Universitat Autònoma
de Barcelona

EDITOR DE IMÁGENES POR CAPAS Y CON ANOTACIONES

Memoria del proyecto de
Ingeniería Técnica en
Informática de Sistemas
realizado por

Víctor Manuel García Ortega

y dirigido por

Jordi Pons Aróztegui.

Escola d'Enginyeria

Sabadell, julio de 2011

Jordi Pons Aróztegui,
profesor de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Víctor Manuel García Ortega**

Y para que conste firma la presente.
Sabadell, **julio** de **2011**

Firmado: **Jordi Pons Aróztegui**

Ezequiel Parra Mestre, de la empresa UNIT4,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Víctor Manuel García Ortega**

Y para que conste firma la presente.
Sabadell, **julio** de **2011**

Firmado: **Ezequiel Parra Mestre**

HOJA DE RESUMEN – PROYECTO FIN DE CARRERA DE L'ESCOLA D'ENGINYERIA

Titulo del proyecto: Editor de imágenes por capas y con anotaciones	
Autor: Víctor Manuel García Ortega	**Fecha:** *Julio de 2011*
Tutor: Jordi Pons Aróztegui	
Titulación: Ingeniería Técnica en Informática de Sistemas	

Palabras clave

- Castellano: imágenes, editor, visor.
- Catalán: imatges, editor, visor.
- Inglés: images, editor, viewer.

Resumen del proyecto

- Castellano:

El proyecto se desarrolla en marco de un convenio de colaboración entre la Universitat Autònoma de Barcelona y la empresa UNIT4 en las oficinas de UNIT4 Ibérica en Barberà del Vallès. El proyecto tiene por finalidad crear un control que permita la edición de imágenes mediante capas y anotaciones, que se añadirá al software de gestión empresarial que comercializa UNIT4. Este control está desarrollado en la plataforma Java.

- Catalán:

El projecte es desenvolupa en marc d'un conveni de col·laboració entre la Universitat Autònoma de Barcelona i l'empresa UNIT4 en les oficines de UNIT4 Ibèrica a Barberà del Vallès. El projecte té per finalitat crear un control que permeti l'edició d'imatges mitjançant capes i anotacions, que s'afegirà al programari de gestió empresarial que comercialitza UNIT4. Aquest control està desenvolupat en la plataforma Java.

- Inglés:

The project is developed under a collaboration agreement between the Universitat Autònoma of Barcelona and the company UNIT4 at the UNIT4 offices in Barbera del Valles. The project aims to create a control that allows editing of images using layers and annotations, the project will be added to the business software of UNIT4. This control is developed in Java platform.

TABLA DE CONTENIDOS

INTRODUCCIÓN

ESTUDIO DE VIABILIDAD

ANÁLISIS

IMPLEMENTACIÓN

PRUEBAS

CONCLUSIONES

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

1.1. MARCO DE TRABAJO

1.1.1. *Convenio de colaboración entre UNIT 4 y la UAB*

Este proyecto se ha desarrollado dentro del marco de un convenio entre la Universitat Autònoma de Barcelona y la empresa UNIT4, en el cual, se estipulan 560 horas para realizar un trabajo de desarrollo de software. Mediante este convenio, el alumno puede adquirir conocimientos de cómo funciona una empresa de desarrollo de software de gestión y de tecnologías de la información, a la vez que este desarrollo sirve como trabajo de final de carrera.

1.1.2. *La empresa*

UNIT 4 tiene una experiencia durante más de cuarenta años en el mundo de la ingeniería de software. Posee un amplio abanico de soluciones de gestión de última generación lo cual la hace ser una de las primeras empresas españolas en tecnologías de la información y comunicaciones.

UNIT4 dispone actualmente de 4.230 trabajadores y cuenta con oficinas y distribuidores en todo el mundo para garantizar un acceso fácil y local a las ventas, servicios y soporte. Se encuentran en: Alemania, Australia, Bélgica, Canadá, Dinamarca, España, Estados Unidos, Estonia, Francia, Holanda, Hungría, Irlanda, Malasia, Noruega, Portugal, Reino Unido, República Checa, Singapur, Suráfrica, Suecia y Uganda.

Mediante la oferta de soluciones y objetivos, su misión es la de proporcionar una auténtica ventaja competitiva y diferenciadora a sus clientes mediante la implantación de soluciones informáticas de gestión.

Las soluciones de UNIT4 ofrecen un apoyo funcional de negocio tanto genérico como especializado, y benefician a todos los tipos de organizaciones del sector público y privado en el mundo entero. Además, algunas de ellas están específicamente centradas en las necesidades de sectores de mercado concretos.

Como principales objetivos tienen mantener su posición de liderazgo como proveedor global de soluciones y servicios TIC y estar por delante en investigación de nuevas tecnologías y desarrollo de nuevas soluciones.

Por los requisitos del proyecto este se sitúa dentro del departamento de *"Research & Development"* que es el grupo que se encargar del desarrollo y el soporte de la herramienta karat una completa plataforma tecnológica para la gestión de las empresas, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos. Este departamento además se ocupa del desarrollo de nuevas aplicaciones y mantenimiento de las ya existentes.

1.2. OBJETIVOS DEL PROYECTO

El proyecto tiene por objetivo crear una aplicación en la plataforma Java que consistirá en un control que permita la edición de imágenes mediante capas y anotaciones, este control se añadirá al software de gestión empresarial que comercializa UNIT4.

· A nivel personal:

- Adquirir experiencia en el trabajo diario dentro de una gran empresa y aprender metodologías de trabajo.
- Aprender Java, un lenguaje de programación orientado a objetos con un gran uso en el mundo laboral.
- Desarrollar una herramienta desde cero, donde poder realizar un ciclo completo en el desarrollo de software (investigación, análisis, codificación y pruebas).

· En el ámbito de la empresa UNIT4:

- Desarrollar una aplicación que permita la edición de imágenes mediante capas y anotaciones.
- Adaptar las funcionalidades para que sean sencillas y útiles, los usuarios no tienen que verse desbordados por las posibilidades de la herramienta.
- Debe incorporar todas las herramientas necesarias para que el usuario tenga una experiencia satisfactoria.

1.3. CONTENIDO DE LA MEMORIA

Seguidamente, se describirán los puntos de los cuales constará la memoria del proyecto:

· **Introducción**: Apartado en el cual se explica de forma breve el marco de ubicación del proyecto, así como los objetivos marcados.
· **Estudio de viabilidad**: Se hará una pequeña descripción del sistema actual, del sistema propuesto (objetos, funcionalidades, etc.), y de los beneficios y riesgos del proyecto.
· **Planificación del proyecto**: En este capítulo se podrá observar la planificación inicial de las tareas a realizar en el proyecto con su duración.
También se explicará el método de desarrollo de software utilizado.
· **Análisis**: Se profundizará en lo especificado en el capítulo anterior y se especificarán sus componentes en los dos niveles, así como una descripción de los procesos involucrados.
· **Implementación**: Apartado en el cual se describirá la tecnología utilizada en el desarrollo, las funcionalidades del software y estilo de codificación usado.
· **Pruebas**: Se describen en este apartado todas las pruebas realizadas sobre la aplicación y el resultado de éstas para comprobar la eficacia del programa.
· **Conclusiones**. Se redactarán los objetivos conseguidos, las líneas de trabajo abiertas así como una valoración personal sobre el proyecto.
· **Bibliografía**: Detalle de recursos informativos utilizados para realizar el aplicativo.

ESTUDIO DE VIABILIDAD

2.1. INTRODUCCIÓN

2.1.1. *Descripción*

Unit4 dispone de *karat*, una completa plataforma tecnológica para la gestión de las empresas, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos.

UNIT4 ha desarrollado una nueva interfaz de usuario denominada Walnut, tiene como lema "User experience". Con el fin de optimizar el trabajo cotidiano con la aplicación, centra la innovación en aprovechar la experiencia de los usuarios. Además es mucho más dinámica, personalizable y potencia la operatividad.

Desarrollada en Java, es independiente del entorno TI del cliente (Linux, Unix, Apple, Windows, etc.), no precisa de explorador para trabajar en web, puede usarse en cualquier dispositivo (PC, PDA, teléfonos móviles, etc.) y soporta todo tipo de idiomas y alfabetos (Unicode).

Algunos productos, especialmente aquellos orientados a entornos sanitarios, solicitan un control para la visualización de imágenes que, además de permitir incrustar imágenes y anotaciones, permita la edición de dichas imágenes al estilo de un editor sencillo de imágenes.

El objetivo que se persigue ahora es disponer de un nuevo control que incorpore la gestión de edición de imágenes mediante capas y anotaciones, además nuevas funcionalidades de inclusión de imágenes desde galerías, adaptándolo a walnut.

2.1.2. *Objetivos del proyecto*

A continuación detallamos los objeticos propuestos para el proyecto así como su clasificación según su importancia en tres categorías: críticos, prioritarios y secundarios.

1. Posibilidad de editar una imagen ya existente sin alterarla indefinidamente.
2. Desarrollo de un visor de las imágenes con las modificaciones y anotaciones realizadas.
3. Creación de un sistema de capas y grupos para la edición y recuperación.
4. Incluir herramientas básicas de edición de imágenes.
5. Incluir filtros para imágenes.
6. Importar proyectos mediante documentos XML.
7. Posibilidad de incluir imágenes prediseñadas desde archivo.
8. Permitir poder incluir anotaciones en las modificaciones de la imagen.
9. Añadir herramientas adicionales en la aplicación.

	Crítico	Prioritario	Secundario
O1	×		
O2	×		
O3	×		
O4	×		
O5		×	
O6		×	
O7		×	
O8	×		
O9			×

Tabla 1: Tabla de clasificación de objetivos.

2.1.3. *Partes interesadas*

A continuación presentemos las partes interesadas del proyecto.

Nombre	Descripción	Responsabilidad
Dept. "*Research & Development*" UNIT4	Departamento de la empresa UNIT4 dedicado a la parte de desarrollo y mantenimiento del software.	Departamento encargado de desarrollar y mantener nuevos productos de software.

Tabla 2: Tabla de las partes interesadas.

2.1.4. *Equipo de proyecto*

En la figura siguiente se muestra como queda confeccionado el equipo del proyecto y una pequeña explicación de la responsabilidad de cada uno.

Nombre	Descripción	Responsabilidad
Jordi Pons Aróztegui	Tutor del Proyecto(DP)	Supervisa el trabajo del alumno durante el proyecto.
Ezequiel Parra Mestre	Jefe de Proyecto UNIT4 (CP)	Define, gestiona y controla el proyecto.
Víctor Manuel García Ortega	Analista (A)	Desarrolla el estudio de viabilidad y la planificación. Análisis de la aplicación: arquitectura, metodología, especificación, estándares... Participa en el diseño y validación.

Francesc Martínez Morlanes	Analista (A)	Asesoramiento en el análisis de la aplicación y asesoramiento técnico a lo largo del proyecto.
Víctor Manuel García Ortega	Programador (P)	Desarrolla la aplicación de acuerdo al análisis y la planificación prevista. Hace la implantación del proyecto.
Víctor Manuel García Ortega	Diseñador (D)	Diseña y desarrolla el aspecto de la aplicación de acuerdo al análisis y normas de usabilidad establecidas.

Tabla 3: Tabla del equipo del proyecto.

2.2. ESTUDIO DE LA SITUACIÓN ACTUAL

2.2.1. *Contexto*

Actualmente en el producto karat dispone de un control con soporte para inclusión de imágenes prediseñadas. Se trata de un control simple donde es posible añadir imágenes prediseñadas arrastrándolas a una imagen estática no modificable.

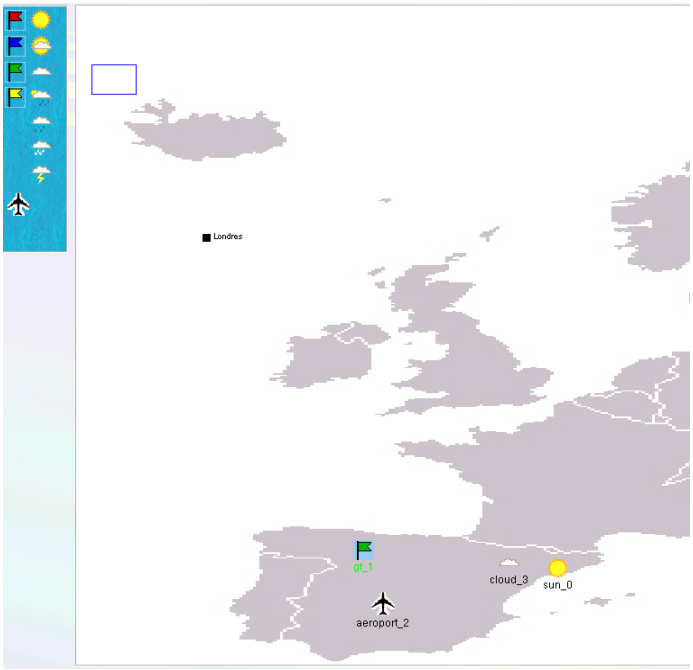

Figura 1: Contexto actual del sistema.

2.2.2. *Diagnóstico del sistema actual*

Aunque el control fue útil en su creación, actualmente no satisface todas las necesidades que se han solicitado y dispone de muchas carencias.

Carencias:

- Sistema sin capas.
- No es posible añadir anotaciones.
- No es posible la aplicación de dibujo sobre la imagen.
- Imagen predeterminada como fondo y no modificable.
- Falta de herramientas necesarias para la edición de imágenes.

2.3. REQUISITOS DEL PROYECTO

2.3.1. *Requisitos funcionales*

A continuación una lista de los requisitos funcionales que tendrá que cumplir el software que se va a crear.

RF1. Herramientas básicas de un editor de imágenes (creación de nuevo proyecto, modificar tamaño, zoom…)

RF2. Sistema de capas y grupos.

RF3. Creación de formas simples (línea, elipse, rectángulo).

RF4. Dibujo a mano alzada.

RF5. Posibilidad de añadir enumeraciones.

RF6. Posibilidad de añadir texto.

RF7. Posibilidad de añadir anotaciones en las modificaciones y visualizarlas en el visor.

RF8. Insertar imágenes prediseñadas.

RF9. Insertar imágenes de archivo.

RF10. Copiar/pegar, hacer/rehacer.

RF11. Filtros en la imagen (brillo / contraste)

RF12. Guardar proyecto y poder recuperarlo.

2.3.2. *Requisitos no funcionales*

A continuación los requisitos no funcionales que se han de cumplir.

RNF1. El diseño ha de seguir el estilo Walnut.

RFN2. El guardado de proyecto ha de ser en formato XML.

RFN3. Tolerancia de errores y acciones incorrectas.

RFN4. Facilitar el trabajo al usuario mediante una interfaz amigable.

RFN5. El software ha de permitir ser ampliable.

2.3.3. *Restricciones del sistema*

1. Utilización de software libre.
2. Seguir los estándares de UNIT4.
3. El proyecto ha de estar finalizado antes del 30 de junio de 2011.

2.3.4. *Catalogación y priorización de los requisitos*

#Req	Esencial	Condicional	Opcional
RF1	×		
RF2	×		
RF3	×		
RF4		×	
RF5			×
RF6	×		
RF7	×		
RF8		×	
RF9			×
RF10		×	
RF11		×	
RF12	×		
RNF1		×	
RNF2		×	
RNF3		×	
RNF4			×
RNF5		×	

Tabla 4: Tabla de catalogación y priorización de los requisitos.

2.4. ALTERNATIVAS Y SELECCIÓN DE LA SOLUCIÓN

2.4.1. *Alternativa 1*

Utilizar un software ya creado para la edición de imágenes.

En este caso hemos seleccionado el más famoso de todos, adobe Photoshop que se adapta en gran medida a las especificaciones.

Se trata de un producto muy completo pero a su vez muy caro. Cada licencia en España ronda los 1500€. Si se ha de poner a todos los usuarios que solicitasen el editor de imágenes este software supondría un incremento enorme en la licencia de karat.

Figura 2: Logotipo de Adobe Photoshop.

Además este software dispone de varias herramientas que no son necesarias para el usuario y su uso es bastante complejo si no se dispone de conocimientos previos en la edición de imágenes.

Tampoco se ajusta al guardado de proyecto interactivo mediante la plataforma karat.

2.4.2. *Alternativa 2*

Buscar un programa con la licencia Open Source[1] y adaptarlo a nuestras necesidades.

En este caso el coste de adquisición sería de 0€ y se facilitaría el trabajo del desarrollador.

El software ha de ser en la plataforma Java y ha de disponer de un mínimo de requisitos para que el trabajo de entender el software sea beneficioso.

Figura 3: Logotipo de Open Source.

2.4.3. *Alternativa 3*

Desarrollar un programa desde 0, adaptándolo a todas las necesidades y cubriendo todos los requisitos que se nos indican.

En este caso el coste de adquisición también sería 0€, su implementación implica un mayor trabajo por parte del desarrollador pero asegura que todos los requisitos serán satisfechos.

[1] es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

2.4.4. *Solución propuesta*

En la siguiente comparativa se ponen en común los diferentes aspectos más relevantes y la solución más óptima.

	Coste adquisición	Coste adaptación	Coste exploración	Coste desarrollo	Ajuste requerimientos
Alternativa1	1500€ por licencia	-	-	-	bajo
Alternativa2	0€	bajo	alto	medio	medio
Alternativa3	0€	bajo	bajo	alto	alto

Tabla 5: Tabla de comparativa de alternativas.

Analizando las diferentes alternativas, la **Alternativa1** queda totalmente descartada debido a su gran coste.

Ha sido necesario realizar una investigación previa para poder evaluar la **Alternativa2**, en esta investigación se han encontrado varios programas Open Source para intentar adaptarlo a nuestras necesidades, a continuación se nombran los que destacaban sobre los demás y estaban programados en Java:

- ImageJ
- Tudor DICOM
- Osirix Viewer

Por desgracia ninguno de los programas que se han encontrado satisface todos los requisitos mínimos. En el caso de ImageJ no era posible su integración y adaptación, Tudor DICOM debido a su complejidad nos obligaba a un tiempo demasiado extenso en adaptación y finalmente Osirix Viewer no cumplía los requisitos mínimos además aportaba funcionalidades que no eran necesarias.

Después de realizar esta investigación se ha descartado totalmente la Alternativa 2 ya que si se intentara utilizar alguno de esos programas acarrearía más problemas que beneficios.

Por lo mencionado arriba hemos escogido realizar la **Alternativa3**, que aunque supone un tiempo más elevado de desarrollo se ajusta perfectamente a las necesidades y requerimientos que se nos indican.

Al escoger la Alternativa 3 se suprimirán todos los problemas y costes de adaptación, además nos permite una seguridad en cuanto a ampliaciones futuras y problemas imprevistos en la codificación ya que todo el código se creará desde cero.

2.5. PLANIFICACIÓN DEL PROYECTO

2.5.1. *Recursos del proyecto*

Aquí se presentará todo lo referente a la planificación del proyecto y también de qué recursos se disponen para realizarlo.

El coste de los recursos humanos es aproximado, ya que por este proyecto el alumno recibirá un total de 2.352€. El coste ficticio del proyecto supondría alrededor de unos 25.000€, como se detalla en el apartado 2.6 de presupuesto.

- Recursos humanos:

Función	Coste/h
Jefe de proyecto	100€/h
Analista	50€/h
Programador	30€/h
Técnico de pruebas	20€/h

Tabla 6: Tabla de recursos humanos del proyecto.

- Recursos materiales:

  · Licencia Karat.
  · Equipo:
    PC – Intel Core 2Q8400 @ 2.66GHz
    6,00 GB RAM
    Disco duro 500GB
    Conexión a internet
  · Software:
    Windows 7
    Eclipse
    Editor de archivos XML

2.5.2. *Planificación temporal*

El proyecto se desarrollará de Noviembre de 2010 hasta Junio de 2011, con una dedicación de aproximadamente 20h semanales y un total de 560h.

Fecha de inicio: 8 de noviembre de 2010.

Fecha de finalización: 9 de junio de 2011.

Este calendario puede variar, ya que está planificado para hacer una jornada de 4 horas diarias, si se decide ampliar la jornada laboral o reducirla.

Todas las fases del proyecto se desarrollan mediante un modelo lineal. Por lo tanto, cada fase no comienza hasta que no se ha terminado la fase anterior.

En la fase de desarrollo, se ha decidido utilizar un modelo modular. La aplicación se ha dividido en diferentes funciones (módulos), y hasta que no se haya acabado el modulo actual y se haya paso dl fase de pruebas, no se seguirá con el siguiente.

La fase de documentación se realizará en la parte final del proyecto.

A continuación explicamos mediante esta tabla las diferentes tareas que componen el proyecto así como su duración en horas.

	Nombre de la tarea	Duración	Predecesoras
1	**Editor de imágenes por capas y con anotaciones**	**154 días**	
2	**Curso de Formación**	10 días	
3	**Planificación**	**6 días**	
4	Estudio de viabilidad	2 días	
5	Aprobación Estudio de Viabilidad	1 día	4
6	Plan del Proyecto	3 días	4
7	Aprobación Plan de Proyecto	1 día	6
8	**Análisis de la aplicación**	**14 días**	
9	Reunión con departamento de I+D	1 día	3
10	Análisis de requisitos	2 días	9
11	Investigación de software Open Source y competencia	7 días	10
12	Documentación del Análisis	3 días	11
13	Aprobación del Análisis	1 día	12
14	**Diseño de la Aplicación**	**39 días**	13
15	Diseño modular de la Aplicación	30 días	
16	Diseño de la interfaz gráfica	10 días	
17	Diseño de los test de pruebas	5 días	15
18	Documentación del diseño	3 días	17
19	Aprobación de diseño	1 día	18
20	**Desarrollo de la Aplicación**	**67 días**	14
21	Preparación del entorno de desarrollo	1 día	
22	Configuración de librerías	1 día	21
23	Desarrollo de las clases	65 días	22
24	Desarrollo de la interfaz gráfica	20 días	22
25	**Test y pruebas**	**17 días**	20
26	Pruebas unitarias	5 días	
27	Pruebas de integración	2 días	26
28	Pruebas de estrés	2 días	27
29	Documentación de las pruebas	2 días	28
30	Aprobación de las pruebas y resultados	1 día	29
31	Corrección de errores	5 días	30

32	**Implantación**	**4 días**	25
33	Instalación en karat	2 días	
34	Formación de usuarios	2 días	33
35	**Generación de la Documentación**	**5 días**	32
36	**Cierre del proyecto**	**1 día**	35
37	**Defensa del Proyecto**	**1 día**	36

Tabla 7: Tareas del proyecto.

A continuación se muestra el diagrama de Gantt de la planificación del proyecto con las tareas numeradas para facilitar su comprensión.

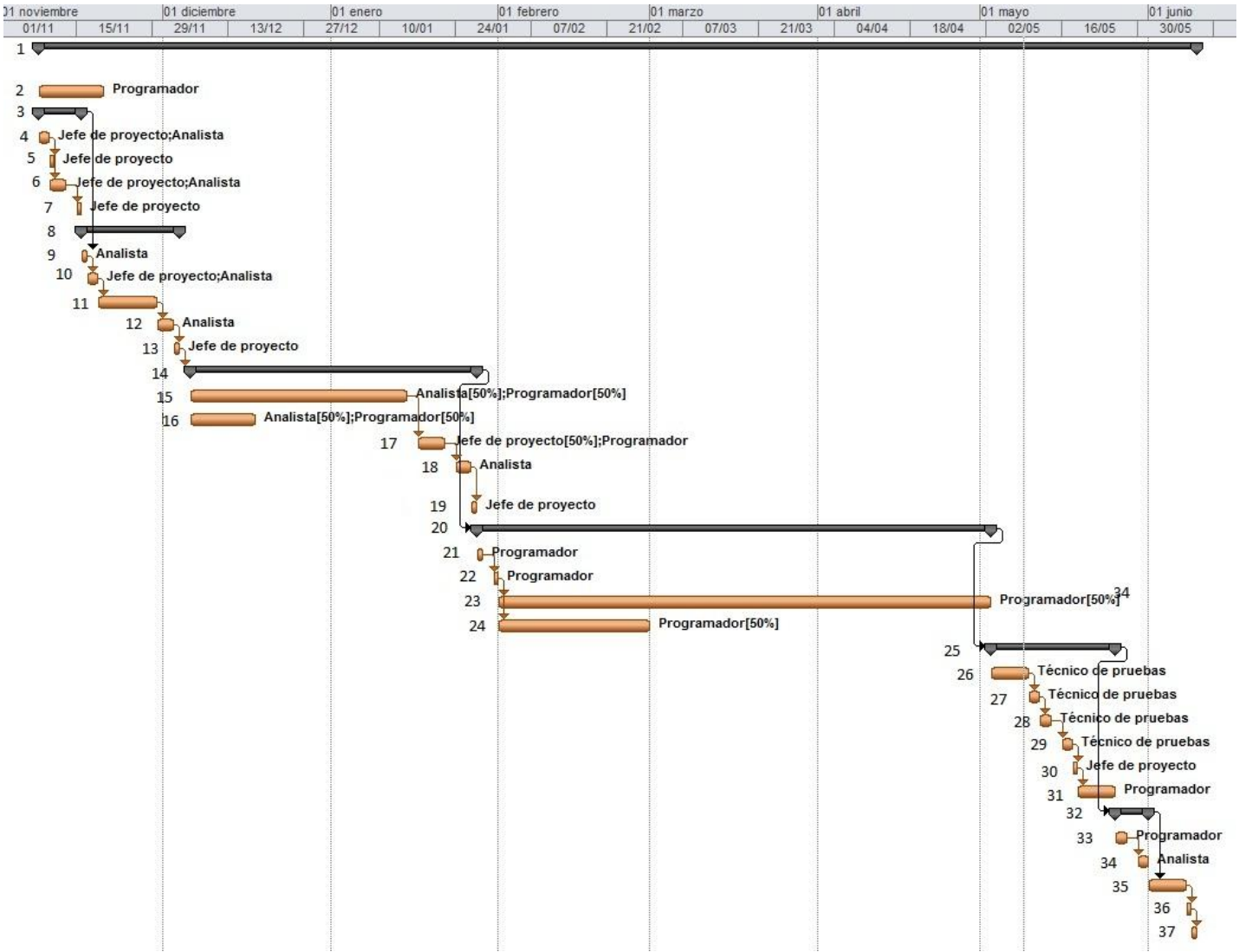

Figura 4: Diagrama de Gantt.

2.6. PRESUPUESTO

2.6.1. *Estimación coste de personal*

A continuación se enumeran los costes aproximados de personal previstos.

Función	Coste/h	Horas	Coste total
Jefe de proyecto	100€/h	80	8000 €
Analista	50€/h	120	6000 €
Programador	30€/h	320	9600 €
Técnico de pruebas	20€/h	40	800 €
TOTAL		560	24400 €

Tabla 8: Tabla con los costes estimados de personal.

2.6.2. *Estimación coste de los recursos*

La siguiente tabla muestra los costes de los recursos necesarios para desarrollar el proyecto.

Recurso	Coste
PC	600€

Tabla 9: Tabla con los costes estimados de los recursos.

2.6.3. *Resumen y análisis coste beneficio*

Detalle de coste:

Coste personal……..	24.400
Coste recursos……	600
TOTAL……..	--------- 25.000 €

Tabla 10: Resumen y análisis coste beneficio.

El proyecto tiene un coste razonable si miramos las licencias de los productos actuales que se asemejan en el mercado, igualmente como hemos comentado antes esto es un coste ficticio, el coste real que supone son los 2.352€ que percibirá el alumno.

2.7. CONCLUSIONES

Para concluir el estudio de viabilidad se va a realizar una valoración de los beneficios y los inconvenientes que acarreará el proyecto.

2.7.1. *Beneficios esperados*

La realización de un proyecto informático es siempre una oportunidad para que el alumno pueda poner en práctica los conocimientos que ha ido adquiriendo a lo largo de la carrera. También para profundizar en aquellas materias específicas que hace referencia el proyecto.

Como beneficios que el proyecto aportará a la empresa UNIT4:

- ✓ Incorporar un editor de imágenes dentro de la herramienta karat, barato, fiable y totalmente adaptado.
- ✓ Mejorar la experiencia de uso de los usuarios de la herramienta karat.
- ✓ Abrir el mercado y poder hacer competencia con herramientas similares.

2.7.2. *Inconvenientes*

Como inconvenientes el único que podemos encontrar es la necesidad de un periodo de formación, ya que el alumno no dispone de los conocimientos necesarios para poder realizar la aplicación. Igualmente esto supone un gran beneficio didáctico para el alumno donde podrá ampliar sus conocimientos.

Una vez analizados los beneficios e inconvenientes se demuestra que el proyecto resulta totalmente viable.

Beneficios + Inconvenientes = Proyecto Viable

ANÁLISIS

3.1. PERFILES DE USUARIO

La aplicación solo es usada por un tipo de usuario.

PU1. Usuario

Se les permite poder crear nuevos proyectos, además pueden abrir y modificar proyectos existentes.

3.2. REQUISITOS FUNCIONALES

A continuación se detallaran los requisitos funcionales del software.

RF1. Herramientas básicas de un editor de imágenes.

El software ha de disponer de las herramientas básicas de un editor de imágenes, estas herramientas aparte de algunas que se explicarán en requisitos posteriores son las herramientas de configuración y manejo de proyectos que contiene normalmente un editor, como la creación de un proyecto o la posibilidad de ajustar tamaños.

Además para comodidad del usuario en la edición de las imágenes el editor ha de permitir realizar un Zoom sobre el proyecto.

RF2. Sistema de capas y grupos.

Ya que las imágenes que se necesita modificar pueden ser importantes, es necesario un sistema de modificación que sea reversible, por este motivo se ha de incorporar un sistema de capas y grupos.

Cada capa ha de permitir tanto su modificación individual como su eliminación, además se han de poder hacer visibles e invisibles y bloquearlas.

Los grupos de capas están formadas por capas individuales permitiendo así la comodidad de los usuarios. Cada usuario debe poder crear su propio grupo y realizar sus propias modificaciones sin que esto repercuta en el trabajo de otros.

Los grupos han de permitir hacer visibles e invisibles todas las capas que contengan y además bloquearlas si es necesario.

RF3. Creación de formas simples (línea, elipse, rectángulo).

El software dispondrá de herramientas simples de dibujo como líneas, elipses y rectángulos. Estas herramientas podrán ser modificadas en tamaño, color, opacidad y posición.

En el caso de la elipse y el rectángulo además se podrá rellenar y cambiar el color de relleno.

RF4. Dibujo a mano alzada.

También se permitirá la creación de dibujos a mano alzada, donde se podrá modificar su tamaño, color, relleno, opacidad y posición.

RF5. Posibilidad de añadir enumeraciones.

El software incluirá una herramienta que se considera muy útil para los usuarios, se trata de mediante la secuencia de clics con el ratón, ir incrementando una enumeración. Esto permitirá al usuario poder enumerar diferentes objetos o posiciones dentro de una imagen.

En las enumeraciones se podrá modificar su color, tamaño, posición, fuente y estilo.

RF6. Posibilidad de añadir texto.

El usuario ha de poder añadir texto dentro de la imagen, este texto ha de ser multilinea.

Se podrá modificar su color, tamaño, posición, fuente, estilo y el propio texto que se haya insertado anteriormente.

RF7. Posibilidad de añadir anotaciones en las modificaciones y visualizarlas en el visor.

Cualquier elemento que se inserte dentro de la imagen ha de permitir la inclusión de una anotación. Esta anotación será visible mediante el paso del ratón por encima del elemento.

En el caso de las enumeraciones, las anotaciones han de ser individuales por cada número, no como conjunto de la enumeración.

Estas anotaciones han de poderse ver en el visor.

RF8. Insertar imágenes prediseñadas.

Karat dispone de unas listas de sistema donde se incluyen imágenes prediseñadas que el usuario ha de poder insertar dentro de la imagen.

En las imágenes prediseñadas se ha de poder modificar su tamaño, posición y opacidad.

RF9. Insertar imágenes de archivo.

Se ha de permitir que el usuario pueda insertar imágenes desde cualquier archivo. Estas imágenes se insertarán dentro de la imagen de fondo.

En las imágenes de archivo se tiene que poder modificar su tamaño, posición, opacidad, brillo, contraste y permitir incorporar un borde con diferentes estilos y colores.

RF10. Copiar/pegar, hacer/rehacer.

El software permitirá la copia y pegado de capas, todo esto se podrá hacer mediante teclas rápidas de teclado.

También se podrán realizar acciones de marcha atrás y adelante para no tener que volver a empezar en una modificación errónea.

RF11. Filtros en la imagen (brillo / contraste).

Se han de poder realizar filtros en la imagen e fondo, el usuario podrá ajustar el brillo y el contraste de la imagen a su gusto para poderla visualizar mejor.

RF12. Guardar y recuperar proyecto.

Es necesario que el proyecto permita su guardado. Esto se realizará mediante un archivo XML[2], que contendrá toda la información necesaria para su posterior recuperación.

[2] ***eXtensible Markup Language*** ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

3.3. REQUISITOS NO FUNCIONALES

A continuación se muestran los requisitos no funcionales que tendrá que cumplir el software.

RNF1. El diseño ha de seguir el estilo Walnut.

Es necesario que el diseño de la interfaz de usuario cumpla con los estándares de Walnut. Esto es necesario porque el editor de imágenes estará insertado en Walnut y de esta manera todo tendrá un aspecto homogéneo.

RFN2. El guardado de proyecto ha de ser en formato XML.

Para comodidad en el trato de los elementos del editor es necesario que sean guardados en ficheros XML. Con esto conseguiremos una mayor simplificación del guardado y su posterior recuperación.

RFN3. Tolerancia de errores y acciones incorrectas.

La aplicación tendrá que estar desarrollada de forma que tendrá en cuenta que el usuario que la utilice puede cometer errores durante la entrada de datos, modificaciones en la imagen y realizar acciones de forma diferente a la prevista. Así pues, cualquier entrada de datos, secuencia de acciones, etc... tendrá que controlar las posibles entradas incorrectas permitiendo al usuario la corrección de las mismas e intentando minimizar la repetición de entrada de datos en los sucesivos intentos.

RFN4. Facilitar el trabajo al usuario mediante una interfaz amigable.

La interfaz del editor ha de ser comprensible para el usuario y ha de facilitar en todo momento su uso. Esto es necesario para que el usuario no se vea abrumado por las herramientas del editor y pueda realizar su trabajo de una forma cómoda y sencilla.

RFN5. El software ha de permitir ser ampliable.

La aplicación tiene que estar diseñada correctamente para permitir posibles modificaciones y ampliaciones en un futuro.

IMPLEMENTACIÓN

4.1. INTRODUCCIÓN

En el momento de desarrollar una aplicación es importante que anteriormente se haya hecho un buen diseño, esto facilitará su posterior codificación y comprensión de la estructura y funcionamiento.

Se estructurará el software mediante módulos independientes para conseguir un nivel de abstracción lógico y mayor comprensión, además se separará completamente el modelo de datos de la parte visual, para que modificaciones, actualizaciones o nuevos visores no afecten a nuestro modelo de datos.

4.2. SELECCIÓN DE TECNOLOGÍA DE DESARROLLO

Debido a que el software va a estar integrado dentro de la plataforma Karat es necesario que nuestro software esté completamente codificado en Java. Además, para mayor comodidad y facilidad para guardado, el modelo de datos se guardará mediante documentos XML.

4.2.1. *Java*

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Figura 5: Logotipo Java.

Uno de los principales motivos de la elección del lenguaje Java, aparte de su enorme potencial, es su independencia de plataforma con la cual nos aseguramos que nuestra aplicación va a funcionar perfectamente en todas las plataformas del mercado.

Java dispone de una excelente documentación de ayuda y una comunidad muy grande detrás, aspecto que nos facilita enormemente el trabajo.

4.2.2. *XML*

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información.

Para nuestro software la necesidad de guardar datos es realmente importante y es necesaria una manera simple y eficaz de realizarlo, mediante XML todo resulta mucho más fácil.

4.3. SELECCIÓN DEL ENTORNO DE DESARROLLO

Para poder realizar el software, en primer lugar se debe escoger un entorno de desarrollo integrado (IDE). Por su comodidad y ventajas se ha seleccionado eclipse Helios.

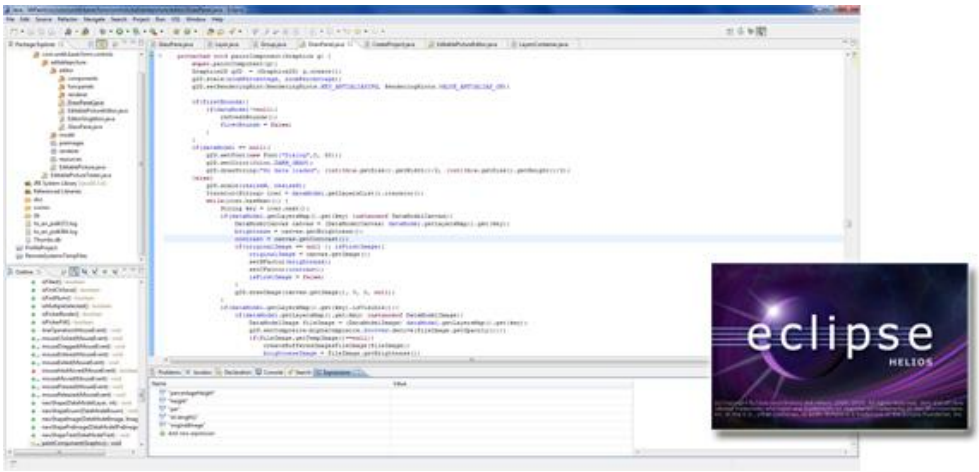

Figura 6: Eclipse Helios.

Eclipse es uno de los IDE más usados en todo el mundo. Se trata de un software con licencia OpenSource y es totalmente gratuito. Consta de infinidad de herramientas para hacer más cómodo el desarrollo de aplicaciones y la gestión de proyectos. Además dispone de una gran comunidad que actualiza constantemente utilidades y plug-ins[3] para mejor la experiencia de desarrollo.

[3] Un **plug-in** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica.

4.4. FUNCIONAMIENTO Y DISEÑO DE LA APLICACIÓN

La aplicación se ha diseñado de manera que sea comprensible y sea realmente fácil diferenciar los diferentes elementos. Principalmente consta de un visor, un editor y un modelo de datos.

El visor permite visualizar el proyecto que queremos editar. Este proyecto estará guardado mediante un fichero XML que contendrá todo el modelo de datos.

En el **modelo de datos** se guardarán todos los elementos de modificación de nuestra imagen, como pueden ser líneas, rectángulos, propiedades…

El editor permitirá realizar modificaciones sobre el proyecto abierto y actualizar el visor.

El funcionamiento de la aplicación es bastante simple. A partir del visor se abre un proyecto, se puede abrir mediante un fichero XML o crear uno nuevo, con lo cual el visor tendrá su propio modelo de datos.

Si queremos realizar una modificación del proyecto se deberá abrir el editor. Cuando se realiza esta acción se crea una copia del modelo de datos mediante otro fichero XML que se le pasa al editor. En este momento tendremos el visor con su modelo de datos y el editor con una copia del modelo de datos del visor, con esto conseguimos que si se realizan modificaciones sin guardar no afecten directamente al visor. Finalmente si guardamos el proyecto desde el editor se actualizará el modelo de datos del visor.

En la figura 7 vemos un pequeño esquema del funcionamiento para su mayor comprensión:

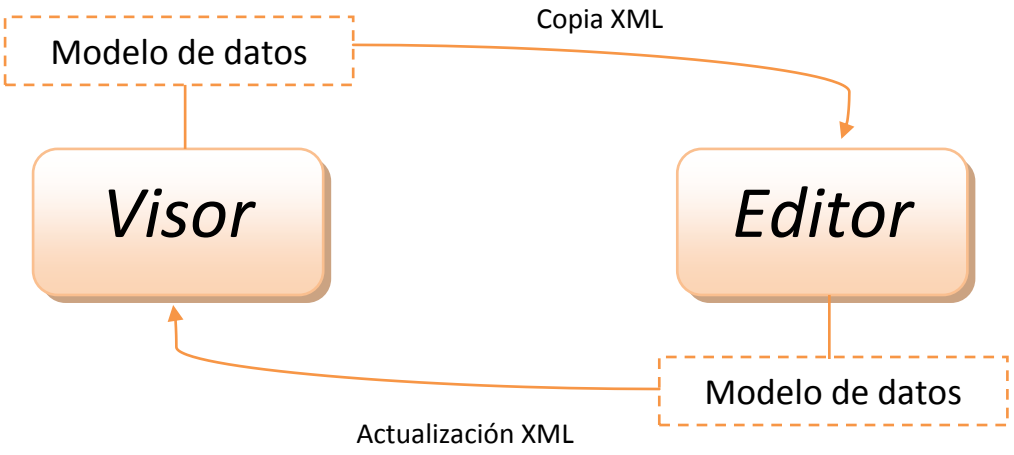

Figura 7: Esquema funcionamiento de la aplicación.

4.5. DISEÑO DE CLASES

Como se ha comentado anteriormente la aplicación tiene un diseño modular donde los módulos principales pueden dividirse en submódulos, los cuales a su vez se pueden también fraccionar.

En esta sección se muestran los diagramas de clases de las principales partes del programa divididas para su mayor comprensión.

El modelo de datos: en el modelo de datos es donde se guarda toda la información referente a las modificaciones del proyecto y se trata para importarlo/exportarlo mediante ficheros XML.

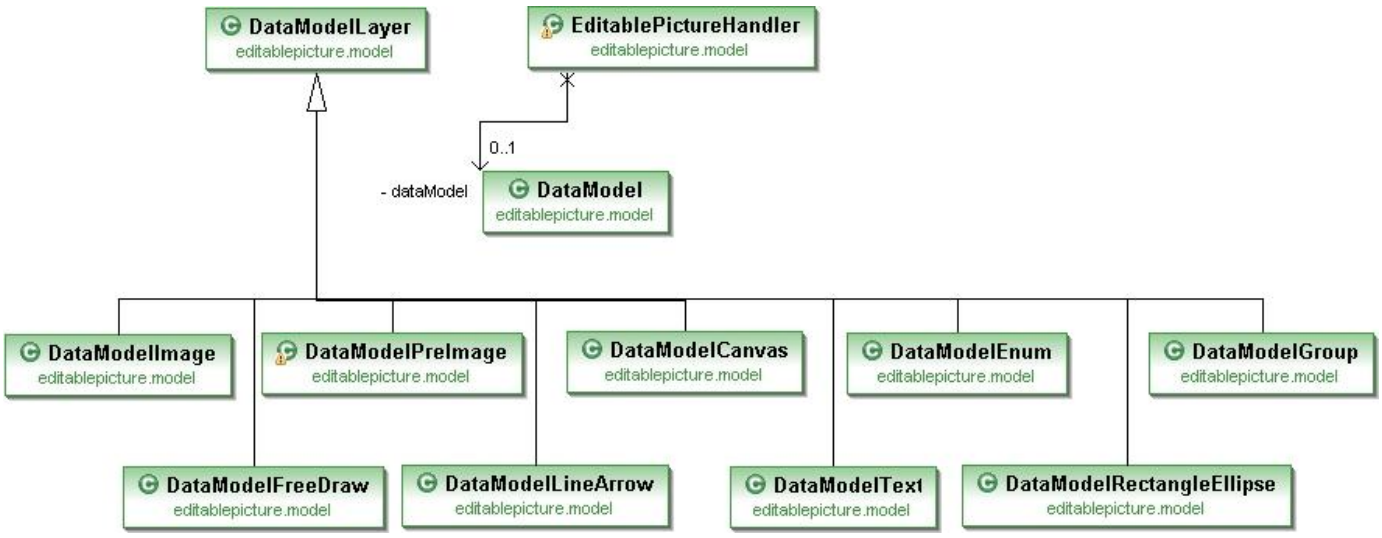

Figura 8: Diagrama de clases del modelo de datos.

En el diagrama podemos apreciar que nuestro modelo de datos se compone de *DataModelLayer,* esta es una clase abstracta, que posee los atributos comunes de las subclases *DataModelImage, DataModelFreeDraw…* que son los elementos que podemos introducir en el editor (líneas, rectángulos, trazo libre, imágenes…).

También se puede apreciar la clase *EditablePictureHandler,* que se ocupa de tratar los ficheros XML, tanto en pasar de modelo de datos a XML como a la inversa.

El visor: como se ha comentado en el apartado de funcionamiento la aplicación consta de un visor y un editor. Este visor muestra el proyecto y llama al editor en el momento que queramos realizar alguna modificación en la imagen mostrada.

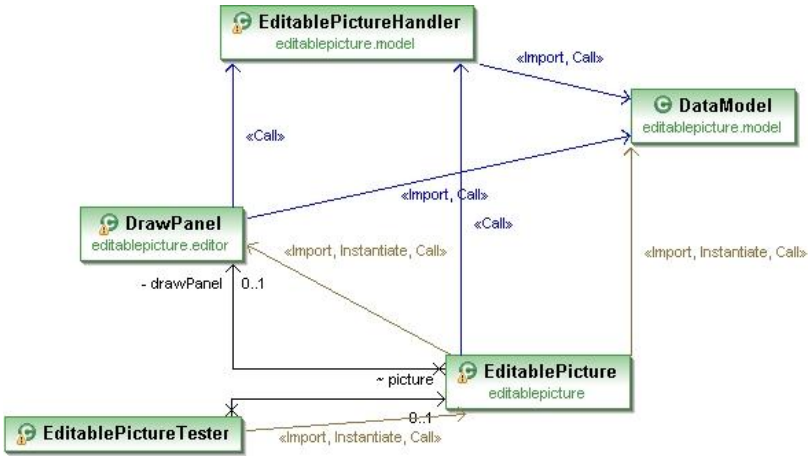

Figura 9: Diagrama de clases del visor.

El visor (*EditablePicture*) se compone de una clase *DrawPanel* que es donde se mostrará el proyecto, una interfaz *EditablePictureTester* y un modelo de datos que será usado cuando se importe un fichero XML.

El editor: el editor es el módulo de la aplicación donde se realizan todos los cambios en el proyecto. Debido a su gran número de módulos en la siguiente imagen sólo se han insertado los más importantes, pero cabe decir que por ejemplo los paneles de control de herramientas (*GeneralPanel, TextPanel, ImagePanel…*) están compuestos de varios submódulos para su funcionamiento.

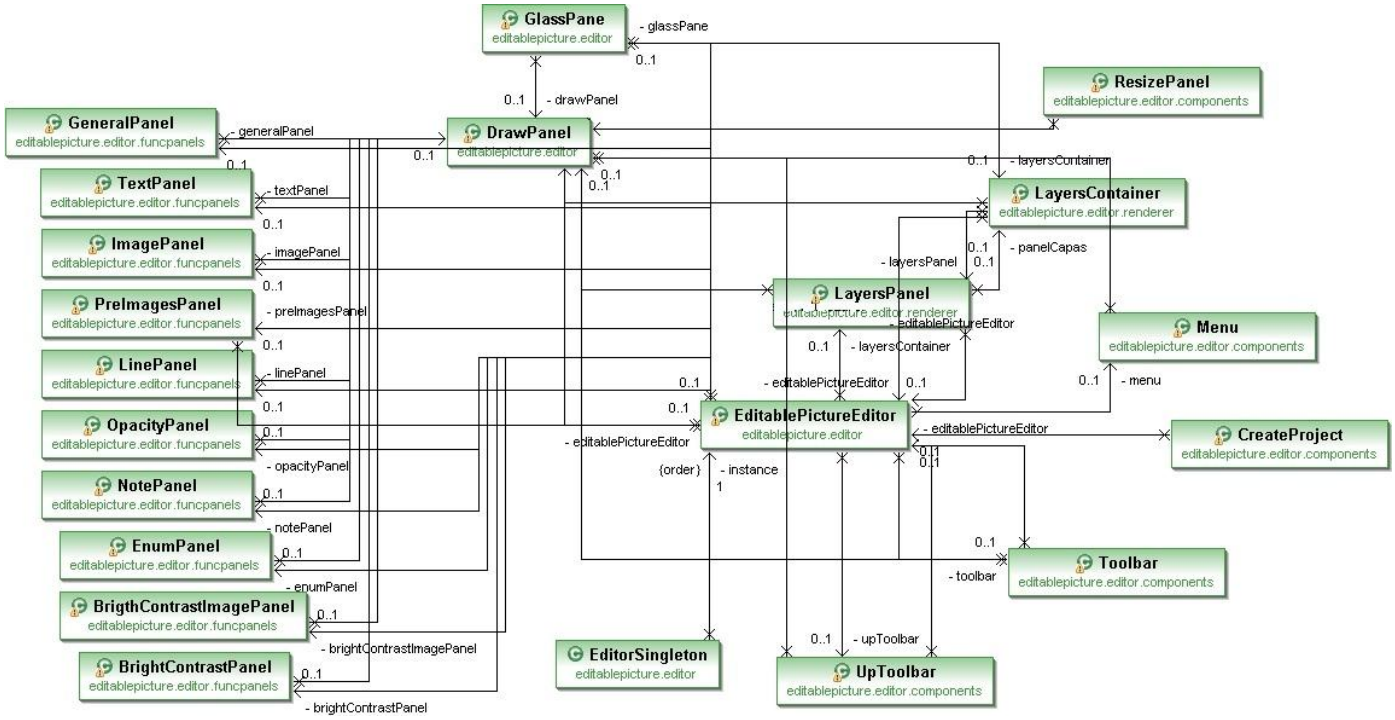

Figura 10: Diagrama de clases del editor.

Nuestro editor (*EditablePictureEditor*) está compuesto principalmente por una clase *Drawpanel* donde se harán las modificaciones en el proyecto, y por herramientas del editor.

Estas herramientas son las barras donde se encuentran las utilidades y las opciones de dibujo (*Menu, Toolbar, UpToolbar*), los paneles de control de nuestras herramientas (*GeneralPanel, TextPanel, ImagePanel, NotePanel…*), las pantallas de configuración y control del editor (*CreateProject, ResizePanel*) y el contenedor de nuestro sistema de capas *Layerspanel* que se comentará posteriormente.

Además también hay otras clases interesantes como la clase *EditorSingleton* que se ocupa de asegurar que solo se pueda abrir un editor del mismo proyecto o la clase *GlassPane* que añade funcionalidades visuales a nuestro editor.

Al editor se le añadirá una copia del modelo de datos desde el visor cuando el usuario decida realizar una modificación en el proyecto.

Sistema de capas: en el apartado de editor se ha hecho una mención a la clase *LayerPanel.* Esta clase es la que contiene nuestro sistema de capas que se explicará con más detalle en capítulos posteriores, pero a modo de introducción se muestra cómo están diseñadas las clases.

Cada elemento que se introduce en nuestro proyecto tiene una capa que lo controla, mediante la cual se puede seleccionar fácilmente el elemento y realizar los cambios que se deseen.

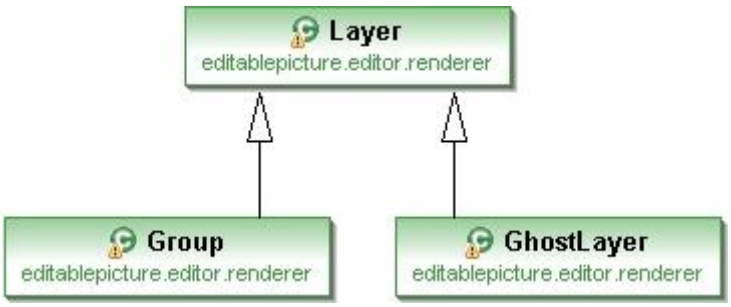

Figura 11: Diagrama de clases del sistema de capas.

El sistema de capas está compuesto por clases *Layer,* que es una capa simple de los elementos y también actúa como clase abstracta para las clases *Group* y *GhostLayer* que añaden funcionalidades al sistema de capas.

Estas clases, como se ha comentado antes, hacen referencia a los elementos introducidos en nuestro proyecto y tienen la necesidad de estar relacionados con el modelo de datos.

4.6. LIBRERÍAS EXTERNAS

Para el desarrollo de la aplicación se han necesitado varias librerías externas que han facilitado la codificación. A continuación se indicarán cuales han sido y una pequeña explicación de cada una:

- Karat-base: una de las librerías del programa Karat, que es usada para algunas funciones de codificación de imágenes.
- Karat-laf: otra de las librerías del programa Karat, que es usada para el aspecto visual de nuestra aplicación.
- JDom: librería necesaria para el tratamiento de ficheros XML.
- SwingX: librería que mejora y añade nuevas funcionalidades a la parte visual por defecto de Java.
- JForms: librería que permite la localización de los objetos en las pantallas de Java con mucha mayor facilidad y comodidad.

4.7. INTERFAZ DE USUARIO

Esta sección tiene como objetivo la definición de la interfaz de usuario de nuestra aplicación, que se ha diseñado de manera que cumpla con los estándares de Walnut y sea cómoda para el usuario.

A continuación se muestra un esbozo del editor, que es la parte donde hay más elementos y se debían organizar de manera que el usuario tuviese una experiencia satisfactoria en su uso.

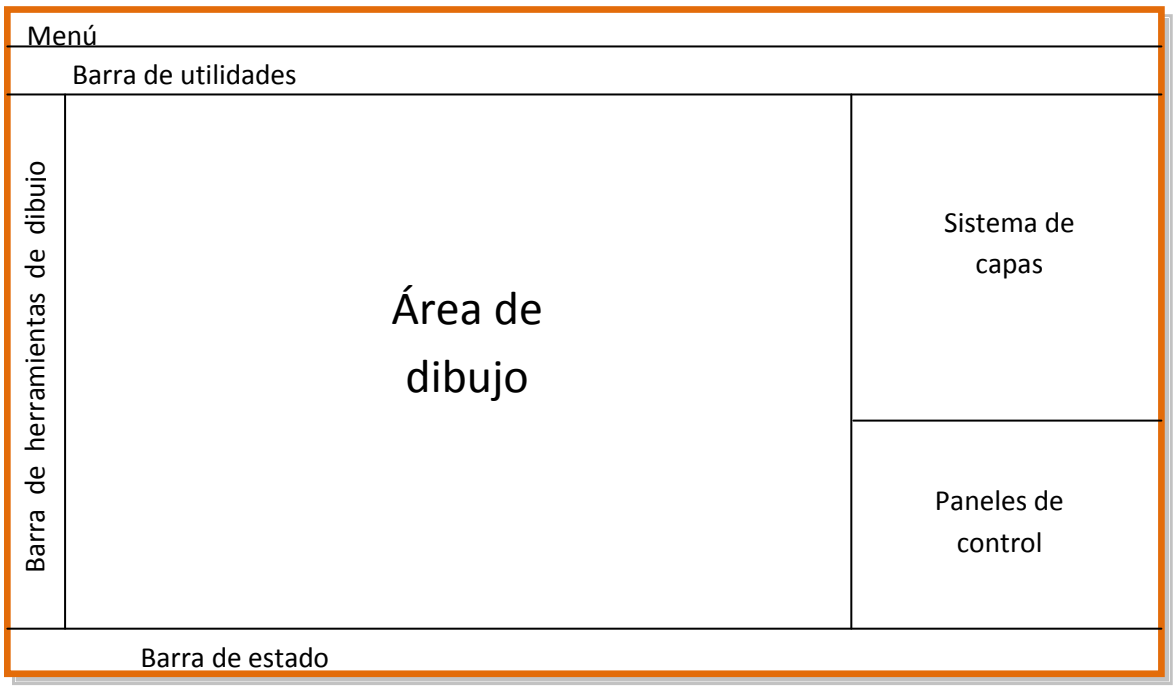

Figura 12: Esbozo del editor.

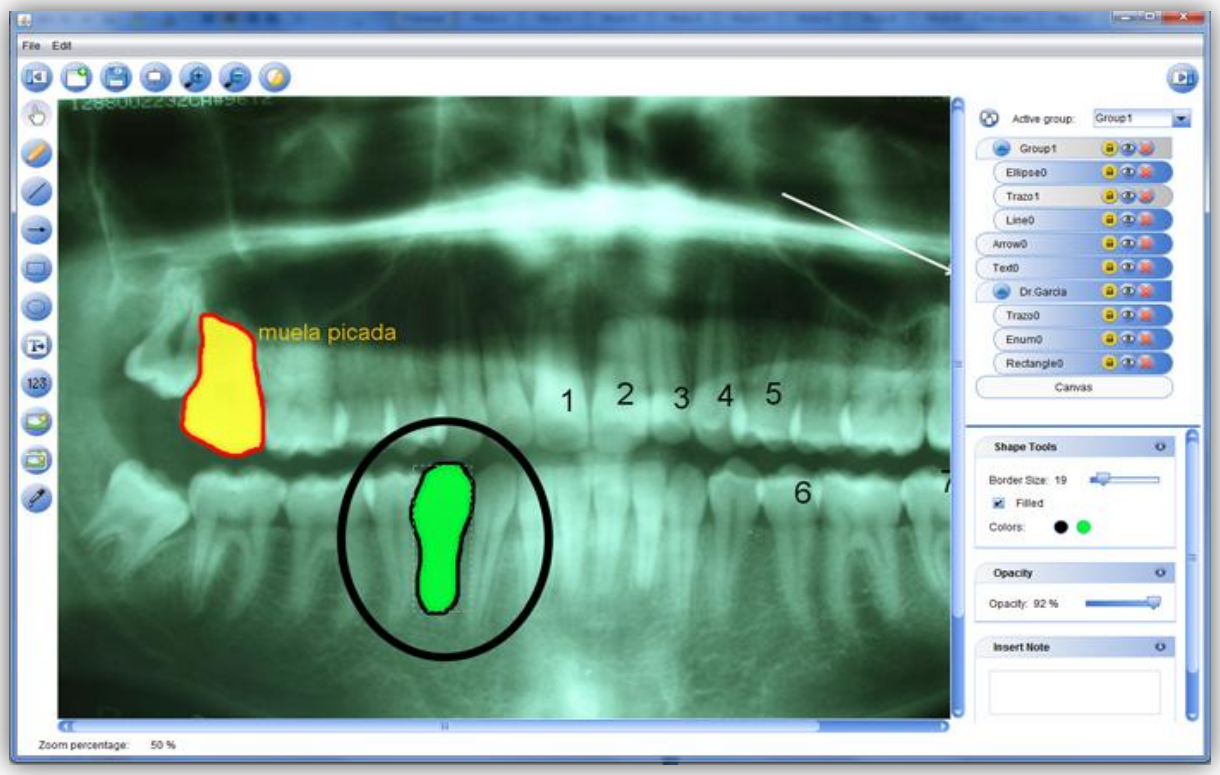

Figura 13: Imagen real del editor.

- Menú: donde encontraremos funciones de guardado de imagen a .JPEG[4] y teclas rápidas de editor.
- Barra de utilidades: en esta barra se muestran utilidades para la gestión del editor, tales como el salvado del proyecto, creación de uno nuevo, ajuste del tamaño, herramientas de zoom y filtros.
- Barra de herramientas de dibujo: aquí tendremos las herramientas de dibujo que podremos aplicar sobre nuestro proyecto.
- Área de dibujo: es la parte principal del editor. En esta área insertaremos nuestra imagen a modificar y aplicaremos todas las modificaciones pertinentes.
- Sistema de capas: esta zona incluye el sistema de capas que se explicará más detalladamente en capítulos posteriores.
- Paneles de control: en estos paneles podremos editar las propiedades de los elementos que introduzcamos en el proyecto, además de incluir notas.
- Barra de estado: en esta barra encontraremos información referente al editor, como el estado del zoom.

[4] **JPEG** (del inglés *Joint Photographic Experts Group*, Grupo Conjunto de Expertos en Fotografía), además de ser un método de compresión, es a menudo considerado como un formato de archivo. JPEG es el formato de imagen más común utilizado por las cámaras fotográficas digitales y otros dispositivos de captura de imagen.

Toda la interfaz está pensada para que el usuario se encuentre cómodo trabajando. El usuario podrá ajustar el marco de trabajo a su gusto para poder visualizar mejor todas las herramientas y no verse restringido por resoluciones bajas de pantalla.

A continuación detallamos las posibilidades de ajuste que ofrece la interfaz del editor:

La interfaz permite que se puedan ocultar ciertos paneles para mayor comodidad en el trabajo. Mediante los botones de ocultar que se encuentran en la barra de utilidades es posible hacer desaparecer y aparecer la barra de herramientas de dibujo, el sistema de capas y los paneles de herramientas.

Figura 14: Botones de ocultar.

Otra opción que permite la interfaz es ajustar el tamaño visual del sistema de capas y los paneles de control, esto se realiza mediante una barra horizontal que arrastrándola se posiciona en el lugar deseado.

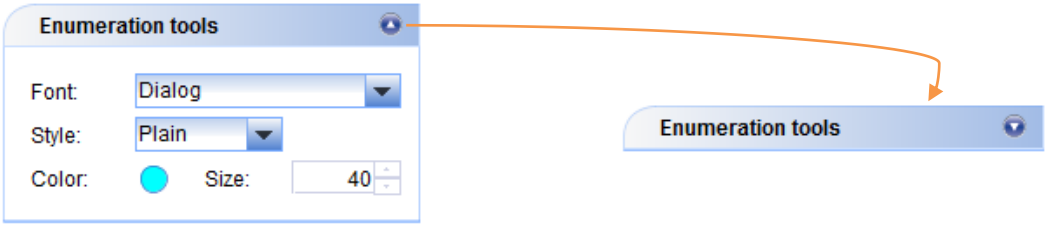

Figura 15: Minimización de un panel de control.

En los paneles de control es posible la minimización de cada uno por separado, de esta manera el usuario podrá escoger el panel con el que más trabaje, ocultando los demás para no ocupar espacio.

Finalmente, ya sea en el área de dibujo, sistema de capas o paneles de control, cuando sea necesario aparecerá una scrollbar[5] para permitir la visualización de todo el contenido ajustable en pantalla.

4.8. BARRA DE UTILIDADES

En este apartado se van a especificar todas las utilidades que incluyen esta barra de herramientas y la función de cada una.

[5] La barra de desplazamiento (scrollbar) es un elemento de las interfaces gráficas que constan de una barra horizontal o vertical con dos extremos con flechas que apuntan en sentidos contrarios y que suelen ubicarse en los extremos de una ventana o recuadro. Las barras de desplazamiento permiten desplazar el contenido del cuadro hacia un lado u otro. Las barras suelen aparecer o activarse cuando el recuadro no es lo suficientemente grande como para visualizar todo su contenido.

4.8.1. *Nuevo proyecto*

Como hemos comentado en el apartado de funcionamiento y diseño (4.4) el uso normal de la aplicación es abrir un nuevo proyecto desde el visor y luego abrir el editor, pero el editor permite poder reconvertir totalmente el proyecto que hemos abierto desde el editor para volver a empezar desde el inicio reemplazando el proyecto abierto en el visor.

Esto es útil si en una futura ampliación se decide separar completamente el visor del editor y no necesitar ese paso intermedio. También aporta una usabilidad para el usuario de que en cualquier momento puede cambiar la imagen del fondo y crear un nuevo proyecto sin necesidad de tener que volver al visor.

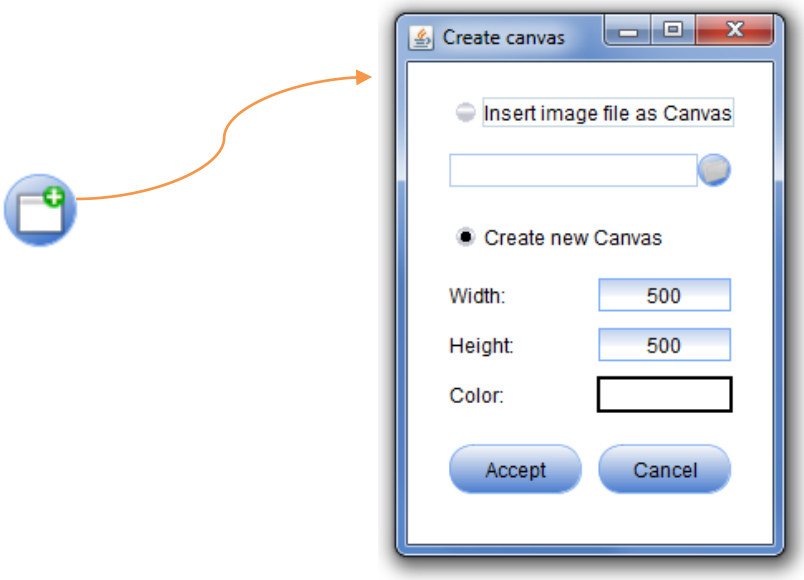

Figura 16: Botón y panel de nuevo proyecto.

Presionando el botón de nuevo proyecto nos aparece un panel donde se permitirá al usuario poder seleccionar las propiedades del nuevo proyecto.

- Insertar una imagen externa como fondo: con esta opción conseguimos insertar una imagen desde archivo como fondo para trabajar en su edición.
- Crear un nuevo fondo: es posible la creación de un nuevo fondo para trabajar sobre él. En este podremos seleccionar sus medidas y el color.

Una vez seleccionadas las propiedades simplemente se deberá pulsar el botón aceptar y ya se podrá empezar a usar el nuevo proyecto. Presionando el botón cancelar ninguna modificación surtirá efecto.

4.8.2. *Salvar*

Esta es una de las funciones más importantes del editor. Es la función que se ocupa de aplicar los cambios realizados en la imagen.

Figura 17: Botón Salvar.

Como hemos comentado en el apartado 4.4 una vez realizados los cambios en el editor, si se pulsa el botón de Salvar, se crea un archivo XML que será enviado al visor donde realizará una sustitución de su modelo de datos.

4.8.3. *Reajustar tamaño del proyecto.*

El editor permite que se pueda ajustar el tamaño del proyecto a gusto del usuario, este cambio afecta a todo el proyecto incluido las modificaciones.

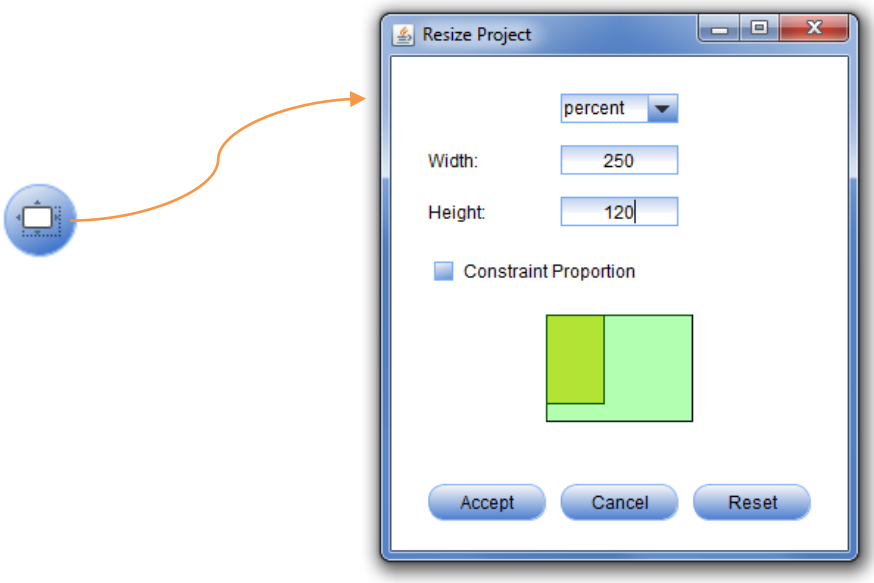

Figura 18: Botón y panel de reajustar tamaño del proyecto.

Al presionar el botón de reajuste de tamaño nos aparece un panel donde se pueden seleccionar las propiedades:

- Una caja de opciones donde se puede seleccionar si se quiere trabajar en porcentajes o en píxeles[6].
- Los tamaños: donde se ha de insertar el tamaño deseado de reajuste.
- Mantener proporciones: si se tiene seleccionada esta opción, al insertar un tamaño ya sea en altura o anchura, se mantendrán las proporciones en el otro.

[6] Píxel es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico.

En el momento que insertamos un tamaño automáticamente se actualiza un pequeño avance visual de cómo va a quedar nuestro proyecto. Esto facilita al usuario saber exactamente cuáles van a ser sus modificaciones.

Un apartado interesante es la opción de reset (restablecer). La aplicación está pensada para que profesionales editen sus imágenes y estas imágenes, en muchas ocasiones, son importantes (como por ejemplo una radiografía). Para el usuario no es bueno que se modifiquen realmente las imágenes por posibles pérdidas de calidad. Por este motivo en el editor la imagen no se modifica completamente una vez se ajusta el tamaño, sino que se guardan unos valores de reajuste que el visor y el editor comprenden.

<PROPERTY type="0" resize="2.5,1.2">

Por ejemplo este es el código que se guardaría en el archivo XML y que nuestro editor y visor comprendería. Son los valores que se pueden observar en la figura 14, anchura de 250% y altura de 120%.

La imagen, en todo momento, está intacta y no se ve afectada en pérdidas de calidad. Con el botón reset podremos volver a los tamaños originales de la imagen en cualquier momento.

Los cambios surtirán efecto una vez se presione el botón de aceptar y no se aplicará ningún cambio si se presiona el botón cancelar.

4.8.4. *ZoomIN / ZoomOUT*

Para la comodidad de edición en el editor en ocasiones es necesario ampliar o reducir nuestra área de dibujo, esto se puede conseguir mediante los botones de zoom.

Figura 19: Botones de Zoom.

Estas modificaciones son simplemente visuales y no afectan al proyecto final aunque se decida aplicar los cambios.

4.8.5. *Filtros sobre la imagen de fondo*

En ocasiones las imágenes que se quieren modificar pueden contener elementos que son difíciles de ver, como por ejemplo en radiografías donde alguna parte puede verse muy oscura, en estos casos el editor permite reajustar el brillo[7] y el contraste[8] de la imagen para que todo pueda verse correctamente.

[7] El **brillo** se puede definir como la cantidad de "oscuridad" que tiene un color, es decir, representa lo claro u oscuro que es un color respecto de su color patrón.

[8] El **contraste** se define como la diferencia relativa en intensidad entre un punto de una imagen y sus alrededores.

Figura 20: Botón y panel de filtros.

Al presionar sobre el botón de filtros, aparece un panel emergente donde se puede modificar el brillo y el contraste. Esta modificación es dinámica, cuando modificamos algún valor inmediatamente se ve el resultado en nuestra imagen de fondo.

En la figura 21 tenemos un ejemplo de una imagen en la que cuesta identificar los elementos y mediante filtros se consigue poder visualizarlos mejor.

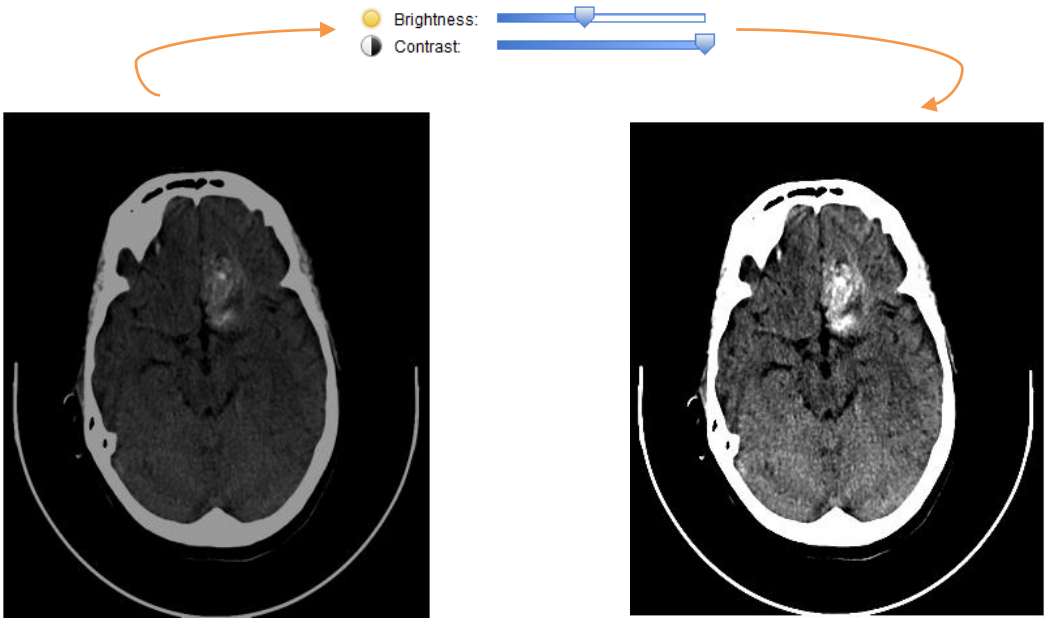

Figura 21: Muestra de uso de los filtros.

Los filtros, al igual que el reajuste de tamaño (apartado 4.7.3), es necesario que no modifiquen definitivamente la imagen. En este caso nuestro visor y editor también comprenden los valores que le indicamos.

<LAYER type="0" filter="-50.0,1.8">

Este sería el código que guardaríamos en el archivo XML donde se pueden ver los valores de filtro que nuestro editor y visor comprenden. Si presionamos el botón reset la imagen restablecería los valores originales de brillo y contraste sin perder ninguna calidad.

4.9. BARRA DE HERRAMIENTAS DE DIBUJO

En esta barra se encuentran todas las herramientas de dibujo que podemos aplicar sobre nuestro proyecto. A continuación se detallarán las herramientas, sus propiedades y su método de uso.

4.9.1. *Línea*

La línea es el elemento más simple de todos, permite al usuario poder incluir una línea sobre la imagen en el área de dibujo.

Para incluir una línea basta con presionar sobre el botón línea y situarnos dentro de nuestro área de dibujo, después realizando un clic[9] y arrastrando el ratón podremos visualizar como va a quedar la línea que queremos insertar. La línea finalmente se insertará una vez acabamos de arrastrar.

Figura 22: Botón línea.

A continuación se muestra un ejemplo de cómo quedaría una línea incrustada.

Figura 23: Muestra de línea incrustada.

Cada línea tiene diferentes propiedades que se pueden modificar, estas propiedades las podremos tratar en su propio panel de control.

A continuación se explica el panel de control de la línea y las diferentes propiedades:

[9] En informática, se denomina **clic**, **hacer clic**, "**clicar**", "**cliquear**" o **pinchar** a la acción de pulsar cualquiera de los botones de un mouse o ratón de computadora. Como resultado de esta operación, el sistema aplica alguna función o proceso al objeto señalado por el cursor o el puntero en el momento de realizarla.

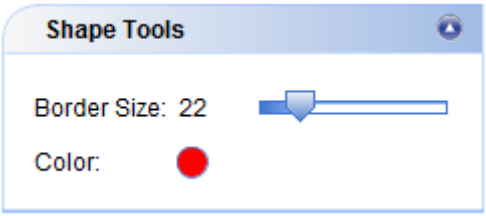

Figura 24: Panel de control de la línea / flecha.

En el caso de la línea se podrá modificar su grosor y su color, el grosor simplemente modificando el valor mediante el deslizador se realizará el cambio automáticamente. En el caso del color será necesario efectuar un clic del ratón sobre el círculo del color y aparecerá un selector de color con una paleta de colores.

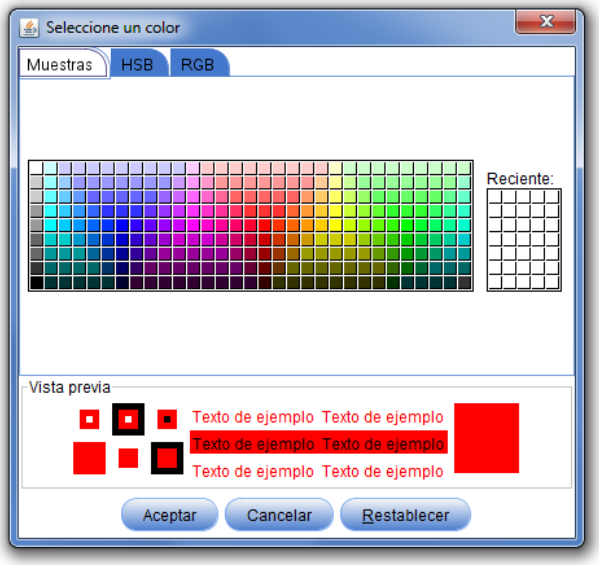

Figura 25: Panel selector de color.

Una vez seleccionado el color pulsaremos el botón aceptar y automáticamente nuestra línea cambiará de color.

Como apunte de codificación nuestra flecha guardará diferentes valores únicos para nuestro modelo de datos, estos valores son:

- Punto inicial
- Punto final
- Grosor de borde
- Color

Estas propiedades son las que definen en esencia a nuestra línea y modificándolas se consigue poder actualizar el aspecto de ésta.

4.9.2. *Flecha*

La flecha es muy parecida a la línea, con esta herramienta podremos insertar una flecha dentro de nuestra área de dibujo.

Figura 26: Botón flecha.

El método para insertarla es igual al de la línea, simplemente deberemos arrastrar y soltar.

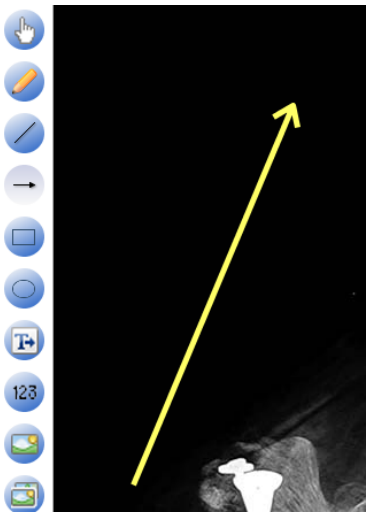

Figura 27: Muestra de flecha incrustada.

Como aspecto interesante y problemático en la codificación se ha de mencionar que la flecha es una forma que se va actualizando dependiendo de la posición, si fuese una forma estática en el momento que el aspa se situara en una posición no controlada haría que el dibujo no pareciera una flecha.

La flecha es la única herramienta con una forma creada aparte, pero puede servir de ejemplo para posteriores actualizaciones a la hora de crear nuevas formas.

Como ya se ha comentado anteriormente las líneas y las flechas son muy similares, tanto que comparten el mismo panel de herramientas y propiedades.

4.9.3. *Rectángulo*

El rectángulo es otro elemento que se puede insertar y que puede ser muy útil a la hora de remarcar zonas u otros usos al gusto del usuario.

Figura 28: Botón rectángulo.

El método de insertar un rectángulo también sigue los pasos de simplemente arrastrar sobre nuestra área de dibujo en la posición deseada y soltar.

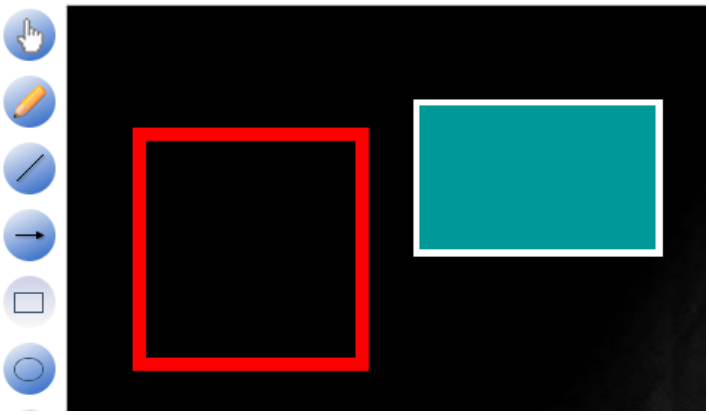

Figura 29: Muestra de rectángulos insertados.

Las propiedades del rectángulo se podrán modificar mediante el panel de control pertinente. A continuación se explicarán las diferentes propiedades del rectángulo.

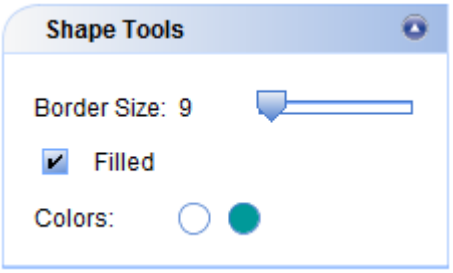

Figura 30: Panel de control del rectángulo / elipse / trazo libre.

En el rectángulo es posible, al igual que los elementos anteriores, la modificación del grosor del borde y su color, pero también cuenta con propiedades adicionales. Las propiedades adicionales son el relleno y el color de éste. Es posible rellenar un rectángulo simplemente marcando la casilla de relleno en el panel de control y ajustar su color mediante el segundo círculo selector de color que nos aparece.

Nuestros rectángulos tendrán unos valores únicos que lo describirán, que son:

- Punto inicial
- Altura
- Anchura
- Si está relleno
- Color de borde
- Color de relleno

Estas son las propiedades que lo definen dentro del modelo de datos y cualquier modificación actualizará el aspecto visual de nuestro rectángulo.

4.9.4. *Elipse*

La herramienta elipse es muy similar al rectángulo. Se trata de insertar una elipse dentro de nuestra área de dibujo.

Como en todas las anteriores herramientas la inserción es muy simple, solo se ha de arrastrar y soltar.

Figura 31: Botón elipse.

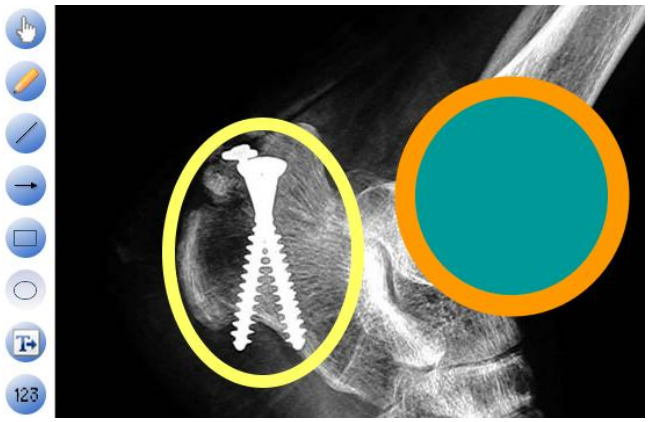

Figura 32: Muestra de elipses insertadas.

Como se ha comentado anteriormente el rectángulo y la elipse son muy similares, tanto que comparten el mismo panel de herramientas y las mismas propiedades en el modelo de datos.

4.9.5. *Trazo libre*

La herrmienta trazo libre consiste en, como su propio nombre indica el poder insertar un trazo dibujado libremente por el usuario. Esta herramienta puede resultar muy útil para remarcar zonas, hacer indicaciones rápidas…

Figura 33: Botón trazo libre.

Figura 34: Muestra de trazo libre.

Las propiedades de nuestro trazo libre son las mismas que el rectángulo y la elipse, por lo tanto para editarlas se usa el mismo panel de control (Figura 30).

Como aspecto interesante de la codificación se ha de mencionar que para guardar nuestro trazo libre a medida que el usuario arrastra el ratón por el área de dibujo se van guardando los puntos por los que pasa como si fuese un camino, estos puntos son guardados en un ArrayList[10] de puntos y más tarde transformado en una forma para su edición.

En nuestro modelo de datos los valores que describen al trazo libre son los siguientes:

- Forma
- Grosor de borde
- Si está relleno
- Color de borde
- Color de relleno

4.9.6. *Texto*

El editor también permite la inserción de texto, que puede ser muy útil para hacer pequeñas indicaciones.

La manera de insertarlo es algo diferente que las demás pero muy simple. Se ha de situar el ratón en la zona deseada, realizar un clic y empezar a escribir. Una vez hecho esto el texto se verá reflejado en pantalla y se actualizará a medida que escribamos algún carácter.

Figura 35: Botón texto.

Figura 36: Muestra de texto insertado.

[10] Un **ArrayList** es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, que sigue un orden. Seincrementa y decrementa automáticamente.

Las propiedades del texto se modifican mediante el panel de control de texto, a continuación sus caracteristicas:

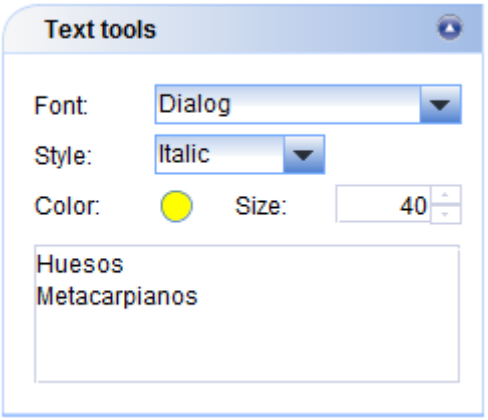

Figura 37: Panel de control texto.

Mediante este panel podemos modifcar todos los valores de nuestro texto. Se puede modificar la fuente[11] del texto, el estilo (normal, negrita, cursiva, negrita-cursiva), el color y el tamaño.

En el panel también podemos modificar el texto en cualquier momento mediante una caja de texto incorporada. Cualquier modificación tendrá efecto visual inmediato en el área de dibujo.

Cabe mencionar en el tema de la codificación dos aspectos que han sido más laboriosos de solucionar.

- El texto necesariamente debía permitir ser multilínea: algunos elementos de Java como las cajas de texto permiten el texto multilínea, pero esto sólo nos permitía captarlo correctamente, para dibujarlo en pantalla se usa la función:

  drawString(texto, posición)

  Al usar esta función el texto aparecía visualmente todo en la misma línea. Para solucionarlo fue necesario dividir el texto en líneas, introducirlas en un ArrayList, indicarles una nueva posición dependiendo del tamaño del texto e insertarlas una a una dentro del área de dibujo.

- Actualización visual del texto: otro de los problemas que surgió al inicio del diseño era que sólo se podía mostrar el texto una vez se había acabado de escribir y presionado un botón de aceptar, esto era algo engorroso para el usuario y se intentó solucionar.
  La manera de solucionarlo fue añadiendo eventos de Java que detectaban las pulsaciones del teclado. Una vez detectada una pulsación se procedería a repintar la pantalla, dando un efecto dinámico al introducir el texto.

[11] **Fuente**, conjunto de caracteres tipográficos de un determinado diseño y tamaño de estilo o tipo de letra.

En el modelo de datos nuestro elemento texto tiene los siguientes valores descriptivos:

- Punto inicial
- Fuente
- Estilo
- Color
- Tamaño
- Texto

El campo texto no se guarda en el modelo de dato como un ArrayList de líneas como hemos comentado anteriormente, sino que este procedimiento se trata en el propio visor y editor para ahorrar espacio en nuestros archivos XML.

4.9.7. *Enumeración*

En muchas imágenes en las que tengamos que realizar modificaciones, el usuario puede necesitar enumerar algún elemento. Esto se podría hacer mediante la herramienta texto pero resultaría algo engorroso para el usuario tener que ir punto a punto escribiendo el número.

Figura 38: Botón enumeración.

Para aumentar la eficiencia y la comodidad se ha desarrollado esta herramienta que es muy sencilla de usar. El usuario simplemente ha de situar el ratón en la posición deseada y realizar un clic, con esto conseguimos que se inicialice la enumeración apareciendo un 1, si nos volvemos a mover a otra posición y realizamos otro clic aparecerá un 2 incrementándose la enumeración y así sucesivamente.

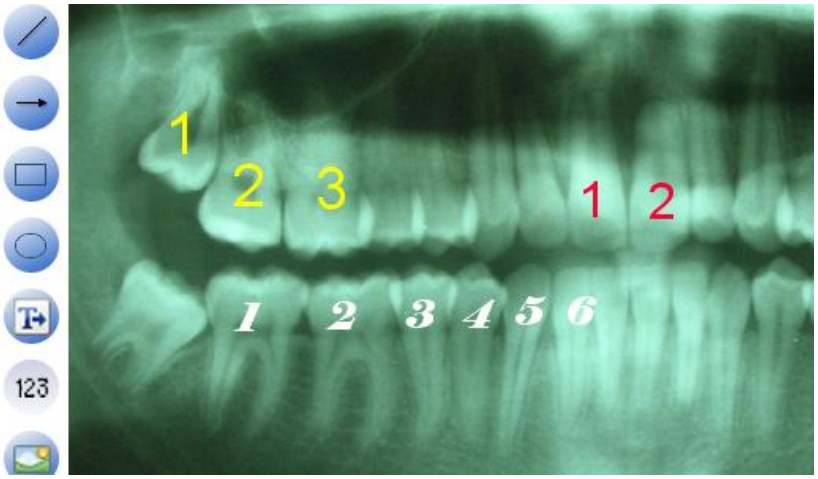

Figura 39: Muestra de enumeraciones insertadas.

Las propiedades de cada enumeración se pueden modificar mediante el panel de control de las enumeraciones.

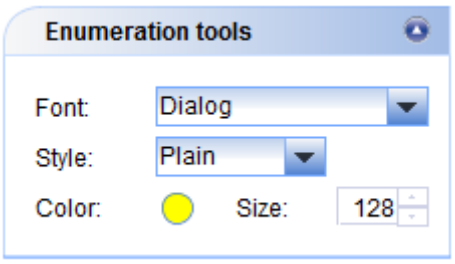

Figura 40: Panel de control enumeraciones.

Este panel es similar al del texto, ya que podremos modificar su fuente, estilos, color y tamaño.

Para codificar esta herramienta se tuvo que pensar una manera que fuese óptima y manejable, ya que posteriormente a cada elemento de la enumeración se le pueden aplicar algunos cambios.

La creación de una enumeración se consigue guardando cada posición del clic de ratón en un ArrayList, donde cada una de estas posiciones es un número de la enumeración que no es necesario indicar cuál es porque siguen un orden, por ejemplo:

ArrayList: P (5, 10), P (15, 20), P (1, 5), P (7, 15).

Esta enumeración contiene 4 números y por ejemplo el número 3 estará situado en la posición (1,5).

En el modelo de datos de nuestras enumeraciones se guardan los valores:

- ArrayList de puntos
- Fuente
- Estilo
- Color
- Tamaño

Uno de los apuntes curiosos sería el cómo guardar un ArrayList dentro del archivo XML, como los archivos XML solo permiten guardar texto hemos de crear un texto mediante el ArrayList d puntos, entonces en el ejemplo anterior quedaría así:

ArrayList: P (5, 10), P (15, 20), P (1, 5), P (7, 15).

XML: "5, 10|15, 20|1, 5|7, 15"

Posteriormente ya se tratar este texto para volverlo a transformar en una ArrayList de puntos.

4.9.8. *Imágenes de archivo*

Otra herramienta que incluye el editor es la posibilidad de insertar imágenes de archivo al gusto del usuario.

Figura 41: Botón imagen de archivo.

El método para insertarla es simplemente presionando el botón de imágenes de archivo y seleccionando la imagen mediante una pantalla de selección de fichero.

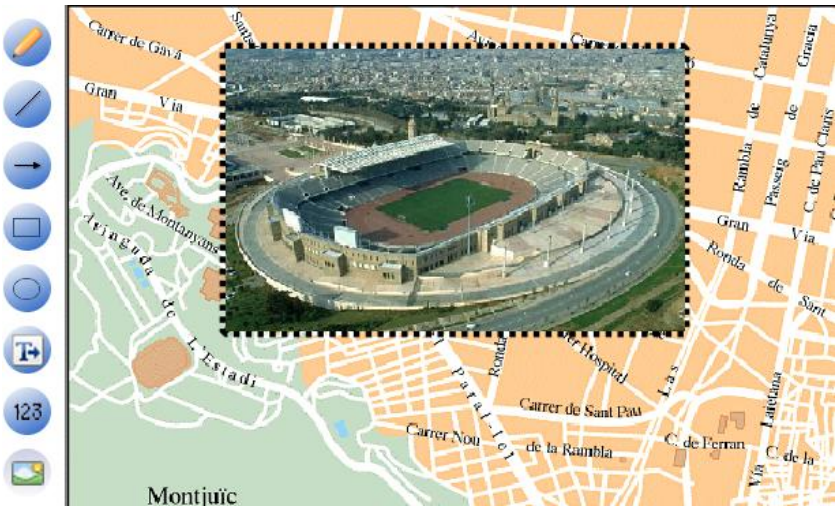

Figura 42: Muestra de imagen de archivo insertada.

A las imágenes de archivo se le pueden aplicar diversas propiedades, que se pueden configurar en los paneles de control de imágenes.

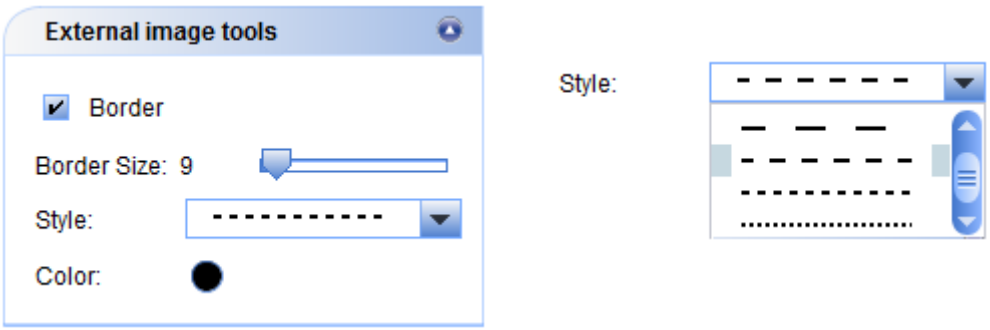

Figura 43: Panel de control de imágenes de archivo con tipos de estilo de borde.

Este panel da la posibilidad de añadir un borde sobre la imagen que hemos insertado. Además a este borde se le puede aplicar un tamaño de grosor, un estilo que obtendremos de un muestrario de estilos y un cambio de color.

Además del panel anterior, las imágenes de archivo permiten aplicar filtros, que son los mismos que sobre el fondo, el brillo y el contraste.

Figura 44: Panel de control de filtros sobre imágenes de archivo.

Uno de los problemas en la codificación ha sido el guardado de la imagen en formato XML. Como se ha mencionado antes los archivos XML sólo permiten insertar texto, por este motivo era necesario transformar la imagen a texto. Esto se ha conseguido mediante funciones de la librería Karat-base que obtiene una lista de los bytes de la imagen y los codifica en texto.

En el modelo de datos se guardarán los siguientes valores:

- Imagen
- Posición
- Altura
- Anchura
- Si tiene borde
- Grosor del borde
- Estilo del borde
- Color del borde
- Brillo
- Contraste

Una vez se ha insertado una imagen de archivo dentro de nuestro proyecto no es necesario que esta imagen siga estando dentro del ordenador. El uso de este proyecto en otro ordenador no supondrá ningún problema, ya que la imagen se queda guardada completamente dentro del fichero XML.

4.9.9. *Imágenes prediseñadas*

Las imágenes prediseñadas son imágenes que ya están insertadas dentro del editor. Una vez abierto el editor el usuario tiene disponible una serie de imágenes para poderlas añadir dentro de su proyecto.

Figura 45: Botón imagen de archivo.

Esta es una herramienta muy útil por su comodidad y posibilidades que ofrece ya que para añadirlas en el proyecto simplemente se ha de

arrastrar la imagen desde el panel de control a la posición deseada dentro del área de dibujo y esta ya se adjuntará a nuestro proyecto.

Esta herramienta abre una enorme posibilidad de usos ya que cambiando las imágenes prediseñadas se puede abarcar muchos marcos de trabajo, a continuación un posible uso de las imágenes prediseñadas para mostrar la meteorología de Cataluña.

Figura 46: Muestra de imágenes prediseñadas.

Para añadir la imagen prediseñada se debe arrastrar la imagen desde el panel de control, que tiene el siguiente aspecto:

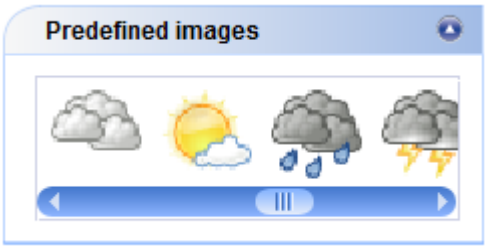

Figura 47: Panel de control de imágenes prediseñadas.

El panel muestra en miniatura todas las imágenes prediseñadas de las que disponga el editor y una barra de scroll para poder movernos entre las imágenes.

En la parte de codificación unos de los problemas más grandes ha sido como realizar el arrastre de la imagen para adjuntarla al proyecto. Para solucionar este problema se han tenido que aprender conceptos sobre las interfaces *DropTargetListener, Transferable, DragSourceListener* y *DragGestureListener*, que se ocupan de controlar todos los eventos de arrastre del ratón y dan más opciones de control que otras interfaces más automáticas.

Visualmente también ha habido complicaciones en la codificación, ya que cuando se arrastra la imagen prediseñada aparece en pantalla una copia de la imagen acompañando el movimiento. Esto hace que el arrastre sea mucho más dinámico y el usuario vea en todo momento qué es lo que está realizando. Para solucionar este problema se ha hecho uso del concepto de la clase *GlassPane* que se tratará en el apartado 4.14 Otros aspectos interesantes de la codificación.

En nuestro modelo de datos guardaremos los siguientes valores:

- Código de la imagen prediseñada
- Altura
- Anchura
- Posición

4.9.10. *Herramienta mano*

La herramienta mano es una de las herramientas más importantes del editor, con ella el usuario puede hacer modificaciones directas en los elementos añadidos. Cuando se pulsa el botón de la herramienta mano y se tiene un elemento seleccionado en nuestro sistema de capas, este elemento pasa a un estado de edición y se muestra visualmente con "cajas de modificación", que permiten que mediante el arrastre con el ratón nuestro elemento modifique sus propiedades.

Figura 48: Botón herramienta mano.

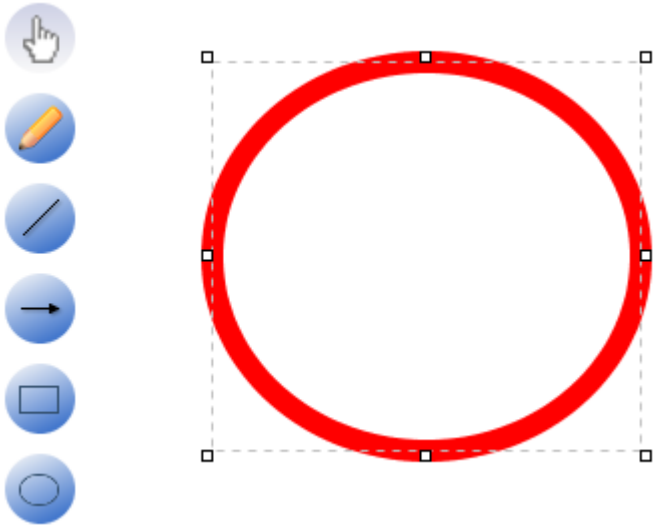

Figura 49: Muestra de uso de la herramienta mano.

En esta imagen se puede apreciar que cuando tenemos pulsada la herramienta mano han aparecido cajas de modificación alrededor de nuestro elemento elipse. Dependiendo de donde situáramos el cursor nos permitiría realizar diferentes acciones, por ejemplo si situáramos el cursor en la caja inferior derecha y arrastráramos el ratón podríamos modificar el tamaño de nuestra elipse.

Esta herramienta está pensada de manera que el usuario pueda modificarlo todo de forma dinámica. El elemento se actualiza constantemente dependiendo de la acción del usuario, dando una sensación de fluidez.

Dependiendo del elemento a modificar la herramienta mano permite unas acciones u otras. A continuación tenemos una lista de lo que se permite dependiendo de los elementos:

- Trazo libre: modificación de posición.
- Línea y flecha: modificación de posición, alargamiento desde el punto inicial o punto final.
- Rectángulo, elipse, imagen de archivo y imagen prediseñada: modificación de posición y modificación de altura y anchura. Esta modificación, si se realiza presionando la tecla SHIFT+clic, permite mantener proporciones para que la imagen/forma no se vea deformada.
- Texto: modificación de posición y modificación del tamaño del texto.
- Enumeración: modificación de la posición de todos los elementos.

Para la comodidad del usuario, cuando se sitúa encima de alguna de las cajas de modificación el cursor se cambia para dar una pista de cuál va a ser la acción que realiza esta caja.

En la codificación esta herramienta ha sido una de las más laboriosas, ya que tiene que tener acceso total y control sobre todos los elementos del modelo de datos además ha de detectar las pulsaciones del ratón sobre el área de dibujo identificando cual es el elemento y la acción deseada.

Al realizar una acción sobre un elemento mediante esta herramienta, dependiendo del movimiento de arrastre, se deberán incrementar/decrementar los valores pertinentes. Todo esto se controla mediante algoritmos que realizan llamadas al modelo de datos y capturan los eventos del ratón.

4.9.11. *Color Picker*

Mediante esta herramienta el usuario podrá seleccionar un color que se encuentra dentro del área de dibujo. Esto puede resultar útil cuando no se sabe exactamente el color que queremos mostrar dentro de la paleta de colores.

Figura 50: Botón Color Picker.

Esta herramienta detecta automáticamente si el elemento al que queremos modificar el color es un elemento con posibilidad de relleno. Si esto es afirmativo al presionar el botón activa un menú desplegable que dará la opción al usuario de si quiere modificar el color del borde / relleno / borde-relleno.

El uso es muy sencillo, simplemente se ha de situar el cursor sobre el área de dibujo y clicar o arrastrar sobre el color deseado. Inmediatamente se mostrará el color sobre el elemento que queremos modificar.

4.10. SISTEMA DE CAPAS

El sistema de capas es una de las partes más importantes del editor.

Un sistema de capas es necesario y muy importante en un editor de imágenes con el que se quiera realizar un trabajo profesional y cuidado, especialmente en nuestra aplicación, la cual está pensada para la edición de imágenes importantes, en las que no se puede realizar una modificación permanente. Mediante un sistema de capas se puede modificar la imagen a nuestro gusto sin preocuparnos de alterar la imagen original.

Como se ha comentado anteriormente, cada elemento que insertamos dentro de nuestro proyecto tiene una capa como referencia, mediante la cual se pueden realizar modificaciones sobre el elemento e incluso eliminarlo si es conveniente. Es además una manera rápida de organizar nuestras modificaciones y poder acceder rápidamente a los elementos.

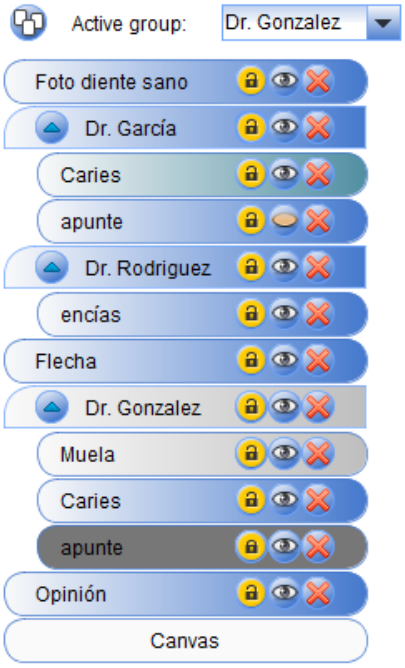

Figura 51: Sistema de capas.

En este apartado se van a tratar todas las posibilidades que ofrece el sistema de capas que se ha diseñado.

4.10.1. *Capa*

Una capa ofrece varias herramientas para poder controlar nuestros elementos insertados en el proyecto. Además una capa puede pasar por diferentes estados los cuales afectan directamente al elemento referenciado. Los posibles estados por los que puede pasar una capa son:

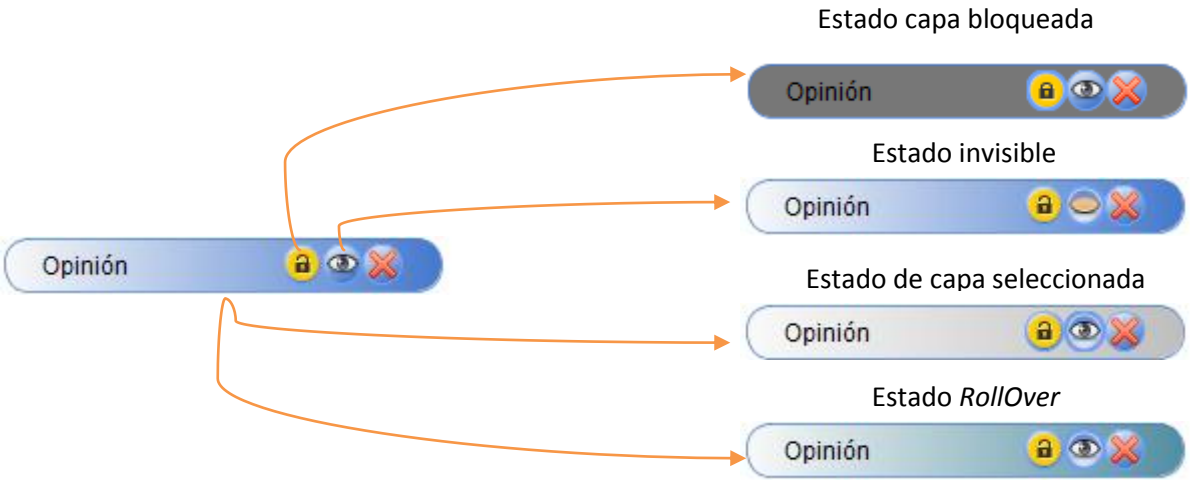

Figura 52: Posibles estados de una capa.

- Estado capa bloqueada: este estado se produce cuando presionamos el botón con forma de candado que incluye la capa. El botón cambiará visualmente como se puede observar en la Figura 51 a un candado cerrado y la capa tomará un color oscurecido.
  Cuando una capa se encuentra en estado bloqueado no se permite realizar ninguna modificación sobre el elemento, incluso no permite su eliminación.

- Estado invisible: este es otro de los estados que se produce presionando un botón de la capa, el que tiene forma de ojo, y que cambiará visualmente a un ojo cerrado.
  Una capa en estado invisible repercute directamente en el elemento que referencia. Este elemento dejará de verse inmediatamente en nuestro proyecto, pero esta modificación no es permanente, presionando otra vez el botón de visibilidad nuestro elemento volverá a aparecer en pantalla.
  Este estado es muy útil, imaginemos que se tienen muchas modificaciones en pantalla, aunque todas sean importantes pueden hacer el proyecto algo engorroso y cargado, si a un usuario sólo le interesan algunas anotaciones en concreto puede hacer invisibles todas las demás para facilitar su visión.

- Estado de capa seleccionada: para poder realizar modificaciones sobre nuestros elementos del proyecto es necesario que esté elemento este seleccionado, esto se consigue con este estado. Para llegar a este estado de capa basta con realizar un clic con el ratón sobre la capa, que tomará un color gris que indicará que es la capa seleccionada.
  El sistema de capas permite la selección múltiple de capas, que puede ser útil si queremos realizar las mismas modificaciones en diferentes elementos. Además el editor está diseñado para detectar si los elementos seleccionados tienen características comunes y mostrar los paneles de control que puedan modificar a todas las capas seleccionadas al mismo tiempo. Para realizar una selección múltiple basta con presionar CTRL+clic en las capas deseadas.

- Estado *RollOver:* este es un estado para facilitar al usuario a saber que capa referencia a que elemento. Aunque las capas permitan la modificación del nombre por uno más adecuado como comentaremos más adelante, puede ser una tarea difícil. Esto se soluciona fácilmente con este estado.
  El estado de *RollOver* se produce cuando situamos nuestro ratón encima de una capa sin realizar ningún clic. El sistema de capas detecta sobre que capa está situado el ratón, busca su elemento referenciado y lo muestra enmarcado dentro del área de dibujo. Cuando se produce este estado la capa cambia a un color verdoso.

Cuando se crea una capa aparece con un nombre por defecto, que puede ser modificado para mejorar la comprensión del proyecto. Este cambio se realiza simplemente realizando un clic sobre la capa y escribiendo el nuevo nombre dentro de la caja de texto que aparecerá donde estaba situado el nombre.

Para finalizar como se ha comentado antes, una capa puede eliminarse cuando lo desee el usuario. Para eliminarla se ha de pulsar el botón con una X dentro de la capa y automáticamente la capa desaparecerá y el elemento referenciado también.

4.10.2. *Grupo*

En muchas ocasiones un usuario querrá gestionar sus capas de una manera que sea cómoda para su comprensión o dividir varias capas dependiendo de quién las haya creado. Esto lo podemos conseguir mediante los grupos, que nos ofrecen poder agrupar varias capas y gestionarlas conjuntamente.

Un ejemplo de uso de los grupos podría ser que en una misma radiografía varios médicos dieran su opinión añadiendo modificaciones. Estos médicos podrían agrupar sus modificaciones en grupos diferentes y así poder saber entre ellos de quien es cada anotación.

Los grupos también dan la posibilidad de poder modificar su nombre para personalizarlos y detectarlos más fácilmente.

Un grupo es muy similar a una capa y contiene los mismos botones de utilidades. Cuando presionamos en un botón del grupo provocará que todas las capas contenidas en él adquieran el mismo efecto. Además el grupo permite minimizarse de manera que si un grupo no interesa lo podremos minimizar para tener mayor visibilidad de los otros.

Figura 53: Grupo y minimización de un grupo.

Para crear un grupo simplemente deberemos presionar el botón de nuevo grupo y automáticamente éste aparecerá en nuestro sistema de capas.

Figura 54: Botón de nuevo grupo.

Para insertar una capa nueva dentro de un grupo se hace mediante la opción del grupo activo. Esta indica que éste es el grupo activo en ese momento y todas las capas nuevas se añadirán a él. Para activar un grupo se puede hacer mediante el panel de grupos activos o simplemente realizando un clic sobre el grupo.

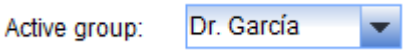

Figura 55: Panel de grupos activos.

4.10.3. *Copiar y pegar capas*

Muchas veces un usuario puede necesitar duplicar una capa. Esto se puede realizar mediante la selección de la capa o capas deseadas y pulsando la tecla CTRL+C o mediante el Menú Edición>Copiar. Una vez realizada una acción de copiar podremos pegar las nuevas capas, para realizarlo basta con presionar CTRL+V o, como en la acción de copiar, mediante el Menú Edición>Pegar.

4.10.4. *Orden de las capas y arrastre*

Las capas siguen un orden de dibujo donde cada capa nueva se dibujará sobre la anterior teniendo como inicial la imagen de fondo.

El sistema de capas permite modificar el orden de las capas de una manera dinámica y fácil. Simplemente se ha de arrastrar la capa deseada a la nueva posición y la capa irá mostrando los cambios dinámicamente en nuestra área de dibujo. Para facilitar el arrastre se mostrará una capa fantasma de donde se posicionará la capa que se está arrastrando.

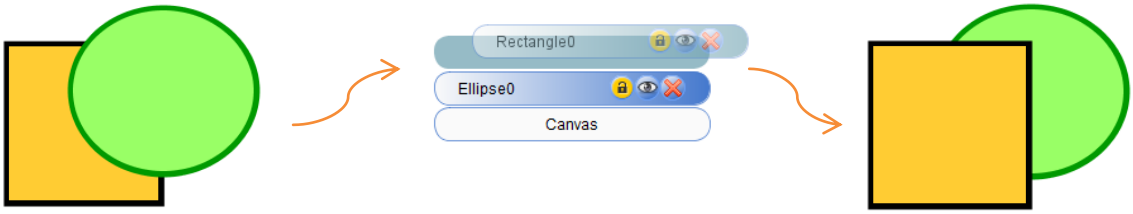

Figura 56: Muestra de un cambio de orden en el sistema de capas.

Además de ordenar capas el arrastre de una capa sirve para poder insertarlas en los grupos.

4.10.5. *Codificación del sistema de capas*

El sistema de capas puede dar la sensación de estar integrado en el modelo de datos y por lo tanto en los ficheros XML por su estrecha relación con los elementos de nuestro proyecto, pero esto en realidad no sucede así.

Las capas se crean a partir de la lectura del modelo de datos y son objetos que se crean y actualizan dinámicamente. Se podría decir que una capa tiene una referencia a un elemento del modelo de datos, pero el modelo de datos y, por consiguiente, sus elementos no tienen ninguna referencia sobre las capas. Esta propiedad es útil para que nuestro modelo de datos sea totalmente independiente del editor y si en un futuro se hacen modificaciones en el sistema de capas no afecte a los proyectos antiguos.

Cuando insertamos un elemento sobre el proyecto, este elemento llama a una función que se encarga de crear una capa nueva. Esta capa obtendrá algunos valores necesarios del elemento, como su clave única para poder acceder a él cuando lo desee y se posicionará en orden en un ArrayList de capas.

A partir de la clave única nuestra capa puede acceder en cualquier momento al elemento que referencia en nuestro modelo de datos y poder realizar los cambios que el usuario le indique.

Los grupos, al contrario que las capas, sí que se guardan en el modelo de datos, pero igualmente esto no afecta a la independencia del modelo de datos con el editor ya que el guardado es simplemente para indicar que existe un grupo y lo considera como si fuese un elemento más dentro de nuestro modelo de datos.

Este grupo está diseñado de tal manera que ocupe el menor espacio posible dentro de nuestro fichero XML, por este motivo simplemente se guarda el nombre del grupo y el número de capas que contiene. Esto puede resultar lioso porque la pregunta inmediata es como el grupo sabe qué capas contiene si sólo dispone de un número. El fichero XML también sigue un orden y como hemos diseñado nuestro grupo como un elemento más, si le indicamos que contiene 2 capas al crear el sistema de capas sabremos que el grupo contiene las 2 capas anteriores que se lean del fichero XML. Veamos un ejemplo:

```
<?xml version="1.0" encoding="windows-1250"?>
<EDITABLE_PICTURE name="TEST">
  <PROPERTIES>
    <PROPERTY type="0" resize="1.0,1.0"></PROPERTY>
  </PROPERTIES>
  <DATA>
    <LAYERS>
      <LAYER type="0" filter="0.0,1.0">/9j/4AAQSkZJRgABAgEASABIAAD/4Q8pRXhpZgAATU0AKgAAAAgABwESAAMAAAABA
      <LAYER type="3" bounds="119,159,282,110" opacity="1.0" color="255,0,0" width="1" visible="true" lo
      <LAYER type="3" bounds="259,121,132,111" opacity="1.0" color="255,0,0" width="1" visible="true" lo
      <LAYER type="2" bounds="108,53,363,233" caption="Nota" opacity="1.0" color="255,0,0" width="1" loc
      <LAYER type="3" bounds="454,39,55,77" caption="Opinión" opacity="1.0" color="255,0,0" width="1" vi
      <LAYER type="10" members="2" caption="Dr. García" locked="false" visible="true"></LAYER>
      <LAYER type="3" bounds="217,339,124,53" opacity="1.0" color="255,0,0" width="1" visible="true" loc
    </LAYERS>
  </DATA>
</EDITABLE_PICTURE>
```

Figura 57: Muestra de un grupo en fichero XML.

Los grupos creados en el sistema de capas adquieren las claves únicas de todos los elementos que contienen, de esta manera se pueden hacer modificaciones sobre las capas accediendo al modelo de datos.

Otro de los problemas en la codificación del sistema de capas ha sido la parte visual. Aparte de que se tenía que asemejar al estilo de Walnut, las capas debían cambiar de tamaño dependiendo de si se encuentran dentro de un grupo. Este cambio se realiza mediante una variable que indica si la capa está integrada en el grupo y si esto sucede se realiza un repintado de la capa adaptándola al tamaño necesario.

4.11. OPACIDAD

En algunas ocasiones un usuario puede necesitar ver qué hay debajo de un elemento. Para que no sea necesario moverlo existe la opción de opacidad, modificando la opacidad podemos dar a un elemento una cierta transparencia.

Para modificar la opacidad tenemos el panel de control de opacidad.

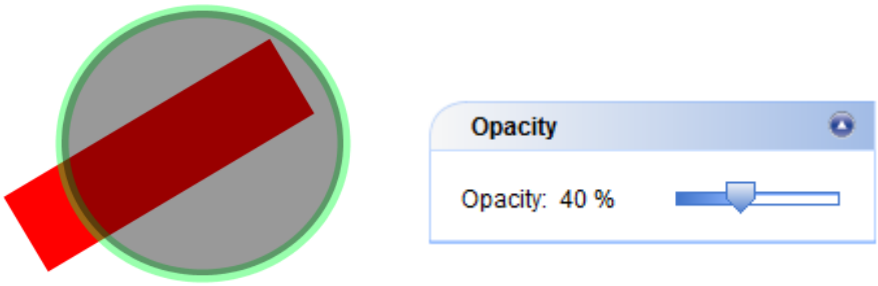

Figura 58: Muestra de opacidad y panel de control.

El valor de opacidad se guarda en el modelo de datos, de esta manera se conservan las modificaciones que se hagan sobre los elementos.

4.12. NOTAS

Cuando insertamos un elemento dentro de nuestro proyecto podemos tener la necesidad de aplicar una nota sobre él. Estas notas son visibles en el visor pasando el ratón por encima del elemento deseado.

Figura 59: Muestra de una nota.

Para insertar una nota se debe realizar mediante el panel de control de notas. Una vez seleccionada la capa deseada simplemente se ha de escribir la nota y presionar el botón de salvar nota. En el caso de las enumeraciones el panel cambiará para mostrar una caja de opciones donde se podrá indicar exactamente sobre qué elemento de la enumeración queremos aplicar la nota.

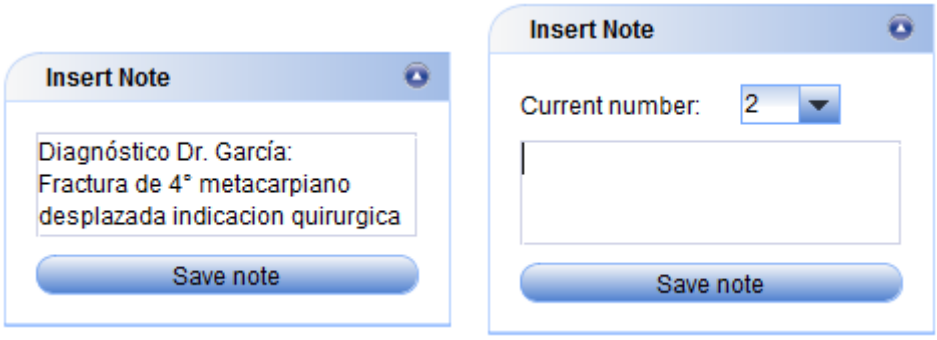

Figura 60: Panel de control de notas, en estado normal y estado de enumeraciones.

Referente a la codificación se puede comentar cómo se realiza el muestreo de las notas en el visor. El visor detecta cuando pasamos el ratón por encima de un elemento insertado y en ese momento busca en el modelo de datos si el elemento tiene una nota, y si la tiene la muestra por pantalla.

4.13. DESHACER / REHACER

Para comodidad del usuario se ha implementado la funcionalidad de deshacer y rehacer, que es una funcionalidad muy importante ya que en un editor de imágenes normalmente el usuario puede cometer fallos al realizar una acción. Mediante esta funcionalidad el usuario puede realizar un paso atrás en el momento que lo desee sin necesidad de tener que borrar completamente su modificación y volver a empezar.

Para realizar un paso atrás en la acción basta con pulsar la tecla CTRL+Z y para volver adelante pulsar CTRL+I, o también se pueden realizar mediante el Menú editar.

Esta funcionalidad, aunque sea una de las más famosas en un editor, acarrea muchos problemas en la programación. La manera en la que se ha implementado es ir guardando todas las acciones que se realizan en un ArrayList de ficheros XML, de esta manera si queremos volver a una acción anterior se cargará automáticamente el fichero XML y ya habremos realizado un paso atrás.

Nuestro ArrayList de XML está limitado en tamaño para no sobrecargar la memoria de la aplicación. Por ello una vez llegado al límite se borran las primeras acciones.

4.14. OTROS ASPECTOS INTERESANTES DE LA CODIFICACIÓN

En este apartado se van a tratar algunos temas interesantes de la codificación que no se han podido ubicar cuando se explicaban las funcionalidades de la aplicación.

4.14.1. *Codificación y accesibilidad al modelo de datos*

El modelo de datos, como se ha comentado antes, está compuesto de elementos, pero para guardar estos datos de forma óptima en la memoria del programa y poder acceder a ellos fácilmente se tuvo que pensar una manera de almacenarlos.

Este método de almacenamiento consiste en un ArrayList y un HashMap.

Un HashMap es una colección de objetos que se guardan de manera óptima en memoria y se accede a ellos mediante una clave única. La pega de los Hashmap es que no siguen un orden y nuestros elementos necesariamente necesitan un orden de dibujo. Esto lo solucionamos mediante la ArrayList.

De esta manera tenemos una ArrayList con las claves únicas ordenadas con las cuales acceder al HashMap para seleccionar el elemento.

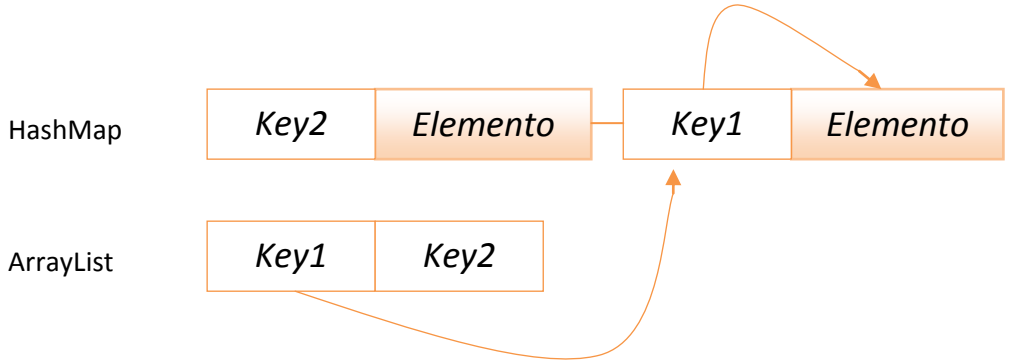

Figura 61: Guardado y accesibilidad del modelo de datos en memoria.

4.14.2. *El parseador de archivos XML*

Se ha hablado de que nuestro modelo de datos se guarda en archivos XML, pero los elementos que contienen estos archivos no son reconocidos directamente por la aplicación.

El parseador se ocupa de leer el archivo XML y transformarlo en un modelo de datos.

Primero se crea un modelo de datos y se procede a leer el archivo XML línea a línea. A medida que el parseador interpreta que está leyendo un elemento lo crea y lo introduce con todas sus propiedades dentro del modelo de datos, además el parseador tiene control de posibles fallos en los archivos XML y indicará si un elemento es correcto o no.

4.14.3. *GlassPane*

Una ventana o frame en Java se compone de muchos paneles, ya que todos los componentes que insertamos los situamos en un panel llamado *Content Pane,* pero en muchas ocasiones podemos necesitar pintar o capturar eventos sobre todos esos elementos a la vez, para eso usamos el GlassPane.

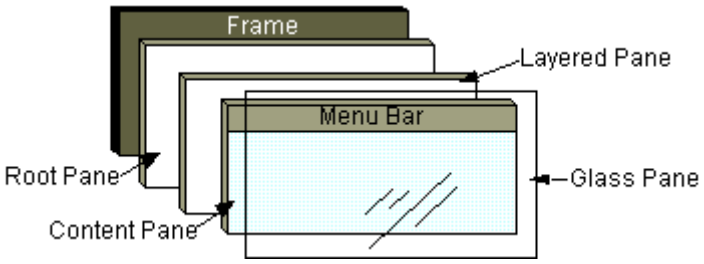

Figura 62: Partes de una ventana o frame en Java.

En nuestra aplicación el GlassPane es usado para mejorar visualmente la experiencia del usuario y añadir funcionalidades muy importantes.

- En las imágenes prediseñadas: una vez se detecta que se está produciendo un arrastre de alguna imagen prediseñada se activa el GlassPane. En nuestro GlassPane se pinta la imagen y se va actualizando su posición dependiendo del movimiento del mouse. La ventaja de usar el GlassPane es que esta imagen fantasma que se mueve antes de insertarla en el proyecto se puede mover por toda la ventana. Una vez se termina la acción de arrastre, el GlassPane detecta si se ha soltado dentro del área de dibujo y si es afirmativo inserta la imagen en el proyecto, sino la descarta.

- En el sistema de capas: en el sistema de capas el uso del GlassPane es algo más complejo y se ocupa de toda la gestión del cambio de orden de las capas y grupos.
  Cuando se detecta algún arrastre en las capas se activa también el GlassPane, en éste se va pintando una imagen que se ha creado a partir de la capa/grupo que se está arrastrando. La posición de esta imagen también se actualiza dependiendo del movimiento del ratón si la acción de arrastre está activa. Al contrario que las imágenes prediseñadas a las capas no se les permite moverse por toda la ventana, simplemente por un tema de lógica hacia el usuario, dándole a entender que sólo va a poder soltar la capa dentro del sistema de capas. Este pintado es controlado por el GlassPane detectando en todo momento por donde pasa el ratón y restringiendo su pintado en caso que no sea el deseado.
  Si se detecta que se pasa sobre una capa o grupo se activa la funcionalidad de orden de capas comentada en el apartado 4.10.2.

El desarrollo de esta funcionalidad ha sido una de las más complejas de toda la aplicación. Antes de explicar el proceso hay que hacer un pequeño apunte sobre la detección en las capas.

Para detectar las capas se dividen en dos partes, de esta manera sabremos si pasamos sobre la parte superior o inferior de una capa, esto será útil posteriormente.

Figura 63: División de las capas.

A continuación se muestra un diagrama de flujo de los pasos que sigue el arrastre de una capa.

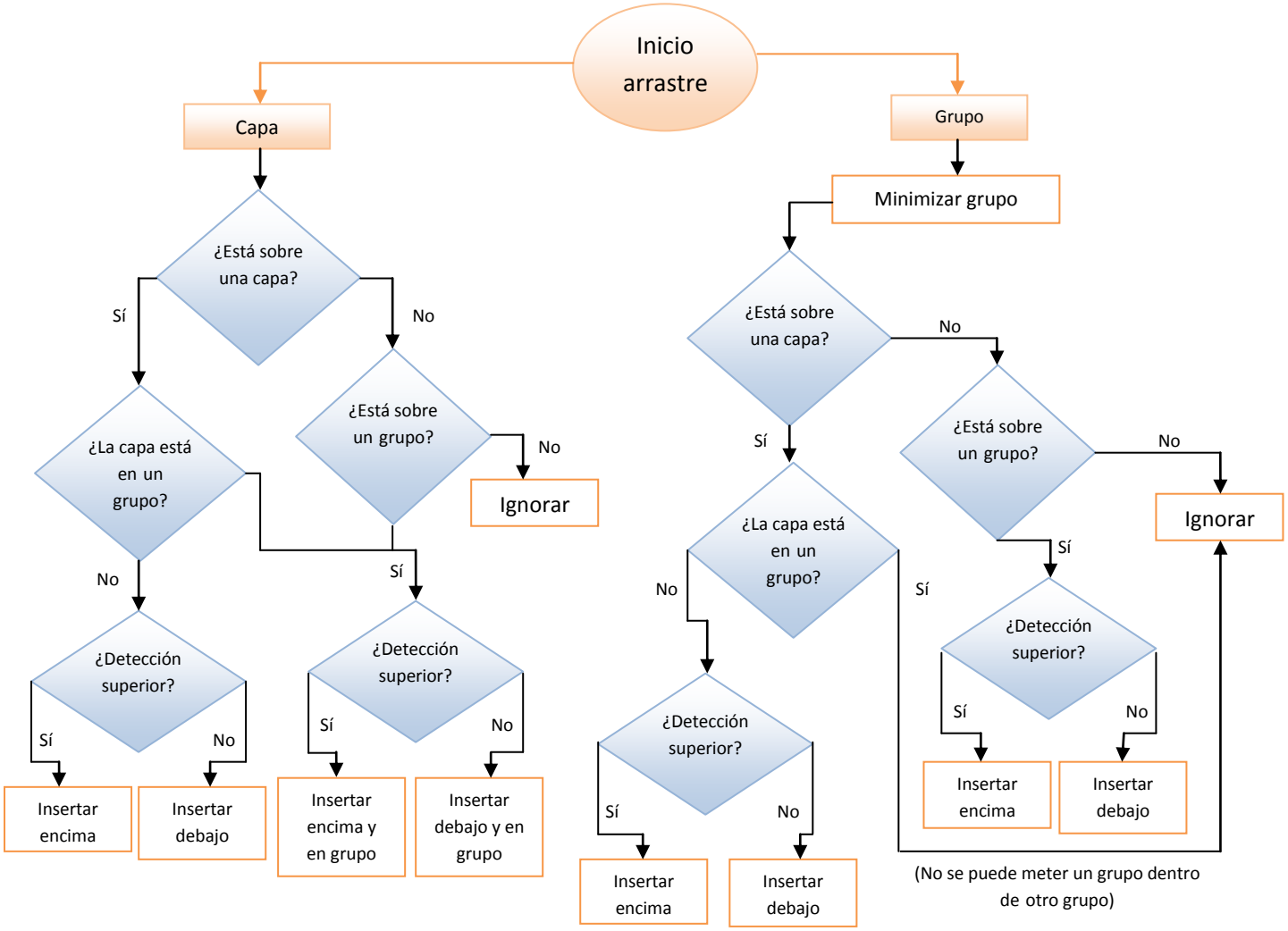

Figura 64: Diagrama de flujo orden de capas.

Como se muestra en el diagrama el GlassPane detecta continuamente sobre qué componente estamos y realiza la acción conveniente dependiendo de los factores en que se encuentre.

PRUEBAS

5.1. INTRODUCCIÓN

Garantizar el correcto funcionamiento de la aplicación y la integridad de los datos que se manipulan es básico para completar el ciclo de desarrollo. Hay muchos factores diferentes a tener en cuenta en el ciclo de desarrollo que minimizan la incidencia de errores y facilitan su detección.

Aunque en un ciclo clásico de desarrollo las pruebas se realizan al final, en nuestro caso las pruebas se han ido intercalando a medida que se desarrollaba la aplicación.

5.2. PRUEBAS UNITARIAS

Las pruebas unitarias son aquellas que nos permiten asegurar que un determinado componente de la aplicación devuelve una salida correcta por una determinada entrada.

En nuestra aplicación se han hecho diversas pruebas de cada una de las herramientas modificando además todas sus propiedades.

Las diferentes pruebas han sido:

- Validación de herramientas de elementos: se han insertado los elementos (líneas, rectángulos, flechas, elipses...) en un proyecto, se han modificado sus propiedades y se ha comprobado que se guarden y recuperen correctamente mediante el fichero XML.

- Validación de los Menús y barras de herramientas: se ha comprobado que todos los menús y barras de herramientas estén correctamente enlazadas con sus funcionalidades.

- Validación de utilidades: se ha comprobado que cada utilidad realice la acción para la que está diseñada. Las utilidades que se han comprobado son:
    - Creación nuevo proyecto.
    - Cambiar tamaño del proyecto.
    - Herramienta Mano.
    - Salvar proyecto.
    - ZoomIN / ZoomOUT.
    - Deshacer / rehacer.
    - ColorPicker.
    - Brillo y contraste.
    - Opacidad.

- Comprobación del sistema de capas: se ha comprobado que se inserten bien las capas, que todas estén referenciadas a su elemento correctamente y que realicen sus propias funcionalidades.

- Comprobación de inserción de notas y visualización en el visor: se ha comprobado que se pueda insertar una nota en cada elemento y se visualice correctamente en el visor.

- Comprobación de la interfaz: se ha comprobado que todas las partes de la interfaz se desplegasen correctamente (minimizaciones, scrollobars, paneles de control).

5.3. PRUEBAS DE INTEGRACIÓN

Las pruebas de integración tienen como finalidad probar el funcionamiento conjunto de la aplicación. Esto quiere decir asegurar que los módulos y acciones probadas en el punto anterior funcionan correctamente de forma conjunta.

Los procesos verificados se detallan en los siguientes puntos:

- Prueba 1: se ha creado un nuevo proyecto, se han insertado todos los elementos y se ha guardado correctamente.

- Prueba 2: se ha abierto un proyecto ya existente, se han modificado los elementos que lo contenían y se ha vuelto a guardar correctamente.

- Prueba 3: se ha abierto un proyecto ya existente, se han modificado e insertado nuevos elementos, se han añadido notas y se ha guardado correctamente. Las notas se ven en el visor.

- Prueba 4: se ha abierto un proyecto existente, se han usado todas las utilidades del editor y se ha comprobado que todas hicieran su función sobre el proyecto.

- Prueba 5: se ha comprobado que al abrir un proyecto ya existente el sistema de capas recupere toda la información correctamente.

CONCLUSIONES

6.1. OBJETIVOS CONSEGUIDOS

A grandes rasgos se puede afirmar que el producto resultante de este proyecto es una solución que encaja perfectamente con los objetivos que se enumeraron durante la definición del proyecto.

La aplicación desarrollada permite de forma sencilla y dinámica la edición de imágenes poniendo al alcance del usuario todas las herramientas que pueda necesitar para conseguir resultados profesionales. Además la aplicación se ha desarrollado de manera que el usuario se sienta cómodo utilizándola y entienda rápidamente todas las funcionalidades. Este era uno de los principales objetivos ya que no está pensada (en principio) para profesionales informáticos y de esta manera cualquier usuario normal puede explotar todas las posibilidades de la aplicación.

Finalmente hay que destacar que la aplicación se ha desarrollado de manera que se pueda adaptar perfectamente al programa Karat y a su nueva interfaz Walnut de UNIT4, tanto visualmente como en la codificación.

6.2. DESVIACIONES DE LA PLANIFICACIÓN

Se ha conseguido seguir la planificación temporal inicial en gran medida aunque algunos apartados como la investigación y aprendizaje de la tecnología a usar han llevado más tiempo del planificado, pero una vez conseguido se ha recuperado fácilmente.

Además cabe destacar incorporaciones nuevas de requisitos planteados por el propio alumno como nuevas herramientas y mejoras (enumeración, ColorPicker, flechas y propiedades de algunas otras) que en un principio no estaban planificadas y que han supuesto un mayor tiempo de diseño y desarrollo.

Hay que indicar también que en varias ocasiones a los largo del diseño y codificación se ha tenido que rehacer trabajo por un mal planteamiento o por decisiones del tutor. Esto no ha supuesto ningún problema y el resultado ha mejorado con cada modificación.

La siguiente tabla muestra las modificaciones temporales que se han producido respecto a la planificación inicial.

Nombre de tarea	Duración planificada	Duración real
Investigación de software Open Source y competencia	7días	12 días
Diseño de la Aplicación	39 días	45 días
Desarrollo de la Aplicación	67 días	56 días
Test y pruebas	17 días	14 días
Generación de la Documentación	5 días	9 días

Tabla 11: Desviaciones de la planificación.

Finalmente se va a hacer una pequeña muestra de la evolución visual que ha seguido el programa ya que se ha tenido que modificar en varias ocasiones porque debía adaptarse a la interfaz Walnut.

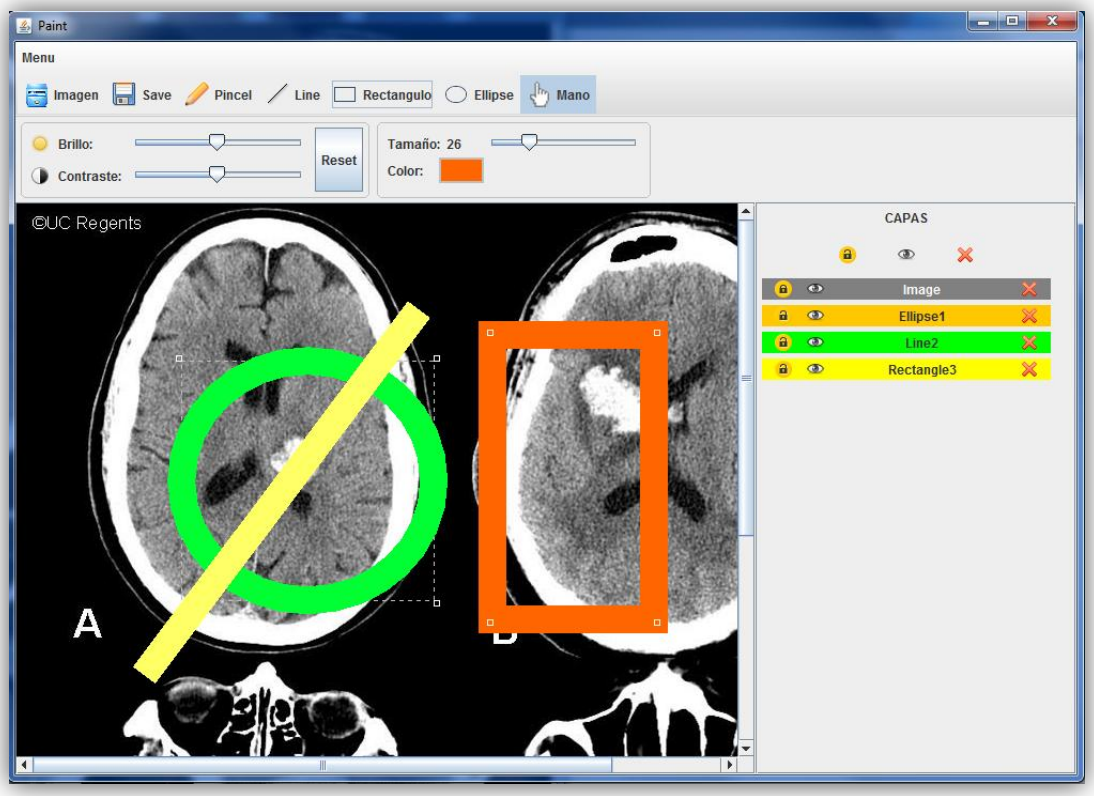

Figura 65: Primer diseño de la interfaz.

Inicialmente se planteó un diseño funcional y bastante rústico para centrarse más en las funcionalidades de las herramientas y pensar posteriormente en adaptarlo a Walnut. Al haber pocas herramientas la barra se situó en la parte superior y el sistema de capas se diseñó de manera sencilla.

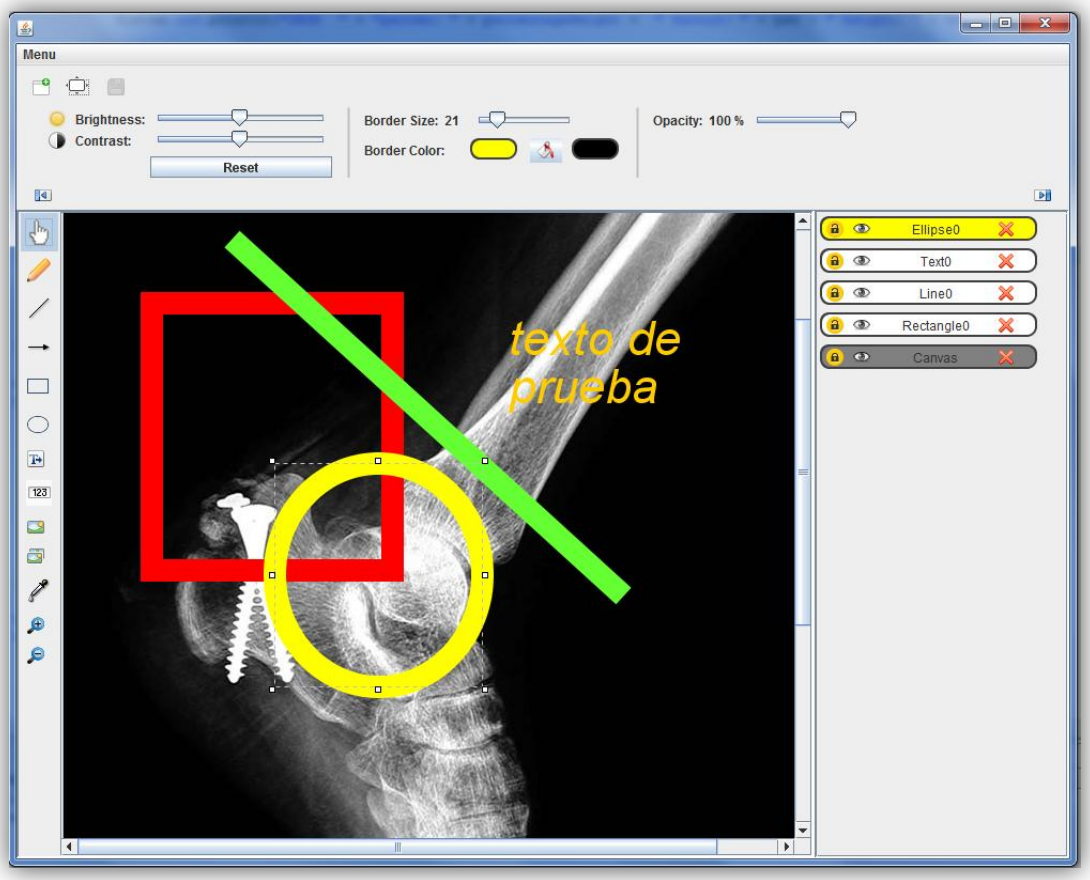

Figura 66: Segundo diseño de la interfaz.

Aproximadamente a mediados de la fase de codificación se modificó la interfaz para ordenar mejor las herramientas y empezar a parecerse a Walnut.

En este nuevo diseño se optó por insertar la barra de herramientas en la parte izquierda debido a que el número de herramientas creció considerablemente y que por experiencia en editores profesionales las herramientas en un lateral de la pantalla facilitan el trabajo al usuario.

El sistema de capas fue modificado para hacerlo más atractivo visualmente y ya empezar a adaptarse a Walnut por las formas redondeadas que este presenta en muchas de sus propiedades.

Finalmente se añadió una barra de utilidades en la parte superior y unos botones para poder esconder el sistema de capas y la barra de herramientas para facilitar la visión del proyecto en monitores con poca resolución.

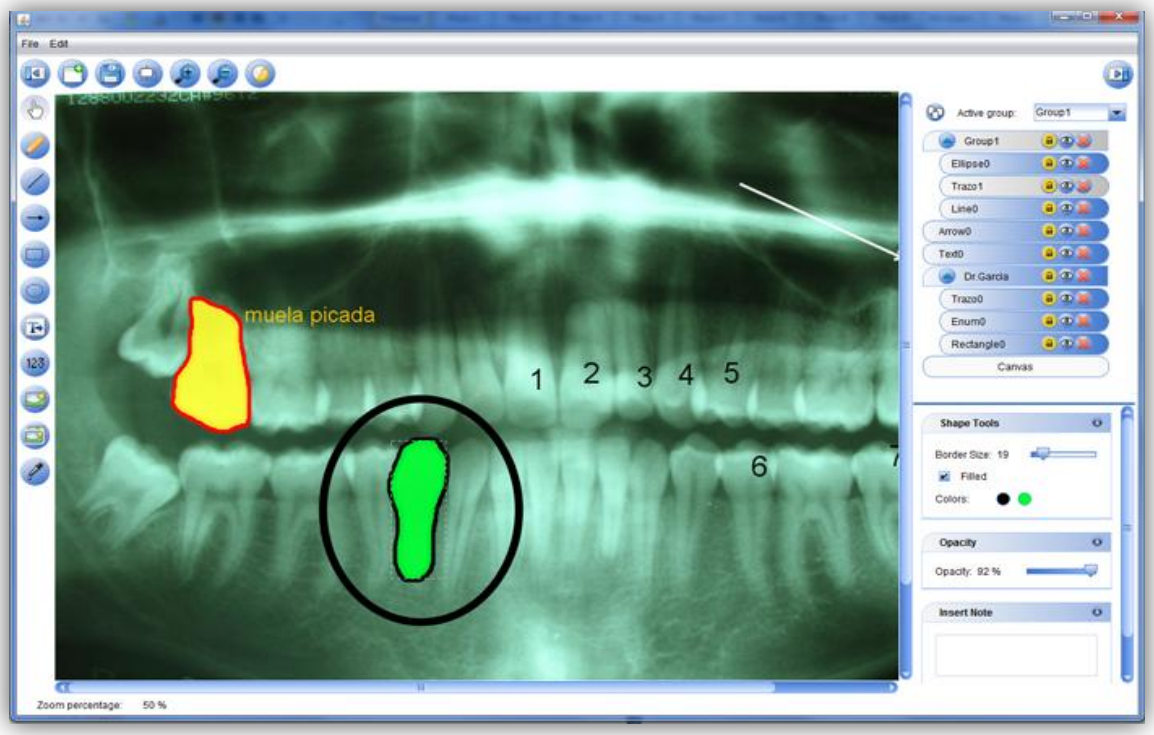

Figura 67: Diseño final de la aplicación.

Una vez acabadas todas las funcionalidades de la aplicación se optó finalmente por realizar los cambios necesarios a la interfaz para que se adaptara perfectamente a Walnut.

Mediante la librería *Karat-laf* comentada en el apartado 4.6 librerías externas se cambió parte del aspecto visual de la interfaz como los botones que pasaron a ser redondeados, los colores azulados y las scrollbars.

Para mejorar el trabajo del usuario se modificó la interfaz para dar más visibilidad al área de dibujo pasando los paneles de control a la derecha, debajo del sistema de capas, además estos paneles como se ha comentado en apartados anteriores también permiten su minimización cuando se desee.

Finalmente el sistema de capas se modificó completamente, no había ninguna referencia en la que orientarse dentro de Walnut pero se diseñó de manera que siendo útil e intuitivo no perdiese la homogeneidad con el resto de la aplicación.

6.3. LÍNEAS DE AMPLIACIÓN

Como se ha comentado anteriormente aunque la aplicación cumple con todos los requisitos que se han pedido, está diseñada de forma modular, por lo tanto es fácilmente ampliable.

La edición de imágenes es un mundo el cual abarca muchos temas de desarrollo y deja libre a la imaginación posibles ampliaciones, por ejemplo nuevas formas de dibujo o herramientas de

diseño. Por lo tanto en este apartado no se puede decir concretamente qué ampliaciones se podrían llevar a cabo, pero si hacer ver que sí un programa está bien diseñado aunque sea pequeño se puede transformar en una grandísima aplicación con la imaginación y el tiempo suficiente.

6.4. VALORACIÓN PERSONAL

El desarrollo de este proyecto me ha parecido una experiencia muy enriquecedora por varios motivos.

He podido aprender un lenguaje tan importante y potente como es Java que yo desconocía en gran medida porque en la universidad no tuve oportunidad de hacer asignaturas en que se profundizara en su uso.

También ha sido importante el hecho de realizar un proyecto desde cero con todo lo que ello conlleva, diseñar las cosas bien, codificar profesionalmente y documentarlo todo. Estoy seguro que me será muy útil en un futuro.

Me he dado cuenta de lo que realmente cuesta un proyecto de esta envergadura y la satisfacción que produce ver cuando se obtienen resultados.

Gracias al convenio con UNIT4 he podido realizar el proyecto en una gran empresa y darme cuenta de cómo funciona el mundo de la informática a nivel laboral donde es bastante diferente del mundo idílico que se obtiene mientras se está estudiando.

Y finalmente por lo mencionado antes, gracias a estar trabajando el UNIT4 me he quitado el miedo que tenía de no saber realmente como funciona una empresa del sector y si estaba capacitado con mis estudios para desempeñar el papel que me tocase.

Por todos estos motivos mi valoración personal del proyecto es muy positiva y me alegro enormemente de haber escogido trabajar en una empresa que aunque el número de horas ha sido mayor que un proyecto convencional estoy seguro que la recompensa lo vale.

BIBLIOGRAFÍA

A continuación las referencias bibliográficas que se han usado para realizar el proyecto, se han incluido las más usadas pero hay que indicar que se han realizado muchas más búsquedas por varias webs de internet en momentos puntuales.

- http://es.wikipedia.org
  Es una enciclopedia libre y políglota de la Fundación Wikimedia (una organización sin ánimo de lucro).

- http://www.java2s.com/
  Web muy completa sobre tutorías y ejemplos en Java.

- http://download.oracle.com/javase/tutorial/uiswing/
  Web de tutoriales sobre la interfaz en Java.

- http://www.w3schools.com/xml/default.asp
  Web sobre tutoriales de XML.

- http://www.google.com
  Famoso buscador.

AGRADECIMIENTOS

En este apartado me gustaría agradecer a las personas que han hecho posible que este proyecto siguiera adelante.

- A Francesc Martinez mi tutor de UNIT4 por enseñarme todo lo que he necesitado, aguantarme en mis innumerables dudas y animarme en todo momento.

- A Jordi Pons por su apoyo, rapidez en la corrección del proyecto y por darme esta oportunidad de realizar el proyecto en una empresa, lo cual valoro mucho para mi futuro laboral.

- Al departamento de *Research & Development* de UNIT4 por ayudarme cuando lo he necesitado.

Víctor Manuel García Ortega

a 09 de Junio de 2011