TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INFORMÁTICA

Desarrollo de un Sistema de Recuperación de Imágenes para Plataformas Móviles

Autor
Alejandro Casado Quijada

Director
Jesús Chamorro Martínez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Resumen

Este proyecto busca permitir al usuario realizar consultar a un sistema de recuperación de imágenes, visualizando las imágenes resultantes de dichas consultas de una manera atractiva, interactiva y amigable. En esta ocasión dicho sistema ha sido desarrollado para plataformas móviles, concretamente *Android*.

Cuando hablamos de un sistema de recuperación de información, *CBIR*, nos referimos a los sistemas basados fundamentalmente en descriptores de bajo nivel (color, textura, etc.) obtenidos directamente a partir de la imagen. En ellos la idea es, mediante una imagen consulta, comprobar como de parecidas son el resto, imágenes resultado, y presentar los resultados

Al tratarse de una plataforma móvil hay que tener muy en cuenta los recursos que va a requerir dicho sistema, por lo que hay que esta especialmente atentos a ellos, ya que si el sistema necesita demasiados recursos puede funcionar incorrectamente y provocar incluso malfunciomaniento del propio teléfono.

Se busca interactividad por parte del usuario, por ello será capaz de moverse a través de las imágenes, tanto consulta como resultado, realizando movimientos de *scroll*. A su vez, se ha añadido mecanismos de ayuda, para que el usuario sepa en cada momento en que lugar se encuentra, ya que el resultado de una consulta puede ser de cientos de imágenes. Por otro lado, será capaz de modificar ciertos parámetros, como descriptor asociado, número de imágenes, para adecuar el uso del sistema a su experiencia deseada.

Todo lo descrito se va a desarrollar a partir de un CBIR concreto, Java Multimedia Retrieval©.

Para realizar la planificación se realizaron una serie de reuniones iniciales con mi tutor en las cuales se acordaron los objetivos y elementos que debía de tener este proyecto. Estos objetivos se dividieron entre una serie de semanas. Pero en todo momento sabía lo que debía de realizar. Por lo que cada pocas semanas realizábamos reuniones para

comprobar si dichos objetivos acordados y planificados se cumplían, a su vez discutíamos detalles secundarios del proyecto, como leves mejoras en la interfaz.

Para llevar a cabo la planificación se ha usado la herramienta conocida como diagramas de Gantt, en el que se especifican los objetivos a cumplir en un periodo concreto de tiempo. Este diagrama se detallará más adelante en su correspondiente sección.

Para la realización de este proyecto se partía de una situación que podría considerarse prácticamente desde 0, debido a que no hay muchos proyectos relacionados. También hay que tener en mente el problema de lidiar con una plataforma móvil, ya que sus recursos son limitados y no tan potentes como los de un computador. A su vez, hay que cuidar el tiempo de ejecución de las consultas, pues si es demasiado elevado, provocaría que el usuario dejara de usar la aplicación.

Teniendo en cuenta lo descrito anteriormente, el resultado del proyecto ha sido satisfactorio. La aplicación resultante nos permite realizar consultas sobre las imágenes del teléfono, pudiendo elegir la imagen consulta desde la galería o desde la cámara del propio teléfono. Los tiempos de cálculo de consulta son más que aceptables, teniendo en cuenta que tras un primera consulta, se almacenan los resultados en una pequeña base datos, esto se detallará detalladamente más adelante. Por otro lado, el usuario puede editar ciertos parámetros de la consulta, lo que hace que la aplicación sea ajustable a las necesidades del usuario en cualquier momento.

Mencionar brevemente, que las conclusiones del trabajado han sido satisfactorias. El proyecto se ha llevado a cabo correctamente, cumpliendo todos los objetivos establecidos, se ha cubierto un hueco de mercado que estaba prácticamente vacío, los *CBIR* para *Android*, son puramente académicos y sus interfaces son demasiado simples.

A su vez, se especifican dos posibles trabajos futuros:

- Trabajar con código nativo usando *Android ndk*
- Mostrar estadísticas apoyándose en conjuntos difusos.

Capítulo 2

Introducción

2.1 Contextualización

Debido al avance de la tecnología, cada vez disponemos de más dispositivos con la capacidad de capturar imágenes. Esto ha provocado un importante incremento en las bases de datos de imágenes, lo que a su vez ha propiciado la aparición de metodologías para solventar el problema de la manipulación, gestión y recuperación de dicha información. En este caso cuando hablamos de metodologías nos referimos a los sistemas de recuperación de información, basados fundamentalmente en descriptores de bajo nivel (color, textura, etc.) obtenidos directamente a partir de la imagen. Estos sistemas se denominan CBIR.

La recuperación de información es la actividad de obtener recursos de información relevantes para una necesidad de información a partir de una colección de recursos de información. Las búsquedas pueden ser basadas en texto, o en otro tipo de indexación basada en contenido. Para realizar esta actividad se utilizan los sistemas de recuperación de información, que se encargan de gestionar todo lo necesario.

Un proceso de recuperación de información comienza cuando un usuario realiza una consulta al sistema. Estas consultas son declaraciones de necesidad de información, por ejemplo, buscar una frase en un motor de búsqueda. Las consultas no identifican a un único objeto de la base de datos o colección, si no que afecta a varios con distintos grados de relevancia.

Un objeto, en este ámbito, es una entidad que representa la información en una colección o base de datos. Sin embargo, al contrario de lo que pasaría en una consulta a un sistema clásico SQL, los resultados son clasificados. Esta clasificación es la principal diferencia respecto a las búsquedas en bases de datos tradicionales.

Hay que tener en cuenta que dependiendo de la aplicación, los objetos pueden ser textos, imágenes, audio o vídeos, por ejemplo. La mayoría de los sistemas de recuperación

de información manejan una puntuación numérico sobre como coincide cada objeto de la base de datos con el objeto consulta, y ordena los objetos acorde a este valor numérico. Los que se encuentran en las primeras posiciones del ranking se muestran al usuario.

Para realizar las consultas, los sistemas de recuperación utilizan los denonimados descriptores. Estos describen las características de los objetos que se encuentran en la colección de objetos accesibles por el sistema. Dado que en este trabajo nos vamos a centrar en un sistema de recuperación de información, vamos a hablar sobre sus mútliples descriptores, denominados descriptores visuales.

Estos descriptores representan características visuales de los objetos, siendo estos vídeos o imágenes. Describen características elementales como la forma, el color, o la textura, entre otros muchos.

Los descriptores visuales se encuentran divididos en dos grandes grupos:

- **Descriptores de información general**. Proporcionan descripciones sobre el color, formas, regiones y texturas.
- **Desciptores de información de dominio específico**. Proporcionan información sobre sucesos que van apareciendo en escena. El reconocimiento de caras sería un ejemplo delimitado de este tipo.

2.1.1 Descriptores de información general

Como se ha mencionado, estos descriptores se encargan de representar distintas propiedades visuales elementales, como el color y la textura.

Color

Puede ser considerada como el atributo elemental del contenido visual. A continuación se especifican cinco formas para describir el color. Las tres primeras simbolizan la distribución del color. Las otras dos describen la distribución espacial del color y la relación del color en un conjunto o secuencia de imágenes.

- Dominant Color Descriptor (DCD)
- Scalable Color Descriptor (SCD)
- Color Structure Descriptor (CSD)
- Color Layout Descriptor (CLD)
- Group of frame (GoF) o Group-of-pictures (GoP)

Textura

Esta propiedad se encuentra desarrollada para caracterizar las regiones, o texturas de una imagen. Esto se lleva a cabo teniendo en cuenta la homogeneidad de las regiones y los histogramas de los bordes de esas regiones. Como representantes de este conjunto tenemos:

- Homogeneous Texture Descriptor (HTD)
- Texture Browsing Descriptor (TBD)
- Edge Histogram Descriptor (EHD)

Forma

En esta ocasión, estamos hablando de información semántica de gran relevancia, ya que los objetos son reconocidos visualmente por los seres humanos. La única manera de obtener dicha información es mediante la segmentación de la imagen. Desafortunadamente, aún no se encuentra completamente disponible, pero si disponemos de una serie de algoritmos que nos ofrecen buenas aproximaciones. Estos descriptores detallan formas, regiones y contornos, si hablamos de imágenes 2D, y volúmenes en el caso de 3D. Estos descriptores son:

- Region-based Shape Descriptor (RSD)
- Contour-based Shape Descriptor (CSD)
- 3-D Shape Descriptor (3-D SD)

Como se ha mencionado anteriormente, nos vamos a centrar en los sistemas de recuperación de imágenes, por lo que vamos a ver una serie de ejemplos sobre ellos.

2.1.2 CBMIR

Este CBIR es usado en ámbitos médicos, de ahí su nombre Content-based medical image retrieval, ha sido desarrollado por la universidad nacional de Malasia. Se centra en ayudar a diagnosticar irregularidades vertebrales, lo cual es muy importante si se realiza a tiempo para disminuir el riesgo de sufrir fracturas vertebrales. Las fracturas aparecen en radiografías, pero su detección es poco frecuente por parte de médicos y radiólogos. Por esta razón, el objetivo de este CBIR es ayudar al personal sanitario a detectar estas irregularidades como se ha comentado anteriormente.

Una vez definido este CBIR, veamos como visualiza la información de una consulta.

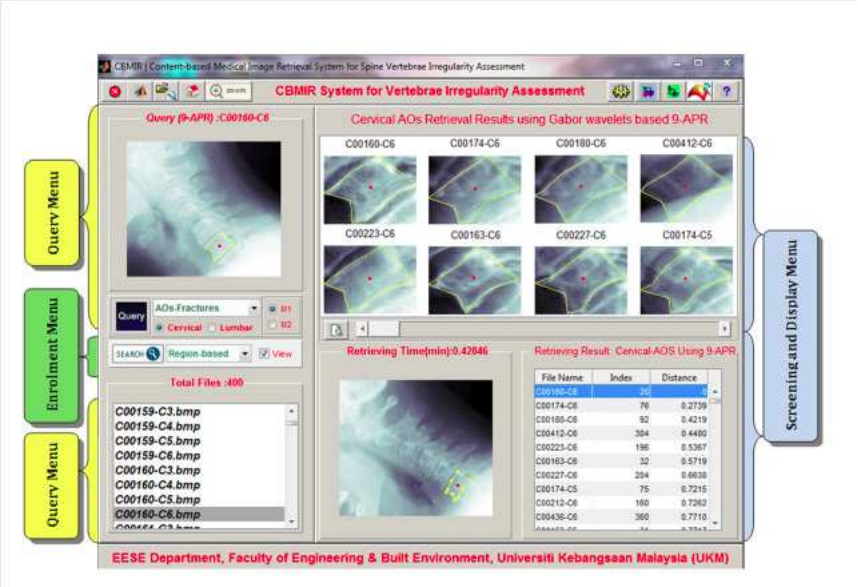

Figure 2.1: Sistema de recuperación de imágenes CBMIR

La visualización se realiza en la parte derecha de la pantalla. En la parte superior se aprecian todas las imágenes resultantes de la consulta, mientras que en la parte inferior podemos obtener mas información sobre cada una de esas imágenes.

Mientras que para la consulta se utiliza la parte izquierda. En ella se puede seleccionar el descriptor y la imagen a usar.

2.1.3 Visual Clustering of Image Search Results

La visulaización de ese sistema implementado por Trystan G. Upstill, Rajehndra Nagappan y Nick Craswellb se basa a su vez en un modelo primavera desarrollado por Olsen y Korfhage. Esta técnica fue adaptada de RadViz. En este sistema, RadViz, los puntos de referencia se distribuyen en torno de un círculo, mientras que los elementos de datos están distribuidos en el círculo según su atracción a los puntos de referencia.

En esta imágen se puede apreciar lo explicado:

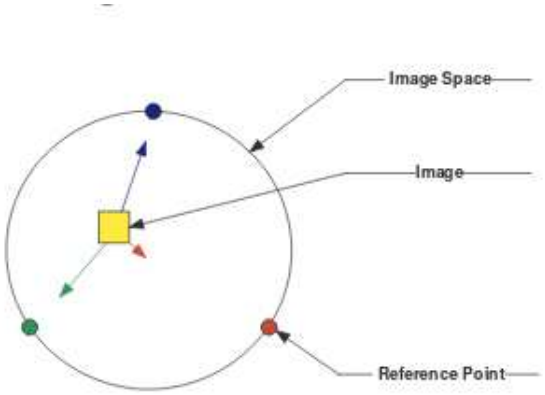

Figure 2.2: Puntos de atracción VCISR

Ahora veamos la representanción visual de una salida producida por una consulta.

Figure 2.3: Salida VCISR

Podemos apreciar como se distribuyen las imágenes a lo largo del círculo siguiendo el patrón comentado anteriormente.

Por otro lado podemos ver lo que ocurre al hacer zoom sobre una región. Al hacerlo apreciamos nuevas imágenes que permanecían ocultas.

2.2 Fundamentos

Este trabajo se basa en un sistema de recuperación de imágenes existente llamado *Java Multimedia Retrival*, *JMR*. Se trata de un *CBIR* de código libre cuyo desarrollador principal e impulsor es el profesor Jesús Chamorro Martínez del departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Se va hablar durante esta memoria sobre el espacio de color, por lo que es necesario darle una definición para saber a que nos referimos concretamente.

Por espacio de color nos referimos a un sistema de interpretación de color, una manera específica estructurar los colores de una imagen o vídeo. También es necesario hablar sobre los modelos de color, que se tratan de modelos matemáticos abstractos que describen la forma en la que cual los colores pueden ser representados como tuplas de números, normalmente como tres o cuatro valores o componentes de color, un ejemplo de esto sería RGB, en el que un color se descompone en tres valores, el valor de rojo, verde y azul.

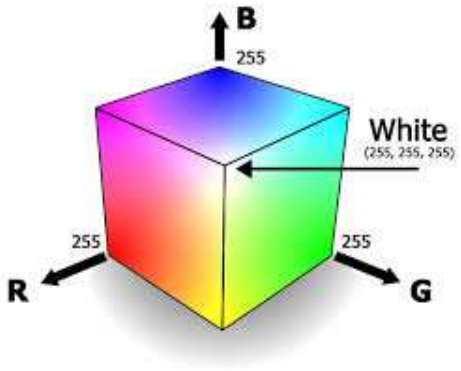

Figure 2.4: Modelo de color RGB

Otro ejemplo de espacio de color sería el *HMMD*. Este espacio de color (Hue-Max-Min-Diff), es cercano a un espacio de color perceptualmente uniforme. Los nombres componentes, correspondientes a los distintos nombres, se pueden calcular a partir de RGB siguiente las siguientes transformaciones:

- **Max**: máximo(R,G,B)
- **Min**: mínimo(R,G,B)
- **Diff**: Max-min

Por otro lado, incluso se puede definir otro componente, llamado Sum = (Max+Min)/2.

En total habría una cantidad de 5 componentes en este espacio de color. Sin embargo, un cojunto de 3 elementos, H,Max,Min o H,Diff,Sum, es suficiente para para formar este espacio de color y especificar un punto de color.

Hue, tiene la misma propiedad que su equivalente en el espacio de color HSV. Hue se mueve en el rango [0º, 360º]. *Max* se mueve en el rango [0,1] y especifica cuanto color negro se encuentra presente, dando la sensación de sombra u oscuridad. *Min*, rango [0,1], especifica cuanto color blanco se encuentra presente, dando la sensación de blanqueado. *Diff*, rango [0,1], especifica como de cercano es un color a los colores puros, dando la sensación de tono o colorido. Finalmente, *Sum*, especifica el brillo del color.

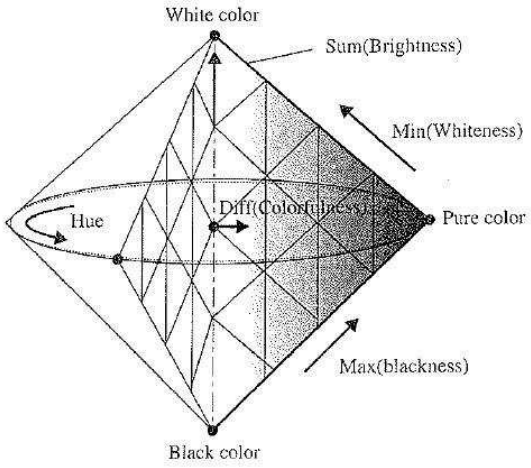

Figure 2.5: Modelo de color HMMD

```
if( Max == Min ) Hue=0; /* It is achromatic color */
otherwise:
if( Max == R && G >= B )
  Hue = 60*(G-B)/(Max-Min)
else if( Max == R && G < B )
  Hue = 360 + 60*(G-B)/(Max-Min)
else if ( G == Max )
  Hue = 60*(2.0 + (B-R)/(Max-Min))
else
  Hue = 60*(4.0 + (R-G)/(Max-Min))
```

Figure 2.6: Cáculo Hue HMMD

Para este proyecto van a ser implementados una serie de descriptores. Estos descriptores se corresponden al tipo información general, previamente comentado, concretamente sobre el color. Estos son:

- Single Color Descriptor
- MPEG7ColorStructure

2.2.1 Single Color Descriptor

Este descriptor de color se basa en el color medio de la imagen. Para calcularlo recorre una imagen en el espacio de color RGB pixel a pixel, obteniendo sus valores R, G y B. Cuando termina de recorrer todos los píxeles, calcula el color medio.

2.2.2 MPEG7ColorStructure

MPEG7ColorStructure es un descriptor de color que captura el contenido del color (similar a un histograma del color) e información sobre la estructura de este contenido. Su principal funcionalidad es la comparación imagen-imagen y se usa en la recuperación de imágenes, donde puede consistir una imagen de un solo marco rectangular o de forma arbitraria, posiblemente desconectado, regiones. El método de extracción incrusta información de estructura de color en el descriptor teniendo en cuenta todos los colores en un elemento estructurado de 8x8 píxeles que se desliza sobre la imagen, en lugar de considerar cada píxel por separado. A diferencia del histograma de color, este descriptor puede distinguir entre dos imágenes en las que un determinado color está presente en cantidades idénticas pero donde la estructura de los grupos de píxeles que tienen ese color es diferente en las dos imágenes. Los valores de color son dados por el ColorSpace MMD que se cuantifica de forma no uniforme en *bin*, recipientes, de tamaño 32, 64, 128 o 256. Cada valor de amplitud de *bin* está representado por un código de 8 bits. Este descriptor proporciona funcionalidad adicional y rendimiento en la recuperación de imágenes, basado en la similitud similitud, prsentando una mejora en comparación con el histograma de color ordinario.

2.3 Objetivos

Como todo proyecto que se realiza, este ha de tener una serie de objetivos que justifiquen su elaboración. Por lo que en este apartado se van a discutir los objetivos de este, diviéndose en objetivos principales y secundarios.

Los principales, son los objetivos que el proyecto debe cumplir completamente. Mientras que los secundarios son objetivos que el proyecto no debe cumplir integramente, pero que en caso de hacerlo, suponen un gran valor añadido al proyecto. Cabe destacar que en este caso, los objetivos secundarios han sido satisfechos en su totalidad.

2.3.1 Objetivos principales

- Desarrollar un sistema de recuperación de imágenes para plataformas móviles, para la plataforma Android concretamente. Actualmente el número de CBIR no es muy grande y no son muy conocidos por los usuarios, por lo que puede verse como una gran oportunidad.

- Este sistema debe ser capaz de permitir al usuario elegir la imagen consulta de su galería, o mediante la cámara del dispositivo, permitiendo realizar la consulta con dichas imágenes.
- Los resultados deben ser obtenidos en un periodo de tiempo aceptable. Esto ha de ser así, ya que en caso contrario, la experiencia del usuario se resentiría, lo que podría traducirse en un abandono de la aplicación.

2.3.2 Objetivos secundarios

Una vez desarrollados los principales objetivos del proyecto, se explicarán los secundarios:

- Utilizar una base de datos para almacenar los resultados. De esta manera, una vez calculada la primera consulta, el tiempo del resto se reduce drásticamente, lo que se traduce en una mejor experiencia para el usuario.
- Permitir al usuario modificar ajustes de la consulta. Logrando esto, el usuario puede adecuar la consulta a sus necesidades concretas, mejorando su experiencia con la aplicación.

Capítulo 3

Planificación

Después de aclarar los objetivos del proyecto y requisitos del mismo, es tiempo de realizar la planificación.

Para realizar la planificación del proyecto disponemos de múltiples alternativas, pero en este caso la técnica a usar son los diagramas de Gantt. Un diagrama de Grantt puede ser considerado una herramienta visual en la cual se ve reflejado el tiempo que va a ser empleado en cada una de las tareas del proyecto.

La planificación de este proyecto ha sido dividida en varios bloques que serán comentados a continuación:

3.1 Primer acercamiento

Este bloque ha sido dedicado a un estudio preliminar sobre nociones básicas de *Android*, para garantizar que las bases del proyecto fuesen correctas y así asegurarse de partir de una buena base. Por otro lado, una vez terminado dicho estudio, se empleó el resto del tiempo en asegurar la posibilidad de elegir la imagen consulta usando la galería, o mediante la cámara del dispositivo, cosa que puede complicarse si no se realiza correctamente.

Como detalle, mencionar que mi experiencia con *Android* era poca, por lo que este primer acercamiento ha sido de gran importancia, ya que ha sentado las bases del proyecto.

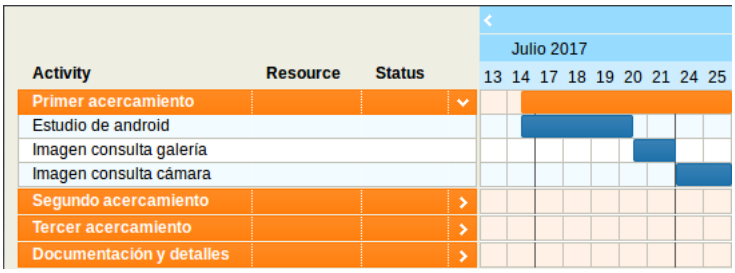

Figure 3.1: Diagrama Gantt. Primer acercamiento

El proyecto fue comenzado el 15 de julio. Los primeros días fueron empleados en el estudio de *Android*, punto fundamental para empezar el proyecto. El resto de días se usaron para posibilitar la elección de la imagen consulta a través de la cámara o galería, gestionando los permisos de una forma adecuada.

3.2 Segundo acercamiento

Una vez establecido el punto de partida e instalado todo lo necesario pasamos al segundo bloque.

En este bloque se realizó un análisis para establecer cuales eran los requisitos necesarios para el correcto funcionamiento del proyecto. Por otro lado, se desarrolló una interfaz prototipo, en la cual era posible seleccionar la imagen consulta, como hemos comentado anteriormente. Las imágenes consulta se mostraban en la parte superior de la pantalla, mientras que las imágenes resultado se mostraban en la parte inferior.

También se desarrolló un descriptor prototipo, para comprobar que realmente podíamos trabajar bien con las imágenes en *Android*. Dicho descriptor únicamente cogía los píxeles de las imágenes y hacía cálculos triviales. Con esto, se estableció la estructura que debían tener los descriptores que se implementaron en el siguiente bloque.

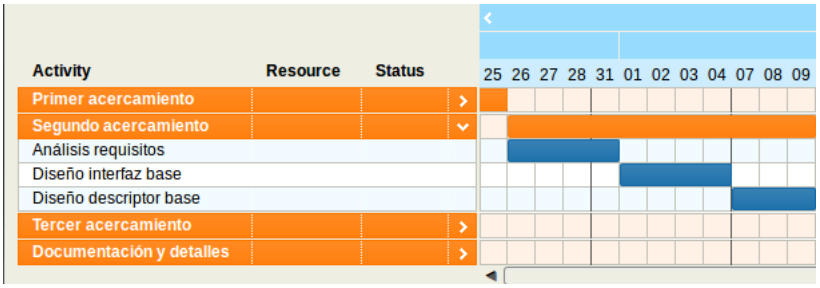

Figure 3.2: Diagrama Gantt. Segundo acercamiento

En este bloque el tiempo ha sido repartido de manera casi equitativa entre el análisis de requisitos, el diseño de la interfaz prototipo y del descriptor prototipo. Cosa crucial

para la correcta realización del proyecto, ya que, aparte de implementar los prototipos, todos los requisitos deben estar establecidos antes de avanzar en el proyecto.

3.3 Tercer acercamiento

Este es el bloque al cual se le ha dedicado más tiempo en este proyecto, debido a que es en el que se ha procedido a la implementación de la interfaz final, de los descriptores y de la base de datos.

La interfaz sigue un estilo similar a la prototipo, con las imágenes consulta en la parte superior y en la inferior las imágenes resultado de la consulta. A su vez cuenta con otros apartados como ajustes e información adicional. Todo esto se comentará con detalle en su correspondiente sección.

Por el momento se han implementando dos descriptores:

- Media de color
- Color estructurado

El primero se basa en el color medio de las imágenes, mientras que el segundo utiliza el histograma. Como en el caso de la interfaz, esto se detallará con más detalle.

Finalmente, después de todo lo anterior se decidió a implementar una pequeña base de datos que almacenase los valores obtenidos durante las consultas, de esta manera, en lugar de calcular el color medio o el histograma, se consultará en la base de datos y se obtendrá su valor, lo que reduce el tiempo de consulta drásticamente.

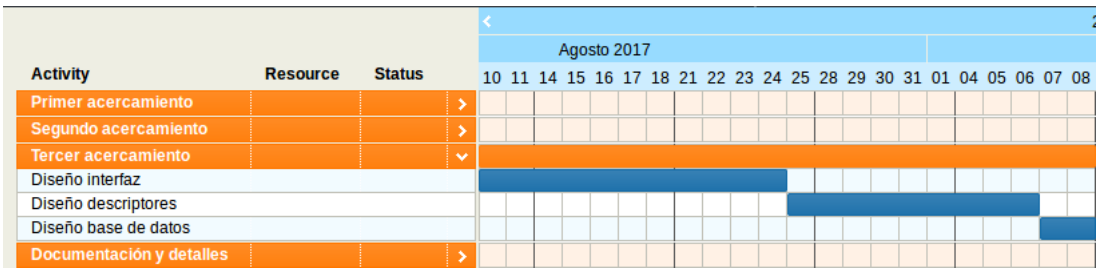

Figure 3.3: Diagrama Gantt. Tercer acercamiento

3.4 Documentación y detalles

En este último bloque se ha prodecido a la realización de esta memoria y a su vez a la revisión de todo el proyecto en busca de erratas y malfuncionamiento.

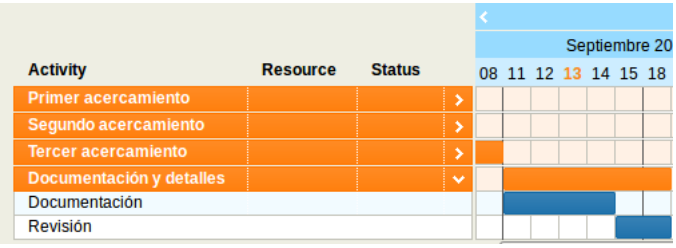

Figure 3.4: Diagrama Gantt. Documentación y detalles

Tras terminar todo el proyecto es muy importante revisarlo minuciosamente con el objetivo de encontrar fallos o elementos que se encuentren mal explicados. Por eso este bloque es gran importancia a pesar de ser el último.

Capítulo 4

Análisis

4.1 Introducción al problema

Nos encontramos ante la siguiente situación: Debemos permitir al usuario realizar consultas sobre las imágenes de su dispositivo, tomando como imagen consulta la obtenida desde la cámara o desde la galería.

Los valores proporcionados por cada descriptor se encuentran en el rango 0-1. Los cálculos para determinar la posición se realizarán con esos datos. Para ello, se colocarán de izquierda a derecha, formando un total de 4 columnas, siguiendo un orden lógico y natural que es entendido rápida y claramente por el usuario.

Dado el número de imágenes y el tamaño de pantalla de los dispositivos, no todas podrán ser vistas a la vez ni, por lo que el usuario debe ser capaz de desplazarse por las imágenes resultado contando con guas que le indiquen el orden de las diferentes imágenes mostradas.

También deberá ser posible obtener información de las imágenes mostradas con el fin de comprender mejor la visualización y su vez comprobar el correcto funcionamiento de los descriptores.

El proyecto está pensado por el momento sólo para smartphones, debido a falta de medios, por lo que la posibilidad de girar pantalla ha sido desactivada. Como trabajo futuro, se realizarán los cambios oportunos para que pueda ser utilizado en tablets.

4.2 Especificación de requisitos

Por requisitos podemos entender: Condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado.

Como en todo proyecto de software, los requisitos deben ser especificados antes de empezar el desarrollo del mismo.

En esta sección vamos a detallar cada uno de los distintos requisitos necesitados agrupados en requisitos funcionales, no funcionales y de información.

4.2.1 Requisitos funcionales

Los requisitos funcionales se encargan de especificar cómo se realizará la interacción entre el sistema y su entorno, indicando los servicios que ha de tener el sistema o cómo responderá ante ciertos estímulos.

Este proyecto cuenta con los siguientes requisitos funcionales:

RF-1. Imágenes: Gestión de imágenes.

RF-1.1. Se debe permitir cargar cualquier imagen desde cámara o galería como imagen consulta.
RF-1.2. Se debe permitir cargar cualquier número de imágenes para su consulta.
RF-1.3. La elección de la imagen consulta ha de ser sencilla.

RF-2. Descriptores: Gestión de descriptores.

RF-2.1. Ha de ser posible la elección de cualquier descriptor disponible.
RF-2.2. La consulta ha de realizarse con únicamente con el descriptor seleccionado.

RF-3. Base de datos: Gestión de base de datos.

RF-3.1. El usuario debe poder precalcular la base de datos antes de realizar cualquier consulta.
RF-3.2. El usuario debe poder eliminar la base de datos, en caso de que lo considere necesario.

RF-4 Interacción Gestión de la interacción.

RF-4.1. El usuario debe poder interactuar a través de la pantalla.
RF-4.2. El usuario debe ser capaz de desplazar las imágenes consulta de izquierda a derecha y viceversa.
RF-4.3. El usuario debe ser capaz de desplazar las imágenes resultado de arriba hacia abajo y viceversa.
RF-4.4. El usuario debe poder ver las imágenes a tamaño real al pulsar sobre ellas.
RF-5 Información adicional Gestión de la información adicional.

RF-5.1. El usuario debe poder consultar información sobre el desarrollo del proyecto.

RF-5.2. El usuario debe ser capaz de obtener más información sobre sistemas de recuperación de imágenes y sobre los descriptores.

4.2.2 Requisitos no funcionales

Los requisitos no funcionales describen cualidades o restricciones del sistema que no se relacionan de forma directa con el comportamiento funcional del mismo.

Este proyecto cuenta con los siguientes requisitos no funcionales:

RNF-1 Debe de utilizar la mínima cantidad de memoria posible.

RNF-2 Ha de ser implementando en Android.

RNF-3 Ha de requerir los permisos mínimos para funcionar.

RNF-4 Las consultas se realizan utilizando una única imagen consulta.

RNF-5 El tamaño de las imágenes mostradas ha de ser lo suficientemente grande para poder verse correctamente.

RNF-6 Se debe indicar el orden en el que fueron dibujadas las imágenes.

4.2.3 Requisitos de información

Este tipo de requisito de detallar la necesidad por parte del sistema del almacenamiento de la información.

RI-1 Almacenar información de la consulta para evitar nuevas cálculos innecesarias.

4.3 Historias de usuario

Cuando nos referimos a historias de usuario, nos refererimos a la representación de un requisito software que se encuentra escrito en varias frases cortas, utilizando el lenguaje común del usuario. Esto facilita la comprensión y realización de los requisitos de un proyecto. Se trata de otra manera de ver los requisitos software que se acaban de explicar. Por lo que vamos a ver sólo los pricipales

1	Interacción con la interfaz
Descripción	
Se podra mover con libertad por los distintos menús diponibles en la interfaz.	
Pruebas de aceptación	
- Comprobar que al pulsar en al pulsar el item **consulta** se muestra la interfaz asociada. - Comprobar que al pulsar en el action button, este se despliega mostrando dos opciones, cámara y galería. - Comprobar que al pulsar en al pulsar el item **ajustes** se muestra la interfaz asociada.	
- Comprobar que al pulsar en al pulsar el item **adicional** se muestra la interfaz asociada.	

Table 4.1: Historia de usuario - Interacción con la interfaz

2	Cargar imagen cámara
Descripción	
Se podra mover cargar la imagen consulta utilizando la cámara del dispositivo.	
Pruebas de aceptación	
- Comprobar que al pulsar en al pulsar el item **cámara** del *floating button*, se lanza la actividad cámara, pudiendo elegir la delantera o trasera en caso de que se diponga de ellas. - Comprobar que al tomar una fotografía esta se añade a la sección de imágenes consulta. - Comprobar que al pulsar en al pulsar en el item **consultar**, se realiza la consulta con esta nueva imagen.	

Table 4.2: Historia de usuario - Cargar imagen consulta cámara

3	Cargar imagen galería
Descripción	
Se podra cargar la imagen consulta utilizando la galería del dispositivo.	
Pruebas de aceptación	
- Comprobar que al pulsar en al pulsar el item **galería** del *floating button*, se lanza la galería. - Comprobar que al tomar seleccionar una imagen, esta se añade a la sección de imágenes consulta. - Comprobar que al pulsar en al pulsar en el item **consultar**, se realiza la consulta con esta nueva imagen.	

Table 4.3: Historia de usuario - Cargar imagen consulta galería

4	Cambiar de descriptor
Descripción	
Se podra cambiar el descriptor que se emplea durante las consultas.	
Pruebas de aceptación	
- Comprobar que al pulsar sobre el descriptor deseado este queda maracado como activo. - Comprobar que solo un descriptor puede ser seleccionado. - Comprobar que al pulsar en al pulsar en el item **consultar**, se realiza la consulta con este descriptor seleccionado.	

Table 4.4: Historia de usuario - Cambiar descriptor

5	Eliminar base de datos
Descripción	
Se podra eliminar la base de datos.	
Pruebas de aceptación	
- Comprobar que al pulsar sobre la opción *Eliminar BD* se lanza un desplegable para realizar la acción. - Comprobar que al pulsar en aceptar, se notifica de que la BD ha sido eliminada.	

Table 4.5: Historia de usuario - Eliminar base de datos

6	Calcular base de datos
Descripción	
Se podra precalcular la base de datos para agilizar consultas.	
Pruebas de aceptación	
- Comprobar que al pulsar sobre la opción *Calcular BD* se lanza un diálogo sobre el proceso. - Comprobar que al realizar una consulta, el tiempo es menor que si no hubiese base de datos.	

Table 4.6: Historia de usuario - Calcular base de datos

7	Elegir número de imágenes
Descripción	
Se podra elegir el número de imágenes que serán consultadas.	
Pruebas de aceptación	
- Comprobar que al pulsar sobre la opción *Elegir número de imagenes* se lanza un desplegable permitiendonos elegir el número. - Comprobar que el número de las imágenes resultado se corresponde con el establecido previamente.	

Table 4.7: Historia de usuario - Elegir número de imágenes

8	Interactuar con las imágenes consulta
Descripción	
Se podra interactuar con las imágenes consulta.	
Pruebas de aceptación	
- Comprobar que se puede realizar movimiento scroll de derecha a izquierda y viceversa sobre las imágenes consulta, siempre que haya las suficientes. - Comprobar que al pulsar sobre una, esta se muestra a pantalla completa indicándonos la posición que ocupa respecto a las demás.	

Table 4.8: Historia de usuario - Interactuar con las imágenes consulta

9	Interactuar con las imágenes resultado
Descripción	
Se podra interactuar con las imágenes resultado.	
Pruebas de aceptación	
- Comprobar que se puede realizar movimiento scroll hacia arriba, hacia abajo y viceversa sobre las imágenes resultado, siempre que haya las suficientes. - Comprobar que al pulsar sobre una, esta se muestra a pantalla completa indicándonos la posición que ocupa respecto a las demás, y la distancia respecto a la imagen consulta.	

Table 4.9: Historia de usuario - Interactuar con las imágenes consulta

4.4 Diagramas de secuencia

Establecidas las historias de usuario, procedamos con los diagramas de secuencia. Estos se usan para establecer el modo en que interaccionan objetos en un sistema atendiendo a UML.

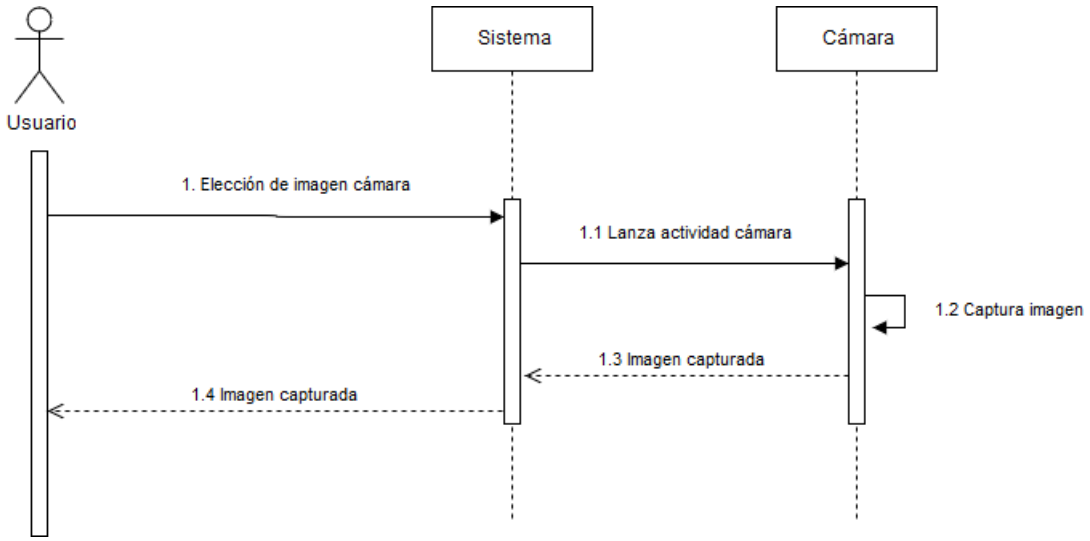

Figure 4.1: Diagrama de secuencia imagen consulta cámara

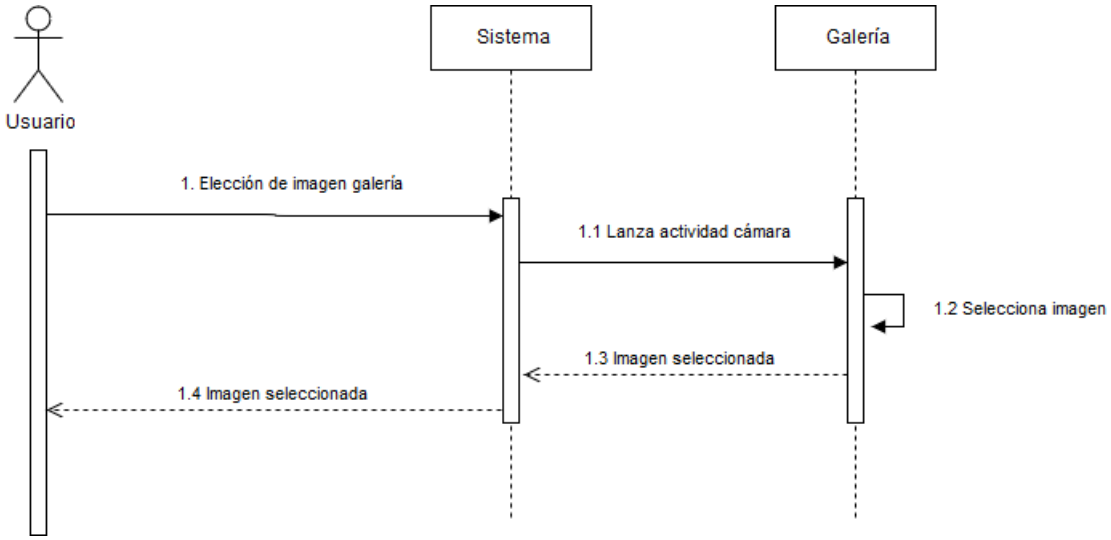

Figure 4.2: Diagrama de secuencia imagen consulta galería

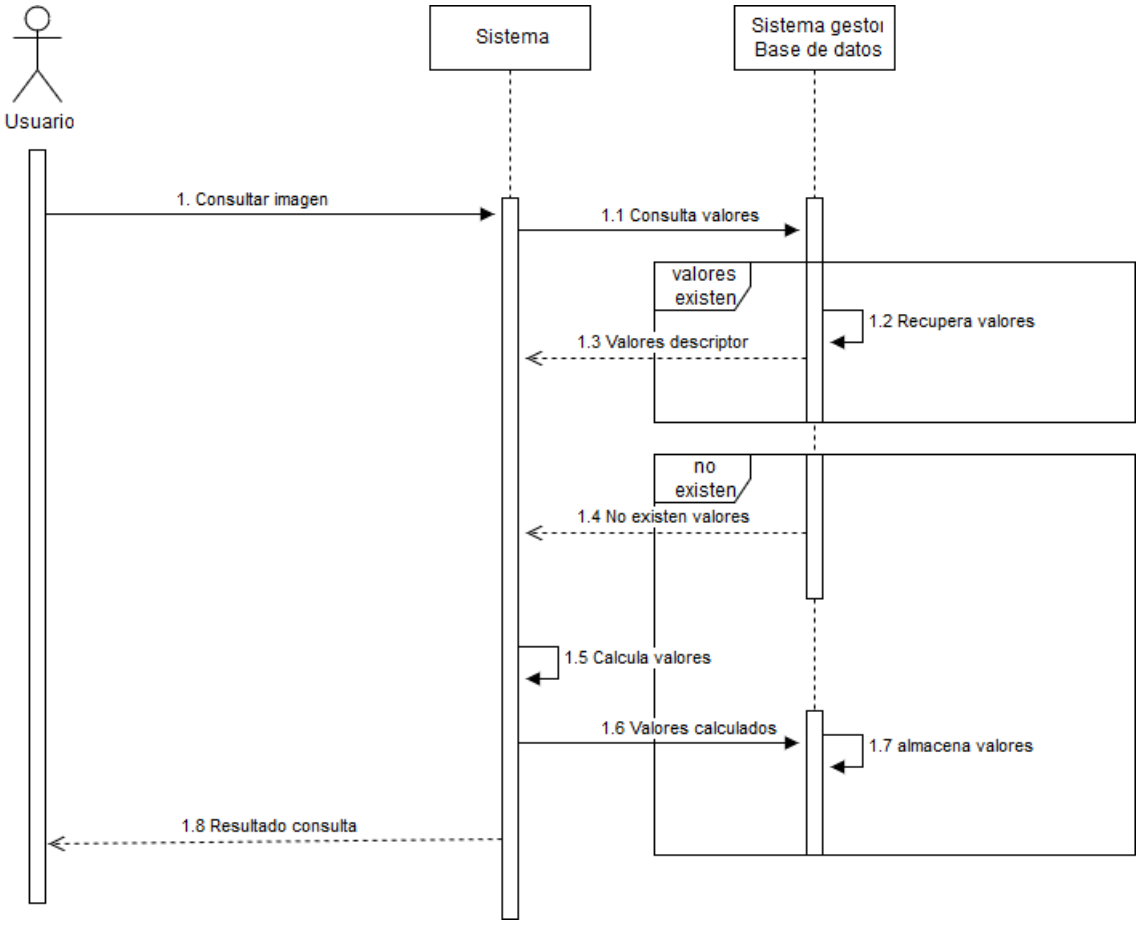

Figure 4.3: Diagrama de secuencia imagen realizar consulta

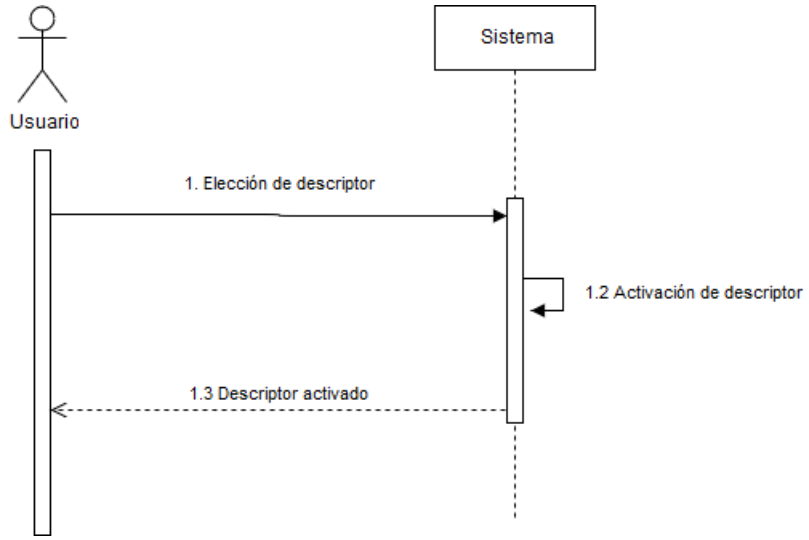

Figure 4.4: Diagrama de secuencia elección descriptor

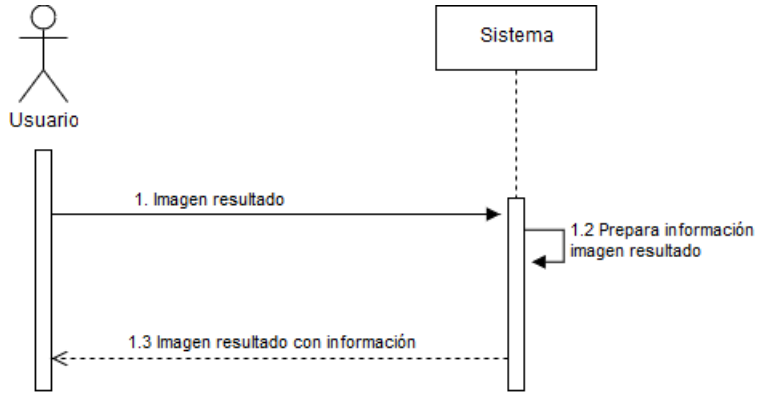

Figure 4.5: Diagrama de secuencia información imagen resultado

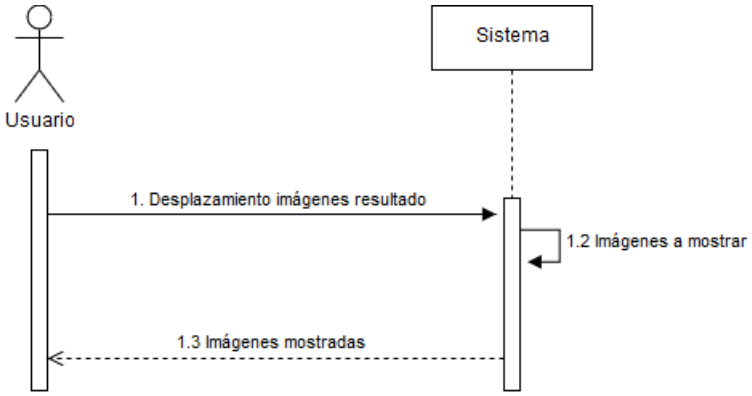

Figure 4.6: Diagrama de secuencia desplazamiento imagen resultado

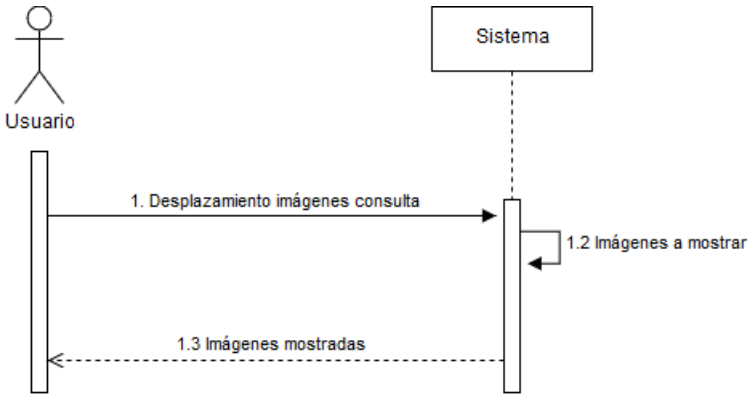

Figure 4.7: Diagrama de secuencia desplazamiento imagen consulta

Capítulo 5

Diseño

En este capıtulo se va a hablar sobre el diseño del proyecto, ası como de la arquitectura, herramientas y técnicas elegidas para el desarrollo del mismo.

El diseño es el desarrollo de aplicar una serie de métodos, técnicas y principios de diseño para traducir el modelo del análisis a una representación la cual sea posible ser codificada.

5.1 Metodología

La metodología seguida puede considerarse como *Scrum*.

Al comienzo de este proyecto se tuvieron varias reuniones para establecer los objetivos y requisitos que debía de cumplir este proyecto, aunque no en demasiada profundidad, siendo esto una tarea que se llevo a cabo según lo establecido en el capítulo anterior.

Una vez finalizadas dichas reuniones iniciales, se establecieron una serie de fechas en las cuales se tuvieron otras reuniones para comprobar el estado del proyecto. Estas fechas coinciden con la planificiación. Para cada reunión, se establecía una serie de objetivos que se debían alcanzar para el correcto desarrollo del proyecto y garantizar que se cumplía con la planificación establecida. De esta manera, se intercambiaron opiniones sobre la situación del proyecto, comentando posibles mejoras y solucionando las dudas surgidas durante las distintas etapas del desarrollo.

5.2 Arquitectura

El estilo arquitectónico utilizado para la realización de este proyecto ha sido el modelo vista controlador, ya que se adapta correctamente a las necesidades del sistema. Los elemenos de una arquitectura MVC son:

- Modelo: Es el encargado del conocimiento del dominio de la aplicación.
- Vista: Responsable de mostrar las diferentes caracterısticas del modelo al usuario.
- Controlador: Elemento que responde a la interacción del usuario, realizando las peticiones necesarias al modelo y la vista.

Una vez definino el estilo arquitectónico se va a pasar a hablar de las herramientas que se han usado.

5.3 Herramientas usadas

En esta sección vamos a comentar las herramientas que han sido usadas durante la realización del proyecto.

5.3.1 Android Studio

Aunque no es estrictamente necesario para desarrollar aplicaciones Android, es posible utilizar eclipse por ejemplo, he decidio usarlo ya que es muy sencillo de entender y facilita al progamador muchas tareas. Esto ha sido muy importante ya que, como he comentado antes, mi experiencia con Android era muy reducida.

Android Studio es el entorno de desarrollo integrado, *IDE*, oficial para la plataforma Android. Por esta razón la documentación es abundante, lo que supone un gran punto a su favor.

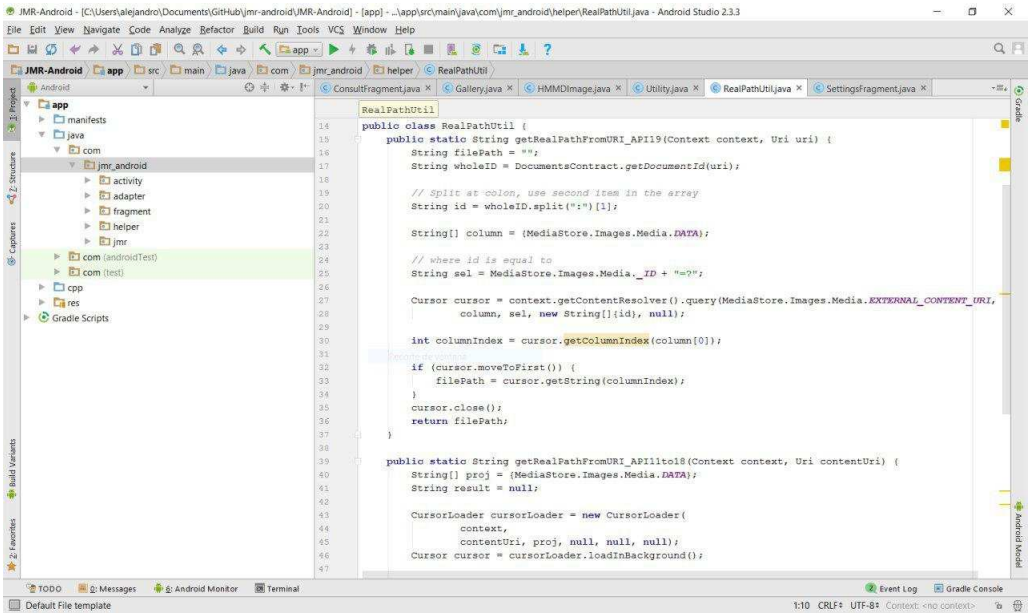

Figure 5.1: Ejemplo de proyecto Android Studio

Otra de las cosas interesantes de este *IDE* es la posibilidad de diseñar interfaces de una manera muy sencilla e intuitiva, permitiendo arrastar los elementos a las posiciones deseadas. Por lo que no se requiere un gran nivel de programación para estas tareas. Aunque si hay que comentar, que si se necesita hacer cosas más complicadas, o que no sean las estándar, si es necesario un nivel de programación avanzado, ya que en dicho caso, la ayuda proporcionada por Android Studio para estas tareas se reduce.

5.3.2 Git y GitHub

Al tratarse de un proyecto de esta magnitud, ha sido necesario utilizar una herramienta de control de versiones, como es natural se ha utilizado git.

También se ha usado GitHub, que se trata de un lugar donde alojar nuestros proyectos utilizando el sistema de control de versiones Git. Por lo tanto, podemos entender que git y GitHub van de la mano, al menos en este caso.

El respositorio del proyecto se puede consultar aquí.
Para llevar un control del proyecto se han usado los elementos conocidos como *Milestones* y *Issues* por GitHub.

Podemos entender un *Issue* como una tarea por realizar, siendo un ejemplo, *seleccionar imagen de la galería*. Se les puede añadir información extra, como a que *Milestone* está asociado, que persona es la encargada de solucionarlo, o se puede añadir una eti-

queta para establecer el tipo.

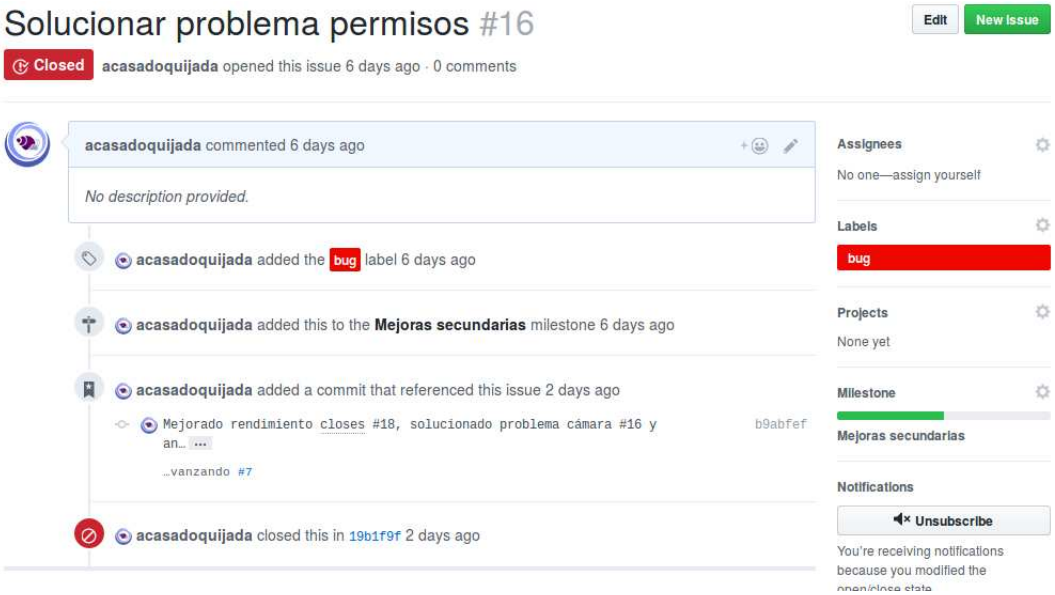

Figure 5.2: Ejemplo de issue

Por otro lado, se encuentran los *Milestones*, que podemos considerarlos como hitos, es decir, un *Milestone* está compuesto por varios *Issues*. Por lo que también puede ser vistos como una agrupación de *Issues*, una gran tarea dividida en pequeñas subtareas.

Figure 5.3: Ejemplo de milestone

Como se puede entender, usar ambos es de vital importancia si se desea llevar a cabo un proyecto de gran magnitud.

5.3.3 Dispotivio de pruebas

Aunque Android Studio nos ofrece la posibilidad de utilizar un emulador para lanzar la apliación, he decidido utilizar mi dispositivo móvil, por motivos de eficiencia y comodidad. Debido a que no se podía comprobar de manera real algunos aspectos de la propia aplicación, como el consumo de memoria o el propio rendimiento de las consultas.

Mi smartphone es un *Xiaomi redmi 4 pro*, y cuenta con las siguientes características destacables:

- CPU: Qualcomm Snapdragon 625, con ocho núcleos y 2GHz
- Memoria: 3 GB
- Almacenamiento: 32 GB

Se tratan de unas características que suelen ser habituales de encontrar en los smartphones actuales, por lo que ha sido un gran sujeto de pruebas.

5.4 Tecnicas

Para programar en Android se utiliza *Java* para la lógica, y *XML* para las interfaces de usuario.

Como es habitual, el desarrollo en Java ha seguido un paradigma de programación orientada a objetos, *POO*. En el que se han desarrollado una serie de clases que se han organizado en distintos paquetes. Esto será explicado posteriormente.

Comentar que se ha utilizado también *Android NDK*. Se trata de un conjunto de herramientas que nos permiten implementar partes de la aplicación en código nativo, como *C* o *C++*. Esta opción es perfecta para usar en los cálculos que realiza la aplicación. Como en el caso anterior se comentará con más detalle en su correspondiente sección.

```
Java_com_example_alejandro_jmr_1android_jmr_MPEG7ColorStructure_quantFuncC(JNIEnv *env,
                                                                           jobject instance,
                                                                           jdouble x) {

    // TODO

    int i = 0;
    double stepIn[] = {0.000000001, 0.037, 0.08, 0.195, 0.32, 1};
    int stepInLength = 6;
    int stepOut[] = {-1, 0, 25, 45, 80, 115};
    int y = 0;
    if (x <= 0) {
        y = 0;
    } else if (x >= 1) {
        y = 255;
    } else {
        y = (int) round(((x - 0.32) / (1 - 0.32)) * 140.0);
        for (i = 0; i < stepInLength; i++) {
            if (x < stepIn[i]) {
                y += stepOut[i];
                break;
            }
        }
        // Since there is a bug in Caliph & emir version the data
        // are between -66 and 255
        y = (int) (255.0 * ((double) y + 66) / (255.0 + 66.0));
    }
    return y;

}
```

Figure 5.4: Ejemplo de código Android ndk

Por último comentar que se ha usado *XML* para las interfaces de usuario, la mayor parte del tiempo apoyándose en el soporte proporcionado por Android Studio, pero que a la hora de realizar cosas mas complejas se ha tenido que escribir manualmente dicho código XML.

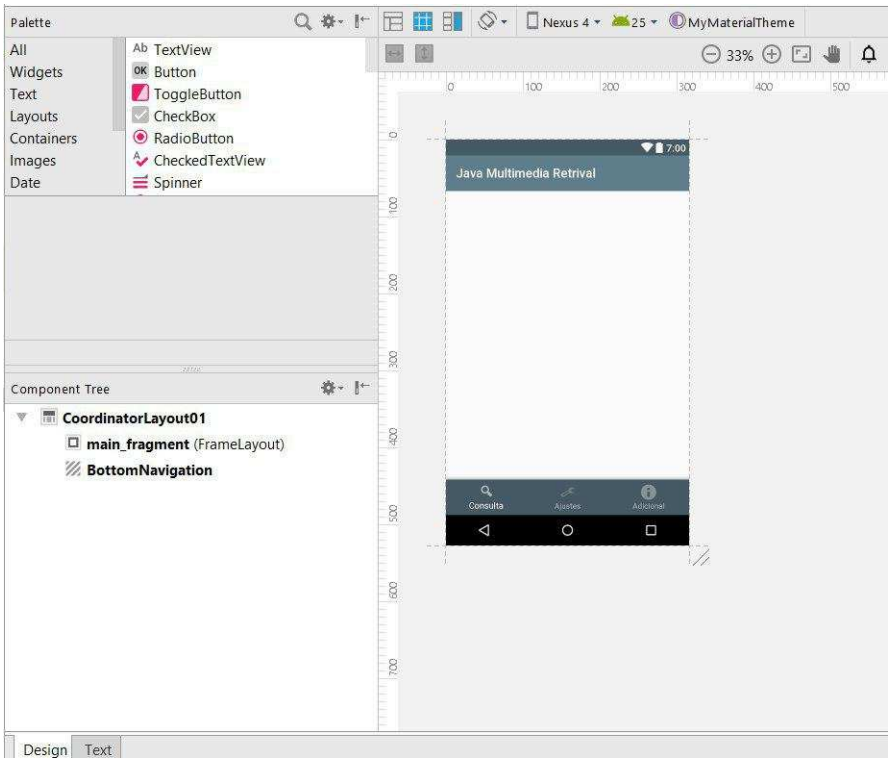

Figure 5.5: Ejemplo de proyecto Android Studio

Capítulo 6

Desarrollo

En este capítulo se va a llevar a cabo el estudio del arte relativo a este proyecto y se van a comentar el proceso de desarrollo realizado para conseguir los objetivos establecidos.

6.1 Estudio del arte

El estudio del arte debe hacerse por partida doble. Debe estudiarse el estado en el que se encuentra Android y a parte, los CBIR en dicha plataforma.

6.1.1 Estado arte Android

Por la parte de la lógica no hay ningún tipo de misterio, ya que tratamos con Java y XML, cosas que apenas cambian y ya se tienen muy estudiadas y asimiladas, por lo que no es necesario un gran estudio de esta parte.

Lo que si hay que tener en cuenta es el desarrollo de interfaces de usuario. Este es un tema muy importante, ya que aunque la aplicación funcione correctamente y presente unas novedades increibles, un mal diseño de interfaz puede provocar su abando por parte de los usuarios. Resulta natural dedicar un tiempo de investigación a esta parte.

Material design

Al realizar cualquier búsqueda sobre diseño de interfaces en Android nos encontramos con *Material design*, que se trata de una guía integral para el diseño visual, de movimientos y de interacción en distintas plataformas y dispositivos. Presenta una gran serie de nuevos elementos y novedades. Actualmente es el estándar de Android. Debido a esto, se debe seguir *Material design* a la hora de realizar una interfaz.

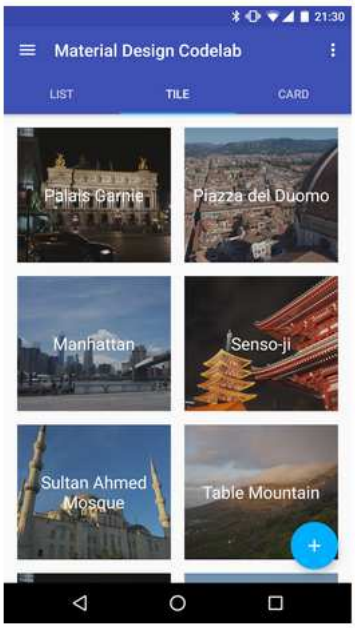

Figure 6.1: Ejemplo de Material Design 1

Figure 6.2: Ejemplo de Material Design 2

Tras un estudio preliminar de *Material design* se procedió a estudiar con más detalle dos elementos. Estos son *floating action button*, *Bottom Navigation* y *Glide*.

Floating action button

Un *Floating action button* representa la acción principal en una aplicación. Normalmente se encuentra en forma de un icono en cicular que se encuentra flotando sobre la interfaz de usuario, cambia de color al pulsar y se eleva al seleccionar. Cuando se presiona, puede contener más acciones relacionadas.

En mi caso he utilizado la implementación usada por *Clans*, el código fuente se puede encontrar en su repositorio

Figure 6.3: Ejemplo Floating action button

Bottom Navigation

Los *Bottom Navigation* nos proporciona una navegación rápida entre las vistas de nivel superior de una aplicación. Está diseñado principalmente para su uso en dispositivos móviles. Las pantallas más grandes, como el escritorio, pueden lograr un efecto similar usando la navegación lateral. Por ejemplo, el tratamiento compacto "rail" muestra los iconos de navegación de forma predeterminada.

En mi caso he utilizado la implementación usada por *sephiroth74*, el código fuente se puede encontrar en su repositorio

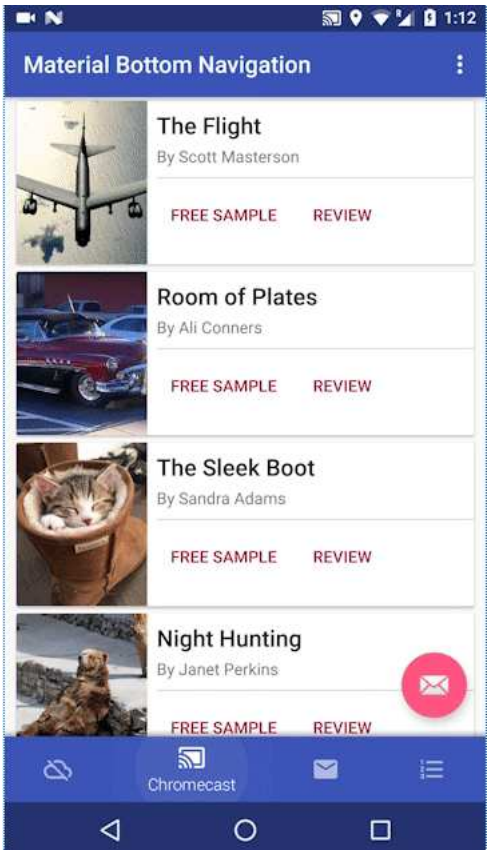

Figure 6.4: Ejemplo Bottom Navigation

Glide

Se trata de un marco de trabajo rápido y eficiente de código abierto que se encarga de la gestión de imágenes, agrupando la decodificación de medios, la gestión de memoria y el almacenamiento de cache en disco mediante una interfaz simple y sencilla de utilizar.

Actualmente se encuentra recomendada por *Google*, está desarrollada por *Bump Technologies*, y se puede consultar en el enlace de su repositorio

```
// For a simple view:
@Override public void onCreate(Bundle savedInstanceState) {
  ...
  ImageView imageView = (ImageView) findViewById(R.id.my_image_view);

  GlideApp.with(this).load("http://goo.gl/gEgYUd").into(imageView);
}

// For a simple image list:
@Override public View getView(int position, View recycled, ViewGroup container) {
  final ImageView myImageView;
  if (recycled == null) {
    myImageView = (ImageView) inflater.inflate(R.layout.my_image_view, container, false);
  } else {
    myImageView = (ImageView) recycled;
  }

  String url = myUrls.get(position);

  GlideApp
    .with(myFragment)
    .load(url)
    .centerCrop()
    .placeholder(R.drawable.loading_spinner)
    .into(myImageView);

  return myImageView;
}
```

Figure 6.5: Ejemplo de uso de Glide

En nuestro caso es ideal, ya que nos proporciona una manera sencilla de trabajar con imágenes y a su vez, nos permite utilizar movimientos de desplazamiento scroll de una manera simple, lo que nos posibilita centrarnos en la manera en la que queremos representar las imágenes.

6.1.2 Estado CBIR Android

Lo primero que podemos encontrar es información abundante sobre CBIR en computadores, pero en nuestro caso no nos es necesaria, ya que partimos de uno previo, *Java Multimedia Retrieval*, que cuenta con todo lo que necesitamos sobre este tipo de sistemas. Aunque siempre es interesante estar al día en estos asuntos.

Por lo que vamos a centrarnos en los CBIR desarrollados exclusivamente para sistemas Android, aunque sobre este tema concreto, no hay demasiada información.

Content based image retrieval for mobile systems

En esta ocasión nos encontramos ante un artículo escrito por *P Jeyanthi* profesor asociado del departamento de tecnologías de la información de la universidad Sathyabama,

Rajiv Gandhi Salai, India.

En el artículo se habla sobre un enfoque para la realización de la extracción de características basadas en textura por la co-ocurrencia de nivel de gris y la matriz de color basado en la extracción de características por color vector de concurrencia. La mayor parte del artículo se centra en el cálculo de descriptores, cosa que como se ha mencionado antes, a nosotros no nos es relevante. Por otro lado podemos ver un poco de la interfaz, lo que si nos interesa, aunque comprobamos de que se trata de una interfaz muy pobre, ya que se concentran en el cálculo de descriptores más que en la representación de los resultados.

Figure 6.6: Interfaz CBIR for mobile systems 1

Figure 6.7: Interfaz CBIR for mobile systems 2

Otro de los aspectos que nos interesan es el tiempo de cómputo y el consumo de recursos, pero no se hace ningún tipo de referencia a estos aspectos en el artículo.

Tras realizar un estudio del estado del arte de los CBIR en Android podemos llegar a la conclusión de que se tratan de sistemas que han sido poco explotados en dicha plataforma, y lo más importante, el usuario medio no conoce de su existencia, por lo que este proyecto cubre un espacio de mercado que se encuentra desocupado.

6.2 Proceso de desarrollo

En esta sección vamos a detallar los distintos elementos que componen el proyecto con un nivel de detalle suficiente para dar una visión global del proyecto y entender cada uno de sus componentes.

Como visión global podemos ver la aplicación como una única actividad que va cambiando el fragment que se muestra en pantalla, cada fragment se encarga de sus propias tareas, dejando a la actividad encargándose únicamente de cambiar fragments y otras cosas triviales.

6.2.1 Paquetes

Vamos a comenzar hablando de los distintos paquetes de los que se compone el proyecto.

Figure 6.8: Paquetes del proyecto

Activity

En él se cuentran las activities que forman parte del proyecto. En este caso solo forma parte de él una única actividad, *MainActivity* que es la encarga de gestionar la aplicación.

Adapter

Formado por *GalleryAdapter* y *ViewPagerAdapter*. El primero se encarga de renderizar las imágenes para su correcta visualización, mientras que el segundo se encarga se usa para permitir que podamos interactuar con las imágenes.

Fragment

En este paquete se encuentan los distintos framgents de los que se componen el proyecto, un total de 4. Los tres primeros se corresponden a cada una de las opciones del menú inferior de la aplicación, mientras que el último se encarga de que podamos interactuar con las imágenes al pulsar sobre ellas, aportandónos información.

Helper

Formado por clases que se encargan de *ayudar* al resto para que su funcionamiento sea el esperado. Entre algunos de los miembros de este paquete destacan *DBHelper*, que se encarga de la gestión de la base de datos y *GalleryHelper*, encargado de la gestión de la galería del usuario.

JMR

Paquete en el que se encuentra todo lo relacionado con el cálculo de descriptores y la organización de sus resultados. A su vez cuenta con clases como *HMMDImage* que se encarga de pasar una imagen en el espacio de color *RGB* al espacio de color *HMMD*. Esto se explicará con detalle en el próximo apartado

Preference

En este paquete se encuentran una serie de clases que representan los elementos que se encuentran en la sección adicional de la aplicación. Son necesarios para que esta responda como es esperada, ya que *Android* no nos proporcionaba por defecto lo que buscábamos.

6.2.2 Clases

Una vez que se ha terminado de hablar de los paquetes, es hora de hablar de cada clase individualmente. Al igual que en el caso anterior, lo haremos con cierto nivel de detalle, sin entrar en cuestiones demasiado técnicas.

Comenzaremos mostrando los diagramas de clases individuales de cada proyecto, para una mejor compresión de las clases.

6.2.3 Diagramas de clases

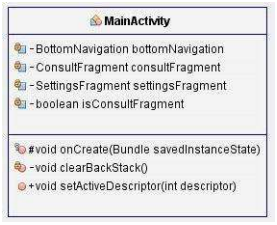

Figure 6.9: Diagrama clase paquete Activity

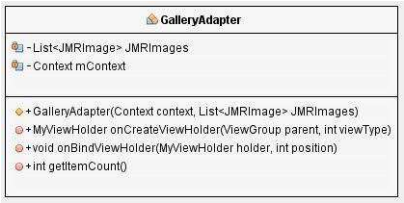

Figure 6.10: Diagrama clase paquete Adapter

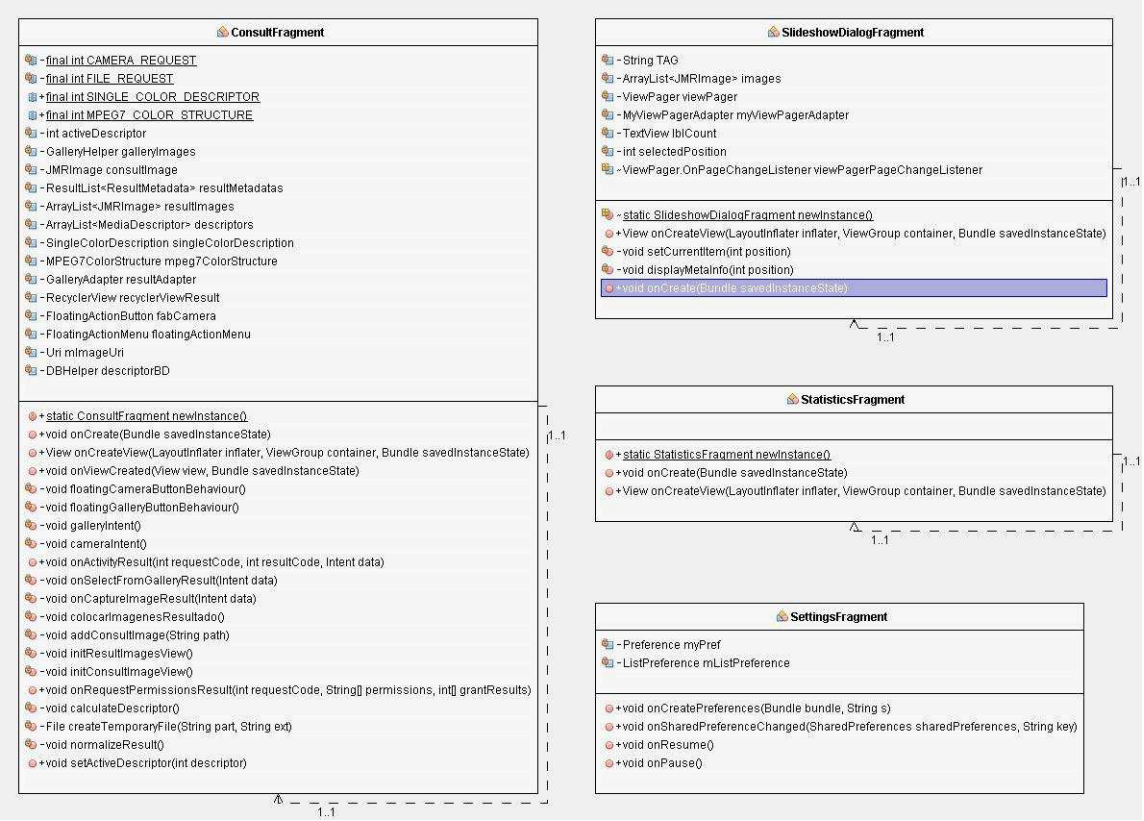

Figure 6.11: Diagrama clase paquete Fragment

Figure 6.12: Diagrama clase paquete Helper

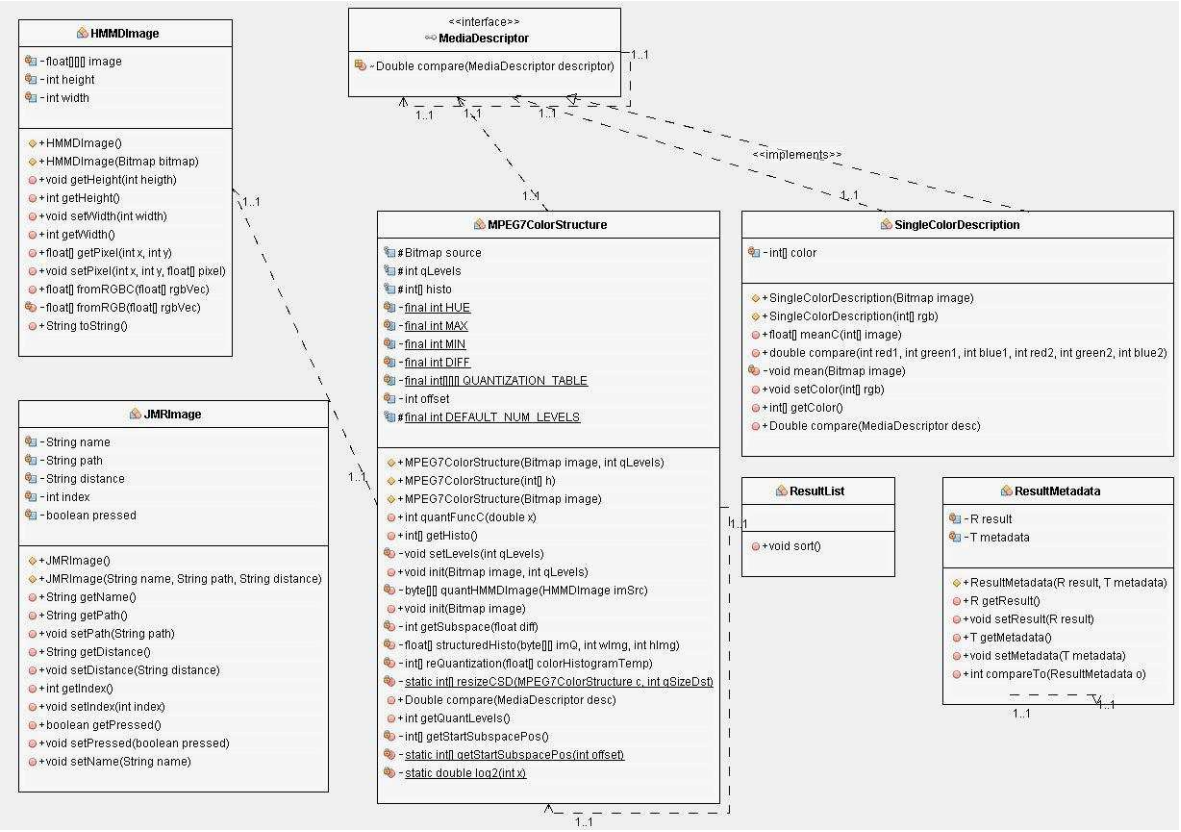

Figure 6.13: Diagrama clase paquete JMR

MainActivity

Es la única clase actividad, *Activity*, del proyecto. Se encarga de instanciar e iniciar las 3 clases *Fragment* que se corresponden a cada una de las opciones del menú inferior. A su vez, se ocupa de la comunicación entre *Fragments*, necesaria para un correcto funcionamiento de la aplicación, y también se ocupa de gestionar la memoria, para evitar que la aplicación ocupe demasiados recursos. Esto se consigue controlando la cola de *Fragments* de la aplicación.

GalleryAdapter

Esta clase se encarga de *hinchar*, *inflate*, un fichero XML llamado gallery_thumbnail.xml, en el cuál se colocan las imágenes, una por cada instancia del fichero, y a su vez, se encarga de reenderizar las imágenes. Se puede decir, que esta clase se encarga de colocar todas las imágenes y nos permite interactuar con ellas a través de eventos.

ConsultFragment

Clase de tipo *Fragment* que puede considerarse como el núcleo del proyecto.

En ella se inicializa su propia interfaz, al igual que el resto de sus clases *Fragment* hermanas. Garantiza que el usuario pueda elegir la imagen consulta tanto de la cámara como de la galería, estas imágenes se van acumulando en la parte superior de la pantalla, pudiendo moverse realizando un movimiento de scroll. También gestiona los permisos, ya que es aquí donde se encuentran los métodos que requieren de ellos para poder desempeñar correctamente su función.

Todos los cálculos de descriptores son realizados aquí, consultado previamente si los valores a calcular se encuentran en la base de datos y actuando en consecuencia. Para estos cálculos se utilizan las clases que se encuentran en el paquete JMR. Las imágenes resultantes de una consulta, imágenes resultado, se colocan justo debajo de las imágenes consulta, en cuatro columnas, pudiendo realizar un movimiento de scroll en caso de que no cupiesen todas en la pantalla del dipositivo.

SettingsFragment

Esta clase se encarga de gestionar el apartado de opciones de la aplicación. Permite al usuario elegir con que descriptor desea realizar las consultas, entre otras cosas. Básicamente, a través de eventos recoge las acciones del usuario y se las transmite a otras clases, en caso de ser necesario, usando la clase *MainActivity*.

Un ejemplo de esto es cuando el usuario decide cambiar el descriptor activo. En esa ocasión, esta clase notifica a *ConsultFragment* utilizando a *MainActivity*

SlideshowDialogFragment

Esta clase, junto con su clase interna adapter, *MyViewPageAdapter*, nos proporciona los métodos necesarios para que, cuando pulsemos una imagen, esta aparezca ocupando toda la pantalla y a su vez, nos muestra información sobre ella. Esta información es el número de imagen que ocupa del total, imagen 5 de 400, y la distancia de la imagen consulta, un valor entre 0 y 1 que nos indica como de parecidas son las imágenes entre si, siendo 0 iguales y 1 totalmente distintas.

Fragment sobre nosotros

Poner licencias, que hable un poco del proyecto.

DBHelper

Clase encargada de la gestión de la base de datos. Cuenta con métodos para crearla, alterarla y destruirla. Cuando se están realizando los cálculos de los descriptores, se

consulta la base de datos a través de los métodos de esta clase, para comprobar si el valor que se desea calcular se encuentra en ella, o en caso contrario, introducirlo. Esta base de datos es de tipo relacional.

GalleryHelper

Se encarga de gestionar todas las imágenes disponibles en el dispositivo del usuario, como es natural pensar, requiere de permisos por parte de él. Dispone de todo tipo de métodos para obtener las imágenes, y para minimizar el consumo de recursos lo único que se mantiene en memoria es la ruta de las imágenes, *path*. Cuando es necesario cargar una imagen, se hace en base a su ruta, lo que evita tener todas las imágenes en memoria, un escenario que sería imposible.

RealPathUtil

Clase que se ocupa de tratar la ruta o *path* de las imágenes, tarea que puede resultar trivial, pero no lo es.

SquareLayout

En esta ocasión, esta clase extiende de *RelativeLayout* y se encarga de representar una imagen de manera cuadrada en un *grid view*, es decir, en las 4 columnas de las que hablabamos previamente.

Utility

Se encarga de gestionar los permisos que necesitan el resto de clases para actuar correctamente. Cuando una clase requiere cualquier tipo de permiso se pone en contacto con ella y actua en consecuencia.

HMMDImage

Dado que el descriptor *MPEG7ColorStructure* necesita que la imagen en el espacio de color HMMD, esta clase se encarga de dicha tarea. Utiliza una imagen en espacio de color RGB y la transforma HMMD, con una serie de operaciones matemáticas no muy complejas.

JMRImage

Siempre que nos referimos a imagen, nos referimos a esta clase. Cuenta con todo lo necesario para garantizar que el consumo de memoria sea lo mínimo posible, como se ha comentado anteriormete. Simplemente guarda información sobre la imagen, pero no a la propia imagen. Esta información es utilizada a la hora de cargar la imagen, y de añadirle más nivel de detalle, tal que el número de la imagen respecto a las demás o su distancia a la imagen consulta, en caso de que se trate de una imagen resultado.

MPEG7ColorStructure

Clase que representa al descriptor de mismo nombre y que utiliza el histograma de una imagen en espacio de color HMMD, que posteriormente se comparará con el resto de histogramas del resto de las imágenes para ver cuan parecidas son entre ellas.

ResultList

Extiende de *LinkedList*, en la cual se almacenaran los objetos de la clase *ResultMetadata*.

ResultMetada

Clase parametrizable que representa el resultado de una descriptor. Contiene dos atributos que pueden ser de cualquier tipo, en este caso dichos atributos son de tipo *String*, que representa la ruta o *path* de la imagen y un *Double*, que representa la distancia entre dos imágenes, como de parecidas son en un valor que va desde 0 a 1.

A su vez implementa la interfaz *Comparable*, de esta manera garantizamos de que se disponga de un método para comparar distintos *ResultMetada*

SingleColorDescriptor

Representa al otro descriptor disponible en el proyecto. En este caso se calcula el color medio de una imagen en espacio de color RGB, recorriendo la imagen pixel a pixel.

AditionalFragmet

Clase que extendiende de *PreferenceFragmentCompat* e implementa la interfaz *SharedPreferences.OnSharedPreferenceChangeListener*. Esta clase representa la

6.2.4 Código nativo

Como se ha comentado en la memoria, algunas partes del código han sido implementandas en código nativo utilizando *Android nkd*. Solo ha sido implementado un poco del proyecto por dos razones:

- Carencia de tiempo y complejidad: Dado que he estado realizando prácticas a media jornada en una empresa y a la dificultad de encontrar la resolución a algunas dudas, me ha sido imposible aumentar la cantidad de código nativo del proyecto. Aunque, se va a tener muy en cuentra como trabajo futuro del proyecto.
- Malos resultados: En algunas ocasiones, al implementar algunas funciones sencillas en código nativo, se obtenían unos tiempos peores que empleando la misma función implementada en Java. Por lo que como trabajo futuro, se investigará el motivo con mas detalle.

Capítulo 7

Resultados

7.1 Tamaños y tiempos

Lo primero que debemos considerar como resultado, es el tiempo de ejecución de la aplicación, ya que nos encontramos ante un dispositivo movil, que no dispone de la misma cantidad de recursos que un ordenador portatil o de sobremesa.

También hay que tener en cuenta que estamos trabajando con imágenes, por lo que a mayor cantidad de píxeles de esta, mayor será el tiempo de cómputo, por lo que se debe encontrar un equilibrio entre el tamaño de las imagénes y el resultado obtenido. De nada nos sirve que la aplicación tarde muy poco tiempo en comparar un gran número de imágenes si el resultado es totalmente erroneo.

En una serie de pruebas iniciales, el tamaño de las imágenes no se tuvo en cuenta, por lo que incluso con el descriptor prototipo, no realizaba níngún tipo de cálculo, los tiempos de consulta eran demasiado elevados, a pesar de usar un número pequeño de imágenes.

El siguiente paso fue redimensionar las imágenes a un tamaño concreto, se comenzó con 200x200. Con este tamaño se redujo el tiempo de cómputo, y los resultados comenzaban a ser prometedores. Pero se comprobó que dicho tiempo crecía demasiado con el número de imágenes, por lo que se descartó este tamaño.

Con los tamaños 64x64 y 32x32, este último es el definitivo, se obtuvieron unos buenos tiempos de ejecución y unos buenos resultados, aunque se decidió usar 32x32 ya que el descriptor MPEG7 realiza una gran cantidad de operaciones con el histograma de la imagen, por lo que ese tamaño se adapta mejor a nuestras necesidades. Se probaron estos dos últimos tamaños, ya que en una asignatura de este máster se tuvo que trabajar con imágenes, y dichos tamaños, son los que mejores resultados nos proporcionaron.

A continuación se van a presentar una serie de tablas con tiempos para distintos

tamaño y número de imágenes que justifican lo que acaba de ser comentado.

Tamaño/nº imágenes	10	100	500	1000	2500	5000
200x200	3s	14s	1m 2s	2m 21s	6 m 20s	13m 4s
64x64	2s	5s	19s	46s	2m 27s	5m 8s
32x32	1s	4s	14s	37s	2m 5s	4m 29s

Table 7.1: Tiempos SingleColorDescriptor sin base de datos

Tamaño/nº imágenes	10	100	500	1000	2500	5000
200x200	1s	4s	6s	16s	53s	1m 53s
64x64	1s	4s	6s	16s	52s	1m 51s
32x32	1s	4s	6s	15s	50s	1m 51s

Table 7.2: Tiempos SingleColorDescriptor con base de datos

Tamaño/nº imágenes	10	100	500	1000	2500	5000
200x200	13s	1m 58s	6m 2s	9m 21s	-	-
64x64	3s	19s	1m 5s	2m 17s	6m 9s	12m 32s
32x32	2s	12s	30s	43s	2m 58s	6m 6s

Table 7.3: Tiempos MPEG7ColorStructure sin base de datos

Tamaño/nº imágenes	10	100	500	1000	2500	5000
200x200	1	5s	10s	20s	-	-
64x64	1s	5s	9s	20s	1m 3s	2m 12s
32x32	1s	5s	8s	20s	1m 1s	2m 8s

Table 7.4: Tiempos MPEG7ColorStructure con base de datos

Vemos como el tiempo de consulta en ambos descriptores es idéntico cuando se dispone de los datos previamente calculados en la base de datos. Esto es lógico, ya que en este caso se consulta en la base de datos si el valor del descriptor esta calculado para cada imagen y se recupera de ahí.

7.2 Consultas

Una vez discutidos los tiempos y tamaños, es hora de comprobar que los resultados de las consultas son los esperados. Para ello vamos a utilizar dos bases de imágenes:

- Banderas. Un cojunto de 160 con las banderas de los principales países del mundo.

- VisTex. Conjunto de imágenes de texturas, constituida por un total de 669 elementos.

Figure 7.1: Base de datos de banderas

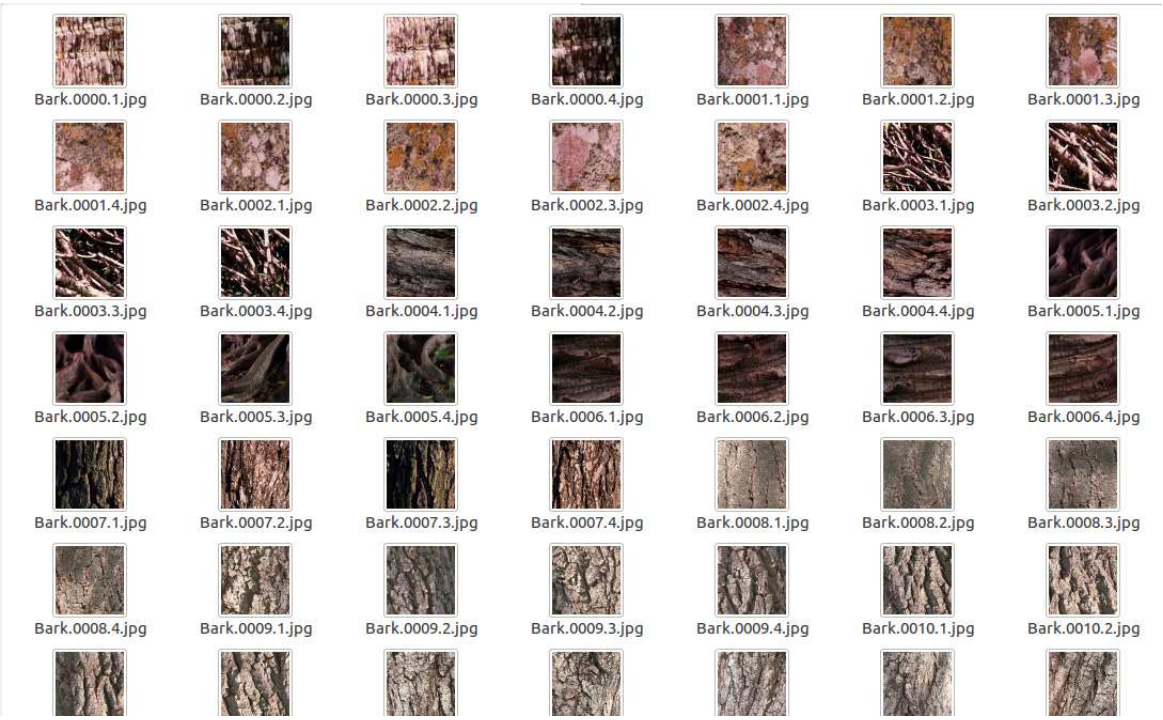

Figure 7.2: Base de datos de VisTex

Como es natural, también se han realizado las pruebas con imágenes propias de mi dispositivo, pero por comodidad y para ver mejor los resultados se ha decidido usar estas dos bibliotecas.

7.2.1 SingleColorDescription

Este descriptor se basa en el cálculo del color medio de la imagen encontrándose esta, en un espacio de color RGB. Por lo que las imágenes con colores más parecidos aparecerán juntas.

Ahora vamos a comentar una consulta utilizando este descriptor y VisTex:

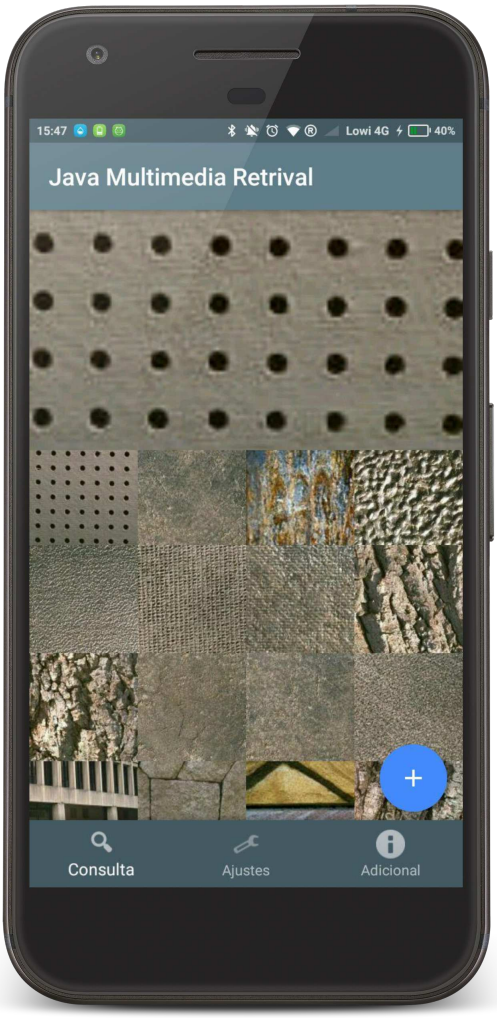

Figure 7.3: Resultado consulta SingleColorDescriptor 1

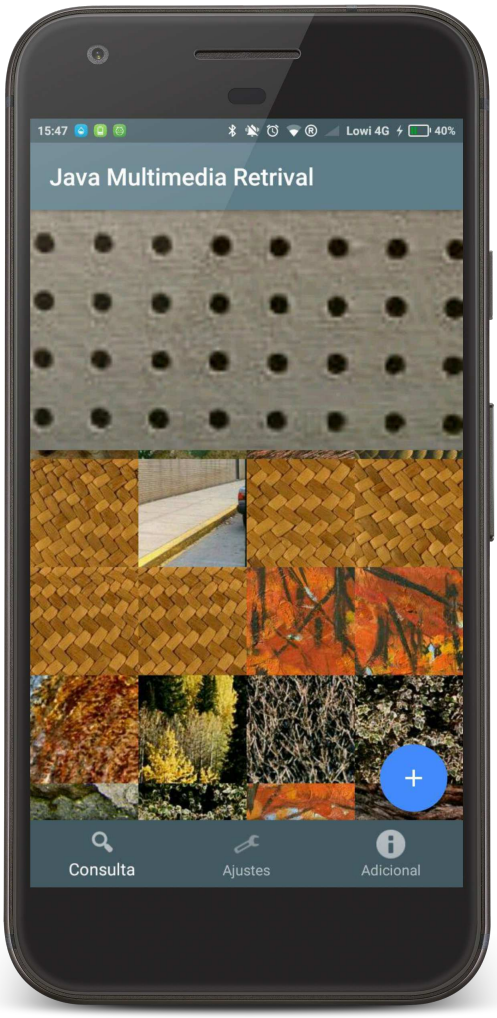

Figure 7.4: Resultado consulta SingleColorDescriptor 2

Podemos comprobar como en la primera imagen, que sería lo primero que veríamos al realizar una consulta, las imágenes resultado, son muy parecidas en color a la imagen original. Por otro lado, en la segunda imagen se aprecian imágenes que se parecen menos a la consulta, por lo que se encuentran en posiciones más alejadas. Para llegar a ellas podemos usar el movimiento de scroll.

7.2.2 MPEG7ColorStructure

El descriptor MPEG7ColorStructure se basa en el cálculo del histograma empleando imágenes en el espacio de color HMMD. Los resultados serían parecidos a los del descriptor anterior.

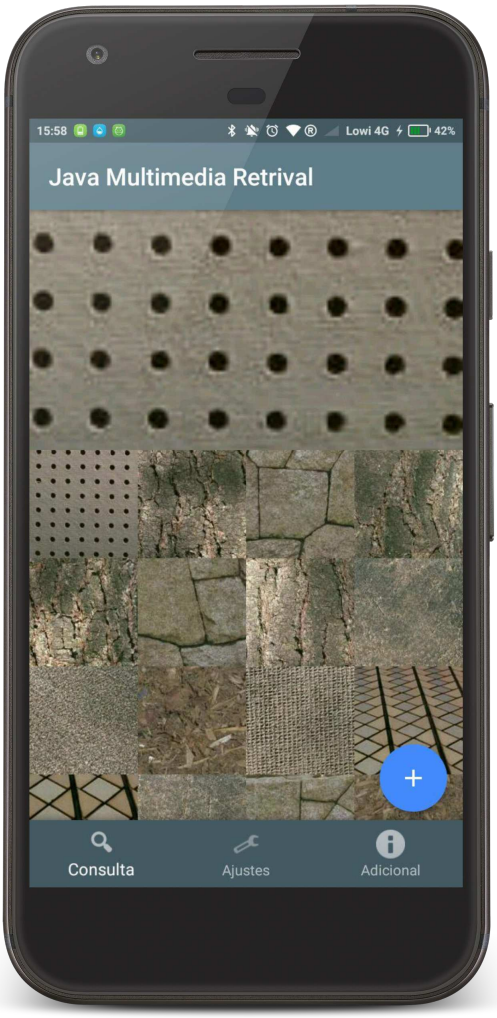

Figure 7.5: Resultado consulta SingleColorDescriptor 1

Figure 7.6: Resultado consulta SingleColorDescriptor 2

Se puede ver como este descriptor obtiene mejores resultados. Las primeras imágenes resultado son más uniformes, mientras que en el caso anterior, habia un mayor número de diferencias. Y por otro lado, como se agrupan mejor las imágenes menos parecidas.

Capítulo 8

Conclusiones

En este último capítulo del proyecto se va a comentar las conclusiones fruto de su realización. Por otro lado se explicarán trabajos futuros para este proyecto que no han podido ser adaptados.

8.1 Conclusiones

El mundo de los *CBIR* es extenso, y cubre un gran amplio rango de sectores. Aunque bien es cierto que apenas han hecho su entrada en los dispotivios móviles, al menos, de cara al usuario. Por lo que este cubre un nicho de mercado que no se encuentra muy explotado.

A la hora de realizar el proyecto partíamos con una serie de objetivos en mente, los cuales se debían satisfacer para un correcto desarrollo del proyecto. Una vez finalizado este proyecto, podemos asegurar que los objetivos establecidos se han cumplido correctamente, incluso los objetivos secundarios. Por lo que el balance es altamente positivio.

Comentar también, como se ha hecho algunas veces a lo largo de la memoria, de que carecía de experiencia con el desarrollo en Android, por lo que este trabajo ha sido un reto que me ha permitido aprender sobre esta plataforma, lo cual ha sido notablemente satisfactorio.

Con todo lo anterior, el resultado del proyecto ha sido un *CBIR* para plataformas Android, totalmente funcional, sencillo de usar y comprender por parte del usuario, pudiendo ser ajustado a sus necesidades, lo que se traduce en una gran experiencia de uso por parte del usuario.

8.2 Trabajos futuros

A continuación vamos a hablar sobre algunos trabajos que son interesantes para este proyecto, pero que por diversos motivos no se han podido incluir y se han quedado en el tintero.

El principal, es añadir una mayor cantidad de código nativo en la aplicación. Este código realiza las mismas tareas pero de una manera más eficiente. En nuestro caso, por falta de tiempo y unos resultados negativos, más tiempo de código nativo que de código Java, no se incluyó más de este código. Invirtiendo una mayor cantidad de tiempo en su estudio, estos problemas se solucionarán, lo que se traducirá en un menor tiempo de cómputo y una mejora en la experiencia del usuario.

Otro trabajo que se pensó en la inclusión de un apartado sobre estadísticas. En este apartado, se incluirían una serie de gráficos sobre las imágenes resultado.

Para ello se agruparían las imágenes en varios conjuntos, siendo estos: muy parecida, parecida, no parecida, nada parecida. Estos cojuntos, serían conjuntos difusos, por lo que tendríamos que trabajar también con lógica difusa para establecer el parentesco de las imágenes.

Por otro lado, se podría aumentar el número de ajustes a modificar por el usuario, como por ejemplo el tamaño de la imagen a la hora de realizar una consulta. Esto supondría un aumento de la precisión de la consulta, a coste de un mayor tiempo de cómputo, por lo que también habria que proporcionar una guía al usuario sobre como escoger el tamaño apropiado para las imágenes.

Bibliography

[1] Oracle. Javadoc Java 3D. http://download.java.net/media/java3d/javadoc/1.5.2/allclasses-noframe.html.

Appendix A

Manual de usuario

A.1 Permisos

Para el correcto funcionamiento de la aplicación es necesario que el usuario de permisos a la aplicación para poder acceder tanto a la cámara, como a las imágenes del dispositivo.

Al iniciar la aplicación se nos pedirá permiso para acceder al contenido del teléfono, ya que es en ese instante, cuando se inicia la aplicación, el momento en el que se listan todas las imágenes del dispositivo.

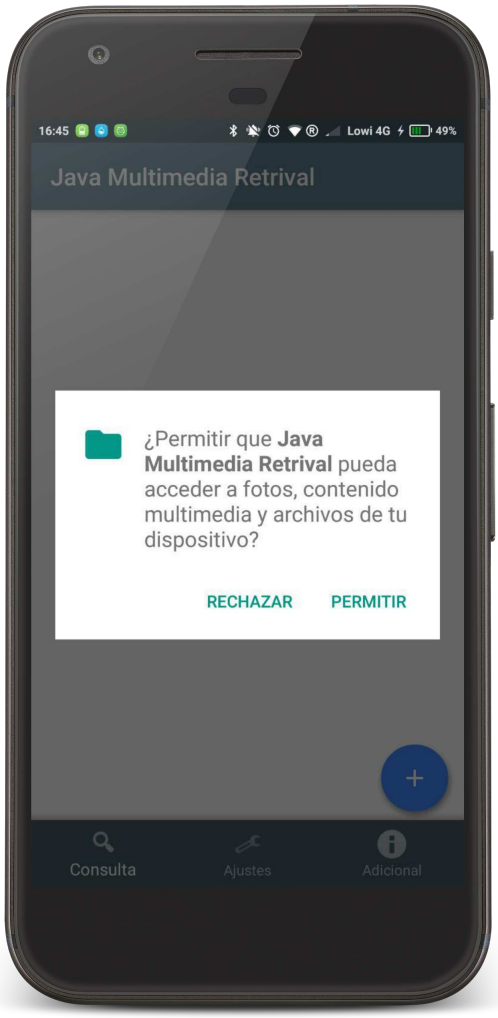

Figure A.1: Permisos aplicación 1

En caso de no conceder el permiso, cuando intentemos acceder al recurso otra vez, galería o cámara, se nos volverá a preguntar. Una vez aceptado, ya podemos acceder a los recursos requeridos sin ningún tipo de problema.

Figure A.2: Permisos aplicación 2

A.2 Navegación

Para moverse a través de la aplicación, se dispone de un menú inferior con 3 elementos o ítems, cada uno representa una pantalla distinta. Por defecto la pantalla inicial es la correspondiente a la de consulta, ya que es el corazón de la aplicación.

Se pueden pulsar cada uno de los items para cambiar a su pantalla correspondiente. Cada pantalla tiene elementos propios y no son compartidos. Por ejemplo, el *floating button* azul solo aparece en la pantalla de consulta.

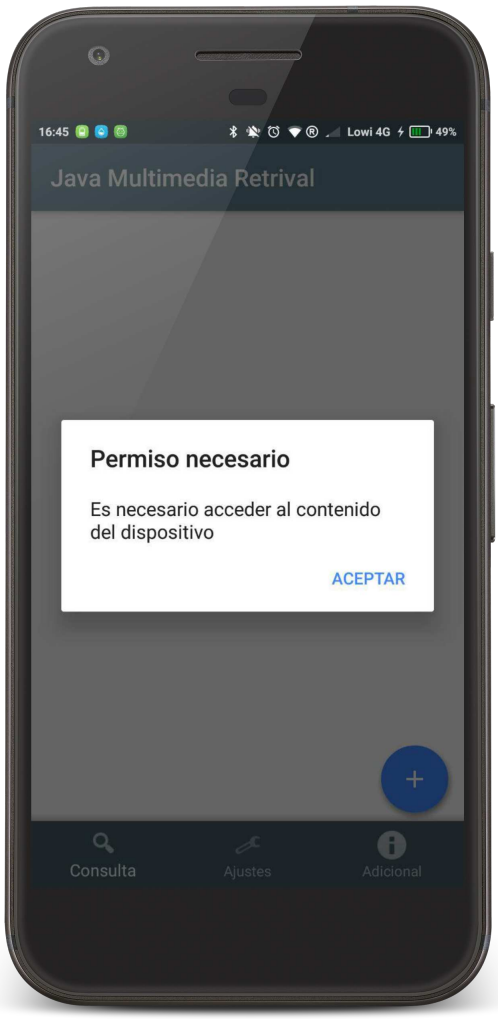

Figure A.3: Ejemplo de navegación 1

Figure A.4: Permisos navegación 2

A.3 Consulta

Para realizar la consulta disponemos de dos opciones, elegir una imagen desde la cámara o desde la galería. Estas opciones aparecen al pulsar sobre el botón flotante. Una vez elegida una, se lanzará la cámara, pudiendo elegir la trasera o delantera, o la galería, mediante la cual podremos acceder a cualquier imagen del dispositivo. Cuando sea

seleccionada, esta aparecerá en la parte superior de la aplicación.

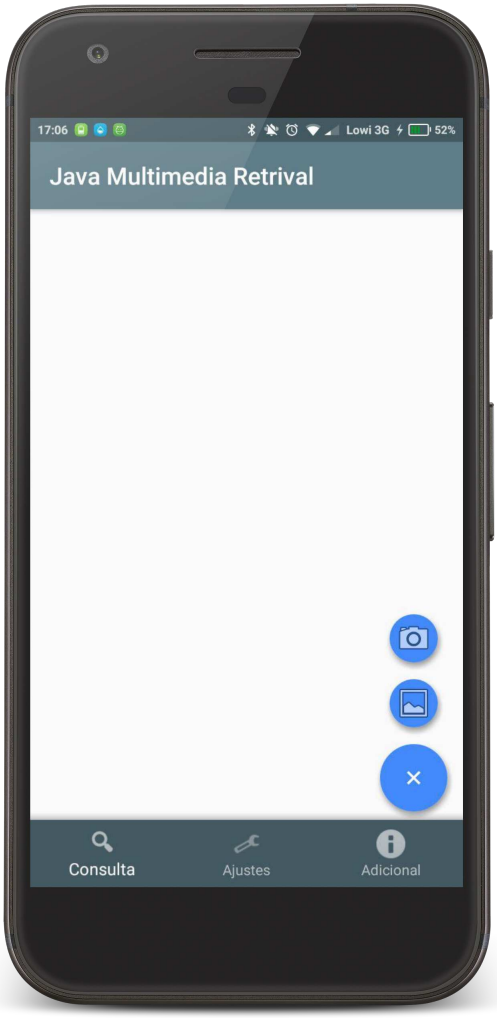

Figure A.5: Realizar consulta

A.4 Ajustes

En este apartado vamos a hablar de los ajutes que puede realizar el usuario sobre las consultas.

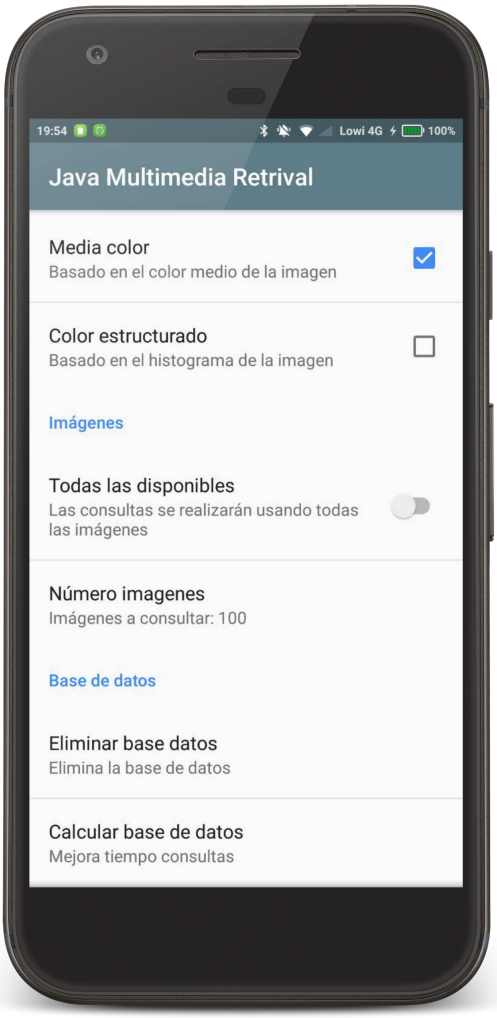

Figure A.6: Pantalla ajustes

Podríamos hablar de tres secciones:

A.4.1 Descriptores

Aquí podemos encontrar los descriptores de los que dispone la aplicación. El usuario puede elegir uno libremente, pero solo un descriptor puede ser elegido a la vez.

Figure A.7: Pantalla ajustes. Sección descriptores

A.4.2 Imágenes

En este apartado el usuario puede elegir el número de imagenes totales sobre las cuales quiere realizar la consulta. Tiene dos opciones, introducir dicho valor manualmente o seleccionar todas las del dispositivo. Este valor también se usa en el caso de querer calcular la base de datos, cosa que veremos a continuación.

Figure A.8: Pantalla ajustes. Sección imágenes

A.4.3 Base de datos

Aquí el usuario puede precalcular la base de datos, de esta manera al realizar la consulta no se calcularán nada, simplemente se obtendrán los valores de la base de datos. Este precalculo se realiza para todos los descriptores del sistema.

Por otro lado, puede borrar la base de datos si así lo desea. Realizando se calcularán los valores de la base de datos durante la consulta.

Figure A.9: Pantalla ajustes. Sección base de datos

A.5 Adicional

En esta sección el usuario puede encontrar información relacionada con el proyecto, así como sobre su desarrollador. Al pulsar en cualquier opción esta le llevará a su correspondiente página web con una mayor cantidad de información.

Debido a que este tipo de sistema, *CBIR*, no es muy conocido por los usuarios, al igual que los descriptores, se proporciona una serie de enlaces con información básica sobre ellos, en caso de que el usuario quiera aprender más o sienta curiosidad sobre el tema.

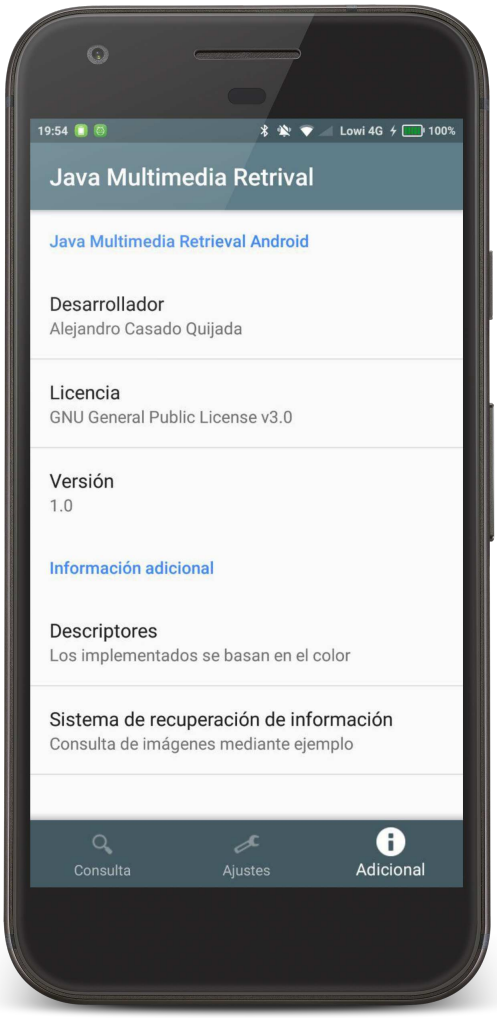

Figure A.10: Pantalla adicional