TRABAJO FIN DE GRADO

MÁSTER EN INGENIERÍA INFORMÁTICA

Desarrollo de un Sistema de Recuperación de Imágenes para Plataformas Móviles

Autor
Alejandro Casado Quijada

Directores
Jesús Chamorro Martínez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, September 15, 2017

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Resumen

1.1 Objetivos

Este proyecto busca permitir al usuario visualizar las imágenes resultantes de una consulta a un sistema de recuperación de información 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, imagenes 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 para un CBIR concreto, Java Multimedia Retrieval©.

1.2 Metodología

A partir de una serie de reuniones iniciales con mi tutor 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 realizabamos 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.

1.3 Resultados

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 pc. 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.

1.4 Conclusiones

Capítulo 2

Resumen

2.1 Contextualización

Debido al avance de la tecnologıa, cada vez disponemos de más dispositivos con lavcapacidad 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 metodologıas para solventar el problema de la manipulación, gestión y recuperación de dicha información. En este caso cuando hablamos de metodologıas 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 funcionalidad de estos sistemas es la siguiente:

Partimos de una imagen, denominada consultada, sobre la que queremos obtener imágenes similares, consideradas resultados. Para ello le proporcionamos al sistema dicha imagen consulta y una serie de criterios, descriptores, para que se realice la consulta. Una vez realizada el sistema nos presenta las imágenes resultado, junto con información sobre ellas, como por ejemplo "como de parecidas" son respecto a la original. Esto suele realizarse con una medida numérica entre 0 y 1.

2.2 Fundamentos

Este trabajo se basa en el sistema de recuperación de imágenes llamado Java Multimedia Retrival, JMR, el cual se encuentra implementado en Java. Este sistema nos permite realizar consultas usando varios descriptores, incluso varios al mismo tiempo, lo cual es muy interesante, ya que cuenta con métodos de visualización que nos permiten comprobar tantos descriptores como queramos a la vez. Con otros métodos los resultados se calcularían utilizando la media de cada uno de los descriptores.

Por lo que este CBIR ha sido de gran utilidad a la hora de realizar el proyecto, ya que ha proporcionado información como sobre realizar cálculos con los descriptores, por ejemplo, aunque como es natural, todo se ha debido de reimplementar para que funcione correctamente y siga una lógica concreta.

Capítulo 3

Objetivos

Como todo proyecto que se realiza, este ha de tener una serie de objetivos que justifiquen su realización. Por lo que en este apartado se van a discutir los objetivos de este, diviendose 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 cumplirlos, suponen un gran valor añadido al proyecto. Cabe destacar que en este caso, los objetivos secundarios han sido satisfechos en su totalidad.

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 resintiría, lo que podría traducirse en un abandono de la aplicación.

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 4

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:

4.1 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.

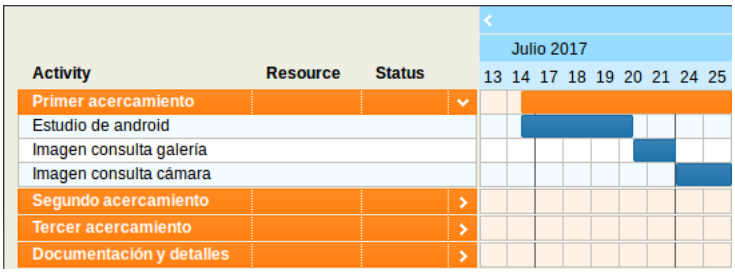

Figure 4.1: Diagrama Gantt. Primer acercamiento

El proyecto fue comenzado el 15 de julio. Los primeros dias 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.

4.2 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 funciomaniento 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.

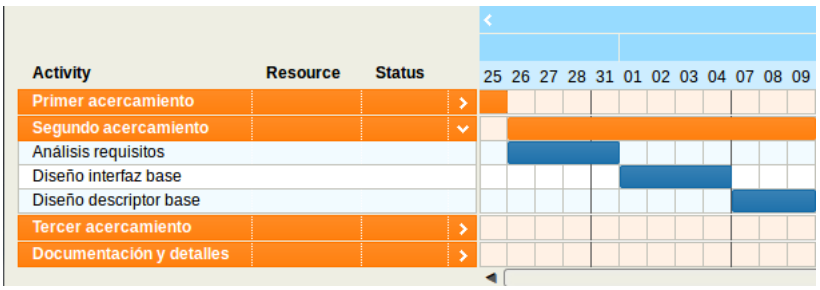

Figure 4.2: Diagrama Gantt. Segundo acercamiento

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.

4.3 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.

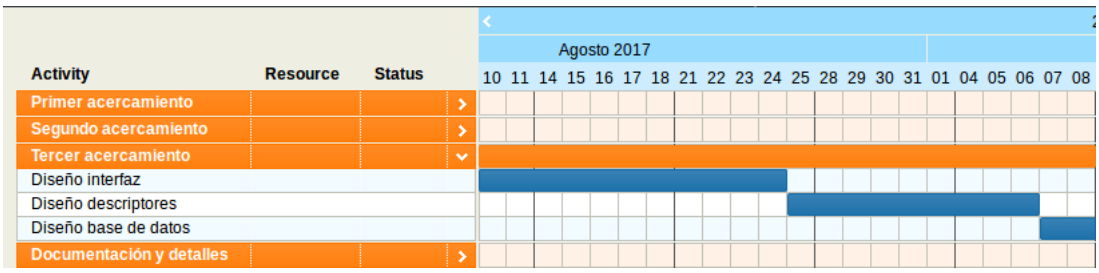

Figure 4.3: Diagrama Gantt. Tercer acercamiento

4.4 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.

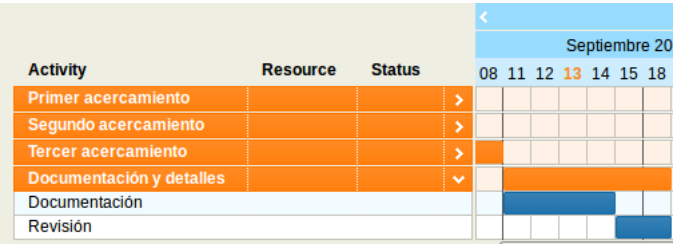

Figure 4.4: Diagrama Gantt. Documentación y detalles

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 5

Metodología

En este apartado se va a describir la metodología utilizada durante la realización de este proyecto. A su vez, se va a incluir una descripción de las herramientas, tecnologías y técnicas usadas, asi como de una justificación de dichas elecciones.

5.1 Metodología

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, 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. 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 Herramientas usadas

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

5.2.1 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.

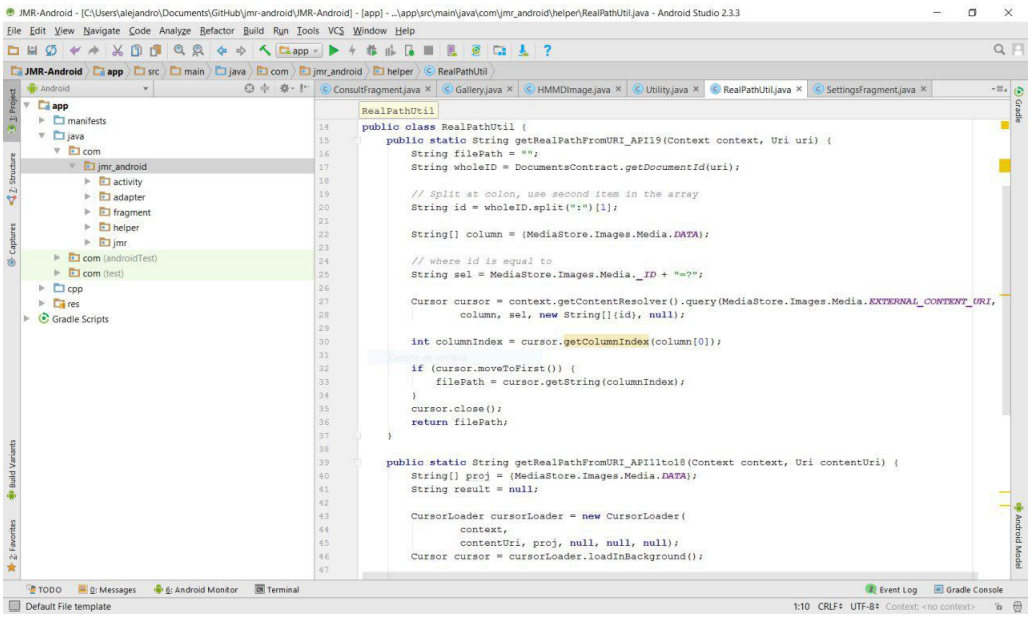

Figure 5.1: Ejemplo de proyecto Android Studio

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.

5.2.2 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 etiqueta para establecer el tipo.

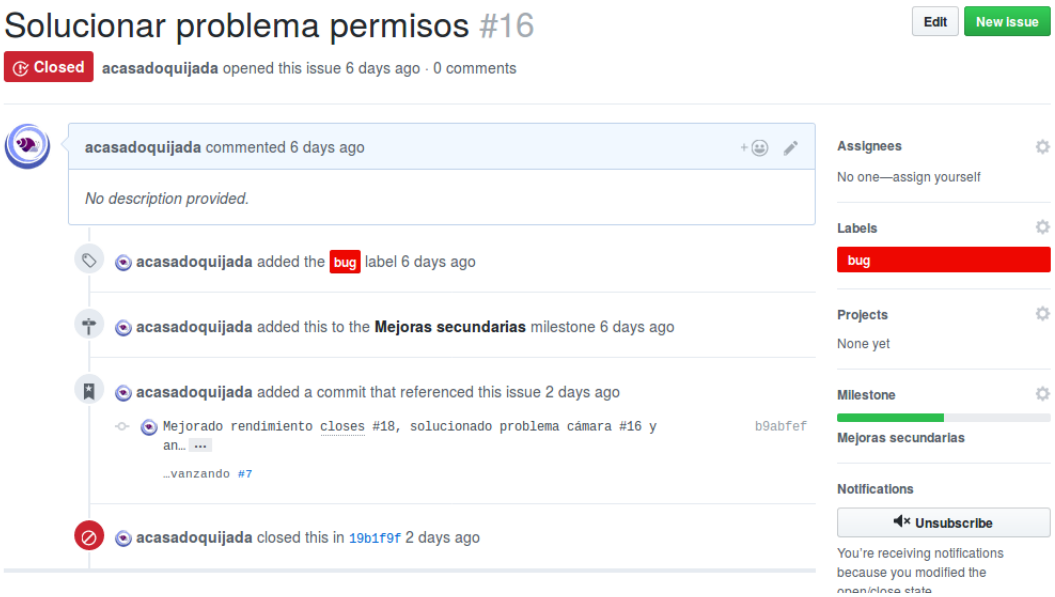

Figure 5.2: Ejemplo de issue

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. hablar sobre mi movil, carácterisiticas y tal

5.2.3 Dispotivio de pruebas

Aunque Android Studio nos ofrece la posibilidad de utilizar un emulador para lanzar la apliación, he decidido utilizar mi dispositivo movil, por motivos de eficiencia y comodidad. Debido a que no 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.

5.3 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.

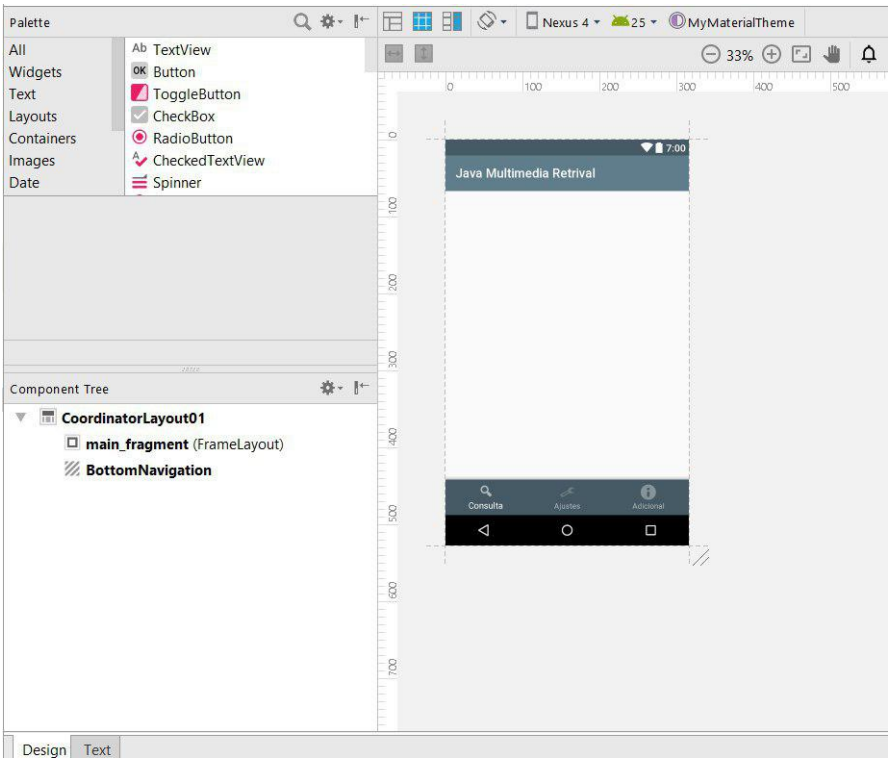

Figure 5.5: Ejemplo de proyecto Android Studio

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 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 e 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.

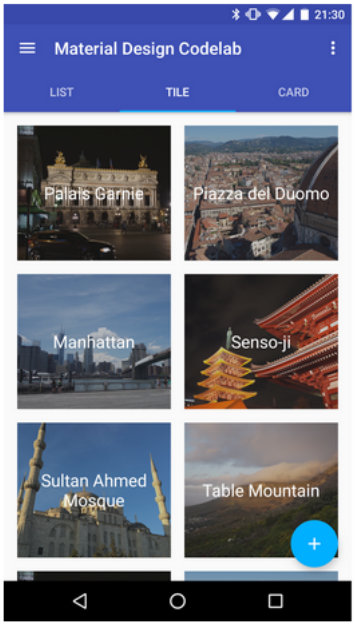

Figure 6.1: Ejemplo de Material Design 1

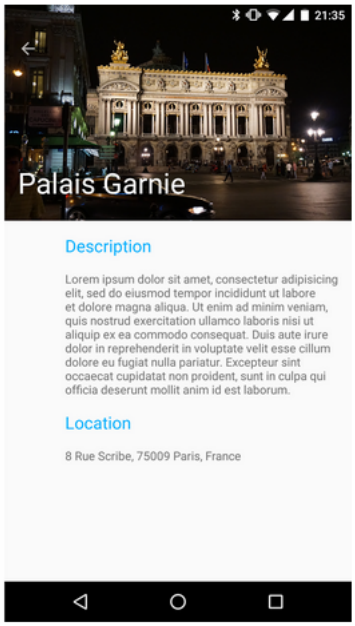

Figure 6.2: Ejemplo de Material Design 2

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

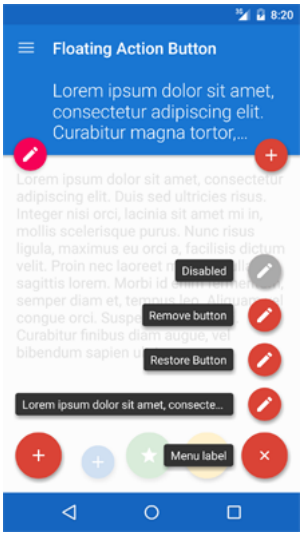

Figure 6.3: Ejemplo Floating action button

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

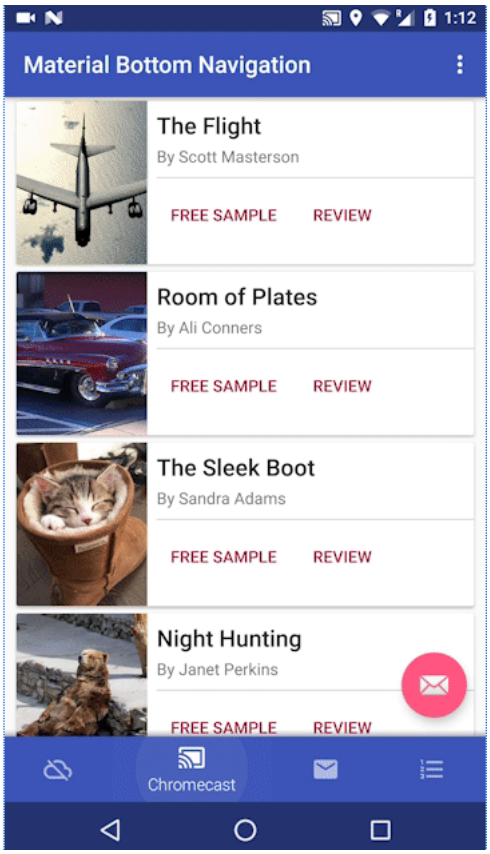

Figure 6.4: Ejemplo Bottom Navigation

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

```java
// 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 Retrival, 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, sin embargo, si vamos a comentar el CBIR Java Multimeda Retrieval, ya que partimos de él.

Java Multimedia Retrieval

Se trata de un CBIR desarrollado en Java, iniciado por Jesus Chamorro, profesor de la universidad de Granada.

Este CBIR trabaja con 3 descriptores distintos, basados en el color dominante, estructurado y escalable. Nos ofrece la posibilidad de realizar las consultas con cualquiera de dichos descriptores, incluso con 2 o 3 a la vez. A su vez cuenta con una pequeña base de datos donde se almacenan los resultados de las consultas para agilizar las posteriores.

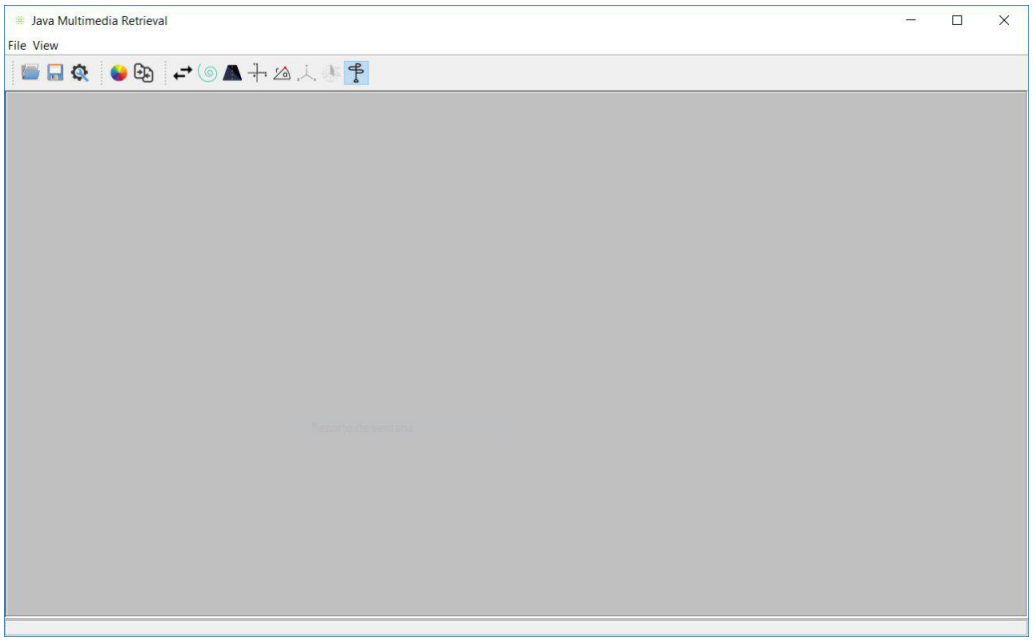

Figure 6.6: Interfaz Java Multimedia Retrieval

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 6.7: Interfaz CBIR for mobile systems 1

Figure 6.8: 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.

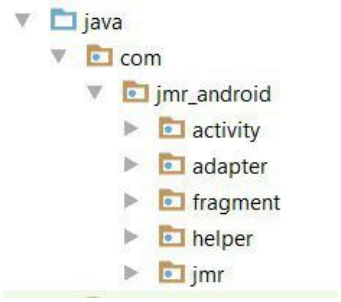

Figure 6.9: Paquetes del proyecto

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

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 diagramas de clase, uno que muestre todas las clases y otro de las clases de cada paquete, para su mayor comprensión

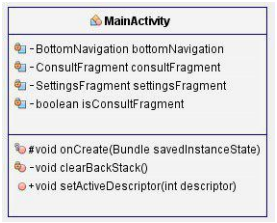

Figure 6.10: Diagrama clase paquete Activity

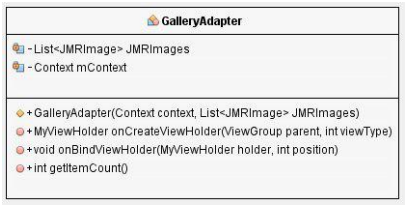

Figure 6.11: Diagrama clase paquete Adapter

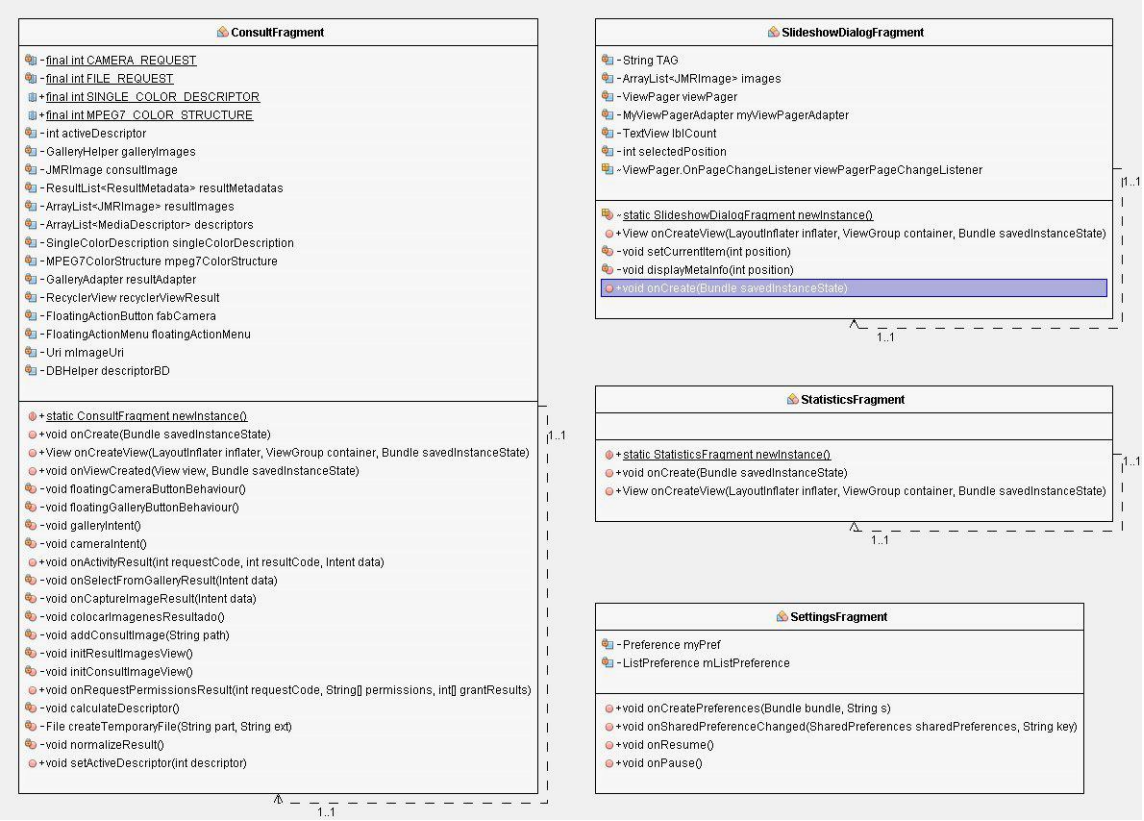

Figure 6.12: Diagrama clase paquete Fragment

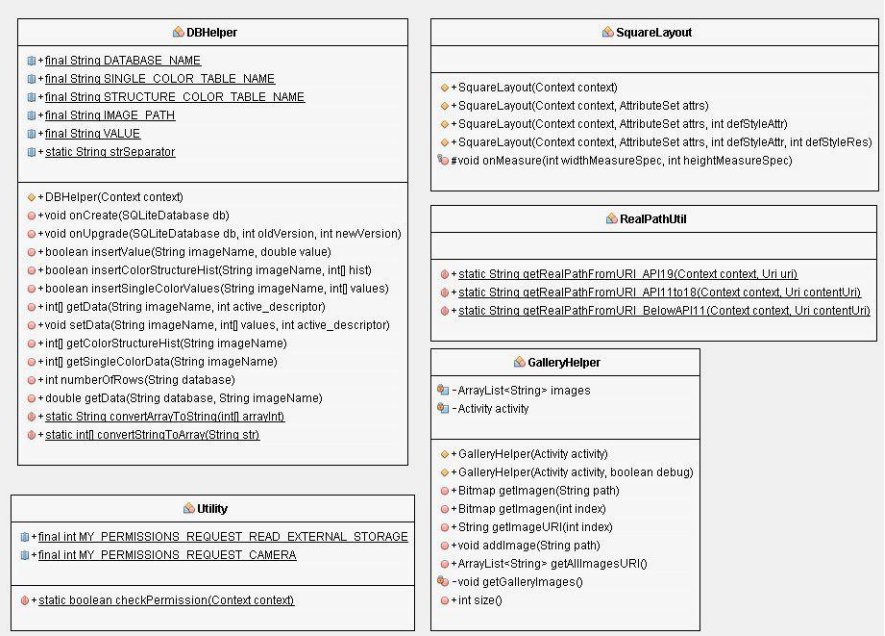

Figure 6.13: Diagrama clase paquete Helper

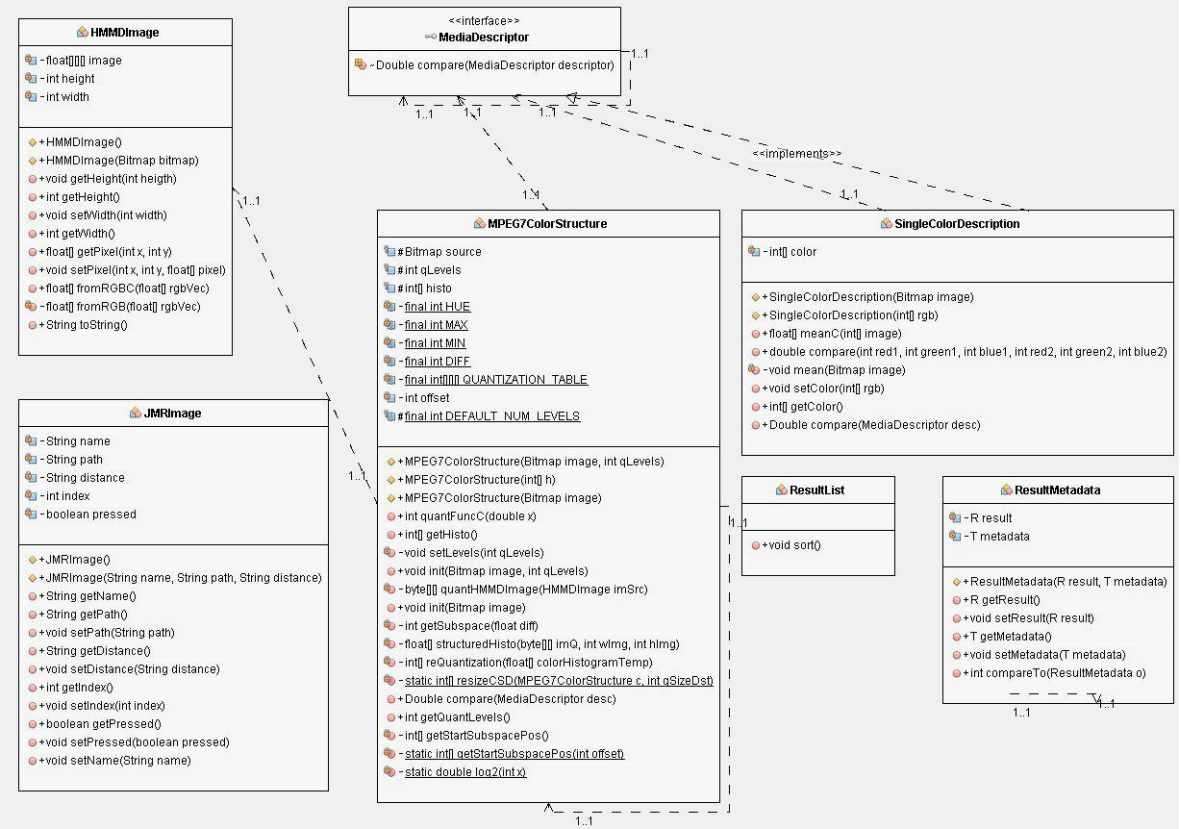

Figure 6.14: Diagrama clase paquete JMR

Una vez vistos los distintos diagramas, es hora de hablar de cada clase.

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.

6.2.3 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.

6.2.4 Historias de usuario

Cuando nos referimos a historias de usuario, nos refererimos a la representación de un requisito que se encuentra escrito en varias frases cortas, utilizando el lenguaje común

del usuario.

A continuación vamos a comentar las historias de usuario correspondientes a este proyecto.

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 6.1: 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 6.2: Historia de usuario - Cargar imagen consulta cámara

3	Cargar imagen galería
Descripción	
Se podra mover 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 6.3: Historia de usuario - Cargar imagen consulta galería

4	Realizar consulta
Descripción	
Se podra mover realizar una consulta usando una de las fotos consulta.	
Pruebas de aceptación	
- Comprobar que al pulsar durante unos segundos sobre una imagen esta se establece como la imagen consulta. - Comprobar que solo una imagen puede ser seleccionada para su consulta al mismo tiempo. - Comprobar que al pulsar en al pulsar en el item **consultar**, se realiza la consulta con esta imagen seleccionada.	

Table 6.4: Historia de usuario - Realizar consulta

5	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 6.5: Historia de usuario - Cambiar descriptor

6	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 6.6: Historia de usuario - Eliminar base de datos

7	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 6.7: Historia de usuario - Calcular base de datos

8	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 6.8: Historia de usuario - Elegir número de imágenes

9	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 6.9: Historia de usuario - Interactuar con las imágenes consulta

10	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 6.10: Historia de usuario - Interactuar con las imágenes consulta

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 el tiempo de cómputo 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.

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 Color medio

Hablar sobre el color medio, y poner resultados usando las imágenes bandera, y vistex. 2/4 pantallazos

7.2.2 MPEG7

Hablar sobre el MPEG7, y poner resultados usando las imágenes banedra y vistex. 2/4 pantallazos

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.

Bibliography

[1] Dennis J Bouvier. Getting Started with the Java 3D API - Chapter 4 Interaction. http://www.cs.stir.ac.uk/courses/CSC9N6/Java3D/Tutorial/j3d_tutorial_ch4.pdf.

[2] Oracle Corporation. Instalación Java 3D. http://download.java.net/media/java3d/builds/release/1.5.2/README-download.html.

[3] Oracle Corporation. Netbeans. https://netbeans.org/.

[4] Freepik de flaticon.com. Icono n-caminos. http://www.flaticon.com/free-icon/directions_169295#term=road&page=3&position=47.

[5] Madebyoliver de flaticon.com. Icono ecuaciones cartesianas 3D. http://www.flaticon.com/free-icon/coordinates_136810#term=coordinate&page=1&position=2.

[6] Madebyoliver de flaticon.com. Icono espiral. http://www.flaticon.com/free-icon/spiral_137099.

[7] E.T.S.I.I.T. Apuntes de fundamentos de ingeniería del software de la E.T.S.I.I.T. de la Universidad de Granada.

[8] Free Software Foundation. Licencia gpl-3.0. https://www.gnu.org/licenses/gpl-3.0.html.

[9] BrainC from markmail.org. MouseOver - ToolTip for Shape3D. http://markmail.org/message/bddcbgkfpnc6uen3#query:+page:1+mid:de5o6ppps3fv6dmp+state:results.

[10] Greg Hopkins. The Joy of Java 3D. http://www.java3d.org/introduction.html.

[11] László Kozma Arto Klami and Samuel Kaski. GaZIR: Gaze-based Zooming Interface for Image Retrieval. http://www.lkozma.net/mlmi09gazir.pdf.

[12] Chaoli Wang John P. Reese Huan Zhang Jun Tao Yi Gu Jun Ma and Robert J. Nemiroff. Similarity-Based Visualization of Large Image Collections. http://www3.nd.edu/~cwang11/research/iv15-imap.pdf.

[13] Sun Microsystems. ShadowApp creates a single plane. http://www.java2s.com/Code/Java/3D/ShadowAppcreatesasingleplane.htm.

[14] Trystan G. Upstill Rajehndra Na-gappan and Nick Craswellb. Visual Clustering of Image Search Results. http://es.csiro.au/pubs/upstill_spie01.pdf.

[15] Henry A. Sowizral David R. Nadeau. An Introduction to Programming AR and VR Applications in Java3D. http://viz.aset.psu.edu/jack/java3d/java3d.htm.

[16] Tom's Planner N.V. Toms planner software de planificación. https://www.tomsplanner.es/.

[17] Oracle. Java 3D API. http://www.oracle.com/technetwork/articles/javase/index-jsp-138252.html.

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

[19] QCAD. Icono ecuaciones polares 2D. http://www.qcad.org/doc/qcad/3.1.0/reference/en/scripts/Snap/SnapCoordinatePolar/doc/SnapCoordinatePolar_en.html.

[20] Francisco Velasco Anguita Francisco Javier Melero Rus. Apuntes de Sistemas Gráficos de la E.T.S.I.I.T. de la Universidad de Granada.

[21] Ismael H. F. Santos. Setting geometry appearances. http://webserver2.tecgraf.puc-rio.br/~ismael/Cursos/Cidade_CG/labs/Java3D/Java3D_onlinebook_selman/Htmls/3DJava_Ch09.htm#9-1.

[22] Sevarac. easyUML. http://plugins.netbeans.org/plugin/55435/easyuml.

[23] Sukirgenk. Icono ecuaciones polares 3D. http://sukirgenk.dvrlists.com/3d-cartesian-coordinate-system.html.

[24] Aouache Mustapha Aini Hussain Salina Abdul Samad Mohd Asyraf Zulkifley Wan Mimi Diyana Wan Zaki and Hamzaini Abdul Hamid. Design and development of a content-based medical image retrieval system for spine vertebrae irregularity. http://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-14-6.