Universidad de las Ciencias Informáticas

Facultad 1

Título: **Análisis, diseño y arquitectura de la información de la herramienta SLAM-GraphicDesigner**.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Danieska Ramirez Molina

Roxana Duarte Páez

Tutor:

Ing. Antonio Membrides Espinosa

Ciudad de la Habana, junio del 2010

Año 52 de la Revolución.

"El futuro de Cuba tiene que ser necesariamente un futuro de hombres de ciencia, de hombres de pensamiento."

Fidel

Declaración de autoría

Declaramos que nosotras Danieska Ramirez Molina y Roxana Duarte Páez somos las únicas autoras de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio siempre y cuando se nos acredite.

Para que así conste firmamos el presente trabajo a los 24 días del mes de junio del año 2010.

_____ _____

Danieska Ramirez Molina Roxana Duarte Páez

Firma del autor Firma del autor

_____

Ing. Antonio Membrides Espinosa

Firma del tutor

Dedicatoria

A mis padres, por constituir este momento uno de sus mayores sueños, porque sin su ayuda y sacrificio no hubiera sido posible realizarlo.

A mis hermanas por quererme tanto y darme siempre aliento y fuerzas para seguir adelante.

A Maikel por su amor, su apoyo y sobre todo por esperarme tanto tiempo.

Danieska Ramirez Molina

A mi mamá por ser mi amiga en todo momento y apoyarme cuando más la he necesitado.

A tata por darme fuerza y sustento en todo momento como si fuera mi papá, guiándome siempre por el camino correcto.

Al flaco por ser ese niño con pensamiento de hombre que me ha apoyado constantemente con la fe de que al final lo lograría.

A Alexander por estar todo este último tiempo apoyándome cuando más lo he necesitado.

A mi abuela la Prieta que esté donde esté siempre la llevo en mi corazón.

Roxana Duarte Páez

Agradecimientos

A mis padres y hermanas a quienes les debo hoy todo lo que soy.

A Maikel, que sin su apoyo no hubiera sido posible.

A mi abuela Benedicta, que donde quiera que esté estará orgullosa de mí.

A toda mi familia y a Yoandry, por preocuparse siempre por mí.

A mi tutor Antonio Membrides por su paciencia y dedicación.

A todos mis amigos(as) con los que he tenido la dicha de compartir estos maravillosos años que nunca olvidaré, y sobre todo a Roxana por aguantarme en todo este tiempo de difícil trabajo.

Danieska

A mi mamá y mis hermanos por ser mi todo y por apoyarme siempre.

A Alexander por compartir conmigo momentos inolvidables y convertir mis tristezas en alegrías, y cada alegría en un sueño. Te quiero.

A mi mejor amigo Albertico por quererme y apoyarme siempre.

A mi tía Laura y mi prima Milena por todo el amor que me han transmitido.

A mi abuela Cheche y a mi tía Lidia por permitirme contar con ellas siempre.

A mis amigos Yaima, Puchy, Mary, Taimí, Juanqui y Karima, por ser excelentes amigos.

A Danieska que a pesar de que teníamos nuestras discrepancias estuvo ahí siempre conmigo para hacer realidad este sueño.

A mi suegra Ilse que en este tiempo ha sabido ganarse mi respeto y cariño, gracias por tu preocupación.

Roxana

Agradecemos a nuestro comandante en jefe Fidel Castro Ruz y a la Revolución cubana por darnos la posibilidad de desfilar en esta gran tropa de futuro.

A Mora y a todas las personas que de alguna forma han colaborado con la realización del presente trabajo.

Resumen

La industria del software constituye una alta prioridad para el gobierno cubano, especialmente aplicada al desarrollo de herramientas de diseño destinadas a la confección de productos multimedia, que juegan un importante papel en el desarrollo y aprendizaje de las nuevas generaciones. Sin embargo es además una de las razones por las que se debe alcanzar la independencia tecnológica y evitar el uso de herramientas propietarias. Es significativo destacar que la mayoría de estas herramientas están orientadas a plataformas como Windows, rompiendo con la estrategia de migración a software libre que se ha trazado el país con el objetivo de lograr una mayor independencia tecnológica.

Teniendo en cuenta que todos estos elementos limitan considerablemente el uso de este tipo de software, sumado a la necesidad de herramientas totalmente libres y robustas para la construcción de multimedia, se decidió confeccionar una herramienta de diseño multiplataforma como complemento del marco de trabajo SLAM-C++, dedicado a la construcción de multimedia.

En el presente trabajo se exponen los resultados de todo el proceso investigativo, así como las razones que hacen de vital importancia desarrollar herramientas de diseño propias para la construcción de productos multimedia, pues se muestran los argumentos que demuestran que la situación problémica requiere de un sistema que cumpla con los requisitos propuestos para satisfacer las necesidades identificadas. Posteriormente se procede a conceptualizar el sistema propuesto y se lleva a cabo la modelación del mismo.

PALABRAS CLAVES.

Análisis, Arquitectura de la información, Diseño, *Framework,* Multimedia

Índice

Introducción

En la actualidad la evolución de las tecnologías de la informática y las comunicaciones(TIC), ha tenido un impacto trascendental en varios niveles de la sociedad internacional, donde los medios de comunicación como el correo electrónico, el fax y el satélite han provocado altos grados de desarrollo en esta rama, ocasionando que las multimedia estén evolucionando cada vez más como una forma sencilla de comunicación, siendo estas un producto que combina textos, gráficos, sonidos, animaciones y videos.

La tecnología multimedia forma parte de la informática por su capacidad de interactividad. Esta se ha convertido en un medio de comunicación tecnológico desempeñándose en diferentes ámbitos sociales. Entre sus principales y conocidas utilidades están la diversión y entretenimiento, posibilitando la creatividad mediante los sistemas de computación. Las multimedia convierten el diálogo, hombre-máquina en algo intuitivo, espontáneo y divertido**[1]**. Son empleadas en pasatiempos de tipo cultural como trabajos infantiles y video juegos, en el mundo del negocio se emplean en la publicidad del marketing para las ofertas y difusión de los productos **[2]**. Para su desarrollo se utilizan herramientas de diseño que actúan como programas de computador de gran importancia, ampliando sus posibilidades creativas y la eficacia en una amplia variedad de operaciones como trabajo con fotos, y otros recursos media.

Desde sus inicios las herramientas de diseño han sufrido una serie de cambios, convirtiéndose en la solución estándar de la industria para el desarrollo profesional, al incorporar una amplia función de soporte para el diseño y mejora del software. Estas permiten agilizar del flujo de trabajo para que los usuarios puedan hacer más cosas en menos tiempo. También aportan las más avanzadas metodologías de diseño profesional, crean atractivos gráficos, películas animadas dadas por la fusión de pantalla de agujas e imágenes digitales obtenidas directamente de programas informáticos de trabajo en 3 dimensiones. Se usan además para enaltecer y retocar el trabajo realizado por los diseñadores que hacen posible las presentaciones multimedia.

Estas tecnologías no solo son empleada en la creación de aplicaciones multimedia, también se utilizan en el desarrollo de páginas web*,* entre otros. Cada una tiene una forma diferente de trabajar, muchas de ellas permiten dibujar, pintar, seleccionar y modificar ilustraciones basándose en la metáfora cine, donde los actores son los recursos textos, sonidos, imágenes, entre otros.

La compañía *Adobe Systems* es el líder del mercado, constituyendo un significativo paquete de desarrollo que se dedican a la elaboración de herramientas de diseño para construir aplicaciones multimedia. Entre las herramientas más exitosas que desarrollan se encuentran: *Adobe Photoshop*,

Adobe Flash, Adobe Director, Adobe Fireworks CS4 y otras. Un elemento preponderante de todas estas herramientas sofisticadas para el diseño es que son privativas y producen dependencia tecnológica. Por estas razones existe la necesidad de crear herramientas libres para el desarrollo de productos multimedia y sobre todo que sean multiplataforma.

Los grandes monopolios de la informática se adueñan cada vez más de las aplicaciones y servicios que se ofrecen en el mundo, la mayoría son industrias norteamericanas que impiden la comercialización con Cuba debido al bloqueo impuesto por el gobierno de los Estados Unidos. Por estas razones el país decide migrar a software libre por los altos precios que hay que pagar para adquirir estos productos. GNU/Linux es la alternativa a este problema porque posibilita a los usuarios trabajar libremente, eliminar la dependencia tecnológica y abrir las puertas a la industria del software en el mercado nacional e internacional.

GNU/Linux es un sistema operativo muy potente que dispone de herramientas de diseño para multimedia pero estas no cumplen con los requisitos que imponen estos productos para desarrollarse y su convergencia podría ser extremadamente compleja. En la Universidad se encuentra actualmente en desarrollo un proyecto llamado AUVIX que pretende unir cuatro herramientas libres para la producción de audiovisuales, sin embargo su desarrollo es aún incipiente constituyendo un inconveniente para el crecimiento de estos productos. Existe otro proyecto con similares perspectivas llamado Grupo de Alternativas Libres para Multimedia (GALM) que se encuentra actualmente en evolución.

De igual forma a finales del año 2007 se inició el desarrollo del marco de trabajo SLAM-C++, este tenía como objetivo hacer uso de tecnologías libres para construir multimedia, teniendo gran importancia debido a la carencia de herramientas sólidas para crear dichos productos sobre plataformas libres. Este *framework* a pesar de ser una solución innovadora no consta de una herramienta de diseño para la generación de contenidos multimedia basados en la metáfora cine, que agilice el proceso de construcción de estos productos y amplíe los límites de las posibilidades creativas, a partir del control visual que esta le pueda ofrecer a los diseñadores sobre los recursos media.

A raíz de esta problemática surge como **problema a científico**: ¿Cómo lograr la generación de contenidos interactivos basados en la metáfora cine para el *framework* SLAM-C++?

Por tanto, como **objeto de estudio** se define: las herramientas de diseño.

El **campo de acción** que abarca esta investigación es: la herramienta SLAM-GraphicDesigner.

Teniendo como **objetivo general** del trabajo proponer el diseño de una herramienta para el *framework* SLAM-C++ que permita la generación de contenidos interactivos. Con el objetivo de darle cumplimiento al presente trabajo se definieron los siguientes **objetivos específicos***:*

- Determinar aspectos teóricos - conceptuales asociados con las herramientas de diseño para multimedia.
- Proponer prototipos funcionales de acordes con los esquemas de organización definidos en la arquitectura de la información para SLAM_GraphicDesigner.
- Delimitar las necesidades del cliente para el análisis y diseño de la herramienta SLAM_GraphicDesigner
- Establecer una arquitectura que permita al sistema adaptarse a nuevas funcionalidades, incluyendo elementos visuales.
- Proponer un estándar para la generación de proyectos con SLAM_GraphicDesigner.

Para el desarrollo del trabajo se cuenta con una serie de **tareas** que ayudarán a cumplimentar la construcción del mismo:

- Elaboración del estado del arte atendiendo a las herramientas de diseño estudiadas, principalmente aquellas que modelen la metáfora de trabajo cine.
- Análisis de los estándares para la generación de proyectos basados en la metáfora de trabajo cine.
- Definición y documentación de los estándar para generación de proyectos basados en la metáfora de trabajo cine que utilizará SLAM_GraphicDesigner.
- Estudio de las técnicas de diseño gráfico con el objetivo de obtener un sistema de organización y estructuración de los contenidos más completos.
- Definición de las técnicas de representación de la información que serán utilizadas en la construcción de SLAM_GraphicDesigner.
- Análisis de las metodologías de desarrollo del software que permitan cumplir con los requerimientos del sistema propuesto con la mayor documentación posible.
- Definición de los requerimientos que debe cumplir la herramienta de diseño SLAM_GraphicDesigner.
- Estudio de las principales tendencias arquitectónicas en el desarrollo de herramientas de diseño.
- Definición del patrón de arquitectura a utilizar en la construcción de SLAM_GraphicDesigner.
- Análisis y definición de las tecnologías con las que se desarrollará la herramienta.
- Documentación de los resultados de los estudios realizados.

Partiendo de lo planteado anteriormente la **idea a defender** sería: *e*l diseño de la herramienta SLAM_GraphicDesigner contribuirá a la generación de contenidos interactivos basados en la metáfora cine para el *framework* SLAM-C++.

A lo largo de esta investigación se utilizarán un conjunto de métodos científicos que servirán de guía y facilitarán un mejor entendimiento del problema.

Métodos empíricos:

Encuestas y entrevistas: se entrevistarán a varios expertos vinculados con el tema de la arquitectura de la información en la UCI, para recopilar toda la información que estos pudieran suministrar y de esta forma validar la propuesta de solución.

Métodos teóricos:

Histórico lógico: se realizará un estudio de la trayectoria hasta la actualidad de las herramientas de diseño en el mundo, en el país y en la UCI, para conocer con mayor profundidad los antecedentes y las tendencias actuales. Esto dará un preámbulo de lo que está sucediendo y un punto de partida para proponer posibles soluciones.

Analítico–sintético: Se llevará a cabo un análisis de las teorías, documentos, entre otros, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Estructuración del contenido por capítulos:

Capítulo 1. Fundamentación teórica: en este capítulo se abordan los elementos conceptuales que se tratarán a lo largo del documento. Se realiza un estudio de las herramientas y lenguajes que se utilizarán para la construcción del sistema propuesto, así como las razones por las que se decidió desarrollar el mismo, incluyendo el análisis de la situación problémica así como la descripción del objeto de estudio.

Capítulo 2. Descripción de la solución propuesta: en este capítulo se describe el funcionamiento del sistema. Se muestran los requisitos funcionales y no funcionales del sistema, así como los modelos conceptuales que describen el proceso, incluyendo aspectos importantes de la arquitectura de la información.

Capítulo 3. Construcción de la solución propuesta: este capítulo aborda los temas relacionados con el análisis y diseño de la herramienta, donde se exponen los modelos y vistas que facilitan su comprensión desde el punto de vista arquitectónico. También se propone el estándar de proyecto que será generado por la herramienta.

Capítulo 1: Fundamentación teórica

1.1 Introducción

Este capítulo comprende las terminologías relacionadas con un producto multimedia, marco de trabajo y metáfora cine. Se exponen aquellos conceptos que permitan un mejor entendimiento del tema de la investigación, realizándose un análisis de los conceptos asociados al dominio del problema, así como la descripción del objeto de estudio, que permitirán tener una visión general del trabajo.

1.2 Conceptos asociados al dominio del problema

1.2.1 *Framework* (Marco de trabajo)

Es una abstracción de una máquina/artefacto para resolver un problema complejo en un contexto de desarrollo. Además es un conjunto de herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista. Son diseñados con el objetivo de facilitar la evolución del software y acelerar el proceso de desarrollo de un proyecto **[3]**.

1.2.2 Herramientas de autor

Las herramientas de autor también denominados entornos de autor son aplicaciones informáticas que permiten elaborar sistemas multimedia. Ofrecen un entorno de trabajo que permite una programación basada en íconos, objetos y menús de opciones, que posibilitan al usuario realizar disímiles productos sin necesidad de escribir una sola línea en un lenguaje de programación. Dentro de las funcionalidades que este tipo de herramientas posee se pueden destacar la posibilidad de crear actividades o pequeñas aplicaciones desde la misma herramienta. **[4]**.

1.2.3 Metáfora cine

Es un paradigma de sistema de autor que le permite a los desarrolladores construir aplicaciones informáticas como si fuera una película, definida por una secuencia de escenas en el tiempo que pueden o no tener comportamientos asociados. Cada escena contiene un conjunto de actores que definen la representación de los recursos multimedia como son las imágenes, videos, menús, botones, componentes visuales, entre otros componentes que se aprecian en la cinematografía, de igual forma se podrán integrar elementos gráficos, conjuntamente con sus transformaciones realizadas sobre el escenario para producir transiciones y efectos especiales. Cuando una secuencia de fotogramas es visualizada con una determinada frecuencia de imágenes por segundo se logra generar la sensación de movimiento en el espectador y se le denomina metáfora cine **[5]**.

1.2.4 Herramientas de diseño

Las herramientas de diseño son programas creados con el objetivo de diseñar y confeccionar proyectos creativos con uno o más propósitos, las mismas están encaminadas a idear un procedimiento útil, funcional y estético en la creación de productos multimedia. Son aplicaciones en forma de estudio de animación, que integran medios digitales debido a su versatilidad. Integran un taller de pintura y fotografía que brindan la oportunidad de edición y retoque de imágenes para multimedia. Programas con estas características son utilizados en el sector de la creación artística, constituyendo una innovadora alternativa para el desarrollo de las técnicas digitales. Algunas de estas son *Adobe Flash, Adobe Photoshop, Director,* y otras **[6]**.

1.2.5 Estándar

El término estándar de origen inglés en tecnología es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. Un estándar es una guía técnica que se define y no varía, es utilizado para desarrollar hardware y software de tal forma que cualquier desviación produce problemas de compatibilidad. Este es el resultado de un proceso basado en especificaciones elaboradas por un determinado grupo de trabajo después de un estudio profundo de los métodos, aproximaciones y tendencias a desarrollos tecnológicos existentes. **[7]**

1.3 Objeto de estudio

1.3.1 Descripción general

El desarrollo de aplicaciones multimedia se ha convertido en un auténtico trabajo donde el diseño gráfico, la interactividad con el usuario y la realidad virtual son cada día más utilizados. Actualmente existen herramientas de diseño para el desarrollo de estas, sin embargo, la mayoría se ejecutan sobre la plataforma Windows, significando una desventaja para los desarrolladores de esos productos debido a las restricciones planteadas por los representantes de las compañías y los altos costos que hay que pagar para adquirir las licencias.

En sus inicios la mayoría de las herramientas de diseño estaban representadas por la empresa de software gráfico Macromedia, que fue creada por la fusión de *Authorware, Inc*. y *MacroMind-Paracomp*, siendo una gran compañía que se dedicaba al desarrollo de estos productos, que han tenido gran aceptación por todos los usuarios. En el año 2005 *Adobe Systems Incorporated* anunció un acuerdo para adquirir Macromedia ampliando su portafolio con aplicaciones y tecnologías que se complementaban con las que ya poseía.

A medida que ha ido pasando el tiempo han tomando una serie de características que las han hecho muy importantes, por tal motivo su utilidad ha sido necesaria para la confección de productos multimedia obteniendo resultados satisfactorios, por su versatilidad al poder integrar imágenes, audio, video digital, películas *flash*, entre otras, en una sola aplicación. Fueron diseñadas para administrar los elementos de multimedia individualmente. Ofrecen facilidades para crear y editar texto y tienen extensiones para controlar los reproductores de video disco, y otros periféricos relacionados.

Otro aspecto a destacar es su profesionalidad en el retoque fotográfico, en los dibujos vectoriales, maquetación, edición de video, contendido web, entre otras, que mejoran cada vez más en sus nuevas versiones al salir al mercado. Algunas se han perfeccionado de forma vertiginosa proporcionando una exitosa fusión entre la pintura, el retoque de color, y los efectos creativos.

Existen algunas que se ejecutan sobre plataformas libres que no cuentan con mucha aceptación por parte de los usuarios, pero no dejan de ser importantes. Permiten la edición de imágenes digitales en forma de mapa de bits, y de dibujos como fotografías, además del modelado, animación y creación de gráficos tridimensionales; se utilizan además para la autoedición. La principal característica que presentan es la consistencia de su interfaz y que algunas son multiplataforma.

Cada una tiene definido un estándar para la generación de contenidos que los define la Organización Internacional de Normalización (ISO), ya sea de video, o de sonido, que han sido determinados por ISO/IEC 14496-12, ISO/IEC 14496-10, ISO/IEC 14496-3. La relevancia que han adquirido estos estándares dirigidos a la creación de recursos dentro de la acción del trabajo con herramientas de diseño es el aspecto fundamental para poder realizar trabajos con estas.

Teniendo en cuenta todos estos aspectos mencionados surge la idea de desarrollar una herramienta de diseño que permita realizar todas estas acciones y otras nuevas, brindándole la posibilidad al usuario de desarrollar multimedia de forma sencilla.

1.4 Soluciones existentes para el desarrollo de multimedia

Actualmente el desarrollo de herramientas libres para la construcción de aplicaciones multimedia no es lo suficientemente consistente como para saciar las necesidades que estos productos implican. La comunidad de software libre ha puesto todo su empeño en desarrollar dichas herramientas, aunque todavía no se cuenta con sistemas a la altura de *Adobe*. Existen varios proyectos muy prometedores y ya se han empezado a ver los resultados solo que estos permiten diseños específicos.

Son pocas las soluciones que se podrían encontrar hasta el momento debido al escaso desarrollo de aplicaciones dentro de la naciente industria del software en el país. Debido a esto se realizó un estudio

exploratorio basado en un grupo de propuestas desarrolladas por equipos independientes en el mundo. Las candidatas a un análisis riguroso fueron.

1.4.1 *Adobe Director*

Adobe Director es una herramienta de autor utilizada para desarrollar multimedia, considerada una de las más poderosas para integrar y programar medios digitales, debido a su facilidad de poder incorporar imágenes en 3 dimensiones, audio, video digital, películas *flash* entre otros, en una sola aplicación, utiliza como lenguaje de programación Lingo y *JavaScript*.

Director soporta formatos, vectores y capacidades extensivas de video, que permiten a los desarrolladores ofrecer una amplia gama de contenidos multimedia. Ofrece además una interfaz mejorada lista para usar fragmentos de código y así de este modo acortar el tiempo de desarrollo de los productos finales.

Entre los usos de la herramienta se pueden mencionar el empleo de modelos en 3 dimensiones, a través de *plugin* para navegadores web que permiten la reproducción de contenidos interactivos, instalados en un 50% de los navegadores y que generan ficheros con extensión DCR desarrollados con la herramienta **[6]**.

1.4.2 *Adobe Flash*

Adobe Flash es un sistema de autor en forma de estudio de animación que trabaja sobre fotogramas, destinado a la producción de multimedia, utiliza gráficos vectoriales e imágenes ráster, sonido, código de programa, flujo de video, audio bidireccional, entre otros y usa el lenguaje de programación *ActionScript* utilizado especialmente en aplicaciones web animadas. Puede exportar las extensiones JPEG, GIF, PNG, PICT en el sistema operativo Macintosh y Windows **[8]**.

Esta herramienta ofrece aplicaciones, contenido y videos de gran atractivo, una interfaz de usuario agilizada, herramientas de video avanzada y es utilizada para crear animaciones 2 dimensiones.

Entre las principales aplicaciones que puede realizar *Flash* se encuentran:

- Incluir fácilmente en las películas archivos de sonido con la tecnología MP3 y comprimirlos conjuntamente con fotogramas.
- Desarrollar proyectores ejecutables que sean independientes, para que las películas se puedan ver sin necesidad de un navegador, o sea, desde un CD-ROM.
- Producir cambios en vivo dentro de los elementos de la película.

1.5 Herramientas de diseño existentes en Linux

1.5.1 Gimp

Programa de manipulación de imágenes GNU (Gimp), es un programa gratuito y robusto para pintar, procesar y manipular gráficos e imágenes. Disponible para varios sistemas operativos e idiomas. El Gimp es apropiado para tareas como retoque fotográfico, composición y edición de imágenes. Es especialmente útil para la creación de logotipos y otros gráficos para páginas web.

Puede utilizarse como un simple programa de dibujo, un programa experto de retoque fotográfico, o como un convertidor de formatos de imagen como TIFF, GIF, JPEG, BMP, PPM, entre otros. La potencia de Gimp es similar a la de *Photoshop* debido a que es una alternativa firme y rápida de este, con la diferencia de que es software libre.

Esta herramienta de diseño permite la automatización de muchos procesos mediante macros o secuencias de comandos. Para ello incluye el lenguaje *Scheme* y otros como Perl, Python, Tcl entre otros utilizados para este propósito.

Uno de los inconvenientes del Gimp es la falta de soporte nativo para *Pantone* y *Cmyk*, que son los modelos de colores utilizados por las imprentas. Estos sistemas son los más utilizados en la industria y se basan en una muestra o gama de colores, a cada color le corresponde un único número. **[9]**.

1.5.2 Inkscape

Inkscape es una herramienta para el trabajo con dibujo y para la edición de gráficos vectoriales SVG, es multiplataforma y gratuito. Su objetivo principal es crear una herramienta de dibujo muy completa y cómoda, totalmente compatible con los estándares XML, SVG y CSS. En la mayoría de los casos se utiliza junto a un programa de edición de imágenes como Gimp.

Esta herramienta puede importar formatos como EPS, JPEG, PNG, TIFF, entre otros, basados en vectores, es similar a *Adobe Illustrator*, *Corel Draw, Freehand, o Xara X*. Lo que diferencia a *Inkscape* de estas herramientas es que utiliza *Scalable Vector Graphics* (SVG), un estándar abierto del *W3C* basado en XML como formato nativo.

Con Inkscape se puede realizar edición de nodos, capas, operaciones complejas con trazos, vectorización de archivos gráficos, texto, alineación, edición de XML directo y otros. Soporta características que incluyen formas, marcadores, clones, mezclas de canales alfa, transformaciones, gradientes, patrones y agrupamientos **[10]**.

1.5.3 Ktoon

Ktoon es una aplicación gráfica y libre, se usa para hacer la labor de dibujo y es muy eficaz en la animación debido a que es una alternativa pública a *Adobe Flash*, sin llegar a considerarlo un clon

aunque tiene muchas de sus opciones. Está orientada a la producción de dibujos animados basada en la animación tradicional 2 dimensiones.

La herramienta puede exportar archivos en formato SWF, además posee facilidad de uso y calidad de sus animaciones y aunque su orientación tiende más al desarrollo multimedia no se puede considerar como una herramienta para el desarrollo de multimedia **[11]**.

1.5.4 F4L

Flash para Linux (F4L) fue una aplicación informática que se desarrolló como una propuesta para software libre de *Adobe Flash*, pero no llegó a su culminación, incluye los contenidos SWF, SVG y ofrecía una interfaz de usuario similar a la de *Flash MX*, permitió el dibujo y animación de elementos básicos. Fue pausado porque aunque era muy prometedor estaba lejos de poder soportar la creación de fotogramas claves. **[12]**.

1.5.5 *Qflash*

Qflash fue una propuesta para realizar un clon de *Adobe Flash* para Linux, esto se logró principalmente en la parte de la interfaz, que es en gran medida muy parecida a la de *Flash*, pero aún así carece de muchas de sus funcionalidades. Entre los principales logros de este proyecto se destacan el dibujo de formas básicas, creación de animaciones fotograma a fotograma y el trabajo con capas. Este se unió con F4L para crear un programa conjunto llamado Uira.

1.5.6 Uira

Uira fue concebida como una alternativa de *Flash* en Linux. En sus inicios se propuso combinar los recursos y el conocimiento del proyecto *Qflash* y F4L con intención de unir las experiencias contenidas en ambas, con el fin de proporcionar una solución completa para la creación de contenido multimedia interactivo para plataformas libres como GNU/Linux. Esta herramienta estaba encaminada a ser un poco más que *Adobe Flash*, pues no solo se limitaría a crear ficheros SWF, sino que permitiría generar binarios en formato SVG. Permitiendo construir imágenes bidimensionales totalmente libres utilizando XML proporcionándole un gran impulso al formato SVG frente al SWF. El proyecto no se pudo terminar, en gran parte debido a la ley DADVSI en Francia, fue interrumpido completamente en enero 2008 **[13]**.

1.6 Soluciones existentes en Cuba

1.6.1 HAEduc

HAEduc es una herramienta de autor libre y a su vez multiplataforma para el desarrollo de multimedia. Está concebida para la gestión de aplicaciones de escritorio. Esta emplea un lenguaje *script* similar al

Basic, proveniente de *wxBasic* con las modificaciones necesarias para su adaptación a la metáfora libro, establecida por los autores. Los formatos de imágenes que soporta son JPEG, PNG, GIF, BMP, entre otros, también así los de video MPEG, AVI, dependiendo de los códec instalados.

El entorno de desarrollo integrado (IDE) con el que trabaja el usuario contiene las herramientas necesarias para el desarrollo de los proyectos educativos planificados, al contar con un árbol de navegación por el libro, página de propiedades de los objetos de usuario, editor de variables por ámbito, editor de *script*, visualizador de errores de sintaxis y ejecución, paleta de objetos de usuario, entre otros. Para el diseño del software el usuario cuenta con once objetos multimedia imagen, video, reloj, etiqueta, texto, caja de texto, polígono, tabla, botón, lista y contenedor HTML **[14].**

1.6.2 Picasso.

Picasso es una herramienta para el diseño multimedia actualmente en desarrollo en la Universidad, que tiene amplias perspectivas para crear productos multimedia, con características similares a las herramientas que existen en Linux para el diseño. Soporta formatos de imagen, audio, video, y el fichero SWF, esta herramienta fue creada para que todos los usuarios pudieran tener acceso a ella debido a que se ejecuta sobre la plataforma Linux. Comenzó su desarrollo con amplias expectativas de trabajo pero debido a razones ajenas de los desarrolladores actualmente se encuentra en pausa.

1.7 Justificación de la solución propuesta

Después de realizar un análisis detallado de las soluciones existentes, para la construcción de aplicaciones multimedia que se ejecuten sobre plataformas como GNU/Linux, se puede decir que no es factible tomar ninguna de las herramientas de diseño existentes por las siguientes razones:

Todas las opciones que brinda *Adobe* para la construcción de multimedia son buenas porque le permiten a los usuarios innovar en el mundo del diseño, pero estas tienen muchas limitantes entre las que se destacan los altos costos que hay que pagar para adquirir las licencias siendo esto una solución no rentable para el país.

Qflash, F4L y *Uira* son herramientas que se comenzaron a desarrollar con grandes expectativas pues tenían previsto realizar funciones similares al *Flash* pero en Linux. Debido a que no llegaron a su final, hoy en día no son muy conocidas ni utilizadas. En el caso de Inkscape y Gimp son herramientas conocidas, pero realizan diseños específicos por separado, por tal motivo no se pueden considerar como una alternativa para desarrollar multimedia. Además, no proveen mecanismos de comunicación inter procesos (IPC) que permitan embeberlas dentro de otra aplicación lo que justifica la imposibilidad de usarlas como un anexo a SLAM-C++ siendo esto un inconveniente para su uso.

Otra expectativa muy relevante es el caso de la herramienta HAEduc que la misma fue desarrollada en Cuba y satisface en gran medida a las necesidades económicas del país, pero a pesar de que el proyecto es una alternativa para desarrollar multimedia, no se ajusta a las necesidades del marco de trabajo porque no da soporte para JavaScript siendo esta una proyección de SLAM-C++. Además esta herramienta trabaja con la metáfora libro y eso es uno de los principios más importantes que debe cumplir la herramienta propuesta.

Por todas las razones anteriormente expresadas se decidió realizar la herramienta SLAM-GraphicDesigner desde cero con una interfaz amigable, incorporándole nuevas funcionalidades como el uso del mapeo de objetos-relacional (ORM) para el mapeo de las tablas de la base de datos y la arquitectura basada en plugin, que posibilitará agregar muchas funcionalidades, tantas como desee el usuario sin necesidad de modificar las versiones ya existentes de la herramienta; además permitirá trabajar con la misma sin tener amplios conocimientos de programación, de forma tal que con pocos pasos se pueda crear recursos multimedia relativamente complejos y avanzados en cuanto a interactividad y funcionalidades.

1.8 Tecnologías, metodologías y herramientas

Las especializaciones del software están impulsando cada vez más el desarrollo de las tecnologías, y metodologías que se utilizan para crear herramientas de diferentes tipos. Existen herramientas que permiten la realización de grandes sistemas en tiempos breves y con gran calidad debido a las facilidades que implementan.

Para desarrollar un software se deben tener en cuenta ciertos parámetros como metodologías a usar, lenguajes de programación, librerías multiplataforma, IDE, entre otros. Como consecuencia a lo planteado a continuación se presenta el resumen realizado durante la investigación, y la propuesta tecnológica realizada.

1.8.1 Lenguaje de programación

Un lenguaje de programación es un idioma diseñado para expresar instrucciones que son llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina para expresar algoritmos con precisión. También esta palabra se define como el proceso de creación de un programa mediante la aplicación de procedimientos lógicos.

El lenguaje de programación tiene tres clasificaciones entre las que se encuentran:

- Lenguaje de bajo nivel: generación de código máquina y lenguaje ensamblador.
- Lenguaje de medio nivel: *ActionScript*, *JavaScript*, ASP, PHP, *Perl, Python, Ruby*.
- Lenguaje de alto nivel: C++, *Java*, C#, *Python*.

La Programación Orientada a Objetos es uno de los paradigmas más modernos usados en el desarrollo de software, enfocado en disminuir el coste y aumentar la eficiencia. Esto ha inducido a un aumento considerable en la cantidad de lenguajes de programación que la soportan, entre los que se pueden mencionar el C++, *Python*, PHP, *ActionScript,* entre otros. La orientación a objetos es especialmente adecuada para realizar una amplia gama de aplicaciones, sobre todo realización de prototipos y la simulación de programas.

El lenguaje seleccionado para el desarrollo de este trabajo es C++ debido a que es uno de los lenguajes más cercanos al código de máquina, este es capaz de proveer los recursos de los lenguajes de alto nivel, como la programación orientada a objeto. C++ es el que más se ajusta a las necesidades de la investigación**[15]**.

Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones. Además, se trata de un lenguaje de programación estandarizado, ampliamente difundido, y con una biblioteca estándar de C++ que lo ha convertido en un lenguaje universal, de propósito general, y ampliamente utilizado tanto en el ámbito profesional como en el educativo. Actualmente se han desarrollado un gran número de librerías estándar que facilitan el trabajo con estructuras de datos de alta complejidad, manipulación de cadenas, así como interfaz gráfica de usuario por citarse algunos ejemplos.

Con lo expresado anteriormente no se quiere decir que se desechan los lenguajes de mayor grado de abstracción, sino que se valora realmente desde un punto de vista ingenieril el trabajo de los desarrolladores y se identifica el potencial que C++ ofrece. Al seleccionar este lenguaje se piensa en desarrollar una aplicación lo más robusta y eficiente posible siguiendo las especificaciones del marco de trabajo SLAM-C++ **[16]**.

1.8.2 Librería multiplataforma

Las librerías multiplataforma son utilizadas para dar solución a un determinado problema, son una lista de instrucciones perfectamente definidas, ordenadas y de carácter finito. Brindan ayuda a diferentes

programas independientes admitiendo que el código y los datos se compartan, para que de esta forma puedan transformarse modularmente.

Basándose en los objetivos de esta investigación de desarrollar una herramienta que conste de una interfaz gráfica de usuario, se debe tener presente la importancia de un paquete de librerías multiplataforma que no solo sea independiente de la plataforma de hardware, sino que también lo sea del sistema operativo, que provea los elementos necesarios para el manejo de los recursos que brindan incluyendo manipulación de audio y video.

Existen muchos marcos de trabajo multiplataforma utilizados en la construcción de aplicaciones para interfaces de usuarios ejemplos de ellos son:

- GTK+
- *wxWidgets*
- Qt

Todas estas son muy importantes, pero para el desarrollo de la herramienta SLAM-GraphicDesigner se optó por Qt, que es un marco de trabajo para el desarrollo de aplicaciones multiplataforma, que contiene una biblioteca de clases C++ creada para construir interfaces de usuario, tiene una interfaz de programación totalmente orientada a objetos.

Las aplicaciones basadas en Qt tienen una buena respuesta y un uso aceptable de la memoria. El desarrollo con este marco de trabajo es muy apropiado para proyectos software de larga escala, tanto comerciales como de libre distribución. Permite escribir código que se compilará y ejecutará en distintas plataformas, incluyendo Unix, Linux, FreeBSD o incluso Windows.
Características:

- Es una librería que se basa en los conceptos de *widgets* (objetos), señales-*slots* y eventos.
- Las señales y los *slots* es el mecanismo para que unos *widgets* se comuniquen con otros.
- Algunos atributos como el texto de etiquetas y otros se modifican de modo similar al lenguaje HTML.

Qt proporciona además otras funcionalidades:

- Librerías básicas para entrada/salida, manejo de red, XML
- Interfaces con bases de datos como Oracle, MySQL, PostgreSQL, ODBC.
- Plugin, librerías dinámicas, imágenes, formatos.

- Compatibilidad multiplataforma con un sólo código fuente.
- *Performance* de C++.
- Disponibilidad del código fuente.
- Excelente documentación.

Qt funciona en las principales plataformas. Las API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red para el manejo de ficheros, y estructuras de datos tradicionales. Además, para su mejoramiento lanzó *QtCreator*, una de las herramientas más potentes que la componen, debido a que es un IDE para el desarrollo de aplicaciones con C++ basado en componentes de Qt.

1.8.3 Metáfora de trabajo

En sentido general los paradigmas representan un conjunto de conocimientos, con el fin de conformar una visión global que facilita la resolución de un determinado problema en determinadas circunstancias, fundamentados en experiencias personales.

En el desarrollo de aplicaciones multimedia las metáforas de trabajo representan un papel fundamental ya que permiten determinar las direcciones en las que se desarrollarán por medio de la propuesta de enigmas. Las herramientas de desarrollo para multimedia basada en línea de tiempo gozan de mayor aceptación en el mercado. Cada una utiliza su propio enfoque e interfaces gráficas de usuario para administrar eventos en el tiempo y por lo general emplean una línea de tiempo visual para representar la secuencia de objetos de una aplicación multimedia.

Aunque el objetivo de este trabajo no sea construir una herramienta de autor, si se debe proveer un paradigma de trabajo para la construcción de aplicaciones multimedia, teniendo en cuenta los aspectos que se han ido abordando a lo largo de este capítulo se decidió seleccionar la metáfora cine, siendo la más apropiada para SLAM-GraphicDesigner, de forma tal que los desarrolladores acostumbrados a trabajar en herramientas como *Adobe Flash* y *Adobe Director* no noten un cambio radical.

1.8.4 Metodologías

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente que se ajuste al equipo de desarrollo, guiando y organizando las actividades que conlleven a las metas trazadas. Hoy en día existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso. Además han

demostrado ser efectivas y necesarias en un gran número de proyectos, sobre todo aquellos proyectos de gran tamaño respecto a tiempo y recursos.
Sin embargo, la experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución para proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud porque no están pensadas para trabajar con incertidumbre.

Entre las metodologías tradicionales se pueden citar:

- Desarrollo de Sistemas de Jackson (JSD)
- Información metodológica de la ingeniería
- Sistema de Análisis Estructurado y Método de Diseño.
- Métrica promovida por el Ministerio de las Administraciones Públicas.
- Proceso Unificado del Software (RUP).

También surgieron otras metodologías que trataban de adaptarse a la realidad del desarrollo de software, estas son las metodologías ágiles que cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientadas al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien orientados al código siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

Las metodologías más utilizadas dentro de las ágiles se encuentran:

- Programación extrema.
- *Scrum.*
- *Feature Driven Development* (FDD).
- Desarrollo de software adaptable.
- Procesos ágil unificado (AUP) y proceso esencial unificado (EUP).

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1: Diferencias entre metodologías ágiles y tradicionales.

Como se mencionaba anteriormente existen varias metodologías muy importantes y utilizadas para el desarrollo del software, por ejemplo:

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Divide en 4 faces el desarrollo del software:

- **Inicio**: el objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: el objetivo es determinar la arquitectura óptima.
- **Construcción**: el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transición**: el objetivo es llegar a obtener el *release* del proyecto **[17]**.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

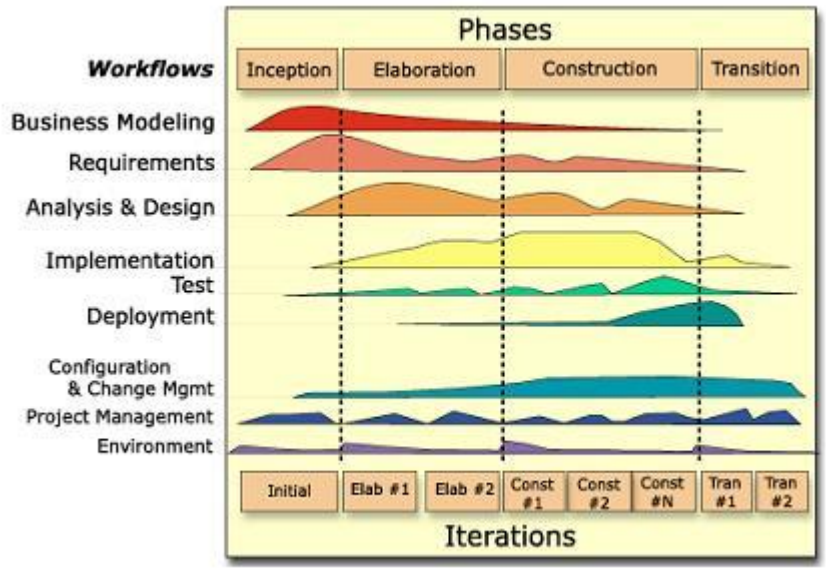


Figura 1: RUP en dos dimensiones

Los Flujos de trabajo que propone RUP:

• **Modelamiento del negocio:** describe los procesos del negocio, identificando quiénes participan y las actividades que requieren automatización.

• **Requerimientos:** define qué es lo que el sistema debe hacer identificando las funcionalidades requeridas y las restricciones que se imponen.

• **Análisis y diseño:** describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), que indica con precisión lo que se debe programar.

• **Implementación:** define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la estructura de capas de la aplicación.

• **Prueba (Testeo):** busca los defectos a lo largo del ciclo de vida.

• **Instalación:** produce el *release* del producto y realiza actividades de empaque, instalación, asistencia a usuarios, entre otras, para entregar el software a los usuarios finales.

• **Administración del proyecto:** involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

• **Administración de configuración y cambios:** describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a utilización/actualización concurrente de elementos, control de versiones, entre otros.

• **Ambiente:** contiene actividades que describen los procesos y herramientas que soportará el equipo de trabajo del proyecto, así como el procedimiento para implementar el proceso en una organización.
Los elementos de RUP son:

- **Actividades**: son los procesos que se llegan a determinar en cada iteración.
- **Trabajadores**: son las personas o entes involucrados en cada proceso.
- **Artefactos**: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

El ciclo de vida de RUP se caracteriza por:
• **Dirigido por casos de uso:** los casos de uso reflejan la necesidad de los usuarios, captándose esta cuando se modela el negocio representándose a través de los requerimientos.

• **Centrado en la arquitectura:** la arquitectura muestra la visión común del sistema completo en la que el equipo del proyecto y los usuarios deben estar de acuerdo, por esto se describen los elementos del modelo que son más importantes para su construcción y los cimientos del sistema que son necesarios como base para comprenderlo.

• **Iterativo e incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros **[18]**.
Por todo lo expresado anteriormente se escoge RUP como metodología, por la eficacia con que guía el proceso de desarrollo de software, convirtiéndose en la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El mismo está diseñado para adaptarse a cualquier tipo de proyecto por muy difícil y complejo que sea. RUP desde sus inicios cuenta con una documentación profunda y detallada de todo el proceso en sentido general que permite tener un mayor control y seguimiento del proceso de desarrollo debido a que es centrado en la arquitectura. Se opta por esta metodología debido a que está basada en componentes e interfaces

bien definidas, además está basada en un conjunto de actividades necesarias para convertir los requisitos del usuario en un software. Teniendo en cuenta que el desarrollo de este trabajo aborda temas de arquitectura de la información se consideró factible esta metodología.

1.8.5 Lenguaje de modelado

Lenguaje de modelado es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML es un lenguaje de modelado que se utiliza para definir un sistema, detallar los artefactos del mismo, documentar y construir.

UML centra el desarrollo en tres modelos diferentes:

• **Modelo funcional**: diagramas de casos de uso, describen el sistema desde la perspectiva del usuario.

• **Modelo objeto**: diagrama de clases, describen la estructura de un sistema partiendo de objetos, atributos, asociaciones y operaciones.

• **Modelo dinámico**: diagramas de secuencia y de estados, describen el comportamiento del sistema **[19]**.

RUP propone UML como lenguaje de modelado para representar todos los esquemas de un sistema de software. Partiendo de un enfoque sistemático permite construir estos modelos de una forma consistente demostrando su utilidad en sistemas de gran envergadura como la solución propuesta, se puede afirmar que RUP y UML son la combinación perfecta para desarrollar SLAM-GraphicDesigner.

1.8.6 Herramientas

1.8.6.1 CASE

Conjunto de métodos y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas, completamente o en alguna de sus fases, brindan una ayuda que da asistencia a los analistas, ingenieros de software y desarrolladores.

Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación del código

automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras. En la actualidad existen un gran número de herramientas CASE, tal es el caso de *Unbrello*, BOUML, ArgoUML, Eclipse UML, *Rational Rose* y *Visual Paradigm*.

Visual Paradigm es una herramienta muy poderosa y profesional que soporta el ciclo de vida del desarrollo de un software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de mejores aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.
Facilita a las organizaciones y el diagrama de diseño visual, integrar y desplegar sus aplicaciones empresariales de misión crítica y sus bases de datos. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios realizados por sus compañeros.

Entre sus principales características están las siguientes:

- Visualiza, comprende y mejora sus procesos de negocio con la más completa herramienta de modelado de procesos de negocio *Business Process Modeling Notation* (BPMN).
- Posee una excelente interoperabilidad en una única plataforma de desarrollo.
- Genera la documentación del proyecto en varios formatos como HTML, MS Word y PDF.
- Ingeniería inversa *Java*, C++.
- Generador de informes para la generación de documentos.
- Distribución automática de diagramas **[20]**.

Se utilizará la herramienta case *Visual Paradigm*, porque además de cumplir perfectamente con todos los estándares UML, la Universidad pagó por su licencia y tiene como política que sea la herramienta case a utilizar.

1.8.6.2 IDE

Un entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o varios.
Proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación como C++, *Python*, *Java*, C#, *Delphi*, *Visual Basic*, entre otros. En algunos lenguajes puede funcionar como un

sistema en tiempo de ejecución, donde se permite utilizar el lenguaje de programación en forma interactiva sin necesidad de trabajo orientado a archivos de texto, como es el caso de *Smalltalk* u Objetivo-C.

Existen varios IDE que soportan el lenguaje C++ tal es el caso de:

- *Kdevelop*
- Código en bloque
- Dev C++
- *QtCreator*

Por sus características el que más se ajusta para el desarrollo de SLAM-GraphicDesigner es:
QtCreator es un excelente IDE multiplataforma para desarrollar aplicaciones en C++ de manera sencilla y rápida. Como su nombre lo indica, está basado en la librería Qt y cuenta con las siguientes características:

- Editor avanzado para C++.
- Diseñador de formulario con interfaz gráfica de usuario (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completador automático.
- Depurador visual.

El objetivo de este IDE no es reemplazar a *Eclipse* ni a *Visual Studio*, sino convertirse en un IDE ligero pensado especialmente para el desarrollar aplicaciones en múltiples plataformas como:

Windows XP, Vista, Linux y Mac OSX. Actualmente soporta el desarrollo de aplicaciones de escritorio y además incluye las funcionalidades necesarias para programar dispositivos móviles y embebidos. Proporciona una GUI con múltiples elementos para insertar y confeccionar la interfaz del programa. Se basa en la idea de proveer a los usuarios de Qt las funcionalidades requeridas para ponerse en marcha, ya sean principiantes en Qt o desarrolladores avanzados. Este IDE ofrece varias funcionalidades únicas para que la creación de Qt resulte fácil y satisfactoria **[21]**.

Posee un revisor de código estático y suministra sugerencias de estilo a medida que se tipea, *code folding* y edición avanzada. Se puede obtener con el lenguaje ayuda sensible al contexto, una combinación de paréntesis y modos de selección. *QtCreator* contiene herramientas de navegación que

permiten administrar el código de origen. Admite abrir archivos, incluso cuando el desarrollador no tiene conocimiento de dónde se encuentran ubicados o cuál es su nombre exacto. También cuenta con un diseñador de *layout* y de formas GUI integrado, un depurador visual, herramientas de administración y construcción de proyecto. Se puede obtener *QtCreator* combinando la última versión de las bibliotecas Qt, con herramientas adicionales **[22]**.

1.8.6.3 MindManager

Es un programa para la creación de mapas, permitiendo trabajar con más profundidad, pensar de forma creativa, ahorrar tiempo para la captura visual y gestión de la información. Es un excelente gestor de proyectos que permite tener organizadas todas las ideas, objetivos y opciones, además de tener una perspectiva general del trabajo sin olvidar ningún detalle. Su utilidad fundamental radica en las facilidades que ofrece para la construcción de esquemas de planteamiento y solución de problemas de información. Posee una intuitiva interfaz que permite empezar a usarlo, los mapas que genera tienen soporte para documentos y enlace.

1.9 Conclusiones

Luego de haber realizado un estudio de todas de las metodologías, herramientas y tecnologías que existen en el mundo para el diseño se arribó de manera breve a las siguientes conclusiones:

En el desarrollo de SLAM-GraphicDesigner se utilizarán:

- El lenguaje de programación C++ porque además de todas las características que tiene es el establecido por el *framework* para su desarrollo.
- RUP como metodología de desarrollo de software que regirá el proceso de construcción de la misma y UML como lenguaje de modelado.
- *Visual Paradigm* como herramientas CASE para la modelación del análisis y diseño de la herramienta.

Capítulo2: Descripción de la solución propuesta

2.1 Introducción

En este capítulo se realizará un análisis de la herramienta a desarrollar, se definirán las características que tendrá. Además se presenta la descripción del modelo de dominio, así como el diagrama de clases correspondiente al modelo de objetos, incluyendo la gestión de requerimientos a través de diferentes técnicas que existen para la captura, definición y validación de los mismos, pues estos constituyen la base para la realización del modelo de casos de uso.

2.2 Modelo de dominio

El proceso de modelar posibilita obtener una visión de la organización, permitiendo definir los procesos y roles, relacionado con la obtención de requerimientos. Debido al bajo nivel de estructuración que presenta el negocio que se está estudiando se decide desarrollar un modelo de dominio, porque este permite capturar los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema.

2.2.1 Descripción general

Aplicando los conceptos de la metáfora cine, un producto multimedia es considerado como una película interactiva, a su vez está constituida por una secuencia de escenas que pueden o no tener comportamientos asociados. Cada escena contiene un conjunto de actores que definen la representación de los recursos multimedia a los que se hacen referencia, así como los comportamientos asociados a los mismos. Para mayor organización del sistema los recursos multimedia son almacenados en la librería.

ORM es una técnica de programación que potencia la reutilización y abstracción a los diferentes Sistemas Gestores de Base de Datos (SGBD), permitiendo trabajar con las tablas de una base de datos como si fuese orientado a objeto. Crea una capa de abstracción a datos que posibilita tener un control estricto sobre los datos y tratarlos lógicamente orientado a objeto, admitiendo la implementación de la seguridad en la capa de acceso a datos.

2.2.2 Diagrama de clases

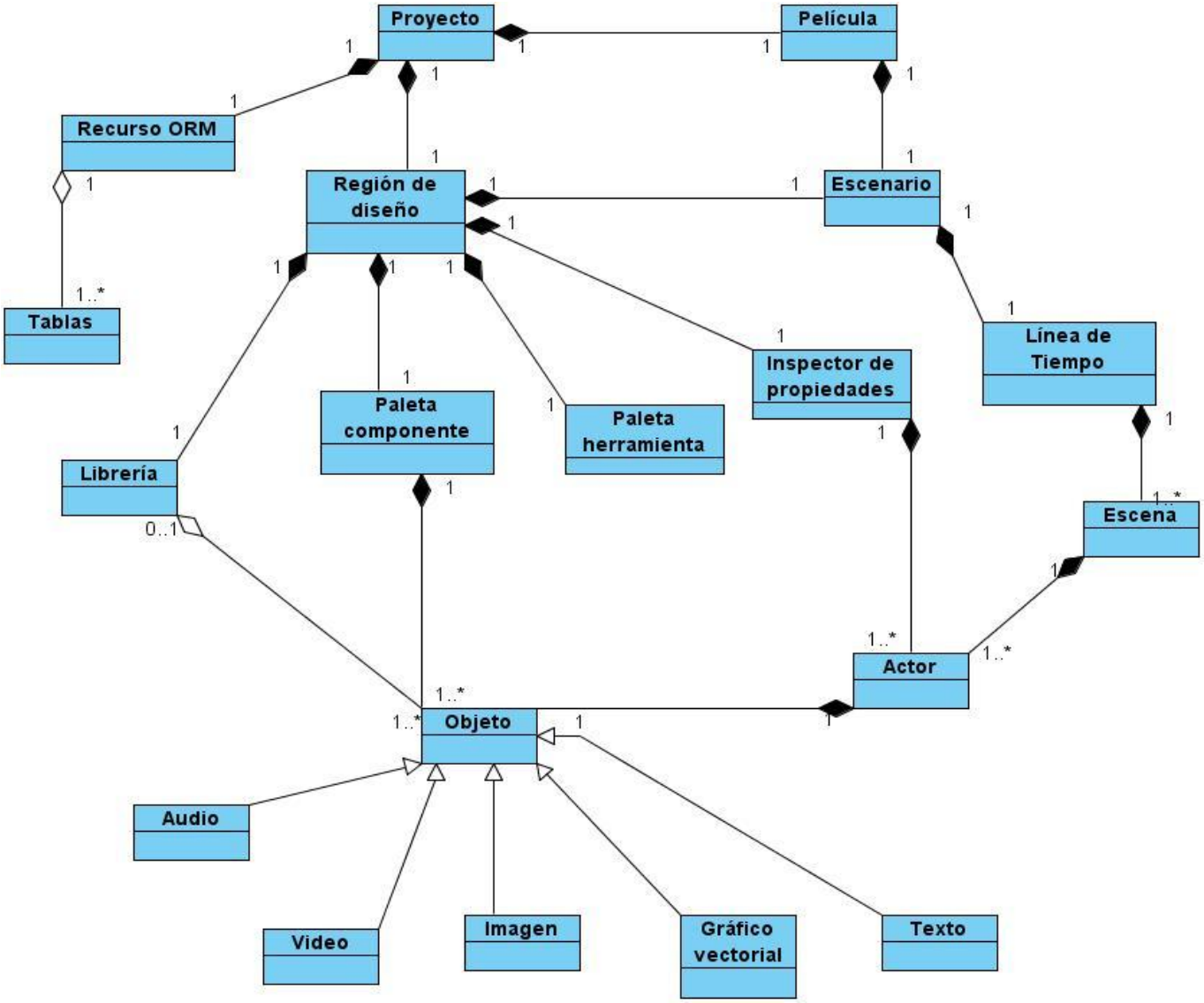


Figura 2: Diagrama de clases del dominio

2.2.3 Conceptos principales del dominio

Película: constituye el soporte de la animación, define las propiedades globales de las multimedia, controla la aparición de los componentes gráficos en la misma y controla los eventos asociados a cada una de las escenas **[23]**.

Escena: es la unidad mínima de una animación, que determina un comportamiento independiente en el tiempo y constituye una estructura organizativa donde se definen los elementos a representar en el escenario en un instante determinado.

Escenario: es el área de trabajo donde se compone o diagrama el contenido de cada una de las escenas, que constituyen la película durante su edición. Representa el espacio a ocupar por el proyector durante la reproducción del proyecto.
Línea de tiempo: organiza y controla el contenido de una película a través del tiempo, en capas y fotogramas. Sus elementos principales son las escenas y la cabeza lectora **[23]**.
Paleta de herramientas: recurso que permitirá acelerar las operaciones más comunes del menú principal como crear formas, además de seleccionarlas, moverlas, duplicarlas y modificarlas. En ella se podría además configurar los efectos de color para borde o relleno, así como texturas, gradiente, transparencia y otras propiedades con las que trabaja el escenario.
Paleta de componentes: contiene todos los componentes primarios que son útiles para el diseñador como *CheckBox, ListBox, RadioButton, ComboBox, PushBotton, ScrollBar y ScrollPane.*

Región de diseño: es el área donde se realizan las acciones de diseño, integra todas las paletas como inspector de propiedades, biblioteca, herramientas y componentes.

Actor: es un objeto dentro de un escenario, que tiene comportamientos asociados al mismo y está definido por un conjunto de propiedades.

Objeto: es el contenedor de un recurso multimedia. Llámese recurso multimedia a todos los elementos gráficos o no, como por ejemplo los mapas de bit, gráficos vectoriales, audio, videos, textos, entre otros.

Librería: es el contenedor de todos los objetos que intervienen en la multimedia.

Inspector de propiedades: es un panel donde se muestran las características de cada atributo y efecto de un actor previamente seleccionado en el escenario.

Tabla: es una representación matricial de aplicaciones donde se guardan los datos recogidos por un programa. Es considerada como una forma de representar mucha información de manera esquematizada, ordenada y compacta.

Proyecto: es una estructura organizativa, que almacena toda la información desarrollada con un objetivo y puede ser utilizada nuevamente después de haberse cerrado en un lugar determinado.

2.3 Descripción del sistema propuesto

El propósito de SLAM-GraphicDesigner es convertirse en una herramienta de diseño para el desarrollo aplicaciones multimedia a través de la metáfora cine, con la misma se podrán integrar elementos gráficos, conjuntamente con las transformaciones realizadas sobre el escenario para producir efectos especiales. Para ello se propondrá una interfaz similar al modelo conceptual del dominio, véase figura #2.

Las animaciones se basarán principalmente en la utilización de gráficos vectoriales, definiéndose para esto el formato SVG que no es más que un estándar abierto, basado en XML **[24]**, también será posible trabajar con imágenes en diferentes formatos. SLAM-GraphicDesigner proveerá mecanismos sencillos para crear efectos y animaciones complejas, de forma tal que se puedan asociar interpolaciones de movimiento, forma y transparencia a un determinado actor en el escenario. Para el trabajo con archivos de audio y video se utilizará un conjunto de interfaces que simplifica su utilización, apoderándose de las funcionalidades que brinda el reproductor *mplayer*, así como todos los formatos que este soporta.

El sistema proveerá mecanismos para el tratamiento de datos. Esto incluye trabajo con XML, así como acceso a bases de datos mediante un ORM que soporta los siguientes servidores *MySQL* y *PostgreSQL.* La herramienta tiene como propósito además ofrecerle al usuario un panel que brinda una serie de opciones para el diseño, permitiéndole trabajar de manera rápida y sencilla sobre el escenario, proporcionando además opciones de colores, figuras, botones, menús, entre otras.

Formatos soportados por la herramienta

- **Formatos de gráficos vectorial:**

SVG

- **Formatos de mapas de bit:**

JPEG/PNG/GIF/PGM/BMP/DIB

- **Formatos de audio:**

MP3/WAV/WMA/MP2

- **Formatos de video:**

MPEG/VOB/IFO/AVI/WMV/ASF/OGG/OGM/MKV/MP4/DAT/RM/RMVB.

2.4 Especificación de requerimientos de SLAM-GraphicDesigner

Los requerimientos funcionales describen las funcionalidades que el sistema debe ser capaz de cumplir. Establecen los comportamientos y describen las transformaciones que el mismo debe realizar para obtener los resultados deseados por los programadores. Por otra parte, los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. A continuación son detallados cada uno de estos:

2.4.1 Requerimientos funcionales

La herramienta debe proveer mecanismos para:

RF1 Importar archivos

RF2 Exportar Archivos

La herramienta debe proveer mecanismos para el trabajo con la librería:

RF3. Insertar un recurso multimedia.

RF4. Eliminar un recurso multimedia.

RF5. Insertar un recurso multimedia desde un directorio.

RF6. Eliminar un conjunto de recursos multimedia.

La herramienta debe proveer mecanismos para el trabajo con proyectos:

RF7. Crear nuevo proyecto.

RF8. Abrir un proyecto existente.

RF9. Guardar cambios.

RF10. Cargar más de un proyecto a la vez.

RF11. Guardar como.

RF12. Cerrar un proyecto.

RF13. Salir del proyecto.

La herramienta debe proveer mecanismos para el trabajo con el escenario:

RF14. Modificar las propiedades de los objetos.

RF15. Eliminar objetos.

RF16. Copiar objetos.

RF17. Cortar objetos.

RF18. Pegar objetos.

La herramienta debe proveer mecanismos para el trabajo en la región de diseño:

RF19. Crear figuras

RF20. Dibujar línea.

RF21. Dibujar poli línea.

RF22. Dibujar rectángulo.

RF23. Dibujar cuadrado.

RF24. Dibujar elipse.

RF25. Dibujar círculo.

RF26. Trazar con lápiz.

RF27. Rellenar figura.

RF28. Insertar texto.

RF29. Utilizar la selección libre.

RF30. Representar el contenido de la escena.

RF31. Realizar acciones de zoom.

RF32. Utilizar gradiente.

RF33. Utilizar transformación libre

RF34. Utilizar transformación de relleno.

RF35. Utilizar pluma.

RF36. Utilizar la goma.

RF37. Utilizar pincel.

La herramienta debe proveer mecanismos para el trabajo con la línea de tiempo:

RF38. Eliminar una escena.

RF39. Insertar una escena.

La herramienta debe proveer mecanismos para el trabajo con textos:

RF40. Visualizar archivos de textos.

RF41. Definir el tipo alineación.

RF42. Definir color del texto.

RF43. Definir estilo de letra.

RF44. Definir tamaño de letra.

RF45. Definir formato de letra.

RF46. Definir tipo de archivo texto.

RF47. Definir dimensiones de archivo de texto.

RF48. Definir la profundidad del archivo de texto.

La herramienta debe proveer mecanismos para el trabajo con gráfico vectorial:

RF49. Definir dimensiones de gráfico vectorial.

RF50. Definir profundidad de gráfico vectorial.

RF51. Visualizar archivos de gráfico vectorial.

RF52. Cargar gráfico vectorial.

La herramienta debe proveer mecanismos para el trabajo con imágenes:

RF53. Importar imágenes

RF54. Definir profundidad de la imagen.

RF55. Definir dimensiones de la imagen.

RF56. Definir brillo y contraste de la imagen.

RF57. Definir color de la imagen.

La herramienta debe proveer mecanismos para el trabajo con audio:

RF58. Cargar audio.

RF59. Reproducir audio.

RF60. Detener reproducción del audio.

RF61. Controlar volumen del audio.

RF62. Definir frecuencia de reproducción del audio.

62.1 Definir la frecuencia de reproducción de audio a través de sus propiedades.

- Compresión.
- Calidad
- Velocidad.

RF63. Definir si utilizar la reproducción cíclica o no del audio.

La herramienta debe proveer mecanismos para el trabajo con video:

RF64. Cargar video desde archivos externos.

RF65. Reproducir archivos de video.

RF66. Detener reproducción del video.

RF67. Controlar volumen del video.

RF68. Visualizar archivos de video.

RF69. Definir las dimensiones del archivo de video.

La herramienta debe proveer mecanismos para el trabajo con actores:

RF70. Especificar el tiempo de visualización de un actor.

RF71. Definir el objeto a referenciar en la librería.

RF72. Definir dimensiones de un actor.

72.1 Delimitar las extensiones de un actor mediante las propiedades.

- Largo

- Ancho

RF73. Definir la profundidad de un actor.

73.1 Delimitar la profundidad de un determinado actor en el escenario donde cada uno de estos pueda observarse según su posición.

- Profundidad X
- Profundidad Y

La herramienta debe proveer mecanismos para el trabajo con la película:

RF74. Definir el tiempo de duración de la película.

RF75. Definir la velocidad de reproducción de la película.

RF76. Definir las dimensiones de la película.

76.1 Delimitar las extensiones de una película mediante las propiedades.

- Largo
- Ancho
- Máscara

RF77. Modificar la cantidad de escenas de la película.

RF78. Modificar la secuencia de escenas de la película.

La herramienta debe proveer mecanismos para el trabajo con ORM:

RF79. Definir el tipo de servidor de bases de datos.

79.1 Permitir puntualizar que servidor de base de datos se utilizará en la aplicación.

- *MySQL*
- *PostgreSQL*
- *ODBC*

RF80. Definir el IP del servidor de bases de datos.

RF81. Definir el puerto a utilizar para la conexión.

RF82. Definir el nombre de la base de dato a utilizar.

RF83. Definir el usuario y contraseña para la conexión.

RF84. Seleccionar las tablas que serán mapeadas.

RF85. Desmarcar las tablas seleccionadas.

RF86. Crear tablas virtuales.

RF87. Mapear tablas.

RF88. Ejecutar instrucciones SQL

La herramienta debe proveer mecanismos para el trabajo con efectos:

RF89. Crear interpolación de forma.

RF90. Crear interpolación de movimiento.

RF91. Crear interpolación de transparencia.

RF92. Asociar un efecto a un actor.

RF93. Quitar un efecto a un actor.

La herramienta debe proveer mecanismos para visualizar las barras que se muestran en el entorno de trabajo:

RF94. Visualizar barra de herramientas.

RF95. Visualizar inspector de propiedades.

RF96. Visualizar biblioteca.

RF97. Visualizar paleta de componentes.

La herramienta debe proveer mecanismos para el trabajo con XML:

RF98. Crear un XML.

RF99. Cargar un XML. .

RF100. Escribir en un XML externo.

La herramienta debe proveer mecanismos para el trabajo con plugin:

RF101. Adicionar nuevas funcionalidades a la herramienta.

RF102. Eliminar nuevas funcionalidades a la herramienta.

RF103. Modificar funcionalidades de la herramienta.

RF104. Visualizar nuevas funcionalidades de la herramienta.

La herramienta debe proveer mecanismos para el trabajo con texto plano:

RF105. Crear un texto plano.

RF106. Cargar un texto plano.

RF107. Escribir en un texto plano externo.

La herramienta debe proveer mecanismos para el trabajo con ficheros binarios:

RF108 Crear un fichero binario.

RF109. Cargar un fichero binario.

RF110. Escribir en un fichero binario externo.

La herramienta debe permitir el trabajo con la ayuda:

RF111. Cargar ayuda de SLAM-GraphicDesigner.

RF112. Mostrar información de los desarrolladores.

RF113. Mostrar información de la herramienta.

2.4.2 Requerimientos no funcionales

Soporte y portabilidad

- Se debe garantizar el uso de estándares de diseño y arquitectura que faciliten el mantenimiento y ampliación del sistema en caso de que sea necesario.
- Se debe garantizar compatibilidad en versiones, donde una composición creada bajo la primera versión debe poder abrirse transparentemente bajo una versión posterior.
- El sistema podrá ejecutarse en entornos basados en Linux y Windows teniendo en cuenta que el lenguaje de programación y las herramientas sean multiplataformas.

Usabilidad

- La herramienta podrá ser utilizada por todas las personas con conocimientos básicos sobre el manejo de la computadora.

Interfaz

- Se debe garantizar que la interfaz de la aplicación sea lo más atractiva y clara posible para el desarrollador.
- Se debe garantizar que la interfaz sea amigable y de fácil comprensión para los usuarios que por primera vez la utilicen.
- La aplicación debe proveer al usuario una gran versatilidad en la presentación de la información en las pantallas, en cuanto a disponibilidad de los recursos y organización de los mismos.

Requerimientos de licencias y patentes

- Se debe garantizar que el sistema será software libre y por tanto, los componentes de software que se utilicen también deberán serlo.

Restricciones en el diseño y en la implementación

- Utilizar los patrones de diseño establecidos.
- Para la implementación de la herramienta se deberá utilizar como lenguaje de programación C ++.
- Se deberá utilizar *QtCreator* como IDE.

Hardware

- 256 RAM.
- 200 Mb de espacio libre en memoria.
- Microprocesador 2.4 GHz mínimo.

2.5 Modelo de casos de uso del sistema

Referencia a requerimientos	Nombre del caso de uso	Prioridad
RF: 3,4,5,6	Gestionar recurso en la librería	Crítico
RF: 40,41,42**,**43,44,45,46,47,48	Especificar texto	Crítico
RF: 49,50,51,52	Definir gráfico vectorial	Crítico
RF:58,59,60,61,62,63	Controlar audio	Crítico
RF: 14,15,16,17,18	Configurar objetos	Crítico
RF: 53,54,55,56,57	Especificar recurso imagen	Crítico
RF: 38,39	Gestionar escenas	Crítico
RF: 70,71,72 73	Definir actor	Crítico
RF: 94,95, 96 , 97	Visualizar paneles	Crítico
RF:19,20**,**21,22**,**23,24,25,26,27,28,29,30, 31,32, 33,34,35,36,37	Proyectar en el escenario	Secundario
RF: 74,75,76,77,78	Editar película	Secundario
RF: 64,65,66,67,68,69	Controlar video	Secundario
RF: 87,88	Ejecutar ORM	Secundario
RF: 7,8,9,10,11,12,13	Gestionar proyecto	Secundario
RF: 92,93	Interpolar	Secundario
RF: 98,99,100	Utilizar XML	Secundario
RF: 79,80,81,82,83	Especificar conexión	Secundario
RF: 84,85,86	Accionar con las tablas	Secundario
RF: 105,106, 107	Utilizar fichero de texto plano	Secundario
RF: 108,109,110	Utilizar fichero binario	Secundario
RF: 89	Interpolar forma	Opcional
RF: 91	Interpolar transparencia	Opcional
RF: 90	Interpolar movimiento	Opcional
RF: 101,102,103,104	Gestionar plugin	Opcional
RF: 1	Importar	Opcional
RF: 2	Exportar	Opcional
RF: 111,112,113	Realizar ayuda	Opcional

Tabla 2: Clasificación de los casos de uso y sus dependencias.

Debido a la complejidad de la herramienta y por la gran cantidad de requisitos que contiene, se decidió dividir el sistema en paquetes según las funcionalidades de cada caso de uso. Para este diseño se tuvo en cuenta algunos patrones de casos de uso entre los que se encuentran:

- Extensión concreta o inclusión
- Concordancia por especificación
- CRUD total

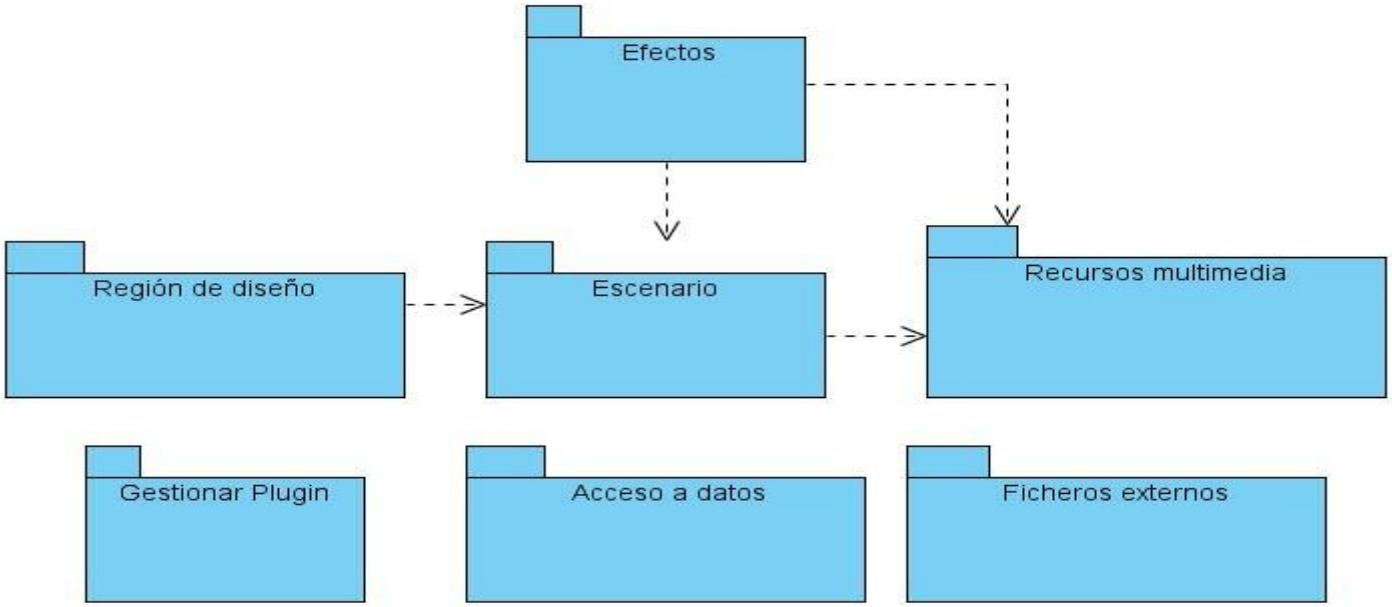


Figura 3: Vista de gestión del modelo.

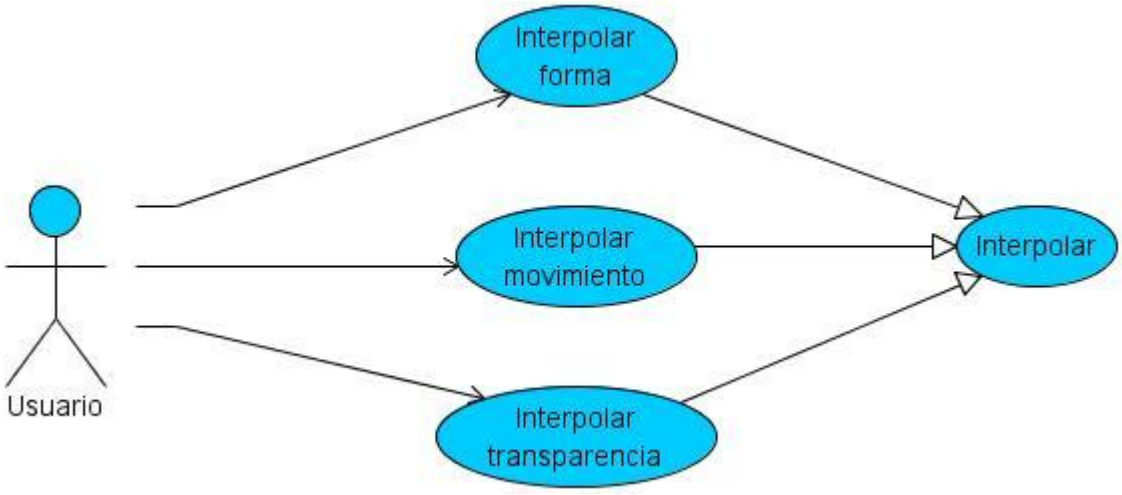


Figura 4: Diagrama de casos de uso del paquete: Efectos.

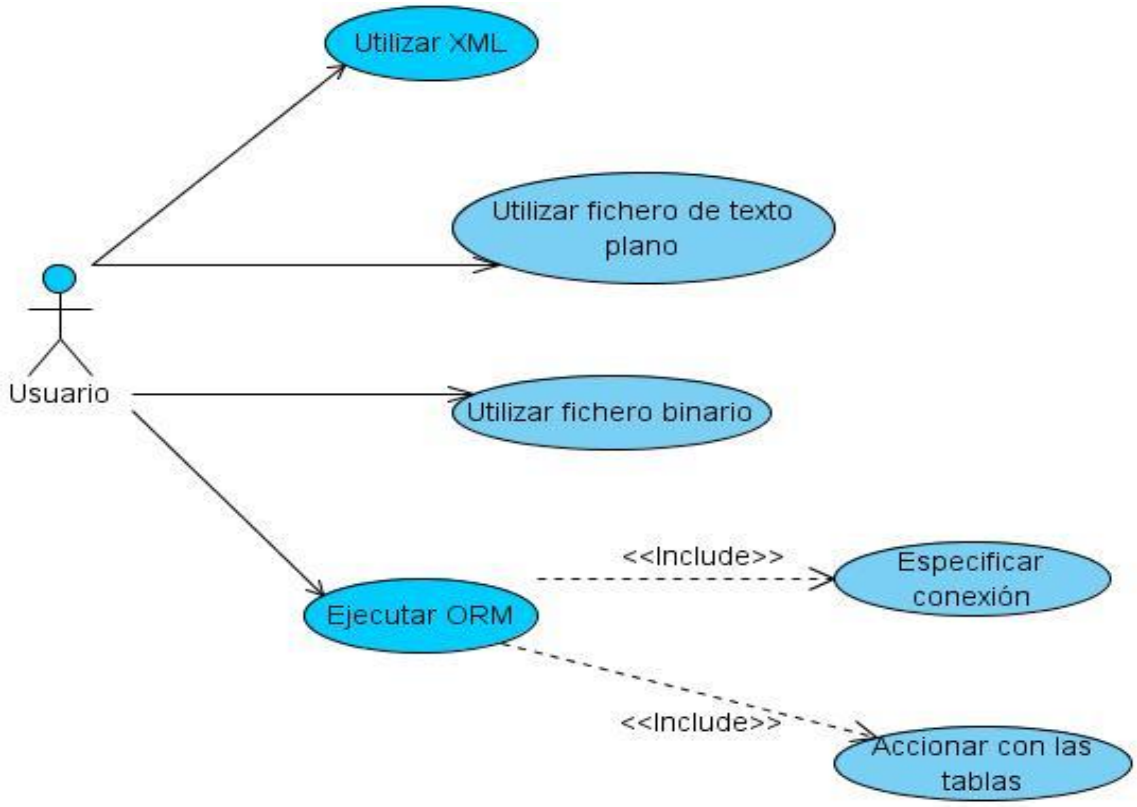


Figura 5: Diagrama de casos de uso del paquete: Acceso a datos.

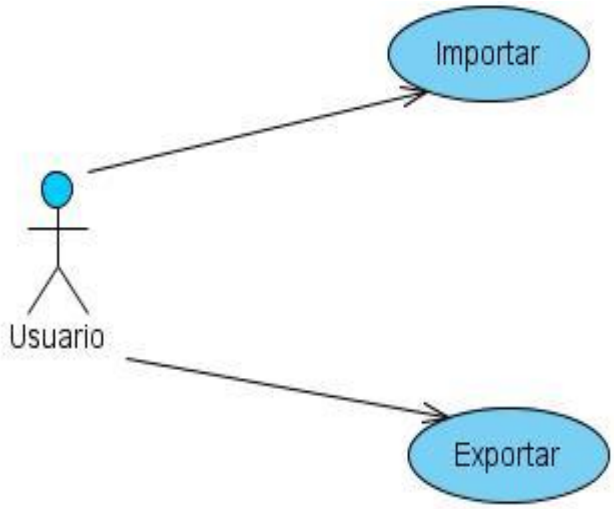


Figura 6: Diagrama de casos de uso del paquete: Ficheros externos.

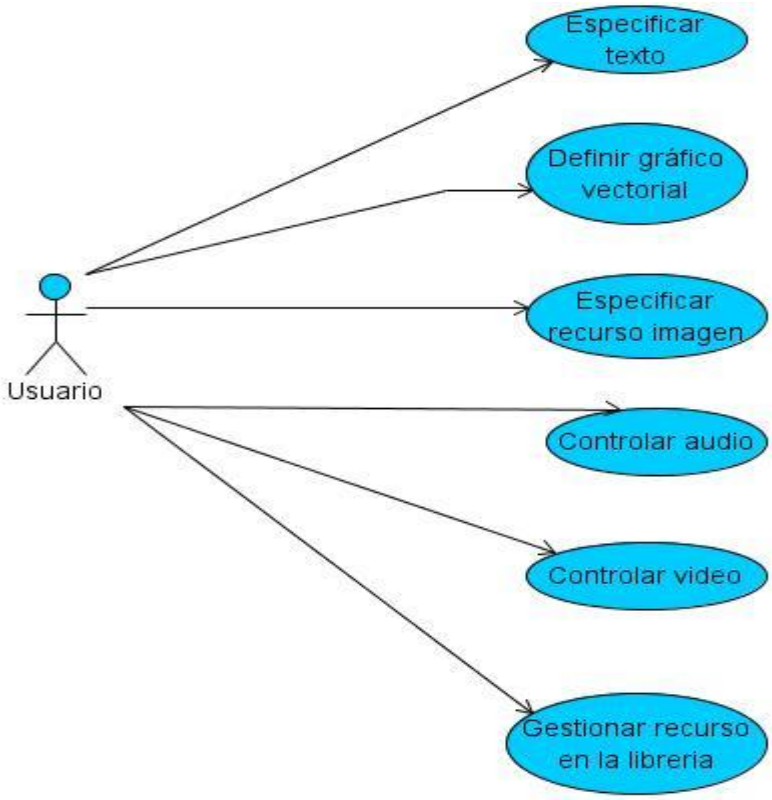


Figura 7: Diagrama de caso de uso del paquete: Recursos multimedia.

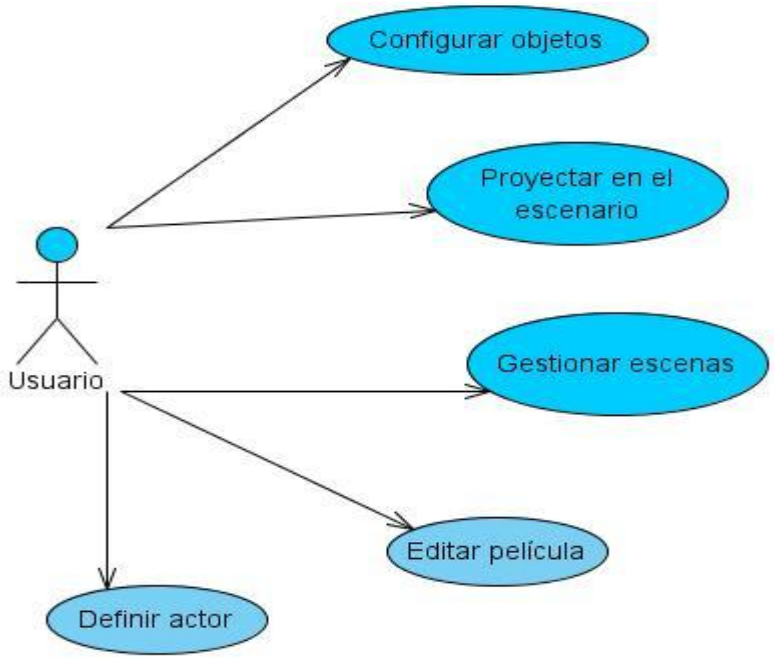


Figura 8: Diagrama de caso de uso del paquete: Escenario.

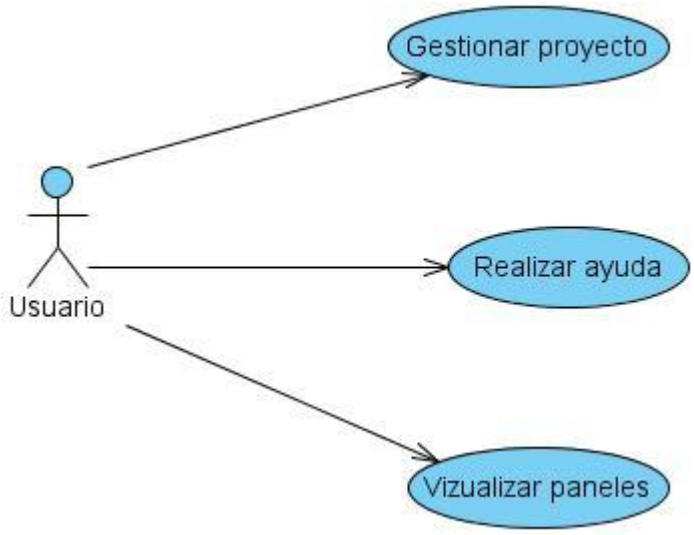


Figura 9: Diagrama de caso de uso del paquete: Región de diseño.

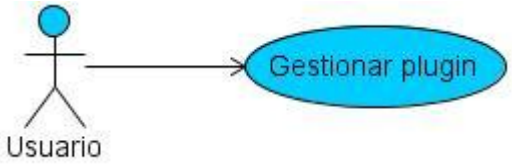


Figura 10: Diagrama de caso de usos del paquete: Gestionar plugin.

2.6 Descripción extendida del actor del sistema

Actor	Descripción
Usuario	Es el único actor de SLAM_GraphicDesigner. Inicializa todas las operaciones que se realizan en el ambiente de trabajo. Representa a una persona sin conocimientos en programación capaz de utilizar el sistema para el desarrollo de productos multimedia.

Tabla 3: Descripción del actor del sistema: usuario

2.7 Descripción extendida de los casos de uso del sistema

Caso de uso	
CU	Controlar audio
Propósito	Permitir realizar acciones con el audio.
Actor: usuario	
Resumen: El caso de uso se inicia cuando el usuario desea cargar el audio, definir las acciones y propiedades que este recurso tendrá. El sistema permite cargar el audio y realizar acciones con el mismo, finalizando así el caso de uso.	
Referencias	RF: 58,59,60,61,62,63
Prioridad	Crítico
Precondiciones	La barra de componentes debe estar activada.
Poscondiciones	El componente queda cargado con el archivo de audio.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario selecciona de la barra de componentes el componente de audio.	1.1- El sistema muestra la selección realizada.
2- El usuario arrastra el componente de audio para el escenario.	2.1- El sistema muestra en el escenario y en la biblioteca el elemento arrastrado.
3- El usuario carga el audio desde un directorio.	3.1- El sistema muestra el audio cargado en el escenario y muestra sus propiedades (control de volumen, frecuencia de reproducción, reproducción cíclica).
4- El usuario define las propiedades que desea modificar.	4.1- El sistema modifica las propiedades del audio 4.2- El sistema muestra las propiedades del audio modificadas y culmina de esta forma el caso de uso.

Tabla 4: Descripción del caso de uso del sistema: Controlar audio

Caso de uso	
CU	Especificar recurso imagen
Propósito	Definir las propiedades de la imagen.
Actor: usuario	

Resumen: El caso de uso se inicia cuando el usuario carga un objeto de tipo imagen y procede a definir las propiedades que este recurso tendrá. El sistema permite la realización de estas acciones finalizando así el caso de uso.	
Referencias	RF: 53,54,55,56,57
Prioridad	Crítico
Precondiciones	La barra de componentes debe estar activada.
Poscondiciones	La imagen queda mostrada en el escenario.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario selecciona de la barra de componentes, el componente de la imagen.	1.1- El sistema muestra la selección realizada.
2- El usuario arrastra el componente de imagen para el escenario.	2.1- El sistema muestra en el escenario y en la biblioteca el elemento arrastrado.
3- El usuario carga la imagen.	3.1- El sistema muestra la imagen cargada en el escenario y muestra sus propiedades (profundidad, dimensiones, brillo, contraste, color).
4- El usuario define que propiedades desea modificar.	4.1 -El sistema modifica las propiedades. 4.2-El sistema muestra las propiedades modificadas y culmina el caso de uso.

Tabla 5: Descripción del caso de uso del sistema: Especificar recurso imagen

Para apreciar otras descripciones de los casos de uso, ver **anexo 6.**

2.8 Arquitectura de la información

Existen muchas definiciones de arquitectura de la información que permiten comprender mejor el término, por ejemplo Richard Saul Wurman, que la define como el estudio de la organización de la información con el objetivo de permitir al usuario encontrar la vía de navegación hacia el conocimiento y la comprensión de la información.

Existen otros autores que definen a la arquitectura como:

- El arte y la ciencia de estructurar y clasificar sitios web e intranets con el fin de ayudar a los usuarios a encontrar y manejar la información. Determina el contenido y las funcionalidades que la herramienta va a tener **[25]**.

De forma general la arquitectura de la información es la disciplina que organiza la estructuración del trabajo, permitiendo que todas las personas lo entiendan de manera simple. Se utiliza fundamentalmente en espacios virtuales donde se requiere que el propio usuario obtenga la información.

2.8.1 Actividades en el proceso de levantamiento de información

En esta etapa se determinan los objetivos principales de la organización a la que se le entregará el producto, permite desarrollar un satisfactorio levantamiento de la información. En las primeras iteraciones se deben analizar algunos aspectos como la misión y visión de la organización.

En el caso de SLAM-GraphicDesigner el objetivo general que rige el desarrollo de la aplicación es que se desea construir una herramienta de diseño, que les permita a los usuarios desarrollar aplicaciones multimedia de una forma rápida, organizada y detallada facilitando así la eficiencia de la misma. La visión es establecer una herramienta dentro de la Universidad, que de la posibilidad a varios usuarios de acceder a la misma sin poseer grandes conocimientos sobre temas de programación, ampliando las posibilidades creativas.

2.8.1.1 Definición de los objetivos que se persiguen con el desarrollo de SLAM-GraphicDesigner:

- Desarrollar la herramienta para que sirva de guía en la construcción de futuras herramientas de diseño.
- Permitir a los diversos usuarios desarrollar multimedia interactivas complejas con pocos pasos.
- Convertirse en una fuente de conocimiento para la comunidad universitaria.

2.8.1.2 Definición de la audiencia del sistema

Clasificación de la audiencia

Las personas que deseen realizar multimedia con la herramienta.

Necesidades de la audiencia

En sentido general la necesidad de la audiencia es poder desarrollar multimedia de forma rápida con la herramienta.

2.8.1.3 Análisis de homólogos

Nombre del producto

Adobe Flash

URL del producto
http://www.adobe.com

Objetivo de la creación del producto

1. El principal objetivo de la herramienta es brindar oportunidades para que todos los usuarios puedan desarrollar productos multimedia de forma eficiente.

Servicios que brinda *Adobe Flash*

- Recursos para el trabajo con audio
- Recursos para el trabajo con video
- Recursos para el trabajo con imágenes
- Recursos para el trabajo con gráficos vectoriales
- Recursos para el trabajo con efectos
- Recursos para el trabajo con animaciones
- Recursos para el trabajo con base de datos
- Recursos para el trabajo con textos

Ofrece componentes que permiten el trabajo con botones, menús, entre otros.

2.8.2 Organización de la información

En la organización de la información la función principal del arquitecto de información es organizar las grandes cantidades de contenidos con el objetivo de que el usuario que trabaje con la herramienta pueda manejarlos fácilmente y satisfacer sus necesidades de creación.

2.8.2.1 Realización de la taxonomía

Es una representación de la estructura de la aplicación en cuanto a las etiquetas y a la jerarquía de los contenidos. En esta sección se realiza una descripción textual de los elementos de la estructura, características y comportamiento.

Diagrama de la taxonomía.

1. Archivo
1.1 Nuevo proyecto

5.2.4 Rectángulo
5.2.5 Poli línea

1.2 Abrir proyecto
1.3 Guardar proyecto
1.4 Guardar como
1.5 Cerrar proyecto
1.6 Importar
1.7 Exportar
1.8 Salir

2. Editar

2.1 Eliminar
2.2 Copiar
2.3 Cortar
2.4 Pegar

3. Texto

3.1 Fuente
3.2 Tamaño
3.3 Estilo
3.4 Color
3.5 Alineación
3.6 Dimensiones

4. Control

4.1 Reproducir
4.2 Detener
4.3 Control de volumen

5. Ventana

5.1 Línea de tiempo
5.2 Barra de herramientas
5.2.1 Seleccionar
5.2.2 Texto
5.2.3 Línea
5.2.6 Cuadrado
5.2.7 Elipse
5.2.8 Círculo
5.2.9 Lápiz
5.2.10 Relleno
5.2.11 Zoom
5.2.12 Gradiente
5.2.13 Transformación libre
5.2.14 Transformación de relleno
5.2.15 Pluma
5.2.16 Goma
5.2 17 Pincel

5.3 Inspector de propiedades

5.3.1 Propiedades
5.3.1.1 Largo
5.3.1.2 Ancho
5.3.1.3 Profundidad X
5.3.1.4 Profundidad Y

5.4 Paleta de componentes

5.4.1 Interfaz de usuario
5.4.1.1 Etiqueta
5.4.1.2 Botón
5.4.1.3 Menú
5.4.1.4 *RadioButton*
5.4.1.5 *CheckBox*
5.4.1.6 *ComboBox*
5.4.1.7 *ColorPicker*
5.4.1.8 Lista
5.4.2 Video
5.4.2.1 *BackButton*
5.4.2.2 *PlayButton*
5.4.2.3 *PauseButton*
5.4.2.4 *MuteButton*

5.4.2.5 PlayPauseButton

5.4.2.6 *StopButton*

5.4.2.7 *VolumenBar*

5.4.3 Efectos

5.4.3.1 Interpolación de transparencia

5.4.3.2 Interpolación de movimiento

5.4.3.3 Interpolación de forma

5.5 Librería

6. ORM

6.1 Seleccionar

6.2 Conexión

6.2.1 Tipo de servidor

6.2.2 IP

6.2.3 Nombre de la base de datos

6.2.4 Usuario y contraseña

6.3 Desmarcar

6.4 *Join*

6.5 Ejecutar instrucciones

7. Ayuda

7.1 Ayuda de SG

7.2 Contactos de los desarrolladores de SGD

7.3 Más sobre la documentación de SGD

Se decidió realizar la taxonomía de esta forma porque mejora de la navegación y el desarrollo de sistemas de búsqueda basados en la exploración. Además porque la realización de la arquitectura de la información se efectuó a través del apoyo de plantillas establecidas por el proyecto de Informatización de la Universidad de las Ciencias Informáticas.

2.8.2.2 Definición del sistema de navegación

El sistema de navegación es una representación gráfica de cómo estará organizada la información dentro de la herramienta, para definir todas las etiquetas del sistema, se desarrolló un mapa de navegación que representa como queda cada una de estas estructurada. Véase mapa de navegación en **anexos 8.**

Propuesta de etiquetado

Archivo: etiqueta que muestra un menú desplegable con una serie de opciones entre las que se encuentran:

Nuevo proyecto: mostrará una ventana posibilitando crear un nuevo proyecto donde se especificará el tamaño deseado para la región de diseño. Pertenece a la categoría de archivo y estará a un segundo nivel.

Abrir un proyecto: mostrará una ventana posibilitando abrir un proyecto desde la computadora o un dispositivo. Pertenece a la categoría de archivo y estará a un segundo nivel.

Guardar proyecto: mostrará una ventana posibilitando guardar un proyecto en un directorio. Pertenece a la categoría de archivo y estará a un segundo nivel.

Guardar como: mostrará una ventana posibilitando guardar un proyecto con las extensiones definidas por el sistema en un directorio. Pertenece a la categoría de archivo y estará a un segundo nivel.

Importar: mostrará una ventana posibilitando importar cualquier archivo existente desde un directorio. Pertenece a la categoría de archivo y estará a un segundo nivel.

Exportar: mostrará una ventana posibilitando exportar el proyecto hacia la computadora o cualquier dispositivo sea externo o interno. Pertenece a la categoría de archivo y estará a un segundo nivel.

Editar: etiqueta que muestra un menú desplegable con una serie de opciones entre las que se encuentra:

Eliminar: opción que brinda la posibilidad de excluir cualquier acción dentro del escenario.

Copiar: opción que brinda la posibilidad de reproducir cualquier acción dentro del escenario.

Pegar: opción que brinda la posibilidad de fijar cualquier acción dentro del escenario.

Cortar: opción que brinda la posibilidad de fragmentar cualquier acción dentro del escenario.

Texto: etiqueta que muestra un menú desplegable con una serie de opciones entre las que se encuentra:

Fuente: mostrará un menú que pertenece a la categoría de texto que estará a un segundo nivel, posibilitando definir la fuente del texto y estará a un tercer nivel.

Tamaño: mostrará un menú que pertenece a la categoría de texto, posibilitando definir el tamaño de las letras y estará a un segundo nivel.

Estilo: mostrará un menú que pertenece a la categoría de texto, posibilitando definir el estilo de las letras y estará a un segundo nivel.

Color: mostrará un menú que pertenece a la categoría de texto, posibilitando definir el color de las letras y estará a un segundo nivel.

Alineación: mostrará un menú que pertenece a la categoría de texto, posibilitando definir el tipo de alineación que tendrá el texto y estará a un segundo nivel**.**

Dimensiones: mostrará un menú que pertenece a la categoría de texto, posibilitando definir las dimensiones que tendrá el texto y estará a un segundo nivel.

Control: mostrará una serie de opciones que podrán ser ocultadas o mostradas según las necesidades de los usuarios entre ellas se encuentran:

Reproducir: opción que brindará la posibilidad de reproducir todas las acciones de la película. Pertenece a la categoría control y estará a un segundo nivel.

Detener: opción que brindará la posibilidad de pausar los archivos de audio y video. Pertenece a la categoría control y estará a un segundo nivel.

Control de volumen: opción que brindará la posibilidad de chequear el volumen de los archivos de audio y video. Pertenece a la categoría control y estará a un segundo nivel.

Ventana: etiqueta que muestra un menú desplegable con una serie de opciones que podrán ser ocultadas o mostradas según las necesidades de los usuarios entre ellas se encuentra:

Línea de tiempo: posibilita realizar acciones en la película. Pertenece a la categoría de ventana y estará a un segundo nivel.

Barra de herramientas: posibilita o no trabajar con todas las herramientas disponibles para la película. Pertenece a la categoría de ventana y estará a un segundo nivel.

Inspector de propiedades: barra que contiene los siguientes elementos:

Propiedades: ventana que permitirá mostrar o no las propiedades generales de cada actor. Pertenece a la categoría inspector de propiedades y estará a un tercer nivel.

Parámetros: ventana que permitirá mostrar o no los parámetros generales de cada actor. Pertenece a la categoría inspector de propiedades y estará a un tercer nivel.

Paleta de componentes: barra que contiene los siguientes elementos:

Interfaz de usuario: menú desplegable que se mostrará en la paleta de componentes a partir de la selección en un menú, que abrirá los componentes que existen en la herramienta, posibilitando trabajar con cada uno de ellos. Pertenece a la categoría paleta de componentes y estará a un tercer nivel.

Video: menú desplegable que se mostrará en la paleta de componentes a partir de la selección en un menú, que abrirá los componentes que existen en la herramienta, posibilitando trabajar con cada uno de ellos. Pertenece a la categoría paleta de componentes y estará a un tercer nivel.

Biblioteca: mostrará la biblioteca de la herramienta posibilitando almacenar todos los actores de la película. Pertenece a la categoría de ventana y estará a un segundo nivel.

Opciones: posibilita o no especificar todas las preferencias generales del sistema. Pertenece a la categoría ventana, estará a un segundo nivel y contiene los siguientes elementos:

Generales: define las propiedades globales que se aplicarán a la película. Pertenece a la categoría opciones y estará a un tercer nivel.

Interfaz: define las propiedades que se podrán aplicar en la interfaz a todos los elementos de la película. Pertenece a la categoría opciones y estará a un tercer nivel.

Manejo de archivo: define todas las opciones para el trabajo con archivos. Pertenece a la categoría opciones y estará a un tercer nivel.

Unidades y reglas: define las dimensiones que tendrán todos los elementos del diseño. Pertenece a la categoría opciones y estará a un tercer nivel.

Plugin: permite seleccionar los plugin que podrán ser utilizados en el diseño. Pertenece a la categoría opciones y estará a un tercer nivel.

Texto: ofrece todas las propiedades relacionadas con el texto que tendrá el diseño. Pertenece a la categoría opciones y estará a un tercer nivel.

Plugin: etiqueta que muestra un menú desplegable con una serie de opciones que podrán ser ocultadas o mostradas según las necesidades de los usuarios, entre ellas se encuentra:

ORM: es un plugin en específico que trabaja con base de datos y a su vez contiene:

Conexión: mostrará una ventana que dará la posibilidad de introducir los datos necesarios para hacer la conexión con la base de datos. Pertenece a la categoría de ORM y estará a un segundo nivel.

Ejecutar instrucciones: mostrará una ventana dando los resultados de las instrucciones que fueron ejecutadas. Pertenece a la categoría de ORM y estará a un segundo nivel.

Seleccionar: brindará la opción de elegir las tablas con las que el usuario desea trabajar.

Desmarcar: brindará la opción de deshacer la elección realizada en las tablas.

Ayuda: etiqueta que muestra un menú desplegable con una serie de opciones entre las que se encuentran:

Ayuda de SGD: mostrará toda la información relacionada con el uso de la herramienta para las personas que tengan duda de cómo trabajar con la misma. Pertenece a la categoría de ayuda y estará a un segundo nivel.

Contactos de los desarrolladores: brindará todos los datos de contacto de los desarrolladores de la herramienta. Pertenece a la categoría de ayuda y estará a un segundo nivel.

Más sobre la documentación de SGD: mostrará toda la información relacionada con la herramienta así como temas legales y otros. Pertenece a la categoría de ayuda y estará a un segundo nivel.

Contactos de los desarrolladores de SGD: mostrará todos los datos de contacto de los desarrolladores de la herramienta. Pertenece a la categoría de ayuda y estará a un segundo nivel.

2.8.3 Diseño de la arquitectura información

Maquetas del producto

Las maquetas son la representación visual de los prototipos de interfaz de usuario que tendrá la herramienta, están formadas por un conjunto de componentes que los usuarios ven y con los que interactúan para comunicarse con la computadora **[26].** A continuación se muestra el prototipo general de la herramienta. Véanse otros prototipos de interfaz en **anexos 1**.

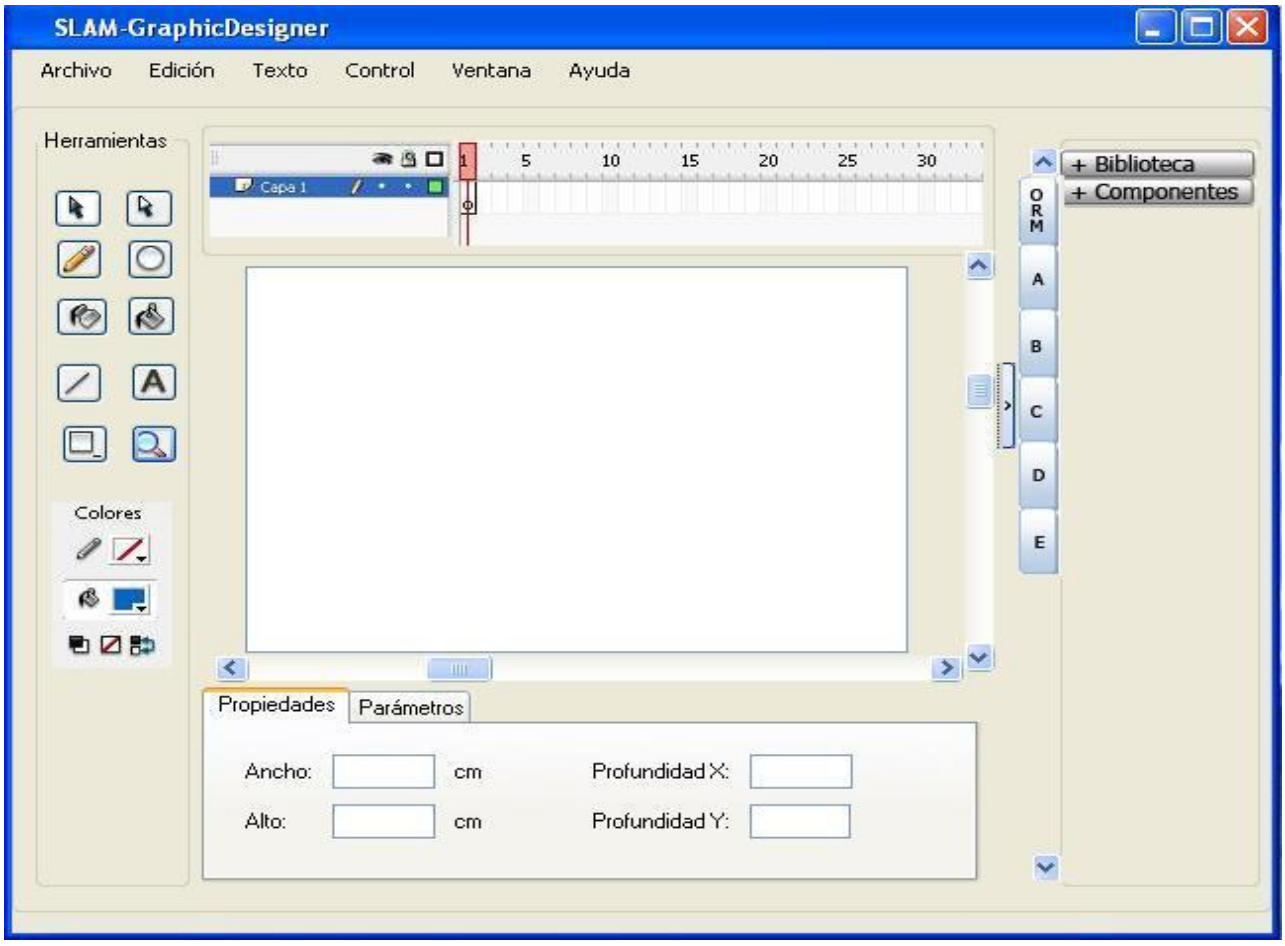


Figura 11: Pantalla general de la herramienta

2.9 Conclusiones

El análisis de cada uno de estos aspectos ofrecidos a lo largo de este capítulo permite arribar de manera parcial a las siguientes conclusiones:

- Para desarrollar una herramienta de diseño sólida que permita la construcción de aplicaciones multimedia, es necesario tener en cuenta todos los requerimientos especificados.
- La herramienta se basa en el modelado del contenido multimedia y no en la creación de estos recursos.
- Para lograr un buen diseño de SLAM-GraphicDesigner es necesario seguir cada especificación de la arquitectura de la información que dará una panorámica general de cómo deberá quedar el producto una vez terminado.

Capítulo 3: Construcción de la solución propuesta

3.1 Introducción

El presente capítulo está dedicado a la disciplina análisis y diseño, donde se describen los principales elementos de la herramienta, profundizándose en los casos de usos detallados en el capítulo anterior, de forma tal que permitan reflejar una vista interna del sistema descrito. Se documenta dicho flujo de trabajo, obteniéndose una serie de artefactos que permitirán sentar las bases para su futura implementación.

3.2 Modelo de análisis

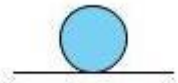
El modelo de análisis del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño que se utilice. Este consiste en obtener una visión de la herramienta que indica el "qué" debe de hacer la misma, de modo que sólo se toman en cuenta los requisitos funcionales.

El modelo de análisis puede considerarse como una primera aproximación al modelo de diseño. Los principales propósitos del mismo son:

- Conseguir una comprensión más precisa de los requisitos, refinarlos y estructurarlos.
- Utilizar el lenguaje de los desarrolladores para analizar con profundidad los requisitos funcionales.
- Proporcionar una visión general de la herramienta.

3.2.1 Definición de los estereotipos del modelo de análisis

Para la modelación de los diagramas de clases del análisis se utilizarán los estereotipos básicos que propone UML, estos representan los tres elementos principales para la construcción de un sistema. Véase la **tabla 6.**

Nombre	Características	Representación
Entidad	Modelan información que posee larga vida y que a menudo es persistente	
Control	Coordinan las realizaciones de los casos de uso, regulando las actividades de los objetos que implementan su funcionalidad	

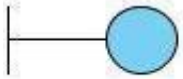
Interfaz	Modelan la interacción entre el sistema y los actores.	

Tabla 6: Definición de los estereotipos del modelo de análisis

3.2.2 Diagrama de clases del análisis

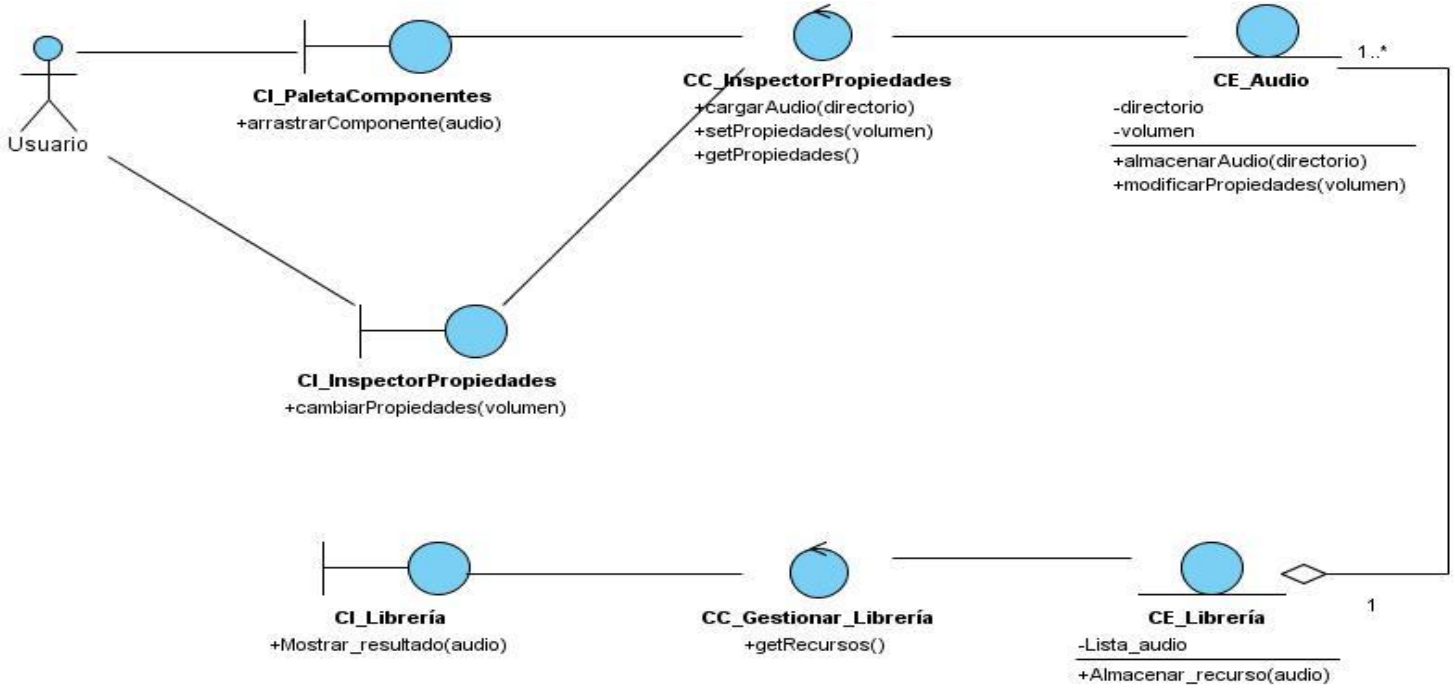


Figura 12: Descripción del caso de uso del sistema: Controlar audio

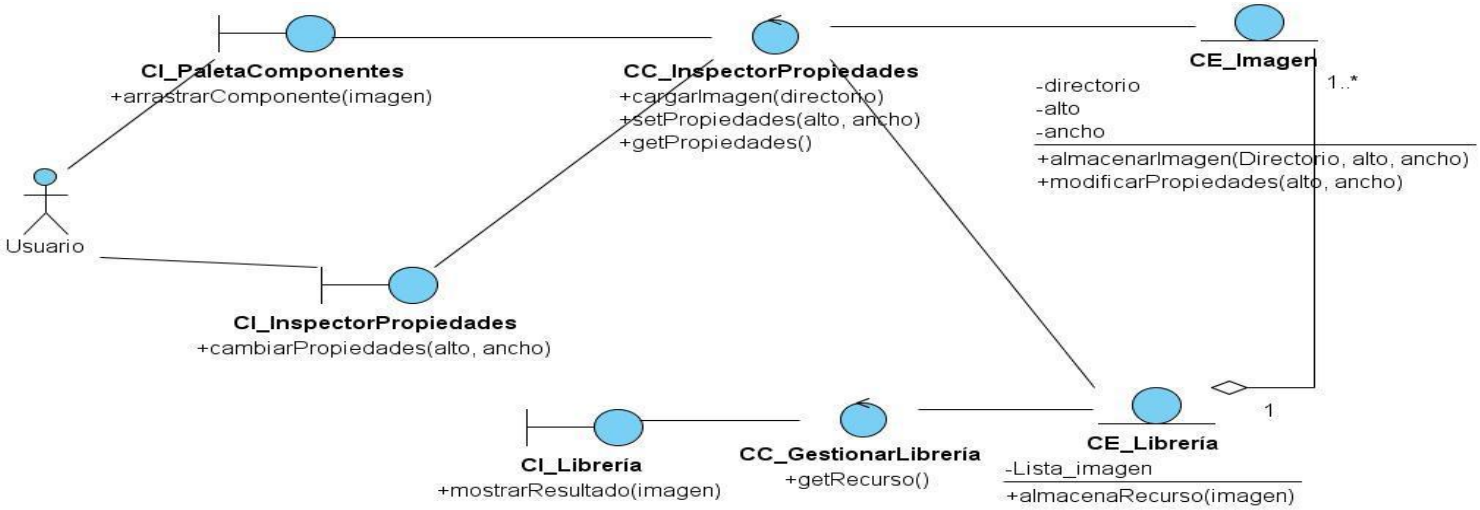


Figura 13: Descripción del caso de uso del sistema: Especificar recurso imagen

Para apreciar otros diagramas de clases del análisis referirse al **anexo 2.**

3.3 Modelo de diseño

El modelo de diseño es un refinamiento del modelo de análisis que tiene en cuenta los requisitos no funcionales e indica el "cómo" cumple el sistema sus objetivos. Una vez que se analizan y se especifican los requisitos del software, esta es la primera y más importante de las tres actividades técnicas que son diseño, implementación y pruebas que se requieren para construir y verificar el software.

3.3.1 Patrones de diseño GRASP

Teniendo en cuenta que la asignación de responsabilidades es importante en el diseño orientado a objetos, se decidió utilizar la mayoría de los patrones GRASP. Estos constituyen la base de cómo se diseñará el sistema, pues describen los principios fundamentales del diseño y la asignación de responsabilidades, expresados como patrones.

Entre los utilizados se encuentran

- Experto en información
- Creador
- Alta cohesión
- Bajo acoplamiento
- Controlador

3.3.2 Patrones de diseño GOF

Los patrones de diseño son una forma de reutilizar la experiencia de los desarrolladores, clasificando y describiendo las posibles soluciones a problemas que ocurren frecuentemente en el desarrollo. Todos ellos contribuyen a la construcción de un diseño más elegante y robusto. Con el objetivo de lograr un mayor entendimiento de los mismos. El grupo de GOF *(Gang Of Four*) los clasificó en 3 categorías basadas en su propósito:

Creacionales: el objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

Estructurales: estos describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades.

Comportamiento: definen la comunicación e iteración entre los objetos de un sistema con el fin de reducir el acoplamiento entre los objetos.

Patrón estructural Fachada

Este patrón se emplea en los subsistemas administrar proyecto, interfaz de usuario y gestionar plugin porque en varias ocasiones en el sistema existen grupos de tareas muy frecuentes. Para dar solución a este problema se define una clase fachada que realizará el papel de clase intermediaria para poder acceder a cada uno de estos subsistemas agregándole funcionalidades que agrupen tareas de forma sencilla y clara.

Patrón de comportamiento Observador

Conocido como "*spider"* se emplea en los subsistemas administrar proyecto, interfaz de usuario y gestionar plugin porque se detecta una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes de él.

Patrón estructural Decorador

Se emplea en el subsistema efecto porque permite configurar de forma dinámica las propiedades de un objeto.

3.3.3 Patrones arquitectónicos

Existen diferentes patrones arquitectónicos dentro de los que se pueden mencionar el patrón arquitectónico modelo vista controlador (MVC),el patrón arquitectónico basado en componentes y el patrón arquitectónico de tres capas. A continuación se explican las características principales:

MVC

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el sistema de gestión de base de datos y el controlador representa la lógica del negocio.

Arquitecturas en Capas

Define cómo organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo que quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores.

Arquitectura basada en Componentes

Describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de

comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos.

Principios fundamentales

Un componente es un objeto de software específicamente diseñado para cumplir con cierto propósito. Los principios fundamentales cuando se diseña un componente deben ser:

• **Reusable**: los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.

• **Extensible**: un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.

• **Encapsulado:** los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, ni detalles del proceso o estado.

• **Independiente**: los componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas **[27]**.

Después de haber realizado un estudio de los patrones arquitectónicos se escogió para la realización de la herramienta SLAM-GraphicDesigner, la arquitectura basada en componentes, porque ofrece una interfaz bien definida para proveer funcionalidades específicas permitiendo el desarrollo sin impactar otras partes del sistema. Además potencia el uso de componentes reutilizables que pueden ser empleados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas.

3.3.4 Diagramas de clases del diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones con el fin de modelar la vista de diseño estática del sistema. Para un mejor entendimiento del diseño ver glosario de términos para el diseño en **anexo 5.**

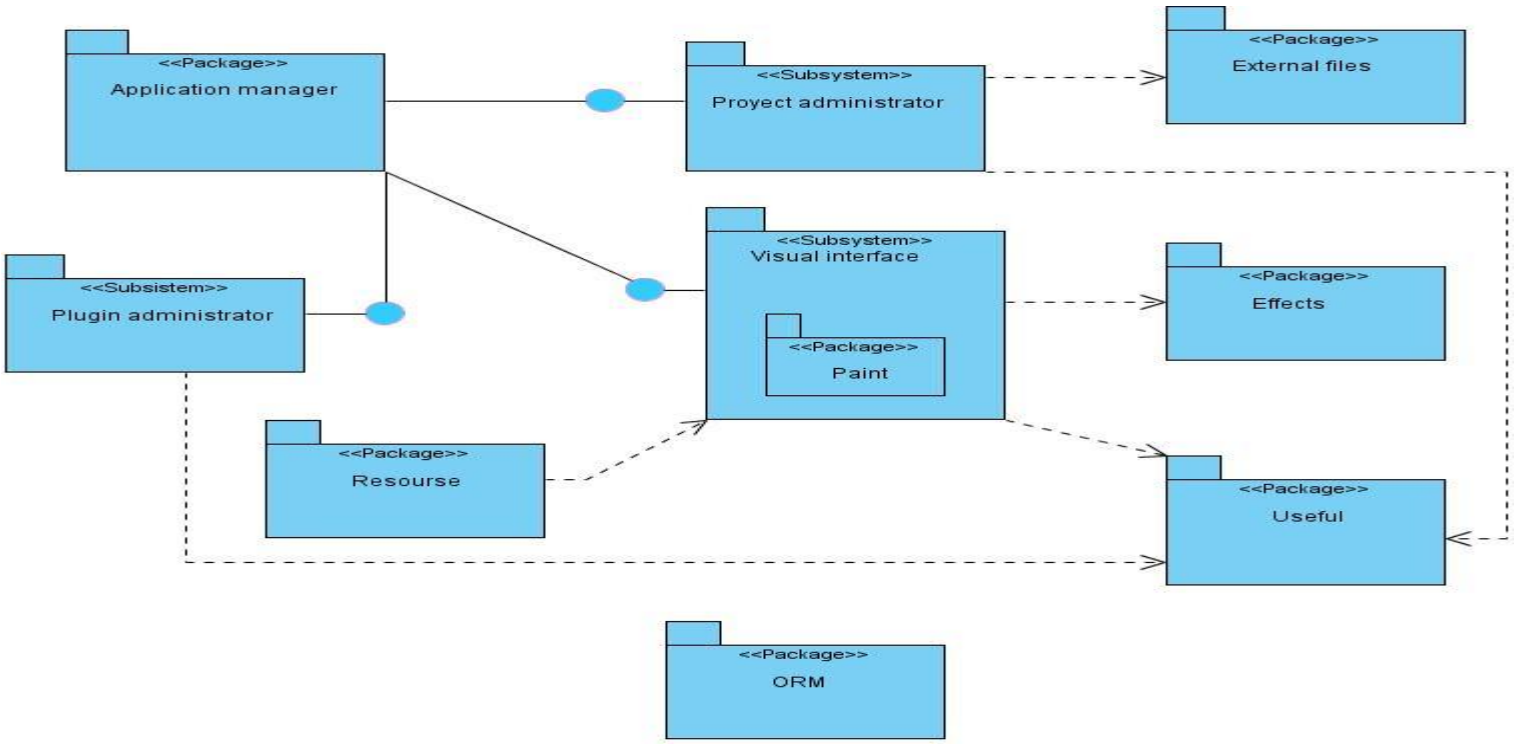


Figura 14: Relación entre los subsistemas del modelo de diseño

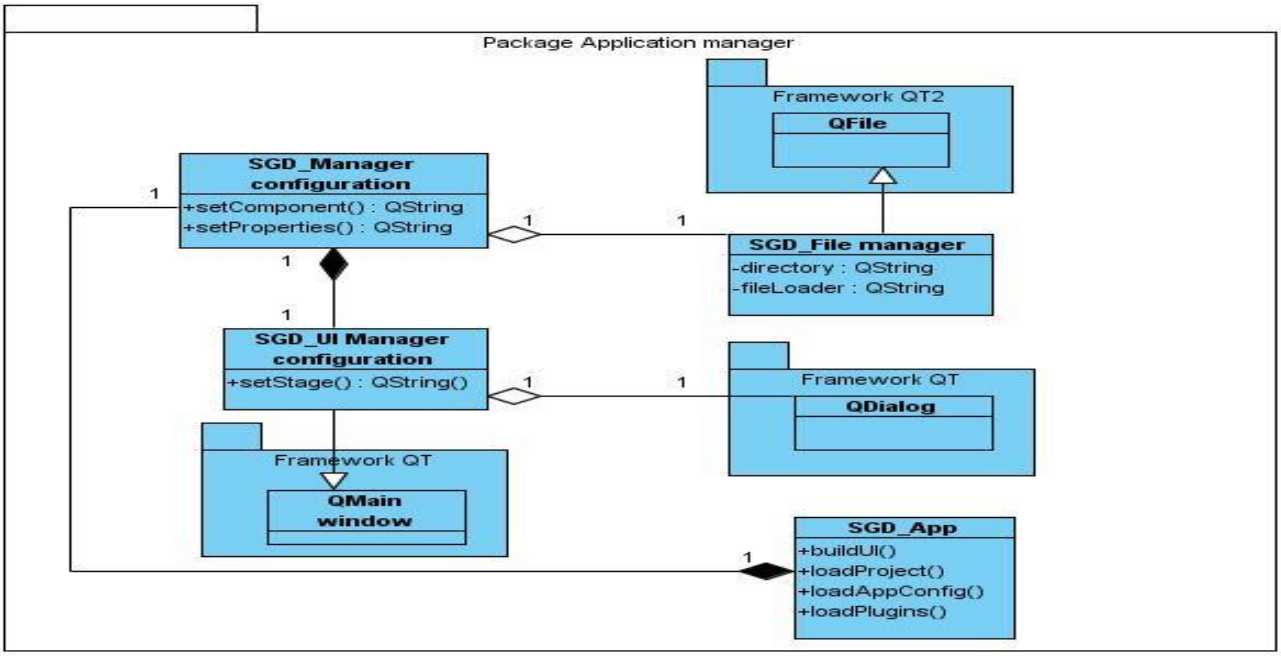


Figura 15: Diagrama de clases del paquete: Administración aplicación

Los diagramas de los subsistemas de diseño restantes y los diagramas de secuencia, se muestran en los **anexo 3 y 7** respectivamente**.**

3.4 Diagrama de despliegue

El modelo de despliegue describe cómo una aplicación se desarrolla. La intención de este modelo no es describir la infraestructura, sino el camino en que los componentes específicos deben corresponder a una aplicación que se despliega a través de él. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada con la expansión del sistema propuesto. Muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, las instancias de los componentes y objetos que residen en ellos.

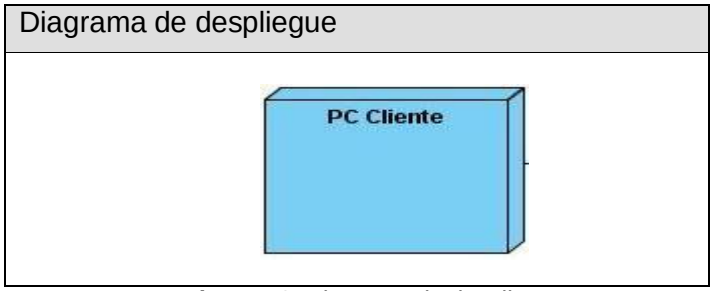


Figura 16: Diagrama de despliegue

3.5 Estándar del proyecto

Después de haber realizado un estudio de los estándares que existen, se tomaron como ejemplo el que brinda *QtCreator* que es el IDE seleccionado y el que ofrece el lenguaje HTML porque son interpretados con facilidad, tienen una excelente organización y son fáciles de entender. Tomando como punto de partida los ejemplos analizados, se llegó a la conclusión que era necesario definir un estándar para SLAM-GraphicDesigner porque se garantiza una mejor organización del contenido generado por la herramienta y para que todos los usuarios que deseen agregarle nuevas funcionalidades a esta deban regirse por el estándar establecido, con el objetivo de que el trabajo realizado quede con mayor calidad y organización. Además para que otros sistemas que deseen anexarla puedan entender como esta funciona sin complejidad alguna, ver **figura 17.**

Etiquetas del estándar

Proyect: es la principal que contiene todos los elementos de la película.

Property: es la que contiene las características generales de la película.

Tittle: muestra el título de la película en general.

Name: muestra el nombre del proyecto que será modificado o no por el usuario.

Width: muestra el ancho que tiene el proyecto.

Height: muestra el alto que tiene el proyecto.

FullScreen: muestra el proyecto que se está realizando en la pantalla completa.

Mask: contiene todas las propiedades de la máscara donde se está trabajando.

Type: especifica el tipo de formato del archivo con que se está trabajando.

Path: especifica la dirección en la que se encuentra situado el archivo con que se está trabajando.

Filedname: especifica el nombre de la máscara con que se está trabajando.

TimeLine: está contenida dentro del proyecto principal, contiene sus propiedades específicas y los fotogramas definidos en cada escena.

Property: especifica las propiedades de la línea de tiempo.

Frames: son los fotogramas de la película que contiene una o muchas escenas en un determinado instante de tiempo.

Actor: contiene las propiedades de un determinado actor en un instante de tiempo en la escena.

Property: muestra las propiedades generales de los actores en la película.

Objeto: contiene los objetos de la película con las propiedades específicas de cada uno de ellos en un instante de tiempo, estos podrán ser video, audio, imagen, texto.

Type: contiene el nombre del objeto.

Property: contiene las propiedades específicas de un objeto.

Scene: contiene los actores en cada una de las escenas de la película en un determinado instante de tiempo.

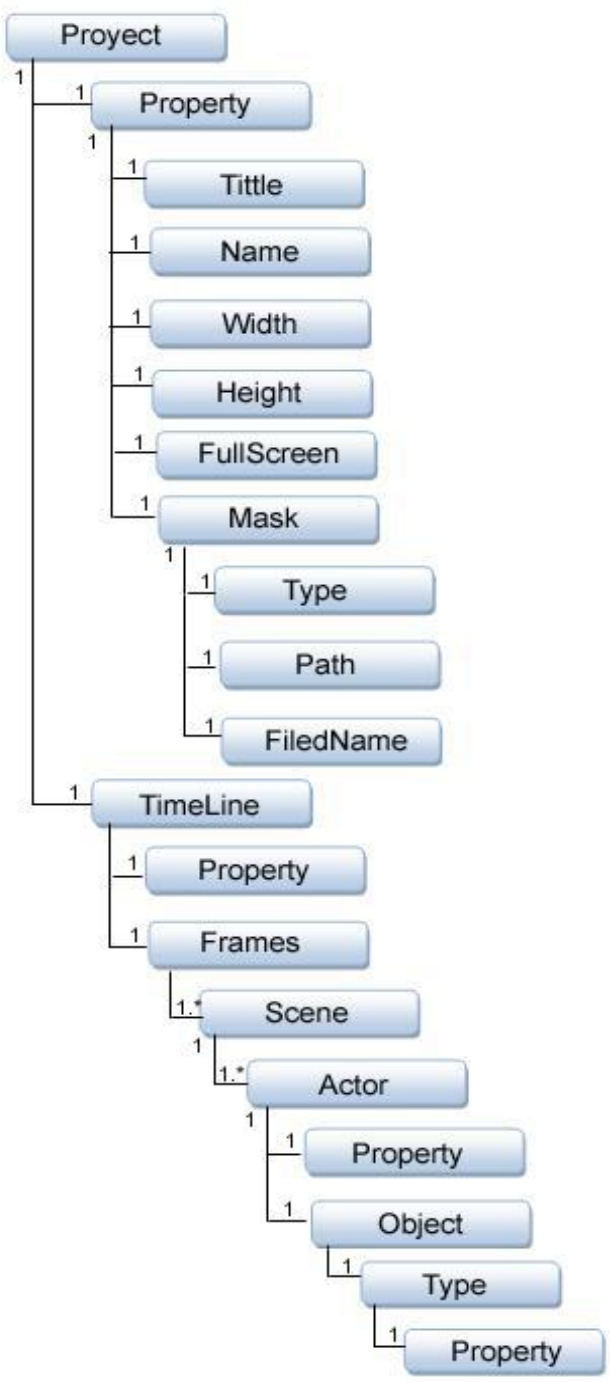


Figura 17: Estándar de proyecto

Para ver como quedaría el estándar en una situación dada, ver **anexos 4.**

3.6 Conclusiones

El análisis de cada uno de los aspectos ofrecidos a lo largo de este capítulo permite arribar de manera parcial a las siguientes conclusiones:

Los módulos más significativos que conforman el sistema son:

- El subsistema *Proyect administrator* que facilita el trabajo con Ficheros externos y con temas específicos de configuración de proyecto, siendo estos elementos importantes a la hora de hacer productos multimedia.
- El subsistema *User interface* que provee de un conjunto de clases que implementan la metáfora cine, abstrayendo de esta forma a los desarrolladores del proceso de construcción de multimedia.
- El subsistema *Plugin* provee mecanismos para desarrollar aplicaciones multimedia aplicando arquitecturas de componente, un elemento importantísimo que facilita la reutilización de código y el soporte de las mismas.
- La arquitectura utilizada en la construcción de la herramienta es la basada en componentes porque permite anexar nuevas funcionalidades al sistema sin necesidad de modificarlo.
- Para la construcción de SLAM-GraphicDesigner es indispensable la utilización de patrones. Estos comunican los estilos y soluciones consideradas como buenas prácticas, que los expertos en el diseño orientado a objetos utilizan para la creación de sistemas.

Conclusiones

Las herramientas de diseño se han ido insertando de manera satisfactoria en el gran mundo de los recursos media, brindándoles a los usuarios una mayor facilidad y sencillez a la hora de desarrollar sus aplicaciones, e insertando nuevas técnicas que permiten desarrollar productos multimedia de forma eficiente.

- Se estudiaron las herramientas y tecnologías actuales empleadas en el desarrollo de aplicaciones multimedia, lo que permitió determinar los elementos reutilizables y genéricos, así como requerimientos más críticos con los que debe cumplir SLAM_GraphicDesigner con el objetivo de que esta pueda extenderse al contexto productivo nacional de dicha esfera.
- A partir de los problemas encontrados se propone que esta herramienta de diseño utilice la arquitectura basada en componentes, teniendo en cuenta el éxito que ha alcanzado dicha arquitectura en el desarrollo de aplicaciones, permitiendo un alto nivel organizacional y solidez en su interfaz
- La herramienta en desarrollo contribuye a la robustez del marco de trabajo SLAM-C++, permitiendo realizar productos multimedia de forma rápida y sencilla, garantizando la eficiencia y creatividad del usuario.
- La solución propuesta brinda una interfaz visual amigable y sencilla que facilita el trabajo de los desarrolladores para crear aplicaciones multimedia.
- SLAM-GraphicDesigner permitirá el trabajo en paralelo debido a la arquitectura que presenta permitiendo a los desarrolladores liberar nuevas versiones que serán anexadas a las anteriores.

Recomendaciones

Una vez concluido el análisis y diseño de la herramienta SLAM-GraphicDesigner se recomienda:

- Enriquecer los elementos conceptuales de la metáfora utilizada, sobre la base de conceptos de la programación orientada a objetos con la finalidad de facilitar el trabajo para quienes utilicen la herramienta.

- Realizar un estudio profundo sobre la posible integración de SLAM_GraphicDesigner con otro marco de trabajo para probar la fortaleza y la adaptabilidad de la herramienta.

- Mejorar las técnicas de dibujado incluyendo nuevas herramientas y perfeccionando las posibilidades de configuración y funcionamiento de las actuales.

- Implementar la herramienta SLAM_GraphicDesigner usando el diseño propuesto.

Referencias bibliográficas

1. **Díaz, C.C.** ITESO. ITESO. 1994; Available from: **http://iteso.mx**.
2. **PC, w.**, Comprendiendo el sonido multimedia. 23, 2003. No.119.
3. **Campos, C.** 2005; Available from: **http://www.portalfox.com**.
4. **Berrocoso, J.V.** UNEX, Usos educativos de la informática. 2003; Available from: **http://www.unex.es/**.
5. **Espinosa, A.M.**, Soluciones libres para aplicaciones multimedia basadas en C++. 2008, Universidad de las Ciencias Informáticas
6. **Incorporated, A.S.** 2009; Available from: **www.adobe.com**.
7. **Consortium, W.W.** Estándares y formatos libres. 2007; Available from: **http://www.w3.org**.
8. **Incorporated, A.S.** 2008; Available from: **www.adobe.com/es/**.
9. **Tecniber. 2009**; Available from: **http://www.bcn.es**.
10. Inkscape: Guía para un programa de dibujo vectorial Tavmjong Bah: 2009.
11. González, G. Herramienta de Animación en 2D. 2008; Available from: **http://ktoon-es.toonka.com**.
12. **Balboa, M.D.** LinWind. 2008; Available from: **http://www.dacostabalboa.com**.
13. Uira, in WorldLingo, Encyclopedia.
14. **Lliteras, O.F.,** Entrevista sobre HaEduc. 2010.
15. **Sierra, F.J.C.**, Enciclopedia Del Lenguaje C++. Vol. 2da Edición.
16. **Sierra, F.J.C.,** C/c++ Curso De Programación Vol. 2da Edición.
17. **Ivar Jacobson,** G.B., James Rumbaugh Proceso Unificado De Desarrollo De Software. 2000.
18. **Ivar Jacobson**, G.B.y.J.R., El Proceso Unificado De Desarrollo De Software. 2004.
19. **Larman, C.**, UML y Patrones: Introducción al análisis y programación orientada a objetos. 1999.
20. Sitio de descarga de software. 2007; Available from: **http://www.visual-paradigm.com**.
21. **Benjamin Herila**, P.S., Introducction to Qt, in Qt Lab. 2010.
22. **Garrido, S.A**., Introducción a Qt.Programación gráfica en C++ con Qt4. 2009.
23. **Anisley , Y**., Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia. Subsistema de gestión visual. . 2008, UCI: Ciudad de La Habana.
24. **LILLEY, C**. Scalable Vector Graphics (SVG). 2004; Available from: **www.w3.org**
25. **Morville, L.R.y.P.**, Information Architecture for the World Wide Web. 2002: 2da Edición.
26. **Catalán, M**. Metodologías de evaluación de interfaces gráficas de usuario. 2000; Available from: **http://www.r020.com.ar**.
27. **Addison Wesley, J.C.y.J.D.**, UML Components. 2001.

Bibliografía

Díaz, Carlos. 1994. *LA TECNOLOGIA MULTIMEDIA: Una Nueva Tecnología de Comunicación e Información. Características, concepciones y aplicaciones.*

Díaz, S, **2007**. *Ingeniería para productos con tecnología multimedia*, II Taller de Software Educativo y Multimedia.

Gehrmann, B. 1998. KDevelop - un Entorno de Desarrollo Integrado.

GONZÁLEZ, S. C. 2002. *Diseño teórico. El paradigma cuantitativo de la investigación científica.* 2002.

Hann, J.; Skinner G, 2005. *Guía de aprendizaje de los efectos gráficos de Flash*.

Herrera, Y, 2007**.** *Una Nueva Versión del Lenguaje de Modelación para Aplicaciones Educativas Multimedia*.

Peter Collin Publishing, 2002. *Dictionary of Multimedia*, 3era edición.

Planchette J, Summerfield M, 2006. *C++ GUI Programming with Qt4*, Prentice Hall.

Pressman, Roger S. 2005. *Ingeniería del Software*. *Un enfoque práctico.* La Habana: Félix Varela, 2005.

Rodríguez, Montero M, 2009. *Análisis y Diseño de un Sistema de Gestión y Control de Trámites para el Ministerio del Turismo (MINTUR).* Ciudad de La Habana.

Sánchez, M.A.M. 2004. *Metodologías De Desarrollo De Software.*

Glosario de términos

ActionScript: lenguaje de *scripts* muy parecido al *JavaScript* pero diseñado específicamente para la creación de multimedia interactivas utilizando *Flash*. Es orientado a objetos y semejante a los lenguajes de programación tradicionales tales como C++, Java entre otros.

Adobe Systems Incorporated: es una empresa privada de software con sede en San José (California, USA) fundada en diciembre de 1982. Destacada en el mundo del software por sus programas de edición de páginas web, video e imagen digital.

Animación: es un proceso utilizado para dar la sensación de movimiento a imágenes o dibujos. Para la realización de esta, existen numerosas técnicas que van más allá de los dibujos animados.

API: acrónimo de interfaz de programación de aplicaciones. Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción, son usados generalmente en las bibliotecas.

Caso de uso: es una descripción de un conjunto de secuencia de acciones, incluyendo variaciones que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

Fichero: un archivo informático que contiene un conjunto de información que se almacena en algún medio de escritura que permita ser leído o accedido por una computadora. Los archivos informáticos son los equivalentes digitales de los archivos en tarjetas, papel o microfichas del entorno de oficina tradicional, estos facilitan una manera de organizar los recursos usados para almacenar permanentemente información dentro de un computador.

***Framework*:** es una estructura que da soporte a otro proyecto de software, que puede ser organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con otros detalles de bajo nivel que provee un sistema funcional. Fuera de las aplicaciones en la informática, un *framework* puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo.

GTK: acrónimo de un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario.

Herramienta CASE: son programas diseñados para el modelado de sistemas, generalmente soportan varias metodologías de desarrollo como RUP, XP, MSF. Ejemplo de estas aplicaciones son *Visual Paradigm y Rational Rose*.

IDE: acrónimo de entorno de desarrollo integrado. Constituye un entorno de programación que ha sido empaquetado como un programa de aplicación, contiene un editor de código, compilador, depurador y constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Librería en programación: es una colección estándar de clases y funciones, escritas en el núcleo del lenguaje. La biblioteca estándar proporciona varios contenedores genéricos, funciones para utilizar y manipular esos contenedores, funciones objeto, cadenas y flujos genéricos.

Mac OSX: es el actual sistema operativo de la familia de ordenadores Macintosh. Es un sistema operativo basado en UNIX.

OS: acrónimo de sistema operativo.

Plugin: es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, su relevancia radica en que se convierten en una vía de expandir programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal.

SLAM-C++: acrónimo de Soluciones Libres para Aplicaciones Multimedia basadas en C++.

Software: es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados a estos.

Taxonomía: procede de los términos griegos "taxis", ordenación, y "nomos", norma. Aristóteles fue uno de los primeros en utilizar este término, para designar esquemas jerárquicos orientados a la clasificación de objetos científicos. El objetivo de la taxonomía es organizar los contenidos de manera lógica utilizando diversos criterios. Esto permite ordenar los contenidos en un sistema estructurado y eventualmente jerarquizado.

UI: es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora.

UML: es el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

***Widgets*: e**s un término que acatan las librerías Qt para referirse a todo componente de interfaz de usuario como botones, ventanas, cajas de texto, debido a que cada una tiene la capacidad de mostrarse individualmente en una ventana flotante si no le es definido otro *widgets* para incrustarse.

XML: extensible markup language (lenguaje de Marcas Extensibles). Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos de la misma manera que HTML.

Anexos

Anexo 1: Prototipos de interfaz

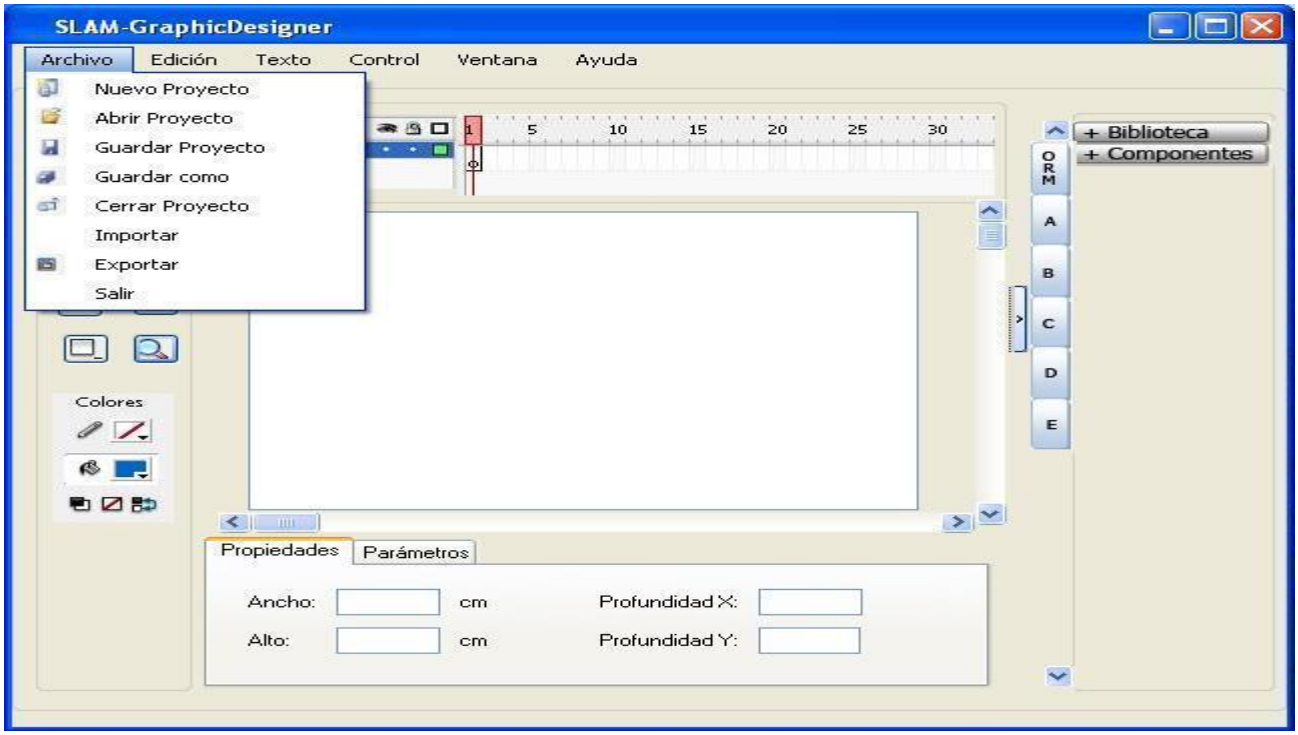


Figura 18: Pantalla general con el menú Archivo

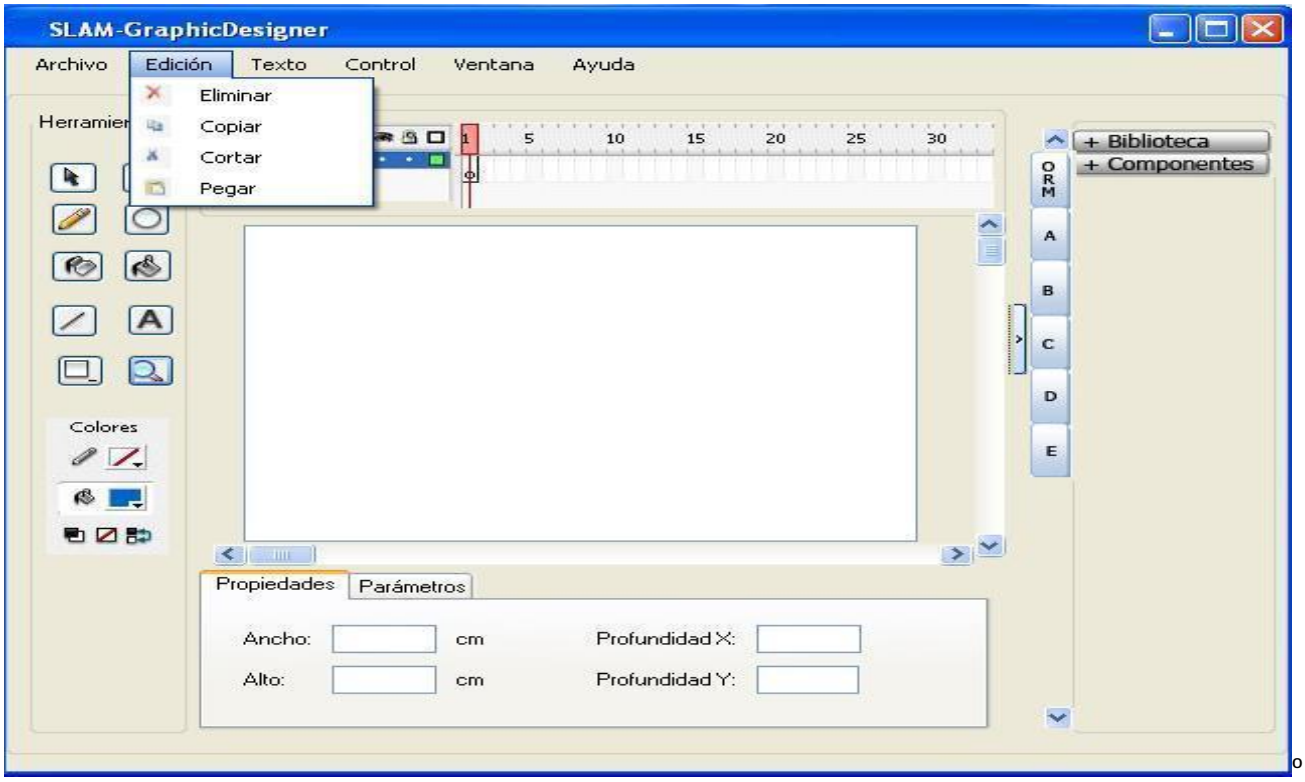


Figura 19: Pantalla general con el menú Edición

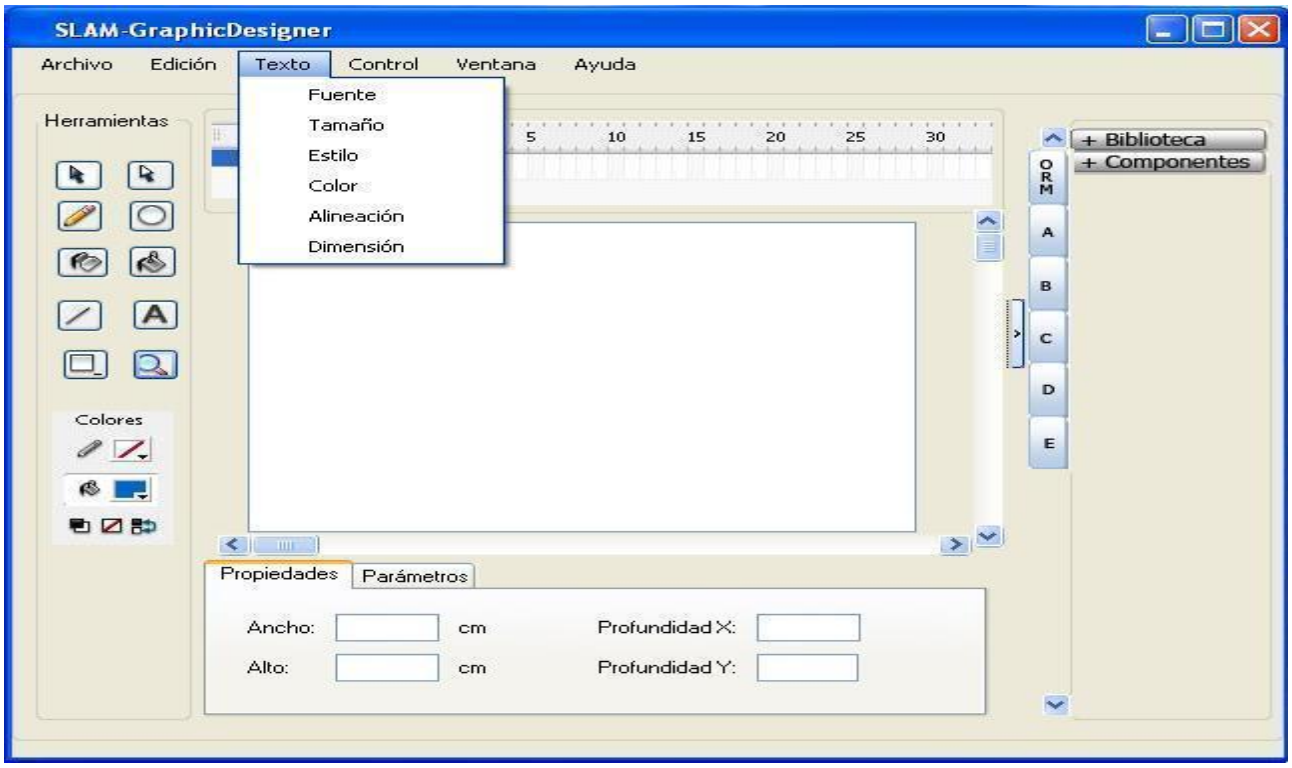


Figura 20: Pantalla general con el menú texto

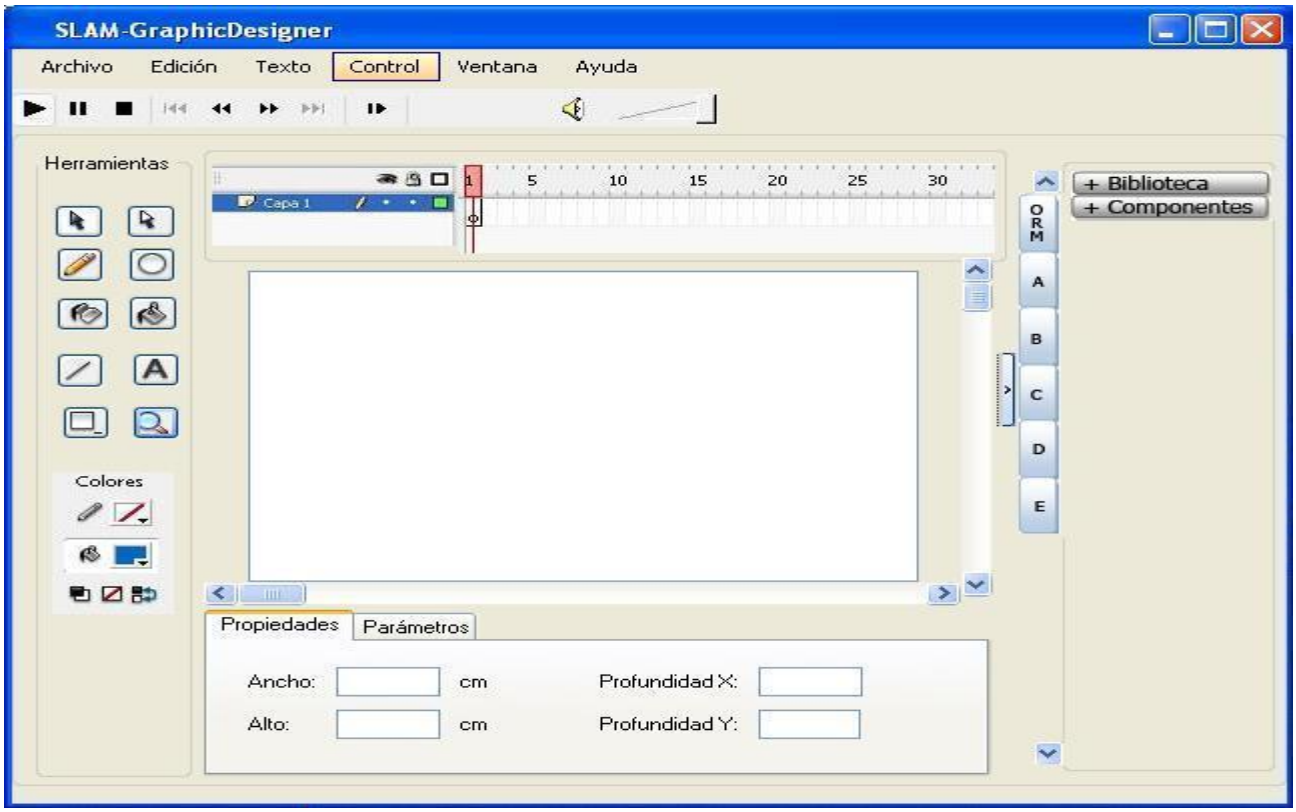


Figura 21: Pantalla general con la etiqueta Control activada.

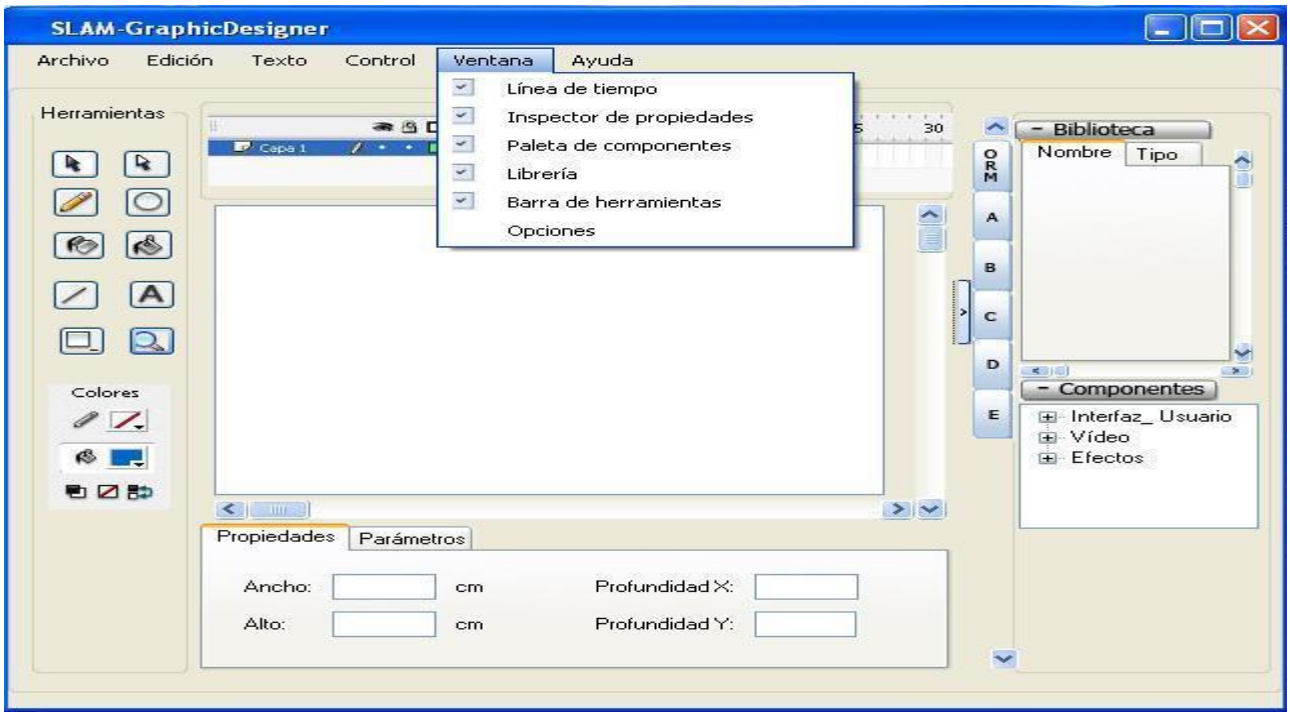


Figura 22: Pantalla general con el menú Ventana activado

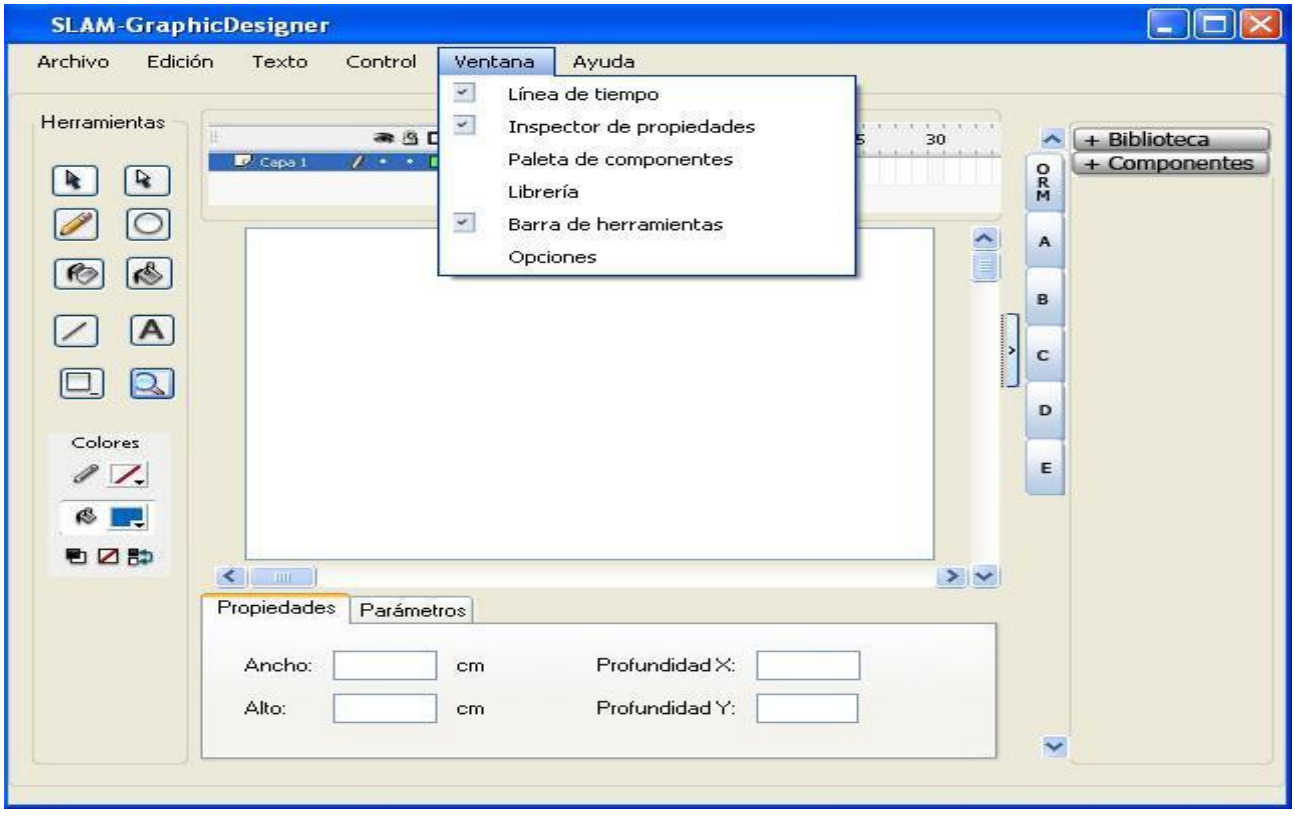


Figura 23: Pantalla general con la biblioteca y la paleta de componentes desactivadas

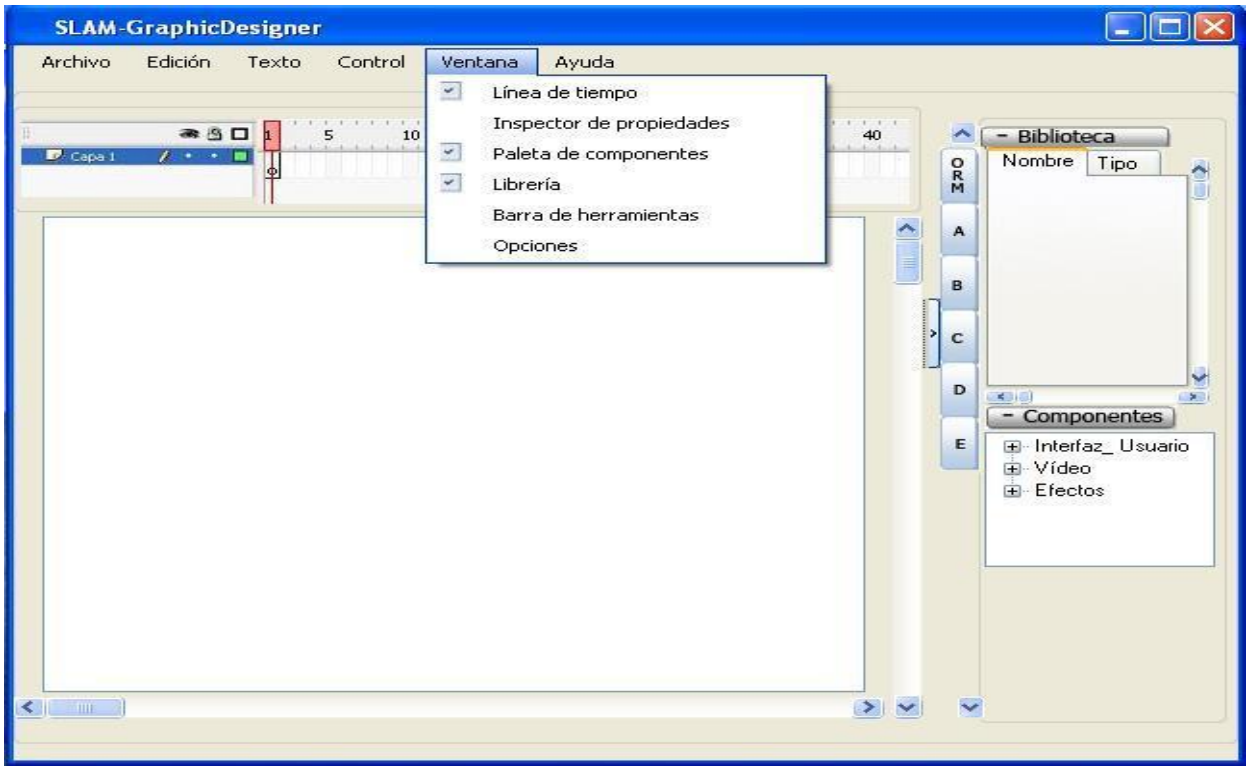


Figura 24: Pantalla general con las barra de herramientas y el inspector de propiedades desactivadas

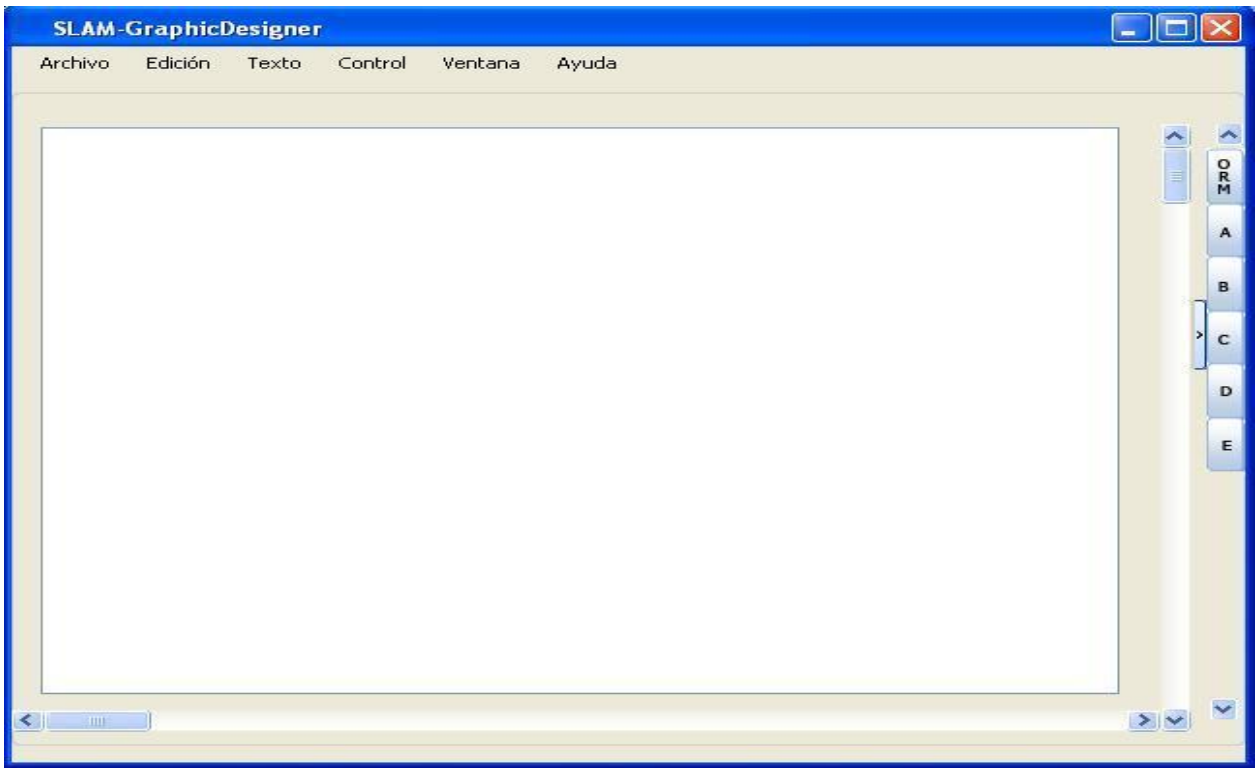


Figura 25: Pantalla con todas la barras desactivadas

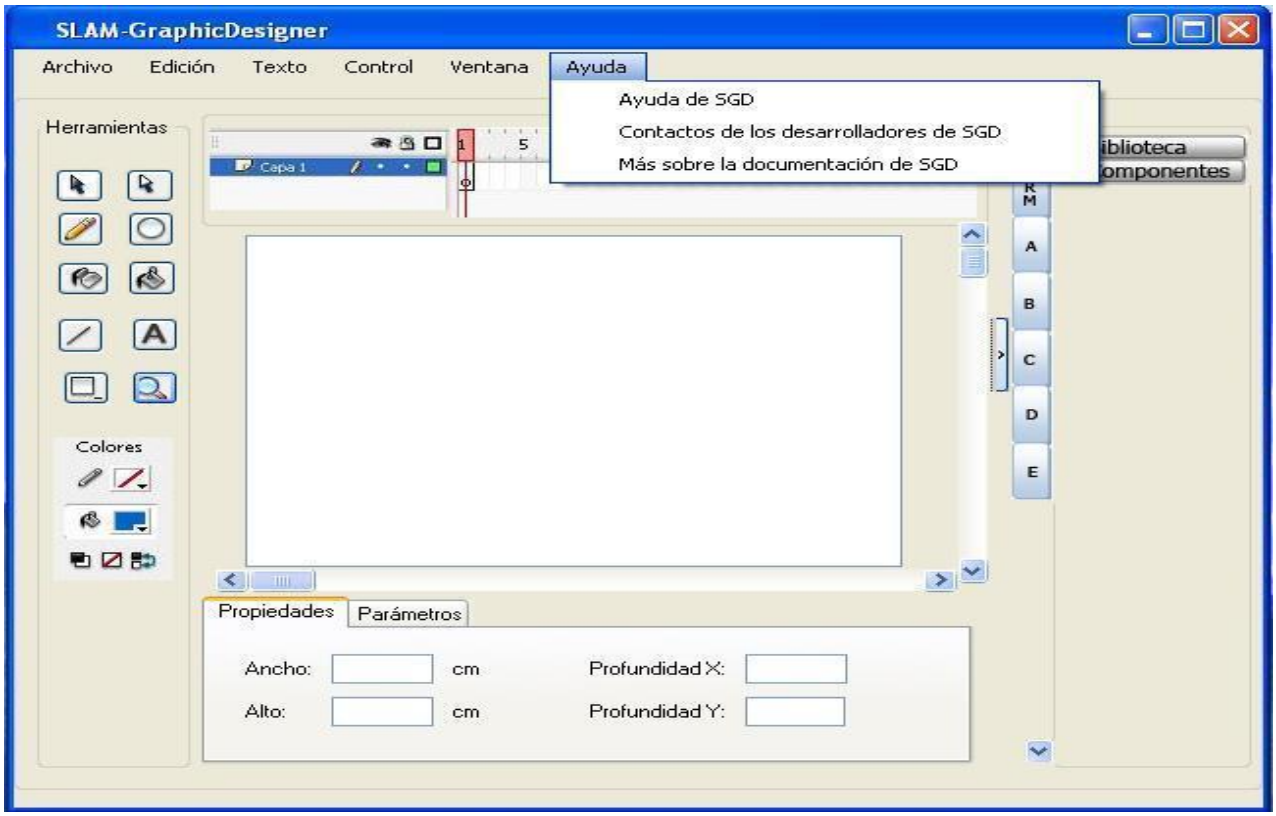


Figura 26: Pantalla general con el menú Ayuda activado

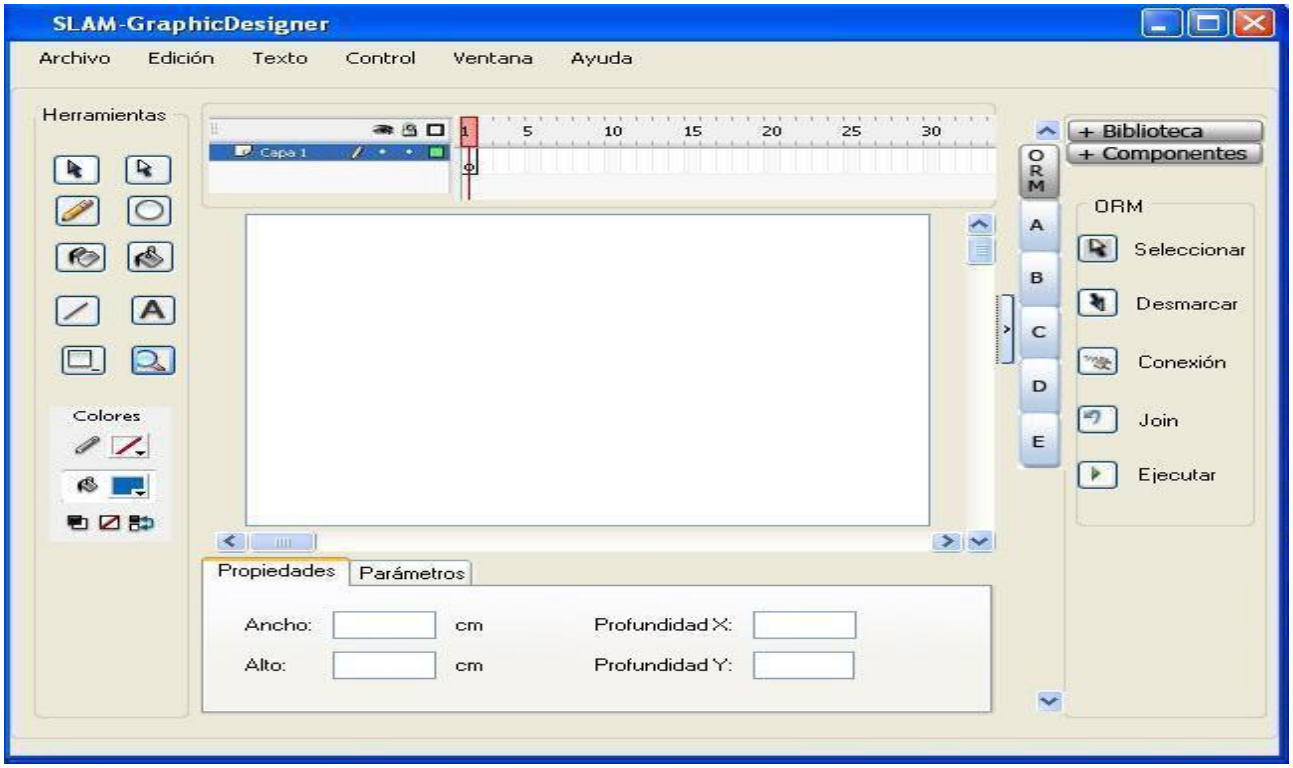


Figura 27: Pantalla general con los componentes del plugin ORM

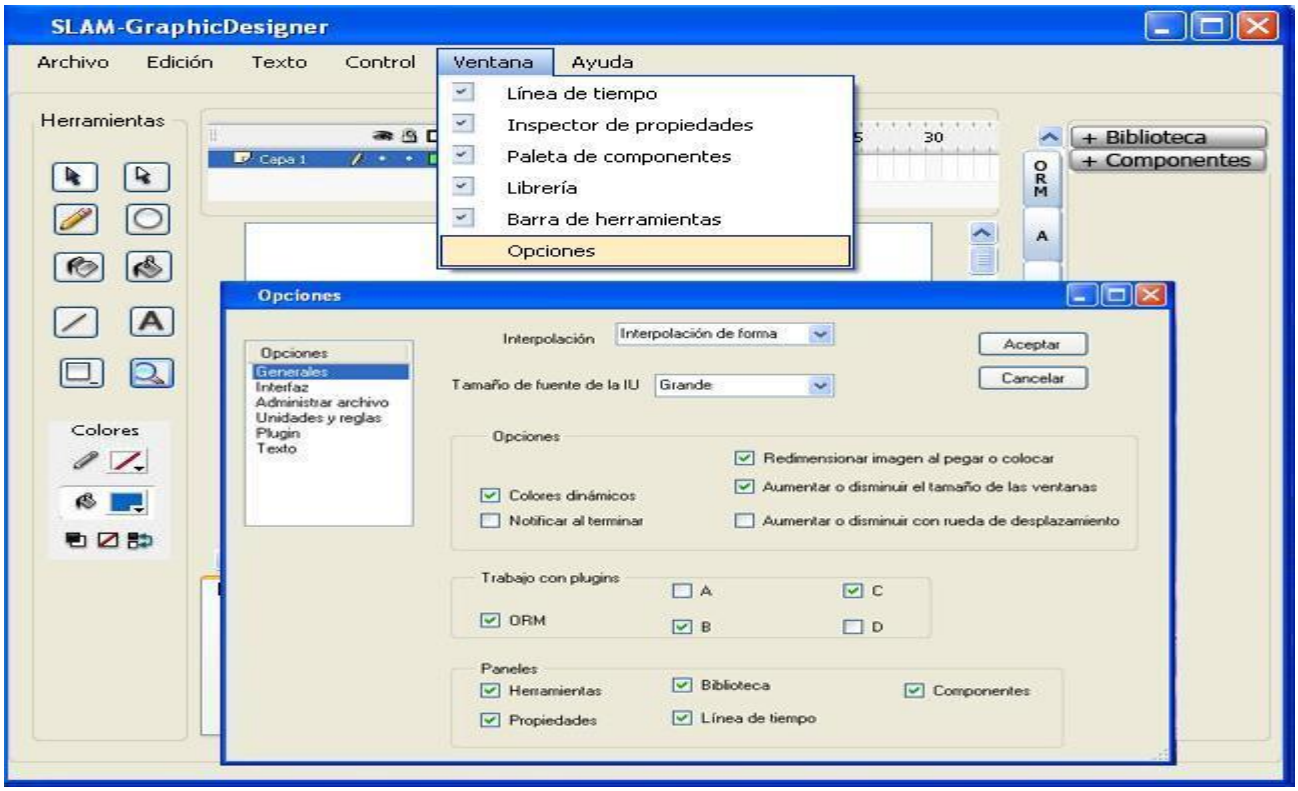


Figura 28: Pantalla general mostrando las opciones generales

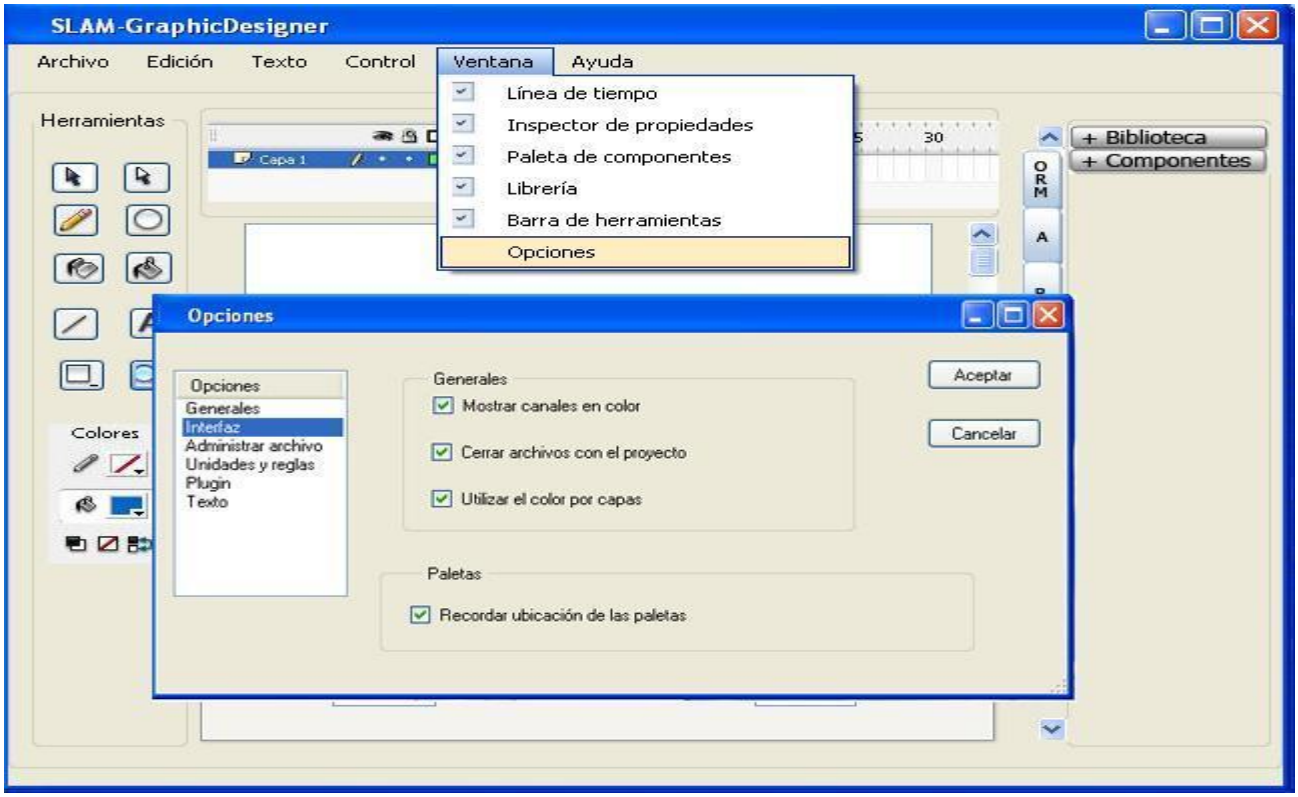


Figura 29: Pantalla general mostrando las opciones generales de la interfaz

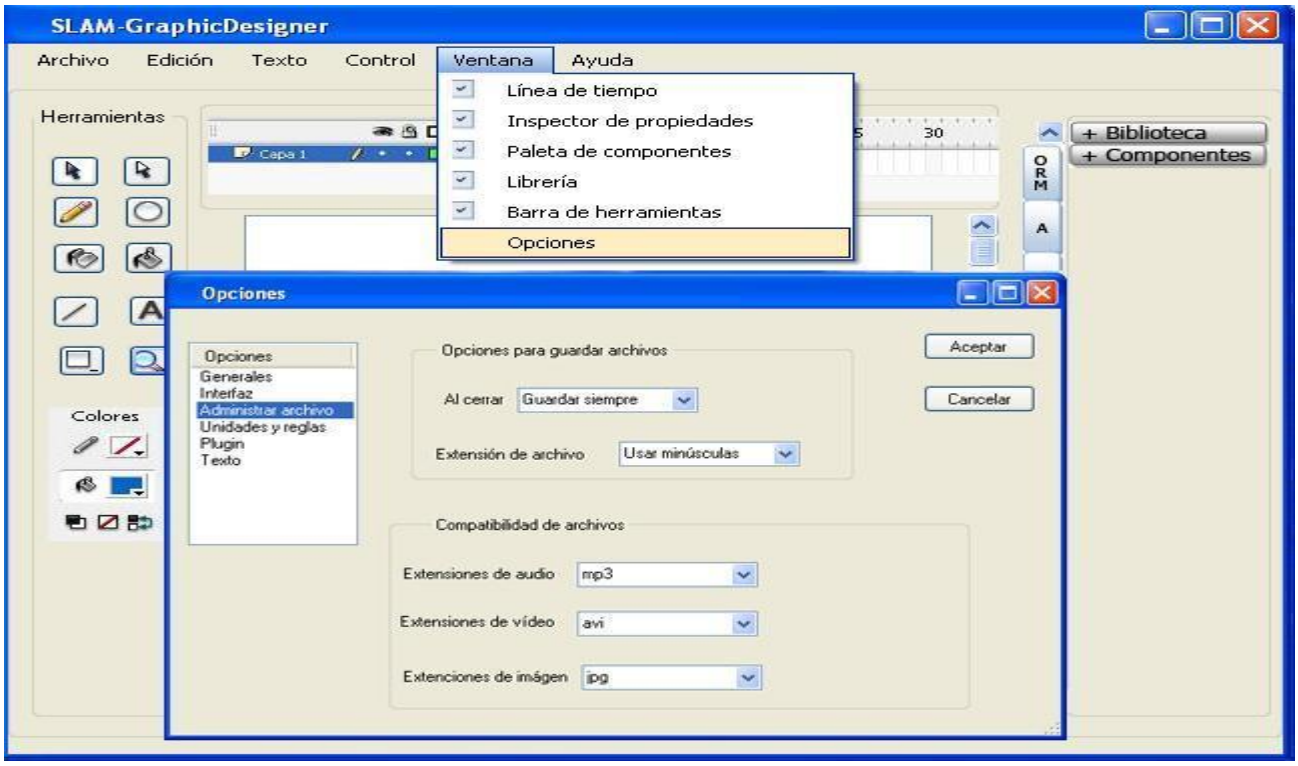


Figura 30: Pantalla general mostrando las opciones generales para administrar archivo

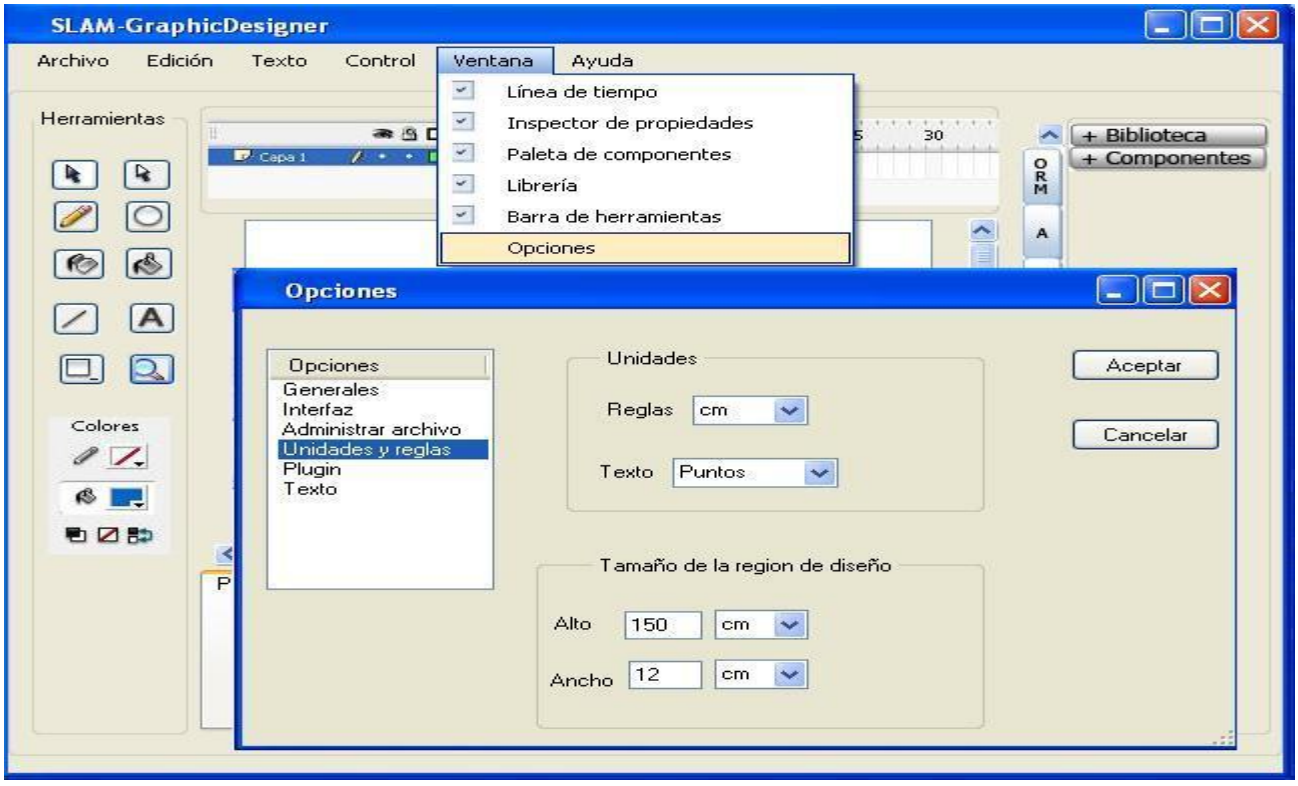


Figura 31: Pantalla general mostrando las opciones generales para las unidades y reglas

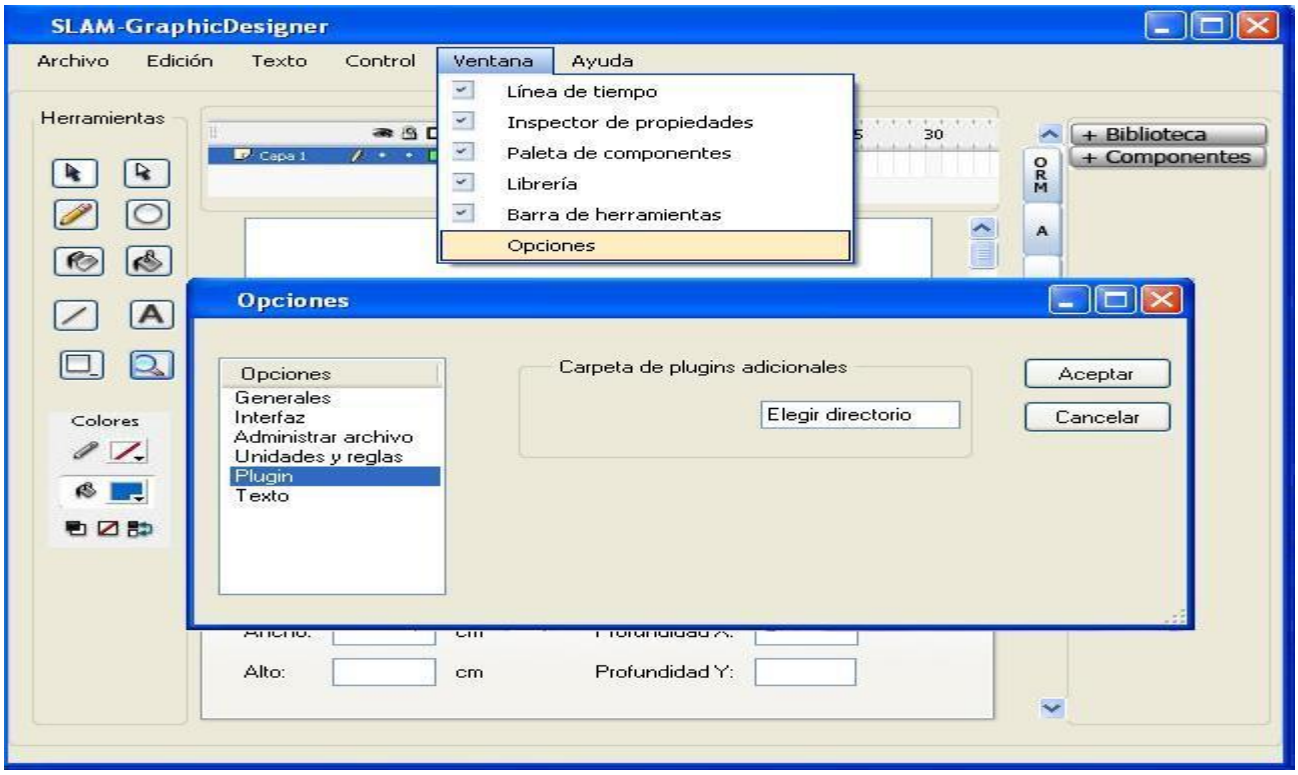


Figura 32: Pantalla general mostrando las opciones generales para los plugin

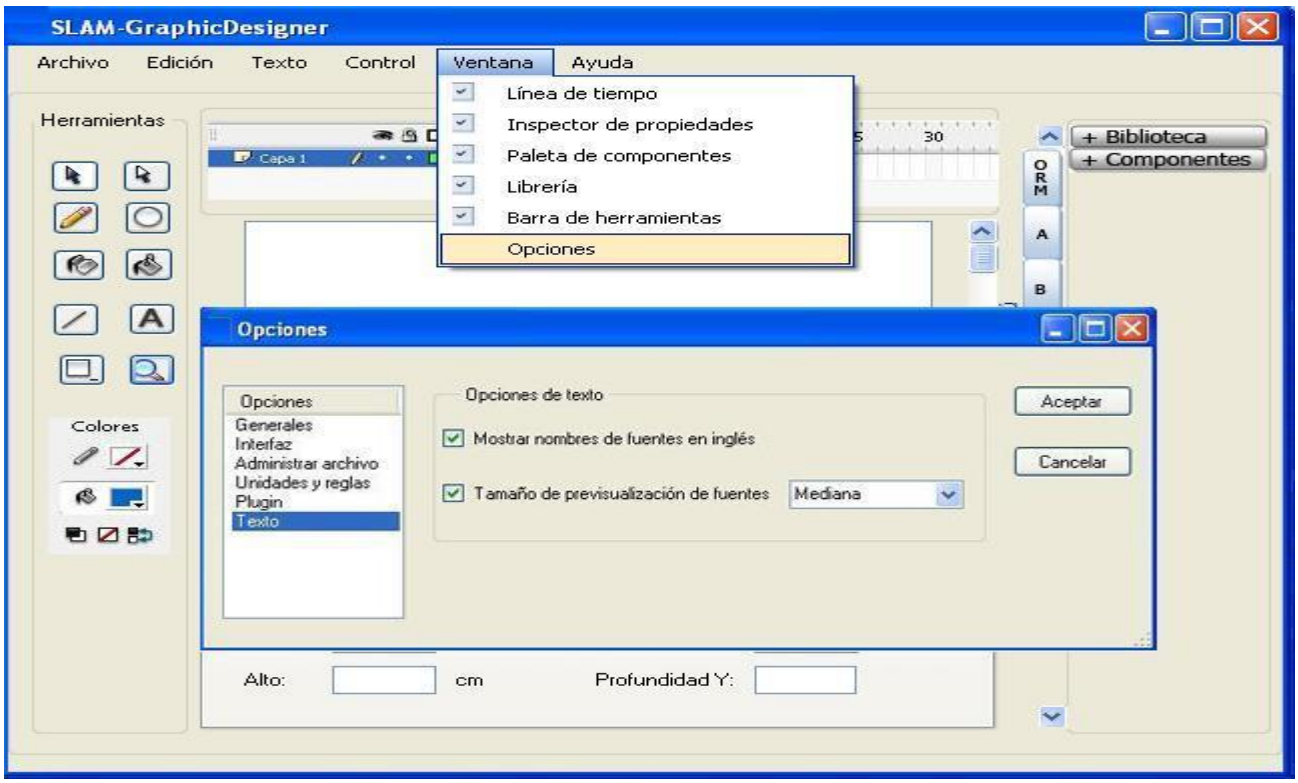


Figura 33: Pantalla general mostrando las opciones general del texto

Anexos 2 Diagramas clases de análisis

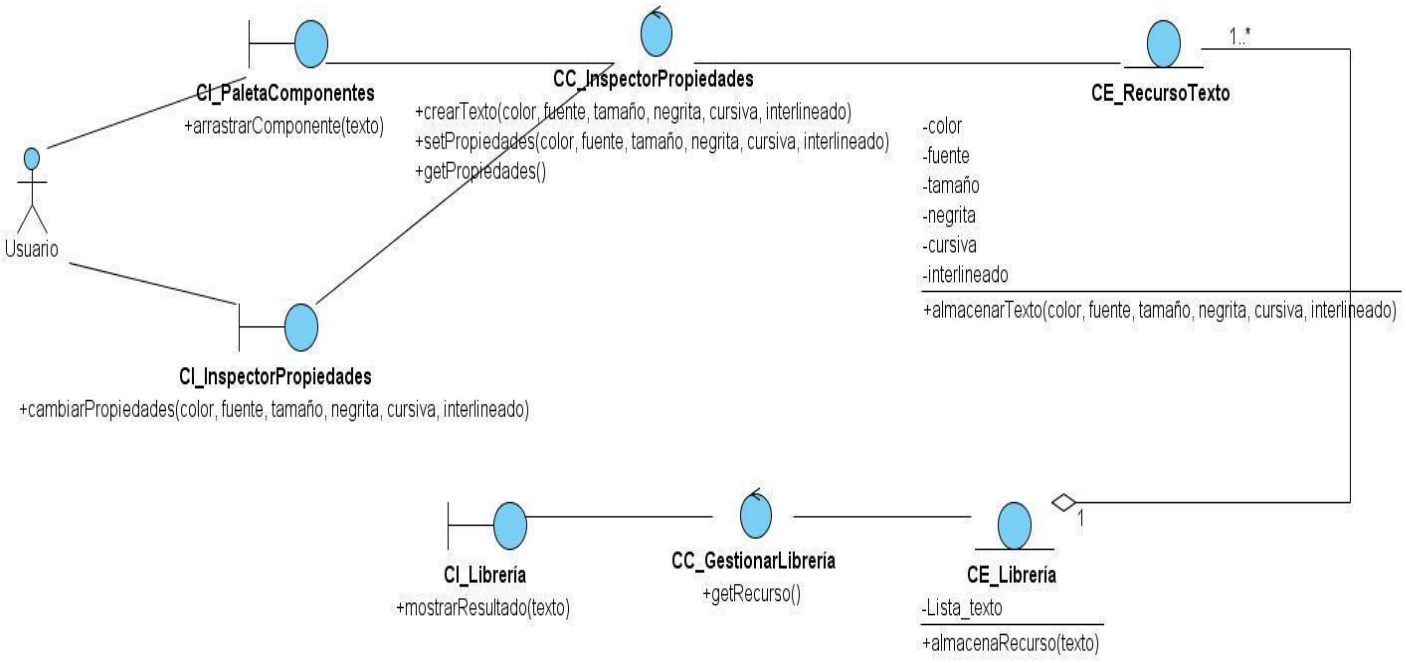


Figura 34: Diagrama de clases del análisis del caso de uso: Especificar texto.

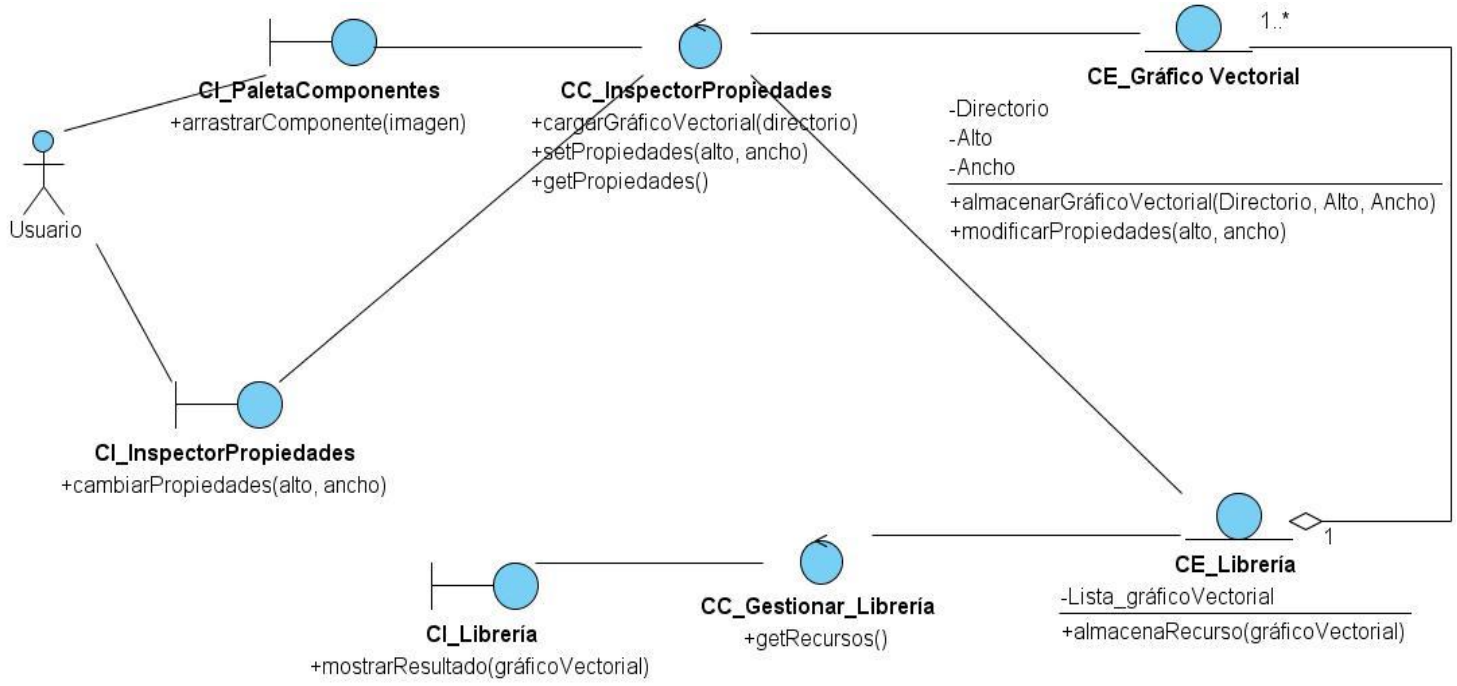


Figura 35: Diagrama de clases del análisis del caso de uso: Definir gráfico vectorial

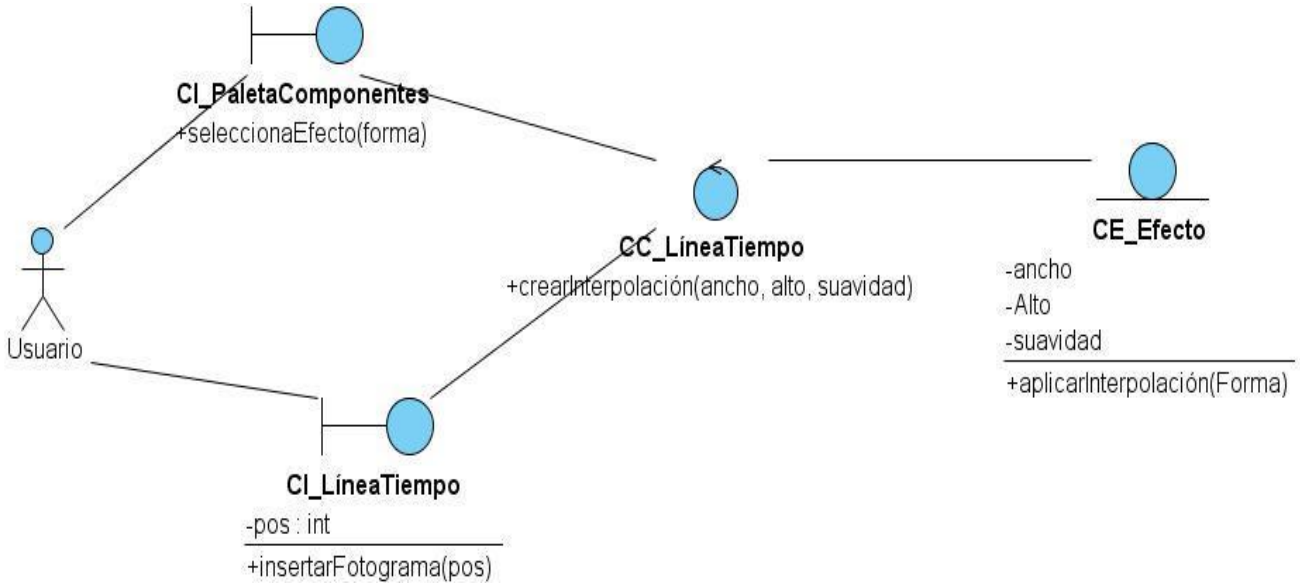


Figura 36: Diagrama de clases del análisis del caso de uso: Interpolación de forma

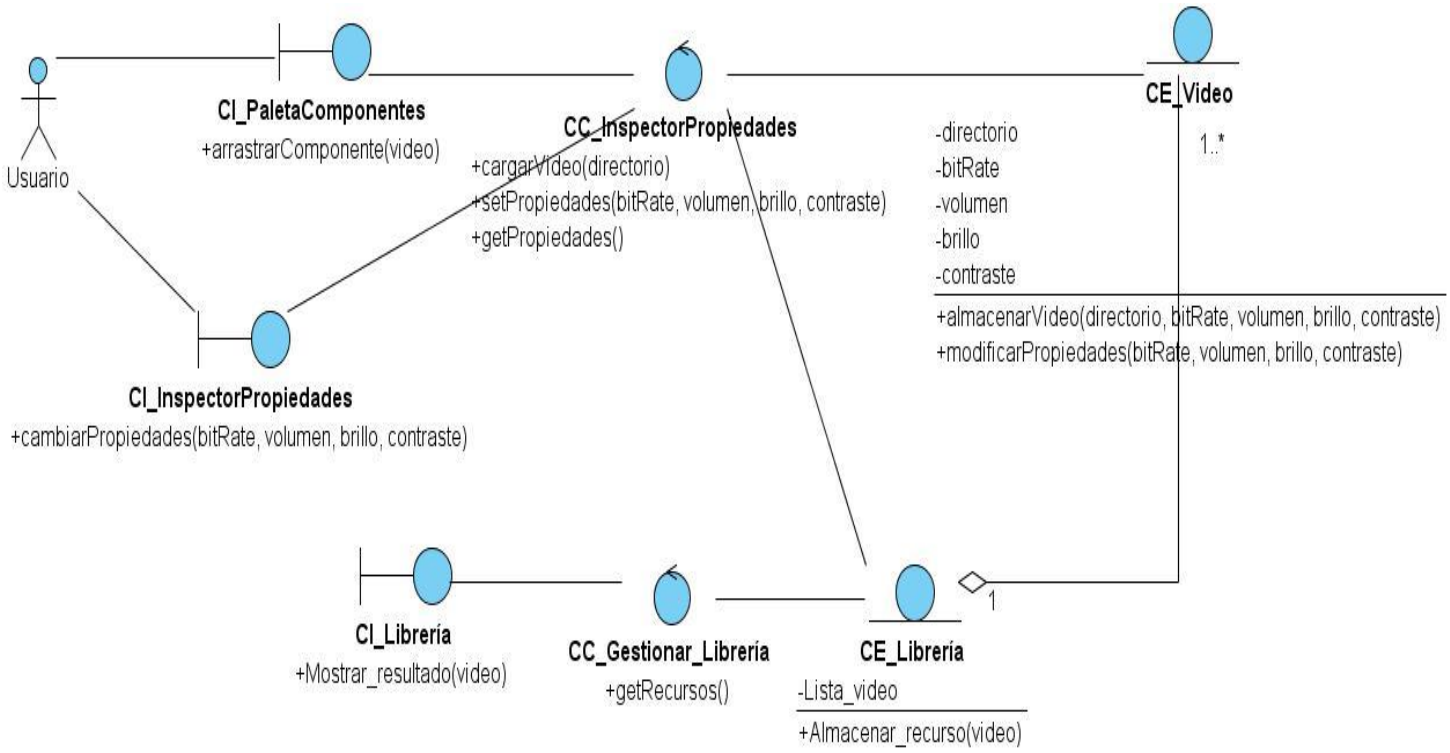


Figura 37: Diagrama de clases del análisis del caso de uso: Controlar video

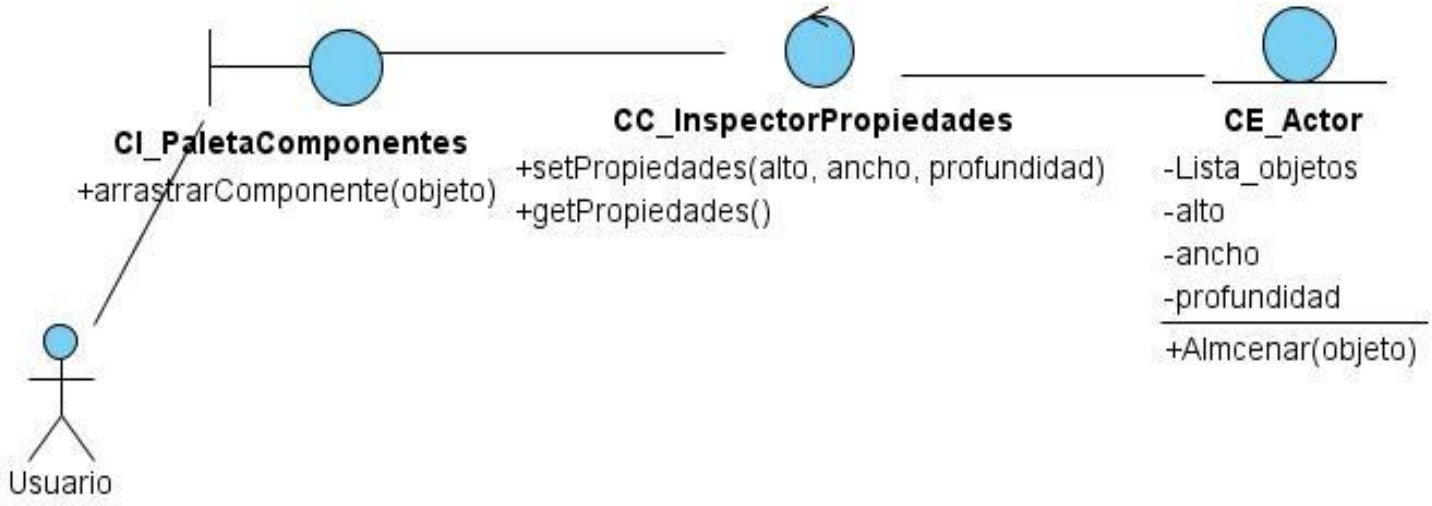


Figura 38: Diagrama de clases del análisis del caso de uso: Definir actor

Anexo 3 Subsistemas del diseño

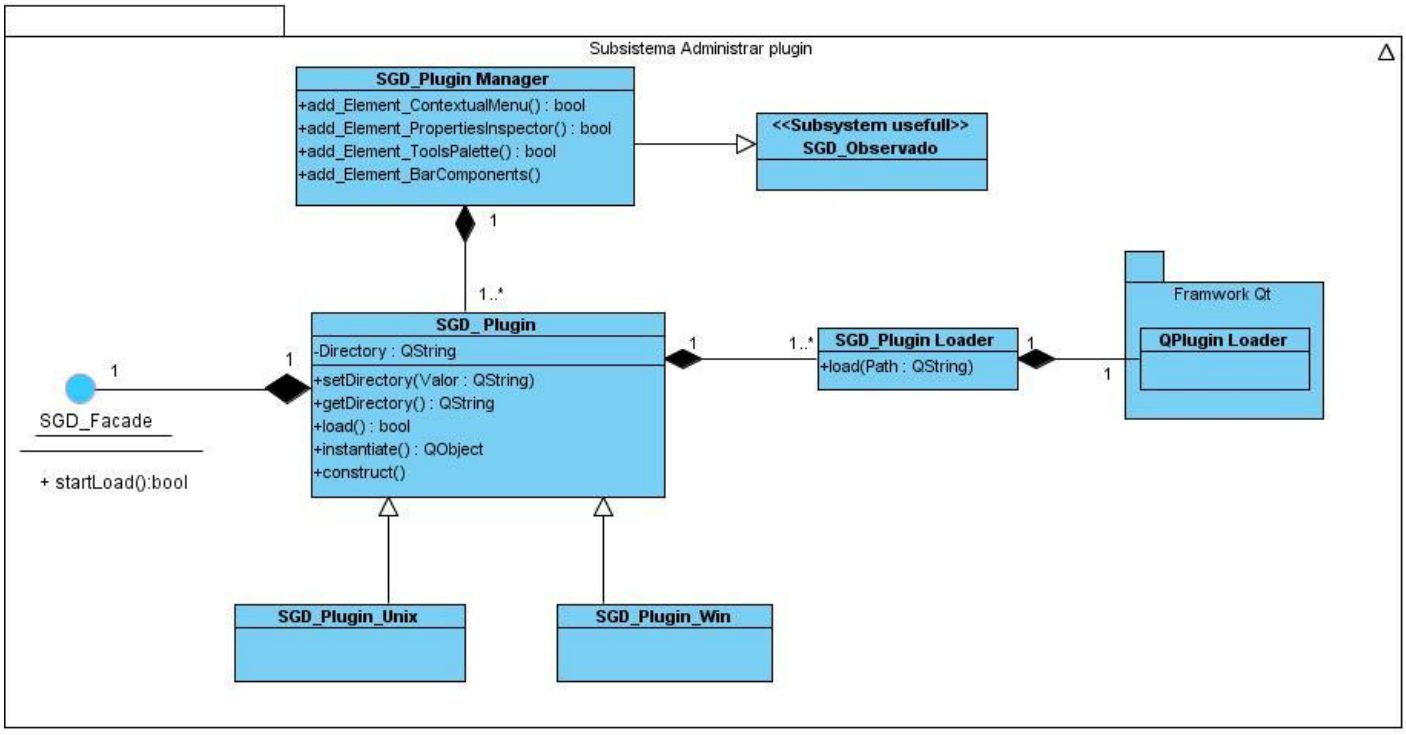


Figura 39: Diagrama de clases del subsistema: Administrar plugin

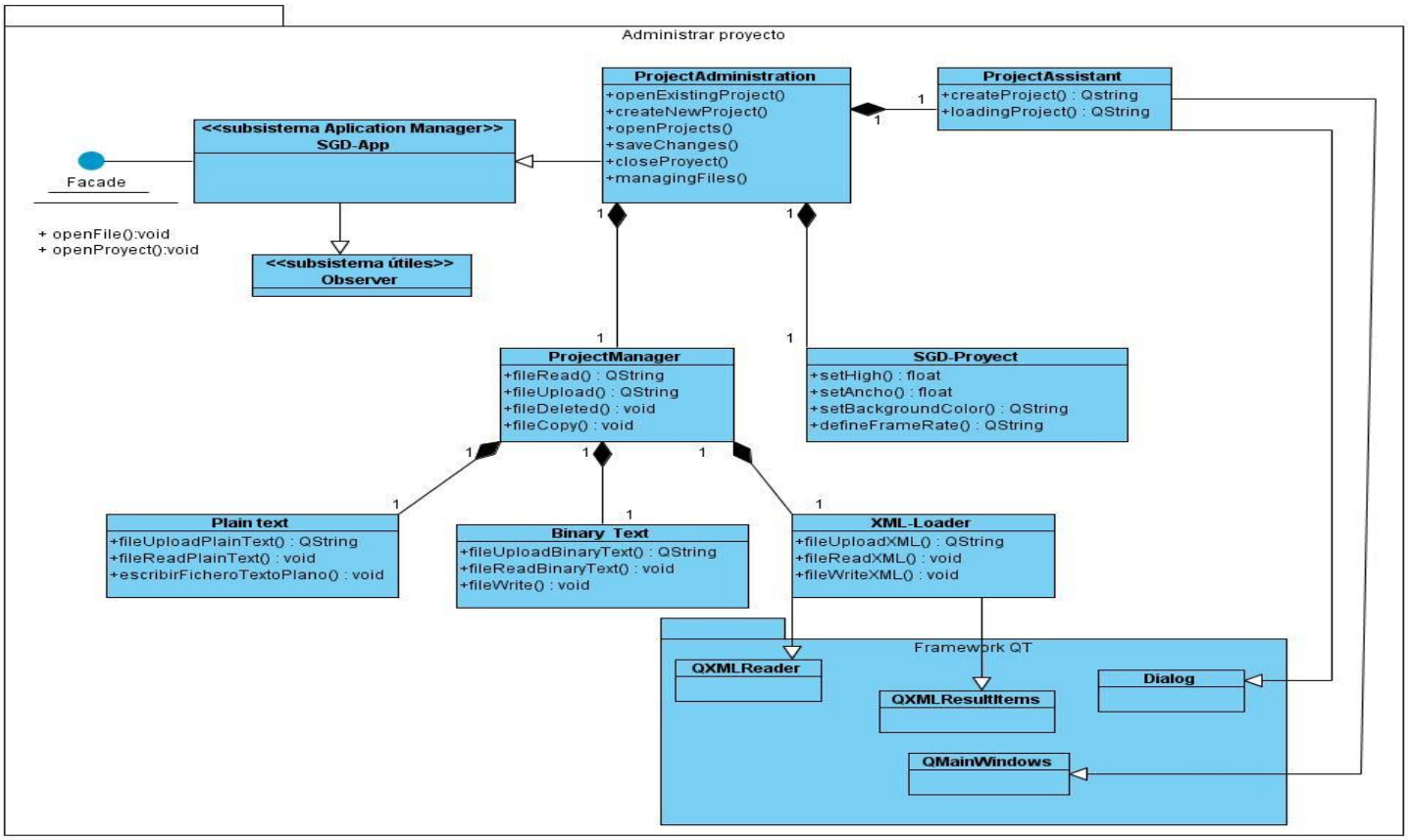


Figura 40: Diagrama de clases del subsistema: Administrar proyecto

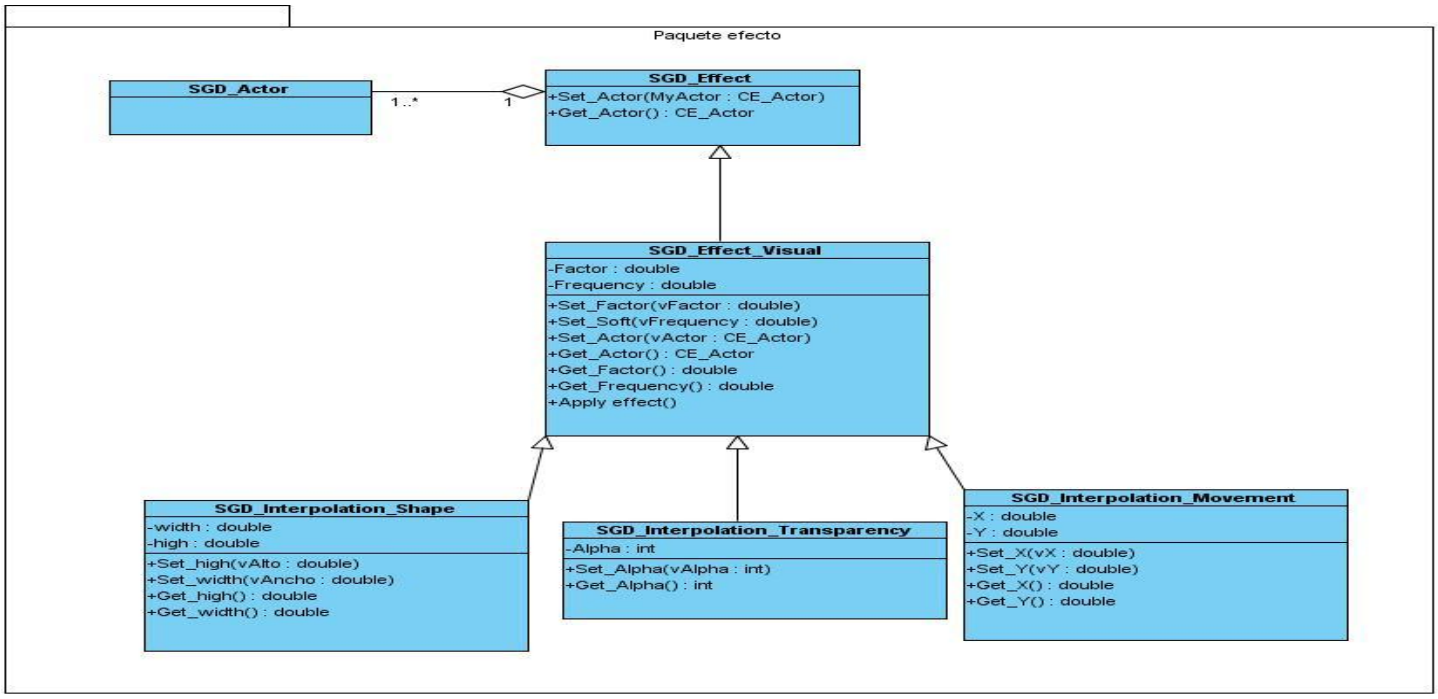


Figura 41: Diagrama de clases del paquete: Efectos

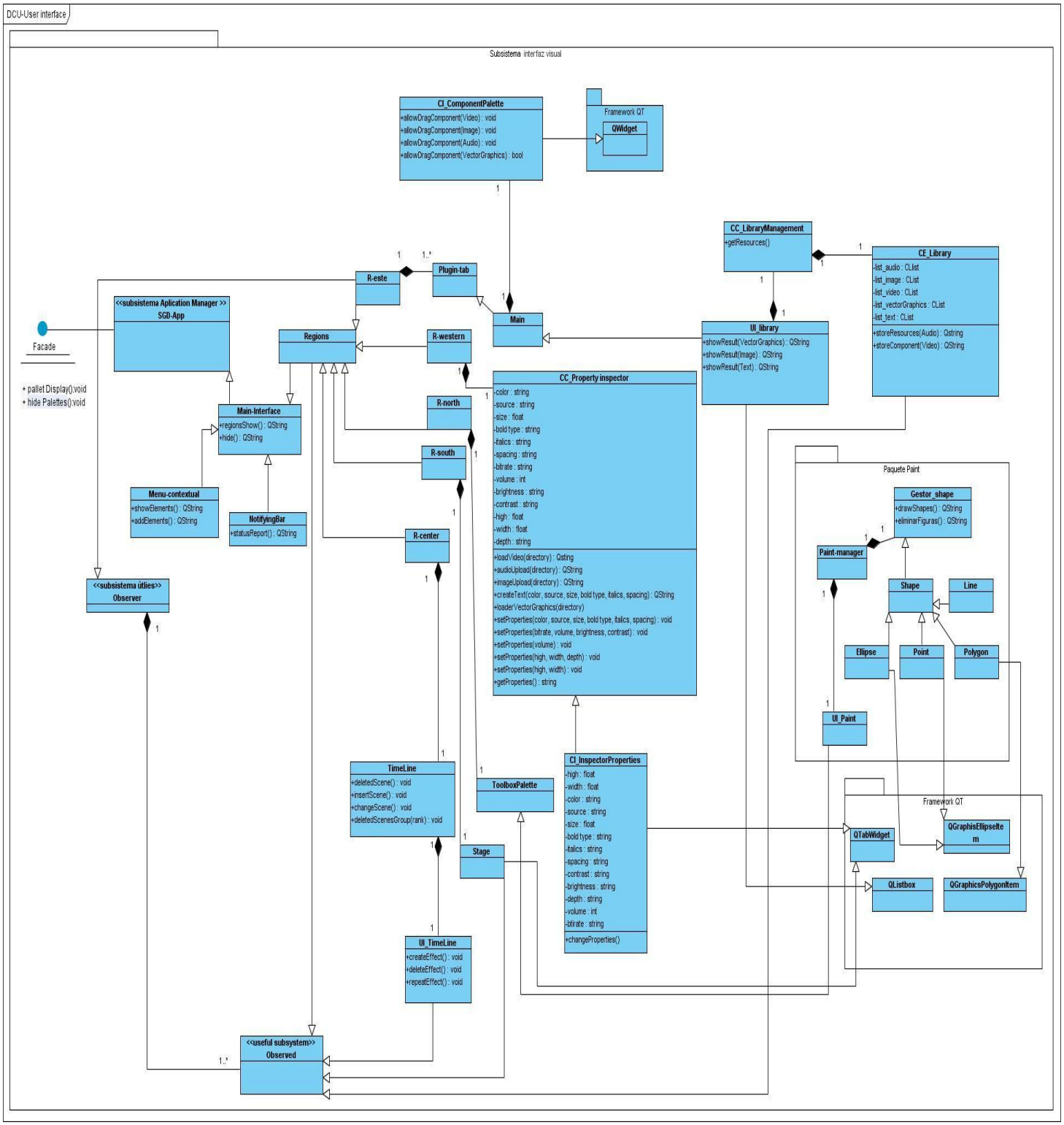


Figura 42: Diagrama de clases del subsistema: Interfaz de usuario

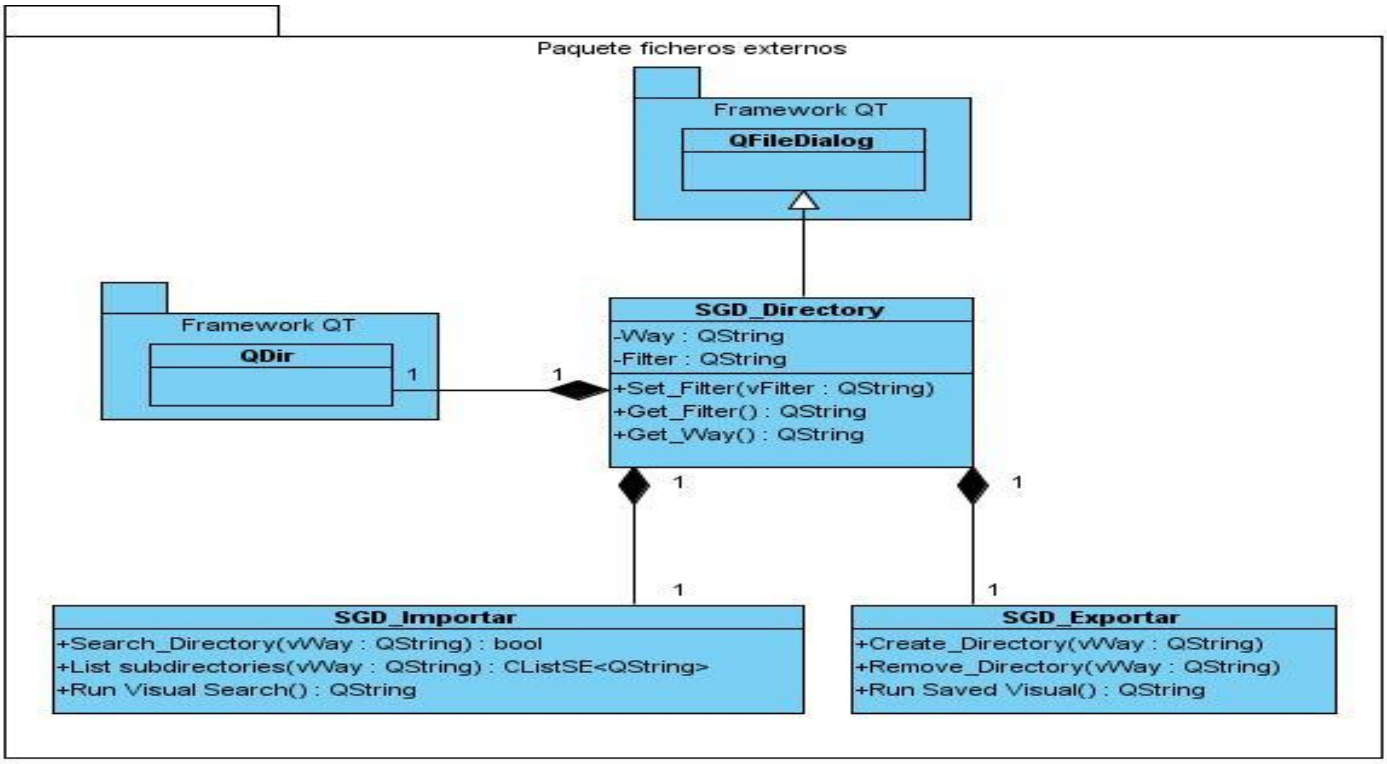


Figura 43: Diagrama de clases del paquete: Ficheros externos

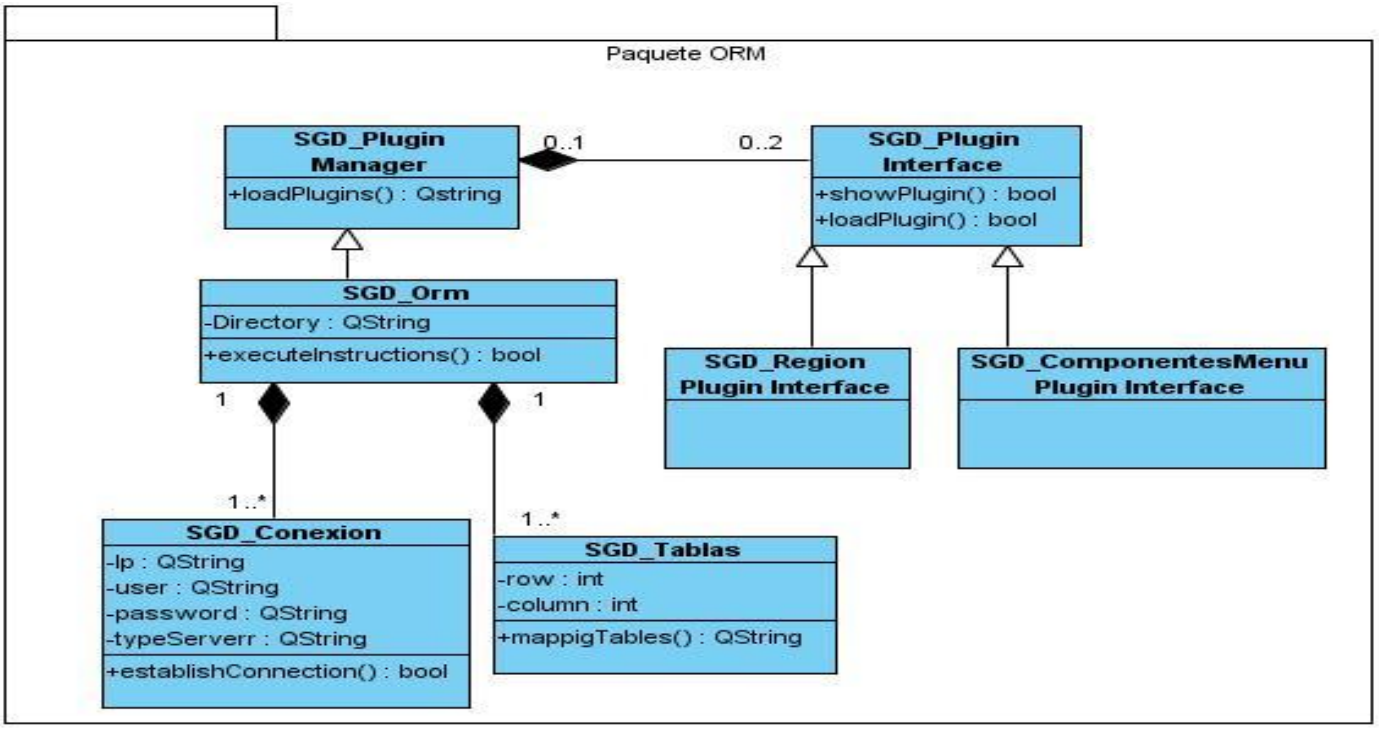


Figura 44: Diagrama de clases del paquete: ORM

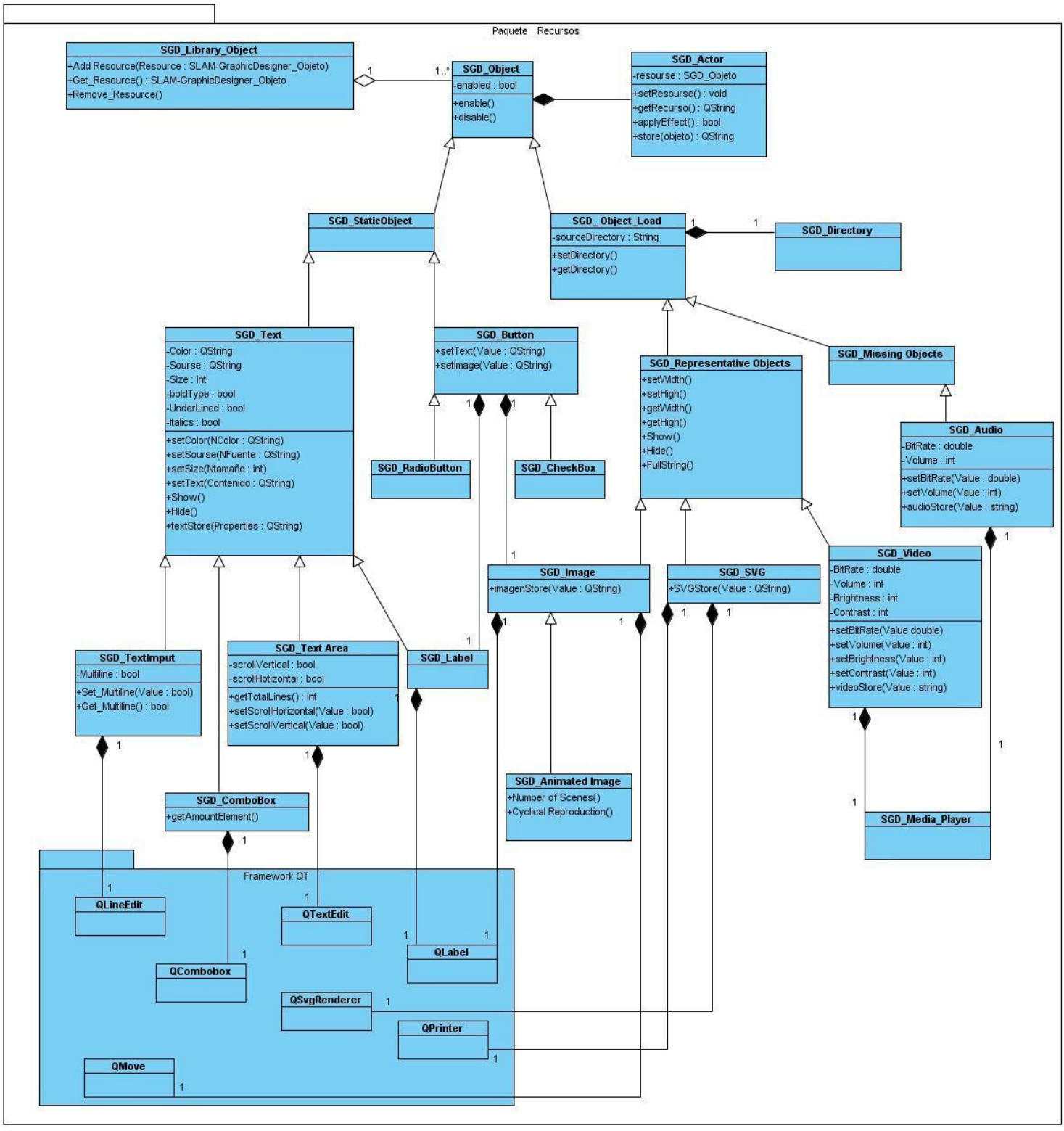


Figura 45: Diagrama de clases del paquetea: Recursos

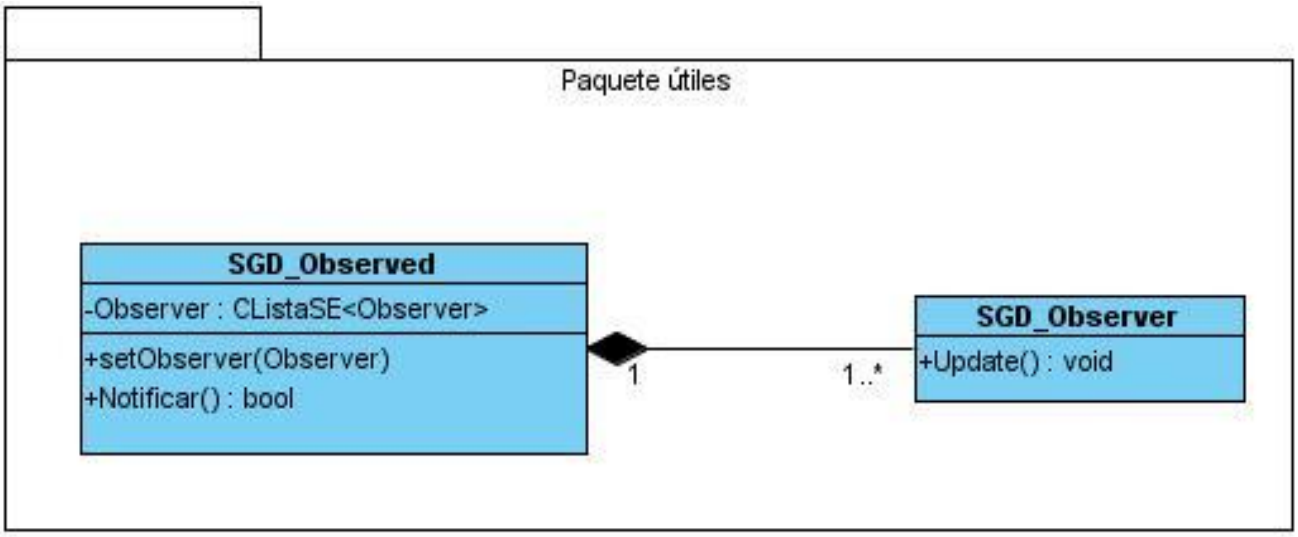


Figura 46: Diagrama de clases del paquetea: útiles

Anexo 4 Ejemplo de cómo queda el estándar del proyecto aplicado a un problema

Un proyecto presenta una serie de propiedades entre las que se encuentran: título del mismo que es Multimedia y nombre *Move*. Presenta además un ancho en la pantalla de 640 cm y un alto de 480 cm. Tiene una máscara de la misma se especifica el tipo de máscara, la dirección donde se encuentra guardada la misma y el nombre que esta tiene. Este proyecto también contiene una línea de tiempo, la que contiene velocidad de reproducción y número de fotogramas, estos contienen dos escenas de las que se especifica el id. La primera escena contiene un actor del que se define las propiedades que son alto, ancho y profundidad. También el actor contiene en este caso un objeto de tipo video, del que se detalla las propiedades que son velocidad de reproducción de 20.07 fotogramas / segundo y el brillo que es de 50. La segunda escena contiene otro actor del que se precisan las mismas propiedades. Además el actor contiene en esta segunda escena un objeto de tipo imagen, de la que se detallan las propiedades que son color de 40, la transparencia que va a ser de 85 y brillo que es de 55.

```
<gui verison="1.0">
   <proyect>
      <property>
        <tittle>Multimedia</title>
        <name>Movie</name>
        <width>640</width>
        <height>480</height>
        <fullscreen>false</fullscreen>
        <mask>
            <type>png</type>
            <path>d:/datos/proyect/multimedia1/mask</path>
            <filedname>mascara1</filedname>
        </mask>
      </property>
      <timeLine>
            <property>
                <birate>23</birate>
                <numberframes>3</numberframes>
            </property>
            <frames>
              <scene id ="0">
                <actor id ="0">
                    <property>
                        <width>0</width>
                        <height>0</height>
                        <x>0</x>
                        <y>0</y>
                    </property>
                    <object>
                        <property>
                            <type>video</type>
                             <birate>20.07</birate>
                             <bridge>50</bridge>
                         </property>
                    </object>
                </actor>
            </scene>
                <scene id ="1">
                <actor id ="1">
                    <property>
                        <width>0</width>
                        <height>0</height>
                        <x>0</x>
                        <y>0</y>
                    </property>
                    <object>
                      <property>
                          <type>image</type>
                          <transparency>85</transparency>
                          <color>40</color>
                          <bridges>55</bridges>
                      </property>
                    </object>
                </actor>
                </scene>
              </frames>
    </timeLine>
    </proyect>
</gui>
```

Figura 47: Estándar de proyecto

Anexo 5 Glosario de términos para el diseño

Conceptos:

película -> movie
forma -> shape
imagen > image
ilustración > picture
cabezal > bolster
escena -> Scene
escenario -> stage
actor -> actor
comportamiento > behavior
temporizador -> timer
tiempo -> time
observador -> observer
observado -> observed
ventana -> window
objeto -> object
señal -> signal
ranura -> slot
recurso -> recourse
recurso -> resort
polígono -> polygon
línea -> line
tabla -> table
punto -> point
fotograma -> frame
fichero -> file
reproductor -> player
impresora -> printer
librería -> library
contenedor -> container
gráfico -> graph
gráfico -> chart
factor ->factor

Propiedades:

duración -> duration
posición -> position
velocidad -> Speed
intérprete -> performer
rendimiento -> performance
omitido -> omitted
visual -> visual
representativo -> representative
alto -> high
ancho -> width
profundidad -> depth
superior -> upper
tamaño -> size
actual -> current
directorio -> Directory
camino -> path
apertura -> opening
grupo -> group
estático -> static
desplegable -> down
transparencia -> transparency
movimiento -> movement
movimiento -> motion
proyector -> projector
reflector -> searchlight
carga -> load
suavidad -> soft
sonido -> sound
efecto -> effect
frecuencia ->frequency

datos -> data
binario ->binary
operación ->operation
camino -> way

Funciones:

continuar -> continue
aguantar -> keep
incrementar -> increase
renderear -> renderer
renderizar -> rendering
realizar -> perform
reproducción -> rendering
conexión -> connection
reproducir -> play
detener -> stop
subir -> rise
bajar -> let down
ceder -> let up
descargar -> download
redimencionar -> resize
arrastrar -> drag
soltar -> drop
liberar -> release
notificar -> notify
avisar -> warn
navegar -> browse
editar -> edit
imprimir -> print
QT acrónimo de Quasar Technologies

agrupar -> grouping
fusionar -> merge
interpolar -> interpolate
interpolación -> interpolation
acceso a datos -> data access
modelo -> model
patrón -> pattern
construir -> construct
instanciar -> instantiate
aplicar -> apply
cargar -> load
leer -> read
crear ->create
copiar -> copy
buscar ->search
negrita ->bold type
cursiva ->italics
subrayada -> underlined
mostrar ->show
ocultar ->hide
publicar -> publish
exportar -> export
habilitar -> enable
deshabilitar -> disable

Anexo 6 Descripción de los casos de uso

Caso de uso	
CU	Especificar texto
Propósito	Definir la estructura del texto.
Actor: usuario	
Resumen: El caso de uso se inicia cuando el usuario crear un objeto de tipo texto y procede a definir las acciones y propiedades que este recurso tendrá. El sistema permite la realización de estas acciones finalizando así el caso de uso.	
Referencias	RF: 40,41,42,43,44,45,46,47,48
Prioridad	Crítico
Precondiciones	La barra de componentes debe estar activada.
Poscondiciones	El texto queda creado en el escenario.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario selecciona de la barra de componentes el elemento de texto que desee.	1.1- El sistema muestra la selección realizada.
2- El usuario arrastra el componente de texto deseado para el escenario.	2.1- El sistema muestra en el escenario y en la biblioteca el elemento arrastrado dando la posibilidad de definir sus propiedades (tipo alineación, estilo de letra, tamaño de letra, formato de letra, dimensiones del archivo texto).
3- El usuario define que propiedades desea modificar.	3.1- El sistema modifica las propiedades. 3.2-El sistema muestra las propiedades modificadas y culmina el caso de uso.

Tabla 7: Descripción del caso de uso del sistema: Especificar texto

Caso de uso	
CU	Definir gráfico vectorial
Propósito	Definir las propiedades del gráfico vectorial.
Actor: usuario	
Resumen: El caso de uso se inicia cuando el usuario carga un objeto de tipo gráfico vectorial y procede a definir las propiedades que este recurso tendrá. El sistema permite la realización de estas acciones finalizando así el caso de uso.	
Referencias	RF: 49,50,51,52
Prioridad	Crítico
Precondiciones	La barra de componentes debe estar activada.
Poscondiciones	El gráfico vectorial queda mostrado en el escenario.
Flujo Normal de Eventos	

Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario selecciona de la barra de componentes, el componente de la imagen.	1.1- El sistema muestra la selección realizada.
2- El usuario arrastra el componente de imagen para el escenario.	2.1- El sistema muestra en el escenario y en la biblioteca el elemento.
3- El usuario carga el gráfico vectorial.	3.1- El sistema muestra el gráfico vectorial cargado en el escenario y muestra sus propiedades (profundidad, dimensiones).
4- El usuario define que propiedades desea modificar.	4.1 -El sistema modifica las propiedades. 4.2-El sistema muestra las propiedades modificadas en el gráfico vectorial y culmina el caso de uso.

Tabla 8: Descripción del caso de uso del sistema: Definir gráfico vectorial.

Caso de uso	
CU	Gestionar escenas
Propósito	Definir las escenas de una película.
Actor: usuario	
Resumen: El caso de uso se inicia cuando el usuario inserta o modifica una escena a partir de la línea de tiempo. El sistema permite la realización de estas acciones finalizando así el caso de uso.	
Referencias	RF: 38,39
Prioridad	Crítico.
Precondiciones	La línea de tiempo debe estar activada.
Poscondiciones	La escena queda establecida en la película.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario da clic derecho sobre la línea de tiempo.	1.1-El sistema muestra las opciones de la línea de tiempo. Si selecciona 1. Eliminar escena (Sección eliminar escena). 2. Insertar escena (Sección insertar escena).
Sección 1 "Eliminar escena"	

2- El usuario selecciona la escena que desea eliminar.	2.1- El sistema eliminará la representación visual de la escena en la línea de tiempo. 2.2- El sistema reducirá la duración de la película. 2.3 Fin de la Sección 1. Regresa al paso 1 del Flujo Normal de Eventos.
Sección 2 "Insertar escena"	
3- El usuario escoge la opción insertar escena.	3.1- El sistema insertará una nueva escena en la línea de tiempo. 3.2 El sistema aumentará la duración de la película. Fin de la sección.

Tabla 9: Descripción del caso de uso del sistema: Gestionar escenas

Caso de uso	
CU	Configurar objetos
Propósito	Definir las propiedades y las acciones sobre un objeto.
Actor: usuario	
Resumen: El caso de uso se inicia cuando el usuario trabaja sobre el escenario con la herramienta selección luego de activar alguna de las opciones de la etiqueta edición que se muestran en un menú. El sistema permite la realización de estas acciones finalizando así el caso de uso.	
Referencias	RF:14,15,16,17,18
Prioridad	Crítico.
Precondiciones	Debe haber un objeto representado en el escenario.
Poscondiciones	La representación de objeto modificado en el escenario.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario seleccionará una de las opciones de la etiqueta edición mostrada por el sistema.	1.1- El sistema llamará a la selección correspondiente de acuerdo con la opción seleccionada. a. Eliminar (Sección Eliminar objetos). b. Copiar (Sección Copiar objetos). c. Cortar (Sección Cortar objetos). d. Pegar (Sección Pegar objetos).
Sección 1 "Eliminar objetos"	
2- El usuario seleccionará en el escenario mediante la herramienta selección lo que desee eliminar. Luego irá a la etiqueta Edición.	2.1- El sistema eliminará el objeto seleccionado. 2.2-El sistema muestra automáticamente los cambios en el escenario.

Flujos Alternos	
	3.0- El sistema mostrará un mensaje “Está seguro que desea eliminar.”.
4- El usuario aceptará o no el mensaje lanzado por el sistema.	4.1- El sistema eliminará o no según la selección del usuario.
Sección 2 “Copiar objetos”	
5- El usuario seleccionará en el escenario lo que desee copiar.	5.1- El sistema permitirá copiar lo seleccionado anteriormente. 5.2- El sistema muestra automáticamente los cambios en el escenario.
Sección 3 “Cortar objetos”	
6- El usuario seleccionará en el escenario lo que desee cortar.	6.1- El sistema permitirá cortar lo seleccionado anteriormente. 6.2- El sistema muestra automáticamente los cambios en el escenario.
Sección 4 **“Pegar objetos”**	
7- El usuario seleccionará en el escenario lo que desee pegar.	7.1- El sistema permitirá pegar lo seleccionado anteriormente. 6.2- El sistema muestra automáticamente los cambios en el escenario.

Tabla 10: Descripción del caso de uso del sistema: Configurar objetos

Caso de uso	
CU	Controlar video
Propósito	Permitir realizar acciones con el video.
Actor: usuario	
Resumen: El caso de uso se inicia cuando el usuario desea cargar un video, definir las acciones y propiedades que este recurso tendrá. El sistema permite cargar el audio y realizar acciones con el mismo, finalizando así el caso de uso.	
Referencias	RF: 64,65,66,67,68,69
Prioridad	Secundario
Precondiciones	La barra de componentes debe estar activada.
Poscondiciones	El video queda cargado en el escenario.
Flujo Normal de Eventos	
Sección “General”	
Acción del actor	**Respuesta del sistema**
1- El usuario selecciona de la barra de componentes el	1.1- El sistema muestra la selección realizada.

componente de video.	
2- El usuario arrastra el componente de video para el escenario.	2.1- El sistema muestra en el escenario y en la biblioteca.
3- El usuario carga el video desde un directorio.	3.1- El sistema muestra el video cargado en el escenario y muestra sus propiedades (control de volumen y dimensiones).
4- El usuario define las propiedades que desea modificar.	4.1- El sistema modifica las propiedades del video. 4.2- El sistema muestra las propiedades del video modificadas y culmina de esta forma el caso de uso.

Tabla 11: Descripción del caso de uso del sistema: Controlar video

Caso de uso	
CU-7	Visualizar paneles.
Propósito	Permitir que un panel determinado sea visible en el escenario.
Actores: usuario	
Resumen: El caso de uso inicia cuando el usuario decide visualizar un panel en el escenario. El sistema debe permitir mostrar o no el panel, finalizando de esta forma el caso de uso.	
Referencias	RF: 94,95,96,97
Prioridad	Crítico.
Precondiciones	
Poscondiciones	Queda visualizado en el escenario el panel deseado.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**
1- El usuario selecciona la etiqueta ventana y marca el panel que desea visualizar.	1.1- El sistema muestra los paneles - Si desea visualizar la barra de herramientas ver sección "Visualizar barra de herramientas". - Si desea visualizar el inspector de propiedades ver sección "Visualizar inspector de propiedades". - Si desea visualizar la biblioteca ver sección "Visualizar biblioteca". - Si desea visualizar la barra de componentes ver sección "Visualizar barra de componentes".
Sección 1 "Visualizar barra de herramientas"	

	1.1- El sistema activa la barra de herramientas. 1.2- El sistema modifica automáticamente el escenario. 1.3- Fin de la sección 1. Regresa al paso 1 del flujo normal de eventos.
Sección 2 "Visualizar inspector de propiedades"	
	2.1- El sistema activa el inspector de propiedades. 2.2- El sistema modifica automáticamente el escenario. 2.3- Fin de la sección 2. Regresa al paso 1 del flujo normal de eventos.
Sección 3 "Visualizar biblioteca"	
	3.1- El sistema activa el inspector de propiedades. 3.2-El sistema modifica automáticamente el escenario. 3.3-Fin de la sección 3. Regresa al paso 1 del flujo normal de eventos.
Sección 4 "Visualizar barra de componentes"	
Acción del actor	**Respuesta del sistema**
	4.1- El sistema activa el inspector de propiedades. 4.2-El sistema modifica automáticamente el escenario. 4.3-Fin de la sección 4.

Tabla 12: Descripción del caso de uso del sistema: Visualizar paneles

Caso de uso	
CU-7	Definir actor.
Propósito	Delimitar propiedades de un actor.
Actores: usuario	
Resumen: El caso de uso inicia cuando el usuario decide visualizar un actor en el escenario, definiendo las propiedades de este. El sistema debe permitir accionar con el actor, finalizando de esta forma el caso de uso.	
Referencias	RF: 70,71,72,73
Prioridad	Crítico.
Precondiciones	Debe estar la paleta de componentes activada.
Poscondiciones	Queda visualizado en el escenario el actor con todas las acciones asignadas a este.
Flujo Normal de Eventos	
Sección "General"	
Acción del actor	**Respuesta del sistema**

1- El usuario selecciona en la barra de componentes un objeto determinado.	1.1- El sistema muestra la selección realizada.
2- El usuario arrastra el objeto para el escenario convirtiéndose este en un actor.	2.1- El sistema muestra el objeto referenciando en la biblioteca y las propiedades de este automáticamente en el escenario (tiempo de visualización, dimensiones, profundidad).
3- El usuario selecciona propiedades a modificar del actor.	3.1- El sistema modifica las propiedades del actor. 3.2- El sistema muestra automáticamente como queda el actor en el escenario y culmina de esta forma el caso de uso.

Tabla 13: Descripción del caso de uso del sistema: Definir actor

Anexo 7 Diagramas de secuencia del diseño

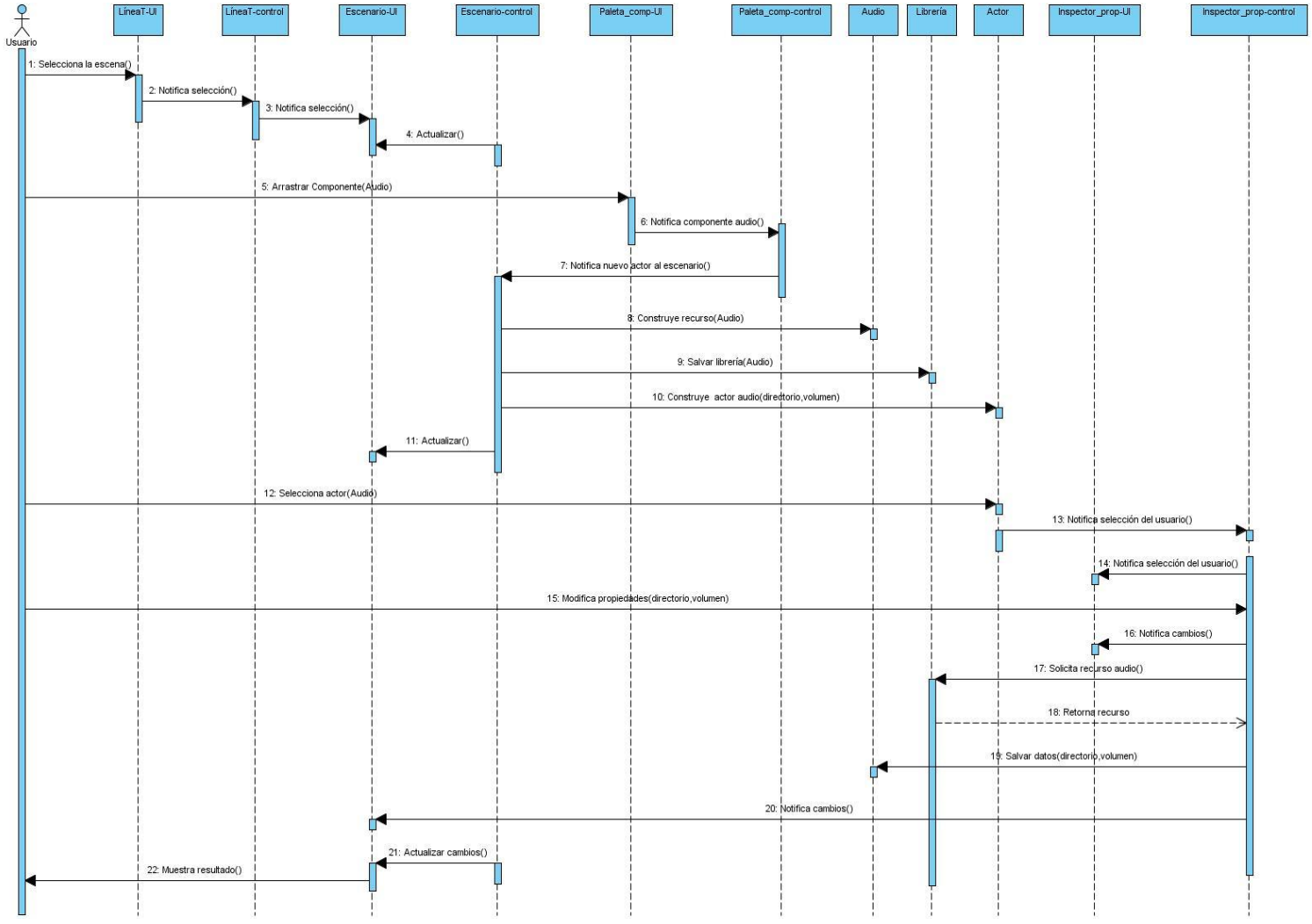


Figura 48: Diagrama de secuencia del caso de uso: Controlar audio.

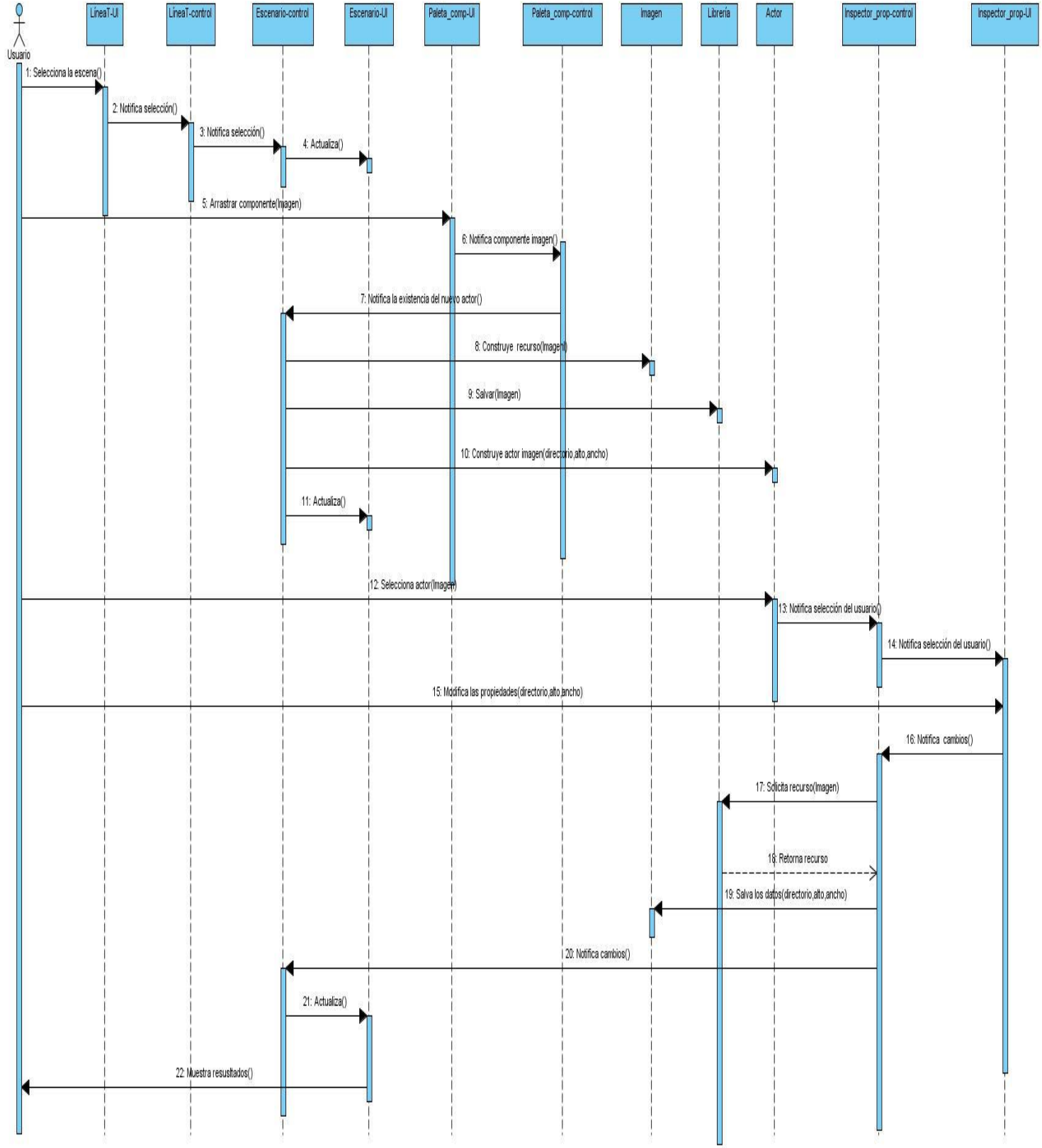


Figura 49: Diagrama de secuencia del caso de uso: Especificar recurso imagen

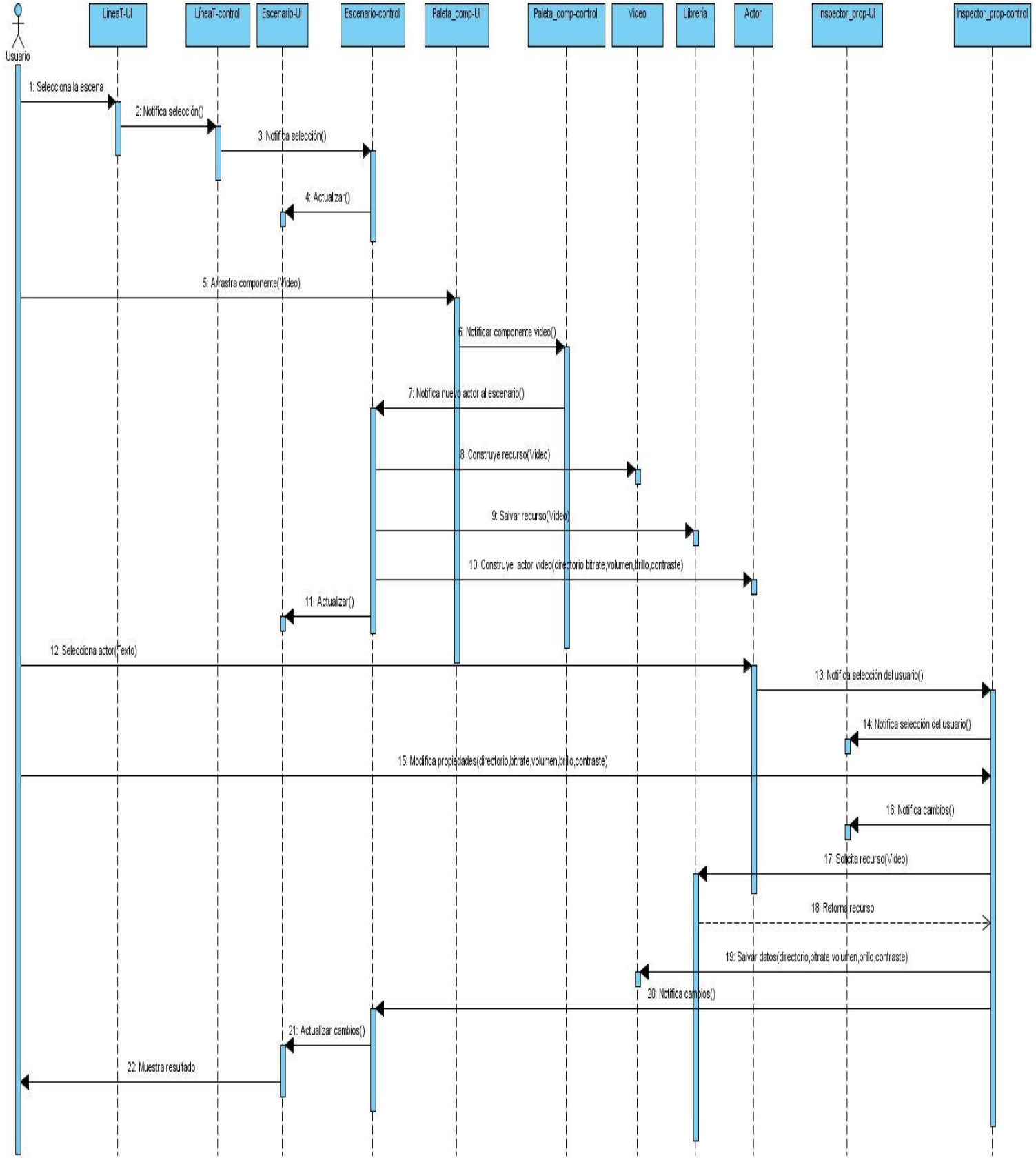


Figura 50: Diagrama de secuencia del caso de uso: Controlar video

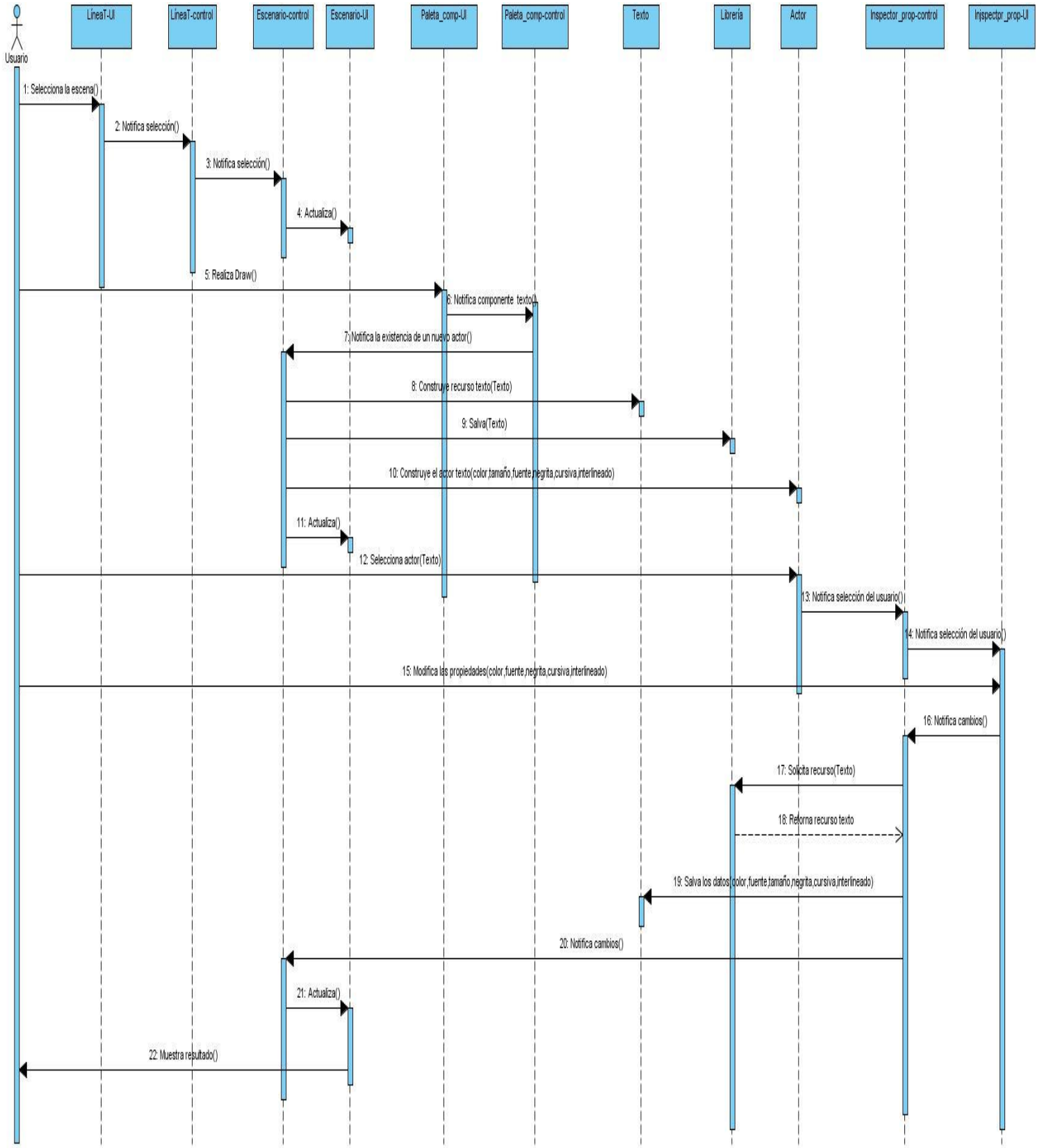


Figura 51: Diagrama de secuencia del caso de uso: Especificar texto

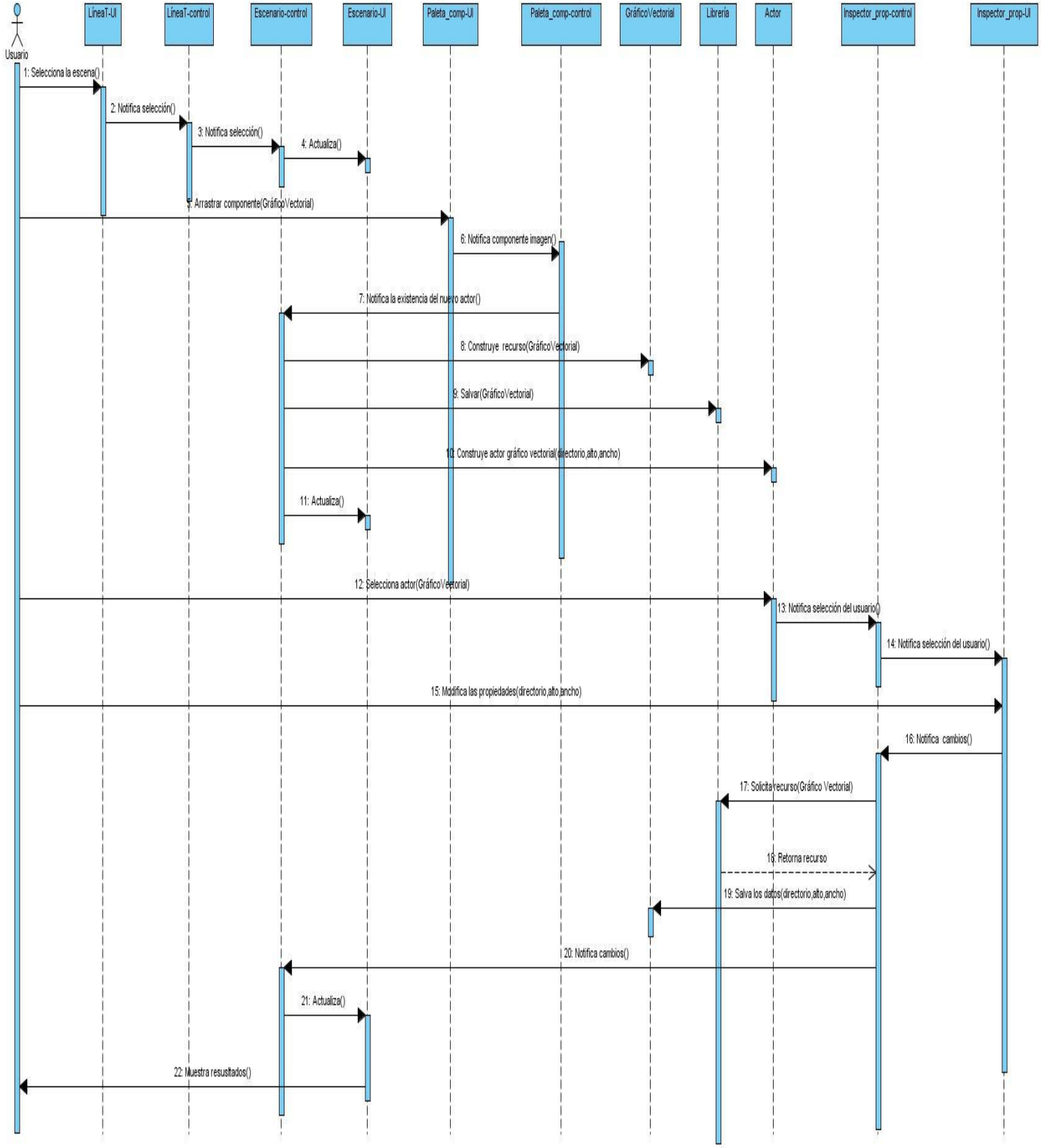


Figura 52: Diagrama de secuencia del caso de uso: Definir gráfico vectorial

Anexo 8: Mapa de navegación

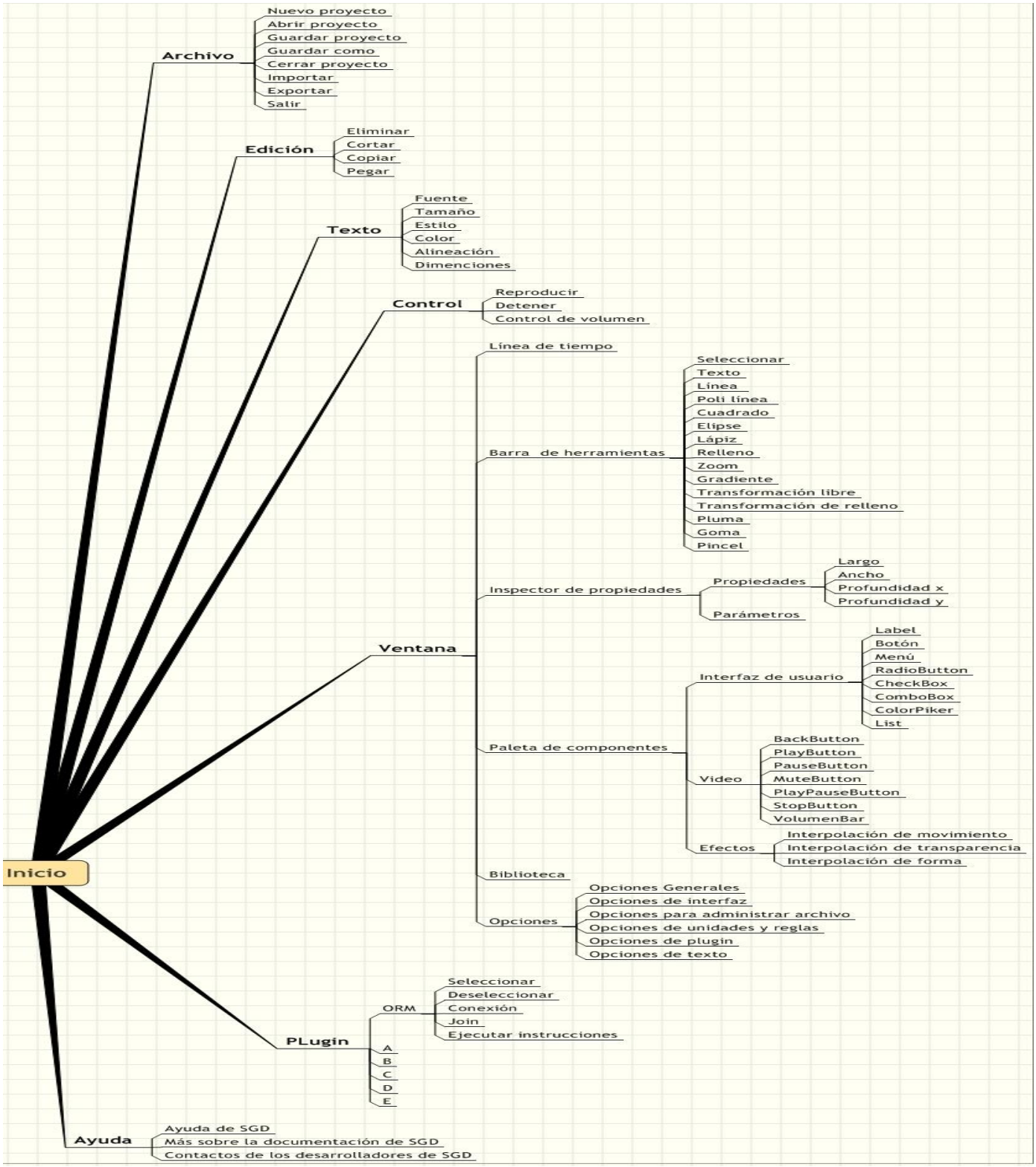


Figura 48: Mapa de navegación