo.....	49
Diagrama 5: Diagrama de clases del diseño de la vista para macros PSTricks.....	50
Diagrama 6: Diagrama de clases del módulo traductor.....	51
Diagrama 7: Diagrama de clases de la jerarquía de las vistas para generar las macros PSTricks. ....	51
Diagrama 8: Diagrama de clases de las utilidades diseñadas para realizar la traducción de código LaTeX. ....	52
Diagrama 9: Diagrama de clases representando el módulo de persistencia. ....	53
Diagrama 10: Diagrama de clases para la implementación del módulo de persistencia por exportar. ....	54
Diagrama 11: Diagrama de clases del diseño de la interfaz de usuario de la aplicación. ....	56
Diagrama 12: Diagrama de clases del diseño de menú de la aplicación. ....	57
Diagrama 13: Diagrama de clases para diseño de Barra de Herramientas. ....	58
Diagrama 14: Diagrama de clases del conjunto de algunas acciones realizadas por la aplicación.....	59
Diagrama 15: Diagrama de clases del área de trabajo de la aplicación.....	60
Diagrama 16: Diagrama de clases de modelo de controladores relacionados con UMLDiagramToolBar .....	63
Diagrama 17: Diagrama de clases para el controlador de figuras .....	63
Diagrama 18: Diagrama de clases que representa el diseño del controlador de conectores. ....	65
Diagrama 19: Diagrama de clases del controlador de arraste de figuras.....	66
Diagrama 20: Diagrama de clases para el Controlador de Teclado .....	67

## **1. Introducción**

Un desarrollador de software puede utilizar el lenguaje unificado de modelado UML, para crear diagramas que le apoyen en el análisis y diseño de sistemas de información. Para la creación de los diagramas se puede auxiliar de un editor gráfico.

Dada la necesidad de documentar dichos diagramas, ya sea en medios digitales o impresos, hay usuarios que desean incluir los diagramas UML que diseñaron en documentos LaTeX.

LaTeX es un lenguaje de marcado de documentos formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX.

En este documento se presenta UML4TeX, una aplicación que permite crear, editar y guardar diagramas UML. Además, la aplicación es capaz de exportar los diagramas como macros PSTricks.

En este trabajo se discute la motivación y objetivos de diseño de UML4TeX, así como las distintas fases de su desarrollo y los resultados obtenidos de la construcción de esta aplicación de edición gráfica.

## **2. Antecedentes**

### **2.1. Referencias internas**

Dentro de la Universidad Autónoma Metropolitana Unidad Azcapotzalco se han desarrollado editores gráficos para diferentes ámbitos de estudio, los cuales se mencionan a continuación:

- El proyecto terminal titulado “Ambiente de programación visual para construir geometrías simples mediante transformaciones ortogonales con álgebra geométrica y álgebra matricial” [4], ofrece una aplicación enfocada a la creación y manipulación de figuras geométricas mediante puntos, líneas, círculos, rotaciones y reflexiones.
- En el proyecto terminal titulado “EDAPIX: Una aplicación gráfica para asistir al proceso de enseñanza-aprendizaje de las estructuras de datos” [5], se desarrolló un editor gráfico para modelar, editar y simular distintas estructuras de datos (pilas, colas, listas, grafos).
- En el proyecto terminal titulado “Visualización de algoritmos de ordenamiento y búsqueda interna con *TikZ*” [6], se implementó una interfaz gráfica para visualizar algoritmos de ordenamiento y búsqueda y con la capacidad de generar código fuente en lenguaje *TikZ* [7], para crear gráficos integrables a documentos LaTeX.

Aunque estos proyectos ofrecen entornos donde se pueden editar gráficos, no están enfocados en la manipulación de diagramas UML.

### **2.2. Referencias Externas**

También existen diversas aplicaciones que han sido desarrolladas para la edición de diagramas UML:

- LaTeXDraw [8] es un editor gráfico para LaTeX y puede generar código PSTricks o exportar directamente en imágenes PDF o PostScript. Sin embargo, no tiene soporte para dibujar diagramas UML.
- Dia [9] es una aplicación desarrollada en gtk+ [10] para la plataforma GNOME [11] orientado a la creación de distintos tipos de diagramas, entre ellos, diagramas UML. También puede exportar directamente a macros LaTeX o PSTricks. Sin embargo, al ser un editor de múltiples diagramas no ofrece soporte completo para los de tipo UML. Además, al estar orientado a entornos GNOME, su portabilidad se ve disminuida.
- Umbrello [12] es un editor de diagramas UML desarrollado para el entorno gráfico KDE [13]. Permite editar y exportar diagramas UML hacia distintos lenguajes de programación, pero no permite exportar diagramas hacia formatos para su inclusión en archivos LaTeX. Además, al estar desarrollado para KDE no es portable.

Por esta razón el proyecto presentado tiene como objetivo desarrollar un editor gráfico que ofrezca las características de ser multiplataforma, permitir la edición de diagramas UML y la exportación en código fuente utilizable en documentos LaTeX.

### 3. Justificación

Un desarrollador de software puede utilizar el lenguaje unificado de modelado UML [1], para crear diagramas que le apoyen en el análisis y diseño de sistemas de información. Para la creación de los diagramas se puede auxiliar de un editor gráfico.

Dada la necesidad de documentar dichos diagramas, ya sea en medios digitales o impresos, hay usuarios que desean incluir los diagramas UML que diseñaron en documentos LaTeX [2].

LaTeX es un lenguaje de marcado de documentos formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX. Para poder incluir un diagrama en un documento de este tipo existe la posibilidad de insertar el diagrama en formato de imagen o como macros. Sin embargo, definir y especificar las macros para realizar un diagrama requiere de un conocimiento amplio sobre las funcionalidades que ofrece LaTeX y sobre cómo generar sus propias macros y paquetes. En lugar de definir y especificar cada elemento de un diagrama UML como macro, también existe la opción de utilizar paquetes ya diseñados como MetaUML [4] o uml.sty [5]. No obstante, estos no cuentan con un soporte completo para la generación de código para diagramas UML y todavía requiere que el usuario tenga experiencia en codificar la estructura de los diagramas en término de las macros especificadas en esos paquetes.

Por lo tanto, para facilitar la tarea de integrar diagramas UML como macros, podemos recurrir a las aplicaciones orientadas a editar diagramas UML y que sean capaces de exportar a macros compatibles con LaTeX.

Es por esta razón que el proyecto presentado ofrece UML4TeX, un editor gráfico y multiplataforma que permite al usuario editar diagramas UML y con la posibilidad de exportarlos como macros PSTricks [6], y por lo tanto, integrables en documentos LaTeX. La aplicación desarrollada tiene el propósito de ahorrar la tarea de desarrollar macros



especiales y evitar la difícil curva de aprendizaje para estructurar el código necesario para generar las imágenes asociadas a los diagramas, particularmente, diagramas UML.

Cabe aclarar al lector que también existen editores de diagramas UML que permiten generar código fuente en distintos lenguajes de programación o incluso realizar ingeniería inversa generando los diagramas UML a partir de código fuente. Sin embargo, la generación de código y la ingeniería inversa no son objeto de estudio en este proyecto.

## **4. Objetivos**

### **4.1. Objetivo general**

Diseñar e implementar un editor gráfico que permita crear, editar y guardar diagramas UML y sea capaz de exportarlos como macros PSTricks.

### **4.2. Objetivos específicos**

- Diseñar e implementar el módulo para representar la estructura lógica de diagramas UML.
- Diseñar e implementar el módulo de visualización de la estructura lógica de diagramas UML.
- Diseñar e implementar el módulo para la edición (manipulación gráfica) de diagramas UML.
- Diseñar e implementar el módulo de la interfaz gráfica de usuario.
- Diseñar e implementar el módulo de persistencia de los diagramas.
- Diseñar e implementar el módulo de traducción de la estructura lógica de diagramas UML como macros PSTricks.
- Integrar los módulos en el entorno de la aplicación.

- Realizar las pruebas de integridad a los distintos módulos de la aplicación desarrollada.

## **5. Marco teórico**

### **5.1. Editor de diagramas**

Un diagrama es una representación simbólica de información que se ajusta a una técnica de visualización.

Un editor de diagramas es un software que es utilizado para modelar, representar y visualizar información. Esta información se utiliza en desarrollo de software, técnico o de negocio para representar flujos de datos, flujos de trabajo, arquitecturas de software o estructuras organizacionales.

Por lo tanto, un editor de diagramas puede estar diseñado para tener un uso específico dependiendo el área en el que se encuentre enfocada y el tipo de información que se quiera representar. Existen varios tipos de editores para el modelado y diseño de diagramas. Por mencionar algunos:

- Editor de diagramas de flujo.
- Editor de diagramas UML.
- Editor de diagramas conceptuales.
- Editor de mapa de ideas y de conceptos.
- Editor de diagramas de estados.
- Editor de diagramas de proceso de negocio.

Para efectos del proyecto solo incluye la conceptualización de diagramas UML.

### 5.1.1. Editor de diagramas UML

Un editor de diagramas UML suele ser llamado también **Herramienta de Modelado UML**, una aplicación de software que soporta algunas o todas las notaciones asociadas al lenguaje unificado de modelado.

Estas herramientas ofrecen normalmente las siguientes características:

- **Diagramas:** Consiste en crear y editar diagramas UML que sigan con la notación gráfica del lenguaje de modelado unificado.
- **Generación de código:** La capacidad de generar código a cierto lenguaje de programación, a partir de los modelos especificados por los diagramas realizados.
- **Ingeniería inversa:** La capacidad de generar diagramas UML a partir de código fuente de algún lenguaje de programación.

Es importante mencionar al lector que la generación de código y la ingeniería inversa no son objeto de estudio en este proyecto.

## 5.2. UML

UML (Unified Modeling Language) es un lenguaje de modelado estándar que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. [17]

Dentro de las especificaciones de UML 2.0 se determinan trece diagramas [21], los cuales se clasifican en tres categorías (ver Figura 1):

- **Diagramas de comportamiento:** Incluyen los diagramas de actividad, estado, caso de uso y de interacción.
- **Diagramas de interacción:** Incluyen los diagramas de comunicación, vista general de interacciones, de secuencia y de tiempo.
- **Diagramas de estructura:** Incluyen los diagramas de clases, estructura de componentes, de componente, de despliegue, de objetos y de paquetes.

Para efectos de este proyecto se considera un diagrama de cada categoría. De esta manera, el proyecto se da por concluido cuando la aplicación desarrollada sea capaz de crear, editar y guardar los siguientes diagramas UML:

- Diagrama de clases
- Diagrama de casos de uso
- Diagramas de secuencia

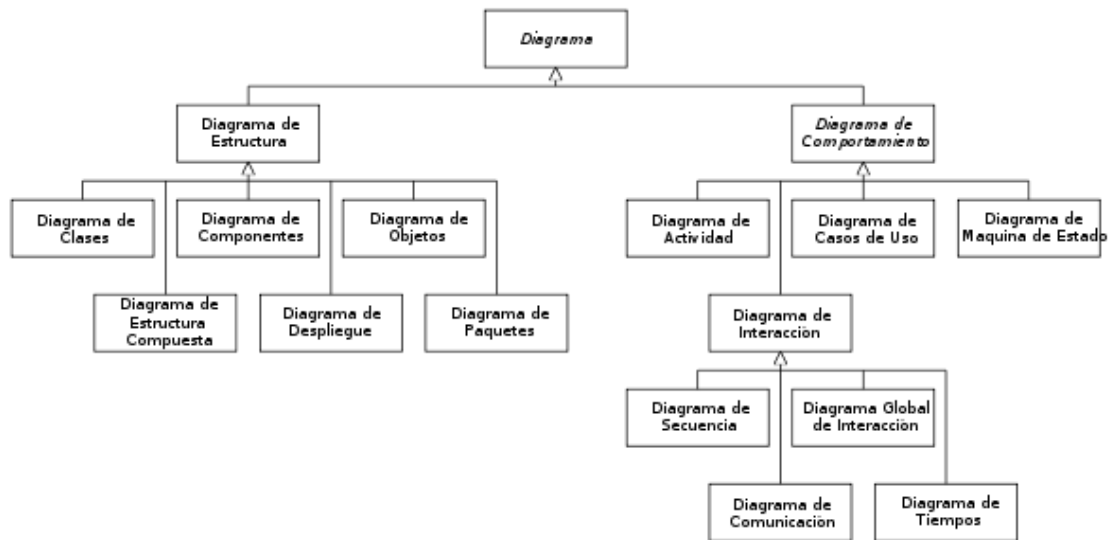


Figura 1: Estructura de la organización de los distintos diagramas UML.

### 5.2.1. Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de sistemas de información, donde se crea el diseño conceptual de la información que se manejará, los componentes que se encargarán del funcionamiento y la relación entre uno y otro. En un diagrama de clases se pueden distinguir principalmente dos elementos: las clases y sus relaciones.

#### 5.2.1.1. Clase

La clase es la unidad básica que encapsula toda la información de un objeto a través de la cual podemos modelar el entorno en estudio. En UML, una clase es representada por un rectángulo que posee tres divisiones (ver Figura 2).

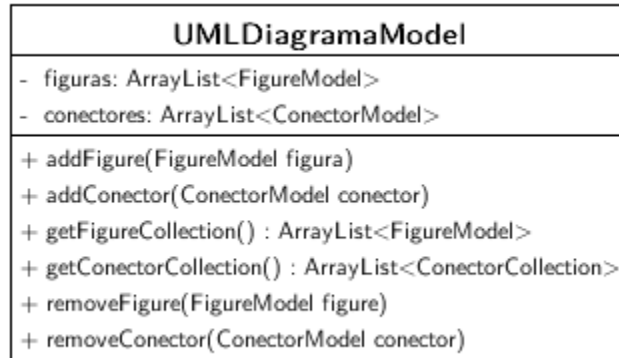


Figura 2: Ejemplo de una clase en UML.

En donde:

- El rectángulo superior contiene el nombre de la clase
- El rectángulo intermedio contiene los atributos (o variables de instancia) que caracterizan a la clase.
- El rectángulo inferior contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno.

#### 5.2.1.1.1. Atributos

Los atributos representan la información acerca de un objeto (ver Figura 3).

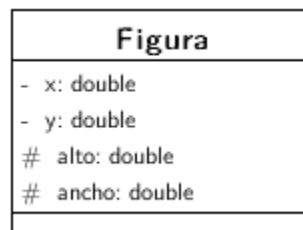


Figura 3: Representación de las propiedades de un objeto en una clase UML.

#### 5.2.1.1.2. Métodos u operaciones

Las operaciones son funciones o transformaciones que se aplican a todos los objetos de una clase particular. La operación puede ser una acción ejecutada por el objeto o sobre el objeto (ver Figura 4).

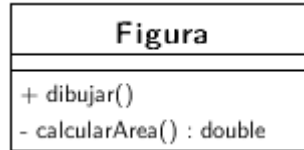


Figura 4: Representación de los métodos u operaciones de un objeto en una clase UML.

### 5.2.1.1.3. Visibilidad de los atributos y los métodos de la clase

Los atributos y los métodos deben tener una visibilidad asignada, que puede ser:

- + Visibilidad pública.
- # Visibilidad protegida.
- - Visibilidad privada.
- ~ Visibilidad de paquete.

### 5.2.1.2. Relaciones

Una relación representa una interacción entre una o más clases. A continuación se enlistan los tipos de relaciones utilizados en los diagramas de clases de UML.

#### 5.2.1.2.1. Herencia (Especialización/Generalización)

Indica que una clase (clase derivada) hereda los métodos y atributos especificados por una clase (clase base), por lo cual una clase derivada además de tener sus propios métodos y atributos, podrá acceder a las características y atributos visibles de su clase base.

En la Figura 5 podrá observar un ejemplo de este tipo de relación:

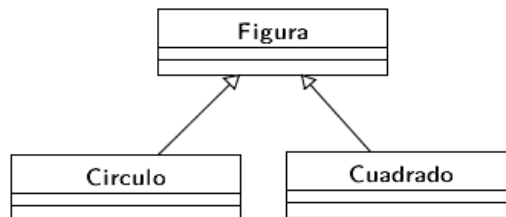


Figura 5: Relación de Especialización/Generalización en UML

### 5.2.1.2.2. Composición

La composición es un tipo de relación estática, que permite expresar que un objeto se compone de otros objetos.

En la Figura 6 podrá observar un ejemplo de este tipo de relación:

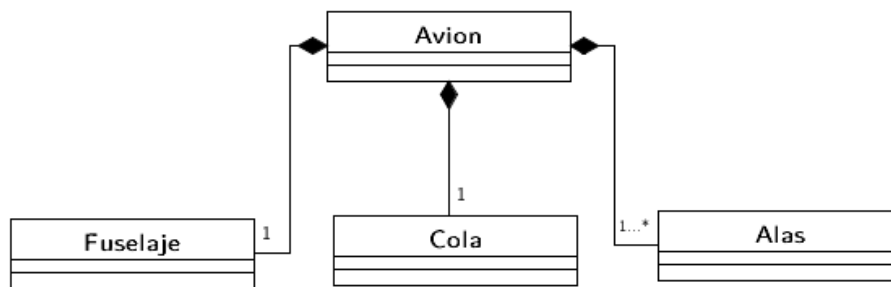


Figura 6: Relación de Composición en UML

### 5.2.1.2.3. Agregación

La agregación es un tipo de relación dinámica, que permite expresar que un objeto agrupa a otros objetos.

En la Figura 7 podrá observar un ejemplo de este tipo de relación:

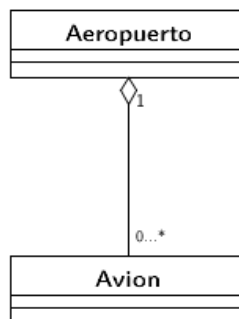


Figura 7. Relación de Agregación en UML

#### 5.2.1.2.4. Dependencia o Instanciación

Representa un tipo de relación en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada. El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra (ver Figura 8).

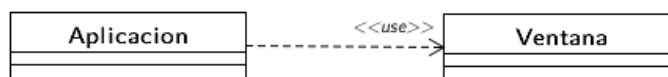


Figura 8: Relación de Dependencia o Instanciación en UML

#### 5.2.1.2.5. Asociación

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí. En la Figura 9 podrá observar un ejemplo de este tipo de relación:



Figura 9: Relación de Asociación en UML

Los elementos adicionales que pueden aparecer en una relación de este tipo son los siguientes:

##### **Rol**

Identifica con nombres a los elementos que aparecen en los extremos de la línea que denota la relación, dicho nombre describe la semántica que tiene la relación en el sentido indicado.

##### **Multiplicidad**

La multiplicidad de una relación determina cuantos objetos de cada tipo intervienen en la relación y las siguientes características (ver Figura 10):



MULTIPLICIDAD	SIGNIFICADO
1	Uno y solo uno
0...1	Cero o uno
X...Y	Desde X hasta Y
*	Cero o varios
1...*	Uno o varios

Figura 10. Multiplicidad de una asociación UML

## 5.2.2. Diagrama de caso de uso

Los diagramas de casos de uso son diagramas que sirven para modelar el comportamiento de un sistema, un subsistema o una clase. Cada uno muestra un conjunto de casos de uso, sus actores y sus relaciones.

Un diagrama de casos de uso consta de los siguientes elementos:

- Actor
- Casos de Uso
- Relaciones de Uso, Herencia y Comunicación

### 5.2.2.1. Actor

Un Actor es un rol que un usuario o subsistema juega con respecto al sistema en construcción. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema (ver Figura 11).



Figura 11: Representación de un Actor en UML

### 5.2.2.2. Caso de uso

Es una tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso (ver Figura 12).

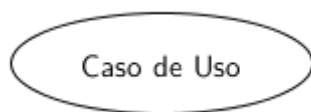


Figura 12: Representación de un caso de uso en UML

### 5.2.2.3. Relaciones

#### 5.2.2.3.1. Comunicación

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una línea simple (ver Figura 13).

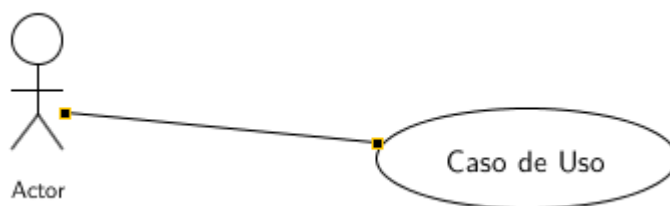


Figura 13: Representación de una relación de comunicación en UML

#### 5.2.2.3.2. Inclusión

Indica que una instancia del caso de uso fuente comprende también el comportamiento descrito por el caso de uso destino. Es decir, denota la inclusión del comportamiento de un escenario en otro (ver Figura 14).

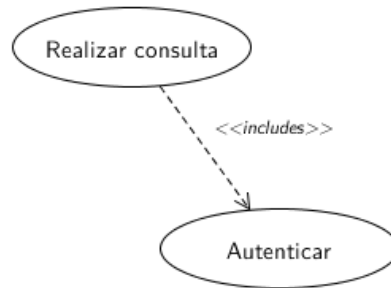


Figura 14: Representación de la relación de inclusión de casos de uso en UML.

### 5.2.2.3.3. Extensión

Indica que el caso de uso fuente extiende el comportamiento del caso de uso destino. Es decir, denota cuando un caso de uso es una especialización de otro (ver Figura 15).

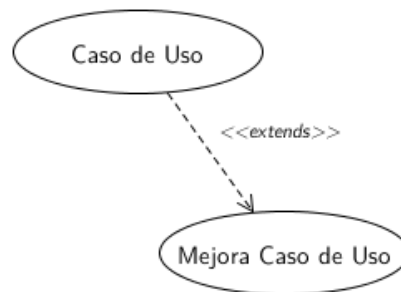


Figura 15: Representación de la relación de extensión de casos de uso en UML

### 5.2.2.3.4. Generalización

Un caso de uso (subcaso) hereda el comportamiento y significado de otro, es decir las relaciones de comunicación, inclusión y extensión del supercaso de uso. También puede darse la herencia entre actores (ver Figura 16).

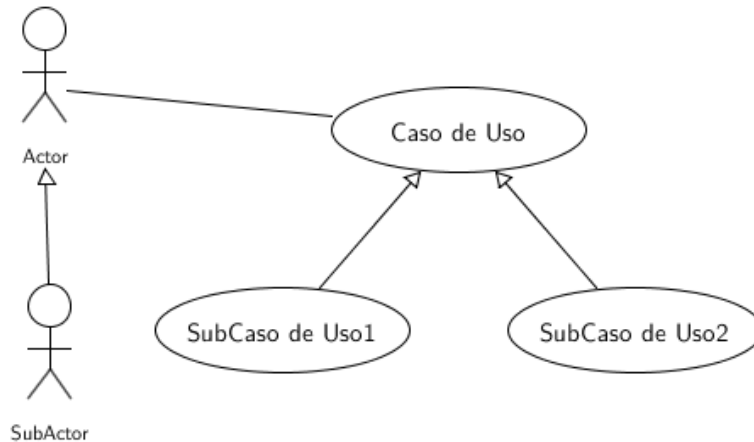


Figura 16: Representación de herencia de casos de uso/actores en UML

### 5.2.3. Diagrama de secuencia

El diagrama de secuencias muestra la forma en que los objetos se comunican entre sí a través del tiempo.

El diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos.

#### 5.2.3.1. Tipos de mensajes

##### 5.2.3.1.1. Mensajes síncronos

Los mensajes síncronos se corresponden con llamadas a métodos del objeto que recibe el mensaje. El objeto que envía el mensaje queda bloqueado hasta que termina la llamada. Este tipo de mensajes se representan con flechas con la cabeza llena.

##### 5.2.3.1.2. Mensajes asíncronos

Los mensajes asíncronos terminan inmediatamente, y crean un nuevo hilo de ejecución dentro de la secuencia. Se representan con flechas con la cabeza abierta.

### 5.3. PSTricks

PSTricks es una colección de macros PostScript que es compatible con la mayoría de las macros TeX y LaTeX, y que soporta color, gráficas, movilidad, árboles y otros.

Las macros PSTricks permiten la inclusión de dibujos PostScript directamente dentro de código TeX o LaTeX.

Originalmente fue un trabajo del Profesor Timothy Van Zandt y en años recientes lo han mantenido Denis Girou, Sebastian Rahtz y Herbert Voss [16].

PSTricks cuenta con una lista de comandos que te permiten especificar las características que tendrá un gráfico que se esté especificando (Ver Figura 17).

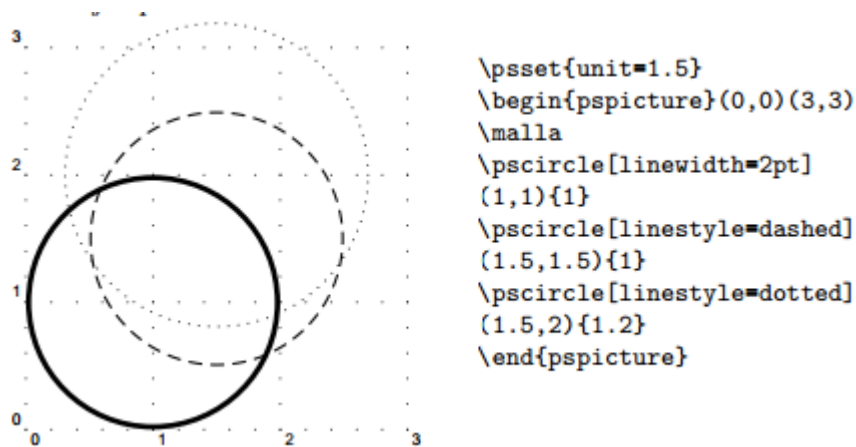


Figura 17: Ejemplo de codificación de macros PSTricks y la figura resultante para dibujar una un círculo.

## 6. Desarrollo del 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 [17]) el cual es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El modelo en espiral en el cual está basado este proceso permitió que la aplicación se fuese 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

#### 6.2.1. Documentación de Actores

Para el diseño del editor gráfico se identificó el siguiente actor:

<b>Actor:</b>	Usuario
<b>Tipo:</b>	Primario
<b>Descripción:</b>	Este actor representa a cualquier usuario que interactúe con la aplicación.
<b>Comentarios:</b>	Este usuario posiblemente sea un programador o diseñador de diagramas UML. Puede ser algún estudiante o profesor que requiera de realizar diagramas básicos de UML y a su vez requiera integrarlos a sus documentos LaTeX

## 6.2.2. Casos de uso

A un nivel de abstracción alto, el siguiente diagrama de casos de uso muestra que hay un actor “Usuario” que interactúa con el sistema. En la Figura 18 se muestra el diagrama de casos de uso de la aplicación.

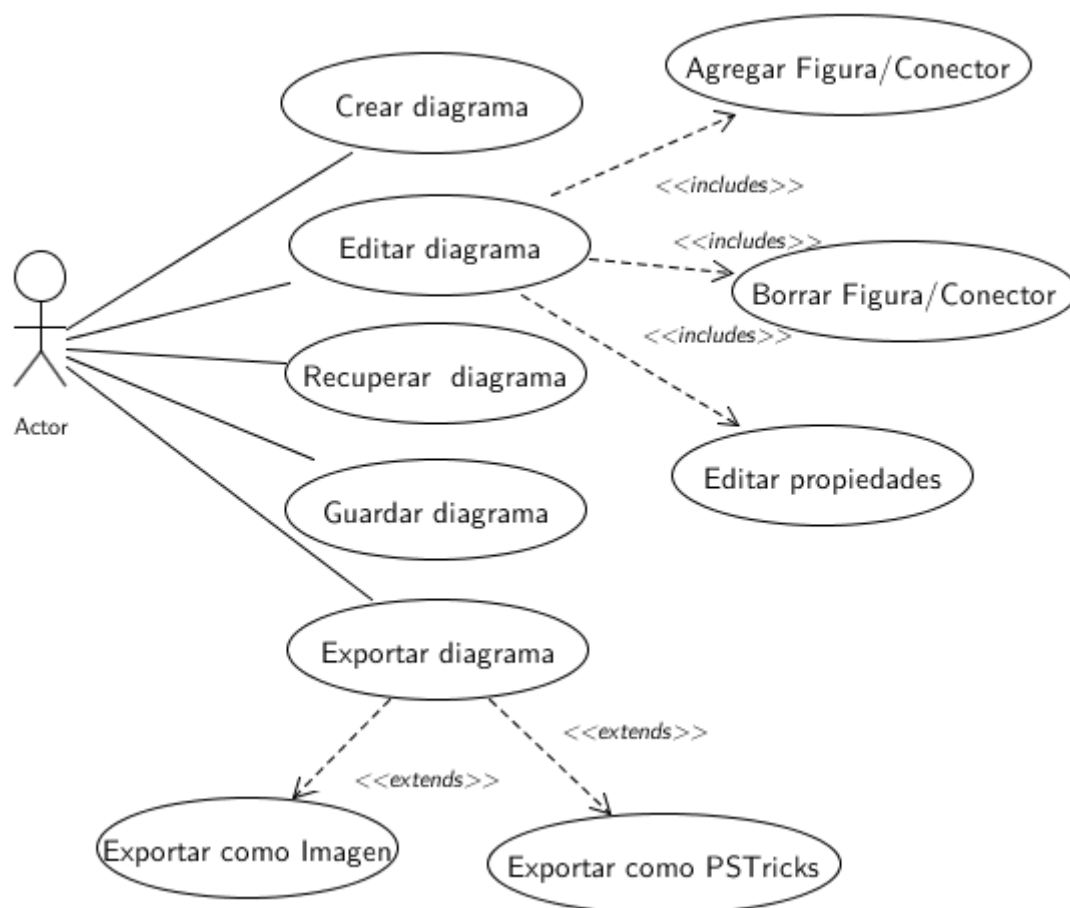


Figura 18: Diagrama de Casos de uso para la aplicación UMLaTeX.

A continuación se describen los casos de uso más importantes de la aplicación.

### 6.2.2.1. CU-01 Crear Diagrama

<b>CU-01</b>	<b>Crear Diagrama</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Básico
<b>Propósito:</b>	Crea un diagrama UML.
<b>Resumen:</b>	El usuario solicita a la aplicación crear un nuevo diagrama UML para su posterior edición. Los diagramas que puede elegir son los siguientes: <ul style="list-style-type: none"> <li>- Diagrama de casos de uso</li> <li>- Diagrama de clases de clases</li> <li>- Diagrama de secuencia</li> </ul>
<b>Pre condiciones:</b>	Tener la aplicación abierta.
<b>Post condiciones:</b>	La aplicación muestra un panel de trabajo con un diagrama vacío y una barrar de herramientas con los artefactos UML correspondientes al tipo de diagrama elegido.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario solicita a la aplicación la realización de la operación de creación de nuevo diagrama y especifica el tipo de diagrama.</li> <li>2. Si hay un diagrama abierto ir a excepción &lt;E1 Diagrama Existente&gt;. Si no hay diagrama continuar en 3).</li> <li>3. La aplicación crea un nuevo panel de trabajo (área de edición).</li> <li>4. La aplicación obtiene la barra de herramientas con los artefactos UML correspondientes al tipo de diagrama especificado.</li> <li>5. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código).</li> </ol>
<b>Sub flujo:</b>	Ninguno.
<b>Excepciones:</b>	<p>&lt;E1 Diagrama Existente&gt;</p> <ol style="list-style-type: none"> <li>1. El sistema notificará al usuario mediante un cuadro de diálogo si desea descartar el diagrama actual con las opciones SI y NO.</li> <li>2. Si la respuesta el SI. Regresar al flujo principal en el punto <b>3</b>)</li> <li>3. Si la respuesta es NO. Finalizar caso de uso (No crea diagrama).</li> </ol>
<b>Frecuencia</b>	Alta.
<b>Comentarios</b>	Ninguno.



### 6.2.2.2. CU-02 Recuperar Diagrama

<b>CU-02</b>	<b>Recuperar Diagrama</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Básico
<b>Propósito:</b>	Recupera un diagrama UML.
<b>Resumen:</b>	El usuario puede recuperar un diagrama previamente guardado por la herramienta.
<b>Pre condiciones:</b>	Existe un archivo con la descripción del modelo del diagrama (previamente guardado por la aplicación).
<b>Post condiciones:</b>	Se muestra el diagrama en el panel de trabajo listo para editar.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario solicita a la aplicación recuperar (abrir) un diagrama previamente guardado por la aplicación.</li> <li>2. La aplicación muestra un cuadro de diálogo con un sistema de archivos para que el usuario escoja el archivo a abrir. Si el usuario no elige un archivo, ejecuta &lt;SI Cancela Abrir&gt;, de lo contrario continuar en 3).</li> <li>3. Si el archivo no es válido ejecutar &lt;E1 Archivo no válido&gt;. Si el archivo es válido continuar en 4)</li> <li>4. La aplicación decodifica el archivo para obtener el modelo del diagrama y crea el panel de trabajo (área de edición) con la barra de herramientas con los artefactos UML correspondientes al tipo del diagrama recuperado.</li> <li>5. La aplicación muestra el diagrama recuperado.</li> </ol>
<b>Sub flujo:</b>	<S1 Cancela Abrir> <ol style="list-style-type: none"> <li>1. El usuario decide cancelar la operación Abrir.</li> <li>2. Termina Caso de Uso.</li> </ol>
<b>Excepciones:</b>	<E1 Archivo no válido> <ol style="list-style-type: none"> <li>1. El archivo no es válido, la aplicación mostrará un aviso indicando que el archivo elegido no es válido.</li> <li>2. Termina Caso de Uso.</li> </ol>
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

### 6.2.2.3. CU-03 Editar diagrama

<b>CU-03</b>	<b>Editar Diagrama</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Básico
<b>Propósito:</b>	Edición de un diagrama UML.
<b>Resumen:</b>	<p>El usuario puede realizar una serie de operaciones para editar gráficamente el modelo de un diagrama. Las operaciones que puede hacer son:</p> <ul style="list-style-type: none"> <li>- Añadir un artefacto UML al diagrama (figura/conector)</li> <li>- Eliminar un artefacto UML del diagrama (figura/conector)</li> <li>- Editar las propiedades de un artefacto UML (figura/conector)</li> </ul>
<b>Pre condiciones:</b>	El usuario ha creado o recuperado un diagrama.
<b>Post condiciones:</b>	Se cambia la estructura del modelo del diagrama y se actualizan las vistas del diagrama.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario desea agregar un artefacto UML al diagrama. Ejecuta el Caso de Uso <b>CU-04 Agregar Figura/Conector</b>.</li> <li>2. El usuario desea editar un artefacto UML. Ejecuta el Caso de Uso <b>CU-05 Editar Propiedades</b></li> <li>3. El usuario desea borrar un artefacto UML del diagrama. Ejecuta el Caso de Uso <b>CU-06 Borrar Figura/Conector</b>.</li> </ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	Ninguno.
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

#### 6.2.2.4. CU-04 Agregar Figura/Conector

<b>CU-04</b>	<b>Agregar Figura/Conector</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Inclusión
<b>Propósito:</b>	Agrega un nuevo componente en el diagrama.
<b>Resumen:</b>	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.
<b>Pre condiciones:</b>	La aplicación tiene un diagrama abierto para su edición.
<b>Post condiciones:</b>	Actualiza el modelo y la vista del diagrama editado.
<b>Flujo principal:</b>	<ol style="list-style-type: none"><li>1. El usuario elige de la barra de herramientas un artefacto UML (figura o conector).</li><li>2. El usuario se posiciona en el área de edición y coloca el artefacto UML previamente elegido.</li><li>3. La aplicación agrega el nuevo componente al modelo del diagrama.</li><li>4. La aplicación actualiza el modelo y las vistas asociadas al diagrama editado.</li></ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	Ninguno
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

### 6.2.2.5. CU-05 Editar propiedades

<b>CU-05</b>	<b>Editar propiedades.</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Inclusión
<b>Propósito:</b>	Edita las propiedades de un componente del diagrama.
<b>Resumen:</b>	Muestra pantallas y diálogos donde se pueden editar las propiedades alusivas a un componente (figura o conector) del diagrama editado y actualiza su modelo.
<b>Pre condiciones:</b>	Existe un diagrama a editar y se ha seleccionado un componente para modificar propiedades.
<b>Post condiciones:</b>	Actualiza el modelo lógico y las vistas del componente modificado.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona.</li> <li>2. El usuario pide a la aplicación Editar el artefacto UML seleccionado.</li> <li>3. La aplicación muestra el formulario relacionado con el artefacto UML a editar.</li> <li>4. El usuario edita las propiedades y acepta los cambios. Si el usuario decide no aceptar los cambios ejecutar <i>&lt;E1 No aceptar cambios&gt;</i></li> <li>5. Se actualiza el modelo y la vista del diagrama editado.</li> </ol>
<b>Sub flujo:</b>	Ninguno.
<b>Excepciones:</b>	<i>&lt;E1 No aceptar cambios&gt;</i> <ol style="list-style-type: none"> <li>1. La aplicación no actualiza el diagrama.</li> <li>2. Fin del caso de uso.</li> </ol>
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

#### 6.2.2.6. CU-06 Borrar Figura/Conector

<b>CU-06</b>	<b>Borrar Figura/Conector</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Inclusión
<b>Propósito:</b>	Borra un componente en el diagrama.
<b>Resumen:</b>	Elimina un artefacto UML (figura o conector) del diagrama, actualiza el modelo del diagrama y su vista en la aplicación.
<b>Pre condiciones:</b>	La aplicación tiene un diagrama abierto para su edición.
<b>Post condiciones:</b>	Actualiza el modelo y la vista del diagrama editado.
<b>Flujo principal:</b>	<ol style="list-style-type: none"><li>1. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona.</li><li>2. El usuario pide a la aplicación borrar el artefacto UML seleccionado.</li><li>3. La aplicación elimina el nuevo componente al modelo del diagrama.</li><li>4. La aplicación actualiza el modelo y las vistas asociadas al diagrama editado.</li></ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	Ninguno
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

### 6.2.2.7. CU-07 Guardar Diagrama

<b>CU-07</b>	<b>Guardar Diagrama</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Básico
<b>Propósito:</b>	Guarda un diagrama a un archivo.
<b>Resumen:</b>	Guarda un diagrama en un formato específico para su posterior edición. (Persistencia)
<b>Pre condiciones:</b>	Existe un diagrama con elementos.
<b>Post condiciones:</b>	Se genera un archivo que tiene los datos del modelo y representación del diagrama a salvar.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario solicita a la aplicación que se ejecute la operación de guardar.</li> <li>2. La aplicación muestra un diálogo con un sistema de archivos para elegir el nombre y ubicación del archivo a guardar.</li> <li>3. Si el archivo elegido ya existe, ejecutar &lt;E1 Archivo existente&gt;, de lo contrario continuar en 4.</li> <li>4. La aplicación realizará la operación de guardado. Si existe un error en el guardado, ejecutar &lt;E2 Error Grabado&gt;</li> </ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	<p><b>&lt;E1 Archivo existente&gt;</b></p> <ol style="list-style-type: none"> <li>1. El archivo elegido ya existe, la aplicación mostrará un diálogo con las opciones “Si”/”No” confirmando si se sobrescribirá el archivo. Si el usuario elige “Si” ejecutar el flujo principal en el paso 4. Si el usuario elige “No” continuar.</li> <li>2. Fin del caso de uso.</li> </ol> <p><b>&lt;E2 Error Grabado&gt;</b></p> <ol style="list-style-type: none"> <li>1. La aplicación mostrará un mensaje de error, indicando que hubo un error al guardar el archivo.</li> <li>2. Fin del caso de uso.</li> </ol>
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Se utilizará como formato de salida el tipo de archivo XML. La aplicación podría manejar su extensión de archivo final.

### 6.2.2.8. CU-08 Exportar Diagrama

<b>CU-08</b>	<b>Exportar Diagrama</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Básico
<b>Propósito:</b>	Exportará el diagrama a otro formato.
<b>Resumen:</b>	Exportará el diagrama actual en el formato elegido por el usuario.
<b>Pre condiciones:</b>	Existe un diagrama con elementos.
<b>Post condiciones:</b>	Se genera el archivo correspondiente con la representación del diagrama en el formato elegido por el usuario.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario solicita a la aplicación que se ejecute la operación de exportar:</li> <li>2. Elige “Exportar como código PSTricks”. Ejecuta <b>CU-09 Exportar Diagrama como PSTricks</b>.</li> <li>3. Elige “<b>Exportar como imagen</b>”. Ejecuta CU-10 Exportar como imagen.</li> </ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	<ol style="list-style-type: none"> <li>2. Si la operación no puede ser realizada por la aplicación, esta no se realiza y muestra un mensaje de error.</li> </ol>
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

### 6.2.2.9. CU-09 Exportar Diagrama como PSTricks

<b>CU-09</b>	<b>Exportar Diagrama como PSTricks</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Extensión
<b>Propósito:</b>	Exporta un diagrama a un archivo con macros PSTricks.
<b>Resumen:</b>	Exporta el diagrama en un archivo *.tex con macros PSTricks.
<b>Pre condiciones:</b>	Existe un diagrama con elementos.
<b>Post condiciones:</b>	Se genera un archivo de texto que contendrá las macros PSTricks para dibujar el diagrama UML a exportar.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario solicita a la aplicación que se ejecute la exportación como macros de PSTricks.</li> <li>2. La aplicación muestra un diálogo al usuario para saber dónde guardar el archivo *.tex de salida.</li> <li>3. El usuario introduce el nombre del archivo.</li> <li>4. La aplicación genera el archivo *.tex con las macros PSTricks del diagrama actual. Si existe un error ejecutar &lt;E1 Error al exportar&gt;.</li> </ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	<p>&lt;E1 Error al Exportar&gt;</p> <ol style="list-style-type: none"> <li>3. Si la operación no puede ser realizada por la aplicación, esta no se realiza y muestra un mensaje de error.</li> </ol>
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.



### 6.2.2.10. CU-10 Exportar Diagrama como imagen

<b>CU-10</b>	<b>Exportar Diagrama como imagen</b>
<b>Actores:</b>	Usuario
<b>Tipo:</b>	Extensión
<b>Propósito:</b>	Exporta el diagrama como imagen.
<b>Resumen:</b>	Exporta el diagrama actual en un formato de imagen (JPG/PNG).
<b>Pre condiciones:</b>	Existe un diagrama con elementos.
<b>Post condiciones:</b>	La aplicación genera un archivo de imagen (JPG/PNG) que tiene la representación gráfica del diagrama actual.
<b>Flujo principal:</b>	<ol style="list-style-type: none"> <li>1. El usuario solicita a la aplicación que se ejecute una operación de exportar como imagen.</li> <li>2. La aplicación muestra al usuario un diálogo para saber dónde guardar el archivo de imagen de salida.</li> <li>3. El usuario escribe el nombre del archivo.</li> <li>4. La aplicación realiza la operación de exportación en el formato indicado (PNG/JPG). Si existe un error ejecutar <b>&lt;E1 Error al exportar&gt;</b></li> </ol>
<b>Sub flujo:</b>	Ninguno
<b>Excepciones:</b>	<p><b>&lt;E1 Error al Exportar&gt;</b></p> <ol style="list-style-type: none"> <li>1. Si la operación no puede ser realizada por la aplicación, esta no se realiza y muestra un mensaje de error.</li> </ol>
<b>Frecuencia</b>	Alta
<b>Comentarios</b>	Ninguno.

### 6.2.3. Diagrama de clases

Dado que utilizaremos el patrón de diseño **Modelo-Vista-Controlador** [19], es necesario identificar las clases que corresponden a cada aspecto del diagrama.

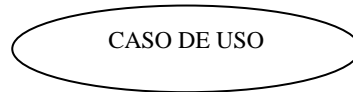
#### 6.2.3.1. Modelo del dominio del problema

Para el diseño de la aplicación se identificaron las clases conceptuales relacionadas con los requisitos para crear un modelo de dominio inicial que corresponda a una representación de clases conceptuales del mundo real.

En general, un diagrama está compuesto de figuras y conectores (similar a un grafo, que contiene nodos y aristas).

En particular, un diagrama de UML tiene que representar algo en la notación UML con decoradores y conectores entre ellos, los cuales representan relaciones entre los elementos del diagrama (ver Figura 19).

Ejemplo de Figura de UML:



Ejemplo de Conector UML:



*Figura 19: Ejemplos de artefactos UML.*

Dentro de las especificaciones de UML 2.0 se determinan trece diagramas. Para efectos de este proyecto se implementaron los siguientes diagramas:

- Diagrama de clases
- Diagrama de casos de uso
- Diagramas de secuencia

Los diagramas tienen en común que son una colección de componentes UML (tienen una representación) y de relaciones entre ellos que se representan por medio de conectores.

Las clases que fueron elegidas como modelo de la aplicación son las siguientes:

- ❖ **Clase Diagrama**
- ❖ **Clase Figura**
- ❖ **Clase Conector**

#### 6.2.3.1.1. Clase Diagrama

Una clase Diagrama tiene una colección de Figuras y una colección de Conectores (ver Figura 20).

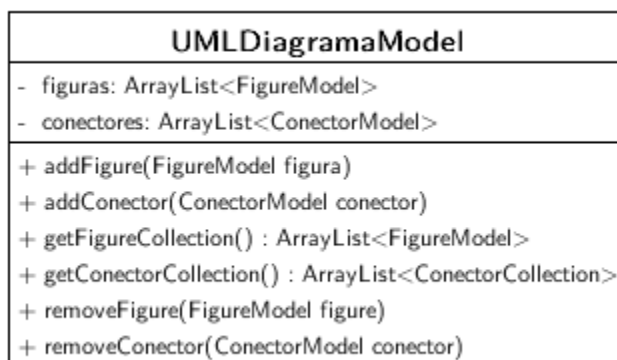


Figura 20: Clase Diagrama

Además debería ser capaz de:

- **Conectar** dos Figuras (que ya existen en la colección de Figuras del diagrama).
- **Agregar** nuevas Figuras a la colección de Figuras.
- **Agregar** nuevos Conectores a la colección de Conectores.
- **Eliminar** las **figuras** en la colección de Figuras.
- **Eliminar** los **conectores** en la colección de Conectores.

Los diagramas pueden ser de distintos tipos (diagrama de clase, diagrama de caso de uso, diagrama de secuencia). Estos se generalizan para cada diagrama específico. Luego se

puede implementar como una clase abstracta que encapsule las propiedades y métodos comunes a todos los diagramas.

Los Diagramas concretos deberían proporcionar los tipos de Figuras o de Conectores que utiliza.

Al tratarse de un modelo, esta clase implementa la interfaz *Observer* para que las vistas que observen a este modelo sean notificadas sobre los cambios realizados en el modelo.

### 6.2.3.1.2. Clase Figura

Una figura es un elemento gráfico. En particular, enfocándonos en los diagramas UML, una figura debería dar una representación gráfica de algún componente de la notación UML (ver Figura 21).

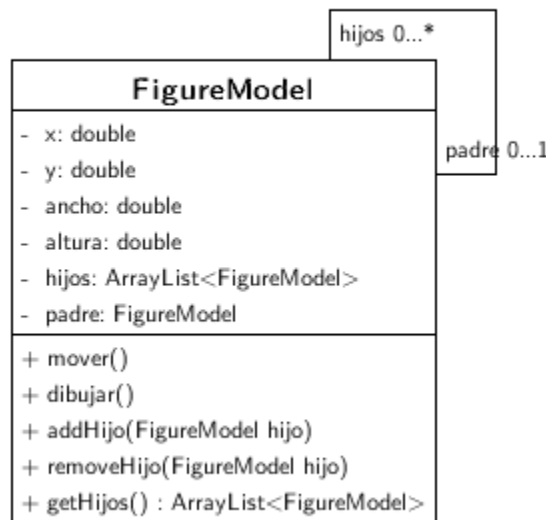


Figura 21: Clase Figura

Entre las características más importantes de una figura se encuentran las siguientes.

- Una figura puede tener alguna forma (circular, rectangular, etc.)
- Tienen propiedades **específicas** para cada tipo de Figura.
- Debe ofrecer métodos que modifiquen su comportamiento (ya sea **mover** la figura sobre el espacio de trabajo, dar los puntos de su marco delimitador, ofrecer puntos de conexión a los vértices, etc...).

- Ya que cualquier componente en UML termina siendo una figura, sería conveniente encapsular los métodos y atributos que sean comunes a ellos y dejar las implementaciones concretas en clases heredadas.

Para la implementación de esta clase se utilizará el patrón de diseño *Composite* [19]. Este patrón de diseño sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

Tratar la clase figura de esta manera ayudará a la implementación de las distintas vistas que se crearán a partir de este modelo.

### 6.2.3.1.3. Clase Conector

La clase Conector tiene como tarea principal **conectar dos figuras**. (Una figura origen y otra figura destino). Además, debe proveer los métodos que den información sobre las Figuras que conecta y proveer los métodos para crear sus vistas (ver Figura 22).

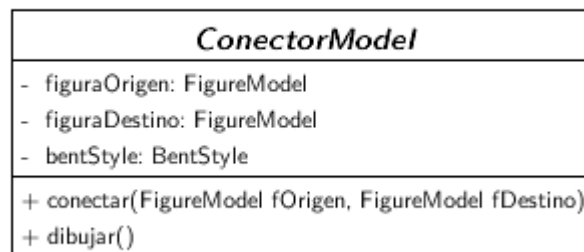


Figura 22: Clase Conector

Los conectores en UML son similares ya que están compuestos de 3 partes (ver Figura 23).

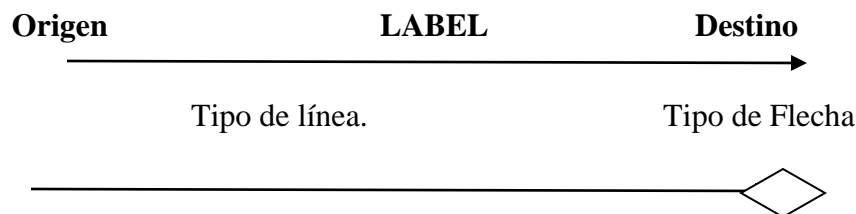


Figura 23: Características de un conector.

Luego, se pueden englobar las propiedades en una clase abstracta que permita la elección del tipo de línea o de flecha en los conectores (dependiendo del tipo de diagramas a elegir).

#### 6.2.3.1.4. Identificación de relaciones

Se enumeran a continuación las relaciones existentes entre cada clase que se escogió para el modelo del sistema.

- Un **Diagrama** puede ser de distintos tipos. Un diagrama de UML (de clase, caso de uso, de secuencia...) **es un Diagrama**.
- Un **Diagrama** está compuesto de una colección de **Figuras** y **Conectores**

Un Diagrama Especifico tiene asociado un tipo específico de Figuras y Conectores.

- Los diversos tipos de componentes o artefactos UML pueden representarse como figuras. Un componente UML **es una Figura**.
- Un conector puede tener varias formas. En especial para UML, las Relaciones son un tipo de conector. Una **Relación es un Conector**.

#### 6.2.3.1.5. Identificación de multiplicidad

- Un diagrama tiene **una o muchas** figuras [1...\*]
- Un diagrama tiene **ceros o muchos** conectores [0...\*]
- Una figura puede estar atada a una o muchas figuras por medio de conexiones, una figura tiene **ceros o muchas** conexiones [0...\*]
- Una conexión tiene cero o una figura origen a conectar. [0...1]
- Una conexión tiene cero o una figura destino a conectar. [0...1]

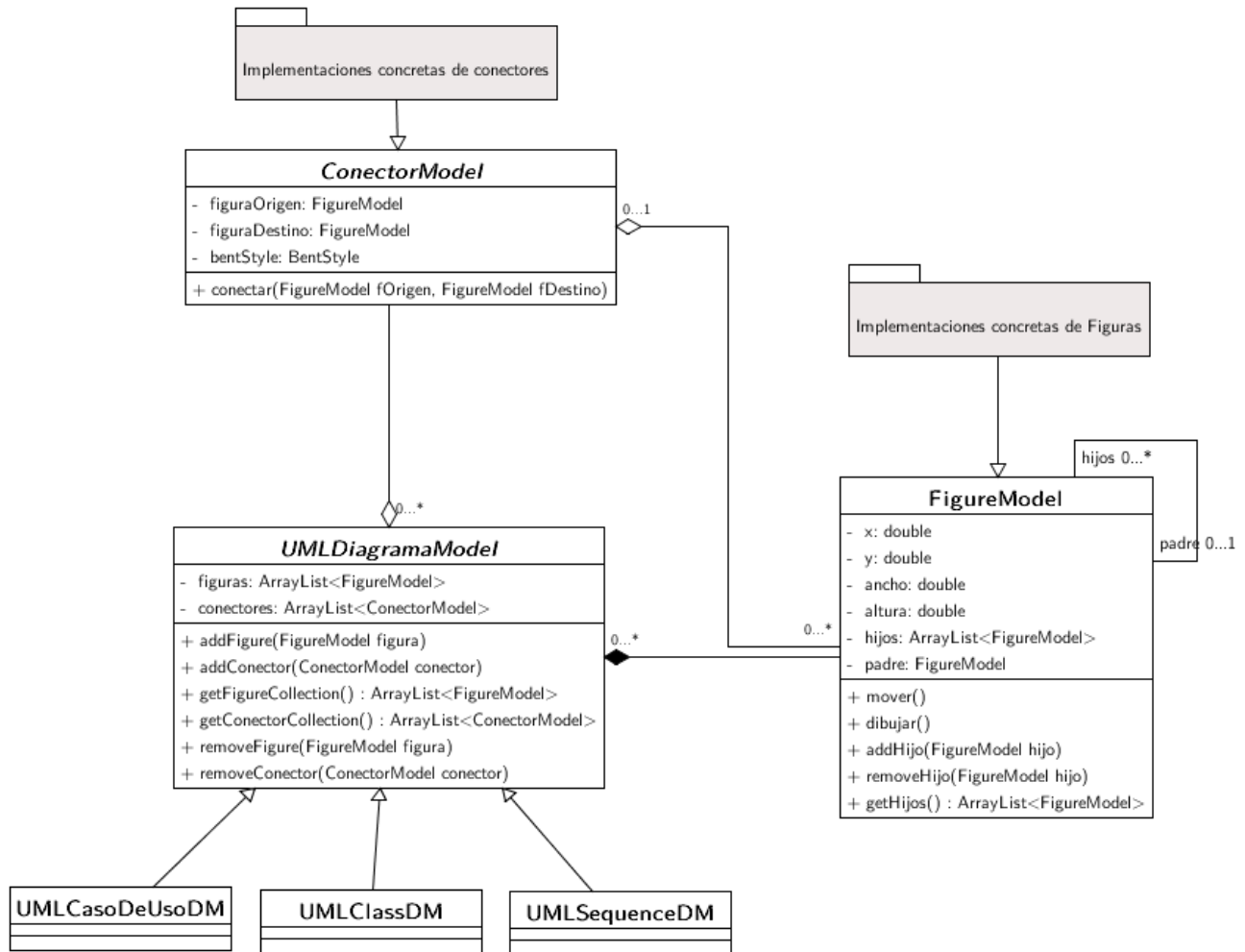


Diagrama 1: Diagrama de clases en alto nivel para modelo del problema.

El Diagrama 1 muestra las clases generadas que representan el modelo de la aplicación en el esquema Modelo-Vista-Controlador para la resolución del problema. La clase **UMLDiagramaModel** representa algún tipo de diagrama y de éste se realizan las implementaciones concretas del mismo, del cual se generarán los distintos diagramas de UML. La clase contiene una colección de figuras y conectores, mismos que representan a los distintos artefactos UML que puede utilizar y que varían en función del tipo de diagrama UML que se quiere representar. La clase **ConectorModel** será la clase base para representar las relaciones entre las figuras de un diagrama UML y de la cual se heredarán se realizarán las implementaciones concretas. Lo mismo ocurre para la clase **FigureModel** que representa a la clase base de la cual heredarán las implementaciones concretas para cada figura de un diagrama UML.

### 6.2.3.2. Vistas

El modelo del diagrama UML puede tener varias representaciones las cuales son mostradas en la aplicación. Cada vista cambia en función de los datos que contiene el modelo que se observa. En el diseño de la solución del problema se identificaron 3 vistas que pueden generarse a partir del modelo.

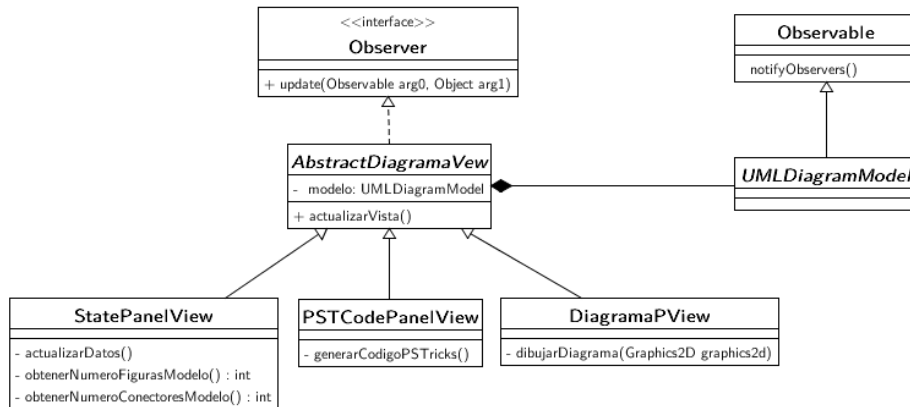


Diagrama 2: Diagrama de clases para diseño de las vistas de la aplicación.

El Diagrama 2 muestra el diagrama de clases que se utilizará para generar las diferentes vistas del modelo de diagrama UML de la aplicación. Para la implementación de las vistas se utilizó el patrón de diseño *Observador* [19], representado por las clases **Observable** que hereda el modelo de diagrama. Todas las vistas que deseen observar el modelo deberán implementar la interfaz **Observable** la cual se encargará de monitorear los cambios asociados al modelo de datos. El modelo representado por la clase **UMLDiagramModel** se encargará de notificar a todas las vistas que tenga asociadas cada ocasión que reciba un cambio en su estructura.

Se detectó que las vistas a generar tienen como común que se pueden representar en un panel que puede estar presente en la interfaz de usuario, por esta razón creará una clase abstracta que encapsule los datos y métodos comunes a todas las vistas. Esto es representado por la clase **AbstractDiagramaView**.

#### 6.2.3.2.1. Vista 1. Representación gráfica del modelo

Para representar gráficamente el modelo se necesita un módulo que pueda renderizar las figuras y conectores que están definidos dentro del diagrama. El Diagrama 3 muestra cómo se diseñó el módulo para generar la vista gráfica del modelo de diagrama UML.



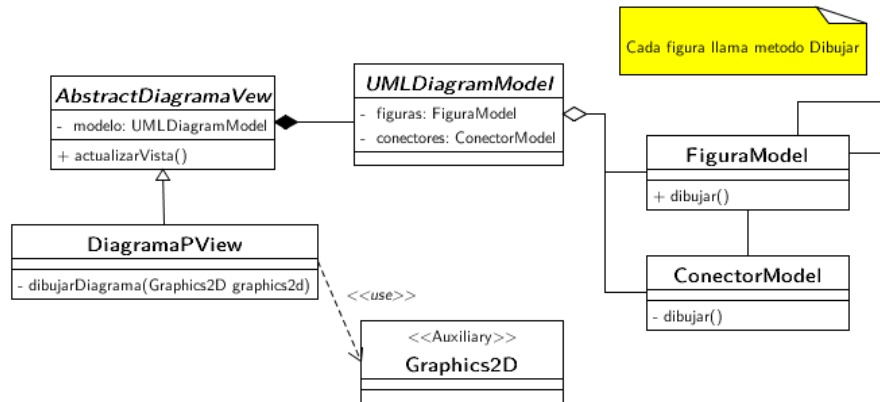


Diagrama 3: Diagrama de clases del diseño de la vista de representación gráfica del modelo.

La clase **DiagramaPView** representa la vista que renderiza el modelo de diagrama UML, la cual se auxilia de la clase **Graphics2D** [20], que es un motor de renderizado para utilizar en Java, que permite generar gráficos. El método **dibujarDiagrama** utiliza una instancia de Graphics2D para hacer que el modelo de diagrama UML vaya dibujando la colección de figuras y conectores que contiene y vayan llamando recursivamente al método **dibujar** definido en las clases **FiguraModel** y **ConectorModel**.

De esta manera, cada vez que el modelo se actualice, la vista hará el llamado al método dibujarDiagrama para dibujar la vista (ver Figura 24).

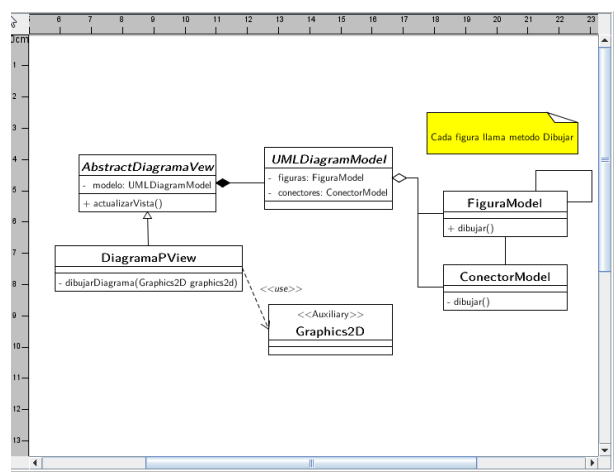


Figura 24: Ejemplo de la vista del modelo del diagrama en la aplicación.

### 6.2.3.2.2. Vista 2. Representación en forma de información de los elementos del modelo

La tercera vista requerida por la aplicación es mostrar la información básica del modelo como saber el número de figuras que contiene o el número de conectores, o bien saber el estado de cambio del modelo para saber si ya fue guardado por la aplicación o necesita guardarse.

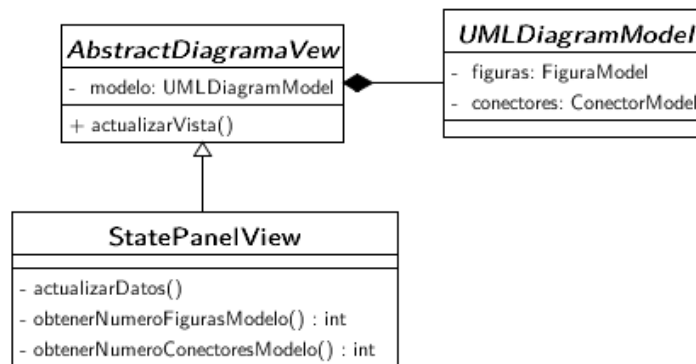


Diagrama 4: Diagrama de clases para vista de información general del modelo.

El Diagrama 4 muestra los métodos que se utilizarán para obtener la información del estado del modelo como lo son el número de elementos que contiene, si ya está grabado o si ya cambió y actualizar los datos en la vista que representa la clase **StatePanelView**.

La Figura 25 muestra un ejemplo de la vista generada.

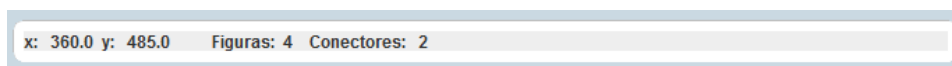


Figura 25: Ejemplo de la vista que muestra la información básica del modelo de diagrama.

### 6.2.3.2.3. Vista 3. Representación en código PStricks del modelo

La siguiente vista requerida por la aplicación es el equivalente del diagrama en forma de macros PStricks.

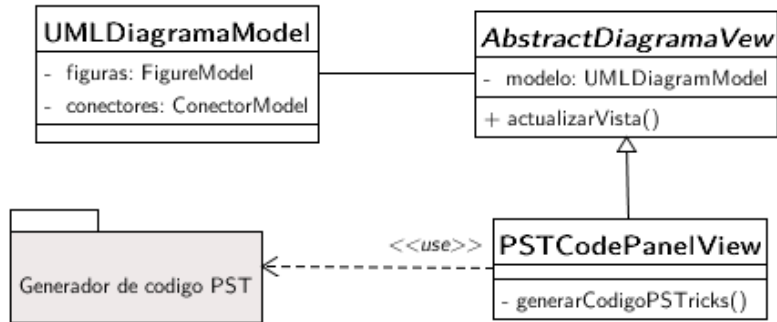


Diagrama 5: Diagrama de clases del diseño de la vista para macros PStricks.

En el diagrama de clases mostrado en Diagrama 5 se observa los elementos que entran en juego para la generación de la vista del diagrama UML en su equivalente PStricks. La clase **PSTCodePanelView** se encargará de generar la vista del modelo en forma de macros. Para realizar esto se ayudará de un módulo que se encargará de realizar la traducción indicada por el paquete “Generador de código PST”.

### 6.2.3.3. Módulo traductor a código PStricks

El módulo que realizará la traducción del modelo del diagrama hacia código PStricks se construye de varias clases y utilidades, las cuales se pueden visualizar en el Diagrama 6.

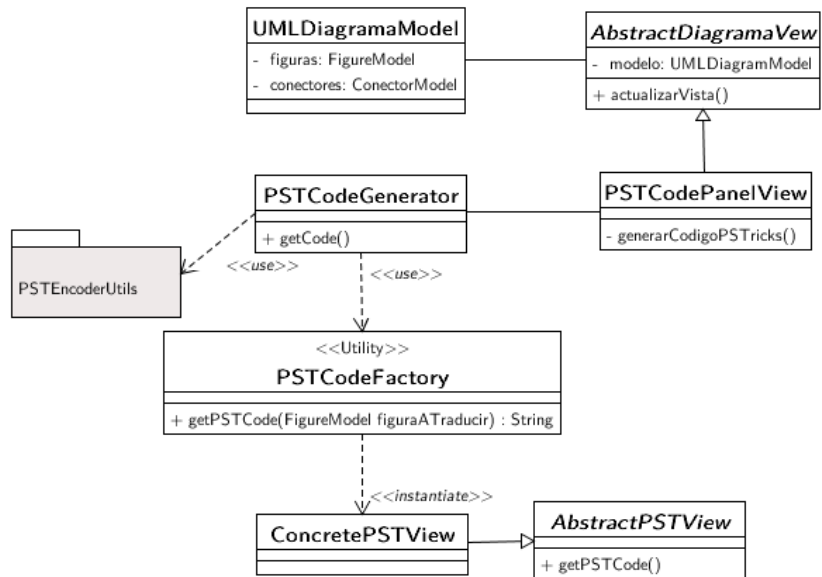


Diagrama 6: Diagrama de clases del módulo traductor.

La clase **PSTCodeGenerator** se encargará de realizar la traducción de cada uno de los elementos (figuras y conectores) que construyen el diagrama. Para realizar esta tarea se apoya de una fábrica de vistas **PSTCodeFactory** implementando así el patrón de diseño creacional *Factory* [19], que servirá para crear instancias que heredan la clase **AbstractPSTView**. La fábrica obtendrá una instancia del objeto correspondiente a cada figura del diagrama UML y realizará su traducción.

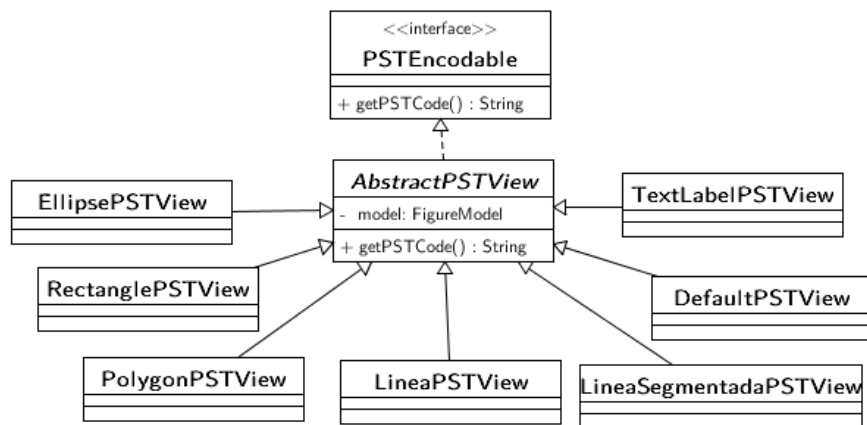


Diagrama 7: Diagrama de clases de la jerarquía de las vistas para generar las macros PSTricks.

El Diagrama 7 muestra la jerarquía de clases que implementan las vistas para generación de macros PSTricks para las figuras básicas. Todas implementan el método **getPSTCode()**

y tienen asociado un modelo del tipo *FigureModel*, al cual hace referencia para realizar la traducción.

El modulo traductor hace uso de clases que le servirán de ayuda para realizar la traducción y armar un documento LaTeX coherente y que pueda ser utilizable en una aplicación que genere documentos TeX.

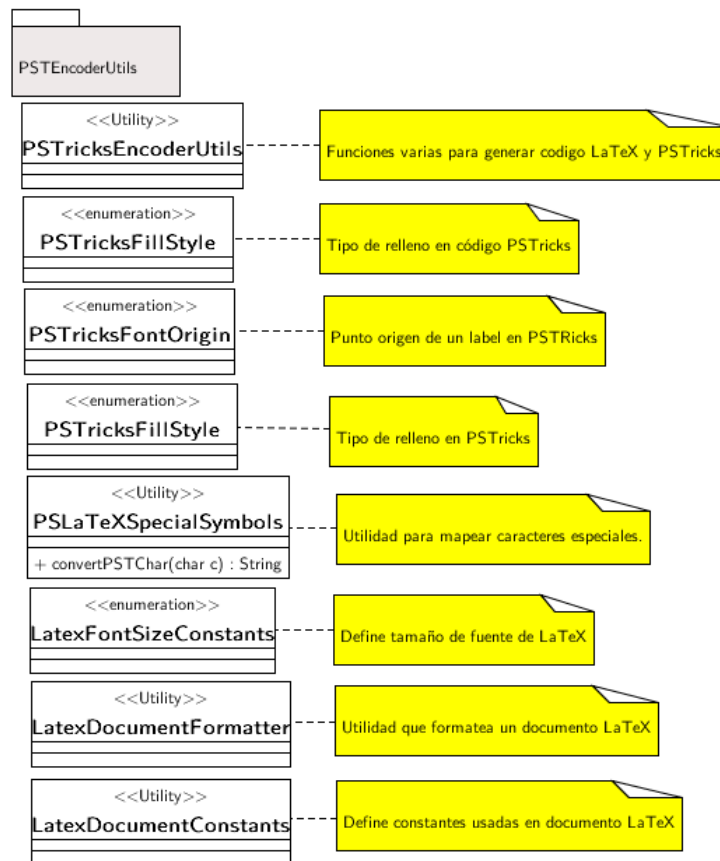


Diagrama 8: Diagrama de clases de las utilidades diseñadas para realizar la traducción de código LaTeX.

El Diagrama 8 muestra algunas de las clases que se diseñaron para el módulo traductor del modelo de diagramas, la cual consta principalmente de utilidades y definiciones que apoyarían al codificador de documentos LaTeX a generar las macros adecuadas.

### 6.2.3.4. Módulo para implementar la persistencia

Otro de los objetivos particulares del proyecto de integración fue el desarrollo del módulo que implementaría la persistencia de los diagramas generados por la aplicación, de tal forma que existiera una manera de persistir y recuperar los diagramas UML.

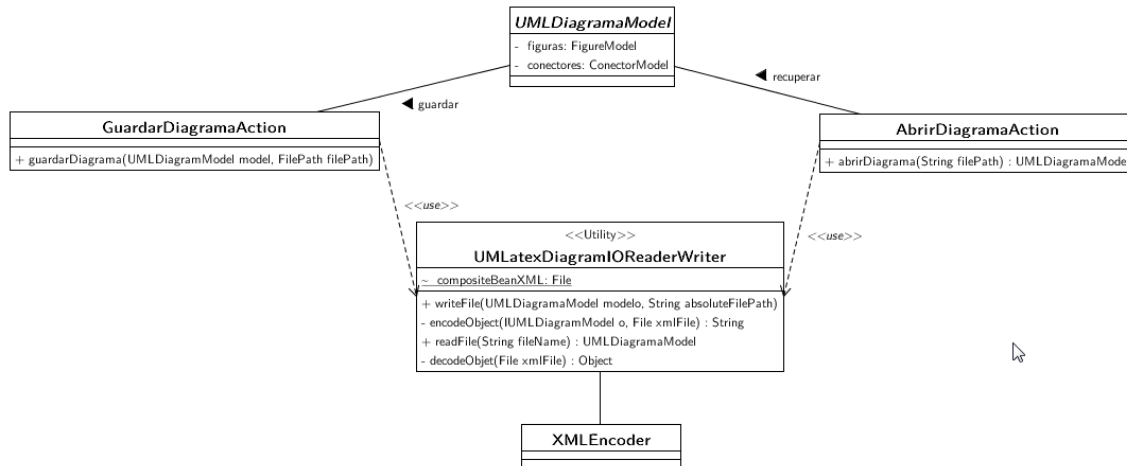


Diagrama 9: Diagrama de clases representando el módulo de persistencia.

El Diagrama 9 muestra a la clase **UMLatexDiagramIOReaderWriter** la cual es una clase utilidad que se encargará de escribir un archivo utilizando la clase **XMLEncoder** y **codifica** el modelo del diagrama en el lenguaje XML [21]. Adicionalmente se utiliza la misma clase para recuperar un archivo XML y **decodificarlo** de tal manera que si resulta exitosa la lectura y decodificación se recupera un modelo de diagrama que posteriormente es mostrado en el área de trabajo de la aplicación.

Cabe mencionar que el módulo está preparado para actuar ante cualquier eventualidad de errores de lectura y escritura de archivos.

### 6.2.3.5. Módulo para exportar archivos

La aplicación puede persistir el modelo de diagrama de otra manera. La aplicación puede exportar los diagramas generados en distintos formatos de archivo.

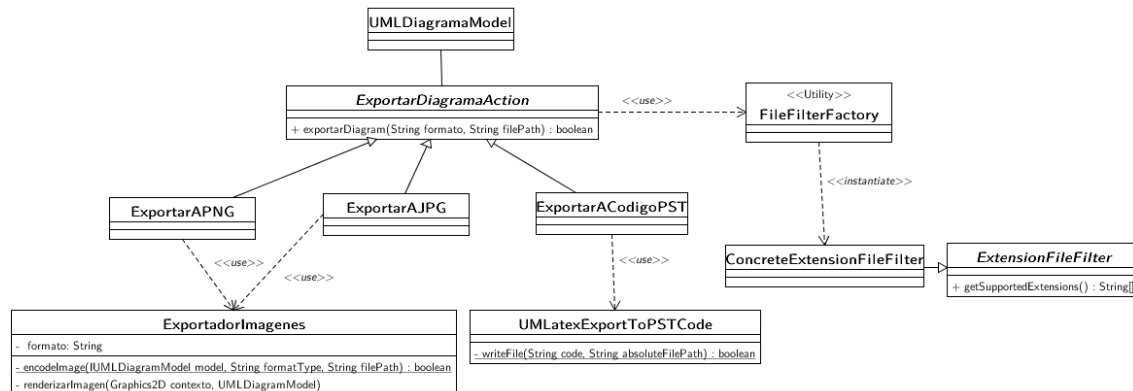


Diagrama 10: Diagrama de clases para la implementación del módulo de persistencia por exportar.

El diagrama de clases mostrado en Diagrama 10 muestra el diseño generador del módulo para exportar el diagrama en distintos formatos. La clase **ExportarDiagramaAction** es heredada de acuerdo al tipo de formato en el cual se quiere preservar el diagrama. Por ejemplo, para imágenes se utiliza la acción **ExportarAPNG** o **ExportarAJPG**, de las cuales utilizan otra clase llamada **ExportadorImagenes** la cual se encargará de llevar la tarea de renderizar el contenido del modelo de diagrama en el formato de imagen especificado por el usuario.

La clase **ExportarACodigoPST** puede exportar el diagrama en un archivo de texto \*.tex utilizando el código generado por el módulo de traducción y también se auxilia de la clase **UMLatexExportToPSTCode** que lleva toda la lógica para poder escribir en archivo de texto el código generado.

Este módulo utiliza como ayuda una factoría de filtros de extensión de archivo **FileFilterFactory** la cual se encargará de crear instancias de las clases concretas que heredan de **ExtensionFileFilter**. Esta clase ayudará a guardar y ubicar los archivos resultantes en el formato correcto.

### 6.2.3.6. Interfaz de usuario.

La interfaz del usuario o GUI tiene que ser modelada de manera que el usuario tenga fácil entendimiento y uso de la herramienta.

Debe proveer las herramientas y un área de trabajo para la creación y edición de los diagramas que se desean trabajar.

En la Figura 26 se muestra un ejemplo de la interfaz de usuario.

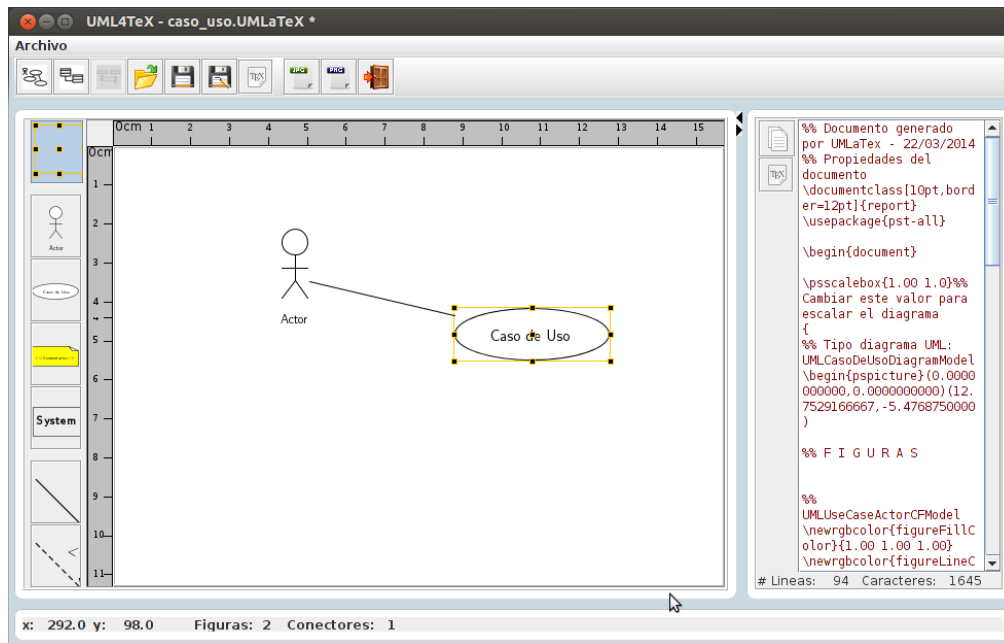


Figura 26: Interfaz de usuario para la aplicación.

Se mencionan a continuación los elementos más importantes en la interfaz 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.

Parte importante del diseño de la interfaz es que las clases generadas deben proveer los métodos para poder informar sobre los cambios del modelo y así actualizar las vistas del modelo (en este caso, los diagramas dibujados, así como el código de las macros de PSTricks).

Se analizan a continuación las clases diseñadas por la interfaz.



### 6.2.3.6.1. Clase de la aplicación principal

Esta clase representa la aplicación principal y es la que coordina la interacción de todos los componentes con el usuario.

El Diagrama 11 muestra el conjunto de clases que se obtuvieron en relación del diseño de la interfaz de usuario.

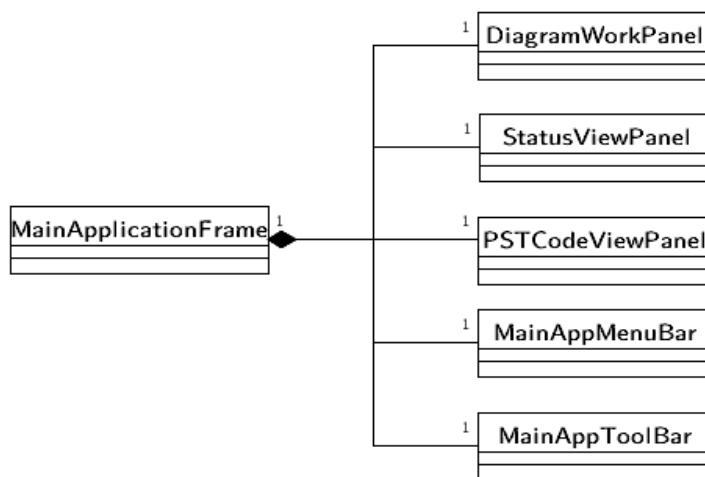


Diagrama 11: Diagrama de clases del diseño de la interfaz de usuario de la aplicación.

La aplicación representada por la clase **MainApplicationFrame** contiene una serie de componentes que representan los diferentes aspectos de la interfaz del usuario.

### 6.2.3.6.2. Clase Barra de Menú

La clase barra de Menú contiene una serie elementos que pueden elegirse y que están ligadas a diferentes tipos de acciones que pudiera realizar el usuario (ver Figura 27).

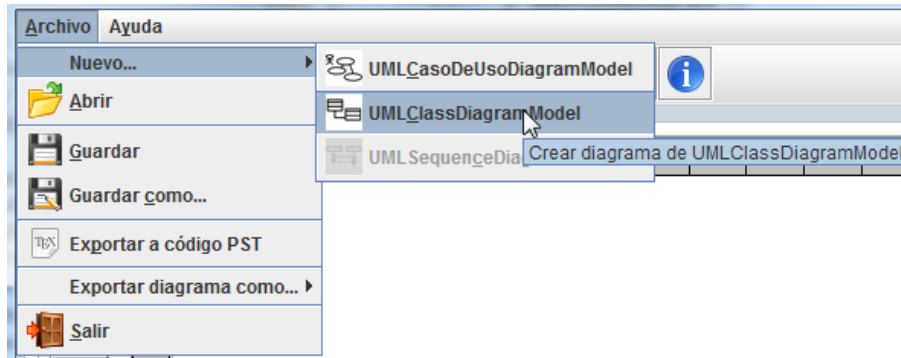


Figura 27: Ejemplo de barra de menú Archivo.

El Diagrama 12 muestra las clases relacionadas para la implementación de los menús y cómo se relacionan con las acciones que tienen asociadas. La clase **MainAppMenuBar** implementa una serie de elementos de menú (**JMenu** y **JMenuItem**) y a los cuales se les asocia una acción determinada para que en cada especialización de la clase **Action** realice una tarea determinada.

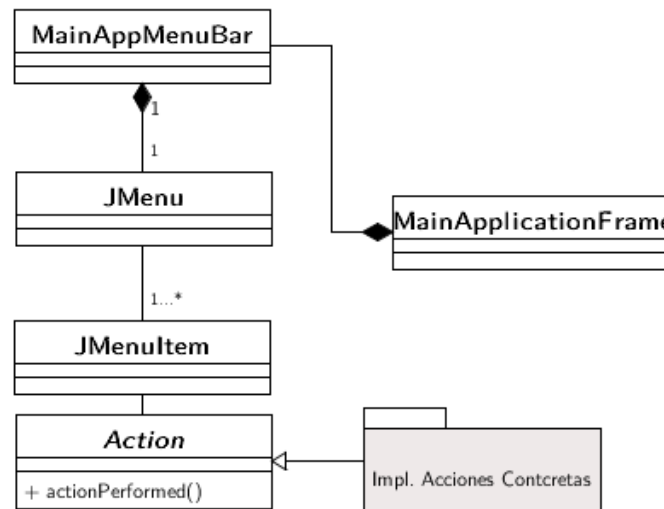


Diagrama 12: Diagrama de clases del diseño de menú de la aplicación.

### 6.2.3.6.3. Clase Barra de Herramientas

Similar a la barra de menús de la aplicación, la barra de herramientas contiene una serie de Botones a los cuales se les asocia algún tipo de comando o acción. En este es posible reutilizar las mismas acciones que se utilizan en la barra de menús (ver Figura 28).



Figura 28. Ejemplo de un diseño de barra de herramientas de la aplicación.

El Diagrama 13 muestra las relaciones entre las clases que conforman la barra de herramientas de la aplicación. La clase **MainAppToolBar** contiene una serie de botones a los cuales se les asocia una acción genérica y de la cual sus implementaciones concretas se encargan procesar la acción concreta que desea el usuario en la aplicación (ejemplo: guardar un diagrama en archivo).

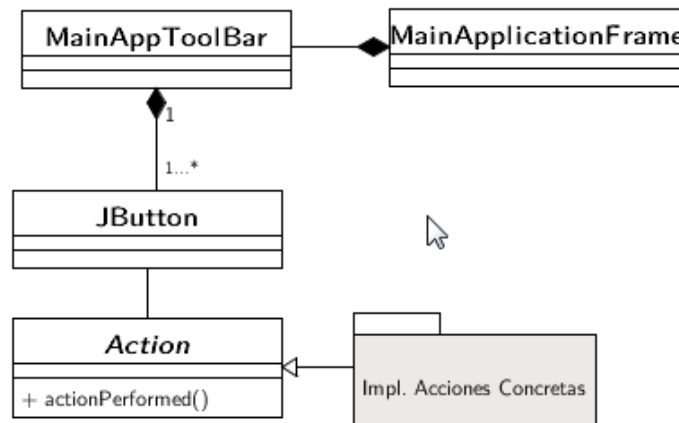


Diagrama 13: Diagrama de clases para diseño de Barra de Herramientas.

Ha de notarse que las acciones pueden ser compartidas entre las clases de Barra de Menús y la de Barra de Herramientas para evitar realizar desarrollo extra para cada componente que realice la misma tarea.

#### 6.2.3.6.4. Diseño de las acciones realizadas por la aplicación

La aplicación se puede modularizar en distintas acciones que representan una tarea específica de las operaciones que puede realizar el usuario. Es por eso que también se diseñaron de tal manera que estén relacionadas ya que comparten características comunes como el nombre o un icono. Por lo tanto, definió una clase abstracta que describe las características comunes de las distintas acciones y después se deja la implementación a cada una de las acciones concretas.

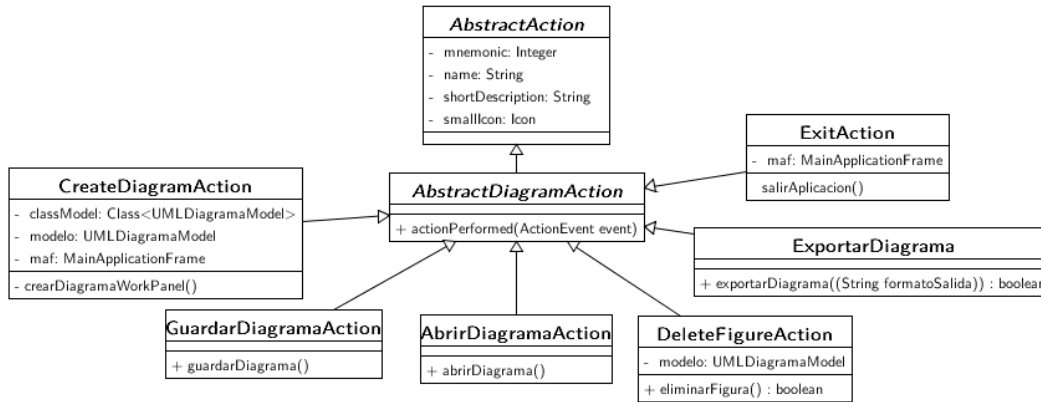


Diagrama 14: Diagrama de clases del conjunto de algunas acciones realizadas por la aplicación.

El Diagrama 14 muestra la jerarquía de clases que constituyen algunas de las acciones que realiza la aplicación y que funcionan a modo de controladores de la aplicación. Es decir a cada acción corresponde un comando y una actividad a realizar, llamando al módulo correspondiente que resuelve la tarea asignada.

### 6.2.3.6.5. Clase Área de Trabajo

El Contenedor o Área de Trabajo de la aplicación muestra un área de dibujo con la barra de herramientas asociada a un tipo de diagrama, es decir, por cada tipo de diagrama se tiene una instancia de barra de herramientas de diagrama, así como su propia área de dibujo (ver Figura 29).

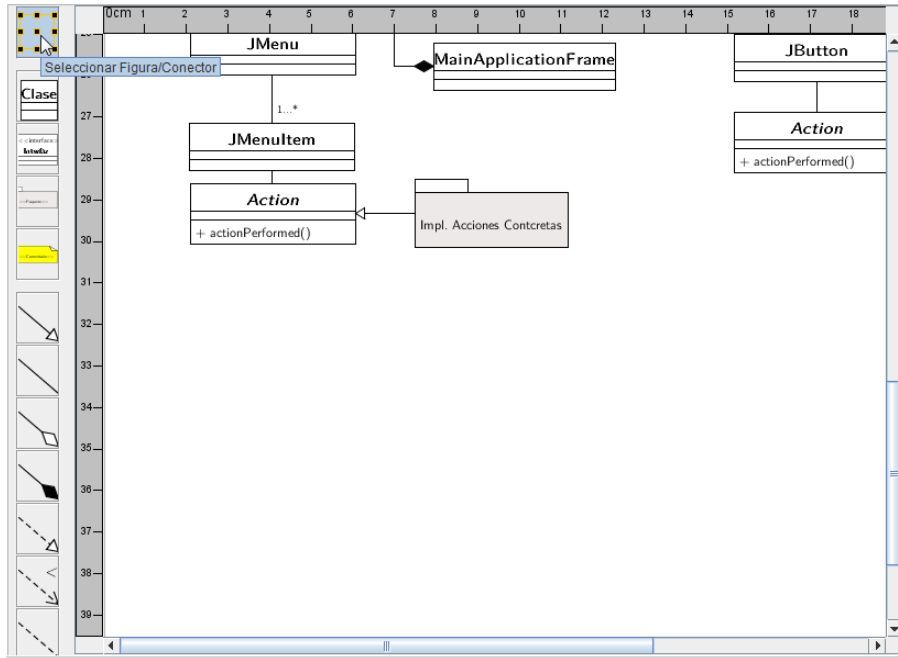


Figura 29: Ejemplo del área de trabajo de la aplicación.

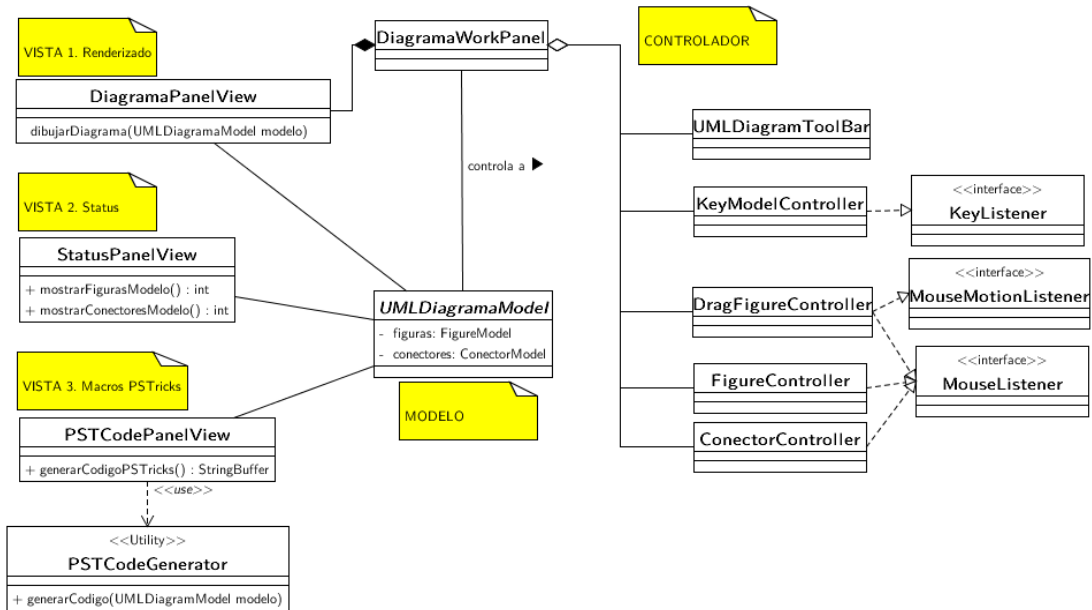


Diagrama 15: Diagrama de clases del área de trabajo de la aplicación.

El en Diagrama 15 se puede observar el diagrama de clases que conforma el diseño del área de trabajo de la aplicación y cómo está estructurada de manera que se apega al patrón de diseño **Modelo-Vista-Controlador**.

La clase *DiagramaWorkPanel* representa el área de trabajo de la aplicación, y es la encargada de integrar los distintos módulos que representan a cada uno de los aspectos de la aplicación:

**Modelo:** Representada por la clase *UMLDiagramModel* y la cual se explica en la sección 6.2.3.1 es el modelo sobre el cual se trabajará.

**Controlador:** La clase *DiagramaWorkPanel* resulta ser el controlador principal ya que se encarga de dirigir todos los eventos de entrada del usuario a subcontrolador que corresponda. Cada controlador puede modificar un aspecto del modelo, por ejemplo la clase *DragFigureController* se encarga de procesar los eventos recibidos por el mouse para controlar los movimientos de la figura en el panel de dibujo. Más adelante se tratarán las clases que fungen como controladores.

**Vistas:** El modelo del diagrama UML puede tener varias representaciones las cuales son mostradas en la aplicación. Cada vista cambia en función de los datos que contiene el modelo que se observa. Como se puede ver en el diagrama de clases, el modelo cuenta con tres vistas diferentes. Por ejemplo, la clase *DiagramaPanelView* representa el módulo que dibujará el modelo de diagrama UML en su representación gráfica.

### 6.2.3.7. 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.

#### 6.2.3.7.1. Clase *UMLDiagramToolBar*

Para comenzar a dibujar artefactos UML es necesario contar con un catálogo de figuras y conectores y una herramienta que les permita poder dibujar en el área de edición.

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.

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 utilidad *AFigureAndConectorControllerFactory* que es una fábrica abstracta para instanciar el controlador de figura o conector adecuado para poder dibujarlo en el área de trabajo (ver Diagrama 16).

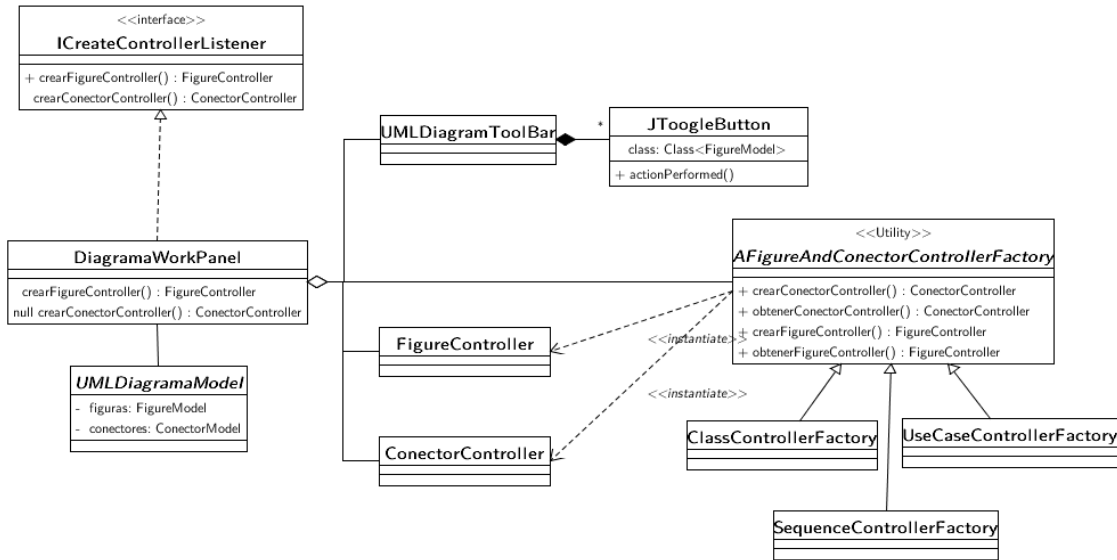


Diagrama 16: Diagrama de clases de modelo de controladores relacionados con UMLDiagramToolBar

### 6.2.3.7.2. Clase Controlador de Figuras

Este tipo de controlador se encargará de recibir y procesar todos los eventos que estén relacionado con un artefacto UML que represente una Figura.

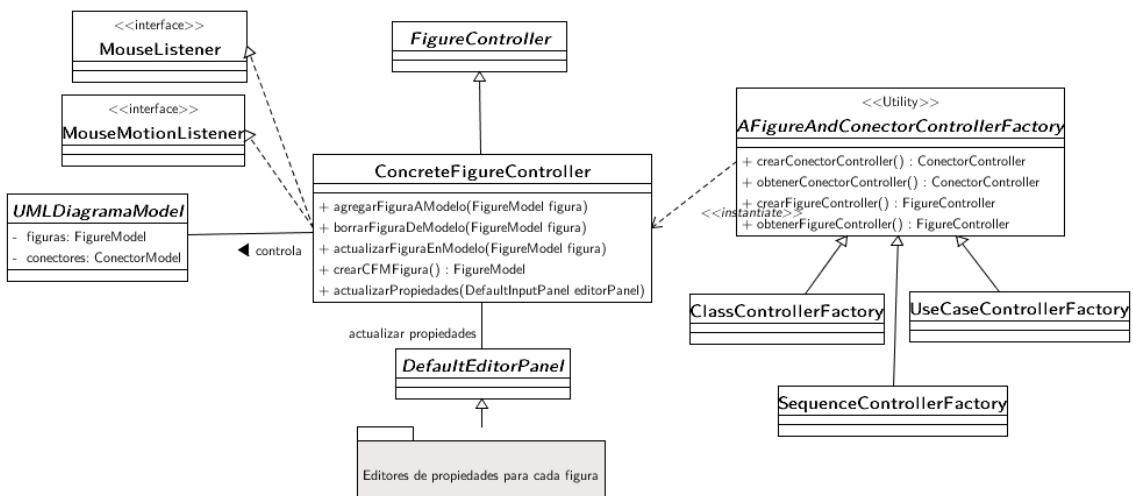


Diagrama 17: Diagrama de clases para el controlador de figuras

Todo controlador de figuras **ConcreteFigureController** deberá ser obtenido a partir de una factoría de controladores que herede de la clase **AFigureAndConectorControllerFactory**.



En el Diagrama 17 se puede observar como las clases *ClassControllerFactory*, *UseCaseControllerFactory* y *SequenceControllerFactory*, de esta manera, para cada tipo de modelo de diagrama UML que se esté trabajando en la aplicación, le corresponderá una y solo una factoría de controladores y estos a su vez, tendrán asociados un número limitado de las figuras que soporta.

El controlador de figuras tiene como base una clase abstracta *FigureController* que define cuales son las acciones deberá realizar cualquier controlador de figuras que la implemente. En específico el controlador de figuras deberá realizar las siguientes tareas:

- **Crear y agregar** una figura al modelo
- **Eliminar** una figura del modelo
- **Actualizar las propiedades** de una figura en el modelo.

El controlador de figuras también implementa métodos del mouse para reconocer cuando una figura está creándose o bien para despertar el flujo de actualización de propiedades.

Cuando el controlador recibe el evento para actualizar las propiedades de la figura seleccionada en el modelo de figuras, crea un Editor de propiedades, el cual es representado por la clase abstracta *DefaultEditorPanel*, y para la cual existen sus implementaciones concretas que mostrarán al usuario un cuadro de diálogo con las opciones que editará para cada clase del modelo de Figura.

#### **6.2.3.7.3. Clase Controlador de Conectores**

Este controlador se encargará de interceptar y procesar los eventos relacionados con un artefacto UML que represente un conector (ver Diagrama 18).

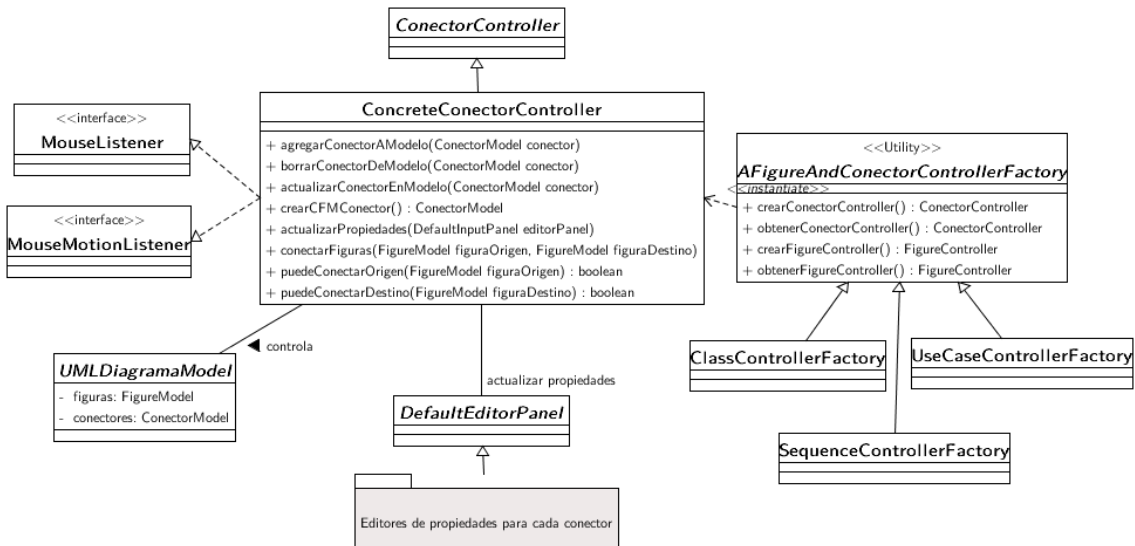


Diagrama 18: Diagrama de clases que representa el diseño del controlador de conectores.

Todo controlador de figuras *ConcreteConectorController* deberá ser obtenido a partir de una factoría de controladores que herede de la clase *AFigureAndConectorControllerFactory*, de esta manera, para cada tipo de modelo de diagrama UML que se esté trabajando en la aplicación, le corresponderá una factoría de controladores los cuales tendrán asociados un número limitado de las conectores.

El controlador de conectores tiene como base una clase abstracta *ConectorController* que define cuales son las acciones deberá realizar cualquier controlador de figuras que la implemente. En específico el controlador de figuras deberá realizar las siguientes tareas:

- **Crear y agregar** un conector al modelo
- **Eliminar** un conector del modelo
- **Actualizar las propiedades** de un conector en el modelo.
- **Conectar** dos figuras.
- **Evaluar** si las figuras a conectar son válidas para el controlador de conectores actual.

El controlador de conectores también implementa métodos del mouse y puede recibir los eventos para actualizar las propiedades del conector al que controla. Esto también lo hace usando un Editor de propiedades, el cual es representado por la clase abstracta *DefaultEditorPanel*, y para la cual existen sus implementaciones concretas que mostrarán al usuario un cuadro de diálogo con las opciones que editará para cada clase del modelo de Conector..

#### 6.2.3.7.4. 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 19).

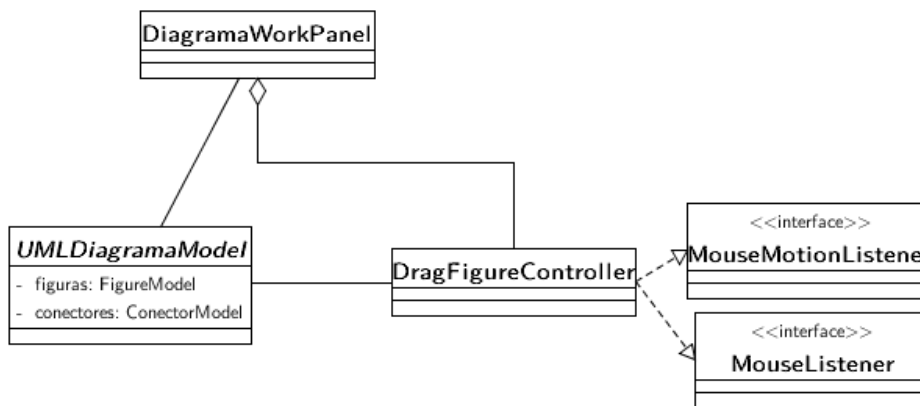


Diagrama 19: Diagrama de clases del controlador de arrastre de figuras.

Estos eventos de mouse normalmente estarán asociados a los cambios de posición de las figuras o conectores UML que se encuentren en el diagrama

#### 6.2.3.7.5. Clase KeyModelController

Este controlador está encargado de interpretar aquellos eventos que estén relacionados con el teclado. Por lo tanto por eso implementa la interfaz *KeyListener*.

Las acciones que tiene implementadas para controlar el modelo del diagrama es cuando presionan la tecla **DELETE** para borrar la figura si esta seleccionada, o bien, desplazarla en el área de trabajo con las flechas del teclado.

El Diagrama 20 muestra el diseño de esta clase y como se relaciona con el modelo y con el diagrama de trabajo que lo contiene.

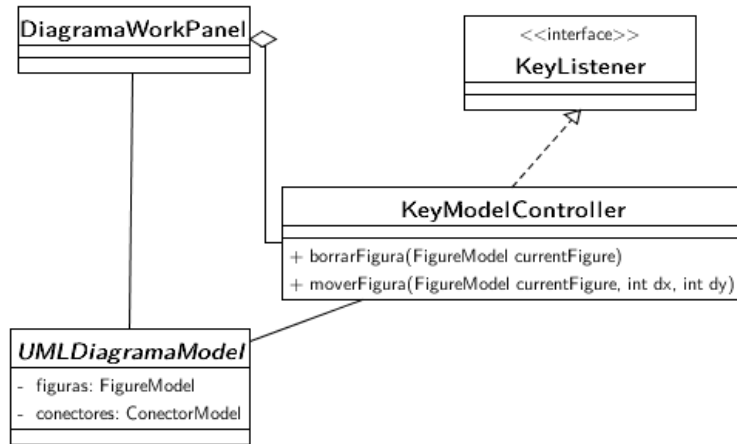


Diagrama 20: Diagrama de clases para el Controlador de Teclado

Se pueden ir agregando más acciones relacionadas con este controlador a futuro para permitir una interacción más dinámica entre el usuario y la aplicación.

#### 6.2.4. Arquitectura del sistema

En la Figura 30 se muestra el diagrama de la arquitectura funcional de la aplicación.

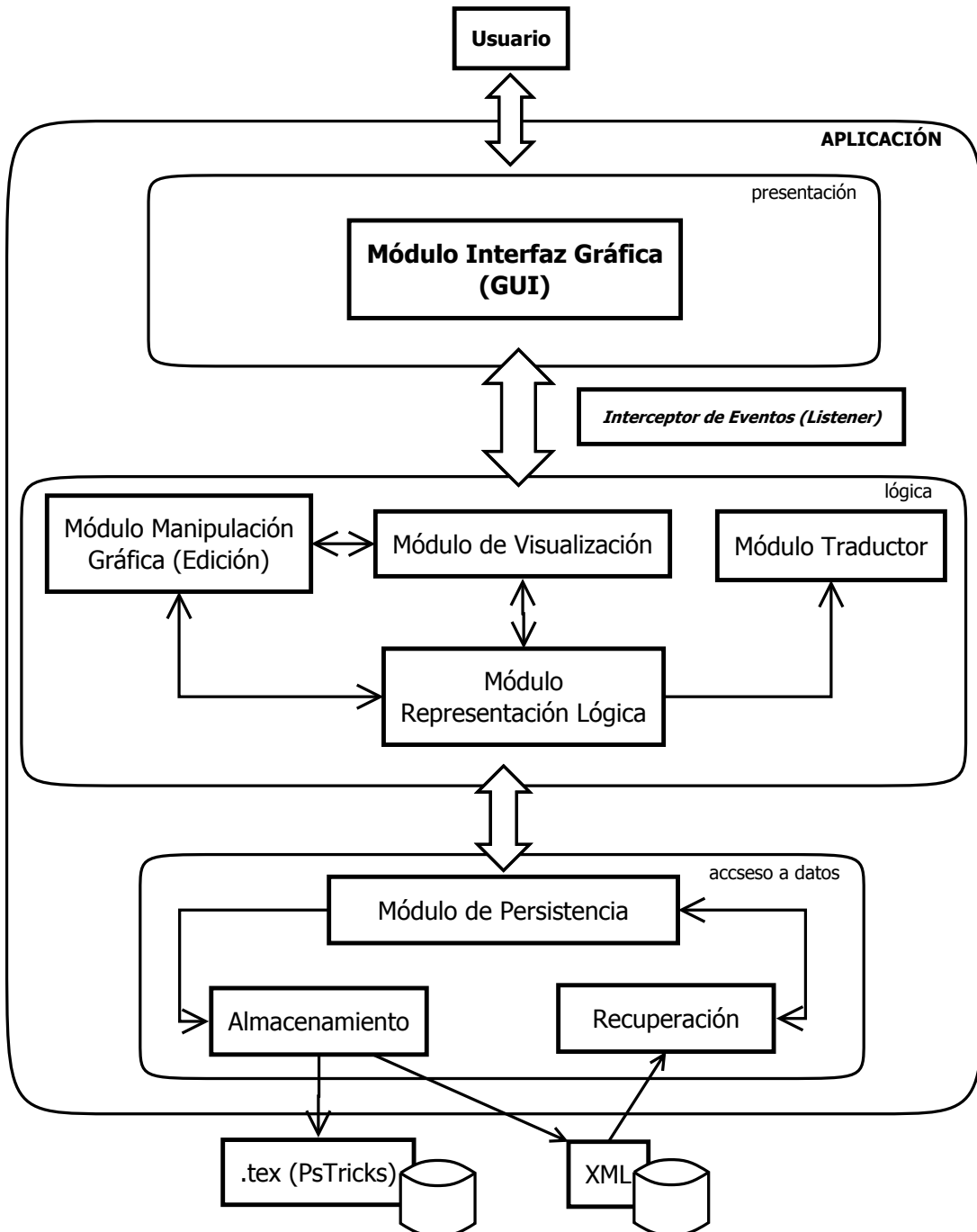


Figura 30: Diagrama de la arquitectura funcional de la aplicación.

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 [22]. De esta manera, se garantizó la separación de roles en la aplicación y hará más fácil la adición de módulos en trabajos futuros.

En la **capa de presentación** se implementa el módulo de la interfaz gráfica (GUI). Tiene los componentes necesarios para la interacción con el usuario.

- Un área de trabajo donde se podrán dibujar y editar los diagramas.
- El conjunto de menús y barras de herramientas necesarios para implementar las operaciones necesarias para la edición de los diagramas.

El módulo interceptor de eventos, es el encargado de enlazar con las capas subsecuentes y de crear la interacción de la interfaz gráfica con los demás módulos.

En la **capa de lógica de negocio** se encuentran descritos los módulos que definen el comportamiento de la aplicación al interactuar con el módulo interceptor de eventos, a saber:

- El módulo de representación lógica que es el encargado de construir el esquema de los diagramas UML; es decir, define los componentes que contiene cada uno de los diagramas UML y las propiedades asociadas a los mismos. En conjunto, define cómo está constituido cualquier diagrama.
- El módulo de visualización es el encargado de interpretar la representación lógica de los diagramas y sus componentes y comunicarlo a la capa de presentación para que este lo visualice en función de los valores de la representación lógica del diagrama.
- El módulo de edición (manipulación gráfica) tiene la labor de permitir la edición de los diagramas y en conjunto con el módulo de representación lógica tener la capacidad de cambiar la estructura de los diagramas y pasarlos al módulo de visualización para su despliegue en la interfaz.
- El módulo traductor es el encargado de convertir los datos contenidos en la estructura de los diagramas proporcionadas por el módulo de representación lógica

en macros PSTricks que, en conjunto con el módulo de persistencia, generará salidas en formato `.tex`.

Finalmente, la **capa de acceso de datos** está constituida por el módulo de persistencia, que a su vez, está formado por los módulos siguientes:

- Módulo de Almacenamiento. Con la ayuda del módulo de representación lógica, este módulo genera una salida para ser almacenada en algún dispositivo, conteniendo en un archivo la estructura y descripción lógica del diagrama almacenado. La salida es bajo el estándar XML.
- Módulo de Recuperación. Es el encargado de obtener de un medio de almacenamiento un archivo que contiene la información sobre la estructura de los diagramas UML (representación lógica) que hayan sido guardados previamente.
- También, en colaboración con el módulo traductor el módulo de persistencia es capaz de exportar las macros PSTricks generadas por éste en un archivo.

#### **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) [19] y [23].

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.

## 6.2.6. Uso del sistema

### 6.2.6.1. Crear Diagrama

1. El usuario solicita a la aplicación la realización de una operación de creación de nuevo diagrama. (Elegir menú **Archivo** → **Nuevo...** → [Elegir el diagrama que se desea crear]). (Ver Figura 31)

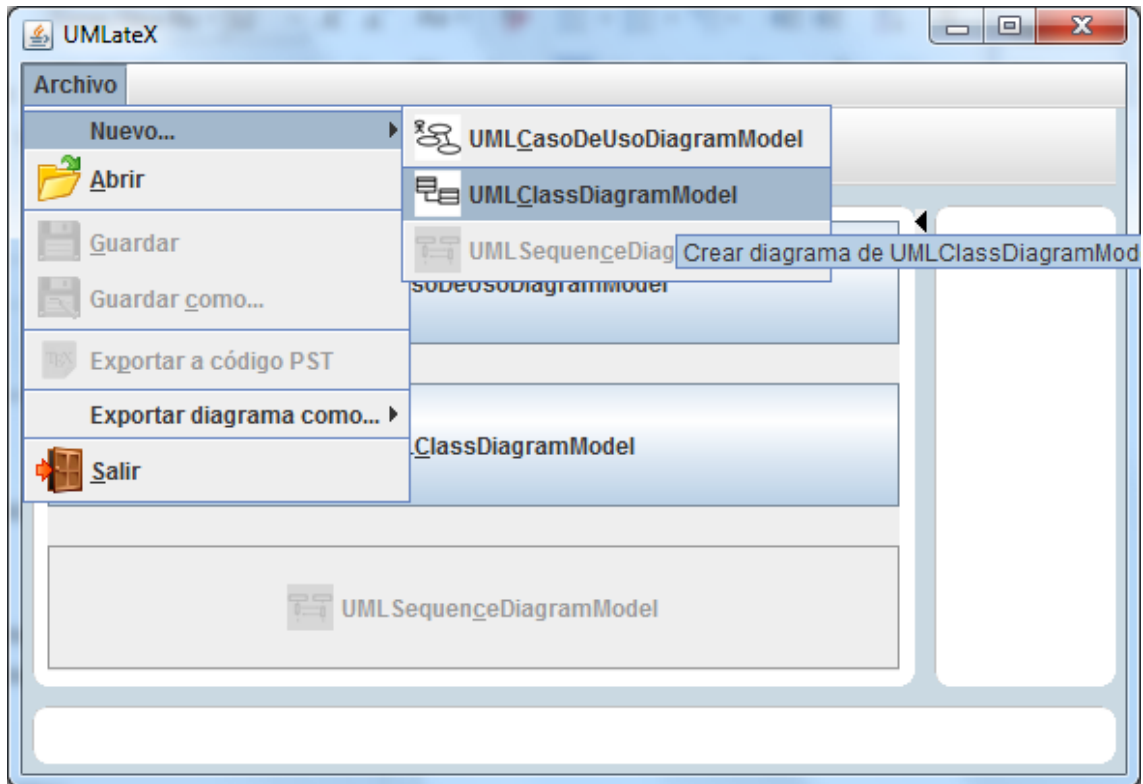


Figura 31: UMLATeX, elegir nuevo diagrama

2. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código). (Ver Figura 32)



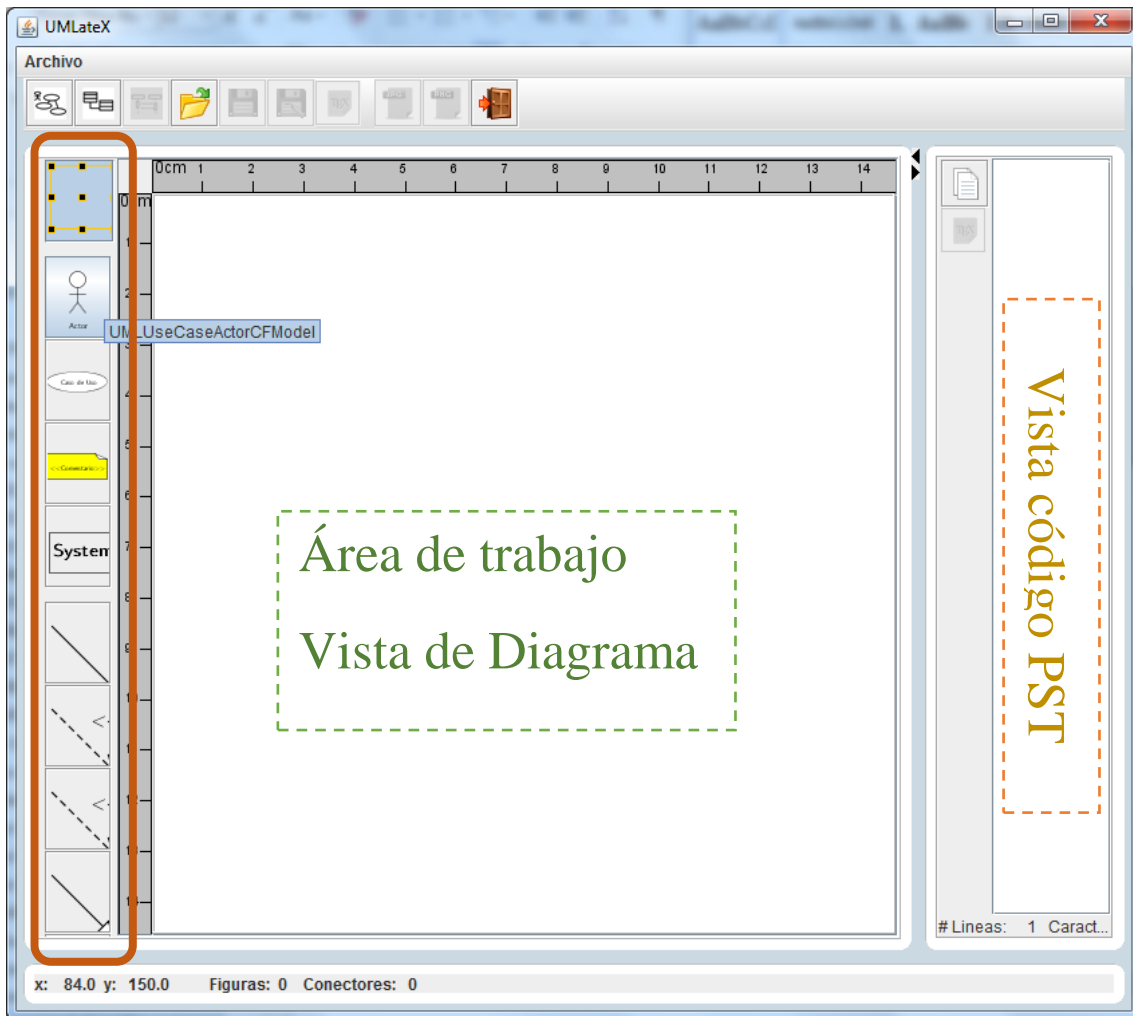


Figura 32: UMLateX, área de trabajo mostrada tras crear un nuevo diagrama. A la izquierda, la barra de herramientas correspondiente al diagrama UML elegido. Al centro el área de trabajo con la Vista del diagrama. A la derecha la Vista de código PSTricks.

### 6.2.6.2. Recuperar Diagrama

1. El usuario solicita a la aplicación recuperar (abrir) un diagrama previamente guardado por la aplicación (ver Figura 33).

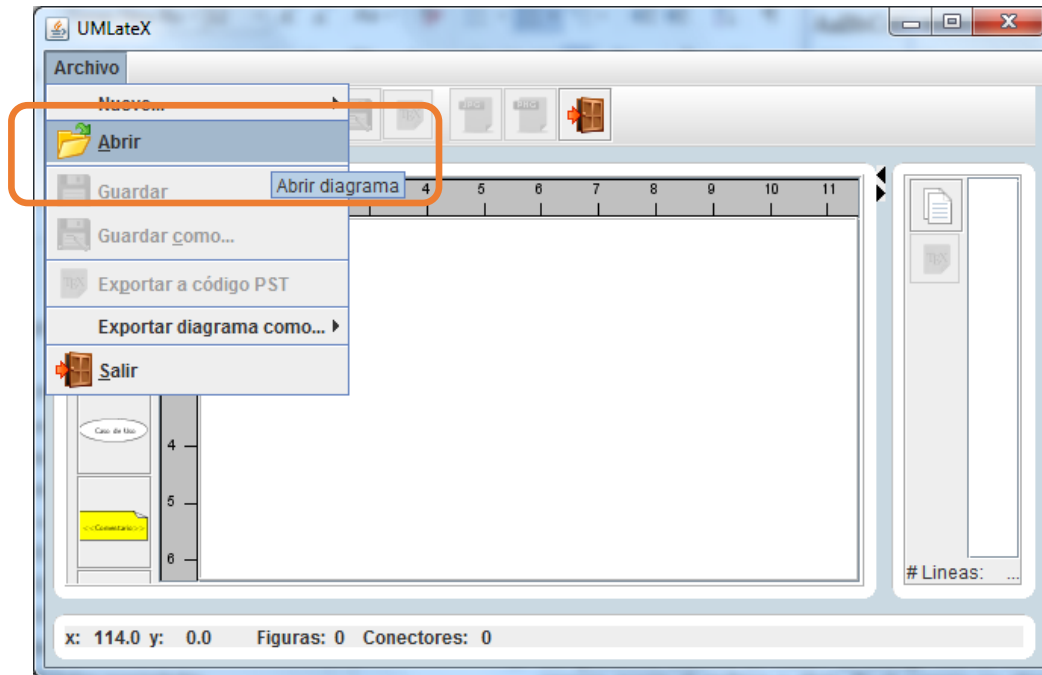


Figura 33: UMLaTeX. Abrir un diagrama

2. La aplicación muestra un cuadro de diálogo con un sistema de archivos para que el usuario escoja el archivo a abrir. (Ver Figura 34).

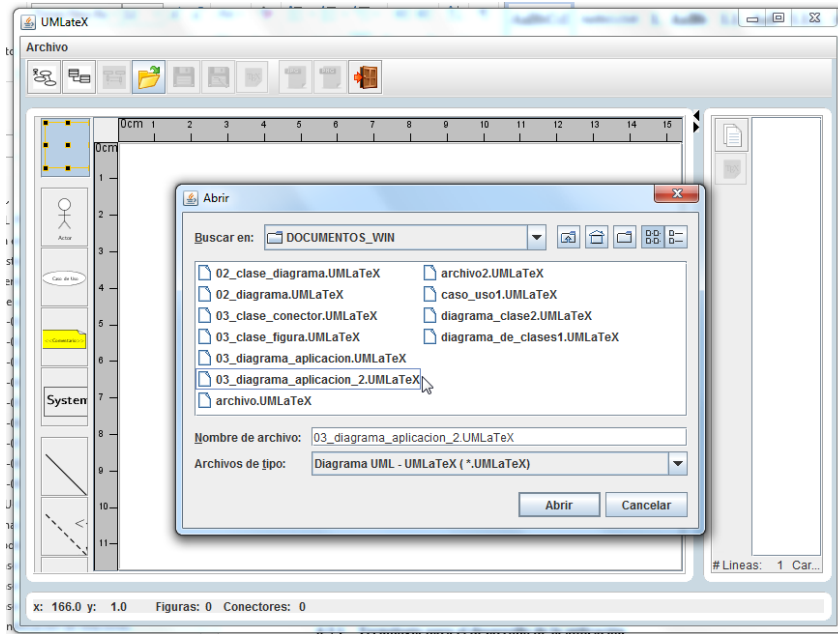


Figura 34: UMLaTeX. Cuadro de diálogo para abrir un diagrama \*.UMLaTeX

3. La aplicación muestra el diagrama recuperado (ver Figura 35).

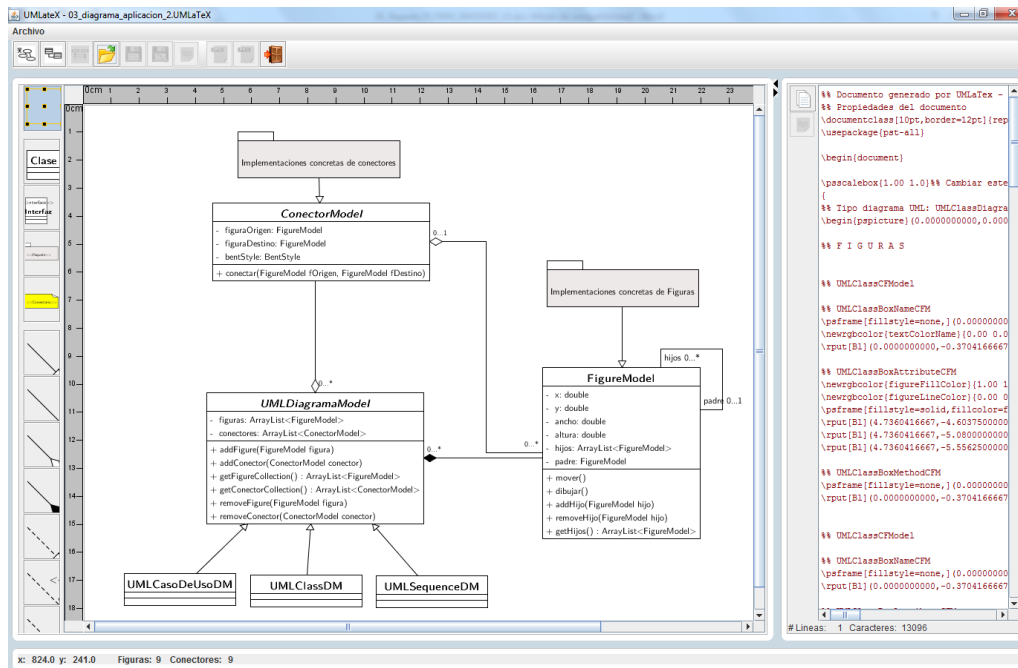


Figura 35: UMLaTeX. Muestra diagrama recuperado tras abrir el archivo.

### 6.2.6.3. Editar diagrama. Mover figuras

1. El usuario elige un artefacto UML (figura o conector) y lo arrastra (ver Figura 36).

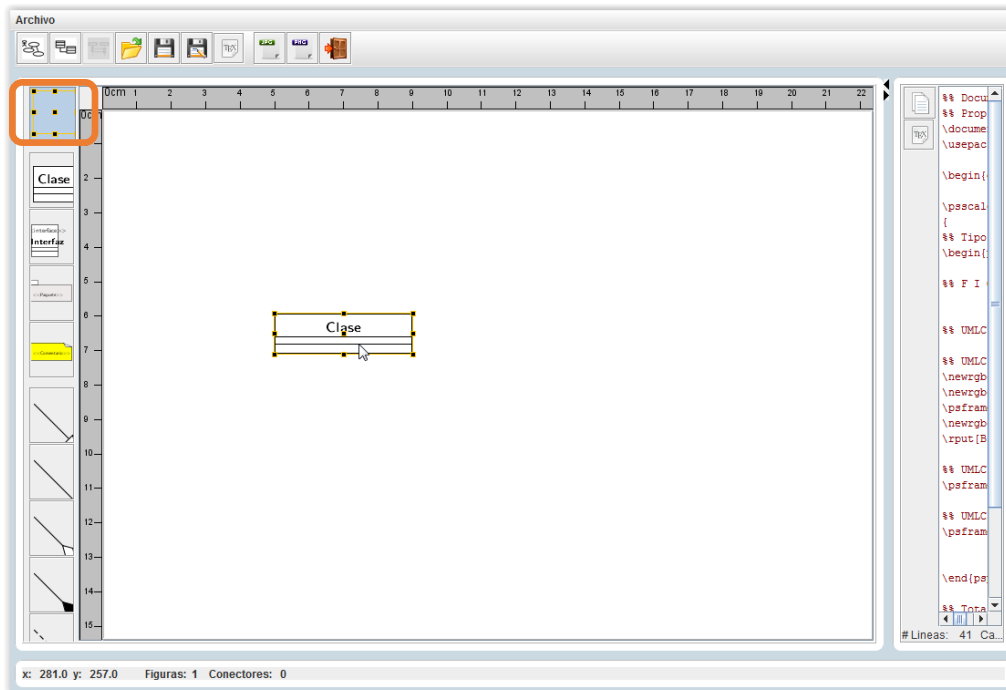


Figura 36:UML4TeX. La herramienta de selección de figura siempre estará en la parte superior de la barra de herramientas.

### 6.2.6.4. Agregar Figura/Conector

#### Agregar Figura:

1. El usuario elige de la barra de herramientas un artefacto UML (figura). (Ver Figura 37)

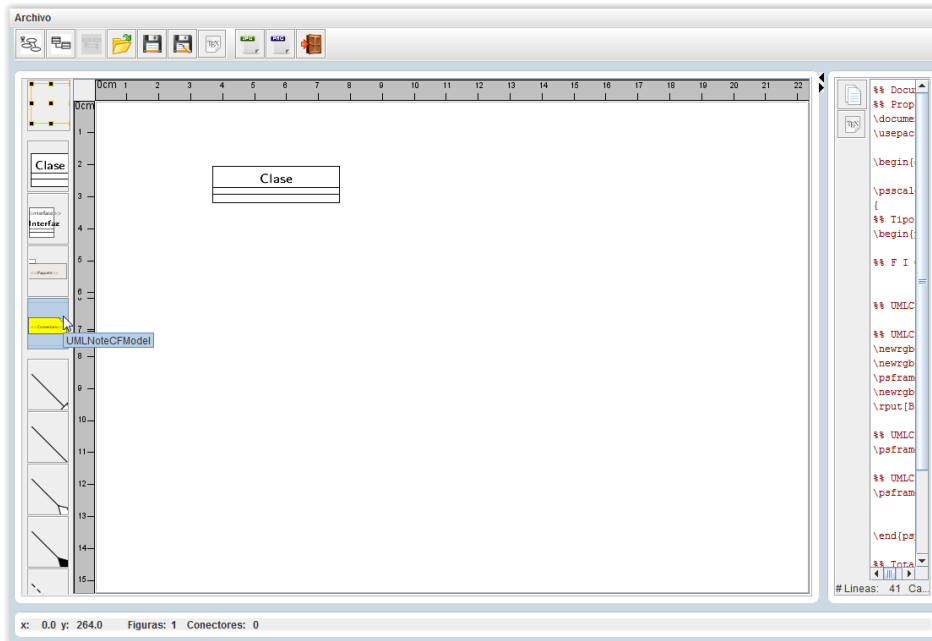


Figura 37: UMLaTeX. El usuario puede elegir de la barra de herramientas el artefacto UML a agregar al diagrama.

2. El usuario se posiciona en el área de edición y oprime el botón izquierdo para colocar el artefacto UML previamente elegido (ver Figura 38).
3. La aplicación agrega el nuevo componente al modelo del diagrama.

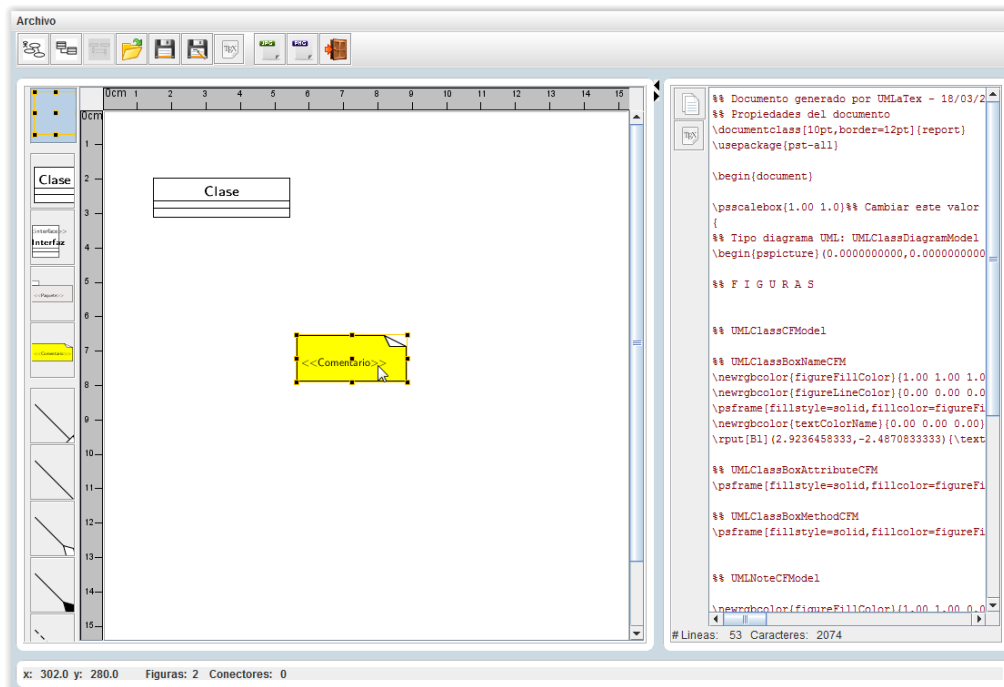


Figura 38: UMLaTeX. La aplicación actualiza el modelo y las vistas al agregar una nueva figura.

## Agregar Conector:

1. El usuario elige de la barra de herramientas un artefacto UML (conector). (Ver Figura 39).

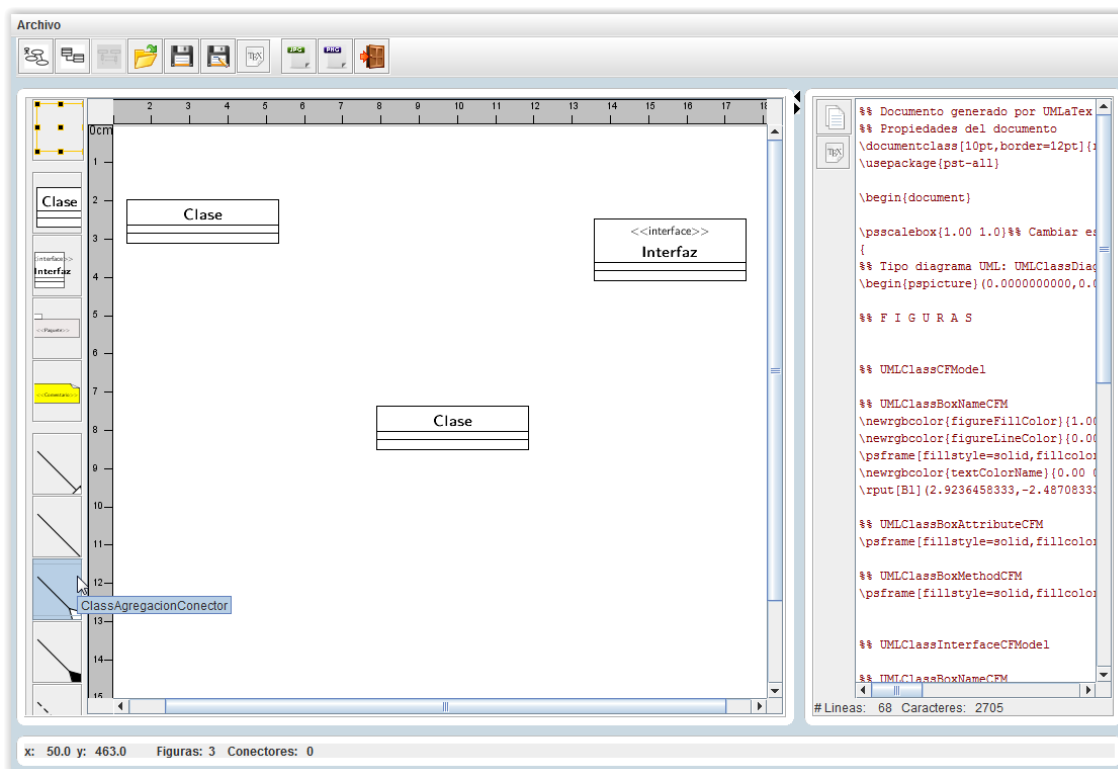


Figura 39: UMLaTeX. El usuario puede elegir de la barra de herramientas el artefacto UML a agregar al diagrama.

2. El usuario se posiciona en el área de edición y oprime el botón izquierdo del ratón para colocar el artefacto UML previamente elegido (Ver Figura 40).
3. La aplicación agrega el nuevo componente al modelo del diagrama.

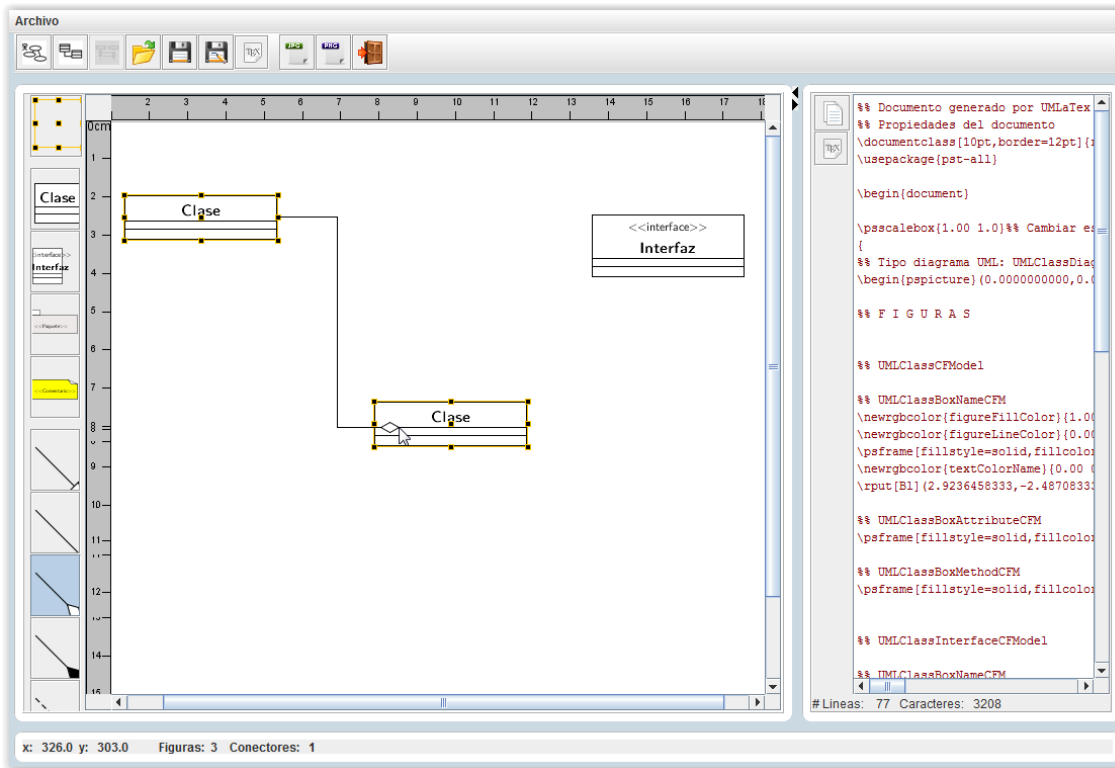


Figura 40:UMLaTeX. La aplicación actualiza el modelo y las vistas al conectar dos figuras.

### 6.2.6.5. Editar propiedades

1. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona (ver Figura 41).

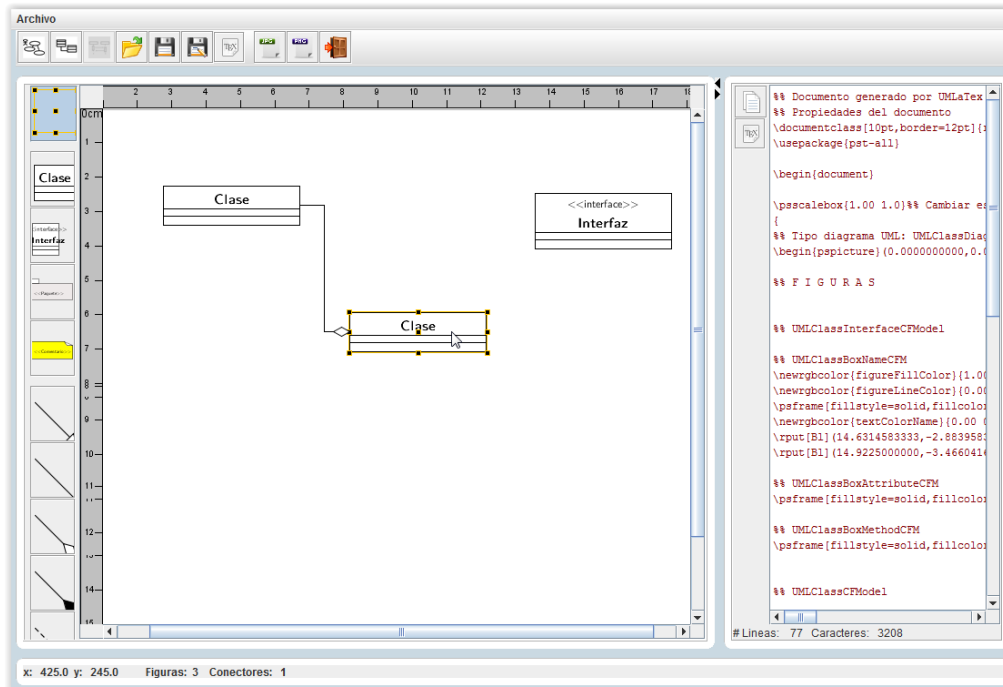


Figura 41. UMLaTeX. Al elegir una figura se resaltarán sus bordes en color naranja.

2. El usuario pide a la aplicación Editar el artefacto UML seleccionado. Esto se logra haciendo dos clics en la figura seleccionada.
3. La aplicación muestra el formulario relacionado con el artefacto UML a editar (ver Figura 42).

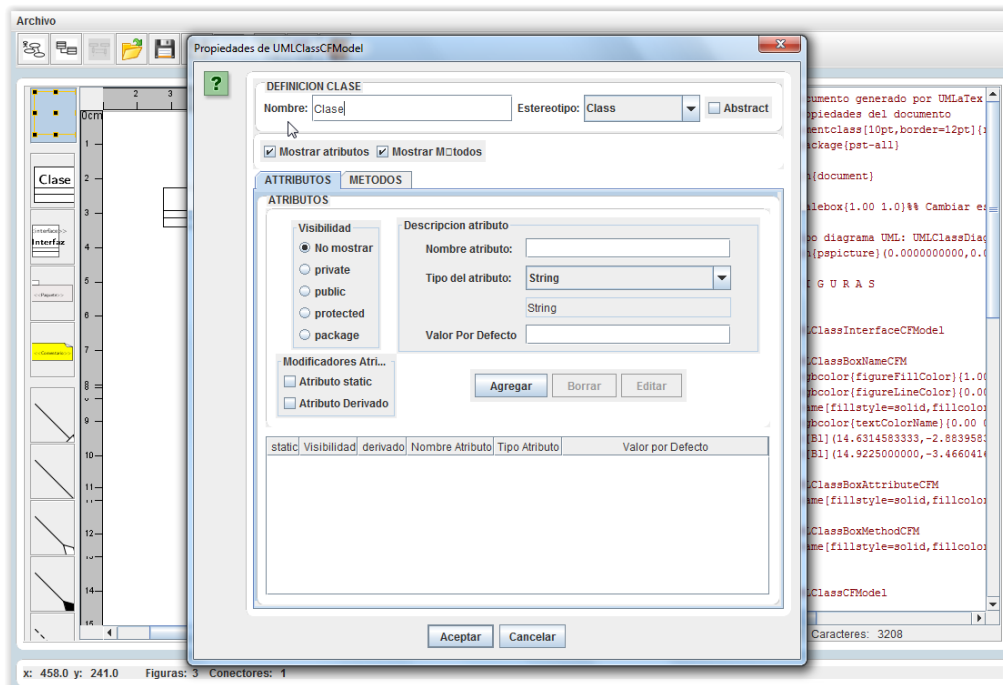




Figura 42:UMLaTeX. Cada artefacto UML (Figura/Conector) tiene un formulario destinado para editar sus propiedades.

4. El usuario edita las propiedades y acepta los cambios (ver Figura 43).

static	Visibilidad	derivado	Nombre Atributo	Tipo Atributo	Valor por Defecto
false	-	false	atributo1	Character	
false	#	false	atributo2	Integer	

Figura 43:UMLaTeX. Formulario de propiedades para el artefacto UML Clase.

5. El diagrama muestra los cambios realizados por el usuario (ver Figura 44).

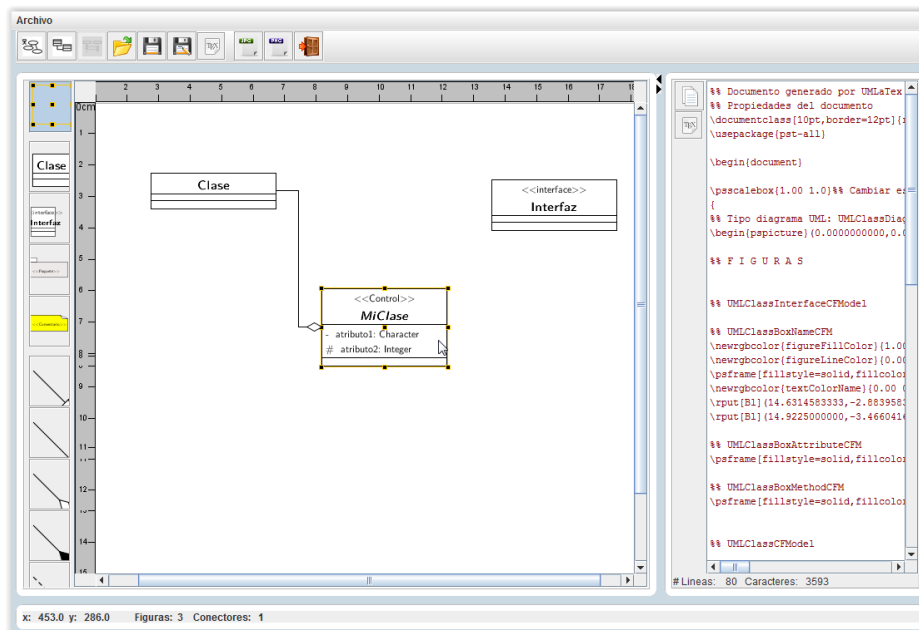


Figura 44:UMLaTeX. Se actualiza el diagrama tras la edición de las propiedades de un artefacto UML.

### 6.2.6.6. Borrar Figura/Conector

1. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona (ver Figura 45).

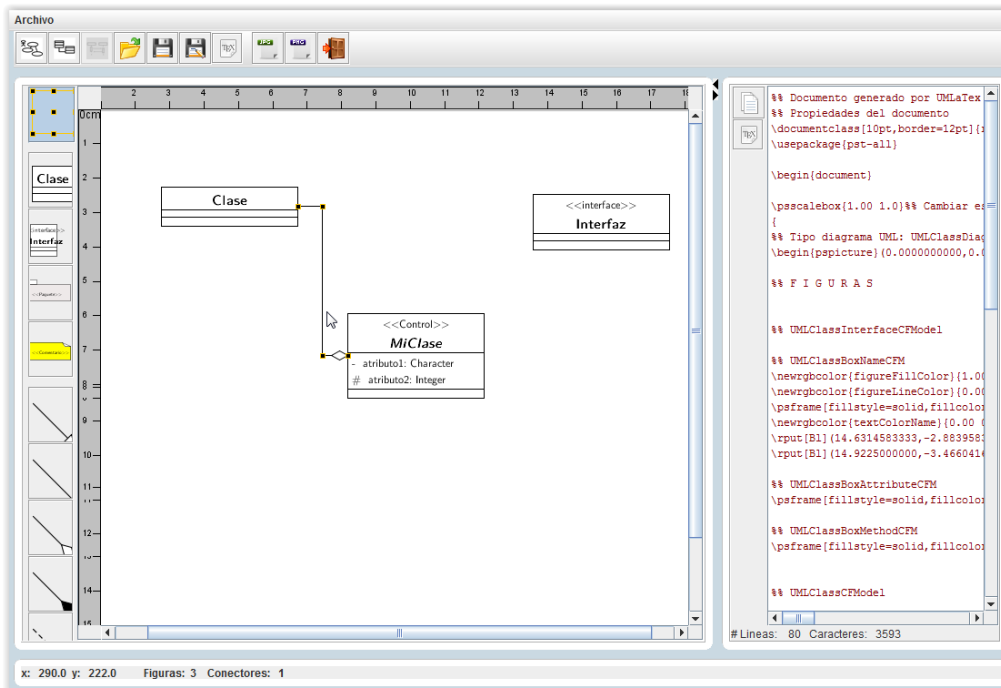


Figura 45: UMLaTeX. La selección de figuras/conectores se realiza con la herramienta de selección.

2. El usuario pide a la aplicación borrar el artefacto UML seleccionado. Esto se logra oprimiendo la tecla **SUPRIMIR**. (Ver Figura 46).

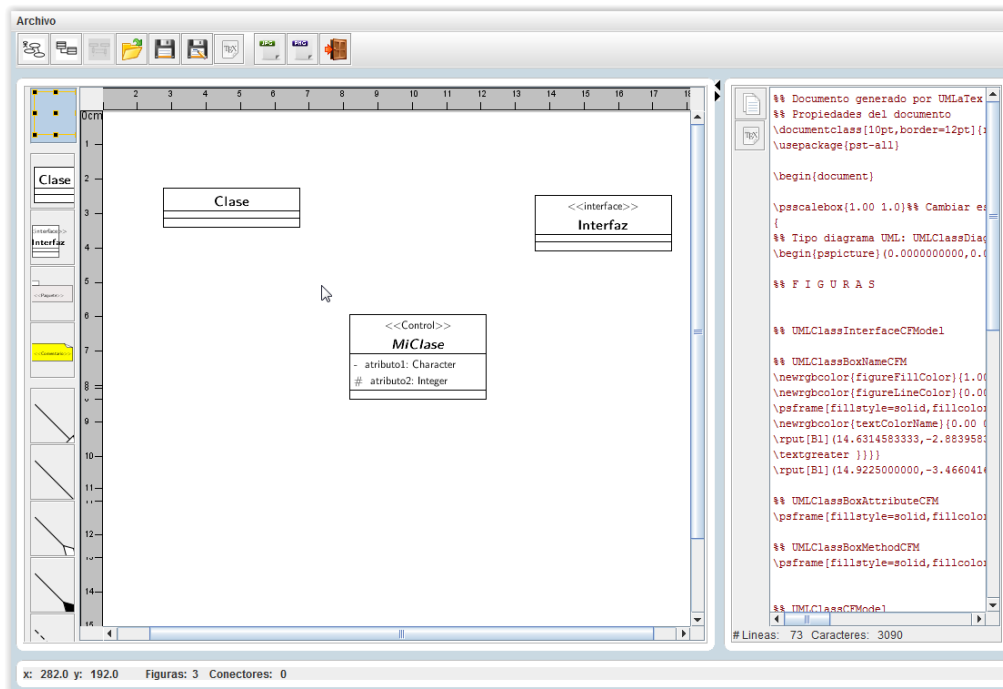




Figura 46: Se pueden eliminar los artefactos UML oprimiendo la tecla SUPRIMIR

### 6.2.6.7. Guardar Diagrama

1. El usuario solicita a la aplicación que se ejecute una operación de salvado. Esto lo puede hacer presionando el botón de la barra de herramientas Guardar  o Guardar Como...  también puede hacerlo seleccionando el menú **Archivo** → **Guardar/Guardar Como...** (Ver Figura 47).

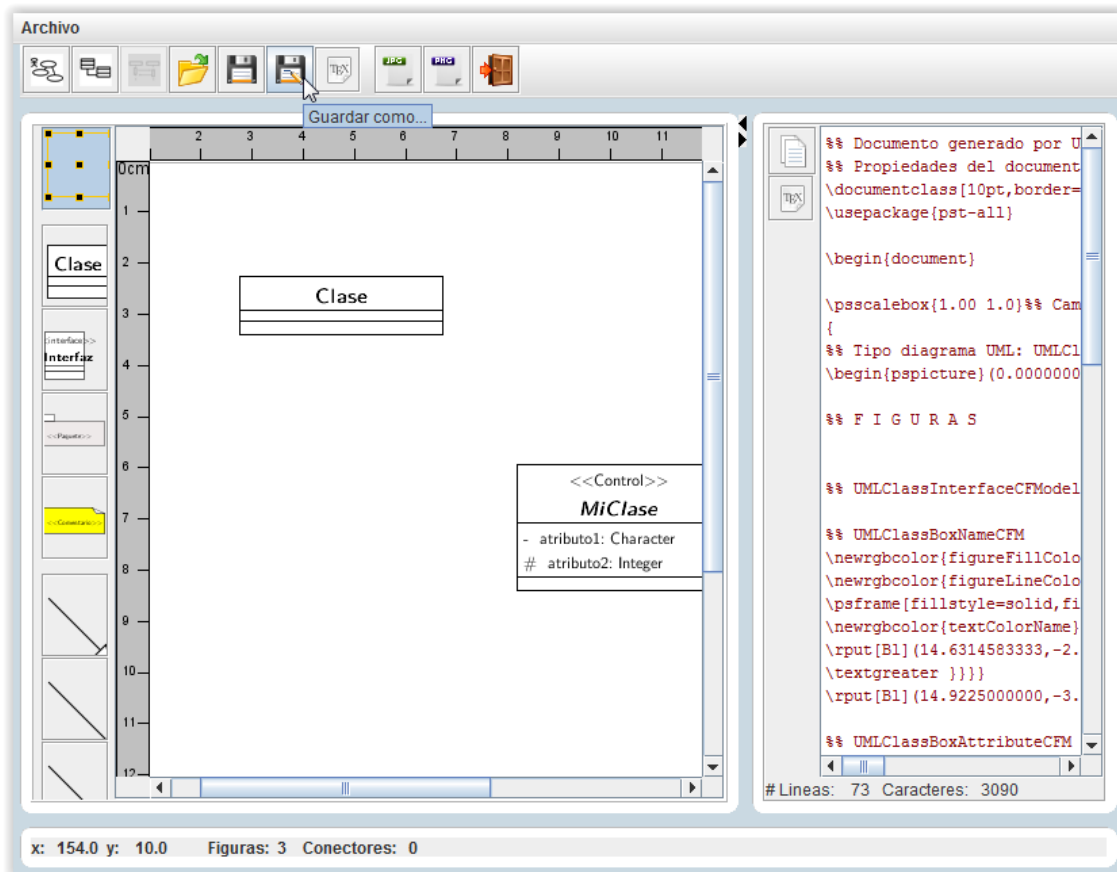


Figura 47: UML4TeX. Las opciones de Guardar/Guardar Como... están disponibles en la barra de herramientas principal de la aplicación o en el Menú.

2. La aplicación muestra un diálogo con un sistema de archivos para elegir el nombre y ubicación del archivo a guardar. El usuario elegirá cómo guardar su archivo (ver Figura 48).

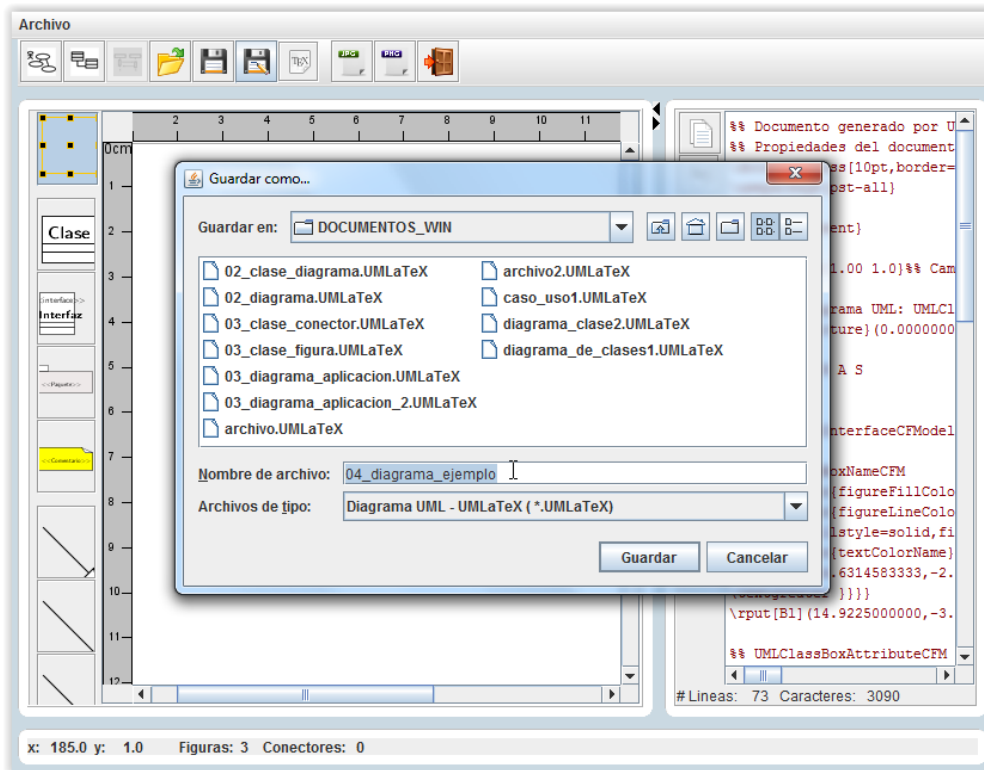




Figura 48: UMLaTeX. Los diagramas generados por la aplicación son guardados en formato XML y con la extensión \*.UMLaTeX

### 6.2.6.8. 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 de la barra de herramientas principal 
  - b. Accediendo al menú **Archivo** → **Exportar a código PST**.
  - c. En el panel de vista de código PST también está habilitada la el botón  para exportar el diagrama (ver Figura 49).

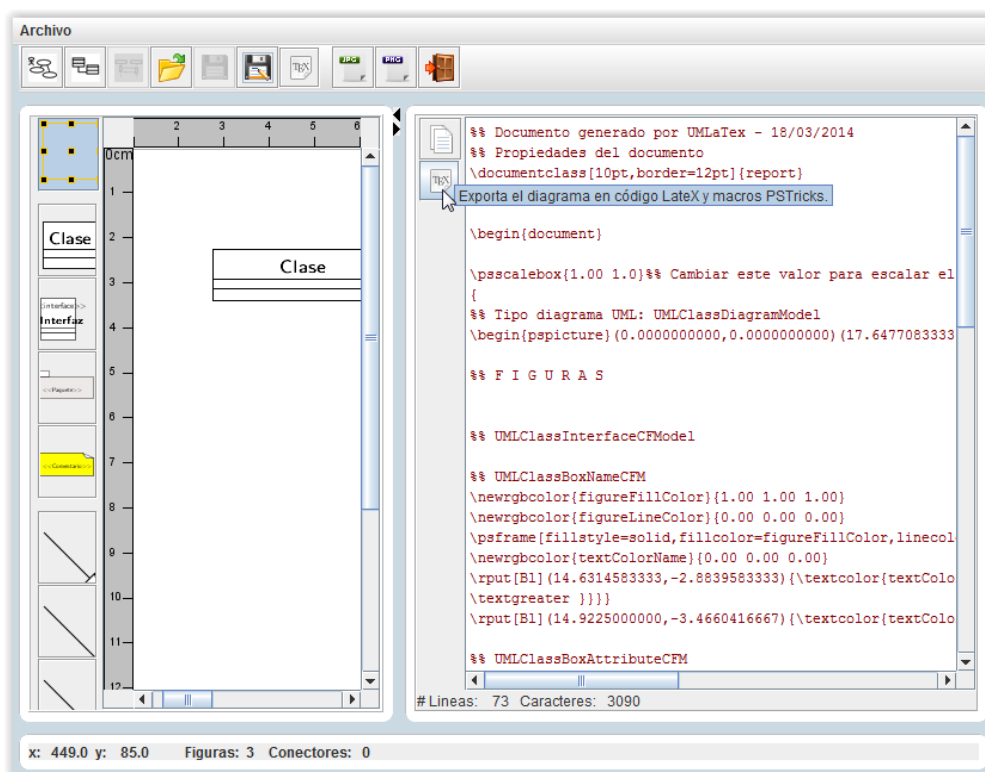


Figura 49: UMLaTeX. La aplicación ofrece varios métodos para exportar los diagramas a macros PSTricks.

2. La aplicación abrirá un diálogo con sistema de archivos donde elegirá la ubicación del archivo \*.tex de salida. (Ver Figura 50).

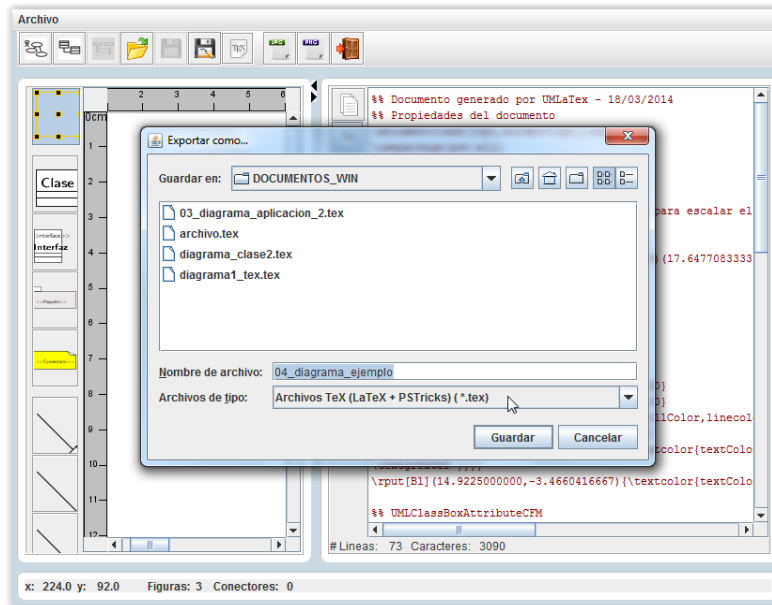


Figura 50: UMLaTeX. Los diagramas pueden ser convertidos a macros PSTricks, que son integrables a documentos LaTeX

3. La aplicación avisará el éxito de la operación (ver Figura 51).

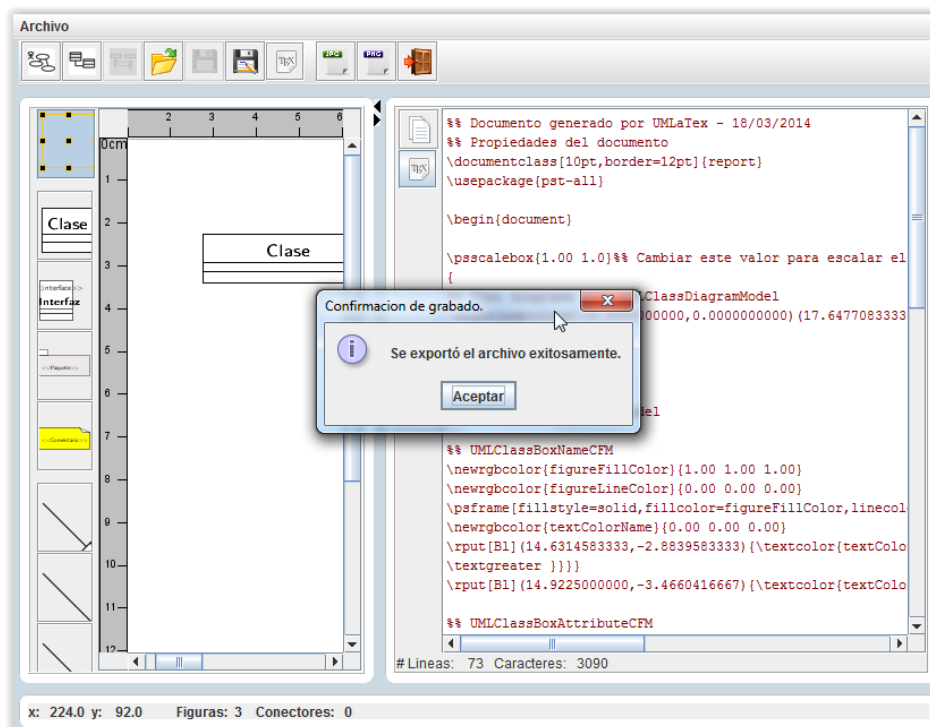




Figura 51: UMLaTeX. La aplicación genera el archivo \*.tex con las macros PSTricks del diagrama actual.

### 6.2.6.9. 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 52). 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**

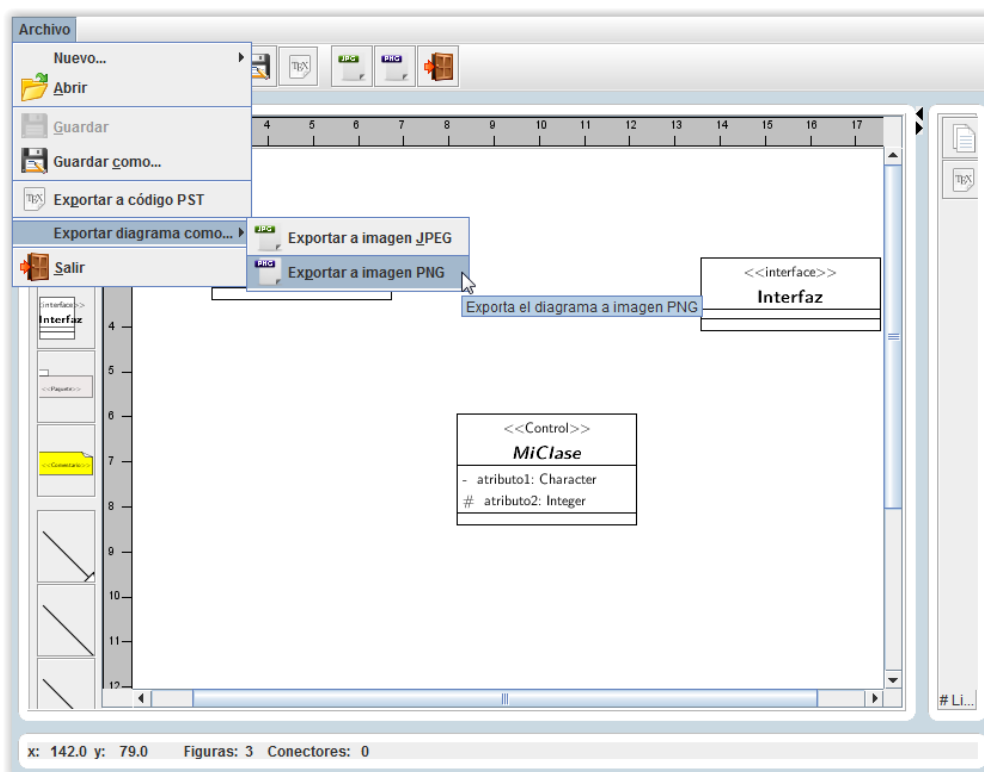


Figura 52: UML4TeX. La aplicación ofrece la posibilidad de exportar el diagrama generado en formato de imagen JPG y PNG

2. La aplicación pregunta al cliente dónde guardar el archivo de imagen de salida (ver Figura 53).



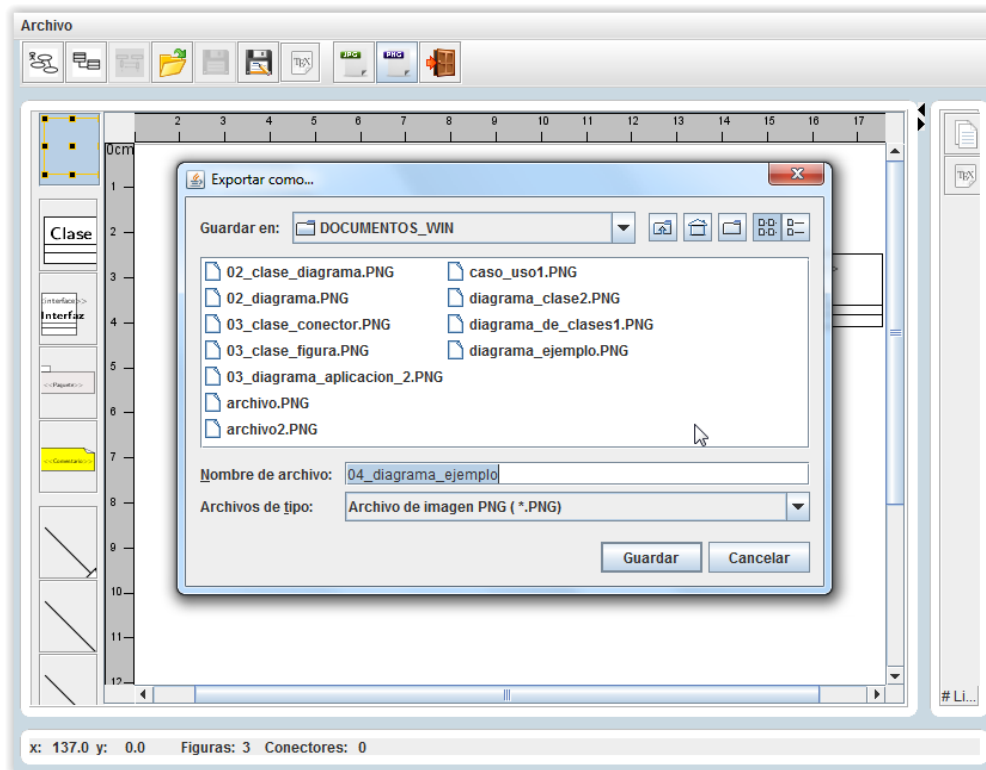


Figura 53: UML4TeX. La aplicación realiza la operación de exportación en el formato indicado (PNG/JPG).

### **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 AMD Athlon™ 64 DX Dual Core a 2.11GHz, con memoria RAM de 2GB.

##### **Software:**

- Entorno de desarrollo integrado (IDE) Eclipse 4.3.1.
- JDK (Java Development Kit) 1.6

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.

#### **6.3.2. Tecnología para la instalación y puesta en marcha de la aplicación**

##### **Hardware:**

Se recomienda utilizar en equipos con 1GB de RAM para un óptimo rendimiento.

##### **Software:**

UML4TeX es una aplicación multiplataforma desarrollada en JAVA, por lo cual para poder ejecutar la aplicación es necesaria que el equipo tenga instalada la máquina virtual de JAVA con una versión de la máquina virtual de Java mayor a 6 (JRE 6).

## 7. Resultados

En la realización de este proyecto se tuvieron los siguientes resultados:

- Se implementó un módulo de representación de la estructura lógica de diagramas UML (Modelo).
- Se implementaron los módulos de representación visual de la estructura de los diagramas UML (Vistas)
- Se implementaron los módulos que gestionan los eventos que hacen que cambien los modelos de los diagramas UML y los asocian a las vistas que presentan en la aplicación (Controladores).
- Se implementó un módulo de la interfaz gráfica de usuario que permite la interacción del usuario final y de los controladores de la aplicación.
- Se construyó un módulo capaz de implementar la persistencia de los diagramas de la aplicación, utilizando como base el lenguaje de marcado XML.
- Se construyó el módulo traductor a macros PSTricks, con el cual los diagramas podrán ser convertidos en macros que se pueden insertar en documentos LaTeX.

La construcción e integración de cada uno de los módulos dio como producto una solución llamada UML4TeX, un editor gráfico que puede crear, editar y guardar los diagramas UML especificados en el alcance del proyecto. Además de tener la capacidad de exportar los diagramas en macros PSTricks, permitiendo que se integren en documentos LaTeX.

## **8. Análisis y discusión de resultados**

En esta sección se hará un recuento de los resultados obtenidos en el desarrollo de la aplicación, esto incluye las pruebas funcionales respecto a los módulos integrados para cada objetivo propuesto en el alcance de este proyecto.

### **8.1. Módulos de estructura lógica y de visualización de los diagramas UML**

Dado que la aplicación se realizó con el esquema Modelo-Vista-Controlador, se implementaron los módulos de estructura lógica de los diagramas UML, así como los distintos módulos de visualización de los diagramas UML.

El conjunto de clases que conforman el módulo del Modelo son las Figuras y Conectores que representan a los distintos artefactos de UML. Todos se unen en la colección que contiene cada diagrama de UML y este representa nuestro modelo.

El modelo tiene varias representaciones en la aplicación:

- El panel de trabajo en la interfaz gráfica de usuario actúa como un renderizador (Vista 1) del modelo de los diagramas.
- El panel de generación de código representa una segunda vista de los diagramas en su representación de macros PSTricks
- El panel de estado es una tercera vista del estado de nuestro modelo, representando el número de figuras y diagramas contenidos en nuestro modelo.

Por cada cambio que exista en el modelo, todas las vistas asociadas a éste se actualizarán. En las imágenes siguientes se muestra a la aplicación integrando los módulos de las vistas en relación al modelo del diagrama (ver Figura 54).

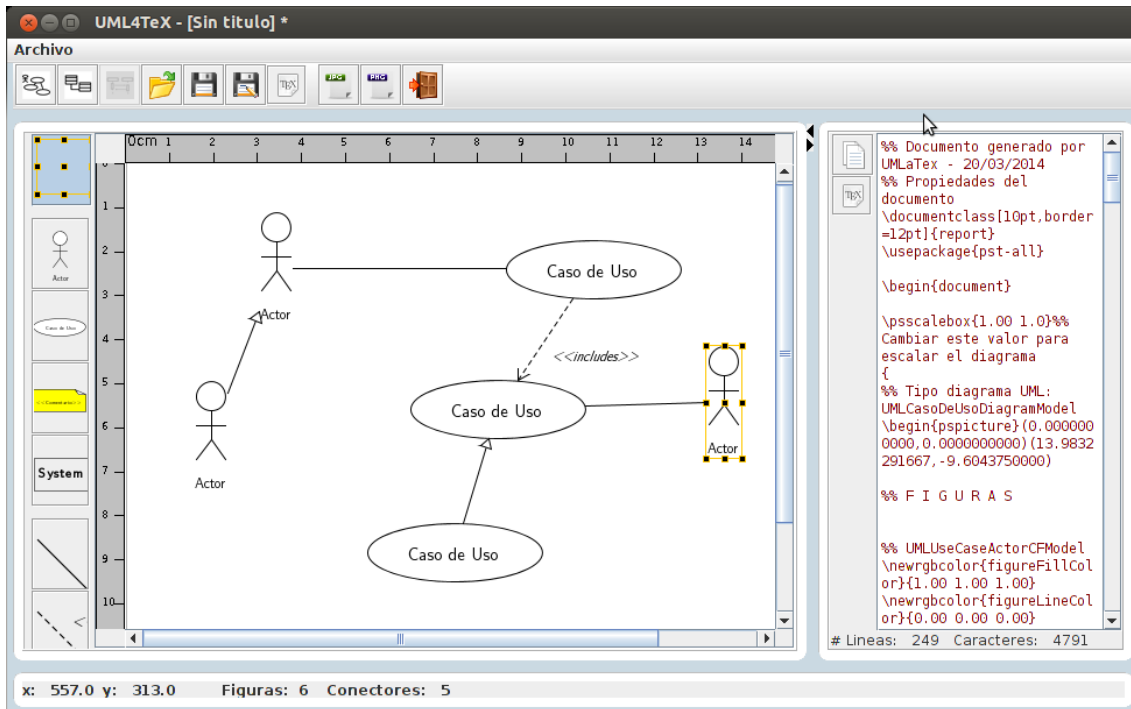


Figura 54: UML4TeX. La aplicación integrando los módulos de estructura lógica (modelo) y de visualización (vistas) de los diagramas UML. La primera vista (panel central) es el renderizador del diagrama, mostrando la representación gráfica de éste. La segunda vista (panel de la derecha) es el diagrama como representación en macros PSTricks. Una tercera vista (panel de abajo) muestra una representación del modelo en función del número de figuras y conectores asociados a él.

## 8.2. Integración de los módulos de edición (manipulación gráfica) de los diagramas UML

Bajo el esquema Modelo-Vista-Controlador, los módulos asociados para la edición de los diagramas UML son manejados por distintos controladores que van interpretando los eventos de entrada (como arrastrar una figura en el panel de dibujo, agregar, borrar o editar un artefacto UML) y van modificando el comportamiento de la estructura lógica del modelo del diagrama, para que a su vez actualice todas las vistas.

Estos controladores están encapsulados en la barra de herramientas de los diagramas UML que asocian el controlador específico de una figura dentro del diagrama con el modelo para decidir cuál acción tomar sobre de este.

### 8.2.1. 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 (ver Figura 55).

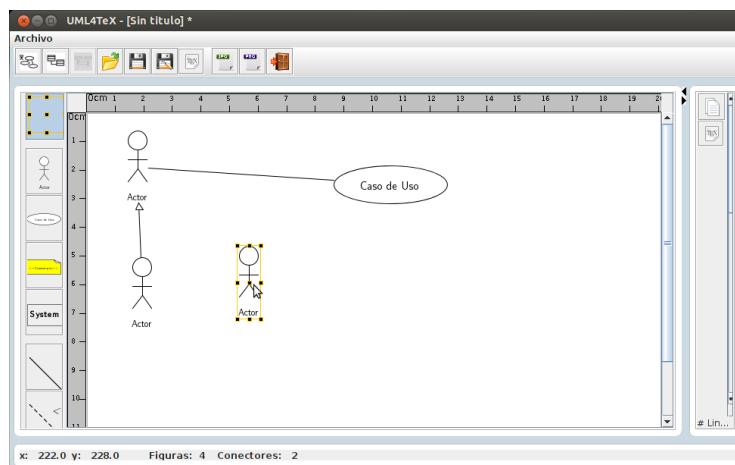


Figura 55. UML4TeX. El controlador de movimiento de figuras actualiza la posición de un artefacto UML dentro del diagrama.

### 8.2.2. 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.

#### 8.2.2.1. 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 panel de edición. Permitiendo así modificar el modelo para agregar una nueva figura o conector (ver Figura 56).

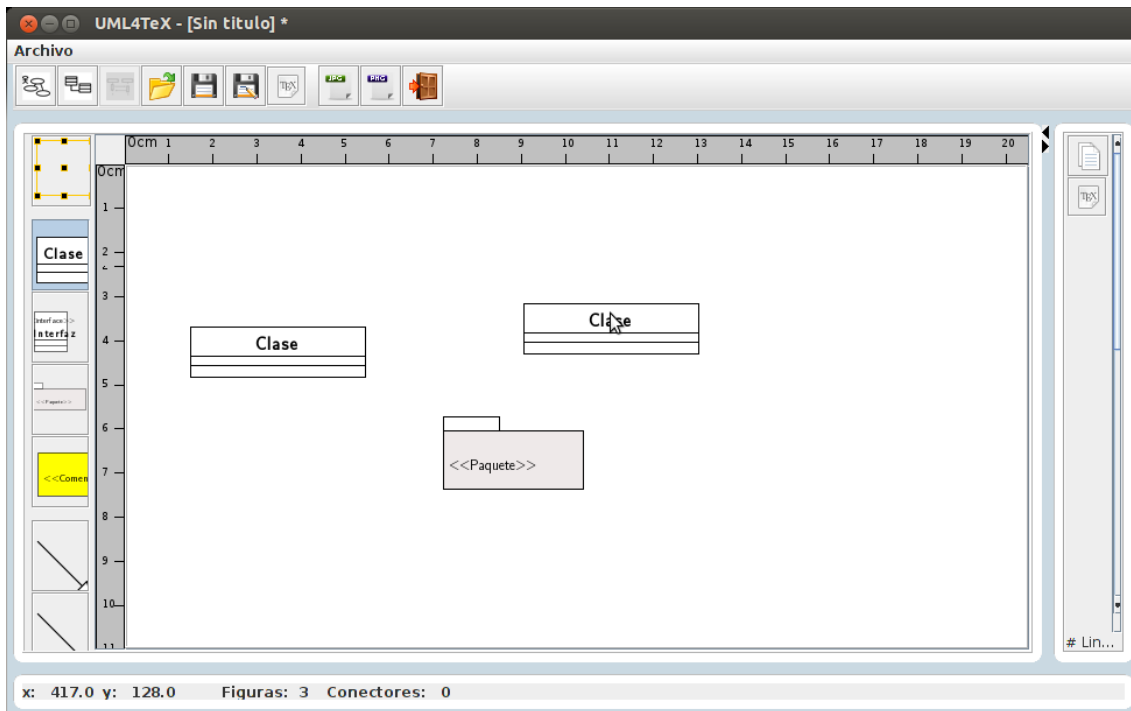


Figura 56. UML4TeX. Selección de figura para agregar, la interfaz recibe los eventos relacionados para obtener el controlador asociado de la figura a insertar.

### 8.2.2.2. Edición de un elemento

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 en cuanto 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 57).

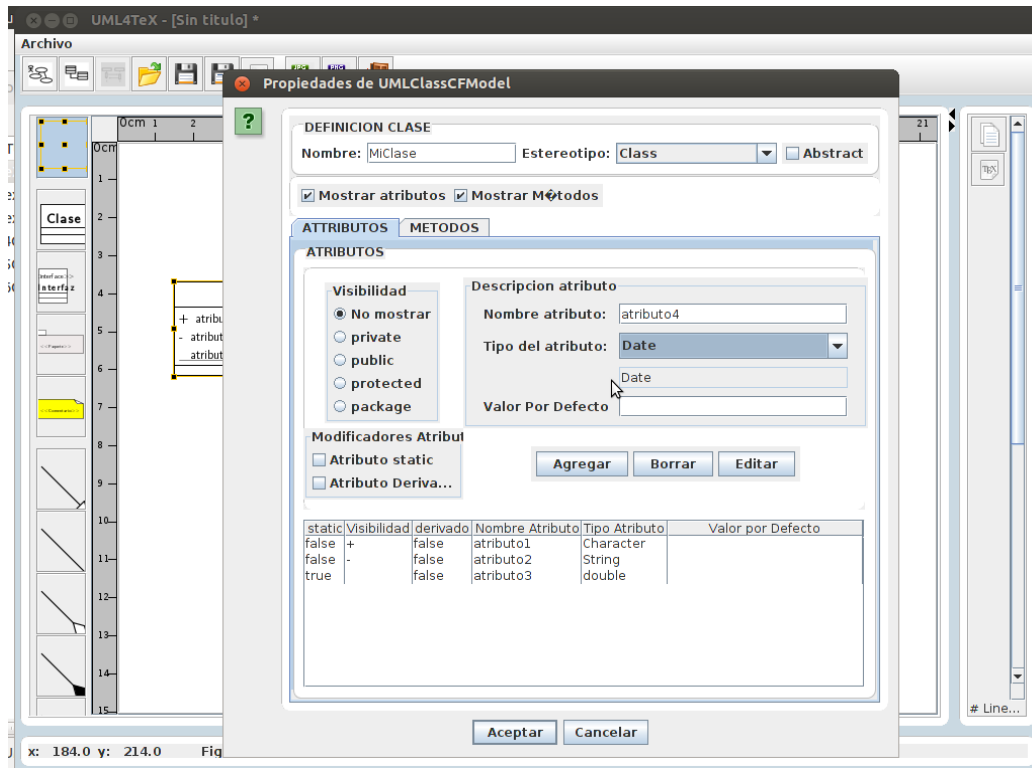


Figura 57: UML4Tex. Edición de las propiedades de un elemento del diagrama.

### 8.2.2.3. Eliminar un elemento

Nuevamente 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 58).



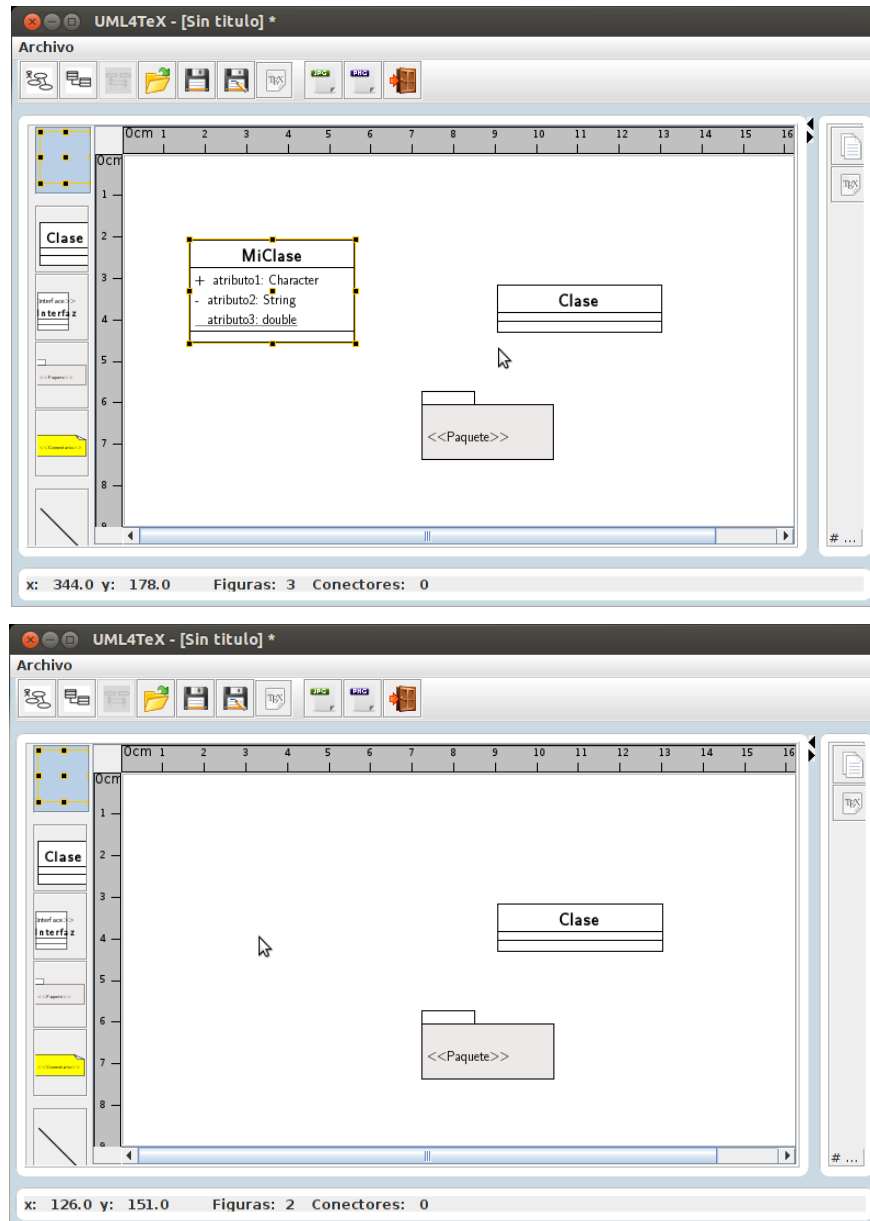


Figura 58: UML4TeX. 1) Se selecciona el elemento Clase del diagrama para eliminar. 2) El controlador detecta que el usuario oprimió la tecla "Suprimir" y lo traduce en la eliminación de la figura del modelo.

### 8.3. Módulo de interfaz gráfica de usuario

Para que la aplicación funcione correctamente y para que el usuario pueda realizar las tareas pertinentes al manejo de la aplicación es necesario una interfaz gráfica. Y esta debe manejar los elementos de usabilidad necesarios para un manejo correcto.

La aplicación se diseñó implementando diferentes paneles de trabajo, de los cuales se destacan barras de herramientas, menús, paneles de edición y paneles para mostrar las distintas vistas (ver Figura 59).

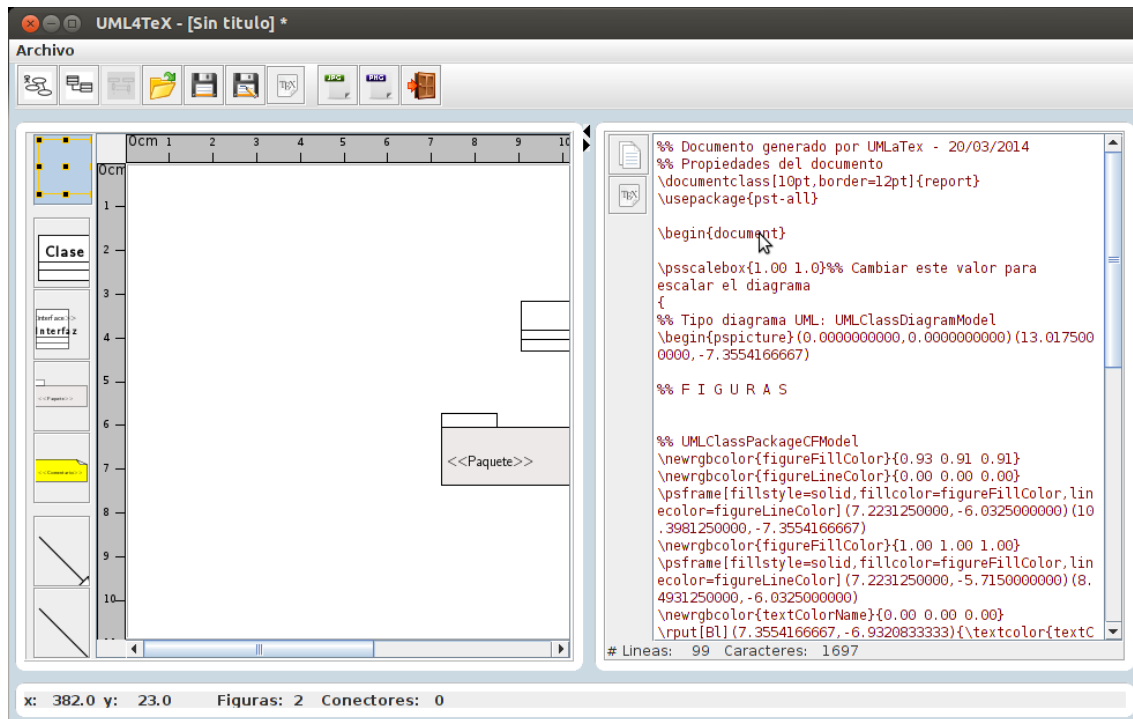


Figura 59: Interfaz gráfica de usuario de UML4TeX. Cuenta con los paneles y menús y barras de herramientas a utilizar.

Cada opción de menú (ver Figura 60), así como cada botón en la barra de herramienta (ver Figura 61) tienen asociado alguna acción y tiene un icono que describe la acción relacionada éste.

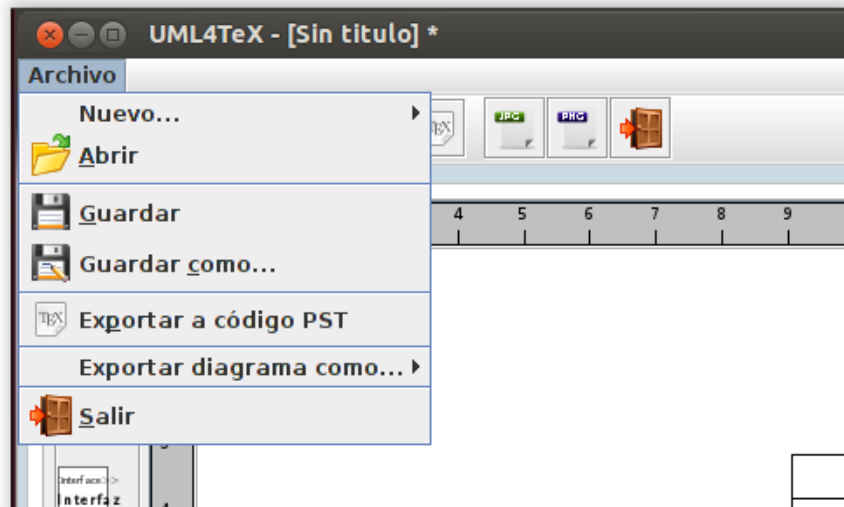


Figura 60: Imagen de uno del Menú Archivo, que utiliza la aplicación.



Figura 61: Barra de herramientas de la aplicación. Cada icono describe la acción que puede realizar.

La interfaz se integra con los controladores y las vistas del proyecto, permitiendo así la interacción del usuario con la aplicación.

#### 8.4. Módulo de persistencia de los diagramas

El resultado de la implementación de este módulo nos permite mantener persistente nuestro diagrama diseñado y poderlo recuperar tiempo después dentro de la aplicación.

Para tener un formato legible se eligió el formato XML. Que describe en sus diferentes etiquetas qué elementos contiene el diagrama que se desea persistir.

El módulo es capaz de escribir y leer un archivo **\*.UMLaTeX** y va codificando o decodificando los distintos elementos UML que los contiene para así mostrarlos en la aplicación.

Las Figura 62 muestra cómo se realiza el proceso de persistencia. Se muestra como un diagrama está preparado para ser guardado.

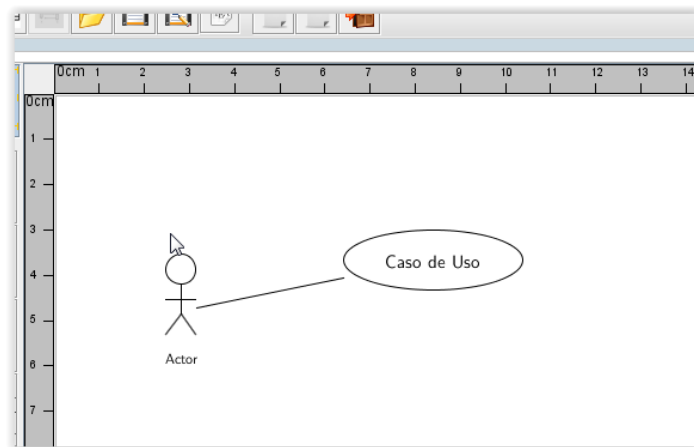


Figura 62. Diagrama de Caso de Uso a ser guardado con el módulo de persistencia.

Al seleccionar de la interfaz de usuario la acción Guardar pregunta al usuario la ubicación donde desear mantener su diagrama persistente (ver Figura 63).

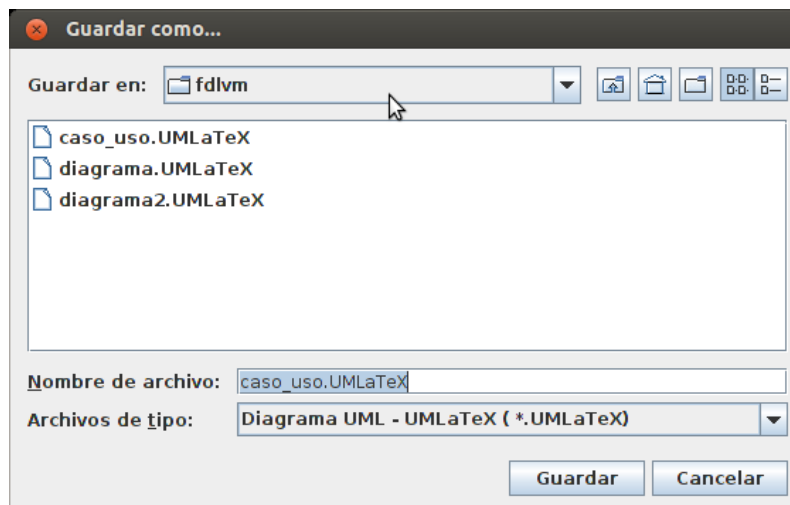
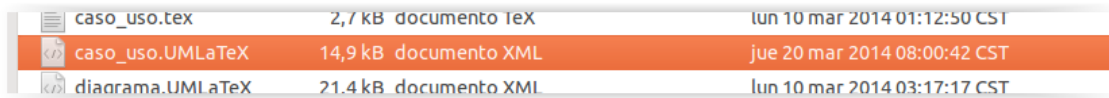


Figura 63. La aplicación pregunta al usuario donde desea guardar el diagrama.

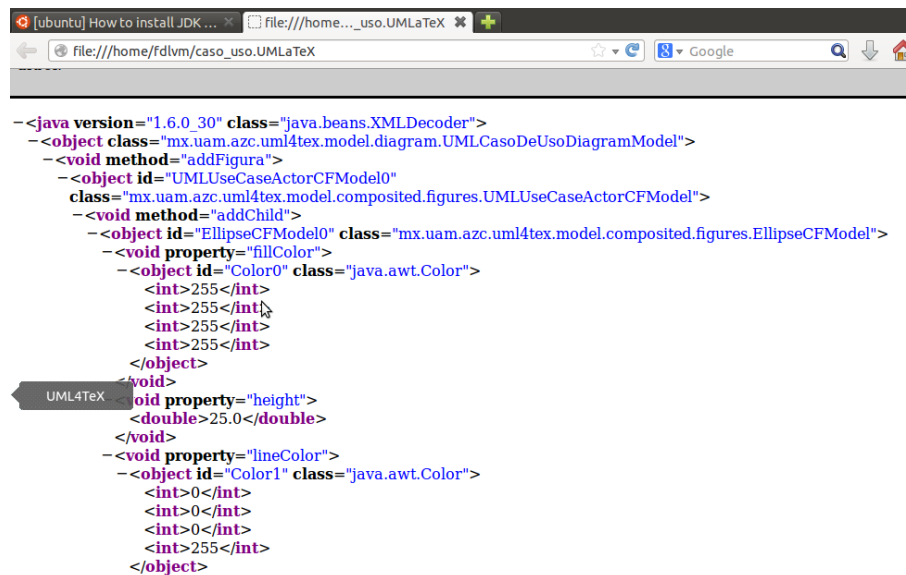
En el sistema de archivos se muestra el archivo generado por el módulo de persistencia. Este es un archivo XML con extensión \*.UMLaTeX (ver Figura 64).



caso_uso.tex	2,7 kB	documento TeX	lun 10 mar 2014 01:12:50 CST
caso_uso.UMLaTeX	14,9 kB	documento XML	jue 20 mar 2014 08:00:42 CST
diagrama.UMLaTeX	21,4 kB	documento XML	lun 10 mar 2014 03:17:17 CST

Figura 64. Archivo \*.UMLaTeX generado por la aplicación

En la Figura 65 se muestra extracto de la información que contiene el archivo XML \*.umllatex generado por la aplicación y que describe los elementos del diagrama.



```
-<java version="1.6.0_30" class="java.beans.XMLDecoder">
-<object class="mx.uam.azc.uml4tex.model.diagram.UMLCasoDeUsoDiagramModel">
-<void method="addFigura">
  -<object id="UMLUseCaseActorCFModel0"
  class="mx.uam.azc.uml4tex.model.composited.figures.UMLUseCaseActorCFModel">
    -<void method="addChild">
      -<object id="EllipseCFModel0" class="mx.uam.azc.uml4tex.model.composited.figures.EllipseCFModel">
        -<void property="fillColor">
          -<object id="Color0" class="java.awt.Color">
            <int>255</int>
            <int>255</int>
            <int>255</int>
            <int>255</int>
          </object>
        </void>
        -<void property="height">
          <double>25.0</double>
        </void>
        -<void property="lineColor">
          -<object id="Color1" class="java.awt.Color">
            <int>0</int>
            <int>0</int>
            <int>0</int>
            <int>255</int>
          </object>
        </void>
      </object>
    </void>
  </object>
</void>
</object>
```

Figura 65: Contenido del archivo XML \*.UMLaTeX

Para la lectura la aplicación decodificará el archivo XML que está en sistema y lo mostrará en la aplicación con el diagrama.

## 8.5. Módulo de traducción de estructura lógica del diagrama UML cómo macros PSTricks

Este módulo tiene como tarea principal traducir el contenido del modelo del diagrama en código LaTeX que incluye macros PSTricks para realizar figuras. Por lo cual se convierte en un módulo que implementa una vista más asociada al modelo del diagrama.

Este módulo aplica las propiedades de cada artefacto UML y los va traduciendo en sus correspondientes macros de acuerdo al formato establecido por PSTricks.

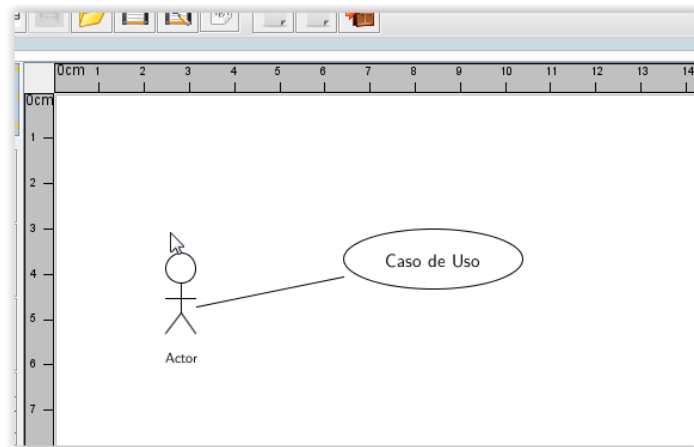


Figura 66. Representación gráfica del Diagrama de Caso de Uso.

Para el diagrama de la Figura 66 se muestra el código con macros PSTricks generado por la aplicación.

```
%% Documento generado por UMLaTex - 20/03/2014
%% Propiedades del documento
\documentclass[10pt,border=12pt]{report}
\usepackage{pst-all}

\begin{document}

\psscalebox{1.00 1.0}%% Cambiar este valor para escalar el diagrama
{
%% Tipo diagrama UML: UMLCasoDeUsoDiagramModel
\begin{pspicture}(0.0000000000,0.0000000000)(12.8852083333,-
5.8208333333)

%% F I G U R A S
```

```

%% UMLUseCaseActorCFModel
\newrgbcolor{figureFillColor}{1.00 1.00 1.00}
\newrgbcolor{figureLineColor}{0.00 0.00 0.00}
\psellipse[fillstyle=solid,fillcolor=figureFillColor,linecolor=figureLi
neColor](4.6831250000,-2.4650347222)(0.3307291667,0.3307291667)
\psline[linecolor=figureLineColor,](4.6831250000,-
2.7957638889)(4.6831250000,-3.4572222222)
\psline[linecolor=figureLineColor,](4.3523958333,-
3.1264930556)(5.0138541667,-3.1264930556)
\psline[linecolor=figureLineColor,](4.6831250000,-
3.4572222222)(4.3523958333,-3.8981944444)
\psline[linecolor=figureLineColor,](4.6831250000,-
3.4572222222)(5.0138541667,-3.8981944444)
\newrgbcolor{textColorName}{0.00 0.00 0.00}
\rput[B1](4.2994791667,-
4.5420138889){\textcolor{textColorName}{\normalsize {\textsf{Actor}}}}

%% UMLUseCaseCFModel
\psellipse[fillstyle=solid,fillcolor=figureFillColor,linecolor=figureLi
neColor](10.9008333333,-5.1593750000)(1.9843750000,0.6614583333)
\rput[B1](9.8425000000,-5.3313541667){\textcolor{textColorName}{\large
{\textsf{Caso de Uso}}}}

%% C O N E C T O R E S

%% RelacionComunicacionCUConector
\psline[linecolor=figureLineColor,](5.0667708333,-
3.5192733135)(10.0541666667,-5.1858333333)
\end{pspicture}%% Fin de Figura

%% Total Figuras: 2
%% Total Conectores: 1
}%% psscalebox

\end{document}

```

*Figura 67. Código LaTeX con macros PSTricks generado por la aplicación.*

El código mostrado en Figura 67 puede ser integrado en un documento LaTeX para integrar el diagrama en forma de código.

La aplicación tiene la posibilidad de exportar el código generado en un archivo \*.tex listo para ser utilizado por cualquier aplicación que lea archivos código fuente de LaTeX.

## **8.6. Integración de los módulos de la aplicación**

De la integración de los distintos módulos de la aplicación nace UML4TeX, el editor gráfico de diagramas UML con capacidad de generar macros PSTricks, la cual cumple con los objetivos propuestos por el proyecto.

La arquitectura de tres capas junto con Modelo-Vista-Controlador permite que se diseñe cada módulo en su particularidad con una función específica, lo cual hace a la aplicación fácil de entender para su mantenimiento y para su extensión.



## **9. Conclusiones**

UML4TeX cumple con los objetivos establecidos en el alcance de este proyecto. Se acopla a una necesidad de generar los diagramas UML y de poder plasmarlos en documentos LaTeX mediante macros PSTricks, de tal forma que el usuario puede ahorrar tiempo y esfuerzo en esta tarea que puede resultar compleja.

El desarrollo de la aplicación se adhiere al concepto de patrón modelo-vista-controlador lo cual permitirá la fácil extensión de los módulos existentes así como la integración de nuevos módulos por parte de otros desarrolladores.

## 10. Perspectivas del proyecto

Este proyecto tiene muchas posibilidades de ampliación. A continuación se enlistan algunas de éstas:

- Integración de más diagramas UML
- Integración de más formatos a los cuales se puedan exportar los diagramas generados.
- Integrar módulos para generación de código fuente a partir de los diagramas a lenguajes de programación.
- Integrar módulos para implementar la construcción de diagramas a partir de ingeniería inversa.

## 11. Referencias bibliográficas

- [1] *Unified Modeling Language™ (UML®)* [En línea]. Disponible: <http://www.omg.org/spec/UML/2.0/>
- [2] L. Lamport, *LATEX: A Document Preparation System*, 2nd Edition, Reading, Massachusetts: AddisonWesley, 1994
- [3] D. E. Knuth, *The TEXbook, Volume A of Computers and Typesetting*, 2nd Edition, Reading, Massachusetts: Addison-Wesley, 1984
- [4] C. Cuéllar y J. García, “Ambiente de programación visual para graficar geometrías simples mediante transformaciones ortogonales con álgebra geométrica y álgebra matricial”, Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2009
- [5] A. Torres, “EDAPIX: Una aplicación gráfica para asistir al proceso de enseñanza-aprendizaje de las estructuras de datos”, Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2009
- [6] J. Bocanegra y Y. Tovar, “Visualización de algoritmos de ordenamiento y búsqueda interna con *TikZ*”, Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2010
- [7] *The TikZ and PGF Packages* [En línea]. Disponible: <http://sourceforge.net/projects/pgf>
- [8] *LaTeXDraw: a graphical drawing editor for LaTeX*. [En línea]. Disponible: <http://latexdraw.sourceforge.net/>
- [9] *Dia: A drawing program* [En línea]. Disponible: <http://projects.gnome.org/dia/>
- [10] *The GTK+ Project*. [En línea]. Disponible: <http://www.gtk.org/>

- [11] *GNOME: The Free Software Desktop Project* [En línea]. Disponible: <http://www.gnome.org/>
- [12] *Umbrello UML Modeller* [En línea]. Disponible: <http://uml.sourceforge.net/>
- [13] *KDE Project* [En línea]. Disponible: <http://www.kde.org/>
- [14] *MetaUML, UML for LaTeX/MetaPost*. [En línea]. Disponible: <http://metauml.sourceforge.net/old/index.html>
- [15] *uml.sty, a LaTeX package for UML* [En línea]. Disponible: <http://heim.ifi.uio.no/~ellefg/uml.sty/>
- [16] *PSTricks website* [En línea]. Disponible: <http://www.tug.org/PSTricks/main.cgi/>
- [17] G. Booch, J. Rumbaugh, I. Jacobson., *El Lenguaje Unificado de Modelado*, Addison Wesley Iberoamericana, 1999.
- [18] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd Edition, Reading, Massachusetts: AddisonWesley, 2003
- [19] E. Gamma et al., *Patrones de diseño. Elementos de software orientado a elementos reutilizable*, 1a ed. Núñez de Balboa, Madrid: PEARSON EDUCACIÓN SA, 2003
- [20] *Graphics2D (Java Platform SE 7) - Oracle Documentation*, [En línea]. Disponible: <http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html>
- [21] *Extensible Markup Language (XML)* [En línea]. Disponible: <http://www.w3.org/XML/>
- [22] R. Pressman, *Ingeniería del Software: un enfoque práctico*, 6a ed. México: McGraw-Hill, 2006

## APÉNDICE A. Entregable: Listado del API del código fuente desarrollado

A continuación se enlistan las clases utilizadas por la aplicación.

Paquete	
mx.uam.azc.uml4tex.app.actions	
Clase	Descripción
AbstractDiagramAction	Clase abstracta para las acciones
AcercaDeAction	Acción Acerca De
CreateDiagramAction	Acción Crear Diagrama
DeleteFigureAction	Acción para borrar una figura
ExitAction	Acción para salir de la aplicación

Paquete	
mx.uam.azc.uml4tex.app.gui	
Clase	Descripción
AppSupportedDiagrams	Indica los diagramas UML soportados por la aplicación
DiagramaWorkPanel	Elemento GUI que muestra el área de trabajo de la aplicación.
HelpFrame	Frame que muestra la ventana de ayuda.
InitialMenuDrawPanel	Panel menú inicial para iniciar la aplicación
MainApplicationFrame	Panel principal de la aplicación
UMLDiagramToolBar	Barra de herramientas de diagrama UML

Paquete	
mx.uam.azc.uml4tex.app.gui.components	
Clase	Descripción
FontLoader	Carga fuentes utilizadas por LaTeX
ReglaComponent	Componente que representa una regla
RoundedBorder	Borde redondeado
RoundedJPanel	Panel con Borde redondeado
SplashScreen	SplashScreen para inicio de la aplicación

Paquete	
mx.uam.azc.uml4tex.app.gui.constants	
Clase	Descripción
ApplicationFrameConstants	Constantes de la interfaz

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.app.main</b>	
<b>Clase</b>	<b>Descripción</b>
UML4TeXMainApp	Clase principal. Lanzador de la aplicación

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.app.persistence</b>	
<b>Clase</b>	<b>Descripción</b>
ExportadorImágenes	Clase para exportar diagramas a formato imagen
ExtensionFileFilter	Filtro de extensión de archivo genérico
FileFilterFactory	Factoría de generación de filtros de extensiones de archivo
LaTeXFileFilter	Filtro de extensión para archivos *.tex
UMLatexDiagramIOReaderWriter	Módulo de persistencia de UML4TeX. Guarda y lee archivos en XML

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.controllers</b>	
<b>Clase</b>	<b>Descripción</b>
IActiveCompositeItemListener	Interfaz para escuchar si una figura/conector esta activa para figuras activas
ICreateControllerListener	Interfaz para escuchar eventos de creación de un controlador de figura/conector
ICurrentPointListener	Interfaz para escuchar eventos de cambio de posición de un punto en el mouse
DragFigureController	Controlador del mouse para el movimiento de los elementos del modelo de un diagrama
KeyModelController	Controlador del teclado para movimiento y eliminación de las figuras del modelo de un diagrama

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.controllers.conector</b>	
<b>Clase</b>	<b>Descripción</b>
AbstractConnectorController	Controlador abstracto para conectores
ACompositedConectorController	Controlador abstracto para conectores compuestos
ClassAssociationConectorController	Controlador para el conector de Asociación de Clases
ClassInterfaceRealizationConectorController	Controlador para el conector de Realización de interfaz
ClassUsageDependencyConectorController	Controlador para el conector de Dependencia de clases
RelacionComunicacionCUConectorController	Controlador para el conector de Comunicación de casos de Uso

RelacionExtensionCUConectorController	Controlador para el conector de Inclusión/Extensión de casos de uso
RelacionGeneralizacionCUConectorController	Controlador para el conector de Generalización de clases y de Caso de uso
UMLNoteConectorController	Controlador para el conector de Notas

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.controllers.conector.gui</b>	
<b>Clase</b>	<b>Descripción</b>
ClassAssociationConectorInputPanel	Panel para la edición de propiedades del conector de Asociación de clases
ClassUsageDependencyConnectorInputPanel	Panel para la edición de propiedades del conector de Dependencia de clases
DefaultConectorInputPanel	Panel default para la edición de propiedades de conectores.
RelacionComunicacionCUConectorInputPanel	Panel para la edicion de propiedades del conector de Relación de caso de uso
RelacionExtensionCUConectorInputPanel	Panel para la edicion de propiedades para el conector de Extensión/Inclusión de casos de uso
RelacionGeneralizacionCUConectorInputPanel	Panel para la edicion de propiedades para el conector de Generalización

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.controllers.factories</b>	
<b>Clase</b>	<b>Descripción</b>
AFigureAndConectorControllerFactory	Factoría abstracta de factorías de controladores
ClassControllerFactory	Factoría de controladores para modelo de diagrama de clase
SequenceControllerFactory	Factoría de controladores para modelo de diagrama de secuencia
UseCaseControllerFactory	Factoría de controladores para modelo de Caso de uso

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.controllers.figures</b>	
<b>Clase</b>	<b>Descripción</b>
ACompositedFigureController	Controlador general de figura compuesta
ARectangleBoundedICFController	Controlador general de figuras rectangulares
UMLActorCFController	Controlador para figura compuesta Actor
UMLClassCFModelController	Controlador para figura compuesta Clase
UMLClassInterfaceCFModelController	Controlador para figura compuesta Interfaz
UMLClassPackageCFModelController	Controlador para figura compuesta Paquete

UMLNoteCFModelController	Controlador para figura compuesta Nota
UMLUseCaseCFController	Controlador para figura compuesta Caso de uso
UMLUseCaseSystemBoundaryCFController	Controlador para figura compuesta Limite sistema

Paquete	
<b>mx.uam.azc.uml4tex.controllers.figures.gui</b>	
DefaultInputPanel	Panel de edicion de propiedades default
UMLActorInputPanel	Panel de edicion de propiedades para Actor
UMLClassAttributesInputPanel	Panel de edicion de propiedades para Atributos de clase
UMLClassHeaderInputPanel	Panel de edicion de propiedades para Cabecera de clase
UMLClassInputPanel	Panel de edicion de propiedades para Clase
UMLClassInterfaceCFModelInputPanel	Panel de edicion de propiedades para Interfaz
UMLClassMethodsInputPanel	Panel de edicion de propiedades para Métodos de clase
UMLClassPackageCFModelInputPanel	Panel de edicion de propiedades para Paquete
UMLNoteCFModelInputPanel	Panel de edicion de propiedades para Nota
UMLUCSystemBoundaryInputPanel	Panel de edicion de propiedades para Limites de sistema
UMLUseCaseInputPanel	Panel de edicion de propiedades para Caso de uso

Paquete	
<b>mx.uam.azc.uml4tex.model.composited.conectores.abs</b>	
Clase	Descripción
AbstractCConectorModel	Conector abstracto
ClassAgregacionConector	Conector de Relación de agregación
ClassAsociacionConector	Conector de Relación de Asociación
ClassComposicionConector	Conector de Relación de Composición
ClassGeneralizacionConector	Conector de Relación de Generalización
ClassInterfaceRealizationConector	Conector de Relación de Realización de interfaz
ClassUsageDependencyConnector	Conector de Relación de Dependencia de clases
RelacionComunicacionCUConector	Conector de Relación de Relación de comunicación
RelacionExtensionCUConector	Conector de Relación de Relación de extensión
RelacionGeneralizacionCUConector	Conector de Relación de Relación de generalización
RelacionInclusionCUConector	Conector de Relación de Relación de inclusión
SimpleConector	Conector simple
UMLNoteConector	Conector de Nota



<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.model.composited.figures</b>	
<b>Clase</b>	<b>Descripción</b>
BoundedBox	Dibuja una caja para resaltar figura rectangular
BoundedLine	Dibuja una caja para resaltar una línea
BoundedShape	Dibuja una caja para resaltar una figura poligonal
Doblez	Figura Doblez (Nota UML)
EllipseCFModel	Figura Elipse
FoldCFM	Figura Folder (Nota UML)
LineaCFModel	Figura Línea simple
LineaSegmentadaCFModel	Figura Línea Segmentada
RectangleCFModel	Figura Rectángulo
TextLabelSFModel	Figura Texto
UMLClassBoxAttributeCFM	Figura que representa los atributos de una clase
UMLClassBoxMethodCFM	Figura que representa los métodos de una clase
UMLClassBoxNameCFM	Figura que representa el header de una clase
UMLClassCFModel	Figura Clase
UMLClassInterfaceCFModel	Figura Interfaz
UMLClassPackageCFModel	Figura Paquete
UMLNoteCFModel	Figura Nota
UMLUseCaseActorCFModel	Figura Actor
UMLUseCaseCFModel	Figura Caso de Uso
UMLUseCaseSystemBoundaryCFModel	Figura Limites del sistema

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.model.composited.figures.abs</b>	
<b>Clase</b>	<b>Descripción</b>
AbstractCFModel	Define Figura abstracta
APolygonCFModel	Define Figura poligonal abstracta
ARectangleBoundedCFModel	Define Figura Rectangular

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.model.composited.figures.arrowheads</b>	
<b>Clase</b>	<b>Descripción</b>
ArrowHeadCFModel	Modelo de Flecha
ArrowHeadFactory	Factoría de figuras "ArrowHead"
DiamondArrowHead	Flecha con forma de diamante
TriangleArrowHead	Flecha con forma triangular

Paquete	
<b>mx.uam.azc.uml4tex.model.composited.figures.interfaces</b>	
Clase	Descripción
ICompositedConector	Interfaz IConectorModel
ICompositedFigureModel	Interfaz IFigureModel
IDrawable	Interfaz para dibujar
IObservable	Interfaz para registrar observadores
IPolygonCFModel	Interfaz para Figura Poligonal
IPSTEncodable	Interfaz para traducir a código PSTricks
ITrasladar	Interfaz para trasladar una figura.

Paquete	
<b>mx.uam.azc.uml4tex.model.composited.figures.props</b>	
Clase	Descripción
BentStyle	Define el tipo de forma de una línea segmentada
Direction	Dirección de una línea
TipoDeLinea	Tipo de trazado de una línea

Paquete	
<b>mx.uam.azc.uml4tex.model.diagram</b>	
Clase	Descripción
IDiagramModel	Interfaz que define modelo de un diagrama
IUMLDiagramModel	Interfaz que define el modelo de un diagrama UML
AbstractDiagramaModel	Modelo de Diagrama abstracto
AbstractUMLDiagramModel	Modelo de Diagrama abstracto UML
UMLCasoDeUsoDiagramModel	Modelo de Diagrama de Caso de Uso
UMLClassDiagramModel	Modelo de Diagrama de Clases
UMLSequenceDiagramModel	Modelo de Diagrama de Secuencia

Paquete	
<b>mx.uam.azc.uml4tex.uml.utils</b>	
Clase	Descripción
TableColumnAdjuster	Ajustador de columna de tablas.
UMLClassArgumento	Define un modelo para argumentos de clase
UMLClassAttributeModel	Modelo de Atributos de clase
UMLClassAttributeModelBeanInfo	BeanInfo del Modelo de Atributos de clase

UMLClassHeaderModel	Modelo para Header de Clase
UMLClassMethodModel	Modelo para Métodos de Clase
UMLClassMethodModelBeanInfo	BeanInfo del Modelo de Métodos de clase
UMLClassTableModel	Modelo de Tabla de Modelo de Clase

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.view</b>	
<b>Clase</b>	<b>Descripción</b>
AbstractDiagramaView	Vista genérica.
DiagramaPView	Vista de diagrama UML gráfico
PSTCodePanelView	Vista de diagrama UML en código PSTricks
StatePanelView	Vista de diagrama UML en información simple

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.view.pstcode</b>	
<b>Clase</b>	<b>Descripción</b>
AbstractPSTView	Vista código PSTricks genérica
DefaultPSTView	Vista código PSTricks default
EllipsePSTView	Vista código PSTricks para Elipse
LineaPSTView	Vista código PSTricks para Línea Simple
LineaSegmentadaPSTView	Vista código PSTricks para Línea Segmentada
PolygonPSTView	Vista código PSTricks para Figura Poligonal
RectanglePSTView	Vista código PSTricks para Figura Rectángulo
TextLabelPSTView	Vista código PSTricks para Texto

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.view.pstcode.factory</b>	
<b>Clase</b>	<b>Descripción</b>
PSTCodeFactory	Factoría de vistas de código PSTricks
PSTCodeGenerator	Módulo traductor de código PSTricks

<b>Paquete</b>	
<b>mx.uam.azc.uml4tex.view.pstencoder.utils</b>	
<b>Clase</b>	<b>Descripción</b>
LatexDocumentConstants	Constantes LaTeX
LatexDocumentFormatter	Formateador de documento LaTeX
PSLaTeXSpecialSymbols	Símbolos especiales LaTeX
PSTricksEncoderUtils	Utilidades para codificar macros PSTricks

LatexFontSizeConstants	Constantes LatTeX de fuentes
PSTricksFillStyle	Tipo de relleno (PSTricks)
PSTricksFontOrigin	Tipo de origen de texto (PsTricks)
PSTricksFontTypes	Tipo de Fuente (LaTeX)
PSTricksLineStyle	Tipo de línea (LaTeX)

## 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
- b) Figura 68).

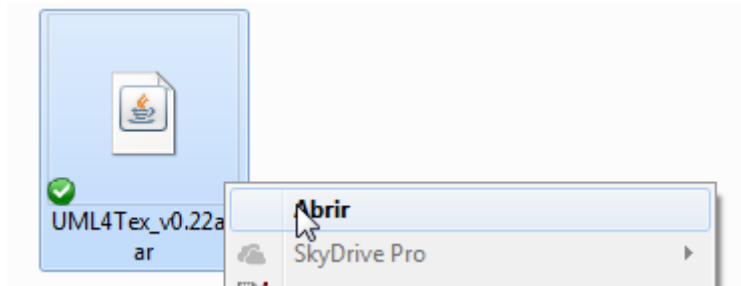


Figura 68. Se puede ejecutar la aplicación con el menú abrir.

#### 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 69).

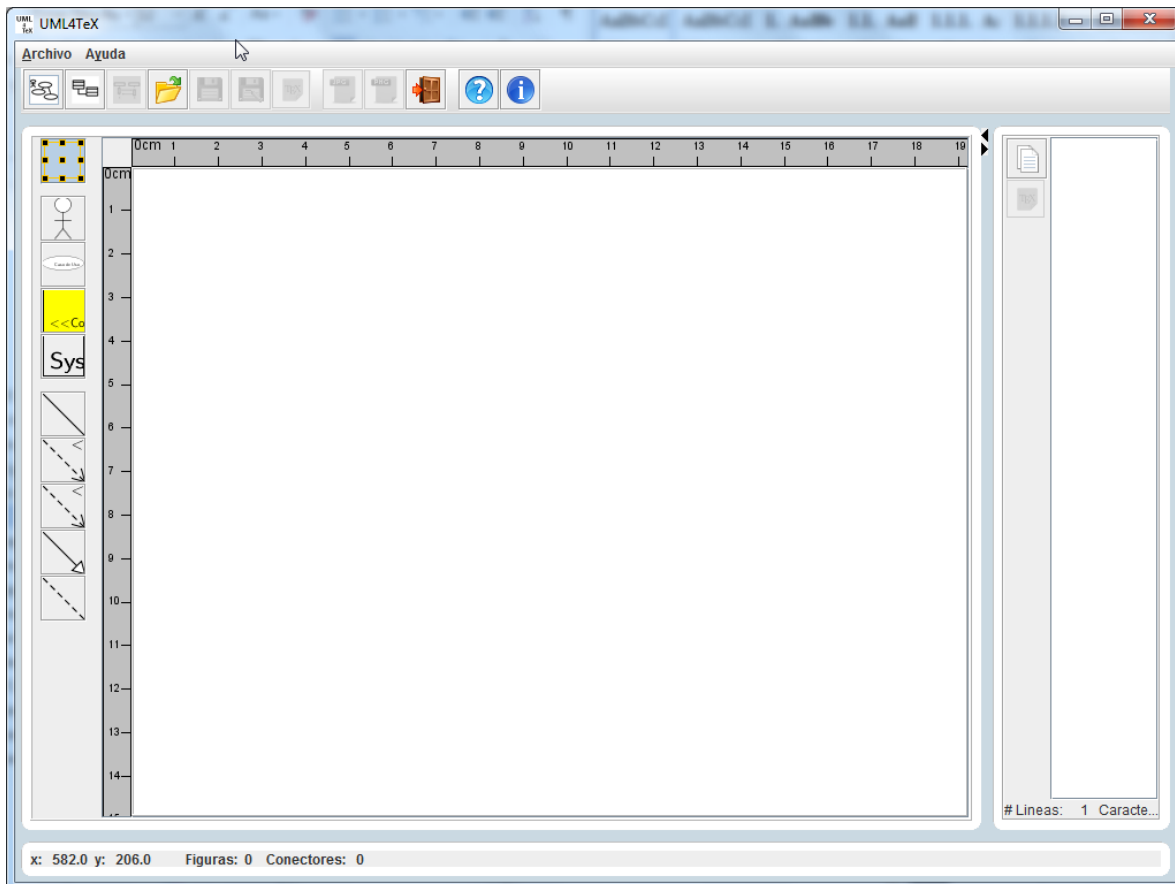


Figura 69. Interfaz gráfica de UML4TeX

#### 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 70).









Figura 70. 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.

Ofrece las siguientes opciones:

Icono	Nombre	Descripción
	Nuevo	Crea un nuevo diagrama UML
	(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 Diagrama de Secuencia
	Abrir	Abre un diagrama UML
	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 cómo...	Exporta el diagrama a una imagen.
	Exportar a imagen PNG	Exporta el diagrama como una imagen formato PNG

	Exportar a imagen JPG	Exporta el diagrama como una imagen JPG
	Salir	Sale de la aplicación

### 3.1.1.3. Menú Ayuda.

Icono	Nombre	Descripción
	Mostrar ayuda	Muestra la ayuda de la aplicación
	Acerca de...	Muestra la información de la aplicación






### 3.1.2. Barra de herramientas

La barra de herramientas, ubicada debajo de la barra de menús, está organizada con botones que corresponden a los comandos principales de la aplicación (ver Figura 71).










Figura 71. Barra de herramientas de la interfaz gráfica.

La siguiente tabla describe las operaciones realizadas con la barra de herramientas.

Icono	Nombre	Descripción
	(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 Diagrama de Secuencia
	Abrir	Abre un diagrama UML
	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 a imagen PNG	Exporta el diagrama como una imagen formato PNG
	Exportar a imagen JPG	Exporta el diagrama como una imagen JPG
	Salir	Salida de la aplicación
	Mostrar ayuda	Muestra la ayuda de la aplicación
	Acerca de...	Muestra la información de la aplicación

### 3.1.3. Panel de dibujo.

En esta área de la aplicación el usuario puede dibujar los diagramas UML (ver Figura 72). 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.

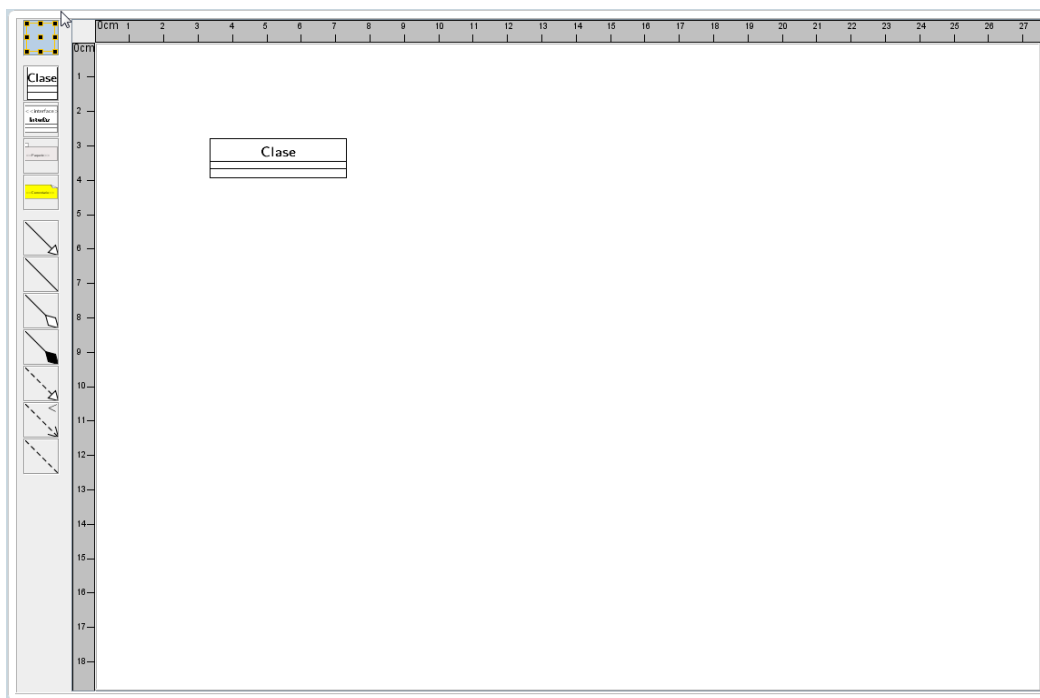







Figura 72. Panel de trabajo de diagrama UML






#### 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		
Icono	Nombre	Descripción
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.






	Dibujar figura 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





### CONECTORES





<b>Icono</b>	<b>Nombre</b>	<b>Descripción</b>
	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 una 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		
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



CONECTORES		
Icono	Nombre	Descripción
	Dibuja una 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 Generalización	Dibuja un conector que representa una relación de generalización de clases (herencia)
	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.

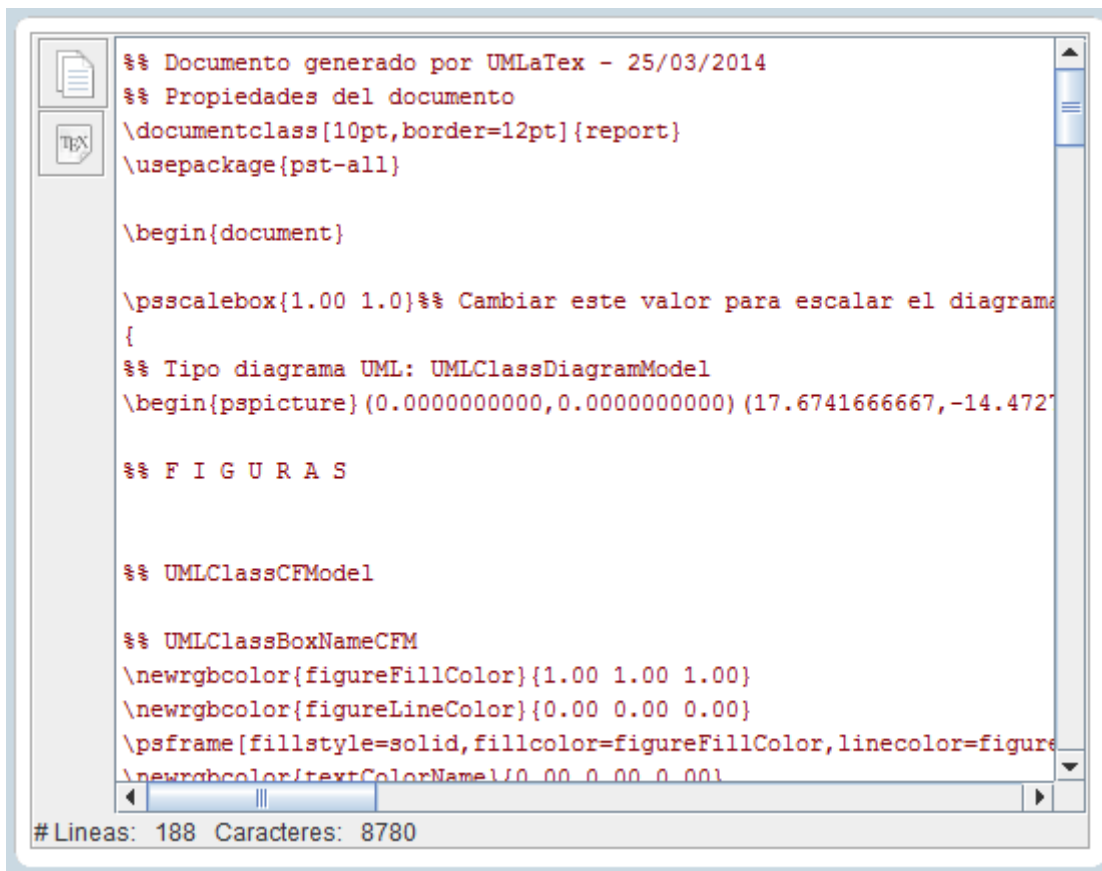
<b>FIGURAS</b>		
<b>Icono</b>	<b>Nombre</b>	<b>Descripción</b>
	Herramienta. Seleccionar.	Selecciona una figura o conector del diagrama.
	Dibujar Objeto	Dibuja una figura representando el Artefacto UML Objeto
	Dibujar Nota	Dibuja una figura representando una Nota

<b>CONECTORES</b>		
<b>Icono</b>	<b>Nombre</b>	<b>Descripción</b>
	Dibujar un mensaje petición	Dibuja un conector que representa un mensaje de petición.
	Dibujar un mensaje de respuesta	Dibuja un conector que representa un mensaje de respuesta.

	Dibujar conector de Nota	Dibuja un conector de figuras de Nota
---	--------------------------	---------------------------------------

### 3.1.4. Panel de generación de código PSTricks.

El panel de generación mostrará las macros PSTricks que se generan con el diagrama UML dibujado (Ver Figura 73).



```

%% Documento generado por UMLaTex - 25/03/2014
%% Propiedades del documento
\documentclass[10pt,border=12pt]{report}
\usepackage{pst-all}

\begin{document}

\psscalebox{1.00 1.0}%% Cambiar este valor para escalar el diagrama
{
%% Tipo diagrama UML: UMLClassDiagramModel
\begin{pspicture}(0.0000000000,0.0000000000)(17.6741666667,-14.4727)

%% F I G U R A S

%% UMLClassCFModel


%% UMLClassBoxNameCFM
\newrgbcolor{figureFillColor}{1.00 1.00 1.00}
\newrgbcolor{figureLineColor}{0.00 0.00 0.00}
\psframe[fillstyle=solid,fillcolor=figureFillColor,linecolor=figureLineColor]
\newrgbcolor{textColorName}{0.00 0.00 0.00}


```

# Lineas: 188 Caracteres: 8780

Figura 73. Panel con código PSTricks generado.

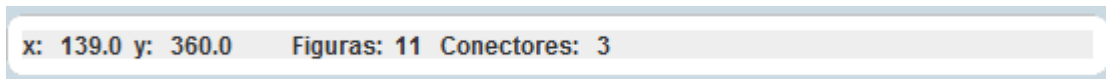
En el panel de barra de herramientas se pueden realizar las siguientes tareas:

Icono	Nombre	Descripción
	Copiar código.	Copia en el portapapeles el código seleccionado en este panel.

	Exportar como código PST	Exporta el diagrama a un archivo .tex como macros PSTricks
---	--------------------------	--

### 3.1.5. Panel de barra de estado de la aplicación.

El panel de barra de estado (ver Figura 74) contiene información del estado de la aplicación.



*Figura 74. Panel de estado de la aplicación.*

- **Posición x:** Posición X del cursor en el panel de dibujo.
- **Posición y:** Posición Y del cursor en el panel de dibujo.
- **Figuras:** Cantidad de figuras en el dibujo actual.
- **Conectores:** Cantidad de conectores del diagrama actual.

## 4. Operaciones de la aplicación

### 4.1. Crear Diagrama

3. El usuario solicita a la aplicación la realización de una operación de creación de nuevo diagrama. (Elegir menú **Archivo** → **Nuevo...** → [Elegir el diagrama que se desea crear]). (Ver Figura 75)

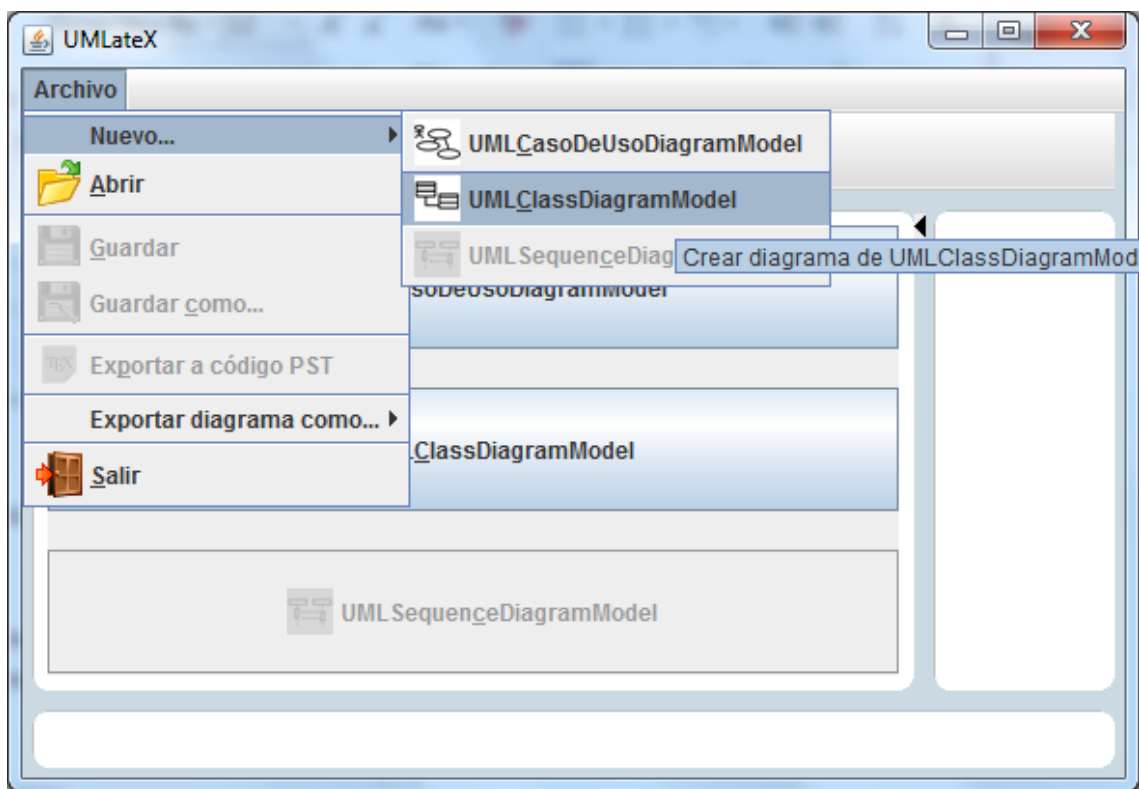


Figura 75: UMLaTeX, elegir nuevo diagrama

4. La aplicación muestra en la ventana principal las diferentes vistas del nuevo modelo (panel de trabajo, panel de código). (Ver Figura 76)



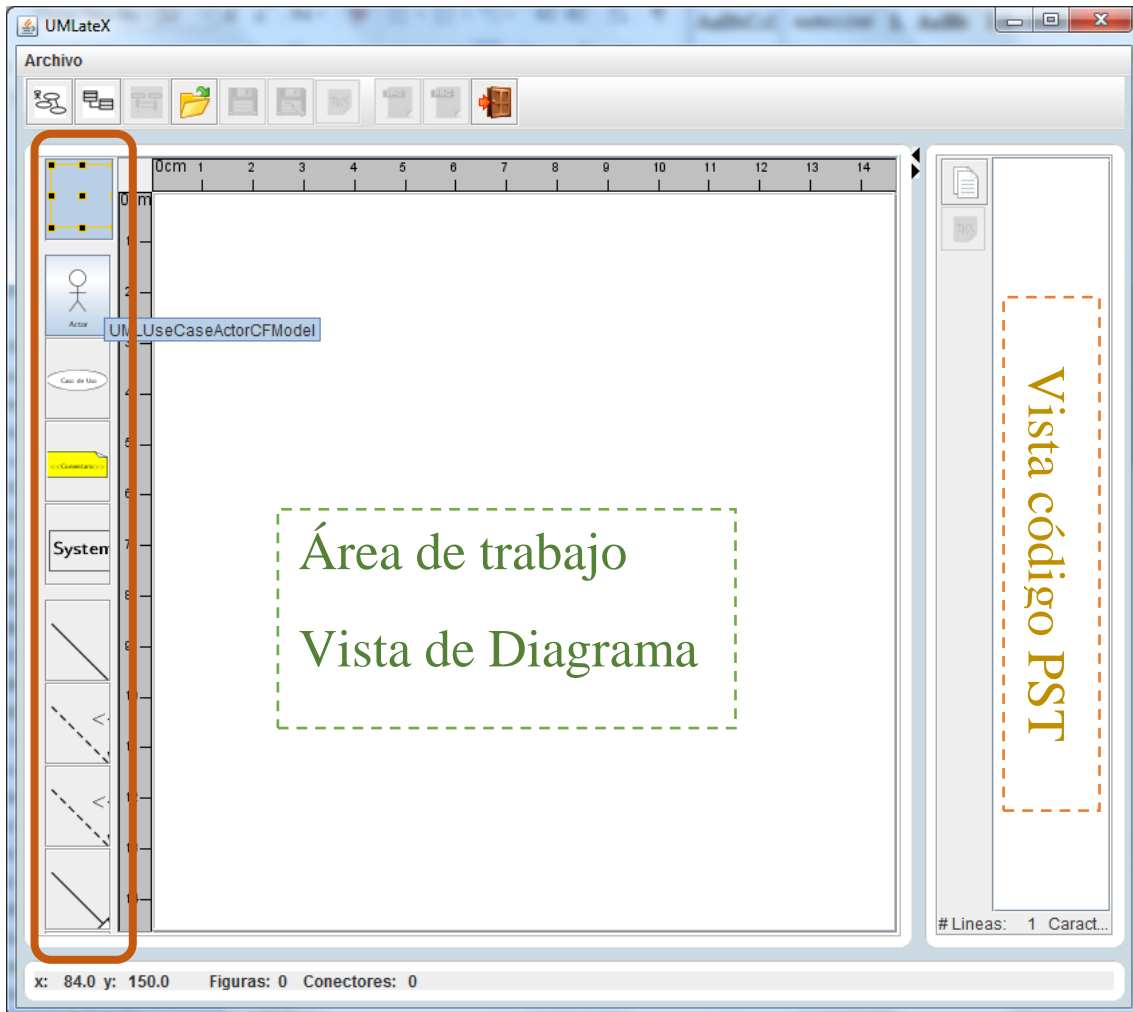


Figura 76: UMLateX, área de trabajo mostrada tras crear un nuevo diagrama. A la izquierda, la barra de herramientas correspondiente al diagrama UML elegido. Al centro el área de trabajo con la Vista del diagrama. A la derecha la Vista de código PSTricks.

## 4.2. Recuperar Diagrama

4. El usuario solicita a la aplicación recuperar (abrir) un diagrama previamente guardado por la aplicación (ver Figura 77).

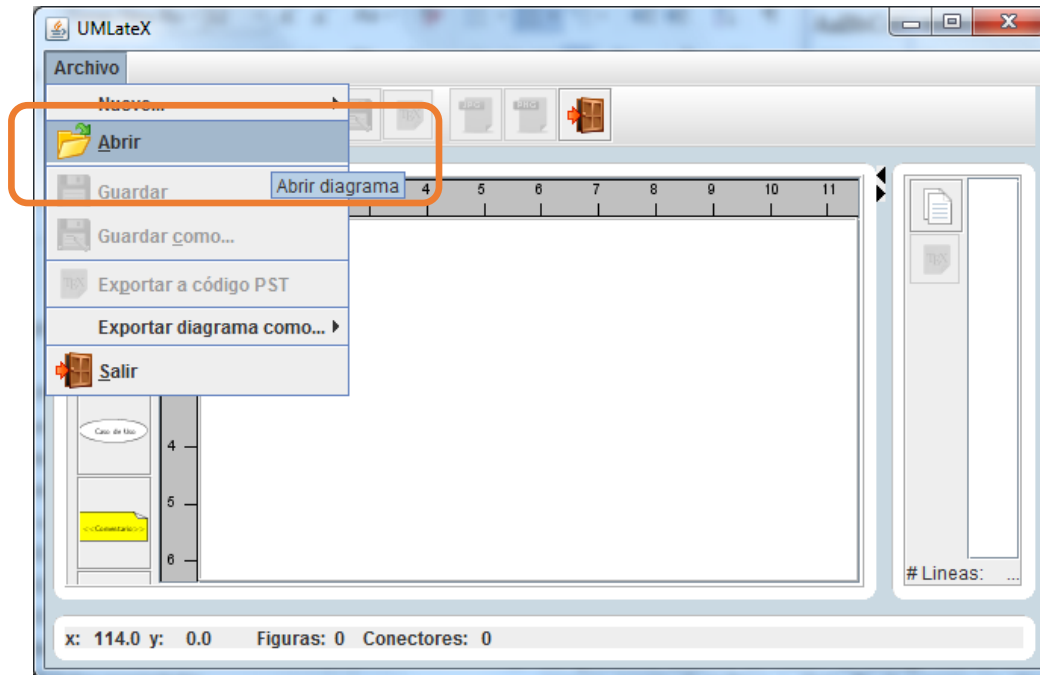


Figura 77: UMLaTeX. Abrir un diagrama

5. La aplicación muestra un cuadro de diálogo con un sistema de archivos para que el usuario escoja el archivo a abrir. (Ver Figura 78).

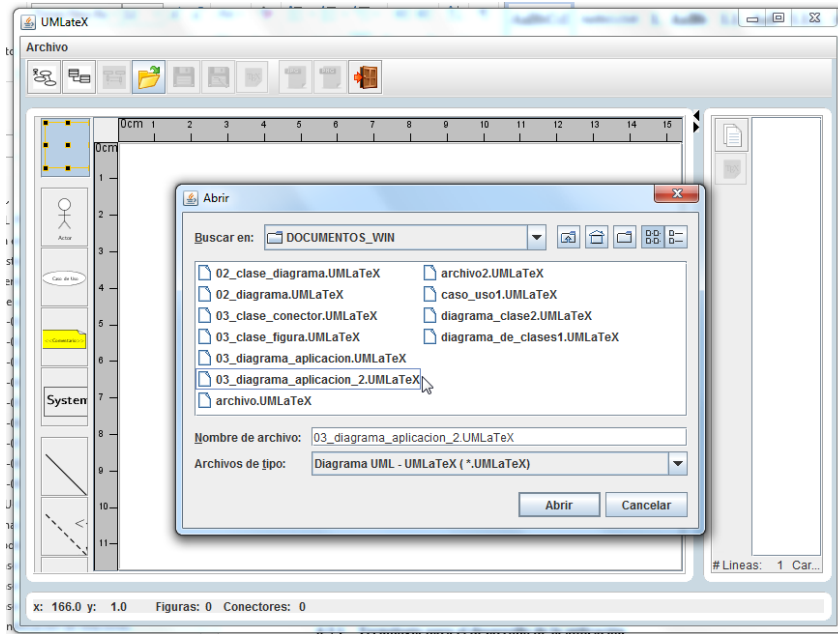


Figura 78: UMLaTeX. Cuadro de diálogo para abrir un diagrama \*.UMLaTeX

6. La aplicación muestra el diagrama recuperado (ver Figura 79).

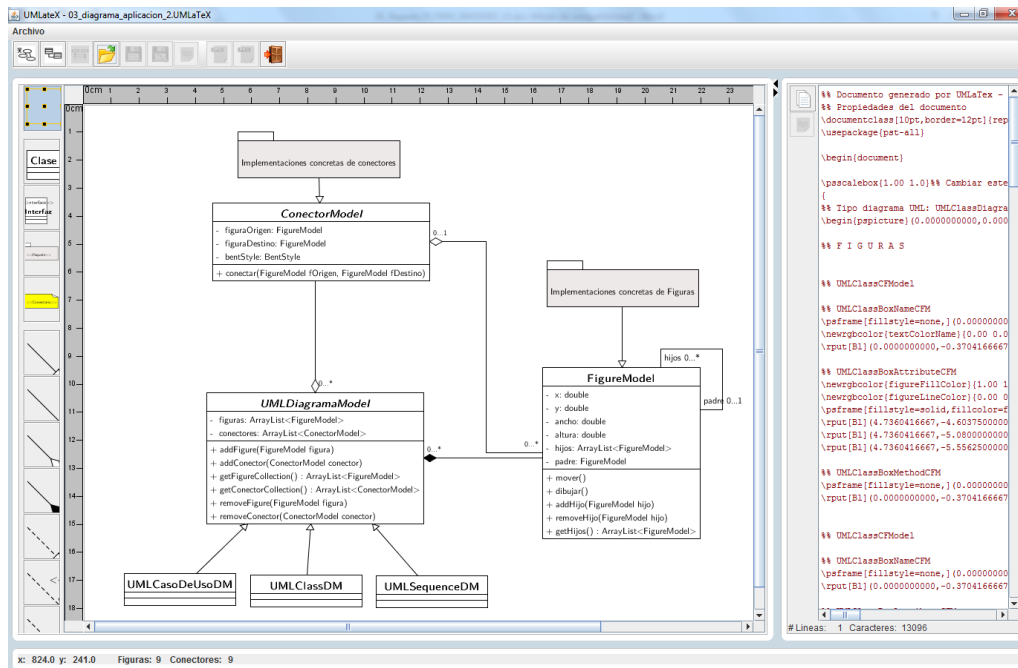


Figura 79: UMLaTeX. Muestra diagrama recuperado tras abrir el archivo.

### 4.3. Editar diagrama. Mover figuras

2. El usuario elige un artefacto UML (figura o conector) y lo arrastra (ver Figura 80).

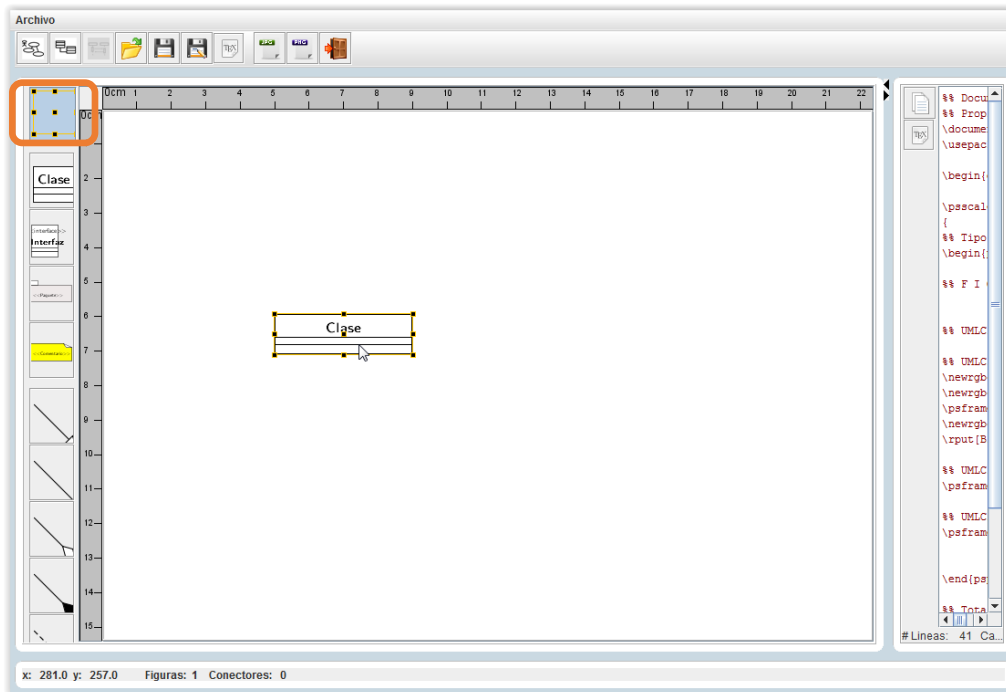


Figura 80:UML4TeX. La herramienta de selección de figura siempre estará en la parte superior de la barra de herramientas.

### 4.4. Agregar Figura/Conector

#### Agregar Figura:

4. El usuario elige de la barra de herramientas un artefacto UML (figura). (Ver Figura 81)

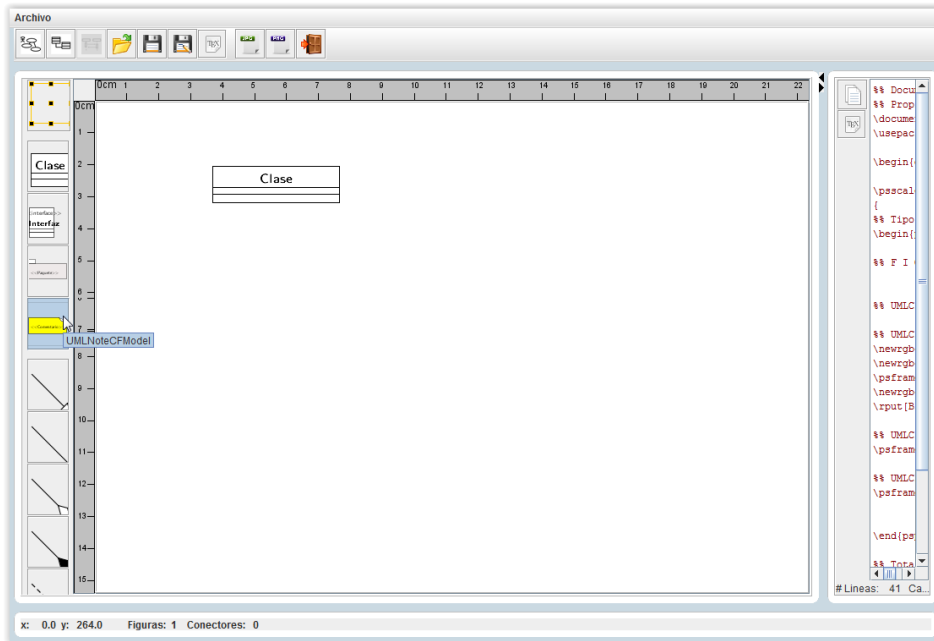


Figura 81: UMLaTeX. El usuario puede elegir de la barra de herramientas el artefacto UML a agregar al diagrama.

5. El usuario se posiciona en el área de edición y oprime el botón izquierdo para colocar el artefacto UML previamente elegido (ver Figura 82).
6. La aplicación agrega el nuevo componente al modelo del diagrama.

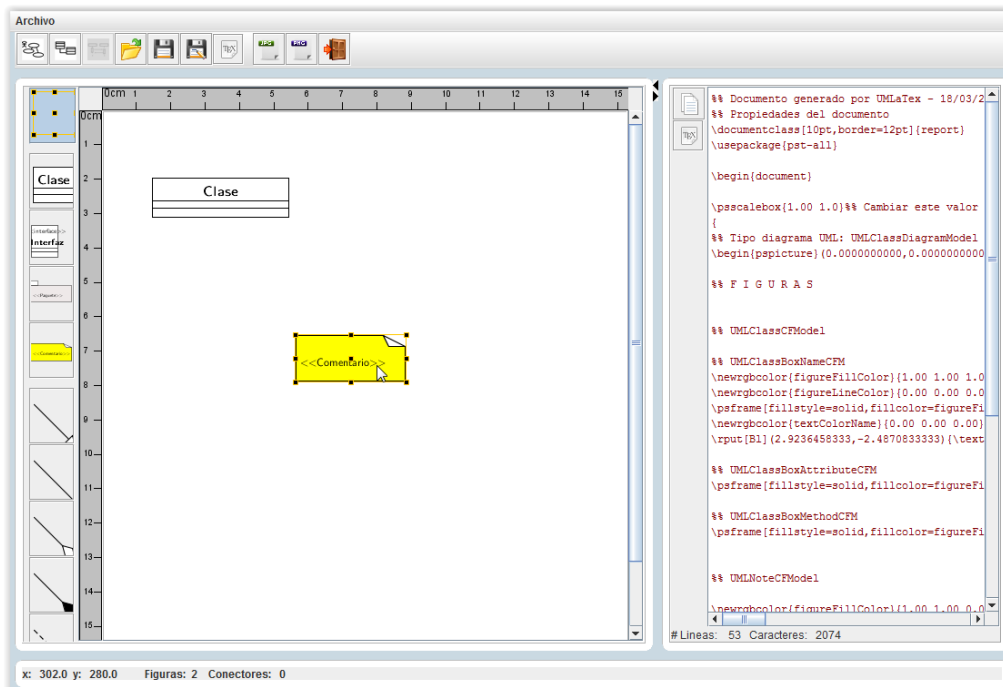


Figura 82: UMLaTeX. La aplicación actualiza el modelo y las vistas al agregar una nueva figura.

## Agregar Conector:

4. El usuario elige de la barra de herramientas un artefacto UML (conector). (Ver Figura 83).

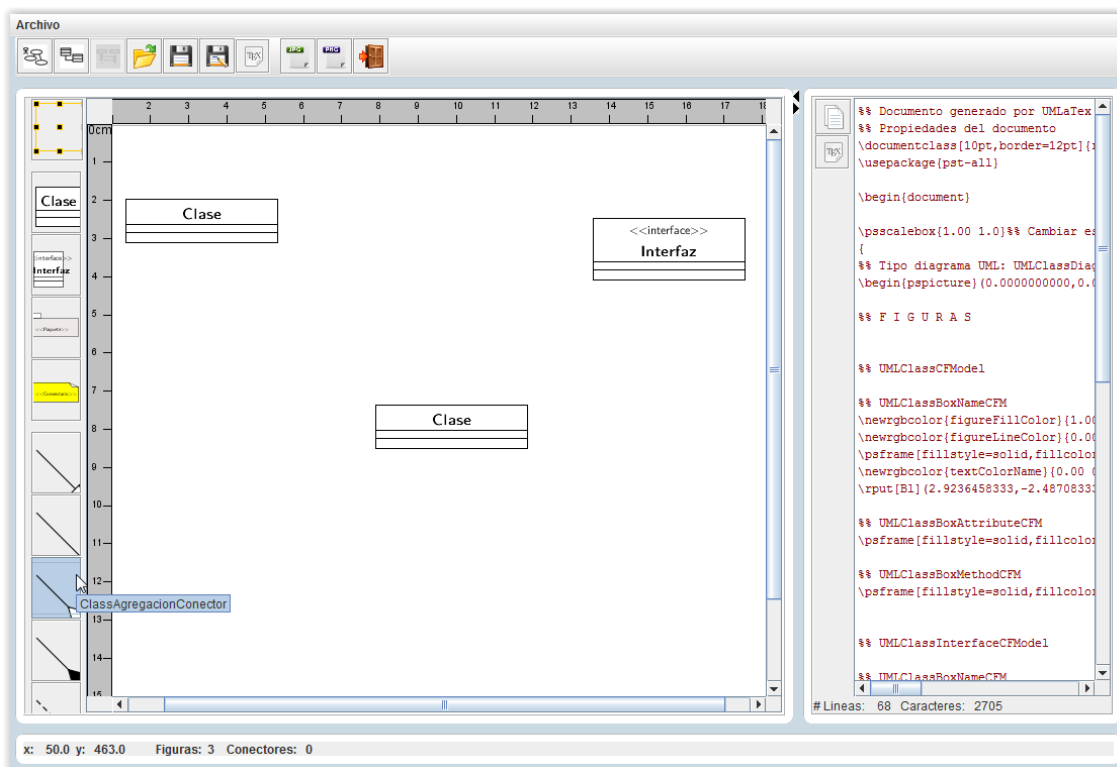


Figura 83: UMLaTeX. El usuario puede elegir de la barra de herramientas el artefacto UML a agregar al diagrama.

5. El usuario se posiciona en el área de edición y oprime el botón izquierdo del ratón para colocar el artefacto UML previamente elegido (Ver Figura 84).
6. La aplicación agrega el nuevo componente al modelo del diagrama.

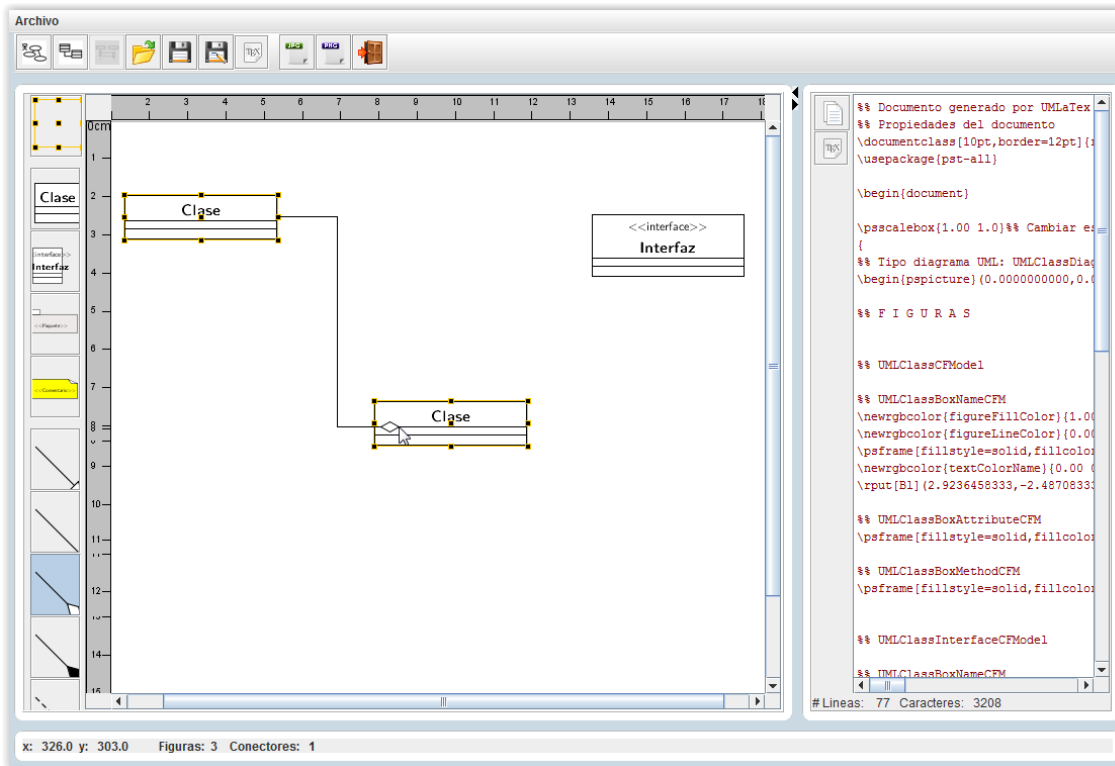


Figura 84:UMLaTeX. La aplicación actualiza el modelo y las vistas al conectar dos figuras.

#### 4.5. Editar propiedades

6. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona (ver Figura 85).

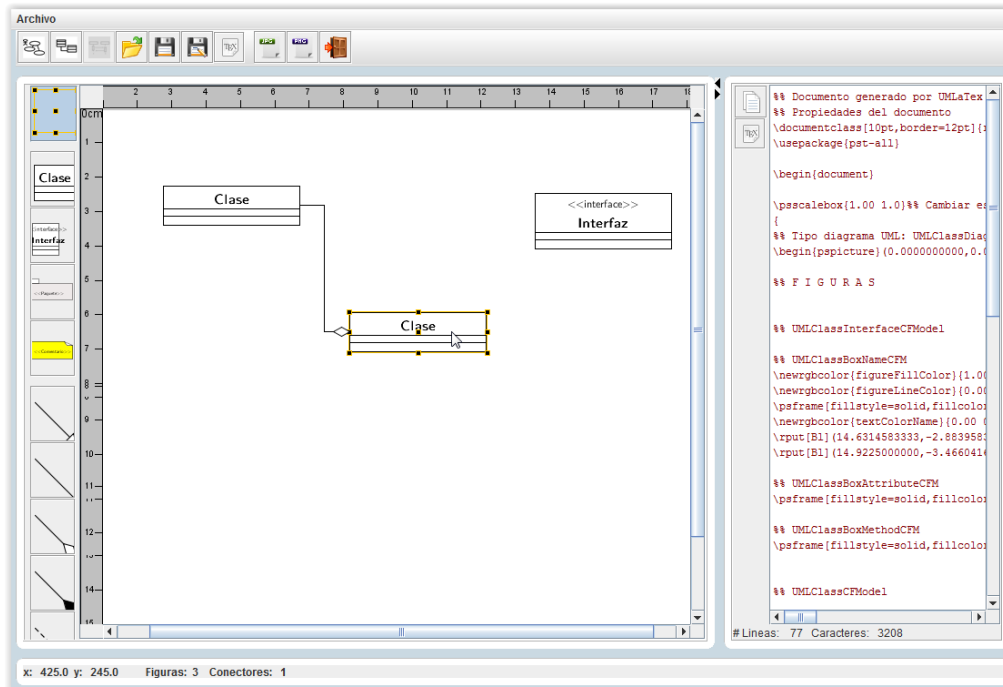


Figura 85. UML4TeX. Al elegir una figura se resaltarán sus bordes en color naranja.

7. El usuario pide a la aplicación Editar el artefacto UML seleccionado. Esto se logra haciendo dos clics en la figura seleccionada.
8. La aplicación muestra el formulario relacionado con el artefacto UML a editar (ver Figura 86).

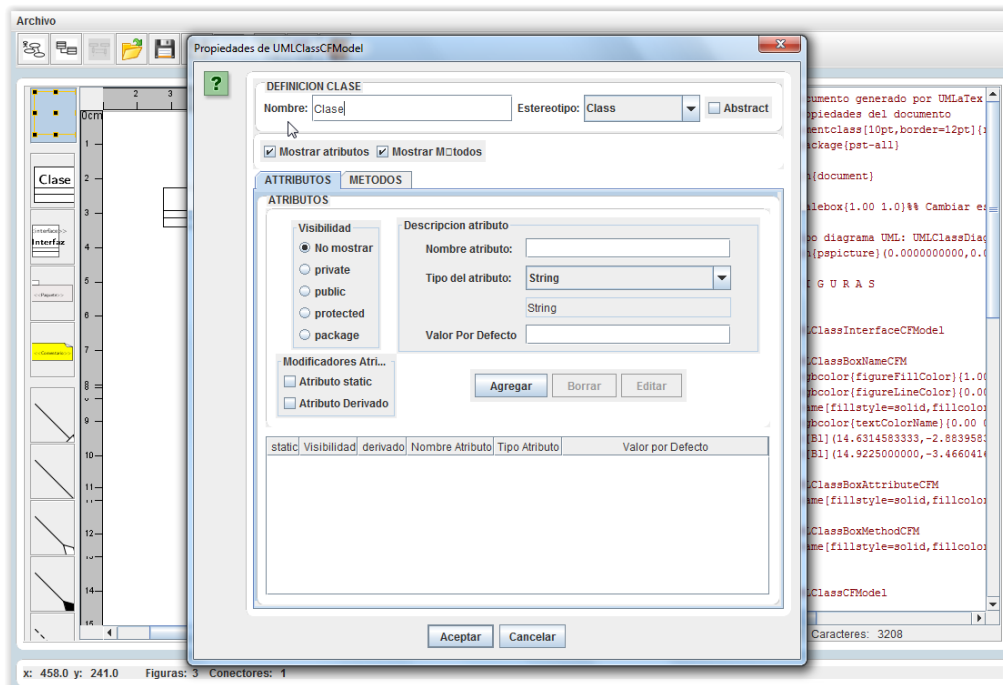




Figura 86:UML4TeX. Cada artefacto UML (Figura/Conector) tiene un formulario destinado para editar sus propiedades.

9. El usuario edita las propiedades y acepta los cambios (ver Figura 87).

Propiedades de UMLClassCFModel

DEFINICION CLASE  
 Nombre: MiClase    Estereotipo: <<Control>>     Abstract

Mostrar atributos     Mostrar Métodos

ATRIBUTOS    METODOS

ATRIBUTOS

Visibilidad  
 No mostrar  
 private  
 public  
 protected  
 package

Descripcion atributo  
 Nombre atributo: atributo2  
 Tipo del atributo: Integer  
 Integer  
 Valor Por Defecto:

Modificadores Atri...  
 Atributo static  
 Atributo Derivado

Agregar    Borrar    Guardar

static	Visibilidad	derivado	Nombre Atributo	Tipo Atributo	Valor por Defecto
false	-	false	atributo1	Character	
false	#	false	atributo2	Integer	

Aceptar    Cancelar

Figura 87:UML4TeX. Formulario de propiedades para el artefacto UML Clase.

10. El diagrama muestra los cambios realizados por el usuario (ver Figura 88).

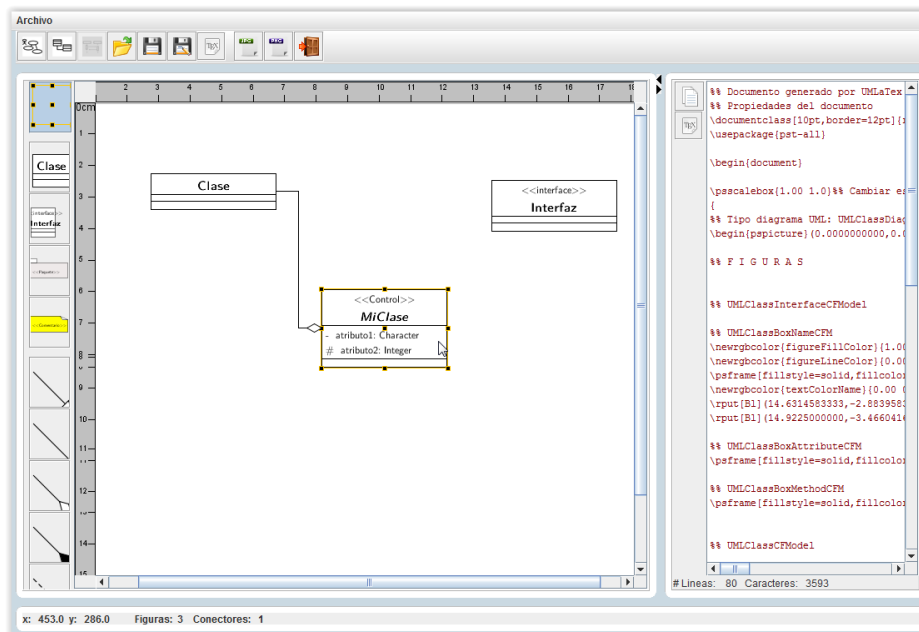


Figura 88:UML4TeX. Se actualiza el diagrama tras la edición de las propiedades de un artefacto UML.

## 4.6. Borrar Figura/Conector

3. El usuario elige un artefacto UML que se encuentra en el área de edición y lo selecciona (ver Figura 89).

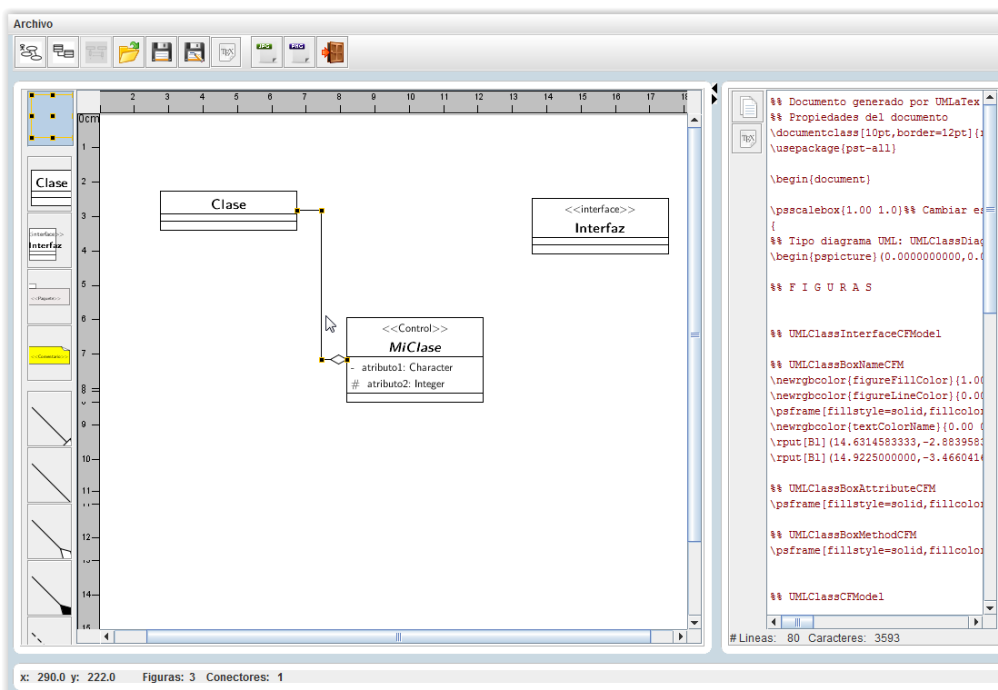


Figura 89:UMLaTeX. La selección de figuras/conectores se realiza con la herramienta de selección.

4. El usuario pide a la aplicación borrar el artefacto UML seleccionado. Esto se logra oprimiendo la tecla **SUPRIMIR**. (Ver Figura 90).

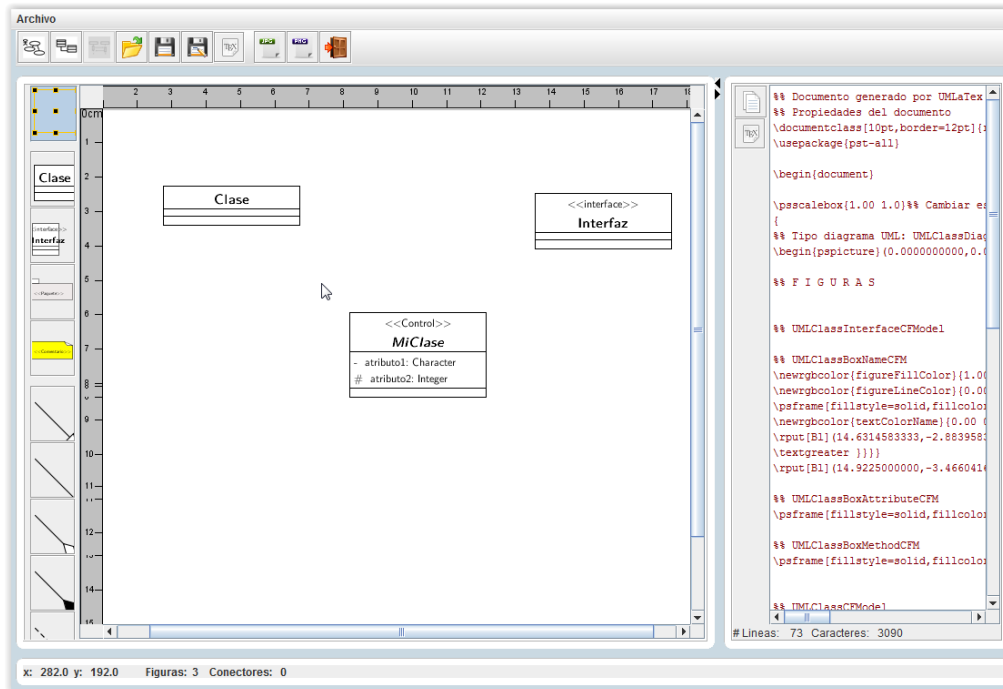




Figura 90: Se pueden eliminar los artefactos UML oprimiendo la tecla SUPRIMIR

## 4.7. Guardar Diagrama

3. El usuario solicita a la aplicación que se ejecute una operación de salvado. Esto lo puede hacer presionando el botón de la barra de herramientas Guardar  o Guardar Como...  también puede hacerlo seleccionando el menú **Archivo** → **Guardar/Guardar Como...** (Ver Figura 91).

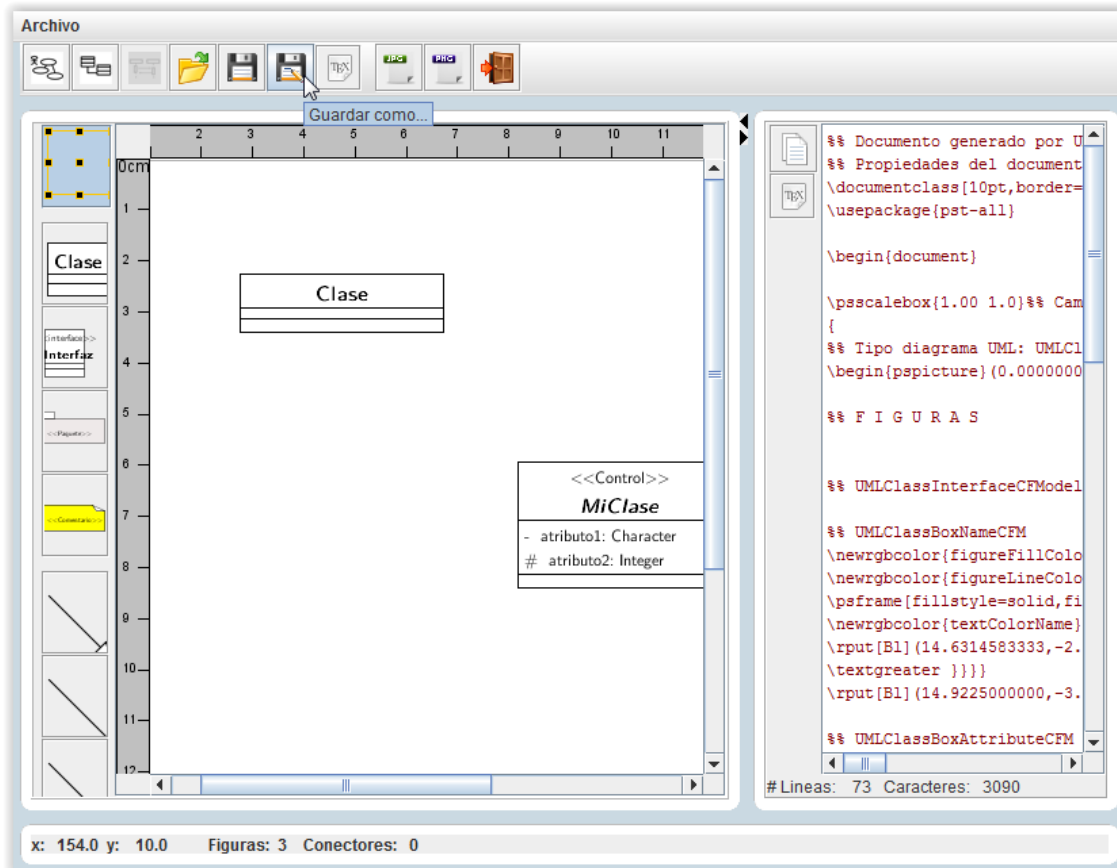


Figura 91: UML4TeX. Las opciones de Guardar/Guardar Como... están disponibles en la barra de herramientas principal de la aplicación o en el Menú.

4. La aplicación muestra un diálogo con un sistema de archivos para elegir el nombre y ubicación del archivo a guardar. El usuario elegirá cómo guardar su archivo (ver Figura 92).

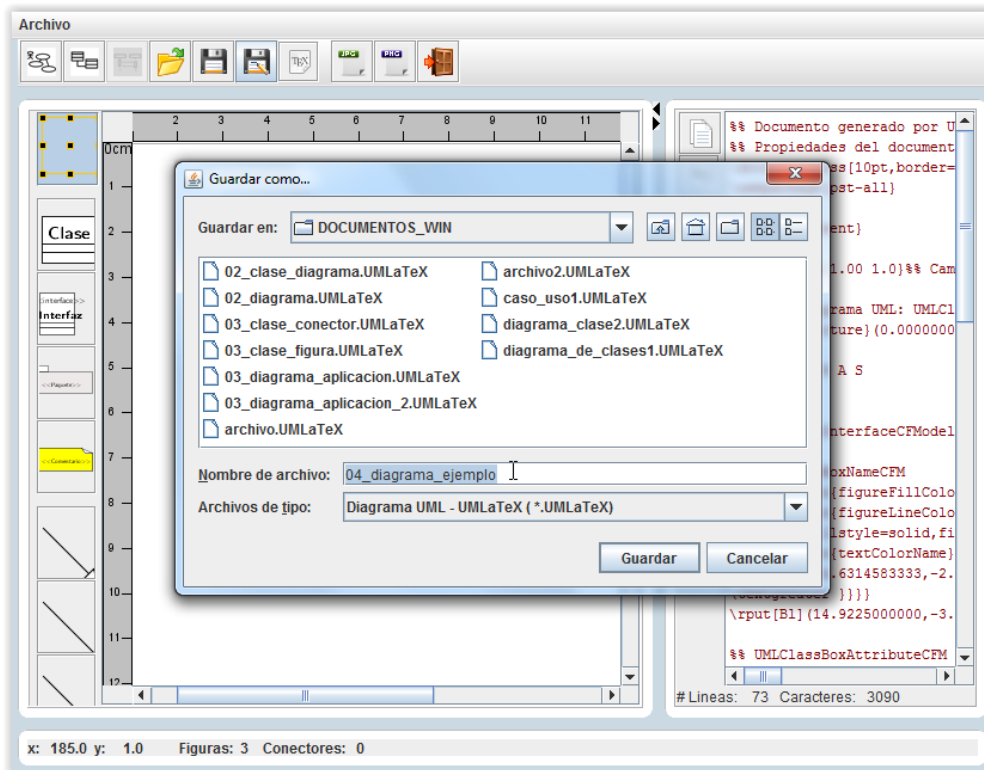




Figura 92: UMLaTeX. Los diagramas generados por la aplicación son guardados en formato XML y con la extensión \*.UMLaTeX

## 4.8. Exportar Diagrama como PSTricks

4. 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 de la barra de herramientas principal 
- b. Accediendo al menú **Archivo** → **Exportar a código PST**.
- c. En el panel de vista de código PST también está habilitada la el botón  para exportar el diagrama (ver Figura 93).

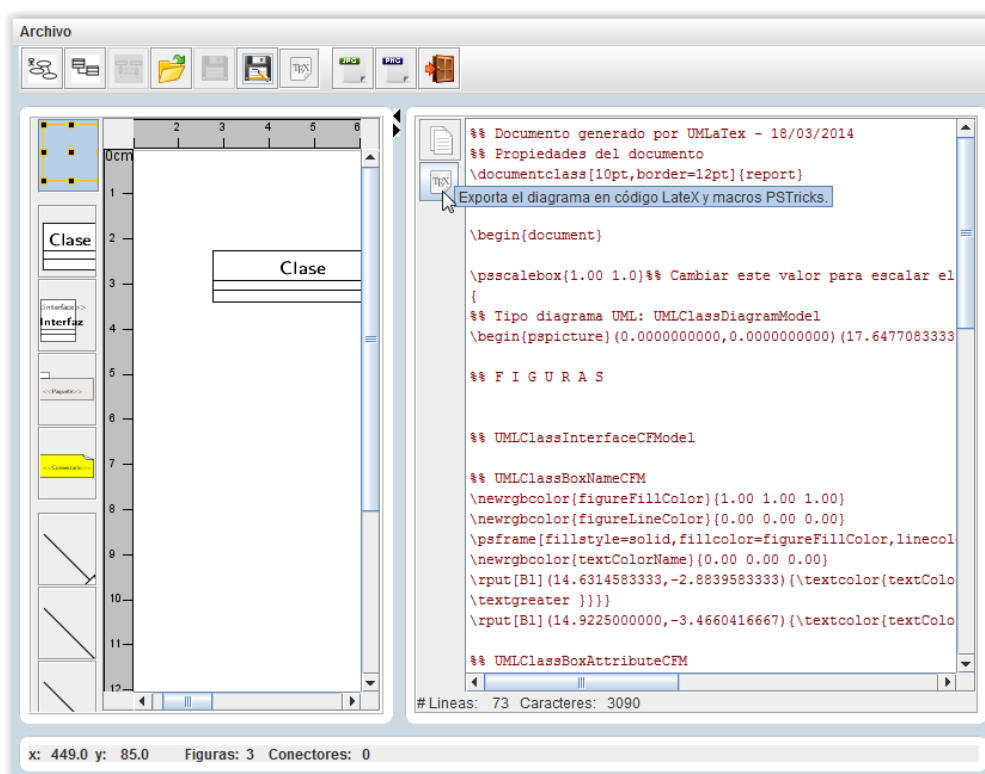


Figura 93: UMLaTeX. La aplicación ofrece varios métodos para exportar los diagramas a macros PSTricks.

5. La aplicación abrirá un diálogo con sistema de archivos donde elegirá la ubicación del archivo \*.tex de salida. (Ver Figura 94).

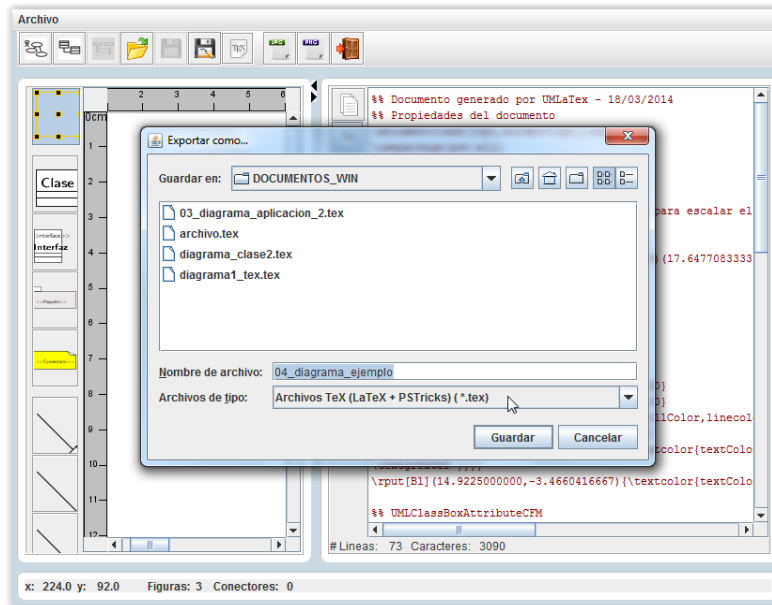


Figura 94: UMLaTeX. Los diagramas pueden ser convertidos a macros PSTricks, que son integrables a documentos LaTeX

6. La aplicación avisará el éxito de la operación (ver Figura 95).

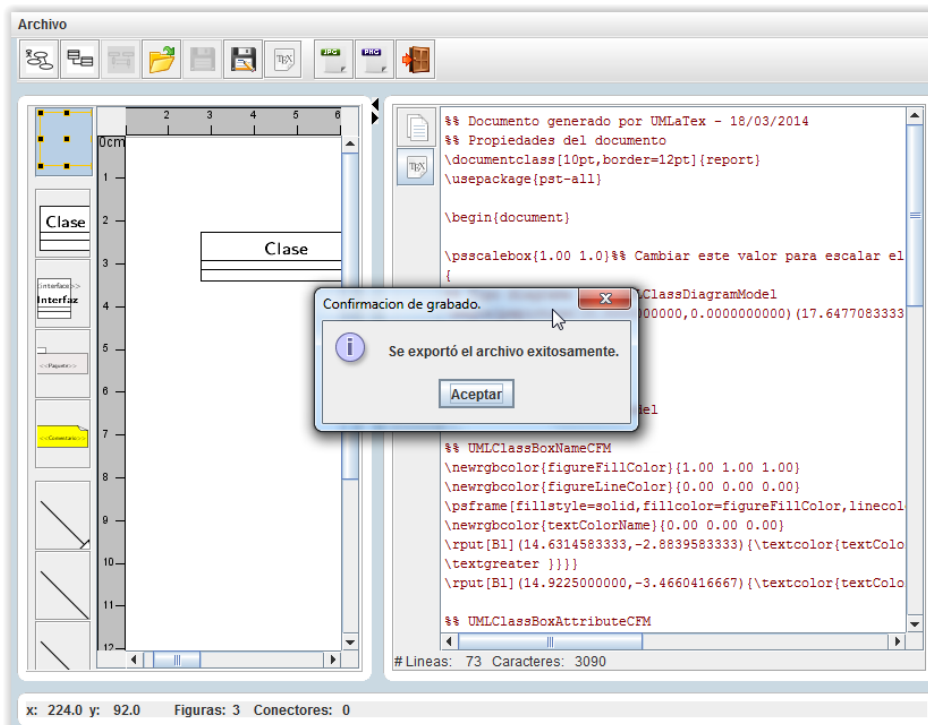




Figura 95: UMLaTeX. La aplicación genera el archivo \*.tex con las macros PSTricks del diagrama actual.

#### 4.9. Exportar Diagramas en formato Imagen (PNG/JPG)

3. El usuario solicita a la aplicación que se ejecute una operación de exportar como imagen (ver Figura 96). 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**

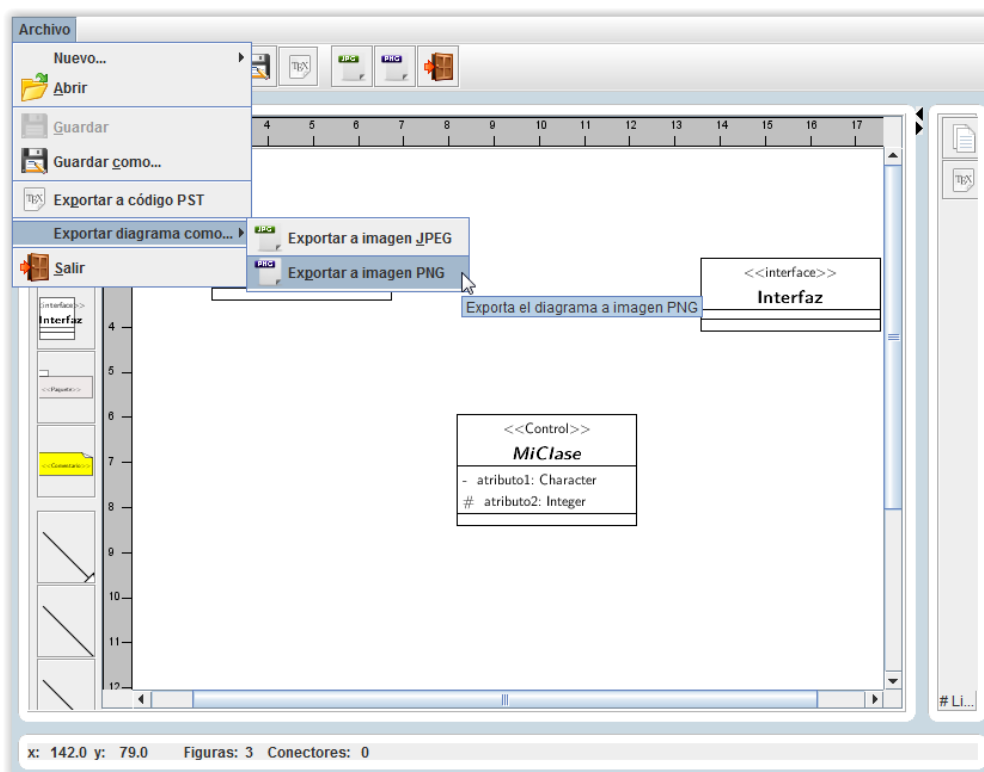


Figura 96: UML4TeX. La aplicación ofrece la posibilidad de exportar el diagrama generado en formato de imagen JPG y PNG

4. La aplicación pregunta al cliente dónde guardar el archivo de imagen de salida (ver Figura 97).



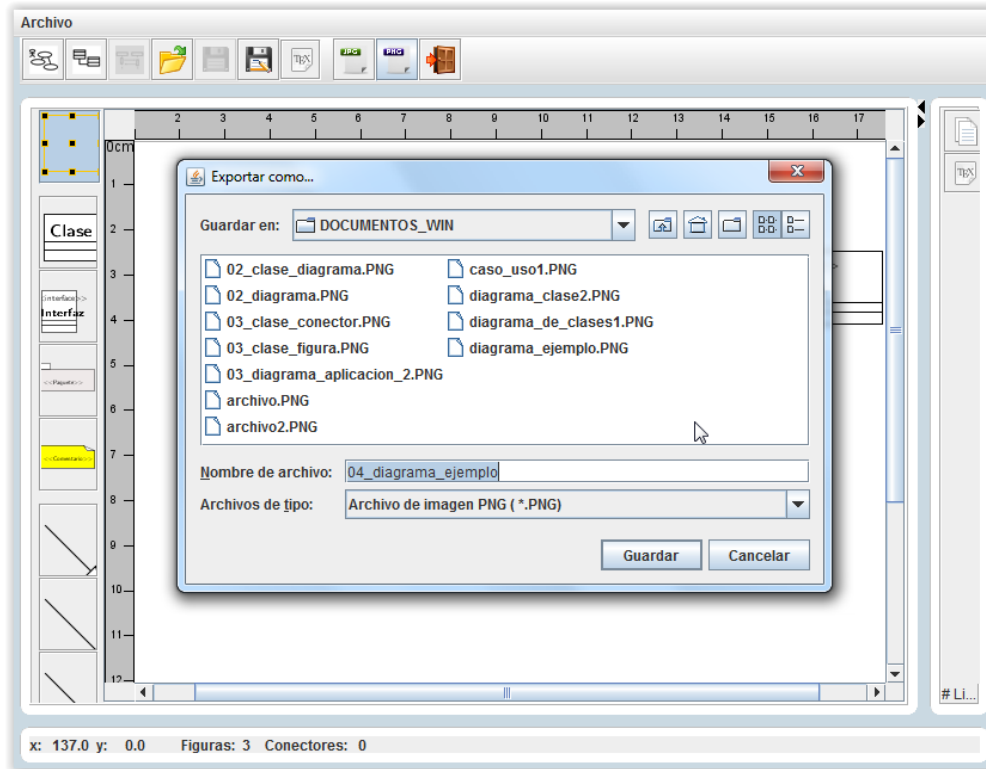



Figura 97: UML4TeX. La aplicación realiza la operación de exportación en el formato indicado (PNG/JPG).

#### 4.10. 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 98).

Alternativamente puede presionar el botón  de la barra de herramientas.

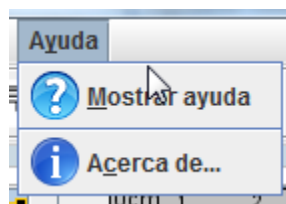


Figura 98. Menú Ayuda

2. La aplicación abrirá una nueva ventana con el contenido de la ayuda de la aplicación (ver Figura 99 ).

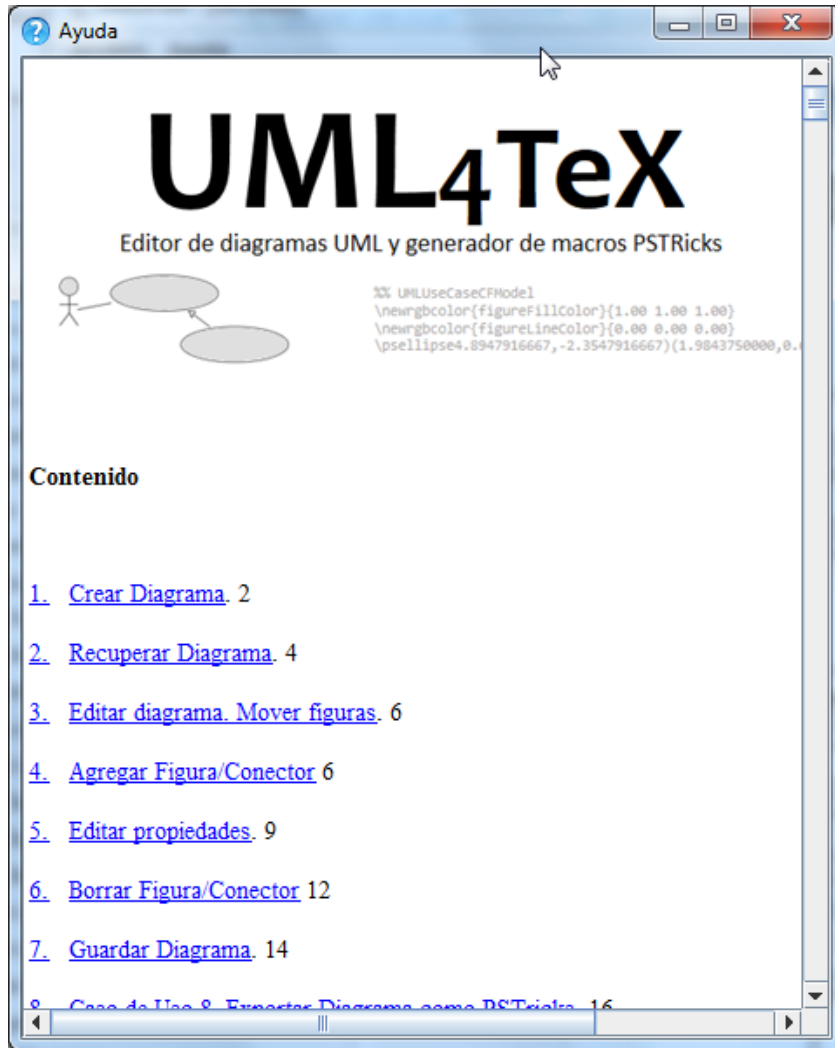


Figura 99. Ventana de ayuda de la aplicación.