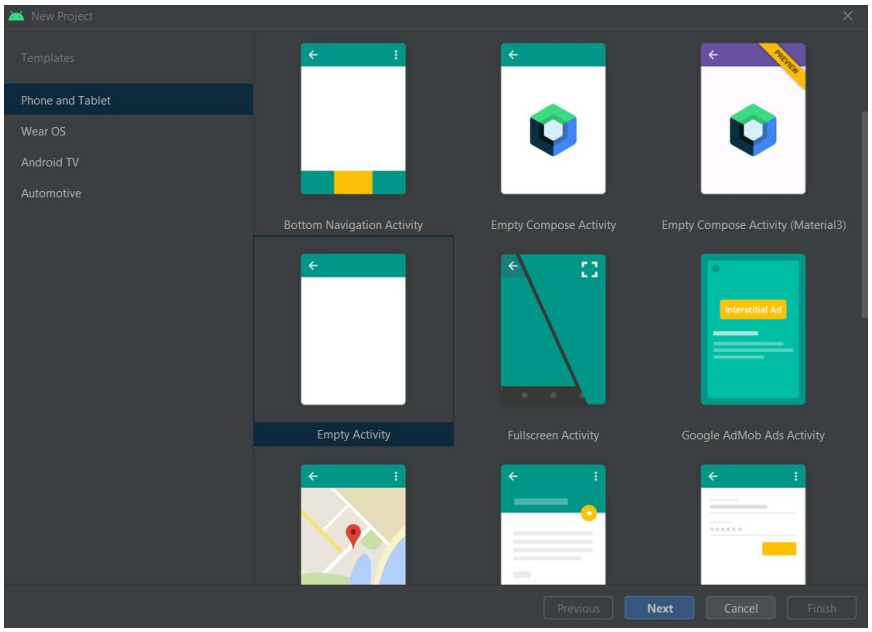
PROGRAMACIÓ MULTIMEDIA Y DISPOSITIVOS MÓVILES

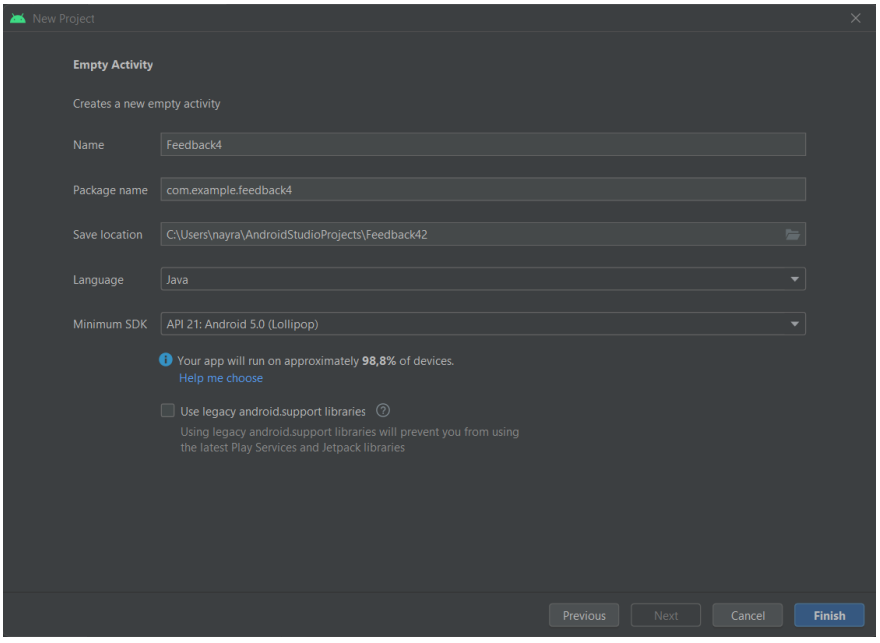
Nayra Infantes Tardón

Ejercicio feedback unidad 4

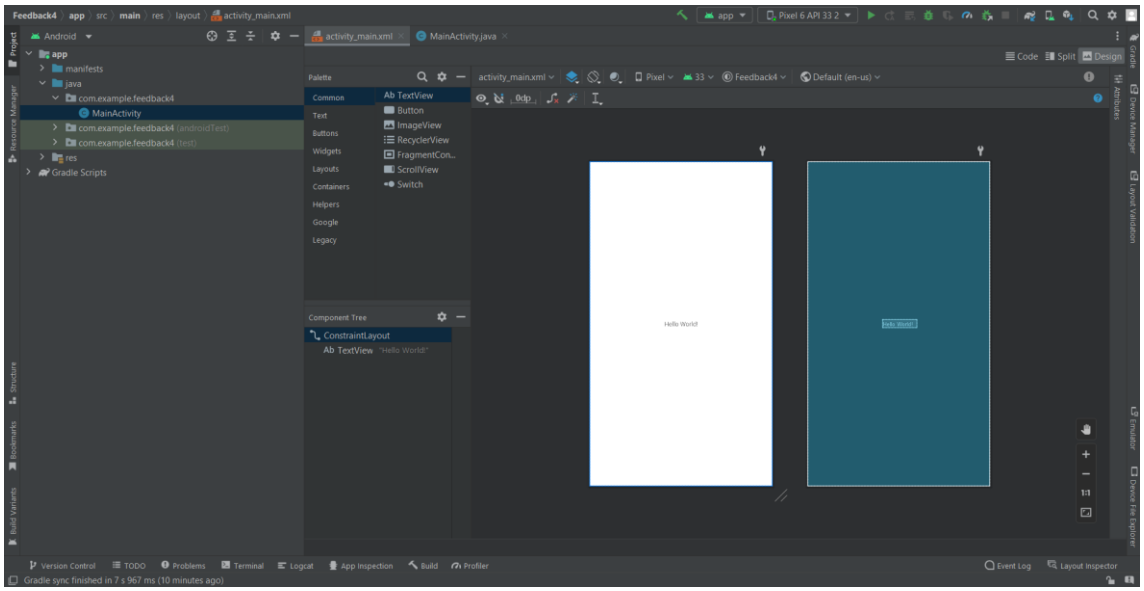
Realiza una app que actúe a modo de agenda y que guarde un nombre y un teléfono por persona introducida. Este guardado se producirá al mismo tiempo en una clase SharedPreferences y en un archivo de texto con el nombre que el usuario indique. Para realizar la consulta de los datos se visualizará el contenido completo del archivo de texto que se seleccione a tal efecto.

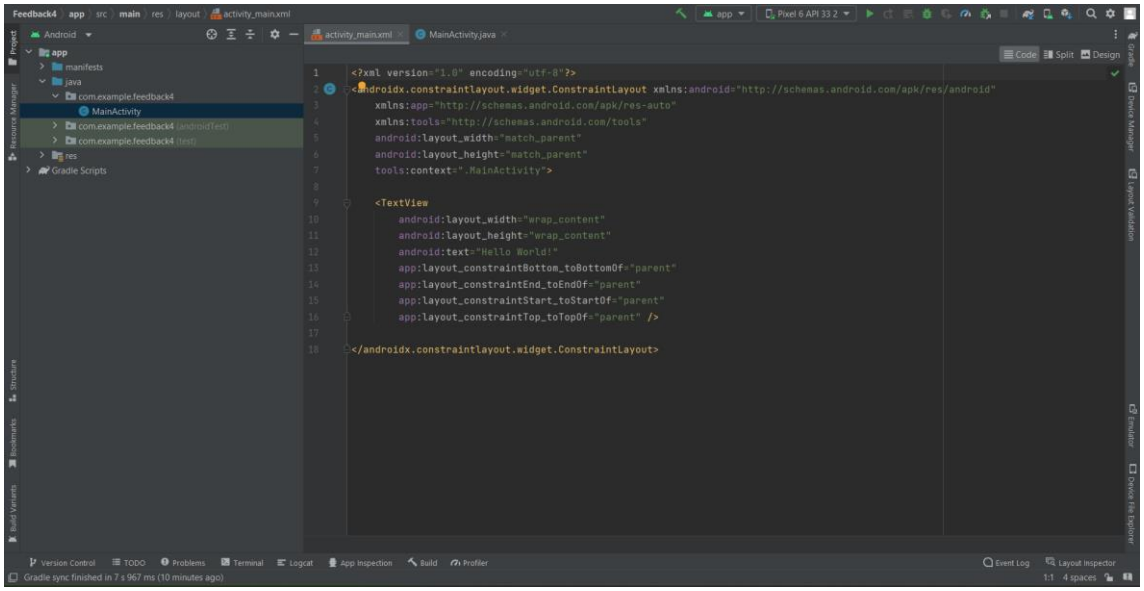
Para poder diseñar nuestra aplicación lo primero que debemos hacer es crear un nuevo proyecto, en nuestro caso se llamará Feedback4, partiremos de una actividad vacía para poder tener una base previa, y el lenguaje a utilizar será Java.



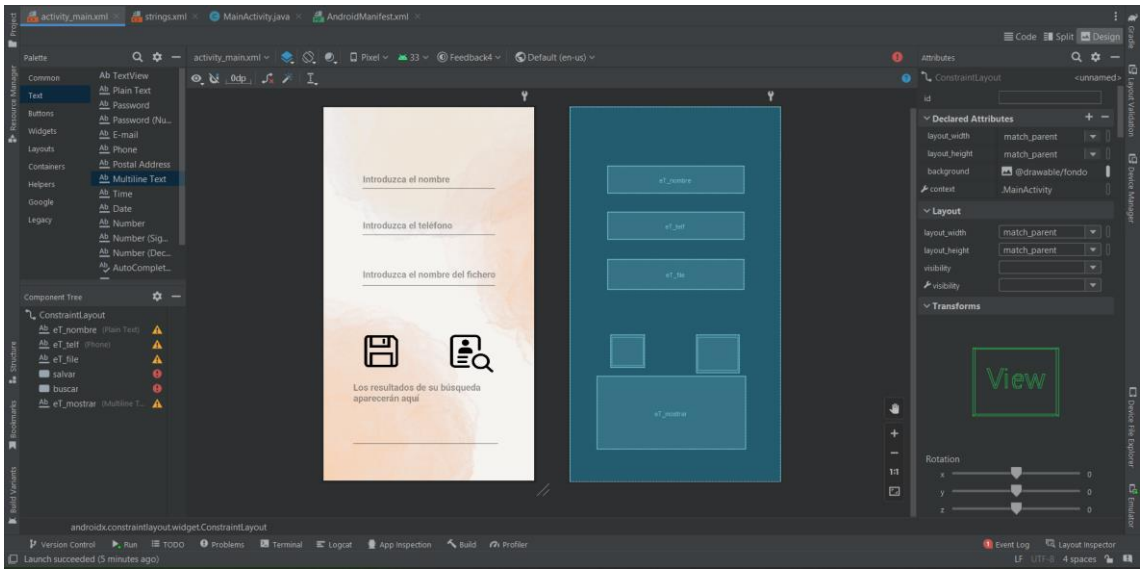


Gracias a la interfaz que nos proporciona Android Studio podemos crear nuestra agenda desde de una parte gráfica en la que funciona todo de una manera muy intuitiva o directamente a través de código en nuestro archivo activity_main.xml.





Para la realización de este ejercicio utilizaremos un ConstrainLayout, para poder posicionar de mejor manera el contenido, al que modificaremos el background.
Nuestra aplicación estará compuesta por cuatro EditText y dos botones en forma de icono que obedecerán a nuestros métodos “Guardar” y “Buscar”; a grandes rasgos, los EditText en Android se consideran un TextView que presenta algunas modificaciones. Puede mostrarle un texto al usuario; no obstante, su función es permitirle o brindarle la opción de introducir texto.




```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"
    android:background="@drawable/fondo"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/eT_nombre"
        android:layout_width="266dp"
        android:layout_height="53dp"
        android:ems="10"
        android:hint="Introduzca el nombre"
        android:inputType="textPersonName"
        android:textAllCaps="false"
        android:textIsSelectable="false"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.169" />
```

Nuestros campos de edición de texto tendrán un id que dependerá del tipo de dato a introducir o mostrar; un layout width, o ancho del grupo; un layout height, que define el alto; un hint o pista, que es el mensaje que se muestra dentro de ellos para que el usuario sepa que dato debe introducir; y un inputType, que determinará el tipo de entrada que se va a utilizar, entre otras propiedades.
Los botones, por otro lado, dispondrán también de un id, ancho y alto, se les asignará un método de nuestro MainActivity.java a través de la propiedad onClick, y les proporcionaremos un icono gracias a la propiedad background para que la aplicación sea más intuitiva.

```
<EditText
    android:id="@+id/eT_telf"
    android:layout_width="266dp"
    android:layout_height="53dp"
    android:ems="10"
    android:hint="Introduzca el teléfono"
    android:inputType="phone"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.302" />

<EditText
    android:id="@+id/eT_file"
    android:layout_width="266dp"
    android:layout_height="59dp"
    android:ems="10"
    android:hint="Introduzca el nombre del fichero"
    android:inputType="text"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.443" />
```

```
<Button
    android:id="@+id/salvar"
    android:layout_width="66dp"
    android:layout_height="63dp"
    android:background="@drawable/salvar"
    android:onClick="Guardar"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.231"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.667" />

<Button
    android:id="@+id/buscar"
    android:layout_width="84dp"
    android:layout_height="73dp"
    android:background="@drawable/buscar"
    android:onClick="Buscar"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.755"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.677" />
```

```
    <EditText
        android:id="@+id/eT_mostrar"
        android:layout_width="290dp"
        android:layout_height="141dp"
        android:hint="Los resultados de su búsqueda aparecerán aquí"
        android:textStyle="bold"
        android:ems="10"
        android:gravity="start|top"
        android:inputType="textMultiLine"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.447"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.89" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

En el archivo strings.xml, ubicado en el directorio “values” que a su vez se encuentra dentro de “res”, crearemos todas las constantes a incluir en nuestro programa. Para ello debemos disponer @string/ y seguidamente el nombre de la constante definida en el archivo. Por defecto siempre podremos encontrar el string app_name que contiene el nombre de nuestra aplicación.

También modificaremos en el archivo colors.xml los colores de nuestra aplicación para que sea mucho más estética.

```
<resources>
    <string name="app_name">AGENDA</string>
    <string name="contacto">Introduzca el nombre </string>
    <string name="telefono">Introduzca el teléfono</string>
    <string name="archivo">Introduzca el nombre del fichero</string>
    <string name="mostrar">Los resultados de su búsqueda aparecerán aquí</string>
</resources>
```

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#000000</color>
    <color name="purple_700">#000000</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

Una vez completados los pasos anteriores comenzaremos a trabajar en nuestro archivo ActivityMain.java.
Crearemos tres objetos de tipo private que serán nuestros EditText.
En el método onCreate, que es donde ejecutamos la lógica de arranque básica de la aplicación que debe ocurrir una sola vez en toda la vida de la actividad, y gracias a la función findViewById() enlazaremos los recursos de la interfaz de usuario de nuestra aplicación, con las variables en nuestro código.

```java
package com.example.feedback4;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    private EditText eT_nombre, eT_telf, eT_file, eT_mostrar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        eT_nombre = (EditText) findViewById(R.id.eT_nombre);
        eT_telf = (EditText) findViewById(R.id.eT_telf);
        eT_file = (EditText) findViewById(R.id.eT_file);
        eT_mostrar = (EditText) findViewById(R.id.eT_mostrar);
    }
```

Para que nuestra aplicación sea completamente funcional debe ser capaz de guardar los datos de nuestro contacto y mostrarlos posteriormente en caso de que el usuario así lo requiera. Crearemos por tanto dos métodos, uno para cada conjunto de acciones.

En nuestro método “Guardar” crearemos dos formas de almacenar datos. La primera, mediante una clase SharedPreferences, un objeto SharedPreferences apunta a un archivo que contiene pares clave-valor y proporciona métodos sencillos para leerlos y escribirlos. El uso principal de SharedPreferences es guardar las preferencias de los usuarios, configuraciones, datos... Para que la próxima vez que la aplicación sea lanzada, estas piezas de información puedan ser recibidas y usadas.

La segunda, debe almacenar nuestros datos en un archivo de texto cuyo nombre nos indica el usuario por teclado. Para ello, dentro de un try/catch, crearemos un objeto de la clase OutputStreamWriter y al constructor de dicha clase le enviaremos el dato que retorna el método openFileOutput al que le pasaremos como parámetro el nombre del archivo de texto y el modo de apertura.
Debemos también llamar al método write y le pasaremos los datos a grabar, introducidos previamente en nuestros EditText, el método flush será el encargado de volcar todos los datos que pueden haber quedado en el buffer y una vez completados los pasos anteriores, procederemos a cerrar el archivo y a limpiar nuestros EditText, además de informar a través de un Toast al usuario de que su fichero se ha guardado correctamente.

```
public void Guardar(View view) {
    String nombre = eT_nombre.getText().toString();
    String telf = eT_telf.getText().toString();
    String file = eT_file.getText().toString();
    String nombrefichero = file + ".txt";
    ////////////////////////////////SHARED PREFERENCES/////////////////////////
    SharedPreferences preferences = getSharedPreferences( name: "file", Context.MODE_PRIVATE);
    SharedPreferences.Editor obj_editor = preferences.edit();
    obj_editor.putString(nombre, telf);
    obj_editor.commit();
    ////////////////////////////////////////////////////////////////////////////
    try{
        OutputStreamWriter archivo = new OutputStreamWriter(openFileOutput(nombrefichero, Activity.MODE_APPEND));
        archivo.write( str: nombre+" "+telf +"\n");
        archivo.flush();
        archivo.close();

        eT_nombre.setText("");
        eT_telf.setText("");
        eT_file.setText("");
    } catch(IOException e){
        System.err.print("ERROR AL ESCRIBIR EL ARCHIVO");
    }
    Toast.makeText( context: this,  text: "El contacto se ha guardado correctamente en su sistema de archivos", Toast.LENGTH_SHORT).show();
}
```

Para buscar y recuperar los datos haremos uso de SharedPreferences, donde hemos almacenado tanto el nombre como el teléfono de nuestro contacto, y también de nuestro archivo de texto.

En el primer caso, recuperaremos los datos almacenados en nuestro archivo de preferencias para poder acceder al teléfono que corresponde al nombre que estamos buscando en nuestra agenda y escribiremos el dato en el EditText correspondiente.

En el segundo caso, crearemos un objeto de la clase InputStreamReader y al constructor de dicha clase le pasaremos el dato devuelto por el método openFileInput, también crearemos un objeto de la clase BufferedReader, que sirve para leer texto desde un flujo de entrada de caracteres, almacenando los caracteres para proporcionar una lectura eficiente de caracteres, arreglos y líneas, y le pasaremos al constructor la referencia del objeto de la clase InputStreamReader.
Mientras el método readLine de la clase BufferedReader devuelva un String lo leeremos línea a línea y tras cerrar nuestros BufferedReader y nuestro InputStreamReader cargaremos el EditText que corresponde con los datos del fichero para que los muestre.

```
public void Buscar(View view){
    String nombre = eT_nombre.getText().toString();
    String file = eT_file.getText().toString();
    //String telf = eT_telf.getText().toString();
    String nombrefichero = file + ".txt";
    ////////////////////////////////SHARED PREFERENCES/////////////////////////
    SharedPreferences preferences = getSharedPreferences( name: "file", Context.MODE_PRIVATE);
    String datos = preferences.getString(nombre, s1: "");
    if(datos.length() == 0){
        Toast.makeText( context: this,  text: "El contacto que busca no existe", Toast.LENGTH_SHORT).show();
    }
    else{
        eT_telf.setText(datos);
    }
    ////////////////////////////////////////////////////////////////////////////
    try{
        InputStreamReader abrirArchivo = new InputStreamReader(openFileInput(nombrefichero));
        BufferedReader leerArchivo = new BufferedReader(abrirArchivo);
        String linea= leerArchivo.readLine();
        String contenidoCompleto= "";
        while(linea != null){
            contenidoCompleto=contenidoCompleto+linea+"\n";
            linea=leerArchivo .readLine();

        }
        leerArchivo.close();
        abrirArchivo.close();
        eT_mostrar.setText(contenidoCompleto);
```

```
        }catch(IOException e){
            Toast.makeText( context: this,  text: "Error al leer el archivo", Toast.LENGTH_SHORT).show();
        }

    }
}
```

Por supuesto a lo largo de la realización del feedback debemos controlar las posibles excepciones que surjan.

Nuestra aplicación debe se mostrar pues de la siguiente forma: