AIKÉN

Aplicación móvil para turismo social

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

Enero de 2019

ANEXO IV – Documentación técnica de programación

Autor

Brenda María Rodríguez Ramírez

Tutores

Jesús Fernando Rodríguez Aragón

D. Carolina Zato Domínguez

Tabla de contenidos

Lista de figuras

1. Introducción

En este anexo se expondrá la información relativa al proyecto que pueda resultar útil para un programador, detallando los diferentes lenguajes de programación y las herramientas utilizadas.

Además, para una mejor comprensión, se especificará la estructura de los ficheros fuente del proyecto, y se proporcionará la documentación del código obtenida de manera automática con la herramienta JavaDoc [1]que posee Android Studio.

[1] JavaDoc es una utilidad proporcionada por Java SDK que permite generar documentación del código desde los archivos fuentes de Java.

2. Lenguajes de desarrollo

Cómo se ha explicado antes, se ha construido una aplicación móvil para dispositivos Android, para su implementación se ha utilizado el entorno Android Studio.

- **ANDROID STUDIO** es un nuevo entorno de desarrollo para Android basado en el IDE IntelliJ IDEA, se ha escogido este entorno debido a que incorpora nuevas características que no encontramos en el tradicional IDE basado en Eclipse. La versión del entorno con la que desarrolló el proyecto fue Android Studio 3.2.1.

Los lenguajes de programación utilizados son: JAVA y XML

- **JAVA** es un lenguaje de programación de propósito general, concurrente, orientado a objetos. Es la base para prácticamente todos los tipos de aplicaciones de red, además del estándar global para desarrollar y distribuir aplicaciones móviles, juegos, contenido basado en web y software de empresa. Este lenguaje permite a los desarrolladores de aplicaciones programar el lenguaje una única vez y sea posible ejecutarlo en cualquier dispositivo.
- **XML** es una especificación para diseñar lenguajes de marcado, permite definir etiquetas personalizadas para descripción y organización de datos. Se encarga de representar información estructurada de modo que se pueda almacenar, transmitir, procesar y visualizar por diversos tipos de aplicaciones y dispositivos. Es un lenguaje fácilmente procesable, separa radicalmente el contenido y el formato de presentación y está diseñado para cualquier lenguaje alfabeto.

La parte lógica de la aplicación será desarrollada en Java, mientras que la interfaz, desarrollará lenguaje XML.

3. Organización del proyecto

En este apartado se va a explicar la distribución del código dentro del proyecto. Dentro del proyecto se explicarán aquellas carpetas que tengan más relevancia.

3.1. Estructura del Proyecto

Cada módulo en Android está formado por:

- un descriptor de la aplicación (manifests)
- el código fuente en Java (java)
- una serie de ficheros con recursos (res)
- ficheros para construir el módulo (Gradle Scripts).

Cada elemento se almacena en su carpeta específica, indicada entre paréntesis.

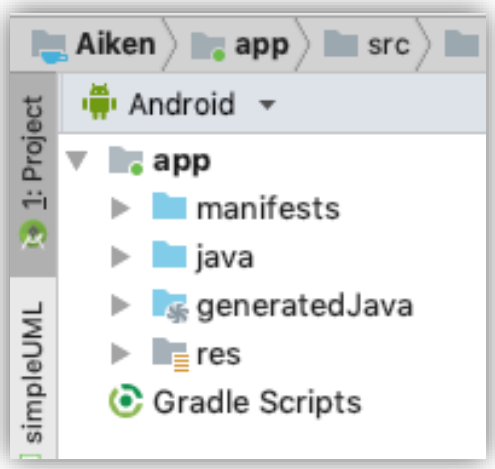


Ilustración 1. Estructura del proyecto Android

Dentro del módulo app tendremos los siguientes archivos:

- **AndroidManifest.xml:** En este fichero describe la aplicación Android. Se define su nombre, paquete, icono, estilos, etc. Se indican las actividades, las intenciones, los servicios y los proveedores de contenido de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica la versión mínima de Android para poder ejecutarla, el paquete Java, la versión de la aplicación, etc.
- **java:** Carpeta que contiene el código fuente de la aplicación. Los ficheros Java se almacenan en carpetas según el nombre de su paquete.
    - **MainActivity**: Clase Java con el código de la actividad inicial.
    - **ApplicationTest**: Clase Java pensada para insertar código de testeo de la aplicación utilizando el API JUnit.
- **res:** Carpeta que contiene los recursos usados por la aplicación.

Dentro del módulo *Gradle Scripts* se almacenan una serie de ficheros que permiten compilar y construir la aplicación. El fichero más importante es build.gradle (Module:app) que es donde se configuran las opciones de compilación del módulo:

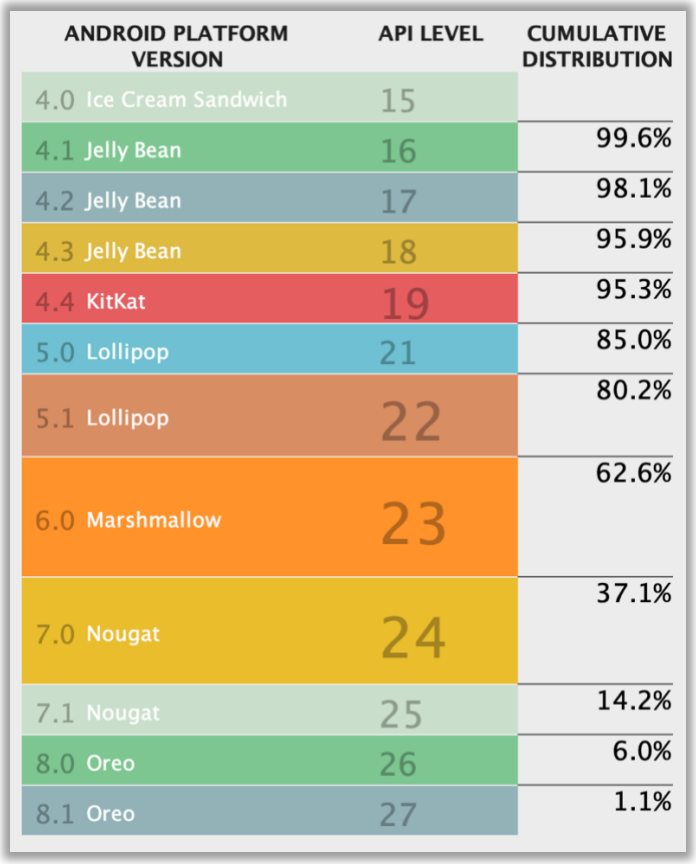
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.android.aiken"
        minSdkVersion 16
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        vectorDrawables.useSupportLibrary true

    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

Ilustración 2. Fichero Gradle(Module:app)

Los parámetros que podemos configurar son:

- **compileSdkVersion:** permite definir la versión del SDK con la se compilará la aplicación.
- **applicationId**: se utiliza como identificador único de la aplicación, de forma que no se permite instalar una aplicación si ya existe otra con el mismo id. Suele coincidir con el nombre del paquete Java creado para la aplicación.
- **minSdkVersion:** especifica el nivel mínimo de API que requiere la aplicación. Es un parámetro de gran importancia, la aplicación no podrá ser instalada en dispositivos con versiones anteriores y solo podremos usar las funcionalidades del API hasta este nivel (con excepción de las librerías de compatibilidad). Para este proyecto se optó por la API 16, que nos permitirá la mayor parte del mercado, cómo nos demuestra la *Ilustración 3. Versiones de Android.* Este valor se puede cambiar, si en determinado momento el programador desea utilizar alguna funcionalidad que requiera una versión superior, en detrimento claro del posible nicho de mercado.

- **targetSdkVersion**: indica la versión más alta con la que se ha puesto a prueba la aplicación.
- **versionCode** y **versionName:** indican la versión de la aplicación.
- **buildTypes:** Dentro se añaden otras configuraciones dependiendo del tipo de compilación que queramos (relase para distribución, debug para depuración, etc.).



ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.6%
4.2 Jelly Bean	17	98.1%
4.3 Jelly Bean	18	95.9%
4.4 KitKat	19	95.3%
5.0 Lollipop	21	85.0%
5.1 Lollipop	22	80.2%
6.0 Marshmallow	23	62.6%
7.0 Nougat	24	37.1%
7.1 Nougat	25	14.2%
8.0 Oreo	26	6.0%
8.1 Oreo	27	1.1%

Ilustración 3. Versiones de Android

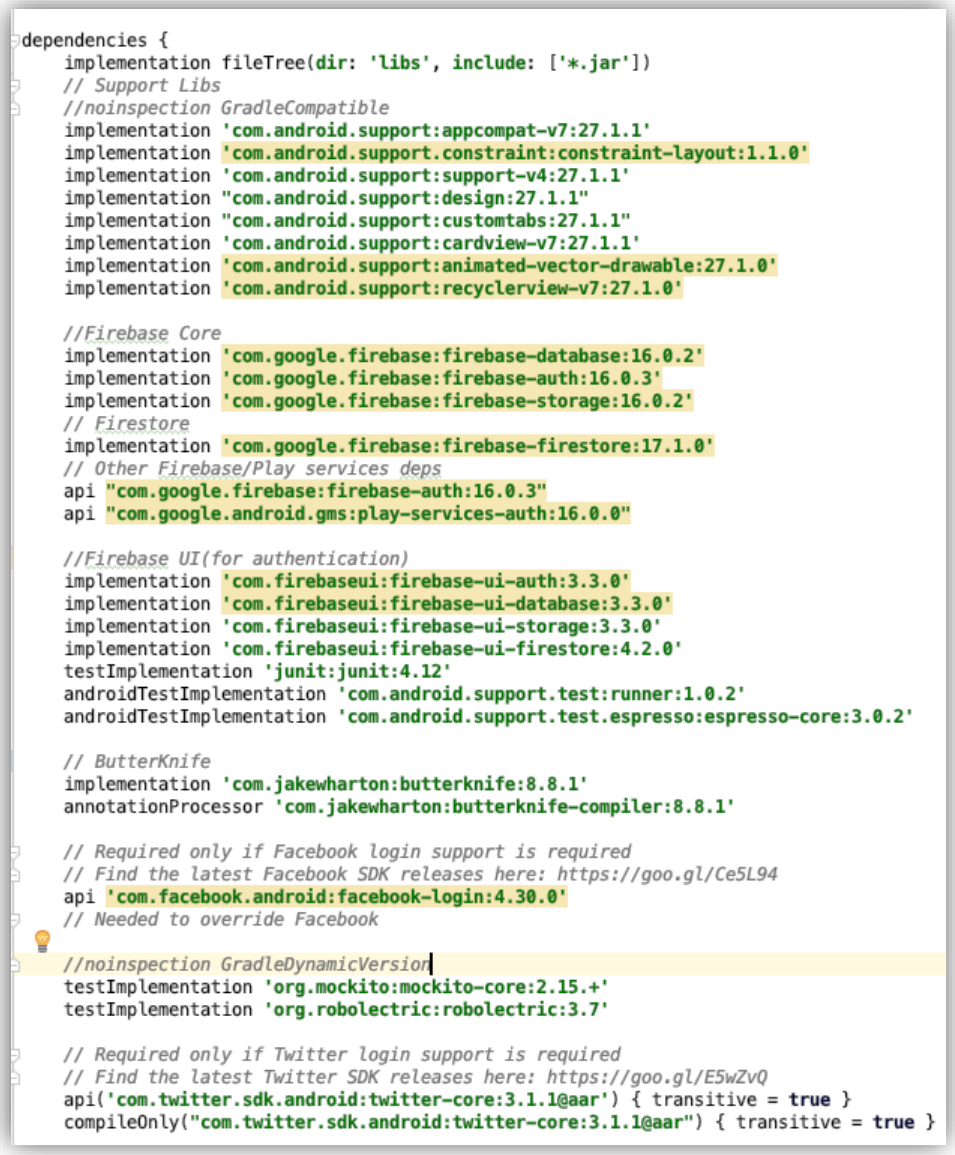
Un apartado importante es el de **dependencies**. En él se debe indicar todas las librerías que han de ser incluidas en el proyecto: librerías de compatibilidad (Support Libs), librerías requeridas para el correcto funcionamiento de Firebase, el registro con Facebook y Twitter, y finalmente las librerías de terceros.

Entre las librerías de compatibilidad incluidas, las principales son las **v7 Libraries**, y se corresponden con librerías que pueden usarse a partir del API 7 (v2.1):

- **v7 appcompat library:** Permite utilizar un IU basado en la Barra de Acciones siguiendo especificaciones de Material Design. Se añade por defecto cuando creamos

un nuevo proyecto. Incorpora las clases: ActionBar, AppCompatActivity, AppCompatDialog y ShareActionProvider.

- **v7 recyclerview library:** Incorpora la vista RecyclerView, una versión mejorada que reemplaza a ListView y GridView.
- **v7 cardview library:** Incorpora la vista CardView, una forma estándar de mostrar información especialmente útil en Android Wear y TV.
- **Design Support Library**: Librería que incorpora varios componentes de Material Design.



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    // Support Libs
    //noinspection GradleCompatible
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.0'
    implementation 'com.android.support:support-v4:27.1.1'
    implementation "com.android.support:design:27.1.1"
    implementation "com.android.support:customtabs:27.1.1"
    implementation 'com.android.support:cardview-v7:27.1.1'
    implementation 'com.android.support:animated-vector-drawable:27.1.0'
    implementation 'com.android.support:recyclerview-v7:27.1.0'

    //Firebase Core
    implementation 'com.google.firebase:firebase-database:16.0.2'
    implementation 'com.google.firebase:firebase-auth:16.0.3'
    implementation 'com.google.firebase:firebase-storage:16.0.2'
    // Firestore
    implementation 'com.google.firebase:firebase-firestore:17.1.0'
    // Other Firebase/Play services deps
    api "com.google.firebase:firebase-auth:16.0.3"
    api "com.google.android.gms:play-services-auth:16.0.0"

    //Firebase UI(for authentication)
    implementation 'com.firebaseui:firebase-ui-auth:3.3.0'
    implementation 'com.firebaseui:firebase-ui-database:3.3.0'
    implementation 'com.firebaseui:firebase-ui-storage:3.3.0'
    implementation 'com.firebaseui:firebase-ui-firestore:4.2.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

    // ButterKnife
    implementation 'com.jakewharton:butterknife:8.8.1'
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.8.1'

    // Required only if Facebook login support is required
    // Find the latest Facebook SDK releases here: https://goo.gl/Ce5L94
    api 'com.facebook.android:facebook-login:4.30.0'
    // Needed to override Facebook

    //noinspection GradleDynamicVersion
    testImplementation 'org.mockito:mockito-core:2.15.+'
    testImplementation 'org.robolectric:robolectric:3.7'

    // Required only if Twitter login support is required
    // Find the latest Twitter SDK releases here: https://goo.gl/E5wZvQ
    api('com.twitter.sdk.android:twitter-core:3.1.1@aar') { transitive = true }
    compileOnly("com.twitter.sdk.android:twitter-core:3.1.1@aar") { transitive = true }
```

Ilustración 4. Gradle (Module:app): Dependencias

3.2. Librerías de terceros

Durante el desarrollo de esta aplicación, ha surgido la opción de utilizar librerías de terceros, para evitar tener que desarrollar funcionalidades que ya están desarrolladas. La ventaja de la ***reutilización de código*** es que dichas librerías están ampliamente testeadas, y se ahorrará tiempo. Las librerías, todas con licencia Apache, son las siguientes:

- **MaterialRatingBar:** Esta librería nos permitirá mejorar el Widget RatingBar de Android, y así poder ver con una mejor apariencia las estrellas que representan la calificación de un tour.
- **FabSpeedDial:** Esta librería nos permitirá desplegar distintas opciones al pulsar en el Floating Action Button del detalle del tour.
- **Circleindicator:** Esta librería nos permitirá añadir un circulo a las fotos de detalle de un tour, para poder desplazarte sobre ellas con el simple gesto del dedo.
- **Picasso:** Esta librería permite cargar imágenes en ImageViews de manera rápida, sin mucho consumo de memoria y en una sola línea de código.

A continuación, se muestran tres imágenes que se ofrecen desde sus respectivas cuentas de GitHub, y nos ilustra mejor sobre su tarea.

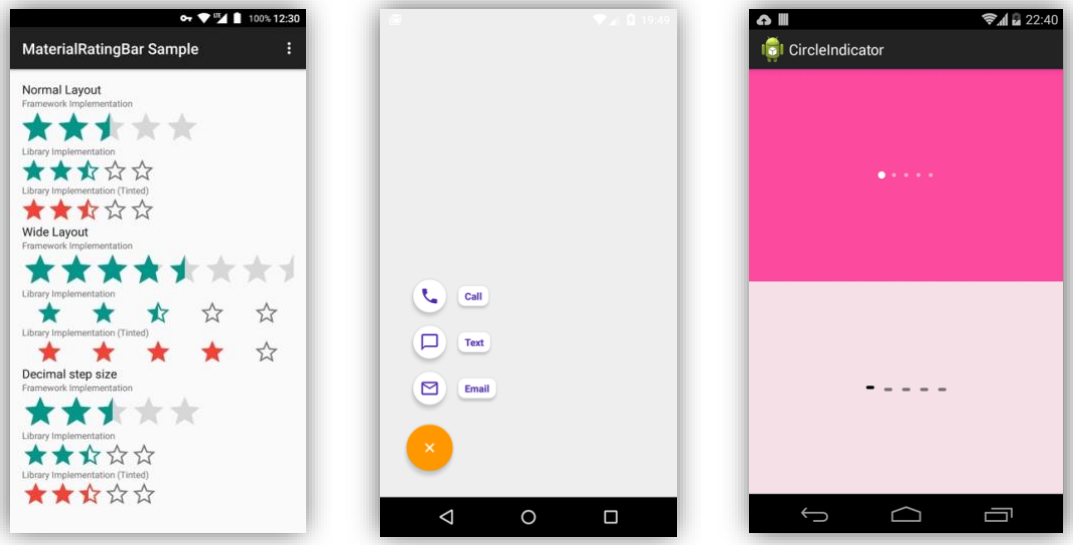


Ilustración 5. Librerías de terceros

3.3. Código fuente

3.3.1 Código JAVA

El código de desarrollo de este proyecto software se encuentra en la carpeta *java*. Durante el desarrollo se han dividido en paquetes, atendiendo al tipo de la clase o su finalidad, de este modo es más fácil su manejo y organización dado el elevado número de clases.

Dentro del paquete ***"com.android.aiken"*** encontraremos los ficheros java. que contendrán toda la lógica de la aplicación. Dichos ficheros han sido explicados en el Anexo 3, Apartado 4.2 Glosario de clases.

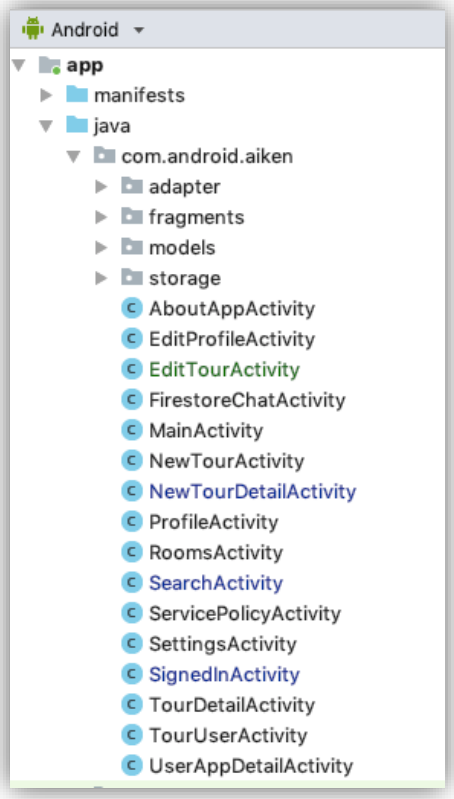


Ilustración 6. Paquete java

A continuación, se detallan los paquetes en los que están organizados los ficheros:

- **"com.android.aiken.adapter":** Este paquete contendrá los ***adaptadores de vista***, y los objetos ***view holder*** y ***view model***, que nos permitirán personalizar cada elemento (tour, usuarios, chat...), recuperar los datos desde la base de datos, representarlos de una manera eficiente (ViewHolder), y asegurándose de que sobreviven a cambios de configuración de la *Activity* (ViewModel).

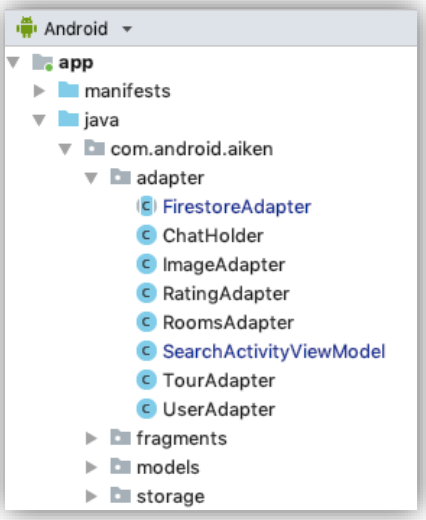


Ilustración 7. Paquete "adapter"

- **"com.android.aiken.storage":** Éste paquete contiene una única clase, que nos permitirá subir, descargar y guardar en memoria cache las imágenes que se deseen mostrar. Para ello, utilizaremos la librería Glide; [2]

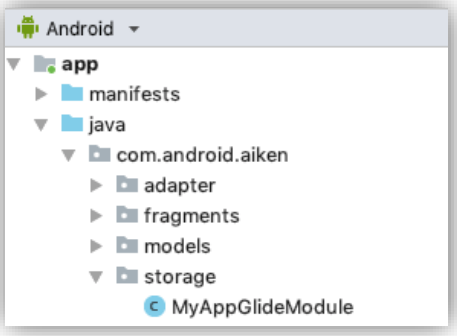


Ilustración 8. Paquete "storage"

- **"com.android.aiken.fragments":** Este paquete contendrá los *Fragments* que hemos creado para la aplicación. Un fragment está formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario, y nos permiten combinar uno o varios fragments dentro de una actividad. El paquete está dividido en dos subpaquetes:
  - **BottomNavigationView**, que es un patrón de navegación que se sitúa en la parte inferior de la pantalla facilitando de esta manera la exploración y el cambio entre vistas de nivel superior con un solo toque. Cada fragment del paquete representará una vista diferenciada de la información de un usuario.
  - **TabLayout**, cada fragment del paquete representará una vista distinta, de los tours *Favoritos* y *Publicados* de un usuario

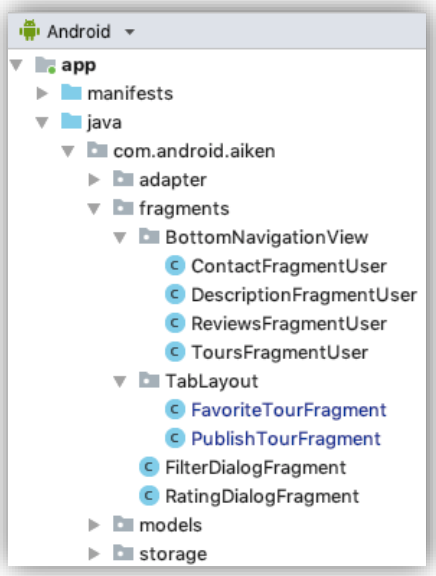


Ilustración 9. Paquete "fragments"

[2] Glide es una librería para Android que fue introducida por Google

- **“com.android.aiken.models”:** Este paquete contendrá las clases que representan el tratamiento de la información del sistema, es decir, son las clases que se comunican directamente con la base de datos.

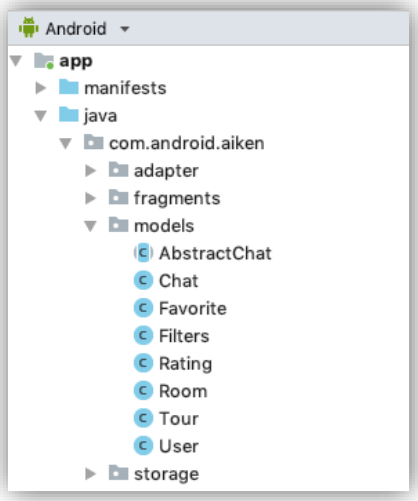


Ilustración 10. Paquete "models"

3.3.2 Código XML

Como se ha indicado en el Apartado *3.1 Estructura del Proyecto*, todos los recursos de la aplicación estarán contenidos en la carpeta **res.** A continuación, se explicará el contenido de cada subpaquete:

- **anim:** Contiene ficheros XML con animaciones de vistas.

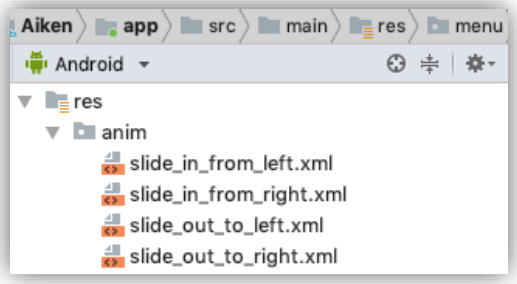


Ilustración 11. Paquete "anim"

- **drawable:** En esta carpeta se almacenan los ficheros descriptores de imágenes en XM, cada fichero representa a un icono de la aplicación.

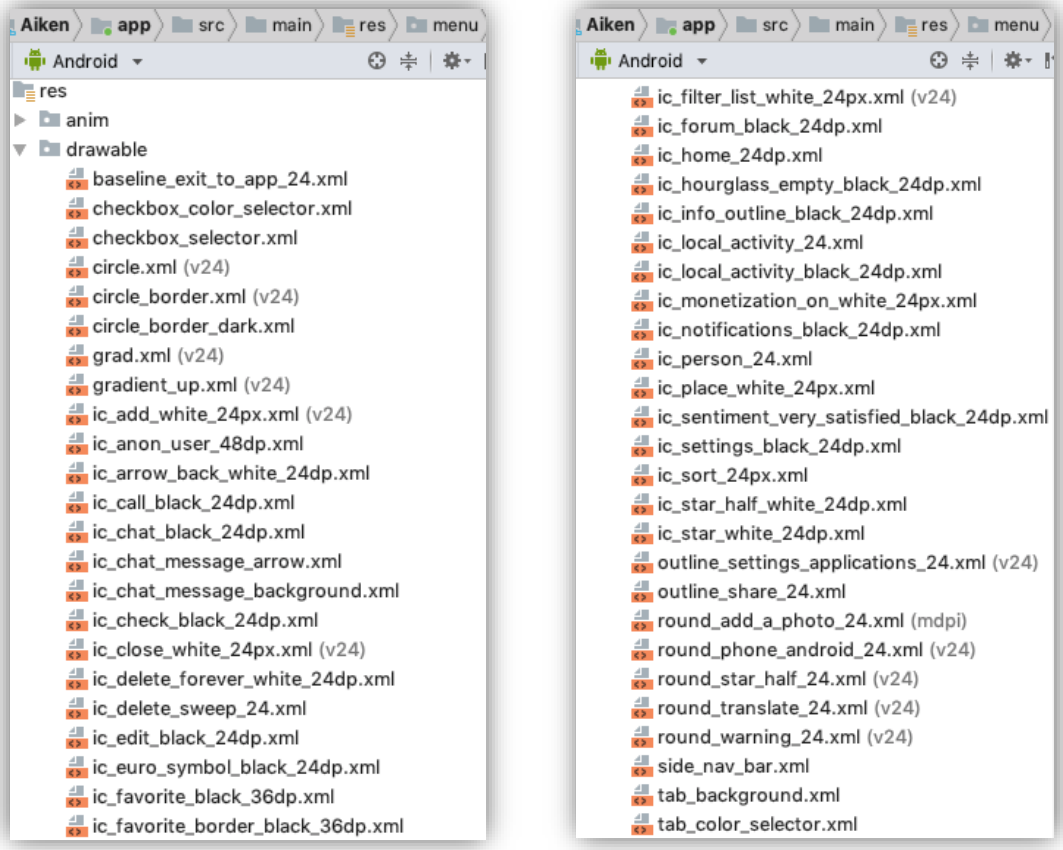


Ilustración 12. Paquete "drawable"

- **menu:** Contiene ficheros XML con los menús de cada actividad.

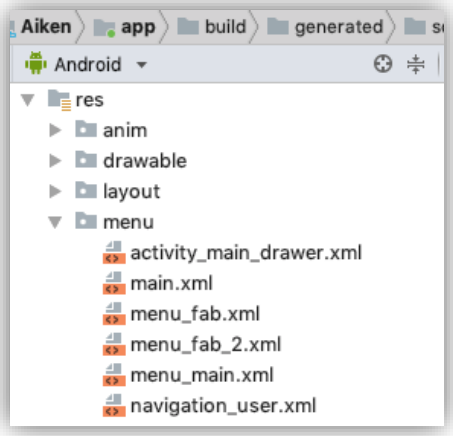


Ilustración 13. Paquete "menu"

- **layout:** Contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación.

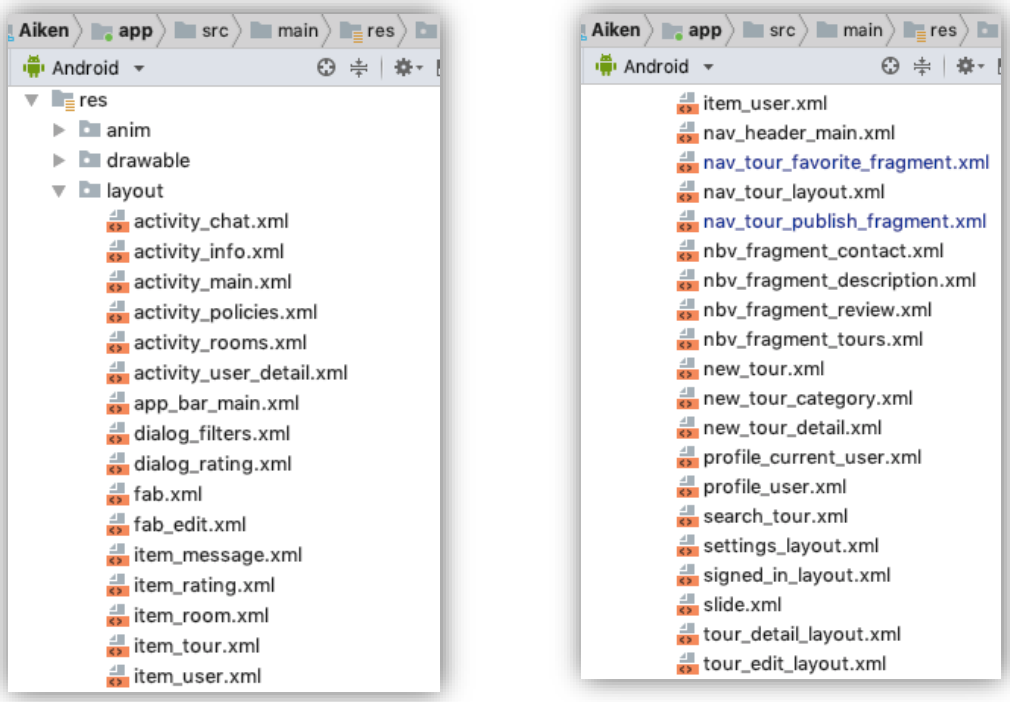


Ilustración 14. Paquete "layout"

- **values:** Contiene ficheros XML para indicar valores usados en la aplicación, de esta manera podremos cambiarlos desde estos ficheros sin necesidad de ir al código fuente.
    - **colors.xml**: se definen los tres colores primarios de la aplicación.
    - **dimens.xml**: se define el margen horizontal y vertical por defecto.
    - **strings.xml:** se definen todas las cadenas de caracteres, que nos permitirá traducir una aplicación a otro idioma.
    - **styles.xml**: se definen los estilos y temas de la aplicación.

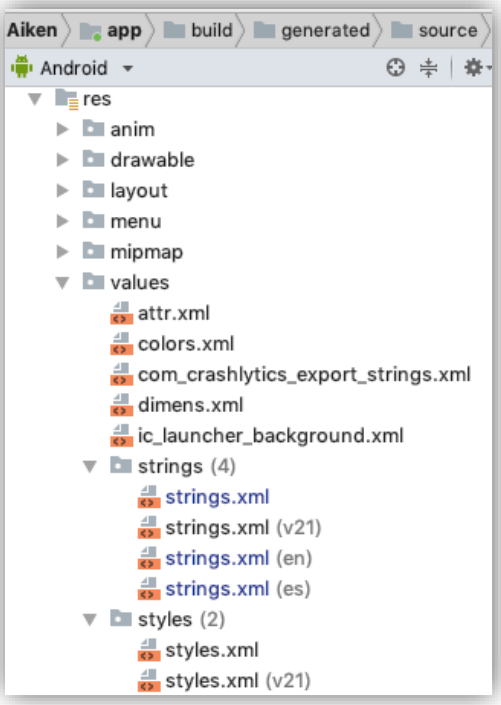


Ilustración 15. Paquete "values"

3.4. Recursos gráficos

Todos los recursos gráficos utilizados para la aplicación se encontrarán en “res.mipmap”. Esta carpeta contendrá el icono de la aplicación y demás imágenes que harán falta para la realización del proyecto.

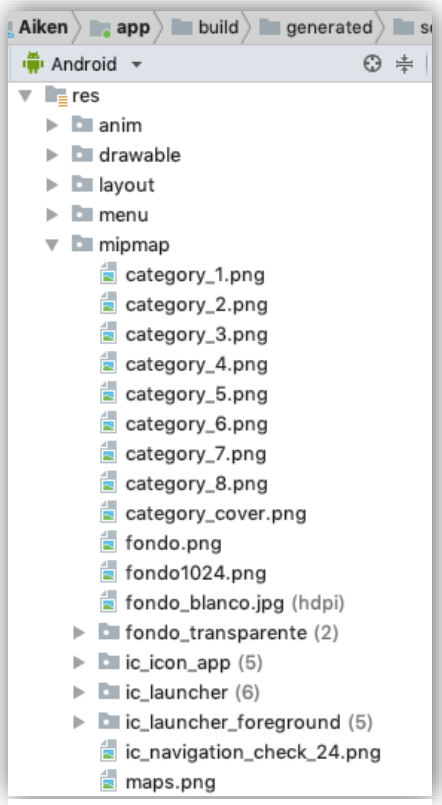


Ilustración 16. Paquete "mipmap"

3.5. Plataforma de desarrollo: Firebase

Firebase se trata de una plataforma de Google pare el desarrollo de aplicaciones móviles y web, cuya principal función es proporcionar un conjunto de herramientas orientadas a la creación de aplicaciones de alta calidad.

Para el programador, Android Studio ofrece Firebase Assistant, que es la manera más simple de conectar la aplicación con Firebase, y sirve de guía proporcionando la información necesaria para cada servicio. También, te conecta directamente la aplicación con la consola, para facilitar su gestión:

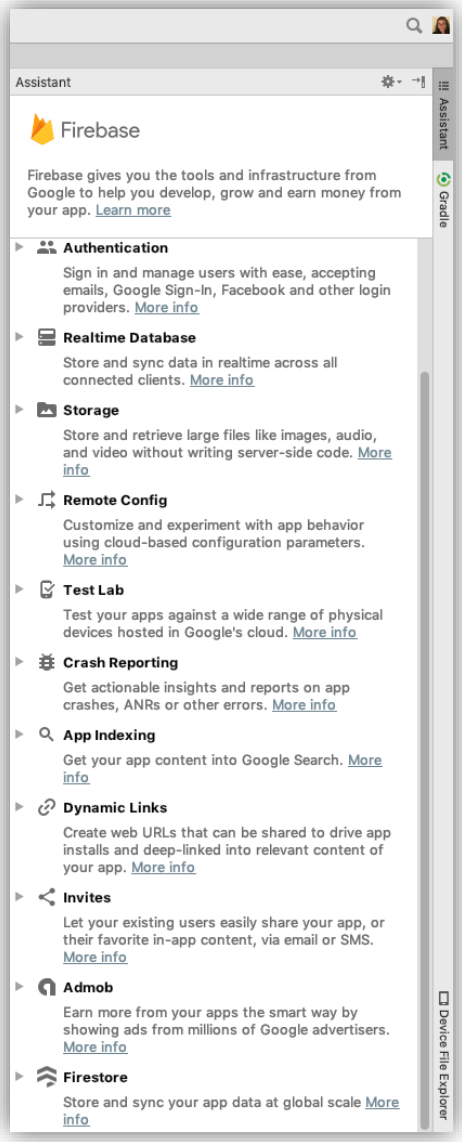


Ilustración 17. Firebase Assistant

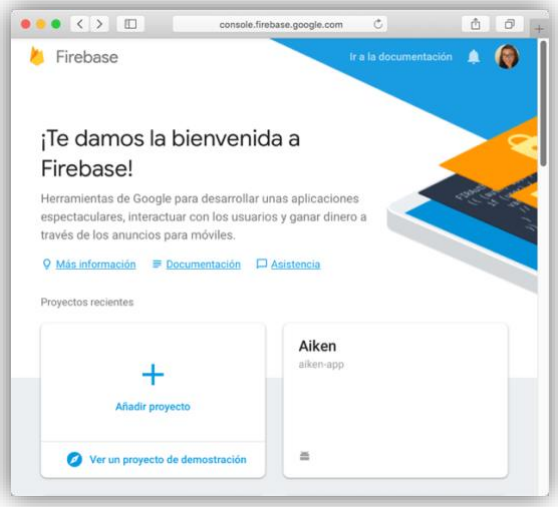


Ilustración 18. Firebase Console

Para este proyecto se ha utilizado los siguientes servicios:

3.5.1 Firebase Authentication

Permite administrar usuarios de manera simple y segura. Firebase Auth ofrece varios métodos para autenticar, entre ellos correo electrónico y proveedores externos. Desde la consola, el programador, podrá modificar los métodos de inicio de sesión, editar una posible plantilla para la recuperación de contraseñas o correo de confirmación, e instrucciones acerca de su uso.

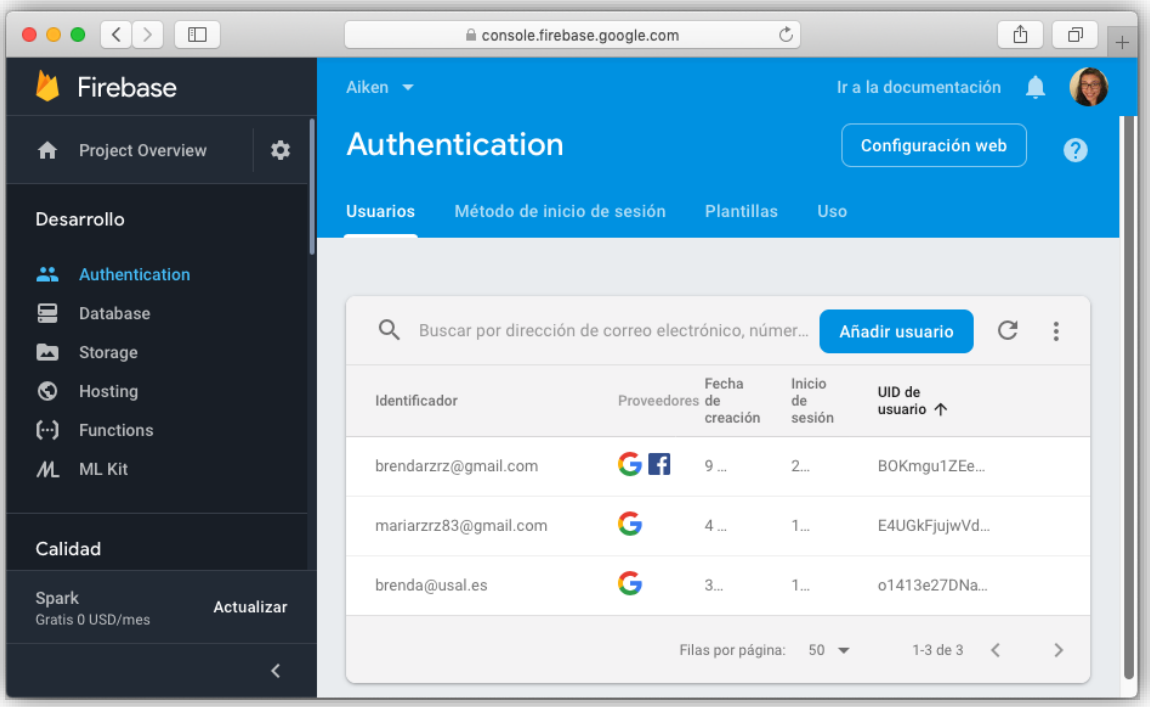


Ilustración 19. Consola de Firebase Authentication

3.5.2 Cloud Firestore

Es una base de datos de documentos NoSQL que permite almacenar, sincronizar y consultar fácilmente datos de apps móviles y web a escala global.

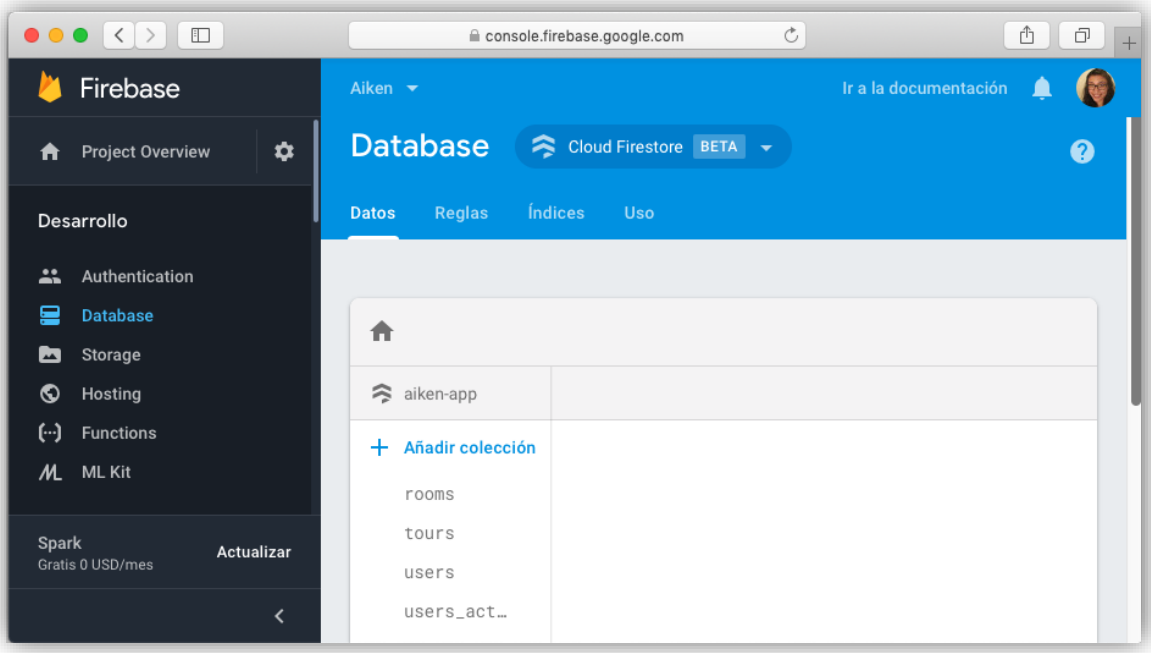


Ilustración 20. Consola de Firebase Database: Cloud Firestore

3.5.3 Cloud Storage

Almacena y comparte imágenes, audio, video y otros tipos de contenido generado por los usuarios fácilmente con el almacenamiento de objetos potente, simple y rentable creado para la escala de Google.

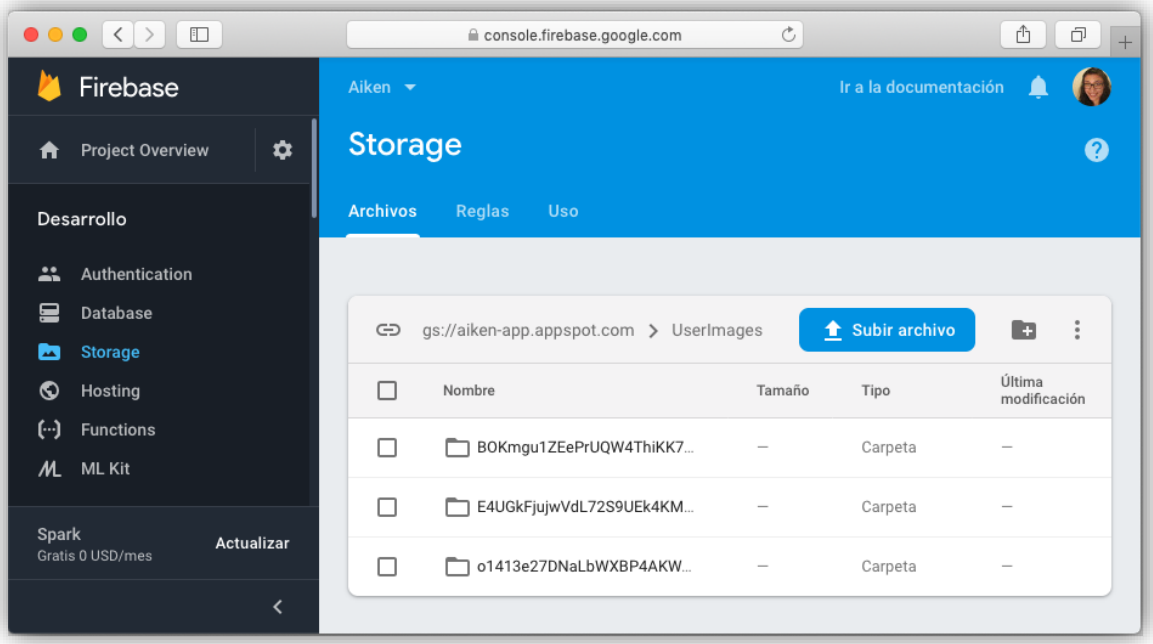


Ilustración 21. Consola de Firebase Cloud Storage

4. Manual del programador

En este documento se describe el proceso seguido para la generación de la documentación de las distintas clases que forman la aplicación.

Se ha hecho uso de la herramienta JavaDoc, que es una herramienta de generación de documentación a partir de código fuente mediante la inserción de bloques de comentarios específicos dentro del código del programa.

4.1. Utilización de JavaDoc

Como se ha mencionado anteriormente, el método de funcionamiento de JavaDoc es a través de bloques de comentarios que comienzan con los caracteres /** y finalizan con */. Dentro de ellos, y mediante una serie de etiquetas, se especifican el formato y la información relevante que se desea incluir en la documentación técnica.

A continuación, se detallan las utilizadas.

- @author: especifica el autor de la clase.
- @version: especifica la versión de una clase.
- @param: sirve describir los parámetros de entrada de los métodos.
- @return: sirve describir los valores de retorno de los métodos.
- @throws / @exception: describe las excepciones generadas por los métodos.
- @see: sirve indicar el nombre de la clase o el método asociado.
- @since: especifica la versión API en la que se incluyó la clase o método.
- @deprecated: indica que la clase o método está obsoleta.

Una vez introducidos estos comentarios en el código, se puede generar la documentación desde la pestaña ***Tools*** de Android Studio.

La documentación técnica del programador puede encontrarse en el soporte digital adjunto en la ruta Documentación/JavaDoc/index.html.

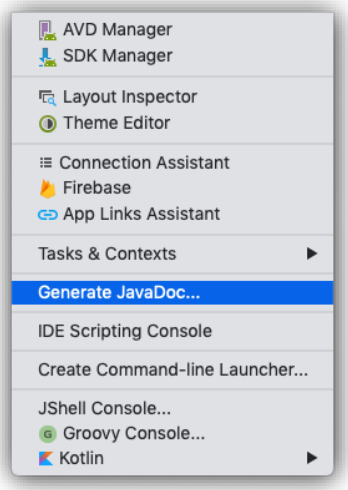


Ilustración 22. Generar JavaDoc

4.2. Depurar y probar

Android Studio nos permite dos maneras de depurar y testear el código: por medio de un dispositivo virtual o por un dispositivo físico.

4.2.1 Pruebas con un dispositivo virtual (AVD)

Un Dispositivo Virtual Android (AVD), nos permite emular desde el ordenador diferente tipos de dispositivos basados en Android.

Esto nos permite probar la aplicación en una gran variedad de teléfonos, tabletas relojes o TV con cualquier versión de Android, tamaño de pantalla o tipo de forma.

Podemos crear un nuevo dispositivo virtual desde la pantalla en la que la vamos a ejecutar la aplicación, seleccionando "Create New Virtual Device", o podemos ir a la pestaña ***Tools*** y seleccionar directamente AVD Manager.

Una vez seleccionada una de las dos opciones, se nos abrirá el AVD Manager, desde donde se podrá escoger el tipo de dispositivo y las características deseadas.

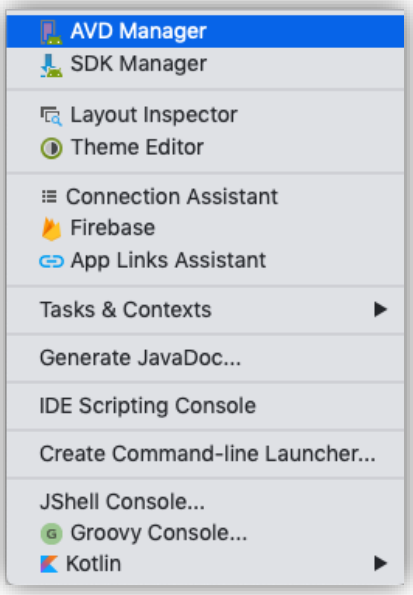


Ilustración 23. Crear un dispositivo virtual Android (AVD)

Ilustración 24. Android Virtual Device Manager

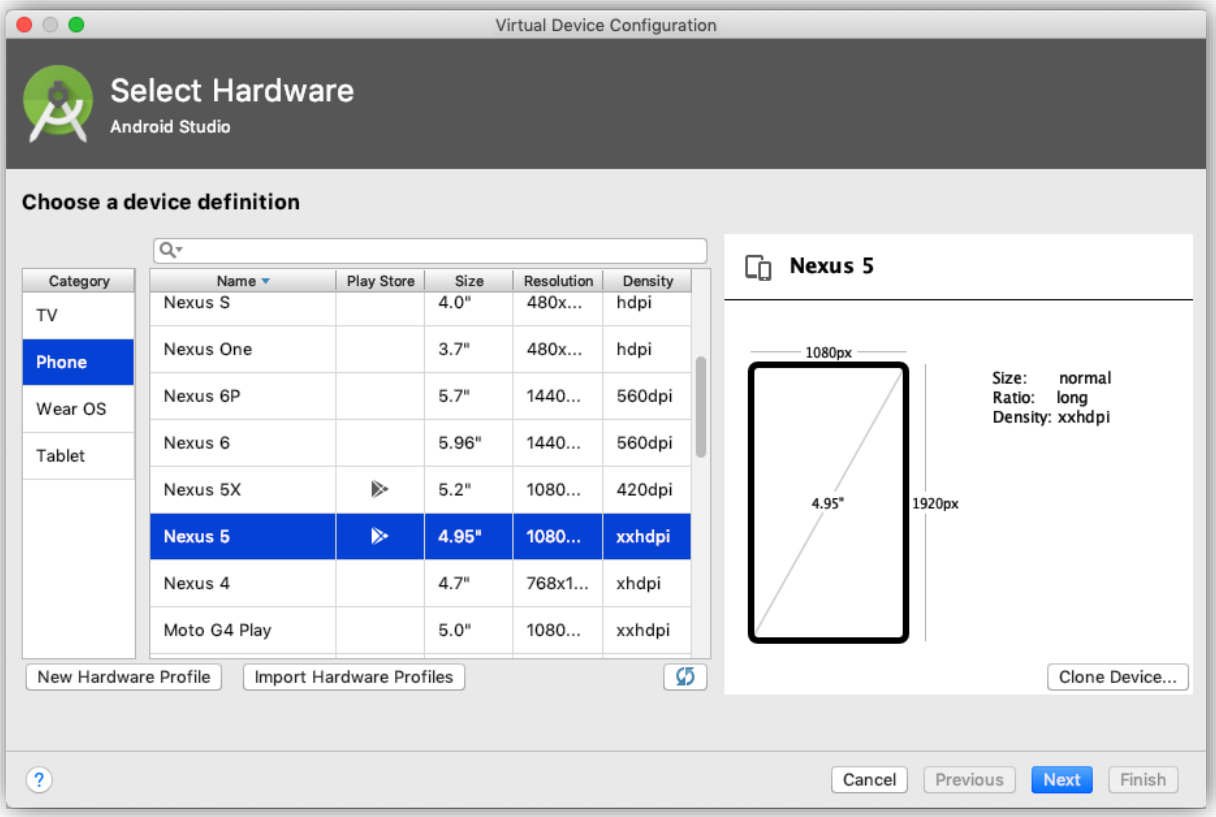


Ilustración 25. Configuración del AVD

En la columna "Category" podremos seleccionar el tipo de dispositivo a emular y en la siguiente tabla junto al nombre de cada dispositivo, se indica si tiene la posibilidad de incorporar Google Play, el tamaño de la pantalla en pulgadas, la resolución y el tipo de densidad gráfica. Una vez introducida la configuración deseada, pulsa el botón *Finish* y aparecerá el dispositivo creado en la lista.

Si al ejecutar la aplicación seleccionamos el nuevo dispositivo virtual, podremos probar la aplicación.

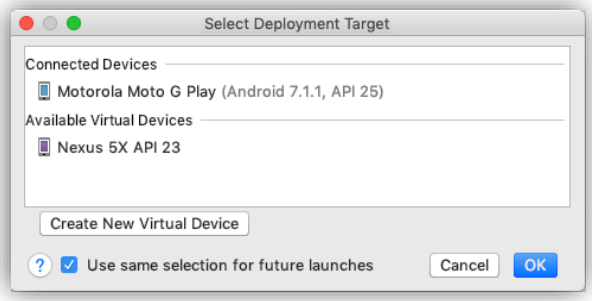


Ilustración 26. Seleccionar dispositivo

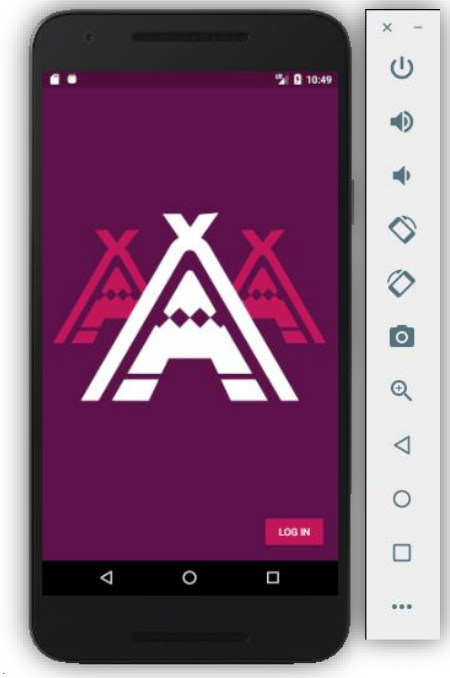


Ilustración 27. Ejemplo en dispositivo virtual

4.2.2 Pruebas con un dispositivo físico

Es posible depurar y realizar pruebas en un terminal real. Esta opción es más rápida y fiable que el emulador. Simplemente hay que instalar un driver genérico en el PC, se encuentra en la carpeta de instalación del SDK \sdk\extras\google\usb_driver, y luego usar un cabrle USB para conectar el terminal al PC.

5. Bibliografía

Android Developers. (25 de abril de 2018). Obtenido de Arquitectura de la plataforma: https://developer.android.com/guide/platform

Documentación oficial de Firebase. (14 de febrero de 2018). Obtenido de Firebase: https://firebase.google.com/docs/guides/

Documentation for app developers. (s.f.). Obtenido de Android Developers: https://developer.android.com/docs/

Durán y Bernárdez. (Universidad de Sevilla de 2002). *Metodología para la Elicitación de Requisitos de Sistemas Software*.

Girones, J. T. (2018). *El gran libro de Android (7ª ED.).* [Ed. S.A. Marcombo].

Ken Schwaber y Jeff Sutherland. (Julio de 2013). *La Guía Definitiva de Scrum: Las Reglas del Juego*. Obtenido de Scrum Guides: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Larman, G. (2002). *UML y Patrones.* [Ed. Prentice-Hall].

Material Design: Design, Develop and Tools. (s.f.). Obtenido de https://material.io

Plataforma EdX. (s.f.). Obtenido de Android: Introducción a la Programación: https://courses.edx.org/courses/course-v1:UPValenciaX+AIP201x+1T2018/course/