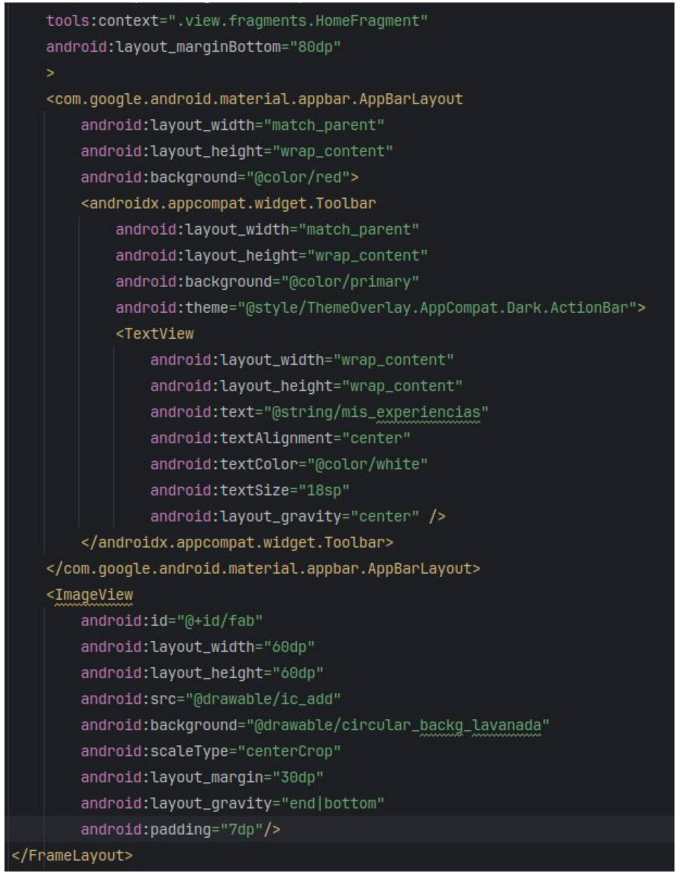
Post en Proyecto integrador

ScrollView,cardView, retrofit, Cloudinary

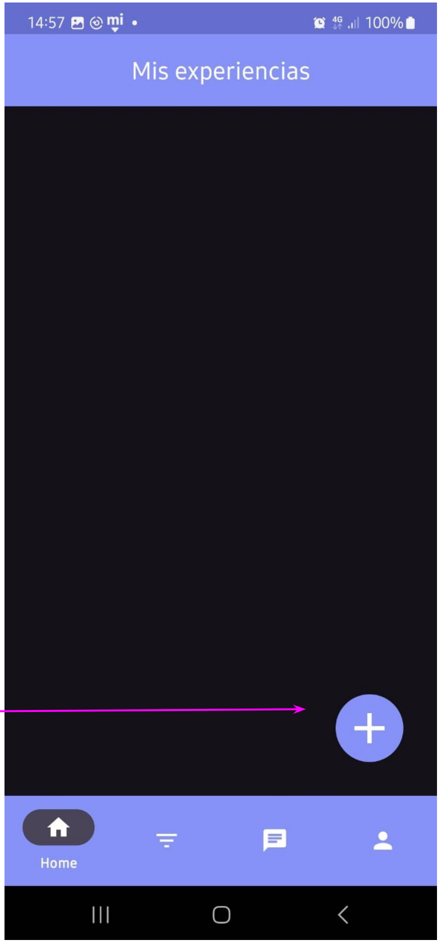
HomeFragment.xml



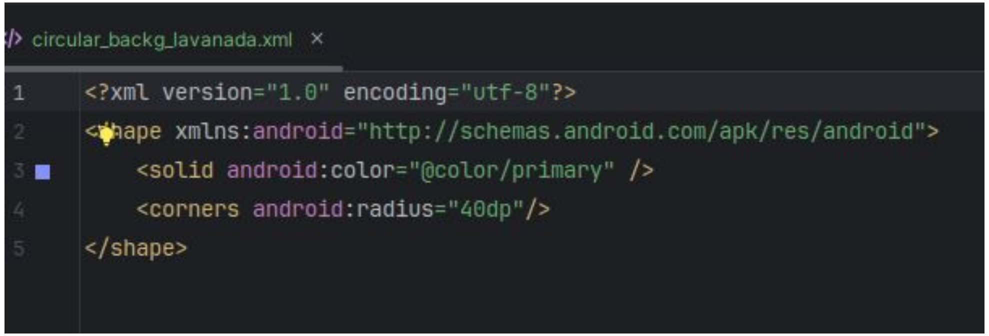
```
    tools:context=".view.fragments.HomeFragment"
    android:layout_marginBottom="80dp"
    >
    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/red">
        <androidx.appcompat.widget.Toolbar
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/primary"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/mis_experiencias"
                android:textAlignment="center"
                android:textColor="@color/white"
                android:textSize="18sp"
                android:layout_gravity="center" />
        </androidx.appcompat.widget.Toolbar>
    </com.google.android.material.appbar.AppBarLayout>
    <ImageView
        android:id="@+id/fab"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:src="@drawable/ic_add"
        android:background="@drawable/circular_backg_lavanada"
        android:scaleType="centerCrop"
        android:layout_margin="30dp"
        android:layout_gravity="end|bottom"
        android:padding="7dp"/>
</FrameLayout>
```

AppBarLayout
Toolbar
TextView

ImageView



Fondo para el botón circular

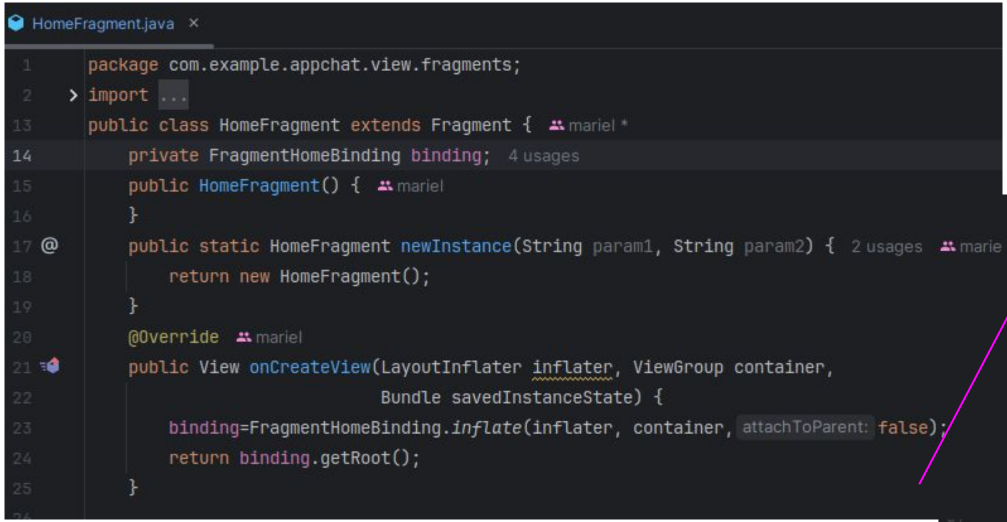

circular_backg_lavanada.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/primary" />
    <corners android:radius="40dp"/>
</shape>
```

Es el archivo **circular_backg_lavanda.xml** en la carpeta drawable usado para el botón circular. Lo usamos en la imageView que actúa como botón circular.

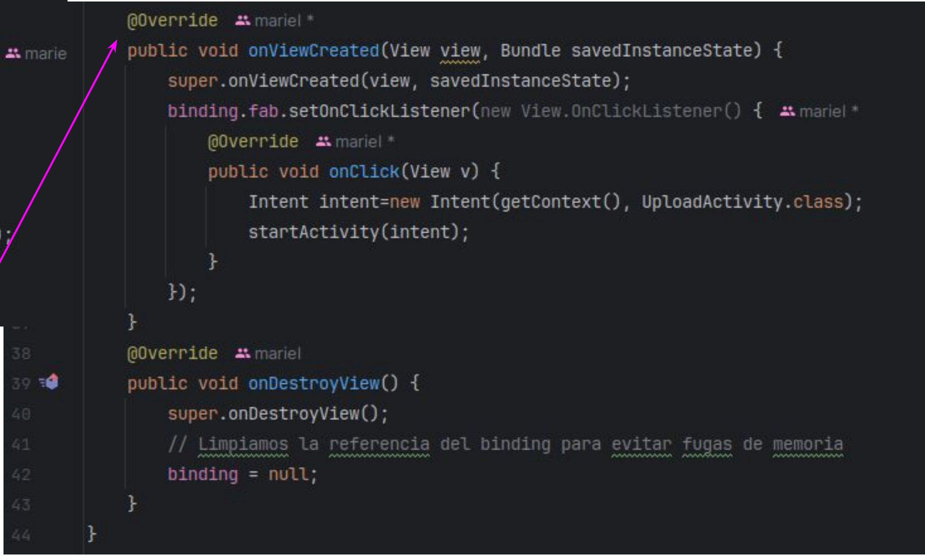
```
android:background="@drawable/circular_backg_lavanada"
```

Asignamos comportamiento al botón circular



HomeFragment.java

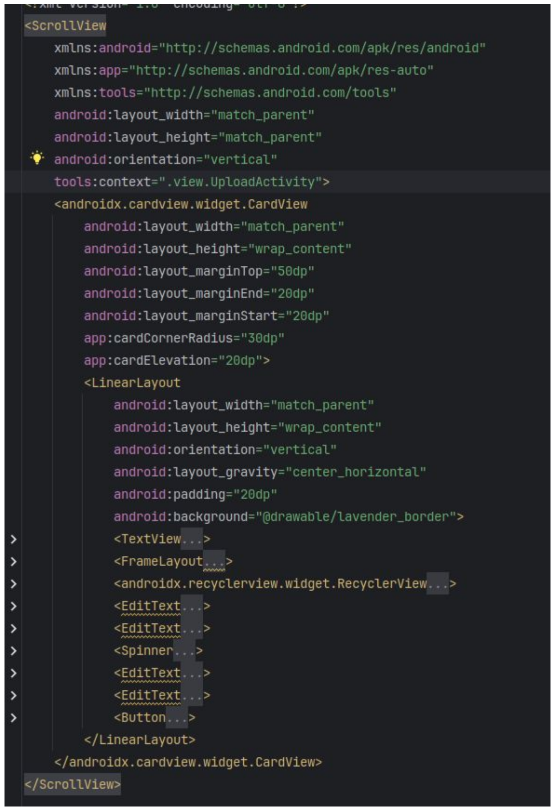
```java
package com.example.appchat.view.fragments;
import ...
public class HomeFragment extends Fragment {  mariel *
    private FragmentHomeBinding binding;  4 usages
    public HomeFragment() {  mariel
    }
    public static HomeFragment newInstance(String param1, String param2) {  2 usages  marie
        return new HomeFragment();
    }
    @Override  mariel
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        binding=FragmentHomeBinding.inflate(inflater, container, attachToParent: false);
        return binding.getRoot();
    }
```



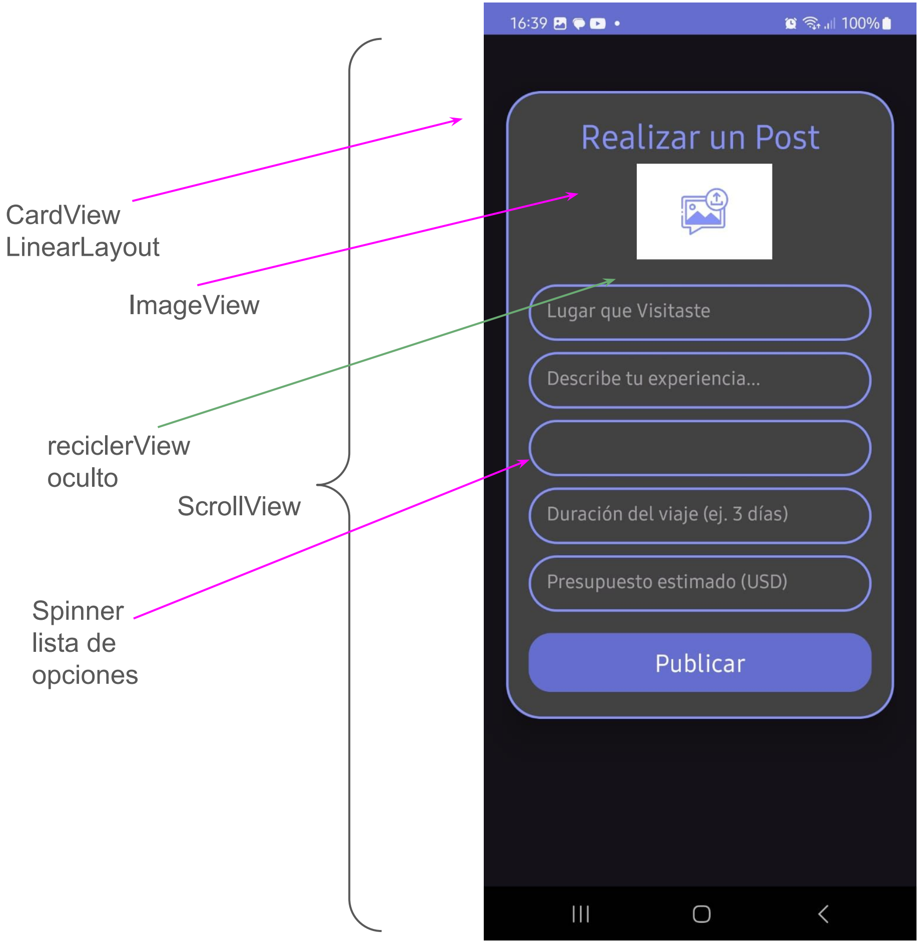
```java
    @Override  mariel *
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        binding.fab.setOnClickListener(new View.OnClickListener() {  mariel *
            @Override  mariel *
            public void onClick(View v) {
                Intent intent=new Intent(getContext(), UploadActivity.class);
                startActivity(intent);
            }
        });
    }
    @Override  mariel
    public void onDestroyView() {
        super.onDestroyView();
        // Limpiamos la referencia del binding para evitar fugas de memoria
        binding = null;
    }
}
```

Prestar atención a cómo se crea el binding, cómo se infla en la vista y lo necesario del método onDestroyView.

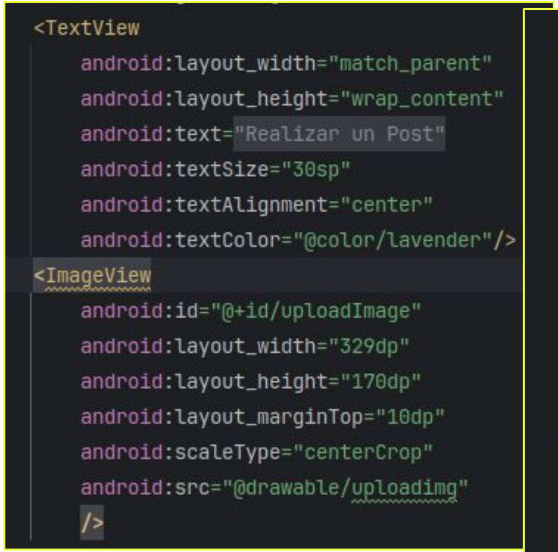
actyity_upload.xml



```xml
<ScrollView
    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="vertical"
    tools:context=".view.UploadActivity">
    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="20dp"
        android:layout_marginStart="20dp"
        app:cardCornerRadius="30dp"
        app:cardElevation="20dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_gravity="center_horizontal"
            android:padding="20dp"
            android:background="@drawable/lavender_border">
            <TextView...>
            <FrameLayout...>
            <androidx.recyclerview.widget.RecyclerView...>
            <EditText...>
            <EditText...>
            <Spinner...>
            <EditText...>
            <EditText...>
            <Button...>
        </LinearLayout>
    </androidx.cardview.widget.CardView>
</ScrollView>
```

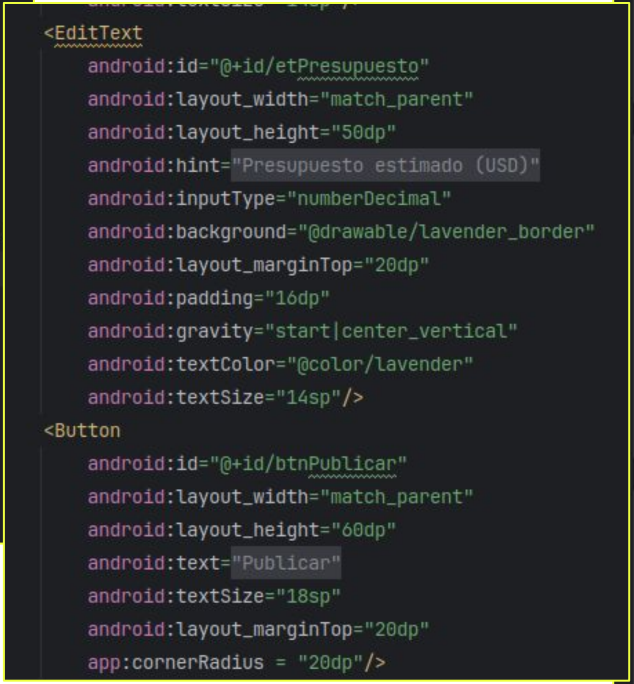


Así lucen los componentes de actyity_upload.xml

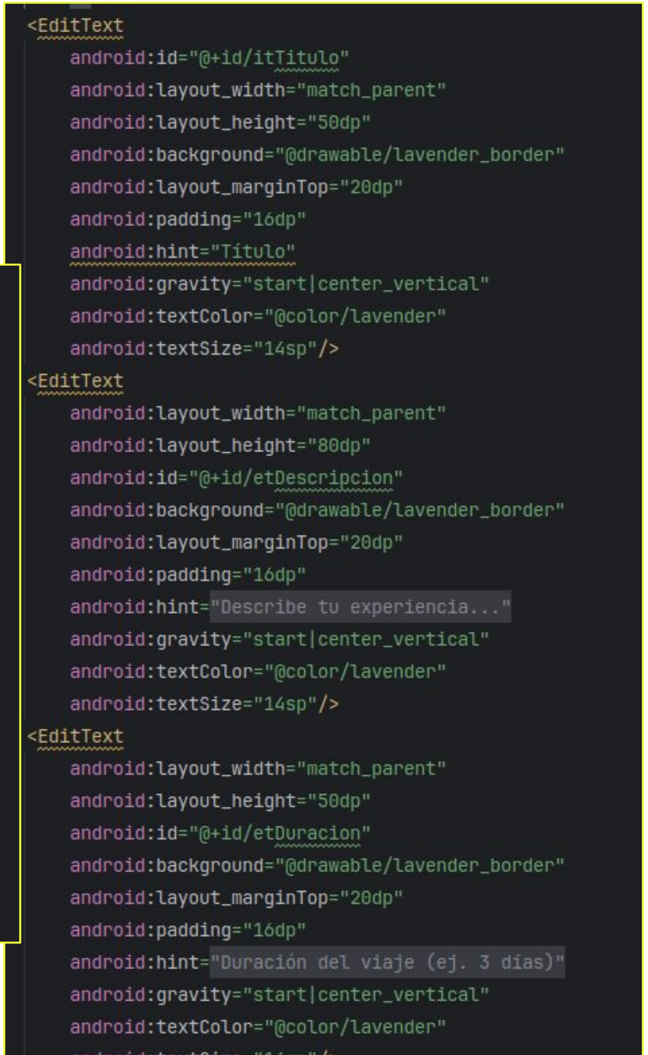


```xml
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Realizar un Post"
    android:textSize="30sp"
    android:textAlignment="center"
    android:textColor="@color/lavender"/>
<ImageView
    android:id="@+id/uploadImage"
    android:layout_width="329dp"
    android:layout_height="170dp"
    android:layout_marginTop="10dp"
    android:scaleType="centerCrop"
    android:src="@drawable/uploadimg"
    />
```

dentro del LinearLayout…

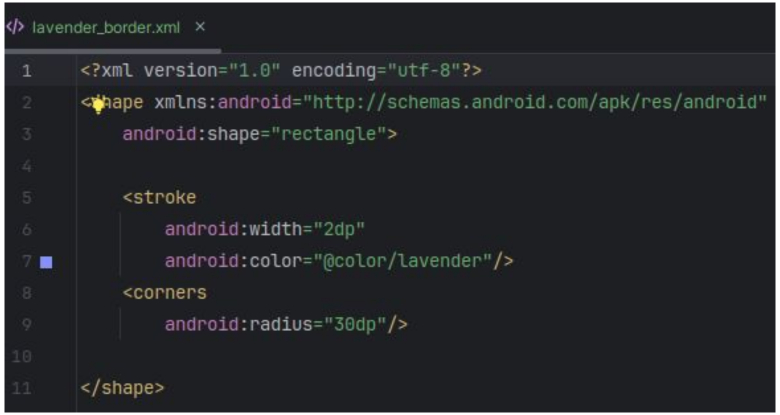


```xml
<EditText
    android:id="@+id/etPresupuesto"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:hint="Presupuesto estimado (USD)"
    android:inputType="numberDecimal"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"
    android:textSize="14sp"/>
<Button
    android:id="@+id/btnPublicar"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:text="Publicar"
    android:textSize="18sp"
    android:layout_marginTop="20dp"
    app:cornerRadius = "20dp"/>
```



```xml
<EditText
    android:id="@+id/itTitulo"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Título"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"
    android:textSize="14sp"/>
<EditText
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:id="@+id/etDescripcion"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Describe tu experiencia..."
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"
    android:textSize="14sp"/>
<EditText
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:id="@+id/etDuracion"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Duración del viaje (ej. 3 dias)"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"
```

lavander_border.xml


lavender_border.xml ×

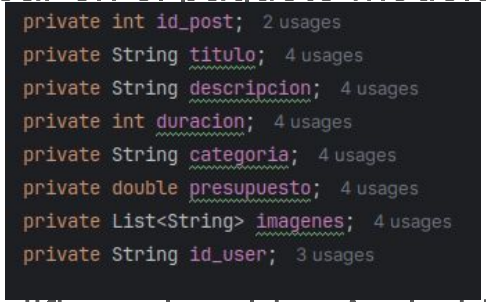
```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <stroke
        android:width="2dp"
        android:color="@color/lavender"/>
    <corners
        android:radius="30dp"/>

</shape>
```

Es el archivo *lavander_border.xml* en la carpeta drawable. Lo usamos en el linearLayout

```
android:background="@drawable/lavender_border"
```

Actividades de hoy

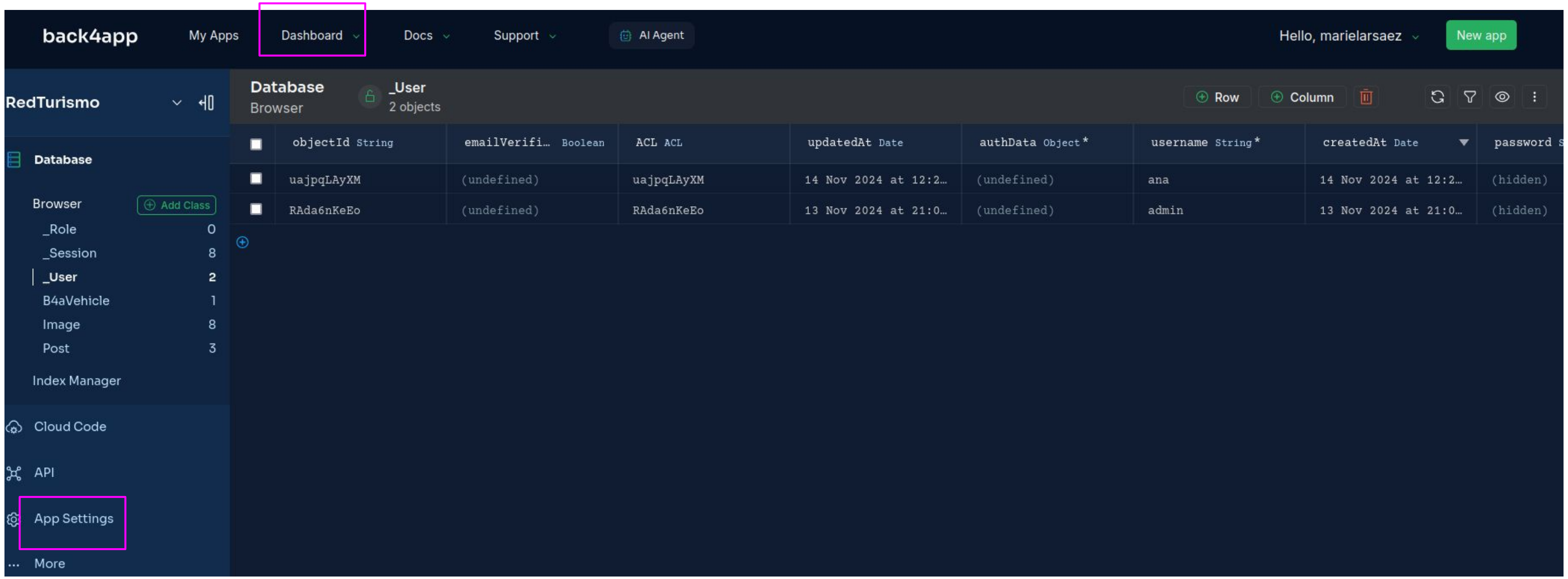
1. Modificar el home_fragment.xml y el HomeFragment.java (agregar botón y comportamiento)
2. Crear ActivityUpload.java y upload_activity.xml (Construir la vista similar a la propuesta)
3. Crear en el paquete modelo la clase Post.java con los siguientes atributos



```
private int id_post;  2 usages
private String titulo;  4 usages
private String descripcion;  4 usages
private int duracion;  4 usages
private String categoria;  4 usages
private double presupuesto;  4 usages
private List<String> imagenes;  4 usages
private String id_user;  3 usages
```

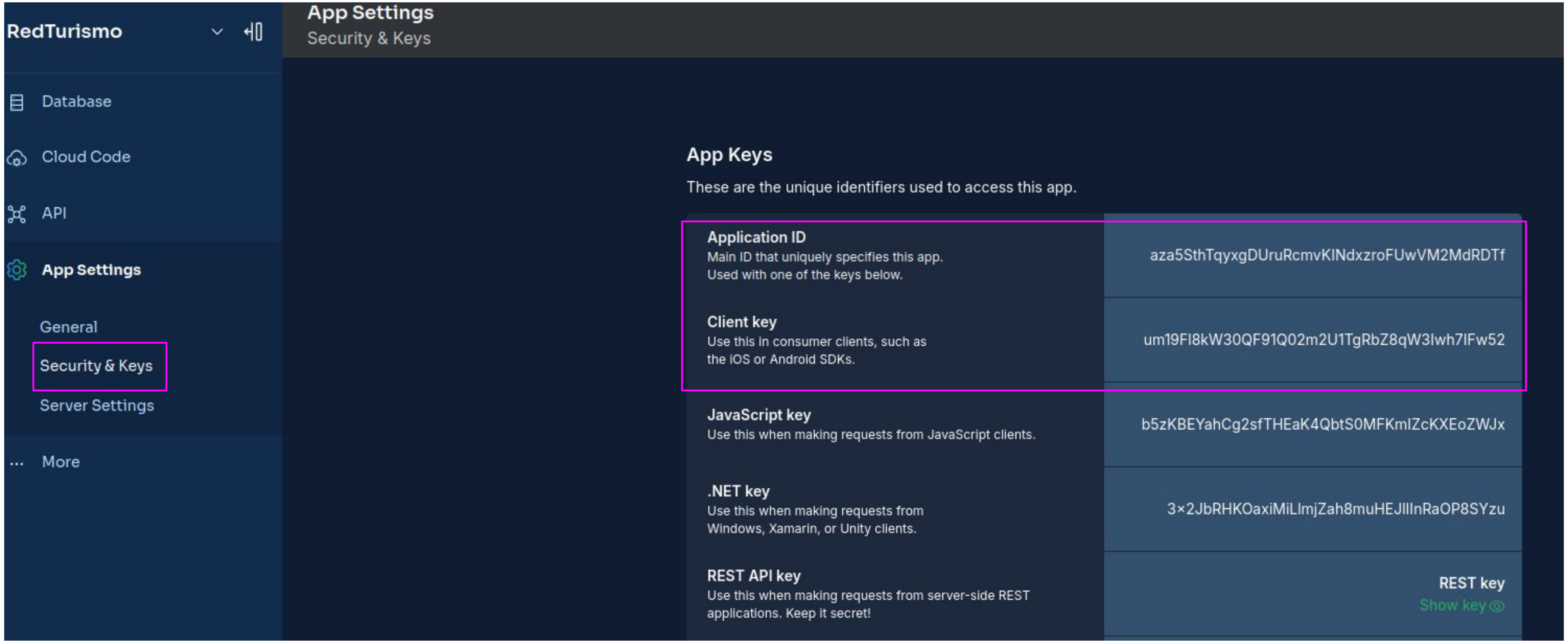
4. Modificar el archivo AndroidManifest.xml asignando permisos y agregando los metadatos
5. Agregar dependencias a build.gradle.kts

Creando nuestro proyecto en parse (Configuración de Parse)



	objectId String	emailVerifi… Boolean	ACL ACL	updatedAt Date	authData Object *	username String *	createdAt Date	password
	uajpqLAyXM	(undefined)	uajpqLAyXM	14 Nov 2024 at 12:2…	(undefined)	ana	14 Nov 2024 at 12:2…	(hidden)
	RAda6nKeEo	(undefined)	RAda6nKeEo	13 Nov 2024 at 21:0…	(undefined)	admin	13 Nov 2024 at 21:0…	(hidden)

Configuración de Parse



App Settings
Security & Keys

App Keys

These are the unique identifiers used to access this app.

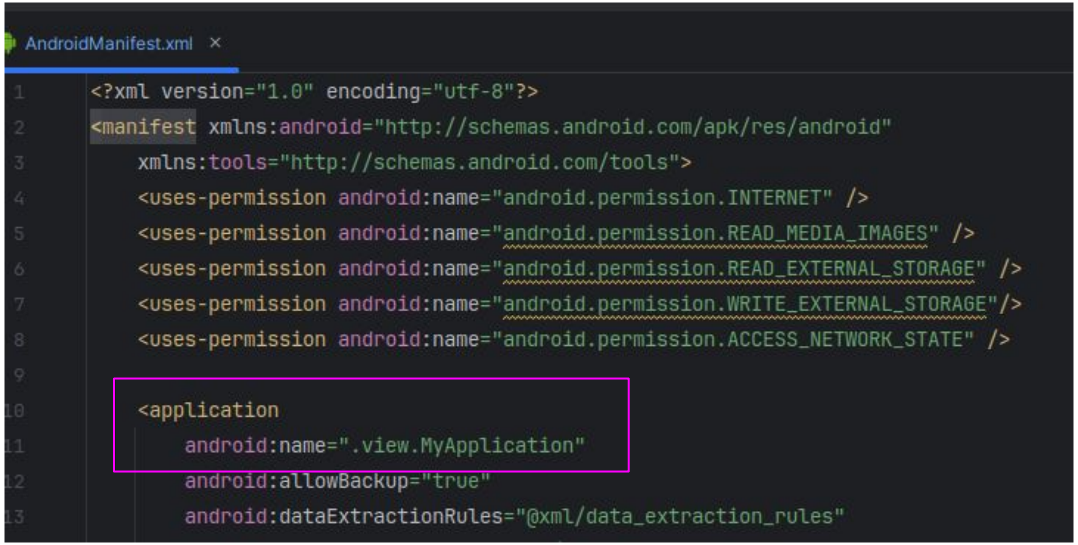
Application ID Main ID that uniquely specifies this app. Used with one of the keys below.	aza5SthTqyxgDUruRcmvKINdxzroFUwVM2MdRDTf
Client key Use this in consumer clients, such as the iOS or Android SDKs.	um19FI8kW30QF91Q02m2U1TgRbZ8qW3Iwh7IFw52
JavaScript key Use this when making requests from JavaScript clients.	b5zKBEYahCg2sfTHEaK4QbtS0MFKmIZcKXEoZWJx
.NET key Use this when making requests from Windows, Xamarin, or Unity clients.	3×2JbRHKOaxiMiLImjZah8muHEJIIInRaOP8SYzu
REST API key Use this when making requests from server-side REST applications. Keep it secret!	REST key Show key

Permisos y Metas en Manifest

```
<uses-permission  android:name="android.permission.INTERNET"  />
<uses-permission  android:name="android.permission.READ_MEDIA_IMAGES"  />
<uses-permission  android:name="android.permission.READ_EXTERNAL_STORAGE"  />
<uses-permission  android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission  android:name="android.permission.ACCESS_NETWORK_STATE"  />
```

```
<meta-data
   android:name="com.parse.SERVER_URL"
   android:value="@string/back4app_server_url"  />
<meta-data
   android:name="com.parse.APPLICATION_ID"
   android:value="@string/back4app_app_id"  />
<meta-data
   android:name="com.parse.CLIENT_KEY"
   android:value="um19Fl8kW30QF91Q02m2U1TgRbZ8qW3Iwh7IFw52"  />
```

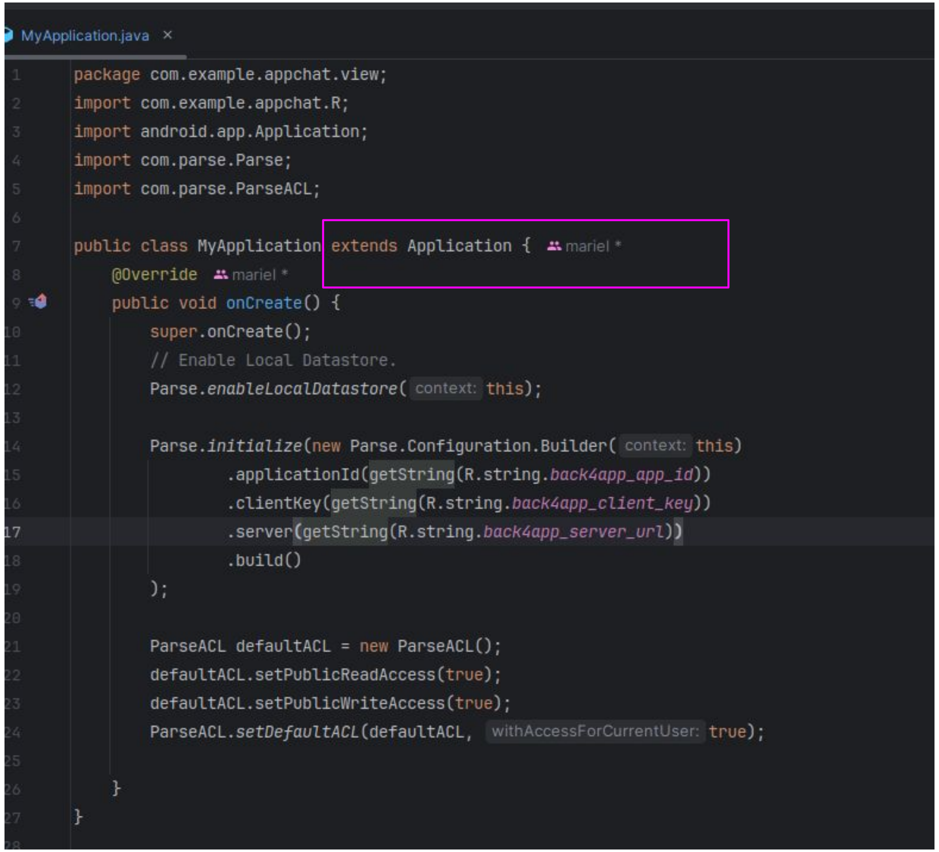
Configurar en Manifest la aplicación de inicio



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:name=".view.MyApplication"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
```

Crear un archivo que extienda de Aplication

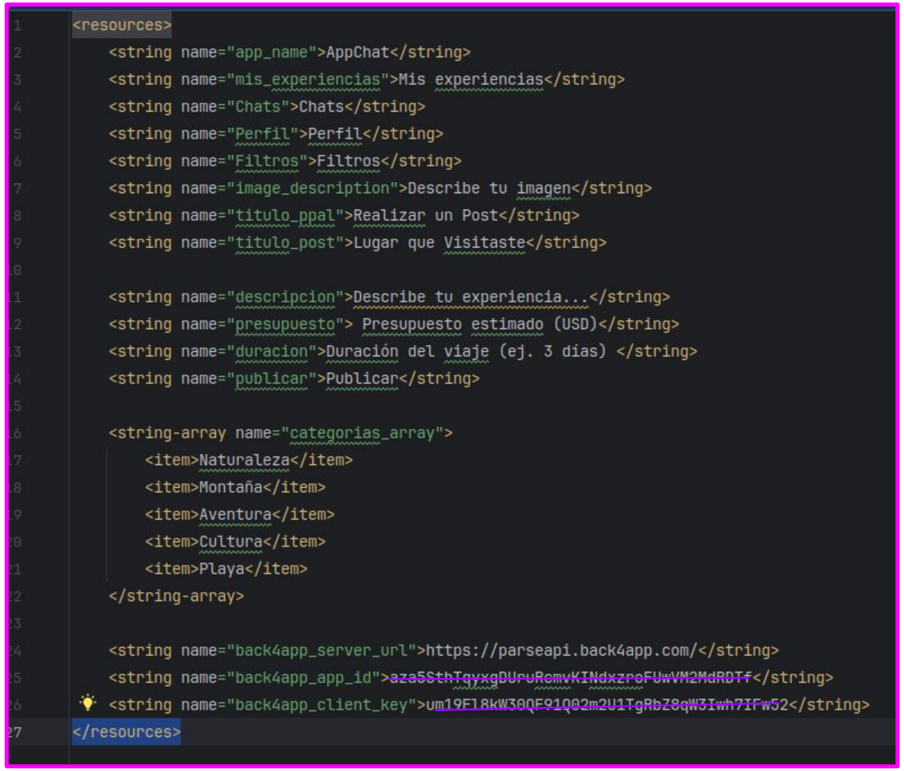


MyApplication.java

```java
package com.example.appchat.view;
import com.example.appchat.R;
import android.app.Application;
import com.parse.Parse;
import com.parse.ParseACL;

public class MyApplication extends Application {  mariel *
    @Override  mariel *
    public void onCreate() {
        super.onCreate();
        // Enable Local Datastore.
        Parse.enableLocalDatastore( context: this);

        Parse.initialize(new Parse.Configuration.Builder( context: this)
                .applicationId(getString(R.string.back4app_app_id))
                .clientKey(getString(R.string.back4app_client_key))
                .server(getString(R.string.back4app_server_url))
                .build()
        );

        ParseACL defaultACL = new ParseACL();
        defaultACL.setPublicReadAccess(true);
        defaultACL.setPublicWriteAccess(true);
        ParseACL.setDefaultACL(defaultACL,  withAccessForCurrentUser: true);

    }
}
```

inicializar parse con los strigns del archivo strings.xml con security keys de parse

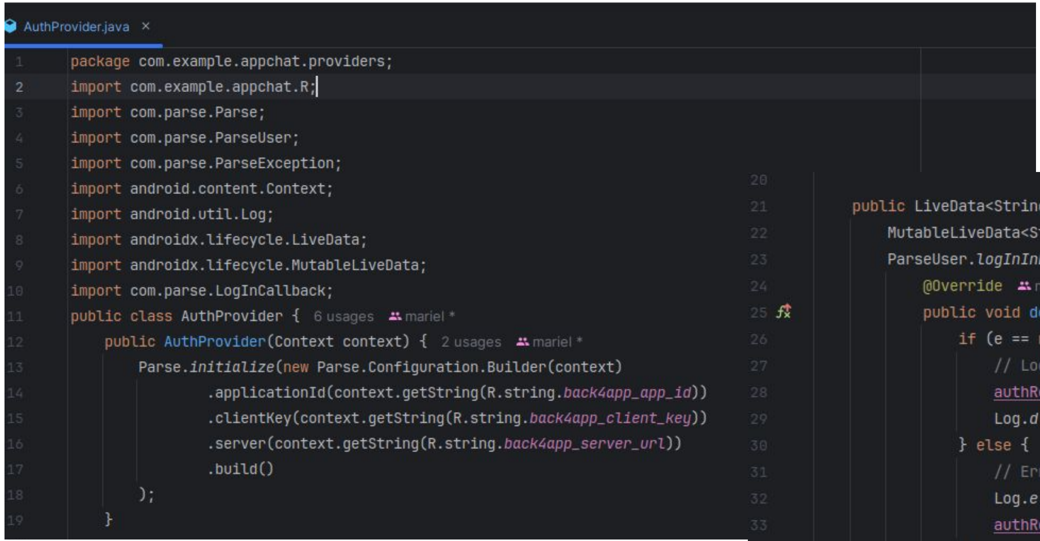
Agregar keys a strings.xml



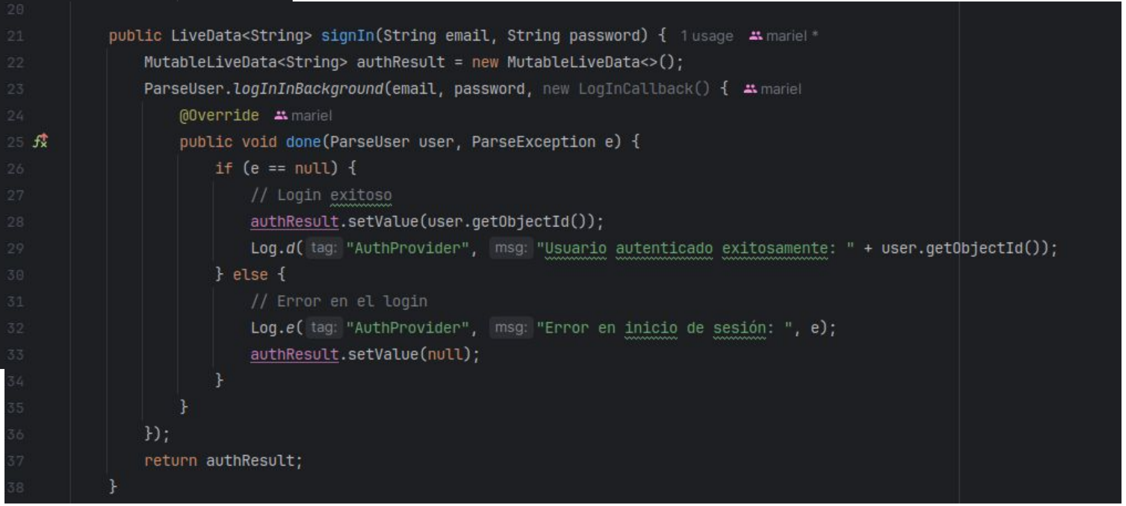
```xml
<resources>
    <string name="app_name">AppChat</string>
    <string name="mis_experiencias">Mis experiencias</string>
    <string name="Chats">Chats</string>
    <string name="Perfil">Perfil</string>
    <string name="Filtros">Filtros</string>
    <string name="image_description">Describe tu imagen</string>
    <string name="titulo_ppal">Realizar un Post</string>
    <string name="titulo_post">Lugar que Visitaste</string>

    <string name="descripcion">Describe tu experiencia...</string>
    <string name="presupuesto"> Presupuesto estimado (USD)</string>
    <string name="duracion">Duración del viaje (ej. 3 días) </string>
    <string name="publicar">Publicar</string>

    <string-array name="categorias_array">
        <item>Naturaleza</item>
        <item>Montaña</item>
        <item>Aventura</item>
        <item>Cultura</item>
        <item>Playa</item>
    </string-array>

    <string name="back4app_server_url">https://parseapi.back4app.com/</string>
    <string name="back4app_app_id">~~aza5SthIgyxgDUruRcmvKINdxzroFUwVM2MdRDTf~~</string>
    <string name="back4app_client_key">~~um19El8kW30QE91Q02m2U1TgRbZ8qW3Iwh7IFw52~~</string>
</resources>
```

pegan sus configuraciones…

Creamos el archivo AuthProvider.java en providers

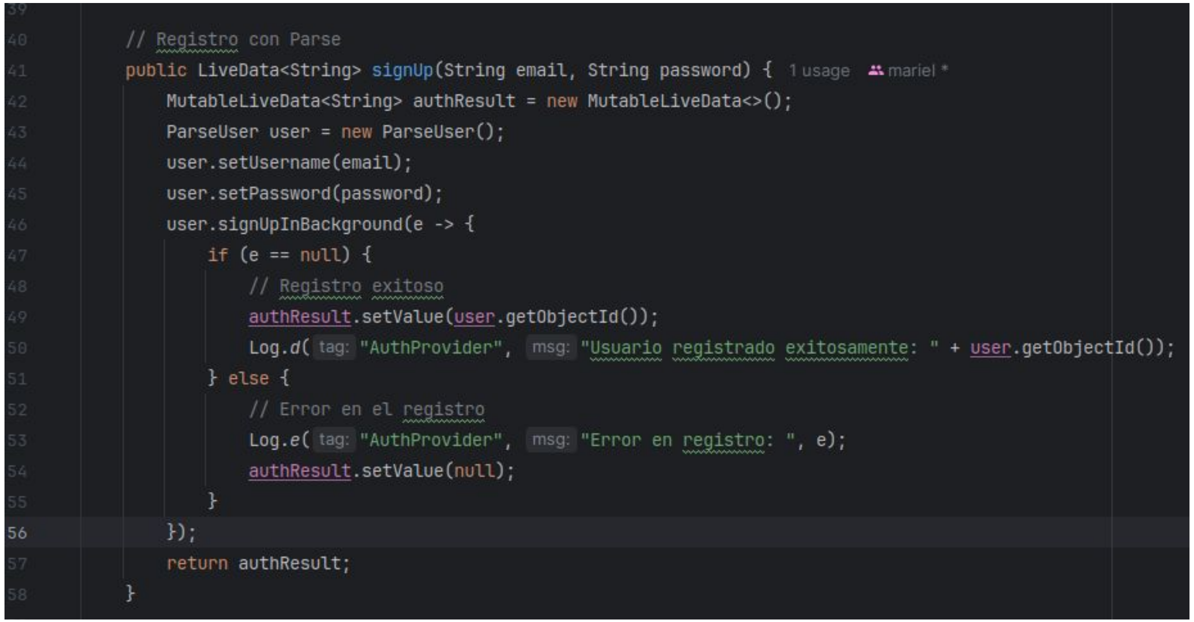

AuthProvider.java ×

```java
package com.example.appchat.providers;
import com.example.appchat.R;
import com.parse.Parse;
import com.parse.ParseUser;
import com.parse.ParseException;
import android.content.Context;
import android.util.Log;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import com.parse.LogInCallback;
public class AuthProvider {  6 usages  mariel *
    public AuthProvider(Context context) {  2 usages  mariel *
        Parse.initialize(new Parse.Configuration.Builder(context)
                .applicationId(context.getString(R.string.back4app_app_id))
                .clientKey(context.getString(R.string.back4app_client_key))
                .server(context.getString(R.string.back4app_server_url))
                .build()
        );
    }
```



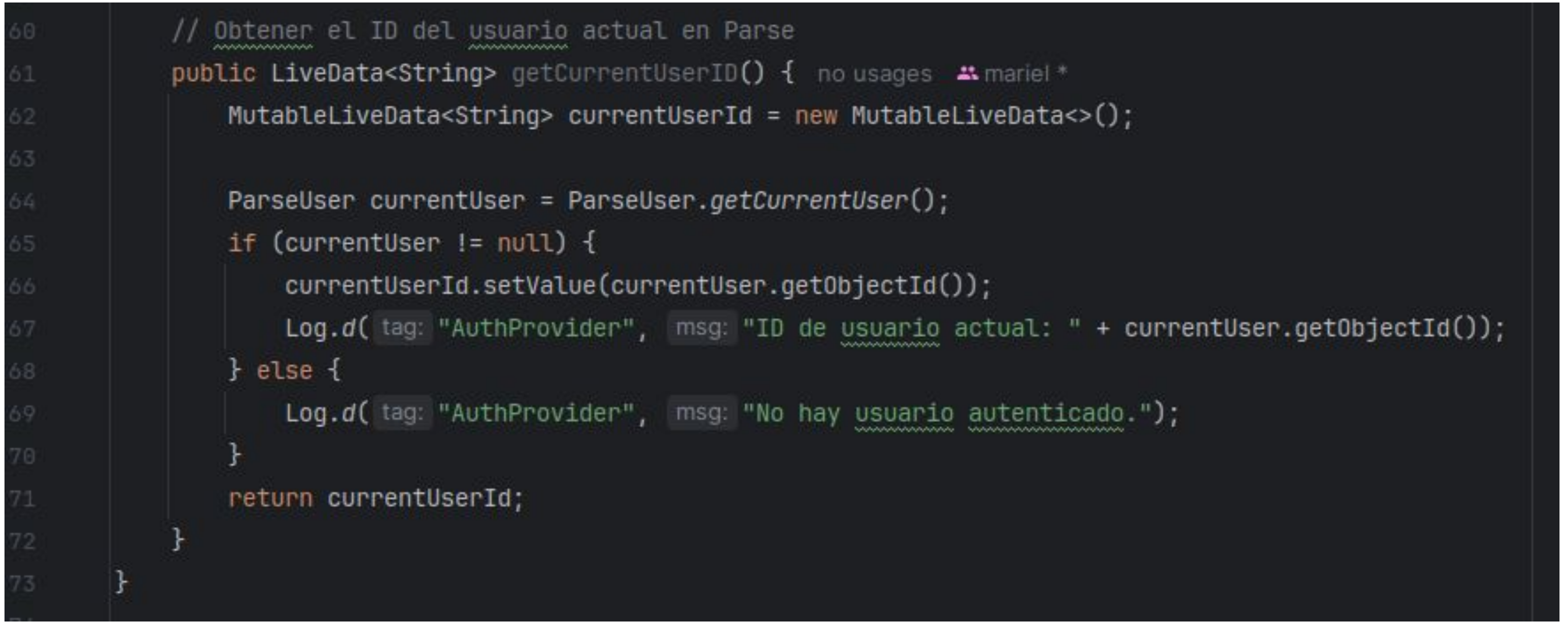
```java
    public LiveData<String> signIn(String email, String password) {  1 usage  mariel *
        MutableLiveData<String> authResult = new MutableLiveData<>();
        ParseUser.logInInBackground(email, password, new LogInCallback() {  mariel
            @Override  mariel
            public void done(ParseUser user, ParseException e) {
                if (e == null) {
                    // Login exitoso
                    authResult.setValue(user.getObjectId());
                    Log.d( tag: "AuthProvider",  msg: "Usuario autenticado exitosamente: " + user.getObjectId());
                } else {
                    // Error en el login
                    Log.e( tag: "AuthProvider",  msg: "Error en inicio de sesión: ", e);
                    authResult.setValue(null);
                }
            }
        });
        return authResult;
    }
```

Modificamos los archivos AuthProvider



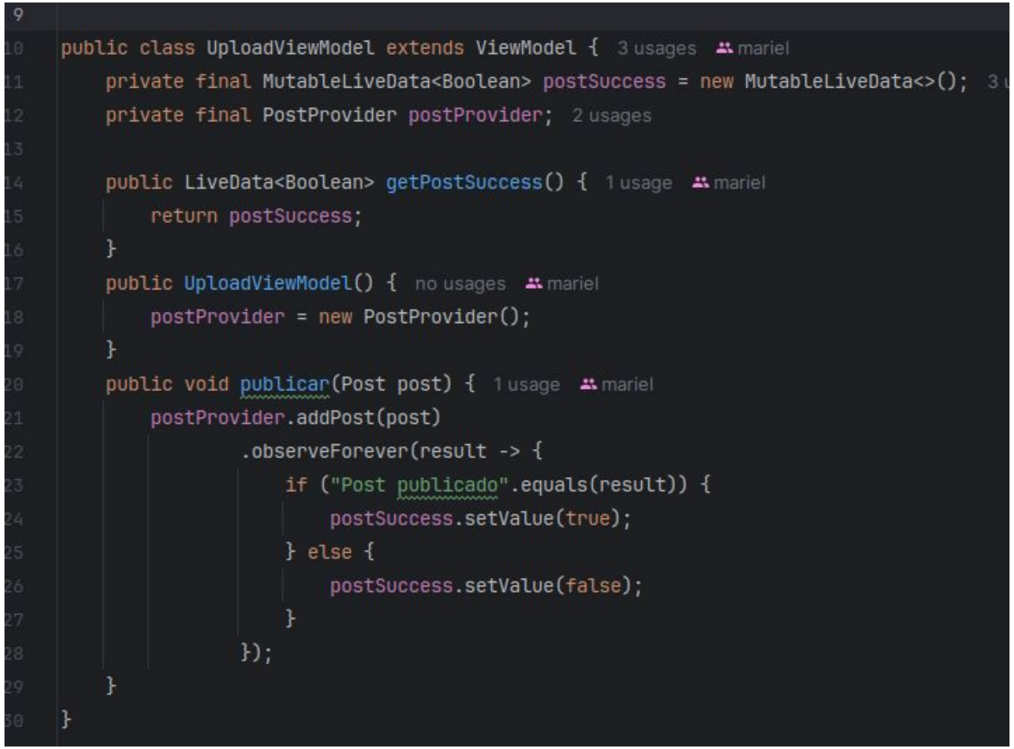
```java
    // Registro con Parse
    public LiveData<String> signUp(String email, String password) {  1 usage  mariel *
        MutableLiveData<String> authResult = new MutableLiveData<>();
        ParseUser user = new ParseUser();
        user.setUsername(email);
        user.setPassword(password);
        user.signUpInBackground(e -> {
            if (e == null) {
                // Registro exitoso
                authResult.setValue(user.getObjectId());
                Log.d( tag: "AuthProvider",  msg: "Usuario registrado exitosamente: " + user.getObjectId());
            } else {
                // Error en el registro
                Log.e( tag: "AuthProvider",  msg: "Error en registro: ", e);
                authResult.setValue(null);
            }
        });
        return authResult;
    }
```

AuthProvider.java getCurrentUserID()



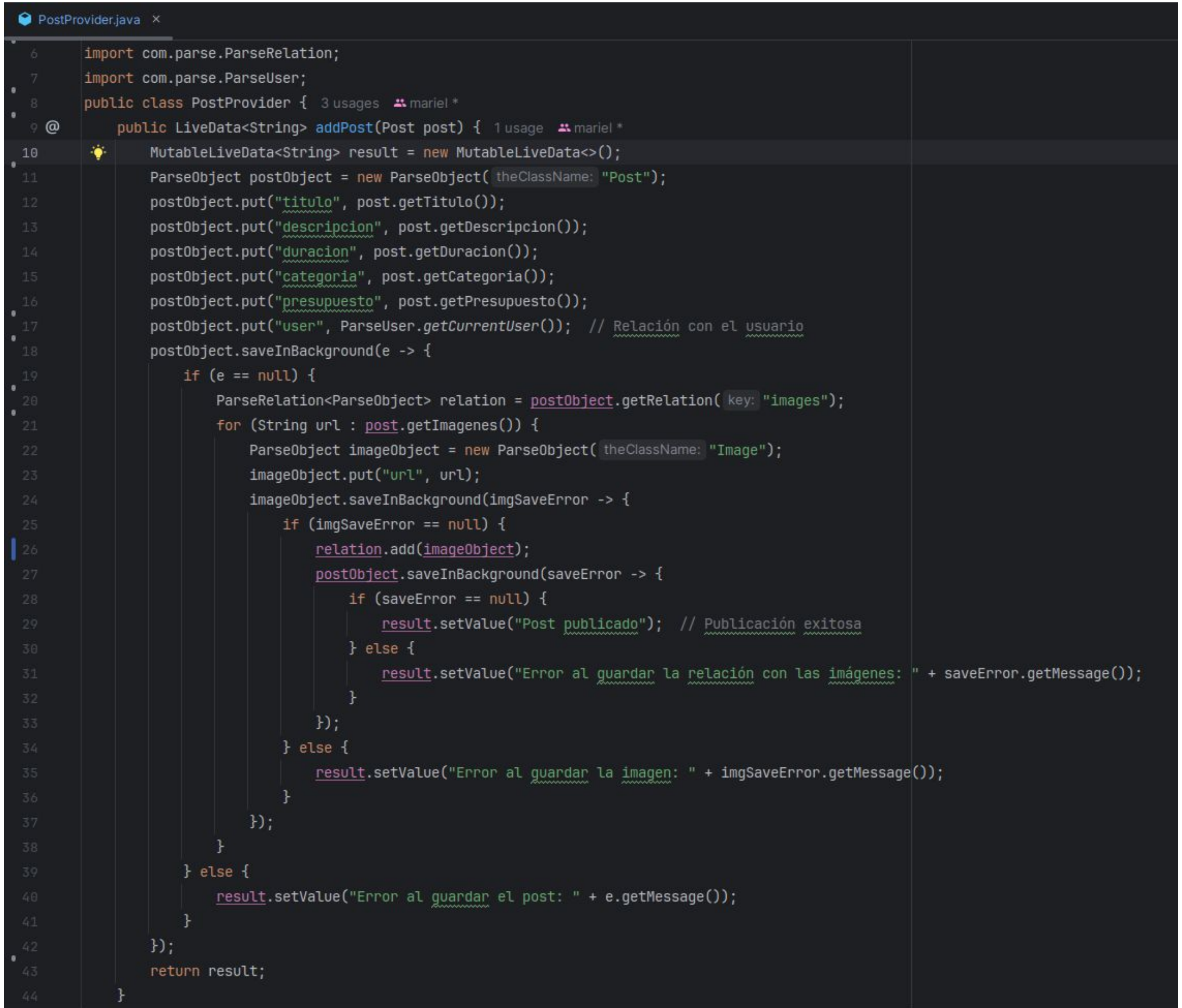
```
    // Obtener el ID del usuario actual en Parse
    public LiveData<String> getCurrentUserID() {  no usages   mariel *
        MutableLiveData<String> currentUserId = new MutableLiveData<>();

        ParseUser currentUser = ParseUser.getCurrentUser();
        if (currentUser != null) {
            currentUserId.setValue(currentUser.getObjectId());
            Log.d( tag: "AuthProvider",  msg: "ID de usuario actual: " + currentUser.getObjectId());
        } else {
            Log.d( tag: "AuthProvider",  msg: "No hay usuario autenticado.");
        }
        return currentUserId;
    }
}
```

UploadViewModel.java



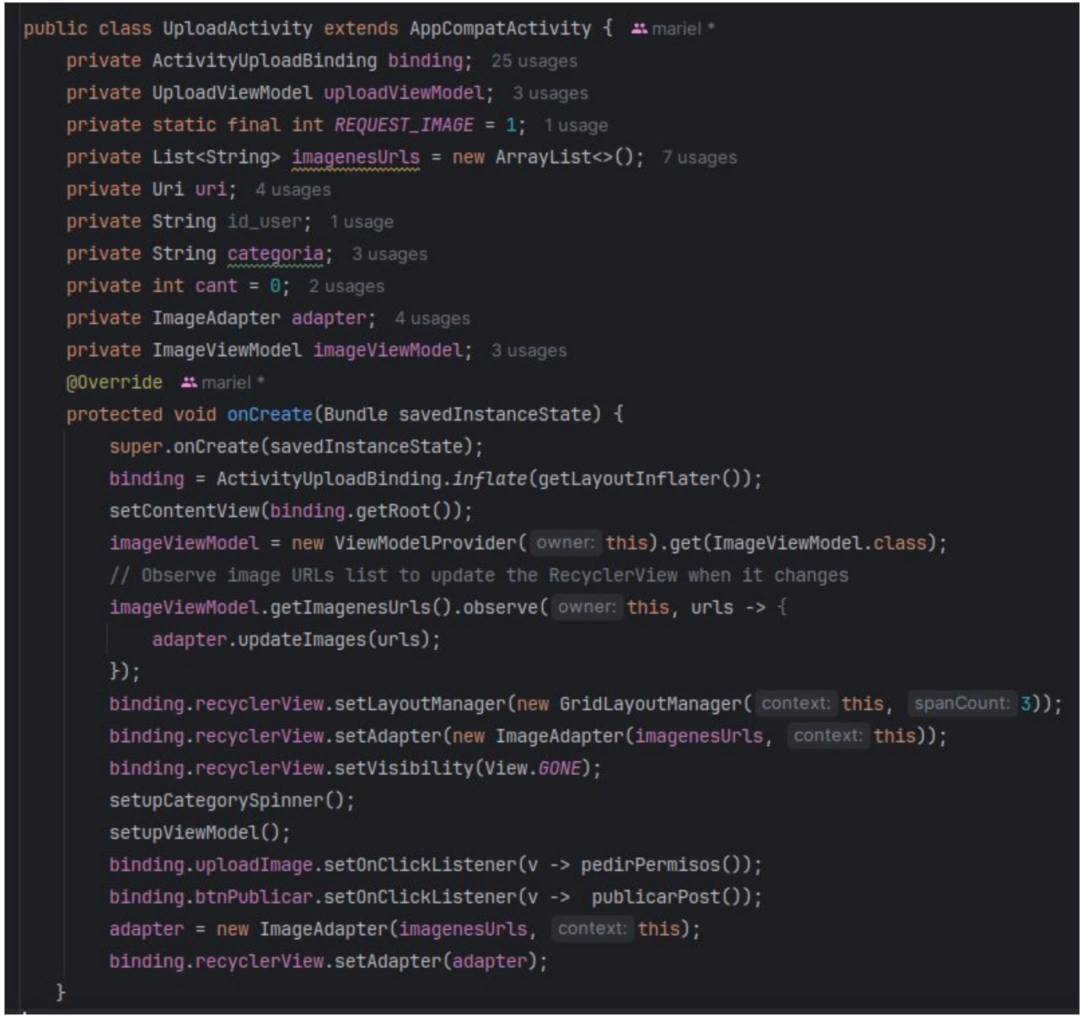
```
public class UploadViewModel extends ViewModel {  3 usages  mariel
    private final MutableLiveData<Boolean> postSuccess = new MutableLiveData<>();  3 u
    private final PostProvider postProvider;  2 usages

    public LiveData<Boolean> getPostSuccess() {  1 usage  mariel
        return postSuccess;
    }
    public UploadViewModel() {  no usages  mariel
        postProvider = new PostProvider();
    }
    public void publicar(Post post) {  1 usage  mariel
        postProvider.addPost(post)
                .observeForever(result -> {
                    if ("Post publicado".equals(result)) {
                        postSuccess.setValue(true);
                    } else {
                        postSuccess.setValue(false);
                    }
                });
    }
}
```

PostProvider


PostProvider.java ×

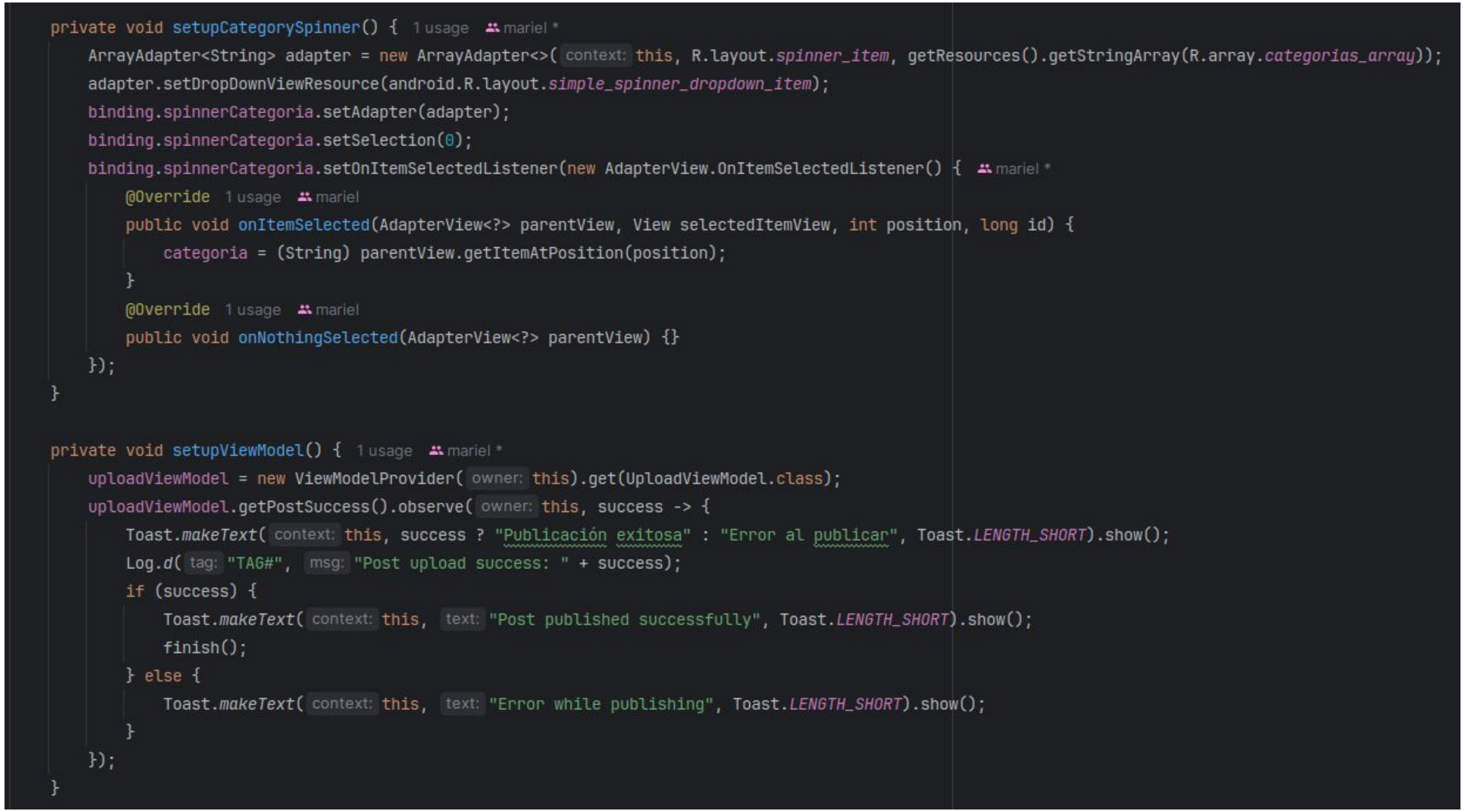
```java
import com.parse.ParseRelation;
import com.parse.ParseUser;
public class PostProvider {  3 usages  mariel *
    public LiveData<String> addPost(Post post) {  1 usage  mariel *
        MutableLiveData<String> result = new MutableLiveData<>();
        ParseObject postObject = new ParseObject( theClassName: "Post");
        postObject.put("titulo", post.getTitulo());
        postObject.put("descripcion", post.getDescripcion());
        postObject.put("duracion", post.getDuracion());
        postObject.put("categoria", post.getCategoria());
        postObject.put("presupuesto", post.getPresupuesto());
        postObject.put("user", ParseUser.getCurrentUser());  // Relación con el usuario
        postObject.saveInBackground(e -> {
            if (e == null) {
                ParseRelation<ParseObject> relation = postObject.getRelation( key: "images");
                for (String url : post.getImagenes()) {
                    ParseObject imageObject = new ParseObject( theClassName: "Image");
                    imageObject.put("url", url);
                    imageObject.saveInBackground(imgSaveError -> {
                        if (imgSaveError == null) {
                            relation.add(imageObject);
                            postObject.saveInBackground(saveError -> {
                                if (saveError == null) {
                                    result.setValue("Post publicado");  // Publicación exitosa
                                } else {
                                    result.setValue("Error al guardar la relación con las imágenes: " + saveError.getMessage());
                                }
                            });
                        } else {
                            result.setValue("Error al guardar la imagen: " + imgSaveError.getMessage());
                        }
                    });
                }
            } else {
                result.setValue("Error al guardar el post: " + e.getMessage());
            }
        });
        return result;
    }
```

UploadActivity.java



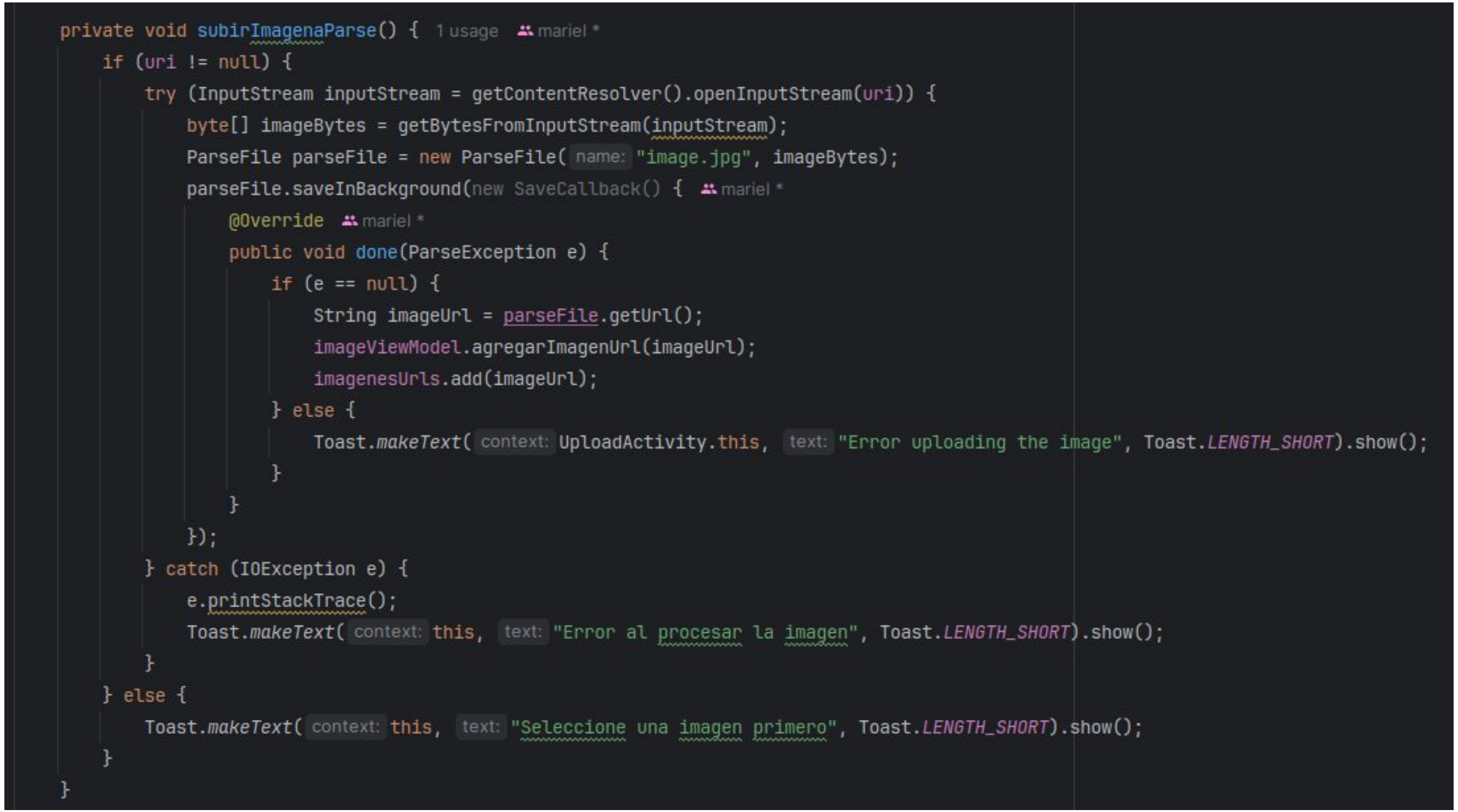
```
public class UploadActivity extends AppCompatActivity {  mariel *
    private ActivityUploadBinding binding;  25 usages
    private UploadViewModel uploadViewModel;  3 usages
    private static final int REQUEST_IMAGE = 1;  1 usage
    private List<String> imagenesUrls = new ArrayList<>();  7 usages
    private Uri uri;  4 usages
    private String id_user;  1 usage
    private String categoria;  3 usages
    private int cant = 0;  2 usages
    private ImageAdapter adapter;  4 usages
    private ImageViewModel imageViewModel;  3 usages
    @Override  mariel *
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityUploadBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        imageViewModel = new ViewModelProvider( owner: this).get(ImageViewModel.class);
        // Observe image URLs list to update the RecyclerView when it changes
        imageViewModel.getImagenesUrls().observe( owner: this, urls -> {
            adapter.updateImages(urls);
        });
        binding.recyclerView.setLayoutManager(new GridLayoutManager( context: this,  spanCount: 3));
        binding.recyclerView.setAdapter(new ImageAdapter(imagenesUrls,  context: this));
        binding.recyclerView.setVisibility(View.GONE);
        setupCategorySpinner();
        setupViewModel();
        binding.uploadImage.setOnClickListener(v -> pedirPermisos());
        binding.btnPublicar.setOnClickListener(v ->  publicarPost());
        adapter = new ImageAdapter(imagenesUrls,  context: this);
        binding.recyclerView.setAdapter(adapter);
    }
```

UploadActivity.java



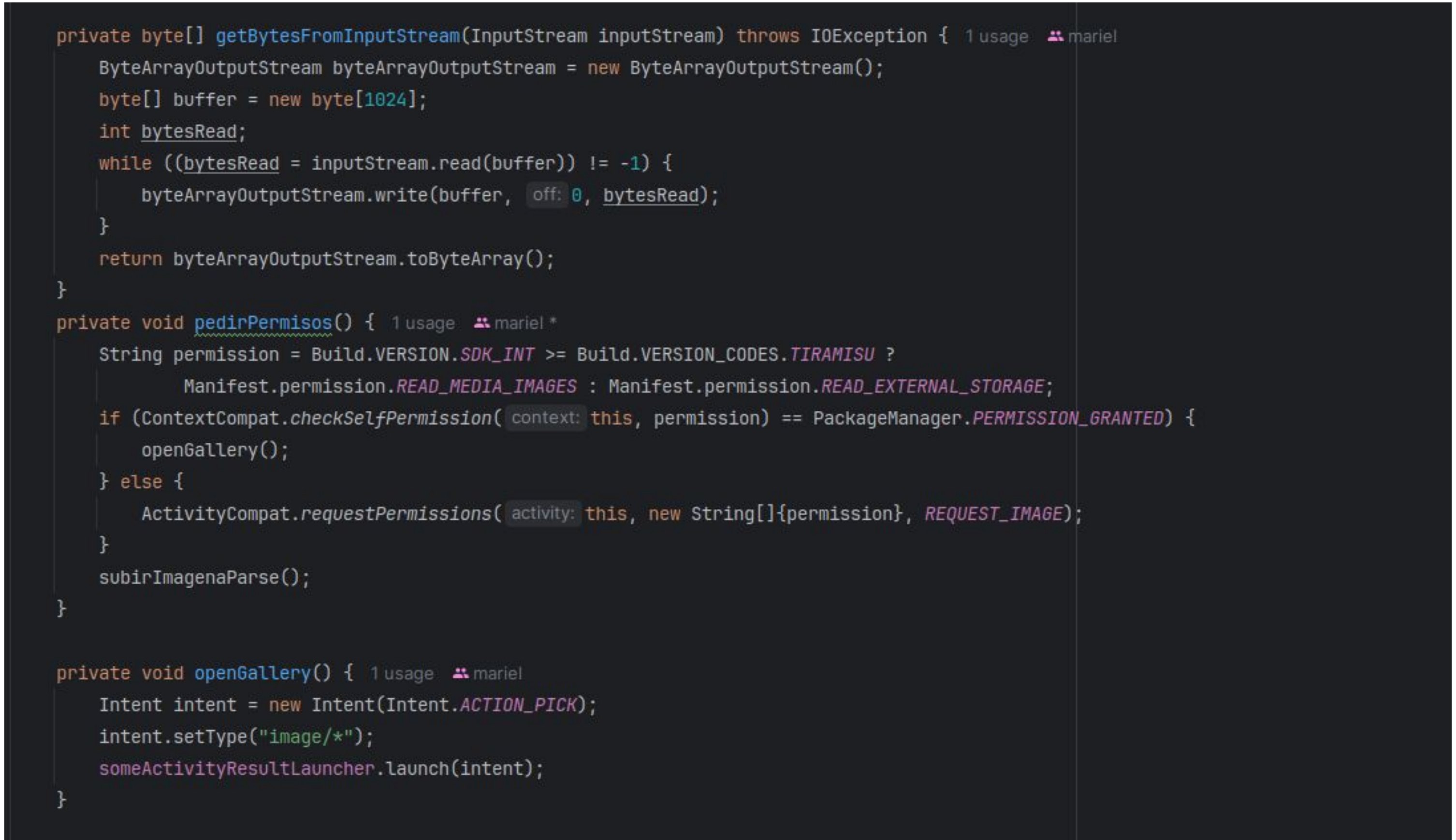
```java
private void setupCategorySpinner() {  1 usage  mariel *
    ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this, R.layout.spinner_item, getResources().getStringArray(R.array.categorias_array));
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    binding.spinnerCategoria.setAdapter(adapter);
    binding.spinnerCategoria.setSelection(0);
    binding.spinnerCategoria.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  mariel *
        @Override  1 usage  mariel
        public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {
            categoria = (String) parentView.getItemAtPosition(position);
        }
        @Override  1 usage  mariel
        public void onNothingSelected(AdapterView<?> parentView) {}
    });
}

private void setupViewModel() {  1 usage  mariel *
    uploadViewModel = new ViewModelProvider( owner: this).get(UploadViewModel.class);
    uploadViewModel.getPostSuccess().observe( owner: this, success -> {
        Toast.makeText( context: this, success ? "Publicación exitosa" : "Error al publicar", Toast.LENGTH_SHORT).show();
        Log.d( tag: "TAG#",  msg: "Post upload success: " + success);
        if (success) {
            Toast.makeText( context: this,  text: "Post published successfully", Toast.LENGTH_SHORT).show();
            finish();
        } else {
            Toast.makeText( context: this,  text: "Error while publishing", Toast.LENGTH_SHORT).show();
        }
    });
}
```

UploadActivity.java



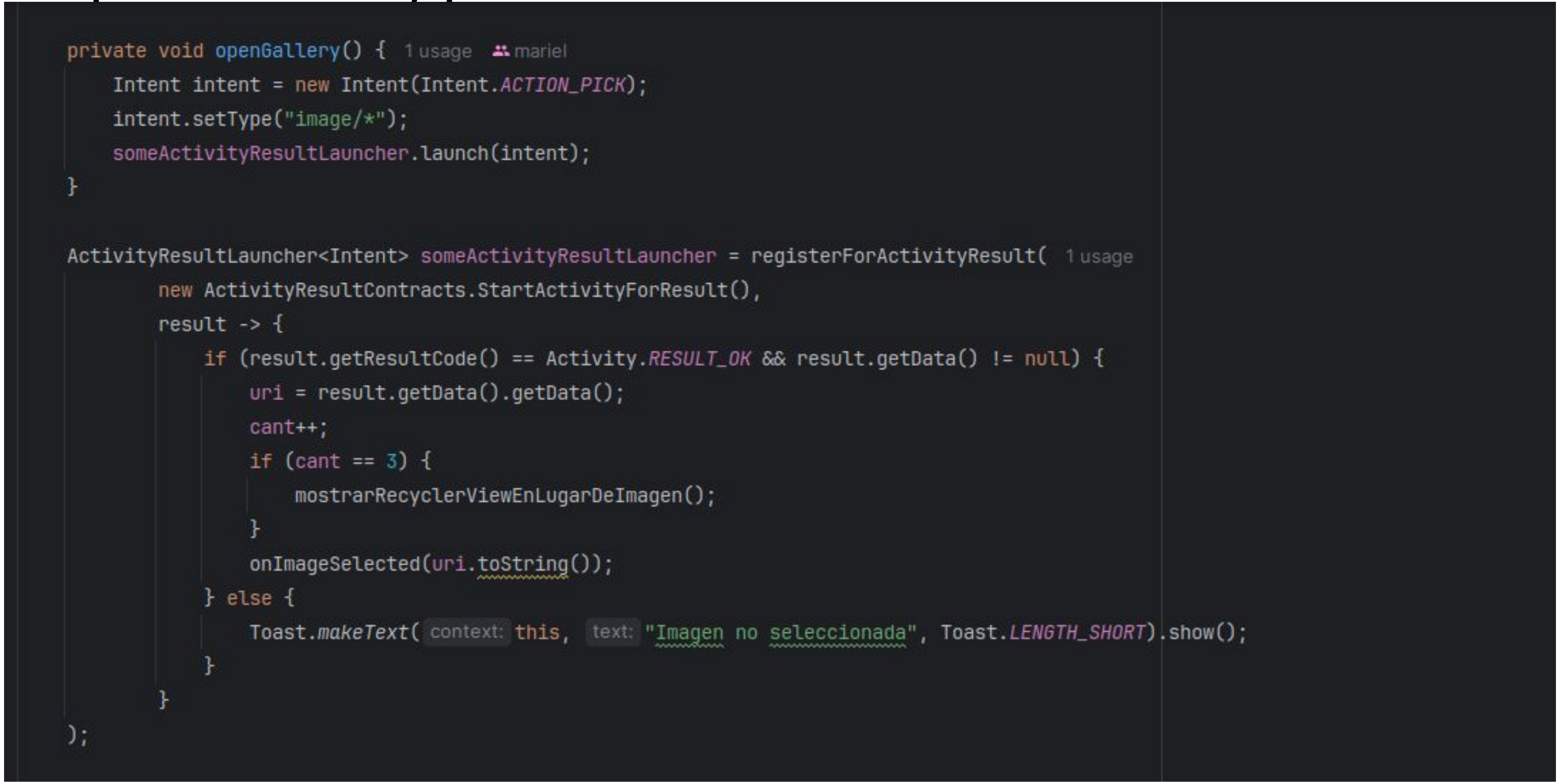
```
private void subirImagenaParse() {  1 usage  mariel *
    if (uri != null) {
        try (InputStream inputStream = getContentResolver().openInputStream(uri)) {
            byte[] imageBytes = getBytesFromInputStream(inputStream);
            ParseFile parseFile = new ParseFile( name: "image.jpg", imageBytes);
            parseFile.saveInBackground(new SaveCallback() {  mariel *
                @Override  mariel *
                public void done(ParseException e) {
                    if (e == null) {
                        String imageUrl = parseFile.getUrl();
                        imageViewModel.agregarImagenUrl(imageUrl);
                        imagenesUrls.add(imageUrl);
                    } else {
                        Toast.makeText( context: UploadActivity.this,  text: "Error uploading the image", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText( context: this,  text: "Error al procesar la imagen", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText( context: this,  text: "Seleccione una imagen primero", Toast.LENGTH_SHORT).show();
    }
}
```

UploadActivity.java



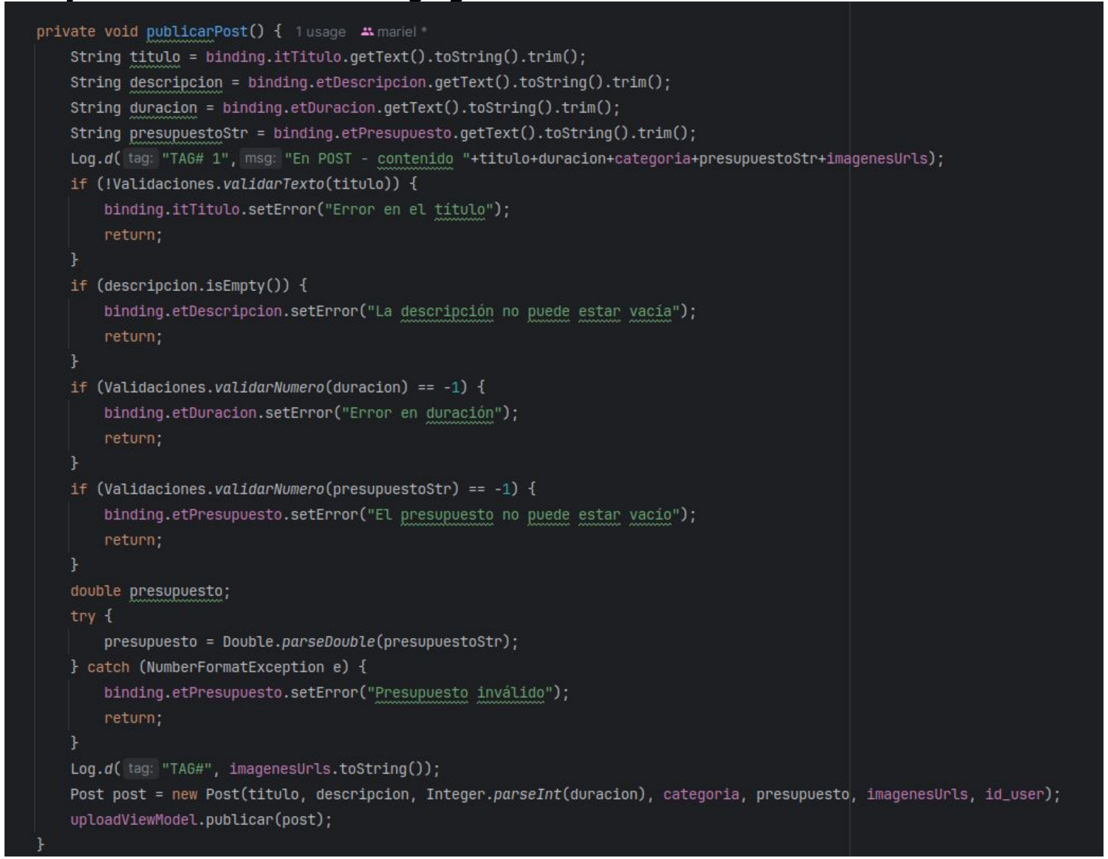
```java
private byte[] getBytesFromInputStream(InputStream inputStream) throws IOException {  1 usage  mariel
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    byte[] buffer = new byte[1024];
    int bytesRead;
    while ((bytesRead = inputStream.read(buffer)) != -1) {
        byteArrayOutputStream.write(buffer,  off: 0, bytesRead);
    }
    return byteArrayOutputStream.toByteArray();
}
private void pedirPermisos() {  1 usage  mariel *
    String permission = Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU ?
            Manifest.permission.READ_MEDIA_IMAGES : Manifest.permission.READ_EXTERNAL_STORAGE;
    if (ContextCompat.checkSelfPermission( context: this, permission) == PackageManager.PERMISSION_GRANTED) {
        openGallery();
    } else {
        ActivityCompat.requestPermissions( activity: this, new String[]{permission}, REQUEST_IMAGE);
    }
    subirImagenaParse();
}

private void openGallery() {  1 usage  mariel
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType("image/*");
    someActivityResultLauncher.launch(intent);
}
```

UploadActivity.java



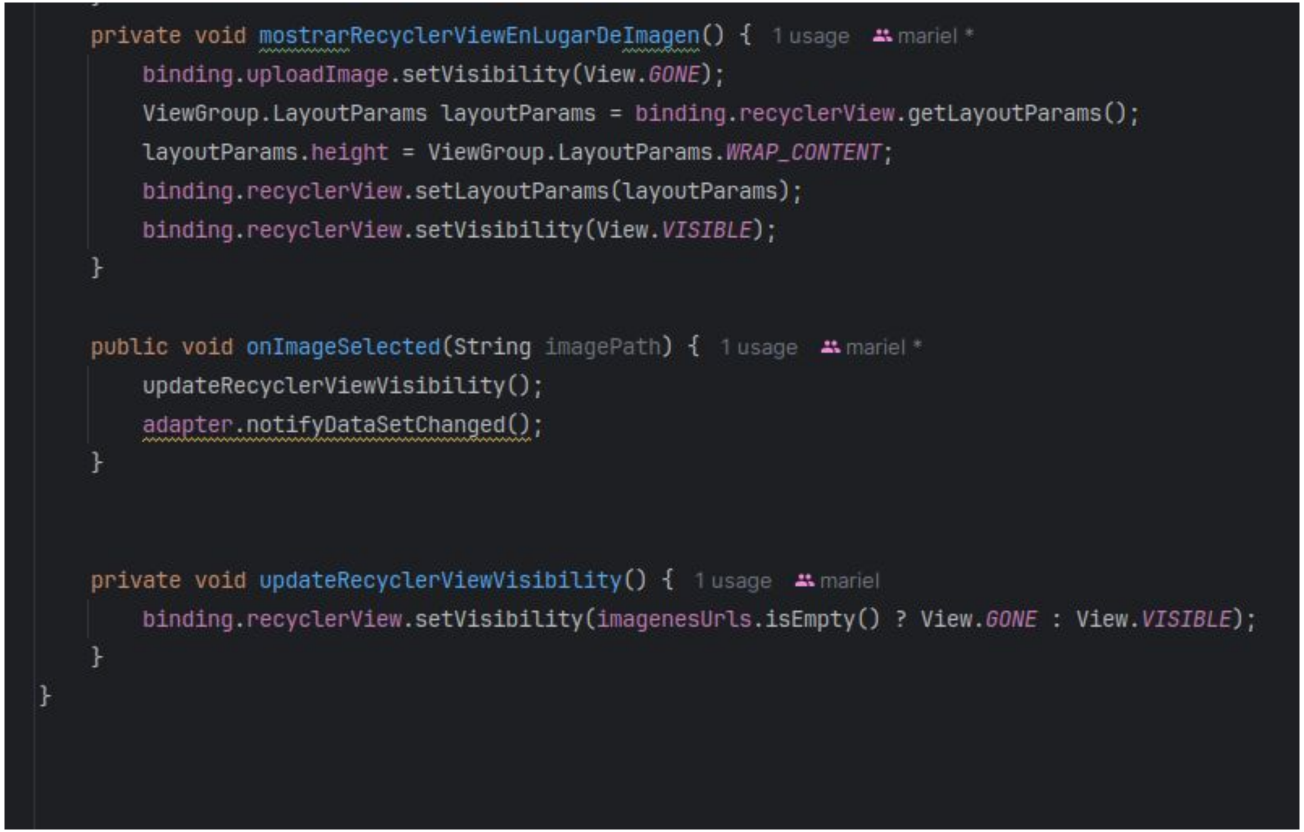
```
private void openGallery() {  1 usage  mariel
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType("image/*");
    someActivityResultLauncher.launch(intent);
}

ActivityResultLauncher<Intent> someActivityResultLauncher = registerForActivityResult(  1 usage
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == Activity.RESULT_OK && result.getData() != null) {
                uri = result.getData().getData();
                cant++;
                if (cant == 3) {
                    mostrarRecyclerViewEnLugarDeImagen();
                }
                onImageSelected(uri.toString());
            } else {
                Toast.makeText( context: this,  text: "Imagen no seleccionada", Toast.LENGTH_SHORT).show();
            }
        }
);
```

UploadActivity.java



```
private void publicarPost() {  1 usage  mariel *
    String titulo = binding.itTitulo.getText().toString().trim();
    String descripcion = binding.etDescripcion.getText().toString().trim();
    String duracion = binding.etDuracion.getText().toString().trim();
    String presupuestoStr = binding.etPresupuesto.getText().toString().trim();
    Log.d( tag: "TAG# 1", msg: "En POST - contenido "+titulo+duracion+categoria+presupuestoStr+imagenesUrls);
    if (!Validaciones.validarTexto(titulo)) {
        binding.itTitulo.setError("Error en el título");
        return;
    }
    if (descripcion.isEmpty()) {
        binding.etDescripcion.setError("La descripción no puede estar vacía");
        return;
    }
    if (Validaciones.validarNumero(duracion) == -1) {
        binding.etDuracion.setError("Error en duración");
        return;
    }
    if (Validaciones.validarNumero(presupuestoStr) == -1) {
        binding.etPresupuesto.setError("El presupuesto no puede estar vacío");
        return;
    }
    double presupuesto;
    try {
        presupuesto = Double.parseDouble(presupuestoStr);
    } catch (NumberFormatException e) {
        binding.etPresupuesto.setError("Presupuesto inválido");
        return;
    }
    Log.d( tag: "TAG#", imagenesUrls.toString());
    Post post = new Post(titulo, descripcion, Integer.parseInt(duracion), categoria, presupuesto, imagenesUrls, id_user);
    uploadViewModel.publicar(post);
}
```

UploadActivity.java



```
    private void mostrarRecyclerViewEnLugarDeImagen() {  1 usage  mariel *
        binding.uploadImage.setVisibility(View.GONE);
        ViewGroup.LayoutParams layoutParams = binding.recyclerView.getLayoutParams();
        layoutParams.height = ViewGroup.LayoutParams.WRAP_CONTENT;
        binding.recyclerView.setLayoutParams(layoutParams);
        binding.recyclerView.setVisibility(View.VISIBLE);
    }

    public void onImageSelected(String imagePath) {  1 usage  mariel *
        updateRecyclerViewVisibility();
        adapter.notifyDataSetChanged();
    }


    private void updateRecyclerViewVisibility() {  1 usage  mariel
        binding.recyclerView.setVisibility(imagenesUrls.isEmpty() ? View.GONE : View.VISIBLE);
    }
}
```