RAVENTALE

Tutor: Ángel Nieto Ros

Alumno: Diego Romero Ramal

ÍNDICE

¿QUÉ ES *RAVENTALE*?

Introducción

¿CUÁL ES LA NECESIDAD?

Introducción

¿A QUE ASPIRAMOS?

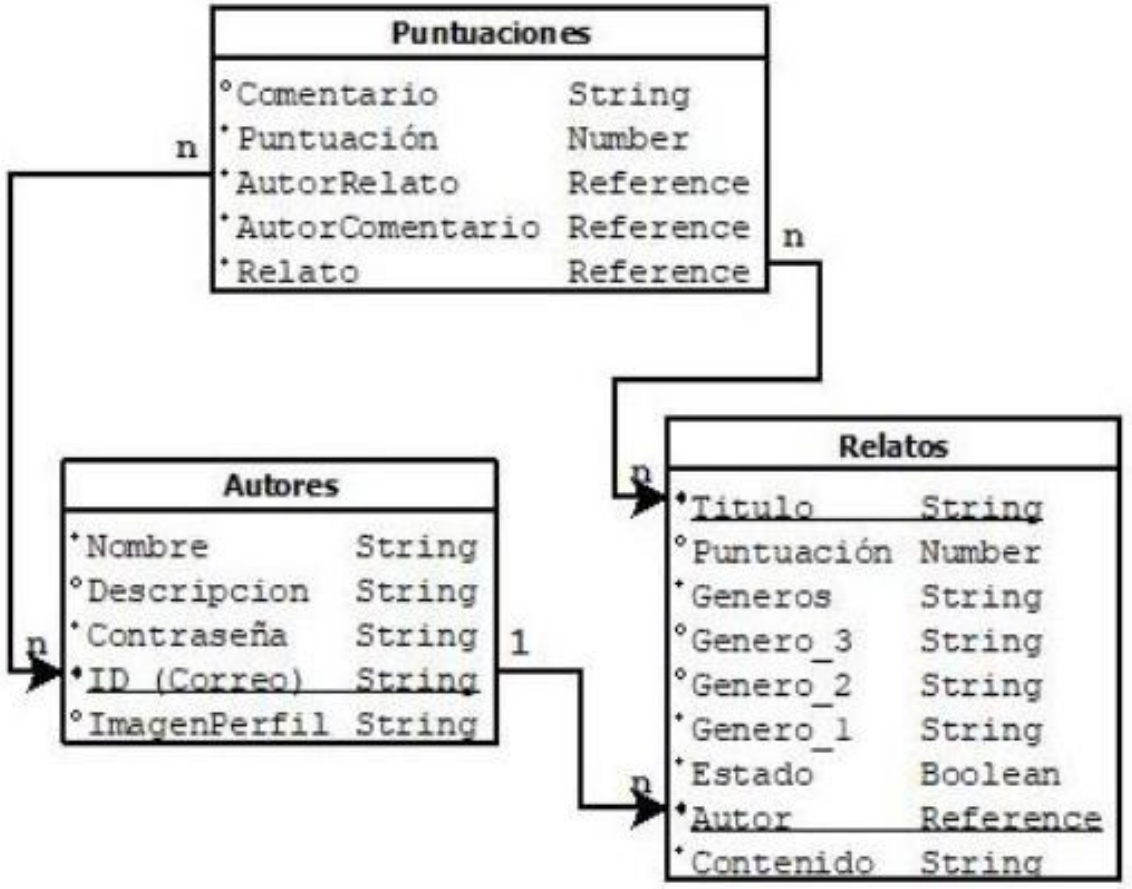
Herramientas y medios

- Base de Datos
- Entorno de Desarrollo
- Lenguaje de programación
- Librerías
- Metodología
- Pruebas reales

Base de Datos

Firebase: Permite crear bases de datos NoSQL escalables de gran rendimiento, con una arquitectura *BaaS (Backend-as-a-Service).* Incluye servicios rápidos como la autentificación sencilla, *Google Analitycs,* almacenaje en la nube…

Arquitectura



- Un autor no podrá repetir el mismo título de relato para otro diferente
- Un autor solo podrá seleccionar un máximo de tres géneros por relato
- Un relato será siempre de un solo autor, mientras que un autor tendrá tantos relatos como quiera
- Un relato no podrá ser publicado sin tener contenido
- Todo el mundo podrá comentar y puntuar cualquier relato
- La contraseña no puede ser menor a seis carácteres

Entorno de desarrollo

Android Studio:

- IDE de compilación rápida
- Renderizado en tiempo real con *ProGuard*
- Integración con *Grandle*
- Entorno dedicado en exclusiva a Android
- Vista previa del programa en diferentes dispositivos y ejecución directa en dispositivos móviles
- Editor de diseño
- Requisitos de instalación bajos

Lenguaje de programación

Java: Lenguaje de programación muy usado, con mucha documentación, con una gran comunidad de desarrolladores y con oportunidades de expandir el programa más allá de Android Studio

Lenguajes de programación

JAVA	KOTLIN
Lenguaje de programación más expresivo y con más líneas de código	Lenguaje de programación más sencillo y menos líneas de código
Ampliamente utilizado	En crecimiento y popularidad
Compatible con todas las plataformas	Interoperable con Java
Gran cantidad de herramientas y bibliotecas	Pocas herramientas y bibliotecas a su disposición

Librerías

- Recycler:
- CardView
- Firebase
- Lottie
- PDFBox
- Glide
- Material

Metodología Kanban

Nuevo

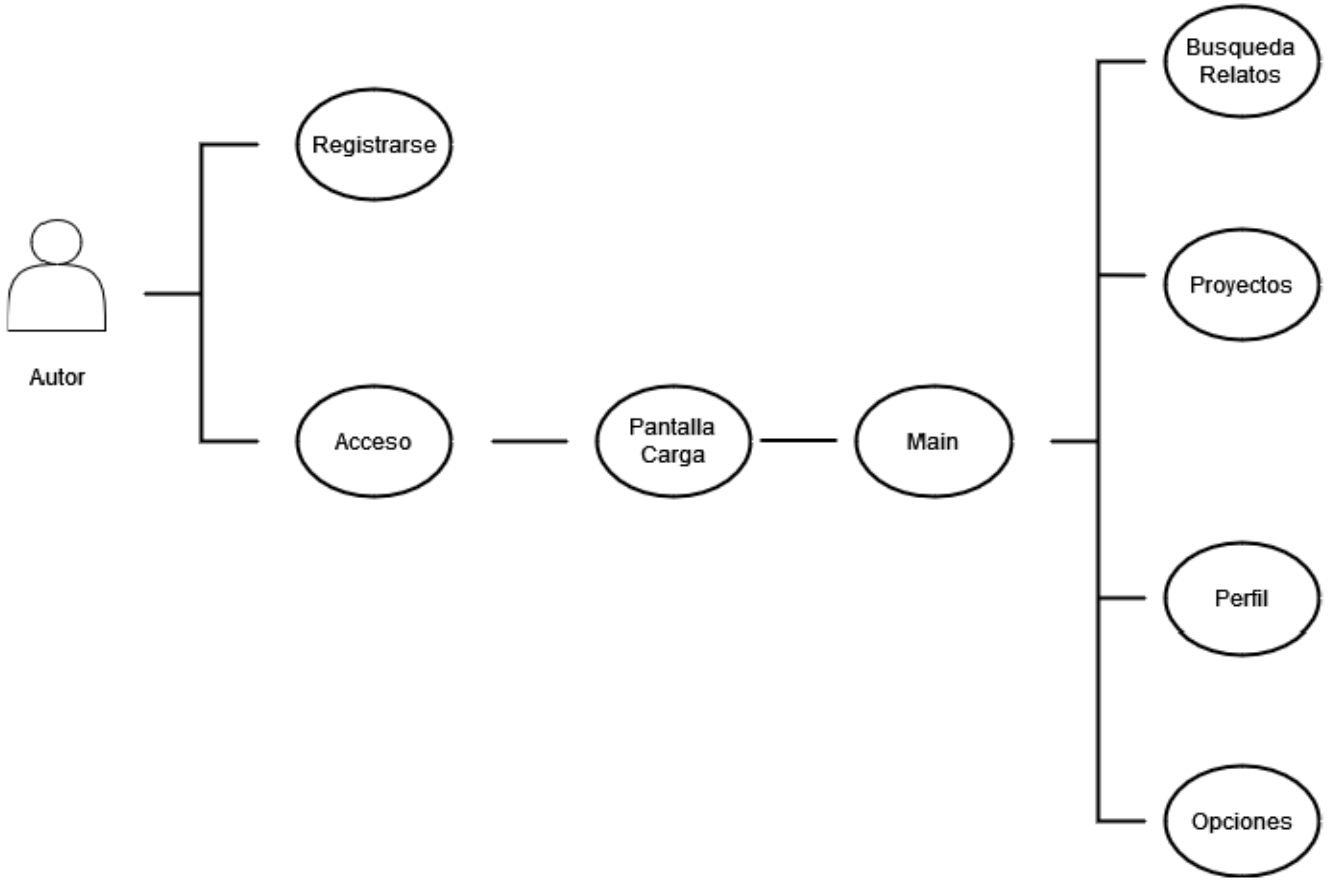
Trabajo en progreso

En espera

Finalizadas

Pruebas reales

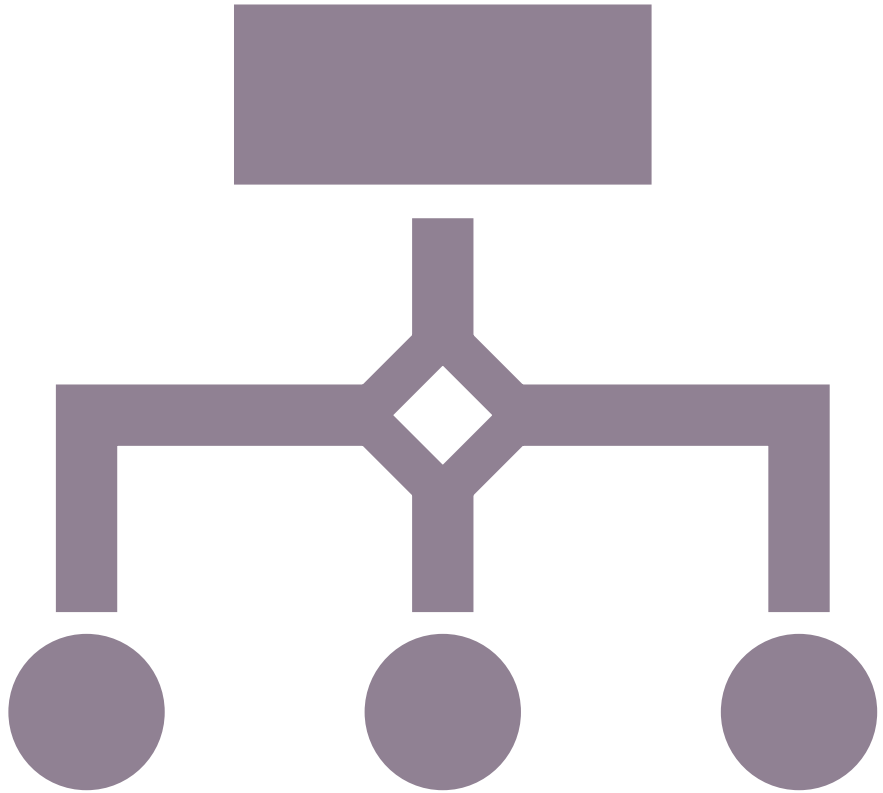
Son las pruebas realistas de la aplicación llevadas al entorno real, pruebas sujetas a un grupo cerrado de usuarios que se encargan de probar la aplicación y darte un *feedback* sobre ella



ARQUITECTURA Y FUNCIONALIDAD

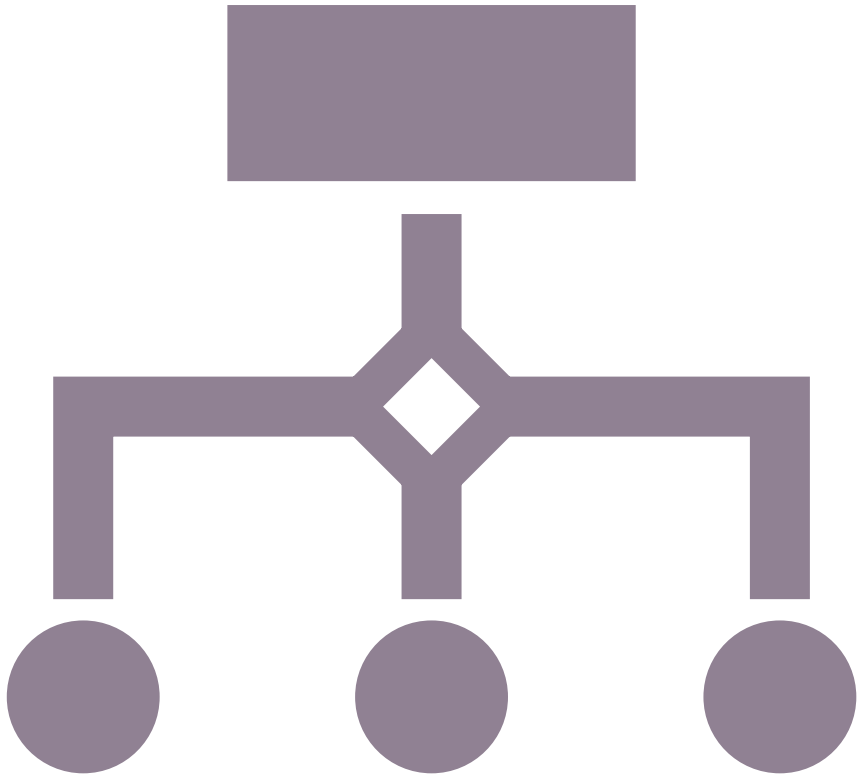
Clase “*Registrarse*”

Dicha clase se encarga de registrar al usuario en el caso de que este no tenga una cuenta gestionada en la base de datos



Método de registro de usuario

```java
//Registrar usuario en la base de datos
private void registrarEnBaseDeDatos() {
    //Creamos una instancia del servicio de autentificación
    aut = FirebaseAuth.getInstance();
    //Pasamos los parametros al servicio
    aut.createUserWithEmailAndPassword(gmail.getText().toString(), contrasenya.getText().toString()).addOnCompleteListener(new OnCompleteListener<
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            //Creamos un map con los valores que queremos almacenar en la base de datos
            Map<String, Object> map = new HashMap<>();
            String id = aut.getCurrentUser().getUid();
            map.put("id", id);
            map.put("Name", nombre.getText().toString());
            map.put("Password", contrasenya.getText().toString());
            map.put("Gmail", gmail.getText().toString());
            map.put("Description", "");

            //Hacemos una consulta a la base de datos
            firebase.collection( collectionPath: "autores").document(id).set(map).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    //En caso de que sea un exito
                    Toast.makeText(getApplicationContext(),  text: "Guardado con exito", Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    //En caso de que falle el registro
                    Toast.makeText(getApplicationContext(),  text: "Error a la hora de registrarse en la base de datos", Toast.LENGTH_LONG).show();
                }
            });
        }
```



Clase "*Acceso*"

Dicha clase se encarga de validar y comprobar que dicho usuario existe en la base de datos

Métodos relativos al acceso del usuario

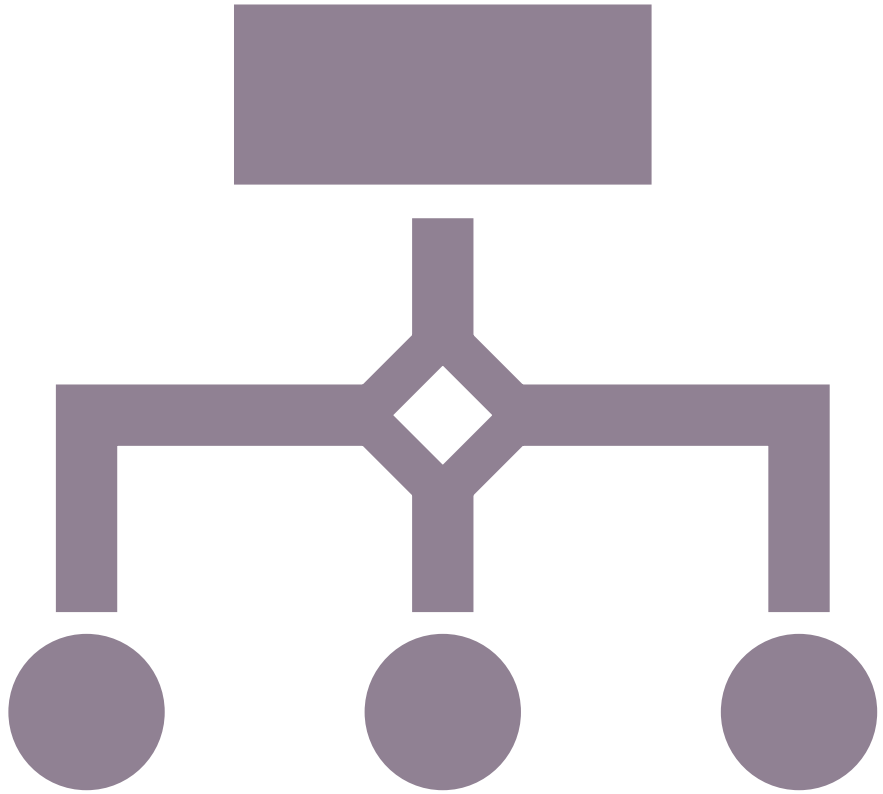
```
//Metodo para comprobar la base de datos
private void comprobarrBaseDeDatosParaEntrarFalse() {
    aut = FirebaseAuth.getInstance();
    aut.signInWithEmailAndPassword(gmail.getText().toString(), pass.getText().toString())
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()){
                        loginExitoso = true;
                        //Llamamos al submetodo login
                        login();
                    }else{
                        loginExitoso = false;
                        //Llamamos al submetodo login
                        login();
                    }
                }
            });
}
```

```
private void login() {
    System.out.println(loginExitoso);
    //En caso de que el login sea exitoso
    if (loginExitoso){
        //cambiamos al main
        Intent Cambio = new Intent(getApplicationContext(), login_complete.class);
        startActivity(Cambio);
        Toast.makeText(getApplicationContext(), text: "¡HOLA!", Toast.LENGTH_LONG).s
    }else{
        //Fallo
        Toast.makeText(getApplicationContext(), text: "Error. Campos introducidos er
    }
}
```

Clase
“login_complete”

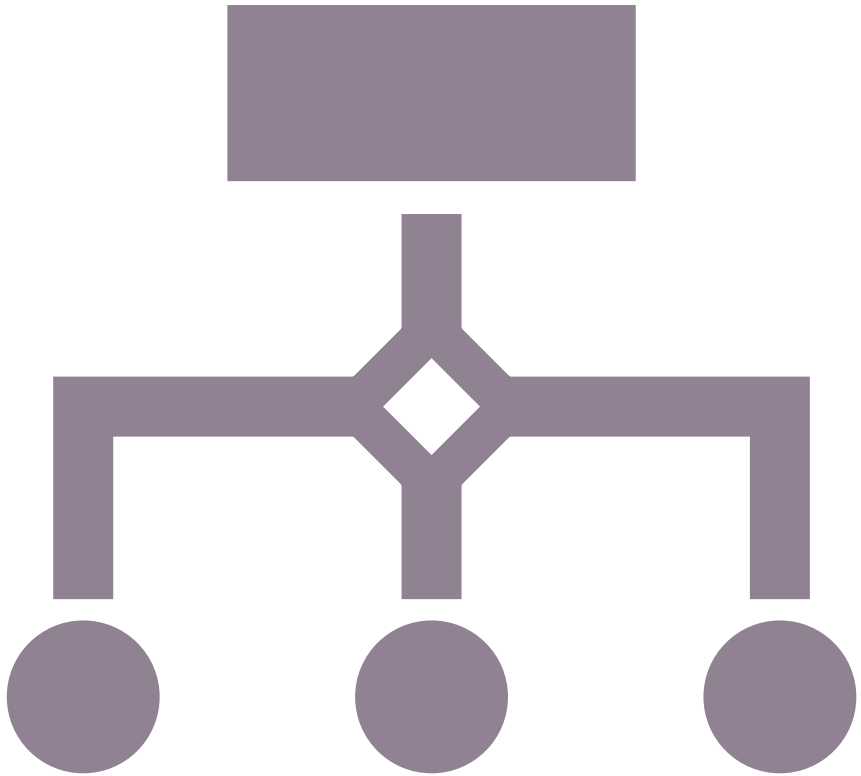
Dicha clase es una clase de corto tiempo que mostrará una animación y el icono emblemático de la aplicación

Mostrará también un conjunto de frases motivadoras e inspiradoras



Fragmento de código relacionado con la clase

```
TextView cit = findViewById(R.id.textView18);
Random r = new Random();
String citaAleatoria = citas.get(r.nextInt(citas.size()));
cit.setText(citaAleatoria);

cit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String citaAleatoria = citas.get(r.nextInt(citas.size()));
        cit.setText(citaAleatoria);
    }
});

/**
 * Crearemos un proceso. Dicho proceso servirá para esperar hasta que termine el tiempo.
 * El tiempo está medido con el Timer, y con el pasamos la tarea que tiene que ejercer una vez pase el tiempo
 * En este caso, tiene 6.5 segundos para iniciar la tarea.
 * Una vez iniciamos la tarea (con el run) hacemos un intent a la clase que será la principal. El menú
 */
TimerTask tarea = () -> {
    Intent intent = new Intent( packageContext: login_complete.this, MainActivity.class);
    startActivity(intent);
    finish();
;
//Temporizador (6.5 seg)
Timer tiempo = new Timer();
tiempo.schedule(tarea,  delay: 7000);
```



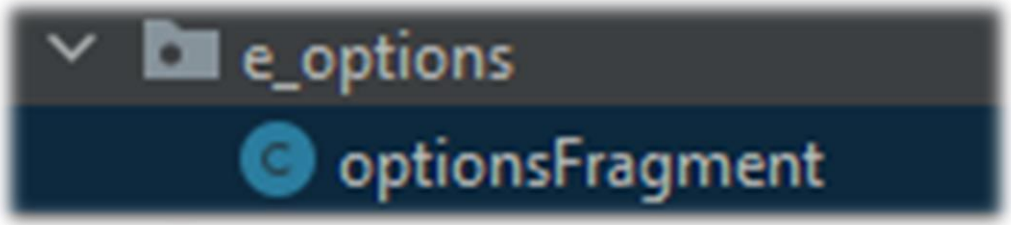
Clase "*relato*" y "*Main*"

Relato: Clase modelo que servirá para pasar datos a los recyclerView como a la base de datos

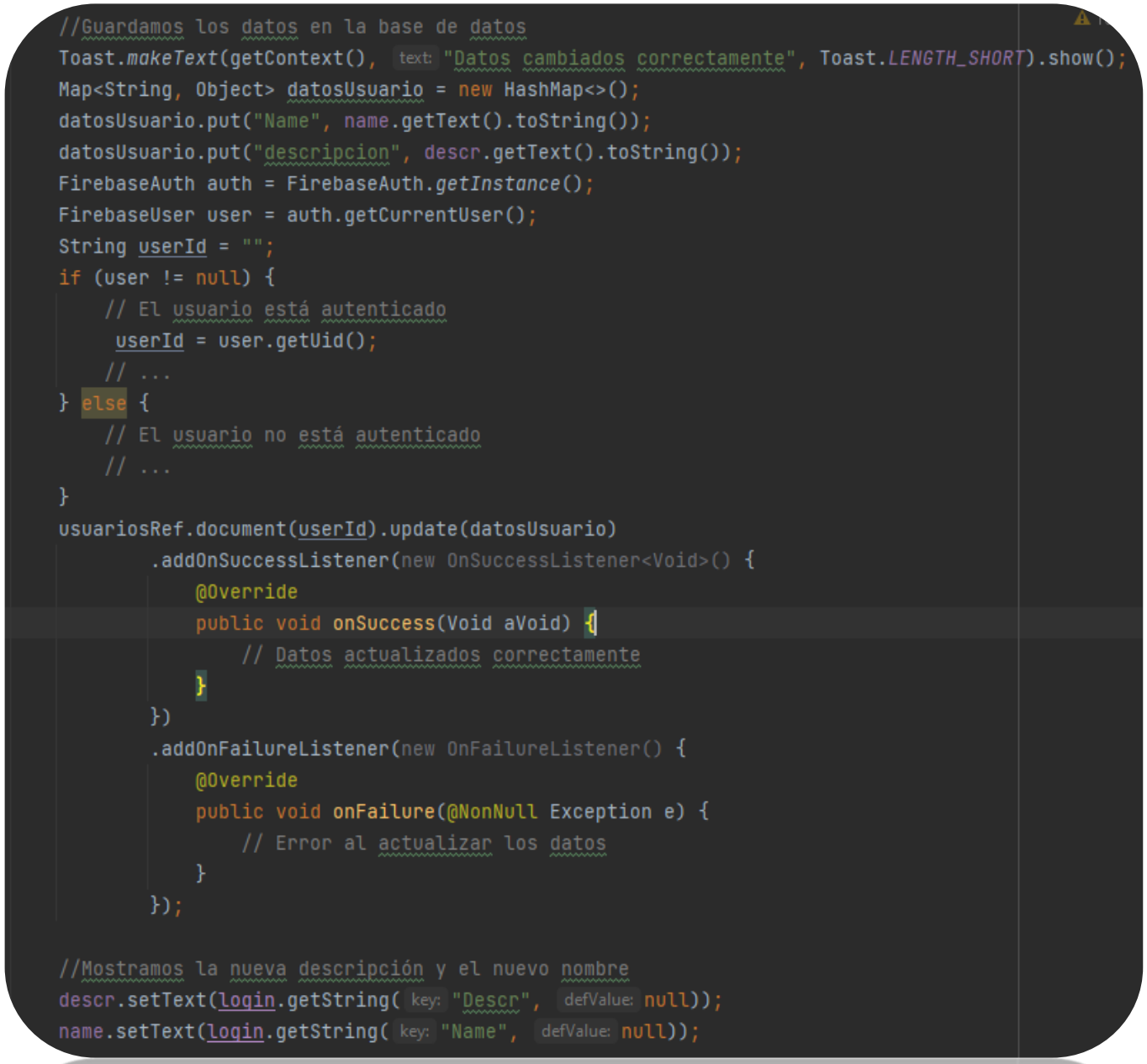
Main: Clase encargada de actualizar visualmente los diferentes fragments

Carpeta Opciones

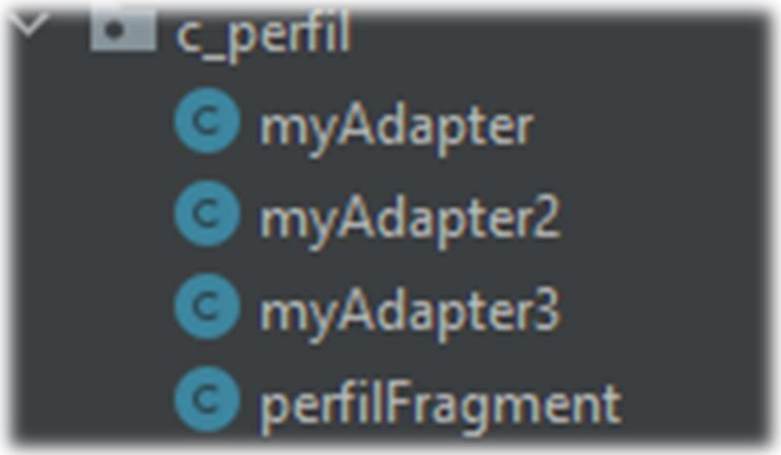
Dicha carpeta contendrá la actividad de OpcionsFragment, encargada de editar el nombre y la descripción del usuario



Método encargado de guardar y actualizar los datos tanto visuales como en la base de datos



```
//Guardamos los datos en la base de datos
Toast.makeText(getContext(), text: "Datos cambiados correctamente", Toast.LENGTH_SHORT).show();
Map<String, Object> datosUsuario = new HashMap<>();
datosUsuario.put("Name", name.getText().toString());
datosUsuario.put("descripcion", descr.getText().toString());
FirebaseAuth auth = FirebaseAuth.getInstance();
FirebaseUser user = auth.getCurrentUser();
String userId = "";
if (user != null) {
    // El usuario está autenticado
     userId = user.getUid();
    // ...
} else {
    // El usuario no está autenticado
    // ...
}
usuariosRef.document(userId).update(datosUsuario)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Datos actualizados correctamente
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Error al actualizar los datos
            }
        });

//Mostramos la nueva descripción y el nuevo nombre
descr.setText(login.getString( key: "Descr",  defValue: null));
name.setText(login.getString( key: "Name",  defValue: null));
```



Carpeta Perfil

Carpeta que contiene todas las clases y adaptadores necesarios para los recycler, para así poder mostrar información mucho más detallada.

También se permite seleccionar una imagen diferente para tu perfil, se guardará directamente en la base de datos

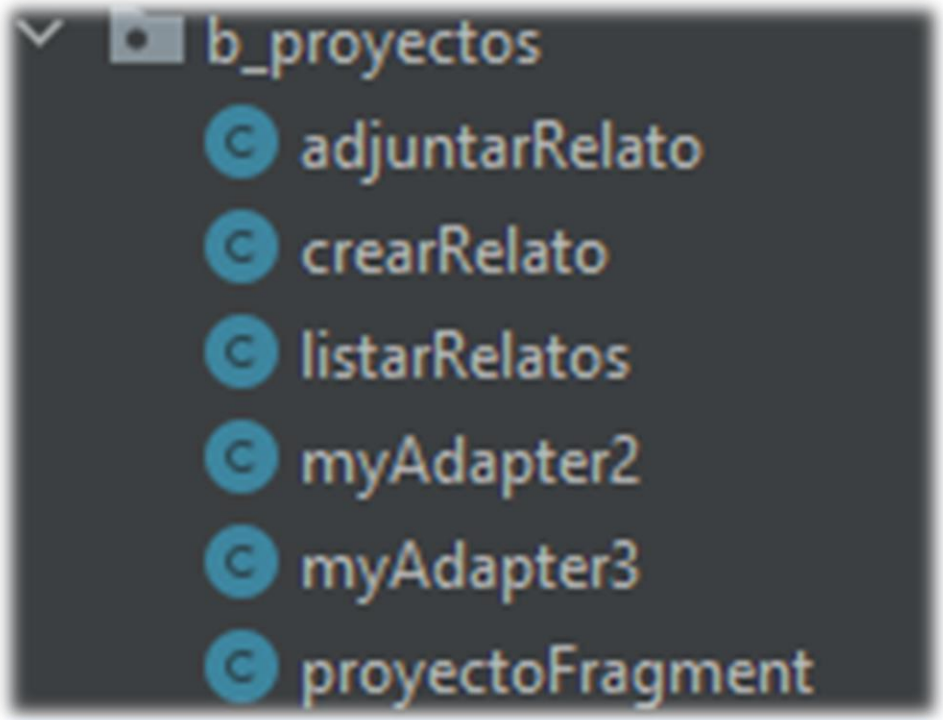
Métodos de actualizar icono y cargar icono con Glide

```java
String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
StorageReference storageRef = storage.getReference();
String storagePath = "autores/" + uid + "/imagen.jpg";
StorageReference imageRef = storageRef.child(storagePath);

imageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
    @Override
    public void onSuccess(Uri uri) {
        Glide.with(getContext()) RequestManager
                .load(uri) RequestBuilder<Drawable>
                .override( width: 305,  height: 183)
                .centerCrop()
                .override( width: 300,  height: 300)
                .into(binding.profileImage);
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        binding.profileImage.setImageResource(R.drawable.iconnav);
    }
```

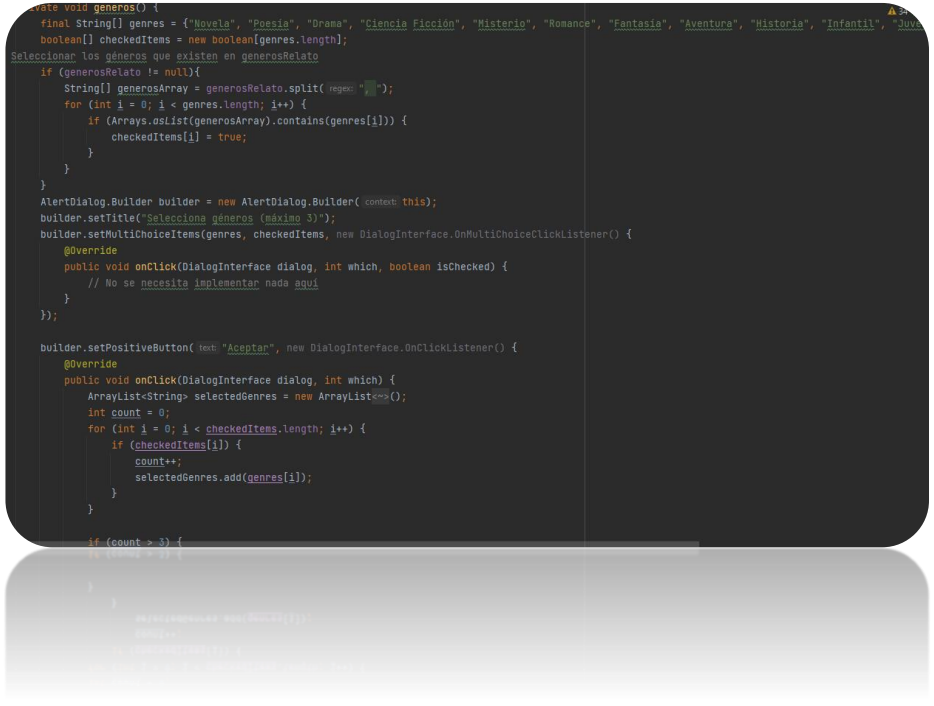
```java
getActivity().getWindow().setFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE,
        WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
FirebaseStorage storage = FirebaseStorage.getInstance();
StorageReference storageRef = storage.getReference();
String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
String storagePath = "autores/" + uid + "/imagen.jpg";
StorageReference imageRef = storageRef.child(storagePath);
imageRef.putFile(selectedImageUri) UploadTask
        .addOnSuccessListener(taskSnapshot -> {
            // Obtener la URL de descarga de la imagen
            imageRef.getDownloadUrl().addOnSuccessListener(uri -> {
                // Guardar la URL en la base de datos de Firebase Firestore
                FirebaseFirestore db = FirebaseFirestore.getInstance();
                DocumentReference userRef = db.collection( collectionPath: "autores").document(uid);
                userRef.get().addOnCompleteListener(task -> {
                    if (task.isSuccessful() && task.getResult() != null) {
                        DocumentSnapshot document = task.getResult();
                        if (document.exists()) {
                            userRef.update( field: "profileImageUrl", uri.toString())
                                    .addOnSuccessListener(aVoid -> {
                                        progressDialog.dismiss();
                                        Toast.makeText(getContext(),  text: "Imagen subida correctamente", Toast.LENGTH_SHORT).show();
                                        // Habilitar interacción del usuario con la pantalla
                                        getActivity().getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                                    })
                                    .addOnFailureListener(e -> {
                                        progressDialog.dismiss();
                                        Toast.makeText(getContext(),  text: "Error al guardar la URL de la imagen", Toast.LENGTH_SHORT).show();
                                        // Habilitar interacción del usuario con la pantalla
                                        // Habilitar interacción del usuario con la pantalla
                                        getActivity().getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                                    });
                        } else {
                            progressDialog.dismiss();
                            Toast.makeText(getContext(),  text: "No existe el usuario en la base de datos", Toast.LENGTH_SHORT).show();
                            // Habilitar interacción del usuario con la pantalla
                            getActivity().getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                        }
                    } else {
                        progressDialog.dismiss();
                        Toast.makeText(getContext(),  text: "Error al obtener el usuario de la base de datos", Toast.LENGTH_SHORT).show();
                        // Habilitar interacción del usuario con la pantalla
                        getActivity().getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    }
                });
            }) StorageTask<UploadTask.TaskSnapshot>
            .addOnFailureListener(e -> Toast.makeText(getContext(),  text: "Error al subir la imagen", Toast.LENGTH_SHORT).show());
    getActivity().getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
}
}
```

Carpeta Proyectos

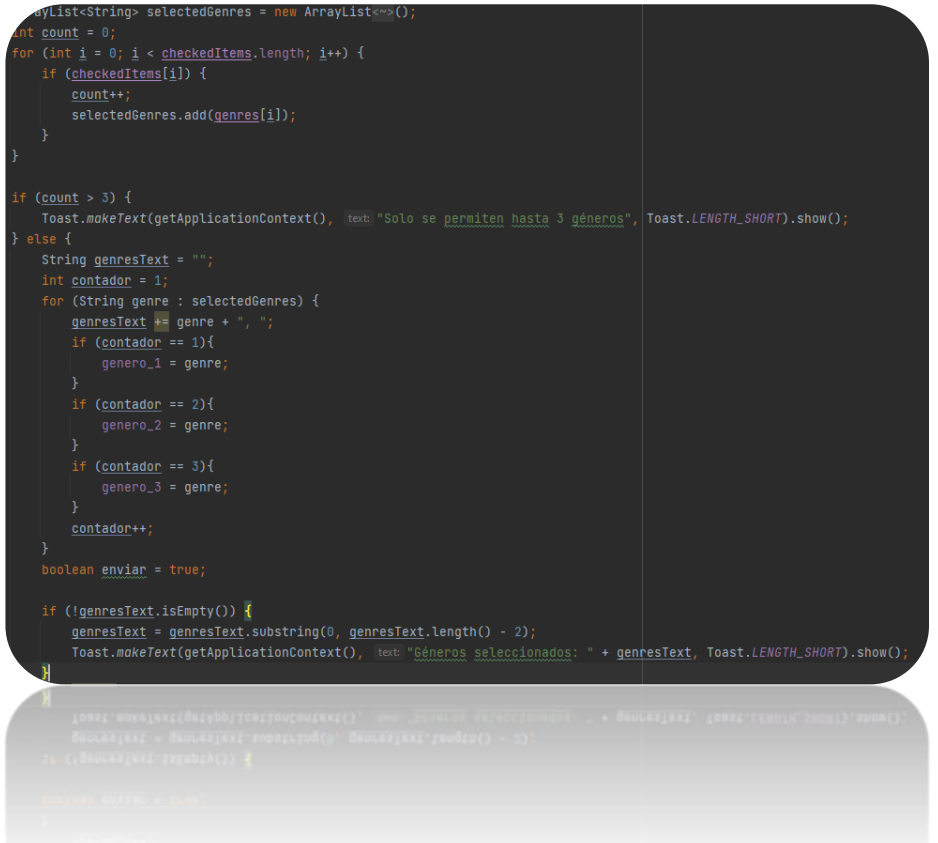
Carpeta que contiene, igual que la anterior, todas las clases y adaptadores para desarrollar las diferentes acciones para poder crear, editar, publicar, cargar y borrar relatos



Método de seleccionar género



```
ivate void generos() {
    final String[] genres = {"Novela", "Poesia", "Drama", "Ciencia Ficción", "Misterio", "Romance", "Fantasia", "Aventura", "Historia", "Infantil", "Juve
    boolean[] checkedItems = new boolean[genres.length];
Seleccionar los géneros que existen en generosRelato
    if (generosRelato != null){
        String[] generosArray = generosRelato.split( regex: ", ");
        for (int i = 0; i < genres.length; i++) {
            if (Arrays.asList(generosArray).contains(genres[i])) {
                checkedItems[i] = true;
            }
        }
    }
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Selecciona géneros (máximo 3)");
    builder.setMultiChoiceItems(genres, checkedItems, new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which, boolean isChecked) {
            // No se necesita implementar nada aquí
        }
    });

    builder.setPositiveButton( text: "Aceptar", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ArrayList<String> selectedGenres = new ArrayList<>();
            int count = 0;
            for (int i = 0; i < checkedItems.length; i++) {
                if (checkedItems[i]) {
                    count++;
                    selectedGenres.add(genres[i]);
                }
            }

            if (count > 3) {
```



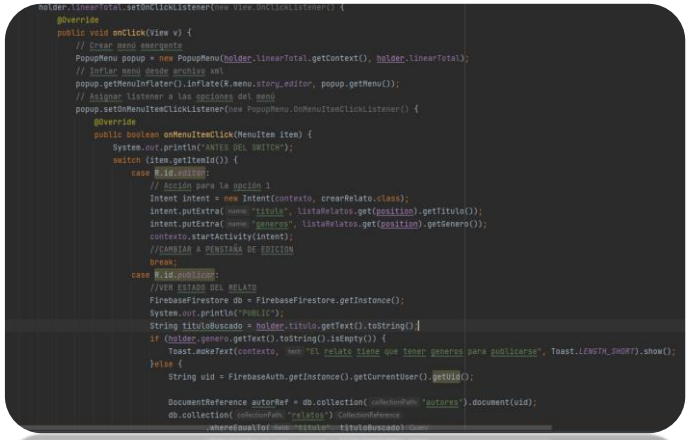
```
ayList<String> selectedGenres = new ArrayList<>();
int count = 0;
for (int i = 0; i < checkedItems.length; i++) {
    if (checkedItems[i]) {
        count++;
        selectedGenres.add(genres[i]);
    }
}

if (count > 3) {
    Toast.makeText(getApplicationContext(), text: "Solo se permiten hasta 3 géneros", Toast.LENGTH_SHORT).show();
} else {
    String genresText = "";
    int contador = 1;
    for (String genre : selectedGenres) {
        genresText += genre + ", ";
        if (contador == 1){
            genero_1 = genre;
        }
        if (contador == 2){
            genero_2 = genre;
        }
        if (contador == 3){
            genero_3 = genre;
        }
        contador++;
    }
    boolean enviar = true;

    if (!genresText.isEmpty()) {
        genresText = genresText.substring(0, genresText.length() - 2);
        Toast.makeText(getApplicationContext(), text: "Géneros seleccionados: " + genresText, Toast.LENGTH_SHORT).show();
    }
```

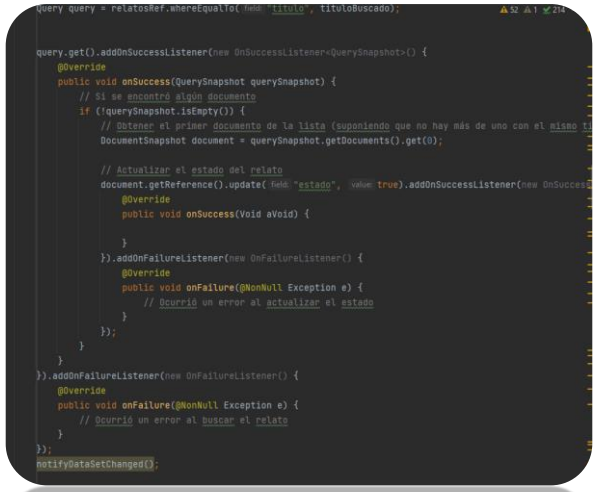
Método de adjuntar archivo PDF y leer contenido PDF

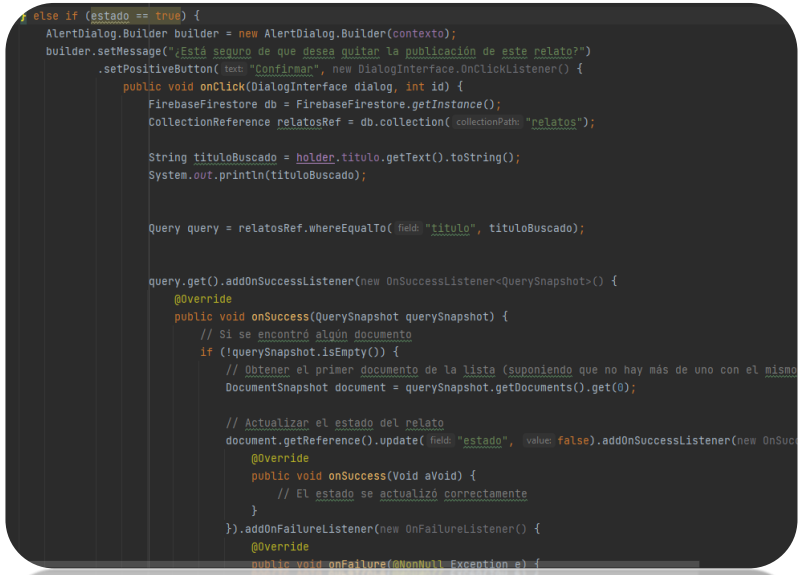
```
builder.setPositiveButton( text: "Aceptar", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        ArrayList<String> selectedGenres = new ArrayList<>();
        int count = 0;
        for (int i = 0; i < checkedItems.length; i++) {
            if (checkedItems[i]) {
                count++;
                selectedGenres.add(genres[i]);
            }
        }

        if (count > 3) {
            Toast.makeText(getApplicationContext(), text: "Solo se permiten hasta 3 géneros", Toast.LENGTH_SHORT).show();
        } else {
            String genresText = "";
            int contador = 1;
            for (String genre : selectedGenres) {
                genresText += genre + ", ";
                if (contador == 1) {
                    genero_1 = genre;
                }
                if (contador == 2) {
                    genero_2 = genre;
                }
                if (contador == 3) {
                    genero_3 = genre;
                }
                contador++;
            }
            enviar = true;
```

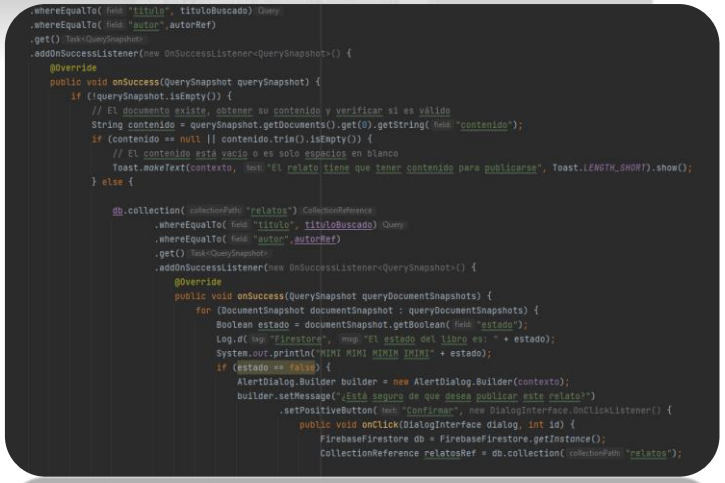
```
        if (!genresText.isEmpty()) {
            genresText = genresText.substring(0, genresText.length() - 2);
            Toast.makeText(getApplicationContext(), text: "Géneros seleccionados: " + genresText, Toast.LENGTH_SHORT).show();
        }
        if (num == 2){
            try {
                guardarPDF(titulo, genresText);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }

        }else if (num == 1){
            try {
                guardarNormal(titulo, genresText);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

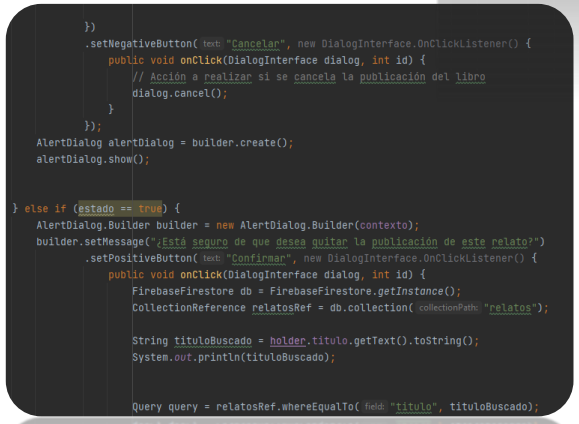
```
private String leerContenidoPDF(Uri uri) {
    try {
        InputStream inputStream = getContentResolver().openInputStream(uri);
        PDDocument document = PDDocument.load(inputStream);
        PDFTextStripper pdfStripper = new PDFTextStripper();
        String texto = pdfStripper.getText(document);
        document.close();
        return texto;
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}
```



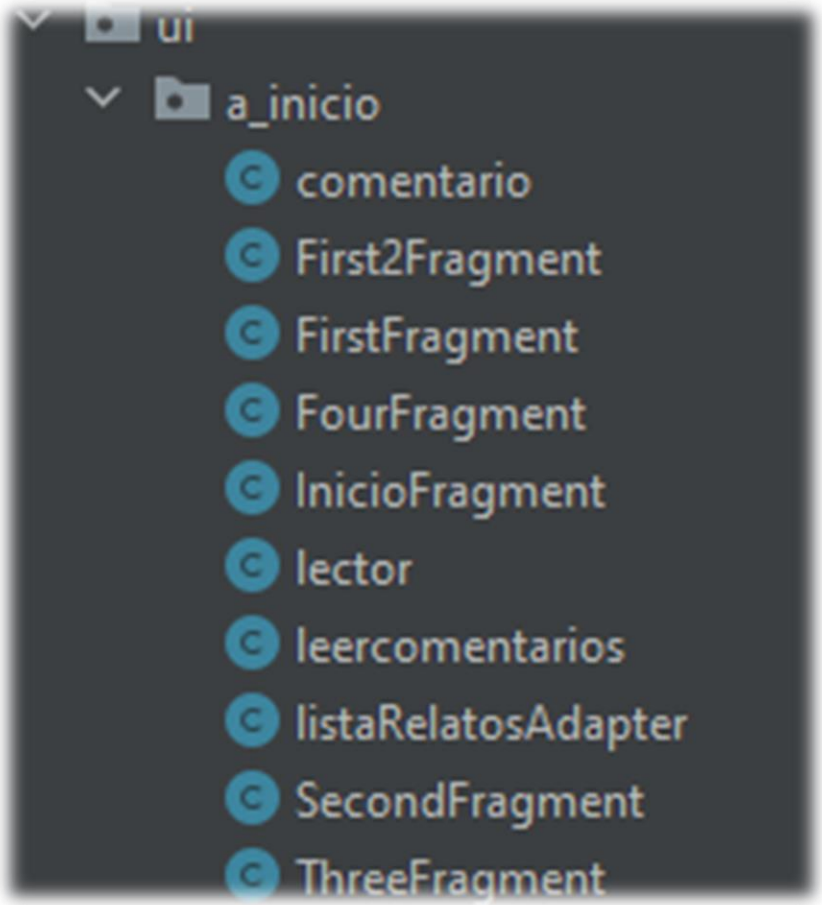






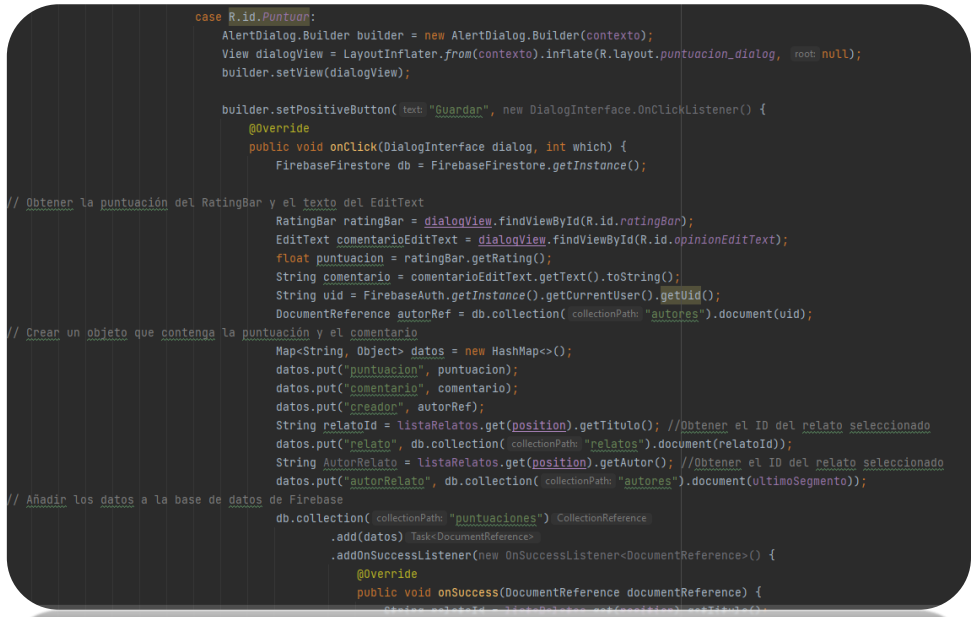


Método de Menú: Publicar, editar, borrar



Carpeta Inicio

Tiene todas las clases del fragmento de inicio, incluido el único adapter para las diferentes consultas, un visualizador para leer el relato y la clase de comentarios



```
case R.id.Puntuar:
    AlertDialog.Builder builder = new AlertDialog.Builder(contexto);
    View dialogView = LayoutInflater.from(contexto).inflate(R.layout.puntuacion_dialog, root: null);
    builder.setView(dialogView);

    builder.setPositiveButton( text: "Guardar", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            FirebaseFirestore db = FirebaseFirestore.getInstance();

// Obtener la puntuación del RatingBar y el texto del EditText
            RatingBar ratingBar = dialogView.findViewById(R.id.ratingBar);
            EditText comentarioEditText = dialogView.findViewById(R.id.opinionEditText);
            float puntuacion = ratingBar.getRating();
            String comentario = comentarioEditText.getText().toString();
            String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
            DocumentReference autorRef = db.collection( collectionPath: "autores").document(uid);
// Crear un objeto que contenga la puntuación y el comentario
            Map<String, Object> datos = new HashMap<>();
            datos.put("puntuacion", puntuacion);
            datos.put("comentario", comentario);
            datos.put("creador", autorRef);
            String relatoId = listaRelatos.get(position).getTitulo(); //Obtener el ID del relato seleccionado
            datos.put("relato", db.collection( collectionPath: "relatos").document(relatoId));
            String AutorRelato = listaRelatos.get(position).getAutor(); //Obtener el ID del relato seleccionado
            datos.put("autorRelato", db.collection( collectionPath: "autores").document(ultimoSegmento));
// Añadir los datos a la base de datos de Firebase
            db.collection( collectionPath: "puntuaciones") CollectionReference
                    .add(datos) Task<DocumentReference>
                    .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                        @Override
                        public void onSuccess(DocumentReference documentReference) {
```

```
db.collection( collectionPath: "puntuaciones") CollectionReference                    ⚠ 33  ✓ 153
        .whereEqualTo( field: "relato", db.collection( collectionPath: "relatos").document(relatoId)) Query
        .get() Task<QuerySnapshot>
        .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
            @Override
            public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                // Calcular la puntuación promedio
                float puntuacionTotal = 0;
                int cantidadTotal = queryDocumentSnapshots.size();
                System.out.println(cantidadTotal+"ESTA ES CANTIDA TOTAL");
                for (DocumentSnapshot documentSnapshot : queryDocumentSnapshots) {
                    double puntuacion = documentSnapshot.getDouble( field: "puntuacion");
                    puntuacionTotal += puntuacion;
                }
                double puntuacionPromedio = puntuacionTotal / cantidadTotal;
                System.out.println(puntuacionPromedio+"ESTA ES LA CANTIDAD PROMEDIO");

                FirebaseFirestore db = FirebaseFirestore.getInstance();
                DocumentReference autor = autorRef;
                String titulo = listaRelatos.get(position).getTitulo();

                db.collection( collectionPath: "relatos") CollectionReference
                        .whereEqualTo( field: "autor", autor) Query
                        .whereEqualTo( field: "titulo", titulo)
                        .get() Task<QuerySnapshot>
                        .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
                            @Override
                            public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                                // Verificar si se encontró un documento que coincida con los criterios de bu
                                if (!queryDocumentSnapshots.isEmpty()) {
                                    // Obtener el primer documento de la lista (suponiendo que no hay docume
                                    DocumentSnapshot relato = queryDocumentSnapshots.getDocuments().get(0);
                                    String relatoId = relato.getId();
```

Muestra de puntuación de relatos

Consulta de puntuaciones de mejor a peor y consulta básica

```
ayList<relato> relatos = new ArrayList<>();
FirebaseFirestore db = FirebaseFirestore.getInstance();
CollectionReference relatosRef = db.collection( collectionPath: "relatos");
FirebaseAuth auth = FirebaseAuth.getInstance();
FirebaseUser currentUser = auth.getCurrentUser();

if (currentUser != null) {
    Query query1 = relatosRef.whereGreaterThanOrEqualTo( field: "genero_1", messages).whereEqualTo( field: "estado",  value: true);
    Query query2 = relatosRef.whereGreaterThanOrEqualTo( field: "genero_2", messages).whereEqualTo( field: "estado",  value: true);
    Query query3 = relatosRef.whereGreaterThanOrEqualTo( field: "genero_3", messages).whereEqualTo( field: "estado",  value: true);

    Task<QuerySnapshot> query1Task = query1.get();
    Task<QuerySnapshot> query2Task = query2.get();
    Task<QuerySnapshot> query3Task = query3.get();

    Tasks.whenAllComplete(query1Task, query2Task, query3Task).addOnCompleteListener(new OnCompleteListener<List<Task<?>>>() {
        @Override
        public void onComplete(@NonNull Task<List<Task<?>>> task) {
            for (Task<?> q : task.getResult()) {
                if (!q.isSuccessful()) {
                    Log.w(TAG,  msg: "Error getting documents.", q.getException());
                    return;
                }
                QuerySnapshot querySnapshot = (QuerySnapshot) q.getResult();
                for (DocumentSnapshot document : querySnapshot.getDocuments()) {
                    // obtén los campos del documento y haz lo que necesites con ellos
                    String titulo = document.getString( field: "titulo");
                    DocumentReference autorRef = document.getDocumentReference( field: "autor");
                    String descripcion = document.getString( field: "contenido");
                    String genero = document.getString( field: "genero");
                    double puntuacion = document.getDouble( field: "puntuacion");
                    Drawable drawable = getResources().getDrawable(R.drawable.imagess);
```

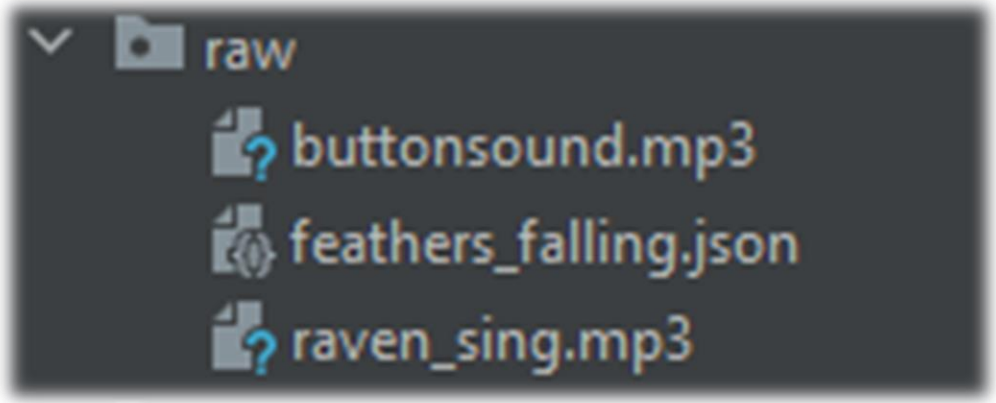
```
                    // crea un objeto "relato" con los campos del documento y agrega a la lista de relatos del autor
                    relato relato = new relato(titulo, autorRef.getPath(), descripcion, genero, puntuacion, drawable);
                    relatos.add(relato);
                }
            }

            // crea un adaptador y configura el RecyclerView
            adapter = new listaRelatosAdapter(relatos, getContext());
            recyclerView.setAdapter(adapter);
        }
    });
}

return view;
}
```

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
db.collection( collectionPath: "autores") CollectionReference
        .whereGreaterThanOrEqual( field: "name", messages) Query
        .addSnapshotListener(new EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot querySnapshot, @Nullable FirebaseFirestoreException e) {
                if (e != null) {
                    Log.w(TAG,  msg: "Listen failed.", e);
                    return;
                }

                if (querySnapshot != null && !querySnapshot.isEmpty()) {
                    // Si se encontró el documento del autor, obtener su referencia
                    DocumentSnapshot document = querySnapshot.getDocuments().get(0);
                    autorRef = document.getReference();
                    System.out.println(autorRef.getPath());

                    recyclerView.setHasFixedSize(true);
                    // Cargar recyclerView con datos
                    linearLayoutManager = new LinearLayoutManager(getContext(), LinearLayoutManager.VERTICAL,  reverseLayout: false);
                    recyclerView.setLayoutManager(linearLayoutManager);
                    recyclerView.setAdapter(adapter);

                    ArrayList<relato> relatos = new ArrayList<>();
                    CollectionReference relatosRef = db.collection( collectionPath: "relatos");
                    FirebaseAuth auth = FirebaseAuth.getInstance();
```

```
uery.get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()) {
            // la consulta se ejecutó correctamente, recorre los documentos resultantes
            Drawable drawable = getResources().getDrawable(R.drawable.imagess);
            System.out.println("ESTOY AQUI 1");
            if (task.getResult().isEmpty()) {
                System.out.println("Hay " + task.getResult().size() + " documentos que cumplen con el filtro.");
            }
            for (DocumentSnapshot document : task.getResult().getDocuments()) {
                System.out.println("ESTOY AQUI 2");
                // obtén los campos del documento y haz lo que necesites con ellos
                String titulo = document.getString( field: "titulo");
                DocumentReference autorRef = document.getDocumentReference( field: "autor");
                String descripcion = document.getString( field: "contenido");
                String genero = document.getString( field: "genero");
                double puntuacion = document.getDouble( field: "puntuacion");
                System.out.println(titulo + "," + genero);

                // crea un objeto "relato" con los campos del documento y agrega a la lista de relatos del autor
                relato relato = new relato(titulo, autorRef.getPath(), descripcion, genero, puntuacion, drawable);
                relatos.add(relato);
                System.out.println(relatos.size());
                System.out.println(relatos.get(0).getAutor());
                adapter = new listaRelatosAdapter(relatos, getContext());
                recyclerView.setAdapter(adapter);

            }
```

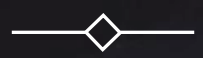
Como detalle…

Existen sonidos implementados en la aplicación, tanto al desplazarse por ella como al cargar la aplicación



¿CUALES SON LOS SIGUIENTES OBJETIVOS?

- Implementar la subida de archivos PDF directa a la base de datos
- Mejorar los filtros del inicio
- Crear un nuevo menú llamado “nido” donde la gente pueda crear eventos conectándose a una API
- Poder guardar relatos como favoritos
- Poder ver los comentarios ajenos sobre un relato
- Poder seguir a gente para que se te notifique cuando sube un relato
- Poder crear un chat entre usuarios



FIN