Práctica 2

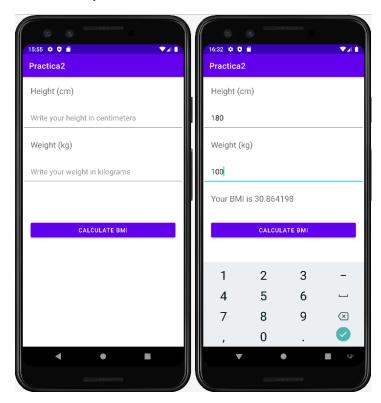
Contenido

D	ata Bindingata	3
	Creando pantalla de la aplicación	[
	·	
	Construyendo lógica de la aplicación	7

Objetivo: Configurar una aplicación Android para el uso de Data Binding Layout y acceder a los Views desde la lógica de la aplicación.

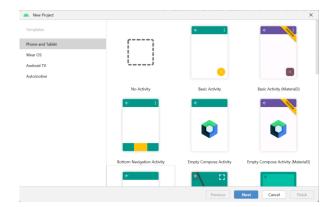
Funcionamiento: La aplicación calcula el índice de masa corporal a partir de la estatura y peso ingresado por el usuario.

Resultado Esperado:



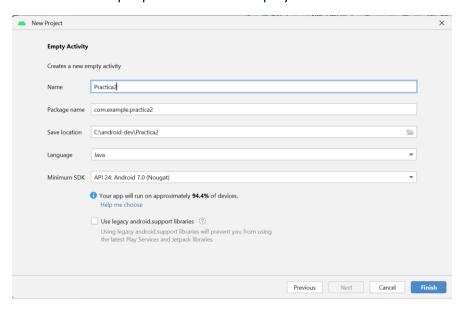
Crear un nuevo proyecto File->New Proyect

Seleccionamos Empty Activiy

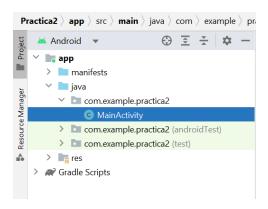




Llenamos los campos para definir nuestro proyecto.



Después de pulsar el botón finalizar, se cargarán los elementos básicos para el funcionamiento de nuestra aplicación



Data Binding

De acuerdo con la documentación de Android, un Data Binding es una biblioteca de soporte que permite vincular los componentes de la interfaz de usuario a fuentes de datos en la aplicación mediante un formato declarativo en lugar de mediante programación.

De esta manera se aligera la carga de los componentes en tiempo de operación.

Para incluir el uso de esta biblioteca, nos iremos al panel del proyecto y en aparatado de Gradle Scripts, buscaremos el archivo **build.gradle (Module :app)** lo abrimos y acontinuación veremos su contenido.

```
⊕ 至 🛨 🗢 —
 ▲ Android
                                                🚜 activity_main.xml 🗴 🏿 🖒 MainActivity.java 🗡 🔊 build.gradle (:app) 🗡
🗡 📭 арр
                                                 You can use the Project Structure dialog to view and edit your project configuration
  > manifests
                                                         plugins {

✓ iava

                                                 2
                                                              id 'com.android.application'
     com.example.practica2
                                                 3
                                                       △}

    MainActivity

                                                 4
     > com.example.practica2 (androidTest)
                                                 5
                                                       eandroid {
     > com.example.practica2 (test)
                                                              namespace 'com.example.practica2'
                                                 6
  > res
                                                              compileSdk 33
  Gradle Scripts
      📝 build.gradle (Pi
                                                 8
     build.gradle (Module :app)
                                                 9
                                                              defaultConfig {
     proguard-rules.pro (Pro
                                                                  applicationId "com.example.practica2"
     gradle.properties (Project Properties)
                                                                  minSdk 24
     gradle-wrapper.properties (Gradle Version)
                                                                  targetSdk 33
     local.properties (SDK Location)
                                                                  versionCode 1
     settings.gradle (Project Settings)
                                                14
                                                                   versionName "1.0"
                                                16
                                                                   testInstrumentationRunner "androidx.test.
```

Ahora, dentro del apartado android{... escribiremos la siguiente entrada

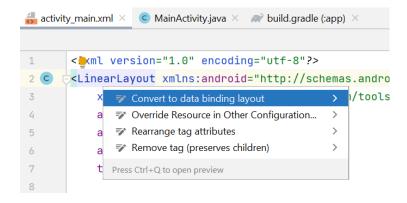
```
buildFeatures{
   dataBinding true
}
```

Posterior nos aparecerá un mensaje en la parte superior de la pantalla, indicando que hubo cambios en el archivo de configuración; por lo cual pulsaremos el hipervínculo "Sync Now"

```
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.
                                                                                         Sync Now
                                                                                                  Ignore these changes
               versionCode 1
               versionName "1.0"
               testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
18
           buildTypes {
19
20
               release {
                   minifyEnabled false
                   proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
           }
           compileOptions {
               sourceCompatibility JavaVersion. VERSION_1_8
               targetCompatibility JavaVersion.VERSION_1_8
           buildFeatures{
               dataBinding true
```

Posteriormente, nos pasamos al activity_main.xml, nos posicionamos sobre la etiqueta del Layout principal (en este ejemplo, se está usando un LinearLayout).

- Pulsamos las teclas Alt+Enter
- Se desplegará un menú contextual del cual seleccionaremos la opción "Convert to data binding layout" y pulsamos Enter



A continuación, automáticamente nuestro LinearLayout quedará dentro de una etiqueta de tipo Layout

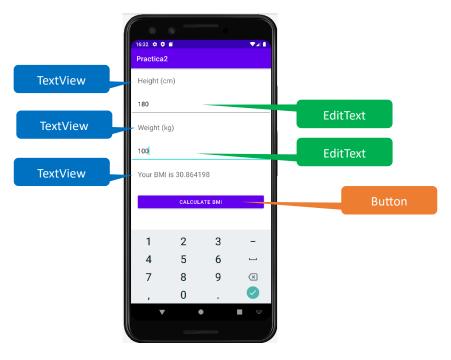
```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
```

Además, se incluirá la etiqueta <data> </data>

```
activity main.xml ×
                  © MainActivity.java × ⋒ build.gradle (:app) ×
        <?xml version="1.0" encoding="utf-8"?>
        <layout xmlns:android="http://schemas.android.com/apk/res/android"</pre>
            xmlns:tools="http://schemas.android.com/tools">
            <data>
 5
            </data>
8
9
            <LinearLayout
                android:layout_width="match_parent"
10
                android:layout_height="match_parent"
                android:orientation="vertical"
                tools:context=".MainActivity">
13
14
15
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
17
                    android:text="Hello World!" />
18
19
        </LinearLayout>
20
        </layout>
```

Creando pantalla de la aplicación

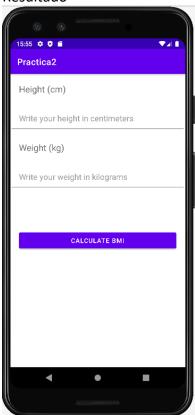
Elementos de la interfaz de usuario



A continuación, podemos generar la pantalla de nuestra aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"</pre>
   xmlns:tools="http://schemas.android.com/tools">
    <data>
    </data>
    <LinearLayout
        android:layout width="match parent"
        android:layout height="match parent"
        android:orientation="vertical"
        tools:context=".MainActivity">
        <TextView
            android:layout width="match parent"
            android:layout height="wrap content"
            android:padding="20dp"
            android:text="@string/add height"
            android:textSize="20dp" />
        <EditText
            android:id="@+id/user height"
            android:layout width="match parent"
            android:layout height="wrap content"
            android:inputType="number"
            android:padding="20dp"
            android:hint="@string/write height cm" />
        <TextView
            android:layout width="match parent"
            android:layout height="wrap content"
            android:padding="20dp"
            android:text="@string/add weight"
            android:textSize="20dp" />
        <EditText
            android:id="@+id/user weight"
            android:layout width="match parent"
            android:layout height="wrap content"
            android:inputType="number"
            android:padding="20dp"
            android:hint="@string/write weight kg" />
        <TextView
           android:id="@+id/text result"
            android:layout width="match parent"
            android:layout height="wrap content"
            android:padding="20dp"
            android:textSize="20dp" />
        <Button
            android:id="@+id/button"
            android:layout width="match parent"
```

Resultado



Construyendo lógica de la aplicación

Dentro de nuestra clase MainActivity vamos a declarar un ActivityMainBinding

```
public class MainActivity extends AppCompatActivity {
    ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
    }
}
```

La clase ActivityMainBinding se generó cuando transformamos nuestro **activity_main.xml** en un data binding layout.

El método **inflate()** crea un objeto de tipo data binding layout. En ese momento podemos acceder a los datos del layout principal. Lo podemos observar en la línea **setContentView(binding.getRoot())**;

inflate() es una función en Android que se utiliza para convertir un archivo de diseño XML en una jerarquía de vistas en tiempo de ejecución. Es decir, se utiliza para inflar o crear vistas a partir de un archivo XML en la aplicación de Android.

La línea de código **ActivityMainBinding.inflate(getLayoutInflater())** se utiliza para inflar el archivo de diseño XML "activity_main.xml" en una instancia de la clase "**ActivityMainBinding**".

La clase "ActivityMainBinding" es generada automáticamente por la herramienta de vinculación de datos (data binding) de Android y contiene referencias a todas las vistas definidas en el archivo de diseño. La llamada al método "inflate" en la clase "ActivityMainBinding" devuelve una instancia de la vista raíz del archivo de diseño.

La función "getLayoutInflater()" devuelve una instancia de LayoutInflater que se utiliza para inflar vistas a partir de archivos de diseño XML. En este caso, se pasa la instancia de LayoutInflater como argumento al método "inflate()" de la clase "ActivityMainBinding" para inflar el archivo de diseño "activity main.xml".

Una vez inflada la vista, se puede acceder a todas las vistas y elementos del archivo de diseño mediante las referencias generadas en la clase "ActivityMainBinding"

A continuación, creamos nuestra función para el cálculo del índice de masa corporal (IMC)

```
private double calc_imc(double peso, double altura) {
    double altura_m = altura/100;
    return peso/(altura_m*altura_m);
}
```

Finalmente agregamos el evento del método setOnClickListener del botón de nuestra pantalla

```
binding.button.setOnClickListener(v->{
    try {
        double peso = Double.parseDouble(binding.userWeight.getText().toString());
        double altura = Double.parseDouble(binding.userHeight.getText().toString());
        binding.textResult.setText(getString(R.string.result_message, calc_imc(peso,
        altura)));
    }catch(NumberFormatException ex) {
        Log.e("MainActivity", ex.toString());
        Toast.makeText(this, getString(R.string.error_message), Toast.LENGTH_SHORT).show();
    }
});
```

Quedando de la siguiente manera

```
activity_main.xml × 🌏 strings.xml × 🥲 MainActivity.java × 🔊 build.gradle (:app)
                                                                                                            %3 ^ v
       public class MainActivity extends AppCompatActivity {
           ActivityMainBinding binding;
           @Override
15 🍑
           protected void onCreate(Bundle savedInstanceState) {
               super.onCreate(savedInstanceState);
               binding = ActivityMainBinding.inflate(getLayoutInflater());
               setContentView(binding.getRoot());
19
               binding.button.setOnClickListener(v->{
                    try {
                       double peso = Double.parseDouble(binding.userWeight.getText().toString());
                       double altura = Double.parseDouble(binding.userHeight.getText().toString());
                       binding.textResult.setText(getString(R.string.result_message, calc_imc(peso, altura)));
                   }catch(NumberFormatException ex){
26
                       Log.e( tag: "MainActivity", ex.toString());
                        Toast.makeText(context this, getString(R.string.error_message), Toast.LENGTH_SHORT).show();
28
29
               });
30
```

- Con las líneas 22 y 23 obtenemos el peso y altura introducidos a la aplicación por el usuario.
- La línea 24 muestra la asignación del resultado de la función calc_imc() que creamos anteriormente

El resultado de la aplicación la podemos ver a continuación

