FACULTAD DE CIENCIAS

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

Desarrollo de un Clasificador de Imágenes Digitales inspirado en los Clasificadores de Diapositivas de la Fotografía Analógica

(Development of a Digital Image Sorter inspired on the Classical Slide Sorters for Analogical Photography)

Para acceder al Título de
INGENIERO EN INFORMÁTICA

Autor: Ángel Tezanos Ibáñez
Julio 2011

El Dr. Ing. Don Pablo Sánchez Barreiro, Ayudante Doctor del Área de Lenguajes y Sistemas Informáticos de la Facultad de Ciencias de la Universidad de Cantabria,

Certifica que D. Ángel Tezanos Ibáñez, alumno de Ingeniería Informática, ha realizado en el Departamento de Matemáticas, Estadística y Computación de la Universidad de Cantabria, bajo mi dirección, el Proyecto Fin de Carrera titulado:

Desarrollo de un Clasificador de Imágenes Digitales inspirado en los Clasificadores de Diapositivas de la Fotografía Analógica

Revisado el presente trabajo, estimo que puede ser presentado al tribunal que ha de juzgarlo, y autorizamos la presentación de este Proyecto Fin de Carrera en la Universidad de Cantabria.

Santander, Junio de 2011

Fdo.: Pablo Sánchez Barreiro
Ayudante Doctor del Área de Lenguajes y Sistemas Informáticos.

FACULTAD DE CIENCIAS

INGENIERÍA EN INFORMÁTICA

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por:	
	Ángel Tezanos Ibáñez
Director del PFC:	Pablo Sánchez Barreiro
Título:	Desarrollo de un Clasificador de Imágenes Digitales inspirado en los Clasificadores de Diapositivas de la Fotografía Analógica
Title:	Development of a Digital Image Sorter inspired on the Classical Slide Sorters for Analogical Photography

Presentado a examen el día:

para acceder al Título de
INGENIERO EN INFORMÁTICA

Composición del Tribunal:

Presidente (Apellidos, Nombre):
Secretario (Apellidos, Nombre):
Vocal (Apellidos, Nombre):
Vocal (Apellidos, Nombre):
Vocal (Apellidos, Nombre):

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: Vocal

Fdo.: Vocal

Fdo.: Vocal

Fdo.: El Director del PFC

II

Agradecimientos

Como suele ser natural en cualquier proyecto fin de carrera, no es un trabajo solitario. Durante el desarrollo del mismo, uno se va dando cuenta de la importancia de las personas más cercanas que están a su alrededor (padres, hermanos, novia, amigos y compañeros de universidad) y lo necesario que fue tenerlos cerca.

En primer lugar quiero agradecer a Pablo Sánchez Barreiro, por guiarme durante la realización del proyecto y ayudarme a solventar todos los problemas que aparecieron, así como sus propuestas de mejora, las cuales ayudaron en gran medida a crear un software competitivo y manejable.

También quiero agradecer a todas aquellas personas que se interesaron por mi trabajo y preguntaban por el estado del mismo, escuchando con atención todas las explicaciones y descripciones que les daba.

Por último, quiero agradecer a mis padres Ángel y María José, mis hermanas María José y Andrea y a mi novia María, por no sólo comprenderme, sino aguantarme con la paciencia y cariño que demostraron.

Con este proyecto finaliza una etapa de mi vida, sin duda alguna una de las mejores y más enriquecedoras. Por todo ello y a todos, muchas gracias.

Ángel Tezanos.

IV

Resumen

El objetivo del presente Proyecto Fin de Carrera es crear una aplicación software sencilla, usable y ligera para la clasificación de un conjunto de imágenes digitales.

La interfaz gráfica de dicha aplicación debe ser lo más parecida posible a un clasificador retroiluminado de diapositivas analógicas. El usuario debe interactuar con la aplicación como si un clasificador de este tipo se tratase. Una vez seleccionadas clasificadas y ordenadas las fotografías de una colección de imágenes, la aplicación debe permitir exportarlas a una carpeta concreta como un conjunto de archivos, ordenadas de manera adecuada.

Las aplicaciones existentes actualmente no tienen como finalidad principal la ordenación y clasificación de fotografías, por lo que, aunque muchas de ellas son capaces de clasificar y ordenar fotografías, no permiten renombrar las imágenes de manera efectiva. Además, para su ejecución requieren gran cantidad de recursos del sistema.

La finalidad del proyecto es crear un software capaz de clasificar y organizar fotografías con un consumo de recursos bajo y una curva de aprendizaje muy suave. El software será desarrollado en Java, utilizando un desarrollo basado en componentes, adoptando una metodología iterativa e incremental.

Palabras Clave

Imágenes, Fotografías, Clasificación de diapositivas, JavaBeans, Drag&Drop

VI

Preface

This Master Thesis aims to create a simple software application, with a high usability degree, for the ordering classification of a set of digital images.

The application user interface should be as similar as possible to the traditional backlight slide sorters commonly used for the selection, classification and ordering of analogic slides. The user must interact with the application such as if he or she were using one of these classical slide sorters. Once a set photographs from a collection has been properly classified, ordered and selected, the application will have to be able to export these images to a folder specified by the user, ensuring that the ordering between these images is kept.

Similar state-of-art applications do not have as a main goal keeping the ordering between images once they are exported. Most of these applications are able to classify and sort pictures, but the ordering is lost when the images are displayed outside the application. Moreover, these application are often too much heavyweight, demanding a high amount of computer resources.

Thus, the project aims to create a lightweight and usable software application that can sort and organise photos, and whose learning curve is very small. The software will be developed in Java using a component-based development approach. An iterative and incremental software development process is followed.

Keywords

Images, Photography, Pictures, SlideClassification, JavaBeans, Drag&Drop

VIII

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este capítulo se describe una breve introducción al problema que pretende solventar este Proyecto Fin de Carrera. Tras la descripción se expone la estructura del documento.

Índice

1.1. Introducción

Actualmente la fotografía digital ha ocupado el lugar de la fotografía analógica, quedando esta última reducida al ámbito artístico o profesional de altas prestaciones. No obstante, las aplicaciones software dedicadas actualmente a la clasificación o gestión de colecciones de imágenes digitales, adolecen de varios inconvenientes para el usuario medio; entre ellos:

1. Suelen ser programas pesados que consumen bastantes recursos, por lo que son difíciles de ejecutar en computadores con pocas prestaciones;

2. La mayoría de ellos no son intuitivos, poseyendo una curva de aprendizaje en absoluto despreciable; y

3. La clasificación y ordenación de imágenes de una colección de fotografías digitales suele ser tediosa, y cuando es realizable, no suele ser posible exportar las imágenes de forma que éstas puedan ser correctamente visualizadas, en el orden especificado, en un computador que carezca de la aplicación que se ha utilizado para ordenarlas. De hecho en muchos casos, es incluso imposible visualizar dichas colecciones en computadoras con la misma aplicación que se ha usado para ordenarlas pero distinta a la usada para crear dicha clasificación/ordenación.

El objetivo del presente proyecto fin de carrera es crear una aplicación software sencilla, usable y ligera para la clasificación de un conjunto de imágenes digitales. La interfaz gráfica de dicha aplicación deberá ser lo más parecida posible a un clasificador retroiluminado de diapositivas analógicas. El usuario deberá interactuar con la aplicación como si un clasificador de este tipo se tratase. Una vez seleccionadas, clasificadas y ordenadas las fotografías de una colección de imágenes, la aplicación debe permitir exportarlas a una carpeta concreta como un conjunto de archivos ordenados de manera adecuada.

El proyecto se realizará en Java, con objeto de hacer la aplicación multiplataforma. Para la interfaz gráfica se desarrollarán una serie de componentes visuales (o widgets), usando para ello las facilidades proporcionadas por las Graphical Beans de Java para la creación de nuevos componentes software para interfaces gráficas.

1.2. Estructura del Documento

Tras este capítulo introductorio, el resto de la presente memoria se estructura tal como se describe a continuación:

Capítulo 2: Planificación del Proyecto. Describe el ámbito funcional del proyecto y en que casos resulta apropiado su uso. El tipo de metodología usada, así como la planificación del proyecto. Definiendo las iteraciones que habrá, así como los requisitos que existen.

También se hablará de los componentes básicos de los que constará la aplicación, las herramientas que se utilizaran durante el desarrollo, y de la construcción de prototipos.

Capítulo 3: Antecedentes. Como antecedentes se hace una breve introducción a la programación por componentes, explicando en que consiste y las ventajas que conlleva.

A continuación se habla sobre la tecnología de componentes usada en Java, denominada *Java Beans* y las características que debe cumplir para definirse como componente.

Capítulo 4: Creación de los primeros componentes. Es la primera iteración, se muestra la construcción del primer prototipo cumpliendo las finalidades que se marcaron para esa iteración. De esta manera vemos la creación de la mesa y la diapositiva. Así como el comportamiento de *Drag And Drop*.

Se observarán los casos de uso detectados, así como el refinamiento de requisitos, para posteriormente diseñar los componentes y finalmente implementarlos y probarlos.

Capítulo 5: Creación de subsecuencias. Es la quinta iteración, se muestra la construcción de los últimos componentes de la aplicación, y de esta manera finalizar los requisitos de esta iteración, en la cual ya será posible ordenar y clasificar fotografías, aunque de momento no se podrán exportar hasta algunas iteraciones después.

Se observa la los casos de uso detectados, así como el refinamiento de requisitos, para posteriormente diseñar los componentes y finalmente implementarlos y probarlos.

Capítulo 6: Mejora de la interfaz visual. Es la décima iteración, se muestra el cambio visual de toda la aplicación con el fin de hacerla más atractiva para el usuario final.

Capítulo 7: Despliegue y Aceptación. Muestra las decisiones tomadas para el despliegue de la aplicación en los distintos sistemas operativos en los que funciona. Del mismo modo, se informa de la creación de una página web para dar difusión a la aplicación.

También se hablará de la aceptación y de los problemas encontrados en esta fase, así como de las peticiones de los usuarios, y los cambios realizados para satisfacerlas.

Capítulo 8: Conclusiones y Trabajos Futuros. Concluye la presente memoria y comenta posibles trabajos futuros.

Capítulo 2

Planificación del Proyecto

En este capítulo se describe la planificación que tendrá el proyecto, mostrando en líneas generales su ámbito funcional, la metodología de desarrollo utilizada, los requisitos de alto nivel de la aplicación, las iteraciones programadas, el diseño de los artefactos base, y las herramientas utilizadas para la construcción de prototipos.

Índice

2.1. Ámbito Funcional del Proyecto

En esta sección se describe el ámbito funcional de la aplicación *Apolo* la cual consiste en un clasificador y organizador de fotografías.

El objetivo general de la aplicación *Apolo* es ofrecer al usuario la posibilidad de ordenar cómodamente sus fotografías digitales, de acuerdo al criterio que éste elija, y descartar las que no satisfagan la calidad esperada o deseada.

Actualmente gracias a la incorporación de una cámara fotográfica a casi cualquier aparato electrónico, podemos tomar fotografías en prácticamente cualquier instante de nuestra vida. Eso unido a que ya no es necesario un proceso de revelado, nos impulsa a almacenar bastantes fotografías, muchas de ellas totalmente innecesarias. La mayoría de las veces,

Figura 2.1: Ejemplo de clasificador de diapositivas analógicas

las fotografías de interés se encuentran mezcladas con fotografías que no salieron como se esperaba (movidas, mal enfocadas o sobreexpuestas). Otras veces, el orden en el que se almacenan no es el deseado para su exposición, o nos gustaría filtrar algunas de ellas. Por ejemplo, tras un viaje de negocios puede que no sea el mismo conjunto de fotos el que deseemos enseñar a nuestro jefe que a nuestros amigos.

Mantener las fotografías almacenadas de este modo no resulta práctico. Si el usuario desea exhibir un subconjunto de una selección de fotografías, tendrá primero que seleccionar las fotografías que desea mostrar, copiarlas en un lugar aparte y luego ordenarlas de acuerdo al orden de exposición deseado. Dicho trabajo, si se ha de realizar con los gestores de ficheros actuales, puede resultar una tarea ardua y tediosa, sobre todo si se han de renombrar los archivos de forma manual para darles el orden deseado.

Para resolver estos problemas y hacer esta tarea menos tediosa, nace la idea de Apolo, un *Clasificador de Imágenes Digitales inspirado en los Clasificadores de Diapositivas de la Fotografía Analógica*.

Los clasificadores de fotografía analógicos, tal como el que aparece en la Figura 2.1, no es más que una superficie retroiluminada, e inclinada generalmente, con unas filas al estilo de baldas, donde es posible depositar las diapositivas. De esta manera gracias a la luz que emite es posible visualizar la imagen de la diapositiva mientras ésta permanece posada en la balda.

El objetivo del clasificador analógico es que la persona sea capaz de organizar y clasificar las diapositivas de una manera rápida y simple. La persona sólo debe ir posicionándolas en las filas según el orden que desee, e ir descartando las que no le satisfagan. Al finalizar la tarea tendrá las diapositivas situadas encima del clasificador en el orden deseado, y se tratará simplemente de recogerlas y guardarlas en el mismo orden.

El objetivo de Apolo es imitar en la medida de lo posible el funcionamiento de estos clasificadores de diapositivas analógicos. Para ello Apolo ofrece las siguientes funciones:

1. Dentro de la aplicación cada fotografía aparecerá mostrada como si de una diapositiva *clásica* se tratara.

2. Cada diapositiva tiene un menú asociado, el cual ofrece la posibilidad de ver la imagen (a tamaño completo) o descartarla, así como otros datos de interés que se consideren relevantes.

3. Para empezar a clasificar y ordenar fotografías es necesario primero haberlas importado a la aplicación. Una vez importadas, estas aparecerán en la aplicación como un conjunto de diapositivas esparcidas sobre una mesa. A esta zona la denominaremos precisamente *Mesa*.

4. El usuario puede a partir de ese momento empezar a seleccionar las diapositivas que considere adecuadas y arrastrarlas a la *zona de clasificación o estantería*. En esta zona se simularán una especie de baldas, similares a los de la Figura 2.1, donde se puedan depositar las diapositivas arrastradas.

5. El usuario puede en cualquier instante alterar el orden de las diapositivas en las estanterías simplemente arrastrando la diapositiva hacia el lugar que desee.

6. También puede cambiar la diapositiva de estantería de la misma forma, arrastrándola y soltándola en la estantería deseada.

7. Cada vez que el usuario haya conseguido ordenar de forma satisfactoria una secuencia de diapositivas, puede marcarla como subsecuencia ordenada y darle un nombre. A continuación, todas las diapositivas se agruparán en un paquete, el cual tendrá una portada representativa de la subsecuencia; y se moverán a una zona diferenciada que denominaremos de *subsecuencias ordenadas o álbum*. El aspecto de esta zona será como el de una estantería diferenciada, donde se disponen las subsecuencias horizontalmente.

8. El usuario también puede descartar todas las diapositivas que se encuentren en una estantería, dejando éstas de aparecer en la aplicación.

9. En la zona de *subsecuencias ordenadas o álbum* se puede alterar el orden de las subsecuencias. Es decir, una subsecuencia recién añadida, y que inicialmente se situaría al final de la lista de subsecuencias, puede ser desplazada para colocarse entre dos subsecuencias ya existentes o al principio de la lista de subsecuencias.

10. Cuando en la zona superior de la aplicación se encuentren todas las subsecuencias de diapositivas ordenadas de acuerdo a los deseos del usuario, éste podrá exportarlas a un directorio o carpeta a su elección. La aplicación entonces creará una copia de cada una de las fotografías seleccionadas en tal carpeta y las renombrará de manera que preserven el orden deseado.

11. En ningún caso se modificarán o suprimirán las fotografías originales (las que se importan). En cada exportación se duplican tantas fotografías como sean necesarias.

12. Debe ser posible guardar el estado actual de la aplicación en caso de que se tenga que interrumpir el proceso de clasificación y se desee continuarlo más tarde.

13. El usuario, basándose en estos estados parciales de la aplicación, podrá crear álbums de fotos de manera rápida. Solo deberá abrir el fichero que contiene la clasificación y orden de las diapositivas deseadas, y exportarlas a una determinada carpeta para obtener un conjunto de fotografías digitales en el orden deseado.

Tras describir a grandes rasgos el funcionamiento de la aplicación, la siguiente sección proporciona una visión de la metodología que utilizaremos para su desarrollo, así como las justificaciones para la elección de la misma.

2.2. Metodología de Desarrollo

Esta sección muestra detalladamente la metodología de desarrollo que será utilizada durante la construcción de la aplicación *Apolo*.

La metodología de desarrollo que se seguirá es la del **Modelo Iterativo Incremental**[Lar03]. Se trata de un proceso de desarrollo evolutivo, en el cual la aplicación se irá construyendo mediante iteraciones, en cada una de las cuales, en base a incrementos, se otorgarán más funcionalidades al sistema.

Esta metodología requiere, inicialmente, una buena descripción del sistema o aplicación a desarrollar. Es esencial que sea clara y lo más completa posible, pues a partir de ella se sustentará todo el proceso de desarrollo.

Basándose en la descripción se definen una serie de incrementos en donde cada uno añade más funcionalidades a la aplicación, y por tanto cumple con una serie de requisitos. Debe comenzarse con la funcionalidad más básica de la aplicación, de manera que pueda ir construyéndose incrementalmente, es decir que cada versión incorpore a la anterior una funcionalidad nueva, la cual hará que se cumpla un requisito o varios.

Dentro de cada iteración hay un proceso interno, en el que pueden darse o no, las siguientes fases: análisis de los requisitos de esa iteración, diseño, implementación y finalmente prueba del correcto funcionamiento. La primera iteración dará como resultado la versión inicial de la aplicación con una funcionalidad muy básica, y solo cumpliendo los requisitos más básicos. A medida que vayan sucediendo iteraciones la aplicación ira cobrando funcionalidad. En la última iteración se obtendrá el producto final, el cual debe cumplir todos y cada uno de los requisitos.

Las ventajas que tiene aplicar esta metodología al desarrollo del proyecto Apolo es, que al ser la interfaz gráfica una parte imprescindible al igual que la usabilidad, se puede en cada incremento ver si la solución tomada es la adecuada para las expectativas esperadas, y en caso contrario corregirla antes de seguir avanzando en el desarrollo de la misma.

Además gracias a que es un modelo evolutivo, se permiten (y es más, se esperan) cambios

en los requisitos en tiempo de desarrollo. Lo cual permite cierto margen de cambio en el funcionamiento de la aplicación.

2.3. Requisitos de Alto Nivel de la Aplicación

Esta sección muestra la identificación de los requisitos de alto nivel que ha de satisfacer nuestra aplicación software, de acuerdo a la descripción del ámbito funcional proporcionada en la Sección 2.1.

Los requisitos de alto nivel encontrados son los descritos en el cuadro 2.1.

2.4. Iteraciones

En la siguiente sección se muestran las iteraciones planificadas a partir de la división de los requisitos de alto nivel encontrados, como puede verse en la sección 2.3.

Las iteraciones planificadas de acuerdo a la agrupación de funcionalidades son las siguientes:

1. Importar fotografía, moverla (Drag & Drop), visualizarla y descartarla. (Ver R01 y R02)

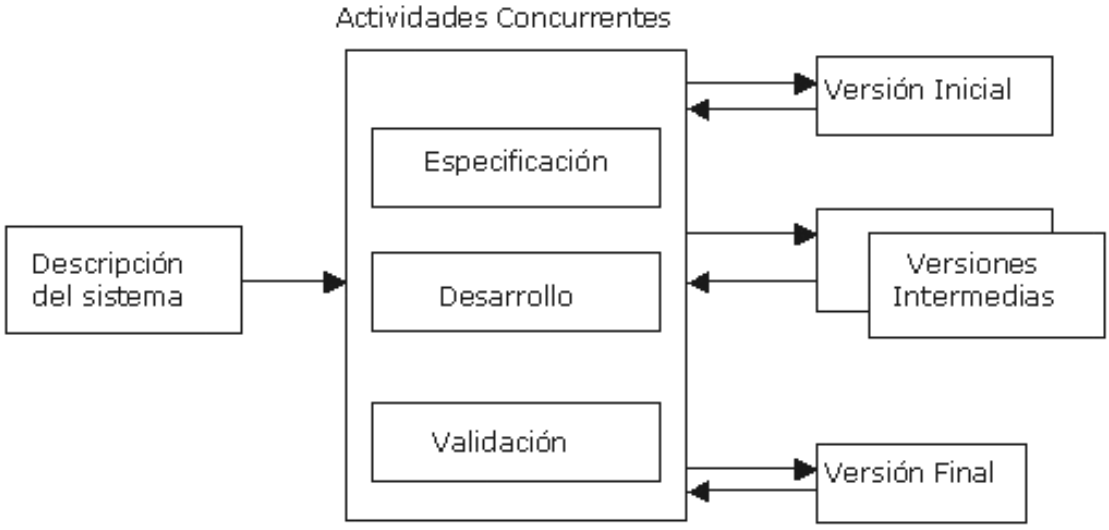

Figura 2.2: Modelo Incremental Iterativo

Referencia	Requisito
R01	Una fotografía deberá ser representada como una diapositiva.
R02	La diapositiva podrá visualizarse o descartarse.
R03	La aplicación importará las fotografías para trabajar con ellas.
R04	Tras la importación aparecerán todas ellas en la zona baja de la aplicación.
R05	Las diapositivas deberán poderse arrastrar hasta la zona media de la aplicación donde habrá unas zonas donde depositarlas (baldas).
R06	Una vez soltada la diapositiva en la zona central, deberá permanecer allí anclada.
R07	Se podrán recolocar las diapositivas dentro de cada estantería arrastrándolas.
R08	Se podrá cambiar diapositivas entre estanterías arrastrándolas.
R09	Una balda, o subsecuencia de diapositivas, ya ordenada y clasificada, podrá ser almacenada en la zona superior de la aplicación.
R10	Una balda, o subsecuencia de diapositivas, podrá ser descartada.
R11	En la zona superior, se podrá reordenar subsecuencias de diapositivas.
R12	Se podrá exportar la clasificación y ordenación a un directorio.
R13	La exportación deberá conservar el orden fijado durante el uso de la aplicación.
R14	No se modificarán las fotografías originales, se copiarán.
R15	Se podrá guardar el estado de la aplicación en un fichero.
R16	Se podrá cargar la aplicación a un estado previo por medio de un fichero.

Cuadro 2.1: Requisitos de alto nivel

2. Importar varias fotografías, llenar pool, realizar estanterías. (Ver R03 y R04)

3. Mover las diapositivas a la parte central y que se queden fijadas. (Ver R05 y R06)

4. Recolocación de diapositivas entre la misma estantería, cambio de diapositivas entre estanterías. (Ver R07 y R08)

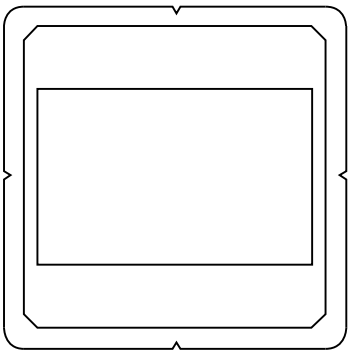

Figura 2.3: Esquema del Diseño del Marco de la diapositiva.

5. Poder añadir una balda a la zona superior, poder descartar balda. (Ver R09 y R10)
6. Reordenar subsecuencias arrastrándolas entre la zona superior. (Ver R11)
7. Exportar subsecuencias en el orden fijado. (Ver R12 y R13)
8. Guardar el estado de la aplicación. (Ver R15)
9. Cargar la aplicación al estado. (Ver R16)
10. Retoques, Efectos visuales, corrección de bugs.

Por cada iteración pueden darse, si fueran necesarias cada una de las siguientes fases: análisis de los requisitos de esa iteración, diseño, implementación y finalmente prueba del correcto funcionamiento.

Una vez descritas las iteraciones planteadas, en el siguiente capítulo se diseñaran los prototipos de los artefactos base del proyecto.

2.5. Diseño de los Artefactos Base del Proyecto

En este capítulo se describen los artefactos básicos de los que constara el proyecto.

Los artefactos básicos del proyecto Apolo son la interfaz gráfica y la representación de la fotografía como diapositiva dentro de la aplicación. Son los elementos más básicos para el funcionamiento de la aplicación.

La representación de la fotografía como una diapositiva conlleva un diseño de un componente que aparezca en la aplicación simulando ser una diapositiva, para ello se ha esquematizado el diseño del marco que tendrá que intentar evocar, lo máximo posible, el recuerdo de la diapositiva clásica. En la imagen 2.3 mostramos el diseño planteado.

La interfaz gráfica[KW04] es un componente muy importante en el proyecto, pues se hace hincapié en que debe ser atractiva y con una curva de aprendizaje muy corta y de pendiente muy suave. Para ello se pensó que la aplicación constaría de tres zonas claramente diferenciadas:

Mesa Se encuentra en la zona inferior de la aplicación. En ella todas las fotografías importadas a la aplicación y con las que se desea trabajar. Aparecerá una a continuación de otra, de manera que el usuario sea capaz de seleccionar la que desee.

Zona de Clasificación o Estantería Esta en la zona central de la aplicación, está compuesta por baldas donde el usuario podrá ir *posando* las diapositivas que vaya arrastrando, de manera que vaya ordenando y clasificando según su gusto y criterio.

Zona de Subsecuencias Ordenadas o Album Se ubica en la parte superior de la aplicación, allí el usuario almacenará las subsecuencias que considere ya ordenadas, de manera que cuando desee exportar, serán estas, según el orden en el que se encuentren, las que serán exportadas.

La imagen de la Figura 2.4 muestra un primer boceto de la interfaz gráfica de la aplicación, donde se pueden ver las diferentes zonas mencionadas en el listado anterior.

Para completar el diseño de los artefactos que componen la aplicación, se realizó a muy alto nivel la estructura de datos que tendrá *Apolo*. Puede verse en la figura 2.5 como se relacionan los componentes de la aplicación.

Una vez descritos los artefactos base de los que constará la aplicación, en el siguiente capítulo se hablará sobre las herramientas usadas para la implementación de los mismos.

2.6. Herramientas utilizadas para el desarrollo de la aplicación

En esta sección se describen las herramientas utilizadas para la creación de la aplicación.

Para el diseño UML[1][JC00] se ha utilizado la herramienta Magic Draw La implementación de la aplicación se desarrolló con ayuda del entorno de desarrollo Eclipse[Hol04], junto con unos plugins para la creación de interfaces visuales y el control de versiones.

Como repositorio donde almacenar las distintas versiones, se utilizó el servicio ofrecido por google code.

El sistema operativo donde se desarrollará será a parte igual en un sistema *Microsoft Windows 7*[EB09] y *Ubuntu 10.10*[MH10].

[1]Unified Modeling Language

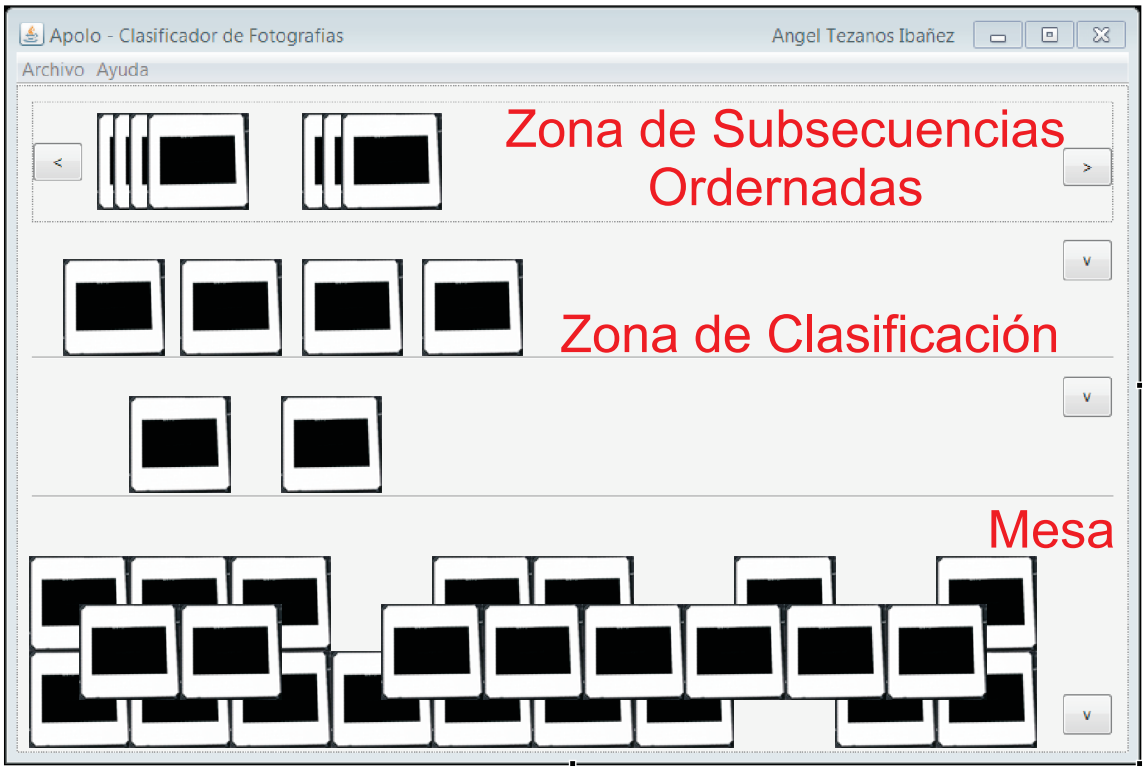

Figura 2.4: Esquema del Diseño Inicial de la Interfaz Gráfica.

2.7. Construcción de Prototipos

En esta sección se detalla la construcción de un primer prototipo de diapositiva, donde se investigará que solución tomar para representar mejor el movimiento de *Drag and Drop*.

El prototipo construido es una aplicación simple en la que aparece una diapositiva y esta puede ser arrastrada por la aplicación. Al arrastrarla entra en acción un efecto de *Drag And Drop* el cual consiste en el la aparición del componente (en este caso la diapositiva) que se arrastra de forma translúcida siguiendo el movimiento del ratón, de esta manera ayuda al usuario a localizar con exactitud donde se arrastra la diapositiva y cuál de todas está arrastrando. Esto puede verse en la figura 2.6, este prototipo servirá para investigar y conocer la técnica del *Drag and Drop* algo fundamental en el uso de la aplicación si queremos que su uso resulte fácil y atractivo.

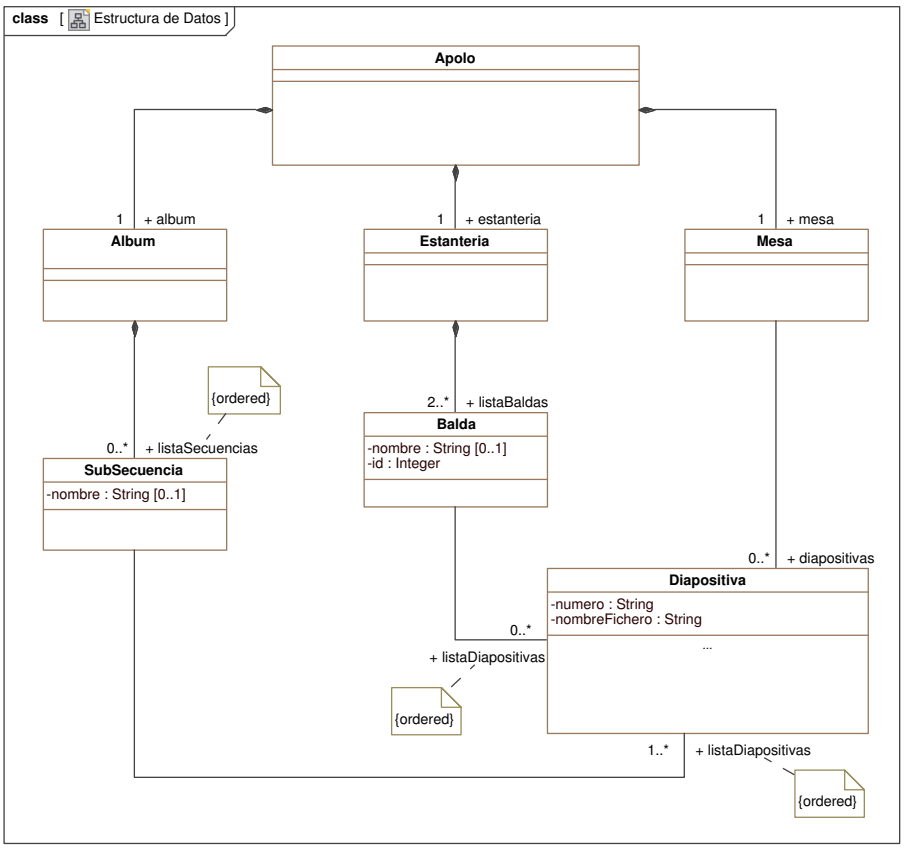

Figura 2.5: Esquema del Diseño de la estructura de datos.

2.8. Sumario

Durante este capítulo se ha descrito la planificación del proyecto fin de carrera, indicando el ámbito funcional en el que se encuentra, así como la metodología que se usará. También se describieron los requisitos de alto nivel y las distintas iteraciones de las que se compondrá el proceso de desarrollo.

También se describió el diseño de los componentes más básicos del proyecto así como las herramientas que se utilizaran para llevarlo a cabo. Finalmente se mostró la construcción del prototipo con el que se ensayaran pruebas del *Drag and Drop*.

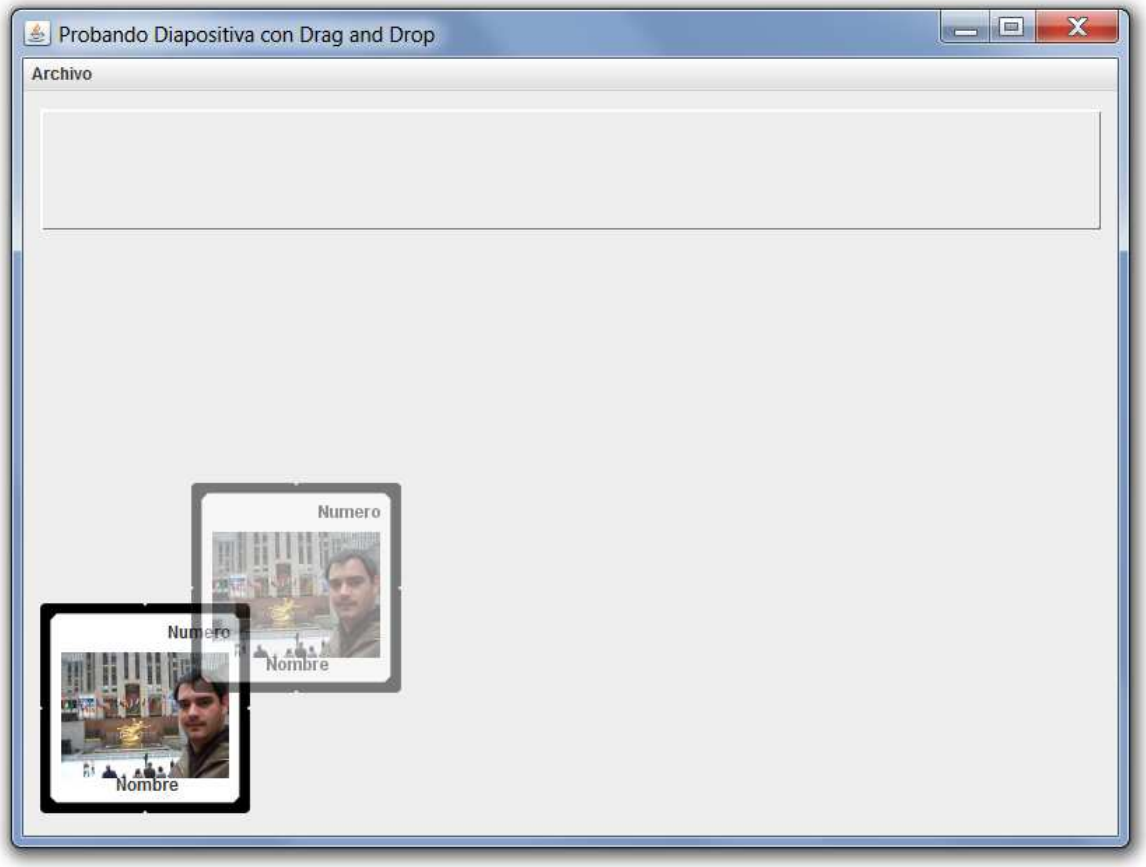

Figura 2.6: Efecto de Drag and Drop

Capítulo 3

Tecnología usada

El presente capítulo describe brevemente las tecnologías sobre las que se fundamenta el presente proyecto. Más concretamente, se explica el funcionamiento del software basado en componentes, en especial Java Beans. Así como el desarrollo de interfaces gráficas en Java, y efecto de *Drag and Drop*.

Índice

3.1. Desarrollo de Software basado en Componentes

En esta sección de hace una introducción a la programación basada en componentes, haciendo hincapié en las ventajas que supone su utilización.

El proyecto se desarrollara bajo una programación orientada a componentes[Szy11]. Esta rama de la ingeniería software trata de construir sistemas a base de componentes funcionales, como si de un juego de piezas se tratase. Para ello cada componente debe tener una interfaz bien definida.

El nivel de abstracción de los componentes se considera más alto que el de los objetos al agrupar unidades funcionales autónomamente. De esta manera se explota en gran medida las posibilidades de reutilización. Pudiendo utilizar componentes ya creados por otros, y/o en otros proyectos de manera rápida y sencilla.

Cada componente software es un elemento o pieza del sistema final que ofrece un servicio y es capaz de comunicarse con el resto de componentes. Básicamente un componente es un

objeto escrito siguiendo unas especificaciones, si las cumple adquiere la característica de **reusabilidad**.

Los componentes deben poder ser serializados para garantizar el envío del estado del objeto a través de flujos de datos.

Para que un componente este bien diseñado requiere un esfuerzo en la fase de diseño, pues se debe tener en cuenta que puede ser reutilizado por muchos programas, debe estar debidamente documentado, probado de manera enfática, es decir, se debe probar la validez de las entradas y que sea capaz de mostrar mensajes de error claros y oportunos; también se debe prever el uso del componente de manera imprevista o incorrecta.

En la siguiente sección se hablará de la programación basada en componentes aplicada a Java.

3.2. Java beans como modelo de componentes

En esta sección se habla de la tecnología JavaBeans[PC98, CSH04] desarrollada por Java.

JavaBeans es la tecnología de componentes de Java, cada componente se le conoce como bean, un bean no es más que una clase de objetos con unas características especiales:

1. Es una clase pública que implementa la interfaz serializable.
2. Expone una serie de propiedades que pueden ser leídas y modificadas por el entorno de desarrollo.
3. Los eventos que posea pueden ser capturados y asociados a una serie de acciones.

Las propiedades no son más que atributos del objeto que pueden ser modificados y leídos por el entorno de desarrollo. Cada propiedad debe tener al menos un método *get* para obtener el valor, y un *set* para modificarlo. En caso de que no se implemente el método set se entenderá que es una propiedad de solo lectura.

Existen varios tipos de propiedades:

- Simples: Representa un único valor.
- Indexadas: Representa un array de valores.
- Ligadas (Bound): Notifican un cambio de la propiedad a otros objetos (listeners).
- Restringidas (Constrained): Similar a la Ligada salvo que los objetos notificados tienen la opción de vetar el cambio.

Gracias a esta tecnología podemos diseñar componentes que iremos añadiendo a nuestra aplicación de manera incremental. Es decir, nos centraremos en la creación de un componente, y cuando este esté implementado, procederemos al siguiente, de manera que el trabajo de diseño e implementación este repartido en fases. Posteriormente, se realizara la conexión entre componentes, para que puedan realizar las acciones que se consideren oportunas.

3.3. Interfaces gráficas

En esta sección se detalla la importancia de construir interfaces gráficas amigables con el usuario.

Una interfaz gráfica debe ser lo más intuitiva que sea posible, de manera que el usuario que haga uso de ella, le sea muy fácil familiarizarse, y no le suponga mucho esfuerzo trabajar con ella.

Una buena interfaz logra que el usuario se sienta cómodo y realice la interacción con la aplicación de manera no tediosa. Esto ayuda y aumenta la productividad, logrando que el usuario realice las acciones de una manera mucho más rápida que si la interfaz de usuario resulta incómoda.

Por ello se desarrollará una interfaz gráfica de usuario, lo más sencilla posible utilizando para lo cual, los fundamentos de un buen diseño de interfaces explicados en varios libros[Gea98, Gea99].

3.4. Drag and Drop

En la siguiente sección se describe el proceso que se adoptará para mostrar el efecto de *Drag and Drop* en la aplicación.

Un campo importante en el desarrollo de la aplicación será la decisión de adoptar la forma de guiar al usuario a la hora de mover diapositivas por la aplicación. Esto es como representar visualmente el *Drag and Drop*.

Para ello se leyeron varios libros especializados[Sch06, JM05] en interfaces gráficas, y se tomó la decisión de adoptar un efecto que mostrase el fantasma[1] de la diapositiva arrastrada, siguiendo en todo momento al cursor mientras este se desplaza. Una vez soltada el efecto fantasma debe desaparecer.

El sistema se basa en añadir al componente, que se desea dotar de un efecto *Drag and Drop*, unos controladores de los eventos *mover ratón y pulsaciones del ratón* de manera que podamos detectar esos eventos sobre el componente. Esos eventos internamente activaran un panel de *cristal* que se sitúa por encima del resto de componentes, y donde aparecerá el fantasma del componente movido, mientras este sea movido.

[1] Aparición del componente semitransparente

Al detener el arrastre del componente y soltar el botón izquierdo del ratón, el controlador desactivará el panel de *cristal* y lanzará un evento a todos los componentes en los que se pueda depositar el elemento arrastrado.

De esta manera, cada componente detectará si el elemento soltado se encuentra dentro de sus coordenadas, y si es así, procederá a añadirlo a su lista.

De esta manera tenemos dos posibles componentes movibles: la diapositiva y la subsecuencia. La diapositiva podrá ser depositada única y exclusivamente en baldas, y la subsecuencia en el Álbum.

3.5. Sumario

El este capítulo se habló sobre la programación basada en componentes, indicando las ventajas que ello conlleva en el desarrollo de software y la reutilización de componentes. También se habló de JavaBeans que será la tecnología usada para desarrollar componentes en Java.

Por otro lado, también se habló de la creación de interfaces gráficas de usuario, las cuales deben ser sencillas para su uso, y sobre todo intuitivas, para que la curva de aprendizaje del usuario sea lo más suave posible. Finalmente se describió el efecto de *Drag and Drop* que se adoptara para la clasificación y ordenación de diapositivas.

Capítulo 4

Creación de los primeros componentes

Este capítulo trata sobre la primera iteración del proyecto. En ella se desarrolla los componentes más básicos de la aplicación, de forma que el resto de componentes se apoyaran en los creados en esta iteración. Serán mostrados los casos de uso pertenecientes a esta iteración figura 4.1, así como los nuevos requisitos descubiertos, tabla 4.1. A continuación se mostrará el proceso de implementación, y finalmente las pruebas realizadas, para comprobar el correcto funcionamiento y cumplimiento de los requisitos.

Índice

4.1. Objetivos

En esta primera iteración se construye la base del proyecto. En ella se crearán los principales componentes de la aplicación. Esto es la interfaz visual de la diapositiva, así como la mesa donde aparecerán una vez importadas. De la misma forma se desarrollará el comportamiento de la diapositiva, su movimiento de *Drag and Drop* y sus opciones de visualización y descarte (o eliminación de la aplicación).

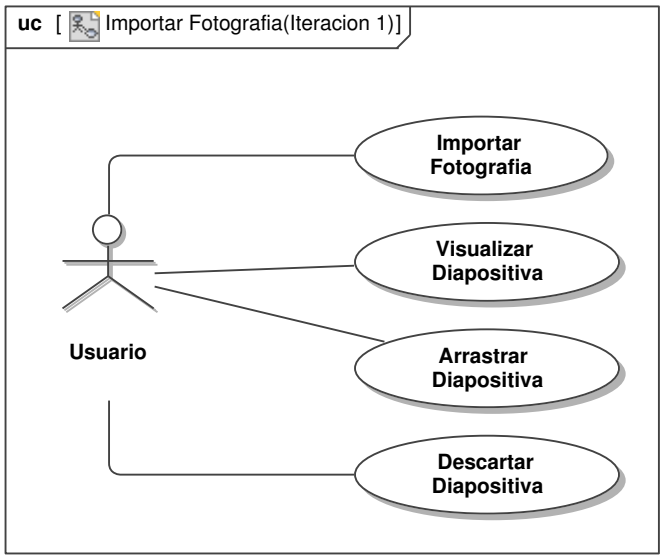

Figura 4.1: Casos de uso de la iteración 1

4.2. Ingeniería de Requisitos

En esta sección se muestran los casos de uso que correspondientes y el refinamiento de requisitos, así como la aportación de nuevos.

4.2.1. Casos de uso

Los casos de uso de la primera iteración corresponden a las acciones de importar fotografía, moverla (Drag and Drop), visualizarla y descartarla. Como puede observarse en la figura 4.1.

Un usuario debe poder importar una fotografía a la aplicación, y que una vez seleccionada ésta aparezca en la mesa. Este es el primer caso de uso detectado. El segundo es que el usuario sea capaz de moverla, y con ello activar los eventos de *Drag and Drop*. Como tercer caso de uso, está la posibilidad de visualizar la diapositiva, esto es ver la imagen que representa, y no la miniatura que se hizo de ella. Como último caso de uso detectado en esta iteración, está la necesidad de poder descartar diapositivas que no queramos que se encuentren en nuestra aplicación.

4.2.2. Refinamiento de requisitos

Como consecuencia del análisis y diseño de esta iteración surgieron una serie de refinamientos de los requisitos. Véase la figura 4.1.

Numero	Nuevo Requisito
R01.1	La importación de Diapositivas debe hacerse desde un hilo aparte, para que no se congele la interfaz de usuario, y éste pueda realizar otras operaciones durante el proceso.
R01.2	Debe poderse importar más de una fotografía a la vez. De manera que el usuario no deba importar una por una.
R03.1	Durante la importación se hace necesario ofrecer al usuario un progreso sobre la operación, de manera que sepa en todo momento en que estado va la importación así como cerciorarse de posibles cuelgues, aunque estos no se esperan.
R05.1	Para la realización del Drag & Drop debe aplicarse algún tipo de sistema para que usuario sea consciente y vea por donde arrastra la diapositiva.

Cuadro 4.1: Refinamiento de Requisitos Iteración 1

4.3. Implementación

En esta sección se describe el proceso de implementación de la primera iteración así como las decisiones tomadas durante esta etapa, de manera que se satisfagan los requisitos descritos en la sección anterior.

La implementación se realizó empezando por el componente más básico, esto es la lógica de la diapositiva, a esta clase las denominamos *Diapositiva.java*. Posteriormente se creó la interfaz visual de la diapositiva, y su comportamiento interno. A continuación se creó la interfaz visual de la mesa y finalmente el comportamiento de la misma. Cada una de las interfaces visuales se llamas *GUIDiapositiva* y *GUIMesa* respectivamente. Y con ello tenemos ya dos componentes de nuestra aplicación.

Una vez finalizada *GUIMesa.java* se procedió desarrollar el método que importa una fotografía e integrar está a la ventana principal. Al utilizar la técnica de desarrollo por componentes resulta sumamente sencillo añadir este tipo de widgets o componentes visuales a la ventana principal. Todo este proceso se creó en un nuevo thread[SO04].

Posteriormente, se implementó el resto de componentes necesarios para la realización de la primera iteración, esto es: un menú desde el que se ofrezca la opción de importar fotografía, un menú en cada diapositiva para poder descartarla o visualizarla. Para visualizarla, se optó por delegar la acción al programa que tenga el sistema operativo asignado por defecto para esa tarea.

Puede verse el aspecto de la aplicación al final de la iteración en la figura 4.2.

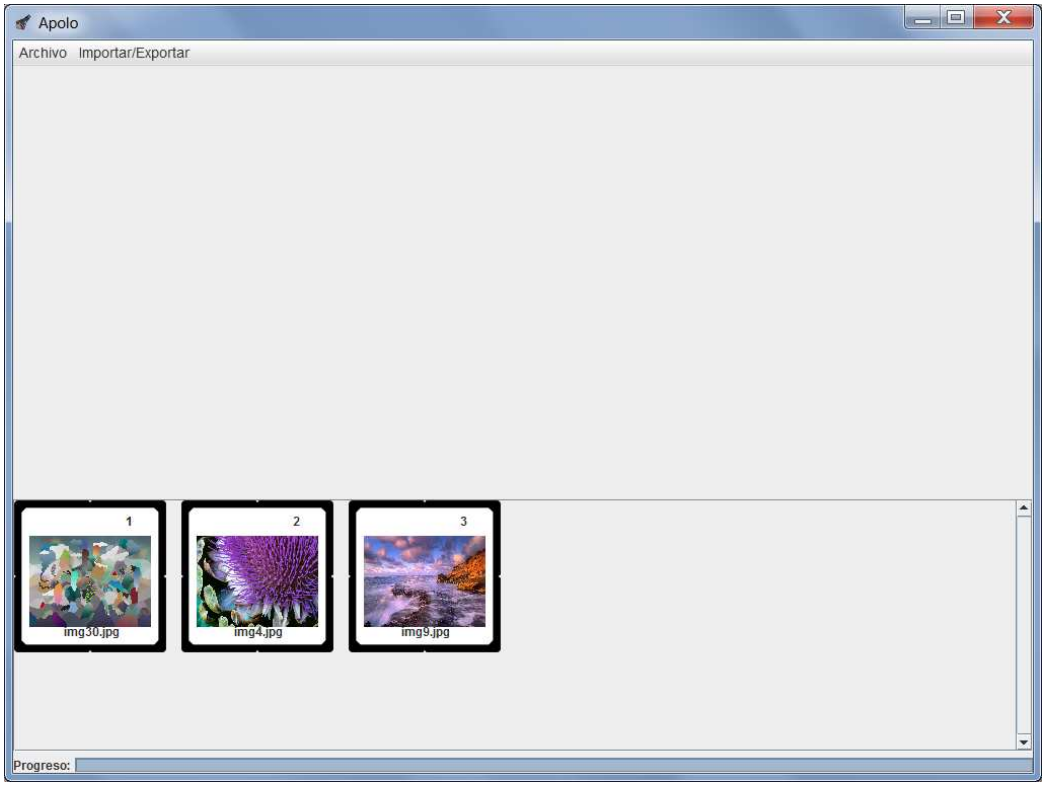

Figura 4.2: Interfaz Gráfica al finalizar la iteración 1

4.4. Pruebas

En esta sección se relatan las pruebas realizadas para la comprobación del cumplimiento de los requisitos descritos en la tabla 4.1.

Una vez concluida la fase de implementación, realizamos las pruebas necesarias para comprobar que los requisitos que se marcaron en la primera iteración están cumplidos.

Para ello ejecutamos la aplicación y observamos si los resultados al ejecutar distintas acciones corresponden con los esperados.

1. **Acción:** Pulsar en el menú la opción importar y seleccionar una fotografía.

   Resultado Esperado: Aparición de la interfaz visual de la diapositiva en la interfaz visual de la mesa.

2. **Acción:** Pulsar en el menú la opción importar y seleccionar varias fotografías.

   Resultado Esperado: Aparición de la interfaz visual de la diapositiva por cada una de las fotografías importadas, en la interfaz visual de la mesa.

3. **Acción:** Pulsar con el botón izquierdo del ratón sobre una diapositiva y sin soltarlo desplazar el cursor.

   Resultado Esperado: Aparición de un efecto que represente el arrastre de la diapositiva seleccionada.

4. **Acción:** Pulsar con el botón derecho del ratón sobre una diapositiva y seleccionar la opción visualizar.

   Resultado Esperado: Visualización de la diapositiva en el programa asignado por defecto del sistema operativo.

5. **Acción:** Pulsar con el botón derecho del ratón sobre una diapositiva y seleccionar la opción descartar.

   Resultado Esperado: Supresión de la diapositiva en cuestión de la mesa. Eliminación de visual de la interfaz.

6. **Acción:** Intentar importar una fotografía ficticia, para ello se introduce un nombre ficticio en el cuadro de dialogo y se pulsa aceptar.

   Resultado Esperado: No aparece ninguna fotografía importada.

4.5. Sumario

En este capítulo se habló sobre las acciones que se realizaron durante la primera iteración del proyecto. Se indicaron los casos de uso, y los nuevos requisitos encontrados. También se indicó la forma de implementar los componentes y funcionalidades correspondientes a esta iteración y finalmente se muestran las pruebas realizadas para comprobar el correcto desempeño de las acciones.

Capítulo 5

Creación de subsecuencias

Este capítulo trata sobre la quinta iteración del proyecto. Con ella se termina prácticamente el proceso de clasificar y ordenar fotografías, a partir de entonces el resto de iteraciones añadirán más funcionalidades a los componentes ya desarrollados. Serán mostrados los casos de uso pertenecientes a esta iteración figura 5.1, así como los nuevos requisitos descubiertos, tabla 5.1. A continuación se mostrara el proceso de implementación, y finalmente las pruebas realizadas, para comprobar el correcto funcionamiento y cumplimiento de los requisitos.

Índice

5.1. Objetivos

En la quinta iteración se encuentra la aplicación a la mitad de su implementación, con la mitad de sus requisitos ya resueltos. En esta iteración se añade el nuevo componente *Álbum*, el último que queda de añadir a la aplicación. A partir de entonces el resto de iteraciones consistirán en ir añadiendo funcionalidades a la aplicación de manera que se vayan cumpliendo los requisitos pedidos.

Al final de esta iteración será posible hacer gran parte de la clasificación, pues ya estarán operativas las baldas, y el alojamiento de subsecuencias en la zona superior, por lo que la

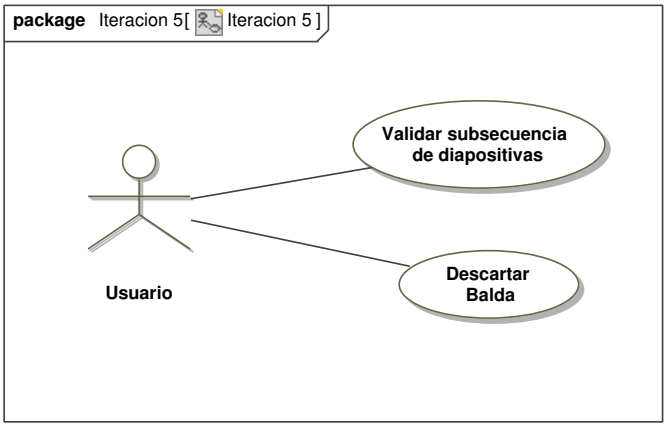

Figura 5.1: Casos de uso de la iteración 5

funcionalidad principal de la aplicación estará prácticamente finalizada, recordemos que es clasificar y organizar fotografías.

5.2. Ingeniería de Requisitos

En esta sección se muestran los casos de uso que la correspondes y el refinamiento de requisitos, así como la aportación de nuevos.

5.2.1. Casos de uso

Los casos de uso de la quinta iteración corresponden a las acciones de añadir el conjunto de diapositivas de una balda a la zona superior, *Álbum*, manteniendo el orden, y de esta forma validar un subconjunto de diapositivas. El otro caso de uso será deshacerse de la balda y de esta manera suprimir las diapositivas que se encuentren en la balda. Ver Figura 5.1.

El usuario al finalizar esta iteración debe ser capaz de poder añadir o validar una subsecuencia de fotos a la zona superior o *Album* de manera que queden bloqueadas y ordenadas para su posterior exportación. También deberá ser posible eliminar una balda de la estantería de la aplicación, y que todas las diapositivas desaparezcan con ella.

5.2.2. Refinamiento de requisitos

A consecuencia del análisis de requisitos de esta iteración surgieron una serie de refinamientos en los requisitos iniciales, así como nuevos requisitos. Ver la sección 5.2.2.

Numero	Nuevo Requisito
R09.1	Para indicar al sistema que se desea validar una balda o subsecuencia de diapositivas, se deberá pulsar un botón.
R09.2	No será posible añadir una balda si no hay sobre ella ninguna diapositiva.
R09.3	A cada balda de la estantería, se la podrá dar un nombre o descripción, de manera que al usuario le resulte más sencillo identificar qué tipos de diapositivas van en ella.
R09.3.1	La descripción que posea la balda, la tendrá el subconjunto de diapositivas, de manera que pueda seguir siendo identificadas como tal.
R09.4	Debe ser posible que un usuario pueda volver a modificar la subsecuencia de diapositivas. Es decir, la subsecuencia debe poder volverse a la estantería como balda.
R09.4.1	Antes de proceder a convertir la estantería en balda, se debe mostrar un mensaje advirtiendo al usuario de la acción que va a realizar.
R09.4.2	Cuando una subsecuencia se convierta en balda, ésta debe contener todas las diapositivas que contenía la subsecuencia, y deben poder ser arrastradas de nuevo.
R09.4.3	La balda que aparecerá como resultado de un subconjunto de diapositivas, tendrá la misma descripción del subconjunto.
R10.1	Antes de ser descartada se le debe preguntar al usuario si es esto lo que realmente desea.

Cuadro 5.1: Refinamiento de Requisitos Iteración 5

5.3. Implementación

En esta sección se describe el proceso de implementación de la quinta iteración así como las decisiones tomadas durante esta etapa, de manera que se satisfagan los requisitos descritos en la sección anterior.

Llegados a esta iteración ya disponemos de la aplicación con su mitad de funcionalidades implementadas, y su principal funcionalidad está prácticamente acabada[1].

Para proceder con la implementación de esta iteración, lo primero que se hizo fue diseñar y posteriormente crear los últimos componentes de la aplicación.

[1] Clasificar y organizar fotografías.

La zona *Álbum o de subsecuencias ordenadas*, una vez creada se añadió al resto de la aplicación y se comprobó su correcta integración en la aplicación.

Una vez integrada se comenzó a enlazar el nuevo componente con los ya presentes, de manera que respondiera a los eventos necesarios para la correcta inclusión en la aplicación. De esta forma, se programó que al pulsar el botón de alguna balda, las diapositivas depositadas en ella, fueran enviadas al nuevo componente, para que éste las incluyera en sí, de manera que mostrara dicha subsecuencia contenida en la balda. Para ello se creó un nuevo componente que representa al conjunto de diapositivas de una balda, y este sólo aparece en la zona superior.

El componente que representa a un conjunto de diapositivas ordenadas en una balda se denomina *Paquete de diapositivas*. Muestra el número de secuencia que tiene, usado para su exportación final, así como la descripción si la tuviera.

Se dotó de la posibilidad de añadir una descripción a las baldas, por lo que hubo que modificar el componente *Balda* para que esto fuera posible. Gracias al diseño por componentes, fue un cambio relativamente sencillo, pero que dota de un detalle bastante útil a la aplicación.

Puede verse el aspecto de la aplicación al final de la iteración en la figura 5.2.

5.4. Pruebas

En esta sección se relatan las pruebas realizadas para la comprobación del cumplimiento de los requisitos descritos en la tabla 5.1.

Una vez concluida la fase de implementación, realizamos las pruebas necesarias para comprobar que los requisitos que se marcaron en la primera iteración están cumplidos.

Para ello ejecutamos la aplicación y observamos si los resultados al ejecutar distintas acciones corresponden con los esperados.

1. **Acción:** Pulsar el botón de la balda que añade una subsecuencia a la zona superior, conteniendo la balda diapositivas.

   Resultado Esperado: Aparición de la representación de la subsecuencia ordenada en la parte alta de la aplicación y desaparición de la balda.

2. **Acción:** Pulsar el botón de la balda que añade una subsecuencia a la zona superior sin que la balda contenga diapositivas.

   Resultado Esperado: Mensaje de error, informando de lo ocurrido.

3. **Acción:** Pulsar el botón de la balda que añade una descripción a la balda.

   Resultado Esperado: Renombre de la balda.

4. **Acción:** Pulsar el botón de la balda que añade una subsecuencia a la zona superior, conteniendo la balda diapositivas y una descripción.

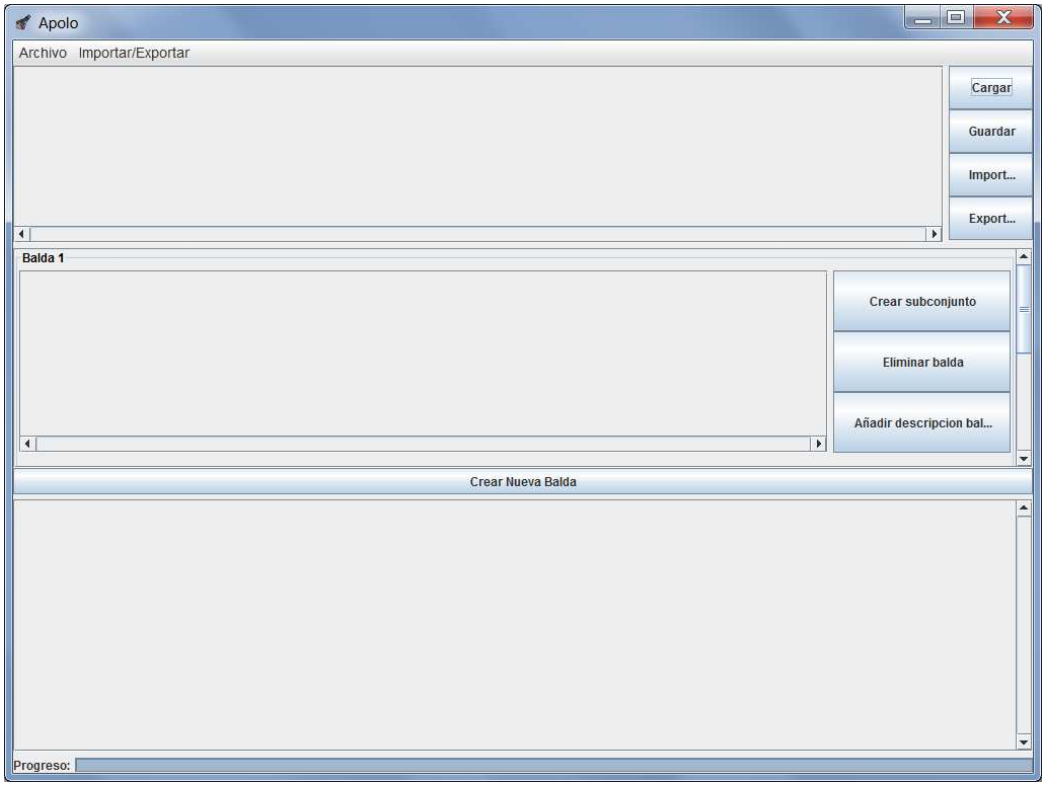

Figura 5.2: Interfaz Gráfica después de la iteración 5

Resultado Esperado: Aparición de la representación de la subsecuencia en la zona superior con la descripción.

5. **Acción:** Pulsar con el botón derecho sobre una subsecuencia y seleccionar en el menú la opción desempaquetar, sin que esta contenga una descripción.

   Resultado Esperado: Aparición de un mensaje y tras su confirmación, creación de una balda conteniendo todas las diapositivas que había ordenadas.

6. Convertir subsecuencia ordenada con descripción en balda:

   Acción: Pulsar con el botón derecho sobre una subsecuencia y seleccionar en el menú la opción desempaquetar, teniendo la subsecuencia una descripción.

   Resultado Esperado: Aparición de un mensaje y tras su confirmación, creación de una balda con descripción conteniendo todas las diapositivas que había ordenadas.

7. **Acción:** Pulsar el botón de la balda que la elimina.

   Resultado Esperado: Aparición de un mensaje solicitando confirmación de la acción y en su caso, desaparición de la balda y todas las diapositivas que contenía,

de contenerlas.

5.5. Sumario

En este capítulo se habló de la quinta iteración del proyecto, así como las decisiones que se tomaron durante la misma. Con esta iteración prácticamente finaliza el proceso de clasificación y ordenación de fotografías, uno de los más importantes objetivos de la misma. En iteraciones posteriores se irán añadiendo funcionalidades a los componentes creados de manera que se terminen de satisfacer los requisitos.

Durante este capítulo se mostraron los casos de uso de la iteración quinta, así como los nuevos requisitos descubiertos y el refinamiento de los ya existentes. Se indican las decisiones tomadas durante la implementación de esta iteración y las pruebas realizadas con el objetivo de comprobar el correcto funcionamiento.

Capítulo 6

Mejora de la interfaz visual

Este capítulo trata sobre la décima y última iteración del proyecto. En ella se realiza un cambio en todo el apartado gráfico de la aplicación, de manera que será mucho más atractiva virtualmente para el usuario. En esta capítulo se muestran los nuevos requisitos, tabla 5.1. A continuación se mostrara el proceso de implementación, y finalmente las pruebas realizadas, para comprobar el correcto funcionamiento y cumplimiento de los requisitos.

Índice

6.1. Objetivos

Esta iteración corresponde a la última planificada para el desarrollo de la aplicación. En ella se pulirán todos los aspectos visuales de la aplicación. De manera que quede mucho más agradable a la vista.

Básicamente esta es una iteración cuya finalidad es cambiar la apariencia básica de la interfaz gráfica, por una más atractiva. Añadir opciones al menú superior. Inclusión de iconos que faciliten el reconocimiento de las acciones de cada botón u opciones de menú. Adición de mensajes informadores sobre cada control, que aparezcan tras dejar el cursor encima. Cambio del formato de mensajes de error que salían por consola, por unos visuales, evitando de esta manera cualquier tipo de interacción del usuario con la consola de Java.

También se cambió la apariencia de algunos Beans de manera que resultaran más atractivos visualmente, y se añadió algunas funcionalidades extra, claramente sin modificar las

ya existentes.

En esta última iteración, se corrigieron algunos comportamientos no deseados que se detectaron mientras se ejecutaba y probaba las versiones intermedias. Estos errores no corresponden a fallos en algún requisito, sino controles extras para garantizar un buen uso de la aplicación.

Por último se implementó un control de resolución. Apolo ha sido diseñado pensando en pantallas con resoluciones altas, por lo que si la pantalla tiene una resolución menor que 1024x768, Apolo mostrará un aviso informando al usuario que con esa resolución es posible que no trabaje cómodamente con Apolo; ofreciendo la posibilidad de cancelar la ejecución de Apolo, o continuar de todos modos.

6.2. Ingeniería de Requisitos

En esta sección se muestran el refinamiento de requisitos, así como la aportación de nuevos.

En esta iteración no se añade ninguna nueva funcionalidad, por lo que no se precisa de un diagrama de casos de uso. En cambio se añaden algunos requisitos y se refinan otros para lograr que la interfaz gráfica sea visualmente atractiva y con ello lograr el objetivo de esta iteración.

6.2.1. Refinamiento de requisitos

Para lograr cumplir con los objetivos de esta iteración se definieron una serie de nuevos requisitos, los cuales están reflejados en el cuadro 6.1.

Numero	Nuevo Requisito
R17	Creación de un menú superior desde el cual controlar la aplicación.
R18	Inclusión de iconos de manera que faciliten el reconocimiento de las acciones.
R19	Creación de mensajes de ayuda en cada control.
R20	Sustitución de salida de mensajes por consola a mensajes en ventana gráfica.
R21	Cambio en el aspecto visual del bean balda y paquete diapositiva.
R22	Control de resoluciones.

Cuadro 6.1: Refinamiento de Requisitos Iteración 10

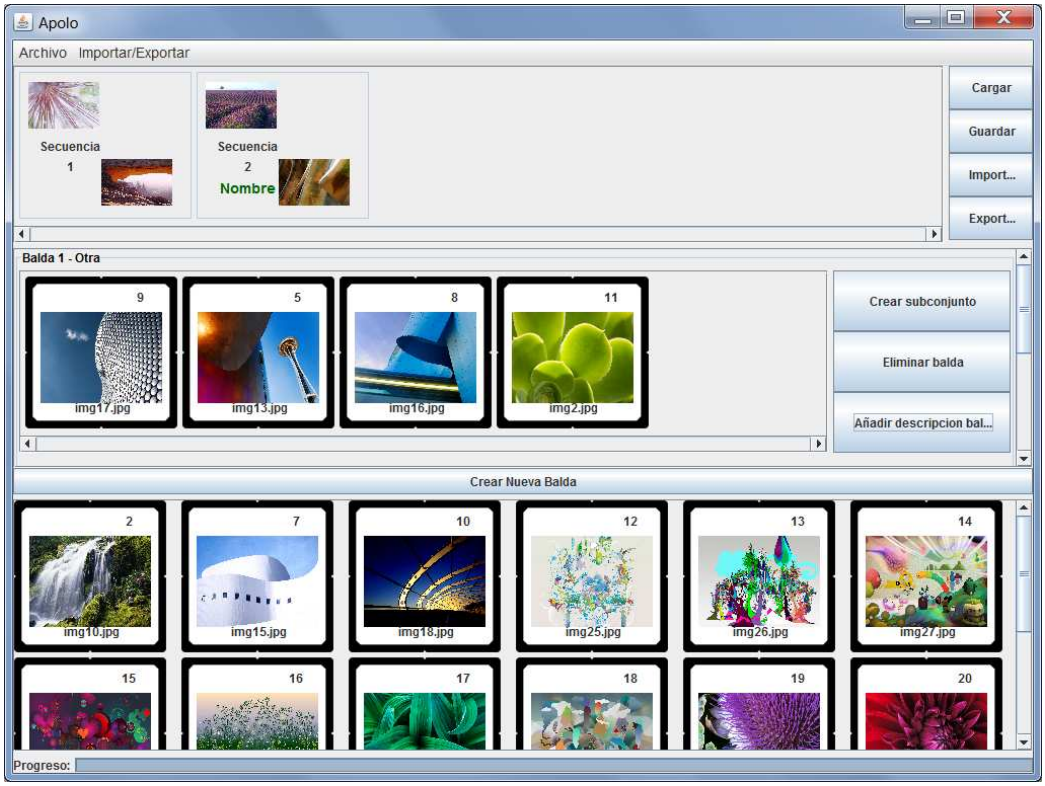

Figura 6.1: Interfaz Gráfica antes de la iteración 10

6.3. Implementación

En esta sección se describe el proceso de implementación de la décima iteración así como las decisiones tomadas durante esta etapa, de manera que se satisfagan los requisitos descritos en la sección anterior.

Al comenzar esta iteración ya se dispone de una aplicación completamente funcional pero con una interfaz pobre visualmente (ver figura 6.1). Por lo que en esta iteración sólo se limitará a mejorar el apartado visual de la aplicación.

Comenzamos por añadir opciones al menú superior, de manera que desde él, pueda controlarse la aplicación, esto es: Cargar un estado, guardar un estado, salir de la aplicación, importar fotografías y exportarlas. Como menú extra se añadió, un acceso al manual de usuario de la aplicación, un acceso a la web de la aplicación, y por último, un *Acerca de* de manera que se muestre una ventana con la autoría y el fin de Apolo.

A continuación se cambió el *Look And Feel*[1][Inc01] de la aplicación, y se eligió una denominada *Nimbus* de manera que todos los componentes visuales utilizados adquieran

[1]Apariencia de los componentes.

otra interfaz visual. Y en su conjunto dotar de un gran atractivo a la Aplicación.

Posteriormente se añadió, y en algunos casos sustituyó, iconos en el texto de botones y menús. De manera que quedara mucho más agradable visualmente. Además se añadió a todos los componentes que son susceptibles de ser manipulados una descripción que aparezca al mantener el cursor del ratón un rato sobre el mismo.

Se cambió el icono por defecto del cursor cuando se encuentre encima de una diapositiva o subconjunto ordenado, de manera que aparezca una manopla como cursor y oriente al usuario que se trata de un componente movible. De la misma forma, al ser arrastrado, la manopla cambia ligeramente, dando la sensación de agarrar la diapositiva.

Se modificó la apariencia del Bean *subsecuencia de diapositivas*, de manera que apareciese visualmente una carpeta con una serie de imágenes sobre ella e información sobre el número de secuencia y descripción. De esta manera se logra que visualmente se relacione el componente con una subsecuencia o paquete de diapositivas.

Se añadió al Bean Diapositiva más opciones a su menú contextual: Editar, actualizar y propiedades. Editar abre el programa que esté definido por defecto en sistema operativo para tal caso. Actualizar, refresca la imagen contenida en la diapositiva, por si se modificó. Propiedades, muestra una ventana con información sobre el fichero representado por la diapositiva.

A continuación se cambió la salida de mensajes de error por consola, a mensajes de error visuales a través de una ventana, de manera que se dejara de usar la consola de Java para tales fines.

Para implementar el control de resolución, antes de la carga del programa, se pregunta al sistema operativo que resolución tiene la pantalla del sistema, si la resolución no es adecuada, se mostrará una ventana informando al usuario y ofreciéndole la oportunidad de detener la ejecución de Apolo, o por el contrario continuar con la carga a pesar del aviso.

Finalmente se solucionó los pequeños bugs que aparecieron según se iba probando la aplicación en distintas manos:

- Imposibilitar un nombre en blanco en el diálogo guardar estado.
- Imposibilitar guardar un estado y no seleccionar al menos una zona.
- Mostrar mensaje de error si el sistema no tiene visualizador o editor por defecto asignado.

La interfaz resultante puede verse en la figura 6.2.

6.4. Pruebas

En esta sección se relatan las pruebas realizadas para la comprobación del cumplimiento de los requisitos descritos en la tabla 6.1.

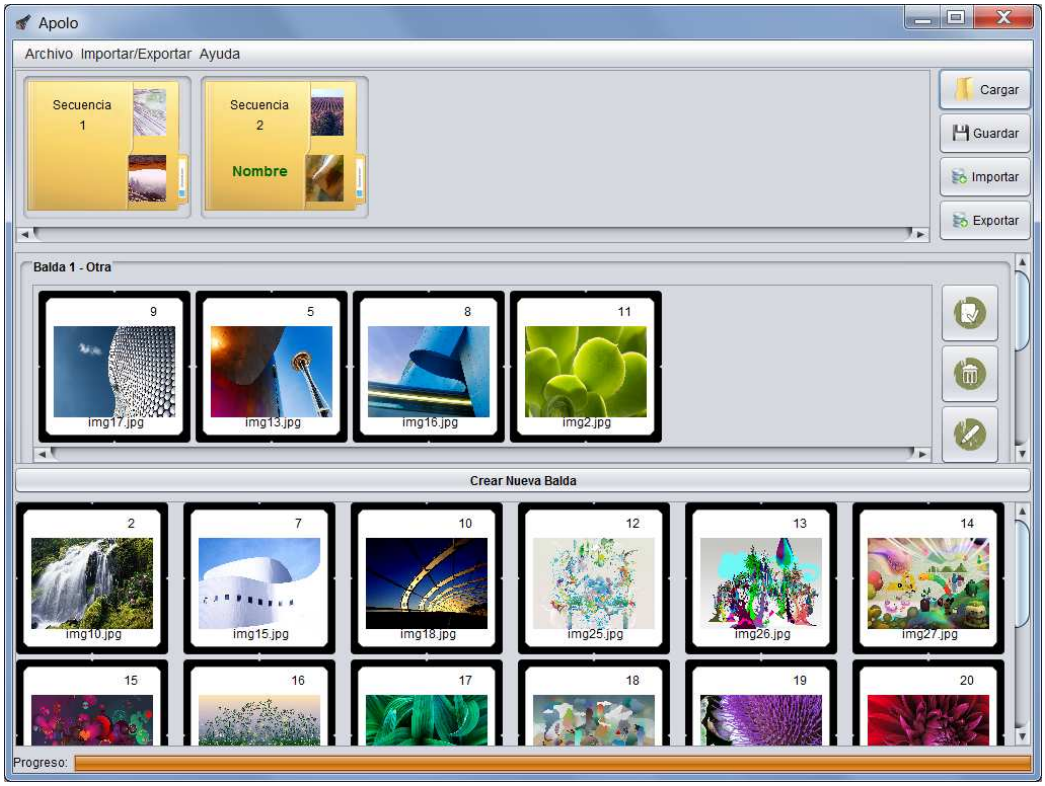

Figura 6.2: Interfaz Gráfica después de la iteración 10

Una vez concluida la fase de implementación, realizamos las pruebas necesarias para comprobar que todo funciona correctamente.

Para ello ejecutamos la aplicación y observamos si los resultados al ejecutar distintas acciones corresponden con los esperados.

1. **Acción:** Ejecutar Apolo.
   Resultado Esperado: Interfaz mejorada visualmente.
2. **Acción:** Ejecución de los menús superiores.
   Resultado Esperado: Todas las opciones funcionan correctamente.
3. **Acción:** Desplegar todos los menús y comprobar la correcta carga de los iconos.
   Resultado Esperado: Todos los iconos cargan correctamente.
4. **Acción:** Provocar mensajes de error o aviso.
   Resultado Esperado: Los mensajes de error aparecen en una ventana.
5. **Acción:** Carga de los componentes modificados.

Resultado Esperado: Nueva interfaz gráfica de componentes.

6. **Acción:** Ejecución de Apolo en pantallas de baja resolución.

   Resultado Esperado: Mensaje de aviso con la posibilidad de seguir ejecutando o detener el programa.

7. **Acción:** Provocar comportamientos no esperados.

   Resultado Esperado: Los comportamientos no esperados son controlados.

6.5. Sumario

Durante este capítulo se trató las acciones realizadas en la décima y última iteración del proyecto. En ella se cambia el aspecto visual de todos los componentes y se modifica algunos. Puede verse el aspecto anterior a esta iteración en la figura 6.1 y posterior a la iteración en la figura 6.2.

Para ello se mostraron los nuevos requisitos encontrados, las decisiones que se tomaron durante la fase de implementación y finalmente las pruebas que se llevaron a cabo para comprobar el correcto cambio visual de la aplicación.

Capítulo 7

Despliegue y Aceptación

Este capítulo trata sobre el proceso de despliegue y aceptación de la aplicación desarrollada. Para ello se describe el proceso utilizado para la creación de instaladores para los distintos sistemas operativos, así como los problemas encontrados durante el proceso.

Índice

7.1. Despliegue de la Aplicación

En esta sección se muestra como se realizará el despliegue de la aplicación en los distintos sistemas operativos, así como la creación de una web para la difusión de la aplicación por internet.

El despliegue de la Aplicación consistirá en crear un archivo *jar* ejecutable, el cual podrá ser ejecutado, en un sistema con una máquina virtual de Java instalada de una versión igual o superior a la *1.6.0*, por el usuario de manera que en ese mismo instante se arranque la aplicación Apolo.

Para que apolo funcione correctamente deberá ejecutarse teniendo, en el mismo directorio donde se localice, la carpeta *recursos*. En la cual se encuentran todos los ficheros que requiere Apolo para su correcta ejecución. Sin estos ficheros, Apolo será funcional, pero a un nivel mínimo, pues parte de su interfaz visual no será cargada.

Como licencia de uso se opta por la GPL v3, de esta manera se permite la libre distribución del software así como su estudio y libre modificación.

7.1.1. Creación de Instaladores

Para facilitar la instalación en equipos *Microsoft Windows y Linux* se crearan instaladores que facilitarán el proceso a los futuros usuarios. De esta manera dejarán de preocuparse de donde se instalan y donde se encuentran a la hora de ejecutarlos, pues el propio instalador se encargará de añadir los accesos directos que se consideren oportunos, ya sea en el menú de programas o en el escritorio.

Sistemas Microsoft Windows

Para los sistemas *Microsoft Windows* se ha utilizado la herramienta *Launch4j* que no es más que un *wrapper* que envuelve el ejecutable *jar* en un ejecutable *exe* para facilitar la ejecución del mismo en entornos *Microsoft Windows*, de manera que si en el sistema operativo no esta asociada la extensión *jar* con la maquina virtual de Java, el ejecutable *exe* le indique que debe lanzar la máquina virtual con el ejecutable embebido *jar*.

A continuación se creó un instalador con la herramienta gratuita *Inno Setup* la cual crea un archivo ejecutable con el programa a instalar así como sus archivos dependientes, todo empaquetado y comprimido en un único archivo.

Para crearlo se realizó un script en el cual se escriben las directivas que debe seguir el programa para la generación del instalador.

Además se indicó que durante esta instalación el sistema informe de la licencia de uso y hasta que el usuario no la acepte, no podrá proseguir con la instalación.

Sistemas Linux basados en Debian

Para los sistemas Linux basados en *Debian*, se creará un paquete *deb* que facilite la instalación del mismo.

Para crear el paquete se utilizó la herramienta *dpkg*[CN07] la cual permite crear paquetes e instalarlos en el sistema. El comando utilizado es el siguiente: *sudo dkpg -b deb/ apolo.deb* estando en deb todo el sistema de fichero que se desea crear, así como la configuración de la post-instalación y la post-eliminación.

Además, en los últimos sistemas operativos basados en *Debian* es posible instalar el paquete sin utilizar comandos de consola, directamente en el centro de software o haciendo doble click en el paquete, depende de la configuración del sistema.

Figura 7.1: Web de Apolo

Otros sistemas

Para el resto de sistemas se dejará a disposición un archivo *tar* en el que se encontrara el archivo ejecutable *jar* Apolo junto con la carpeta *recursos* para que una vez descomprimido y si el sistema dispone de máquina virtual de Java versión 1.6.0 o superior sea ejecutable.

7.1.2. Creación de la web

Se creó una página ***http://www.angeltezanos.com/Apolo*** para una mayor difusión de la aplicación, en la cual se puede encontrar toda la información y descargas de Apolo.

Se trató de crear una web que fuera sencilla y a la vez muy práctica, mostrando una imagen llamativa y agradable de Apolo en todas las secciones.

Se aprovechó el servicio de *Google Analytics*[Cli10] para conocer los visitantes de la web y su procedencia, así como el número de descargas realizadas.

Con el fin de facilitar el aprendizaje del software se crearon una seria de videotutoriales visibles en la web, los cuales muestran como realizar la mayoría de las tareas. De esta manera un usuario, puede conocer cómo realizar una acción sin tener que leerse el manual, o incluso

conocer el funcionamiento de la aplicación sin tener que instalarla.

7.2. Aceptación de la Aplicación

Esta sección trata sobre la implantación del software desarrollado en los equipos finales del usuario, de manera que se descubrieron nuevos problemas que hicieron modificar la aplicación.

La aplicación fue ejecutada en varios sistemas, cada uno de los cuales con configuraciones distintas, de manera que pudiera probarse en los más variados equipos que fuera posible. Se probó tanto en equipos con sistemas Microsoft Windows 7 y XP, como equipos con sistemas Linux. Cada uno de los sistemas operativos en arquitectura de 32 bits, como equipos de 64 bits.

7.2.1. Instalación en distintos equipos

La instalación en los distintos equipos se produjo con normalidad haciendo uso de los instaladores creados con anterioridad. De manera que la tarea se realizó de una manera rápida y sencilla.

Tanto en equipos Microsoft Windows como equipos Linux, la instalación del software como la desinstalación fue correcta, y en todos los casos se encontró un acceso a la aplicación en el menú de programas.

Al ejecutar el programa instalado se descubrió un nuevo problema, en sistemas Linux no cargaba la carpeta de recursos. Así que se pospuso las siguientes fases y se comenzó a analizar la situación de la aplicación, y el motivo por el cual *Apolo* no se ejecutaba como se esperaba.

Problemas encontrados

Analizando la aplicación y haciendo uso de depuradores, se descubrió que la aplicación buscaba la carpeta de recursos en una localización errónea. Sin embargo la localización de los recursos en windows resultaba correcta.

Investigando la causa, se descubrió que el problema se trataba de la distinta forma de trabajar con rutas relativas de *Microsoft Windows y Linux* pues mientras que en *Microsoft Windows* el path relativo que devuelve la máquina virtual es el directorio donde se encuentra el ejecutable *jar*, en *Linux* el path relativo que devuelve la maquina virtual de java es el directorio de usuario, normalmente */home/user*.

Solución de los problemas encontrados

Una vez localizada la causa del mal funcionamiento, se decidió abandonar el uso de rutas relativas, y cambiarlas por absolutas, de manera que nos saltásemos la falta de convención de las distintas implementaciones de la máquina virtual de java para los distintos sistemas.

El problema de trabajar con rutas absolutas es que los programas no pueden ubicarse en otro directorio de donde se definió su uso, ya que las rutas absolutas dejarían de ser correctas. Por lo que teníamos otro problema.

Finalmente se decidió trabajar con una solución mixta, se trabajará con rutas absolutas, pero se definen en la inicialización del sistema. Es decir, se creó una rutina que devuelve el directorio donde se encuentra el fichero ejecutable *jar* y a partir de esta ruta se construye el path donde se encuentra el resto de recursos.

Una vez implementados los cambios, se comprobó que efectivamente tanto en sistemas Linux como Windows, la ejecución del programa era correcta, así como la carga de todos los recursos.

7.2.2. Uso por usuarios finales

Apolo fue probado por usuarios externos al equipo de desarrollo. Rápidamente se adaptaron al uso de la aplicación, sin resultarles compleja su utilización. La aplicación respondió como se esperaba, y ofreció una interfaz gráfica agradable y amistosa para el usuario, sin resultarle, de ninguna manera, complejo su uso.

Los usuarios se mostraron contentos, por disponer finalmente de una herramienta liviana, destinada al único fin de organizar y clasificar fotos, y que renombrase todas las fotografías como ellos desearan.

Petición de los usuarios

Algunos usuarios, nos indicaron que les gustaría que apolo recordase la carpeta última desde donde se importó. Es decir que una vez importadas fotografías, si deseasen volver a importar, directamente les mostrara el directorio de la anterior importación, y no el de por defecto.

Atendiendo a sus peticiones, se realizó los cambios en apolo pertinentes para añadirle esta funcionalidad, la cual aporta mayor comodidad y usabilidad a la aplicación para los usuarios finales.

7.3. Sumario

Durante este capítulo se describieron los procesos de creación de los distintos instaladores necesarios para los sistemas operativos donde se ejecute. A continuación se explica

la creación de la página web *www.angeltezanos.com/Apolo* para ayudar a la difusión de la aplicación.

Finalmente se trata la implantación en los sistemas finales y los problemas encontrados durante la misma, que obligaron a la modificación de ciertas partes de la aplicación.

Capítulo 8

Conclusiones y Trabajos Futuros

Como parte final de la memoria, se muestra las conclusiones del proyecto así como posibles proyectos futuros.

Índice

8.1. Conclusiones

En este proyecto de fin de carrera se ha implementado una aplicación denominada Apolo, la cual es capaz de organizar y clasificar fotografías de manera sencilla, rápida y con una interfaz de usuario amigable. El resto de acciones que permite la aplicación (visualizar y editar fotografías) se delegan en los programas que por defecto tenga instalado el sistema operativo para tal fin. Gracias a ello hemos logrado una aplicación altamente útil y con un consumo de recursos bajo en comparación con otros clasificadores de fotografías, que sin duda alguna resultan mucho más *pesados*.

La aplicación permite importar fotografías, y moverlas una a una, hacia una serie de baldas, en las cuales será depositadas y ordenadas, según el orden que desee el usuario. Una vez ordenado un subconjunto de ellas, podrán ser empaquetadas en subsecuencias de manera que estas subsecuencias también pueden reordenarse entre ellas.

Una vez se desee exportar el conjunto de subsecuencias, el usuario podrá elegir entre un nombre común para todas las fotografías más una numeración, o sólo una numeración. El propio programa se encargará de mostrar la numeración correspondiente, utilizando tantos ceros como sean necesarios, para garantizar una ordenación correcta en cualquier sistema.

El programa también permite guardar el estado de la aplicación en un fichero, de manera que más adelante sea capaz de recuperarlo y continuar con el trabajo. Al guardar el estado,

deberemos elegir el nombre del fichero y el directorio donde deseamos que se guarde. También nos ofrece la posibilidad de elegir que deseamos guardar: La zona superior donde se almacenan los subconjuntos de diapositivas ordenadas; la zona central, donde se encuentran la estantería con sus baldas; o la zona inferior o mesa, con todo el pool de diapositivas.

Durante la realización del proyecto fin de carrera he comprobado la importancia que es tener una buena planificación en el proyecto, dedicar el tiempo necesario para el análisis de los requisitos así como el diseño. Para, de esta manera, poder desempeñar la actividad de implementación lo más rápidamente posible y de una manera fiable, que más tarde supere las pruebas y cumpla con los requisitos esperados.

Gracias a la metodología usada, hemos ido en cada iteración añadiendo más funcionalidades al sistema, de manera que como resultado final de las iteraciones contábamos con un prototipo con en el que analizar el buen rumbo de la aplicación.

8.2. Trabajos Futuros

En esta sección se muestran los posibles trabajos futuros que se podrían realizar si la aplicación tiene la aceptación adecuada por parte de los usuarios.

En el futuro más próximo se creará instaladores para los sistemas Mac, de manera que la aplicación sea fácil de instalar en el prácticamente 100 % de los sistemas operativos de escritorio.

A medio plazo se añadirá la posibilidad de exportar las diapositivas ordenadas y clasificadas en formato de video, de manera que pueda ser visualizadas como si de una película se tratara. También se implementará soporte para otros formatos de imagen, en concreto RAW.

En un futuro a largo plazo y si la aplicación tiene aceptación por parte de los usuarios, está pensado crear un módulo que permita a la aplicación, exportar las fotografías a través de la red. Es decir, la funcionalidad buscada es que un usuario con un conjunto de subsecuencias ordenadas, pueda en un momento dado dejar un socket abierto en el sistema a la espera de peticiones, y que otro usuario con Apolo se pueda conectar a dicho socket y se transfieran las fotografías. De esta manera seria posible intercambiar fotografías sin tener que esperar a ver a nuestro amigo con las fotografías y a nosotros con el USB a mano.

Apéndice A

Contenidos del CD

En este capítulo se muestra el contenido del CD adjunto a esta memoria, el cual aporta contenido multimedia así como instaladores y código fuente.

El CD contiene los siguientes elementos:

- **Código Fuente:** Código Fuente de la aplicación desarrollada.
- **Instaladores:** Instaladores de la aplicación para los distintos Sistemas Operativos.
- **Manual de usuario:** Manual de usuario de la aplicación.
- **Memoria del proyecto:** Memoria del proyecto en formato *pdf*.
- **VideoTutoriales:** Videotutoriales con las acciones más comunes de la aplicación.
- **Web:** Página Web de la aplicación.

Bibliografía

[Cli10] Brian Clifton. *Advanced Web Metrics with Google Analytics*. 2º Edition, 2010.

[CN07] Francois Caen Christopher Negus. *Ubuntu Linux Toolbox: 1000+ Commands for Ubuntu and Debian Power Users*. 1º Edition, 2007.

[CSH04] Gary Cornell Cay S. Horstmann. *Core java*, volume 2 - Caracteristicas Avanzadas. 7º Edition, 2004.

[EB09] Craig Stinson Ed Bott, Carl Siechert. *Windows® 7 Inside Out*. 1º Edition, 2009.

[Gea98] David M. Geary. *Graphix Java Mastering the JFC*, volume 1 - AWT. Prentice Hall, 1998.

[Gea99] David M. Geary. *Graphix Java Mastering the JFC*, volume 2 - Swing. Prentice Hall, 1999.

[Hol04] Steve Holzner. *Eclipse*. 1º Edition, 2004.

[Inc01] Sun Microsystems Inc. *Java(TM) Look and Feel Design Guidelines: Advanced Topics*. 1º Edition, 2001.

[JC00] John Daniels^{o} John Cheesman. *UML Components: A Simple Process for Specifying Component-Based Software*. 1º Edition, 2000.

[JM05] Joshua Marinacci Joshua Marinacci. *Swing Hacks: Tips and Tools for Killer GUIs*. O'Reilly Media, 2005.

[KW04] Alison Huml Sharon Zakhour Kathy Walrath, Mary Campione. *The JFC Swing Tutorial: A Guide to Constructing GUIs*. 2004.

[Lar03] Craig Larman. *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley Professional, 2003.

[MH10] Paul Hudson Matthew Helmke, Andrew Hudson. *Ubuntu Unleashed 2011 Edition: Covering 10.10 and 11.04*. 6º Edition, 2010.

[PC98] Douglas Kramer Patric Chan, Rosanna Lee. *The java Class Libraries*, volume 2. Second Edition, 1998.

[Sch06] Herbert Schildt. *Swing: A Beginner's Guide (Beginner's Guide (Osborne Mcgraw Hill))*. 1º Edition, 2006.

[SO04] Henry Wong Scott Oaks. *The java Threads*. Third Edition edition, 2004.

[Szy11] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. (2nd Edition), 2011.