MANUAL TÉCNICO

Contenido

1. PROPÓSITO

El presente documento tiene como finalidad de proveer información sobre la estructura de la aplicación móvil Android implementada. Conocer sus partes, la manera de cómo fue construida, etc.

2. ANDROID STUDIO IDE

2.1. Introducción

Es el entorno de desarrollo integrado (IDE) exclusivo para Android desarrollado por Google Inc. Incluye el SDK y el emulador de un terminal. Además ofrece las siguientes características:

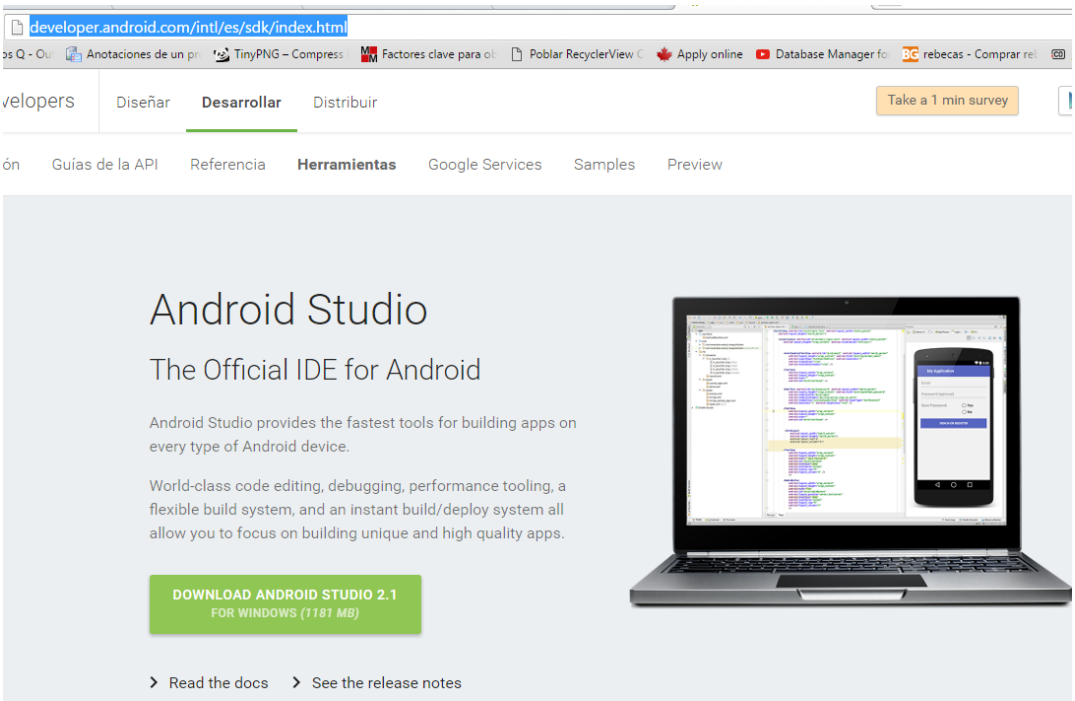
- Un entorno de desarrollo claro y robusto.
- Facilidad para testear el funcionamiento en otros tipos de dispositivos.
- Asistentes y plantillas para los elementos comunes de programación en Android.
- Un completo editor con muchas herramientas extra para agilizar el desarrollo de nuestras aplicaciones.

Al crear un nuevo proyecto en Android Studio, la estructura del proyecto aparece con casi todos los archivos dentro del directorio SRC, un cambio a un sistema de generación basado Gradle que proporcionará una mayor flexibilidad para el proceso de construcción. Además, gracias a su sistema de emulación integrado, Android Studio permite ver los cambios que realizamos en nuestra aplicación en tiempo real, pudiendo además comprobar cómo se visualiza en diferentes dispositivos Android con distintas configuraciones y resoluciones de forma simultánea.

Se seleccionó éste IDE, porque lleva más de 4 años en el mercado y ha sido catalogado cómo excelente por los programadores a nivel mundial, por encima de su predecesor el IDE Eclipse.

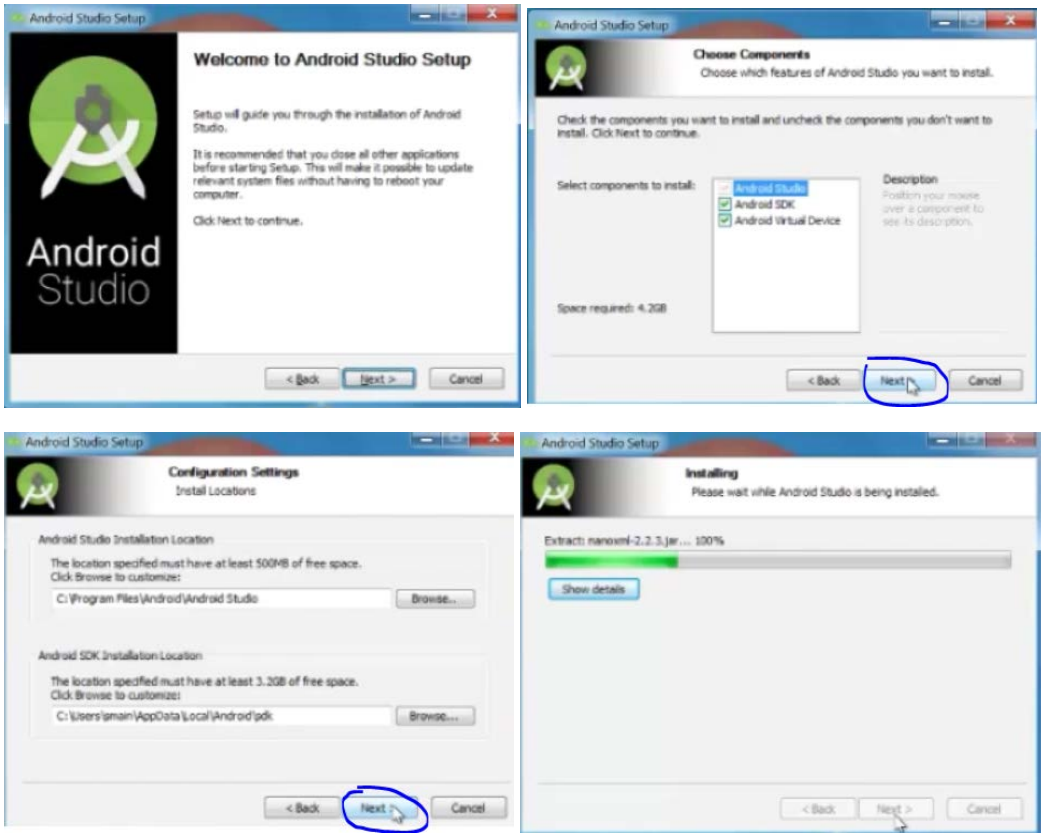
2.2. Instalación

Para descargar el Android Studio IDE, nos dirigimos a la página oficial de Android, http://developer.android.com/intl/es/sdk/index.html, y descargamos el paquete que muestra la página.



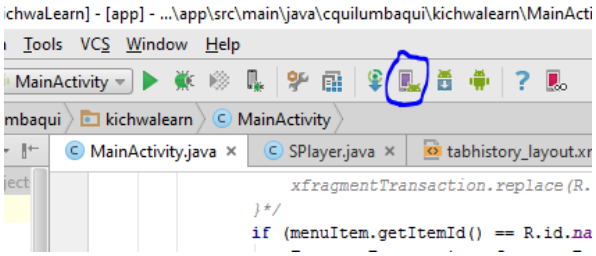
Antes de ejecutar el paquete descargado, se debe verificar que versión de JDK se tiene instalado en la computadora. La versión mínima de JAVA es la 1.8. En caso de no tener instalado JAVA, proceda a instalarlo.

En el paquete descargado se incluye el JKD y el emulador de Android para realizar los test. A continuación unas capturas de la instalación.

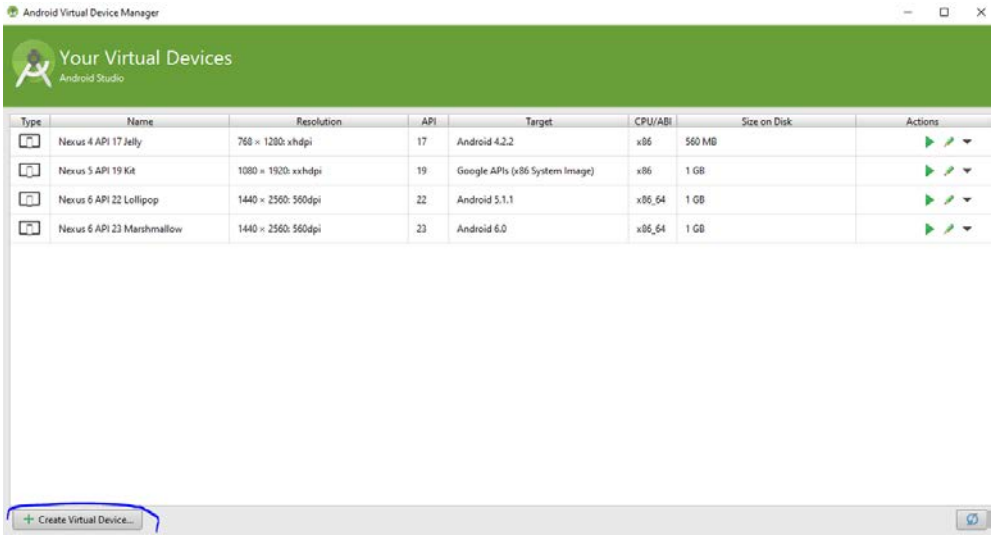


2.3. Emulador

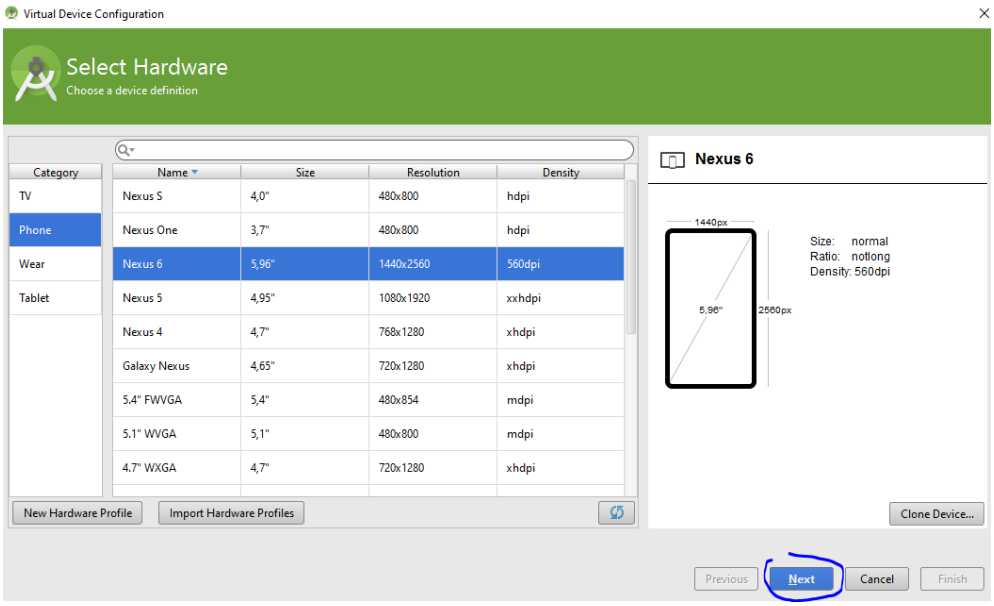
Cuando se instala el AndroidStudio, viene incluido el administrador de emulador. Para lo cual se debe hacer clic en el icono “AVD MANAGER”, para abrir la ventana donde se puede administrar los diferentes emuladores de acuerdo al hardware y software que se requiera.



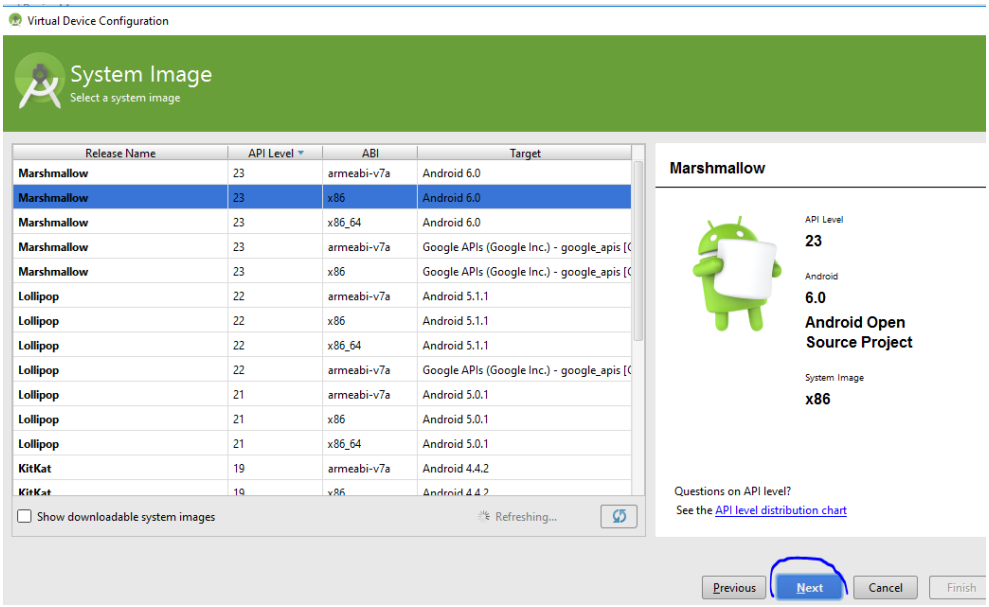
En esta ventana nos muestra una lista de emuladores creados, si deseas crear un emulador nuevo se debe hacer clic en el botón “CREATE VIRTUAL DEVICE”.



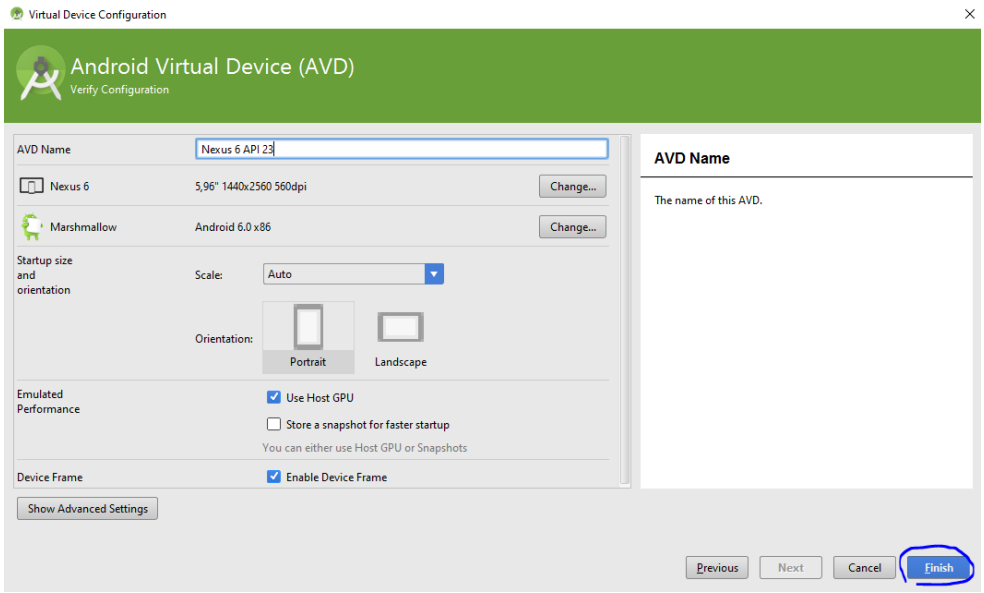
A continuación vamos a crear un emulador con el hardware de un NEXUS 5 de Google.



Y la versión del sistema operativo de Android, es con la que viene por defecto que es Marshmallow, Android 6.0.

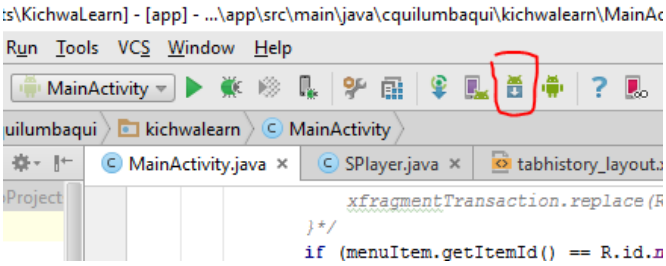


Hacemos clic en siguiente, click en finalizar y obtendremos nuestro emulador listo para arrancarlo y realizar las pruebas de la app que se vaya a desarrollar.

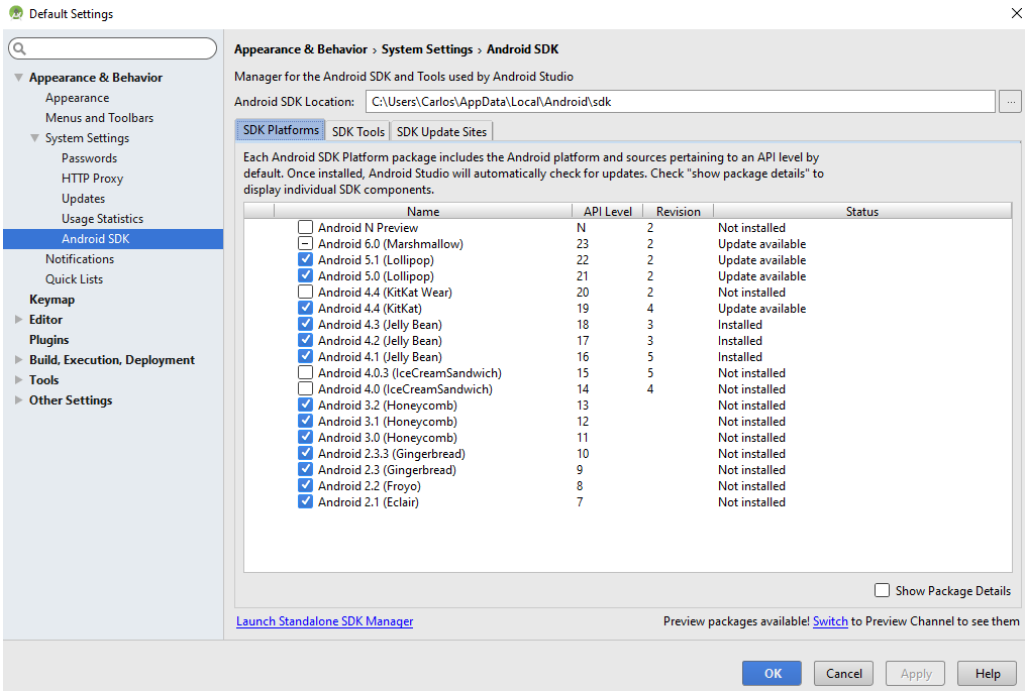


2.4. Descarga e instalación de versiones

Ahora, que pasa si queremos hacer pruebas con versiones de Android más antiguas. Para esto debemos recurrir al botón “SDK MANAGER”, como se muestra en la figura.

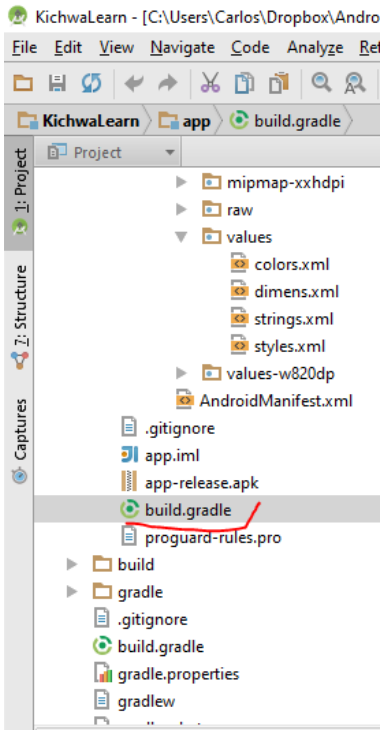


En esta ventana, en la pestaña “SDK PLATFORM”, nos mostrará una lista de las versiones de Android que se encuentran instaladas. Si deseas instalar o desinstalar alguna versión, solamente se debe seleccionar y dar click en “OK”. Una vez finalizada las descargas correspondientes de las versiones de Android, estarán disponibles para crear en el “AVD MANAGER”.

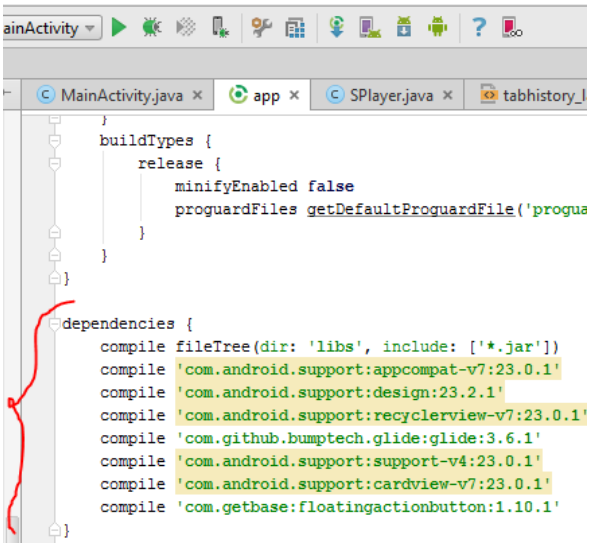


2.5. Librerías

Para la implementación de la app, recurrí a la utilización de algunas librerías que provee Google, las cuales se configura en el archivo “build.gradle”, y se encuentra tal y como se muestra en la figura.



Aquí se procede a escribir las sentencias tal y como se muestra en la figura.

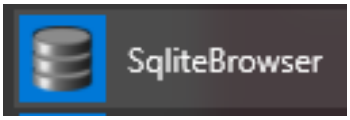


Una vez hecho esto, realizamos un “Build”, para que se sincronice las librerías al proyecto.

3. SQLITE

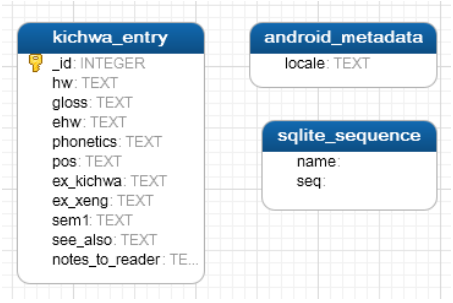
3.1. Herramientas

Para el presente proyecto se realizó con el gestor de base de datos SQLite, utilizando para la administración los pequeños programas: SQLiteBrowser y Kata Kuntur.



3.2. Diagrama Entidad Relacion

El diagrama entidad relación de la app, cuyo contenido se muestra el diseño del esquema de la base de datos del aplicativo, es el siguiente:



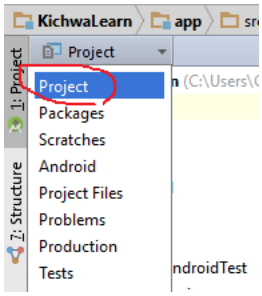
En la tabla “kichwa_entry”, se almacena más de 3000 registros correspondientes al diccionario Kichwa – Inglés, con todas las traducciones, sinónimos, oraciones de ejemplos, etc.

4. PROYECTO APP

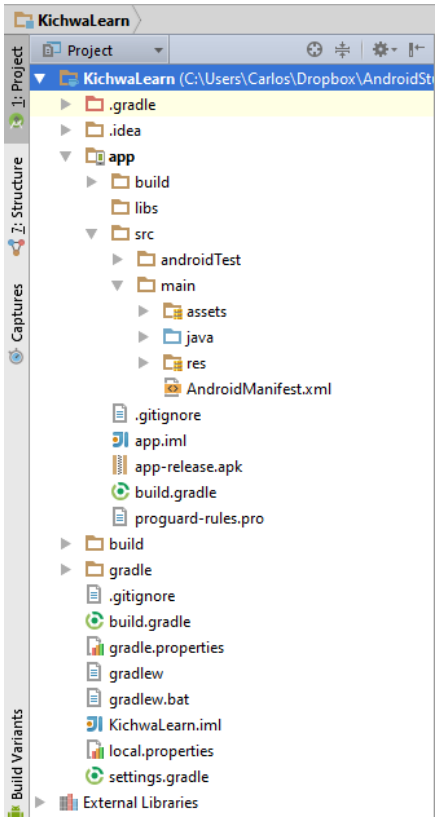
4.1. Estructura

En el IDE Android Studio, en la parte de la izquierda desplegamos las carpetas del proyecto y observamos su estructura en forma de árbol de directorios. Sin embargo por defecto lo vemos de una forma un tanto peculiar que podría llevarnos a la confusión, para entender mejor la

estructura del proyecto vamos a cambiar momentáneamente la forma en que AndroidStudio nos muestra. Para ello pulsaremos sobre la lista desplegable situada en la parte superior de la izquierda, y la cambiaremos la vista de proyecto al modo “Project”.

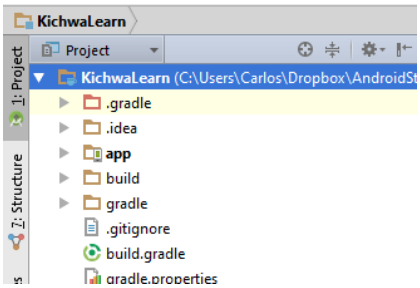


Tras hacer esto, la estructura del proyecto cambia un poco el aspecto y pasa a ser como se observa en la siguiente imagen.



A continuación describiremos los elementos principales de la estructura.

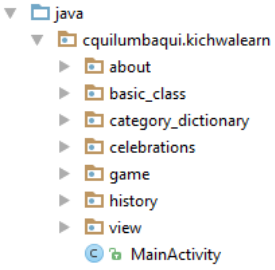
Lo primero que debemos distinguir son los conceptos de proyecto y módulo. La entidad proyecto es única, y engloba a todos los demás elementos. Dentro de un proyecto podemos incluir varios módulos, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación, o distintos componentes de un sistema (aplicación móvil, aplicación servidor, librerías, …). En la mayoría de los casos, trabajaremos con un proyecto que contendrá un sólo módulo correspondiente a nuestra aplicación principal. Por ejemplo en este caso que estamos creando tenemos el proyecto “KichwaLearn” que contiene al módulo “app” que contendrá todo el software de la aplicación de ejemplo.



A continuación se describe los contenidos principales de la aplicación central.

Carpeta /app/src/main/java

Esta carpeta contendrá todo el código fuente de la aplicación, clases auxiliares, etc. Inicialmente, Android Studio creará por nosotros el código básico de la pantalla (actividad o activity) principal de la aplicación, que recordemos que en nuestro caso era MainActivity, y siempre bajo la estructura del paquete java definido durante la creación del proyecto.



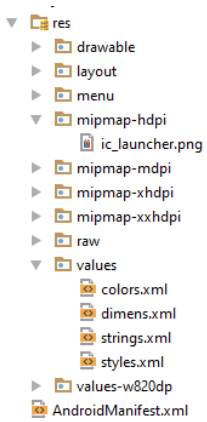
Carpeta /app/src/main/res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, layouts, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas:

Carpeta	**Descripción**
/res/drawable/	Contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas: /drawable (recursos independientes de la densidad) /drawable-ldpi (densidad baja) /drawable-mdpi (densidad media) /drawable-hdpi (densidad alta) /drawable-xhdpi (densidad muy alta) /drawable-xxhdpi (densidad muy muy alta :)
/res/mipmap/	Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de

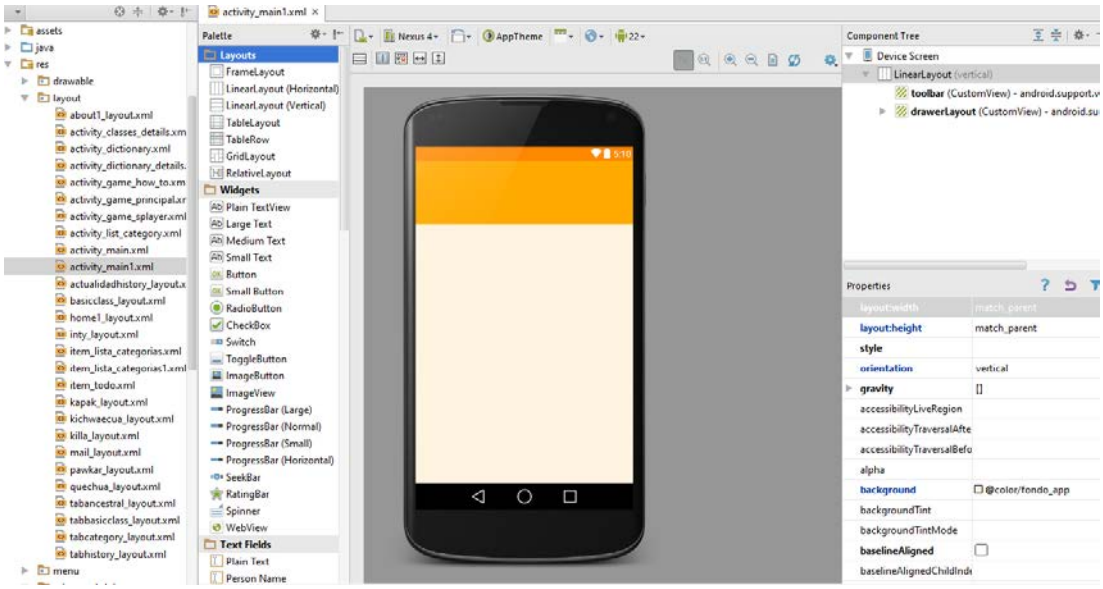
	pantalla existentes. Al igual que en el caso de las carpetas /drawable, se dividirá en varias subcarpetas dependiendo de la densidad de pantalla: /mipmap-mdpi, /mipmap-hdpi, /mipmap-xhdpi, etc.
/res/layout/	Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos layouts dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas: /layout (vertical), /layout-land (horizontal)
/res/anim/ /res/animator/	Contienen la definición de las animaciones utilizadas por la aplicación.
/res/color/	Contiene ficheros XML de definición de listas de colores según estado.
/res/menu/	Contiene la definición XML de los menús de la aplicación.
/res/xml/	Contiene otros ficheros XML de datos utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.
/res/values/	Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), arrays de valores (arrays.xml), tamaños (dimens.xml), etc.

No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten. Como ejemplo, para un proyecto nuevo Android como el que hemos creado, tendremos por defecto los siguientes recursos para la aplicación:

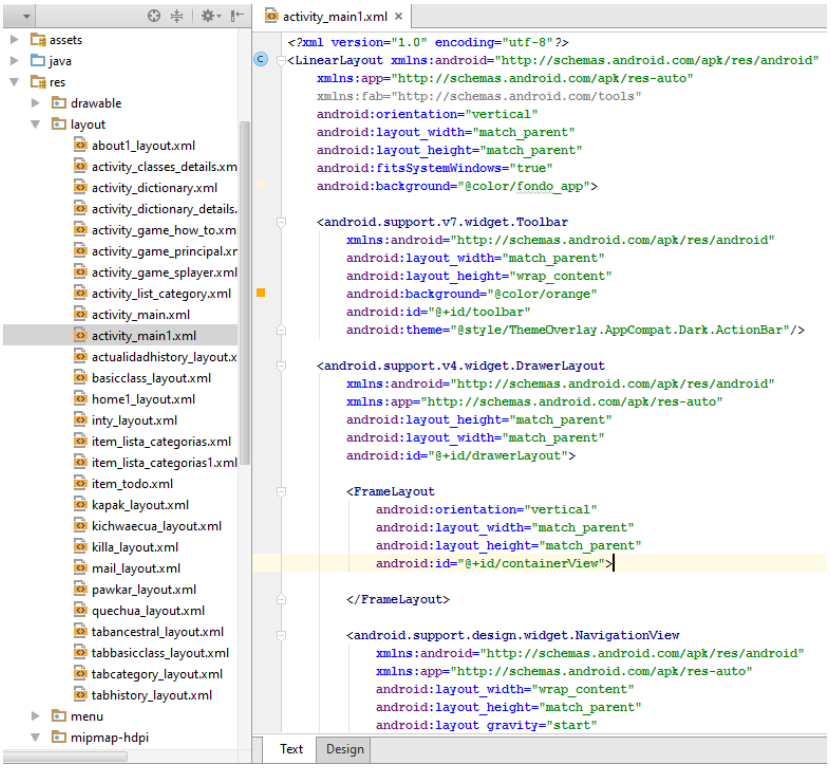


Entre los recursos creados por defecto cabe destacar los *layouts*, en nuestro caso ya se tiene algunos layouts creados, pero el principal es el llamado “activity_main1.xml”, que contienen la definición de la interfaz gráfica de la pantalla principal de la aplicación. Si hacemos doble clic

sobre este fichero Android Studio nos mostrará esta interfaz en su editor gráfico, y como podremos comprobar, contiene toda la interfaz gráfica inicial.

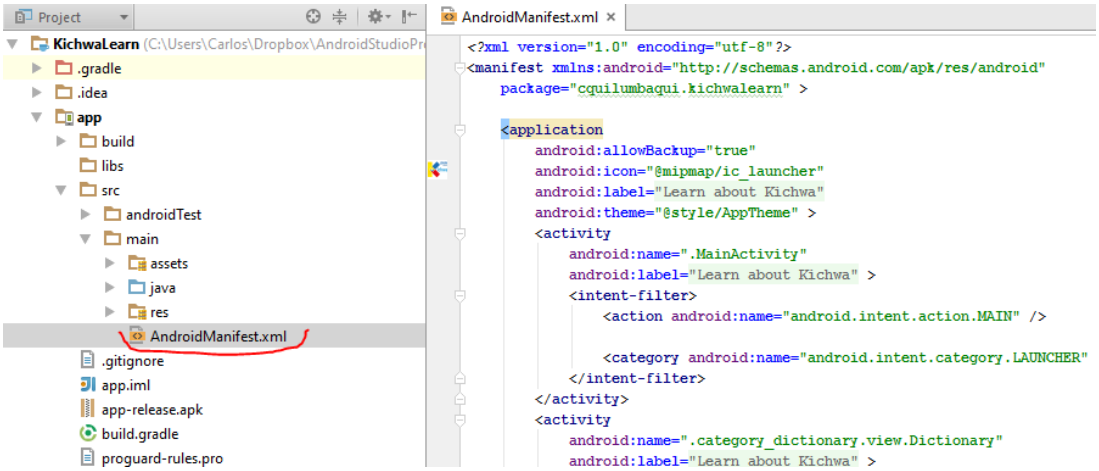


Pulsando sobre las pestañas inferiores “Design” y “Text” podremos alternar entre el editor gráfico (tipo arrastrar-y-soltar), mostrado en la imagen anterior, y el editor XML que se muestra en la imagen siguiente:



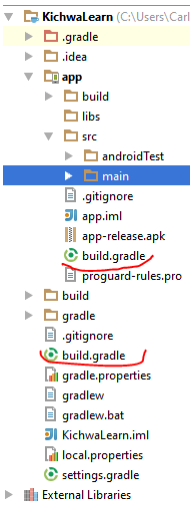
Fichero /app/src/main/AndroidManifest.xml

Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, …), sus componentes (pantallas, servicios, …), o los permisos necesarios para su ejecución. Veremos más adelante más detalles de este fichero.



Fichero /app/build.gradle

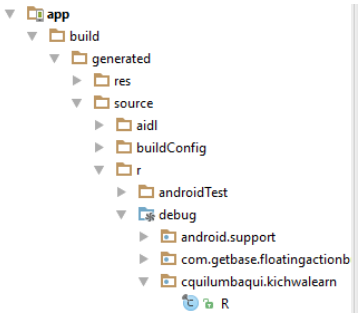
Contiene información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc. Más adelante veremos también más detalles de este fichero.



En un proyecto pueden existir varios ficheros *build.gradle*, para definir determinados parámetros a distintos niveles. Por ejemplo, en nuestro proyecto podemos ver que existe un fichero*build.gradle* a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero de ellos definirá parámetros globales a todos los módulos del proyecto, y el segundo sólo tendrá efecto para cada módulo en particular.

Carpeta /app/build/

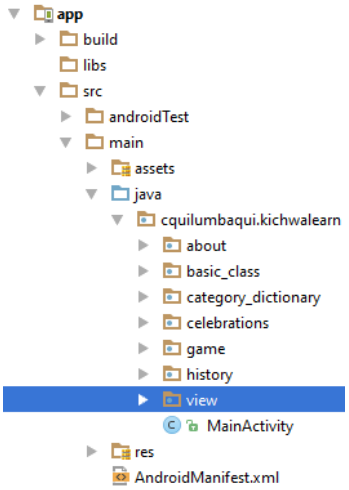
Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que compilamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente java dirigidos, entre otras muchas cosas, al control de los recursos de la aplicación. **Importante**: dado que estos ficheros se generan automáticamente tras cada compilación del proyecto es importante que no se modifiquen manualmente bajo ninguna circunstancia.



A destacar sobre todo el fichero que aparece desplegado en la imagen anterior, llamado "R.java", donde se define la clase R. Esta clase R contendrá en todo momento una serie de constantes con los identificadores (ID) de todos los recursos de la aplicación incluidos en la carpeta /app/src/main/res/, de forma que podamos acceder fácilmente a estos recursos desde nuestro código java a través de dicho dato. Así, por ejemplo, la constante R.layout.activity_main contendrá el ID del layout "activity_main1.xml" contenido en la carpeta /app/src/main/res/layout/.

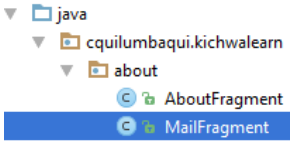
4.2. Paquetes y clases

El proyecto contiene 7 paquetes que contienen distintas clases JAVA que siguiendo el patrón MVC otorgan funcionalidad al aplicativo.

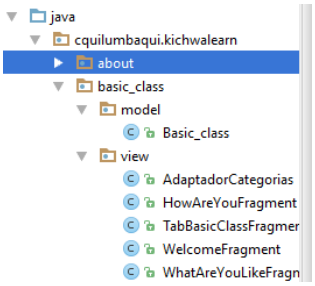


Prosiguiendo se explicarán cada uno de los paquetes a continuación:

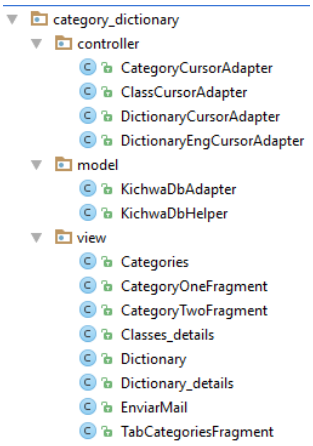
- **about.-** posee las clases cuyos métodos consisten en el envío de un mail (sugerencia, contacto, etc.) al desarrollador desde la app, además de mostrar la interfaz gráfica de los desarrolladores de la aplicación.



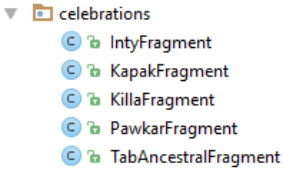
- **basic_class.-** posee las clases clasificadas con el patrón MVC, donde se implementa las "clases básicas" de aprendizaje del Kichwa. Ésta diseñada en un Tab y con Fragments.



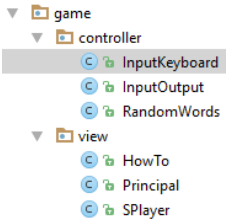
- **category_dictionary.-** contiene las clases JAVA divididas de acuerdo al patrón MVC, donde se implementa la funcionalidad del diccionario Kichwa – inglés con la utilización de la base de datos sqlite.



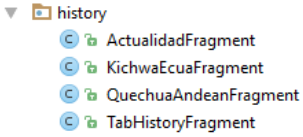
- **celebrations.-** posee las clases donde se implementa las interfaces gráficas informativas de las cuatro celebraciones ancestrales del pueblo Kichwa Otavalo. Se utiliza para el diseño Tabs y Fragments.



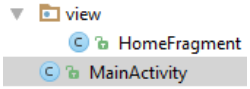
- **game.-** posee las clases donde se implementa el popular juego “el ahorcado”, dividido en dos capas: controlador y la vista. En el paquete de controlador se implementa el teclado portátil y como aparecen las palabras kichwa de manera aleatoria dentro de una lista predefinida.



- **history.-** posee las clases donde se implementa las interfaces gráficas informativas divididas en tres etapas de la historia de la lengua Kichwa Otavalo. Se utiliza para el diseño Tabs y Fragments.

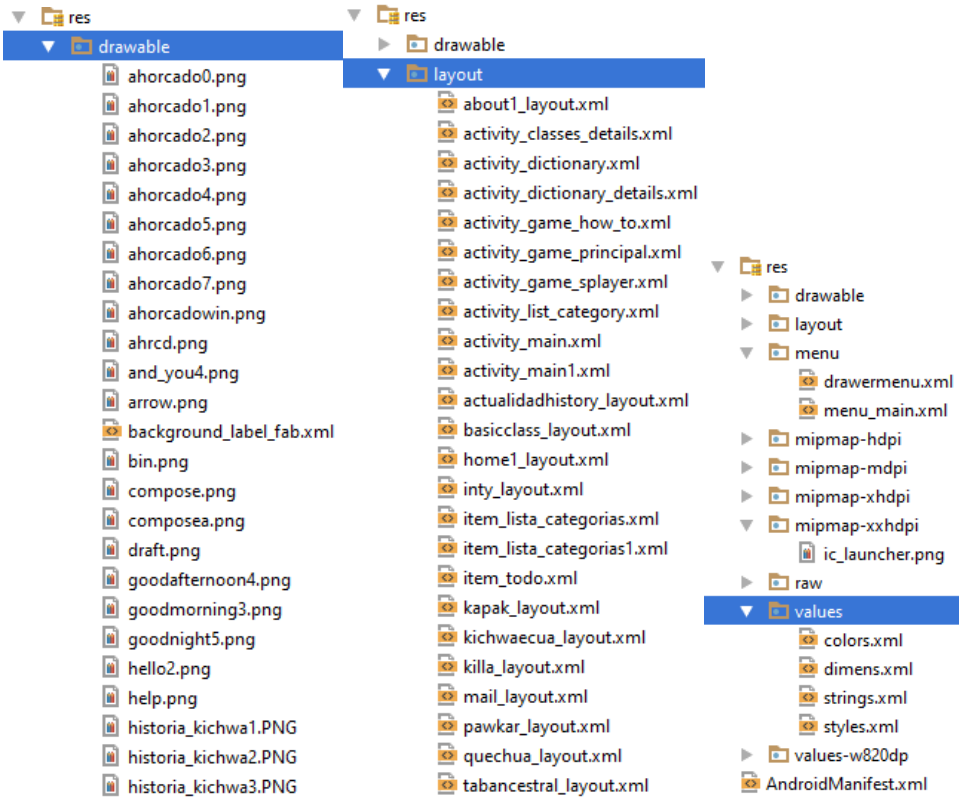


- **View.-** posee la clase java home y principal de todo el proyecto donde se implementa, los floating button action para los diferentes módulos de la app, y demás funcionalidades.



4.3. Vistas

Las pantallas o interfaces gráficas de los diferentes módulos de la app se encuentran dentro de la carpeta de **res/layout**, además dentro de la carpeta **res/values,** encontramos lo concerniente a valores de los colores, fuentes, dimensiones, etc. En la carpeta **res/drawable,** se encuentra las imágenes que se utilizaron en la app.



5. GOOGLE PLAY

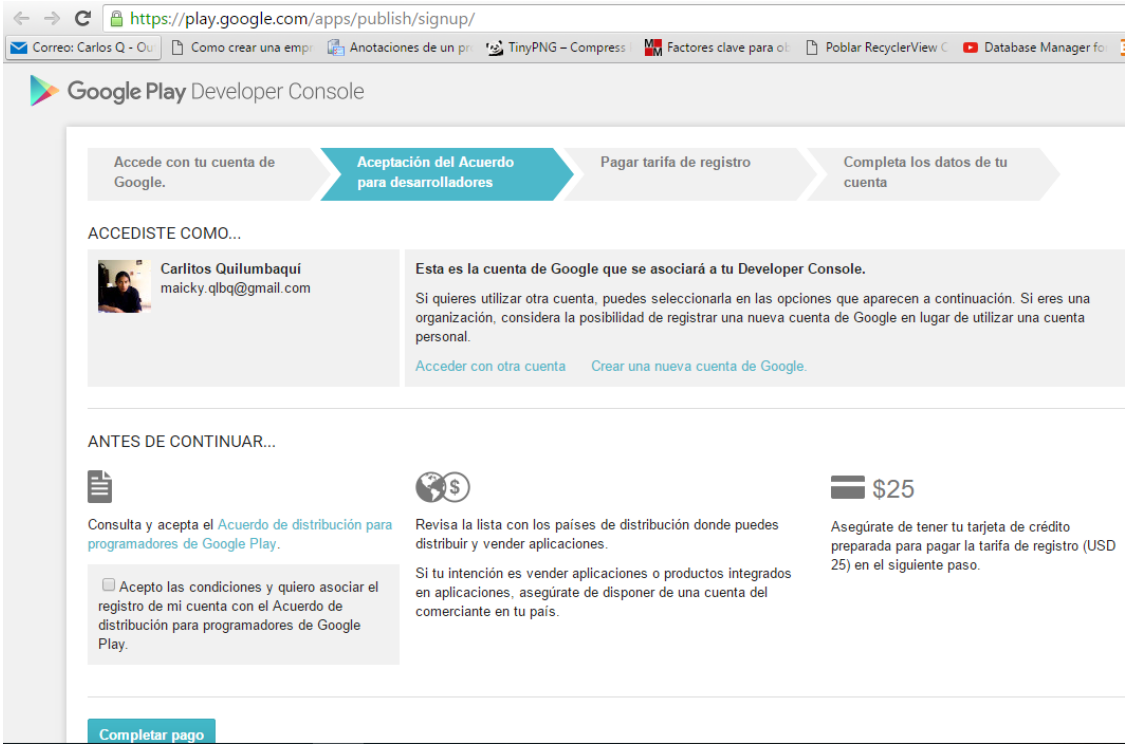
5.1. Introducción

Google Play Store (anteriormente Android Market) es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android, así como una tienda en línea desarrollada y operada por Google. Esta plataforma permite a los usuarios navegar y descargar aplicaciones (desarrolladas mediante Android SDK), juegos, música, libros, revistas y películas. También se pueden adquirir dispositivos móviles como ordenadores Chromebook, teléfonos inteligentes Nexus, Google Chromecast, entre otros.1

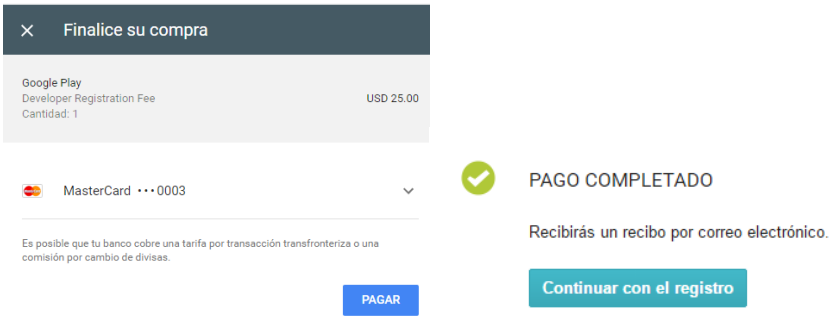
Las aplicaciones se encuentran disponibles de forma gratuita, así como también con costo. Pueden ser descargadas directamente desde un dispositivo con Android a través de la aplicación móvil Play Store.

5.2. Publicación de una app

Lo primero que se debe realizar es convertir la cuenta de Google en una cuenta de desarrolladores. Para ello, se accede a Google Play Developer Console por primera vez en el siguiente link: https://play.google.com/apps/publish/signup/



Click en pago, y llenar el formulario de acuerdo a los requerimientos de facturación y compra online.

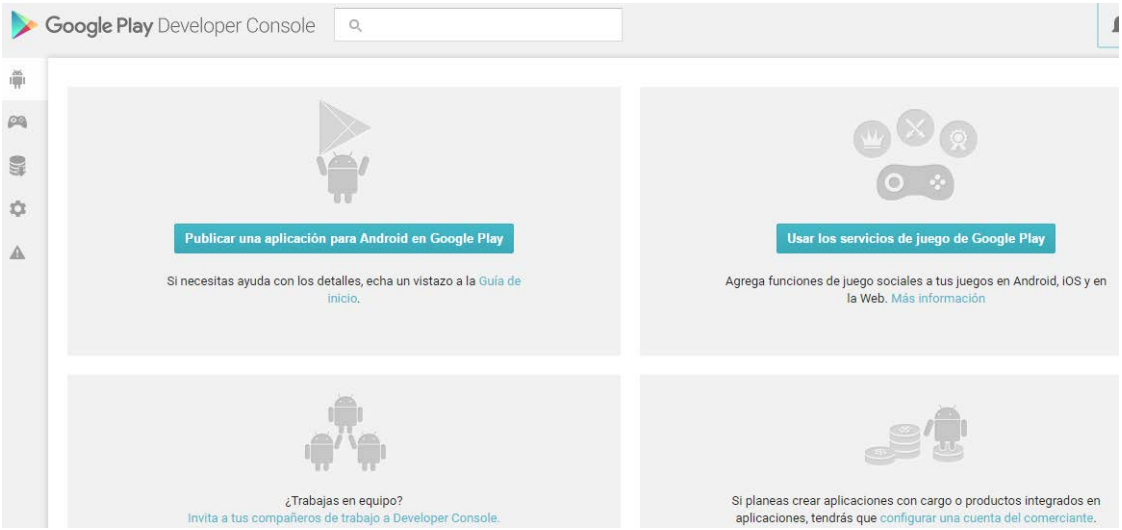


Llenar el formulario con los datos que se mostrará en las aplicaciones a subir a Google Play.

Aquí se debe realizar el pago de $25,00 dólares, por única vez, con lo que ya se podrá ser un desarrollador que distribuye apps en Google Play.

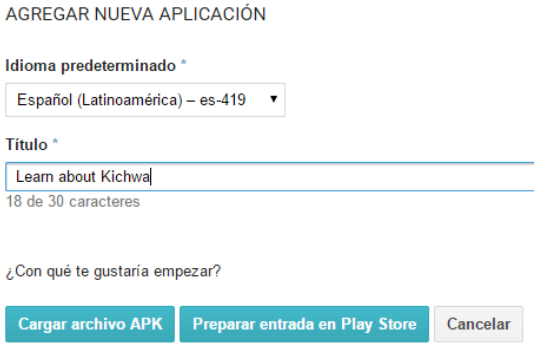
Una vez realizado el pago, podemos entrar en Google Play Developer Console y nos mostrará el centro de gestión e información como desarrolladores, en el cual se puede observar lo siguiente:

- Listado de nuestras aplicaciones.
- Servicios para Google Play Games.
- Informes de nuestros beneficios.
- Configuración
- Anuncios
- Alertas

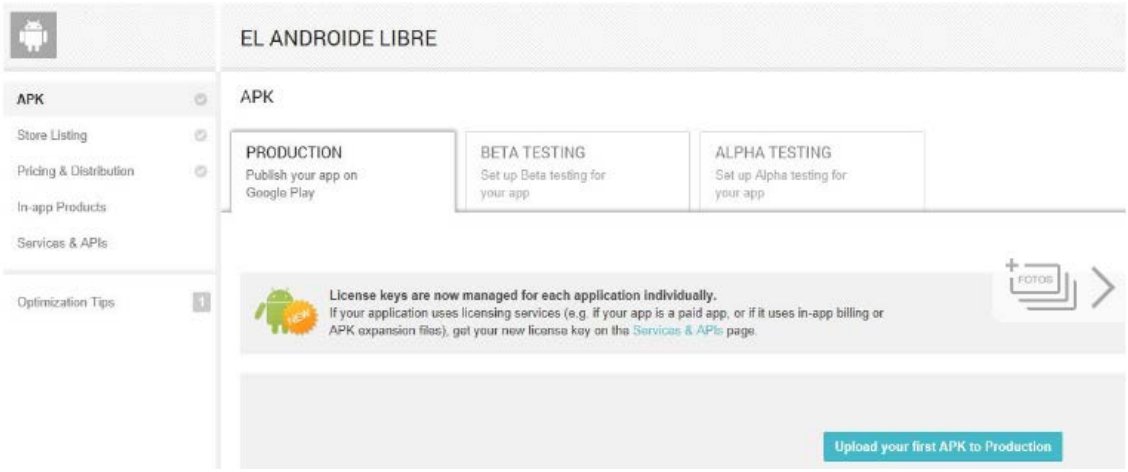


Para agregar una aplicación Android nos vamos al botón “Publicar una aplicación para Android en Google Play”, y luego en agregar nueva aplicación.

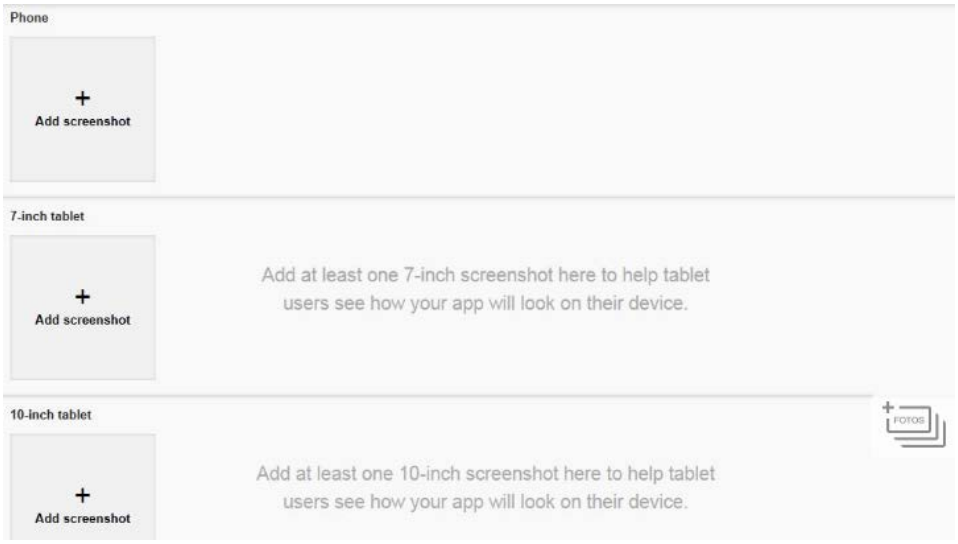
Nos llevará a un cuadro de diálogo donde deberemos escoger el lenguaje por defecto de la aplicación y el título de la misma.



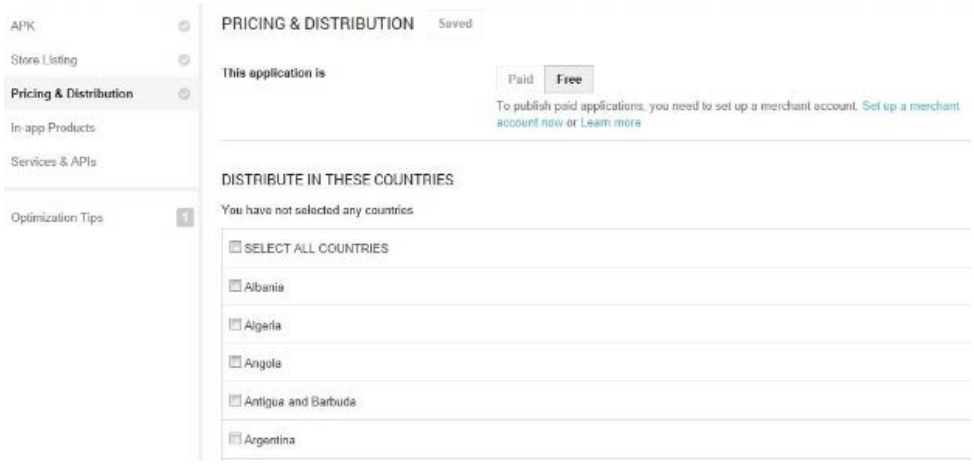
Una vez aquí, se abrirá toda la información acerca de la aplicación. Llenamos todo el formulario, y subimos el APK de la aplicación. Además se puede incluirlo en una fase de testeo alpha o beta, si nos conviene, o directamente a productivo, que lo que aparecerá en Google Play.



Una vez subido, veremos que en APK aparece un **tick verde**, de que esa parte está ya ok para publicar. Ahora pasamos a **Store Listing**, donde rellenaremos la descripción, texto de promoción, pantallazos para diferentes tamaños de pantalla, icono de la aplicación, la categoría de la aplicación, datos de contacto, política de privacidad propia, entre otras cosas.



A continuación, pasamos a la sección **Pricing & Distribution,** donde elegiremos los países donde queremos que la aplicación esté disponible, así como si será gratis o de pago.



Una vez rellenemos esta pestaña, ya estamos listos para publicar la aplicación, cambiando el estado Borrador (*Draft*). También tendremos una pestaña con **consejos para optimizar la información sobre nuestra app en Google Play**.