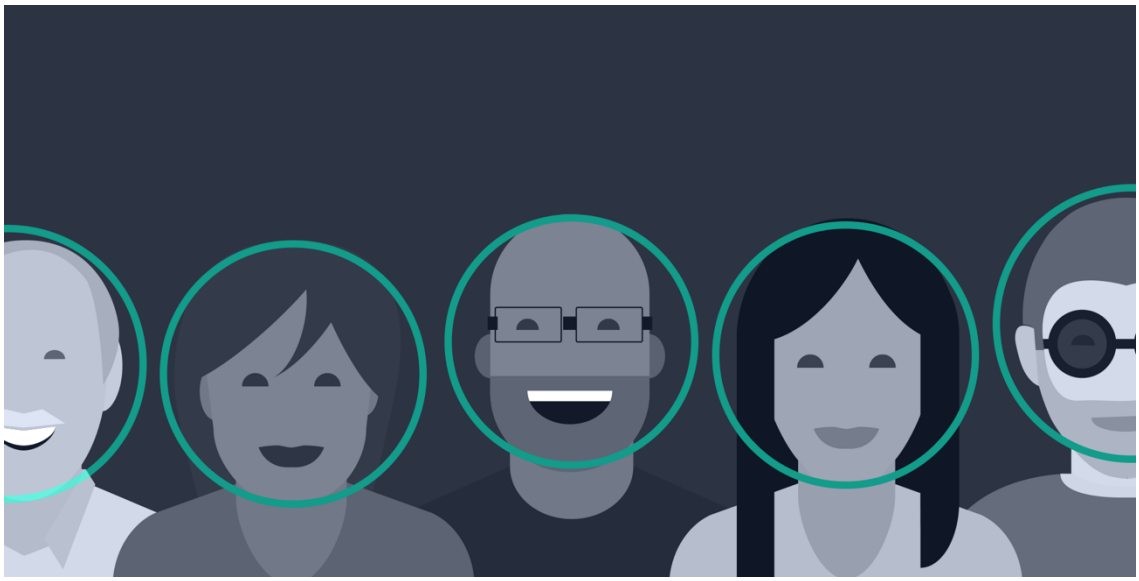
Sistemas Multimedia

WhoRU

Aplicación de Android para reconocimiento facial de profesores



Desarrolladores:

Daniel Casquel Cruz

Nicolás Moral de Aguilar

Índice de contenidos

Nombre del proyecto

La aplicación que hemos desarrollado se llama *WhoRU.* El nombre que le hemos dado refleja a la perfección cual es el objetivo final para la cual fue diseñada: ser capaz de reconocer personas utilizando imágenes. WhoRU es el acrónimo de "Who are you" que en inglés significa quién eres tú. Es un nombre corto y fácil de memorizar y suena bien al escucharlo.

Contextualización y problema

Para este proyecto se nos pide la realización de una aplicación para cualquier plataforma pero con contenido relacionado con multimedia. Como estudiantes, tenemos el problema de tener bastantes profesores y sería muy cómodo para nosotros que existiese una aplicación que nos permitiera identificar al profesorado y al PAS de la universidad. No queríamos quedarnos en una aplicación básica de multimedia como puede ser una galería o un reproductor, que sería la opción más típica de realizar, pasando así a buscar otras ideas factibles con las que poder hacer una aplicación que nos supusiese un mayor reto. Por ello decidimos intentar hacer una aplicación basada en nuevas tecnologías que estuvieran en expansión actualmente, como el reconocimiento facial. Así surgió la idea de realizar WhoRU.

Objetivos del proyecto

El proyecto ha sido enfocado como una herramienta para que los alumnos puedan obtener toda la información acerca de un profesor solamente teniendo una foto suya. De igual forma podría también enfocarse con pocas modificaciones a la inversa, es decir, para ayudar a los profesores a reconocer a sus alumnos. En un principio esta idea habría sido incluso más funcional, pero no tenemos forma de adquirir la información relativa a todos los alumnos para incorporarlos de forma automática a la base de datos de la aplicación, y habríamos gastado muchísimo tiempo de desarrollo sólo en la tarea de realizar un software que realizase la automatización de entrada de datos, o algo parecido a un scrapper. La aplicación posee un sistema de favoritos para poder tener un mejor acceso a unos determinados profesores. Por último, también está dotado de un sistema de búsqueda para poder acceder a los datos de los

profesores que se encuentren en la base de datos sin necesidad de poseer una imagen suya.

Justificación de las tecnologías usadas

Para la aplicación

Hemos elegido Android como sistema para desarrollar la aplicación principalmente porque estábamos familiarizados con el sistema, y con el desarrollo en el mismo. Además es el sistema más utilizado en el panorama actual y en principio daría acceso a más gente a la aplicación. Podríamos haber utilizado un framework web multiplataforma, pero al utilizar componentes de hardware como la cámara, ésta es mucho más rápida desde la API nativa.

Para la API de reconocimiento

Cuando comenzamos a idear el proyecto, estuvimos valorando las diferentes API's de reconocimiento e identificación facial que había en auge, ya que aún es una tecnología muy nueva. Tras bucear un poco en la documentación de las más famosas, encontramos el siguiente diagrama que básicamente resolvió nuestras dudas. La API de Microsoft aún está en sus primeras versiones y no es del todo fiable, la de Facebook tiene las peticiones muy limitadas, y Google tiene reconocimiento de caras, pero no la identificación de las mismas, pero como una imagen vale más que

mil palabras, observemos esto:

	Kairos	Amazon	Google	Microsoft	IBM	Affectiva	OpenCV
Face Detection	✔	✔	✔	✔	✔	✔	✔
Face Recognition (Image)	✔	✔	✖	✔	✖	✖	✖
Face Recognition (Video)	✔	✖	✖	✖	✖	✖	✔
Emotional Depth (%)	✔	✖	✖	✔	✖	✔	✖
Emotions Present (Y/N)	✔	✔	✔	✔	✖	✔	✖
Age & Gender	✔	✔	✖	✔	✔	✔	✖
Multi-face Tracking	✔	✔	✔	✔	✔	✔	✔
SDK	✔	✖	✖	✖	✖	✔	✔
API	✔	✔	✔	✔	✔	✔	✖
Ethnicity	✔	✖	✖	✖	✖	✖	✖

Observando esto, Kairos.io tiene disponibles todas las funcionalidades que podríamos necesitar en el desarrollo, y además la limitación de peticiones gratuitas es mucho mayor que en el resto.

Para la base de datos

En un principio pensamos utilizar sqlite, que suele ser el sistema de bases de datos que se utiliza en el desarrollo de aplicaciones móviles con base de datos local, sin embargo, controlar sqlite es un engorro, y mientras estábamos buscando información relativa al proyecto, descubrimos un

nuevo motor de bases de datos para dispositivos móviles, ligero, más rápido en búsquedas y además con la ventaja de que todas las operaciones CRUD se hacen directamente desde sus clases en JAVA.

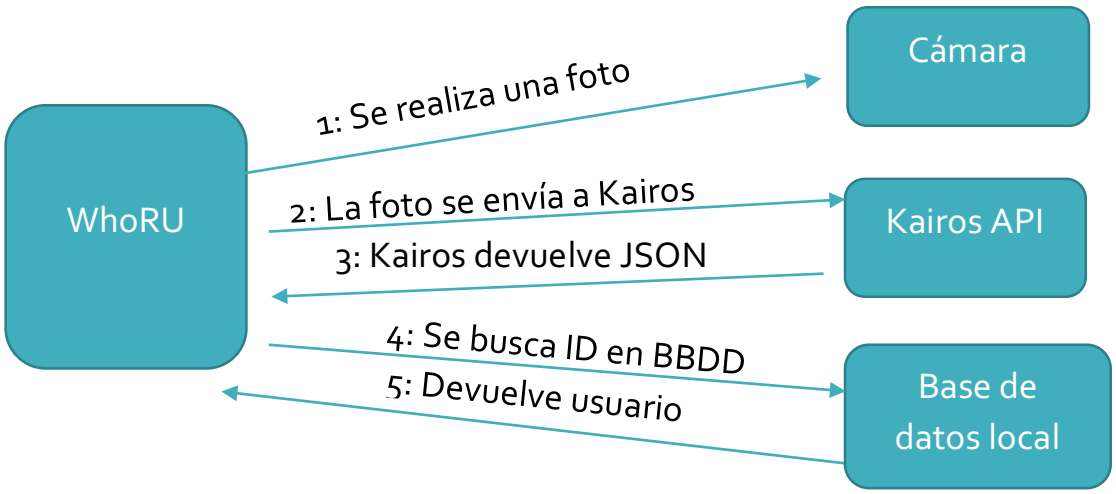
Este motor se llama Realm.io sólo con ver cómo funcionan un par de líneas, nos damos cuenta de la potencia de esta herramienta y decidimos utilizarla.

Ejemplo de uso:

```
public class Dog extends RealmObject {
  public String name;
  public int age;
}

Dog dog = new Dog();
dog.name = "Rex";
dog.age = 1;

Realm realm = Realm.getDefaultInstance();
realm.beginTransaction();
realm.copyToRealm(dog);
realm.commitTransaction();

RealmResults<Dog> pups = realm.where(Dog.class)
                              .lessThan("age", 2)
                              .findAll();
```

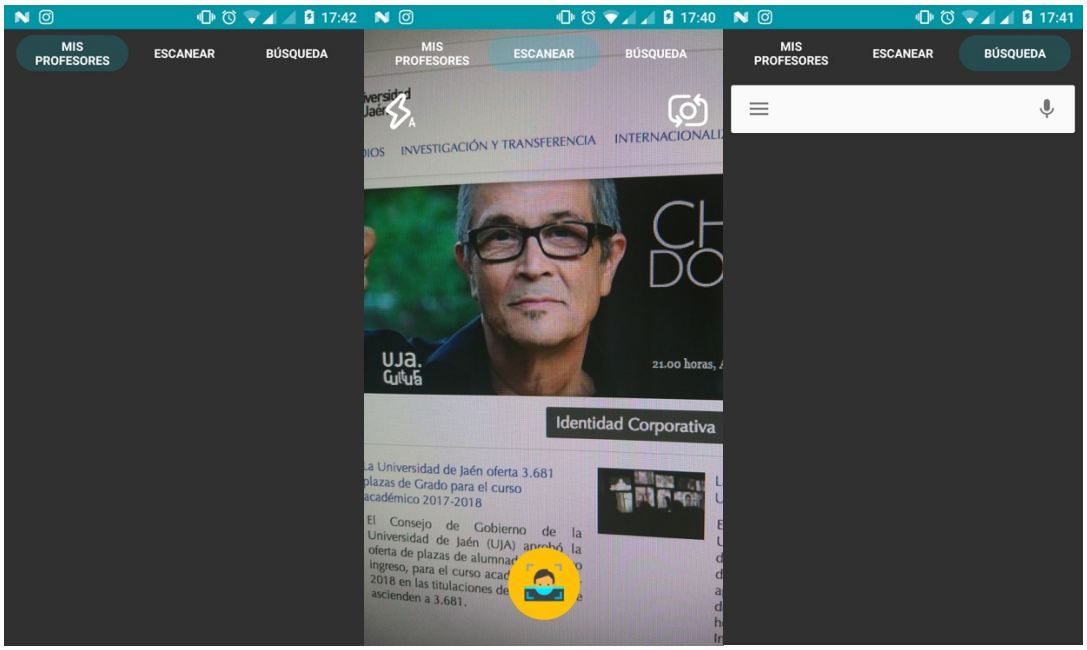
Análisis y diseño

Diagrama de comunicación



Análisis de la UI

Para la interfaz, sabíamos que necesitábamos 3 vistas principales, una para los favoritos, otra para búsqueda, y obviamente la vista para capturar desde la cámara y donde ocurre la mayor parte del funcionamiento. Queríamos una interfaz que se arrastrase hacia los lados para mostrar las diferentes vistas. En Android, este tipo de "Vista principal" que implementa un OnSwipeLeft or Right listener para cambiar de vista se llama ViewPager, componente que se explica más adelante por separado. Además hemos jugado con las transparencias, poniendo en la parte superior un selector de vistas, usando la librería SmartTabLayout para saber en qué vista estamos.



Componentes Android de diseño

Material Design

Es un estilo de diseño desarrollado por Google para los sistemas Android. Este estilo se integró en lollipop reemplazando a su predecesor Holo y poco a poco se va extendiendo al resto de productos. Google también lanzo APIs para que los distintos desarrolladores externos de Google que trabajaran con Android pudieran utilizar Material Design en sus aplicaciones.

Este diseño se caracteriza por bastante muy limpio y predominan las animaciones y las transiciones de respuesta y juega con algunos

elementos como el relleno , la iluminación y las sombras. Cuando este nuevo estilo de diseño apareció, supuso una revolución en el desarrollo de UI's ya que Material Design, pasó de utilizar una interfaz en 2D a utilizar el eje Z (elevation) permitiendo además de transiciones más vistosas, una mayor ergonomía en el diseño e infinitas nuevas funciones para los desarrolladores.

AppCompat

Son las librerías que Google nos proporciona en Android para llevar oficialmente la interfaz Material Design y otras funcionalidades sobre todo de diseño a los dispositivos con versiones anteriores a Lollipop. Así, nosotros hemos programado una aplicación desarrollada sobre la API de sistema v25, es decir para 7.1 y utilizando estas librerías la aplicación es capaz de funcionar en cualquier versión de Android desde Ice Cream Sandwitch. A grosso modo "castea" las nuevas funcionalidades transformándolas a una implementación funcional con código de API's anteriores, aunque ello signifique perder rendimiento. En la realidad, estas bibliotecas son bastante problemáticas, sobretodo por la cantidad de fabricantes que tocan el código oficial de AOSP(Android Oficial Source Project) haciendo que en ocasiones estas librerías dejen de funcionar correctamente. De hecho, incluso Google tiene bastantes bugs en las librerías ya que lógicamente se centran más en el desarrollo de nuevas API's que en dar soporte a versiones con 4-5 años de antigüedad.

RecyclerView

Es una versión más avanzada de listView. Es un contenedor para mostrar una gran cantidad de elementos y que nos podamos desplazar a través de ellos de una manera suave y eficiente.

Esta clase se encarga de simplificar los elementos que aparecen en la pantalla además de hacer más fácil la manipulación de grandes conjuntos de datos gracias a administradores de diseño para controlar la posición de los elementos y a unas animaciones predeterminadas que nos proporciona.

Para poder usar este RecyclerView tienes que especificar un adaptador y un View Holder que es un sostenedor de esas vistas.

CardView

Un cardView es un FrameLayout con un borde redondeado y con sombras. Cardview utiliza elevación para emular un sistema con distintas profundidades.

Constraint Layout

Constraint layout es una herramienta que nos permite mantener la estructura de los elementos de la aplicación sin importarnos la pantalla que posea el dispositivo que ejecuta nuestra aplicación.

View Pager

Es una vista que implementa la funcionalidad de desplazamiento horizontal y fue lanzada con el Compatibility Package, es decir, forma parte de AppCompat. Generalmente, esta vista se incluye en una actividad, y contiene fragmentos, que se cargan sobre ese ViewHolder. Los fragmentos son una especie de actividades que se pueden cargar y descargar dentro de otras actividades, permitiendo así movernos dentro de una aplicación sin tener que generar otra actividad que ocupa mucha más memoria.

Librerías Externas de diseño

SmartTabLayout

Es una librería externa que se vincula a un ViewPager y te indica la página en la que estás en tiempo real en otra vista personalizable, con ello nosotros conseguimos esa tab transparente en la parte superior que nos permite movernos por los diferentes fragmentos.

MaterialProgressBar

Es una librería que nos permite configurar barras de progreso tanto horizontales como circulares. Android ya tiene una librería para esto, pero en los terminales LG y Samsung, tienen problemas para colorearse porque estos fabricantes han tocado el código oficial relativo a esas librerías para implementar sus propias modificaciones, ello hace que no funcione bien en todos los terminales y por eso se recomienda utilizar una librería externa que funcione en todos los dispositivos. Nosotros la hemos utilizado en nuestra aplicación para mostrar el progreso de la petición a la API de Kairos.

CircleImageView

Es una vista que recorta la imagen contenida en forma circular causando un efecto bastante estético y que nosotros utilizamos en la actividad que muestra el perfil de un profesor

SearchView

Es una vista que utilizamos nosotros para realizar las búsquedas en el fragmento de búsqueda de la aplicación, tiene el mismo funcionamiento que la barra oficial de las aplicaciones de google, con animaciones y aspecto ya implementados.

CameraView

Esta vista probablemente sea la más importante de nuestra aplicación, es una vista de superficie que nos muestra en tiempo real la cámara, pudiendo asignar a esta un evento para capturar y procesar la imagen que estuviésemos viendo en la pantalla. Hemos conseguido estirarla hasta los bordes del diseño, para causar un efecto estético como en Instagram Stories y Snapchat.

Implementación

Clases

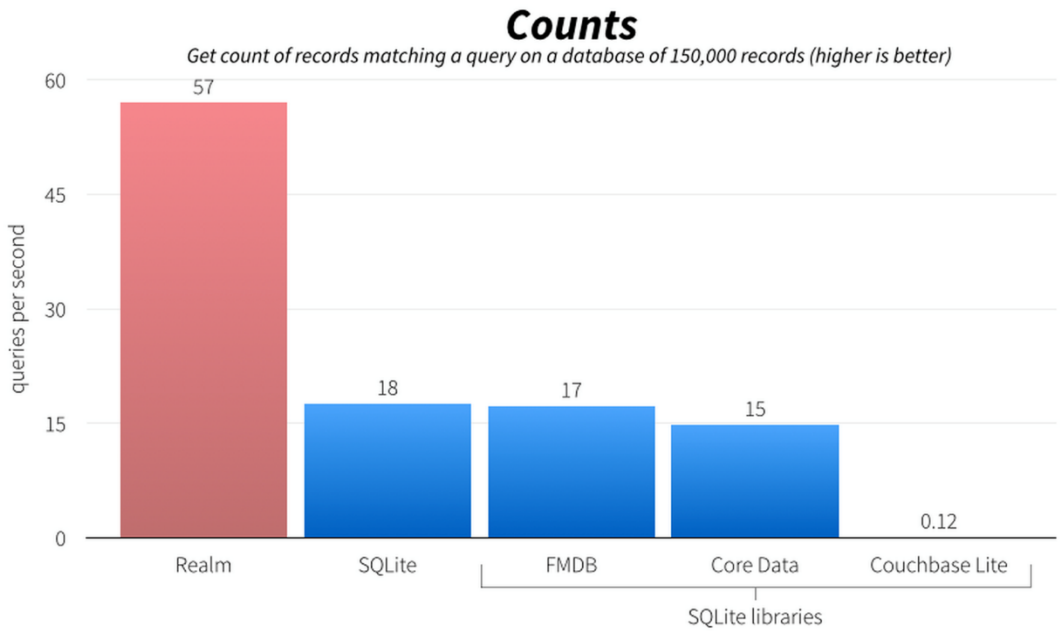
Toda la documentación relativa a las clases se encuentra en el javadoc de la aplicación, tenemos un enlace desde el apartado de Webgrafía.

Persistencia de datos

Realm Mobile Database

Es una alternativa a SQLite. Tiene un diseño de zero-copy, o sea que no se producen copias de los objetos. Es mucho más rápido que otros gestores de bases de datos. Incluso más rápido que el propio SQLite. Es

concurrente y se maneja directamente con clases en java.



Realm Adapter

Generalmente habríamos usado un adapter normal para cargar en un objeto de java el resultado de una query de nuestra base de datos. Sin embargo, utilizar realm significa usar una base de datos viva, que permite cambiar los resultados en tiempo real, así que copiar una información en un objeto definitivo, que puede cambiar, sería contraproducente. Por ello realm nos proporciona unos nuevos adapters que cargan directamente la información desde la instancia de la base de datos. Así ya no mandamos un ArrayList u otra estructura al Adapter, si no que directamente le mandamos un objeto query de nuestro Realm.

Reconocimiento

API

Como ya comentamos anteriormente usamos Kairos, así que nuestra aplicación se comunica con Kairos primero para poder introducir entradas en la base de datos de reconocimiento fácial, si detecta que esa persona tiene más fotos en la API, esta actualiza su perfil automáticamente añadiendo esa nueva imagen para mejorar el reconocimiento, si no tiene ninguna foto, la integra en la base de datos directamente.

Gson(Json)

Es una librería de Google que nos permite la serialización y el manejo de JSON de manera mucho más intuitiva. Usando clases de java con anotaciones para serializar y deserializar el JSON en objetos de Java.

Además, durante el desarrollo, encontramos una web que nos permitía introducir una respuesta JSON y nos daba como resultado los objetos en GSON necesarios para manejar esa respuesta. Recomiendo echar un vistazo JSONschema2pojo.

Camara

En nuestra aplicación hemos "embebido" nuestra previsualización de la cámara frontal y de la trasera. Además añadimos algunos botones para activar y desactivar el flash o cambiar de cámara y obviamente el botón que activa el proceso de reconocimiento facial. Esta superficie devuelve un bitmap que luego transformamos a un byte array para almacenarlo en la base de datos local.

Introducción de datos

Hemos creado 2 actividades principales para gestionar la introducción de datos, por un lado tenemos una actividad que se encarga de introducir nuevas imágenes en la base de datos de la API de Kairos y también una actividad para añadir registros a nuestra base de datos local Realm de profesores. En principio esto último no sería necesario porque en una aplicación final, nos bastaría con cargar todos los profesores desde un archivo de la propia aplicación. Los registros de Realm, se cargan en los fragmentos de búsqueda, y si están marcados como favoritos en el de Mis profesores también.

Control de versiones

Hemos utilizado git desde github como control de versiones, teniendo una rama dev y una master que hemos utilizado para sincronizarnos el proyecto entre ambos compañeros, y para mantener el código de forma más sencilla.

Tenemos más de 50 commits en git, en webgrafía se puede consultar el enlace al repositorio.

Conclusión

El desarrollo de la aplicación nos ha enseñado a aprender bastante técnicas de programación relacionadas con el procesamiento de imágenes e incluso conocimientos relativos a bases de datos y comunicación con API's, aunque ha supuesto un enorme esfuerzo, nos ha resultado bastante ameno. Esta aplicación está acabada y con pocos cambios podría llegar a implementarse con una funcionalidad real. El desarrollo libre de las prácticas creo que es mejor para nosotros como ingenieros, porque nos "obliga" a investigar y arreglar nuestros propios errores a la vez que hemos seguido aprendiendo a desarrollar una tecnología en auge en un área de desarrollo móvil.

Webgrafía

Github del proyecto

Javadoc del proyecto

SmartTabLayout

MaterialProgressBar

CircleImageView

SearchView

CameraView

Kairos

Android

Realm

JSONschema2pojo

Realm Android Adapters