Universidad Nacional Politécnica
Departamento de Ciencia

Ingeniería en Computación
Programación móvil

Nombre: Victor Manuel Leiva

Docente: Ing. Luis

Universidad Nacional Politécnica
Departamento de Ciencia

Descripción de Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para crear aplicaciones móviles en el sistema operativo Android. En este entorno, los proyectos son la base de todo el trabajo, encapsulando todos los elementos necesarios para construir una aplicación, desde el código fuente hasta los recursos y las configuraciones de compilación.

Los proyectos se organizan en módulos, que actúan como unidades de funcionalidad discretas. Estos módulos permiten dividir la aplicación en partes manejables, facilitando la compilación, prueba y depuración independientes de cada uno. Puedes tener varios módulos en un proyecto, cada uno con sus propios archivos y configuraciones.

Android Studio ofrece varios tipos de módulos para satisfacer las necesidades específicas de desarrollo. Por ejemplo, el módulo de "app" para Android es el contenedor principal para el código fuente de tu aplicación, recursos y configuraciones de nivel de app. Además, hay módulos de funciones que representan características modulares de la aplicación y módulos de biblioteca para el código reutilizable.

Dentro de cada módulo, la estructura de archivos se organiza en grupos como "manifest", "java", "res", entre otros, que contienen diferentes tipos de archivos relacionados con la aplicación. Estos grupos facilitan la navegación y la gestión de los recursos de la aplicación.

Para ver la estructura de archivos real del proyecto, incluidos los archivos ocultos, puedes cambiar a la vista de proyecto en Android Studio. Esta vista te permite acceder a todos los archivos y directorios, incluidos los resultados de compilación y bibliotecas privadas.

Además, Android Studio proporciona herramientas para configurar la estructura del proyecto, como el diálogo de estructura del proyecto, que te permite establecer opciones de configuración específicas del proyecto y del módulo, como la versión de SDK, la firma de la aplicación y las dependencias de la biblioteca.

Comandos comunes en Android Studio incluyen:

- Ctrl + Shift + N: Para abrir rápidamente cualquier archivo por nombre.
- Ctrl + B: Para ir a la declaración de una clase, método o variable.
- Shift + Shift: Para buscar en todo el proyecto.
- Ctrl + Alt + L: Para reformatear el código.
- Ctrl + D: Para duplicar una línea de código.
- Ctrl + Space: Para activar la ayuda de autocompletado.
- Ctrl + Alt + O: Para optimizar importaciones.
- Ctrl + Shift + F10: Para ejecutar la app.

Crear proyecto

Android Studio facilita enormemente el proceso de iniciar un nuevo proyecto para desarrollar aplicaciones Android en una amplia gama de dispositivos, como teléfonos, tablets, televisores y dispositivos Wear. Aquí tienes una guía paso a paso sobre cómo comenzar un nuevo proyecto o importar uno existente:

1. Comenzar un Nuevo Proyecto:

- Si no tienes ningún proyecto abierto, simplemente haz clic en "Empezar un nuevo proyecto de Android Studio" desde la pantalla de bienvenida.
- Si ya tienes un proyecto abierto, ve al menú principal y selecciona "Archivo > Nuevo > Nuevo Proyecto".

2. Seleccionar Tipo de Proyecto:

- En la pantalla de creación de nuevo proyecto, encontrarás diversas categorías de dispositivos en el panel de Plantillas.
- Elige el tipo de proyecto que deseas crear, como teléfono, tablet, etc.

3. Configurar Proyecto:

- Una vez seleccionado el tipo de proyecto, haz clic en "Siguiente".
- En la siguiente pantalla, configurarás opciones importantes como:
- Nombre del proyecto: Asigna un nombre descriptivo a tu proyecto.
- Nombre del paquete: Define un nombre único para el paquete de tu aplicación.
- Ubicación de Guardado: Decide dónde almacenar localmente tu proyecto en tu sistema.
- Lenguaje: Selecciona si prefieres usar Kotlin o Java para escribir el código de tu proyecto.
- Nivel de API Mínimo: Elige el nivel de API mínimo requerido por tu aplicación.
- Puedes obtener ayuda para seleccionar el nivel de API haciendo clic en "Ayúdame a elegir".
- Por defecto, se utilizarán las bibliotecas AndroidX, que son recomendadas para nuevos proyectos.

4. Finalizar la Configuración:

- Una vez que hayas configurado todas las opciones según tus preferencias, haz clic en "Finalizar".
- Al seguir estos pasos, habrás creado y configurado tu proyecto en Android Studio, listo para comenzar a desarrollar tu aplicación para Android.
- Para realizar estas acciones utilizando comandos en Android Studio, puedes hacer lo siguiente:
- Para comenzar un nuevo proyecto desde cero, puedes usar el comando `File > New > New Project` en el menú principal.
- Para configurar opciones específicas del proyecto, como el nombre y el lenguaje, simplemente sigue las instrucciones en la interfaz de usuario durante la creación del proyecto.
- Para finalizar la configuración y crear el proyecto, simplemente haz clic en "Finalizar" en la última pantalla de configuración.

Como migrar a Android Studio

Cuando migras tus proyectos desde IntelliJ a Android Studio, debes tener en cuenta varias consideraciones, como la estructura del proyecto, el sistema de compilación basado en Gradle y la gestión de dependencias.

1. Estructura de Proyecto y Módulos:

En Android Studio, los proyectos se organizan en módulos, lo que permite una mejor modularidad y gestión del código. Cada módulo puede tener su propio conjunto de archivos y configuraciones. Además, la estructura del proyecto se basa en convenciones específicas de Android, como los directorios `res` para recursos y `java` para archivos fuente.

2. Sistema de Compilación basado en Gradle:

Android Studio utiliza Gradle como su sistema de compilación. Gradle simplifica la gestión de dependencias, configuraciones de compilación y tareas de construcción. Con Gradle, puedes definir fácilmente las dependencias de tu proyecto, configurar las variantes de compilación y automatizar tareas de construcción.

3. Gestión de Dependencias:

En Android Studio, las dependencias de bibliotecas se gestionan a través de archivos de configuración Gradle. Estos archivos especifican las bibliotecas que tu proyecto necesita y Gradle se encarga de descargarlas automáticamente. Esto proporciona un proceso de desarrollo más fluido y elimina la necesidad de gestionar bibliotecas manualmente.

Para migrar desde IntelliJ, aquí tienes un enfoque detallado:

Migración con Nuevo Proyecto:

1. Abre Android Studio y crea un nuevo proyecto desde la pantalla de inicio o desde el menú `Archivo > Nuevo > Nuevo Proyecto`.

2. Copia los archivos fuente y recursos de tu proyecto de IntelliJ al nuevo proyecto en Android Studio.

3. Configura las opciones de proyecto, como el nombre del proyecto, el nombre del paquete y el nivel de API mínima.

4. Utiliza la interfaz gráfica de usuario de Android Studio para agregar dependencias de Gradle según sea necesario.

5. Para ejecutar pruebas de compilación desde la línea de comandos, utiliza `./gradlew build` en sistemas Unix o `gradlew.bat build` en Windows dentro del directorio del proyecto.

Migración con Archivo de Compilación Gradle Personalizado:

1. Crea un nuevo archivo `build.gradle` o `build.gradle.kts` en el directorio raíz de tu proyecto.

2. Configura Gradle para tu proyecto, incluyendo repositorios, dependencias y configuraciones específicas.

3. Elimina los archivos de configuración de IntelliJ y abre tu proyecto en Android Studio.

4. Utiliza comandos como `./gradlew assembleDebug` o `gradlew.bat assembleDebug` para compilar y empaquetar tu aplicación.

Además de estos comandos, puedes utilizar otros comandos Gradle como `./gradlew clean` para limpiar el proyecto o `./gradlew tasks` para ver todas las tareas disponibles en tu proyecto Gradle. Estos comandos te ayudarán a gestionar y construir tu proyecto de manera eficiente en Android Studio.

Version de control basic

Claro, aquí tienes una versión reformulada y algunos comandos básicos:

En Android Studio, puedes administrar el control de versiones de tu aplicación utilizando diversos sistemas, como Git, GitHub, CVS, Mercurial, Subversion y Google Cloud Source Repositories. A continuación, te explico cómo activar el control de versiones en Android Studio:

1. Activación desde el Menú de VCS:

Después de importar tu aplicación en Android Studio, puedes activar el control de versiones desde el menú VCS (Sistema de Control de Versiones) de Android Studio.

2. Habilitar la Integración del Control de Versiones:

Selecciona la opción "Enable Version Control Integration" desde el menú VCS para habilitar la integración del control de versiones.

3. Seleccionar el Sistema de Control de Versiones:

Luego, elige el sistema de control de versiones que prefieras, como Git, GitHub, CVS, Mercurial, Subversion o Google Cloud Source Repositories.

4. Confirmar la Selección:

Haz clic en "OK" para confirmar la asociación del sistema de control de versiones con el proyecto.

Una vez completados estos pasos, Android Studio mostrará opciones adicionales en el menú VCS según el sistema de control de versiones seleccionado. Desde aquí, podrás realizar operaciones de control de versiones, como commit, push, pull, merge, entre otras.

Para verificar la versión de control de versiones instalada en tu sistema o proyecto, puedes utilizar los siguientes comandos:

- `git --version`: Muestra la versión de Git instalada en tu sistema.

- `svn --version`: Muestra la versión de Subversion instalada en tu sistema.

- `hg --version`: Muestra la versión de Mercurial instalada en tu sistema.

- `cvs --version`: Muestra la versión de CVS instalada en tu sistema.

Configurar Android Studio

Android Studio simplifica la configuración inicial a través de asistentes y plantillas que verifican los requisitos del sistema, como la instalación del JDK y la disponibilidad de memoria RAM, mientras establece ajustes predeterminados como la configuración óptima del AVD (Android Virtual Device) y la utilización de imágenes de sistema actualizadas. Además de estas configuraciones automáticas, Android Studio brinda la posibilidad de personalizar aún más mediante dos archivos de configuración principales: `studio.vmoptions` para ajustes de la JVM (Java Virtual Machine) y `idea.properties` para personalizar funciones específicas del IDE (Integrated Development Environment).

El archivo `studio.vmoptions` permite modificar parámetros como el tamaño de la memoria heap y la caché de la JVM de Android Studio, mientras que `idea.properties` posibilita la personalización de propiedades del IDE, como la ubicación de la carpeta de complementos y el tamaño máximo de archivo admitido. Estos archivos pueden ser ubicados y editados fácilmente desde el menú de ayuda de Android Studio, lo que facilita la adaptación a las necesidades particulares del usuario.

En términos de configuración, Android Studio también ofrece documentación específica sobre la creación y gestión de dispositivos virtuales, ejecución de aplicaciones en dispositivos de hardware, e instalación de controladores USB de OEM, entre otros aspectos.

Además de la edición directa de los archivos de configuración, Android Studio permite establecer variables de entorno que apuntan a archivos de configuración específicos en otros lugares. Estas variables, como `STUDIO_VM_OPTIONS` y `STUDIO_PROPERTIES`, brindan una mayor flexibilidad para ajustar la configuración del entorno de desarrollo según las necesidades individuales del usuario.

Para optimizar aún más el rendimiento de Android Studio, especialmente en equipos con recursos limitados, se pueden realizar ajustes específicos. Por ejemplo, reducir el tamaño máximo de la memoria heap disponible a 512 MB puede resultar beneficioso en equipos con especificaciones más modestas. Además, mantener actualizados Gradle y el complemento de Android para Gradle aprovechará las últimas mejoras de rendimiento, mientras que habilitar el modo de ahorro de energía puede desactivar

operaciones que consumen mucha memoria en segundo plano, como la compilación automática incremental.

Otros ajustes para mejorar el rendimiento incluyen la reducción de las comprobaciones de lint innecesarias, realizar depuraciones en dispositivos físicos en lugar de emuladores para reducir el consumo de memoria, y limitar las dependencias a los Servicios de Google Play necesarios para reducir el uso de memoria. Con una configuración adecuada, los usuarios pueden alcanzar un rendimiento óptimo de Android Studio en una variedad de configuraciones de hardware y redes.

Para configurar Android Studio, algunos comandos útiles:

1. Configuración de Variables de Entorno:

- export STUDIO_VM_OPTIONS=path/to/studio.vmoptions: Establece la ubicación del archivo de configuración de opciones de la JVM.
- export STUDIO_PROPERTIES=path/to/idea.properties: Establece la ubicación del archivo de propiedades del IDE.

2. Edición de Archivos de Configuración:

- nano studio.vmoptions: Abre el archivo de opciones de la JVM en el editor de texto nano.
- nano idea.properties: Abre el archivo de propiedades del IDE en el editor de texto nano.

3. Gestión de Versiones de Gradle:

- ./gradlew clean: Limpia el proyecto y elimina los archivos generados por Gradle.
- ./gradlew tasks: Muestra todas las tareas disponibles en el proyecto Gradle.
- ./gradlew assembleDebug: Compila y empaqueta la aplicación en modo de depuración.

4. Actualización de Android Studio:

- ./studio.sh --disable-update-check: Deshabilita la verificación automática de actualizaciones al iniciar Android Studio.
- ./studio.sh --reset-options: Restablece todas las opciones de configuración de Android Studio a los valores predeterminados.

5. Gestión de Complementos:

- ./studio.sh --list-plugins: Muestra una lista de todos los complementos instalados en Android Studio.

- ./studio.sh --install-plugin <nombre_del_plugin.zip>: Instala un complemento específico desde un archivo ZIP.
- ./studio.sh --uninstall-plugin <nombre_del_plugin>: Desinstala un complemento específico.

Update the IDE and SDK tools

Para mantener tu entorno de desarrollo actualizado, debes actualizar tanto Android Studio como las herramientas del SDK de forma regular. Si instalaste Android Studio con JetBrains Toolbox, este se encargará automáticamente de las actualizaciones. Si lo instalaste manualmente, puedes buscar actualizaciones desde la configuración del IDE. Además, es esencial mantener actualizadas las herramientas del SDK utilizando el Administrador de SDK de Android.

1. Actualizar Android Studio con JetBrains Toolbox:

- Si utilizas JetBrains Toolbox, las actualizaciones se gestionan automáticamente.
- Para instalar Toolbox y Android Studio, sigue las instrucciones en el sitio web oficial de JetBrains

2. Actualizar Android Studio manualmente:

- Para buscar actualizaciones manualmente:
- En Windows/Linux: Haz clic en `File > Settings > Appearance & Behavior > System Settings > Updates`.
- En macOS: Haz clic en `Android Studio > Check for Updates`.
- Alternativamente, puedes utilizar comandos de línea de comandos para buscar actualizaciones.

3. Actualizar herramientas del SDK de Android:

- Utiliza el Administrador de SDK de Android para descargar y actualizar herramientas, plataformas y otros componentes.
- Abre el Administrador de SDK desde Android Studio haciendo clic en `Tools > SDK Manager`.
- Para instalar herramientas desde la línea de comandos, utiliza `sdkmanager`.

Comandos útiles:

- Para abrir Android Studio desde la línea de comandos:

```
./studio.sh       # En sistemas Unix
studio.bat        # En Windows
```

Para buscar actualizaciones manualmente desde la línea de comandos:

```
./studio.sh --checkForUpdate       # En sistemas Unix
studio.bat --checkForUpdate        # En Windows
```

Para abrir el Administrador de SDK desde la línea de comandos:

```
./studio.sh --sdk_manager          # En sistemas Unix
studio.bat --sdk_manager           # En Windows
```

Para actualizar herramientas del SDK desde la línea de comandos:

```
sdkmanager --update                # En sistemas Unix y Windows
```

Siguiendo estos pasos y utilizando los comandos adecuados, podrás mantener tu entorno de desarrollo actualizado y aprovechar las últimas características y mejoras en Android Studio y las herramientas del SDK de Android.