Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Sistema de Composición de Imágenes para Impresión Digital

Tesis que para obtener el título de:

Licenciado en Ciencias de la Computación

Presenta:

Oscar Eduardo Romero Meléndez

Asesor:

Dr. David Eduardo Pinto Avendaño

Febrero 2013

DEDICATORIAS Y AGRADECIMIENTOS

A mis padres Rosa y Oscar.

Porque sin su apoyo este trabajo de tesis no tendría lugar. Por su paciencia, sus consejos, su confianza, su comprensión y su cariño.

A mis hermanas Rosalía y Alejandra.

Por su ayuda y por escuchar siempre mis locuras.

A mis maestros.

Dr. David por su motivación para la culminación del presente trabajo. Al Dr.Manuel Martín y a la Maestra Yalú por brindarme los conocimientos más interesantes en la carrera.

A mis amigos.

Con los que he compartido buenos momentos, me han escuchado y me han aconsejado en este viaje.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la actualidad es sencillo adquirir imágenes digitales gracias a la evolución y aparición de distintos dispositivos como cámaras digitales, cámaras de video, ipod's, smartphones y tablets. Asimismo, se ha incrementado la calidad o resolución de las imágenes que se pueden obtener con dichos aparatos. Estos aparatos resultan cada vez más económicos, pero la característica principal es su tamaño, son pequeños y por lo tanto portables. Estas propiedades permiten que una gran cantidad de personas cuente con alguno de estos dispositivos y tenga la capacidad de capturar imágenes que difícilmente tomaría con una cámara fotográfica tradicional.

De esta forma tenemos un escenario donde una cantidad considerable de personas puede obtener volúmenes enormes de imágenes digitales y con diferentes resoluciones. Gracias a esta comodidad de obtener un buen número de imágenes, resulta común la necesidad de editarlas.

En este trabajo de tesis se aplicarán diversos métodos de manipulación de imágenes con la finalidad de crear un sistema de composición de imágenes útil para impresiones digitales de diferentes dimensiones para la empresa masimpresiones.com. El sistema contará con un catálogo de imágenes dividido en cuatro categorías: fondos, máscaras, marcos y cliparts. El usuario final también será capaz de utilizar sus propias imágenes seleccionándolas a través de un panel de navegación de su sistema de archivos. Todas estas imágenes podrán aplicarse para la composición del proyecto, todo ello en un solo lienzo con el objetivo de crear una sola imagen PNG que será usada para su impresión digital.

Una de las características importantes del proyecto, es que este tendrá que almacenar la imagen final únicamente en un equipo servidor. Por su parte, el proyecto se guardará de manera intermedia en un archivo con formato XML tanto en el equipo cliente como en el equipo servidor.

Adicionalmente, el software verificará en cada inicio por actualizaciones en el catálogo de imágenes y será notificado al usuario para su descarga. Es importante señalar que el administrador del sistema podrá añadir nuevas imágenes al catálogo de manera sencilla y tales cambios se verán reflejados sin necesidad de reinicio del servidor web.

Finalmente, será posible agregar en el proyecto de composición de imágenes cualquier imagen que el usuario tenga disponible en su equipo. El sistema de software será completamente modular y con un desarrollo de software basado en versiones, lo cual facilitará en su momento la generación o modificación de módulos del producto de software.

El documento se encuentra distribuido de la siguiente forma:

En el capítulo I se presenta el marco teórico para todo aquel lector que desee conocer conceptos importantes para comprender a detalle el desarrollo de este trabajo. El capítulo II muestra el análisis y el diseño de la aplicación, se identifican y describen los objetos en el dominio del problema, se determinan las responsabilidades e interacciones de cada objeto y se definen los objetos que serán implementados en el lenguaje de programación Java. El capítulo III muestra la implementación del sistema y por último se presentan las conclusiones y perspectivas de este trabajo de tesis.

CAPÍTULO 1. MARCO TEÓRICO

El marco teórico de este proyecto de tesis se enfoca en presentar las nociones básicas de cada uno de los diferentes temas que se abordarán para que el lector pueda adquirir ciertos conocimientos y familiarizarse con la terminología utilizada a lo largo del desarrollo.

Entre los temas que se presentan se encuentran: procesamiento digital de imágenes, cliente rico, framework, XML, UML, Bases de Datos, Plataforma NetBeans, Servicios Web, Sistema de control de versiones y Programación Orientada a Objetos. Todos ellos tienen su aplicación práctica en alguna parte del proyecto y permitirán el desarrollo de las diversas partes que componen el sistema de composición de imágenes.

1.1 PROCESAMIENTO DIGITAL DE IMÁGENES

El procesamiento digital de imágenes consiste en un conjunto de técnicas para el tratamiento y análisis de las imágenes. El tratamiento se refiere a la aplicación de operaciones sobre la imagen para transformarla, restaurarla y mejorarla. El análisis consiste en la extracción de propiedades y características de las imágenes.

1.1.1 REPRESENTACIÓN DE LA IMAGEN

Una imagen puede ser expresada mediante una función bidimensional $f(x,y)$, donde (x,y) denota la ubicación espacial en la imagen y el valor de la función $f(x,y)$ es proporcional a la luminosidad de la imagen en el punto (x,y). En la literatura de Ciencias de la Computación, la función $f(x,y)$ es llamada función de intensidad de la imagen.[1]

Una imagen digital es una función $f(x,y)$ que ha sido convertida a valores discretos tanto en coordenadas espaciales como en luminosidad. Una imagen digital puede ser expresada mediante una matriz $\{f(i,j),\ i = 1,2,\ldots,n_1,\ j = 1,2,\ldots,n_2\}$ donde i es el índice de filas y j de las columnas, como se muestra a continuación:

$$\begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,n_2) \\ f(2,1) & f(2,2) & \dots & f(2,n_2) \\ \vdots & \vdots & \ddots & \vdots \\ f(n_1,1) & f(n_1,2) & \dots & f(n_1,n_2) \end{bmatrix}$$

Cada elemento es llamado *pixel*, una abreviatura de "picture element". Los pixeles identifican el nivel de gris en cada punto de la imagen.

Para el registro de una imagen se acostumbran usar formatos que están referenciados por el número de bits que son utilizados para la discretización de la cantidad de luz recibida. Cuando se trabaja solo la intensidad de la luz recibida en el detector se dice que la imagen ha sido adquirida en tonos de gris. Actualmente la mayor parte de los dispositivos simples permiten adquirir la imagen con una profundidad de 8 bits, los cuales permiten representar 256 niveles de gris, el rango es el típico [0…255] [1L]. Esto quiere decir que cada pixel está asociado a un valor entero que se encuentra normalmente entre 0 y 255, siendo el valor más alto correspondiente a la máxima brillantez y el mínimo la máxima oscuridad.

Los dispositivos de hoy en día realizan el registro la imagen en colores, de manera que para cada punto en *f*(*x*,*y*) se obtiene un pixel con tres componentes, RGB (Red, Green, Blue, colores básicos de la electrónica de video). La imagen en color cuenta con tres canales RGB y cada uno puede tomar valores enteros en el rango de [0…255]. Debido a que cada pixel corresponde a un valor entero, el color se expresa como una combinación lineal de los colores RGB, es decir:

$$C = rR + gG + bB$$

La *resolución* de una imagen equivale al número de pixeles por pulgada utilizados en el proceso de adquisición, éste número es abreviado como *dpi* (dots per inch). La fidelidad de una imagen depende de su resolución, con mayor resolución se obtienen imágenes con más detalle.

1.1.2 OPERACIONES ORIENTADAS AL PUNTO

Dado que la representación digital de una imagen es expresada como una matriz de valores enteros, es posible acceder desde la computadora a cada punto y modificarlo. Este tipo de modificación de la imagen es llamado *transformación puntual* u operación orientada al punto y puede ser aplicada a toda la imagen o a una región de ella.

Sea **x** un pixel de una imagen **I**. Supongamos que la función **y** = *f*(**x**) transforma a un pixel **x** mediante la regla *f,* generando un nuevo valor para él, digamos **y**. Entonces diremos que la nueva imagen **I'.** donde **y** está en **I'**, es el producto de aplicar *f* sobre **I**. Simbólicamente diremos que,

$$\mathbf{I'} = f(\mathbf{I})$$

Para que la transformación *f* no ocasione problemas de representación, si el dominio de **x** está en el intervalo ***D*** = [0, L - 1], donde $L = 2^p$, donde p es la profundidad en bits de la imagen, entonces se va a exigir que **y** se encuentre dentro del dominio **D'**, donde en general **D'** es un subconjunto del dominio **D**. Lo cual implica que el mecanismo de representación de la imagen sobre elementos de la clase **x**, seguirá siendo válido para la clase a la que pertenece **y**. Esta condición permite que los métodos desarrollados para la visualización de la imagen **I** se pueden utilizar para **I'**.

El algoritmo básico de transformación bajo *f* para una región rectangular de **I** definida por

$$R = [i_1, \ldots, i_2, j_1, \ldots, j_2],$$

es el siguiente:

```
for i = i1, i2{
        for j = j1, j2{
                I'[i,j] = f( I[i,j] )
        }
}
```

En el caso de que $i_1 = 0$, $i_2 =$ M (donde M = Ancho de la imagen - 1), $j_1 = 0$, $j_2 =$ N (donde N = Alto de la imagen - 1); el proceso modificará a toda la imagen.

Al cambiar *f* la transformación será diferente. Si definimos la composición de transformaciones dela manera habitual, tendremos que:

$$f_1 \text{ o } f_2 (\mathbf{I}) = f_1 (f_2 (\mathbf{I}))$$

En general al aplicar dos transformaciones a una imagen en diferente orden, no se debe esperar que la imagen resultante sea la misma, es decir, la composición de transformaciones no es conmutativa [2L].

Algunas de las transformaciones puntuales de una imagen son: negativo, filtros de aclarado, filtros de oscurecimiento, sepia y paso a grises

1.2 PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos es un paradigma de programación que se fundamenta en los conceptos de objeto y clase. Es una forma especial de programar más cercana a la manera en la que expresamos las cosas en la vida cotidiana.

1.2.1 OBJETO

Un objeto representa una entidad, ya sea física, conceptual o de software. Por ejemplo:

Entidad física	**Camión**
Entidad conceptual	Proceso químico
Entidad de software	Lista ligada

Todo objeto cuenta con las siguientes características:

Estado	**se refiere al conjunto de valores de sus atributos en un instante e tiempo dado.**
Comportamiento	determina como un objeto reacciona y responde a las peticiones de otros objetos.
Identidad	propiedad que permite al objeto diferenciarse de los demás, cada objeto es único.

1.2.2 CLASE

Las clases son declaraciones de objetos y capturan solo una abstracción principal. Se pueden ver como una plantilla o estructura de un objeto, es decir, la *clase del objeto*. Un objeto es una *instancia* de una clase.

Una clase se compone de dos partes:

- **Atributos**: representan los datos asociados al objeto, es decir, las propiedades o características.
- **Métodos**: acceden a los atributos del objeto y definen el comportamiento del mismo.

1.2.3 CARACTERÍSTICAS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

1.2.3.1 ABSTRACCIÓN

Es una propiedad de la programación orientada a objetos que se refiere a la capacidad de concentrar las propiedades y comportamientos necesarios de un objeto para su correcta representación dentro del sistema. La abstracción es determinante durante el proceso de análisis y diseño orientado a objeto, ya que permite modelar clases de acuerdo al contexto del problema que se pretende resolver.

1.2.3.2 ENCAPSULAMIENTO

El encapsulamiento es una propiedad importante de la programación orientada a objetos que consiste en reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción,

1.2.3.3 POLIMORFISMO

Es la capacidad de tener métodos con el mismo nombre y comportamientos diferentes. Es conocido como la sobre escritura de métodos y la sobrecarga de operadores. De manera

1.2.3.4 HERENCIA

Es la capacidad de los elementos de una jerarquía de transmitir sus características desde los niveles más abstractos a los más concretos. Es una herramienta importante en aspectos del desarrollo de aplicaciones tales como *organización del diseño*, *reutilización de clases* y *mejora del mantenimiento*.

1.2.3.5 MODULARIDAD

Es la capacidad de dividir una tarea compleja en partes más sencillas y de manejar. Cada parte es llamada módulo y debe contar con las propiedades de *alta cohesión* y *bajo acoplamiento*.

La *cohesión* es la medida de especialización con la que cuenta un objeto dentro de un sistema, entre más alta sea es mejor, es decir, todos los elementos de un módulo han de tener relación.

El acoplamiento es la medida con lo que un objeto depende de otro para funcionar, entre menor sea esta, es mejor.

1.3 FRAMEWORK

En programación un framework o marco de trabajo es una estructura de software destinada a servir como soporte o guía para la construcción de aplicaciones. Un framework es una plataforma de software usada para desarrollar aplicaciones, productos y soluciones que proporciona funcionalidad genérica para el tipo de aplicación para el que fue construido.

El propósito de un framework es proveer una estructura definida mediante la cual el software pueda ser construido de manera organizada de acuerdo a prácticas sanas de programación y a patrones de desarrollo. Normalmente, proporciona un conjunto de APIs y herramientas que ayudan al desarrollador durante la tarea de construcción y creación.

Es bien sabido que para el desarrollo de aplicaciones Web existe un conjunto de frameworks que facilitan el trabajo del programador proporcionándole un conjunto de características que son comunes a todas las aplicaciones y de las cuales él puede disponer a través de ellos con mucha facilidad, permitiéndole enfocarse en resolver los requerimientos relacionados con el dominio del problema y ayudándole a ser más productivo. En contraste con la variedad de frameworks existentes para el desarrollo Web, hay un conjunto muy reducido de estos para la creación de software de escritorio, entre los cuales se encuentra el *Framework de Aplicaciones Swing* (*SAF del inglés Swing Application Framework*), el cual, por cierto, ha dejado de tener soporte desde hace algún tiempo atrás [3L].

1.4 CLIENTE RICO

En la arquitectura cliente servidor el término *cliente rico* es utilizado para referirse a aquellas aplicaciones de software en las que el procesamiento de la información ocurre principalmente del lado del cliente.

En el caso particular de las aplicaciones ricas de dos capas, cliente-servidor, el cliente generalmente es una aplicación de escritorio con una variedad de componentes gráficos que facilitan la interacción con el usuario, partiendo de la captura de datos hasta entrega de informes resultado del procesamiento de los mismos. Dicha aplicación realiza operaciones sobre los datos ingresados por el usuario y, en caso de ser necesario, los hace persistentes mediante el uso de un servidor de base de datos. Es así que el servidor realiza una tarea mínima en comparación con la responsabilidad de la aplicación cliente de proporcionar diferentes funcionalidades al usuario al entregarle resultados correctos [4L].

Clientes ricos son típicamente desarrollados sobre un framework que ofrece un punto de partida básico sobre el que el usuario puede armar partes lógicamente relacionadas, que son llamados *módulos*. Idealmente, soluciones no relacionadas pueden trabajar juntas, de este modo todos los módulos parecen haber sido creados como un todo. Por encima y más allá de todo esto, clientes ricos tienen la ventaja de que son fáciles de distribuir y actualizar, por ejemplo mediante la vía de una función de actualización automática en línea dentro del mismo cliente o a través de un mecanismo que permita al cliente rico iniciar a través de internet.

Algunas de las características de un cliente rico son:

- Arquitectura de aplicación flexible y modular
- Independencia de la plataforma
- Adaptabilidad al usuario final
- Habilidad de trabajar en línea y desconectado
- Simplificación de la distribución al usuario final
- Simplificación de la actualización

1.4.1 PLATAFORMA DE CLIENTE RICO

Una plataforma de cliente rico es un entorno de ciclo de vida de aplicación. Una base para aplicaciones de escritorio. La mayoría de aplicaciones de escritorio tienen características similares tales como menús, barras de herramientas, barras de estado, barras de progreso, ajustes de personalización, pantalla de bienvenida, sistemas de ayuda, etc. Para estas y otras características

típicas de una aplicación cliente, una plataforma de cliente rico proporciona un framework en el cual las características pueden ser aplicadas simple y rápidamente.

El aspecto más importante de una plataforma de cliente rico es su arquitectura. Aplicaciones basadas en plataformas de cliente rico son escritas en forma de módulos, dentro de los cuales las partes lógicamente coherentes de una aplicación se aíslan. Un módulo es descrito declarativamente y automáticamente cargado por la plataforma. Como resultado, no existe vinculación explícita necesaria entre el código fuente y la aplicación. De esta manera, una relación de bajo acoplamiento es establecida entre la funcionalidad independiente de los módulos, por medio de la cual la extensibilidad dinámica de la aplicación y la habilidad para intercambiar sus partes constituyentes se simplifica enormemente. Una plataforma de cliente rico también libera al desarrollador de preocuparse de tareas que poco tienen que hacer con la lógica de negocio de la aplicación [2].

Algunas de las ventajas de este tipo de plataformas son las siguientes:

- Modularidad ofrecida mediante la arquitectura de cliente rico que implica alto grado de robustez y valor para el usuario final.
- Reducción en tiempo de desarrollo. Cada plataforma de cliente proporciona una multitud de interfaces de programación de aplicaciones (APIs) para el desarrollo de aplicaciones de escritorio.
- Interfaz de usuario consistente. La usabilidad de una aplicación siempre es una preocupación fundamental, particularmente cuando la aplicación es destinada a ser usada por profesionales en un campo específico. Una plataforma de cliente rico pone a disposición un framework para desplegar la interfaz de usuario, con énfasis en la consistencia, accesibilidad y usabilidad.
- Actualización de la aplicación. Usando una plataforma de cliente rico es posible distribuir rápida y eficientemente nuevos o actualizados módulos para usuarios finales. Como resultado, no todos los clientes de una aplicación necesitan ser informados por los desarrolladores para cambiar a una nueva versión. Las actualizaciones pueden ser distribuidas e instaladas en forma de módulos, por lo que distintas características pueden ser desarrolladas y entregadas por equipos de desarrollo de forma independiente.

1.5 PLATAFORMA NETBEANS

La Plataforma NetBeans es una framework para desarrollar aplicaciones de escritorio complejas en el lenguaje de programación Java y está basado en la librería gráfica swing. Es una plataforma de cliente rico y por lo tanto tiene todas sus ventajas.

La Plataforma NetBeans hace fuerte hincapié sobre la construcción del software de forma modular. Permite al desarrollador preocuparse por la lógica y desplegar aplicaciones rápidamente pudiendo extender la funcionalidad a medida que pasa el tiempo.

Algunas de las características de la plataforma son:

- Al estar basada en Java, los proyectos desarrollados no dejan de ser multiplataforma.
- Sistema de ventanas práctico para desarrollar las interfaces de usuario.
- Sistema de ficheros virtual en el que se cargan los módulos.
- Su licencia permite construir tanto aplicaciones comerciales como de código abierto.

El bloque de construcción básica de la Plataforma NetBeans es el módulo. Un módulo consiste en una colección de clases funcionalmente relacionadas, junto con una descripción que el módulo expone, así como una descripción de otros módulos que necesita para funcionar.

La Plataforma ayuda a perder acoplamiento de módulos en una aplicación comunicándolos sin ser dependientes de otros módulos.

Un módulo es descrito por los datos de su archivo manifiesto, junto con los datos especificados en archivos XML relacionados. Utilizando archivos XML, la Plataforma NetBeans conoce que módulos están disponibles, así como sus ubicaciones y los contratos que necesitan ser satisfechos por ellos para permitir ser cargados.

1.5.1 NETBEANS RUNTIME CONTAINER

El núcleo de la Plataforma NetBeans se conoce como *NetBeans Runtime Container* y lleva a cabo la carga de los módulos de la aplicación dinámica y automáticamente, después es responsable de ejecutar la aplicación. Además, es la base de la Plataforma NetBeans ya que consiste de un subconjunto mínimo de módulos que requieren todas las aplicaciones de la plataforma:

Módulo	Descripción
Bootstrap	Este módulo es ejecutado antes que cualquier otro. Carga el módulo startup.
Startup	Despliega la aplicación e inicia el sistema de módulos y sistema de archivos.
Module System	Es responsable de la definición de módulos, así como sus configuraciones y dependencias.

File System	Este módulo pone a disposición un sistema virtual de datos, con acceso independiente de la plataforma. Principalmente utilizado para cargar recursos de módulos en una aplicación.
Utilities	Proporciona componentes básicos tales como aquello requeridos para comunicación intermodular.

1.6 SISTEMA DE CONTROL DE VERSIONES[1]

La gran mayoría de los proyectos de desarrollo de software involucran muchos desarrolladores diferentes trabajando concurrentemente sobre una variedad de conjuntos de archivos. Por lo tanto es crítico que los cambios realizados por estos desarrolladores sean rastreados de modo que se pueda saber quién es responsable de cualquier cambio y ser capaz de ver el código fuente como era una hora atrás, una semana o un año. Además es importante tener la habilidad de fusionar las contribuciones de aquellos desarrolladores en un único lugar. Aquí es donde entra el sistema de control de versiones.

Las funcionalidades básicas de un sistema de control de versiones son el mantener el rastreo de cambios de estado de archivos a través del tiempo y combinar las contribuciones de varios desarrolladores. Los sistemas de control de versiones frecuentemente soportan esta característica mediante el almacenamiento de un historial de cambios realizados e a través del tiempo por distintas personas. De esta forma es posible revertir aquellos cambios y ver como eran los archivos antes de que fueran aplicados. Adicionalmente, un sistema de control de versiones proporciona facilidades para la fusión de cambios, utilizando uno o más métodos que van del bloqueo de archivos a la integración automática de cambios conflictivos.

1.6.1 CARACTERÍSTICAS

1.6.1.1 INTEGRIDAD DE DATOS

Un buen sistema de control de versiones ayuda a proteger la integridad de los datos del proyecto de desarrollo. Manteniendo un historial de revisiones, no existe preocupación de que el código que es eliminado en una edición de un día se perderá al termino de una semana.

Contar con un repositorio central del proyecto también ayuda a respaldar datos. Si los desarrolladores regularmente guardan sus cambios en el sistema de control de versiones es posible realizar copias de seguridad por bloques.

[1] Información obtenida de la referencia [4]

1.6.1.2 PRODUCTIVIDAD

Liberando a los desarrolladores de la tarea de integración manual un sistema de control de versiones puede aumentar la productividad. Dado que los proyectos crecen más de una o dos personas aún el proceso mejor organizado perderá incontables horas hombre hacia la integración del trabajo de varios desarrolladores. Con un sistema de control de versiones los desarrolladores son capaces de probar cambios contra el último trabajo de sus compañeros identificando y corrigiendo conflictos antes de que lleguen a ser inmanejables. También son capaces de experimentar más fácilmente. Libres de crear ramas del proyecto y modificar código sin preocuparse de que los cambios afectarán la estabilidad del proyecto principal o del trabajo de otros. Si un cambio experimental rompe algo, puede ser revertido o comparado con el código original rápida y fácilmente.

Un sistema de control de versiones también protege contra la pérdida de productividad para trabajo re-implementado, no sólo evitando pérdidas de datos, sino también haciendo el trabajo del desarrollador fácilmente disponible para otros desarrolladores en el proyecto. Si los desarrolladores son capaces de ver fácilmente donde otros están trabajando en el proyecto, será menos probable duplicar esfuerzos.

1.6.1.3 RESPONSABILIDAD

En cualquier proceso de desarrollo es importante conocer exactamente quién agregó cada fragmento de código al proyecto, así como cuándo lo hizo y quienes han hecho modificaciones desde entonces. Este tipo de responsabilidad es importante no solo por razones técnicas, sino también para propósitos de defensa legal.

1.6.1.4 DESARROLLO EN RAMAS

Conforme el proyecto progresa el desarrollo en ramas ocurrirá naturalmente. Versiones antiguas tendrán que ser soportadas con corrección de errores. Nuevos proyectos pueden ser separados de las bases de código existente para atender a mercados emergentes. Sin importar la razón, la ramificación ocurrirá y a menos que la relación entre ramas sea cuidadosamente mantenida, tenderán a divergir irreconciliablemente. Temas que están resueltos en una rama irán sin resolver en otra. Características implementadas en una rama divergente no podrán ser utilizadas en el tronco principal. En general, aún manteniendo una apariencia de consistencia entre ramas de desarrollo diferentes, el mantenimiento es complicado.

Si se usa de manera organizada y consistente, las características de ramificación incorporados en la mayoría de los sistemas de control de versiones pueden reducir las complicaciones asociadas con el mantenimiento de las ramas divergentes de desarrollo en un proyecto. Utilizando los registros de confirmación generados por el sistema, así como su capacidad de combinar cambios de una rama con otra, los cambios que son aplicables a múltiples ramas

pueden ser claramente implementados en un única rama y entonces aplicados a otras ramas. Similarmente una nueva característica agregada a una rama puede ser migrada a otras ramas donde puede ser útil.

1.6.1.5 DESARROLLO RÁPIDO

Las metodologías de desarrollo de software recientes se han movido hacia el desarrollo rápido y flexible, con procesos como programación extrema y desarrollo ágil siendo adoptados con mayor frecuencia. Estos métodos de desarrollo rápido acentúan políticas de pequeños cambios incrementales y frecuente refactorización. Mediante el uso de buenas prácticas de control de versiones un proyecto mantendrá historias de código de gran utilidad que delimitan las muchas formas que el código desarrollado puede tomar. Adicionalmente, el repositorio central de un sistema de control de versiones es perfecto para automatizar los frecuentes sistemas de construcción llamados por un proceso ágil.

1.6.2 ELEMENTOS DEL CONTROL DE VERSIONES

El control de versiones es esencialmente rastreo, control y combinación de diferentes versiones de un proyecto en el tiempo. Los sistemas de control de versiones son herramientas de software complejas con una gran cantidad de características diferentes que varían ampliamente de sistema a sistema. Conceptualmente, sin embargo, son de hecho bastante simples y la mayoría de los sistema de control de versiones pueden ser comprendidos con un entendimiento de de unos cuantos conceptos básicos.

1.6.2.1 EL REPOSITORIO Y EL DIRECTORIO DE TRABAJO

Algunos sistemas de control de versiones almacenan sus proyectos versionados en un repositorio central. El repositorio puede ser simplemente una estructura de directorio sobre un servidor con cada archivo versionado almacenado de forma separada, o puede ser una base de datos conteniendo entradas para los distintos archivos en un proyecto. Incluso puede ser un complejo sistema distribuido que almacene redundantemente el proyecto versionado en todo el mundo.

Sin importar de como luce el repositorio , la coincidencia entre sistemas de control de versiones es que los desarrolladores no trabajan directamente sobre los archivos del repositorio. En su lugar, tienen algún tipo de directorio de trabajo accesible desde su maquina de desarrollo donde puede realizar modificaciones locales.

Los directorios de trabajo generalmente permiten a desarrolladores trabajar individualmente de manera local, agregando y probando cambios como sea necesario durante el proceso de desarrollo. Una vez que un cambio o conjunto de cambios es considerado completo, el desarrollador es capaz de confirmar los cambios de nuevo en el repositorio, donde llegan a ser parte del proyecto.

Una vez que los cambios han sido confirmados al repositorio, los otros desarrolladores que están trabajando en el proyecto son capaces de actualizar sus copias de trabajo para incluir las últimas versiones de los archivos del proyecto. Esto permite a los demás desarrolladores probarlos nuevos cambios con las modificaciones locales no confirmadas en sus directorios de trabajo y resolver problemas como sea necesario.

1.6.2.2 REVISIONES

Los sistemas de control de versiones no solo almacenan el estado más reciente del proyecto, también guardan un historial de las modificaciones del proyecto. Siempre que un desarrollador confirme cambios sobre el proyecto estos cambios son almacenados en una revisión. Dependiendo de sistema de control de versiones, las revisiones serán puntos globales que hacen referencia al estado del repositorio completo en un momento dado o existirán revisiones de nivel de archivo que hacen referencia al estado de un archivo individual.

En un sistema de revisión de nivel de archivo cada archivo tiene un historial de revisión independiente del resto del proyecto. Este tipo de esquema de revisiones tiende a ser difícil de manejar y puede hacer que sea difícil dar seguimiento al estado general del proyecto en un punto particular en el tiempo.

Otros sistemas de control de versiones tienen revisiones que son globales a través de todo el repositorio. En este tipo de sistemas, una modificación confirmada sobre un único archivo incrementaría las revisiones de todos los archivos en el repositorio. Esto da una gran ventaja sobre las revisiones de nivel de archivo, ya que no es necesario realizar el seguimiento de relaciones entre diferentes revisiones de diferentes archivos.

1.6.2.3 REGISTROS

Mantener un registro de los cambios del código es importante, pero también lo es saber qué líneas de código fueron agregadas y cual es la razón de agregarlas.

Una forma de realizar el seguimiento de cambios lógicos seria tomar notas o escribir comentarios de lo que está sucediendo en los bloques de código fuente en cada confirmación. Los comentarios pueden ser difíciles de encontrar si alguien desea simplemente saber que cambió de una versión a otra.

Una mejor solución de dar seguimiento a los cambios lógicos del proyecto es incluir un registro junto con cada confirmación del repositorio. El registro permitiría a los desarrolladores describir en lenguaje natural la razón de los cambios que están cometiendo.

1.6.2.4 ETIQUETAS

La mayoría de los sistemas de control de versiones proporcionan un medio por el cual es posible etiquetar revisiones, de modo que se puede hacer referencia a ellas posteriormente. Esto libera al desarrollador de la dependencia de referencias con una relación contextual pobre, tales como números de revisiones o fechas. Las etiquetas pueden ser colocadas como hitos de desarrollo, esto permite continuar con el desarrollo de un proyecto sin afectar la capacidad de que otro desarrollador navegue en el código fuente del proyecto en ese hito.

1.6.2.5 RAMAS

Las etiquetas son útiles, pero no son suficientes cuando el código que ha sido etiquetado necesita cambiar, lo que provoca desviarse del tronco principal de desarrollo. Los sistemas de control de versiones soportan este tipo de divergencias, permitiendo crear rutas paralelas de desarrollo para el repositorio. A estas rutas paralelas se les llama ramas de desarrollo.

En cualquier revisión dada una rama puede ser creada y de esta forma es posible el desarrollo continuo de ese punto en las dos rutas paralelas. Cada rama se puede trabajar de forma independiente y los cambios cometidos en una rama no afectarán a ninguna otra rama del proyecto. Posteriormente, si el equipo de desarrollo decide que un cambio realizado en una rama podría ser útil en una o más de las otras ramas de desarrollo, usualmente es posible fusionar partes de dos ramas.

1.7 BASES DE DATOS[2]

Existen diferentes modelos para el diseño de bases de datos, el modelo de red, el modelo jerárquico y el modelo entidad-relación. En la actualidad el modelo entidad-relación es el más difundido por su facilidad de uso, compresión y manera de poder modelar en él problemas del mundo real en un modelo computacionalmente factible, robusto y bien estructurado guardando la definición de las entidades (objetos del mundo real) y sus relaciones con otras entidades.

El modelo relacional fue propuesto originalmente por E.F. Codd en un ya famoso artículo de 1970. Gracias a su coherencia y facilidad de uso, el modelo se ha convertido en los años 80 en el más usado para la producción de DBMS.

[2] Información obtenida de la referencia [5], [6]

La estructura fundamental del modelo relacional es precisamente esa, “relación”, es decir, una tabla bidimensional constituida por líneas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad.

En realidad, siendo rigurosos, una relación es sólo la definición de la estructura de la tabla, es decir su nombre y la lista de los atributos que la componen. Cuando se puebla con las tuplas se habla de “instancia de relación”.

Las tuplas en una relación son un conjunto en el sentido matemático del término, es decir una colección no ordenada de elementos diferentes. Para distinguir una tupla de otra se recurre al concepto de llave primaria, osea a un conjunto de atributos que permiten identificar unívocamente una tupla en una relación. Naturalmente en una relación puede haber más combinaciones de atributos que permitan identificar unívocamente una tupla (“llaves candidatas”), pero entre éstas se elegirá una sola para utilizar como llave primaria. Los atributos de la llave primaria no pueden asumir el valor nulo (que significa un valor no determinado), en tanto que ya no permitirían identificar una tupla concreta en una relación. Esta propiedad de las relaciones y de sus llaves primarias está bajo el nombre de integridad de las entidades.

A menudo, para obtener una llave primaria “económica”, es decir, compuesta de pocos atributos fácilmente manipulables, se introducen uno o más atributos ficticios, con códigos identificativos unívocos para cada tupla de la relación.

Cada atributo de una relación se caracteriza por un nombre y por un dominio. El dominio indica qué valores pueden ser asumidos por una columna de la relación. A menudo un dominio se define a través de la declaración de un tipo para el atributo, pero también es posible definir dominios más complejos y precisos. Característica fundamental de los dominios de una base de datos relacional es que sean atómicos, es decir, que los valores contenidos en las columnas no se puedan separar en valores de dominios más simples. Más formalmente se dice que no es posible tener atributos multivalor.

Una de las grandes ventajas del modelo relacional es que define también un álgebra llamada álgebra relacional. Todas las manipulaciones posibles sobre las relaciones se obtienen gracias a la combinación de tan sólo cinco operadores: restrict, project, times, union y minus. Por comodidad se han definido también tres operadores adicionales que de todos modos se puedes obtener aplicando los cinco fundamentales: join, intersect y divide. Los operadores relacionales reciben como argumento una relación o un conjunto de relaciones y restituyen una única relación como resultado.

A continuación se presenta una breve descripción de estos operadores.

Restrict: restituye una relación que contiene un subconjunto de las tuplas de la relación a la que se aplica. Los atributos se quedan como estaban.

Project: restituye una relación con un subconjunto de los atributos de la relación a la que viene aplicado. Las tuplas de la relación resultado se componen de las tuplas de la relación original, de manera que siguen un conjunto en sentido matemático.

Times: se aplica a dos relaciones y efectúa el producto cartesiano de las tuplas. Cada tupla de la primera relación está concatenada con cada tupla de la segunda.

Join: se concatenan las tuplas de dos relaciones de acuerdo con el valor de un conjunto de susatributos.

Union: aplicando este operador a dos relaciones compatibles se obtiene una que contiene las tuplas de ambas relaciones. Dos relaciones son compatibles si tienen el mismo número de atributos y los atributos correspondientes en las dos relaciones tienen el mismo dominio.

Minus: aplicado a dos relaciones compatibles restituye una tercera que contiene las tuplas que se encuentran sólo en la primera relación.

Intersect: aplicado a dos relaciones compatibles restituye una relación que contiene las tuplas que existen en ambas.

Divide: aplicado a dos relaciones que tengan atributos comunes, restituye una tercera que contiene todas las tuplas de la primera relación que se puede hacer que correspondan con todos los valores de la segunda relación.

Las bases de datos relacionales efectúan todas las operaciones en las tablas usando el álgebra relacional, aunque normalmente no le permiten al usuario utilizarla.

1.8 MODELADO DE APLICACIONES CON UML[3]

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés) es una de las herramientas más útiles en el mundo del desarrollo de sistemas. Es un lenguaje de modelado

[3] Información obtenida de la referencia [7]

visual que permite a los constructores de sistemas crear planos que capturan sus visiones en una forma estándar, fácil de entender y que provee un mecanismo para comunicar y compartir efectivamente éstas visiones con otros.

Comunicar la visión es de suma importancia. Antes del advenimiento de UML, el desarrollo de sistemas era frecuentemente al azar. Analistas de sistemas tratarían de evaluar las necesidades de sus clientes, generando un análisis de requerimientos en alguna notación que el analista entendiera, dando este análisis a un programador o equipo de programadores con la esperanza de que el producto final fuera el sistema que el cliente quería.

Dado que el desarrollo de sistemas involucra comunicación entre personas, el potencial de error acechaba en cada etapa del proceso. El analista podría haber malentendido al cliente o podría haber generado un documento que el cliente no comprendiera. Para agregar más confusión, los analistas frecuentemente creaban documentos de requerimientos de muchas palabras con los que otros miembros del equipo no podían trabajar fácilmente. Así, los resultados del análisis podrían no haber sido claros para los programadores, quien subsecuentemente podrían haber creado un programa que fuera difícil de usar y que no solucionara el problema original del cliente.

En los primeros día de la computación, algunos programadores se basaron en un análisis profundo del problema en cuestión. Si hacían algún análisis, típicamente era en la parte posterior de una servilleta. Frecuentemente escribían programas desde el principio, creando código sobre la marcha. Aunque esto agregaba un aura de romanticismo y audacia en el proceso, ha demostrado ser inapropiado en el mundo actual de los negocios de alto riesgo.

Hoy en día un plan bien pensado es crucial. Un cliente debe de entender lo que el equipo de desarrollo va a hacer y debe ser capaz de indicar cambios si es que el equipo no ha comprendido completamente las necesidades del cliente. Además, el desarrollo suele ser un esfuerzo en equipo, de modo que cada miembro del equipo tiene que saber dónde encaja su trabajo en el panorama general.

Conforme el mundo llega a ser más complejo, los sistemas basados en computadoras que habitan el mundo también incrementan en complejidad. Frecuentemente involucran múltiples piezas de hardware y software, conectados en red a grandes distancias, ligados a bases de datos que contienen montañas de información. La clave es organizar el proceso de diseño de una forma que analistas, clientes, programadores y otras personas involucradas en el desarrollo del sistema puedan entender.

Así como no es posible construir una estructura compleja como una oficina sin primero crear un plano detallado, no es posible construir un sistema complejo para la oficina sin primero crear un plan detallado. El plan debería ser uno que pudiera mostrarse a un cliente con tanta seguridad como un arquitecto muestra un plano a la persona a quién le paga por la construcción. Ese plan de diseño debería ser resultado de un cuidadoso análisis de las necesidades del cliente.

La necesidad de diseños sólidos ha llevado a la necesidad de una notación de diseño que analistas, desarrolladores y clientes acepten como estándar y UML es esa notación.

1.8.1 COMPONENTES UML

UML consiste de un número de elementos gráficos que combinados forman diagramas. Dado que UML es un lenguaje, tiene reglas para combinar estos elementos.

El propósito de los diagramas es presentar múltiples vistas del sistema; este conjunto de múltiples vistas es llamado modelo. Un modelo UML de un sistema es algo así como un modelo a escala de un edificio junto con una representación artística del edificio. Es importante observar que un modelo UML describe lo que un sistema se supone que debe hacer y no dice cómo implementar el sistema.

1.8.1.1 DIAGRAMA DE CLASES

Los diagramas de clases describen la estructura estática de las clases en el sistema e ilustran atributos, operaciones y relaciones entre las clases. La representación de una clase tiene tres componentes:

Nombre: contiene el nombre de la clase así como el estereotipo que proporciona información acerca de la clase. Ejemplos de estereotipos son <<interface>>, <<abstract>> o <<controller>>.

Atributos: lista los atributos de la clase en formato *nombre:tipo*, con la posibilidad de asignar valores iniciales usando el formato *nombre:tipo=valor*.

Operaciones: lista los métodos para la clase en formato *método(parámetros): tipo de retorno*.

Operaciones y atributos pueden tener su visibilidad anotada como sigue: + public, # protected, - private, ~package.

Ejemplo:

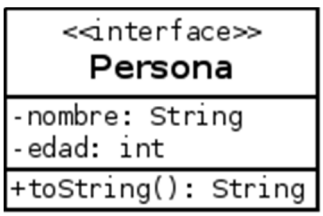


Figura 1.1: Representación de Interface Persona en UML

1.8.1.2 DIAGRAMA DE OBJETO

Diagramas de objetos proporcionan información sobre las relaciones entre instancias de clases en un punto de tiempo particular. Estos diagramas usan algunos elementos de los diagramas de clases.

Típicamente, una instancia objeto se modela usando un simple rectángulo sin compartimentos y con el texto subrayado en el formato *nombreInstancia:Clase*.

1.8.1.3 DIAGRAMA DE COMPONENTES

Diagramas de componentes son utilizados para mostrar cómo los componentes de un sistema están conectados en un nivel más alto de abstracción que los diagramas de clases. Un componente podría ser modelado por una o más clases.

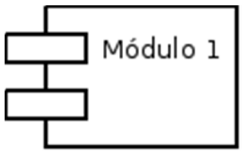


Figura 1.2: Representación gráfica de un módulo en UML

Conectores de montaje pueden ser usados cuando un componente necesita utilizar los servicios proporcionado por otro.

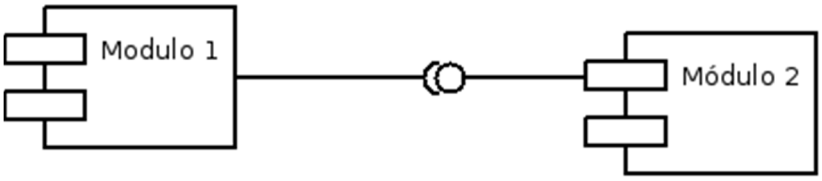


Figura 1.3: Representación de dependencia de módulos en UML

En el diagrama anterior el módulo 1 depende de los servicios que proporciona el módulo 2.

1.8.1.4 DIAGRAMA DE DESPLIEGUE

Diagramas de despliegue modelan la arquitectura en tiempo de ejecución de un sistema en un entorno real. Muestran como las entidades de software son desplegadas sobre nodos físicos de hardware y dispositivos.

Asociación de vínculos entre estas entidades representa comunicación entre nodos y puede incluir multiplicidad.

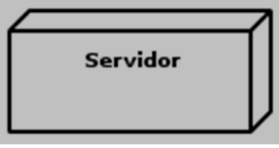
Entidad	Descripción
Nodo 	Elementos de hardware o software mostrados en forma de caja 3D. Los nodos pueden tener muchos estereotipos indicados por un ícono apropiado en la esquina superior derecha.
Artefacto 	Un artefacto es cualquier producto del desarrollo de software, incluyendo código fuente, archivos binarios o documentación. Es representado utilizando un ícono de documento en la esquina superior derecha.

Tabla 1.1: Elementos de un diagrama de despliegue

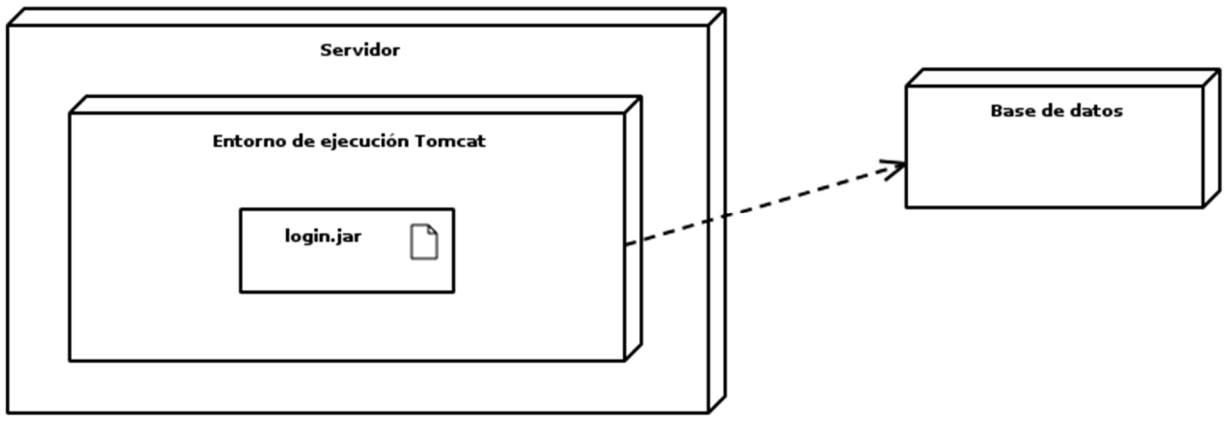


Figura 1.4: Servidor con contenedor web tomcat con un artefacto que depende de la base de datos.

1.8.1.5 DIAGRAMA DE PAQUETES

Diagramas de paquetes muestran la organización de paquetes y los elementos en el interior proporcionan una visualización del espacio de nombres que será aplicado a las clases. Los diagramas de paquetes son comúnmente utilizados para organizar y proporcionar una visión de alto nivel de diagramas de clases.

Figura 1.5: Paquete mx.buap.cs en UML

1.8.1.6 DIAGRAMAS DE CASOS DE USO

Un caso de uso es una descripción del comportamiento de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, los casos de usos son un herramienta valiosa: es una técnica para reunir los requerimientos de un sistema desde el punto de vista del usuario. Obtener información desde el punto de vista del usuario es importante si la meta es construir un sistema que las personas reales puedan usar. El diagrama muestra la interacción de usuarios y otras entidades externas con el sistema que está siendo desarrollado.

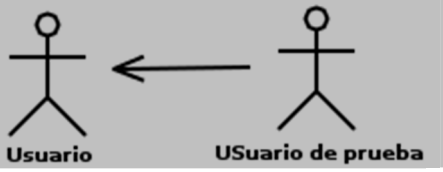
Entidad	Descripción
Actor 	Actores representan entidades externas en el sistema y pueden ser humanas, de hardware u otros sistemas. Relaciones de generalización pueden ser usadas para representar tipos más específicos de actores.
Caso de uso 	Un caso de uso representa una unidad de funcionalidad que puede interactuar con actores externos o relacionados a otros casos de uso.
Límite	Casos de uso están contenidos dentro de un límite del sistema, que es representado con un rectángulo simple. Entidades externas no deben ser colocadas dentro del límite del sistema.

Tabla 1.2: Elementos gráficos de un diagrama de casos de uso

Notación	Descripción

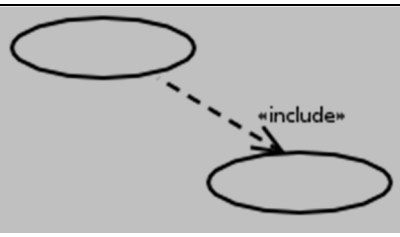
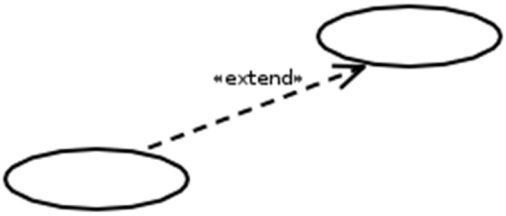
 **Include**	Ilustra que un caso de uso base puede incluir a otro, lo que implica que el comportamiento del caso de uso incluido es insertado en el comportamiento del caso de uso base.
Extends 	Ilustra que un caso de uso particular proporciona funcionalidad adicional al caso de uso base, en algunos flujos alternativos. Esto puede interpretarse en el sentido de que no es necesario para completar el objetivo del caso de uso base.
Generalización	Utilizado cuando existe un caso de uso común que proporciona funcionalidad básica que puede ser usada por casos de uso más especializados.

Tabla 1.3: Relaciones entre casos de uso

Detrás de cada caso de uso debe haber algún texto que lo describa. Las siguientes son secciones típicas en la definición de un caso de uso:

Sección	Descripción
Nombre y descripción	Casos de uso deben tener nombres verbales y tener un breve descripción.
Requerimientos	Esto podría ser un vínculo a una especificación formal externa o a una lista interna de requerimientos que este caso de uso cumple.
Restricciones	Las pre y post condiciones que aplican a la ejecución de este caso de uso.
Escenarios	El flujo de eventos que ocurren durante la ejecución de este caso de uso. Típicamente comienza con una ruta positiva con un número de flujos alternativos referenciado.

Tabla 1.4: Plantilla de descripción de casos de uso

1.8.1.7 DIAGRAMA DE ACTIVIDADES

Los diagramas de actividades capturan el flujo de un programa, incluyendo las principales acciones y puntos de decisión. Estos diagramas son útiles para documentar procesos de negocio.

Sección	Descripción
 **Acción**	Representa un paso en el flujo del programa.
Restricciones	Restricciones de acciones son vinculados en una nota de texto con el formato *<<estereotipo>> restricción*.

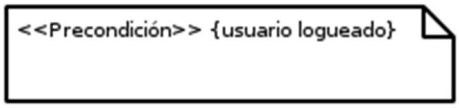
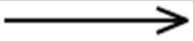
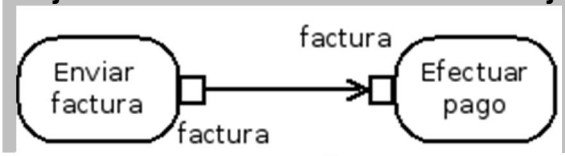
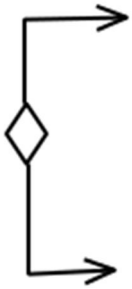
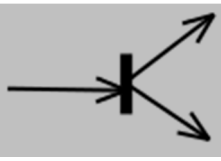
	
Nodo inicial 	El nodo inicial es usado para representar donde el flujo del programa comienza.
Nodo final de actividad 	Representa el fin de todos los flujos de control dentro de la actividad.
Nodo final de flujo 	Representa el fin de un único flujo
Flujo de control 	Representa el flujo de control de una acción a la siguiente como una línea continua con una punta de flecha.
Flujo de objeto 	Si el objeto es pasado entre actividades, una representación del objeto puede ser agregado entre actividades,
Nodo de decisión 	Una forma de rombo anotado es usado para representar decisiones en el flujo de control.
Nodo de bifurcación y unión 	Representado usando una barra horizontal o vertical, un fork node ilustra el inicio de hilos concurrentes.
Partición	Puede ser usado en diagramas de actividades para mostrar actividades realizadas por diferentes actores. Una partición de una actividad se muestra como calles horizontales o verticales.
Región 	Regiones son usadas para agrupar ciertas actividades. Un estereotipo es aplicado a la región para identificar si es iterativa o paralela.

Tabla 1.5: Elementos gráficos de un diagrama de actividades

1.8.1.8 DIAGRAMA DE SECUENCIA

Diagramas de clases y de objetos representan información estática. Sin embargo, en un sistema funcionando los objetos interactúan entre ellos y éstas interacciones ocurren en el tiempo. Los diagramas de secuencia describen cómo interactúan las entidades, incluyendo los

mensajes que son utilizados para esas interacciones. Todos los mensajes son descritos en el orden de ejecución.

Un diagrama de secuencia está compuesto de un número de líneas de vida. Cada entidad tiene su propia columna. El elemento es modelado en la parte superior de la columna y la línea de vida sigue con una línea discontinua.

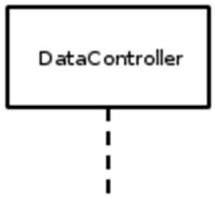
Entidad	Descripción
 **Actor**	Actores representan entidades externas en el sistema. Pueden ser humanos, de hardware u otros sistemas.
Línea de vida general 	Representa una entidad individual en el diagrama de secuencia, desplegada como un rectángulo. Puede tener nombre, estereotipo o podría ser una instancia.
 **Límite**	Elementos de frontera están usualmente en el extremo del sistema, tales como interfaces o lógica de back-end que trata con sistemas externos.
 **Control**	Elementos controladores administran el flujo de información para un escenario. Comportamiento y reglas de negocio son administrados típicamente por estos objetos.
 **Entidad**	Entidades usualmente son elementos que son responsables del almacenamiento de datos. Pueden pensarse como beans u objetos del modelo.

Tabla 1.6 Objetos de línea de vida

El núcleo de los diagramas de secuencia son los mensajes que son enviados entre los objetos modelados. Los mensajes usualmente serán de la forma *nombremensaje(parametros)*. Un rectángulo delgado a lo largo de la línea de vida muestra el tiempo de vida de ejecución para los mensajes del objeto. Los mensajes pueden ser enviados en ambas direcciones y pueden saltar a otras líneas de vida en el camino hacia el destinatario.

Mensaje	Descripción
Síncrono	Un mensaje con una línea continua con punta de flecha

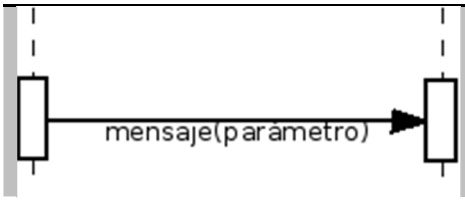
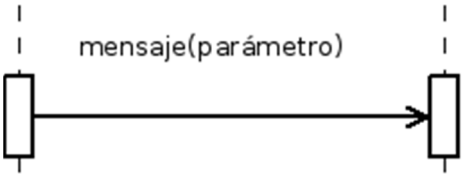
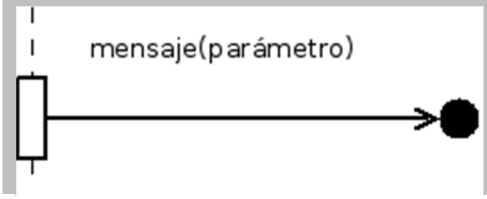
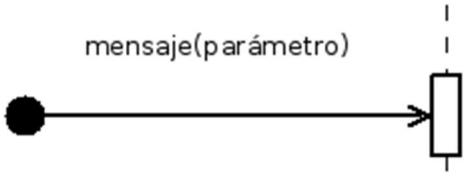
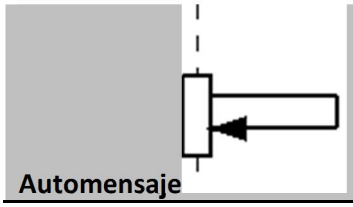
	oscura al final. Si el mensaje es de retorno aparece como una línea discontinua. El remitente espera una respuesta a un mensaje sincrónico antes de continuar.
Asíncrono 	Un mensaje con una línea continua con punta de flecha en línea al final. Si el mensaje es de retorno aparece como una línea discontinua. No requiere una respuesta antes de que el remitente continúe. Un mensaje asincrónico muestra sólo una llamada del remitente.
Perdido 	Un mensaje asincrónico a un receptor desconocido o no especificado.
Encontrado 	Un mensaje asincrónico de un participante desconocido o no especificado.
 **Automensaje**	Un automensaje es usualmente una llamada recursiva o una llama a otro método perteneciente al mismo objeto.

Tabla 1.7: Tipos de mensajes en diagrama de secuencia

1.9 XML[4]

XML, el lenguaje de marcas extensible (por sus siglas en inglés Extensible Markup Language), es un estándar aprobado por la World Wide Web Consortium (W3C) para el marcado de documentos. Define una sintaxis genérica usada para marcar datos con etiquetas simples y legibles. Proporciona un estándar para documentos informáticos que es suficientemente flexible para ser personalizado para campos tan diversos como sitios web, intercambio de datos electrónicos, gráficos vectoriales, llamadas a procedimientos remotos y mucho más.

Un desarrollador puede escribir sus propios programas que interactúen con mensajes y manipulen los datos en documentos XML. Esto da acceso a un amplio rango de librerías en una

[4] Información obtenida de la referencia [8]

variedad de lenguajes que pueden leer y escribir XML de modo que el desarrollador se enfoque únicamente en las necesidades de la aplicación.

XML es un lenguaje de meta marcado para documentos de texto. Los datos son incluidos en documentos XML como cadenas de texto . Estos datos son rodeados por marcas de texto que los describen. La unidad básica de datos y de marcado en XML es llamada *elemento*. La especificación de XML define la sintaxis exacta que ésta marca debe seguir: cuántos elementos son delimitados por etiquetas, como se ve una etiqueta, que nombres son aceptables para elementos, donde son colocados los atributos, etc. Superficialmente, el marcado en un documento XML se parece mucho al marcado en un documento HTML, pero existen algunas diferencias cruciales. Aún más importante, XML es un lenguaje de *meta marcado*. Esto significa que no tiene un conjunto fijo de etiquetas y elementos que se supone que trabajen para todo el mundo en todas la áreas de interés todo el tiempo. Cualquier intento de crear un conjunto finito de tales etiquetas está condenado a fallar. En su lugar, XML permite a los desarrolladores inventar los elementos necesarios a medida que los requieran.

Aunque XML es bastante flexible en los elementos que permite, es muy estricto en muchos otros aspectos. La especificación XML define una gramática para documentos XML que indica donde las etiquetas pueden ser colocadas, como se ven las etiquetas, qué nombres de elementos son válidos, etc. Esta gramática es lo suficientemente específica para permitir al desarrollador de parsers XML el poder leer cualquier documento XML. Los documentos que satisfacen esta gramática se dice que están *bien formados*. Los documentos que no están bien formados son rechazados por procesadores de documentos XML.

Por razones de interoperabilidad, individuos y organizaciones acuerdan utilizar solo ciertas etiquetas. Estas etiquetas son llamadas *aplicaciones XML*. UNa aplicación XML no es una aplicación de software que utiliza XML, es una aplicación de XML en un dominio particular tal como gráficos vectoriales.

El marcado en un documento XML describe la estructura del documento, permite ver cómo se asocian los elementos. En un documento XML bien diseñado el marcado también describe la semántica del documento. Por ejemplo, el marcado puede indicar que un elemento es una fecha, una persona o una imagen. En aplicaciones XML bien diseñadas el marcado no dice acerca de cómo el documento debe ser desplegado, es decir, no dice que un elemento debe ser mostrado en negrita o cursiva. XML es un lenguaje de marcado estructural y semántico, no un lenguaje de presentación.

El marcado permitido en una aplicación XML particular puede ser documentado en un *esquema*. Instancias de un documento particular pueden ser comparadas con el esquema. Los documentos que emparejen con el esquema se dice que son *válidos*, la validez depende del esquema. No todos los documentos deben ser válidos, para muchos propósitos es suficiente que el documento sea bien formado.

Existen muchos lenguajes de esquema XML diferentes con distintos niveles de expresividad. El lenguaje de esquema más soportado y el único definido por la especificación XML es el DTD (*document type definition*). Un DTD lista todos los marcados legales y especifica dónde y cómo pueden ser incluidos en un documento. DTDs son opcionales en XML.

1.9.1 COMO TRABAJA XML

El ejemplo siguiente muestra un documento XML simple. Este documento XML particular podría ser visto en un sistema de control de inventarios o una base de datos de almacén. Marca los datos con etiquetas y atributos que describen el color, el tamaño, número de código de barras, fabricante, nombre del producto, y así sucesivamente.

```
<?xml version="1.0"?>
<product barcode="3141516">
 <manufacturer>Verbatim</manufacturer>
 <name>DataLife MF 2HD</name>
 <quantity>10</quantity>
 <size>3.5"</size>
 <color>black</color>
 <description>floppy disks</description>
</product>
```

El documento puede ser almacenado en un archivo de texto y editarlo con cualquier editor de texto estándar, no es necesario algún editor especial de XML. Los programas que de hecho intentan entender el contenido del documento XML utilizarán un parser XML para leer el documento. El parser es responsable de dividir el documento en elementos individuales, atributos y otros piezas. Pasa el contenido del documento XML a una aplicación pieza por pieza. Si en algún punto el parser detecta una violación de las de reglas de formación correcta de XML, entonces reporta el error a la aplicación y detiene el análisis.

La aplicación que recibe datos desde el parser puede ser:

- Un navegador web que muestra el documento XML a un lector.
- Un procesador de palabras que cargue el documento XML para su edición.
- Una base de datos que almacena los datos XML en un nuevo registro.
- Un programa de dibujo que interpreta XML como coordenadas bidimensionales para contenidos de una imagen.
- Un programa escrito en Java o cualquier otro lenguaje que haga exactamente lo que el desarrollador necesite.

1.10 SERVICIOS WEB[5]

Un servicio web es cualquier servicio que está disponible a través de Internet o redes privadas, utiliza un sistema de mensajería XML estandarizado y no se encuentra atado a algún sistema operativo o lenguaje de programación. Normalmente un servicio web es identificado por una URL como cualquier sitio web. La diferencia entre el sitio y el servicio web radica en el tipo de interacción que pueden proporcionar.

La mayor parte de sitios web están diseñados para proporcionar una respuesta a una petición de una persona. La persona teclea la URL del sitio o hace click en un enlace para crear la petición. Esta petición toma la forma de un documento de texto que contiene algunas instrucciones simples para el servidor. Estas instrucciones están limitadas al nombre de un documento que será devuelto o una llamada a un programa del lado del servidor, junto con unos cuantos parámetros.

Un servicio web es similar en que es accedido mediante una URL. La diferencia reside en el contenido de lo que se está enviando en la petición desde el cliente hacia el servicio. Los clientes del servicio web envían un documento XML formateado de una forma especial de acuerdo con las reglas de la especificación SOAP.

Un mensaje SOAP puede contener una llamada a un método junto con los parámetros que podría necesitar. Además, el mensaje puede contener un número de elementos de encabezado que especifican el propósito del cliente. Estos elementos de encabezado pueden designar qué servicios web obtendrían la llamada al método después de que el servicio actual finalice su trabajo o podrían contener información acerca de la seguridad. En cualquier caso, la complejidad del mensaje SOAP excede por mucho la complejidad que es posible utilizando sólo un navegador.

La arquitectura de los servicios web está basada en enviar mensajes XML en un formato SOAP específico. XML puede ser representado como caracteres ASCII, los cuales pueden ser transferidos fácilmente de una computadora a otra. Las implicaciones de esto son significantes:

- No importa qué tipo de computadora está enviando el mensaje XML o de qué sistema operativo se está ejecutando.
- No importa de qué parte del mundo el cliente está enviando el mensaje.
- No importa en qué lenguaje de programación está escrito el software que el cliente está ejecutando.
- No es necesario para el cliente conocer el tipo de procesador SOAP que está ejecutando el servidor.

[5] Información obtenida de la referencia [9], [10]

Con lo anterior, todas las aplicaciones que utilizan servicios web pueden potencialmente comunicarse entre ellas. Esta comunicación puede tomar lugar a través de viejos límites de ubicación, sistemas operativos lenguajes, protocolos, etc.

1.10.1 COMPONENTES CLAVE

Transacciones de servicios web tienen lugar entre componentes. Tales componentes pueden ser programados desde cero, descargarlos de fundaciones de código abierto o comprarlos de vendedores comerciales. No existen requerimientos de utilizar todos los componentes de un único fabricante. Los siguientes son considerados núcleo de servicios web.

1.10.1.1 SOAP

SOAP viene de Simple Object Access Protocol. Pero SOAP se considera ahora un nombre de especificación y no un acrónimo. SOAP es una especificación que define una gramática XML para enviar mensajes y responder mensajes recibidos de otras partes. La meta de SOAP es describir un formato de mensaje que no está ligado a ninguna arquitectura de hardware o de software, pero que lleva un mensaje desde cualquier plataforma a cualquier otra plataforma de una manera inequívoca.

El estándar SOAP contiene dos partes: los encabezados que llevan instrucciones de procesamiento y el cuerpo que contiene la carga útil. La carga útil contiene la información a enviar. Los dos tipos de mensajes SOAP son documentos y llamadas de procedimientos remotos (*RPC, Remote Procedure Call*). La carga útil de un mensaje de tipo documento es cualquier documento XML que necesite ser enviado de una computadora a otra. Una RPC es una llamada a un método que está destinado a ser ejecutado en el equipo del servicio web. El mensaje RPC realiza la misma función que una llamada de método ordinario en un lenguaje de programación ordinario. La diferencia es que esta llamada puede tener lugar a través de Internet.

1.10.1.2 XML

XML es el lenguaje sobre el que están construidos todos los lenguajes de servicios web. XML es una herramienta para escribir documentos auto descriptibles. De hecho XML es más un metalenguaje que un lenguaje y es usado para crear gramáticas. Estas gramáticas están descritas en esquemas XML que especifican las etiquetas que son permitidas y las relaciones entre los elementos definidos por esas etiquetas. SOAP y WSDL son gramáticas basadas en XML.

1.10.1.3 HTTP

El protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol*) es un estándar que precede a la llegada de los servicios web. Fue desarrollado para facilitar la transferencia de peticiones de un navegador al servidor web. Los servicios web toman ventaja de la existencia de este protocolo maduro para mover los mensajes SOAP y documentos WSDL de un ordenador a otro.

Nuevas versiones de SOAP describen como otros mecanismos de transporte tales como FTP, SMTP y JMS pueden ser utilizados para realizar la misma función. Aunque la mayor parte de los servicios web está construida sobre HTTP.

1.10.1.4 WSDL

El lenguaje de descripción de servicios web (*Web Services Description Language*) es una especificación que dicta cómo describir una pieza de software en términos de las llamadas de métodos a las que responde. Estos métodos son descritos en una forma abstracta que es independiente del lenguaje de programación en el que está escrito el servicio o de la computadora y sistema operativo sobre el que se están ejecutando tales métodos.

WSDL es una gramática basada en XML para especificar una interfaz pública para un servicio web, Esta interfaz pública puede incluir información de todos los métodos disponibles al público, información de tipos de datos para todos los mensajes XML, información acerca del protocolo de transporte específico que será utilizado e información de la dirección para localizar el servicio especificado,

CAPÍTULO 2. ANÁLISIS Y DISEÑO

Durante el presenta capítulo se habla sobre el análisis y diseño del sistema. Estos elementos son utilizados por los programadores para construir el sistema bajo una metodología orientada a objetos. Dicha metodología se compone de diferentes elementos que permiten conocer a detalle cada uno de los componentes que intervienen durante la implementación, pruebas y mantenimiento del producto final.

2.1 ANÁLISIS

2.1.1 DESCRIPCIÓN DEL PROBLEMA

Se desea contar con un sistema multiplataforma para la creación de proyectos de composición de imágenes digitales para usuarios no expertos en el ámbito del diseño gráfico. Para tal fin, el sistema contará con un catálogo de imágenes predeterminadas y el usuario podrá seleccionar imágenes de su ordenador. Como resultado de cada proyecto se obtendrá una imagen lista para su impresión digital.

El sistema a desarrollar contará con cinco categorías de imágenes:

1. Fondos
2. Cliparts
3. Marcos
4. Máscaras
5. Imágenes de usuario

Las imágenes de fondos, marcos, máscaras y cliparts serán proporcionadas por la empresa masimpresiones.com y se almacenarán en un equipo servidor. Estas imágenes tendrán el formato PNG y contarán con una resolución adecuada para realizar proyectos de tamaño 32x48cm.

El usuario creará proyectos con ayuda de la aplicación y éstos únicamente serán una vista previa de la imagen resultante, todas las imágenes en la interfaz de usuario se manejarán a escala dependiendo de la resolución de pantalla del usuario.

Las funciones que llevará a cabo el sistema son:

- Crear proyectos. El sistema permitirá la creación de proyectos desde cero, es decir, con el lienzo vacío. El tipo de proyecto por defecto será de 32x48 cm. El tamaño del lienzo y de las imágenes se ajustarán a la resolución de la pantalla. Las imágenes mostradas en el lienzo serán de menor calidad que las originales, debido a que estas últimas son de muy buena resolución.

- Guardar proyectos. El sistema contará con la opción de guardar su proyecto de forma local como archivo de proyecto XML. También tendrá la opción de guardar proyectos de manera remota, es decir, en el servidor de la aplicación. Para la última opción será posible guardar el proyecto en formato XML y las imágenes del usuario utilizadas en dicho proyecto deberán ser subidas al servidor, evidentemente este proceso necesitará de una conexión a internet y de una cuenta de usuario en el servidor.

- Editar proyectos. El sistema permitirá editar proyectos guardados de manera local, para esto el sistema tomará como guía el proyecto en formato XML.

- Crear cuenta de usuario en el servidor de la aplicación.

- Seleccionar imágenes de marcos, máscaras, fondos y cliparts desde la interfaz de usuario para la creación del proyecto del usuario.

- Seleccionar imágenes de usuario de manera local para la creación de proyectos.

- Mostrar imágenes de la aplicación en la interfaz. El sistema deberá mostrar las imágenes de marcos, máscaras, fondos y cliparts que se encuentran disponibles de manera local. Todas las imágenes deberán estar agrupadas por tipo.

- Cambio de ubicación de imágenes sobre el lienzo.

- Cambio de tamaño de imágenes sobre el lienzo.

- Aplicar máscaras a imágenes de usuario. La aplicación permitirá al usuario seleccionar alguna imagen de máscara disponible y aplicarla a una imagen de usuario sobre el lienzo. Si no existen imágenes de usuario sobre el lienzo, las máscaras no podrán ser seleccionadas desde la interfaz de usuario.

- Operaciones de *drag & drop* en imágenes mostradas en la interfaz. El sistema deberá permitir seleccionar imágenes desde la sección de imágenes disponibles y arrastrarlas hacia el lienzo para ser agregadas. Todos los tipos de imágenes deberán contar con esta característica. Además, se debe tomar en cuenta que una imagen de fondo siempre queda hasta atrás en el lienzo e imágenes de cliparts quedan hasta adelante del mismo.

- Aplicar operaciones puntuales (negativo, paso a grises, corrección gama, etc.) únicamente a imágenes de usuario.

- Obtener imagen final con formato PNG del lado del servidor.

2.1.2 MODELO DE CASOS DE USO

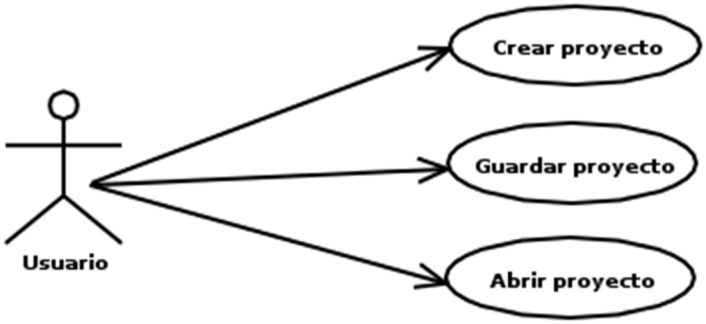


Figura 2.1: Diagrama de casos de uso para la gestión de proyectos

Nombre	**Crear proyecto**
Actor	Usuario
Descripción	Permite crear un proyecto de composición de imágenes de las dimensiones especificadas por el usuario.
Precondiciones	
Postcondiciones	Se muestra el lienzo manteniendo las proporciones del proyecto creado.
Flujo normal	1. El usuario dará clic en el menú Archivo y en la opción Nuevo 2. El sistema mostrará una caja de diálogo 3. El usuario indicará el nombre, dimensiones y resolución del proyecto 4. El usuario dará clic en el botón Aceptar 5. El sistema mostrará el lienzo para el proyecto que se acaba de crear
Flujos alternativos	3. El usuario cierra la caja de diálogo, entonces no se crea un nuevo proyecto y en caso de tener un proyecto abierto, éste no sufre ningún cambio. 3. El usuario indica valores no numéricos para los campos de altura y/o ancho, entonces el sistema deberá mostrar un mensaje que comunique el error para que el usuario lo solucione. 4. El usuario da clic en el botón cancelar, entonces el sistema ignora la petición de crear un nuevo proyecto.

Tabla 2.1: Descripción de caso de uso *Crear proyecto*

Nombre	**Guardar proyecto**
Actor	Usuario

Descripción	Permite al usuario guardar el proyecto actual en formato XML para su posterior edición
Precondiciones	Contar con un proyecto abierto
Postcondiciones	Se almacena archivo XML en la ubicación especificada por el usuario
Flujo normal	1. El usuario dará clic en el menú Archivo y en la opción Guardar proyecto 2. El sistema mostrará una caja de diálogo para elegir la ruta y el nombre con el que será guardado el proyecto 3. El usuario indicará el nombre y la ruta 4. El sistema creará un archivo con extensión xml y lo almacenará en la ruta especificada.
Flujos alternativos	3. El usuario cierra la caja de diálogo, entonces no se guardará el proyecto y no habrá ningún cambio sobre el proyecto actual.

Tabla 2.2: Descripción de caso de uso *Guardar proyecto*

Nombre	**Abrir proyecto**
Actor	Usuario
Descripción	Permite al usuario abrir y editar un proyecto en formato XML creado por el sistema
Precondiciones	Contar con un proyecto en formato XML creado por el sistema
Postcondiciones	El sistema muestra el proyecto en el mismo estado en el que se encontraba al momento de ser guardado
Flujo normal	1. El usuario dará clic en el menú Archivo y en la opción Abrir proyecto 2. El sistema mostrará una caja de diálogo para elegir el archivo en formato XML 3. El usuario indicará el archivo 4. El sistema limpiará el lienzo y mostrará el proyecto en el estado en el que se encontraba al momento de ser guardado
Flujos alternativos	3. El usuario cierra la caja de diálogo, entonces se ignorará la petición de abrir un proyecto

Tabla 2.3: Descripción de caso de uso *Abrir proyecto*

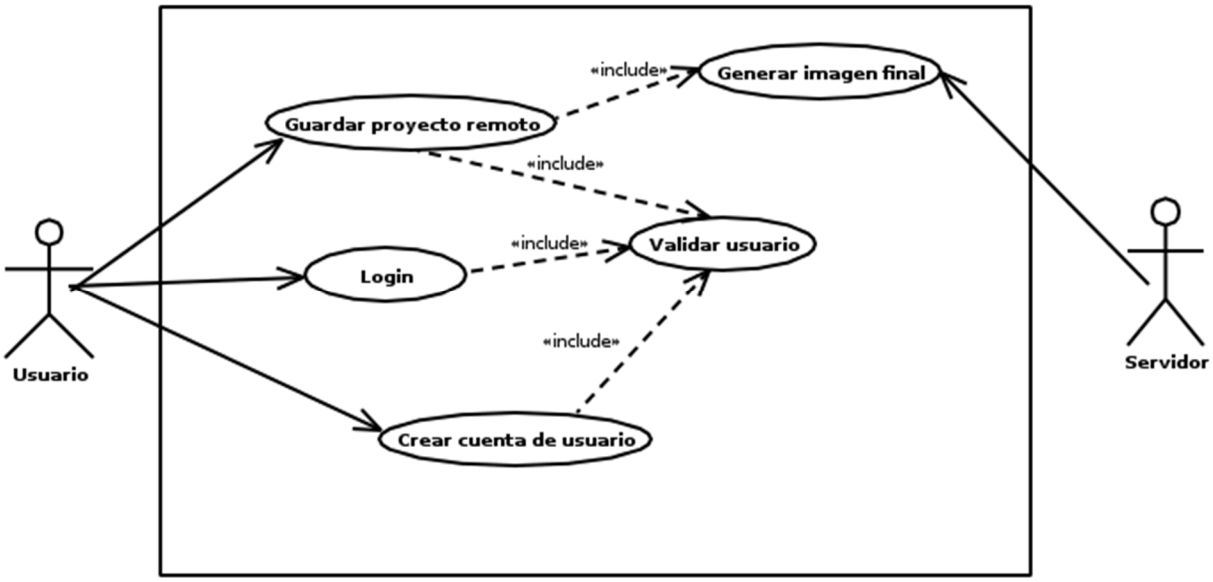


Figura 2.2: Diagrama de casos de uso para funcionalidades que interactúan con el Servidor

Nombre	**Guardar proyecto remoto**
Actor	Usuario
Descripción	Permite al usuario guardar el proyecto en formato XML en el servidor y generar la imagen final a partir del archivo XML almacenado
Precondiciones	El usuario debe estar logueado
Postcondiciones	El proyecto y la imagen resultante quedarán almacenados en el servidor
Flujo normal	1. El usuario dará clic en el menú Archivo y Guardar en el servidor 2. El sistema creará el proyecto en formato XML y lo almacenará en el disco local 3. El sistema creará un proceso que ejecutará en segundo plano para enviar el proyecto al servidor 4. El servidor recibirá el proyecto y lo almacenará en su sistema de archivos 5. El servidor generará la imagen resultante del proyecto almacenado
Flujos alternativos	1. El usuario no se encuentra logueado, entonces el sistema ignorará la petición de Guardar en el servidor

Tabla 2.4: Descripción de caso de uso *Guardar proyecto remoto*

Nombre	**Login**
Actor	Usuario
Descripción	Permite al usuario iniciar sesión
Precondiciones	
Postcondiciones	El usuario habrá iniciado sesión
Flujo normal	1. El usuario dará clic en el menú Archivo y en Iniciar sesión 2. El sistema mostrará una pantalla para proporcionar el nombre y contraseña del usuario 3. El usuario proporcionará los datos 4. El sistema validará los datos 5. El sistema mostrará mensaje de inicio de sesión exitoso

Flujos alternativos	3. El usuario cierra la ventana, entonces el sistema ignorará la petición de iniciar sesión 4. El usuario proporciona nombre o contraseña inválidos, entonces el sistema informará acerca del problema y el usuario podrá corregirlo

Tabla 2.5: Descripción de caso de uso *Login*

Nombre	**Crear cuenta de usuario**
Actor	Usuario
Descripción	Permite al usuario crear una cuenta en la aplicación
Precondiciones	
Postcondiciones	
Flujo normal	1. El usuario dará clic en el menú Archivo y Login 2. El usuario dará clic en el botón Crear cuenta 3. El sistema mostrará una pantalla para proporcionar nombre de usuario y contraseña 4. El usuario proporcionará los datos 5. El usuario dará clic en el botón Aceptar 6. El sistema validará los datos 7. El sistema mostrará mensaje de creación de cuenta exitosa
Flujos alternativos	5. El usuario cierra la ventana de registro, entonces se igaorá la petición de crear cuenta 6. El nombre de usuario ya existe, entonces el sistema informa al usuario y permite cambiar los datos de registro

Tabla 2.6: Descripción de caso de uso *Crear cuenta de usuario*

El caso de uso *Validar usuario* únicamente realiza una consulta sobre la tabla usuarios de la base de datos del servidor, se verifica el usuario y la contraseña y se informará sobre usuario y/o contraseña inválidos.

El caso de uso *Generar imagen final* es disparado por el usuario al guardar un proyecto en el servidor de la aplicación. El servidor es el encargado de crear el proceso que generará la imagen final que quedará almacenada en el sistema de ficheros del propio servidor. Para lograrlo será necesario contar con el proyecto en formato XML en el sistema de ficheros del servidor.

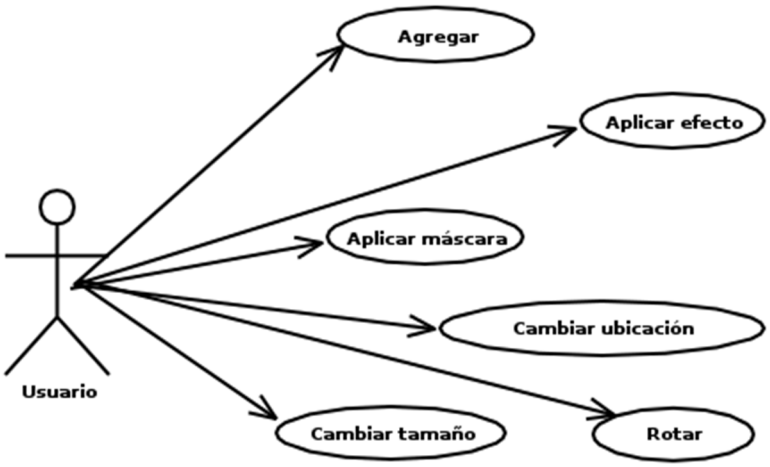


Figura 2.3: Diagrama de casos de uso para la composición de imágenes

Nombre	**Agregar**
Actor	Usuario
Descripción	Permite al usuario agregar imágenes al lienzo
Precondiciones	El usuario debe contar con un proyecto abierto
Postcondiciones	La imagen será agregada al lienzo y se le podrán aplicar diversos filtros
Flujo normal	1. El usuario seleccionará una imagen del catálogo o una imagen propia 2. El usuario arrastrará la imagen y la soltará en un punto sobre el lienzo 3. La imagen será dibujada sobre el lienzo
Flujos alternativos	3. Si la imagen es un fondo y ya existe uno, entonces el fondo será reemplazado 3. Si la imagen es una máscara, entonces no podrá ser agregada sobre el lienzo, las máscaras se aplicarán a imágenes de usuario 3. Si la imagen es de tipo clipart, marco o de usuario, entonces la imagen será agregada encima de todas las imágenes del proyecto

Tabla 2.7: Descripción de caso de uso *Agregar*

Nombre	**Aplicar efecto**
Actor	Usuario
Descripción	Permite al usuario aplicar filtros puntuales a la imagen
Precondiciones	El usuario debe contar con un proyecto abierto que contenga alguna imagen de tipo clipart, marco o de usuario
Postcondiciones	El filtro será aplicado sobre la imagen seleccionada
Flujo normal	1. El usuario seleccionará una imagen dando clic sobre ella 2. El usuario dará clic en el menú Imagen 3. El usuario seleccionará una opción del submenú Efectos 4. El efecto será aplicado sobre la imagen seleccionada
Flujos alternativos	1. La imagen deseada se encuentra por debajo de otra, entonces el usuario enviará la imagen atrás a través de un menú contextual

Tabla 2.8: Descripción de caso de uso *Aplicar efecto*

Nombre	**Aplicar máscara**
Actor	Usuario
Descripción	Permite al usuario aplicar una máscara sobre una imagen de usuario
Precondiciones	El proyecto debe contar con al menos una imagen de usuario
Postcondiciones	La máscara es aplicada sobre la imagen de usuario seleccionada
Flujo normal	1. El usuario seleccionará una máscara del catálogo 2. El usuario arrastrara la máscara hacia la imagen de usuario 3. El usuario soltará la máscara sobre la imagen de usuario 4. La máscara será aplicada sobre la imagen
Flujos alternativos	3. Si la imagen sobre la que se intenta aplicar la máscara no es de usuario, entonces no se aplica la máscara 4. Si la imagen sobre la que se intenta aplicar la máscara ya se le ha aplicado una, entonces la máscara se aplica sobre la imagen original.

Tabla 2.9: Descripción de caso de uso *Aplicar máscara*

Nombre	Cambiar ubicación
Actor	Usuario
Descripción	Permite al usuario cambiar la posición de una imagen en el lienzo
Precondiciones	Contar con un proyecto abierto con al menos una imagen que no sea de fondo
Postcondiciones	La imagen será trasladada a la nueva posición
Flujo normal	1. El usuario seleccionará la imagen a mover 2. El usuario arrastrará la imagen hacia la nueva posición 3. El usuario soltará la imagen en la nueva ubicación
Flujos alternativos	1. La imagen deseada se encuentra por debajo de otra, entonces el usuario enviará la imagen atrás a través de un menú contextual

Tabla 2.10: Descripción de caso de uso *Cambiar ubicación*

Nombre	Rotar
Actor	Usuario
Descripción	Permite al usuario rotar la imagen seleccionada
Precondiciones	Contar con un proyecto abierto con al menos una imagen que no sea de fondo
Postcondiciones	La imagen será rotada de acuerdo a su centro
Flujo normal	1. El usuario seleccionará una imagen sobre el lienzo haciendo clic sobre ella 2. El usuario seleccionará la opción de rotación a través del menú Imagen 3. El sistema mostrará una interfaz con un control para aplicar la rotación sobre la imagen 4. El usuario seleccionará los grados a rotar 5. La aplicación rotará la imagen al momento de cambiar los grados con el control
Flujos alternativos	1. La imagen deseada se encuentra por debajo de otra, entonces el usuario enviará la imagen atrás a través de un menú contextual

Tabla 2.11: Descripción de caso de uso *Rotar*

Nombre	Cambiar tamaño
Actor	Usuario
Descripción	Permite al usuario cambiar una imagen de tamaño
Precondiciones	Contar con un proyecto abierto con al menos una imagen que no sea de fondo
Postcondiciones	La imagen será escalada
Flujo normal	1. El usuario colocará el puntero del mouse la imagen 2. La aplicación mostrará un borde de la imagen con cuadros en las esquinas y los lados para controlar el cambio de tamaño 3. El usuario dará clic en unos de los puntos de control 4. El usuario arrastrará el mouse para cambiar el tamaño de la imagen 5. La imagen será escala al momento de arrastrar el mouse desde un punto de control

Flujos alternativos	1. La imagen deseada se encuentra por debajo de otra, entonces el usuario enviará la imagen atrás a través de un menú contextual

Tabla 2.12: Descripción de caso de uso *Cambiar tamaño*

2.1.3 IDENTIFICACIÓN DE CLASES

A continuación se muestra la identificación parcial de clases a partir del modelo de casos de uso.

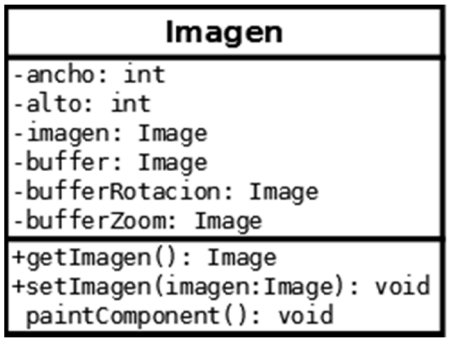


Figura 2.4: Clase *Imagen*

La clase Imagen es la responsable de dibujar imágenes sobre el lienzo. Los buffers son necesarios en caso de que a una imagen se le apliquen múltiples efectos.

Figura 2.5: Clase *FiltrosPuntuales*

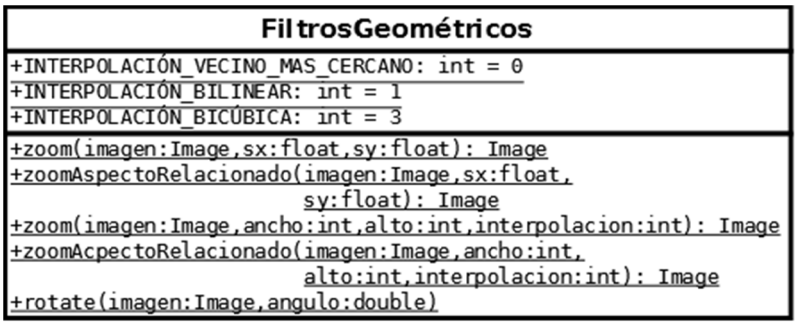


Figura 2.6: Clase *FiltroGeométricos*

Los filtros que se pueden aplicar sobre las imágenes están divididos en Puntuales y Geométricos. Las clases responsables son FiltrosPuntuales y FiltrosGeométricos respectivamente. Cada método recibe como parámetro la imagen a la que se debe aplicar el filtro y devuelve una nueva imagen con el filtro aplicado. En el caso de la operación de escalado es necesario indicar el tipo de interpolación, es por eso que se declaran constantes con para los valores posibles. Las constantes de interpolación están ordenadas de la que produce una imagen con menor calidad de escaldado a mayor calidad de escalado, de esta forma es posible utilizar interpolación con vecino más cercano en el cliente e interpolación bicúbica en el servidor ya que en el cliente las imágenes deben ser escaladas de acuerdo a la resolución del usuario.

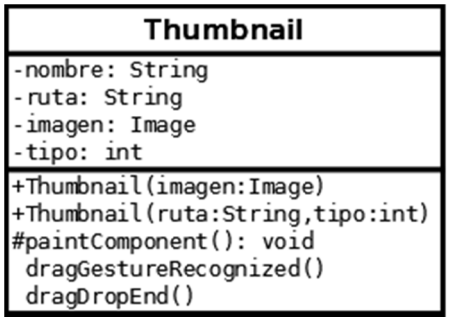


Figura 2.7: Clase *Thumbnail*

La clase Thumbnail representará las imágenes miniatura en la aplicación, esto es, las imágenes del catálogo y las imágenes de usuario. Además las miniaturas deberán contar con identificadores que permitan conocer a qué imagen representan, de qué tipo es y su ruta. Los métodos dragGestureRecognized() y dragDropEnd() son implementados para dar al componente la capacidad de ser arrastrado hacia el lienzo.

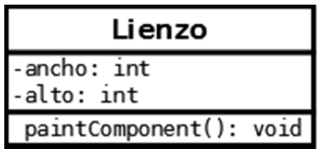


Figura 2.8: Clase *Lienzo*

El Lienzo es el componente sobre el que se realiza el proyecto, por lo tanto debe ser una clase que sepa como dibujar componentes sobre él, esto se logra gracias al método paintComponent.

Figura 2.9: Clase *Proyecto*

Cuando se crea o edita un proyecto es necesario mantener la información básica en memoria, la clase Proyecto encapsula dicha información y la hace accesible a través de métodos getters.

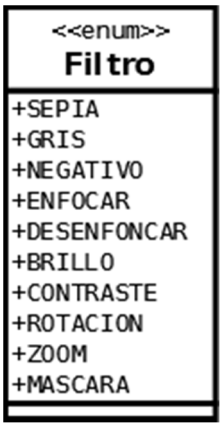


Figura 2.10: Clase *Filtro*

Es útil contar con una serie de constantes con el fin de identificar los filtros aplicados en el proyecto. De esta forma es más sencillo que el sistema guarde y cargue proyectos, además el código se vuelve más legible para su mantenimiento. Esto se logra gracias a las enumeraciones Java.

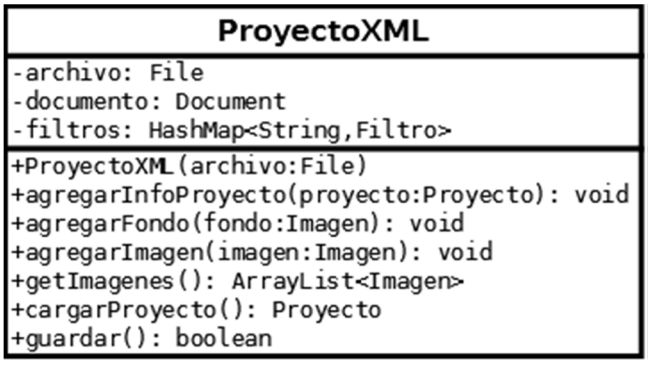


Figura 2.11: Clase *ProyectoXML*

La clase ProyectoXML es la encargada de construir el documento XML y de almacenarlo en disco.

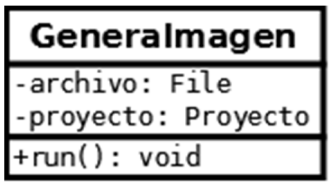


Figura 2.12: Clase *GeneraImagen*

Es necesario contar con una clase que se ocupe únicamente de la generación de la imagen final ya que dicha imagen solo será generada del lado del servidor. Para generar la imagen es necesario contar con el archivo XML del proyecto.

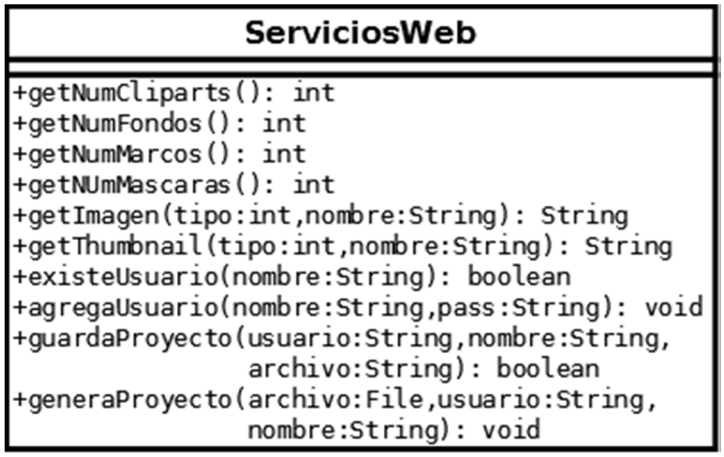


Figura 2.13: Clase *ServiciosWeb*

Toda interacción del cliente con el servidor será llevada a cado mediante una serie de servicios a través de un proyecto web java que los haga disponibles. Una sola clase será la responsable de implementar tales servicios, la clase ServiciosWeb.

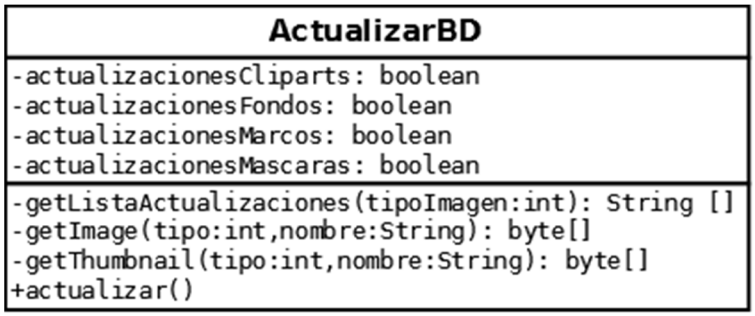


Figura 2.13: Clase *ActualizarBD*

La clase ActualizarBD se comunica con el servidor para decidir si existen actualizaciones en el catálogo de imágenes de la aplicación y actualizar en caso de que éstas existan.

2.2 DISEÑO

2.2.1 MODELO RELACIONAL

El diseño físico de la base de datos produjo un modelo lógico basado en registros. En este caso se utilizó el modelo relacional como modelo final.

Se presentan a continuación las tablas correspondientes a las bases de datos del cliente y del servidor.

Nombre de la columna	**id**	**nombre**	**nickname**	**password**
Tipo de llave	PK			
Nulo	NN	NN	NN	NN
Tipo de dato	Integer	varchar	varchar	blob

Tabla 2.13: Tabla *usuarios*

La base de datos del lado del servidor contará con una tabla para los usuarios, esta base de datos nunca será accesible directamente al cliente, cualquier operación deberá ser realizada a través de servicios.

Nombre de la columna	**id**	**nombre**	**imagen**	**thumbnail**	**tipo**
Tipo de llave	PK				
Nulo	NN	NN	NN	NN	NN
Tipo de dato	Integer	varchar	blob	blob	Integer

Tabla 2.14: Tabla *imagenes*

La aplicación contará con una base de datos local para cada cliente con el catálogo de imágenes disponibles. Las imágenes almacenadas en la máquina del cliente deberán contar con una calidad menor y además serán almacenadas sus imágenes miniaturas en un tamaño de 100x100 pixeles. Esto último para ahorrar operaciones de escalado en cada inicio de la aplicación o en caso de actualización de las imágenes del catálogo.

Las imágenes originales del catálogo no serán almacenadas en una base de datos en el servidor, en su lugar serán guardadas en el sistema de ficheros del servidor.

2.2.2 DIAGRAMA GENERAL DE CLASES

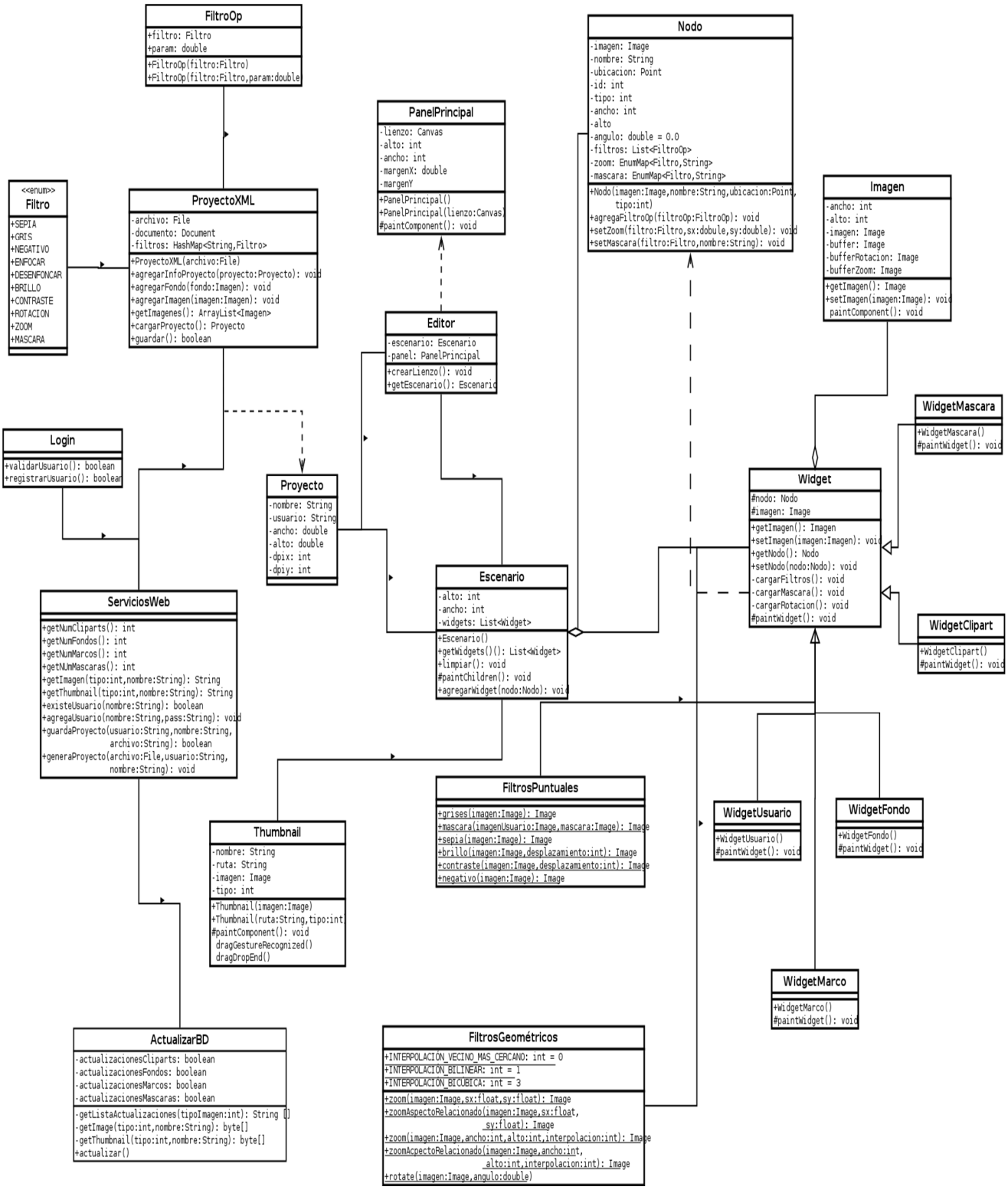


Figura 2.14: Diagrama general de clases

Se muestra el diagrama general de clases de la aplicación. El diagrama contiene el modelo del dominio del sistema. Muestra las clases y la forma en la cual se relaciona una clase con otras.

2.2.3 DIAGRAMAS DE SECUENCIA

Para describir la forma en que los objetos interactúan dentro del sistema, se desarrollaron diagramas de secuencia que se presentan a continuación.

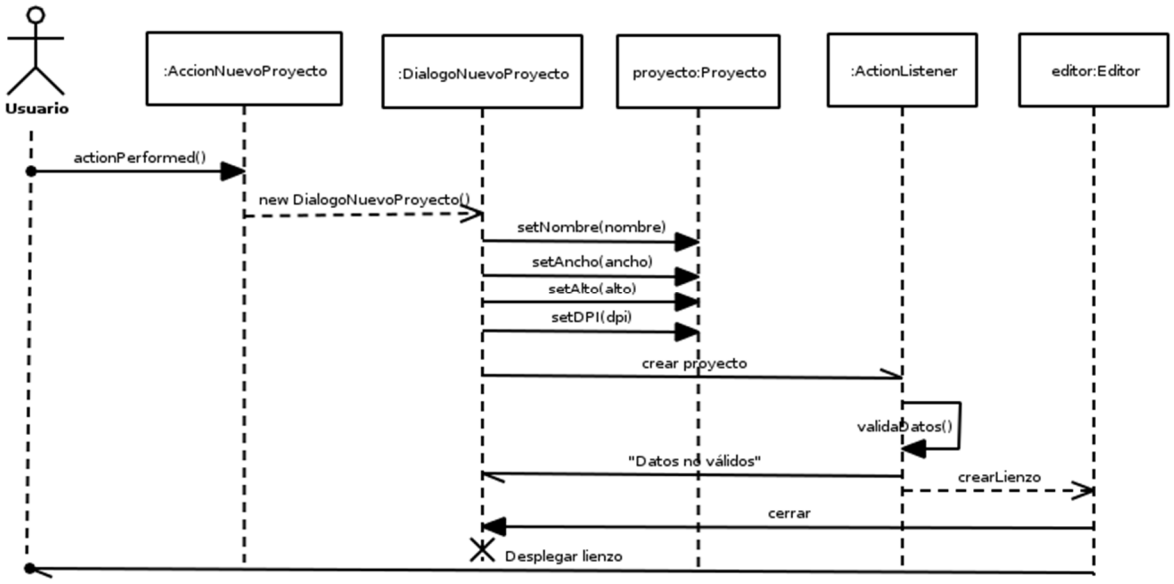


Figura 2.15: Diagrama de secuencia para *Crear proyecto*

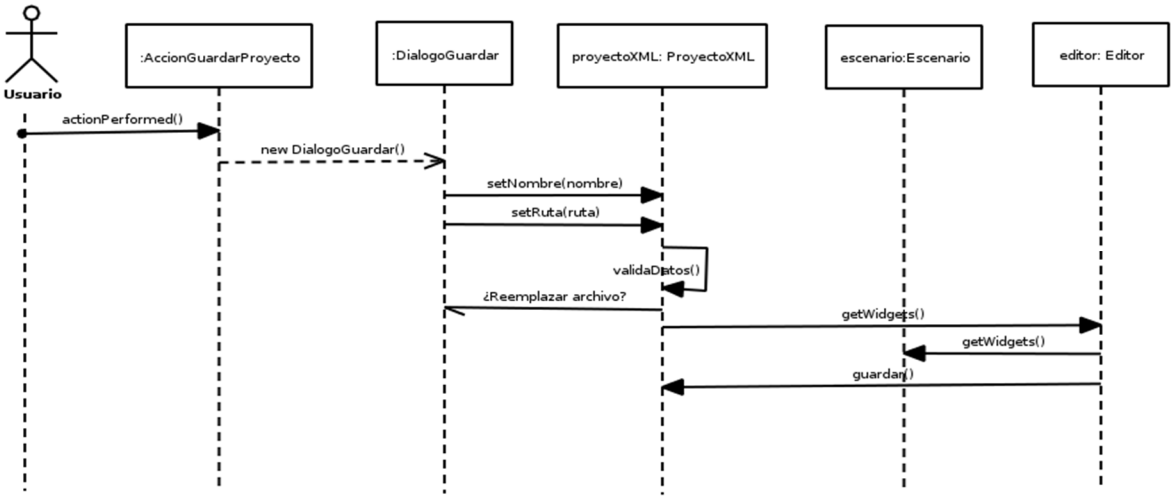


Figura 2.16: Diagrama de secuencia para *Guardar proyecto*

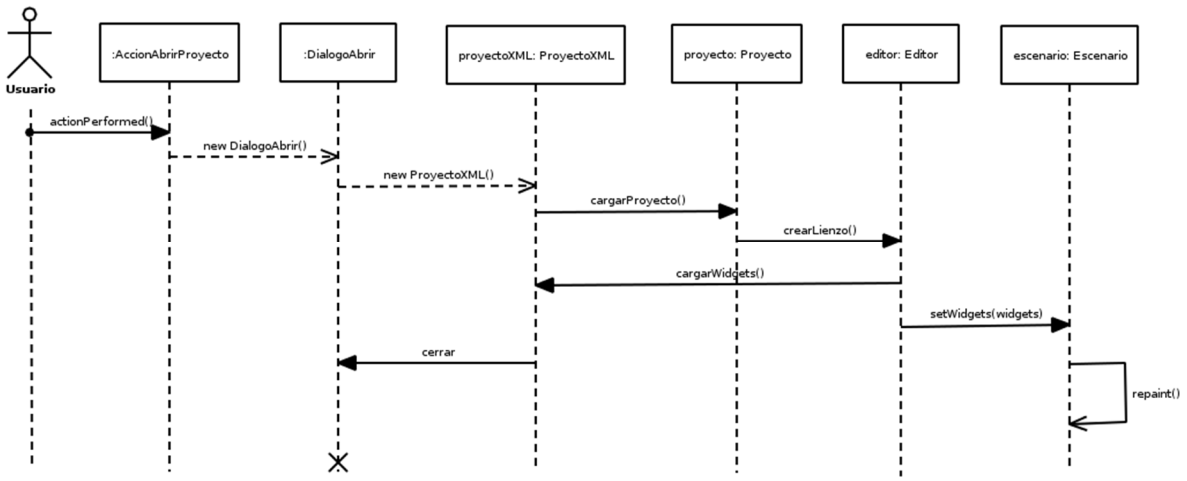


Figura 2.17: Diagrama de secuencia para *Abrir proyecto*

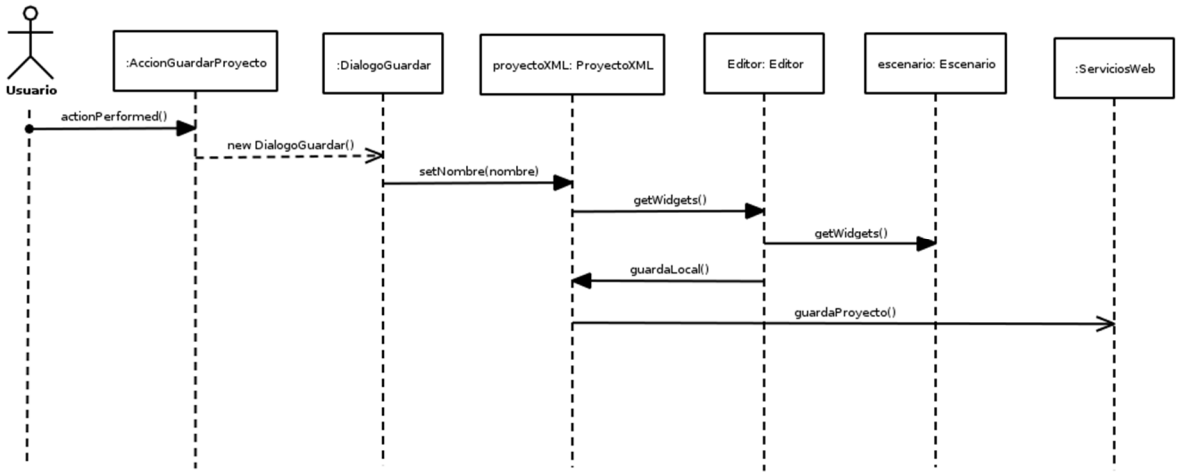


Figura 2.18: Diagrama de secuencia para *Guardar proyecto remoto*

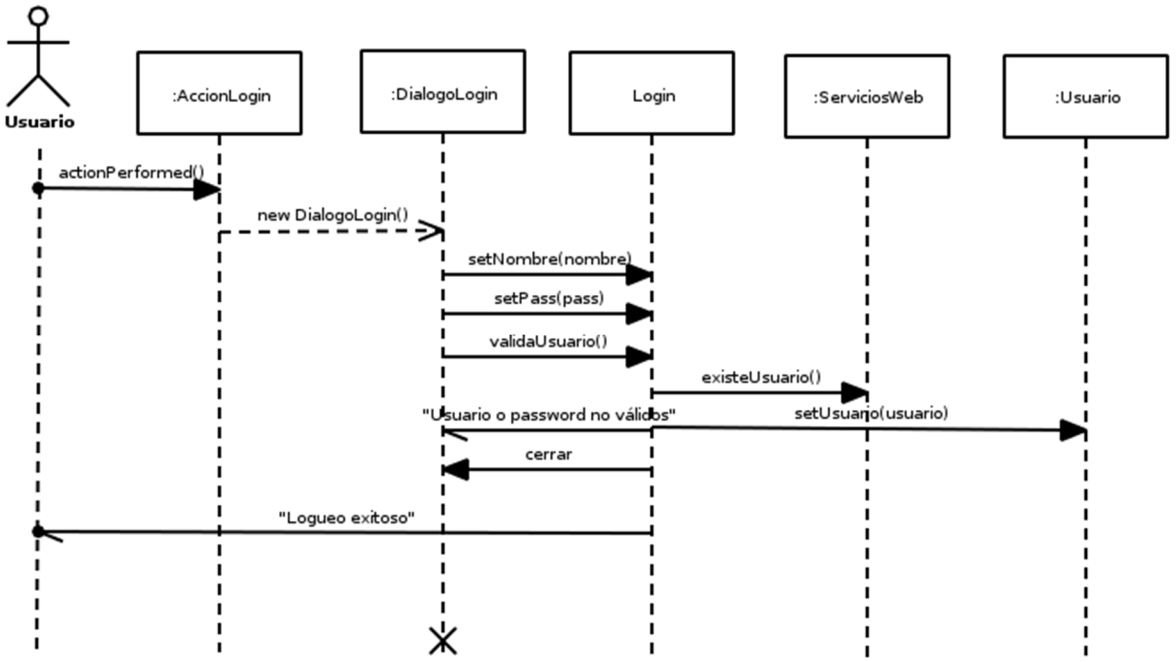


Figura 2.19: Diagrama de secuencia para *Iniciar sesión*

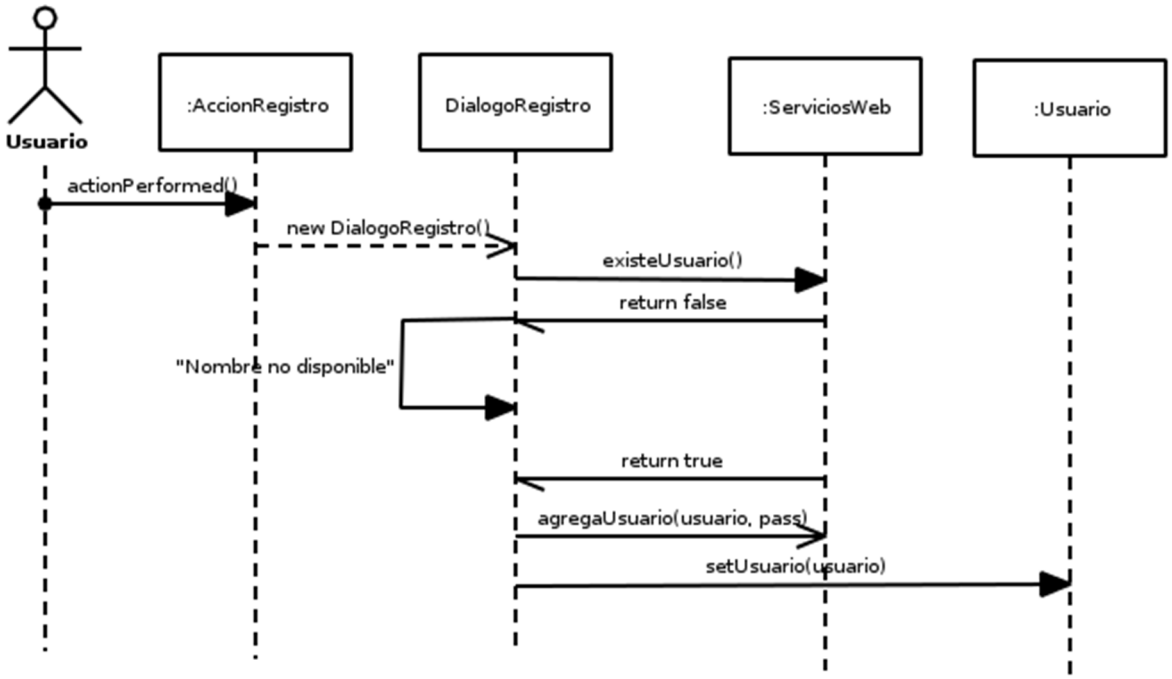


Figura 2.20: Diagrama de secuencia para *Crear cuenta de usuario*

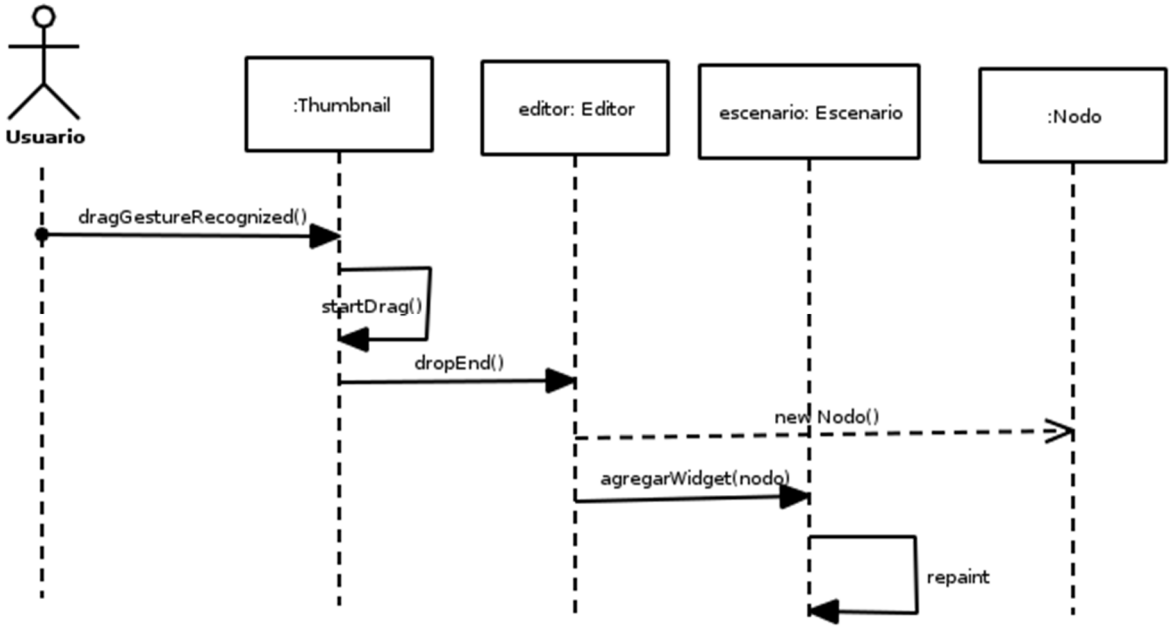


Figura 2.21: Diagrama de secuencia para *Agregar imagen*

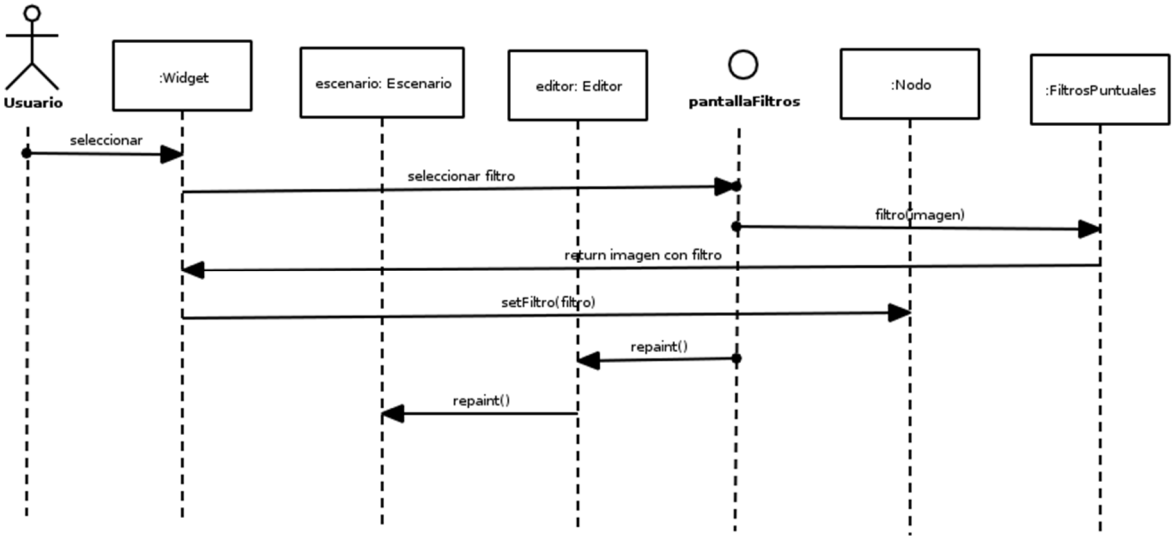


Figura 2.22: Diagrama de secuencia para *Aplicar filtro*

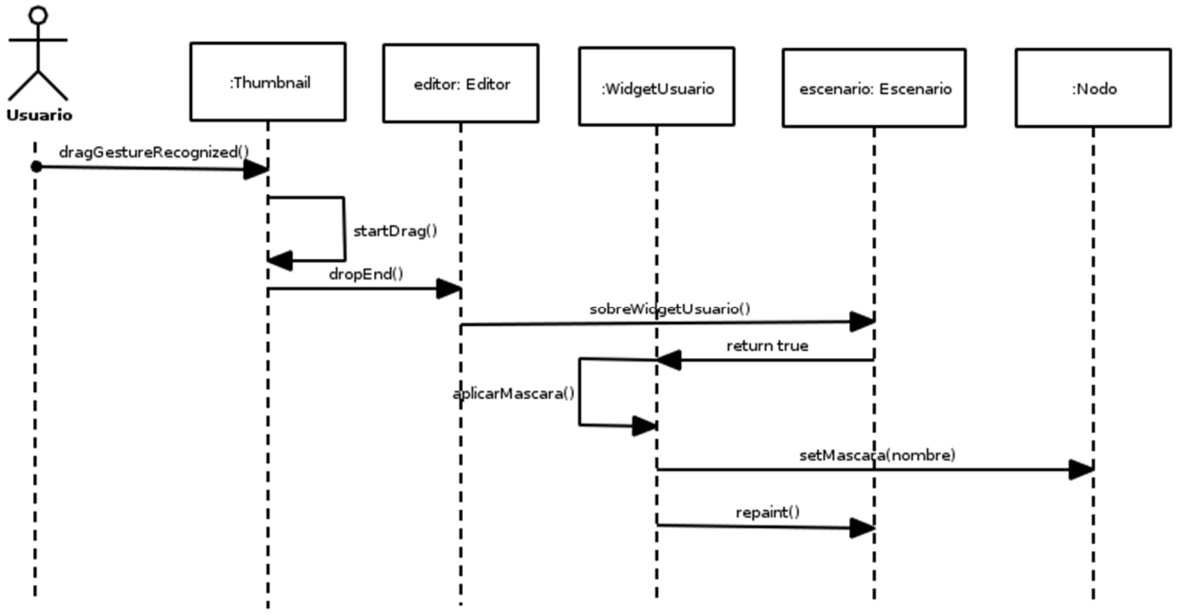


Figura 2.23: Diagrama de secuencia para *Aplicar máscara*

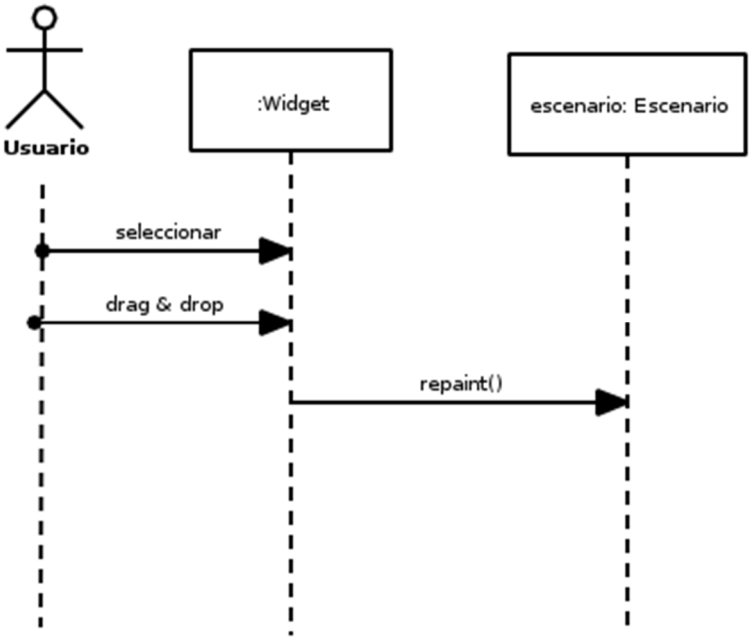


Figura 2.24: Diagrama de secuencia para *Cambiar ubicación*

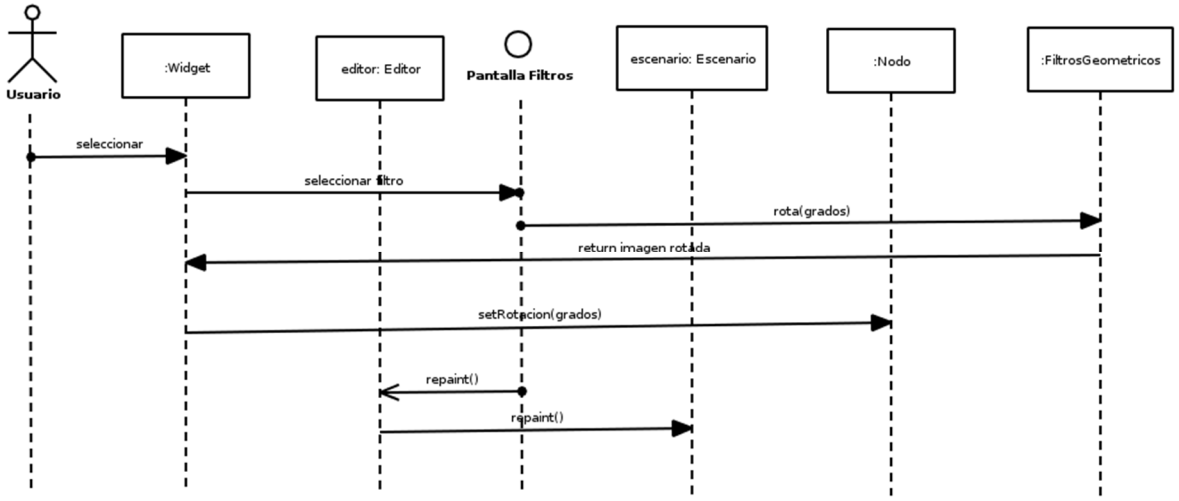


Figura 2.25: Diagrama de secuencia para *Rotación*

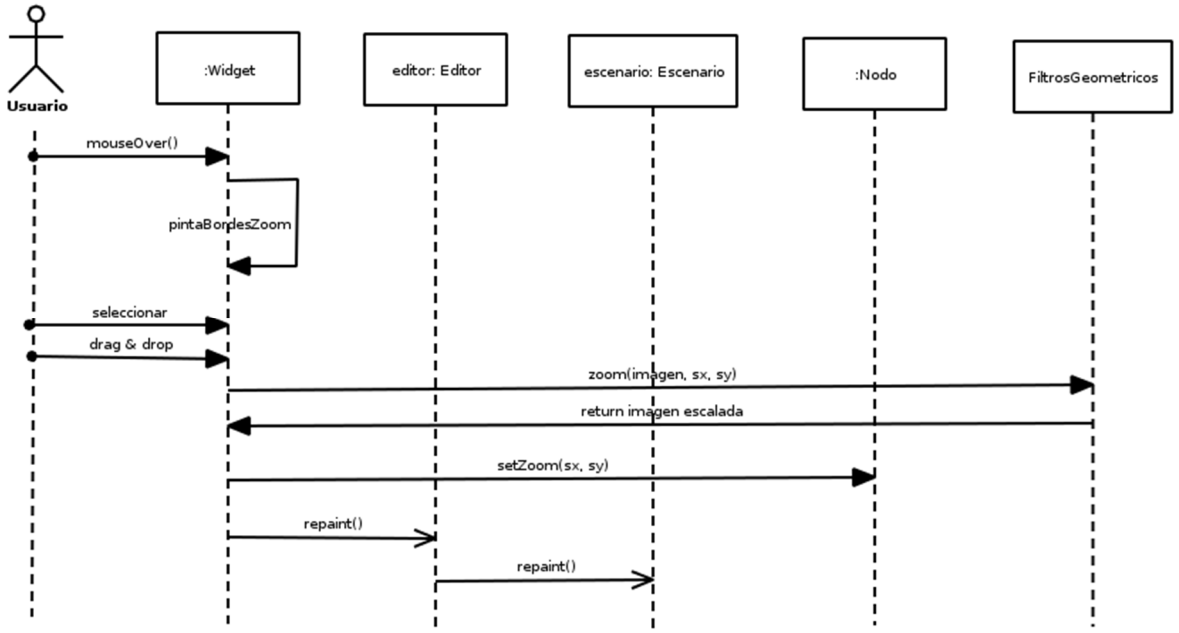


Figura 2.25: Diagrama de secuencia para *Cambio de tamaño*

CAPÍTULO 3. IMPLEMENTACIÓN

3.1 INSTALACIÓN Y PUESTA A PUNTO DE RECURSOS

El sistema objeto de éste documento de tesis se ha trabajado y modelado como un sistema de composición de imágenes digitales, que se refiere al conjunto de operaciones capaces de manipular imágenes digitales para crear una imagen de tal manera que pueda ser impresa en diferentes resoluciones. Cuando se implementa un sistema que manipula imágenes, se deben adoptar tecnologías que permitan almacenar y manipular tales imágenes a partir de una arquitectura debidamente definida por el desarrollador, para tal efecto el sistema se construyó sobre una plataforma de desarrollo basada en las siguientes herramientas:

- Debian GNU/Linux como Sistema Operativo del equipo servidor.
- Servidor web apache para el acceso al sistema de control de versiones.
- MySQL como sistema manejador de bases de datos.
- Servidor de aplicaciones Glassfish para el despliegue de los servicios web.
- Lenguaje de programación orientado a objetos y multiplataforma Java.
- Plataforma NetBeans para el desarrollo modular de la aplicación cliente.
- Subversion como sistema de control de versiones.

3.1.1 INSTALACIÓN DEL SERVIDOR

La computadora que se utilizó cuenta con el sistema operativo Debian GNU/Linux que incluye el servidor web Apache, MySQL y Subversion en sus repositorios de software. Para llevar a cabo su instalación se deben ejecutar los siguientes comandos desde una Terminal en el servidor:

3.1.1.1 APACHE

```
$ sudo apt-get install apache2
```

3.1.1.2 MYSQL

```
$ sudo apt-get install mysql-server
```

Durante la instalación se deberá proporcionar la contraseña para el usuario root de MySQL, no necesariamente la misma contraseña de superusuario.

Por defecto al instalar MySQL se configura como un servicio que se inicia con el sistema operativo. En caso de ser necesario, es posible iniciar, detener o reiniciar el servicio de MySQL con los siguientes comandos:

```
$ sudo /etc/init.d/mysql start
$ sudo /etc/init.d/mysql stop
$ sudo /etc/init.d/mysql restart
```

3.1.1.3 SUBVERSION

```
$ sudo apt-get install subversion
$ sudo apt-get install libapache2-svn
```

Al terminar la instalación de subversion es necesario reiniciar apache con el siguiente comando:

```
$ sudo /etc/init.d/apache2 restart
```

3.1.1.4 JAVA

El servidor también debe contar con Java Standard Edition versión 6 o 7. A continuación se especifica el procedimiento de instalación de Java SE 7.

1. Descargar los binarios del JDK de la página oficial:

http://www.oracle.com/technetwork/java/javase/downloads/java-se-jdk-7-download-432154.html

En este caso se obtiene el archivo **jdk-7-linux-x64.tar.gz** que corresponde a sistemas operativos linux de 64 bits.

2. Descomprimir el contenido del archivo en un lugar apropiado:

```
$ sudo tar zxvf jdk-7-linux-x64.tar.gz -C /usr/local
```

3. Agregar los enlaces simbólicos **java** y **javac** al sistema de alternativas de Debian:

```
$ sudo update-alternatives --install /usr/bin/java java
/usr/local/jdk1.7.0/bin/java 1

$ sudo update-alternatives --install /usr/bin/javac javac
/usr/local/jdk1.7.0/bin/javac 1
```

4. Indicar a Debian que tanto el comando **java** como el comando **javac** serán ejecutados por la versión del JDK de Oracle:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

En terminal se mostrará algo como esto:

Figura 3.1: Configuración del comando *java* en Debian Squeeze

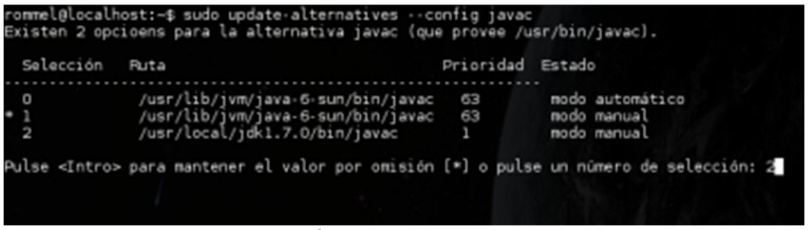


Figura 3.2: Configuración del comando *javac* en Debian Squeeze

5. Verificar las versiones de **java** y **javac** con:

```
$ java -version
$ javac -version
```

6. Editar el archivo **/etc/profile** para indicar al sistema la ruta a seguir para encontrar java:

```
# nano /etc/profile
```

Se añade la siguiente línea al final del archivo y se guardan los cambios (Ctrl+X):

```
JAVA_HOME=$JAVA_HOME:/usr/local/jdk1.7.0/bin/java
```

7. Se indica al sistema que los cambios realizados sobre el archivo **/etc/profile** son permanentes:

```
# source /etc/profile
```

3.1.1.5 GLASSFISH

1. Para instalar el servidor de aplicaciones Glassfish se obtiene el paquete de la página de Oracle:

http://www.oracle.com/technetwork/java/javaee/downloads/index.html

En este caso se elige *Java EE 6 Development Kit Bundles without JDK/ Java EE 6 SDK Update 4* y posteriormente se selecciona *java_ee_sdk-6u3-unix-ml.sh* (multilenguaje)

2. Es necesario otorgar permisos de ejecución al archivo .sh:

```
$ chmod +x java_ee_sdk-6u3-unix-ml.sh
```

3. Ejecutar el instalador:

```
$ sh java_ee_sdk-6u3-unix-ml.sh
```

En el asistente se elige *Instalación típica* y el directorio de instalación sugerido es en la carpeta personal (por ejemplo /home/oscar/glassfish3). Si todo va bien al terminar la instalación tendremos configurado *Glassfish*, en el puerto 4848 para administración y el 8080 para exponer aplicaciones web.

Para parar e iniciar Glassfish se debe crear un script en la carpeta **/etc/init.d**:

```
# nano /etc/init.d/glassfish
```

Del siguiente script se modifica la línea *asadmin="/home/oscar/glassfish3/bin/asadmin"* según la ubicación de la instalación.

```
#!/bin/bash

asadmin="/home/oscar/glassfish3/bin/asadmin"

case $1 in
start)
sh $asadmin start-domain
;;
stop)
sh $asadmin stop-domain
;;
restart)
sh $asadmin restart-domain
;;
*)
#Default case: restart de daemon
sh $asadmin restart-domain
;;
esac
exit 0
```

Se otorgan permisos de ejecución:

```
# chmod +x /etc/init.d/glassfish
```

y se ejecuta alguno de los siguiente comandos para parar, iniciar o reiniciar el servidor de aplicaciones glassfish:

```
# service glassfish stop
# service glassfish start
# service glassfish restart
```

3.1.1.6 REPOSITORIOS

Para llevar el control de versiones de la aplicación se crearán tres repositorios en el servidor, uno para la aplicación encargada del preprocesamiento de las imágenes del catálogo, otro para el

proyecto web que contendrá los servicios y el último que corresponde al de la aplicación de composición de imágenes que será utilizada por los clientes. Por lo tanto los repositorios quedan de la siguiente forma:

- ***repo*:** repositorio principal.
- ***repows*:** repositorio de Servicios web.
- ***repopre*:** repositorio de aplicación para el preprocesamiento de imágenes.

Los pasos para crear tales repositorios son los siguientes:

1. Crear directorios para los tres repositorios:

```
$ mkdir /home/oscar/repositorios/
$ mkdir /home/oscar/repositorios/repo
$ mkdir /home/oscar/repositorios/repows
$ mkdir /home/oscar/repositorios/repopre
```

2. Crear repositorios:

```
$ svnadmin create /home/oscar/repositorios/repo
$ svnadmin create /home/oscar/repositorios/repows
$ svnadmin create /home/oscar/repositorios/repopre
```

3. Asignar permisos:

```
# addgroup desarrollo
# adduser oscar desarrollo
# chown -R www-data:desarrollo /home/oscar/repositorios/repo/*
# chown -R www-data:desarrollo /home/oscar/repositorios/repows/*
# chown -R www-data:desarrollo /home/oscar/repositorios/repopre/*
# chmod -R g+w /home/oscar/repositorios/repo/*
# chmod -R g+w /home/oscar/repositorios/repows/*
# chmod -R g+w /home/oscar/repositorios/repopre/*
```

4. Abrir *dav_svn.conf* para editarlo:

```
# nano /etc/apache2/mods-available/dav_svn.conf
```

5. Buscar y comentar las siguientes líneas:

```
#<Location /svn>
#</Location>
```

6. Añadir lo siguiente:

```
<Location /repo>
      DAV svn
      SVNPath /home/oscar/repositorios/repo
      AuthType Basic
      AuthName "Repositorio principal"
      AuthUserFile /etc/subversion/passwd
      Require valid-user
```

```
</Location>
<Location /repows>
      DAV svn
      SVNPath /home/oscar/repositorios/repows
      AuthType Basic
      AuthName "Repositorio de servicios web"
      AuthUserFile /etc/subversion/passwd
      Require valid-user
</Location>
<Location /repopre>
      DAV svn
      SVNPath /home/oscar/repositorios/repopre
      AuthType Basic
      AuthName "Repositorio para el preprocesamiento de imágenes"
      AuthUserFile /etc/subversion/passwd
      Require valid-user
</Location>
```

7. Crear usuarios que tendrán acceso al repositorio:

```
# htpasswd2 -c /etc/subversion/passwd oscar
# htpasswd2 -c /etc/subversion/passwd dpinto
```

8. Reiniciar apache

```
# service apache2 restart
```

Al terminar los repositorios serán accesible vía HTTP únicamente para los usuarios registrados y tendrá un aspecto como el siguiente:

Figura 3.3: Repositorio SVN con interfaz Web

3.1.1.7 IMPLEMENTACIÓN Y PUESTA A PUNTO DE LA BASE DE DATOS

La base de datos creada recibió el nombre de "editor" y consta de una tabla sencilla para almacenar información de los usuarios. De esta forma, el esquema de la base de datos queda como sigue:

```
CREATE TABLE IF NOT EXISTS `users` (
 `name` varchar(20) NOT NULL,
 `pass` blob NOT NULL,
 PRIMARY KEY (`name`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

El campo *pass* siempre estará cifrado y será necesaria una clave para insertar y recuperar información de contraseñas, dicha clave será responsabilidad del administrador de la aplicación.

3.1.2 PREPROCESAMIENTO DE IMÁGENES

PNG (Portable Network Graphics) es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps [5L]. Las características más importantes del formato PNG para el desarrollo de un programa de edición de imágenes digitales son ***el soporte de transparencias*** e ***imágenes sin pérdida de datos***. Tales características acarrean un pequeño problema para la distribución de las imágenes: *el tamaño*.

Dado que la generación de imágenes de mejor resolución únicamente será llevada a cabo del lado del servidor de la aplicación, se propone realizar un preprocesamiento a las imágenes que serán distribuidas a los usuarios finales, entre las operaciones a realizar se encuentran:

Optimización de imágenes PNG.

Cambio de resolución (especialmente la reducción de tamaño) manteniendo el aspecto relacionado.

Generación de thumbnails.

Almacenamiento de datos en un BD embebida.

Las operaciones de preprocesamiento no serán realizadas necesariamente en el equipo servidor, así que se especifican los requerimientos mínimos:

Hardware

Requisitos mínimos:

Procesador a 2.0 GHz
Memoria RAM: 2GB
Espacio libre en disco: 20 GB

Software

Sistemas Operativos permitidos: GNU/Linux.
Software requerido: java, python, pngnq, phatch, GPRename, *PNGResizer*.

3.1.2.1 OPTIMIZACIÓN DE IMÁGENES

Existen diversos programas informáticos que ya tratan el tema de la optimización de imágenes con formato PNG. PNGNQ utiliza una red neuronal para optimizar la selección de mapa de colores. Para realizar la optimización de manera masiva se utiliza un script creado en python.

Instalación y uso

En sistemas basados en Debian instalamos los siguientes paquetes de los repositorios:

```
$ sudo apt-get install pngnq
$ sudo apt-get install python
```

y se utiliza desde línea de comandosExisten diversos programas informáticos que ya tratan el tema de la optimización de imágenes con formato PNG. PNGNQ utiliza una red neuronal para optimizar la selección de mapa de colores. Para realizar la optimización de manera masiva se utiliza un script creado en python.

```
$ pngnq image.png -d directorioSalida
```

que genera un nueva imagen llamada ***image-nq8.png*** en el directorio especificado.

A continuación se obtiene el script en python del repositorio de preprocesamiento:

```
http://www.postemotion.com/repopre/trunk/optimiza_png.py
```

se asignan permisos y se ejecuta como sigue:

```
$ chmod +x optimiza_png
$ ./optimiza.png /directorio/con/imagenes/png
```

si se omite el segundo parámetro el directorio que tomará por defecto para buscar imágenes png será el directorio donde se está ejecutando el script. El script creará una carpeta llamada *optimizadas* con las imágenes procesadas por **pngnq**.

Nota: Los nombres de las imágenes no deben contener los caracteres **(** , **)** y **,** para que el script en python pueda procesar todas las imágenes correctamente. Se puede utilizar el software GPRename para el renombrado masivo en sistemas linux.
Ejemplo:

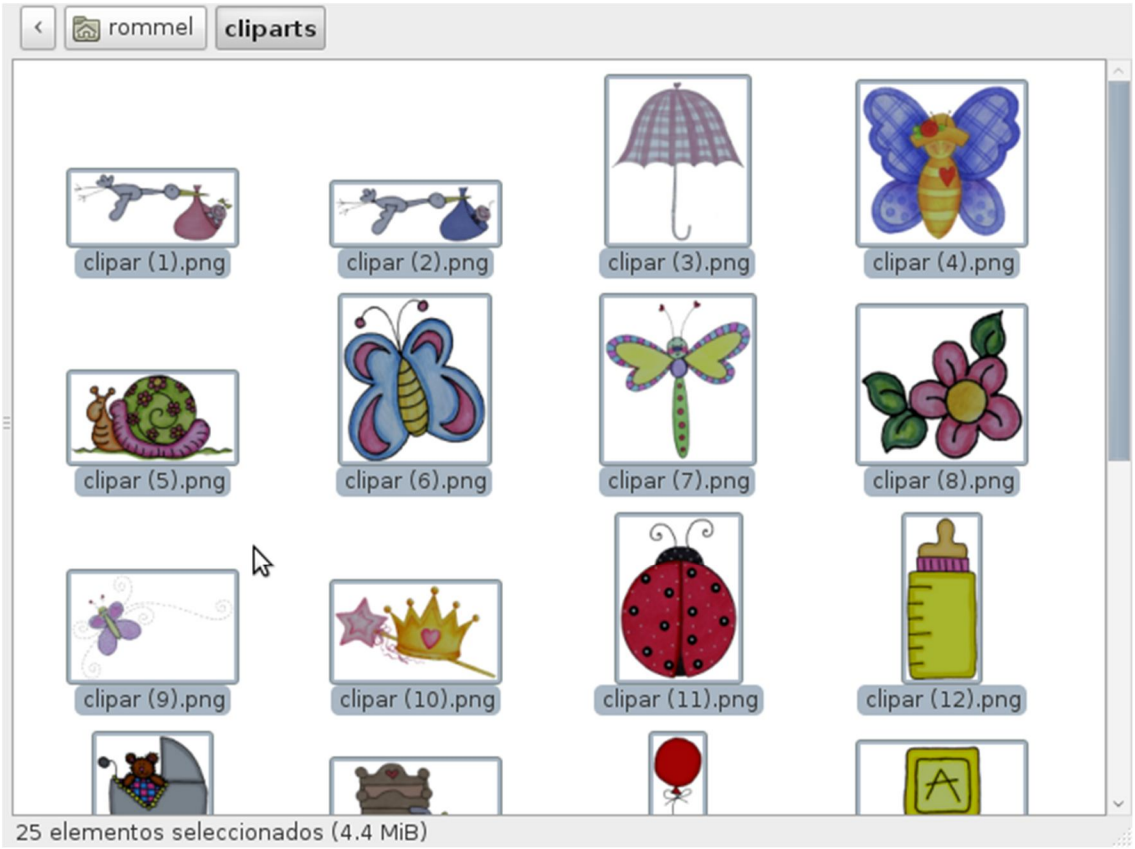


Figura 3.4: Cliparts antes de optimizar

Para optimizar las 25 imágenes primero debemos renombrar de tal forma que no contengan paréntesis en los nombres. Entonces utilizamos *GPRename*:

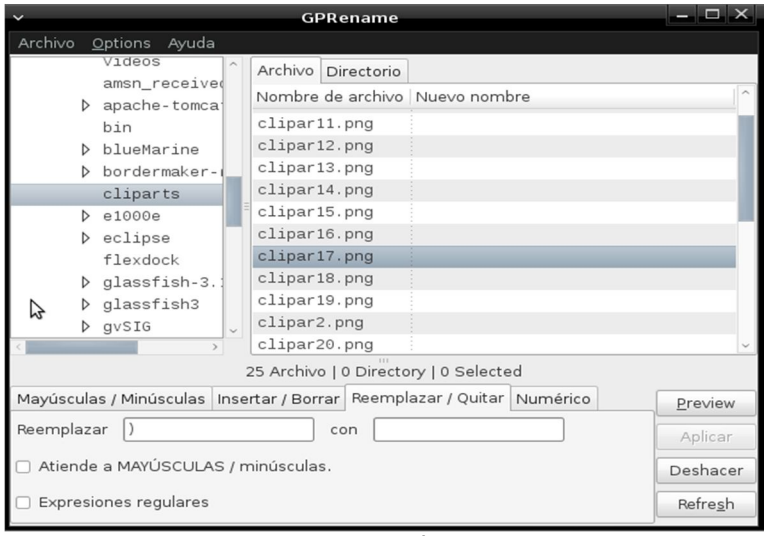


Figura 3.5: Renombrando imágenes con GPRename

Ejecutamos el script y obtenemos las imágenes optimizadas:

```
rommel@localhost:~$ chmod +x optimiza_png.py
rommel@localhost:~$ ./optimiza_png.py cliparts/
```

Figura 3.6: Ejecución de script de optimización

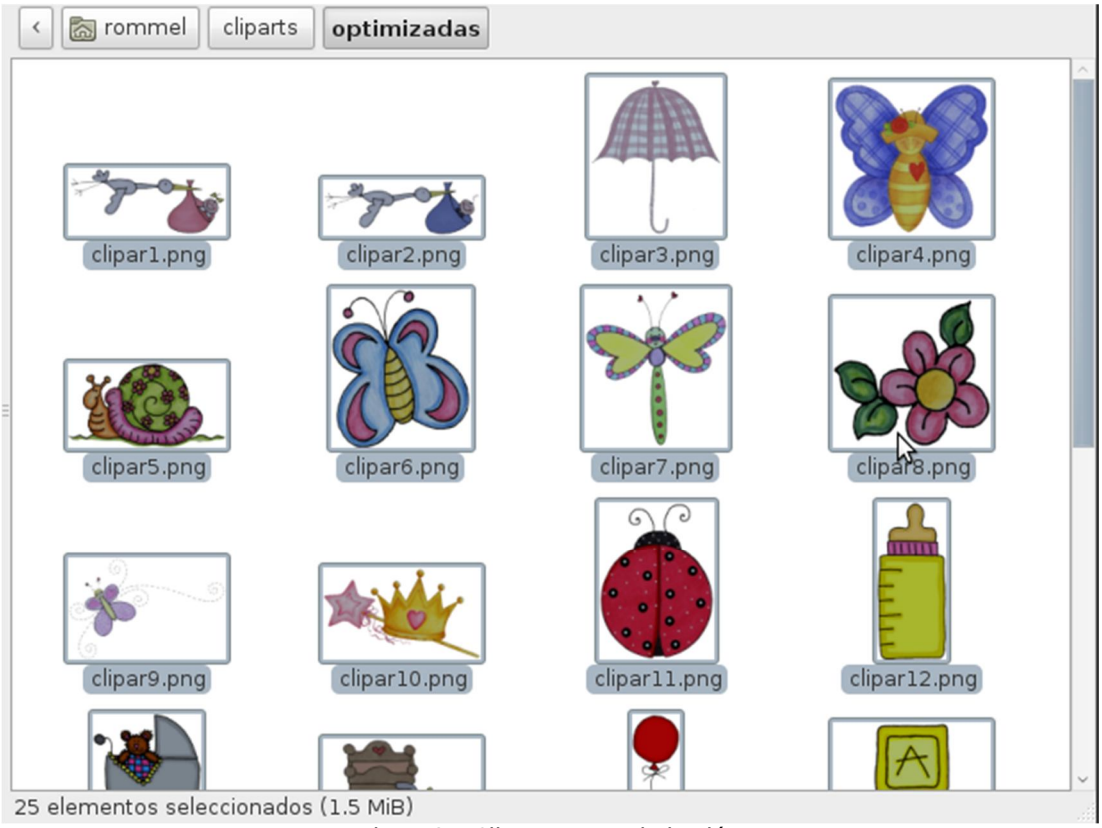


Figura 3.7: Cliparts tras optimización

3.1.2.2 CAMBIO DE RESOLUCIÓN

Para que la distribución de las imágenes sea lo más ligera posible es necesario disminuir su resolución. Además. el cambio de tamaño debe mantener el aspecto relacionado. Para ésta y otras tareas se desarrolló *PNG Resizer*.

Instalación y uso

Se descarga del repositorio de la aplicación:

http://www.postemotion.com/repopre/trunk/Installers/

Ejecutar instalador, aceptar licencia e instalar:

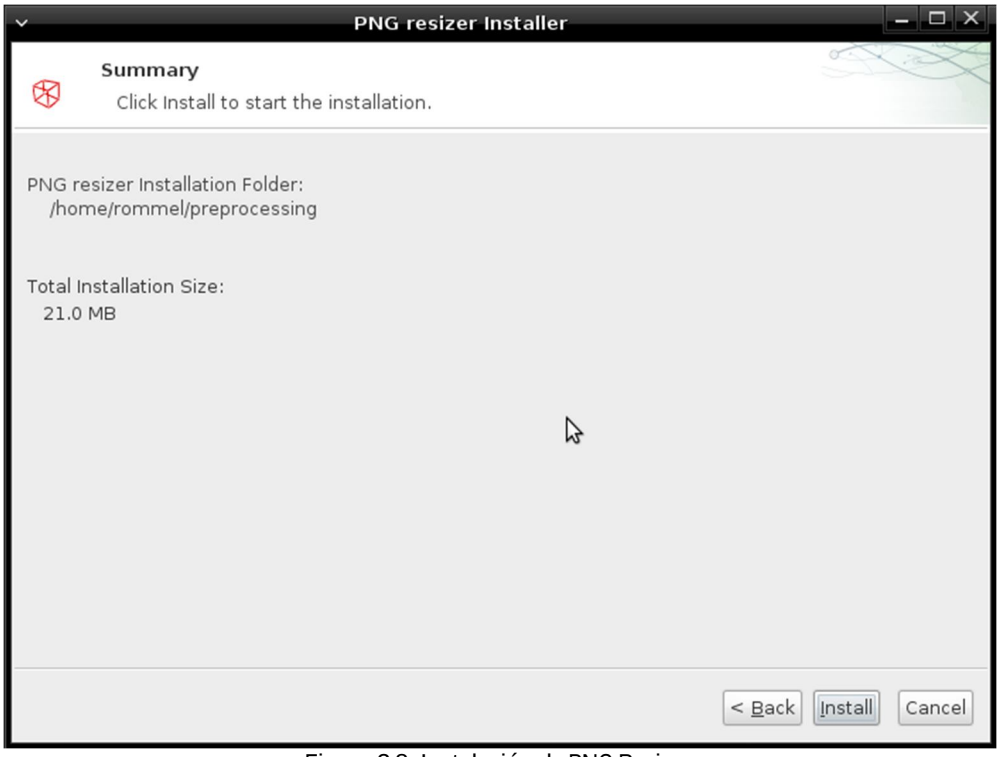


Figura 3.8: Instalación de PNG Resizer

Se creará un acceso directo a la aplicación sobre el Escritorio:

Figura 3.9: Icono de la aplicación

Figura 3.10: Inicio de PNG Resizer

Una vez en la aplicación se selecciona el directorio que contiene las imágenes PNG a procesar:

Figura 3.11: Seleccionando directorio de trabajo en PNG Resizer

Se seleccionan las imágenes de salida de *pngnq,* es decir, el directorio *optimizadas*, entonces la aplicación mostrará miniaturas de las imágenes a procesar y se habilitarán las opciones Redimensionar y BD Embebida:

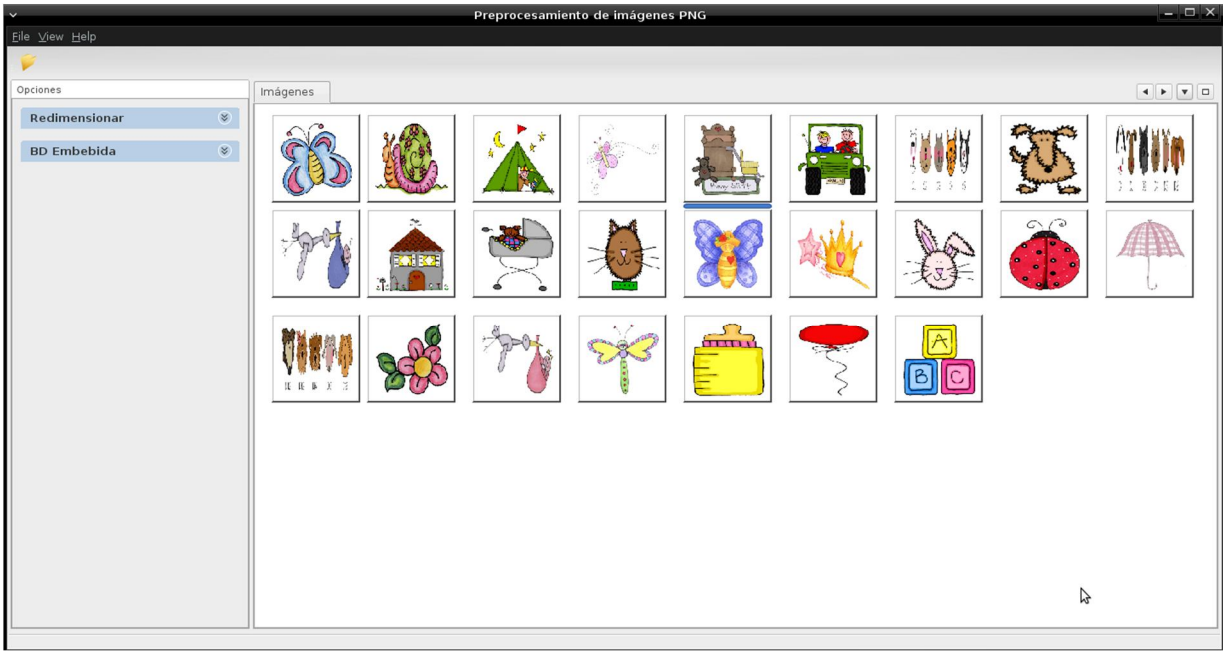


Figura 3.12: Interfaz de PNG Resizer

Se puede seleccionar el **tipo de interpolación** y decidir si las imágenes se redimensionan por **porcentaje** o por **número de pixeles**. Al terminar las operaciones de cambio de tamaño se generará una nueva carpeta con las imágenes procesadas:

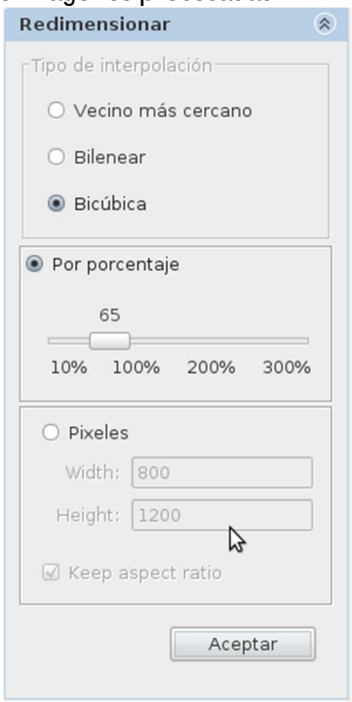


Figura 3.13: Opciones de cambio de tamaño

Figura 3.14: Directorio generado por PNG Resizer

3.1.2.3 POBLANDO LA BASE DE DATOS

La finalidad de aplicar las operaciones de **optimización** y **cambio de tamaño** a las imágenes es hacer posible la distribución de la mayor cantidad de *marcos*, *máscaras*, *fondos* y *cliparts*. Tal distribución será implementada mediante una **Base de Datos embebida** manejada a través de **Apache Derby**.

Apache Derby es un sistema gestor de base de datos relacional escrito en Java que puede ser empotrado en aplicaciones Java y utilizado para procesos de transacciones online. Tiene un tamaño de 2 MB de espacio en disco. Inicialmente distribuido como IBM Cloudscape, Apache Derby es un proyecto open source licenciado bajo la Apache 2.0 License. Actualmente se distribuye como Sun Java DB [6L].

Para la BD se contará con 4 tablas: marcos, máscaras, fondos y cliparts. Cada tabla tendrá los siguientes campos:

ID (INTEGER)	**NOMRE (STRING)**	**IMAGEN (BLOB)**	**THUMBNAIL (BLOB)**

Tabla 3.1: Tabla de imágenes para la Base de Datos embebida

El panel correspondiente a la **BD Embebida** mostrará entonces **a qué tabla se guardarán las imágenes del directorio seleccionado**.

NOTA: En el caso de haber aplicado la optimización elegir el directorio *optimizadas* y en el caso de haber aplicado la opción de *cambio de tamaño* seleccionar el directorio creado (en el ejemplo *percentage65*).

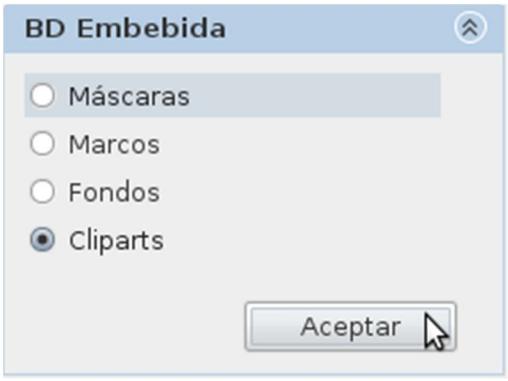


Figura 3.15: Selección de tipo de imágenes que serán almacenadas en la base de datos

La Base de Datos se guardará en el directorio del usuario (**/home/usuario/BDEmbebida** en linux y **Mis Documentos/BDEmbebida** en Windows), si la BD no existe se creará una vacía y en caso contrario se utilizará para el proceso de llenado.

Figura 3.16: BDEmbebida generada por PNG Resizer

Los thumbnails de las imágenes se generarán automáticamente en el directorio seleccionado con un tamaño de 100x100 pixeles manteniendo el aspecto relacionado,

Figura 3.17: Miniaturas generadas por PNG Resizer

Ya que el tamaño de cada proyecto es configurado por cada usuario es posible que las imágenes de **cliparts** no cuenten con la resolución suficiente para proyectos de alta resolución. En la siguiente tabla se puede observar que para proyectos a 300 dpi y de 21.7 x 28.9 cm la imagen resultante es de 3413x2560 pixeles:

Tamaño (pixels)	Mega Pixels	80 dpi (monitor)	133 dpi (en cm)	150 dpi (en cm)	175 dpi (en cm)	200 dpi (en cm)	250 dpi (en cm)	300 dpi (en cm)
640 x 480	0.3	15.3 x 20.3	9.2 x 12.2	8.1 x 10.8	7.0 x 9.3	6.1 x 8.1	4.9 x 6.5	4.1 x 5.4
800 x 600	0.5	19.1 x 25.4	11.5 x 15.3	10.2 x 13.5	8.7 x 11.6	7.6 x 10.2	6.1 x 8.1	5.1 x 6.8
1024 x 768	0.8	24.4 x 32.6	14.7 x 19.6	13.0 x 17.3	11.1 x 14.9	9.8 x 13.0	7.8 x 10.4	6.5 x 8.7
1280 x 960	1.2	30.4 x 40.6	18.3 x 24.4	16.3 x 21.7	13.9 x 18.6	12.2 x 16.3	9.8 x 13.0	8.1 x 10.8
1600 x 1200	1.8	38.1 x 50.9	22.9 x 30.6	20.3 x 27.1	17.4 x 23.2	15.2 x 20.3	12.2 x 16.3	10.2 x 13.5
1920 x 1440	2.6	45.7 x 61.0	27.5 x 36.7	24.4 x 32.5	20.9 x 27.9	18.3 x 24.4	14.6 x 19.5	12.2 x 16.3
2048 x 1536	3.0	48.7 x 65.0	29.3 x 39.1	26.0 x 34.7	22.3 x 29.7	19.5 x 26.0	15.6 x 20.8	13.0 x 17.3
2272 x 1704	3.9	54.0 x 72.1	32.5 x 43.4	28.8 x 38.5	24.7 x 32.9	21.6 x 28.8	17.3 x 23.0	14.4 x 19.2
2731 x 2048	5.3	65.0 x 86.8	39.1 x 52.2	34.7 x 46.2	29.7 x 39.6	26.0 x 34.7	20.8 x 27.7	17.3 x 23.1
3413 x 2560	8.3	81.3 x 108.4	48.9 x 65.2	43.3 x 57.8	37.2 x 49.5	32.5 x 43.3	26.0 x 34.7	21.7 x 28.9

Calidad Baja
Calidad Media
Calidad Alta
Profesional

Figura 3.18: Tipos de proyectos fotográficos

si el clipart mostrado abajo es agregado al proyecto que termina en una imágen de 3413x2560 pixeles se llega a que su tamaño final será de 3.1x2.7cm y esto no deja mucho margen para escalar.

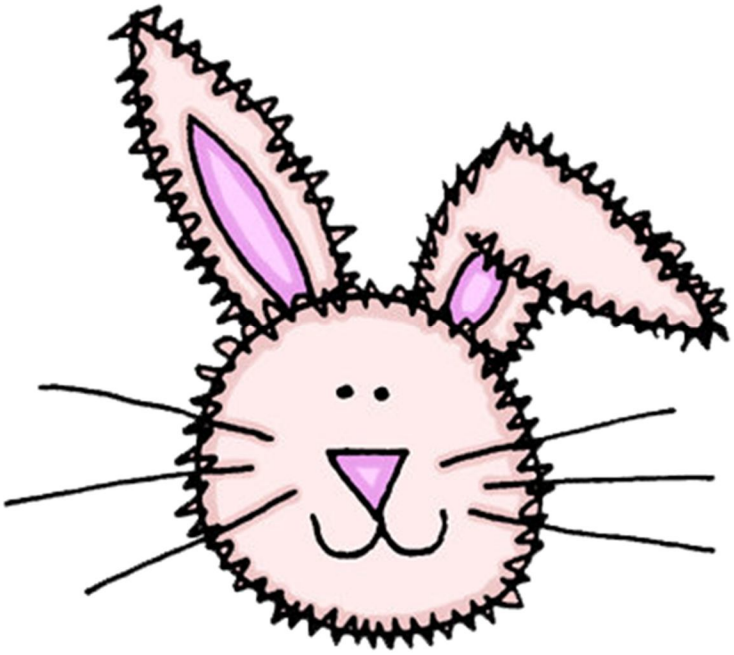


Figura 3.19: Clipart de 369x326 pixeles

El tamaño de impresión se obtiene con la siguiente fórmula:

Tamaño de impresión = Número de píxeles/ Resolución

El resultado está dado en pulgadas.

Suponiendo que el clipart original fuera de una resolución de 1107 x 900 y tomando el mismo tipo de proyecto de a 300 dpi y de 21.7 x 28.9 cm (con una imagen resultante es de 3413x2560 pixeles) el clipart ocuparía 6.23 x 7.62 cm que es cerca del 30% del tamaño del proyecto, dejando margen para escalar.

Por ejemplo como el descrito arriba se recomienda escalar todos los cliparts por lo menos a una resolución de 1000*1000 para evitar pixeleado a la hora de escalar.

3.1.3 ÁRBOL DE DIRECTORIOS PARA IMÁGENES Y PROYECTOS

Las imágenes originales del catálogo serán almacenadas en el servidor de la aplicación y sólo serán accesibles al administrador. Ya que la aplicación debe permitir agregar nuevas imágenes al catálogo, se optó por utilizar el sistema de archivos del servidor para facilidad del administrador de la aplicación. De modo que se tiene la siguiente estructura de directorios en el servidor:

/
-> **media**
…
-> **editor**
-> **usuarios**
-> **imagenes**
-> **cliparts**
-> **optimizadas**
-> **thumbs**
-> **fondos**
-> **optimizadas**
-> **thumbs**
-> **marcos**
-> **optimizadas**
-> **thumbs**
-> **mascaras**
-> **optimizadas**
-> **thumbs**

La raíz de los directorios de la aplicación es la carpeta *editor*. El directorio *usuarios* contendrá una carpeta por cada usuario de la aplicación que guardará sus proyectos. Las carpetas *cliparts*, *fondos*, *marcos y mascaras* almacenarán las imágenes originales y sus directorios *optimizadas* y *thumbs* contendrán las imágenes procesadas por PNGResizer.

Para agregar nuevas imágenes al catálogo es necesario agregar tres imágenes dependiendo su tipo. Por ejemplo, si se desea agregar un nuevo clipart es necesario agregar la imagen original en la carpeta **cliparts**, su imágen optimizada en **optimizadas** y su miniatura (de 100x100) en **thumbs**.

3.1.4 IMPLEMENTACIÓN DE APLICACIÓN CLIENTE

La herramienta más importante para la composición de imágenes es el editor que utiliza el cliente para crear sus proyectos. Tal herramienta fue desarrollada sobre la Plataforma NetBeans, por esa razón la implementación de realizó en módulos. Cada módulo intenta ser lo más cohesivo posible y gracias al estructura del framework para la comunicación entre módulos, se logró perder acoplamiento.

A continuación se muestran los módulos de la aplicación:

- **Editor**: contiene el componente visual del editor, el lienzo o área de trabajo. También incluye a los *Widgets*, que son los componentes sobre los que residen las imágenes en el lienzo. Los Widgets son una modificación de la clase Widget de *Visual Library*[7L] incluída en Netbeans Platform. Visual Library incorpora operaciones tales como zoom, cambio de posición, cambio de plano, drag & drop, entre otras.

- **Flamingo Integration**: responsable de la integración de la barra ribbon [8L] con la aplicación. Esta barra contiene los accesos a las operaciones sobre las imágenes.

- **Login**: módulo dedicado al logueo y registro de usuarios.

- **SSUtils**: contiene clases utilitarias, sobre todo las clases que operan sobre imágenes.

- **Thumbs**: código fuente de los thumbnails utilizados para representar imágenes miniatura del catálogo y de imágenes de usuario.

- **ThumbnailManager**: módulo responsable de la creación de tiras de imágenes miniaturas del catálogo de imágenes.

- **UpdateBD**: verifica si existen actualizaciones para el catálogo de imágenes y actualiza la base de datos local.

- **Usuario**: conjunto de clases que representa el selector de imágenes de usuario.

- **WebServiceClient**: lleva a cabo la comunicación de la aplicación con el servidor.

3.1.5 AGREGAR SERVICIOS

Para agregar algún servicio se debe modificar la clase **org.lri.servcios.InfoImg.java** del proyecto *ServiciosBD*. Si un servicio es agregado y se requiere consumir en el cliente se deben seguir los siguientes pasos:

1. Limpiar y construir  proyecto *ServiciosBD*
2. Desplegar ServiciosBD.war en el servidor de aplicaciones
3. Abrir el proyecto *WebServiceClient*
4. Ir a *Web Services References -> InfoImg*, click derecho -> *Refresh* y Confirmar
5. Ir a *Web Services References -> InfoImg*, click derecho -> *Edit Web Service Attributes*
6. En la pestaña *WSDL Customization* ir a *Port Types -> InfoImg* y deshabilitar la opción *Enable Wrapper Style*
7. Limpiar y construir  proyecto *WebServiceClient*
8. Limpiar y construir  módulo **WebServiceClient**
9. Click derecho sobre módulo **WebServiceClient** -> *Properties -> Libraries -> Wrapped JARs*
10. Remover *WebServiceClient.jar* y agregar el nuevo *WebServiceClient.jar* ubicado en la carpeta dist/ de *WebServiceClient*

3.1.6 GENERANDO INSTALADORES

El instalador necesita una base de datos que contenga el catálogo de imágenes. La base de datos se genera como se explica en el apartado de preprocesamiento**.** Una vez creada la base de datos se comprime en formato ZIP y se llevan a cabo los siguientes pasos:

1. Ubicar el directorio **harness** en el directorio de instalación de Netbeans, en sistemas linux suele encontrarse en /usr/local/netbeans-7.0.1/harness por ejemplo.

2. Copiar el archivo ZIP **BDEmbebida.zip** en:
`harness/nbi/stub/ext/components/products/helloworld/src/org/mycompany`

3. Abrir
`/usr/local/netbeans-7.0.1/harness/nbi/stub/ext/components/products/helloworld/src/org/mycompany/ConfigurationLogic.java`

4. Agregar los siguientes imports:

```
import java.io.BufferedOutputStream;
import java.io.OutputStream;
import java.net.URL;
import java.security.CodeSource;
import java.util.ArrayList;
import java.util.List;
import java.util.Stack;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.ZipEntry;
import java.util.zip.ZipInputStream;
```

5. Agregar el siguiente código antes de la finalización de la clase:

```
public void recursiveUnzip(Class clazz, File outFolder) {
      try{

          URL url = clazz.getResource("/org/mycompany/BDEmbebida.zip");

          this.createFolder(outFolder, true);
          BufferedOutputStream out = null;
          ZipInputStream in = new ZipInputStream(url.openStream());
          ZipEntry entry;
          while((entry = in.getNextEntry()) != null){
              System.out.println("Extracting: " + entry);
              int count;
              byte data[] = new byte[BUFFER];

              // write the files to the disk
              File newFile = new File(outFolder.getPath() + "/" + entry.getName());
              Stack<File> pathStack = new Stack<File>();
              File newNevigate = newFile.getParentFile();
              while(newNevigate != null){
                  pathStack.push(newNevigate);
                  newNevigate = newNevigate.getParentFile();
              }
              //create all the path directories
              while(!pathStack.isEmpty()){
                  File createFile = pathStack.pop();
                  this.createFolder(createFile, true);
              }
              if(!entry.isDirectory()){
                  out = new BufferedOutputStream(
                          new FileOutputStream(newFile), BUFFER);
```

```
                while((count = in.read(data, 0, BUFFER)) != -1){
                    out.write(data, 0, count);
                }
                this.cleanUp(out);
                //recursively unzip files
                if(entry.getName().toLowerCase().endsWith(".zip")){
                    String zipFilePath = outFolder.getPath() + "/" + entry.getName();
                    this.recursiveUnzip(clazz,     new     File(zipFilePath.substring(0,
zipFilePath.length() - 4)));
                }
            }else{
                this.createFolder(new File(entry.getName()), true);
            }
        }
        this.cleanUp(in);
        return;
    }catch(Exception e){
        e.printStackTrace();
        return;
    }
  }


  public void removeAllZipFiles(File folder) {
    String[] files = folder.list();
    for(String file : files){
        File item = new File(folder.getPath() + "/" + file);
        if(item.exists() && item.isDirectory()){
            this.removeAllZipFiles(item);
        }else if(item.exists() && item.getName().toLowerCase().endsWith(".zip")){
            item.delete();
            System.out.println(item.getName() + " Removed!!");
        }
    }
  }


  private void createFolder(File folder, boolean isDriectory) {
    if(isDriectory){
        folder.mkdir();
    }
  }


  private void cleanUp(InputStream in) throws Exception {
    in.close();
  }


  private void cleanUp(OutputStream out) throws Exception {
    out.flush();
```

```
    out.close();
}
private final static int BUFFER = 1024;
```

6. Agregar al final del método **install** el siguiente código:

```
    recursiveUnzip(getClass(), new File(pathDB) );
```

Una vez que ha sido agregada la base de datos como se describe arriba solo falta generar el instalador.

Netbeans Platform permite generar instaladores para Windows, Linux, Mac OS X y Solaris de una forma rápida y sencilla.:

1. Click derecho sobre el proyecto *EditorServicio -> Propiedades -> Installer* y marcar los instaladores deseados:

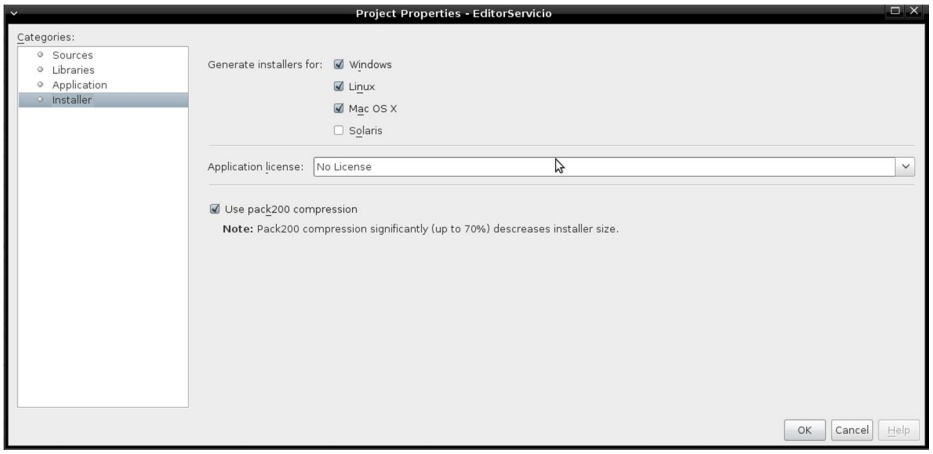


Figura 3.20: Generación de instaladores

2. Click derecho sobre el proyecto *EditorServicio -> Package as -> Installers* y se generarán los instaladores en la carptea **dist/** de *EditorServicio.*

CAPÍTULO 4. FUNCIONAMIENTO DEL PROGRAMA

La aplicación será compatible con Sistemas Operativos Windows Xp, Windows Vista, Windows 7, Sistemas GNU/Linux, Mac OS. Para sistemas windows el instalador constará de un archivo .exe y para sistemas Linux será un archivo .sh.

El icono de la aplicación se encontrará en el Escritorio tras finalizar la instalación. Haciendo doble clic sobre el icono arrancará el programa.

4.1 ENTORNO DE TRABAJO

El entorno de trabajo consta de una interfaz estilo Ribbon, sustituyendo barras de menús y herramientas por pestañas. Además , cuenta con un panel explorador de carpetas, un panel visor de imágenes de usuario, ,un panel visor de imágenes predeterminadas y un espacio de trabajo para la creación de proyectos de imágenes digitales.

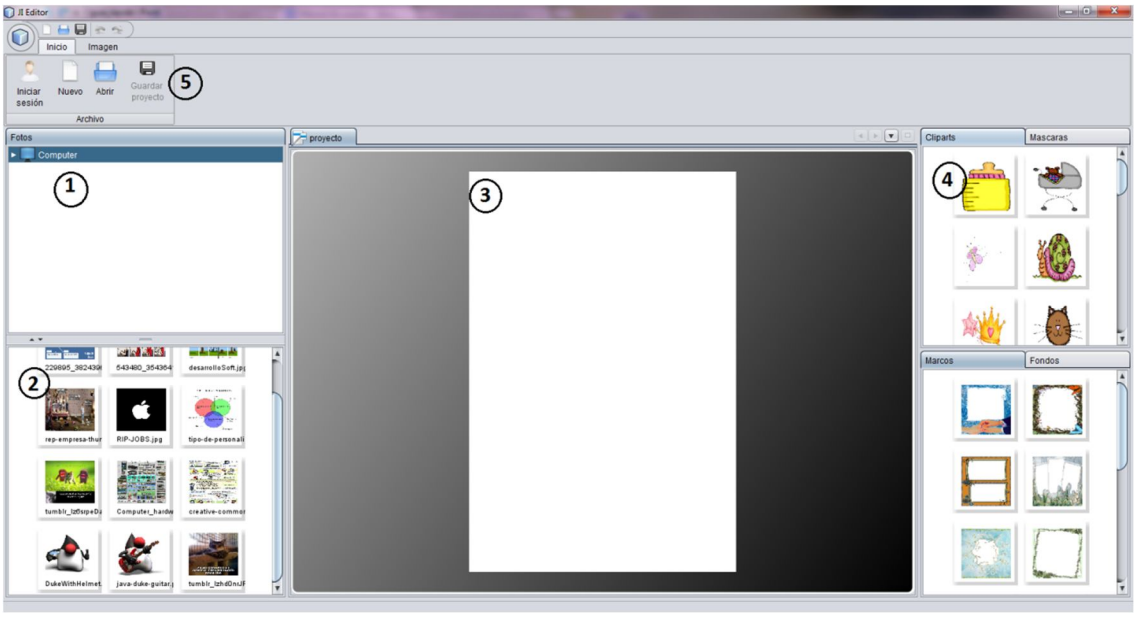


Figura 4.1: Entorno de trabajo

1.- Explorador de carpetas.
2.- Tus imágenes.
3.- Área de trabajo.
4.- Catálogo de imágenes.
5.- Barra de herramientas ribbon

4.1.1 EXPLORADOR DE CARPETAS

El explorador de carpetas permite seleccionar la carpeta dónde se encuentran las imágenes con las que se quiere trabajar. Al hacer un clic en una carpeta, automáticamente se muestran las imágenes que contiene, en miniatura, en el panel visor de imágenes.

Para obtener mayor espacio en este panel es posible cambiar el tamaño desde sus bordes.

4.1.2 PANEL VISOR DE IMÁGENES

El visor de imágenes muestra todas aquellas imágenes que se encuentran en la carpeta seleccionada en el explorador de carpetas. Además permite arrastrar las imágenes dentro del proyecto.

4.1.3 ÁREA DE TRABAJO

En esta zona se visualiza el lienzo sobre el que se arrastran fondos, cliparts, marcos, máscaras y fotografías. El tamaño del área de trabajo será proporcional al tamaño que se indique el crear el proyecto.

4.1.4 CATÁLOGO DE IMÁGENES

La aplicación cuenta con un catálogo de imágenes de fondos, marcos, máscaras y cliparts.

Para agregar un fondo al proyecto se debe arrastrar alguno hacia el proyecto, este fondo cubrirá todo el lienzo y siempre quedará por debajo de las demás imágenes.

Para agregar un marco al proyecto se debe seleccionar la pestaña Marcos y arrastrar alguno sobre el lienzo.

Los cliparts son dibujos útiles para decorar fotografías de eventos especiales como cumpleaños, fiestas, viajes, etc. Estas imágenes también pueden ser agregadas arrastrándolas sobre el lienzo.

La pestaña Máscaras contiene imágenes con una gran cantidad de formas de bordes para fotografías. Las máscaras se aplican únicamente sobre imágenes de usuario. Para agregar una máscara se debe arrastrar y soltar sobre una imagen de usuario.

4.1.5 BARRA HERRAMIENTAS RIBBON

El objetivo de esta barra de herramientas es ubicar todas las funcionalidades de la aplicación en un solo lugar para que sean fáciles de alcanzar. A continuación se explica con detalle cada operación que puede ser realizada a través de esta barra.

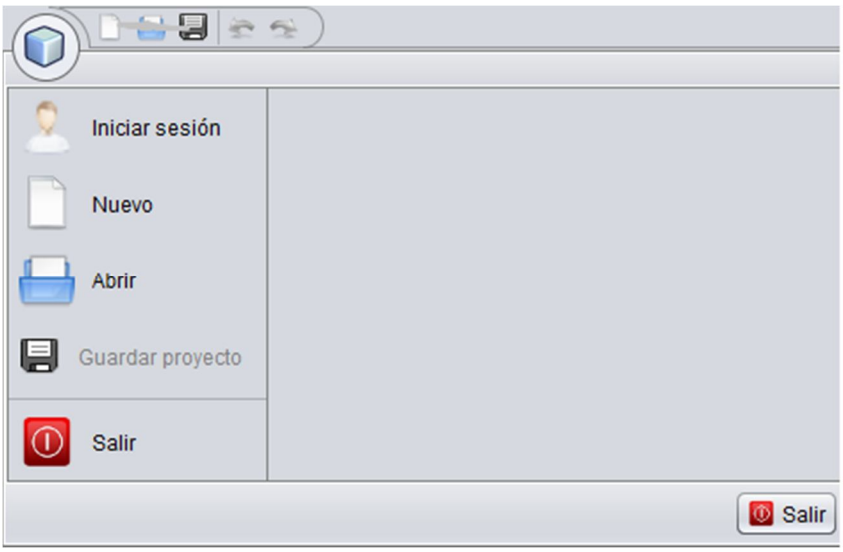


Figura 4.2: Menú principal y barra de accesos rápidos

El menú principal se despliega haciendo clic sobre el ícono  . Este menú es similar al menú *Archivo* en una aplicación con barra de menú.

4.1.5 .1 INICIAR SESIÓN

Haciendo click sobre el elemento *Iniciar sesión* se desplegará la siguiente ventana de diálogo:

Figura 4.3: Dialogo de inicio de sesión

En esta ventana es posible iniciar sesión o registrarse en los servidores de la aplicación para obtener ventajas como generar e imprimir proyectos en alta resolución y recibir actualizaciones del catálogo de imágenes.

Para crear una cuenta en *JI Editor* dar clic en el botón **Crear cuenta** y proporcionar los datos requeridos:

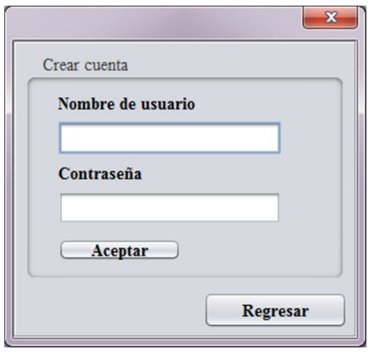


Figura 4.4: Dialogo de Registro

4.1.5 .2 NUEVO (CTRL+N)

Haciendo clic sobre el elemento *Nuevo* se mostrará la siguiente ventana:

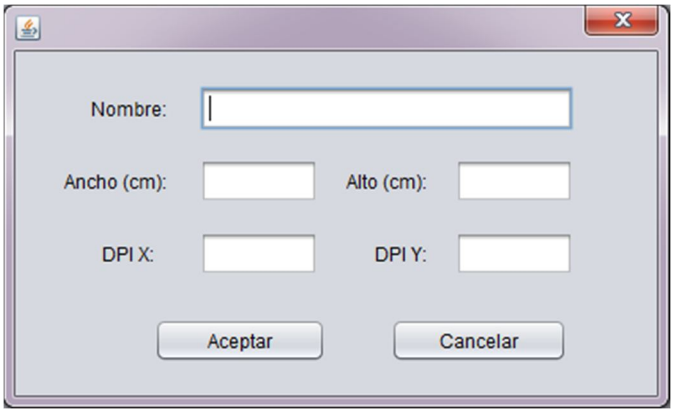


Figura 4.5: Dialogo para la creación de Proyectos

En esta ventana se deben proporcionar los siguientes datos para la creación de un nuevo proyecto:

- Nombre: el nombre del proyecto.
- Ancho: el tamaño de la anchura deseada del proyecto en centímetros.

- Altura: el tamaño de la altura deseada del proyecto en centímetros.
- DPI X: indica el número de píxeles por pulgada a lo ancho para el proyecto.
- DPI Y: indica el número de píxeles por pulgada a lo alto para el proyecto.

La siguiente tabla puede ser útil para tomar una decisión a la hora de crear un proyecto:

Tamaño (pixels)	Mega Pixels	80 dpi (monitor)	133 dpi (en cm)	150 dpi (en cm)	175 dpi (en cm)	200 dpi (en cm)	250 dpi (en cm)	300 dpi (en cm)
640 x 480	0.3	15.3 x 20.3	9.2 x 12.2	8.1 x 10.8	7.0 x 9.3	6.1 x 8.1	4.9 x 6.5	4.1 x 5.4
800 x 600	0.5	19.1 x 25.4	11.5 x 15.3	10.2 x 13.5	8.7 x 11.6	7.6 x 10.2	6.1 x 8.1	5.1 x 6.8
1024 x 768	0.8	24.4 x 32.6	14.7 x 19.6	13.0 x 17.3	11.1 x 14.9	9.8 x 13.0	7.8 x 10.4	6.5 x 8.7
1280 x 960	1.2	30.4 x 40.6	18.3 x 24.4	16.3 x 21.7	13.9 x 18.6	12.2 x 16.3	9.8 x 13.0	8.1 x 10.8
1600 x 1200	1.8	38.1 x 50.9	22.9 x 30.6	20.3 x 27.1	17.4 x 23.2	15.2 x 20.3	12.2 x 16.3	10.2 x 13.5
1920 x 1440	2.6	45.7 x 61.0	27.5 x 36.7	24.4 x 32.5	20.9 x 27.9	18.3 x 24.4	14.6 x 19.5	12.2 x 16.3
2048 x 1536	3.0	48.7 x 65.0	29.3 x 39.1	26.0 x 34.7	22.3 x 29.7	19.5 x 26.0	15.6 x 20.8	13.0 x 17.3
2272 x 1704	3.9	54.0 x 72.1	32.5 x 43.4	28.8 x 38.5	24.7 x 32.9	21.6 x 28.8	17.3 x 23.0	14.4 x 19.2
2731 x 2048	5.3	65.0 x 86.8	39.1 x 52.2	34.7 x 46.2	29.7 x 39.6	26.0 x 34.7	20.8 x 27.7	17.3 x 23.1
3413 x 2560	8.3	81.3 x 108.4	48.9 x 65.2	43.3 x 57.8	37.2 x 49.5	32.5 x 43.3	26.0 x 34.7	21.7 x 28.9

Calidad Baja
Calidad Media
Calidad Alta
Profesional

Figura 4.6: Tipos de proyectos fotográficos

Al crear el proyecto el nuevo lienzo que se muestra cuenta con las dimensiones proporcionales al tamaño que tendrá la imagen final que será impresa, además, todas la imágenes en el proyecto serán escaladas automáticamente para mantener la misma proporción.

4.1.5 .3 ABRIR (CTRL+A)

Haciendo click sobre el elemento *Abrir* se mostrará la siguiente ventana:

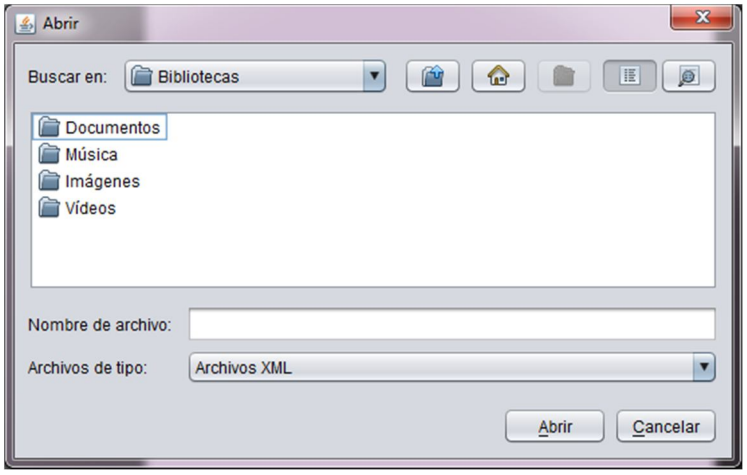
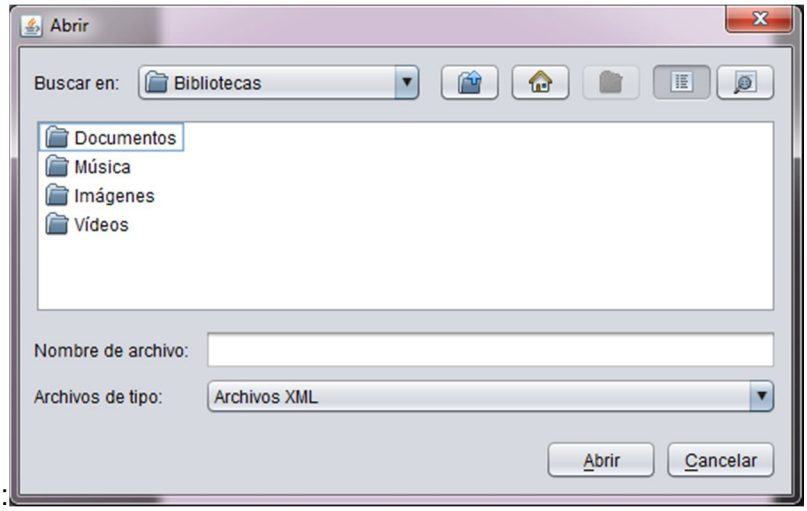


Figura 4.7: Dialogo para Abrir proyectos

En caso de contar con proyectos guardados con *JI Editor* es posible abrirlos para continuar con su edición.

4.1.5 .4 GUARDAR PROYECTO

Esta opción se activa cuando una imagen es agrega al proyecto o cuando se realiza alguna modificación. Al seleccionar *Guardar proyecto* se solicitará la ruta y el nombre con el que se guardará



:

Figura 4.8: Dialogo para Guardar proyectos

Un archivo será generado con el nombre especificado y la extensión xml.

4.1.5 .5 ACCESOS RÁPIDOS 

Esta pequeña barra cuenta con accesos rápidos para las opciones *Nuevo*, *Abrir* y *Guardar Proyecto* que se encuentran en el Menú Principal. También cuenta con dos nuevos elementos: *deshacer* y *rehacer*.

4.1.5 .6 DESHACER/REHACER (CTRL+Z/CTRL+Y)

El elemento *deshacer* es activado cuando el proyecto sufre alguna modificación como agregar algún tipo de imagen, cambiar una imagen de posición o aplicar un filtro a una imagen. El

resultado de aplicar ésta operación es regresar al estado inmediato anterior del proyecto. También es posible *deshacer* un cambio con la combinación de teclas **Ctrl+Z**.

El elemento *rehacer* es activado al *deshacer* un cambio. El resultado de aplicar ésta operación es regresar al estado del proyecto antes de *deshacer* una operación.

4.1.5 .7 PESTAÑA IMAGEN

Al seleccionar la pestaña *Imagen* se desplegará el siguiente panel:

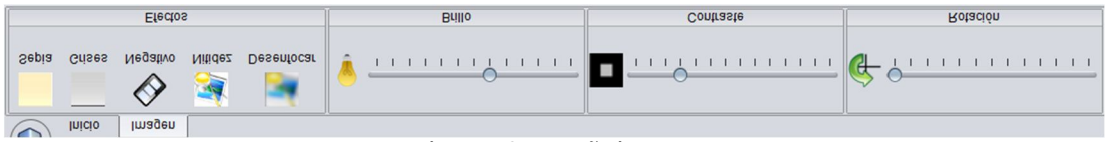


Figura 4.9: Pestaña imagen

Se muestran distintos filtros aplicables a imágenes en el lienzo. Para aplicar cualquiera de los filtros es necesario seleccionar la imagen haciendo clic sobre ella.

Los filtros *nitidez* y *desenfocar* son acumulativos.

4.1.5 .8 RATÓN

También es posible realizar diversas operaciones con el ratón como cambio de posición, cambio de tamaño, traer al frente y enviar atrás.

Para cambiar una imagen de posición basta con arrastrarla hacia el lugar deseado y soltarla en esa posición,
Para cambiar de tamaño una imagen se debe colocar el puntero del ratón sobre la imagen a modificar y arrastrar por las esquinas que aparecen en los bordes:

Figura 4.10: Ejemplo de clipart seleccionado y listo para cambiar su tamaño

También es posible cambiar las imágenes de plano, excepto el fondo. Para esto se debe hacer clic derecho sobre una imagen y seleccionar la opción deseada del siguiente menú contextual:

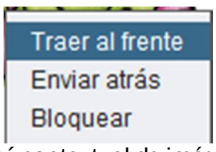


Figura 4.11: Menú contextual de imágenes en el lienzo

4.2 EJEMPLO DE USO

Para crear un nuevo proyecto haremos doble clic sobre el ícono en el Escritorio y se mostrará la siguiente pantalla:

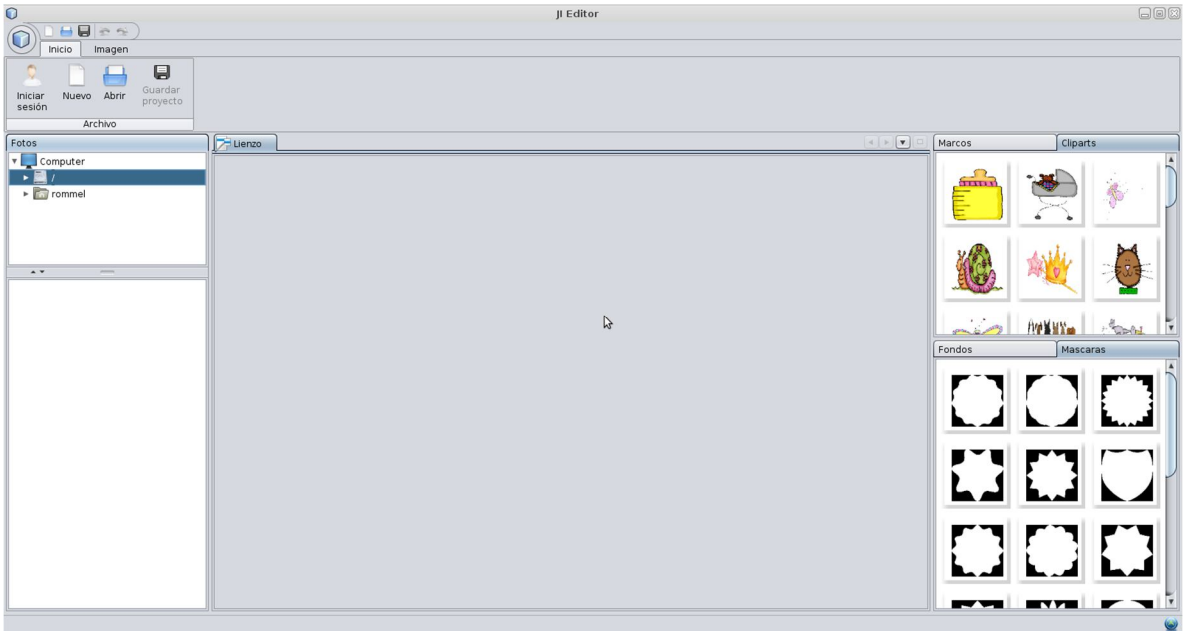


Figura 4.12: Interfaz de la aplicación

Elegiremos la opción *Nuevo* y escribiremos el nombre del proyecto, sus dimensiones y su resolución para impresión:

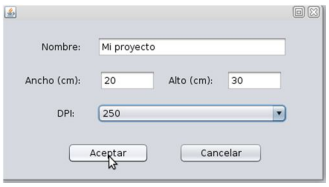


Figura 4.13: Creando un nuevo proyecto

Pulsamos *Aceptar* y veremos el lienzo donde crearemos nuestro proyecto, este lienzo es escalado automáticamente por la aplicación para que mantenga las proporciones reales del proyecto:

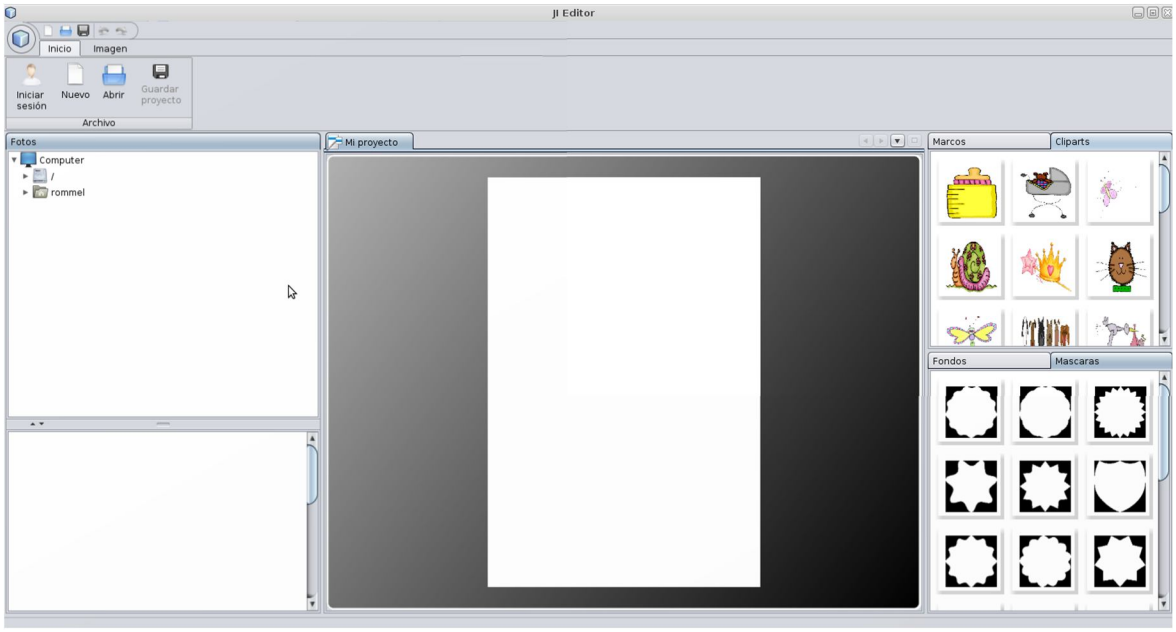


Figura 4.13: Creando un nuevo proyecto

En el explorador de imágenes elegiremos la carpeta donde se encuentran las fotos que utilizaremos en el proyecto:

Figura 4.14: Explorador de archivos de imagen

Para agregar imágenes basta con arrastrarlas hacia el lienzo (zona blanca):

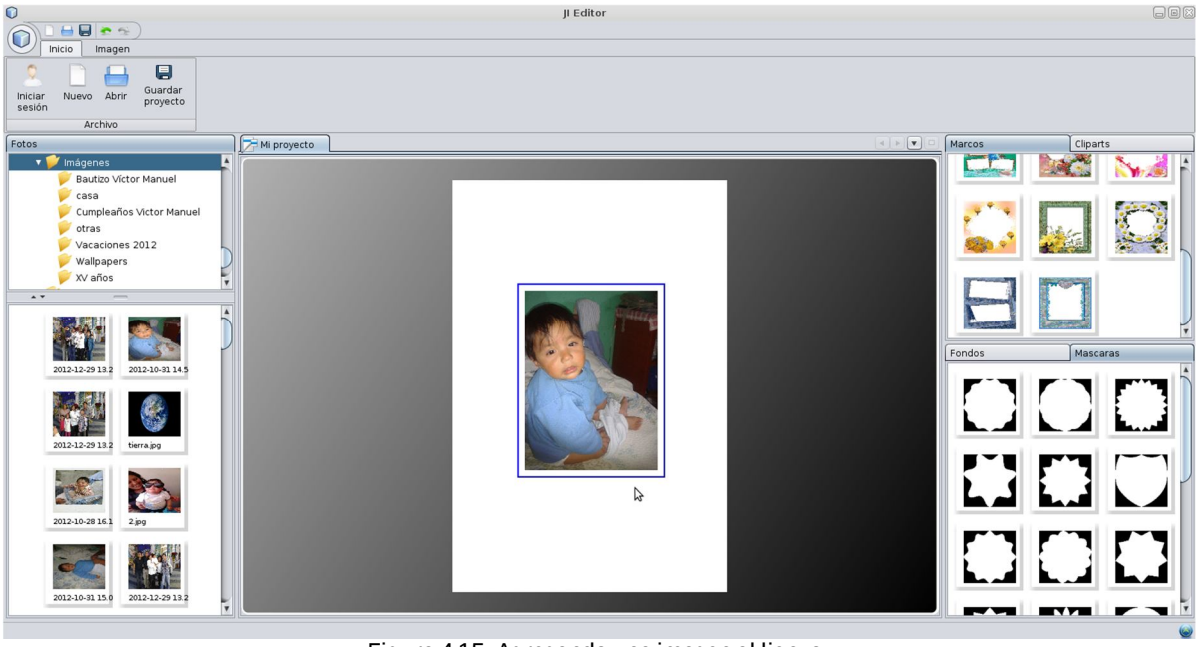


Figura 4.15: Agregando una imagen al lienzo

Es posible aplicar una máscara sobre nuestras imágenes, solo es necesario arrastrar la máscara a aplicar del panel de Máscaras y soltarla sobre la imagen deseada, al soltarla la máscara será aplicada:

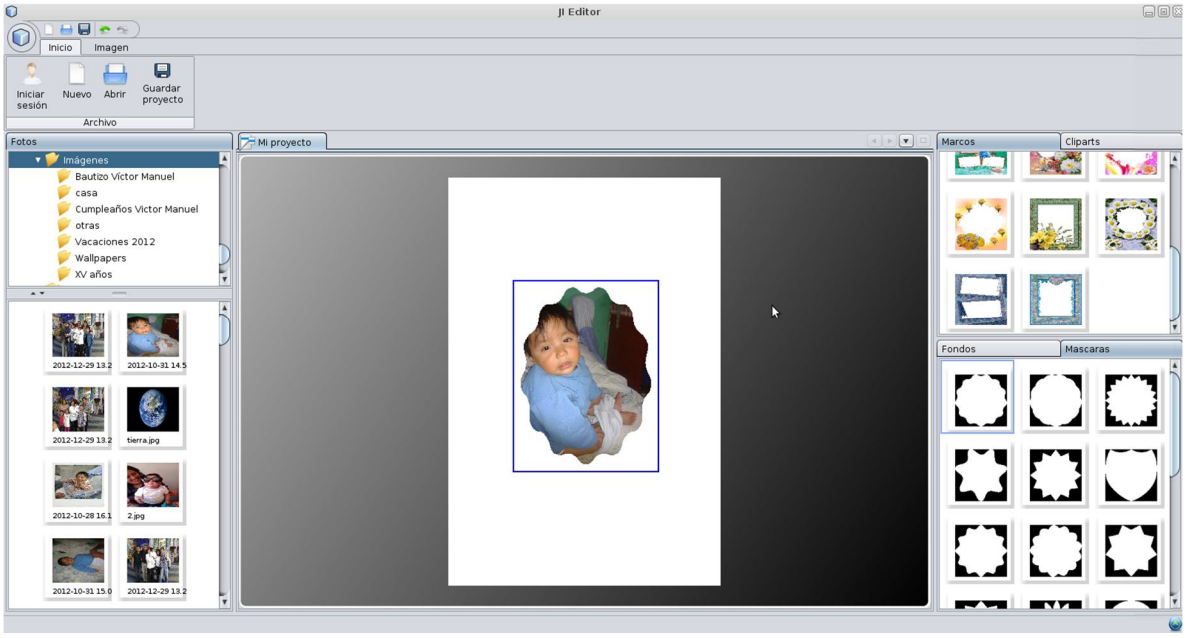


Figura 4.16: Imagen con una máscara aplicada

Es posible agregar un efecto a la imagen a través del menú Imagen, para aplicarlo es necesario seleccionar la imagen deseada haciendo clic sobre ella y eligiendo una opción de la categoría de efectos, por ejemplo el paso a grises:

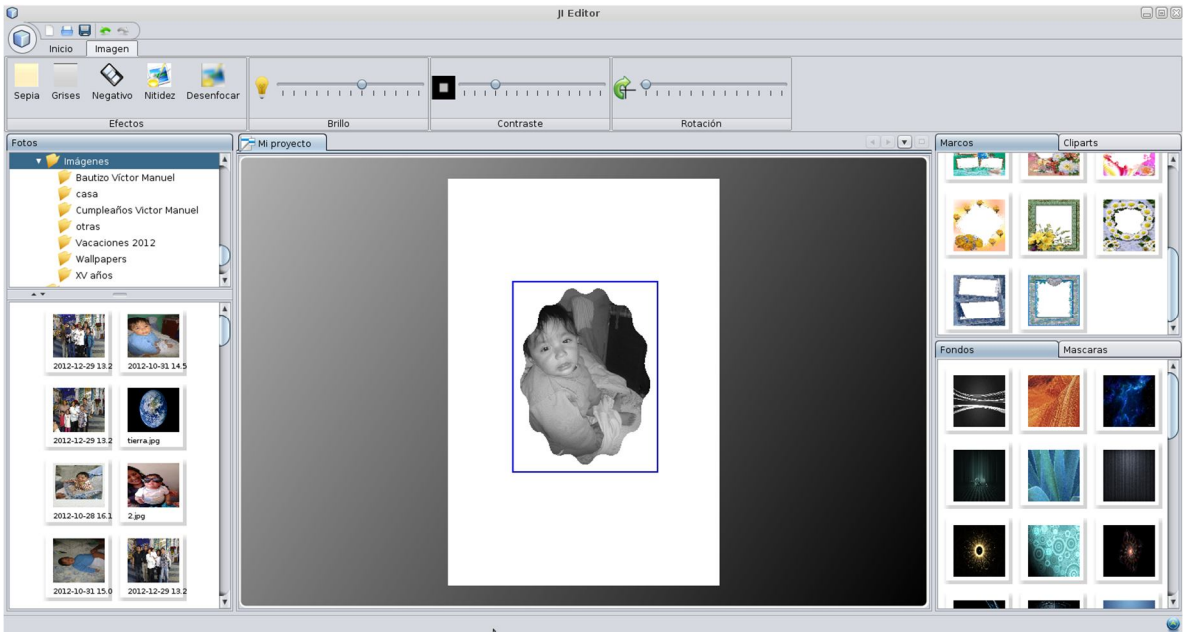


Figura 4.17: Imagen pasada a grises

Para rotar una imagen la seleccionamos y movemos el control de rotación según nos convenga, en el momento del desplazamiento del control la imagen rotará automáticamente:

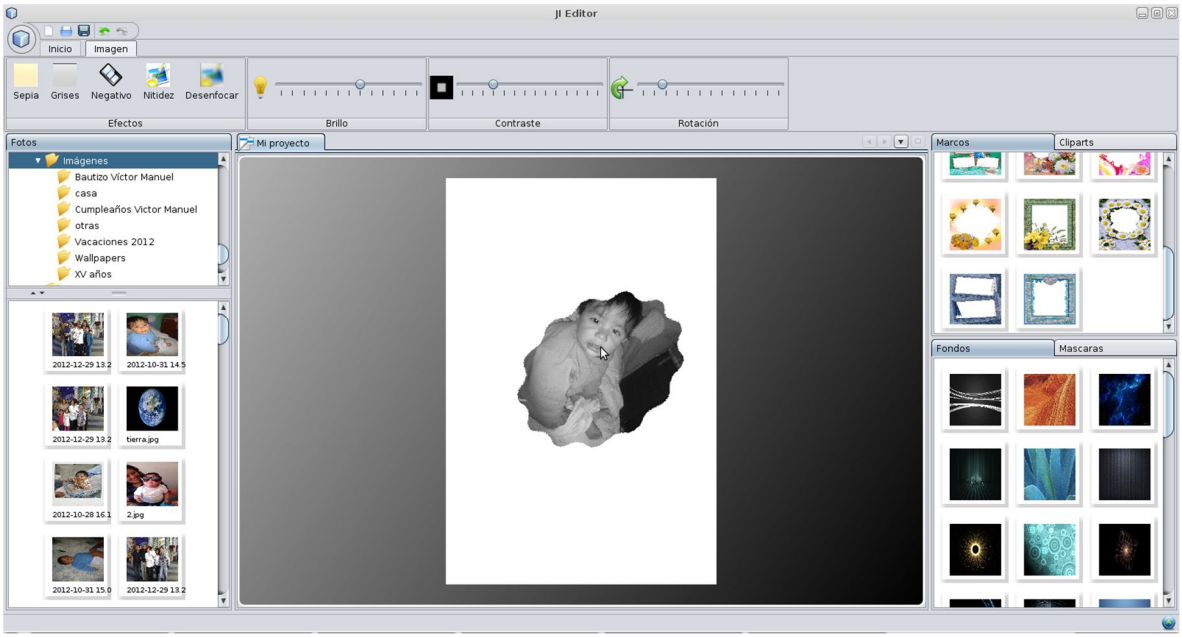


Figura 4.18: Imagen rotada

Como podemos observar la rotación de la imagen no interfiere con la aplicación de máscaras o efectos de imagen.

El cambio de brillo y contraste funciona de la misma manera en que lo hace la rotación:

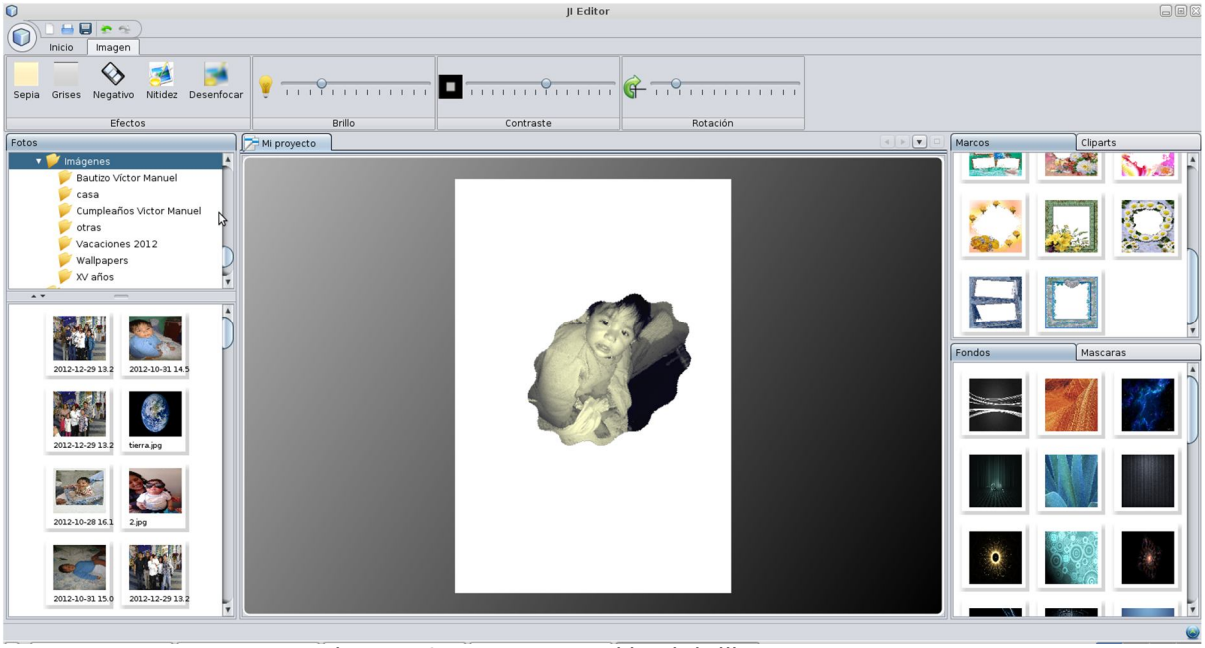


Figura 4.19: Imagen con cambios de brillo y contraste

Para cambiar una imagen de ubicación basta con arrastrarla hacia la posición deseada:

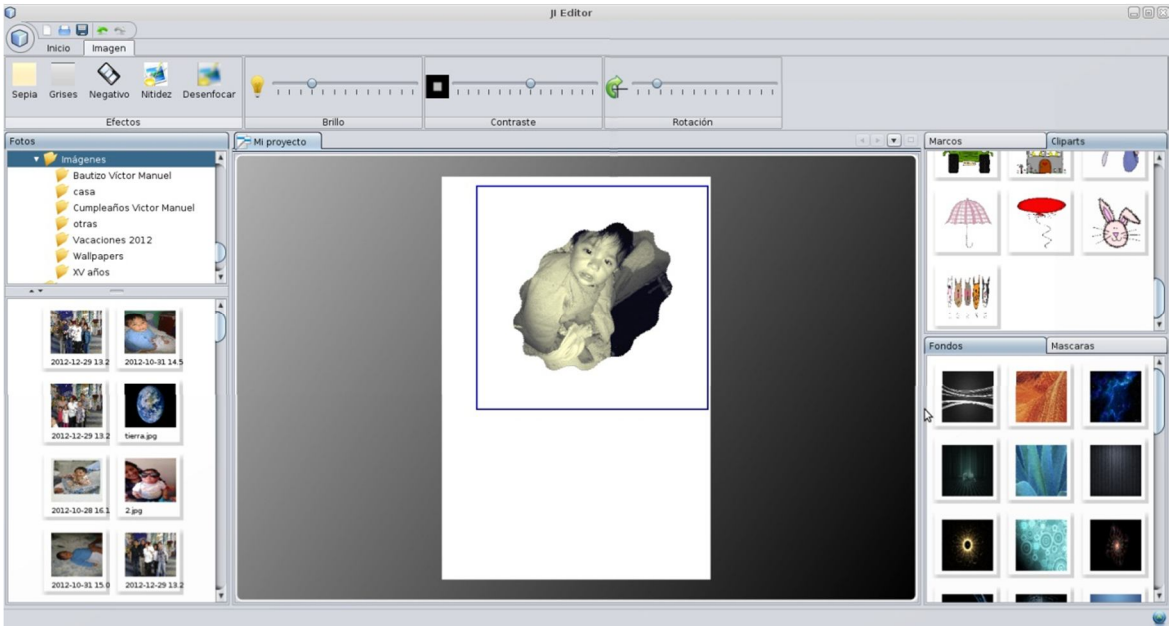


Figura 4.20: Imagen que ha cambiado de ubicación

Ahora aplicamos un fondo sobre nuestro lienzo, para esto, arrastramos el fondo y lo soltamos sobre cualquier punto del lienzo:

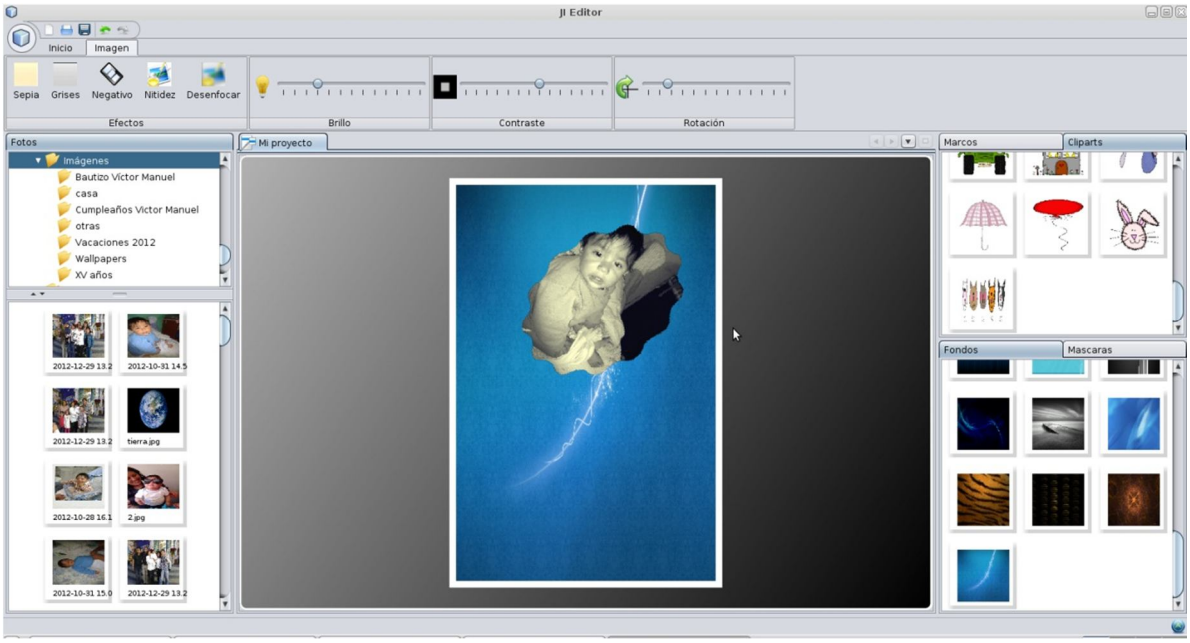


Figura 4.20: Fondo agregado al proyecto

Otro tipo de imágenes en el catálogo es del de Clipart, este tipo de imágenes siempre son agregadas encima de las demás y es posible enviarlas hacia atrás o enfrente de otras imágenes (excepto hacia atrás del fondo):

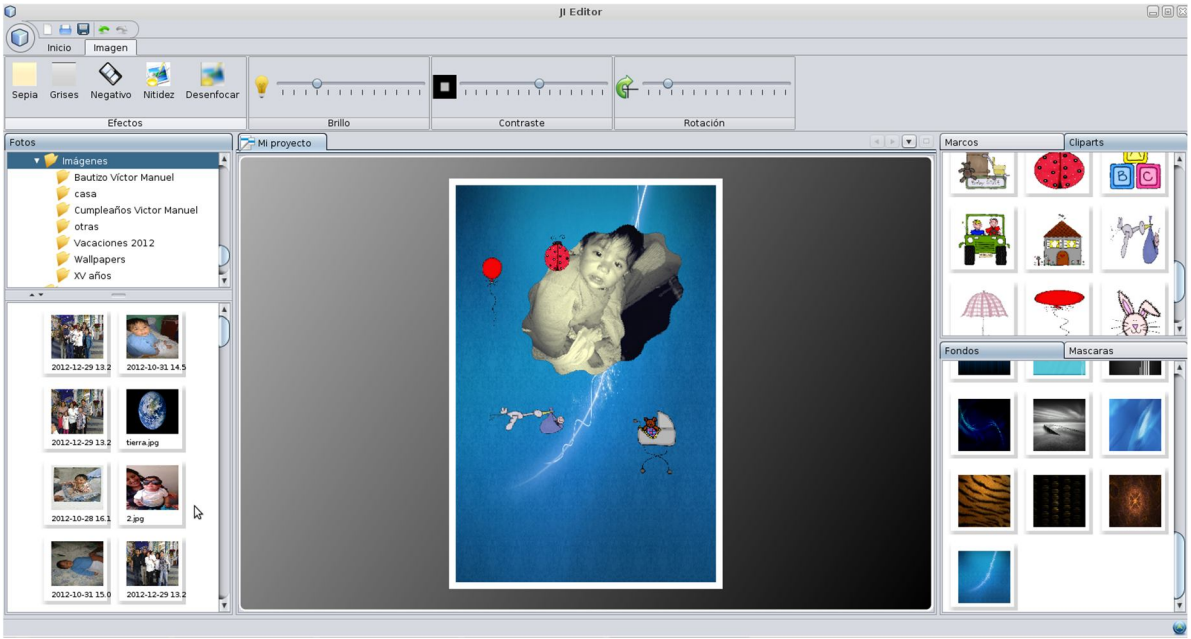


Figura 4.21: Agregando cliparts

Para cambiar el tamaño de las imágenes pasamos el puntero del ratón sobre la imagen deseada, hacemos clic en uno de los puntos de zoom que aparecen y arrastramos hacia adentro o hacia afuera según sea necesario:

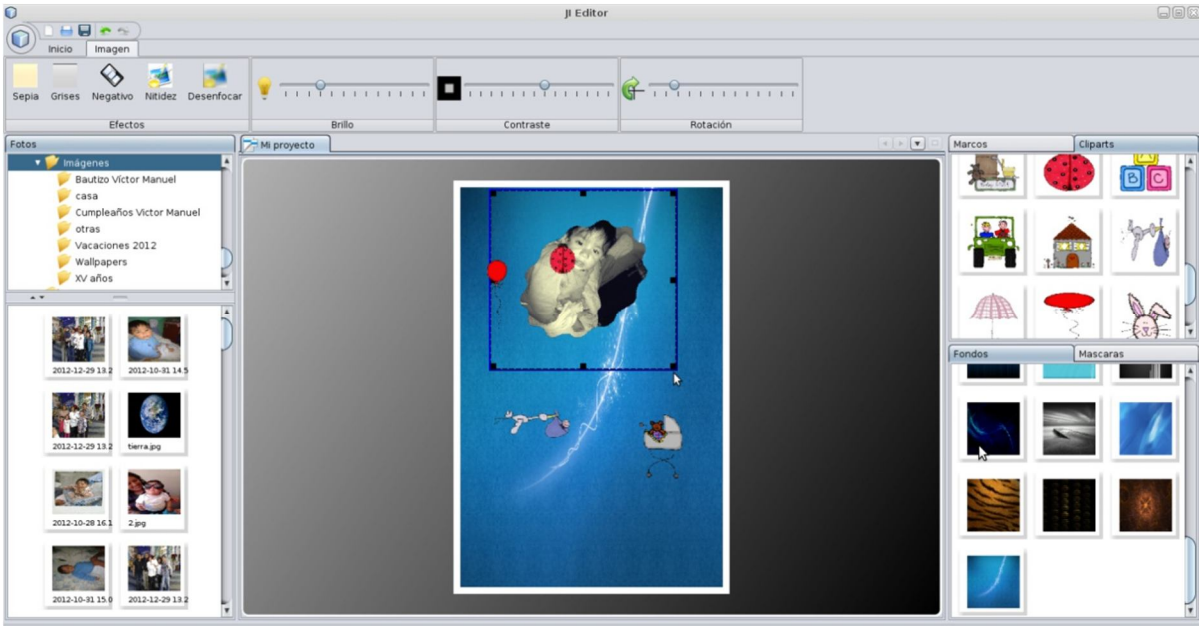


Figura 4.22: Cambiando tamaño

Agregando algunas imágenes más y aplicando las operaciones antes descritas podemos obtener el siguiente proyecto que guardaremos en formato XML:

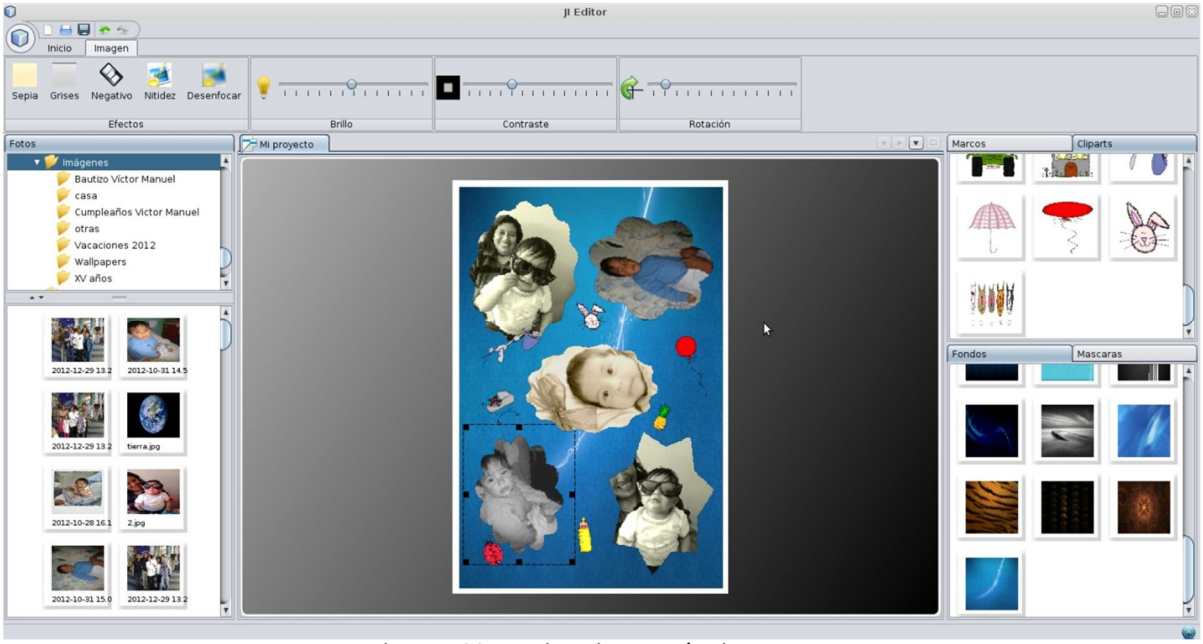


Figura 4.23: Un ejemplo con más elementos

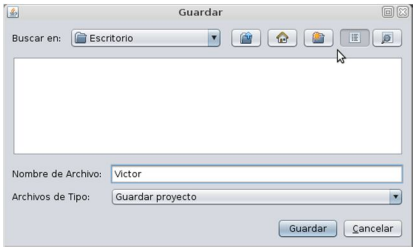


Figura 4.24: Guardando proyecto

Si hemos iniciado sesión antes de guardar, el proyecto será enviado al servidor de la aplicación y la imagen de alta calidad será generada, en este ejemplo la imagen generada es de 1968x2952 pixeles y puede ser impresa en un tamaño de 20x30cm:

Figura 4.24: Imagen final

CONCLUSIONES

En el transcurso de la investigación abordada en esta tesis se fueron generando conceptos, definiciones y diseños, para el desarrollo del sistema de composición de imágenes. Sobre la base del cuerpo teórico desarrollado se construyó un sistema computacional en el que se concreta dicha teoría desarrollada.

El presente proyecto presenta un sistema de composición de imágenes para usuarios no expertos en el tema de diseño gráfico. Se desarrollaron tres proyectos que componen el sistema: la aplicación de preprocesamiento de las imágenes PNG, el proyecto Web que contiene los servicios necesarios para la interacción cliente-servidor y la aplicación de escritorio que es el producto que utilizan los usuarios finales.

Las experiencias obtenidas se describen como:

Se presentó una metodología orientada a objetos para el análisis, diseño e implementación del sistema de composición de imágenes. Se desarrollaron diferentes diagramas UML que describen el modelo de la parte funcional del sistema.

El sistema principal se desarrolló en diferentes módulos con el objetivo de obtener una alta cohesión y un bajo acoplamiento de las clases que los componen. Además de utilizar una herramienta de control de versiones para gestionar los cambios en el código fuente y de esta manera facilitar la actualización o refactorización del sistema.

Se desarrolló un sistema cliente-servidor, donde el cliente corresponde a la aplicación de composición de imágenes y el servidor a una serie de servicios, bases de datos, repositorios de software y una estructura de directorios que tienen la tarea de interactuar de forma adecuada con la aplicación cliente.

TRABAJOS A FUTURO

Gracias al desarrollo por módulos es posible añadir funcionalidad al sistema de una manera sencilla, de esta forma podemos listar algunas mejoras que pueden ser desarrolladas para el sistema:

- Agregar operaciones que pueden ser realizadas sobre las imágenes.
- Agregar un nuevo tipo de imágenes al catálogo.
- Cambiar la forma de registro e inicio de sesión para conectarse con su cuenta de Google o Facebook a través de la modificación del proyecto Web.
- Cambiar la interfaz de usuario si se considera necesario.
- Agregar herramienta de recorte.
- Agregar herramienta que permite escribir textos sobre el proyecto.
- Cambiar los métodos que realizan las operaciones de las imágenes por una librería especializada.

También es posible mejorar la parte dedicada a los servicios y buscar una mejor optimización de las imágenes del catálogo para llevar a cabo una implementación basada en Web y en el diseño orientado a objetos presentado en este trabajo tesis.

REFERENCIAS

INTERNET

[1L] Página web Dr. Manuel Martín Ortiz

Última visita 27 de octubre de 2012

[2L] Página web Dr. Manuel Martín Ortiz

http://www.cs.buap.mx/~mmartin/notas/pdi/PDI-Cap2.pdf

Última visita 27 de octubre de 2012

[3L] Introducción a NetBeans Platform, Alejandro Morales Meza
http://www.ingeniero.moralesm.name/2010/02/25/introduccion-a-netbeans-platform/#punto2
Última visita 29 de octubre de 2012

[4L] Introducción a NetBeans Platform, Alejandro Morales Meza
http://www.ingeniero.moralesm.name/2010/02/25/introduccion-a-netbeans-platform/#punto1

Última visita 29 de octubre de 2012

[5L] Portable Network Graphics

http://es.wikipedia.org/wiki/Portable_Network_Graphics

Última visita 25 de enero de 2013

[6L] Apache Derby

http://es.wikipedia.org/wiki/Apache_Derby

Última visita 25 de enero de 2013

[7L] Visual Library

http://platform.netbeans.org/graph/

Última visita 25 de enero de 2013

[8L] Ribbon Bar

http://platform.netbeans.org/tutorials/nbm-ribbonbar.html

Última visita 25 de enero de 2013

MATERIAL BIBLIOGRÁFICO

[1] Peihua Qiu

Image Processing and Jump Regression Analysis

Wiley, 1965

[2] Heiko Böck

The Definite Guide to NetBeans Platform 7

Apress, 28 de Diciembre de 2011

[3] Alejandro Ramírez

Subversion

Documento publicado bajo Creative Commons Attribution License, marzo de 2004

[4] William Nagel

Using the Subversion Version Control System in Development Projects

Prentice Hall, 2005

[5] Martin Rinehart

Desarrollo de Bases de Datos en Java

McGraw-Hill, Primera edición 1998

[6] Abbey Corey

Oracle 8 Guía de Aprendizaje

McGraw-Hill, Oracle Press, 1998

[7] Joseph Schumuller

Sams Teach Yourself UML in 24 Hours

Sams, Tercera edición, 2004

[8] Elliote Rusty Harold, W. Scott Means

XML in a nutshell

O'Reailly, Tercera edición, Septiembre 2004

[9] Ethan Cerami

Web Services Essentials

O'Reilly, Primera edición, Febrero 2002

[10] Stephen Potts, Mike Kopack

Sams Teach Yourself Web Services in 24 Hours

Sams Publishing, 2003