


Open · Opened 3 weeks ago by  **Rubén Montero**

# Primeros pasos programando para móviles

## Resumen

- Veremos consideraciones especiales de desarrollar código para dispositivos móviles
- Instalaremos y configuraremos Android Studio. Probaremos la aplicación
- Eliminaremos el texto *Hello World!* y comprobaremos de nuevo la aplicación
- Haremos `git add`, `git commit` y `git push` para subir los cambios al repositorio

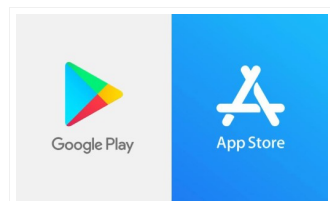
## Descripción

Desarrollar aplicaciones para uso móvil *no* es lo mismo que crear aplicaciones para escritorio (Windows, Linux,...). Es extremadamente importante que:

- Los *recursos* (RAM, ancho de red,...) usados por nuestra aplicación sean los *mínimos*.
- Controlemos el *ciclo de vida* de la aplicación. ¿Qué pasa si llaman por teléfono al usuario mientras nuestra *app* está descargando un vídeo?
- Las aplicaciones sean *responsive*. Deben adaptarse perfectamente a diferentes tamaños de pantalla.
- El *feedback* visual sea inmediato. ¡Nada de tocar la pantalla y que dé la impresión de que la *app* se ha quedado "pillada"!

## Principales plataformas

[Android](#) e [iOS](#) conforman la pareja vencedora en lo que respecta a sistemas operativos móviles [hoy en día](#).



Usaremos Android como vehículo de aprendizaje pues la cuota de mercado es mayor (86% vs 14%), el precio para la publicación de *apps* es menor (pago único de 25\$ frente a 99\$ anuales) y no dependemos de un *hardware* específico (Xcode sólo funciona en MacOS).

## La tarea

Desde la página web de [Android Studio](#), **descarga** el instalador `.exe` haciendo *click* en `Download Android Studio` y **acepta** los términos de licencia<sup>1</sup>.

Una vez descargado, **ejecútalo** y procede con las opciones por defecto, hasta que llegues a la ventana donde permite abrir un proyecto existente o crear un proyecto nuevo.

## Primeros pasos en el repositorio

Necesitas haber instalado [Git](#) y tener tu usuario, email y clave SSH configurados apropiadamente. Si seguiste estos pasos e hiciste `git clone`, ya tienes disponible el repositorio en local. Sólo necesitas abrir un terminal de Windows (*tecla de Windows* > `cmd`), posicionarte en tu repositorio local con `cd` y traer los cambios haciendo *pull*:

```
git pull
```

Puedes verificar desde el explorador de archivos que una carpeta `android-introduction/` ha aparecido.

Luego, desde Android Studio, **selecciona** `Open project` y abre la carpeta `android-introduction/` que se encuentra el repositorio local de nuestro ordenador. Indica **Trust Project** y **Trust Server Certificate**.

## Paciencia 😊

Arrancar un proyecto es lento.

Antes de hacer nada, hay que cerciorarse de que el *banner* amarillo `Gradle Project Sync in Process...` ha desaparecido y abajo no hay barras de progreso indicando tareas en segundo plano.

Aunque tarda hasta 5 ó 10 minutos, terminará estabilizándose y veremos a la izquierda nuestra estructura de proyecto:

```
- app/
  |
  |-- manifests/
  |
  |-- java/
  |   |
  |   |-- com.afundacion.fp.library/
  |   |   |
  |   |   |-- MainActivity.java
  |   |
  |   |-- com.afundacion.fp.library/ (androidTest)
  |   |-- com.afundacion.fp.library/ (test)
  |
  |-- res/

- Gradle Scripts/
```

## Editar preferencias

**Haz click** en *File* (barra superior) > *Settings*.

Se desplegará una ventana con los ajustes de preferencias. En la columna izquierda, busca > **Editor** y **haz click** en la flecha para expandir el menú.

**Selecciona *Design Tools***. En las 4 opciones que aparecen, **elige *Code*** :

- **Default Editor Mode**
  - **Drawables:** *Code*
  - **Other Resources (e.g. Layout, Menu, Navigation):** *Code*
  - **Compose files:** *Code*
  - **Other Kotlin files:** *Code*

De esta manera, preferiremos editar los archivos de interfaz XML a través de código (XML). Existe un editor visual que *nos puede dar muchos dolores de cabeza* si no entendemos *qué XML* estamos manipulando, así que no lo usaremos.

## Lanzar la aplicación

**Haz click** en el botón desplegable *No Devices* en la barra de herramientas superior. Se encuentra a la izquierda del botón verde '**Play**'.

**Selecciona *No Devices* > *Device Manager***. En la pestaña *Virtual* **pulsa** en *Create Device*.

A continuación, crearemos un emulador sencillo. **Elige**:

- **Nexus S** (4.0") de 480x800 hdpi.

En la siguiente ventana del asistente, **selecciona (Download)** la versión:

- **Pie** (API Level 28) para x86

...del sistema operativo que usará la máquina simulada. Acepta el acuerdo de licencia y espera a que se descargue.

---

**Importante:** No descargues una versión de API > 28 o [tendrás problemas en algunos tests](#).

---

En la última página de *Verify configuration* **cambia** *Emulated Performance* > *Graphics* y **elige** *Hardware - GLES 2.0* . Si está disponible, esto debería aprovechar la aceleración *hardware* para que el emulador funcione más deprisa.

Una vez le damos a **Finish**, ¡emulador creado!


**Lanza** la aplicación con el **botón verde 'Play'** en la barra superior. Aparecerá a la derecha de *Nexus S API 28* donde antes aparecía *No Devices*.

¡Y ten paciencia 😊!

Si todo funciona como es debido, aparecerá el emulador a la derecha dentro de Android Studio:



## Posibles problemas

- Si la aplicación no llega a lanzarse, es posible que el dispositivo esté *congelado*. Puedes verificarlo si Android Studio responde a tus clicks (e.g.: Abrir menú *File*) pero el emulador **no** hace **ni caso**. En ese caso:
  - Cerramos el emulador (icono **X** en la pestaña al lado del nombre)
  - Vamos a **Dispositivos** (menú desplegable en la barra de arriba) > *Device Manager* > *Actions* > Flecha  (último icono) > *Wipe Data*.
  - Re-lanzamos** la aplicación con paciencia

Más adelante, veremos alternativas al *emulador de Android Studio*.

## Continuemos...

Trabajemos ahora en la aplicación.

Las *interfaces visuales* de las pantallas se definen en archivos XML.

**Abre** `res > layout > activity_main.xml`. Al hacer doble *click* en ella, verás:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

**Elimina** las líneas 9-16 asociadas al `TextView`.

Si vuelves a lanzar la aplicación verás que el pequeño texto `Hello World!` ha desaparecido.

**¡Enhorabuena!** Primera tarea completada.

## Por último

Desde el terminal de Windows (cmd.exe), nos posicionamos en `android-introduction/` y escribimos:

```
git add *
```


...para marcar los cambios del un nuevo *commit*. Después:

```
git commit -m "Tarea #1: Eliminado TextView hello world en actividad principal"
```

...y por último subimos los cambios al repositorio remoto:

```
git push
```

No está de más visitar la página de GitLab (<https://raspi>) y verificar que el *commit* se ha subido.

1. Estos términos de licencia los impone Google para que usemos su [SDK](#). Básicamente, nos está dando las herramientas necesarias para que nuestras aplicaciones funcionen en dispositivos reales y quiere que, en la medida de lo posible, le exoneremos de responsabilidad ante los daños que nuestras aplicaciones puedan causar 



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 1](#) 3 weeks ago



[Ania Blanco @ania.blanco](#) mentioned in commit [42b57134](#) 2 weeks ago