

Material de estudio obligatorio Eje Temático N° 1

Sitio: [Instituto Superior Politécnico Córdoba](#)
Curso: Programador de Aplicaciones Móviles - TSDWAD - 2022
Libro: Material de estudio obligatorio Eje Temático N° 1

Imprimido por: Ezequiel Maximiliano GIAMPAOLI
Día: jueves, 31 agosto 2023, 2:42 PM

Descripción

Aplicaciones Móviles

Tabla de contenidos

1. Introduccion

2. Tipos de aplicaciones móviles

3. Sistemas operativos móvil

4. Desarrollo de aplicaciones móviles

5. Instalación - Android Studio

6. Introducción a Android Studio

7. Emulador

8. Links - pdfs- java

9. Introduccion java

9.1. ByteCode

9.2. Qué es Java

1. Introduccion

1. Introducción

Los dispositivos móviles forman parte de nuestra vida cotidiana a tal punto que para muchos puede ser preferible utilizar un dispositivo móvil para realizar tareas que antes las hacía en ordenadores.

Partiendo de esto, se desarrollan una gran cantidad de aplicaciones de distintas funcionalidades que buscan cubrir las necesidades actuales.

Las aplicaciones móviles han mejorado la manera en que los usuarios interactúan con los sistemas de información. Por medio de estas de forma cotidiana, se informan, trabajan, se comunican, se divierten, etc

Existen destinadas a la comunicación, a la educación, a la organización, a las actividades diarias como los pagos, a la recreación, al control de aparatos electrónicos, etc.

1.1 Definiciones de aplicaciones móviles

¿Que es una aplicación?

Una aplicación se conoce como un grupo de herramientas creada para realizar tareas y ciertos trabajos en específico. Son herramientas con características especiales, orientadas para dispositivos pequeños como: tabletas o teléfonos inteligentes”.

¿Qué es un smartphone o dispositivo móvil inteligente?

Son dispositivos con diferentes capacidades desde conexión a internet a procesar datos, creados para ejecutar varias funciones generales.

Los teléfonos inteligentes se utilizan para reemplazar otros dispositivos como: cámaras digitales, relojes, grabadoras de video, etc. Ya que un teléfono inteligente es una pequeña computadora.

¿Qué es una aplicación móvil?

Una aplicación móvil es un programa diseñado para ser utilizado en un dispositivo móvil que ofrece una solución a un problema o necesidad.

2. Tipos de aplicaciones móviles

Tipos de aplicaciones móviles

Existen varios tipos de aplicaciones móviles, dependiendo de la necesidad que tengas. Así, cada aplicación móvil se realiza con un propósito bien definido y se busca que cubra la necesidad una necesidad específica.

Es necesario advertir que cada tipo de aplicación que existen, son diferentes entre si y a su vez presentan ventajas y desventajas.

Los tipos de aplicaciones móviles son las siguientes:

- **Aplicaciones nativas**
- **Aplicaciones web**
- **Aplicaciones híbridas**
- **Aplicaciones en la nube**

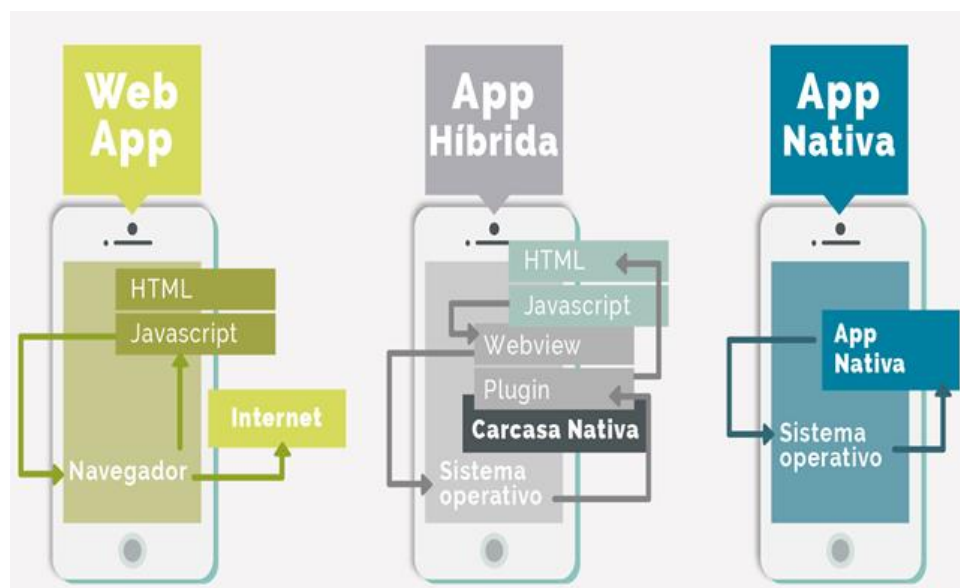


Figura: Tipos de aplicaciones móviles

1.2.1 Aplicaciones nativas

Las aplicaciones móviles nativas como su nombre sugiere son aplicaciones desarrolladas para el sistema operativo donde va a correr la aplicación.

Las aplicaciones nativas:

- no son multiplataformas, entonces se desarrolla una aplicación solo para su respectivo sistema operativo, para los demás se tendría que crear una aplicación para cada uno de los demás sistemas operativos.
- permiten aprovechar al máximo las características de los dispositivos móviles.

Ejemplo:

si desarrollamos una aplicación para el sistema operativo Android, su lenguaje por defecto será Java

(lo mismo sería con los demás sistemas operativos, cada sistema operativo tiene su propio lenguaje por defecto).

Consideraciones

- Una de las grandes ventajas que tienen las aplicaciones nativas es el gran uso que se le da al hardware de los equipos, por esto las aplicaciones nativas sacan todo el potencial del hardware tales como la cámara, GPS, LCD, etc.
- Estas aplicaciones se descargan como todas las demás en el Play Store y App Store si se desea utilizar estos tipos de aplicaciones.
- La principal desventaja de estas aplicaciones es su alto costo. Ya que estas aplicaciones se deben crear para cada sistema operativo y por esto es el costo tan elevado de crear una aplicación para cada sistema operativo. Por ejemplo: se requiere un equipo especializado en dicha tecnología para su desarrollo y luego para su respectivo mantenimiento.

1.2.2 Aplicaciones web

Una aplicación web es aquella que se ejecuta en el navegador web de un dispositivo móvil cuando se ha accedido a una URL.

- Las aplicaciones web están pensadas para multiplataforma para que puedan correr en distintos sistemas operativos a diferencia de las aplicaciones nativas, aunque ellas poseen una interfaz similar a las nativas y una experiencia de navegación.
- Operan desde la web sin la necesidad de ser descargadas e instaladas en los dispositivos móviles obteniendo una gran ventaja, no necesitan cumplir requisitos como en el sistema operativo.

Consideraciones

Las aplicaciones web se pueden desarrollar en lenguajes de programación muy populares tales como: HTML, CSS, JavaScript y PHP y eso hace su desarrollo más fácil.

Las aplicaciones web son multiplataforma capaces de correr en diferentes sistemas operativos, pero no tienen esa visibilidad que las nativas requieren una URL para poder acceder a ellas y si no guardas o recuerdas las URL es difícil acceder a ellas.

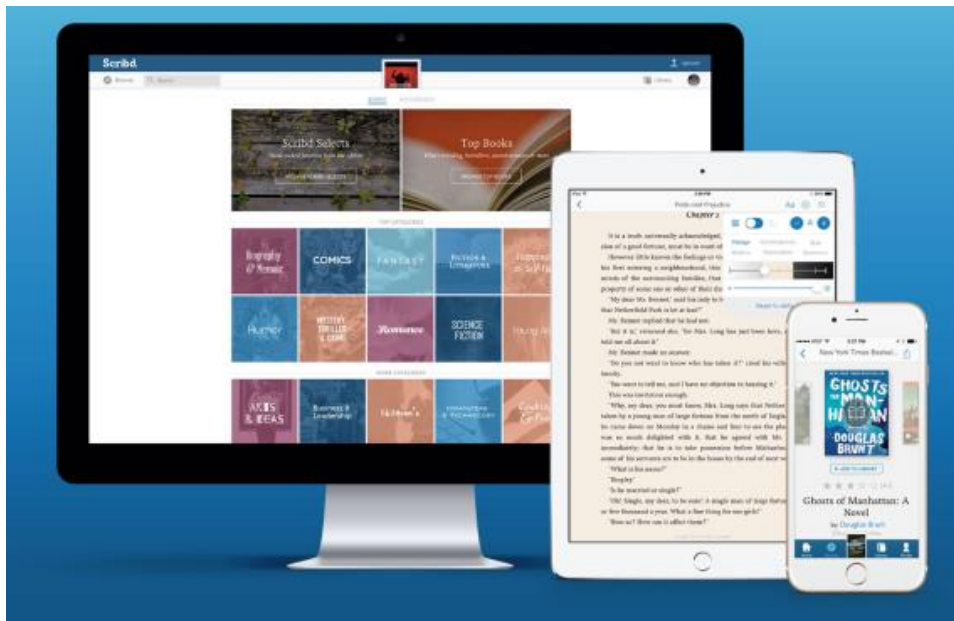


Figura: Aplicaciones web

1.2.3 Aplicaciones híbridas

Las aplicaciones híbridas son aplicaciones que poseen características de aplicaciones nativas y de aplicaciones web que utilizan lenguajes como HTML, CSS y JavaScript y se adaptan a los dispositivos móviles mediante un framework.

Las aplicaciones híbridas:

- Pueden funcionar en diferentes sistemas operativos (capacidad de trabajar en distintas plataformas.)
- Se ahorra tiempo y dinero en el desarrollo
- Son Rápidas de desarrollar

Consideraciones

Es muy difícil diferenciar una aplicación nativa de una híbrida a simple vista, pero una manera regular de hacerlo es ver de manera visual como pueden lucir cada una en distintos sistemas operativos.

- Las aplicaciones híbridas son perfectas cuando queremos llevar una aplicación web ya creada a una aplicación móvil porque podemos usar características de ella.
- La desventaja principal es que no puede acceder a toda la capacidad del hardware y no puede funcionar sin conexión a internet gracias a que parte de la aplicación es web.
- Las aplicaciones están desarrolladas para Android, eso hace que la mayoría de los desarrolladores vean más lógico desarrollar nativo para Android.
- Ciertas especificaciones de algunos dispositivos hace que las aplicaciones híbridas sean algo lentas y funcionan muy pesadas.

Actualmente muchas de las aplicaciones que conocemos por su popularidad han sido desarrolladas de manera híbrida aplicaciones tales como: Facebook, WhatsApp, Uber e Instagram.

1.2.4 Aplicaciones cloud

Las aplicaciones cloud son aplicaciones que son alojadas desde una plataforma de nube, ellas pueden ser accedidas mediante una URL desde el dispositivo móvil.

Cloud: es la capacidad de adquirir recursos bajo demanda de recursos tecnológicos como almacenamiento.

El objetivo de las aplicaciones cloud es poder dar una experiencia similar a las aplicaciones nativas mediante la conexión del navegador, con esto se busca poder realizar pruebas mediante estos tipos de aplicaciones. Esto es una buena manera de probar aplicaciones sin instalarla y que no ocupen espacio en nuestros dispositivos.



Figura: Aplicación cloud

3. Sistemas operativos móvil

Sistemas operativos móvil

1.3.1 Android

Antes de hablar del sistema operativo Android, haremos referencia a Linux. Linux es un sistema operativo que actúa como intermediario, es decir, un puente entre el dispositivo físico y el código de instrucción de un programa.

- Linux es un sistema operativo open source, es decir, posee una licencia descentralizada para que cualquier usuario tenga acceso a sus recursos.
- Al inicio Linux no era un sistema operativo, sino sólo era un kernel
- Un kernel es el núcleo de un sistema operativo creado por Linus Torvalds.
- Este kernel se convirtió en un sistema operativo al unirse el sistema operativo GNU.
- GNU es un conjunto de paquetes que se unió al proyecto Linux, gracias a esto Linux se conoce como GNU/Linux.

1.3.2 ¿Qué es Android?

Es un sistema operativo basado en Linux, diseñado para móviles inteligente táctiles.

- Android como muchas distribuciones de Linux, su kernel está basado en Linux, pero sus componentes fueron desarrollados por Google y otros componentes open source.
- Este sistema operativo inicialmente nació en una empresa que lleva su propio nombre Android inc.
- El proyecto original era crear un sistema operativo para cámaras digitales, pero no tenía a donde llegar así que se integró a dispositivos móviles.
- Luego Google compró la empresa y continuó con el proyecto bajo su nombre. Más adelante Google anunció la primera versión de Android para dispositivos móviles.

Este sistema operativo nos ofrece la capacidad de crear aplicaciones que usen funciones de los dispositivos tales como (GPS, Cámara, Agenda, etc). Esto es gracias a una variación de Java que se conoce como Dalvik. Todo esto se puede lograr ya que las aplicaciones que se desarrollan en Android son desarrolladas en Java.

Una de las grandes ventajas que posee este sistema operativo es que es open source, así que los usuarios no tienen que pagar ninguna licencia para usarlo y pueden ver sin problema el código fuente. Por esto, muchas empresas lo utilizan para desarrollar en sus dispositivos móviles, ya que pueden hacer todo tipo de modificaciones y ajustarlo en sus dispositivos.

1.3.3 IOS

Un sistema operativo propietario de la Apple para sus dispositivos móviles.

Este sistema operativo lleva varios años en el mercado desde la primera vez que fue mostrado con el dispositivo móvil iPhone por el fallecido Steve Job.

Con el transcurso del tiempo Apple ha seguido manteniendo su sistema operativo mediante actualizaciones por sus diferentes dispositivos móviles para mejorar el entorno de los usuarios.

- Este sistema operativo tiene muchas similitudes a Android dando la capacidad al igual que Android de usar las aplicaciones y las funciones de nuestro dispositivo móvil.
- Algunas de las funciones realizadas en los dispositivos móviles, es poder conectarnos a internet, utilizar las cámaras de nuestro dispositivo, poder enviar y recibir mensajes y almacenar información tanto en el dispositivo como en la nube.

La diferencia principal de este sistema operativo con Android es, que es un sistema operativo que solo funciona en dispositivos Apple, a diferencia de Android que está disponible para cualquier dispositivo móvil.

Una gran ventaja que tiene el sistema operativo IOS es la seguridad, posee mejor seguridad y velocidad que el sistema operativo Android gracias a su compatibilidad con los dispositivos.

4. Desarrollo de aplicaciones móviles

Desarrollo de aplicaciones móviles

1.4.1 Conceptos de desarrollo de aplicaciones móviles

El desarrollo de software es el conjunto estructurado y repetitivo de actividades que se realizan de manera sistemática y cuyo objetivo principal es el desarrollo o evolución de un software. Entonces el desarrollo de software de aplicaciones móviles como el procedimiento para crear software para dispositivos inteligentes.

1.4.2 Plataformas de desarrollo de software móvil

Las plataformas de desarrollo son entornos dedicados especialmente para los desarrolladores de aplicaciones. Algunos de los entornos más usados para desarrollo de software móvil tales como:

Xamarin: Se puede describir como una plataforma open source para crear aplicaciones modernas. Su lenguaje principal es C# para crear aplicaciones multiplataformas. Xamarin ayuda a compartir las aplicaciones de los desarrolladores a distintas plataformas. Xamarin studio puede ser añadido a visual studio si ya posees visual studio.

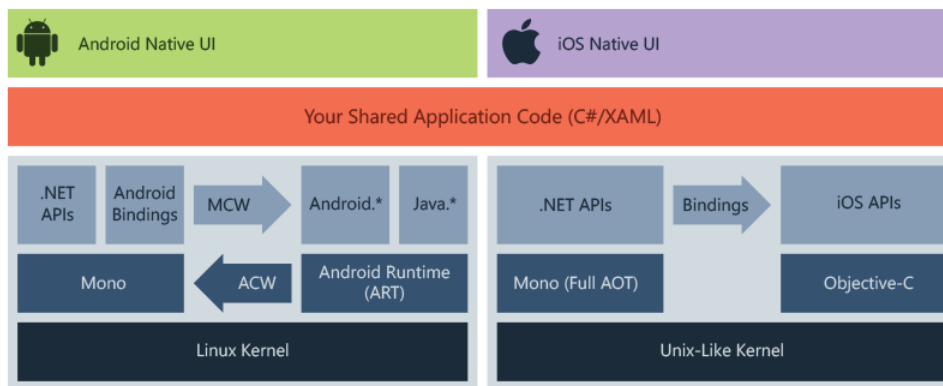


Figura: La arquitectura general de una aplicación Xamarin multiplataforma

Android studio: Se definiría como un entorno de desarrollo para desarrollo de aplicaciones Android, basado en el entorno de intellij. Esto ayuda mucho a los desarrolladores a la hora de escribir código lo que hace que él esté aprendiendo mientras escribes código. El entorno posee un análisis de errores muy eficaz, capaz de detectar errores casi inmediatamente, para brindar soluciones rápidas.



Figura 1.6: Android studio

XCode

Se define como el entorno de desarrollo móvil para el desarrollo de aplicaciones para iOS. Fue creado por Apple para colaborar con interfaz builder, esta posee un grupo de herramientas que incluye compiladores GNU y puede compilar C. El entorno Xcode posee las herramientas necesarias para todo el proceso de desarrollo móvil para los dispositivos móviles Apple.



5. Instalación - Android Studio

Cómo instalar Android Studio

Puedes configurar Android Studio con unos pocos clics.

Primero, asegúrate de [descargar la versión más reciente de Android Studio](#).

Windows

Para instalar Android Studio en Windows, haz lo siguiente:

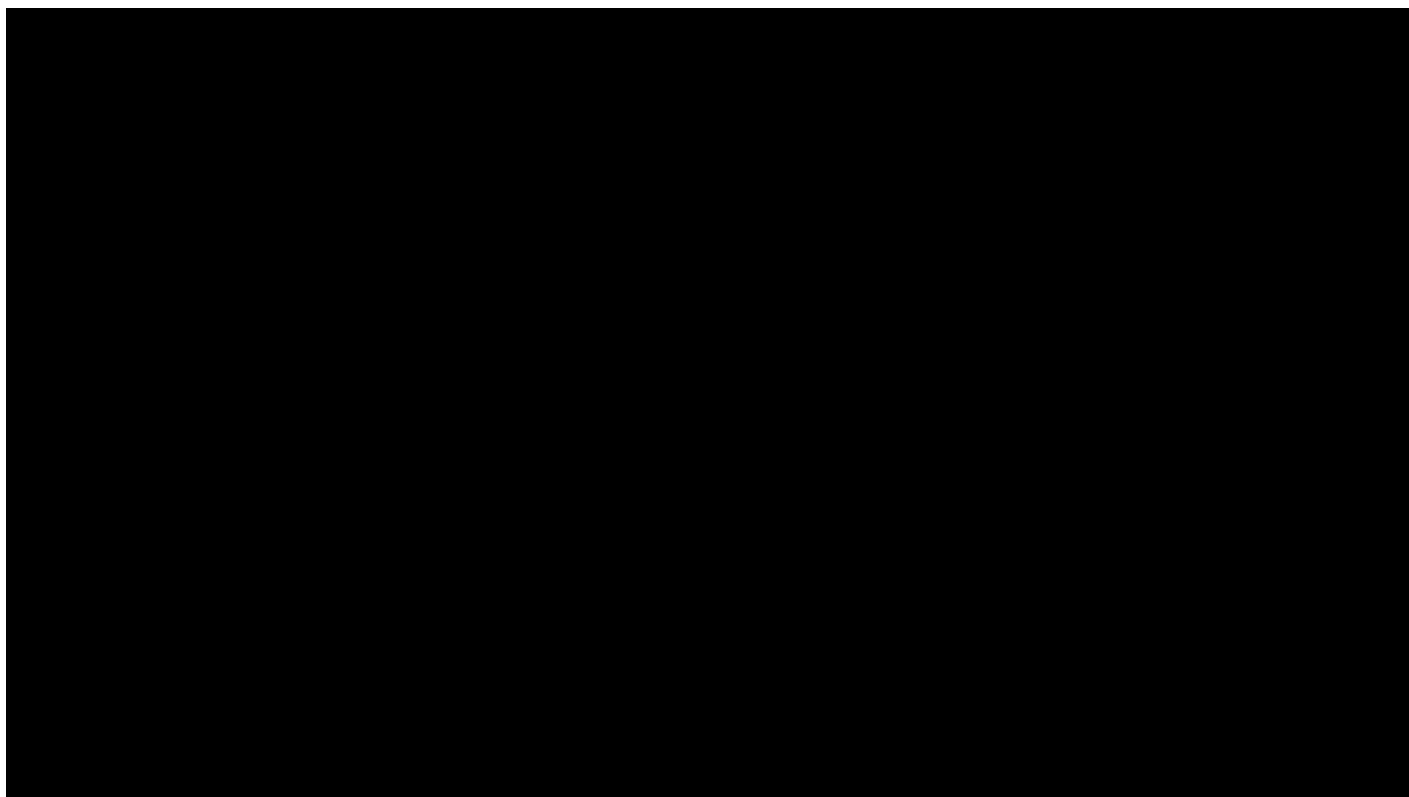
1. Si descargaste un archivo `.exe` (recomendado), haz doble clic en él para iniciarlo.

Si descargaste un archivo `.zip`, extráelo y copia la carpeta android-studio en la carpeta Archivos de programa. A continuación, abre la carpeta android-studio > bin y, luego, inicia `studio64.exe` (para máquinas de 64 bits) o `studio.exe` (para las de 32 bits).

2. Sigue los pasos del asistente de configuración en Android Studio y asegúrate de instalar los paquetes de SDK que recomiende.

En el siguiente video, se muestra cada paso del procedimiento de configuración cuando se usa la descarga de `.exe` recomendada

[Link](#)



Cuando haya herramientas nuevas y otras API disponibles, Android Studio te lo informará por medio de una ventana emergente. También puedes buscar actualizaciones si haces clic en

Help > Check for Update.

Linux

Para instalar Android Studio en Linux, haz lo siguiente:

1. Extrae el archivo **.zip** que descargaste en una ubicación apropiada para tus aplicaciones, como dentro de **/usr/local/** para tu perfil de usuario o **/opt/** para usuarios compartidos.

Si usas una versión de Linux de 64 bits, asegúrate de instalar primero las [bibliotecas requeridas para máquinas de 64bits](#).

2. Para iniciar Android Studio, abre una terminal, navega al directorio **android-studio/bin/** y ejecuta **studio.sh**.
3. Selecciona si deseas o no importar configuraciones anteriores de Android Studio y, luego, haz clic en **OK**.
4. El asistente de configuración de Android Studio te guiará por el resto de la configuración, lo que incluye la descarga de los componentes del SDK de Android que se necesiten para el desarrollo.

Bibliotecas requeridas para máquinas de 64 bits

Si ejecutas una versión de Ubuntu de 64 bits, deberás instalar algunas bibliotecas de 32 bits con el siguiente comando:

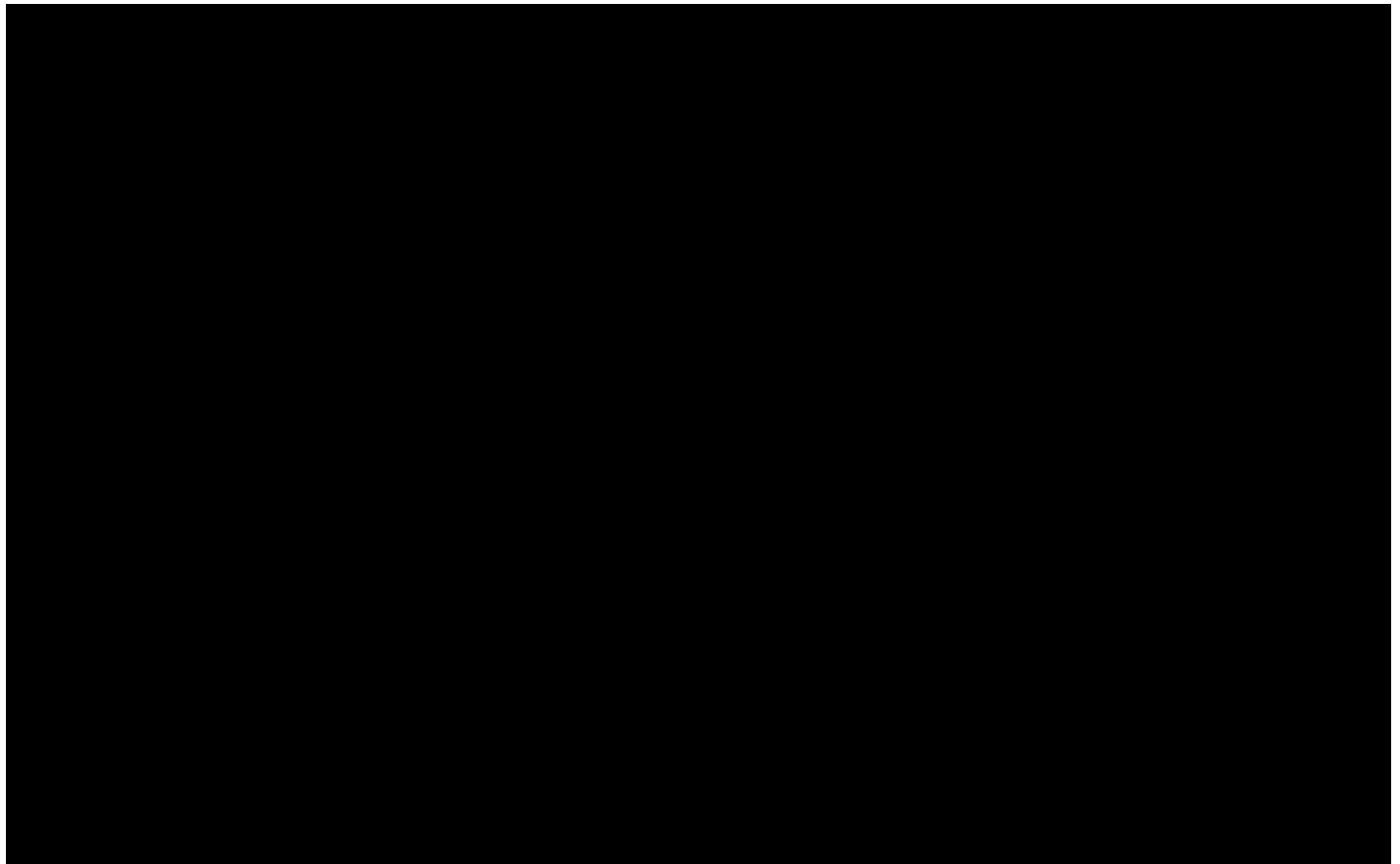
```
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1 libbz2-1.0:i386
```

Si usas Fedora de 64 bits, el comando es el siguiente:

```
sudo yum install zlib.i686 ncurses-libs.i686 bzip2-libs.i686
```

En el siguiente video se muestra cada paso del procedimiento de configuración recomendado.

[Link](#)



Cuando haya herramientas nuevas y otras API disponibles, Android Studio te lo informará por medio de una ventana emergente. También puedes buscar actualizaciones si haces clic en

Help >Check for Update.

6. Introducción a Android Studio

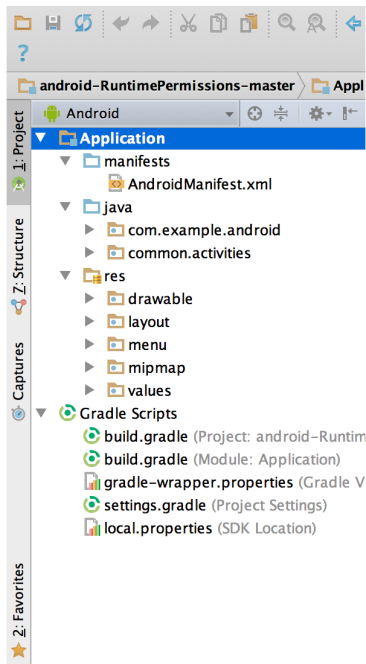
Introducción a Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA (<https://www.jetbrains.com/idea/>) .

Características

Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros Compatibilidad con C++ y NDK
- Compatibilidad integrada con Google Cloud Platform (<https://cloud.google.com/tools/android-studio/docs/?hl=es-419>) , que facilita la integración con Google Cloud Messaging y App Engine



Estructura del proyecto

Cada proyecto de Android Studio incluye uno o más módulos con archivos de código fuente y archivos de recursos.

Entre los tipos de módulos se incluyen los siguientes:

- Módulos de apps para Android
- Módulos de biblioteca
- Módulos de Google App Engine

Vistas

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyecto de Android, como se ve en la figura.

Esta vista está organizada en módulos para que puedas acceder rápidamente a los archivos fuente clave de tu proyecto.

Puedes ver todos los archivos de compilación en el nivel superior de Secuencias de comando de Gradle y cada módulo de app contiene las siguientes carpetas:

1. **manifests:** contiene el archivo AndroidManifest.xml .
2. **java:** contiene los archivos de código fuente Java, incluido el código de prueba de JUnit.
3. **res:** contiene todos los recursos sin código, como diseños XML, strings de IU e imágenes de mapa de bits.

Consideraciones:

1) La estructura del proyecto de Android en el disco difiere de esta representación plana. Para ver

la estructura real de archivos del proyecto, selecciona Project en el menú desplegable Project

(en la figura 1, se muestra como Android).

2) También puedes personalizar la vista de los archivos del proyecto para concentrarte en

aspectos específicos del desarrollo de tu app.

Por ejemplo, si seleccionas la vista Problems de tu proyecto, se mostrarán vínculos a los archivos fuente que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre faltante en un elemento XML en un archivo de Android de diseño.

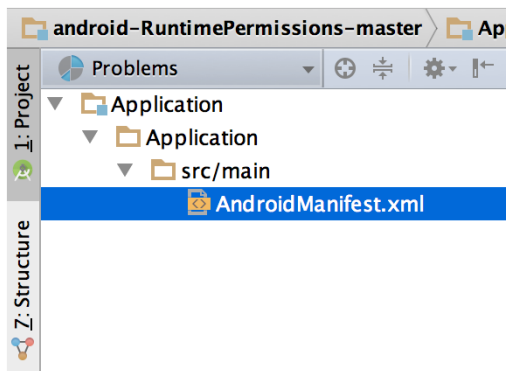
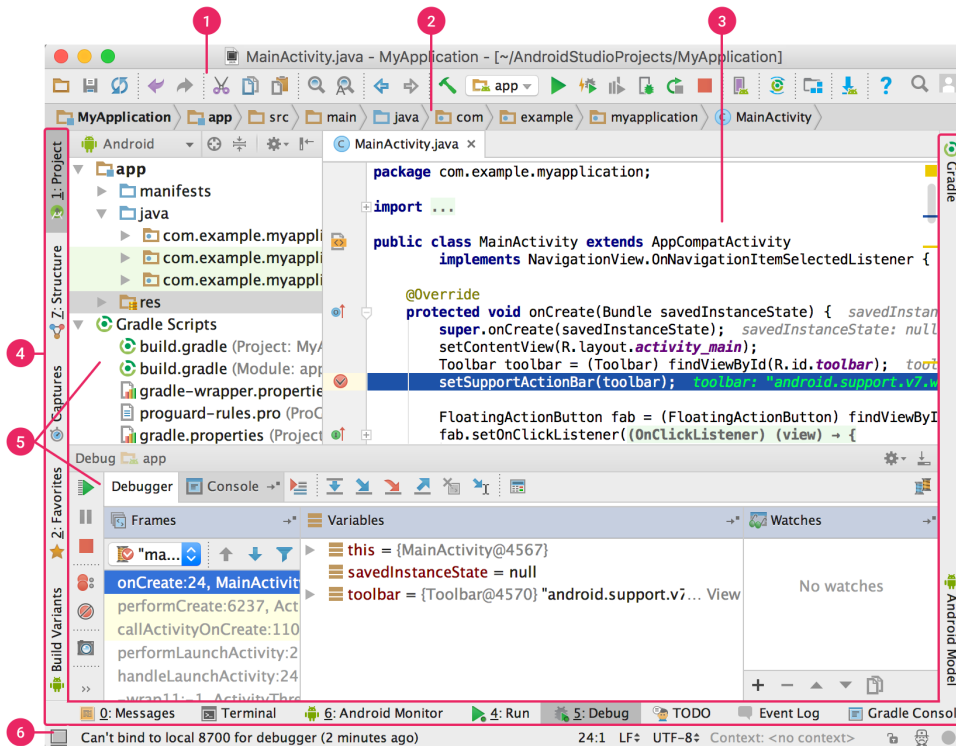


Figura: Archivos del proyecto en la vista Problems, en la que se muestra un archivo de diseño con un problema

Interfaz de usuario

La ventana principal de Android Studio consta de varias áreas lógicas identificadas en la figura siguiente:

Figura: Ventana principal de Android Studio



1- La barra de herramientas te permite realizar una gran variedad de acciones, como ejecutar tu app e iniciar las herramientas de Android.

2- La barra de navegación te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana Project.

3- La ventana del editor es el área en la que puedes crear y modificar código. Según el tipo de actividad actual, el editor puede cambiar.

Por ejemplo, cuando ves un archivo de diseño, el editor muestra el Editor de diseño.

4- La barra de la ventana de herramientas se encuentra afuera de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.

5- Las ventanas de herramientas te brindan acceso a tareas específicas, como la administración de proyectos, la búsqueda, el control de versiones, entre otras. Puedes expandirlas y contraerlas.

6- En la barra de estado, se muestra el estado de tu proyecto y el IDE, además de advertencias o mensajes.

Organización

- Puedes organizar la ventana principal para tener más espacio en pantalla ocultando o desplazando las barras y ventanas de herramientas. También puedes usar combinaciones de teclas para acceder a la mayoría de las funciones del IDE.
- En cualquier momento, puedes realizar búsquedas en el código fuente, las bases de datos, las acciones y los elementos de la interfaz de usuario, entre otros.

Para ello, presiona dos veces la tecla Mayús o haz clic en la lupa, en la esquina superior derecha de la ventana de Android Studio.

Esto puede ser muy útil, por ejemplo, si intentas localizar una acción específica del IDE que olvidaste cómo activar.

Ventanas de Herramientas

En lugar de usar perspectivas predeterminadas, Android Studio sigue tu contexto y te ofrece automáticamente ventanas de herramientas relevantes mientras trabajas.

De manera predeterminada, las ventanas de herramientas más usadas se fijan a la barra de la ventana de herramientas en los bordes de la ventana de la app.

- Para expandir o contraer una ventana de herramientas, haz clic en el nombre de la herramienta, en la barra de la ventana de herramientas. También puedes arrastrar, fijar, no fijar, adjuntar y ocultar ventanas de herramientas.
- Para volver al diseño predeterminado actual de la ventana de herramientas, haz clic en **Window > Restore Default Layout** o personaliza tu diseño predeterminado haciendo clic en **Window > Store Current Layout as Default**.
- Para mostrar u ocultar la barra de la ventana de herramientas completa, haz clic en el ícono de ventana, en la esquina inferior izquierda de la ventana de Android Studio.
- Para ubicar una ventana de herramientas específica, desplázate sobre su ícono y selecciónala en el menú.

También puedes usar combinaciones de teclas para abrir ventanas de herramientas. En la tabla 1, se muestran las combinaciones de teclas para las ventanas más comunes.

Tabla 1: Combinaciones de teclas para algunas ventanas de herramientas útiles

Ventana de herramientas	Windows y Linux
Proyecto	Alt + 1
Control de versiones	Alt + 9
Ejecutar	Mayús + F10
Depurar	Mayús + F9
Logcat	Alt + 6
Volver al editor	Esc
Ocultar todas las ventanas de herramientas	Control + Mayús + F12

Si quieres ocultar todas las barras de herramientas, ventanas de herramientas y pestañas del editor, haz clic en **View > Enter Distraction Free Mode**.

- De esta manera, se habilita el modo Distraction Free Mode. Para salir del modo Distraction Free Mode, haz clic en **View > Exit Distraction Free Mode**.

Puedes usar la Búsqueda rápida para buscar y filtrar en la mayoría de las ventanas de herramientas en Android Studio. Para usar la Búsqueda rápida, selecciona la ventana de herramientas y, luego, escribe tu búsqueda.

Cómo completar el código

Android Studio ofrece tres opciones para completar el código, a las que puedes acceder con combinaciones de teclas.

Tabla 2: Combinaciones de teclas para completar código

Tipo	Descripción	Windows y Linux
Completar de manera básica	<p>Muestra sugerencias básicas para variables, tipos, métodos y expresiones, entre otras.</p> <p>Si completas de manera básica dos veces seguidas, verás más resultados.</p> <p>Entre otros, miembros privados y miembros estáticos sin importar.</p>	Control + Barra espaciadora

Completar de manera inteligente	<p>Muestra opciones relevantes en función del contexto.</p> <p>La función Completar de manera inteligente reconoce el tipo y los flujos de datos previstos.</p> <p>Si completas de manera inteligente dos veces seguidas, verás más resultados.</p> <p>Por ejemplo, cadenas.</p>	<p>Control + Mayús</p> <p>+ Barra espaciadora</p>
Completar instrucciones	<p>Completa la instrucción actual agregando elementos que faltan, como paréntesis, corchetes, llaves y formato, entre otros.</p>	<p>Control + Mayús + Intro</p>

También puedes realizar correcciones rápidas para mostrar acciones de intención si presionas Alt + Intro.

Comandos de Navegación

Sugerencias para ayudarte a desplazarte por Android Studio.

Comando	Descripción
Control + E	<p>Alterna entre los archivos a los que accediste recientemente mediante la acción Recent Files. Presiona Control + E para activar la acción Archivos recientes.</p>
Control + F12	<p>Usa la acción File Structure para visualizar la estructura del archivo actual. Para activar la acción File Structure, presiona Control + F12. Con esta acción, podrás navegar rápidamente hacia cualquier parte del archivo actual.</p>
Control + N	<p>Busca una clase específica en tu proyecto y navega hacia ella con la acción Navigate to Class. Para activar la acción, presiona Control + N.</p>
Control + Mayús + N	<p>Para navegar a un archivo o una carpeta, usa la acción Navigate to File. Si quieres activar la acción Navigate to File, presiona Control + Mayús + N . Para buscar carpetas en lugar de archivos, agrega una / al final de la expresión.</p>
Control + Mayús + Alt + N	<p>Navega a un método o campo por nombre con la acción Navigate to Symbol. Para activar la acción Navigate to Symbol, presiona Control + Mayús + Alt + N</p>
presiona Alt + F7	<p>Encuentra todas las partes de código que hagan referencia a la clase, el método, el campo, el parámetro o la instrucción en la posición actual. Para ello, presiona Alt + F7</p>

Estilo y formato

Para personalizar la configuración de estilo de tu código, haz clic en **File > Settings > Editor > Code Style**

Comando**Descripción****Control + Alt + L**

Si bien el IDE aplica formato de manera automática mientras trabajas, también puedes llamar explícitamente a la acción Reformat Code si presionas **Control + Alt + L**

Control + Alt + I

Aplicar sangrías automáticas a todas las líneas si presionas **Control + Alt + I**

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
}
```

Figura 4: Código antes de la aplicación de formato

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
}

// Get reference to the drawer layout and set event listener
```


Formatted 7 lines
Show reformat dialog:  ⌘ L

Figura 5: Código después de la aplicación de formato

Control de Versiones

Android Studio admite diferentes sistemas de control de versión (VCS), incluidos Git, GitHub, CVS, Mercurial, Subversion y Google Cloud Source Repositories.

Después de importar tu app a Android Studio, usa las opciones del menú del VCS de Android Studio a fin de habilitar la compatibilidad con VCS para el sistema de control de versión deseado, crear un repositorio, importar los archivos nuevos al control de versión y realizar otras operaciones de control de versión:

1. En el menú del VCS de Android Studio, haz clic en Enable Version Control Integration.
2. En el menú desplegable, selecciona un sistema de control de versión para asociarlo con la raíz del proyecto y, luego, haz clic en OK.

En el menú del VCS se mostrarán diversas opciones de control de versión según el sistema que hayas seleccionado.

Nota: También puedes usar la opción del menú **File > Settings > Version Control** para configurar y modificar los ajustes de control de versión.

Sistema de compilación de Gradle

Android Studio usa Gradle como base del sistema de compilación, y el complemento de Android para Gradle proporciona capacidades específicas de Android.

Este sistema de compilación se ejecuta en una herramienta integrada desde el menú de Android Studio, y lo hace independientemente de la línea de comandos.

Puedes usar las funciones del sistema de compilación para lo siguiente:

Personalizar, configurar y extender el proceso de compilación Crear varios APK para tu app; diferentes funciones usan el mismo proyecto y los mismos módulos Volver a usar códigos y recursos en conjuntos de archivos fuente Gracias a la flexibilidad de Gradle, puedes lograrlo sin modificar los archivos fuente de tu app.

Los archivos de compilación de Android Studio se denominan build.gradle . Son archivos de texto sin formato que usan la sintaxis Groovy (<http://groovy-lang.org>) a fin de configurar la compilación con elementos que proporciona el complemento de Android para Gradle.

Cada proyecto tiene un archivo de compilación de nivel superior para todo el proyecto y archivos de compilación de nivel de módulo independientes para cada módulo.

Cuando importas un proyecto existente, Android Studio genera automáticamente los archivos de compilación necesarios.

Variantes de compilación

El sistema de compilación puede ayudarte a crear diferentes versiones de la misma app a partir de un solo proyecto.

- Esto resulta útil cuando tienes una versión gratuita o una versión paga de tu app, o si quieres distribuir múltiples APK para diferentes configuraciones de dispositivos en Google Play.

Administración de dependencias

Las dependencias para tu proyecto están especificadas por nombre en el archivo build.gradle. Gradle se ocupa de buscar tus dependencias y hacer que estén disponibles en tu compilación. Puedes declarar dependencias de módulos, dependencias binarias remotas y dependencias binarias locales en tu archivo build.gradle.

Android Studio configura los proyectos para que usen el repositorio central de Maven de manera predeterminada.

Herramientas de depuración y perfil

Android Studio te ayuda a depurar y mejorar el rendimiento de tu código. Esto incluye herramientas integradas de depuración y análisis de rendimiento.

Depuración integrada Usa la depuración integrada para mejorar las revisiones de código en la vista del depurador con verificación integrada de referencias, expresiones y valores de variables.

La información de depuración integrada incluye:

- Valores de variables integradas
- Referencia a objetos que hacen referencia a un objeto seleccionado
- Valores de retorno de métodos
- Expresiones lambda y de operador
- Valores de información sobre herramientas

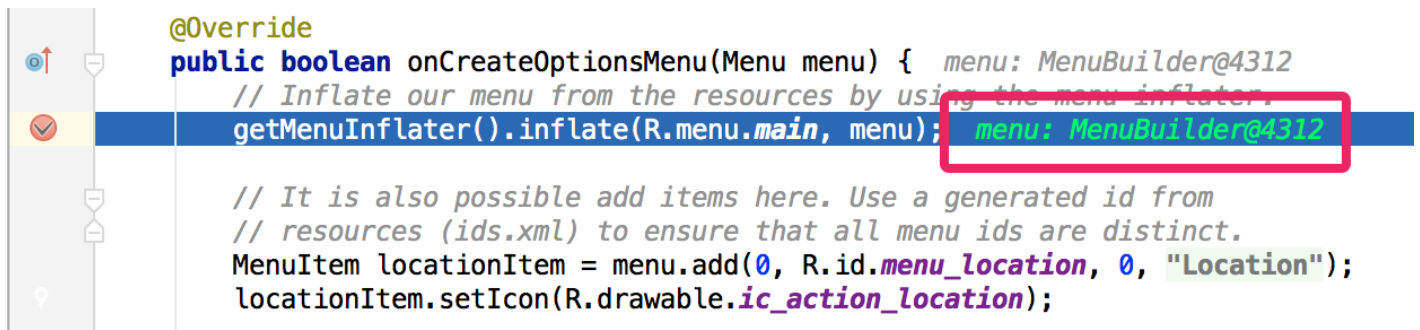


Figura: Valor de una variable integrada

Para habilitar la depuración integrada, en la ventana Debug, haz clic en Settings y selecciona la casilla de verificación para activar la opción Show Values Inline.

Inspecciones de código

Cada vez que compilas tu programa, Android Studio ejecuta automáticamente inspecciones de Lint

y otras inspecciones de IDE (<https://www.jetbrains.com/help/idea/2020.2/code-inspection.html>) para ayudarte a identificar y corregir problemas con la calidad estructural de tu código de manera sencilla.

La herramienta lint comprueba los archivos de origen de tu proyecto de Android en busca de posibles errores y para realizar mejoras relacionadas con la precisión, la seguridad, el rendimiento, la usabilidad, la accesibilidad y la internacionalización.

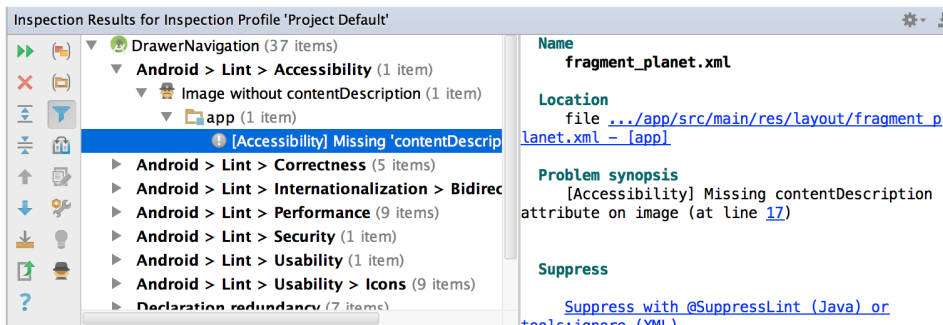


Figura: Resultados de una inspección de Lint en Android Studio

Además de las verificaciones de Lint, Android Studio también realiza inspecciones de código de IntelliJ y valida anotaciones para simplificar tu flujo de trabajo de codificación.

Anotaciones en Android Studio

Android Studio admite anotaciones de variables, parámetros y valores de retorno a fin de ayudarte a detectar errores, como excepciones de puntero nulo y conflictos de tipos de recurso.

Android SDK Manager empaqueta la biblioteca de compatibilidad-anotaciones en el Repositorio de compatibilidad de Android para usarla con Android Studio.

Android Studio valida las anotaciones configuradas durante la inspección del código.

Mensajes de registro

Cuando creas y ejecutas tu app con Android Studio, puedes ver los resultados de adb

y los mensajes de registro del dispositivo en la ventana de Logcat

FUENTE DE INFORMACIÓN: <https://developer.android.com/studio/>

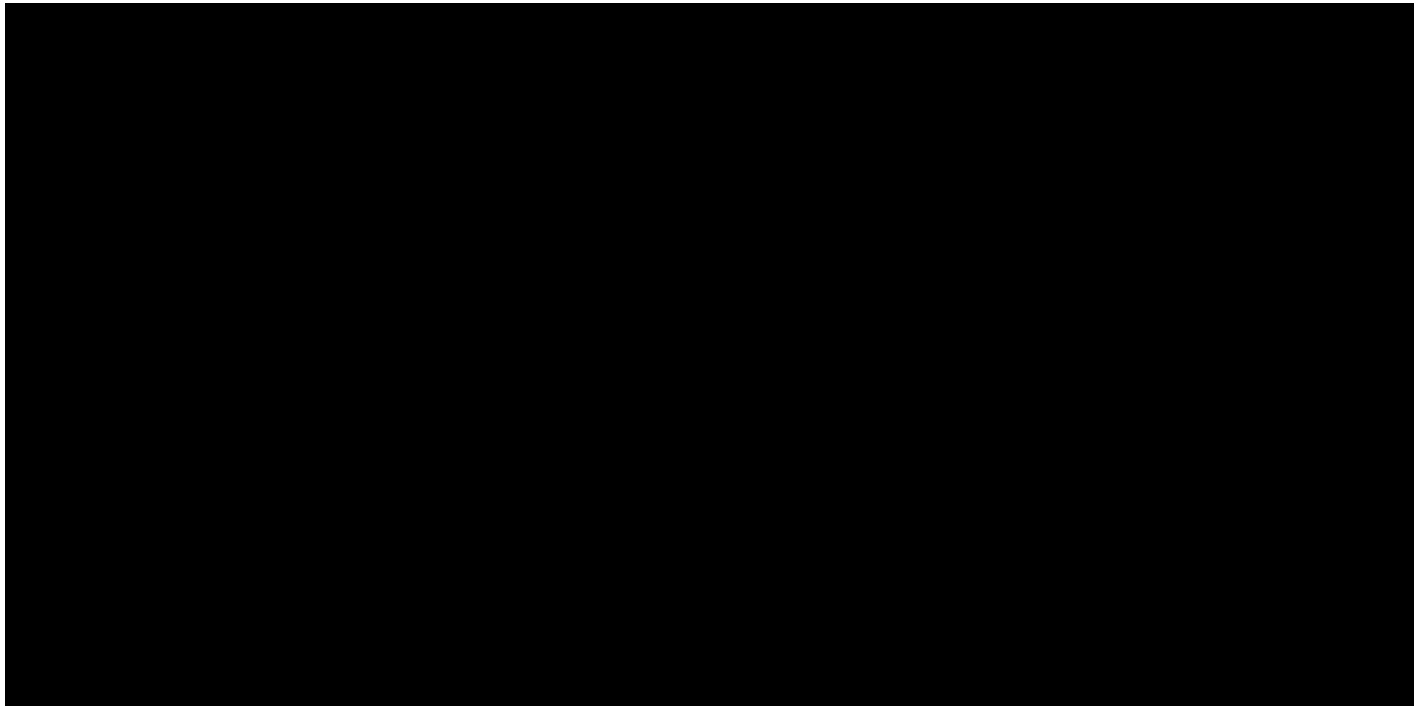
7. Emulador

Click en el link para ir a:

[Emulador Básico](#)

[Emulador Avanzado](#)

Video de Ejemplo - AVD



8. Links - pdfs- java

Algunos links que puede ser de utilidad...(abrir un pdf)

links a webs con pdfs java

libro de java donde pueden ver los conceptos de POO, ademas de las interfaces graficas de usuario - Esta obra de 'Oscar Belmonte et al. está bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 Unported

Libro: Fundamentos de la programación: java - Miguel Toro Bonilla - Esta obra se distribuye con la licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0)

9. Introduccion java

Leer el primer capitulo del libro: [Desarrollo de proyectos informáticos con tecnologia Java](#)

Realice las siguientes actividades

Resuelva el siguiente cuestionario

Material extra: [Lenguajes compilados e interpretados](#)

9.1. ByteCode

Introducción

Un compilador es un programa que traduce código escrito en un lenguaje de programación de alto nivel, (llamado código fuente) a otro lenguaje conocido como objeto, (bajo nivel)

Los interpretes solo realizan la traducción del código a medida que sea necesario, instrucción por instrucción en general no guardan el resultado de dicha traducción.

De acuerdo a esto, notamos claramente que un interprete es diferente de un compilador...

- Ya que interpretar es realizar la ejecución del código sobre la marcha línea por línea...
- Mientras que compilar, es el ensamblaje del código en un solo archivo traduciendo un programa desde su descripción en un lenguaje de programación al código de maquina del sistema subyacente.

Ambos métodos son igual de importantes, aunque los lenguajes interpretados están presente en la mayoría de las actividades que realizamos en Internet, por ejemplo los navegadores que interpretan código javascript.

Ambos métodos tienen sus ventajas y sus desventajas

Compilados

Ventajas

- Están preparados para ejecutarse y distribuirse instantáneamente a prácticamente cualquier maquina y por esta razón, debido a que son compilados son usualmente mas rapidos.
- El código fuente es inaccesible a lo que a veces se le considera una desventaja, y a veces se le considera una ventaja

Desventajas:

- No son programas multiplataformas, es decir, solo va a correr en el sistema en donde se compilo...el sistema operativo
- Son poco flexibles a la hora de querer hacer revisiones o arreglos en el código
- Si se necesitan mejoras o actualizaciones del código se recomienda instalarlos nuevamente, es decir, compilar de nuevo para generar un nuevo ejecutable.

Interpretados

Ventajas:

- Son multiplataforma, cada plataforma añade su propio interprete
- Son mas sencillos de testear y comprobar, cada modificación no requiere de una nueva compilación

Desventajas

- Se requiere un interprete,
- Son mas lentos de ejecutar
- Su código fuente es público

BYTE CODE

Ahora bien tomando en cuenta de que hay cosas buenas en los lenguajes compilados y los interpretados. Nos queda mencionar una tercera forma de ejecutar todo esto es una aproximación intermedia que tiene un poco de ambos.

En el lugar del modelo compilado donde todo el trabajo se hace por adelantado pero que puede ser mas inflexible o el modelo interpretado donde todo el trabajo se realiza en el extremo receptor, pero puede ser algo mas lento.

Se puede hacer algo que este justo en medio entre estos dos procedimientos , es decir, lo que haríamos es compilar parte del código en lo que se llama código intermedio:

- Este sería un archivo que está en medio camino entre el código del lenguaje de programación y el código de máquina.
- Este archivo se puede distribuir, y cada usuario lo que haría es ejecutarlo y compilarlo para su plataforma con la tecnología llamada **just in time** a este lenguaje también se le conoce a veces con el nombre de byte code.
- Bytecode ofrece la posibilidad de garantizar el rendimiento de un programa en todas las plataformas.
- Esto sirve como código intermedio que interpreta los comandos del código fuente y los traduce al lenguaje de destino requerido para el hardware respectivo.

Conclusión

- Así que usando este lenguaje tenemos una mezcla entre los dos, tenemos la rapidez de los lenguajes compilados y tenemos la flexibilidad de los lenguajes interpretados.
- De esta forma por ejemplo los lenguajes de programación C , C++ son lenguajes típicamente compilados, necesariamente necesitan de un compilador, este compilador se puede descargar de forma gratuita y en otras ocasiones viene implementado en aplicaciones de entorno de desarrollos integrados también conocidos como IDEs.
- Luego tenemos lenguajes como php y javascript que funcionan como lenguaje interpretados y que usualmente usamos a diario al utilizar un navegador web.
- Y hay también lenguajes como java, C# Visual basic .net o python que usa este enfoque intermedio de ser compilado e interpretado.

9.2. Qué es Java

Java



Características:

- Es un lenguaje de programación basado en C++
- **Es multiplataforma:** los programas compilados pueden correr bajo distintos sistemas operativos y distintas máquinas, sin la necesidad de recompilarse.
- Es un lenguaje compilado.
- Soporta gestión automática de memoria
- Soporte programación concurrente en forma nativa (aplicaciones multihilo)
- Soporta Herencia simple
- No soporta herencia múltiple
- No soporta punteros ni sentencias de salto goto.

Se puede decir que Java es: un lenguaje de programación ejecutado sobre una plataforma de desarrollo con su propio entorno de ejecución, a su vez posee un conjunto de librerías para desarrollo de programas sofisticados. Las librerías para desarrollo se denominan JAVA Application Programming Interface (Java API).

Compilador:

El compilador de JAVA (javac) genera un archivo en código intermedio llamado bytecode que es ejecutado por la JAVA Virtual Machine (JVM) instalada en cada plataforma.

JAVA Virtual Machine (JVM)

La JVM utiliza la tecnología JIT (just in time) para traducir el archivo bytecode a un archivo ejecutable en la plataforma donde se está realizando el proceso.

