

Unidad 1

Introducción a Android.	2
Google Play Store	2
Lenguajes Oficiales y Alternativos.	2
Android Studio.	2
Creación de una aplicación Android	3
SDK Manager	4
SDK Platforms	4
SDK Tools	5
Emuladores.	5
AVD	5
Genymotion	5
Dispositivo físico.	5
Ejecución de aplicación	6
Estructura de Proyecto.	6
AndroidManifest	6
Java	6
Res	7
Gradle	7
Herramientas generales Android Studio	7
Logcat	7
Componentes Principales de Android.	7
Activity	7
Servicios	8
Tipos de servicios	8
Content Provider	8
Broadcast Receiver	8

Ciclo de Vida de un Activity.	8
Al Iniciar o lanzar el Activity	9
Mientras el Activity se está ejecutando	10
Manejo de Activitys.	10
Estructura de la Activity	10
Método finish()	10
Método onBackPressed()	10
Navegación entre Activitys.	11
Intent	11
Tipos de Intent	11
Enviar parámetros hacia una Activity	11
Obtener parámetros enviados desde una Activity	12
Primera aplicación	12
Diseño de pantalla	12
Maquetado	12
ID	12
Componentes base	12
LinearLayout	12
TextView	12
Button	13
Java	13
Vinculación de vista con objeto	13
Capturar click de una vista	14
Práctica 1. A	14
Práctica 1. B	15
Práctica 1. C	15

Introducción a Android.

Android es un sistema operativo desarrollado por Google, basado en una versión modificada del kernel de Linux, cuyo código es abierto.

Google Play Store

Es la tienda de aplicaciones oficial para el sistema operativo Android, permitiendo a los usuarios navegar y descargar aplicaciones. Además, ofrece a los usuarios secciones de descarga de libros, música y más.

Para acceder a esta plataforma existe una aplicación llamada "Play Store", anteriormente llamada Market, la cual se encuentra preinstalada en todos los dispositivos Android. Google garantiza que su tienda es segura. Ellos mismos se encargan de que las aplicaciones cumplan los términos y condiciones para evitar que aplicaciones corruptas no tengan comportamientos que no cumplan con su ética. Infringir con esta política conduce a la baja directa de la aplicación de su plataforma

Lenguajes Oficiales y Alternativos.

Por lenguajes oficiales se hace referencia a los lenguajes que son respaldados por Google para el desarrollo de aplicaciones Android. Utilizarlos brindan la ventaja de hacer uso de las últimas librerías y funcionalidades desarrolladas. Por un lado, JAVA que es un lenguaje de programación orientada a objetos y primer lenguaje oficial en incorporarse en el desarrollo de Android. Y por otro lado, Kotlin que es un lenguaje con orientación a la programación funcional. Fue anunciado y desplegado para su utilización en el año 2017.

Los lenguajes alternativos son aquellos que no están respaldados por Google, entre los que se encuentran Ionic, Phonegap y React Native.

Android Studio.

Es el IDE (entorno de desarrollo integrado) oficial para la plataforma Android. Fue anunciado en mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS, GNU/Linux y Chrome OS.

Se puede descargar desde [aquí](#). Y hay que tener en cuenta los requisitos

Windows

- 64-bit Microsoft® Windows® 8/10
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a [Windows Hypervisor](#)
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

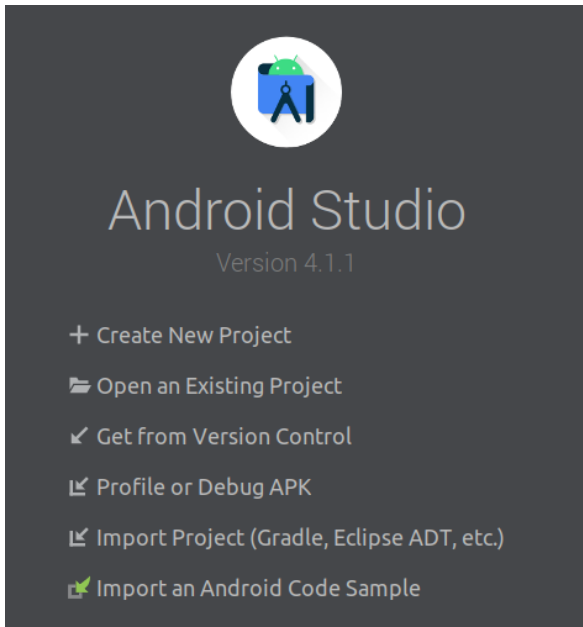
Linux

- Any 64-bit Linux distribution that supports Gnome, KDE, or Unity DE; GNU C Library (glibc) 2.31 or later.
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD processor with support for AMD Virtualization (AMD-V) and SSE3
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

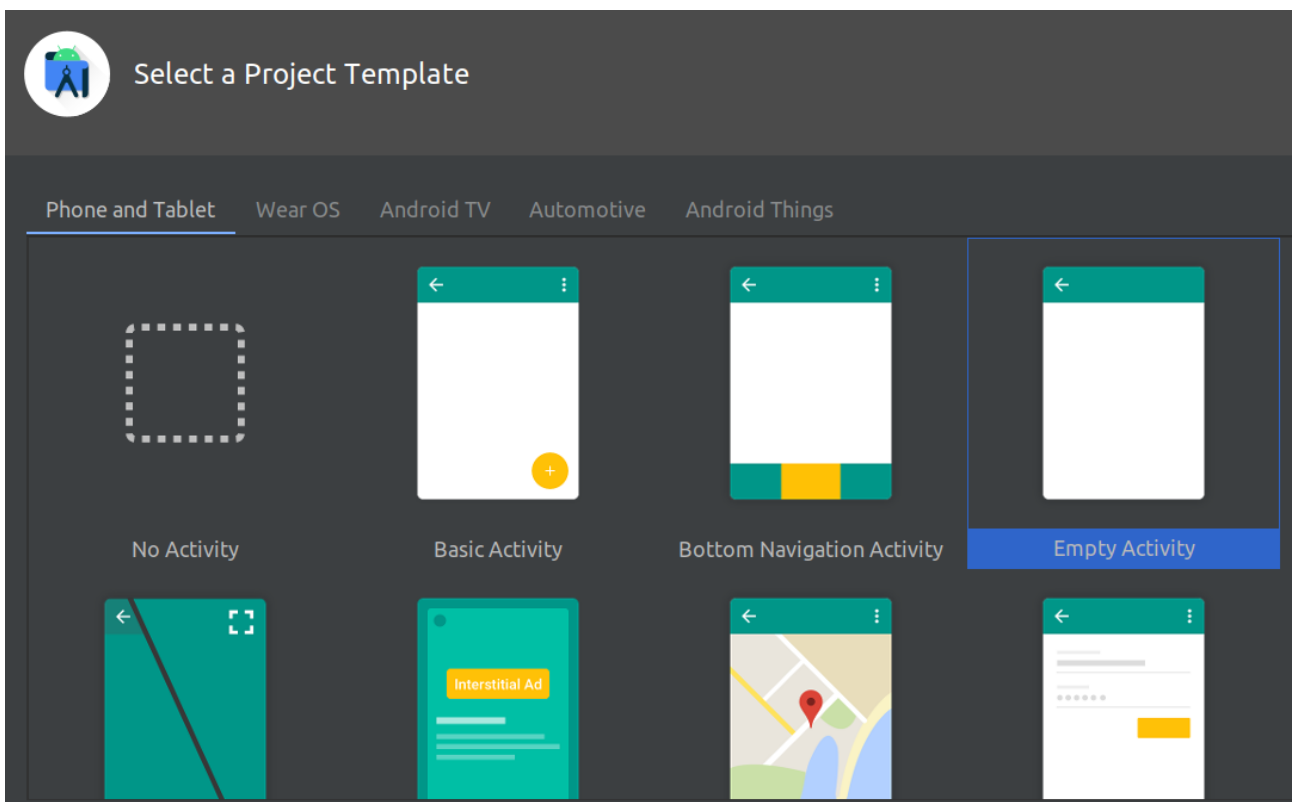
Creación de una aplicación Android

Una vez instalado el Android Studio ya se podrá crear la primera aplicación. Se ejecutará el “.exe” para abrir el IDE y se seguirán los siguientes pasos:

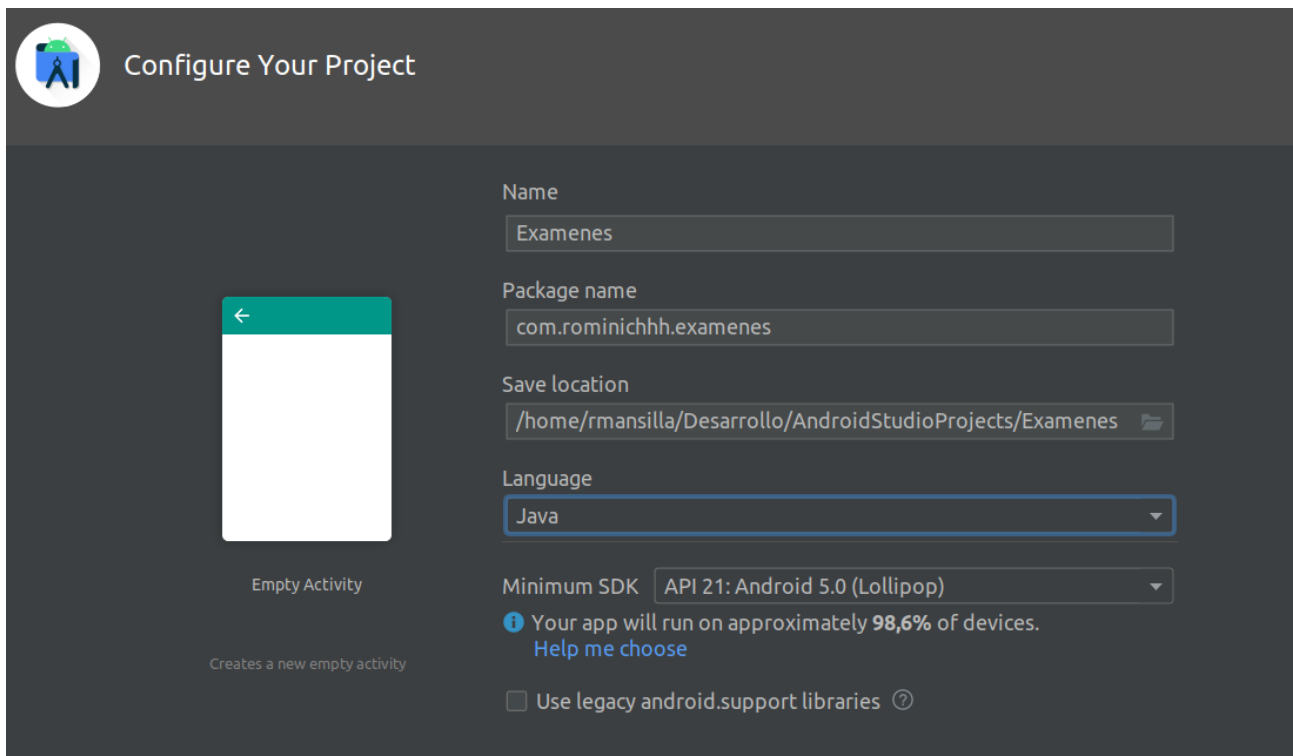
1. Click sobre “Create New Project”.



2. En esta sección se puede definir de qué tipo será el proyecto. Por defecto se deja seleccionado Phone and Tablet. Luego se selecciona Empty Activity y se hace click en “Next”.



3. Se define el nombre de la aplicación. También se puede elegir el nombre del paquete, la ubicación del proyecto, el lenguaje a utilizar. A su vez se podrá definir cuál será la versión de Android mínima que el proyecto soportará. Luego se presionará Finish.



Configure Your Project

Name
Exámenes

Package name
com.rominichhh.exámenes

Save location
/home/rmansilla/Desarrollo/AndroidStudioProjects/Exámenes

Language
Java

Minimum SDK API 21: Android 5.0 (Lollipop)

❗ Your app will run on approximately **98,6%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries ?

Empty Activity
Creates a new empty activity

En esta instancia se estarán descargando y compilando las dependencias/librerías correspondientes.

SDK Manager

El SDK Manager permite tener las herramientas para desarrollar de manera actualizada. Es muy importante tenerlas al día, ya que van a permitir tener las últimas herramientas de desarrollo apuntando hacia las últimas versiones de Android. Podemos acceder a las opciones en Tools -> SDK Manager. A continuación se ven 2 imágenes que indican cuáles son las herramientas obligatorias que deben ser instaladas.

Nota: las versiones de Android irán incrementándose, en esencia hay que tener en cuenta los puntos marcados.

SDK Platforms

Name	API Level	Re...	Status
Android 11.0 (R)			
<input checked="" type="checkbox"/> Android SDK Platform 30	30	3	Installed
<input checked="" type="checkbox"/> Sources for Android 30	30	1	Installed
<input type="checkbox"/> Automotive with Play Store Intel x86 Atom_64 System Image	30	2	Not instal...
<input type="checkbox"/> Android TV Intel x86 Atom System Image	30	3	Not instal...
<input type="checkbox"/> Google APIs ARM 64 v8a System Image	30	11	Not instal...
<input checked="" type="checkbox"/> Google APIs Intel x86 Atom System Image	30	10	Installed
<input type="checkbox"/> Google APIs Intel x86 Atom_64 System Image	30	11	Not instal...
<input type="checkbox"/> Google Play ARM 64 v8a System Image	30	10	Not instal...
<input type="checkbox"/> Google Play Intel x86 Atom System Image	30	9	Not instal...
<input type="checkbox"/> Google Play Intel x86 Atom_64 System Image	30	10	Not instal...
<input type="checkbox"/> Google APIs ATD ARM 64 v8a System Image	30	1	Not instal...
<input type="checkbox"/> Google APIs ATD Intel x86 Atom System Image	30	1	Not instal...
Android 10.0 (Q)			

☒ Hide Obsolete Packages ☒ Show Package Details

SDK Tools

Name	Version	Status
<input checked="" type="checkbox"/> Android SDK Build-Tools 33		Installed
<input type="checkbox"/> NDK (Side by side)		Not Installed
<input type="checkbox"/> Android SDK Command-line Tools (latest)		Not Installed
<input type="checkbox"/> CMake		Not Installed
<input type="checkbox"/> Android Auto API Simulators	1	Not installed
<input type="checkbox"/> Android Auto Desktop Head Unit Emulator	2.0	Not installed
<input checked="" type="checkbox"/> Android Emulator	31.3.9	Installed
<input checked="" type="checkbox"/> Android SDK Platform-Tools	33.0.2	Installed
<input type="checkbox"/> Google Play APK Expansion library	1	Not installed
<input type="checkbox"/> Google Play Instant Development SDK	1.9.0	Not installed
<input type="checkbox"/> Google Play Licensing Library	1	Not installed
<input checked="" type="checkbox"/> Google Play services	49	Installed
<input type="checkbox"/> Google Web Driver	2	Not installed
<input checked="" type="checkbox"/> Layout Inspector image server for API 29-30	6	Installed
<input checked="" type="checkbox"/> Layout Inspector image server for API 31 and T	1	Installed
<input checked="" type="checkbox"/> Layout Inspector image server for API S	3	Installed

☒ Hide Obsolete Packages ☐ Show Package Details

Emuladores.

Un emulador es aquel que permitirá simular la ejecución de la aplicación en un dispositivo similar a los reales. Para crear un emulador, se utilizarán las herramientas descritas debajo, en donde se podrá seleccionar la versión del sistema operativo Android, la RAM que poseerá, el procesador, y muchas más cosas.

AVD

Android Virtual Devices (AVD) es una herramienta que proporciona el Android Studio para gestionar los emuladores (crearlos, editarlos o borrarlos). Luego de crear un dispositivo virtual, se deberá activar la virtualización en la computadora (tener en cuenta que también se deberán haber descargado los módulos correspondientes en el SDK Manager). Este emulador es recomendable para los que poseen en su computadora el procesador Intel y AMD de la línea Ryzen (no Fx).

Genymotion

Es un software que posee una amplia cantidad de emuladores precargados para seleccionar y utilizar. Para la emulación del sistema operativo utiliza Virtualbox. Al momento de la descarga del Genymotion, ofrece también la opción de descargar Virtualbox en conjunto. Tener en cuenta que para utilizarlo, primero se debe registrar en la página. Este tipo de software es recomendado para los usuarios que poseen el procesador AMD y/o quieren agrandar su variedad de dispositivos a emular.

Dispositivo físico.

Para poder utilizar un celular/tablet física para ejecutar la aplicación creada, se deberán seguir los siguientes pasos (puede variar algún paso dependiendo el sistema operativo y la versión de Android):

- Ir a la configuración general del dispositivo.
- Ir a la sección del sistema.
- Ir a la sección “acerca del teléfono”
- Presionar repetitivamente sobre el ítem “Número de compilación”.
- Se debe mostrar “Ya eres programador” o un mensaje similar.
- Ir hacia atrás y buscar el ítem “Desarrollador” ó “Programador”.

- Buscar la opción “Depuración por USB” y seleccionarla.

Una vez realizados estos pasos, al conectar el dispositivo a la computadora, el Android Studio lo detectará y permitirá la ejecución de nuestra aplicación.

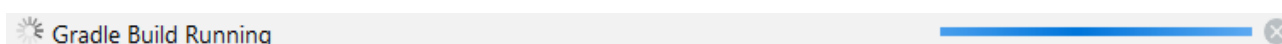
Ejecución de aplicación

Una vez que se encuentre el entorno configurado, se podrá ejecutar la aplicación creada. Para ello se deberá seleccionar el dispositivo en que se ejecutará, y presionar sobre el botón de play ubicado debajo de la barra de herramientas del IDE, ó utilizar el atajo “Shift + F10” en Windows:



Una vez realizado, comenzará la compilación del proyecto y en la barra inferior del IDE del lado derecho se verá una barra de progreso, la cual se deberá esperar a que finalice para poder empezar a visualizar y utilizar la aplicación.

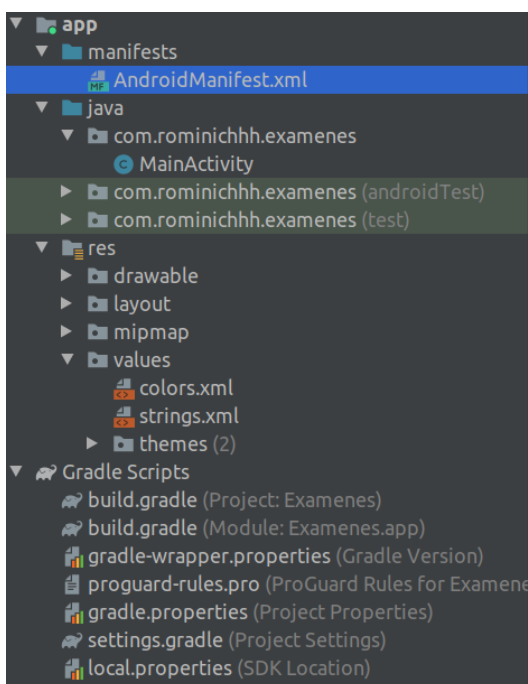
Barra de progreso:



Nota: si es la primera vez que se conecta un dispositivo físico, en el mismo, será necesario aceptar la conexión al IDE.

Estructura de Proyecto.

Un proyecto Android está conformado de la siguiente manera:



AndroidManifest

Es un archivo XML en donde se define un conjunto de configuraciones y definiciones de la aplicación, como por ejemplo el nombre, el ícono, el estilo, la primera pantalla que se va a mostrar al iniciar la aplicación, los permisos que requerirá (ubicación/GPS, internet, etc), y las Actividades. Para más información se puede ingresar [aquí](#).

Java

Carpeta que contiene los distintos paquetes contenedores de los archivos Java.

Res

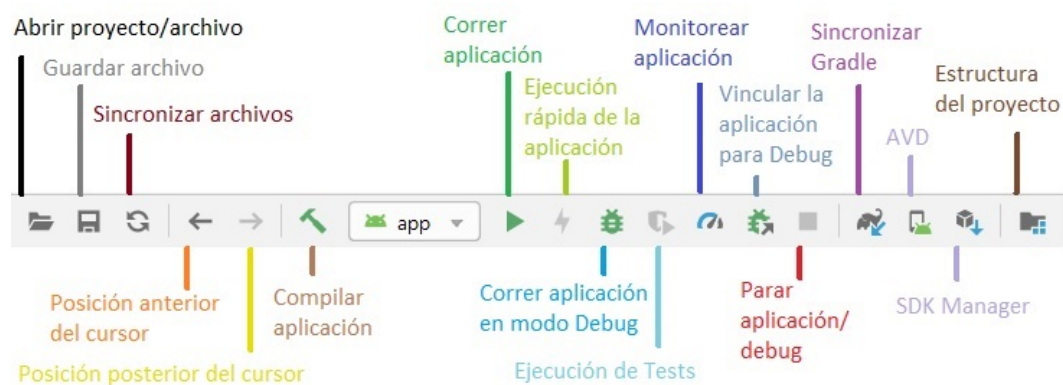
Carpeta contenedora de los recursos que utilizará la aplicación:

- Drawable: utilizado principalmente para el almacenamiento de imágenes.
- Layout: contiene los layouts de la aplicación.
- Mipmap: el ícono de la aplicación.
- Values: subcarpeta que contiene distintos archivos en XML. Entre ellos están:
 - Colors: colores que utilizará la aplicación
 - Strings: cadenas de texto que utilizará la aplicación
 - Styles: los estilos que utilizará la aplicación

Gradle

Es la herramienta que permite la construcción y compilación de la aplicación. Es el encargado de importar las dependencias/librerías de nuestro proyecto. Entre los archivos que se utilizarán está el “build.gradle (Module: app)” en donde entre otras cosas se definirá la versión de SDK con la cual va a compilar el proyecto, la versión mínima de S.O. que se soportará, el número de versión de la app, y las librerías que se utilizarán.

Herramientas generales Android Studio



Logcat

Es una herramienta que permite seleccionar un dispositivo y ver qué es lo que está ocurriendo en vivo. Al ejecutar la aplicación, automáticamente se asocia la información que se muestra con nuestra app.

Al romper la aplicación, en el logcat se escribe un mensaje de error, en donde se podrá analizar y verificar la causa de este crash para poder solucionarlo.

Para acceder al Logcat, se lo deberá buscar en la barra inferior del IDE con tal nombre. Más info [acá](#).

Componentes Principales de Android.

Son los elementos a tener en cuenta al momento de realizar una aplicación en Android. Estos componentes tienen algo en común y es que se definen en el archivo de manifest de la aplicación (AndroidManifest.xml).

Activity

Una Activity representa una pantalla con una interfaz gráfica. Para crear una Activity, se puede utilizar el Wizard que provee el IDE presionando click derecho sobre “app” (en la estructura del proyecto), “new”, “Activity”, “Empty Activity”.

Mayoritariamente las Activities están compuestas por un archivo Java y un archivo XML. Es indispensable que la Activity esté definida en el AndroidManifest, de lo contrario al momento de correr la aplicación, se romperá. Si para crearla se utiliza el Wizard, este automáticamente creará los 2 archivos (Java y XML) junto con la definición en el AndroidManifest.

Una clase es una Activity si esta misma extiende a "Activity" ó "AppCompatActivity".

Definición en el manifest:

```
<activity android:name=".MainActivity"/>
```

Definición de activity indicando que sea el launcher (la primera en mostrarse al iniciar la aplicación):

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Servicios

Un servicio es un componente que permite realizar largas operaciones en background sin necesidad de interacción con el usuario. Los servicios se pueden utilizar para realizar descargas, consultar periódicamente por información, procesamiento de datos, reproducir música, etc.

Tipos de servicios

Existen tres tipos de servicios:

- **Foreground:** realiza operaciones que son perceptibles para el usuario. Estos servicios continúan ejecutándose sin que el usuario esté interactuando con la aplicación. Se debe informar mediante una notificación el estado del servicio. Por ejemplo, cuando se está reproduciendo música.
- **Background:** realiza operaciones que no son perceptibles para el usuario. Sólomente se pueden ejecutar cuando el usuario se encuentra dentro de la aplicación
- **Bound:** se vincula el servicio a un componente. Ofrece una interfaz cliente-servidor que permite a los componentes interactuar con el servicio, enviar requests, recibir resultados, etc. Este tipo de servicios sólo se ejecutan si hay un componente vinculado a él, por lo que si este componente se destruye, el servicio también lo hará.

Content Provider

El Content Provider es un componente que nos permite obtener y compartir información desde y hacia otras aplicaciones. Por ejemplo, si una aplicación necesita leer los contactos del dispositivo, se necesitará utilizar un Content Provider para obtener la información deseada.

Broadcast Receiver

El Broadcast Receiver es un componente que permite el registro de eventos tanto de la aplicación como del sistema. Por ejemplo, se puede realizar el registro del evento de activación/desactivación del modo avión, o también cuando termina de bootear el dispositivo. Una vez que se detecta tal evento, se puede realizar la acción que se necesite.

Más información: [Componentes de Android](#)

Ciclo de Vida de un Activity.

Son las transiciones de estados por las que pasa la Activity. Al iniciarla, el primer método que se ejecuta es el onCreate(). Se pueden sobrescribir los métodos del ciclo de vida de la Activity (ver imagen), para realizar acciones deseadas en momentos específicos. Por ejemplo, si se desea saber si el dispositivo entra en modo avión cuando se encuentra en una pantalla, se definirá la registración del Broadcast Receiver en el método onResume(), y en el método

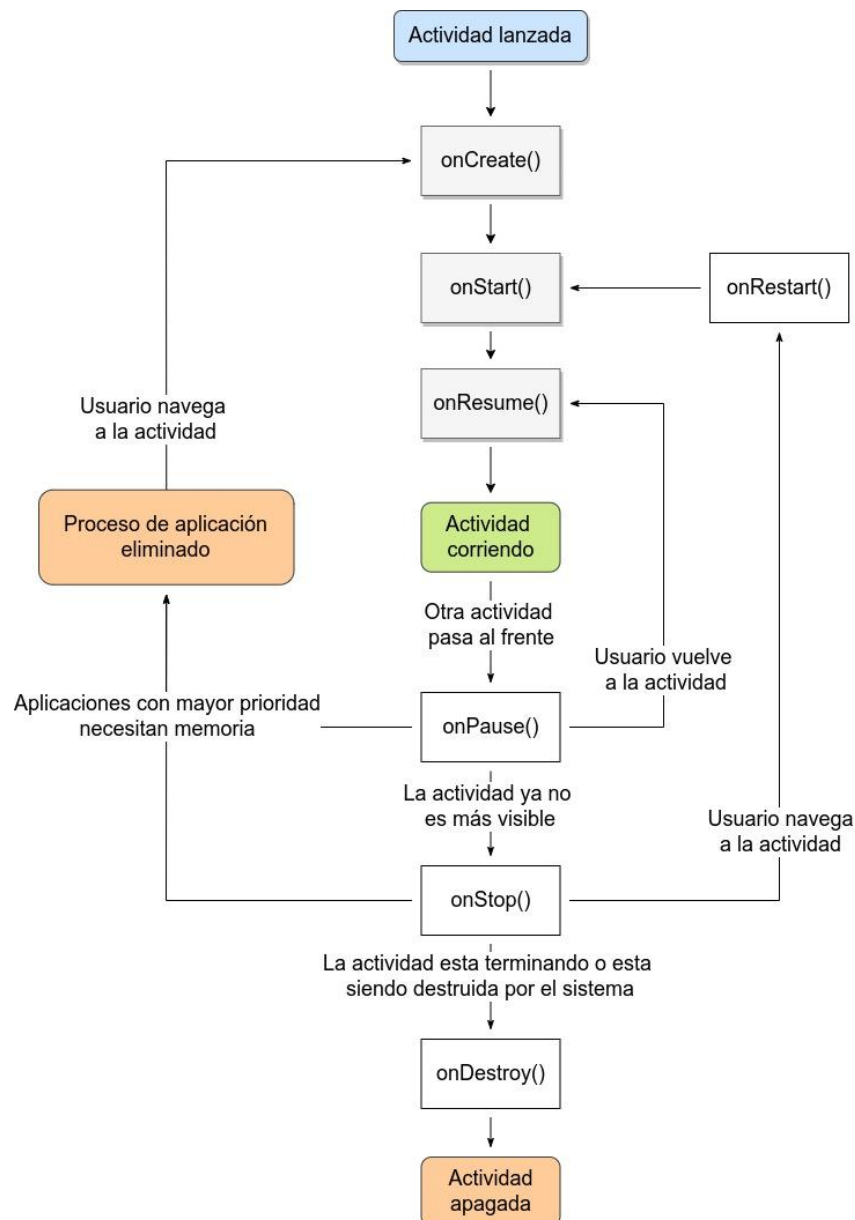
`onPause()` se anulará la registraci3n, ya que no es necesario saber el estado si se sale de la Activity o la aplicaci3n. Al volver a visualizar la Activity, se registrará nuevamente el evento ya que se encontraría dentro del método `onResume()`.

Las razones por las cuales su correcta implementaci3n es muy importante son:

- Evitar posibles crashes
- Reducir la consumici3n de recursos del sistema cuando el usuario no est1 activo
- Manipular correctamente las transiciones de rotaci3n del dispositivo.

A continuaci3n se puede ver un diagrama de c3mo se van ejecutando los m3todos a partir del inicio de una Activity.

M1s informaci3n: [Ciclo de vida Activity](#)



Al Iniciar o lanzar el Activity

`onCreate()`

Como su nombre lo dice es cuando se crea el Activity por primera vez y donde se deben crear vistas, botones, enlace de datos, etc., todas las configuraciones est1ticas que se van a utilizar.

onStart()

Viene después de crear el Activity y es el método que la inicia y hace que se vuelva visible para el usuario.

onResume()

Este método se ejecuta justo antes de que el usuario interactúe con el Activity ya que es el que lo pone en la parte superior de la pila de actividades.

Mientras el Activity se está ejecutando

onPause()

Este método se usa para confirmar los cambios sin guardar, detener animaciones y otros procesos que podrían estar consumiendo CPU. Es llamado por el Sistema Operativo cuando el usuario está a punto de llamar a otro Activity o ponerlo en segundo plano.

onStop()

Cuando el Activity ya no es visible para el usuario ya sea porque se abrió uno nuevo o se reanudó otro que estaba abierto y se puso por delante de este.

onRestart()

Cuando el Activity se pone onStop() pero se quiere interactuar nuevamente con él es cuando se llama a este método siguiendo el método onStart().

onDestroy()

De la misma manera que onRestart(), cuando el Activity se pone en onStop(), pero ahora el Activity ya no es requerido y se va a cerrar por completo se llama a este método para destruir el Activity.

Manejo de Activitys.

Estructura de la Activity

Las Activities se manejan con una estructura del tipo pila (LIFO – last in first out). Esto quiere decir que si se inician dos Activities y se presiona "back", la primera que se destruirá es la última que se agregó.

Para manipular el correcto ordenamiento de Activities dentro del stack, es decir, la pila, se hace uso de dos métodos:

- finish()
- onBackPressed()

Método finish()

Si se necesita cerrar una Activity debido a que el usuario realiza determinada acción, se puede hacer uso del método "finish()". El mismo puede ser utilizado por cualquier Activity. La Activity a destruir será la que llame a este método, y esta va a comenzar a disparar los pasos del ciclo de vida (onPause, onStop, onDestroy).

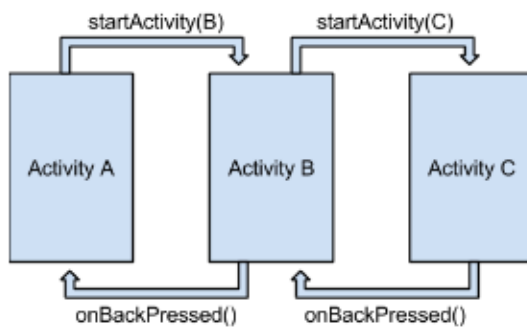
Ej: si se posee la Activity "MainActivity", se genera un Intent, se ejecuta el método "startActivity(intent)" y luego se ejecuta el método "finish()", la Activity que se cerrará es la "MainActivity".

Método onBackPressed()

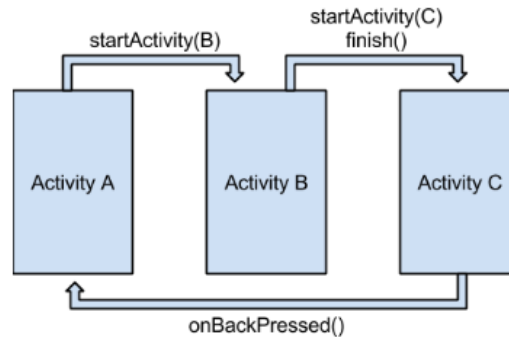
Android brinda la posibilidad de manipular el comportamiento del back. Para ello, dentro de la Activity se podrá sobrescribir el método "onBackPressed()" para modificar su función a lo que se necesite.

Ejemplo: en el caso de que se esté editando algún formulario, al presionar el back, se le puede mostrar un popup preguntándole al usuario si está seguro de que quiere abandonar la Activity sin tener cambios guardados. O también si es la última Activity de la pila, si quiere abandonar la aplicación.

Ejemplo gráfico onBackPressed()



Ejemplo gráfico combinando finish() y onBackPressed()



Navegación entre Activities.

Intent

Un Intent es una clase que permite realizar distintas acciones, como iniciar una Activity o un servicio entre otras cosas.

Tipos de Intent

Explícito: es aquél que permite iniciar un componente específico que fue definido.

Por ejemplo, para iniciar una Activity, al crear el objeto Intent, se debe proveer el nombre de la clase destino.

Ejemplo:

```
Intent intent = new Intent( packageContext: MainActivity.this, HomeActivity.class);
startActivity(intent);
```

Implícito: es aquel al que se le indica la realización de una acción específica. Al instanciar el Intent, se le debe definir la acción de lo que se quiera realizar.

Por ejemplo, se podría abrir la aplicación de Mensajes con un contenido escrito por defecto, ó también se podría abrir el Play Store apuntando hacia alguna app específica.

Ejemplo:

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.setData(Uri.parse("sms:"));
startActivity(sendIntent);
```

Enviar parámetros hacia una Activity

Muchas veces cuando se inicia una Activity se necesita pasarle datos que se encuentran en la pantalla actual. Para esto, una vez instanciado el objeto Intent, se deberá agregarle los parámetros a través del método .putExtra(clave, valor). A este método se le pasan dos parámetros: clave (es la referencia donde se va a guardar el contenido) y valor (es el contenido a almacenar, puede ser un entero, texto, decimal, etc).

Ejemplo, si se obtuvieron datos de una persona que inició sesión, el código podría ser:

```
Intent intent = new Intent( packageContext: MainActivity.this, HomeActivity.class);
intent.putExtra( name: "USUARIO", usuario);
intent.putExtra( name: "CONTRASEÑA", contraseña);
startActivity(intent);
```

Obtener parámetros enviados desde una Activity

Para obtener los parámetros se debe obtener el Intent que se envió mediante el método `getIntent()`. A partir de ese objeto, se obtendrá otro que llamado "Bundle" mediante el método `getExtras()`, el cual funciona como una lista de diccionarios, es decir, clave/valor. Hay que tener en cuenta que este objeto recibido puede ser nulo, por lo que se recomienda siempre hacer la validación correspondiente. Una vez obtenido este Bundle, podemos pedirle que a partir de la clave que nosotros le definimos en el "putExtra" obtengamos el valor requerido.

Ejemplo:

```
Bundle bundle = getIntent().getExtras();
if (bundle != null) {
    usuario = bundle.getString( key: "USUARIO", defaultValue: "");
    contraseña = bundle.getString( key: "CONTRASEÑA", defaultValue: "");
}
```

Nota: se recomienda que el campo clave/key sea definido como una constante para evitar errores tipográficos y tener este valor en un lugar centralizado.

Más información: [Intent](#)

Primera aplicación

Diseño de pantalla

Maquetado

Tanto las pantallas como los componentes de Android se maquetan a partir de la creación de un archivo xml. Existen dos maneras distintas que permitirán seleccionar y posicionar los componentes en donde se desee:

- Drag & drop: herramienta que brinda el IDE. Para utilizarla se deberá seleccionar la opción de modo Design o Split.
- Código XML: se debe visualizar el archivo del tipo layout en modo Code o Split.

El modo Split combina la posibilidad de utilizar la funcionalidad de drag & drop pudiendo visualizar y modificar el código XML que se genera.

ID

Es un atributo que poseen todas las vistas que permitirán que estas puedan ser identificadas y vinculadas a objetos Java para que luego puedan ser manipuladas. Para agregar esta propiedad, hay que posicionarse dentro del tag de la vista y definirlo de la siguiente manera: `android:id="@+id/idAUtilizar"`. Otra manera es a partir de la vista de Diseño, en donde se deberá agregar/modificar el valor del id que se encuentra en la pestaña de Atributos.

Componentes base

Existen VISTAS que se agregan en la pantalla para representar componentes. Para poder agregar una y visualizarla en la pantalla se utiliza primero una capa que es llamada "contenedor".

LinearLayout

Es un contenedor de vistas/componentes. Esto quiere decir que si se desea agregar un texto, un botón u otro componente a la pantalla, se lo deberá agregar dentro del LinearLayout. Estas vistas se van a disponer de manera vertical u horizontal, dependiendo el atributo "orientation" del LinearLayout definido. Para modificarlo se lo puede hacer desde la vista de Diseño en donde se deberá buscar esa propiedad en los atributos.

TextView

Es una vista cuya responsabilidad es mostrar texto. Para modificarlo se puede hacer tanto desde el XML como desde el código. El TextView únicamente permite que se ingresen valores del tipo texto, por lo tanto si se quiere ingresar un número se lo deberá convertir a un texto (String). Para poder definir el valor del texto se puede utilizar un enfoque

mediante el XML (maquetado) o también mediante código java. Mediante XML se utiliza la propiedad text que puede ser definida desde los atributos en la vista de diseño.

```
<TextView
    android:id="@+id/txtNumero"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@@" />
```

Mediante código, una vez realizado el findViewById, se podrán utilizar los siguientes métodos:

1. setText("Nuevo texto")
2. getText()

La primera función permite modificar el texto a partir de un String pasado por parámetro.

La segunda función devuelve un String con el valor del texto que contiene el TextView.

En Java para convertir un número a un texto se utiliza el método toString() a partir de una variable del tipo Integer, Float, o Double (no int, float).

También existe la posibilidad de utilizar String.valueOf() para los casos de los tipos int y float.

Ejemplo 1:

```
Integer numero1 = 1;
String numeroUno = numero1.toString();
```

Ejemplo 2:

```
int numero2 = 2;
String numeroDos = String.valueOf(numero2);
```

Para hacer la operación inversa, es decir, convertir un entero a un objeto del tipo texto se utiliza Integer.parseInt().

```
String numeroTres = "3";
int numero3 = Integer.parseInt(numeroTres);
```

Button

Vista utilizada para capturar clicks sobre la misma.

Java

Vinculación de vista con objeto

Para manipular las vistas de la pantalla se las deberá vincular con los objetos en Java. Primero se los deberá declarar como variables de la clase. Los tipos a definir (las clases) de las vistas son intuitivas. Ej: definir un objeto del tipo botón se hace de la siguiente manera:

```
private Button btnSumar;
```

Para vincular este objeto con una vista, se deberá asignarle lo siguiente dentro del método "onCreate" (este método es el primero en ejecutarse al iniciar una activity):

```
btnSumar = findViewById(R.id.btnSumar);
```

De esta manera se le está diciendo al objeto btnSumar que busque la vista que contiene el id "btnSumar".

Nota: el nombre del objeto es indistinto al del id.

Capturar click de una vista

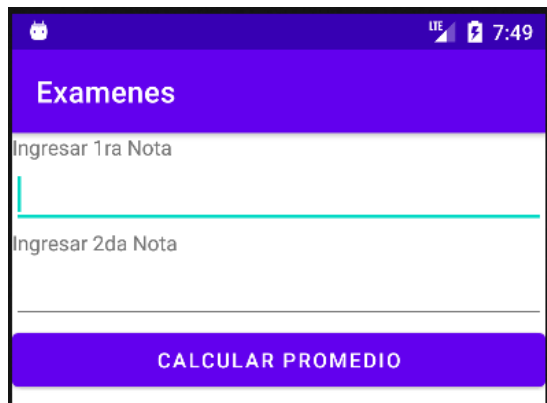
A la vista que se le desea asignar el evento del click (como a un botón), deberá llamar al método `setOnClickListener` haciendo referencia a la variable de la vista. Por parámetro se deberá implementar una interfaz que contiene el método `onClick()` en donde se definirán las acciones a realizar. Ejemplo:

```
btnSumar.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
  
    public void onClick(View v) {  
  
        Toast.makeText(MainActivity.this, "Hello World!", Toast.LENGTH_SHORT).show();  
  
    }  
  
});
```

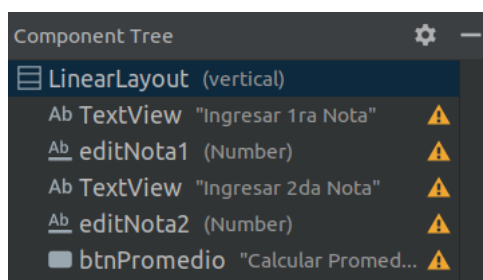
De esa manera, cuando se presiona el botón, se mostrará un Toast (un mensaje temporal en la parte inferior de la pantalla).

Práctica 1. A

En un proyecto llamado Exámenes maquetar la siguiente pantalla:



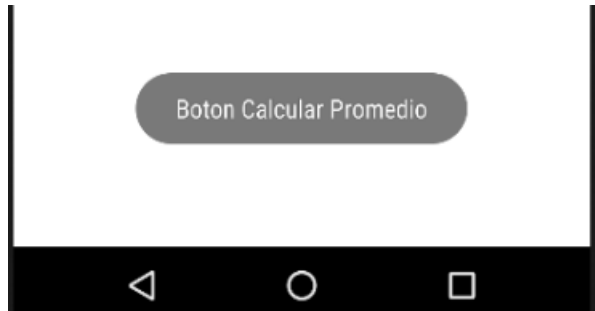
El component tree quedaría de la siguiente manera:



Se deberá agregar IDs para todas las vistas.

Práctica 1. B

El objetivo del mismo es hacer la vinculación de los componentes de la vista con los de la clase Java. A su vez, cuando se presione el botón **Calcular Promedio** se deberá mostrar un mensaje temporal (Toast).



Práctica 1. C

El objetivo del mismo es darle la funcionalidad de matemática. Al presionarlo se deberá obtener el valor ingresado en el texto1, sumarle el valor del texto2 y dividir por dos el resultado.

Nota: el valor inicial de ambos textos será 10.

Recuerden que siempre pueden encontrar más prácticas en la página oficial de Android. Pueden acceder haciendo click [acá](#).