Licenciatura en Ciencias de la Computación, Facultad de Ciencias, UNAM.
Programación de dispositivos móviles.
Profesora: Ana Libia Eslava Cervantes.
Ayudante: Manuel Ignacio Castillo López.

Notas de clase
Android Studio

Introducción

Android Studio es el IDE oficial para desarrollar en Android. Aunque no siempre ha sido así. Cuando se lanzó Android, no había un IDE dedicado para desarrollar apps en Android, se usaba un plug-in de Eclipse para desarrollar Android. Eclipse fue el IDE oficial para desarrollar Android. Google lanzó Android Studio en el 2014, pero Eclipse continuó siendo la herramienta favorita por al menos un año más. Actualmente el plug-in de Eclipse se considera obsoleto e inadecuado para trabajar con Android.

Ejercicio. Instalando Android Studio. Antes de empezar se recomienda tener instalado Oracle Java SE en su versión más reciente (excepto Mac, deberá usar la versión 1.7). Verifique que la variable de entorno JAVA_HOME está dada de alta en su sistema y que está agregada a la variable PATH.

Visite el siguiente enlace y descargue la versión más adecuada para su sistema de Android Studio: https://developer.android.com/studio/

- Para instalar en Windows: basta con ejecutar el instalador descargado y seguir los pasos del asistente automático.
- Para instalar en Mac: abrimos la imagen DMG descargada y arrastramos Android Studio a la carpeta Applications del sistema. Deberá iniciarse un asistente para iniciar Android Studio, al seguir los pasos deberíamos completar la instalación.
- Para instalar en GNU/Linux: extraemos el archivo comprimido descargado en la ubicación en la que queremos instalar Android Studio. Sugerimos usar **/opt** (para hacerlo disponible para todos los usuarios) o en su directorio home. Para iniciar Android Studio ejecutamos el archivo **android-studio/bin/studio.sh**

Conociendo Android Studio

Al iniciar Android Studio, lo primero que debemos hacer es crear un proyecto. Cuando termina de cargar por primera vez, Android Studio nos muestra una pequeña ventana que nos da la opción para crear un proyecto. Al seleccionarla aparece un asistente que nos permite personalizar algunas características del proyecto que es estamos iniciando:

1. Primero elegimos el tipo de dispositivos para el que vamos a desarrollar (o al menos el tipo primario). Podemos elegir entre:
    a. Smartphones y tablets.
    b. Wearables.

c. Smart TV.
d. Automóviles.
e. Sistemas y dispositivos embebidos.

Para los fines de este curso, siempre vamos a usar la opción por defecto: smartphones y tablets.

2. Independientemente del tipo primario de dispositivos que elijamos, lo siguiente es indicar si queremos que se incluya una actividad y si queremos que la actividad tenga alguna funcionalidad especial. En el menú de smartphones y tablets encontramos las siguientes:
    a. **Basic Activity** - Pantalla con flecha de navegación, menú contextual y botón flotante de acción (o FAB por sus siglas en inglés *Floating Action Button*).
    b. **Empty Activity** - Sólo tiene flecha de navegación. Esta es la que vamos a elegir siempre durante el curso.
    c. **Bottom Navigation Activity** - Además de flecha de navegación y menú contextual, incluye un menú de navegación en la parte inferior. Este tipo de menús de navegación no debe confundirse con pestañas conforme a las normas de Material Design. Más adelante aprenderemos la diferencia entre una barra de navegación inferior y pestañas (que se deben ubicar en la parte superior).
    d. **Fullscreen Activity** - Contiene un botón que le permite al usuario ver la actividad en pantalla completa (ocultando la barra de notificaciones y los botones de navegación del sistema).
    e. **Master/Detail Flow** - Este es un tipo de actividad que se usa principalmente en vistas *fragmentables* y con tablets. Son bastante útiles y requieren un diseño un tanto específico que estudiaremos con detalle más adelante.
    f. **Navigation Drawer Activity** - Contiene un menú inicialmente oculto que se revela corriendo desde el extremo izquierdo de la pantalla hacia la derecha.
    g. **Google Maps Activity** - Hospeda una vista especial para mostrar mapas. Los mapas pueden tener todas (o algunas) de las funcionalidades que ofrece el API de Google Maps (cómo colocar marcadores, mostrar una vista satelital o medir distancias).
    h. **Login Activity** - Redefine un formulario para introducir un nombre de usuario y contraseña para iniciar sesión.
    i. **Scrolling Activity** - El área principal de la actividad puede contener elementos cuya posición está fuera del rango visible de la pantalla hacia lo largo, y el usuario puede deslizar la pantalla para revelar dichos elementos. Además, la barra superior y el FAB permanecen visibles siempre (aunque disminuyen el espacio que ocupan en pantalla cuando el usuario desliza la pantalla hacia abajo y restaura su tamaño cuando vuelve a subir).
    j. **Tabbed Activity** - El contenido de esta actividad se página hacia lo ancho.
    k. **Native C++** - Crea una *NativeActivity* para desarrollar una aplicación (o al menos un módulo de la aplicación) usando el Android NDK. Como siempre, solo es necesario usarla cuando las especificaciones de la aplicación a desarrollar así lo requieran (el contenido requerirá de una alta demanda de

procesamiento o queremos un procesamiento dedicado de gráficos más allá de OpenGL que se puede usar sin problemas en una actividad estándar).

Es importante tomar en cuenta que estas actividades pueden usarse para más propósitos que para los que sugiere su nombre y podemos combinar funcionalidades; siempre y cuando hagan sentido bajo los patrones de diseño de móviles y Material Design.

Elegimos; como haremos siempre en este curso, usar una Empty Activity.

3. Esta es la última pantalla del asistente para crear un proyecto. Aquí damos de alta la información más importante:
    a. El nombre de la aplicación (y del proyecto).
    b. El nombre del paquete de mayor jerarquía que contendrá a las clases que componen la aplicación. Este campo es muy importante, ya que siguiendo el estándar de Java, dicho paquete será el identificador de nuestra aplicación. Debemos seguir el estándar para nombrar paquetes de Java.
    c. La ubicación del proyecto en el equipo de desarrollo.
    d. El lenguaje en el que queremos que se creen las clases para la Actividad primaria que solicitamos (Java o Kotlin).
    e. El API mínimo que va a soportar nuestra aplicación. Para fines de este curso se permitirá variar entre API nivel 14 (4.0 Ice Cream Sandwich) y nivel 19 (4.4 KitKat).
    f. Si queremos que soporte Android Instant. No usaremos Android Instante en el curso.
    g. Si queremos que soporte AndroidX. Tampoco haremos uso de AndroidX.

Con esto finaliza el asistente y se abrirá la pantalla principal de Android Studio. Los últimos tres puntos son nuevos hasta donde hemos estudiado, así que vamos a desarrollar brevemente de qué se tratan:

1. **Nivel de API**. Cada versión de Android y sus actualizaciones, definen un nivel de API. Por ejemplo, la primer versión de Android es API nivel 1 y Android Gingerbread en su segunda actualización es API nivel 10. Usualmente las versiones de Android tienen nombre diferente cuando se trata de cambios significativos en el funcionamiento del sistema o las integraciones y servicios en la plataforma.

    Entre más bajo sea el nivel de API que solicitamos, generalmente implica un mayor esfuerzo para escribir instrucciones para las versiones más recientes y las más antiguas. Dependiendo de las funcionalidades que deseemos para nuestra aplicación, es posible que nos baste con solo usar las instrucciones de soporte (que se ejecutan de forma universal en todas las versiones). Indicar que desarrollaremos para un API más bajo también aumenta el número de dispositivos con el que nuestra aplicación es compatible, por lo que puede aumentar el número de sus usuarios. También es útil conocer qué tipo de dispositivos usan nuestros usuarios objetivos, para descartar las APIs anteriores para las que no esperamos un número significativo de usuarios por condiciones socioeconómicas.

2. **Android Instant**. Permite a los usuarios descargar la aplicación y ejecutarla sin instalarla, de forma que pueden usarla a manera de vista previa e instalarla si realmente se sienten satisfecha con ella. Esto es útil sobre todo para aplicaciones que desempeñan alguna tarea para la que ya existe una gran competencia con un número importante de aplicaciones, ya que los usuarios suelen probar las aplicaciones y se sentirán más atraídos a probar una aplicación instantánea; lo que también aumenta las probabilidades de que termine instalada en sus dispositivos (si le encuentran más utilidad y una mejor experiencia que con otras que pueda probar).

3. **AndroidX**. Es el reemplazo a las bibliotecas de soporte introducidas con Android Pie (API nivel 28). Permite desarrollar Aplicaciones usando una extensión del SDK, lo que puede facilitar el desarrollo al tener acceso a más funciones pre-definidas y usar instrucciones de las versiones anteriores de Android (del API nivel 27 hacia atrás).

Como muchos otros IDEs recientes, Android Studio es bastante personalizable y tiene la capacidad de agregar plug-ins. Las áreas de trabajo visibles en pantalla pueden cambiarse de lugar o podemos cambiar su tamaño para ajustarlas a nuestro gusto. La mayor parte del tiempo vamos a poner nuestra atención sobre el área de edición, por lo que es recomendado que sea el área más visible en el editor. Ahora que tenemos la pantalla principal de Android Studio, vamos a describir algunas de las herramientas más utilizadas y sus atajos de teclado:

- **Crear una clase**.
  Sin atajo de teclado.

  Podemos crear una clase de dos formas:
  1. Usando la barra de menús; vamos a **File > New > Java Class**
  2. Usando la vista del proyecto (Alt +1), vamos a **app > java** y elegimos el paquete en el que queremos crear una nueva clase. También podemos elegir un paquete de jerarquía superior si el paquete no existe; o un paquete completamente diferente o la carpeta java o app directamente, podemos especificar el paquete más adelante.
     En todo caso, damos click secundario y del menú contextual elegimos **new > Java Class**.

  Aparecerá un cuadro de diálogo en el que podemos completar la información de la clase que vamos a crear como:
  - Nombre de la clase.
  - Tipo (Clase concreta, clase abstracta, interfaz; entre otros).
  - Superclase. Este campo se autocompleta automáticamente. Note que si la superclase tiene un tipo de datos parametrizado (como es el caso de *java.util.AbstractList*), no podrá indicarlo aquí; pero puede editarlo cuando Android Studio cree la nueva clase.
  - Interface(s) a implementar. Podemos indicar tantas como sean necesarias, separadas por coma. Es mejor escribir el nombre completo del paquete para

evitar ambigüedad al importar. Note que si la interfaz tiene un tipo de datos parametrizado (como es el caso de *java.lang.Comparable*), como en el caso de la superclase, tendrá que esperar a crear la clase para indicarlo.

- **Crear un recurso o un directorio de recursos**.
  Sin atajo del teclado.

  Podemos crear un recurso de dos formas:
  1. Usando la barra de menús; vamos a **File > New > Android Resource File** o **File > New > Android Resource Directory**; según sea el caso.
  2. Usando la vista del proyecto (Alt +1), vamos a **app > res** y elegimos el directorio del tipo y clasificación de recurso que queremos crear. También podemos elegir un directorio de jerarquía superior si el tipo de recurso o clasificación no existe; también podemos elegir un directorio completamente diferente o la carpeta res o app directamente, podemos especificar el tipo y clasificación de recurso más adelante.
     En todo caso, damos click secundario y del menú contextual elegimos **new > Android Resource File** o **new > Android Resource Directory**; según sea el caso.

  Aparecerá un cuadro de diálogo en el que podemos completar la información del archivo de recursos o directorio de recursos que queremos crear:
  - Nombre del archivo o directorio.
  - Tipo de recursos a contenter.
  - Elemento raíz (solo archivo de recursos). No es editable puesto que ya indicamos que queremos crear recursos.
  - Conjunto fuente. Indicamos la variante del proyecto en el que lo queremos.
  - Nombre del directorio padre (solo archivo de recursos).
  - Clasificaciones.

- **Buscar en el archivo actual**.
  Ctrl +F
  En la barra de menús, vamos a **Edit > Find > Find**.

- **Buscar en todo el proyecto**.
  Doble shift -o- Ctrl + Shit + F
  En la barra de menús, vamos a **Edit >Find > Find in path...**

- **Reemplazar en el archivo actual**.
  Ctrl + R
  En la barra de menús, vamos a **Edit > Find > Replace**.

- **Reemplazar en todo el proyecto**.
  Ctrl +Shift +R
  En la barra de menús, vamos a **Edit > Find > Replace in path...**

- **Renombrar una variable, método, clase o un paquete**.
  Primero se debe posicionar el cursor sobre el nombre del elemento que queremos renombrar.

  Shift +F6

  Podemos renombrar de dos formas:
  1. En la barra de menús, vamos a **Refactor > Rename…**
  2. Damos click secundario y del menú contextual elegimos **Refactor > Rename...**

- **Mover una variable (estática), método (estático), clase o paquete**.
  Primero se debe posicionar el cursor sobre el nombre del elemento que queremos mover.

  F6

  Podemos mover de dos formas:
  1. En la barra de menús, vamos a **Refactor > Move…**
  2. Damos click secundario y del menú contextual elegimos **Refactor > Move...**

- **Completar código**.
  Primero se debe posicionar el cursor sobre el texto que queremos autocompletar.

  Ctrl +Barra espaciadora.

  En la barra de menús, vamos a **Code > Completion > Basic**.

- **Organizar imports**.
  Ctrl + Alt +O

  En la barra de menús, vamos a **Code > Optimize Imports**.

- **Borrar una línea**.
  Primero se debe posicionar el cursor en la línea de texto que queremos borrar.

  Ctrl +Y

- **Mover una instrucción o bloque de código hacia arriba o abajo**.
  Primero se debe posicionar el cursor en la instrucción o inicio de bloque de código que se quiere mover.

  Ctrl +Shift +Flecha arriba
  Ctrl +Shift +Flecha abajo

En la barra de menús, vamos a **Code > Move Statement Down** o **Code > Move Statement Up**.

- **Comentar texto existente**.
  Primero se debe posicionar el cursor en la línea que se quiere comentar, o seleccionar el texto de las líneas a comentar.

  Ctrl +/
  Nota: Use la diagonal en el teclado numérico; ya que la tecla “7” que incluye la diagonal con shift no funcionará para este atajo. Si su teclado está en inglés o incluye una tecla con diagonal como carácter primario, debería funcionar sin mayor problema.

  En la barra de menús, vamos a **Code > Comment with Line Comment**.

- **Buscar usos de una variable o un método**.
  Primero se debe posicionar el cursor en la variable o método cuyos usos queremos conocer.

  Alt +F7

  En la barra de menús vamos a **Edit > Find > Find Usages**.

- **Compilar proyecto**.
  Ctrl +F9

  Podemos compilar el proyecto de dos formas:
  1. En la barra de menús vamos a **Build > Make Project**.
  2. En la barra de herramientas, damos clic en el botón Make Project.

- **Exportar APK de depuración**.
  Sin atajo de teclado.

  Podemos exportar un APK de dos formas:
  1. En la barra de menús vamos a **Build > Build Bundle / APK(s) > Build APK(s)**.
  2. Usando la vista de Gradle, vamos a **<nombre app> > :app > Tasks > other > assembleDebug**.
     a. Si usamos este método, la siguiente vez que queramos exportar la aplicación basta con ir a la barra de herramientas y desplegar la lista de Configuraciones de ejecución/depuración y seleccionar assembleDebug directamente.

- **Ejecutar y depurar**.
  Shift +F10 para ejecutar
  Shift +F9 para depurar

En la barra de herramientas, seleccionamos app en la lista desplegable de Configuraciones de ejecución/depuración.

1. Si queremos ejecutar, podemos usar Shift +10 o dar clic en el botón ejecutar.
2. Si queremos depurar, podemos usar Shift +F9 o dar clic en el botón depurar.

- **Dar de alta un repositorio local de manejo de versiones**.
  El atajo de teclado es Alt +´; pero puede no funcionar en teclados no ingleses.

  En la barra de menús vamos a **VCS > VCS operations popup...**

- **Cambiar de rama (solo GIT)**.
  Sin atajo de teclado.

  En la barra de menús vamos a **VCS > Git > Branches...**

- **Hacer commit sobre los cambios y empujar al servidor**.
  Ctrl +K para hacer commit y empujar cambios.
  Ctrl +Shift +K para hacer empujar cambios.

  Podemos hacer un commit de dos formas:

  1. En la barra de menús vamos a **VCS > Commit…**
  2. En la barra de herramientas, damos clic en el botón Commit.

- **Obtener cambios del servidor de manejo de versiones**.
  Ctrl +T

  Podemos obtener cambios de dos formas:

  1. En la barra de menús vamos a **CVS > Update project…**
  2. En la barra de herramientas, damos clic en el botón Update Prject.

  Si estamos usando Git, esto no hará Pull, para hacer pull vamos a **CVS > Git > Pull**

Android SDK Manager

El Android SDK Manager es una de las cuatro principales herramientas para desarrollar Android (junto con Android Studio, ADB y AVD Manager). Android Studio es la herramienta principal de desarrollo, el SDK Manager, ADB y AVD Manager se pueden ejecutar desde Android Studio, pero también se pueden ejecutar de forma independiente desde consola.

El SDK Manager; como su nombre sugiere, nos permite instalar versiones de Android con las cuales compilar nuestros proyectos y herramientas adicionales, como las bibliotecas de soporte, bibliotecas de monetización de Google e indicar el origen de los repositorios de Android.

Para iniciar el SDK Manager desde consola, debemos ubicar el directorio donde instalamos el Android SDK al instalar Android Studio. Luego, ejecutamos el binario **Android/Sdk/tools/bin/sdkmanager** o desde la barra de herramientas de Android Studio.

Al iniciar el SDK Manager, aparece un diálogo que tiene tres pestañas. Cada pestaña nos permite instalar o actualizar las distintas herramientas que necesitamos para compilar un proyecto Android en sus diferentes versiones, con las distintas dependencias que nos ofrece para incluir anuncios, compras desde la aplicación, soporte para versiones anteriores de Android, entre otras.

El contenido de las pestañas es el siguiente:

1. SDK Platforms - Lista las versiones de Android para las que podemos desarrollar. Al menos debemos tener instalada la versión más reciente, que es la que siempre debemos usar al desarrollar.
2. SDK Tools - Son bibliotecas adicionales que nos permiten monetizar la aplicación, dar soporte a versiones anteriores de Android, etc.
3. SDK Update Sites - Lista de repositorios de donde queremos que se busquen los paquetes de Android. Usualmente no necesitamos modificarlo, a menos que queramos usar versiones alternativas (no recomendado en la mayoría de los casos) o Google cambie la URL de los repositorios de Google (que es muy poco probable).

   Tarea. Android SDK. Deberá instalar o actualizar los siguientes paquetes:
   1. Android SDK Platform (al momento de redactar, la versión más reciente es Android 9.0 Pie, nivel de API 28, 6ta. revisión).
   2. Android SDK Build-Tools.
   3. Android Emulator.
   4. Android SDK Platform-Tools.
   5. Android SDK Tools.
   6. Google USB Driver (solo Windows).
   7. Todos los paquetes del repositorio de soporte (Lista desplegable al final de la pestaña).

   Este proceso puede ocupar varios GB en disco y puede tardar varias horas y es un requisito para progresar en las actividades del curso.