ITESO, Universidad Jesuita de Guadalajara

Premio ADA BYRON A LA MUJER TECNÓLOGA

Taller de tecnología móvil

Android

Kotlin

Temario

Módulo 2

- Interfaz gráfica de usuario
- Navegación entre pantallas
- Material Design
- Programación avanzada
- Mensajes y alertas

Interfaz gráfica de usuario

Activity

Un ***activity*** es un componente clave de una aplicación para Android, la forma en que se inician y se crean las actividades es una parte fundamental del modelo de aplicación de la plataforma. Por lo general, esta ventana llena la pantalla, pero puede ser más pequeña y flotar sobre otras ventanas.

La mayoría de las aplicaciones contienen varias pantallas, lo cual significa que incluyen varias ***activities, e***n la cual una se especifica como la ***activity principal***, que es la primera pantalla que aparece cuando el usuario inicia la aplicación. Luego, cada una puede iniciar otras ***activities*** a fin de realizar diferentes acciones.

Fragment

Un ***fragment*** representa una parte reutilizable de la IU de una aplicación. Un ***fragment*** se compone de:

- Diseño
- Ciclo de vida
- Eventos de entrada

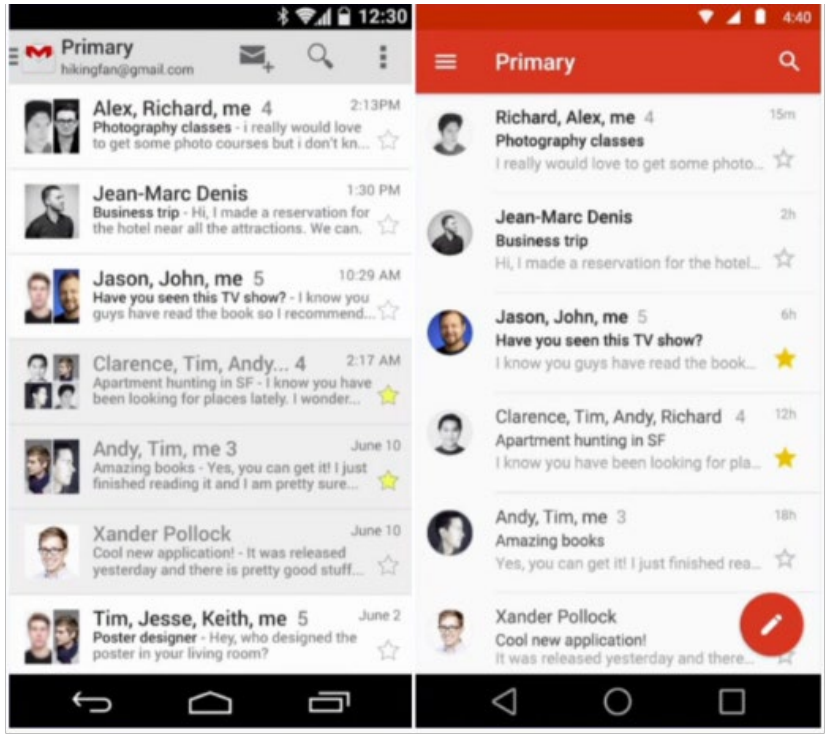
Importante: La jerarquía de vistas del ***fragment*** forma parte de la jerarquía de vistas del host o está *conectada* a ella.

Los fragmentos no pueden existir por sí solos, sino que deben estar *alojados* por una actividad u otro fragmento.

En esta figura podemos ver que contiene una barra de navegación inferior controlada por la actividad y una lista lineal controlada por el fragmento.

Interfaz de usuario

La interfaz de usuario de tu aplicación es todo aquello que el usuario puede ver y con lo que puede interactuar en ella. Android ofrece una variedad de componentes de ***IU*** previamente compilados, como objetos de diseño estructurados y controles de la ***IU*** que te permiten compilar la interfaz gráfica de usuario para tu aplicación. Además, también brinda otros módulos de ***IU*** para interfaces especiales, como diálogos, notificaciones y menús.



Layout

Los ***layout*** son un conjunto de contenedores en donde podemos colocar muchos elementos según el diseño de nuestra aplicación, por ejemplo dentro de un ***layout*** puedes colocar botones, imágenes, formulario, textos, etc. Todos estos deben tener un orden y armonía entre si dentro del diseño, para esto debemos haber elegido un ***layout*** primero para luego colocar dentro de este los elementos que necesitamos.

Constraint Layout

Este ***layout*** es útil para trabajar con Grandes grupos de elementos porque nos crea un orden jerárquico dinámico, podemos agregar elementos adicionales y darles restricciones para que cuando muevas la posición de la pantalla del móvil, este mantenga en su lugar los elementos de una vista, a continuación giramos la pantalla de Portrait (Vertical) a Landscape (Horizontal).

Constraint Layout

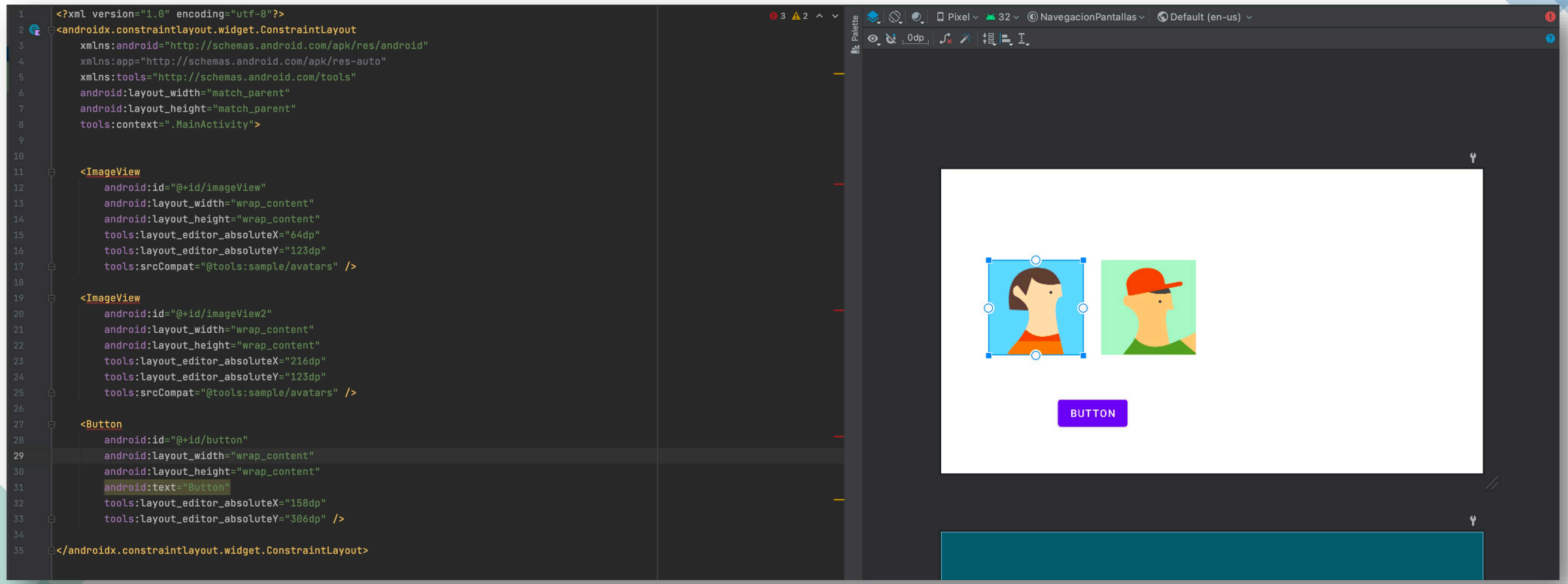
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:layout_editor_absoluteX="64dp"
        tools:layout_editor_absoluteY="123dp"
        tools:srcCompat="@tools:sample/avatars" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:layout_editor_absoluteX="216dp"
        tools:layout_editor_absoluteY="123dp"
        tools:srcCompat="@tools:sample/avatars" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        tools:layout_editor_absoluteX="158dp"
        tools:layout_editor_absoluteY="306dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



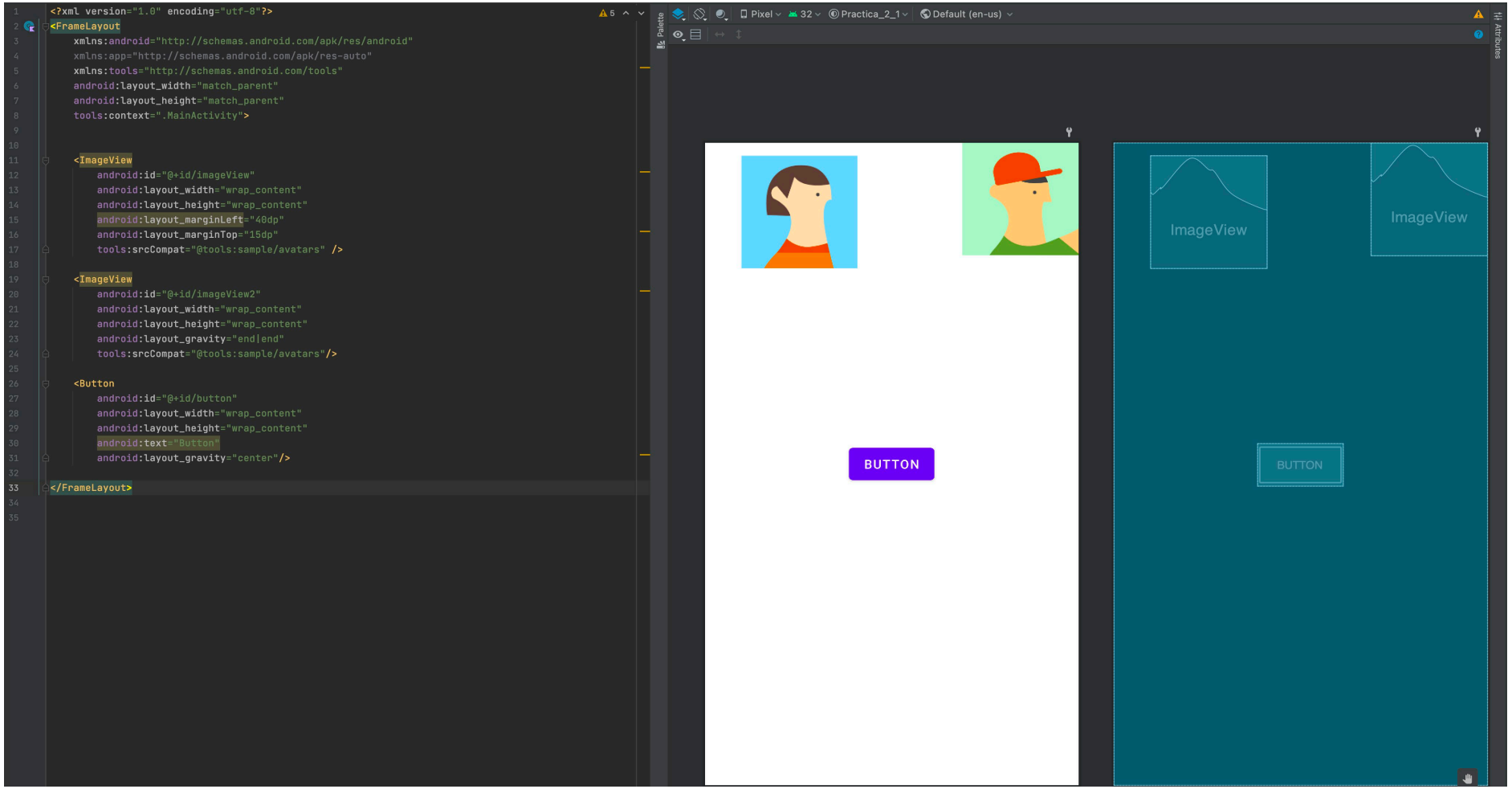
Ejercicio

Ver la práctica 1 del cuaderno de prácticas

Frame Layout

Este ***layout*** alinea todos los elementos de tu diseño (botones, Imágenes, etc.) al lado izquierdo, podemos aplicar márgenes para mostrar un elemento en un punto especifico de la pantalla, puedes agregar varios elementos a tu diseño pero manteniendo el orden de los elementos para que tu diseño se vea agradable.

Frame Layout



ITESO, Universidad Jesuita de Guadalajara

Premio ADA BYRON A LA MUJER TECNÓLOGA

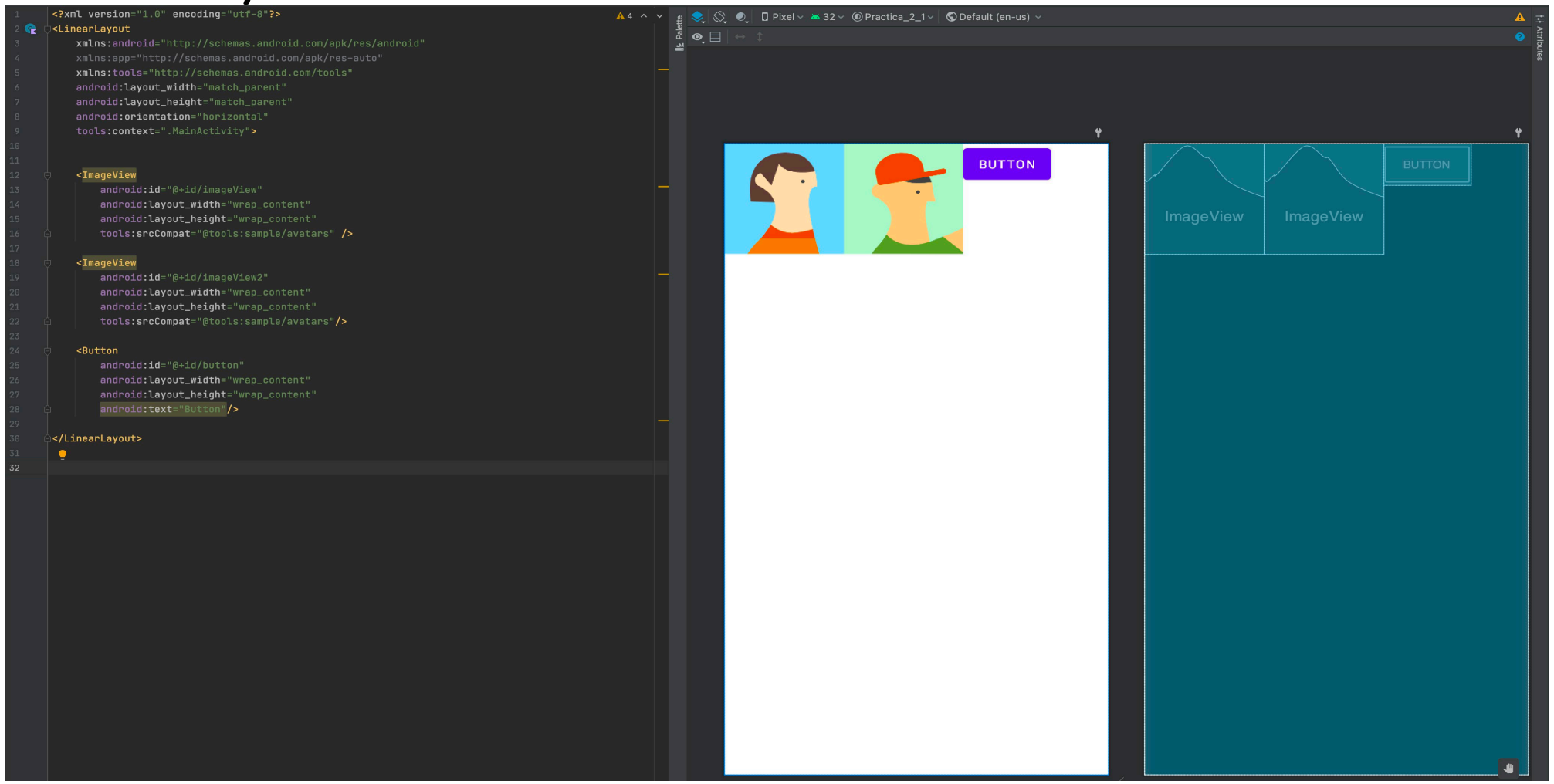
Ejercicio

Ver la práctica 2 del cuaderno de prácticas

Linear Layout

LinearLayout se utiliza para alinear uno o varios elementos de tu diseño de forma vertical u horizontal, la orientación lo definimos con el atributo **android:orientation**=”horizontal” este tipo de ***layout*** los podemos aplicar a los elementos que deben de centrarse en su totalidad, por ejemplo un formulario de contacto o una imagen con un botón, una orientación general a todos los elementos de la vista.

Linear Layout




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:srcCompat="@tools:sample/avatars" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:srcCompat="@tools:sample/avatars"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"/>

</LinearLayout>
```

Ejercicio

Ver la práctica 3 del cuaderno de prácticas

Relative Layout

Relative layout es aconsejable cuando se tenga pocos elementos o cuando es una interface minimalista ya que si no se tiene cuidado al manipular los elementos se puede causar un desorden en el diseño. Los elementos que están dentro de un r***elative layout*** depende entre si. Cuando se mueven los elementos en el diseño se van creando nuevas líneas de atributos.

Relative Layout

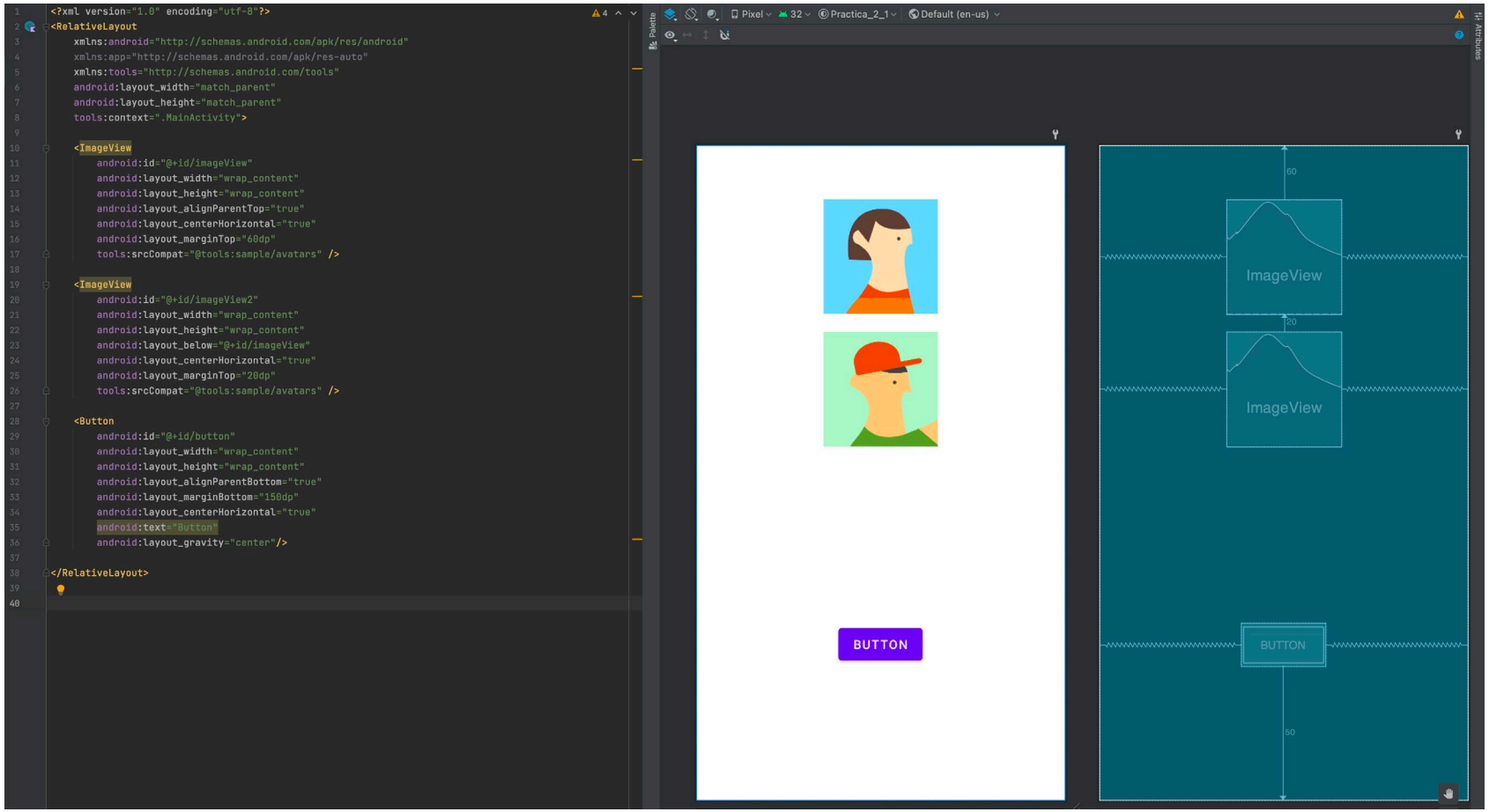
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="60dp"
        tools:srcCompat="@tools:sample/avatars" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        tools:srcCompat="@tools:sample/avatars" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="150dp"
        android:layout_centerHorizontal="true"
        android:text="Button"
        android:layout_gravity="center"/>

</RelativeLayout>
```



Ejercicio

Ver la práctica 4 del cuaderno de prácticas

Navegación entre pantallas

Navegación entre pantallas

Las aplicaciones están diseñadas con más de una sola vista donde podemos observar desde un splashscreen, una vista principal, una vista donde hay algunas configuraciones de la propia aplicación, etc.

Con lo visto anteriormente, podremos hacer una actividad que consiste en una sola pantalla y podemos crear varias activities con lo aprendido, pero ahora veremos como comunicarnos con otra activity y poder visualizarlas.

Ejercicio

Ver la práctica 5 del cuaderno de prácticas

Material Design

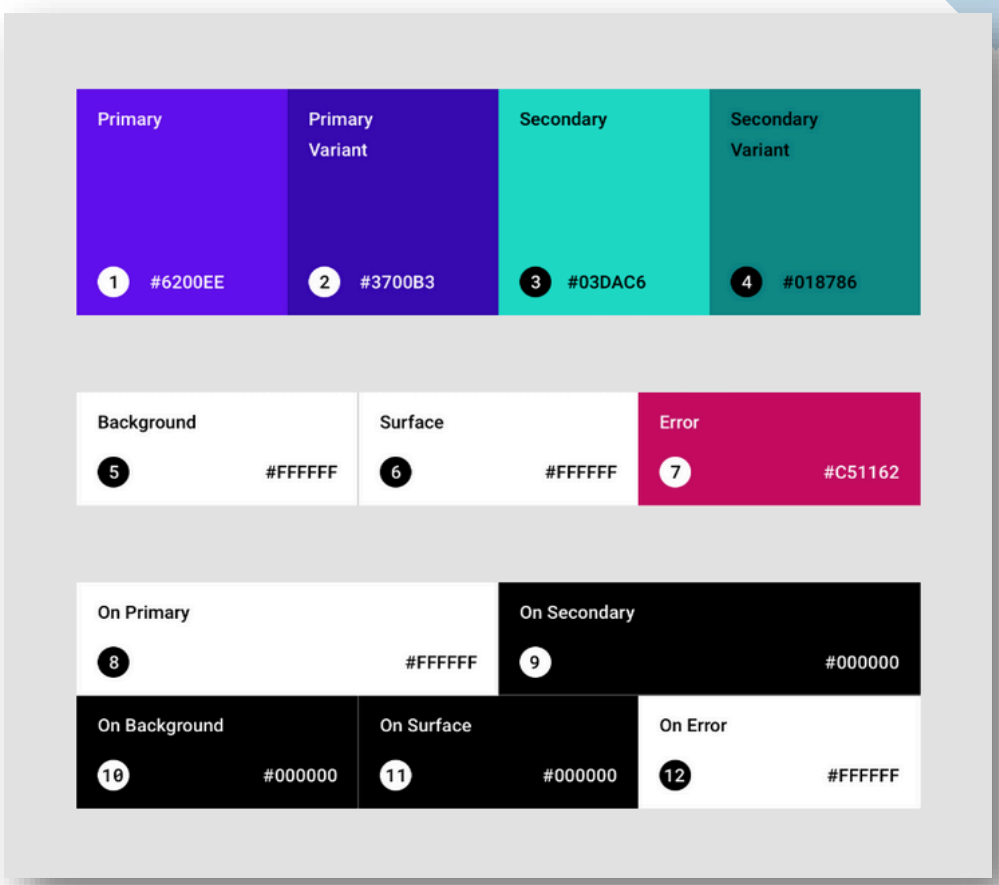
Material design

Material Design no es más que unas guías y pautas para generar un principio de diseño afín en todo nuestro proyecto. Aunque nació con la idea de actualizar el antiguo diseño ***holo*** de Android, Material Design ha sobrepasado la plataforma y está disponible para aplicaciones y páginas webs, no tiene que ser necesariamente Android.

Página oficial de Material Design: https://material.io/

Colores

- **Primary:** Se trata del color más usado de la app, no debe de ser un color que sea muy claro, pues llegaría a ser incómodo al usuario.
- **Primary variant**: Se usa para detalles que acompañen al color principal.
- **Secondary:** Es un color que destaca más que los otros, se utiliza mucho para componentes pequeños que vayamos a usar en la app como f*loating buttons, progress bar*, etc.
- **Secondary variant**: Igual que el *primary variant*, es un diseño más oscuro o claro del color, que podemos utilizar para lo mismo.
- **Background**: Color del fondo de la app.
- **Surface**: Para las superficies de los componentes, como las *cardViews* vistas en capítulos anteriores o menús.
- **Error**: Color muy llamativo para avisar al usuario cuando hay algún problema en la app.



Styles y themes

1

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

2

```xml
<TextView
    style="@style/GreenText"
    ... />
```

3

```xml
<manifest ... >
    <application android:theme="@style/Theme.AppCompat" ... >
    </application>
</manifest>
```

4

```xml
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.AppCompat.Light" ... >
        </activity>
    </application>
</manifest>
```

Los ***styles*** y ***themes*** en Android te permiten separar los detalles de diseño de tu aplicación de la estructura y el comportamiento de la IU, de forma similar a las hojas de estilo en el diseño web.

Un ***style*** es una colección de atributos que especifican la apariencia de una sola View. Un ***style*** puede especificar atributos como el color y el tamaño de fuente, el color de fondo y mucho más.

Un ***theme*** es una colección de atributos que se aplican a toda una aplicación, actividad o jerarquía de vistas, no solo a una vista individual. Cuando aplicas un ***theme***, cada vista de la aplicación o actividad aplica cada uno de los atributos del tema que admite. Los ***themes*** también pueden aplicar ***styles*** a elementos que no se ven, como la barra de estado y el fondo de la ventana.

Button

Un ***button*** consiste en un texto o un ícono (o ambos) que comunica la acción que ocurrirá cuando el usuario lo toque.


```xml
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Ejercicio

Ver la práctica 6 del cuaderno de prácticas

Programación avanzada

Data Classes

Una ***data class*** es una clase que contiene solamente atributos que quedemos guardar. Con esto conseguimos por ejemplo no tener varias variables «nombre1», «nombre2» para almacenar todos los valores con los que vamos a trabajar.

```
data class Superhero(
        var superhero:String,
        var publisher:String,
        var realName:String,
        var photo:String
)
```

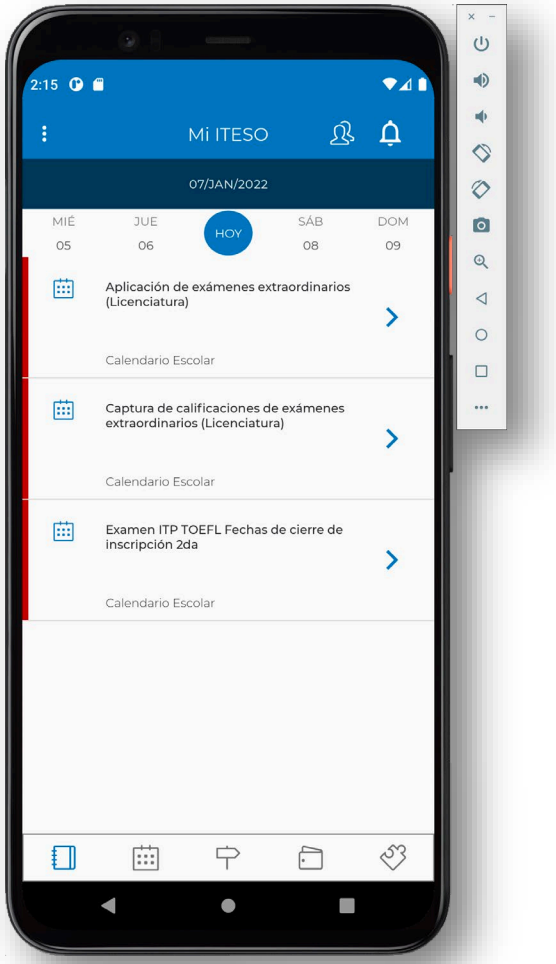
Ejercicio

Ver la práctica 7 del cuaderno de prácticas

RecyclerView

RecyclerView facilita que se muestren de manera eficiente grandes conjuntos de datos. Tú proporcionas los datos y defines el aspecto de cada elemento, y la biblioteca RecyclerView creará los elementos de forma dinámica cuando se los necesite.

Como su nombre lo indica, ***RecyclerView*** recicla esos elementos individuales. Cuando un elemento se desplaza fuera de la pantalla, ***RecyclerView*** no destruye su vista. En cambio, reutiliza la vista para los elementos nuevos que se desplazaron y ahora se muestran en pantalla.



ITESO, Universidad Jesuita de Guadalajara

Premio ADA BYRON A LA MUJER TECNÓLOGA

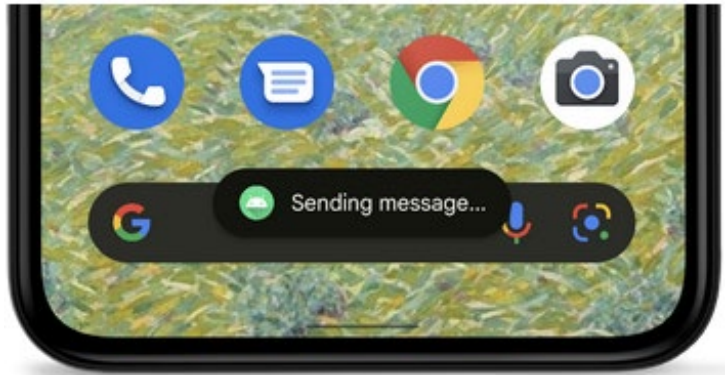
Ejercicio

Ver la práctica 8 del cuaderno de prácticas

Mensajes y alertas

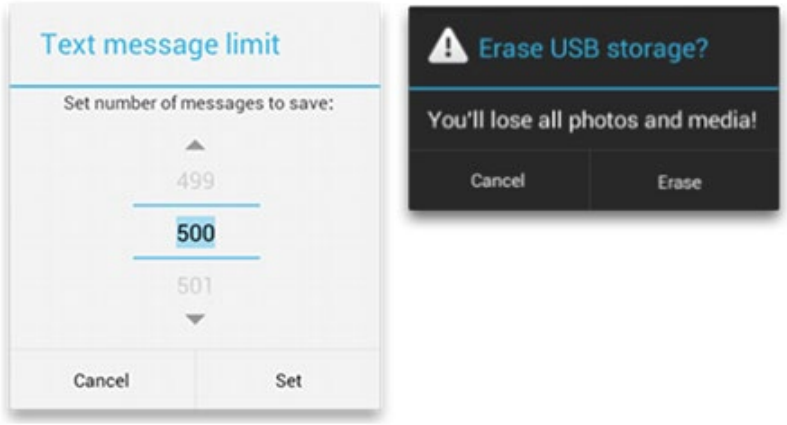
Mensajes Toast

Un ***mensaje toast*** proporciona información simple sobre una acción en una pequeña ventana emergente. Solo ocupa la cantidad de espacio necesario para el mensaje, y la actividad en curso permanece visible y admite la interacción. Los avisos desaparecen automáticamente después de un tiempo de espera.



Mensajes Alerta (Cuadros de diálago)

Un ***cuadro de diálogo*** es una ventana pequeña que le indica al usuario que debe tomar una decisión o ingresar información adicional. Un ***diálogo*** no ocupa toda la pantalla y, generalmente, se usa para eventos modales que requieren que los usuarios realicen alguna acción para poder continuar.



Proyecto final

Empezaremos el desarrollo de nuestro proyecto final, vamos al cuaderno de prácticas al ejercicio 9.

Referencias

- Que son los Layouts y cuales existen en Android Studio. Recuperado de https://blog.nubecolectiva.com/que-son-los-layouts-y-cuales-existen-en-android-studio/
- Material Design y estilos en Kotlin. Recuperado de https://cursokotlin.com/capitulo-24-material-design-y-estilos-kotlin/
- Data Classes en Kotlin. Recuperado de https://cursokotlin.com/capitulo-14-data-classes-en-kotlin/
- Recyclerview en Kotlin. Recuperado de https://cursokotlin.com/capitulo-15-recyclerview-kotlin/

ITESO, Universidad Jesuita de Guadalajara

Premio
ADA BYRON
A LA MUJER TECNÓLOGA

ITESO, Universidad Jesuita de Guadalajara

Premio
ADA BYRON
A LA MUJER TECNÓLOGA