Universidad de las Ciencias Informáticas

Facultad 10

SISTEMA AUTOMATIZADO PARA EL DISEÑO DE PROTOTIPOS DE INTERFACES DE USUARIO

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Autor: LISVAN CORDERO SADRADÍN

Tutor: Ing. SERGIO JESUS GARCÍA DE LA PUENTE

Ciudad de la Habana, Abril de 2009.

“Dime y lo olvido, enséñame y lo recuerdo, involúcrame y lo aprendo.”

Benjamín Franklin

Declaración de Autoría:

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 10 de la Universidad de las Ciencias Informáticas; así como a dicho centro a consultar cualquier cambio que se decida aplicar al mismo con su autor.

Para que así conste firmo la presente a los _____ días del mes de _____ del año 2008.

Lisvan Cordero Sadradín	Sergio Jesús García de la Puente
Firma del autor	Firma del tutor.

Datos de contacto del tutor:

Nombre: Sergio Jesús García de la Puente.
Email: sgarcia@uci.cu

Graduado en el curso 2007-2008 de Ingeniero en Ciencias de la Informática, en la Universidad de las Ciencias Informáticas. Se ha desempeñado como profesor de pregrado en la asignatura de Ingeniería de Software con dos años de experiencia laboral. Ha cursado cursos de superación posgraduada en temáticas afines a su especialidad, así como en la contribución de su formación como docente.

Está vinculado al grupo de Migración y Soporte al Software Libre, adscrito al polo de Software Libre de la Facultad 10 de la Universidad de las Ciencias Informáticas. Se desempeña en los roles de Analista de sistema y como asesor en temas de Arquitectura de Software. Está vinculado a un grupo de desarrollo de software. Pertenece al grupo de Migración y se desempeña como especialista de Migración Parcial.

Ha presentado temas de investigación en los eventos UCIENCIA 2009 e Informática 2009.

Participó en el Comité Organizador del IV Taller Internacional de Software Libre y estándares abiertos de Software, celebrado en el marco de la XIII Convención y Feria Internacional Informática 2009, celebrada en La Habana, Cuba, durante los días del 9 al 13 de febrero de 2009.

Opinión del usuario sobre el trabajo de diploma.

El Trabajo de Diploma titulado Sistema automatizado para el diseño de prototipos de interfaces de usuario, fue realizado en la Facultad 10 de la Universidad de las Ciencias Informáticas. Esta entidad considera que en correspondencia con los objetivos trazados el presente trabajo le satisface.

- Totalmente.

Los resultados de este Trabajo de Diploma le reportan a este ministerio los beneficios siguientes:

_____
_____
_____
_____
_____
_____

Y para que así conste, se firma la presente a los ___ días del mes de _____
del 2008.

_____	_____
Representante de la entidad.	Cargo.
_____	_____
Firma.	**Cuño.**

Agradecimientos

A mi grupo 10504.
A todos los que me apoyaron durante el desarrollo de esta tesis, a Carpio y Feus por el soporte y la capacitación con Qt y C++, a Yenner por sus consejos, a Angel y Jorge por sus críticas, a Tito que fue el beta-tester, en fin, a todos los que hicieron posible la realización de este trabajo y a mi tutor Sergio y su esposa Mileisys, por las revisiones hechas al documento.
A toda la comunidad de software libre de la Universidad de las Ciencias Informáticas.
A todos aquellos que nos han formado durante estos años.
A todo el que me confió en mí o no.

¡Gracias!

Dedicatoria

Lisvan:

Quisiera dedicar especialmente esta tesis a mi abuela Gina, a mi abuelo Lingo y a mi hermana Lisandra por todo su apoyo, cariño y confianza. Para que sepan que siempre están conmigo.

A mi papá, Juanito, por ser mi guía.

A mi mamá, Fredy, porque sin sus regaños no sería nadie.

A mi novia, Linett, por todo el amor que me ha dado.

A toda mi familia porque se lo merece.

Resumen

Esta investigación entrega una solución para suplir la poca presencia de aplicaciones para el diseño de prototipos de interfaces de usuario, en el mundo del Software Libre (indistintamente SWL). En este trabajo se valoran las alternativas existentes para el diseño de prototipos de interfaces de usuario, y se comparan desde varios puntos de vista, definiéndose un proceso de desarrollo para implementar un sistema que permita el diseño de este tipo de prototipos. Con este trabajo los programadores podrán enfocarse mucho mejor en su labor dejando a los diseñadores el diseño del producto, con lo cual se podrá reducir el tiempo de desarrollo de cualquier aplicación y concentrar los esfuerzos de los especialistas en cada rama del proceso.

Índice de contenido

Introducción.

Para el desarrollo de un producto software es muy buena práctica guiar el mismo por un proceso o metodología de desarrollo, esto forma parte de los tres pilares básicos definidos por Pressman a la hora de hacer Ingeniería de Software [1], la cual se apoya en un compromiso de organización y calidad. El proceso es el fundamento de la Ingeniería de Software, ya que mantiene cohesionadas las capas de tecnologías permitiendo un desarrollo racional y oportuno de la propia Ingeniería. El mismo forma la base de la gestión del proyecto y establece el contexto en que se aplican los métodos técnicos, se obtienen productos del trabajo, se establecen hitos, se asegura la calidad y el cambio dentro del propio proyecto se gestiona adecuadamente.

Pressman en su definición de Ingeniería de Software no solo enumeró el proceso, sino que también hizo énfasis en la existencia de los métodos y herramientas, los primeros indican cómo construir técnicamente el software y las últimas proporcionan el enfoque automático o semi-automático, para el propio proceso y los métodos.

Teniendo la base de todo este conocimiento previo se puede entonces concretar en la importancia de los sistemas de prototipos de interfaces de usuario, los cuales son utilizados generalmente en la captura de requisitos, con la intención de validar los requerimientos que el cliente informa debe cumplir el producto. Los prototipos de interfaces de usuario son de gran ayuda al entendimiento entre el equipo de desarrollo y los clientes, puesto que antes de desarrollo alguno se le presenta al cliente dicho prototipo y este expresa su conformidad o no con el mismo, este paso garantiza, generalmente, que no se provoquen cambios por no conformidades del cliente con la interfaz del sistema.

Luego de un estudio de sistemas similares se detectó que la mayoría están diseñados para modelar interfaces de escritorio, excepto en los casos de Microsoft Visio y el complemento Pencil para la versión 3.0 o superior de Mozilla Firefox, que sí permiten el modelado para ambos ambientes: web y escritorio. Los problemas de licencias y las restricciones del país para acceder a los servidores de código de la compañía Google, imposibilita incluir alguna de ellas en el paquete de productos de la estrategia de

migración a software libre. Es por tal motivo que el proyecto Unicornios, del polo Software Libre, de la Facultad 10 de la Universidad de las Ciencias Informáticas, se ha propuesto el desarrollo de un sistema que permita gestionar prototipos de interfaces de usuario, tanto para aplicaciones de escritorio como web, que sea además libre y multiplataforma, brindando soporte a las estrategias de migraciones parciales.

Actualmente, en el proceso de desarrollo del proyecto Unicornios, el diseño de prototipos de interfaces de usuarios se lleva a cabo a mano alzada o no se realiza en su totalidad. En caso de realizarse y que no sea manualmente, se hace con pequeños modelos utilizando aplicaciones propietarias que requieren de licencias y sistemas que no están implementados por el proyecto, haciendo de esto un proceso engorroso y difícil que no arroja buenos resultados, puesto que los desarrolladores o clientes no tienen una visión clara de cómo pudiera quedar la aplicación una vez terminada.

Una vez que se define la metodología y se establecen las pautas a seguir por el proyecto se comienza su desarrollo, pero no se validan correctamente los requisitos[1] ni se comprueba la completa satisfacción del cliente puesto que no existe un diseño de prototipo de la interfaz [2]. Un cambio en el diseño de la interfaz del proyecto implica cambios a gran escala del mismo, por lo tanto, se pierde tiempo de desarrollo y esfuerzos de todo el personal del proyecto.

Problema científico: La ausencia de una aplicación libre para el diseño de prototipos de interfaces de usuarios en el proyecto Unicornios afecta el proceso de validación de requisitos de los productos en desarrollo.

Objeto de estudio: Proceso de desarrollo de software.

Campo de acción: Las herramientas utilizadas para el diseño de prototipos de interfaces de usuarios.

Objetivo general: Desarrollar una aplicación libre para el diseño de prototipos de interfaces de usuarios que permita una mejor validación de los requisitos del sistema.

1 Ver “Validación de requisitos” en el Glosario de Términos.

Para alcanzar el objetivo propuesto se definen los siguientes **objetivos específicos**:

1. Realizar un diagnóstico de la situación actual de las principales herramientas que existen para diseñar prototipos de interfaces de usuario.
2. Implementar una herramienta libre para el diseño de prototipos de interfaces de usuarios.
3. Validar el sistema implementado.

Para dar cumplimiento a estos objetivos se plantean las siguientes **tareas de la investigación**:

1. Realizar un estudio de las principales herramientas existentes para el diseño de prototipos de interfaces de usuarios.
2. Evaluar el contenido de la información obtenida acerca de las principales herramientas y establecer un diagnóstico sobre las tendencias actuales.
3. Identificar las capacidades o condiciones que el sistema debe cumplir.
4. Definir las historias de usuarios.
5. Implementar las historias de usuarios.
6. Integrar los componentes desarrollados y reutilizados.
7. Realizar pruebas de aceptación al sistema.

Los métodos teóricos utilizados en esta investigación son: el método hipotético-deductivo y el método de la modelación. Con el primero a partir de la hipótesis y siguiendo reglas lógicas de deducción se llega a nuevos conocimientos y predicciones sobre las condiciones que debe tener el sistema, las que luego son sometidas a pruebas (verificaciones empíricas), mientras que con el método de la modelación se crean abstracciones con el objetivo de explicar la realidad y aplicarla respecto a lo que debe hacer el software.

Se utilizan como métodos empíricos: la observación y el experimento. La observación científica es la percepción planificada dirigida a un fin, y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado. El experimento es el método empírico para el estudio de un objeto en el cual el investigador crea las condiciones o adapta las existentes para el esclarecimiento de las propiedades, leyes y relaciones del objeto, para verificar una hipótesis, una teoría o un modelo.

Estos métodos permitieron realizar el estudio de las herramientas utilizadas para el diseño de prototipos de interfaces de usuarios, y obtener de ellas una experiencia de usuario que fue utilizada en la implementación del sistema que se necesitaba, así como probar las diferentes funcionalidades que cada una presentaba y realizar las pruebas necesarias para satisfacer las necesidades del cliente.

Con esta investigación se defiende la idea de que: con el sistema gestor de prototipos de interfaces de usuario, desarrollado para el proyecto Unicornios de la Facultad 10 de la Universidad de las Ciencias Informáticas, los equipos de desarrollo contarán con una herramienta que permitirá validar correctamente los requisitos, permitiendo que los programadores se enfoquen mucho mejor en el desarrollo de la aplicación dejando a los diseñadores realizar el diseño de la misma; así, el tiempo de desarrollo de cualquier aplicación disminuirá y los especialistas podrán concentrar sus esfuerzos en cada rama del proceso.

El producto será multiplataforma y libre, permitiendo así que forme parte de los paquetes propuestos por el proyecto en las estrategias de migración de los equipos de desarrollo y estará disponible para toda la comunidad de Software Libre[2]. Se propone su distribución bajo la Licencia Pública General [4], (en lo adelante GPL por sus siglas en inglés), para poder darle solución al problema que existe actualmente en el proyecto Unicornios respecto al uso de herramientas de este tipo.

En el presente trabajo se realiza un estudio de las principales herramientas de diseño de prototipos de interfaces de usuario ya sean libres o no, y se establece a partir de ahí un concepto claro de la necesidad que tienen los diseñadores de aplicaciones como estas. Microsoft Visio, Pencil, Visual Paradigm y Dia son algunos de los programas analizados que permitieron arribar a la conclusión de la escasa presencia de este tipo de aplicaciones desarrolladas bajo software libre, de ahí la importancia que tiene crear un producto propio que aportará un nuevo y novedoso servicio a las estrategias de migración y a la comunidad de Software Libre.

El mismo se encuentra estructurado a lo largo de 3 capítulos principales y anexos, que incluyen toda la

2 Software que permite al usuario copiarlo, estudiarlo, modificarlo y distribuirlo libremente.

documentación teórica así como las pruebas realizadas al software para validar su calidad. A continuación se hace una breve descripción de ellos.

Capítulo 1: Fundamentación Teórica. En este capítulo se realiza un estudio sobre el tema de la presente investigación, herramientas similares y tecnologías utilizadas actualmente para llevar a cabo el proceso de diseño. Además se definen las herramientas, lenguajes de programación y tipos de pruebas utilizados para la implementación de la aplicación.

Capítulo 2: Descripción del Sistema. En este capítulo se define la lista de reserva del producto y se detallan las historias de usuario que guiarán el desarrollo del sistema. Se lleva a cabo la realización de las historias de usuario y se integran los componentes desarrollados y reutilizados.

Capítulo 3: Validación de la solución propuesta. En este capítulo se detallan las pruebas realizadas al software, así como el impacto que tendrá el sistema una vez implantado.

1. Capítulo 1. Fundamentación Teórica.

El uso del Software Libre es cada día más difundido y necesario en Cuba constituyendo una alternativa para los sistemas que necesitan de una disponibilidad mayor de hardware, o para los programas que necesitan licencias para su utilización. Estas licencias tienen un gran coste y muchas veces implican problemas legales con la empresa que las provee debido al injusto bloqueo al que se encuentra sometida la isla. El desarrollo de una aplicación libre proporciona a los desarrolladores independencia absoluta, pues la descripción de las tareas a automatizar está detallada en las historias de usuarios y lo que necesitan los desarrolladores es el boceto de las interfaces de las aplicaciones solamente, ya que en ocasiones, ellos no poseen la experiencia de usuario necesaria para conformarlas.

1.1. Interfaces de Usuarios.

Según la Real Academia Española de la Lengua, una interfaz no es más que:

> "*conexión, física o lógica, entre un computador y el usuario, un dispositivo periférico o un enlace de comunicaciones"* [5].

En resumen, las interfaces de usuario son las partes con las que el usuario entra en contacto de manera física y cognitiva con un sistema y han de ser fáciles de comprender y utilizar [6]. Las mismas pueden tener diferentes funciones como la puesta en marcha y apagado del equipo, control de las funciones manipulables del mismo, manipulación de archivos y directorios entre otras.

Según la forma de interactuar del usuario pueden clasificarse en alfanuméricas, gráficas de usuario o táctiles y según su construcción se denominan de hardware o de software [6]. Las interfaces gráficas de usuarios (indistintamente GUI, del inglés Graphical User Interface) permiten comunicarse con el ordenador de una forma muy rápida e intuitiva, son programas o entornos que gestionan la interacción con el usuario basándose en relaciones visuales como iconos, menús o un puntero.

1.2. El GUI como paradigma.

Un paradigma es un ejemplo o ejemplar [7], de aquí que el científico estadounidense Thomas Khun (1922-1966) introdujera el término de paradigma científico como el conjunto de prácticas que definen una disciplina científica durante un período específico de tiempo. Los paradigmas están constantemente sustituyéndose unos por otros en un proceso llamado "cambio de paradigma" [8].
Douglas Engelbart (1925) a principio de los años '60 y basándose en las teorías de Vannebar Bush (1890-1974), dispuso una serie de conceptos como base de la "manipulación directa[3]" [9]:

- Combinación de ordenador, teclado y pantalla.
- Software de edición de texto.
- Ratón y el principio de apuntar y pinchar.
- Múltiples ventanas.
- Software de hipertexto.
- Conferencia entre ordenadores (basada en texto).

Estos formaron un nuevo paradigma en las interacciones hombre-máquina. Con el tiempo este paradigma fue evolucionando hasta lo que hoy en día pueden representar los entornos gráficos [10].

1.2.1. WYSIWYG.

El acrónimo viene del ingles "What you see is what you get" (lo que ves es lo que es) y describe un interfaz gráfico en el que el usuario ve una representación muy similar al resultado final, (imagen, documento de texto) mientras es creado. Por ejemplo, un usuario puede ver como quedará un documento cuando sea impreso mientras lo está creando. Esto implica el cambio dinámico de la representación por pantalla sin tener que ejecutar ningún comando o proceso en especial.

3 Estilo de interacción hombre-máquina.

Antes de la invención del WYSIWYG los textos aparecían en el mismo formato, con pocas indicaciones sobre margen, tamaño de fuente, etc. Se empleaban lenguajes basados en marcadores (como HTML[4]), que hoy en día se sigue utilizando en algunos contextos debido a su simplicidad.

1.3. Prototipos.

Un prototipo es una representación limitada del diseño de un producto, que permite a las partes responsables de su creación experimentar, probarlo en situaciones reales y explorar su uso. Un prototipo puede ser cualquier elemento, desde un papel con sencillos bocetos a un complejo software [11].

Los prototipos se clasifican en:

- Prototipos de baja fidelidad.
- Prototipos de alta fidelidad.

Los primeros utilizan materiales distintos y no se parecen al producto final. Su ventaja es que son baratos, simples, fáciles de producir y son particularmente útiles en las fases iniciales del desarrollo, durante el diseño conceptual. Por su parte, los prototipos de alta fidelidad son aquellos que utilizan materiales y se parece al producto final [11].

1.3.1. Prototipos de interfaces de usuario.

Es frecuente que los clientes no sepan lo que quieren, pero cuando ven algo pronto saben lo que no desean. Un prototipo de interfaz de usuario no es más que una maqueta de lo que tendrá el cliente en realidad y permiten al mismo interactuar con el equipo de desarrollo en la elaboración de la interfaz de la aplicación, lo cual no solo le permitirá observar el aspecto final del producto, sino que podrá eliminar los detalles que no le agraden del producto sin interrumpir el proceso de desarrollo de la aplicación [12].

4 HyperText Markup Language (Lenguaje de Marcas de Hipertexto).

1.4. Aplicaciones para el diseño de interfaces de usuario de uso actual.

No se conoce en el mundo del Software Libre, al menos hasta el momento, una aplicación que permita diseñar prototipos de interfaces de usuario. Las herramientas que existen son privativas o simplemente no permiten el diseño de prototipos de interfaces en su totalidad.

Las soluciones más completas capaces de diseñar interfaces de usuario son Microsoft Visio y Visual Paradigm, ambas de corte privativo. Las alternativas libres por su parte no poseen esta funcionalidad, aunque sí es posible desarrollar complementos para integrarlos a herramientas como Dia. Como una excepción se puede mencionar a un complemento de Mozilla Firefox: Pencil, el cual permite el diseño de este tipo de prototipos aunque la descarga de su código fuente no está permitida para nuestro país ya que se encuentra en servidores de código de Google (Google Code), ver ANEXO 1.

1.4.1. Microsoft Visio.

Microsoft Visio es un software de dibujo vectorial para Microsoft Windows. Visio comenzó a formar parte de los productos de Microsoft cuando fue adquirida la compañía Visio en el año 2000, su licencia es propietaria y es desarrollado por Microsoft. Está disponible en varios idiomas incluido el español y se puede ejecutar en el sistema operativo Windows.

Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, Lenguaje Unificado de Modelado (UML del inglés Unified Modeling Language), diseños de interfaces de usuario entre otros, que permiten iniciar al usuario en los lenguajes de programación.

Originalmente apuntaba a ser una aplicación para dibujo técnico para el campo de la Ingeniería y la

Arquitectura, pero una vez adquirida por Microsoft sufrió grandes cambios de directrices. En la actualidad (marzo de 2009) se encuentra compitiendo con productos como AutoCad, DesignCad y Microstation.

Su licencia es privativa por lo que se puede concluir en que no constituye una solución al problema que se ha planteado, aunque sí una muy buena fuente de estudio. Su código fuente no se encuentra disponible, pero las funciones que posee son muy útiles a la hora de obtener una experiencia de usuario[5] sobre cómo realizar un diseño de este tipo.

1.4.2. Visual Paradigm.

Visual Paradigm es una herramienta para UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos.

Visual Paradigm tampoco constituye la solución al problema planteado, ya que, para poder utilizarse, es necesario introducir una licencia. Las licencias de Visual Paradigm son Enterprise, Professional, Standar, Modeler, Personal y Community[6]. En dependencia de la licencia proporcionada la aplicación permite al usuario acceder a todas sus funciones o no. Las opciones de diseño de interfaces de usuario necesitan la licencia empresarial para poder ser utilizadas , de lo contrario esta opción ni siquiera es mostrada y si no se proporciona una licencia la aplicación se cierra y no permite su utilización.

5 Conjunto de factores y elementos relativos a la interacción del usuario con un entorno o dispositivos correctos [12].

6 Licencia gratuita de Visual Paradigm.

1.4.3. Dia.

Dia es una aplicación de software libre de propósito general para la creación de variados tipos de diagramas. Es desarrollada como parte del proyecto GNOME: el proyecto de escritorio para software libre, está concebido de forma modular y cuenta con diferentes paquetes de formas, lo cual permite agregarle funcionalidades en dependencia de las necesidades del usuario.

Dia está diseñado como un sustituto de la aplicación comercial Visio de Microsoft. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, entre muchos otros. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto SVG[7] e incluyéndolas en un archivo de Lenguaje de Marcas Ampliable (XML del inglés Extensible Markup Language). Su licencia es la GNU Public License. El formato para leer y almacenar gráficos es XML. Puede producir salida en los formatos de archivos gráficos EPS[8], SVG y PNG[9].

Dia permite crear diagramas personalizados y agregarlos en su paquete de formas, pero las imágenes que se insertan en la escena de trabajo pierden demasiada calidad cuando se trabajan con ellas. Por estas razones no fue elegida como solución aunque sí se tuvo en cuenta como alternativa.

1.4.4. OpenOffice.Org

OpenOffice.Org es una suite ofimática de software libre y código abierto de distribución gratuita que incluye herramientas como: procesador de texto, hoja de cálculo, presentaciones, herramientas para el

7 Scalable Vector Graphics.

8 PostScript Encapsulado.

9 Portable Network Graphics

dibujo vectorial y bases de datos. Tiene características similares, tanto en su interfaz como en su manera de trabajar con ella, respecto a Microsoft Office 2003.

Incluye las herramientas siguientes:

- Writer: un procesador de texto que permite editar HTML.
- Calc: una aplicación para manejar hojas de cálculo.
- Impress: un programa para realizar presentaciones.
- Draw: un editor de gráficos vectoriales.
- Base: un programa de bases de datos similar a Microsoft Access.
- Math: un editor de fórmulas matemáticas.

Draw es un editor de gráficos vectoriales que permite crear prototipos de interfaces de usuario y aunque tiene numerosas opciones para el diseño, todavía no es lo bastante robusto como para representar una alternativa que solucione el problema de esta investigación.

1.4.5. Pencil.

Pencil es un complemento desarrollado para Mozilla Firefox utilizando el motor o engine Gecko. El mismo permite el diseño de interfaces de usuario en su totalidad y con muy buena calidad. Para ello utiliza imágenes que el usuario puede modificar a su antojo y exportar el diseño como imágenes o como un proyecto del propio complemento.

A pesar de que el producto ha sido liberado bajo la licencia GPL se encuentra oficialmente disponible en servidores de la compañía Google por lo que el acceso al servidor de fuentes y a la descarga de todo lo necesario para desarrollar sobre este complemento está totalmente prohibido y el acceso a las mismas incluye violaciones del código de ética de la Universidad de las Ciencias Informáticas y de nuestro país.

Ninguna de las aplicaciones mencionadas anteriormente permiten al usuario el diseño de prototipos de interfaces de usuarios tanto web como de escritorio, las soluciones son incompletas pero contribuyen mucho al estudio de las características que debe tener el sistema que se desea desarrollar. Esta carencia de soluciones prácticas es lo que define la necesidad de crear una aplicación con estas características ya que, al no existir una alternativa, nace la idea de crear una que satisfaga las necesidades del proyecto en cuestión.

1.5. Descripción de las tecnologías, lenguajes y herramientas a utilizar.

En el desarrollo de cualquier sistema es necesario definir las tecnologías a utilizar, lenguajes, herramientas de programación, métricas de evaluación del software y las pruebas que se le realizarán al mismo, lo cual garantizará un mejor desempeño a la hora de implementar y desplegar el sistema. Para el desarrollo del sistema que se desea implementar se ha tenido en cuenta que el sistema debe ser multiplataforma, es decir, debe funcionar tanto en ambientes Windows como en entornos de tipo GNU/Linux y que, además, ha sido diseñado para que sea utilizado por el proyecto Unicornios de la Facultad 10.

1.5.1. Licencia Pública General de GNU (GNU GPL).

La Licencia Pública General de GNU fue creada por la Free Software Foundation a mediados de los años '80. Su objetivo era proteger al software libre de algún intento de apropiación que restringiese de sus libertades al usuario. Es actualmente reconocida por juzgados incluso en países como Alemania, y posee validez en jurisdicciones de derecho civil.

Esta licencia se encuentra actualmente en su versión 3 (GNU GPLv3), y fue elegida porque le da al usuario la fuerza legal para competir y reclamar su derecho respecto al software.

1.5.2. Framework Qt.

Qt es un framework de desarrollo multiplataforma usado para el desarrollo de interfaces gráficas, aunque también puede ser utilizado en el desarrollo de aplicaciones sin interfaz, o sea, de tipo consola o para servidores. Es desarrollado por la compañía noruega Trolltech, la cual es una filial de la compañía NOKIA desde junio de 2008. La última versión disponible de Qt es la 4.5 del 3 de marzo de 2009 hasta la fecha (13 de marzo de 2009).

NOKIA se encarga de mantener disponible Qt para numerosas plataformas entre las que destacan Qt/X11 (para X Window System en Unix/Linux), Qt/Windows (Qt para Microsoft Windows) y Qt/Mac (para MacOs X de Apple) así como una versión para la plataforma S60 de dispositivos móviles.

El uso de Qt se extiende al desarrollo de aplicaciones con gran prestigio a nivel internacional entre las que se destacan KDE, Opera, Google Earth, Skype, Qtopia, Photoshop Elements, VirtualBox, Psi, Last.Fm Player, MythTV, Scribus, VLC Media Player, Motorola A760 y tiene clientes de gran importancia a nivel mundial como son Google, VOLVO, VW, EPSON, AMD, XEROX y PHILIPS.

Qt utiliza C++ de forma nativa aunque cuenta con algunas extensiones no estándar implementadas por un pre-procesador que genera un código C++ estándar antes de la compilación. Posee además numerosas adaptaciones de bibliotecas entre las que se encuentran C# (Qyoto/Kimono), Python (PyQt), Java (Qt Jambi), Perl (PerlQt), Ruby (RubyQt), PHP y Pascal. Cuenta con muy buenas traducciones y posee uno de los mejores soportes que existen hoy día en el mundo.

Otras características que posee son los métodos para el acceso a bases de datos SQL[10], uso y análisis de

10 Structured Query Language (Lenguaje de Consulta Estructurado).

XML, manejo de hilos, y un API[11] multiplataforma para el manejo de archivos. Posee un diseñador de interfaces (Qt4 Designer), que brinda al usuario una mayor comodidad a la hora de crear sus formas y se integra perfectamente con entornos de desarrollo como Eclipse o QtCreator, desarrollado por la compañía NOKIA.

La ayuda de Qt es muy específica y es gestionada mediante la aplicación Qt Assistant, la cual expone todo lo referente a librerías y ejemplos resueltos. Se ha desarrollado además una ayuda rápida (Qt Demo) que cuenta con ejemplos resueltos como navegadores, manejo de imágenes y archivos entre otros que se pueden consultar de manera más detallada en la ayuda general del Qt Assistant.

Qt está disponible bajo una triple licencia: la GNU General Public License en sus versiones 2 (GNU GPL v2) o 3 (GNU GPL v3) con algunas excepciones, la LGPLv2.1 (desde Qt 4.5) y una licencia comercial propietaria en todas las plataformas soportadas. Estas licencias permiten a la aplicación final desarrollada ser licenciada bajo licencias de tipo Free Software, OpenSource o licencias de software privativo.

Todas las versiones soportan un amplio rango de compiladores incluyendo el compilador GCC C++ y la suite Visual Studio.

1.5.3. ¿Por qué Qt?

La versión 4 de Qt (Qt4) fue liberada en junio del 2005 introduciendo 5 nuevas tecnologías en el framework. A partir de entonces todos los años se ha liberado una nueva versión con características adicionales y mejoradas lo que explica por qué se plantea que su soporte es estable, maduro y constante. Existen otros frameworks como WxWidgets y GTK[12]. Las características de ambos frameworks son similares a las de Qt, lo que marca la diferencia es que Qt posee una documentación excelente, con

11 Application Program Interface (Interfaz de Aplicación del Programa)

12 Gimp Tool Kit (o Paquete de Trabajo para Gimp).

ejemplos claros y sencillos para que el usuario, aún siendo inexperto, pueda conseguir una respuesta rápida a su problema en el tiempo mínimo (Qt Demo, Qt Assistant). Se puede agregar que al instalar el paquete de desarrollo de software de Qt, el código que se escriba puede ser compilado sin cambio alguno en el entorno de trabajo deseado, sin importar si es Windows o GNU/Linux; sólo se debe copiar el código y compilarlo con QtCreator, el cual se encuentra disponible para ambas plataformas. A todo esto se suma la funcionalidad de traducción que posee el framework pues la herramienta Qt 4 Linguist constituye un sistema integrado de traducción que permite portar las aplicaciones al lenguaje que se desee. La ventaja de la triple licencia de Qt dependiendo del tipo de aplicación que se desee implementar, su facilidad de uso, su madurez, su constancia y su soporte hacen que se elija este framework por sobre los demás mencionados. Se puede mencionar también que existen numerosos foros en internet donde brindan ayuda en línea y en los que los programadores exponen dudas o problemas que son resueltos la mayoría de las veces en muy corto tiempo.

1.5.4. The Graphics View Framework.

The Graphics View es el framework de dibujo que utilizan las librerías Qt para pintar formas sobre una escena. Posee una superficie para manejar e interactuar con un gran número de formas bidimensionales y un componente vista (traducido del inglés view widget) para visualizar las formas.

El framework incluye una arquitectura de propagación de eventos que permite una interacción con precisión doble de las formas con la escena. Las formas pueden manejar eventos de teclado y de ratón así como rotación y acercamiento.

Graphics View permite además un descubrimiento de formas muy rápido que trae como resultado que se puedan visualizar escenas grandes en tiempo real, incluso con millones de formas. Graphics View fue introducido en la versión 4.2 de las librerías Qt, reemplazando a su predecesor QCanvas.

1.5.5. Lenguaje C++.

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup (1950), como extensión del lenguaje de programación C. Las principales características del C++ son el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica (templates).

Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel como la:

- Posibilidad de redefinir los operadores (sobrecarga de operadores).
- Identificación de tipos en tiempo de ejecución (RTTI).

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel. Por todas estas características es que resulta elegido de entre los muchos para utilizarlo en el desarrollo de aplicaciones rápidas y seguras.

1.5.6. Eclipse.

Eclipse es un entorno de desarrollo integrado (indistintamente IDE) de código abierto y multiplataforma. Es también una comunidad de usuarios que se encarga de extender constantemente las diferentes áreas de la aplicación. Eclipse fue desarrollado inicialmente por IBM pero ahora su desarrollo está en manos de la Fundación Eclipse.

Eclipse emplea módulos o complementos para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están

todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software que le permitir a Eclipse extenderse usando otros lenguajes de programación como el C++ o Python. La arquitectura plug-in permite escribir cualquier extensión deseada en el ambiente, como sería el complemento de Eclipse para Qt, el cual integra el diseñador de interfaces y completa y corrige las funciones de una forma muy óptima.

Principales características:

- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS[13].
- Asistentes (*wizards*): para creación de proyectos, clases, tests, etc.

1.5.7. QtCreator

QtCreator es un poderoso editor de C++ creado por la compañía NOKIA para el desarrollo con las librerías Qt, lo cual implica una integración total con el diseñador de Qt y todas las funciones con las que cuenta. Es un entorno liviano rápido y puede instalarse tanto en Windows como en Linux.

Sus principales características son:

- Es liviano y rápido.
- Es multiplataforma.
- Trabaja directamente en los archivos .pro del proyecto.
- Posee una búsqueda incremental.
- Integra la ayuda de Qt y su diseñador.
- Posee completamiento de código y resaltado de sintaxis.

13 Sistemas de Versiones Concurrentes (del inglés Concurrents Versions System).

- Muestra los errores en línea mientras se escribe.
- Posee una barra de navegación rápida en la que se puede intercambiar entre proyectos, marcadores, lugares del sistema de ficheros o documentos abiertos.
- Posee una interfaz para depurar los proyectos.
- Posee asistentes (wizards): para creación de proyectos, clases, tests, etc.
- Posee integración para control de versiones.
- Cuenta con un SDK[14] que permite su instalación y utilización en cualquier entorno.

Por presentar todas estas características y responder a las necesidades que se buscan para desarrollar el sistema que solucionará el problema planteado, se elige este entorno de desarrollo para implementar la aplicación que se necesita.

1.5.8. The Gimp (*GNU Image Manipulation Program*).

The Gimp es un programa multiplataforma para la edición de imágenes digitales, tanto dibujos como fotografías. Es un programa libre y gratuito englobado en el proyecto GNU y disponible bajo la Licencia Pública General de GNU.

Existen versiones portátiles de GIMP y está traducido al español, inglés, alemán, entre otras lenguas. Su desarrollo original es en idioma inglés. Constituye la alternativa libre al programa Adobe Photoshop y puede leer y escribir en la mayoría de los formatos gráficos incluyendo los de Photoshop.

Principales características que posee:

- Herramientas de selección con varias formas y tamaños.
- Herramientas de modificación de escala, inclinación, deformación, clonado entre otras.
- Herramientas de manipulación de texto y colores.

14 Kit de Desarrollo de Software (SDK del inglés Software Dvelopment Kit).

- Filtros para el manejo de las imágenes.
- Lenguajes para la automatización de procesos.
- Herramientas para el uso y manipulación de capas.
- Gimp utiliza GTK+ como biblioteca de controles gráficos.

Gimp se utiliza en el desarrollo de la aplicación para crear las imágenes que el usuario puede insertar en la escena.

1.5.9. Inkscape

Inkscape es un software libre y multiplataforma para el trabajo con gráficos vectoriales que utiliza la librería gráfica cairo (libcairo) sobre la cual está implementado GTK[15]. Representa la alternativa libre a Corel Draw de los entornos privativos. Es una implementación del estándar de la web3c SVG.
Su principal ventaja es que permite el manejo de gráficos vectoriales.

Existen otras aplicaciones para la edición de imágenes pero ninguna es tan completa como Gimp. El uso de esta aplicación en el desarrollo de la solución al problema planteado se limita a la creación y modificación de imágenes que se insertarán en la aplicación, tanto para su interfaz como para las imágenes que el usuario manejará en la escena.

1.6. Metodología ágil utilizada.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se utiliza una metodología para guiar el proceso, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más inconformes.

15 API (Application Program Interface) Gimp Tool Kit.

Muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo. Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí sí toma sentido el basarse en una metodología de desarrollo, y empezar a buscar cual sería la más apropiada para el desarrollo del software.

Las metodologías ágiles forman parte del movimiento de desarrollo ágil de software, conocidos anteriormente como metodologías livianas, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto. Se le denomina ágil como la habilidad de responder de forma versátil al cambio para maximizar los beneficios. Intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Las Metodologías Ágiles se basan en los siguientes principios :

- Realizar entregas cortas en el tiempo y continuas.
- Dar la bienvenida a los cambios.
- Entregas periódicas y frecuentes que funcionen.
- Los clientes forman parte del equipo de desarrollo.
- Equipo con individuos motivados. Darles para ello el ambiente, apoyo y confianza.
- La comunicación directa es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo. Intenta evitar el teléfono, correos electrónicos, fax, etc.
- La medida principal de progreso es el software que funciona.
- Desarrollo sostenible. Es indispensable que exista paz y armonía en el equipo para que el proyecto tenga éxito.
- Buen diseño y calidad técnica.
- La simplicidad es algo básico.
- Equipos autorganizados.

- El equipo debe realizar reflexiones periódicamente para plantearse cómo llegar a ser más efectivo.

Entre las mas conocidas están :

1. XP (Extreme Programming).
2. SCRUM.
3. Metodología Crystal.
4. Dynamic Systems Development Method (DSDM).
5. Adaptive Software Development (ASD).
6. Feature Driven Development (FDD).
7. Lean Development (LD).
8. BUP.
9. AUP
10. SXP

Para el desarrollo de este trabajo se propone el uso de una de las muchas metodologías ágiles enunciadas anteriormente: SXP.

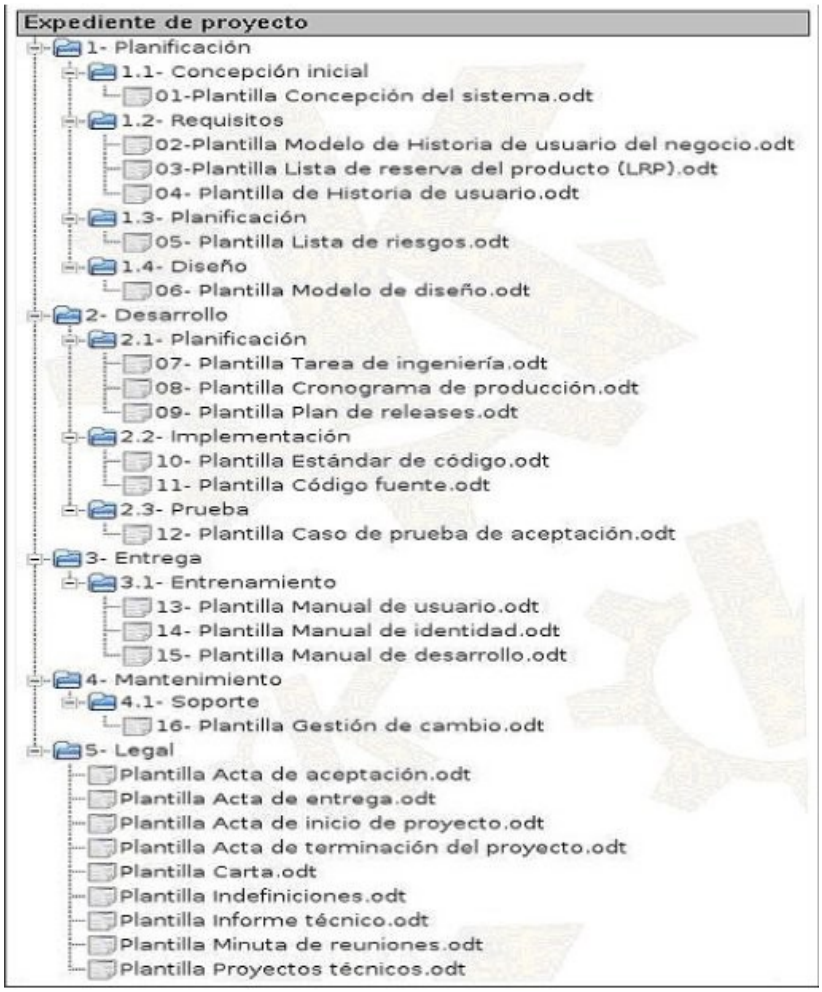
1.6.1. SXP.

Se escoge SXP ya que es una metodología compuesta por las metodologías SCRUM y XP y ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles, que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva, fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo. SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos, tal que sepamos por dónde andamos. XP más bien es una metodología encaminada para el desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario

final, pues es uno de los requisitos para llegar el éxito del proyecto. Consta de 4 fases principales: Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto; Desarrollo, es donde se realiza la implementación del sistema hasta que este listo para ser entregado; Entrega, puesta en marcha; y por último Mantenimiento, donde se realiza el soporte para el cliente. De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

SXP esta especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo [13].

En la siguiente figura se pueden observar el expediente del proyecto de la metodología así como todos los artefactos que genera.



SXP propone los siguientes roles para el trabajo en equipo:

- Líder del proyecto (Scrum Master).
- Gerente (Management).
- Especialista.
- Cliente (Customer).
- Consultor.
- Equipo del proyecto (Scrum Team).

El equipo del proyecto será conformado por otros roles como:

- Programadores (Programmers).
- Analista (Analyst).
- Diseñadores (Designers).
- Encargado de prueba (Tester).
- Arquitecto (Architect).

1.7. Arquitectura del sistema.

Las arquitecturas en capas constituyen uno de los estilos de programación que aparecen con mayor frecuencia mencionados en las diferentes literaturas. El estilo de programación en capas es una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior, y se sirve de las prestaciones que le brinda la inmediatamente inferior, instrumentando una vieja organización estratigráfica que se remonta a las concepciones formuladas por Edsger Dijkstra en la década de 1960. En la práctica, las capas suelen ser entidades complejas, compuestas por varios paquetes o subsistemas. [13] [14].

Para el desarrollo de la aplicación se utiliza una arquitectura en capas. En la **capa de presentación** se encuentra el componente Qt 4.4.3 como framework gráfico, que se utiliza para implementar la interfaz visual de la aplicación, mientras que en la **capa lógica de negocio** se encuentran las clases implementadas para la creación de objetos y la manipulación de los mismos, que constituirán el núcleo de la aplicación.

2. Capítulo 2. Descripción del Sistema.

El sistema a implementar comprende componentes existentes que requieren ser analizados, para reutilizarlos en el desarrollo ágil del Sistema Automatizado para el Diseño de Prototipos de Interfaces de Usuario, mientras que toda la dinámica del proyecto se expresa en forma de historias de usuarios, prototipos de interfaces de usuario y algunos modelos auxiliares expuestos a continuación.

2.1. Propuesta del sistema a implementar.

Después de que se ha realizado el estudio de las principales herramientas existentes para el diseño de prototipos de interfaces de usuario, se concluye en que no representan una solución factible para el proyecto, pues no cumplen con las características necesarias para formar parte del paquete de aplicaciones libres para la migración, aunque sí se toma la experiencia de usuario que ellas aportan. Esto permite que se decida comenzar la implementación de una aplicación libre a partir de la experiencia adquirida en el uso de las aplicaciones antes mencionadas.

Para la implementación de la aplicación se debe utilizar una capa de presentación y una capa lógica de negocio. En la primera estará la interfaz que el usuario observará y que se implementará en Qt, mientras que la capa lógica de negocio contendrá todo el núcleo de la aplicación, principales clases y funcionalidades. La arquitectura en capas permitirá su fácil manipulación y adaptación.

La aplicación estará compuesta por una barra de menú, varias barras de herramientas y componentes flotantes, (para los diferentes tipos de objetos que se podrán insertar y para el fondo de la escena en caso de que el usuario desee modificarlo). Cada acción que el usuario desee ejecutar tendrá un acceso en las barras de herramientas de la aplicación y mediante el clic derecho sobre el objeto insertado en algunos casos. En el centro de la aplicación se podrá observar y trabajar con una escena y sobre la escena una

vista, la cual se encargará de mostrar los objetos insertados. El conjunto de la escena y la vista permitirá acercar o alejar la vista de trabajo.

La aplicación debe permitir al usuario insertar texto, componentes de aplicaciones en forma de imágenes y polígonos. El texto podrá ser modificado por el usuario cuando lo desee mediante un combobox para cambiar el tipo de letra y otro combobox para el tamaño de la fuente. Además se podrá cambiar el tipo de fuente a negrita, subrayada o cursiva. El texto se podrá mover por toda la escena y posicionarlo en donde se desee.

La aplicación debe permitir que el usuario cambie el color del objeto insertado, o si prefiere, le ponga una imagen de fondo. En caso de que el usuario no quede satisfecho con todos los componentes que tiene a su disposición puede crear un componente manualmente e insertarlo en forma de imagen en formato PNG. Todos estos objetos se podrán exportar a los formatos PDF[16], PNG y JPEG[17]. El proyecto se podrá guardar o abrir uno ya existente.

2.2. Análisis de las posibles implementaciones y componentes que pueden ser reutilizados.

La reutilización en un proceso de desarrollo es un pilar clave pues reduce los tiempos y costes del mismo, además de que permite generar aplicaciones eficientes y de gran fiabilidad.

Para dar solución al problema planteado se hace necesario utilizar componentes ya existentes que facilitan el trabajo y agilizan el desarrollo de la aplicación. Algunos componentes han sido localizados realizando una búsqueda en la ayuda de las librerías Qt, específicamente en los ejemplos de Escena para Diagramas (Diagram Scene Example) y algunas implementaciones sobre la misma, además de los

16 Portable Document Format.

17 Joint Photographic Experts Group

ejemplos correspondientes a dibujos (Painting Examples).

De estos ejemplos se utilizan las clases para manipular los polígonos y las escenas para crear una plataforma de dibujo que permita diseñar prototipos de interfaces de usuario, permitiendo al usuario una interacción amigable con el producto final.

El sistema tendrá una clase principal que gestionará toda la información que le brindan las clases internas y una clase que manipulará la escena sobre la cual se encontrará la vista. Esta clase heredará de la clase QGraphicsScene incluida en el framework, y reimplementará algunos de los métodos de la clase padre.

Las demás clases permitirán insertar tanto texto, como imágenes y polígonos. Las mismas heredan de las clases QGraphicsPolygonItem y QGraphicsTextItem que a su vez heredan de QGraphicsItem. Estas clases permitirán que los polígonos sean redimensionados y sus propiedades cambiadas cuando el usuario lo desee. La interfaz de la aplicación se basa en la experiencia de usuario adquirida en el estudio de Galm [15]; una aplicación desarrollada en la Universidad de las Ciencias Informáticas para la creación de animaciones multimedia precisamente con las librerías Qt y C++.

2.3. Planificación del proyecto por roles.

Como SXP define diferentes roles es necesario darle una organización al proyecto utilizando los mismos, quedando estructurado de la siguiente forma:

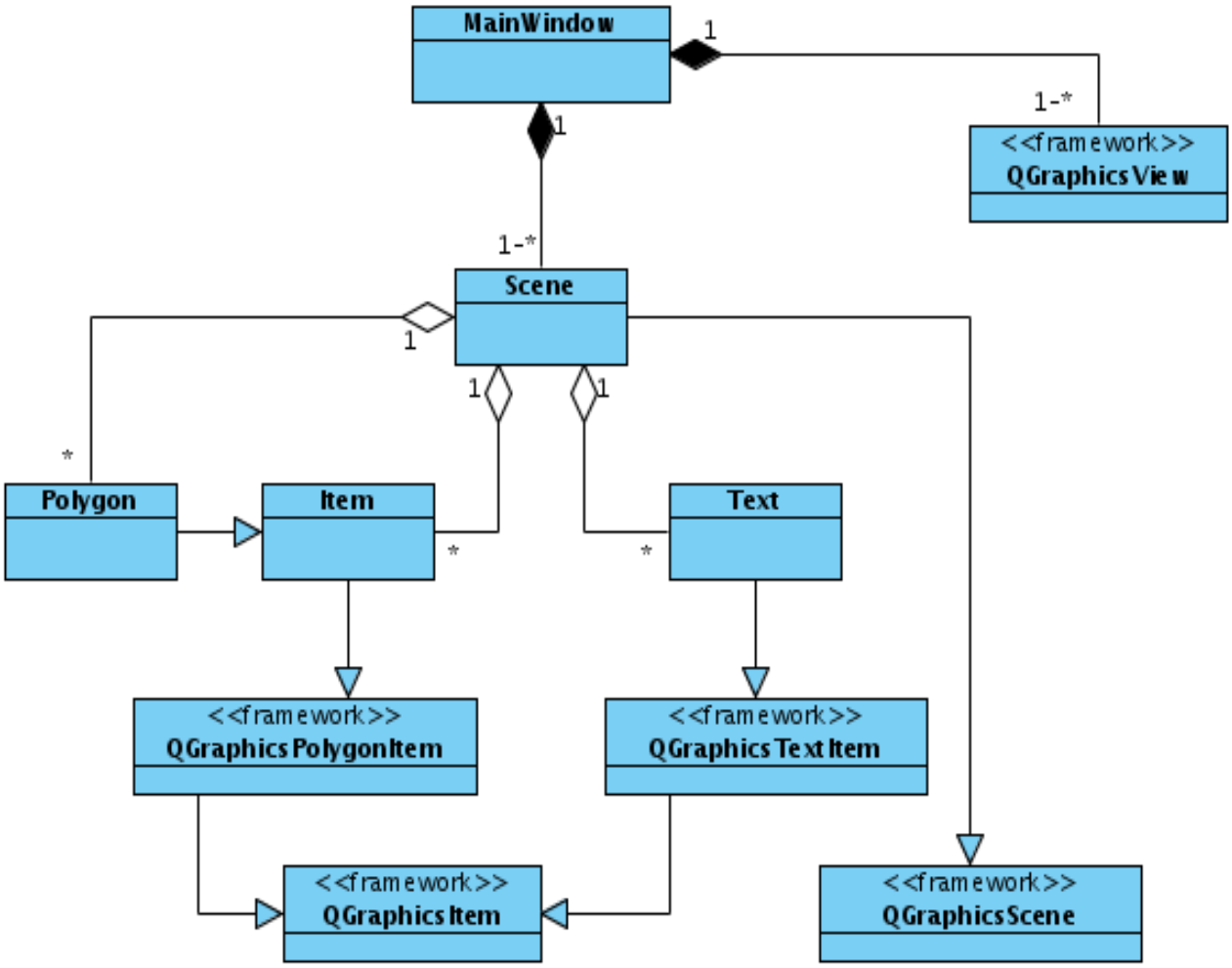
Rol	Responsabilidad	Nombre
Gerente (Management)	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto.	Lisvan Cordero Sadradín.
Cliente (Customer)	El cliente participa en las tareas que involucran	Sergio Jesús García de la

	la lista de reserva del producto.	Puente.
Programadores (Programmers)	Es el encargado de producir el código y escribir las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.	Lisvan Cordero Sadradín.
Analista (Analyst)	Es el encargado de escribir las historias de usuario y las pruebas funcionales para validar su implementación.	Lisvan Cordero Sadradín.
Diseñadores (Designers)	Encargados del diseño del sistema; así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.	Lisvan Cordero Sadradín.
Encargado de Pruebas (Tester)	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Lisvan Cordero Sadradín.
Arquitecto (Architect)	Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas.	Lisvan Cordero Sadradín.

2.4. Modelo de dominio.

El Modelo de las Historias de Usuario del Negocio constituye una descripción detallada del negocio, pero si este no está bien definido entonces se realiza el Modelo de Dominio. A continuación se muestra el

Modelo de Dominio relacionado con la aplicación.



2.5. Lista de Reserva del Producto.

Una de las actividades más importantes en el desarrollo de un sistema es la captura de requisitos, ya que en esta actividad se recogen las necesidades que tienen los clientes para el desarrollo del sistema. Según la metodología de desarrollo SXP se utiliza la Lista de Reserva del Producto para recoger esos requisitos.

La Lista de Reserva del producto no es más que un artefacto generado para recoger las funcionalidades que debe cumplir el sistema durante su desarrollo, y define una lista de actividades que priorizan el trabajo que se va a realizar permitiendo organizarlo y controlarlo. Esta lista puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto y del cliente. Con la restricción de que sólo puede cambiarse entre iteraciones. El objetivo es asegurar que el producto definido al terminar la lista sea el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

Asignado a	**Ítem ***	**Descripción**	**Estimación**	**Estimado por**
Prioridad		**Muy Alta**		
Lisvan	1	Implementar la interfaz principal de la aplicación.	1	Desarrollador
Lisvan	2	Implementar la clase para gestionar los textos en la escena.	1	Desarrollador
Lisvan	3	Implementar la clase para gestionar los componentes en la escena.	1	Desarrollador
Lisvan	4	Implementar la clase para gestionar los polígonos en la escena.	1	Desarrollador

Lisvan	5	Implementar la clase para manejar la escena de la aplicación.	1	Desarrollador
Lisvan	6	Definir los estilos de los polígonos que se insertarán en la escena.	1	Diseñador
Lisvan	7	Diseñar los componentes que el usuario insertará en la escena.	1	Diseñador
Lisvan	8	Definir los estilos de los componentes que utilizará la aplicación.	1	Diseñador.
Lisvan	9	Implementar la funcionalidad que permita redimensionar los componentes que se sitúen en la escena.	1	Desarrollador
Prioridad		**Alta**		
Lisvan	1	Implementar las barras de herramientas para facilitar al usuario el manejo de los objetos en la escena.	1	Desarrollador
Lisvan	2	Implementar las funcionalidades para crear nuevos documentos y guardarlos.	1	Desarrollador
Lisvan	3	Integrar todos los componentes creados.	1	Encargado de pruebas
Lisvan	4	Gestionar el cambio de imagen	1	Desarrollador

		del componente insertado.		
Prioridad		**Media**		
Lisvan	1	Implementar las funcionalidades de exportar e imprimir.	1	Desarrollador
Lisvan	2	Implementar las funcionalidades para salir del proyecto	1	Desarrollador
Prioridad		**Baja**		
Lisvan	1	Implementar las funcionalidades para para mostrar datos acerca de la aplicación.	1	Desarrollador
RNF (Requisitos No Funcionales)				
Lisvan	1	Definir los iconos que usarán los menús de la aplicación.	1	Diseñador

2.6. Historias de usuarios.

SXP en la fase de Planificación-Definición requiere de la entrega de Historias de Usuarios. Las Historias de usuario constituyen la técnica para especificar los requisitos del software y son escritas por los clientes como las tareas que el sistema debe hacer en lenguaje natural.

Historia de Usuario U_IDes_01

Historia de Usuario	
Número: U_IDes_01	**Nombre Historia de Usuario:** Implementar interfaz principal.
Modificación de Historia de Usuario Número: 2.0.	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 6
Riesgo en Desarrollo: Medio	**Puntos Reales:** 6
Descripción: En esta Historia de Usuario se implementa la interfaz de la aplicación. Esta interfaz se crea dinámicamente, es decir, en tiempo de ejecución. Se le agregarán funcionalidades a medida que se implementen. La interfaz tendrá barras de menú, barras de herramientas, componentes flotantes (dockwidgets) y la vista sobre la escena. Las etiquetas se verán en español y todas podrán traducirse.	
Observaciones: Las funcionalidades que necesiten alguna implementación especial podrán utilizarán un mensaje para recordar que faltan.	
Prototipo de interfase: Ver ANEXO 2.	

Historia de Usuario U_IDes_02

Historia de Usuario	
Número: U_IDes_02	**Nombre Historia de Usuario:** Implementar la clase para manipular la escena.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2
Descripción: Se implementa la clase escena (Scene) para manipular los objetos que el usuario debe insertar. Esta clase debe permitir que se gestionen diferentes elementos como texto, imágenes y polígonos.	
Observaciones: La clase no contará con todas sus funcionalidades en un primer momento. Luego se le agregarán más funciones para que el usuario las utilice.	
Prototipo de interfase:	

Historia de Usuario U_IDes_03

Historia de Usuario	
Número: U_IDes_03	**Nombre Historia de Usuario:** Implementar la clase para manejar texto.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2
Descripción: Se implementa la clase para manipular el texto en la escena. La clase debe permitir cambiar el formato del texto (color, tipo de letra, tamaño del texto, si es negrita, cursiva o subrayada).	
Observaciones: Estas funcionalidades se conectarán a los menús para poder ser utilizadas por los usuarios.	
Prototipo de interfase:	

Historia de Usuario U_IDes_04

Historia de Usuario	
Número: U_IDes_04	**Nombre Historia de Usuario:** Implementar aplicación.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2
Descripción: Implementar la aplicación. Se integran los componentes desarrollados hasta el momento y se conectan las barras de menú a sus respectivas señales y/o funciones de forma tal que el usuario ya puede insertar texto y modificarlo.	
Observaciones: Si no se conectan las señales el usuario no puede hacer nada en la escena.	
Prototipo de interfase:	

Historia de Usuario U_IDes_05

Historia de Usuario	
Número: U_IDes_05	**Nombre Historia de Usuario:** Implementar clase para manipular las imágenes.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Implementar la clase para manipular las imágenes que el usuario debe insertar en la escena. Esta clase tendrá un tipo de datos enum que contendrá los tipos de imágenes que se pueden insertar en la escena y las imágenes se cargarán desde un archivo con extensión “.qrc” que contendrá las imágenes que el usuario podrá insertar.	
Observaciones: Las imágenes serán archivos .png y se escalarán con un factor mínimo para no deformarlas demasiado. En las próximas versiones se utilizarán imágenes escalares (SVG) para poder escalarlas sin perder resolución.	
Prototipo de interfase:	

Historia de Usuario U_IDes_06

<table>
<tr><th colspan="2">Historia de Usuario</th></tr>
<tr><td>Número: U_IDes_06</td><td>Nombre Historia de Usuario: Implementar la clase para manipular los polígonos en la escena.</td></tr>
<tr><td colspan="2">Modificación de Historia de Usuario Número: Ninguna</td></tr>
<tr><td>Usuario: Lisvan Cordero Sadradín.</td><td>Iteración Asignada: 1</td></tr>
<tr><td>Prioridad en Negocio: Alta</td><td>Puntos Estimados: 1</td></tr>
<tr><td>Riesgo en Desarrollo: Alto</td><td>Puntos Reales: 3</td></tr>
<tr><td colspan="2">Descripción: Implementar la clase para manipular los polígonos en la escena.</td></tr>
<tr><td colspan="2">Observaciones: Los polígonos se insertarán con un clic arrastrando el ratón hasta el punto final que dará el ancho y el largo del polígono, finalizando la operación de inserción con otro clic del ratón. A los polígonos insertados se les podrá cambiar su color de fondo o establecer una imagen de relleno.</td></tr>
<tr><td colspan="2">Prototipo de interfase:</td></tr>
</table>

Historia de Usuario U_IDes_07

Historia de Usuario	
Número: U_IDes_07	**Nombre Historia de Usuario:** Implementación de funcionalidades.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 3
Descripción: Implementar las funcionalidades que deben permitir a la aplicación imprimir y exportar las imágenes insertadas en la escena.	
Observaciones: Las imágenes podrán ser exportadas a formatos diferentes, tanto “.png” y “.jpeg” como “.pdf”. La impresión enviará una imagen de todos los elementos insertados en la escena al sistema de impresión de la computadora sin importar la plataforma en la que se encuentre en ese momento.	
Prototipo de interfase:	

Historia de Usuario U_IDes_08

Historia de Usuario	
Número: U_IDes_08	**Nombre Historia de Usuario:** Integración del sistema.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisvan Cordero Sadradín.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: Integrar las funcionalidades adicionales del sistema entre las que destacan agregar nuevos tabs o pestañas al área de trabajo principal y acercar o alejar el área de trabajo. Las pestañas se adicionarán de una en una y se eliminarán de la misma forma.	
Observaciones:	
Prototipo de interfase:	

2.7. Plan de iteraciones.

Versión	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 1	Crear los componentes para el diseño de la interfaz de usuario. La vista principal, los menús y barras de herramientas así como los widgets flotantes que utilizará el usuario en su trabajo.	U_IDes_01	27/10/08 - 27/12/08
Iteración 2	En esta iteración el sistema será capaz de permitir al usuario insertar componentes y modificarlos. Los polígonos se podrán modificar y cambiar el color de relleno. La escena permitirá modificar el texto insertado en ella.	U_IDes_02 U_IDes_03 U_IDes_04 U_IDes_05 U_IDes_06 U_IDes_07	20/12/08 - 8/3/09
Iteración 3	En esta iteración se continúa con la implementación de funcionalidades del sistema para agregar o eliminar pestañas al área de trabajo, acercar o alejar esta misma área y mostrar las mensajes que el usuario necesita para manipular la aplicación correctamente.	U_IDes_08	8/03/09-20/03/09

2.8. Tareas de Ingeniería.

En la fase de desarrollo de la metodología SXP se propone la entrega de las tareas realizadas para dar cumplimiento a cada historia de usuario. Estas tareas son llamadas Tareas de Ingeniería.

Tareas de Ingeniería para la HU U_IDes_01

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_01
Nombre Tarea: Estudio de las librerías Qt.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 2
Fecha Inicio: 27/10/08	**Fecha Fin:** 10/11/08
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian las clases presentes en la ayuda de Qt que son necesarias para crear los widgets de la interfaz.	
Observación:	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_01
Nombre Tarea: Estudio de las clases de GALM.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 10/11/08	**Fecha Fin:** 17/11/08
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudia la clase principal de la aplicación GALM para estudiar cómo se crean los componentes de la interfaz.	
Observación:	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** U_IDes_01
Nombre Tarea: Crear menús, barras de herramientas y widgets flotantes de la aplicación.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 19/11/08	**Fecha Fin:** 26/11/08
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se implementan los menús de la aplicación. No se conectan los menús a los SLOTS.	
Observación: Los SLOTS o funciones a las que conectar los menús se implementarán más adelante, por el momento solamente es necesario que se pueda ver cómo va quedando la interfaz de la aplicación.	

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** U_IDes_01
Nombre Tarea: Integrar todos los componentes desarrollados.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 2
Fecha Inicio: 6/12/08	**Fecha Fin:** 20/12/08
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se integran todos los componentes a la aplicación.	
Observación: En este punto el usuario debe poder ver la aplicación y utilizar los menús.	

Tareas de Ingeniería para la HU U_IDes_02

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_02
Nombre Tarea: Estudio de ejemplos.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 2
Fecha Inicio: 20/12/08	**Fecha Fin:** 4/1/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos de la ayuda de Qt y GALM para implementar la clase escena (Scene).	
Observación:	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_02
Nombre Tarea: Implementación de la clase escena.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 3
Fecha Inicio: 4/1/09	**Fecha Fin:** 25/1/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se implementa la clase escena.	
Observación:	

Tareas de Ingeniería para la HU U_IDes_03

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_03
Nombre Tarea: Estudio de ejemplos.	
Tipo de Tarea : Estudio.	**Puntos Estimados:** 1
Fecha Inicio: 25/1/09	**Fecha Fin:** 2/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos en los que se manipula texto en la ayuda de Qt y las principales opciones de los editores de texto disponibles en las plataformas GNU/Linux y Microsoft Windows.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_03
Nombre Tarea: Implementación de la clase texto.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 3/2/09	**Fecha Fin:** 6/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos en los que se manipula texto en en aplicaciones como OpenOffice.org y editores de texto sencillos como Kate para observar cómo se relaciona el usuario con la aplicación.	
Observación:	

Tareas de Ingeniería para la HU U_IDes_04

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_04
Nombre Tarea: Integración de componentes.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 3/2/09	**Fecha Fin:** 6/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se integran todos los componentes desarrollados hasta el momento. El usuario ya es capaz de insertar y modificar el texto en la escena.	
Observación:	

Tareas de Ingeniería para la HU U_IDes_05

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_05
Nombre Tarea: Estudio de ejemplos.	
Tipo de Tarea : Estudio.	**Puntos Estimados:** 1
Fecha Inicio: 6/2/09	**Fecha Fin:** 13/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos de la ayuda de Qt para insertar formas en la escena y se modifican los métodos necesarios para su correcta utilización.	
Observación: Se modifican los métodos que no brindaban ninguna funcionalidad a la aplicación para que esta brinde al usuario las opciones de modificar texto en la escena.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_05
Nombre Tarea: Diseño de componentes.	
Tipo de Tarea : Diseño.	**Puntos Estimados:** 1
Fecha Inicio: 14/2/09	**Fecha Fin:** 20/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se diseñan los componentes que el usuario agregará en la escena. Los componentes son imágenes. Se clasifican en combobox, checkbox, window, radiobutton, sliderbar, textedit, button, progressbar y label.	
Observación:	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** U_IDes_05
Nombre Tarea: Implementación de la clase para manipular imágenes.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 14/2/09	**Fecha Fin:** 20/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se implementa la clase para manipular imágenes. La clase deberá permitir al usuario insertar una imagen prediseñada.	
Observación:	

Tareas de Ingeniería para la HU U_IDes_06

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_06
Nombre Tarea: Estudio de ejemplos.	
Tipo de Tarea : Estudio.	**Puntos Estimados:** 1
Fecha Inicio: 20/2/09	**Fecha Fin:** 29/2/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos de la ayuda de Qt y de la aplicación GALM para manipular polígonos en la escena de manera que permita insertar un polígono de la manera correcta.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_06
Nombre Tarea: Implementación de componentes.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 2
Fecha Inicio: 20/2/09	**Fecha Fin:** 5/3/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos de la ayuda de Qt y de la aplicación GALM para manipular polígonos en la escena de manera que permita insertar un polígono de la manera correcta.	
Observación:	

Tareas de Ingeniería para la HU U_IDes_07

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_07
Nombre Tarea: Estudio de ejemplos.	
Tipo de Tarea : Estudio.	**Puntos Estimados:** 1
Fecha Inicio: 3/3/09	**Fecha Fin:** 8/3/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se estudian los ejemplos de la ayuda de Qt para exportar imágenes en diferentes formatos.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_07
Nombre Tarea: Implementación de funcionalidades.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 3/3/09	**Fecha Fin:** 8/3/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se implementan las funcionalidades de imprimir y exportar las imágenes de la escena.	
Observación:	

Tareas de Ingeniería para la HU U_IDes_08

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U_IDes_08
Nombre Tarea: Implementación de funcionalidades.	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 6/3/09	**Fecha Fin:** 15/3/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se implementan las funcionalidades para adicionar pestañas al área de trabajo y se corrigen posibles errores del sistema.	
Observación: Los errores del sistema pueden ser errores de compilación.	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U_IDes_08
Nombre Tarea: Corrección de errores en la aplicación	
Tipo de Tarea : Implementación.	**Puntos Estimados:** 1
Fecha Inicio: 6/3/09	**Fecha Fin:** 15/3/09
Programador Responsable: Lisvan Cordero Sadradín.	
Descripción: Se corrigen los errores que aparecen al agregar la funcionalidad de las pestañas en el área de trabajo.	
Observación:	

2.9. Estándar de código.

Para el desarrollo de la aplicación se utilizó un estándar de código que permite el entendimiento de todo el código escrito. Este estándar describe como están compuestos los nombres de las clases, los métodos, las variables, los comentarios y la estructura del código en general.
Los nombres de las clases comienzan con mayúsculas:
Ejemplo: class Scene

Los métodos comienzan con mayúsculas y las variables con minúsculas. La segunda palabra de ambos comienza con mayúsculas y los nombres de los métodos proporcionan una visión de la función que cumple en idioma inglés. La selección del idioma inglés para los nombres de los métodos se debe en parte a que la ayuda del framework se encontraba en este idioma y por otro lado era mucho más cómodo para el programador (costumbre).

```
Ejemplo:    void CreateTabWidget()
            int lasty;
```

Las constantes comienzan con mayúsculas y la segunda palabra del nombre también está en mayúsculas.

```
Ejemplo:    const int TextButton = 17;
```

Para la identación se utiliza el estilo Allman, también conocido como “ANSI style”. Este estilo ubica las llaves asociadas a una sentencia de control en la segunda línea, las sentencias son identadas al segundo nivel de las llaves. El editor del IDE QtCreator 1.0 utiliza una tabulación de 4 espacios para identar.

```
Ejemplo:    if (myScene->isEmpty())
            {
                clear();
            }
```

Los comentarios sencillos comienzan con dos barras verticales inclinadas (slash) y los comentarios

detallados con un slash seguido de dos asteriscos.

Ejemplo: //Esto es un comentario

/** Esto es una descripción detallada**/

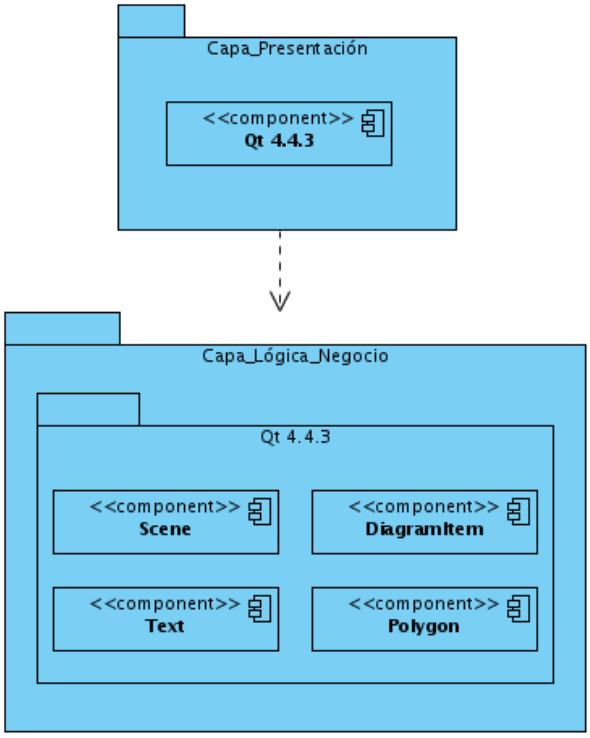
2.10. Diseño con metáforas.

SXP está basada en XP, y dicha metodología define un término llamado metáfora, lo cual según Martin Fowler es una historia compartida que describe como debería funcionar el sistema y define que, la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema.

El diseño con metáforas es sencillamente el diseño de la solución más simple que pueda funcionar y ser implementado en un momento dado del proyecto; lo cual genera el artefacto conocido como Modelo de Diseño, que a su vez está compuesto por un diagrama de paquetes, el cual expone dicho diseño.

Los diagramas de paquetes describen los elementos físicos del sistema y sus relaciones, además la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables.

A continuación se presenta el diagrama de paquetes para el sistema que se propone:



Las clases que se encuentran en la capa lógica de negocio son clases que heredan de otras ya existentes en el framework para vista de gráficos de Qt (traducido del inglés Graphics View Framework). Este framework permite la interacción de la escena y los objetos sobre ella de manera rápida y sencilla. Provee una superficie para manejar e interactuar con una gran número de objetos gráficos bidimensionales, con soporte para acercamientos y rotaciones.

El framework Incluye una sistema de propagación de eventos que permite interactuar con los objetos en la escena; la precisión de esta arquitectura con los objetos es de tipo double y los objetos pueden tratar eventos de teclado y ratón como presión de teclas, clic, doble clic, y seguir el movimiento del ratón.

El framework utiliza un árbol de particionamiento del espacio binario, BSP (del inglés Binary Space Partitioning), que facilita un descubrimiento de objetos muy rápido y como resultado de esto, se pueden visualizar grandes escenas en tiempo real sin limitaciones, incluso con millones de objetos por lo que en una misma escena se pueden tener varias vistas y la escena puede contener objetos de variadas formas geométricas, de esta forma se puede crear una lista de escenas y trabajar sobre la que se desee.

La clase encargada de proporcionar una escena al framework es QGraphicsScene. La escena tiene entre sus responsabilidades, las cuales son las siguientes:

- Proveer una interfaz rápida para manipular gran cantidad de objetos.
- Propagar los eventos a cada objeto que ella contenga.
- Gestionar el estado de los objetos como selección y enfoque.
- Proveer la funcionalidad de renderizado sin cambios, principalmente para dibujar e imprimir.

La escena se utiliza para contener objetos de tipo QGraphicsItem. Los objetos en la escena se adicionan y se obtienen mediante funciones, incluso cuando varios objetos coinciden en un punto es posible obtener el objeto del principio, en un punto determinado o al final de entre todos los objetos coincidentes.

El sistema de propagación de eventos permite enviar eventos a los objetos y manipular la propagación de eventos entre los objetos mismos. Si la escena recibe un evento con el clic del ratón (mouse press event) envía este evento al objeto que se encuentre en la posición en que se presionó el ratón. La clase QGraphicsScene gestiona estados de los objetos como la selección y el enfoque (focus), por lo que es posible obtener todos los objetos seleccionados en una lista. Esto se utiliza en la vista para manipular el cambio de posición de varios objetos al mismo tiempo. QGraphicsScene permite también dibujar toda la escena o solamente una parte de esta.

No sería posible visualizar los objetos que se encuentran sobre la escena sin un widget de vista. Este widget lo proporciona la clase QGraphicsView. Anteriormente se mencionó que se podían tener varios widgets vista sobre una misma escena. El widget vista es un área enrollable (scroll area) con barras de

desplazamiento para escenas muy grandes.

La vista recibe los eventos del teclado y el ratón traduciéndolos a eventos de la escena[18] y enviando los eventos a la escena visualizada.

Para las características de navegación avanzada como el acercamiento (zooming) y la rotación (rotation) la vista utiliza la matriz de transformación (QGraphicsView::matrix()); con esta matriz la vista puede transformar el sistema de coordenadas de la escena. Para más comodidad la clase QGraphicsView provee funciones para traducir desde el sistema de coordenadas de la escena hacia la vista y viceversa.

La clase base de los objetos gráficos en la escena es QGraphicsItem. El framework provee algunos objetos estándares como rectángulos, elipses, y texto pero sin dudas la más poderosa característica de la clase QGraphicsItem es permitir crear objetos personalizados.

Entre otras características QGraphicsItem soporta:

- Detección de colisiones
- Drag and Drop.
- Agrupamiento mediante relaciones padre-hijo.

Los objetos existen en un sistema de coordenadas y como QGraphicsView poseen funciones para mapear las coordenadas entre el objeto y la escena y desde objeto a objeto. Además, como QGraphicsView, puede transformar su sistema de coordenadas para la rotación y escala de objetos individuales.

Los objetos pueden contener otros objetos llamados hijos (children), por lo que el objeto contenedor se denomina padre (parent) de ahí que sea posible obtener el hijo mediante el padre o al contrario.

Por tanto, se define un patrón de programación en capas para el desarrollo de la aplicación que permite

18 Convirtiendo las coordenadas usadas a coordenadas de la escena.

manejar eventos y enviar señales a todos los objetos que se encuentren en la escena, obteniendo un desempeño óptimo en la gestión de estos componentes.

Después de comprender cómo funciona el sistema vista-objeto-escena, se implementa la clase Scene que envía señales (SIGNAL) y gestiona los objetos en la escena mediante funciones que la clase principal se encarga de atender.

Las clases desarrolladas heredan de las clases QGraphicsItem y QGraphicsScene de forma que sea posible manipular más cómodamente los objetos en la escena y modelar un arquitectura fácil de adaptar.

3. Capítulo 3. Validación de la solución propuesta.

La metodología SXP requiere en su etapa de pruebas, que se presenten los casos de pruebas o test de aceptación que se le han realizado al sistema. Estos casos de prueba deben realizarse en cada una de las iteraciones para poder continuar desarrollando y avanzar hacia la siguiente iteración. Las pruebas en un sistema se realizan durante todo el período de implementación y son seguidas todo el tiempo para poder desarrollar de manera iterativa.

La programación extrema define entre iteraciones un conjunto de casos de pruebas para poder avanzar hacia la próxima iteración. El siguiente plan de prueba mantiene controlado el desarrollo de las pruebas que se realizaron a cada historia de usuario.

3.1. Casos de prueba.

Se definieron casos de prueba para todas las historias de usuario. A continuación se muestran las pruebas que se realizaron mediante tablas, definidas en la metodología utilizada para el desarrollo de la aplicación. Los principales elementos que se muestran son las condiciones que el sistema debe cumplir para que se ejecute el caso de prueba así como la entrada y salida de los pasos de ejecución del caso de prueba realizado. Por último se muestra el resultado esperado y la evaluación de la prueba en relación a si fue satisfactoria o no.

Caso de prueba para la historia de usuario U_IDes_01.

Esta sección cubre el conjunto de pruebas que se le realizaron a la historia de usuario: Implementar interfaz principal.

Este caso de prueba pretende comprobar que se crea la interfaz de la aplicación de manera correcta de forma tal que todos los menús sean creados así como las barras de herramientas y demás componentes de la interfaz de la aplicación.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar interfaz principal.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que se han creado todos los componentes implementados. Se comprueba que existen los menús y todas las señales son capturadas correctamente por la aplicación.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Se muestran correctamente todos los componentes de la aplicación: menús, pestañas, barras de herramientas, y escena.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario U_IDes_02.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Implementar la clase para manipular la escena. Este caso de prueba pretende comprobar que la escena permite insertar y eliminar objetos así como moverlos a diferentes posiciones.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que la escena ha sido creada. Se inserta dinámicamente un objeto en la escena con el objetivo de visualizar la escena cuando contiene un objeto y se intenta mover el objeto insertado.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta. Se observa la escena. Se inserta un objeto en la escena y se intenta cambiar su posición arrastrándolo a otro lugar del área de trabajo.	
Resultado Esperado: Se muestra correctamente el objeto insertado en la escena y se puede mover de un lugar a otro.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario U_IDes_03.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Implementación de la clase texto. Este caso de prueba pretende comprobar que el texto en la escena puede ser insertado y modificado.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementación de la clase texto.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que no existan errores. La escena debe permitir insertar texto y modificarlo.	
Condiciones de Ejecución: Se compila la aplicación sin errores.	
Entrada / Pasos de ejecución: Se selecciona el botón para insertar texto y se selecciona el lugar en el que se desea insertar texto en la escena.	
Resultado Esperado: Se muestra el texto insertado en la escena.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario U_IDes_04.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Implementar aplicación.
Este caso de prueba pretende comprobar que todos los componentes se integran en la aplicación permitiendo un correcto funcionamiento de la misma.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar aplicación.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se integran los componentes de la clase texto y escena para permitir al usuario modificar el texto insertado. Para esto se modifica la clase escena y la clase principal de la aplicación.	
Condiciones de Ejecución: La aplicación no puede contener errores en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta. El usuario oprime el botón de insertar texto e insertar un texto. Se selecciona el texto y se intenta modificar mediante las barras de herramientas.	
Resultado Esperado: Se muestra correctamente el texto insertado en la escena y se modifica correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario U_IDes_05.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Implementar la clase para manipular los pixmaps o imágenes. Este caso de prueba pretende comprobar que se puedan insertar imágenes en la escena.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular las imágenes.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se modifican la clase Item y la clase Scene para integrarlas correctamente y que se capturen las señales enviadas por ambas clases. La aplicación debe permitir insertar imágenes en la escena.	
Condiciones de Ejecución: La aplicación no puede contener errores en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación. El usuario selecciona el botón para insertar un componente en la escena. El usuario selecciona con un clic el lugar donde se insertará ese componente.	
Resultado Esperado: No existen errores en el proceso y el usuario puede insertar la imagen correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario U_IDes_06.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Implementar la clase para manipular los polígonos en la escena. Este caso de prueba pretende comprobar que se puedan crear polígonos y gestionarlos en la escena.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular los polígonos en la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que no existan errores en la clase Polygon. El usuario intenta insertar un polígono.	
Condiciones de Ejecución: No existen errores en el proceso de compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación. El usuario selecciona la opción para insertar polígonos. El usuario selecciona el lugar para comenzar a pintar el polígono y con un clic indica dónde terminar.	
Resultado Esperado: Se inserta el polígono correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular los polígonos en la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: El usuario intenta modificar el tamaño del polígono.	
Condiciones de Ejecución: No existen errores en el proceso de compilación.	
Entrada / Pasos de ejecución: El usuario selecciona una esquina para redimensionar el polígono y comienza a desplazarse por la escena. El usuario termina de redimensionar el polígono presionando nuevamente el clic del ratón.	
Resultado Esperado: Se redimensiona el polígono correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular los polígonos en la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: El usuario intenta el color de un polígono mediante las barras de herramientas.	
Condiciones de Ejecución: Existe un polígono insertado en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona la opción para cambiar el color del polígono. Esta opción se encuentra en la barra de herramientas. El usuario selecciona un color.	
Resultado Esperado: El polígono cambia de color correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular los polígonos en la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: El usuario intenta cambiar el color de un polígono mediante la opción Propiedades, a la cual accede pulsando el clic derecho del ratón.	
Condiciones de Ejecución: Existe un polígono insertado en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el polígono y pulsa el clic derecho del ratón para ir a Propiedades en el menú contextual que aparece. El usuario selecciona la opción para cambiar el color del polígono. El usuario selecciona un color.	
Resultado Esperado: El polígono cambia de color correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular los polígonos en la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: El usuario intenta ponerle una imagen de fondo a un polígono mediante las barras de herramientas.	
Condiciones de Ejecución: Existe un polígono insertado en la escena.	
Entrada / Pasos de ejecución: El usuario el polígono y a continuación la opción para cambiar el color del polígono. El usuario selecciona una imagen.	
Resultado Esperado: El polígono cambia su color de fondo por la imagen seleccionada.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementar la clase para manipular los polígonos en la escena.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: El usuario intenta poner una imagen de fondo al polígono mediante la opción Propiedades, a la cual accede pulsando el clic derecho del ratón.	
Condiciones de Ejecución: Existe un polígono insertado en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el polígono y pulsa el clic derecho del ratón para ir a Propiedades en el menú contextual que aparece. El usuario selecciona la opción para insertar una imagen de fondo en el polígono. El usuario selecciona una imagen.	
Resultado Esperado: El polígono cambia su color de fondo por la imagen seleccionada.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario U_IDes_07.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Implementación de funcionalidades. Este caso de prueba pretende comprobar que se puedan exportar e imprimir los objetos que contiene la escena.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Implementación de funcionalidades.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se pueden exportar los objetos de la escena al formato PNG.	
Condiciones de Ejecución: Existe un objeto en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el menú Archivo la opción Exportar. El usuario selecciona un nombre y la extensión “.png”. El usuario selecciona la opción Aceptar en el diálogo mostrado.	
Resultado Esperado: Se guarda la imagen en formato “.png” correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Implementación de funcionalidades.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se pueden exportar los objetos de la escena al formato JPEG.	
Condiciones de Ejecución: Existe un objeto en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el menú Archivo la opción Exportar. El usuario selecciona un nombre y la extensión “.jpeg”. El usuario selecciona la opción Aceptar en el diálogo mostrado.	
Resultado Esperado: Se guarda la imagen en formato “.jpeg” correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 3	**Nombre Historia de Usuario:** Implementación de funcionalidades.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se pueden exportar los objetos de la escena al formato JPEG.	
Condiciones de Ejecución: Existe un objeto en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el menú Archivo la opción Exportar. El usuario selecciona un nombre y la extensión “.jpeg”. El usuario selecciona la opción Aceptar en el diálogo mostrado.	
Resultado Esperado: Se guarda la imagen en formato “.jpeg” correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 4	**Nombre Historia de Usuario:** Implementación de funcionalidades.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se pueden exportar los objetos de la escena al formato PDF.	
Condiciones de Ejecución: Existe un objeto en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el menú Archivo la opción Exportar. El usuario selecciona un nombre y la extensión ".pdf". El usuario selecciona la opción Aceptar en el diálogo mostrado.	
Resultado Esperado: Se guarda la imagen en formato ".pdf" correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 5	**Nombre Historia de Usuario:** Implementación de funcionalidades.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se pueden imprimir los objetos de la escena.	
Condiciones de Ejecución: Existe, al menos, un objeto en la escena.	
Entrada / Pasos de ejecución: El usuario selecciona el menú Archivo la opción Imprimir.	
Resultado Esperado: Se muestra un diálogo de impresión en el que el usuario modifica los parámetros de impresión.	
Evaluación de la Prueba: Prueba satisfactoria.	

Casos de prueba para la historia de usuario U_IDes_08.

Esta sección cubre el conjunto de pruebas realizadas a la historia de usuario: Integración del sistema. Este caso de prueba pretende comprobar que la aplicación funcione correctamente.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se comprueba que no existan errores en el proceso de compilado y que todas las clases se integran correctamente.	
Condiciones de Ejecución: Todas las clases están implementadas correctamente y las señales son capturadas por las clases definidas.	
Entrada / Pasos de ejecución: Se compila la aplicación y se revisa la salida de ejecución.	
Resultado Esperado: No existen errores en el proceso ni se muestran mensajes de error o de señales sin capturar.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se crea una nueva escena para trabajar.	
Condiciones de Ejecución: La aplicación se ejecutó sin errores.	
Entrada / Pasos de ejecución: El usuario selecciona la opción Nuevo en el menú Archivo. Se crea una nueva pestaña que contiene una escena y una vista sobre ella. El usuario inserta un objeto.	
Resultado Esperado: Se muestra el objeto insertado y se puede cambiar de pestañas correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 3	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se inserta texto en varias escenas y se modifica.	
Condiciones de Ejecución: Existen varias escenas.	
Entrada / Pasos de ejecución: La escena cambia al modo insertar texto. El usuario presiona el clic del ratón para insertar el texto. El usuario escribe y se muestra el texto en la escena. El usuario cambia de escena y se repite el procedimiento.	
Resultado Esperado: Se muestra el texto insertado en la escena correspondiente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 4	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se insertan polígonos en varias escenas.	
Condiciones de Ejecución: Existe más de una escena.	
Entrada / Pasos de ejecución: La escena cambia al modo insertar polígono. El usuario presiona el clic del ratón para comenzar a dibujar el polígono. A medida que el usuario mueve el ratón se muestra la figura en la escena. El usuario presiona el clic del ratón para terminar de dibujar el polígono seleccionado. El usuario cambia de escena e inserta otro polígono.	
Resultado Esperado: Se muestran el polígono en la escena correspondiente correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 5	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se insertan imágenes en varias escenas.	
Condiciones de Ejecución: Existen, al menos, dos escenas creadas.	
Entrada / Pasos de ejecución: El usuario selecciona la opción insertar imágenes. La escena cambia al modo insertar imágenes. El usuario presiona clic del ratón para insertar una imagen. Se inserta imagen. La escena se pone en modo mover objeto. El usuario cambia de escena y repite los pasos.	
Resultado Esperado: Se muestran las imágenes en las escenas correspondientes.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 6	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se mueven las imágenes al cambiar de escenas.	
Condiciones de Ejecución: Existen, al menos, dos escenas creadas.	
Entrada / Pasos de ejecución: La escena cambia al modo insertar imagen. El usuario presiona clic del ratón para insertar una imagen. Se inserta la imagen y el usuario la mueve por la escena. El usuario cambia de escena y repite el procedimiento.	
Resultado Esperado: Se mueven las imágenes correctamente en la escena activa.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se elimina un objeto.	
Condiciones de Ejecución: La escena contiene un objeto.	
Entrada / Pasos de ejecución: El usuario selecciona la opción Borrar del menú Forma.	
Resultado Esperado: Se elimina el objeto sobre el que se ejecutó la acción.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 8	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se elimina un objeto.	
Condiciones de Ejecución: La escena contiene un objeto.	
Entrada / Pasos de ejecución: El usuario selecciona un objeto. El usuario selecciona la opción Borrar del menú contextual que aparece al presionar el clic derecho del ratón sobre la forma que se desea eliminar.	
Resultado Esperado: Se elimina el objeto sobre el que se ejecutó la acción de forma correcta.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 9	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se elimina un texto.	
Condiciones de Ejecución: La escena contiene un texto.	
Entrada / Pasos de ejecución: El usuario selecciona la opción Borrar del menú Forma.	
Resultado Esperado: Se elimina el texto sobre el que se ejecutó la acción.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 10	**Nombre Historia de Usuario:** Integración del sistema.
Nombre de la persona que realiza la prueba: Lisvan Cordero Sadradín.	
Descripción de la Prueba: Se elimina una escena.	
Condiciones de Ejecución: Existen, al menos, dos escenas.	
Entrada / Pasos de ejecución: El usuario inserta un objeto en cada escena. El usuario presiona el botón para cerrar la escena. El usuario crea una nueva escena.	
Resultado Esperado: Se elimina la escena activa y al crearse la otra el usuario verifica que se eliminó de la memoria correctamente pues se crea vacía.	
Evaluación de la Prueba: Prueba satisfactoria.	

3.2. Resultados obtenidos.

En este epígrafe se relacionan los resultados obtenidos hasta el momento por el equipo de desarrollo de Qid. Como fruto de este trabajo se encuentra disponible un versión beta del producto que comenzará a utilizarse en el proyecto Unicornios y que recibirá muchas más funcionalidades en próximas versiones.

3.2.1. Acerca del tiempo de desarrollo.

El problema de la inexistencia de una herramienta para el diseño de prototipos de interfaces de usuario en el proyecto Unicornios de la Facultad 10 era una situación problemática que se venía dando hacía mucho tiempo.

Se dió la tarea de implementar una herramienta libre y multiplataforma capaz de suplir las necesidades del proyecto y surge la idea de realizar este sistema. A finales de 2008 se comienza el desarrollo de Qid con tareas de investigación. Con un solo programador que a la vez se convertía en diseñador, analista y encargado de pruebas se logró realizar un producto atractivo y totalmente funcional. El producto estará sujeto a mejoras ya que se implementarán más funcionalidades para su uso en otros ámbitos en la universidad.

Para la realización de este producto se estudiaron las librerías Qt y se comenzó a implementar sin experiencia previa de los elementos y conceptos más importantes para lograr desarrollar un sistema con las características que se necesitaban. El tiempo de desarrollo fue relativamente corto y se cumplieron todas las expectativas que se plantearon durante la planificación inicial.

3.2.2. Acerca de las funcionalidades obtenidas.

Entre algunas de las funcionalidades que presenta el producto software en su primera versión se encuentran que:

- Permite gestionar textos.
- Se pueden gestionar polígonos.
- Se pueden tener varias escenas de trabajo.
- Se pueden exportar las escenas a documentos PDF o imágenes.
- Es multiplataforma.
- Es fácil de utilizar incluso por usuarios inexpertos en el diseño.
- Posee una ayuda que explica paso a paso cómo crear un prototipo de interfaz de usuario.

Estos casos de prueba guiaron la calidad del sistema y determinaron en cada momento si se continuaba avanzando en el desarrollo de la aplicación. Este análisis muestra las funcionalidades alcanzadas por el sistema en el período que se ha trabajando en su desarrollo.

Conclusiones.

Se ha realizado un estudio sobre las aplicaciones utilizadas para el diseño de prototipos de interfaces de usuario, sus características y funcionamiento. Se implementó un sistema libre y multiplataforma que permite crear prototipos de interfaces de usuario y exportar los diseños creados tanto a imágenes como archivos PDF lo cual satisface las necesidades existentes en el proyecto Unicornios de la Facultad 10 de la Universidad de las Ciencias Informáticas de una aplicación con estas características.

Por todo lo anteriormente expuesto se concluye que los objetivos trazados en el presente trabajo se han cumplido satisfactoriamente.

Recomendaciones

- Que el presente trabajo se continúe desarrollando para implementar nuevas funcionalidades y mejorar su calidad.
- Que el producto se comience a utilizar en el diseño de los prototipos de interfaces de usuario en el proyecto con el objetivo de refinar su funcionamiento.

Referencias Bibliográficas.

[1]. Pressman, Roger. *Ingeniería del Software. Un enfoque práctico.* Quinta Edición MacGraw-Hill [cited 21 March 2009]. Available from world wide web: <http://bibliodoc.uci.cu/pdf/reg02689.pdf>

[2]. Ferré, Xavier. Validación de requisitos. [cited 4 April 2009]. Available from world wide web: <http://is.ls.fi.upm.es/xavier/usabilityframework/actividades/val_req.php>

[3]. Free Software Foundation. Licencia GPL o GNU General Public License version 3. June 2007. [cited 13 March 2009]. Available from world wide web: <http://www.gnu.org/copyleft/gpl.html>

[4]. Real Academia Española. Interfaz. In *Real Academia Española* [cited 13 March 2009]. Available from world wide web: <http://buscon.rae.es/draeI/SrvltObtenerHtml?IDLEMA=43177&NEDIC=Si>

[5]. Interfaz de usuario. En *Wikipedia.org*, March 2009 [cited 13 March 2009]. Available from world wide web: <http://es.wikipedia.org/wiki/Interfaz_de_usuario>

[6]. Paradigma. In *WordReference* [cited 14 March 2009]. Available from world wide web: <http://www.wordreference.com/definicion/paradigma>

[7]. Khun, Thomas. *La Estructura de las Revoluciones Científicas*. Universidad de Chicago, 1962.

[8]. Manipulación Directa. In *Wikipedia.org*, March 2009 [cited 13 March 2009]. Available from world wide web: <http://es.wikipedia.org/wiki/Manipulaci%C3%B3n_directa>

[9]. Lagoa Caridad, Federico Borja, Pablo Montero Manso, Carmen Pena Lourés, Juan Padrón González, and Ismael Teijeiro Flores. Interfaces Gráficas de Usuario. [cited 13 March 2009]. Available from world wide web: <http://sabia.tic.udc.es/gc/Contenidos

%20adicionales/trabajos/Interfaces/enlightment/index.html>

[10]. Maner, Walter. Prototipado. [Zaragoza, España]: Departamento de Ingeniería de Diseño y Fabricación, February 2000 [cited 13 March 2009]. Available from world wide web: <http://www.sidar.org/recur/desdi/traduc/es/visitable/maner/Prototipado.htm>

[11]. Lacalle, Alberto. Prototipos de interfaces de usuario. July 2006 [cited 4 April 2009]. Available from world wide web: <http://albertolacalle.com/hci_prototipos.htm>

[12]. Peñalver Romero, Gladys Marsi. MA-GMPR-UR2 Metodología Ágil para proyectos de Software Libre. June 2008

[13]. Pérez Roldán, Dayron. Sistema de Clonación y Distribución de Imágenes de Sistemas Operativos. June 1008

[14]. Arquitectura en capas. In *Wikipedia.org* Available from world wide web: <http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles>

[15]. Martin Olivera, Yonnis Pablo, and Roberto Ferrer Obregón. *Galm*. [cited 13 March 2009]. Available from world wide web: <http://gforge.f10.uci.cu/projects/galm/>

Bibliografía

Dia. [cited 20 March 2009]. Available from world wide web: <http://live.gnome.org/Dia>

Microsoft Office Visio. Microsoft [cited 20 March 2009]. Available from world wide web: <http://office.microsoft.com/en-us/visio/default.aspx>

Microsoft Windows. Microsoft [cited 20 March 2009]. Available from world wide web: <http://www.microsoft.com/WINDOWS/>

Pencil. [cited 20 March 2009]. Available from world wide web: <http://www.evolus.vn/Pencil/>

UML. [cited 20 March 2009]. Available from world wide web: <http://www.uml.org/>

Visual Paradigm. [cited 20 March 2009]. Available from world wide web:<http://www.visual-paradigm.com/>

Mozilla. Mozilla Firefox, Mozilla [cited 20 March 2009]. Available from world wide web: <http://www.mozilla.com/en-US/firefox/>

Olivera, Yonnis Pablo Martin, and Roberto Ferrer Obregón. Galm. [cited 13 March 2009]. Available from world wide web: <http://gforge.f10.uci.cu/projects/galm/>

Sun Microsystems. OpenOffice.org. Sun Microsystems [cited 20 March 2009]. Available from world wide web: <http://es.openoffice.org/>

The GNOME Foundation. GNOME. [cited 20 March 2009]. Available from world wide web: <http://www.gnome.org/>

Qt.Trolltech [cited 20 March 2009]. Available from world wide web: <http://www.qtsoftware.com/products/>

Free Software Foundation. Licencia GPL o GNU General Public License version 3. June 2007. [cited 13 March 2009]. Available from world wide web: <http://www.gnu.org/copyleft/gpl.html>

NOKIA Corporation. QtCreator. NOKIA [cited 20 March 2009]. Available from world wide web: <http://www.qtsoftware.com/products/developer-tools>

Eclipse. Fundación Eclipse Available from world wide web: <http://www.eclipse.org/>

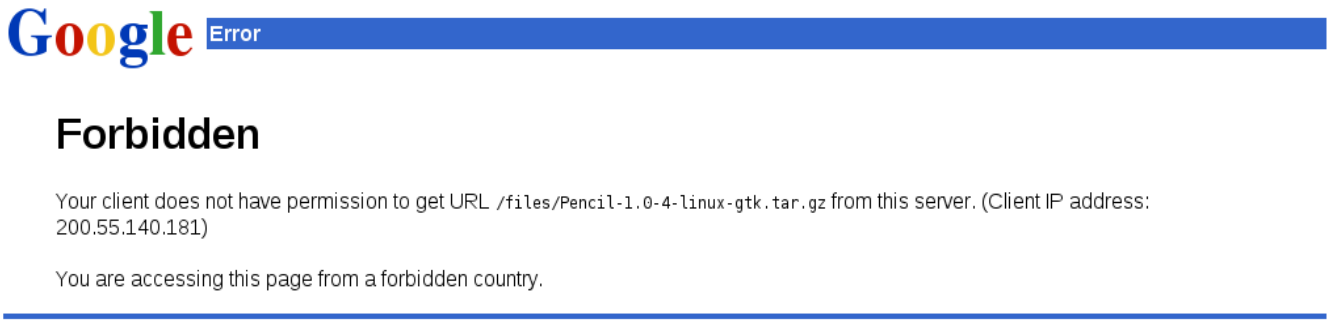
The Gimp. [cited 20 March 2009]. Available from world wide web: <http://www.gimp.org/>

Inkscape. [cited 20 March 2009]. Available from world wide web: <http://www.inkscape.org/>

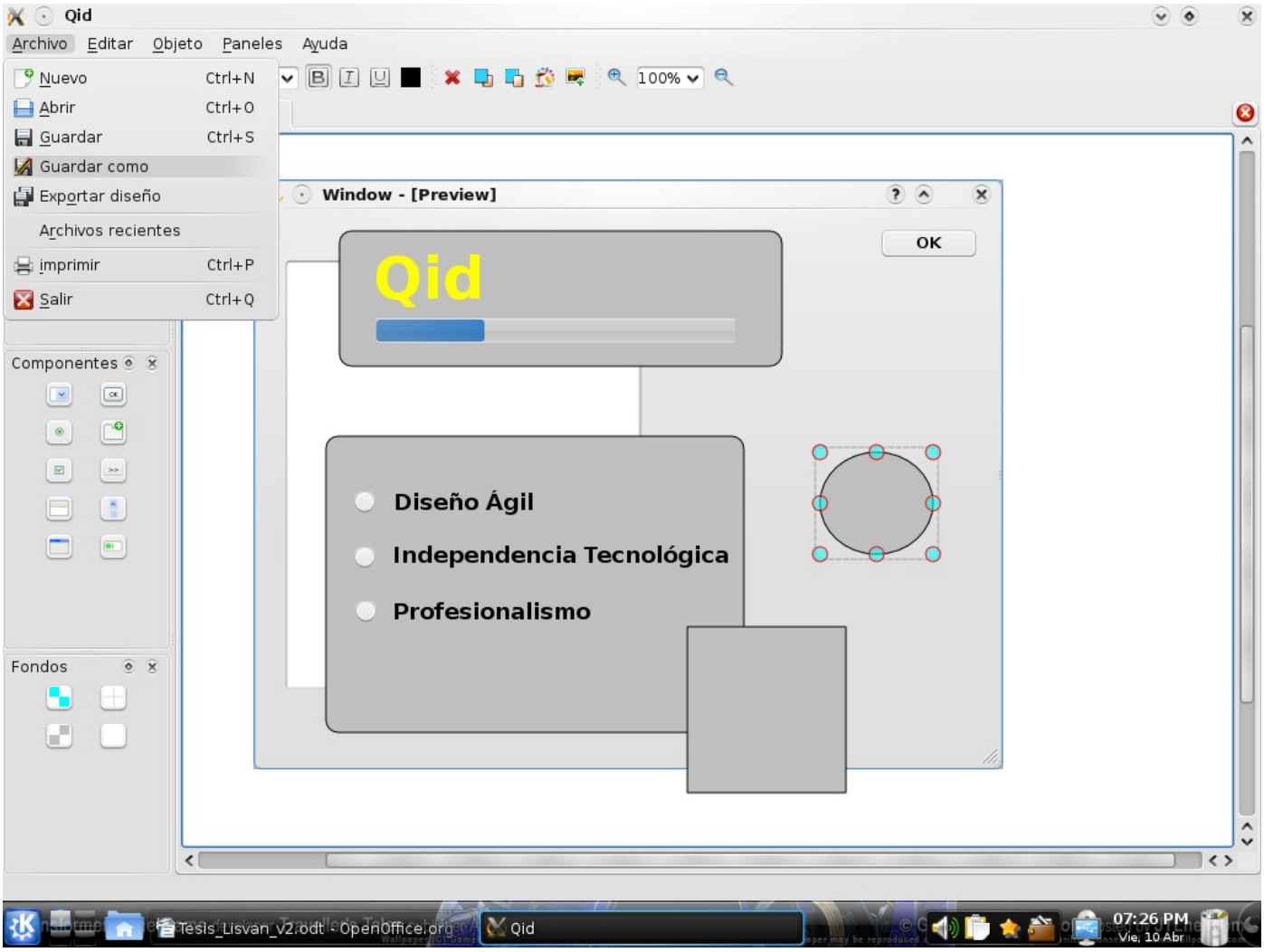
Tutorial de Qt 4.3 [En línea] Available from world wide web: http://doc.trolltech.com/4.3/tutorial.html

Anexos

Anexo No 1. Políticas de Google para descargar archivos o fragmentos de código desde nuestro país.



Anexo No. 2. Interfaz de la aplicación Qid.



Glosario de Términos.

API: Interfaz de Aplicación del Programa (del inglés Application Program Interface). Grupo de rutinas del sistema operativo o de una aplicación que definen cómo invocar cualquier servicio desde un programa.
Empírico: Perteneciente o relativo a la experiencia.
EPS: PostScript encapsulado. Formato de archivo gráfico.
Experiencia de usuario: Conjunto de factores y elementos relativos a la interacción del usuario, con un entorno o dispositivo concretos, cuyo resultado es la generación de una percepción positiva de dicho servicio, producto o dispositivo.
Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
GPL: La GNU General Public License (inglés: Licencia Pública General) es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es Software Libre.
Heurística: capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines.
HTML: siglas de HyperText Markup Language (*Lenguaje de Marcas de Hipertexto*), es el lenguaje de marcado predominante para la construcción de páginas web.
IDE: Entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE'). Es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.
Java: un lenguaje de programación de alto nivel, orientado a objetos.
JPEG: es un método comúnmente utilizado para la compresión de imágenes fotográficas.
Libre: O Software Libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.
Mac OS X: es una línea de sistemas operativos computacionales desarrollada, comercializada y vendida

por Apple Inc.

Paradigma: Modelo o patrón.

PDF: *Portable Document Format*, formato de documento portátil. es un formato de almacenamiento de documentos.

PNG: *Portable Network Graphic*. es un formato gráfico basado en un algoritmo de compresión sin pérdidas para bitmaps.

Privativo: El software no libre (también llamado software propietario, software privativo, software privado, software con propietario o software de propiedad) se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o cuyo código fuente no está disponible o el acceso a éste se encuentra restringido.

SDK: Kit de Desarrollo de Software (SDK del inglés Software Dvelopment Kit).

Software Libre: software que brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

SQL: Lenguaje de Consulta Estructurado (SQL del inglés Structured Query Language).

Subversion: es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido.

SVG: es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados.

Validación de requisitos: Conjunto de actividades que tienen como finalidad examinar los requisitos para asegurarse de que definen el sistema adecuado (el sistema que el cliente/usuario espera).

Widgets: pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o Widget Engine. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual**.**

XML: es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).