Laboratorio 0.- Configuración del entorno de desarrollo

Contenido:

Objetivos: Configurar el entorno para desarrollar aplicaciones para la plataforma Android. Realizar una primera aplicación y ejecutarla tanto en el simulador que acompaña al propio entorno como en nuestro dispositivo móvil.

1 Android Studio

- Descargad la versión de Android Studio correspondiente a vuestro sistema operativo: https://developer.android.com/studio

1.1 Instalación en Windows

- En Windows de 64 bits:
  - Ejecutad el fichero .exe descargado.

1.2 Instalación en Linux

- Si vuestro sistema operativo es de 64 bits, instalad las siguientes librerías

```
↳ sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1
                       libbz2-1.0:i386
```

- Instalad las siguientes librerías

```
↳ sudo apt install libcanberra-gtk-module
```

- Descomprimid el fichero descargado
- Acceded a la carpeta *android-studio/bin* y ejecutad el script studio.sh

```
↳ sh studio.sh
```

- Seguid las instrucciones que van apareciendo en pantalla e instalad la versión Standard del entorno.
- Para más información, consultar: https://developer.android.com/studio/install?hl=es-419#linux

⚠ Si durante la instalación os aparece un mensaje avisando que vuestra máquina soporta KVM (en Windows se llama HAXM) seguid estos pasos al terminar la instalación. KVM (o HAXM) es un modo de virtualización que hace que el simulador sea más rápido.

- Comprobad que la CPU realmente admite KVM

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

- Si el valor que devuelve es > 0, se puede continuar. Si no, dejadlo.
- Instalad KVM.

```
sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils
```

- Agregad el usuario de vuestra máquina al grupo KVM y al grupo libvirtd.

```
sudo adduser nombreusuario kvm
sudo adduser nombreusuario libvirtd
```

- Comprobad que la virtualización está activada en la BIOS.

```
sudo kvm-ok
```

Habrá que sustituir "***nombreusuario***" por el nombre de usuario que corresponda en cada máquina

- Si hace falta habilitad la virtualización en la BIOS (depende de cada modelo). En el laboratorio no se puede hacer.
- Reiniciad el equipo.

➢ Cada vez que queramos arrancar Android Studio habrá que ejecutar el script studio.sh

```
sh studio.sh
```

➢ Para facilitar el arranque de Android Studio, cuando esté lanzado (Fig. 1), a través del menú *Configure → Create Desktop Entry* generad una entrada para el escritorio.

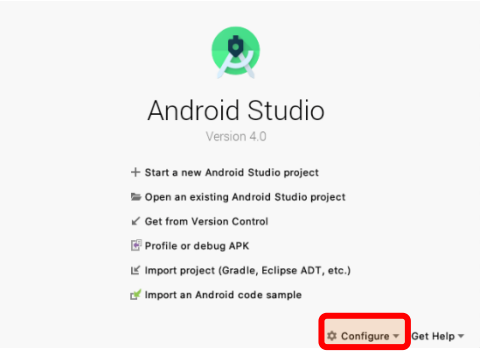

Fig. 1.- Menú principal Android Studio (Opción Configure)

Ahora podréis lanzar Android Studio desde el buscador de aplicaciones e incluso mantenerlo en el lanzador del escritorio.

💡 Cada vez que lancéis Android Studio os avisará si hay actualizaciones pendientes. Es aconsejable tenerlo siempre actualizado.

2 Ejecución de las aplicaciones

- Seleccionad la opción "Start a new Android Studio project" del menú (Fig. 2).

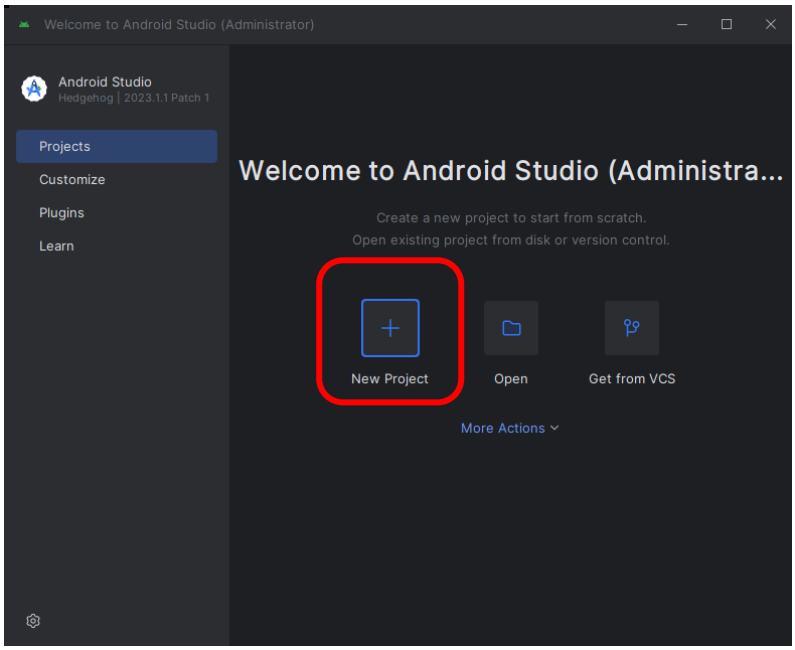

Fig. 2.- Menú principal Android Studio (opción New Project resaltado con un recuadro rojo)

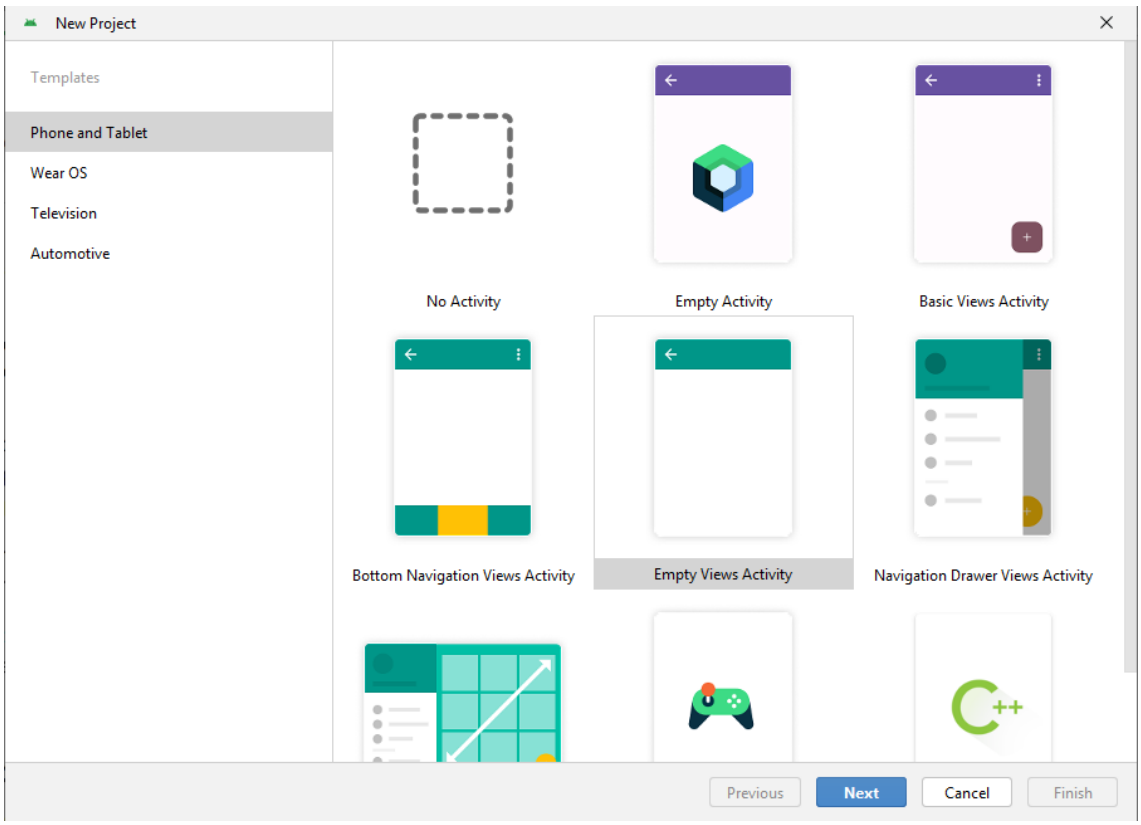

Fig. 3.- Menú New Project (Selecciona Empty Views Activity)

- En las pestañas de la izquierda (Fig. 3) elegid que es una *app* de "Phone and Tablet" (es la opción por defecto).
- Elegid que la *app* tenga una "Empty Views Activity".
- Poned a la aplicación el nombre que queráis. En la opción "Minimum API level" tenéis que indicar la API mínima en la que queréis que funcione vuestra aplicación.
- Selecciona Java como lenguaje y Groovy DSL (build.gradle)

> Si vais a trabajar con un dispositivo real, aseguraos de elegir una versión igual o inferior a la de vuestro dispositivo.

- La primera vez que se crea una nueva aplicación, puede tardar en generar todas las dependencias. Lo mejor es dejarle trabajar y no tocar nada hasta que haya terminado. Sabremos que ha terminado porque lo indica en la parte inferior de la pantalla (Fig. 4) y se activa la opción de ejecutar la aplicación (Fig. 5).

Fig. 4.- Mensaje que indica que el proceso no ha terminado

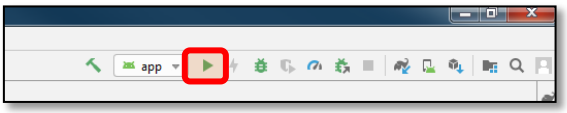

Fig. 5.- Opción de ejecución activada

2.1 Ejecución en un dispositivo real

Las ***apps*** que desarrollemos se pueden probar en dispositivos Android reales.

- Acceded al menú *Ajustes → Información del teléfono* y pulsad 7 veces sobre el "Número de compilación". De este modo se activarán las Opciones de desarrollo del teléfono.
- A través del menú *Ajustes → Opciones de desarrollo* marcad la opción "Depuración USB" que nos permitirá ejecutar en nuestro dispositivo las aplicaciones que desarrollemos lanzándolas directamente desde Android Studio.
- Conectad el dispositivo al ordenador mediante el cable USB. Aceptad el mensaje sobre la clave de la máquina que aparecerá en vuestro dispositivo. La clave es propia de cada máquina, por lo que tendréis que aceptar la clave de todas las máquinas en las que conectéis el dispositivo.

> 💡 Si marcáis la opción de que acepte la clave para siempre, no tendréis que aceptarla cada vez que conectéis el dispositivo a esa máquina concreta.

- Al lanzar la aplicación (icono de Play en la Fig. 5) vuestro dispositivo debería aparecer entre las opciones disponibles para ejecutar la aplicación (Fig. 6). Lo seleccionáis, dais a "OK" y la aplicación se ejecutará en vuestro dispositivo.

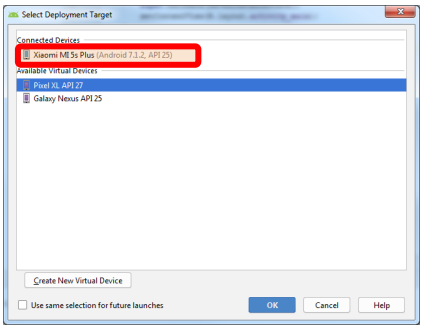

Fig. 6.- Selección del dispositivo para la ejecución

⚠ Si no aparece en vuestro dispositivo un mensaje para aceptar la clave correspondiente a la máquina, es porque no se está reconociendo correctamente.

- Para solucionarlo en Windows hay que buscar e instalar los drivers correspondientes.
- Para solucionarlo en Ubuntu tendremos que añadir el dispositivo manualmente (sólo la primera vez).

  - Con el dispositivo conectado mediante el cable USB ejecutad:

    ```
    ↳ lsusb
    ```

    Se os mostrará un listado de todos los puertos USB de la máquina y los dispositivos que están conectados. Cada línea será similar a esta.

    ```
    Bus XXX Device YYY: ID idFabricante:idProducto Descripción
    ```

  - Buscad la línea correspondiente a vuestro dispositivo y apuntad los 4 caracteres del idFabricante y los 4 del idProducto. Si no sabéis qué línea se corresponde con vuestro dispositivo, desenchufadlo, volved a ejecutar el comando y mirad a ver qué línea falta.
  - Ejecutad el siguiente comando

    ```
    ↳ sudo gedit /etc/udev/rules.d/51.android.rules
    ```

  - Añadid al fichero la siguiente línea:

    ```
    ↳ SUBSYSTEM==”usb”, ATTRS{idVendor}==”idFabricante”,
        ATTRS{idProduct}==”idProducto", MODE=”0666"
    ```

    Sustituid idFabricante e idProducto por los códigos anotados en el paso anterior

  - Guardad el fichero y ejecuta los siguientes comandos:

    ```
    ↳ sudo chmod a+rx /etc/udev/rules.d/51-android.rules
    ↳ sudo service udev restart
    ↳ cd ~/.android
    ↳ sudo gedit adb_usb.ini
    ```

  - Añadid al fichero una línea con el texto

    Sustituid idFabricante por su código

    ```
    ↳ 0xidFabricante
    ```

  - Guardad el fichero y volved a reiniciar el servicio udev

    ```
    ↳ sudo service udev restart
    ```

  - Desenchufad el dispositivo y terminad el servidor adb

    ```
    ↳ adb kill-server
    ```

  - Enchufad el dispositivo y ejecutad el comando devices que pone en marcha el servidor y lista los dispositivos USB con los que adb es capaz de trabajar

    ```
    ↳ adb devices
    ```

  - Vuestro dispositivo debería aparecer correctamente con un código asignado.

2.2 Ejecución en un simulador

➢ Abrid el AVD (***Android Virtual Device***) Manager pinchando en el icono correspondiente (Fig. 7).

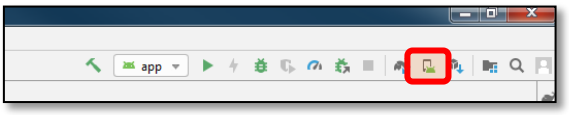

Fig. 7.- Android Virtual Device Manager

En la ventana que se abre (Fig. 6) se mostrará el listado de todos los simuladores que tengamos definidos para probar nuestras aplicaciones. Se pueden tener tantos simuladores como se deseen y cada uno de ellos con las características que se quiera: tamaño de la pantalla, de la RAM, tipo de sensores disponibles, etc.

También permite definir nuevos modelos en función de las necesidades del desarrollador. Para el trabajo en la asignatura, la mayoría de las veces no nos importarán las características concretas del simulador.

➢ Pinchad "***Create Virtual Device***". Se mostrará un listado con una serie de modelos predefinidos. Se pueden crear emuladores en base a esas definiciones o crear un modelo totalmente nuevo mediante la opción "***New Hardware Profile***".
Una vez elegido el modelo pulsad "***Next***" y tendréis que elegir qué versión del sistema operativo deseáis que tenga vuestro emulador.

💡 Si trabajáis en una arquitectura de 64 bits, cuando sea posible, elegid arquitecturas x86_64, que harán que el emulador se ejecute de forma más rápida. Estas arquitecturas se encuentran en la pestaña "x86 Images".

En la columna "Target" se indica si es una imagen con Play Store instalado (Google Play), si sólo tiene instaladas las APIs de Google (Google APIs) – necesarias para acceder a servicios de Google como Google Maps – o si es una versión "limpia". La imagen más completa es la de Google Play, pero hasta que trabajemos con los servicios de Google nos valdría cualquiera de las versiones.

➢ Al lanzar la aplicación (icono de Play en la Fig. 5) vuestro simulador aparecerá entre las opciones disponibles para ejecutar la aplicación. Lo seleccionáis, dais a "OK" y la aplicación se ejecutará en vuestro dispositivo.

💡 El simulador suele tardar bastante en arrancar, por lo que es aconsejable tenerlo arrancado y no cerrarlo cada vez, para no tener que esperar cada vez que se desea probar una aplicación.

⚠ Si estáis usando una arquitectura de 32 bits, os dará un error, ya que los simuladores están optimizados para arquitecturas de 64 bits. Para poder usar una arquitectura de 32 bits, seguid los pasos siguientes:

```
↳ gedit .profile
```

- Añadid una línea al final con el siguiente texto:

```
↳ export ANDROID_EMULATOR_FORCE_32BIT=true
```

- Guardad el fichero y ejecutad: `source .profile`
- Lanzad el emulador de nuevo. Si vuelve a dar error, reiniciad el sistema.