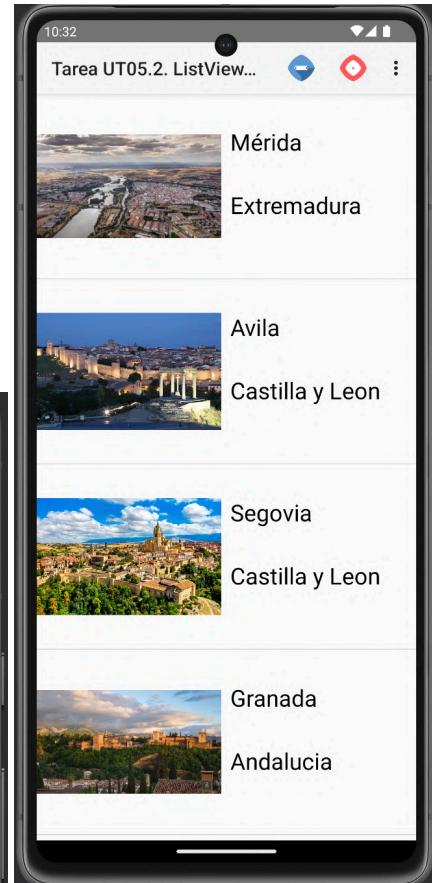


Tarea 5 para Programación Multimedia y Dispositivos Móviles



Diego Manuel Carrasco Castañares

PMDM05.- Interacciones entre clases y actividades. Tarea

Detalles de la tarea de esta unidad.

Enunciado.

Tarea UT05.1. Calcular índice masa corporal. (4 puntos)

Esta tarea consiste en crear una aplicación que permita calcular el índice de masa corporal de una persona.

Para ello en la primera actividad se pedirá al usuario el nombre, altura y peso como se indica en la imagen. Al pulsar el botón Calcular IMC nos llevará a la segunda actividad donde nos da la información que se muestra en la imagen y calculará el IMC e indicará según la tabla el peso de la persona. La tabla resumen IMC se obtiene de internet.

Además, antes de que aparezca la primera actividad, se mostrará una pantalla de presentación, Splash Screen, con una imagen seleccionada por el alumno y con un texto que será el nombre del alumno que realiza la app.

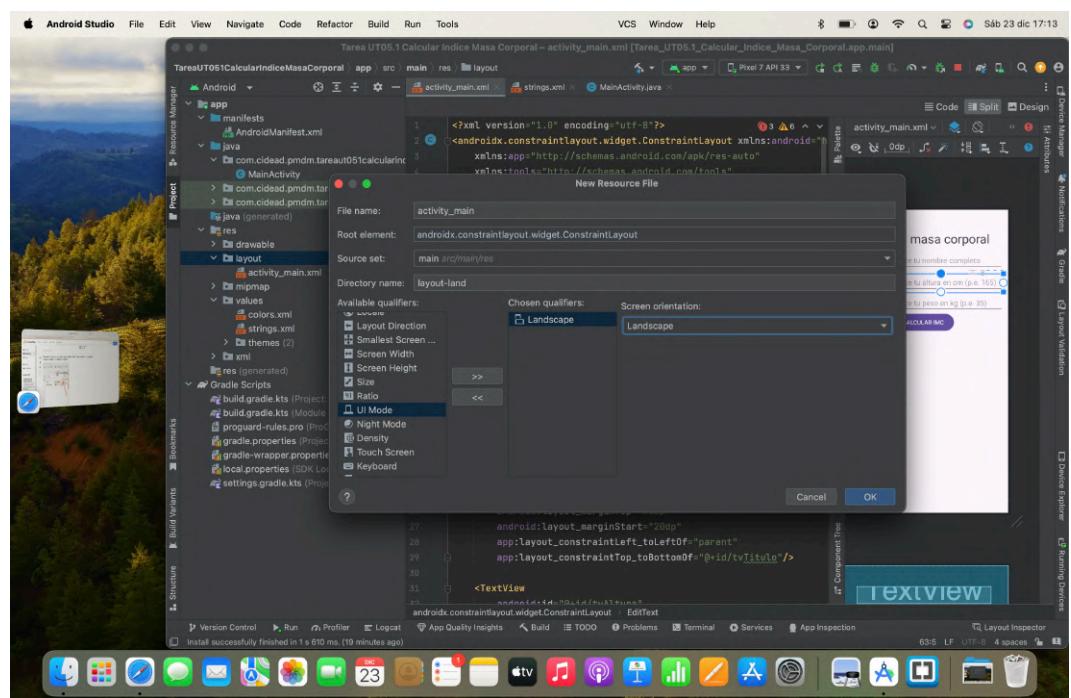
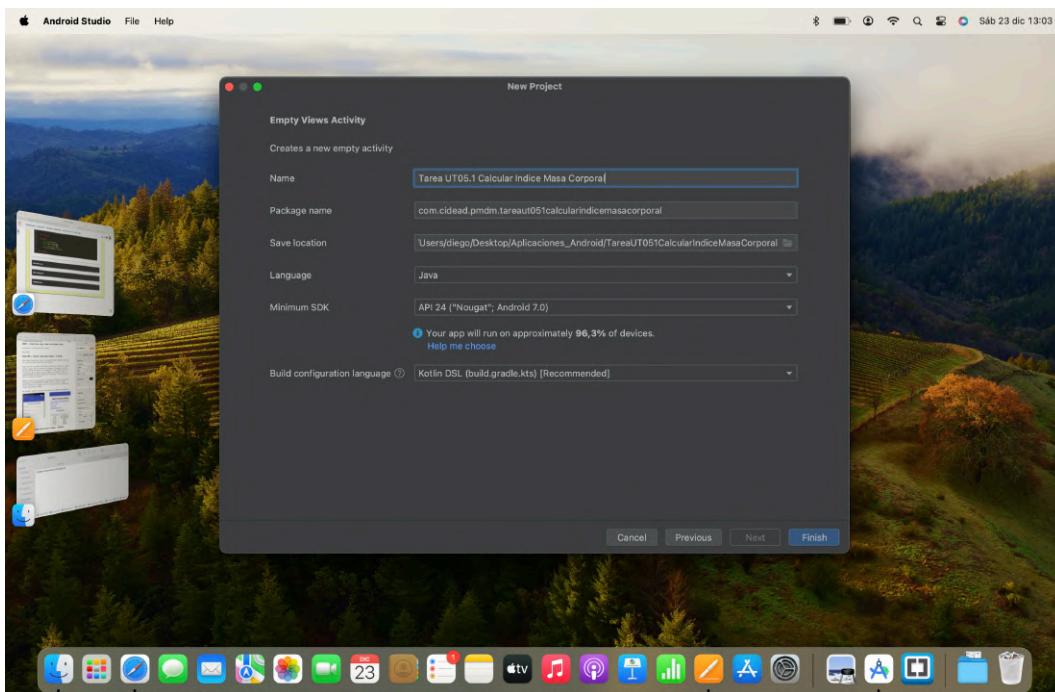
La interfaz tendrá el aspecto mostrado en la captura de pantalla.



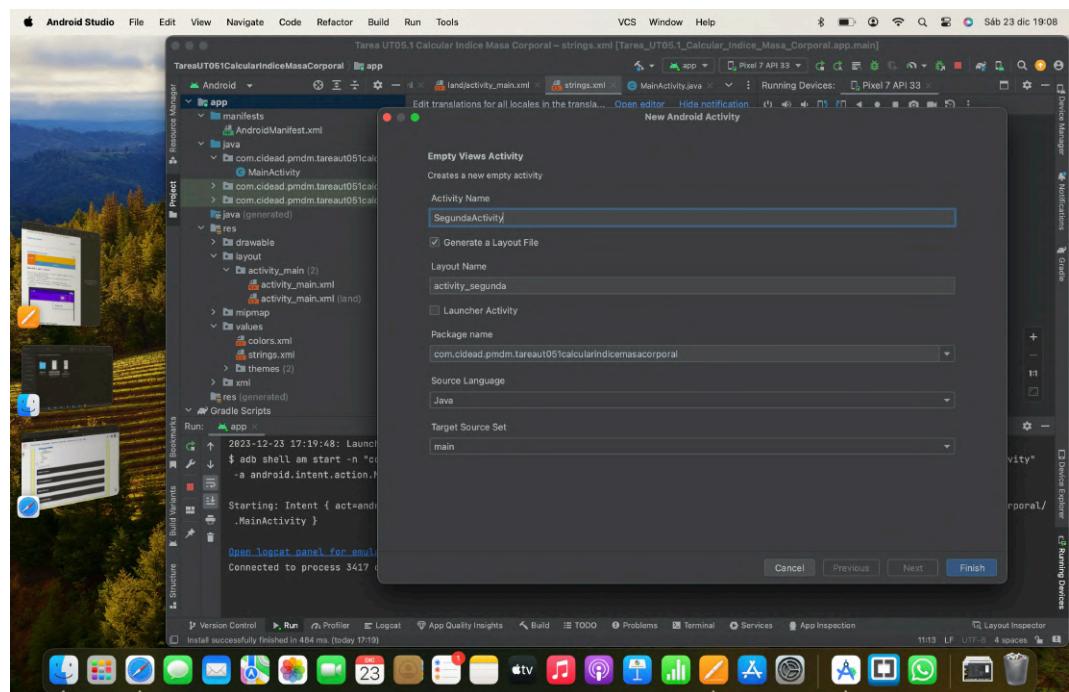
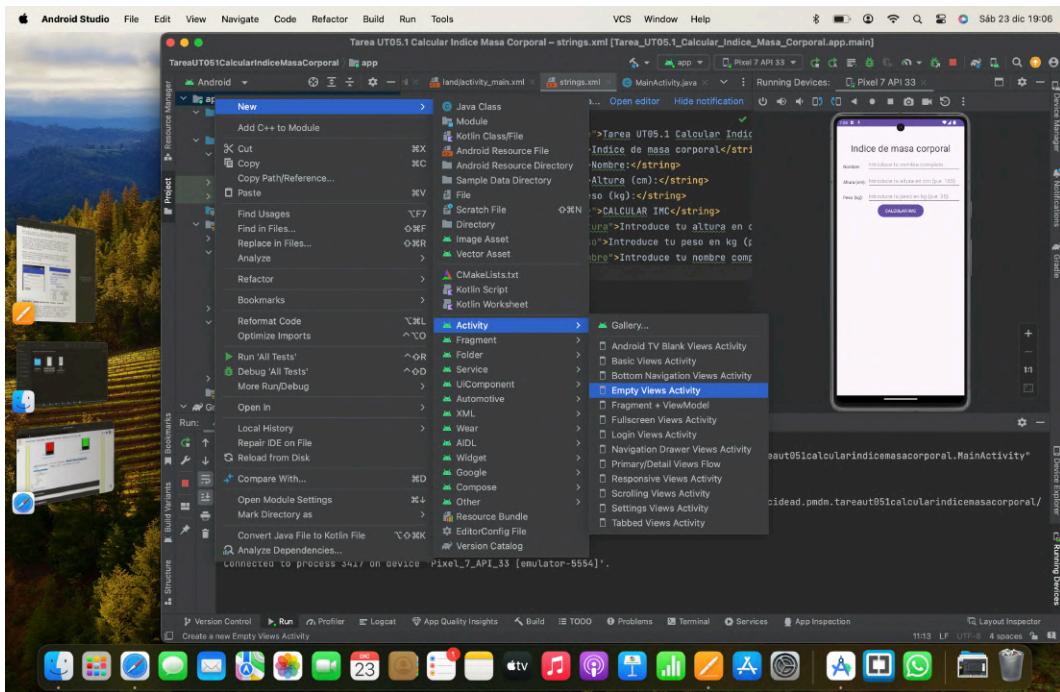
En primer lugar crearemos el proyecto al cual llamaremos "Tarea UT05.1. Calcular índice masa corporal".

Tras ello crearemos el activity_main en xml para pantallas apaisadas.

En las siguientes imágenes podemos ver estos dos pasos.

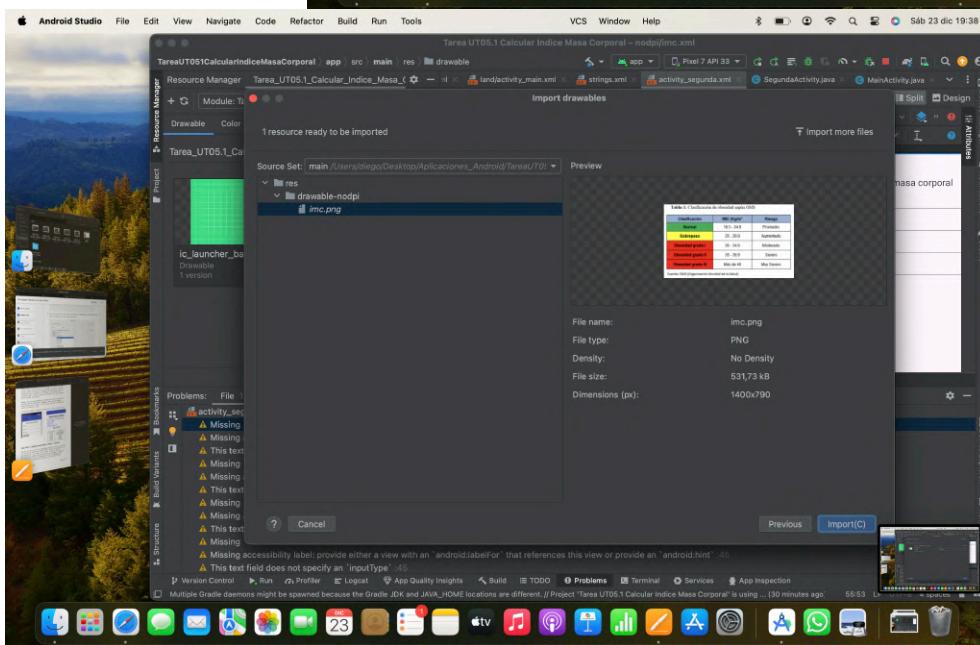
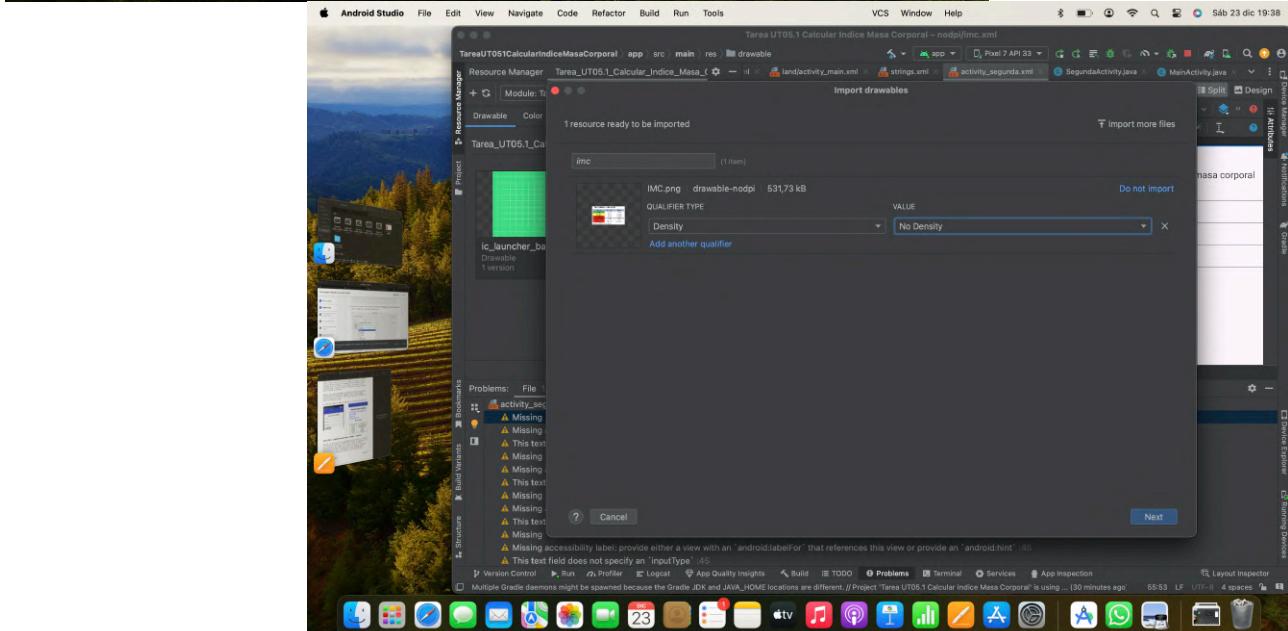
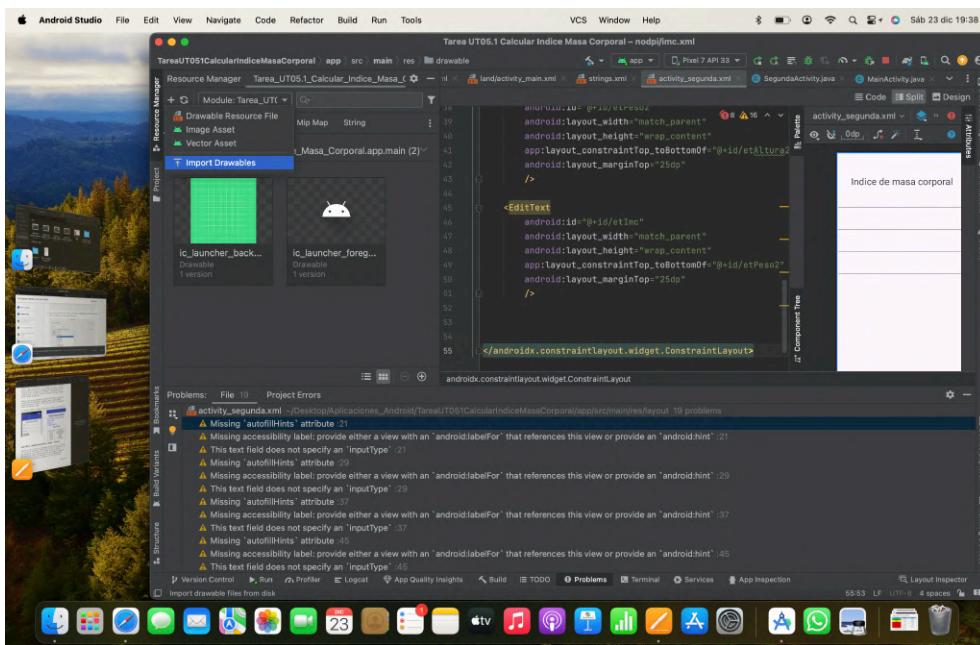


Tras ello crearemos una nueva activity a la cual llamaremos SegundaActivity.

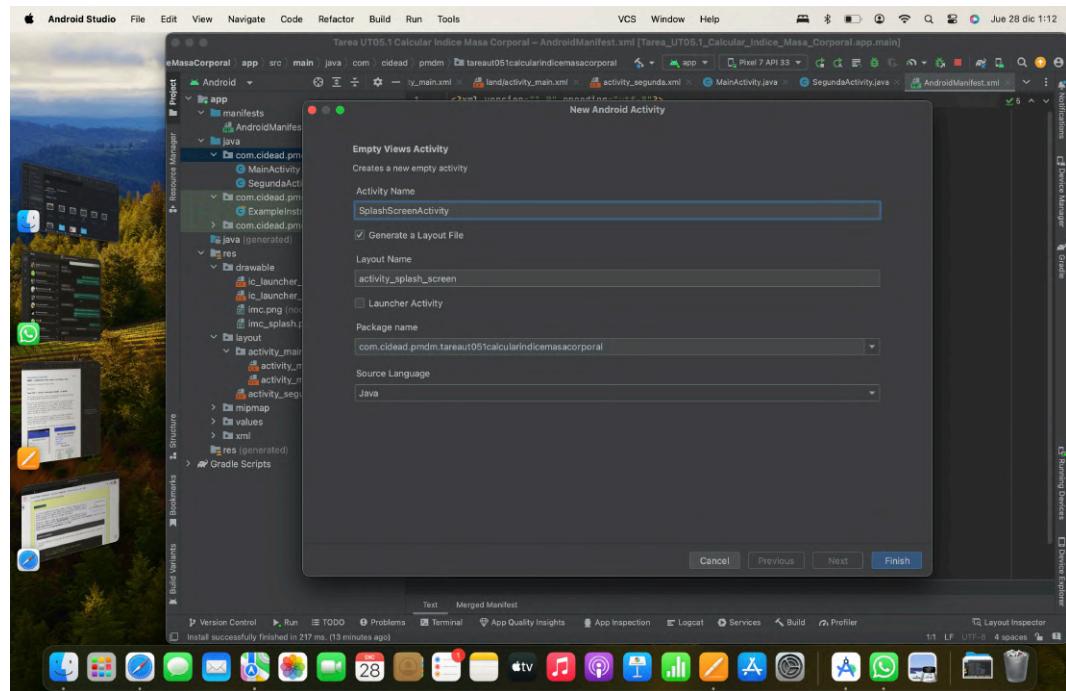
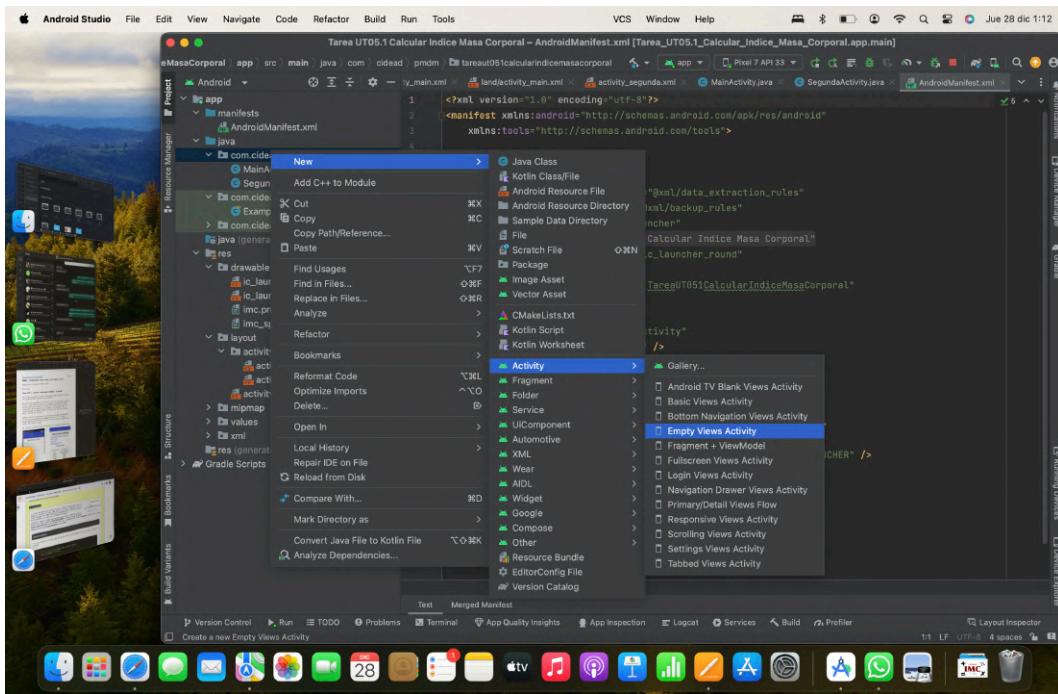


El siguiente paso será añadir la imagen del IMC al proyecto, para ello pincharemos en resource manager, en el botón de +, en import drawables, seleccionamos la imagen le decimos qualifier type es "density" y el valor "no density" y le damos a importar.

De esta manera tenemos la imagen importada en nuestra carpeta de drawables como podemos ver en las siguientes imágenes.



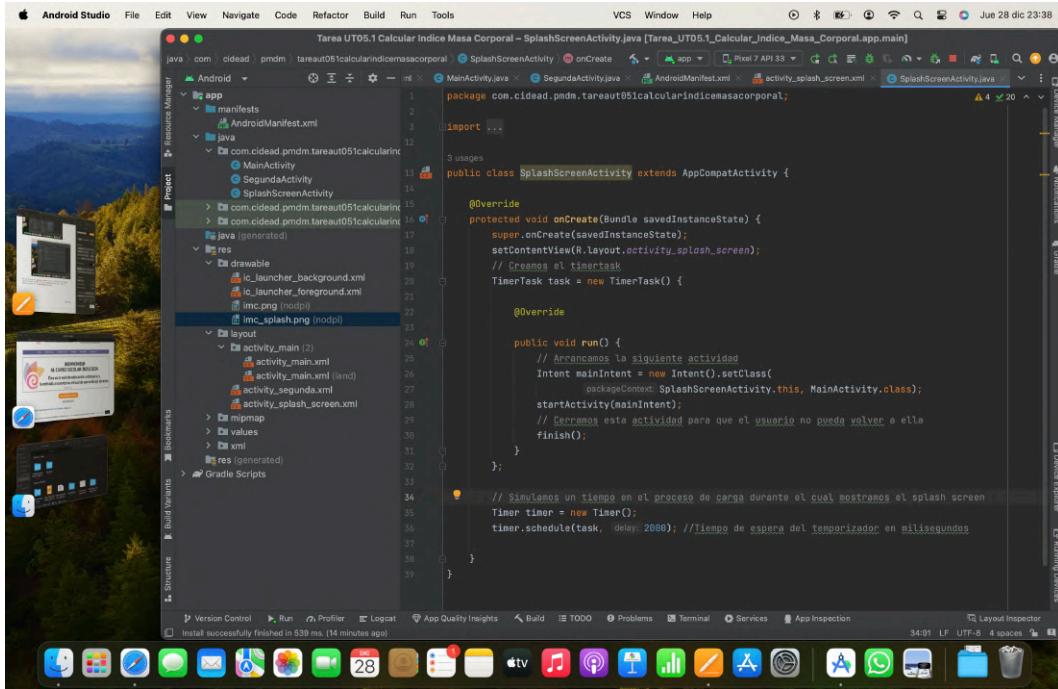
El siguiente paso será crear una nueva activity para el splash_screen.



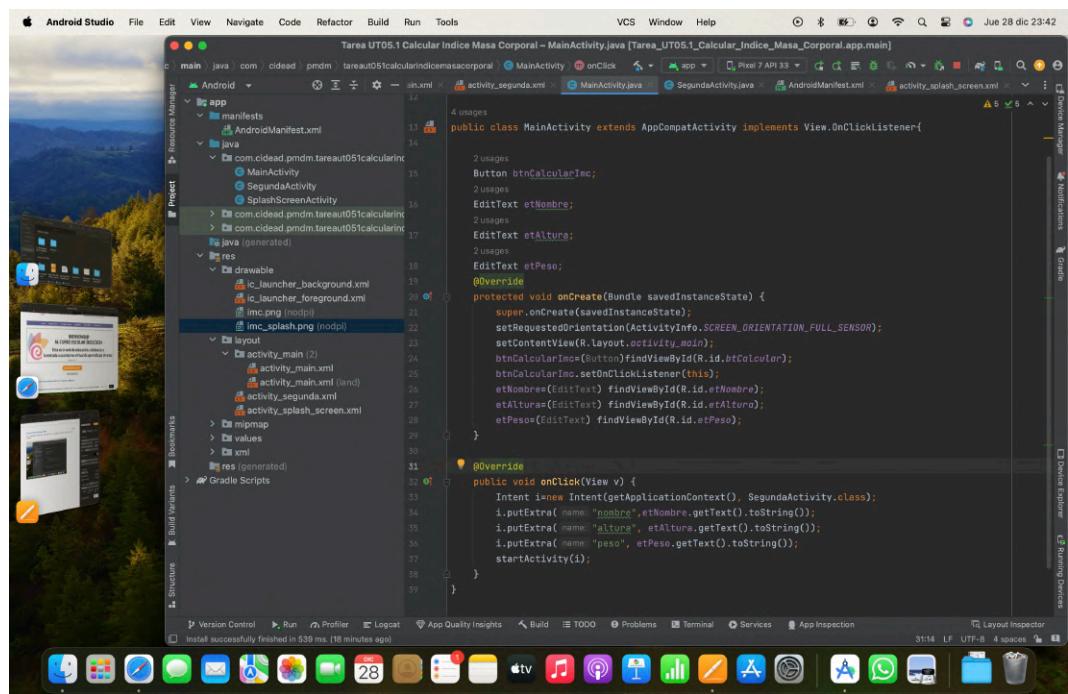
En las siguientes imágenes podemos ver el código de las clases java que hemos creado, de sus correspondientes xml y de las modificaciones en el AndroidManifest.xml.

Empezaremos mostrando el código de SplashScreenActivity, en el cual dentro del método onCreate, creamos un TimerTask, sobreescrivimos el método run, en el cual arrancamos la actividad mediante un intento y la cerramos para que el usuario no pueda volver a ella. Luego simulamos un tiempo de proceso de carga, en nuestro caso de

dos segundos, para que nos muestre el splash screen durante ese tiempo.



En el mainActivity crearemos cuatro variables, una de tipo Button y tres de tipo edittext. Dentro del método onCreate estableceremos el método setRequestedOrientation para que la actividad actúe en función de si esta apaisada o no. Sobreescribiremos el método onClick, en el cual crearemos un intent hacia la SegundaActivity, que guardara los datos mediante el método putExtra y posteriormente iniciaremos dicha actividad.



En la clase SegundaActivity crearemos cuatro variables, una de tipo button y tres de tipo textView.

Posteriormente sobreescribiremos el método onClick para volver a la actividad anterior mediante el método finish.

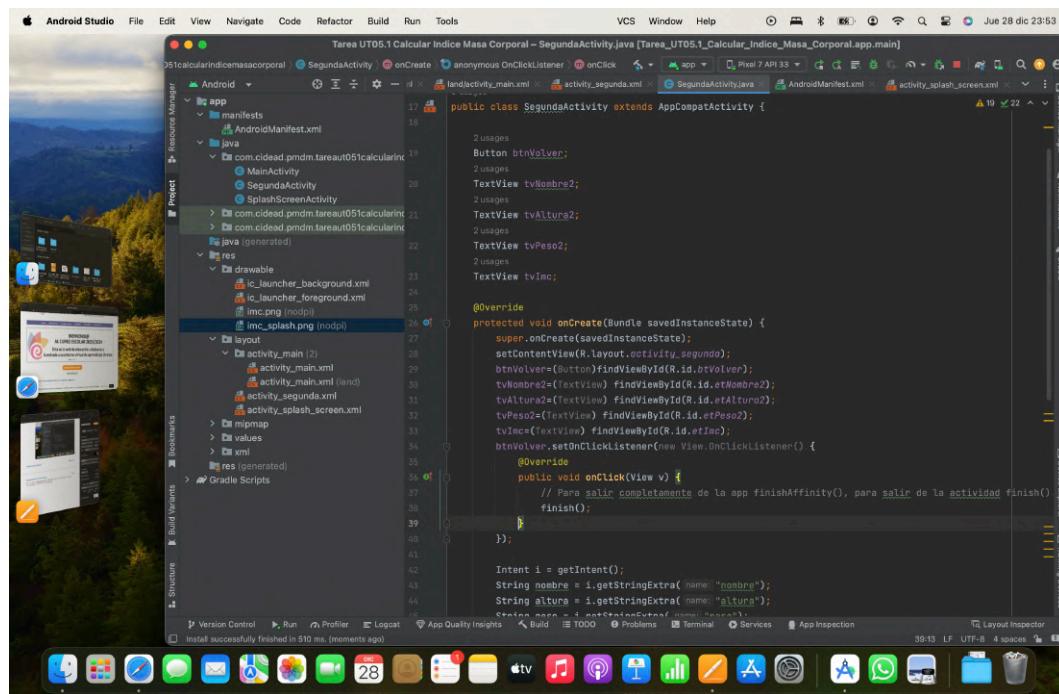
Tras ello crearemos el intent, crearemos las variables para los datos que contiene el intent de tipo string (nombre altura y peso) y las instanciaremos mediante el método getStringExtra.

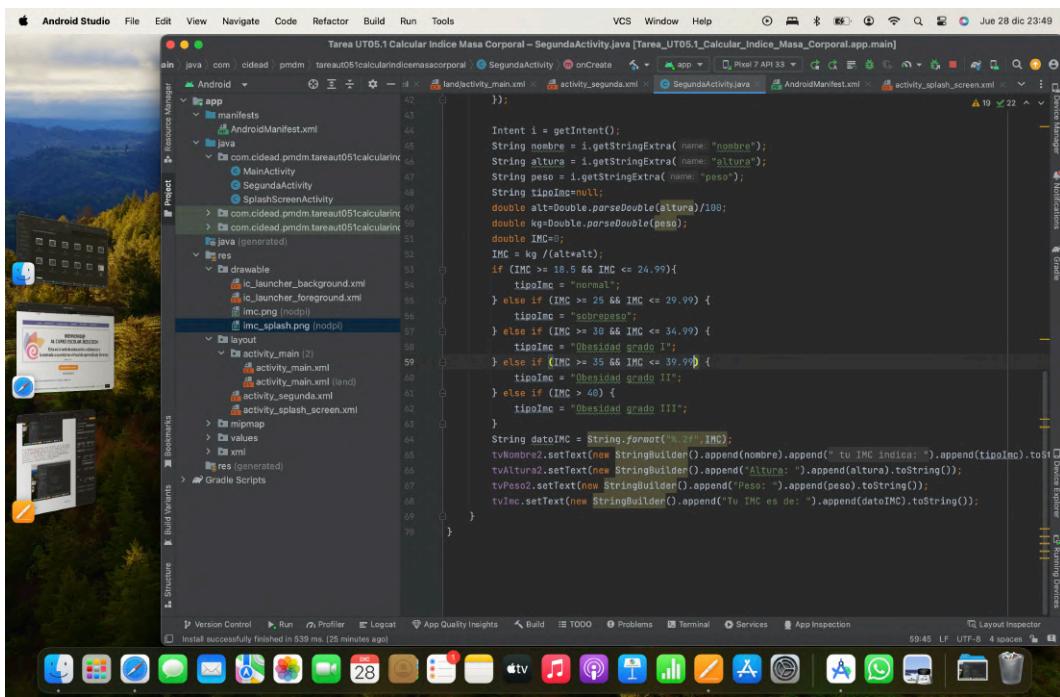
También crearemos otra variable de tipo stirng llamada tipoImc, tres de tipo double llamadas alt (para contener la altura en metros), kg (para contener el peso) y IMC para contener el IMC tras hallar el calculo.

Mediante un if instanciaremos en la variable tipoImc el tipo de IMC en función del valor contenido en IMC.

Tras ello formatearemos el resultado de IMC a dos decimales y lo guardaremos en una nueva variable de tipo strign llamado datoImc.

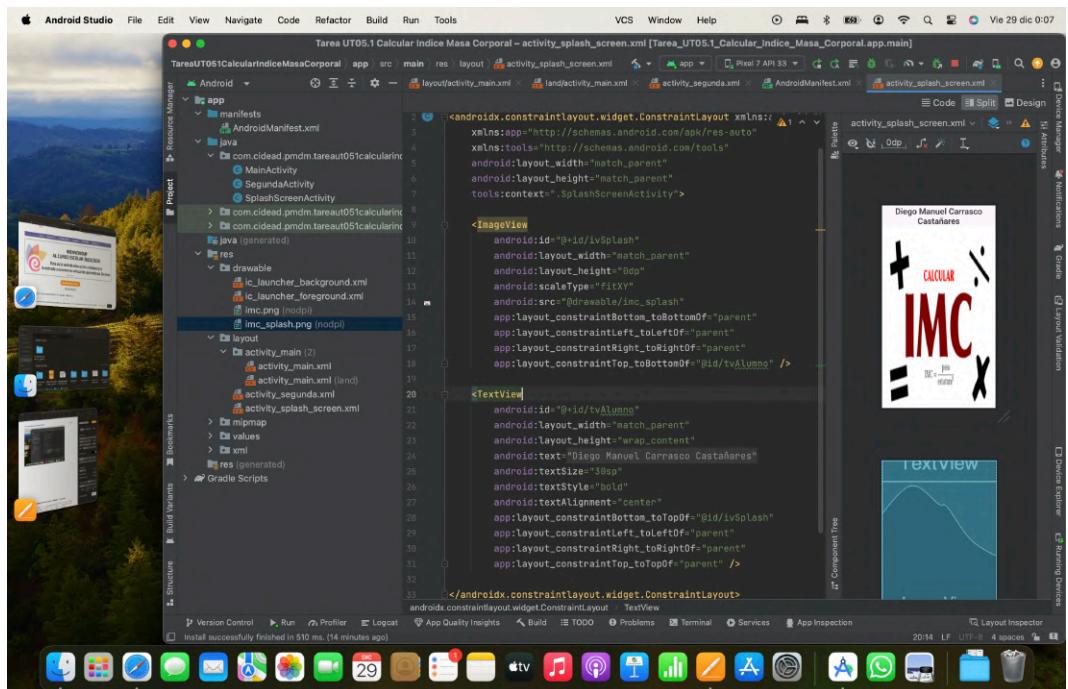
Lo último que haremos será establecer en los textView los datos mediante el método setText al cual le pasaremos los datos que tiene que grabar (las variables creadas anteriormente que nos interese mostrar) y los concatenaremos mediante un stringbuilder().append.





En el xml de activity_splash_Screen un imagenvie que mostrará la imagen que hemos guardado para tal fin tal como hicimos con la imagen del IMC.

También contendrá un textView con nuestro nombre, con tamaño de letra 30, en negrita y centrada, el cual se situara encima del imagenvie mediante sus restricciones.



El activity_main.xml constara de cuatro textView, tres edittext y un button.

El primer textview contendrá la frase "Indice de masa corporal" en un tamaño de letra de 30 y estará situado en la parte superior de la pantalla.

El segundo textview contendrá la frase "Nombre:", se situara justo debajo del textview anterior y tendrá un margen de 30 sobre dicho textview y de 20 sobre el lado izquierdo.

El tercer textview será idéntico al anterior pero contendrá la frase "Altura (cm):" y se situara debajo del mismo.

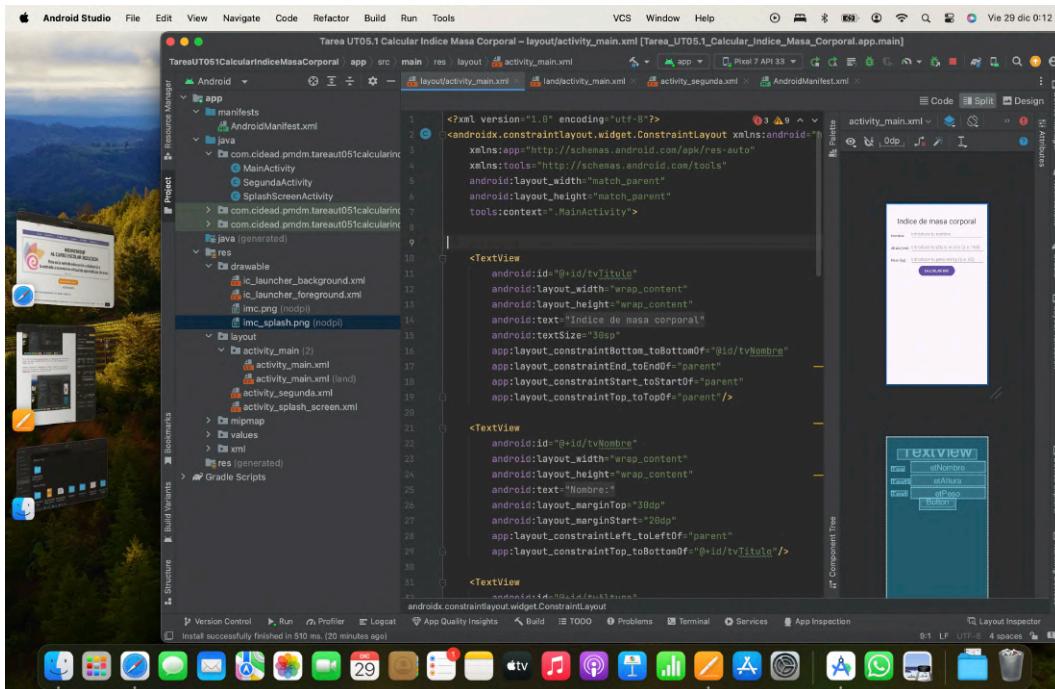
El cuarto textview será idéntico a los dos anteriores pero con la frase "Peso (kg);" y se situara justo debajo del anterior.

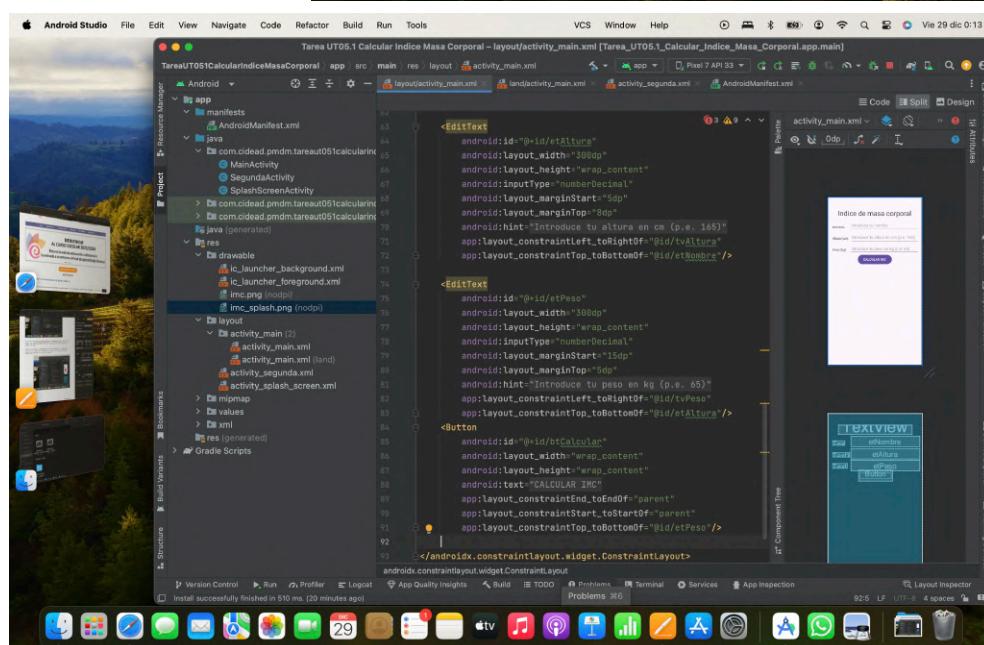
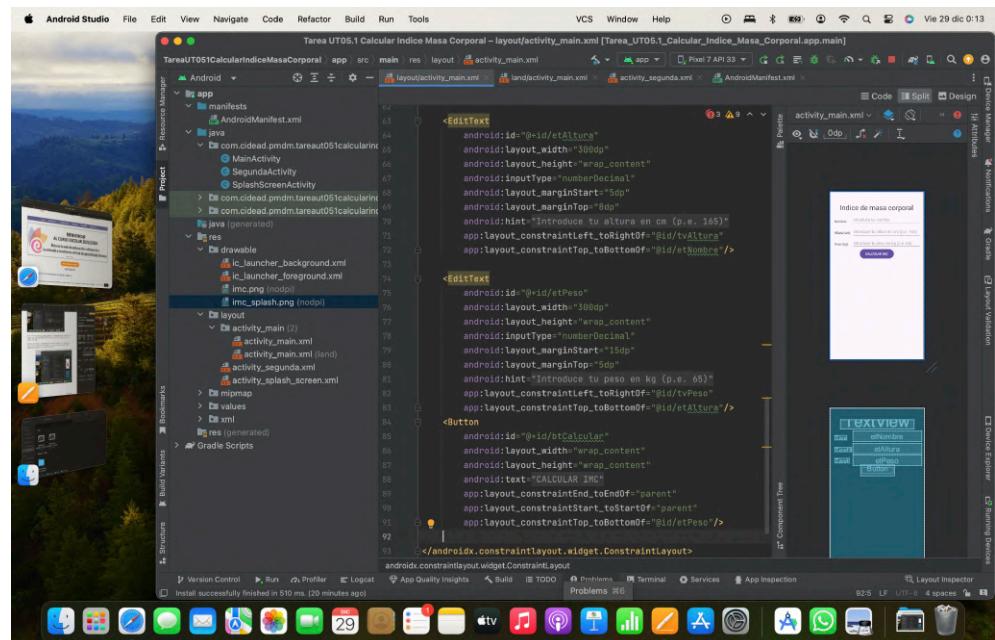
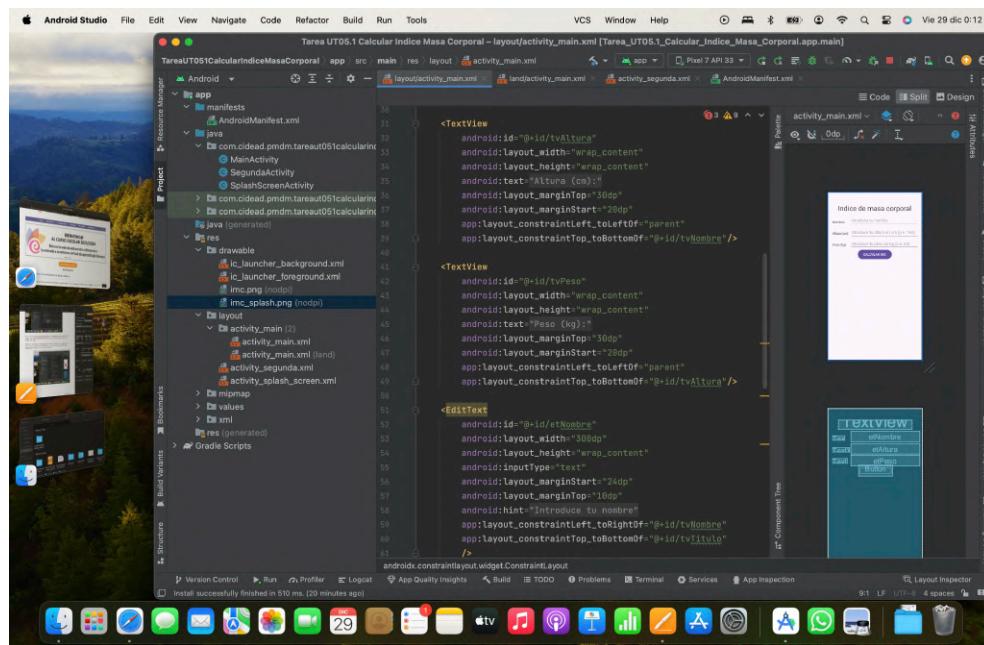
El primer edittext se situara al lado del textview nombre, contendrá un hint co la frase "Introduce tu nombre", dispondrá de un margen de 10 en su parte superior y de 24 en su lado izquierdo.

El segundo edittext sera como el anterior pero estará situado al lado del textview "Altura" y su hint mostrara "Introduce tu altura en cm (p.e. 165) .

El tercer edittext sera como los dos anteriores pero estará situado al lado de textvie "Peso", contendrá un hint "Introduce tu peso en kg (p.e. 65) " .

El activity_main para pantalla apaisada será idéntico al anterior pero adaptando los elementos a dicho tamaño





El activity_segunda contendrá cinco textView, un imagenviwe y un button.

El primer textView contendrá la frase "Indice de masa corporal", con una letra de 30 e irá ubicado en la parte superior de la pantalla.

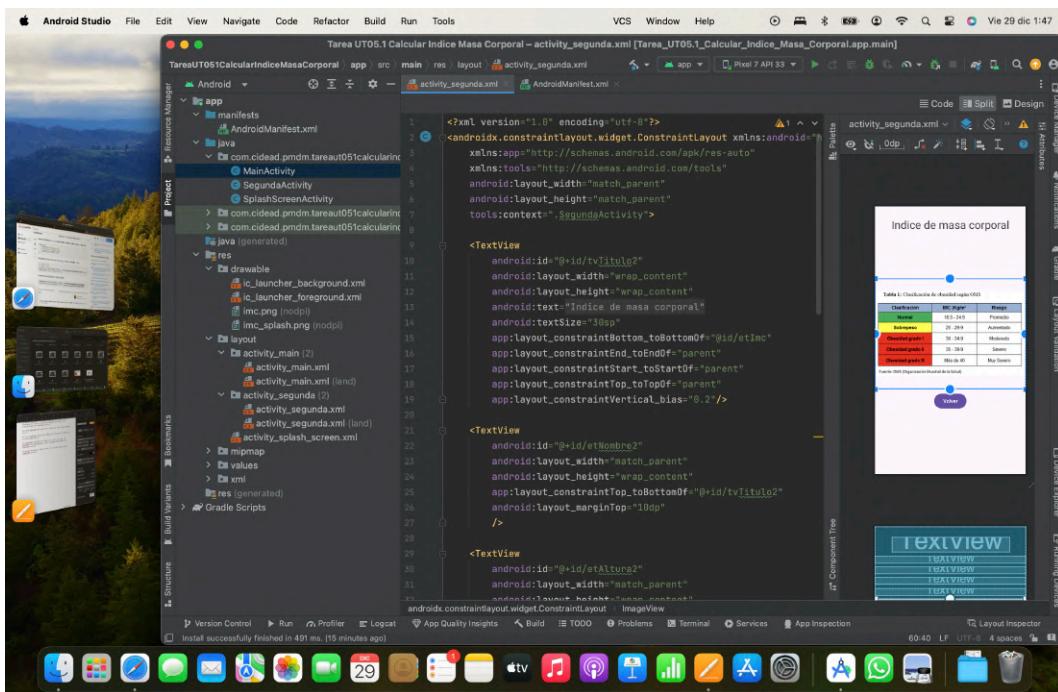
El segundo textView no contendrá ningún dato por defecto, ya se lo añadiremos al clicar al botón de la primera actividad e irá colocado debajo del textView anterior.

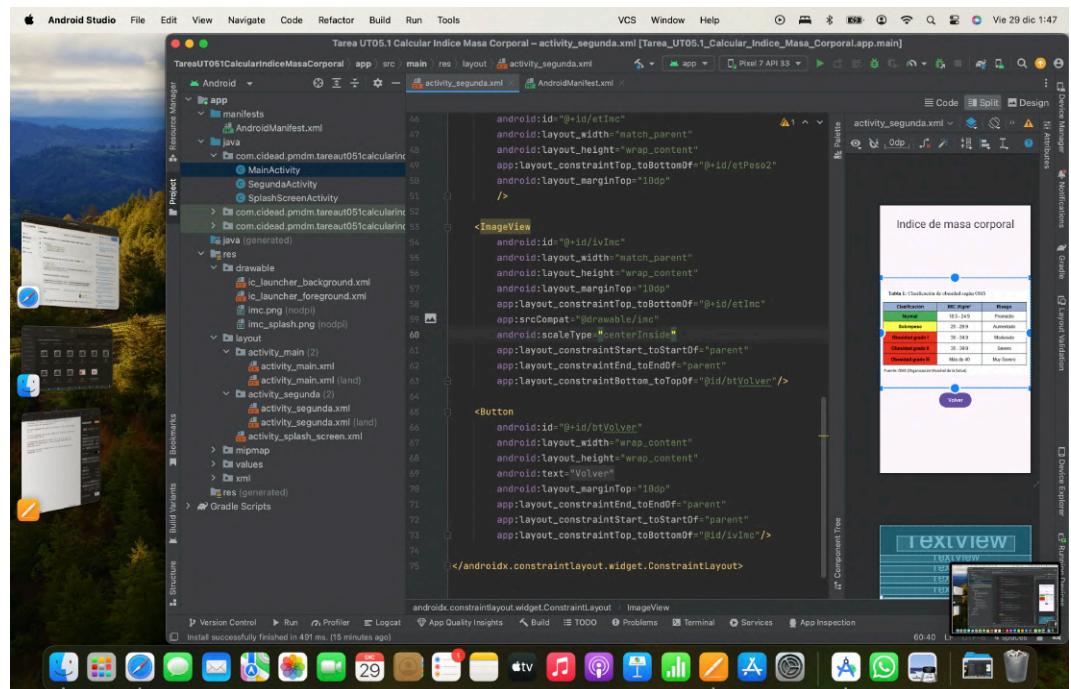
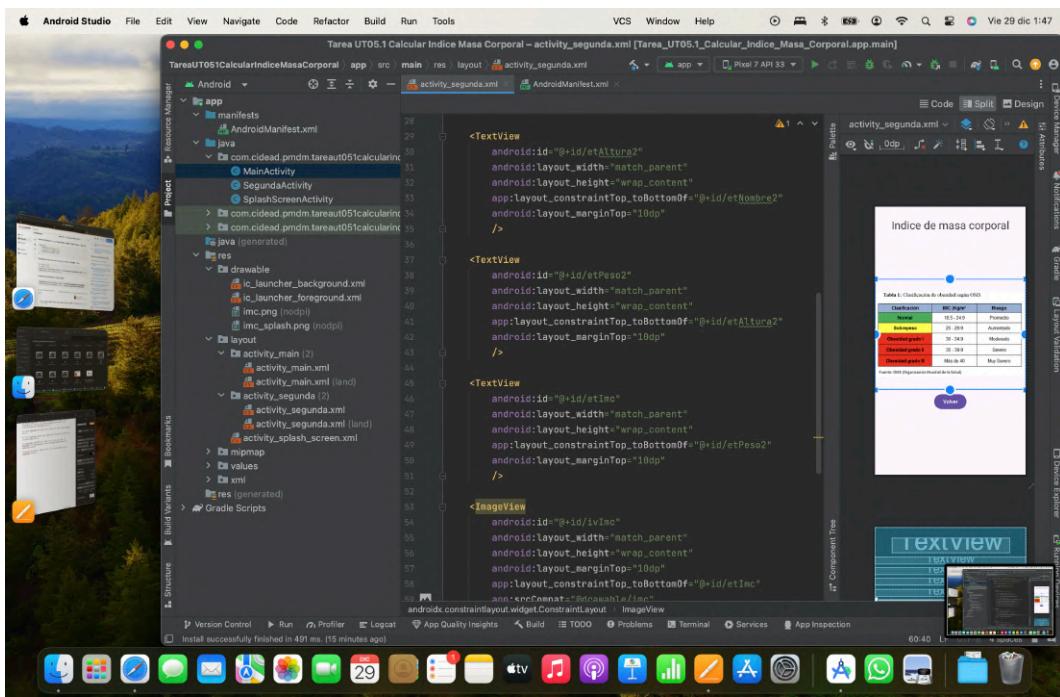
El tercer textView será idéntico al anterior e ira colocado debajo del el.

El cuarto textView también será idéntico al anterior y también ira colocado debajo de el.

La imagenviwe contendrá una imagen con la tabla del IMC, el scaleType será centerInside y estará ubicada justo debajo del textView anterior.

El button contendrá el texto volver e irá ubicado debajo de la imagenviwe anterior.

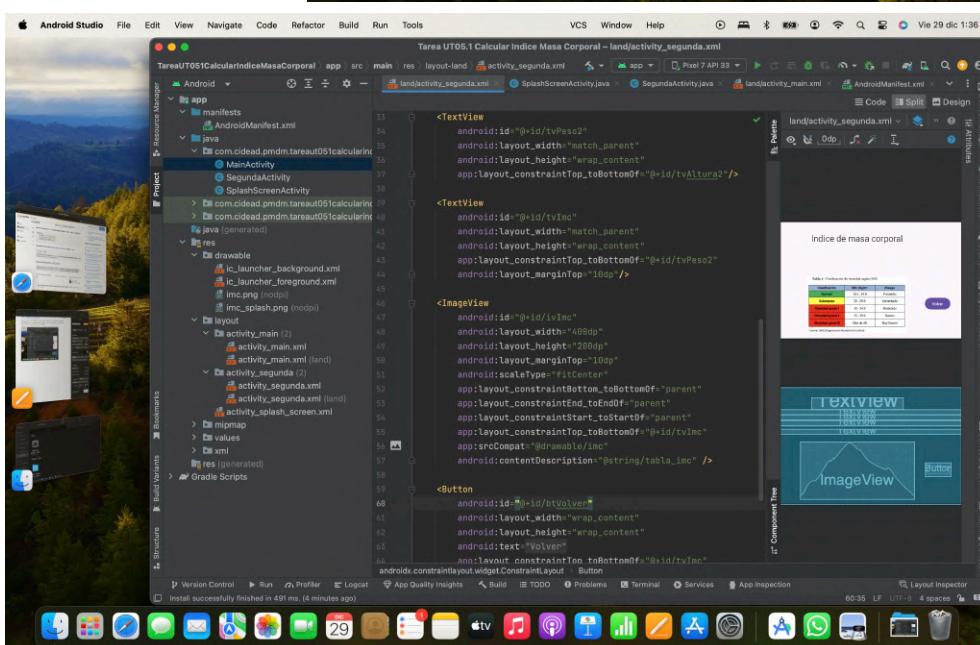
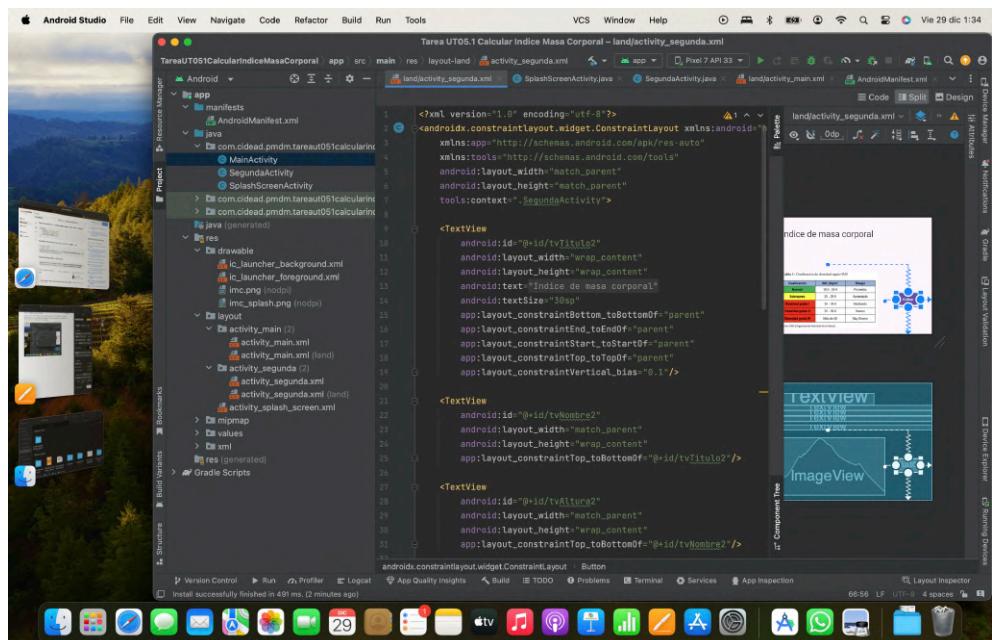
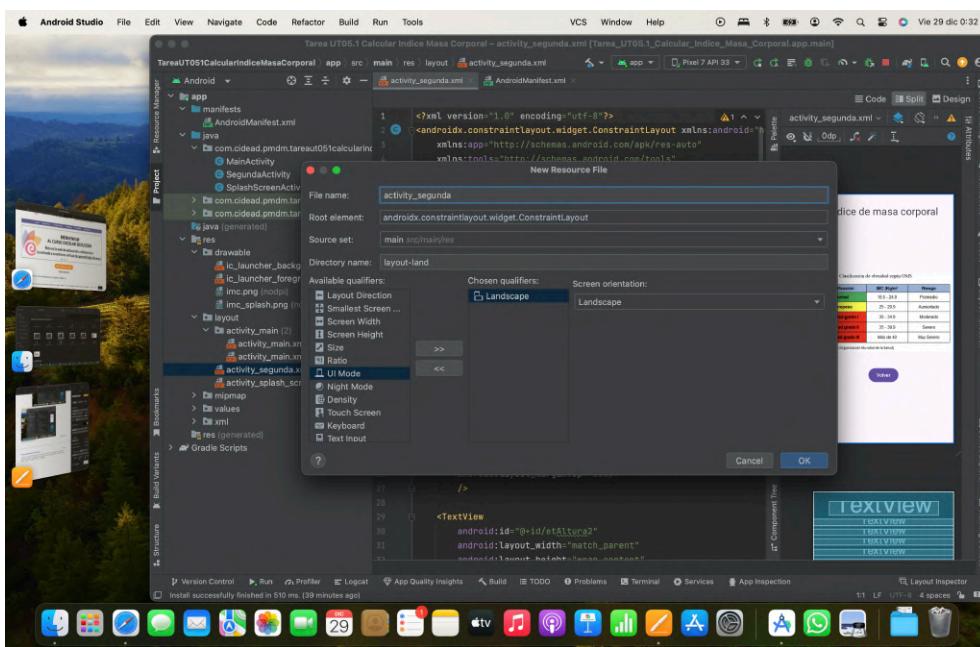


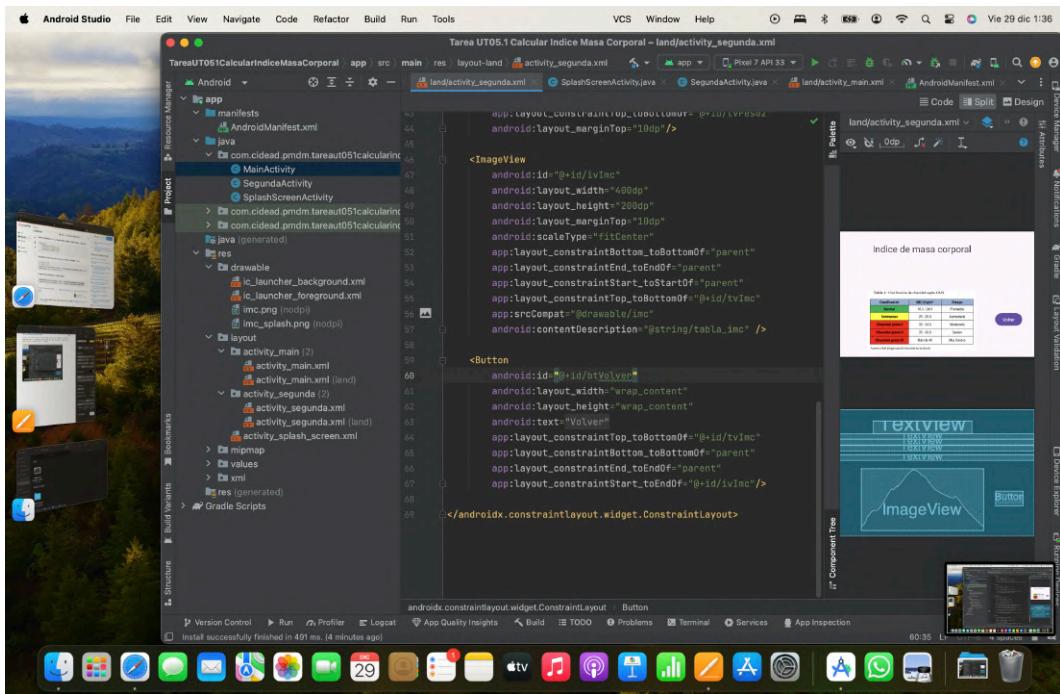


A esta altura me acabo de dar cuenta que no he creado el diseño apaisado para la segunda actividad, así que procedo a crearlo al igual que el diseño para la primera.

Contendrá los mismo componentes que para su versión normal.

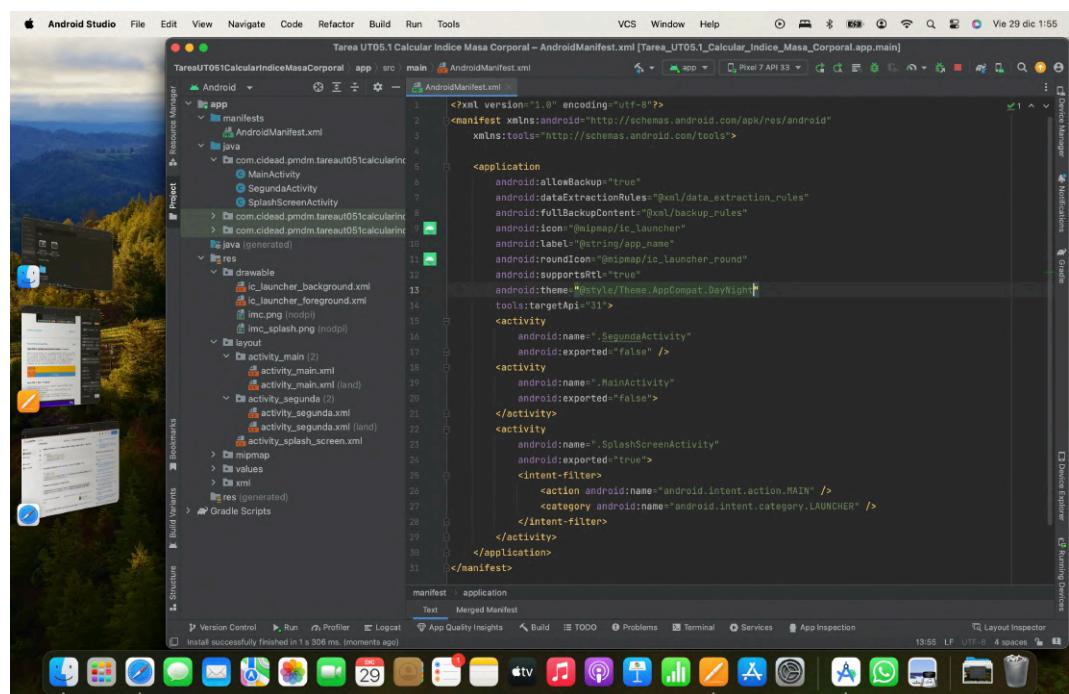
Reajustaremos la ubicación y el tamaño de los componentes y mostrare su código en las siguientes imágenes.





Por último en el androidManifest cambiaremos el intent-filter con el action.main y el launcher a la actividad del splashscreen, a la cual le cambiaremos a true el exported.

En las actividades de MainActivity y SegundaActivity el exported lo pasaremos a false.

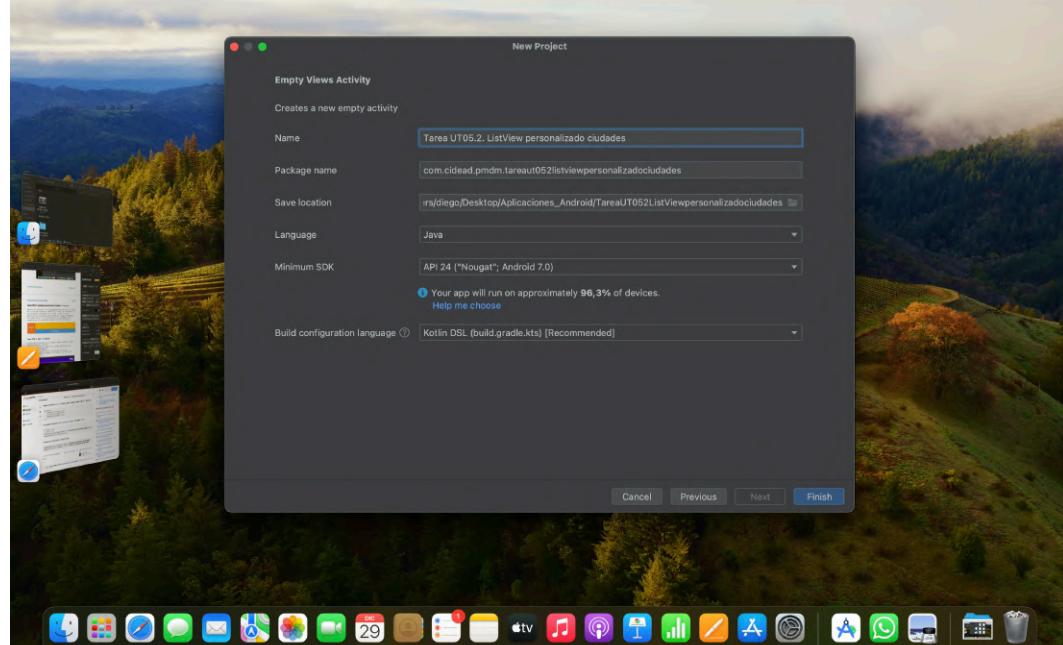
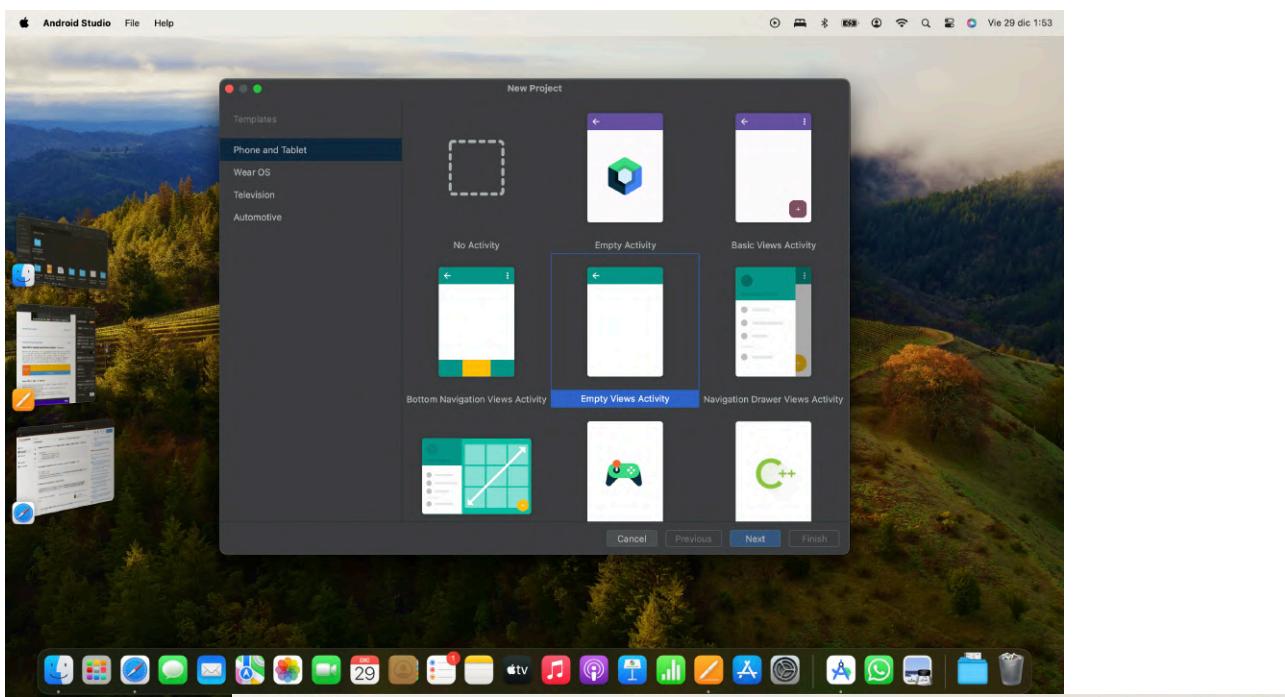


Tarea UT05.2. ListView personalizado ciudades. (4 puntos)

Realiza una aplicación con un ListView personalizado con un mínimo de 4 filas que contenga el nombre de la ciudad, la comunidad a la que pertenece y una imagen de la ciudad. Cuando el usuario seleccione una ciudad tiene que mostrar en un toast el monumento más significativo de la ciudad. Cada fila del ListView estará compuesta por los siguientes elementos con esta estructura:

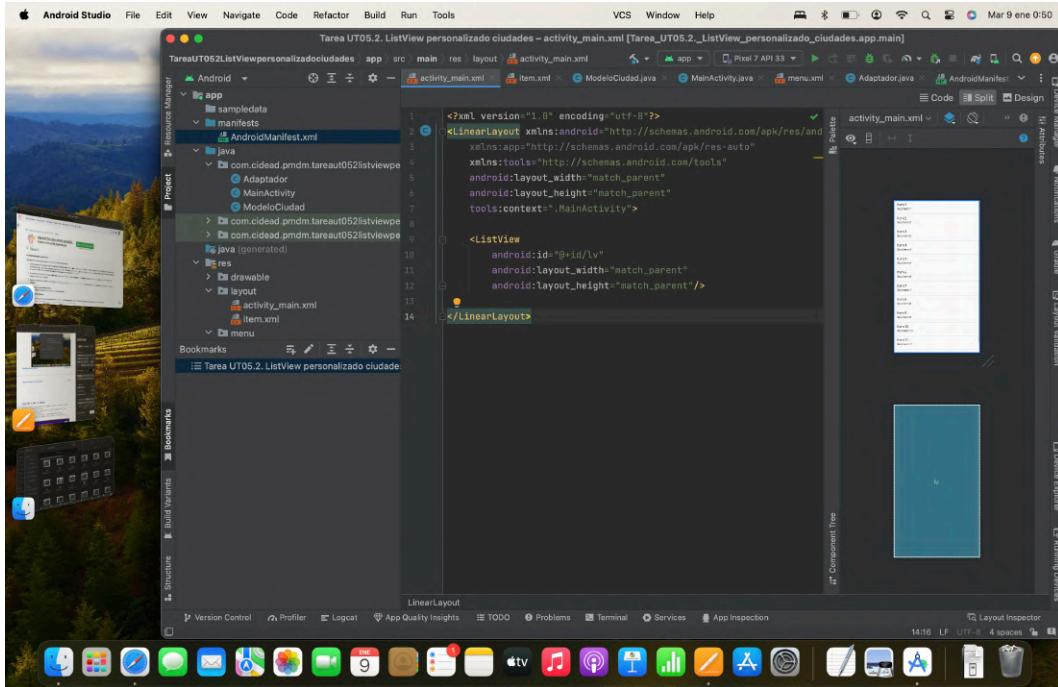


En primer lugar crearemos el proyecto al cual llamaremos Tarea UT05.2 ListView personalizado ciudades.

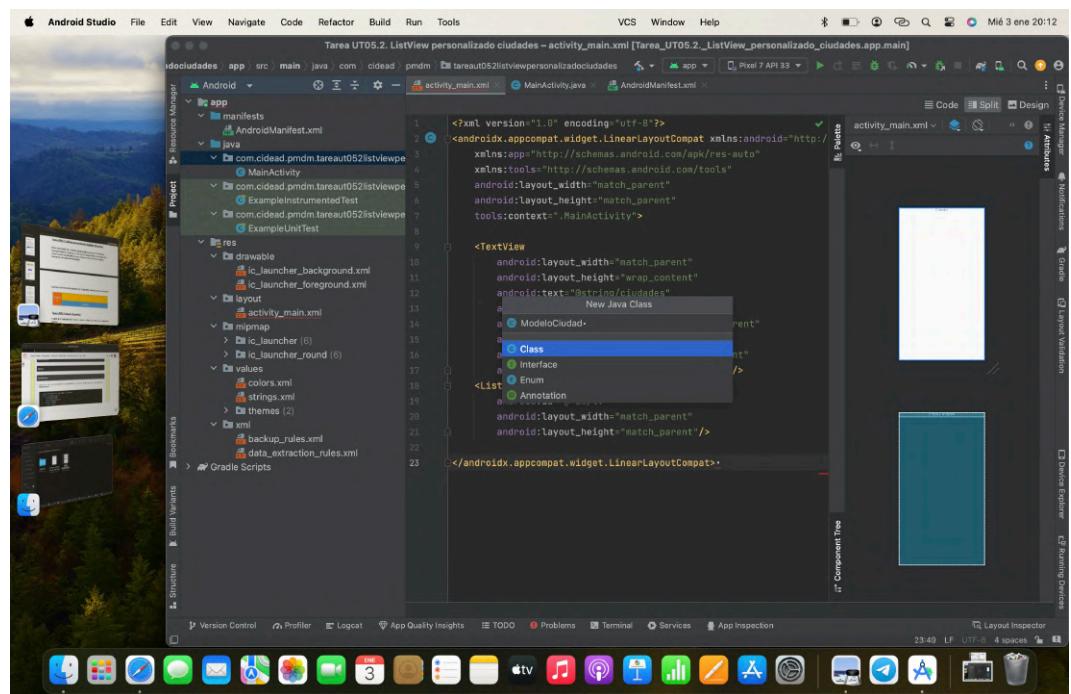


Una vez creada la aplicación nos dirigimos a AndroidManifest y cambiamos el tema que nos aparece por el tema DayNight para que así nos aparezca el título de la aplicación.

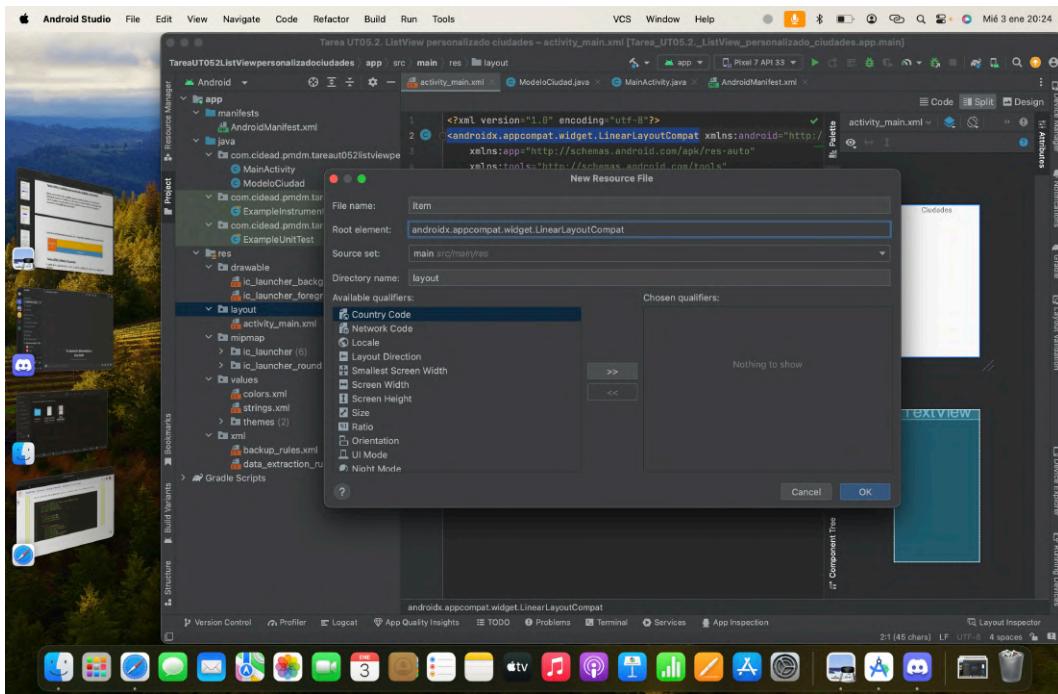
A continuación nos dirigiremos al activity_main, en el cual elegiremos LinearLayout e incluiremos en él un listview



Posteriormente crearemos la clase ModeloCiudad.

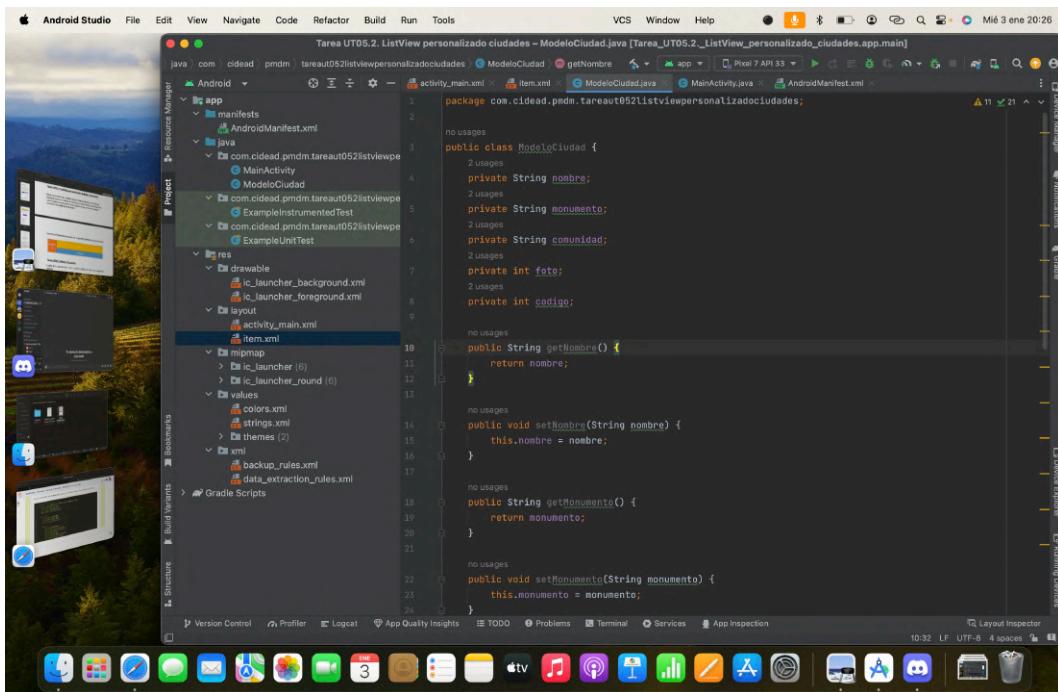


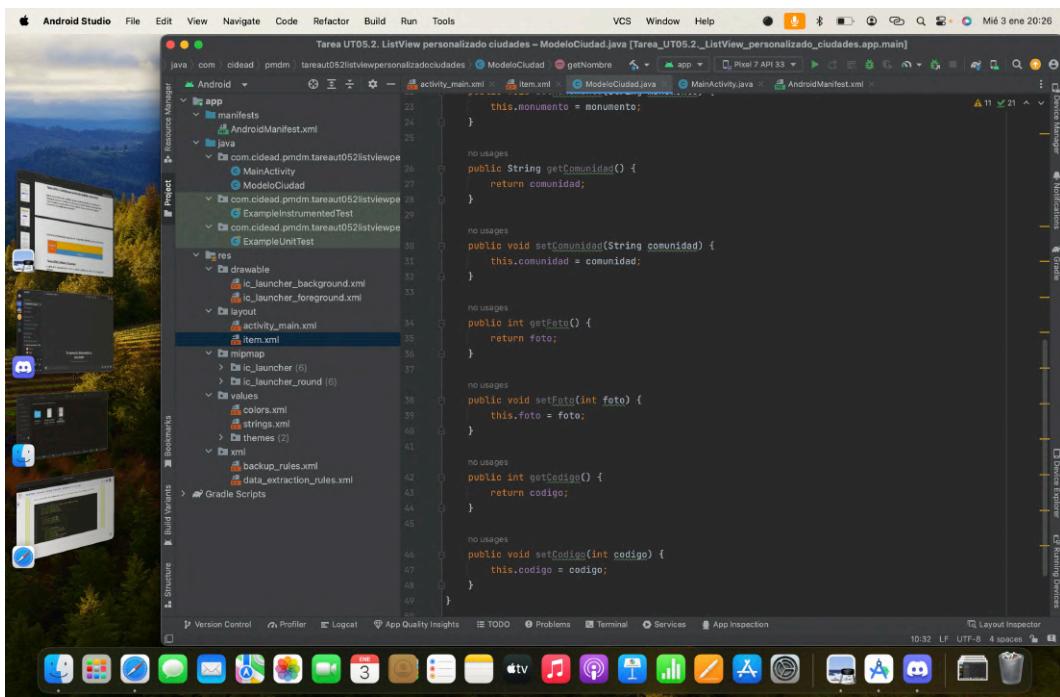
Tras ello crearemos un archivo xml llamado item que estará dentro de res - layout.



Volviendo a la clase `ModeloCiudad`, crearemos su código.

Constara de tres atributos de tipo `String` llamados `nombre`, `monumento` y `ciudad` y otros dos atributos de tipo `int` llamados `foto` y `código` y los métodos `getter` and `setter`.





Tras ello crearemos la clase adaptador.

Dicha clase constara de dos atributos, uno de tipo context llamado miContexto y otro de tipo arraylist de ModeloCiudad llamado miArrayList.

También tendrá el constructor con los dos atributos y los métodos getCount, getItem, getItemId y getView.

El método getCount retornara el tamaño de miArrayList.

El método getItem retornara el objeto donde se encuentre la variable i que le pasemos.

El método getItemId retornara el código del item donde se encuentre la variable i que le pasemos.

Y por último, el método getView retornara un View.

Dentro del método getView crearemos un layoutinflater de miContexto.

Mediante el método setText establecerá el nombre y la comunidad.

Mediante el método setImageResource establecerá la foto.

En las siguientes imágenes podemos ver la creación y el código de dicha clase.

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** Android Studio, File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Title Bar:** Tarea UT05.2. ListView personalizado ciudades – MainActivity.java [Tarea_UT05.2._ListView_personalizado_ciudades.app.main].
- Project Structure:** Shows the project hierarchy under 'app':
 - manifests
 - java
 - com.cidead.pmdm.tarea052listviewpe
 - MainActivity
 - ModeloCiudad
 - com.cidead.pmdm.tarea052listviewpe
 - ExampleInstrumentedTest
 - ExampleUnitTest
 - res
 - drawable
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml
 - layout
 - activity_main.xml
 - item.xml
 - mipmap
 - ic_launcher (6)
 - ic_launcher_round (6)
 - values
 - colors.xml
 - strings.xml
 - themes (2)
 - xml
 - backup_rules.xml
 - data_extraction_rules.xml
 - Gradle Scripts
- Code Editor:** Displays the MainActivity.java code. A context menu is open at the bottom right, with 'Class' selected.

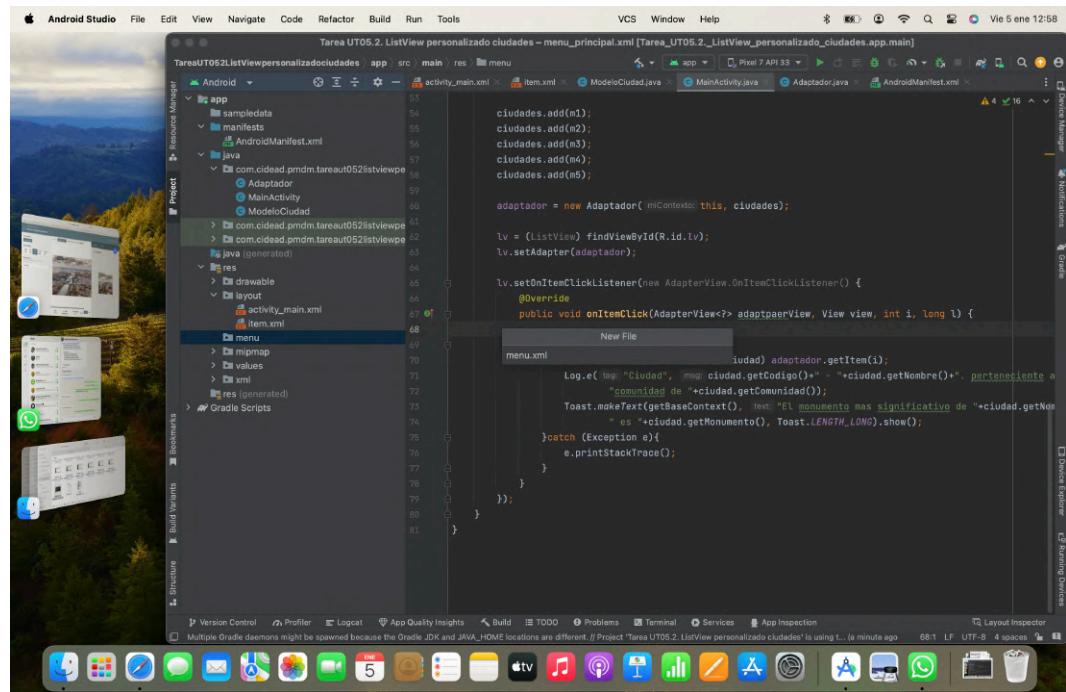
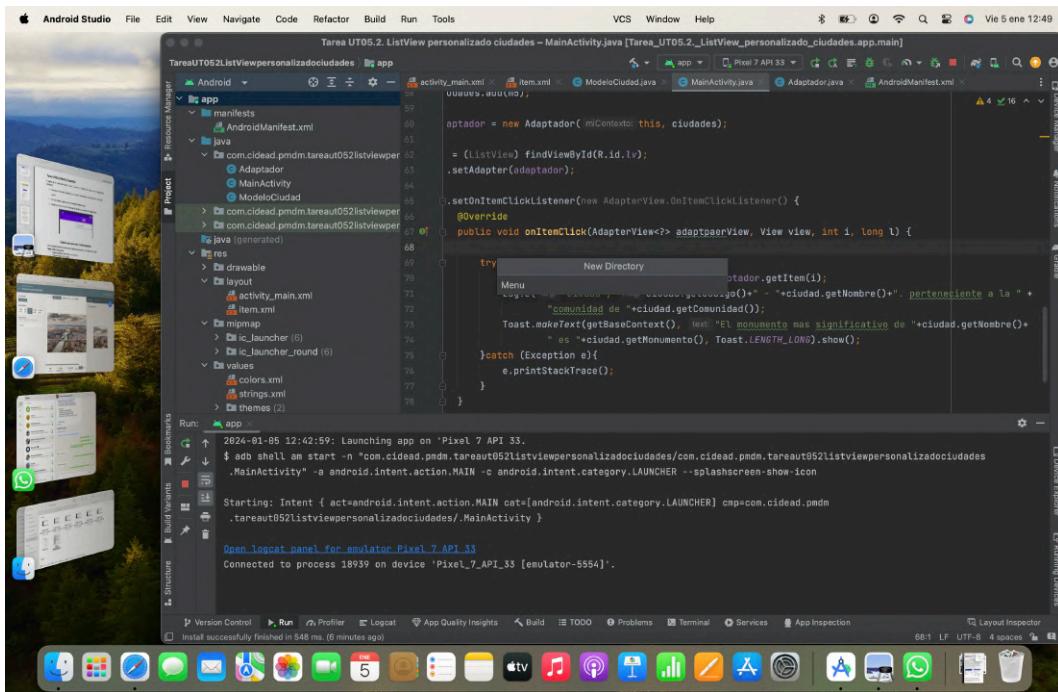
```
1 package com.cidead.pmdm.tarea052listviewpersonalizaciudades;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12
13 }
```
- Bottom Right Context Menu:** Shows options for creating a new Java Class:
 - Adaptador
 - Class** (selected)
 - Interface
 - Enum
 - Annotation
- Sidemenu:** Includes 'Recent Projects', 'Recent Devices', 'Build Variants', 'Structure', and 'Device Explorer'.
- Bottom Status Bar:** Pixel 7 Pro 33°, Mi 6 3 e 20:43.

A screenshot of the Android Studio interface. The top navigation bar includes 'File', 'Edit', 'View', 'Navigate', 'Code', 'Refactor', 'Build', 'Run', 'Tools', 'VCS', 'Window', and 'Help'. The title bar indicates the project is 'Tarea UT05.2. ListView personalizado ciudades - Adaptador.java [Tarea_UT05.2_ListView_personalizado_ciudades.app.main]' and the file being edited is 'Adaptador.java'. The left sidebar shows the project structure with packages like 'app', 'java', 'res', and 'Gradle Scripts'. The main editor window displays Java code for an adapter class. A modal dialog titled 'Select Methods to Implement' is open, listing methods from the 'android.widget.Adapter' interface: getCount(), getItem(position), getItemId(position), and getView(position, convertView, parent). At the bottom of the modal are three checkboxes: 'Copy JavaDoc', 'Generate missed JavaDoc', and 'Insert @Override', with the last one checked. Buttons for 'Cancel' and 'OK' are also present.

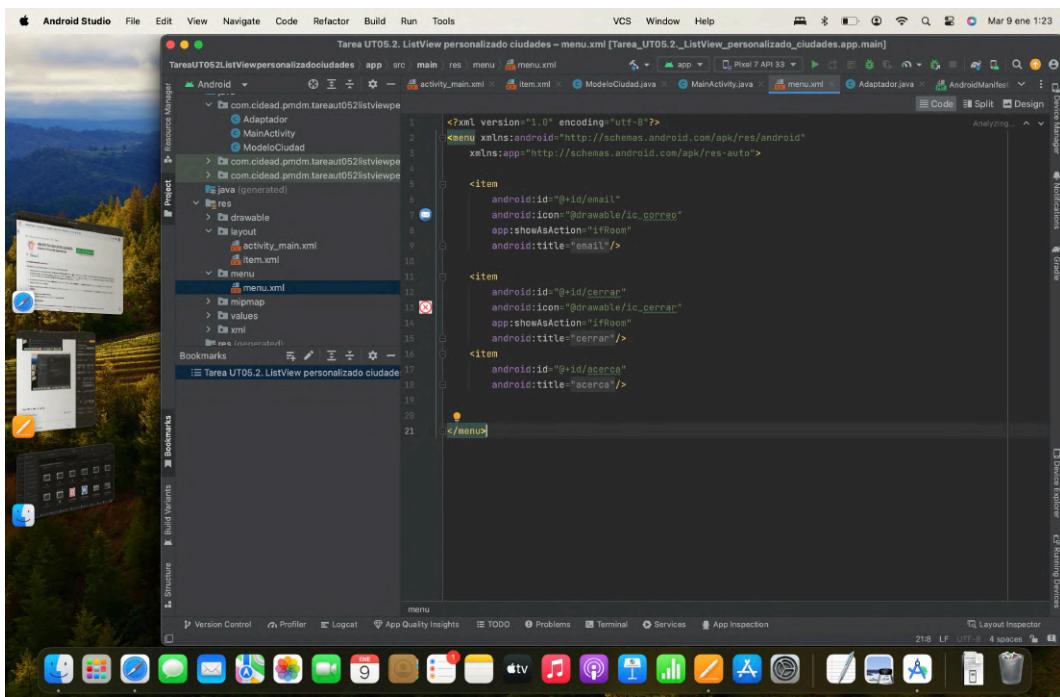
The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "Tarea UT05_2". It contains a main Java package "com.cidcad.pmdm.tarea052.listviewpersonalizado" which includes files like "Adaptador.java", "MainActivity.java", and "ModeloCiudad.java".
- Code Editor:** The current file is "Adaptador.java". The code defines a class "Adaptador" that extends "BaseAdapter". It uses an ArrayList of "ModeloCiudad" objects and various overridden methods to handle item counts, get items, get IDs, and create views.
- Toolbars and Menus:** Standard Android Studio menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, Plugins, Window, Help, and a Pixel 3 API 33 specific toolbar are visible.
- Side Panels:** The Project, Properties, and Tools panels are open on the right side.
- Bottom Navigation:** Includes tabs for Version Control, Profiler, Logcat, App Quality Insights, TODO, Problems, Terminal, Services, and App Inspection.
- Bottom Bar:** Shows the current device as an iPhone X, the time as 34:47, and the system status as LF- UTF-8 4 spaces.

A continuación crearemos un nuevo directorio llamado menú y dentro de él un archivo xml al cual también llamaremos menú.



El archivo menú constará de tres item y el código de podemos verlo en la siguiente imagen.

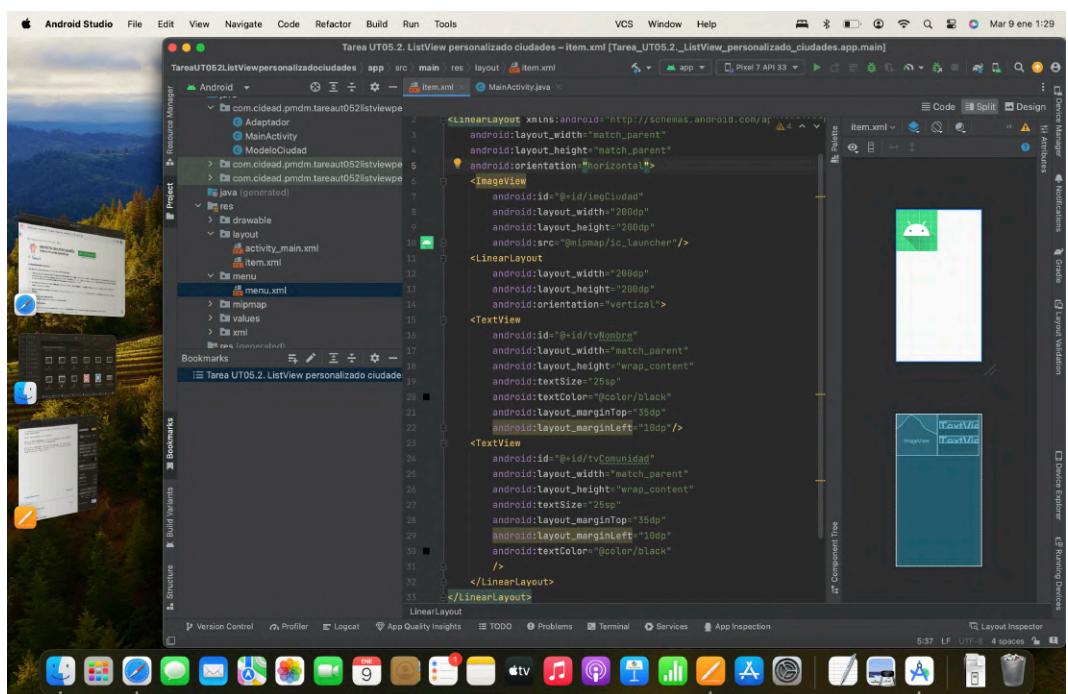


Lo siguiente que haremos será crear el código de la clase item creada anteriormente.

Dicha clase contendrá un LinearLayout.

Dentro de este LinearLayout estarán una ImageView, que contendrá la imagen de la ciudad y otro LinearLayout.

Este último LinearLayout contendrá dos TextView, uno para mostrar el nombre de la ciudad y otro para mostrar la comunidad autónoma donde se encuentra dicha ciudad.



Por último crearemos el código del mainActivity.

Dicha clase constará de dos atributos, uno de tipo ListView llamado lv y otro de tipo Adaptador llamado adaptador.

Contendrá el método onCreate y dentro de él crearemos un ArrayList de tipo ModeloCiudad llamado ciudades.

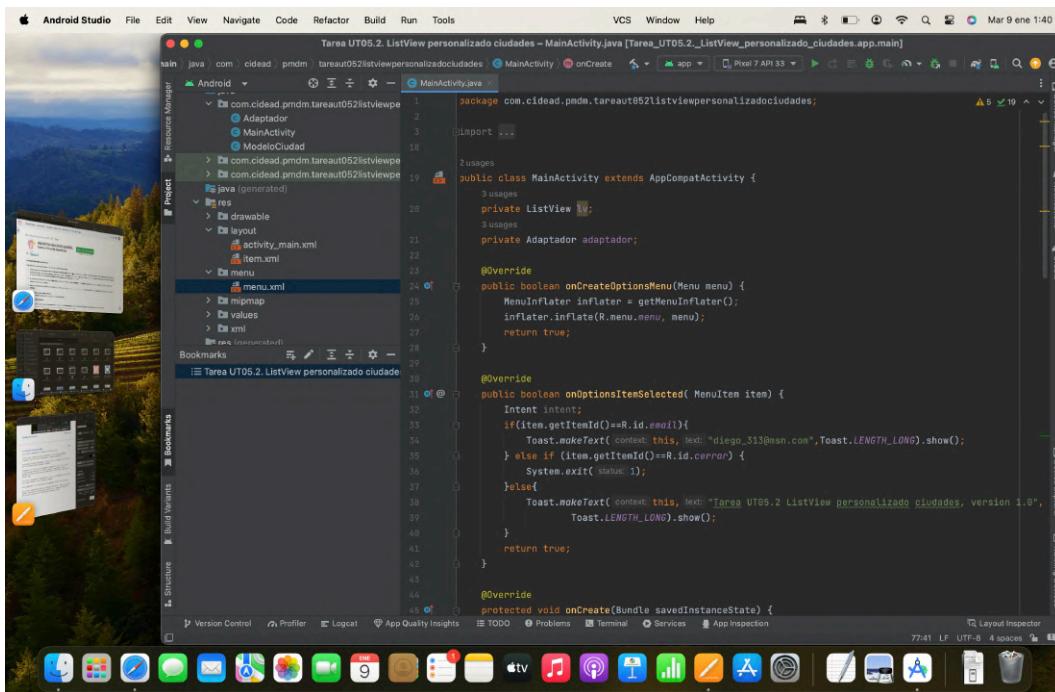
Crearemos en el las cinco ciudades llamadas m1, m2, m3, m4 y m5, instanciaremos en ella su código, nombre, comunidad, foto y monumento para posteriormente, añadirlas al ArrayList mediante el método add.

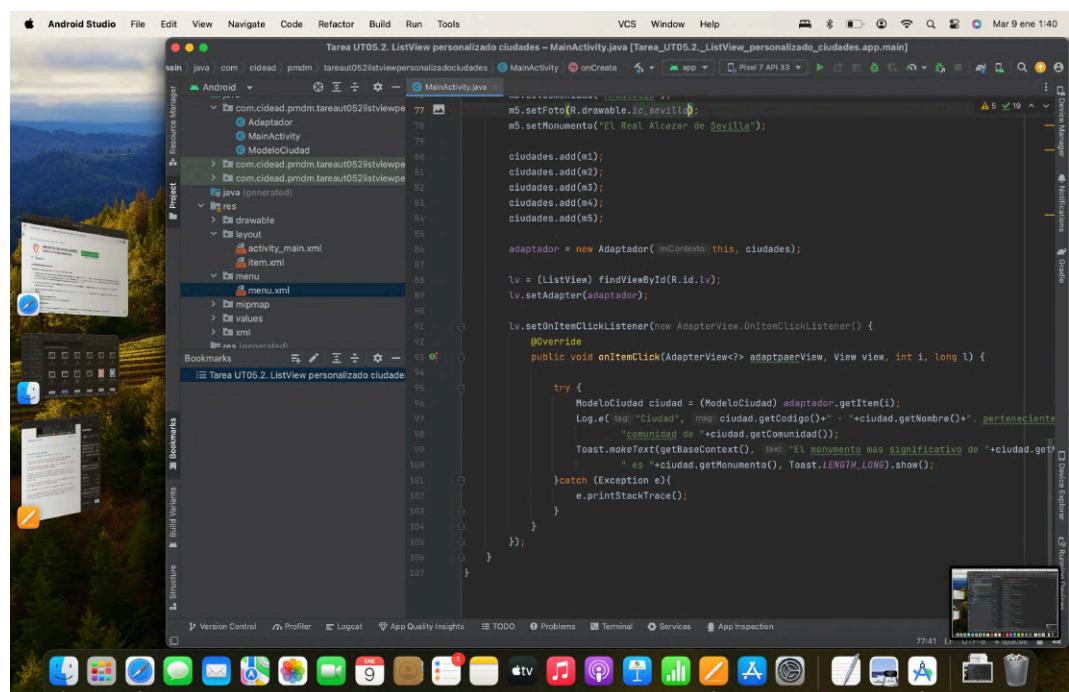
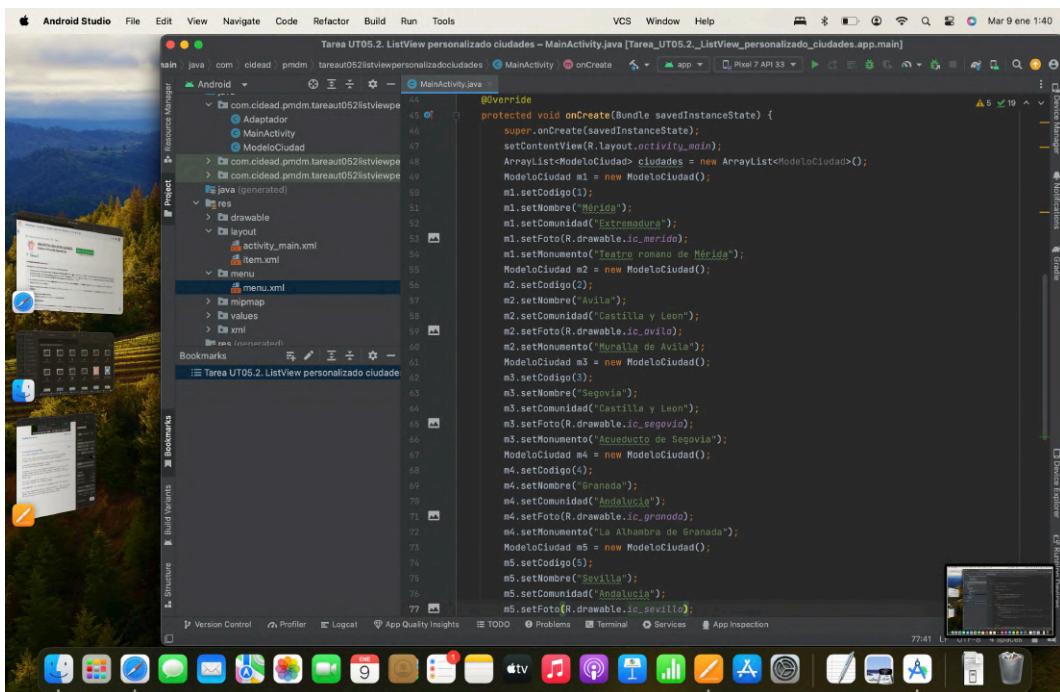
Tras ello crearemos un adaptador, al cual le pasaremos el contexto en el que estamos y las ciudades creadas.

Este adaptador se lo pasaremos al ListView creado anteriormente.

Por último sobreescribiremos el método onItemClic. Dentro de dicho método crearemos el código para mostrar en un Toast el monumento mas significativo de la ciudad donde habíamos clic.

Este código estará envuelto en un try - catch para capturar las excepciones.

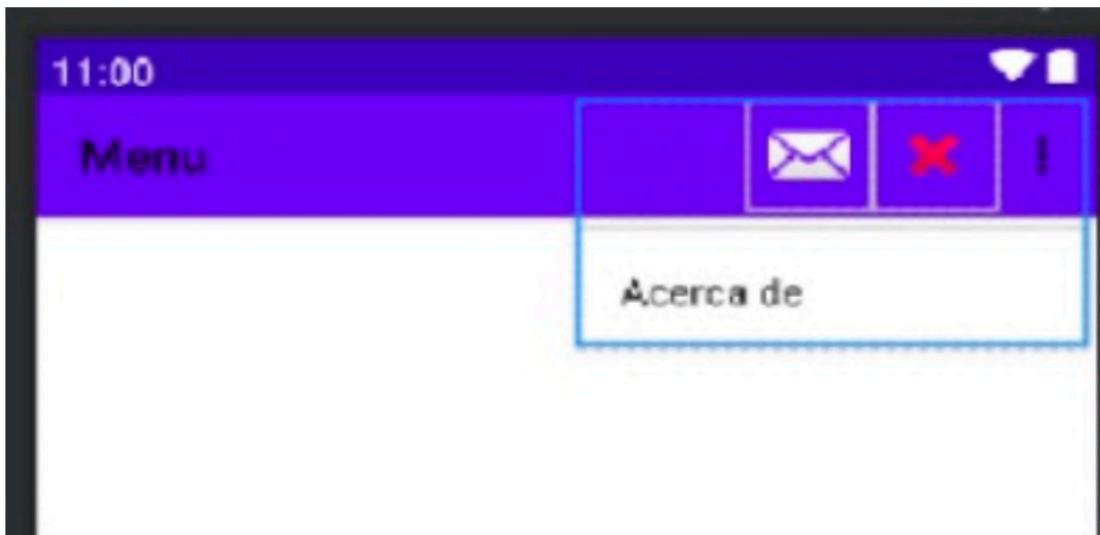




Tarea UT05.3. Menú. (2 puntos)

A partir de la aplicación de la tarea 2, vamos a añadir un menú con las siguientes opciones:

- Contacto: Mostrará un Toast con tu correo electrónico. Se mostrará un ícono de email.
- Cerrar: Cierra la aplicación. Se mostrará como ícono.
- Acerca de: Mostrará un Toast con el nombre y versión de la aplicación.



En primer lugar duplicaremos el proyecto con el nombre nuevo.

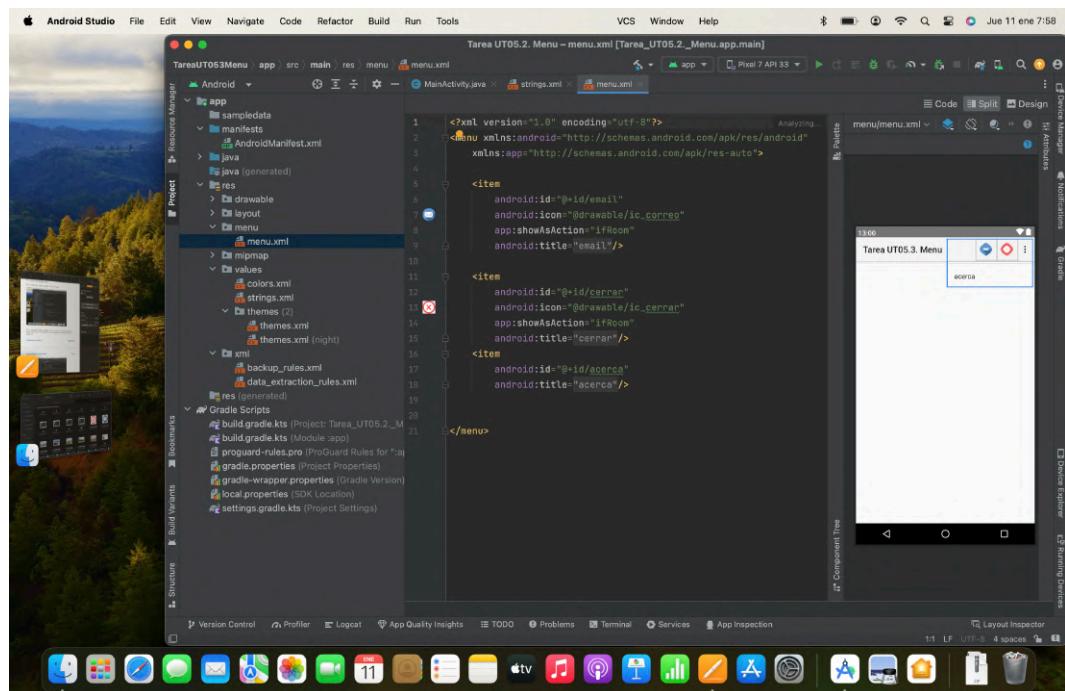
Posteriormente crearemos un nuevo directorio llamado menú y dentro crearemos un archivo xml llamado menú.

Este archivo contendrá tres item contenidos dentro de dos etiquetas de menú.

El primer item será para el email y contendrá la imagen que tenemos preparada para ello (llamada ic_correo) en la carpeta drawable.

El segundo item será para cerrar la aplicación, al igual que el item anterior, contendrá la imagen que tenemos preparada para ello (llamada ic_cerrar) en la carpeta de drawable.

El tercer item se llamará acerca, en este caso no contendrá imagen.



Posteriormente nos dirigiremos al MainActivity.

En el sobreescribiremos los métodos onCreateOptionsMenu y onOptionsItemSelected.

En el primero de los casos, onCreateOptionsMenu, le pasaremos un Menu llamado menú.

Dentro de este método crearemos un MenuInflater llamado inflater, al cual le aplicaremos el método inflater pasando el menú y retornaremos verdadero.

En el segundo caso, onOptionsItemSelected, le pasaremos un MenuItem llamdo item.

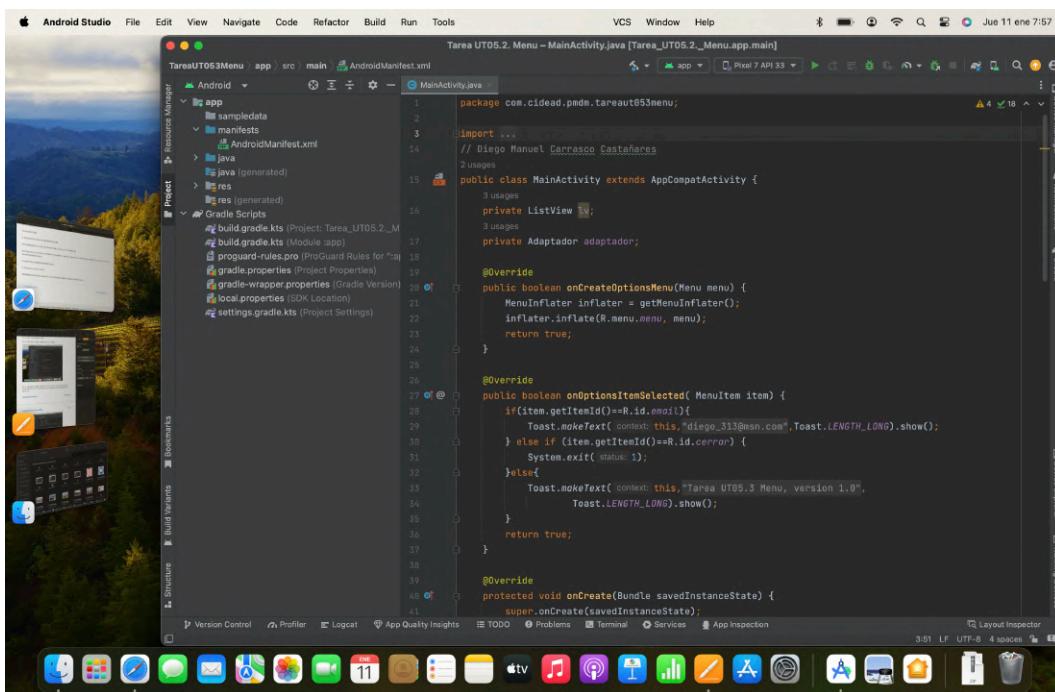
Crearemos un if que comprobara si se ha pulsado el botón de email, si es así mostrara el correo electrónico mediante un toast.

En caso de pulsar el botón de cerrar la aplicación se cerrara mediante el método exit de System.

Si por el contrario no pulsamos ninguno de los dos anteriores entenderá que hemos pulsado el botón de "acerca de" y mostrara en un toast el nombre de la tarea y la versión de la app.

Por ultimo retornaremos true.

En la siguiente imagen podemos ver el código.

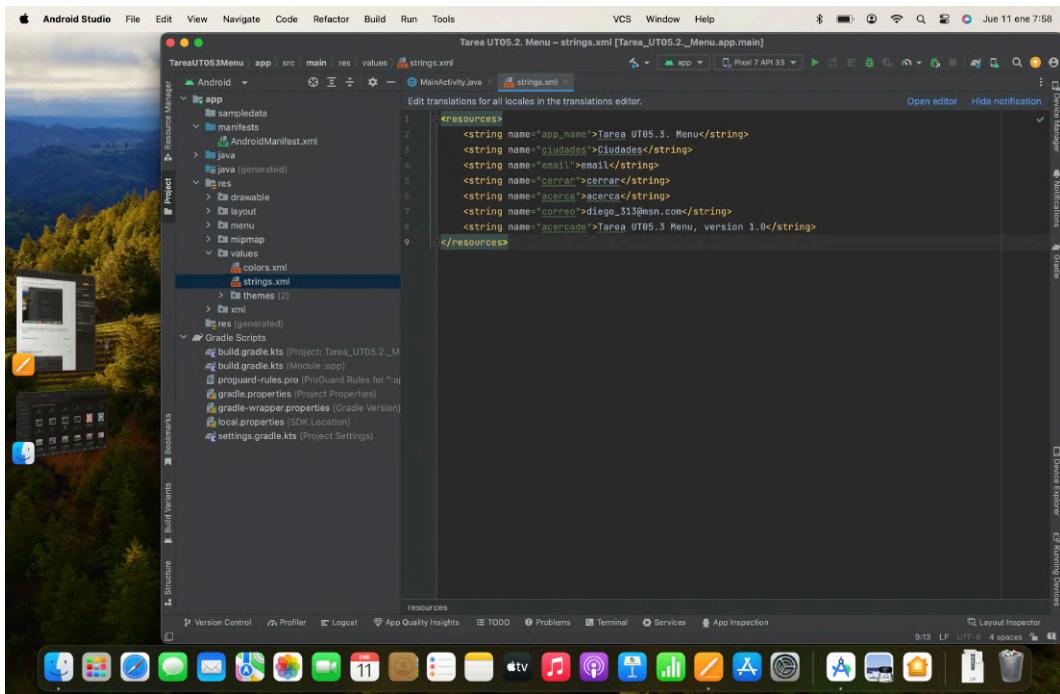


```

Tarea UT05.2. Menu - MainActivity.java [Tarea_UT05.2_Menu.app.main]
Android Studio 4.1.1
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Jue 11ene 7:57
package com.cidead.pndm.tareaut05menu;
import ...
// Diego Manuel Carrasco Castañares
public class MainActivity extends AppCompatActivity {
    private ListView adaptador;
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getItemId()==R.id.email){
            Toast.makeText(context, "diego_313@msn.com", Toast.LENGTH_LONG).show();
        } else if (item.getItemId()==R.id.cerrar) {
            System.exit(status: 1);
        } else{
            Toast.makeText(context, "Tarea UT05.3 Menu, version 1.0",
            Toast.LENGTH_LONG).show();
        }
        return true;
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

En la siguiente imagen podemos ver el archivo string.xml donde puede apreciarse el contenido de los mensajes que mostramos por pantalla.



Y con esto damos por finalizada la tarea al completo.

Criterios de puntuación. Total 10 puntos.

La valoración total de la tarea es de 10 puntos repartidos del siguiente modo:

Tarea UT05.1 (4 puntos)

- Aspecto gráfico de la interfaz: 1 punto.
- Programación: 2,5 puntos.
- Correcta organización de los recursos en ficheros: colores, cadenas de textos, identificadores y aspecto. 0,5 puntos.

Tarea UT05.2 (4 puntos)

- Aspecto gráfico de la interfaz: 1 punto.
- Programación: 2,5 puntos.
- Correcta organización de los recursos en ficheros: colores, cadenas de textos, identificadores y aspecto. 0,5 puntos.

Tarea UT05.3 (2 puntos)

- Aspecto gráfico de la interfaz: 1 punto.
- Programación: 1 punto.

Recursos necesarios para realizar la Tarea.

- Ordenador con Android Studio instalado y suficientes recursos para ejecutar el emulador.
- Contenidos de la Unidad, muy importantes los ejercicios resueltos de la misma.

- ?
- Páginas web de los desarrolladores de los sistemas operativos para móviles.
- ?
- Imágenes de internet.

Consejos y recomendaciones.

- ?
- Lee atentamente el enunciado y asegúrate de haber entendido lo que has de hacer.
- ?
- Intenta reproducir en tu Android Studio los ejercicios resueltos previamente; estos te darán muchas claves para acometer la tarea.
- ?
- No dudes en comentarle a tu tutor o tutora cualquier duda que te pueda surgir.
- ?
- Envíasela a tu tutor o tutora a través del sistema establecido en la plataforma.
- ?
- Las capturas y el contenido de los ficheros deben aparecer en perfecto orden para que esté claro lo que deseas mostrar.
- ?
- Los diseños deben mostrarse lo más parecidos posibles a lo que se pide...recuerda que es una parte importante de la unidad.
- ?
- Debe llevarse a cabo sobre una versión actual de Android.

Indicaciones de entrega.

Una vez realizada la tarea elaborarás un documento PDF con los pasos realizados para la elaboración de cada una de los ejercicios de la tarea. Las capturas deben mostrar el fondo de pantalla y la fecha y hora de la barra de estado de tu sistema operativo para garantizar la originalidad del material. Muy importante que personalices tu nombre en la aplicación.

También se deberá entregar la carpeta comprimida del proyecto con cada uno de los ejercicios en formato .zip. En todos los Activity_main.java debe aparecer el nombre del autor de la entrega.

Entrega los archivos a través de la plataforma.

Se nombrarán siguiendo las siguientes pautas:

Apellido1_apellido2_nombre_PMDM05_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la cuarta unidad del MP PMDM, debería nombrar esta tarea como...

sanchez_manas_begona_PMDM05_Tarea