Documentación

Proyecto con Firebase

Iván Gallego

Índice de contenidos

Login

Variables usadas en el funcionamiento de MainActivity

```
private EditText usuario, pass;
private Button crear, entrar;
private FirebaseAuth auth;
private FirebaseAuth.AuthStateListener authListener;
```

Hay que tener en cuenta que hay que hacer *Override* de los métodos `onStart` y `onStop` indicándole que deben detener la instancia de autenticación

```
@Override
protected void onStart() {
    super.onStart();
```

```
    auth.addAuthStateListener(authListener);
}

@Override
protected void onStop() {
    super.onStop();
    if (auth != null) {
        auth.removeAuthStateListener(authListener);
        auth.signOut();
    }
}
```

Lo primero que debemos de hacer para poder hacer *Login* es obtener una instancia de `FirebaseAuth` con el método estático `FirebaseAuth.getInstance`, y en este caso se le añade un `Listener` para que haga algo cuando cambie el estado de login, es decir, detecte que se ha logeado un usuario

```
auth = FirebaseAuth.getInstance();
authListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser user = firebaseAuth.getCurrentUser();
        if (user != null) {
            Toast.makeText(MainActivity.this, user.getEmail() + " logeado", Toast.LENGTH_LONG).show();
            auth = firebaseAuth;
        }
    }
};
```

A continuación definimos lo que harán los botones para crear y logearse

```
crear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        auth.createUserWithEmailAndPassword(usuario.getText().toString(), pass.getText().toString())
                .addOnCompleteListener(MainActivity.this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(MainActivity.this, "Usuario creado", Toast.LENGTH_LONG).show();
                            iniciarAplicacion(task.getResult().getUser().getEmail().split("@")[0]);
                        } else {
                            Toast.makeText(MainActivity.this,
                                "Problemas al crear el usuario", Toast.LENGTH_LONG).show();
                        }
                    }
                });
    }
});
```

El botón de crear, creará un nuevo usuario a través del método `createUserWithEmailAndPassword` y pasándole los textos de los `EditText`. Trás esto, se le añade un *Listener* que confirmará que la acción de creación se ha completado y finalmente iniciará la aplicación, en caso contrario se mostrará un mensaje de error

El botón de logearse hace algo parecido, pero en vez de crear un usuario, se le consulta a la base de datos por el usuario proporcionado, en esta ocasión se utiliza el método `signInWithEmailAndPassword`

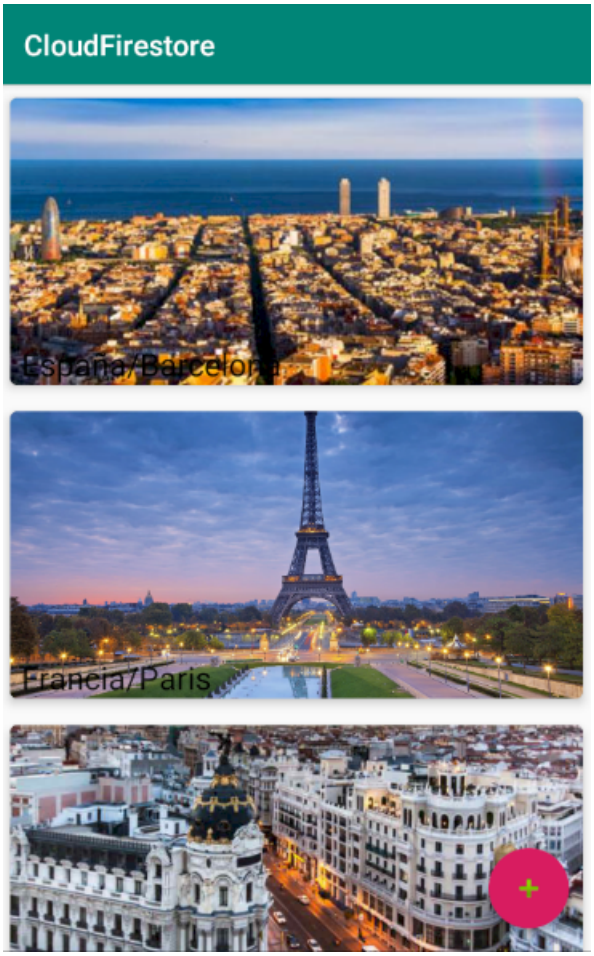
```
entrar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        auth.signInWithEmailAndPassword(usuario.getText().toString(), pass.getText().toString())
```

```
                .addOnCompleteListener(MainActivity.this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            iniciarAplicacion(task.getResult().getUser().getEmail().split("@")[0]);
                        } else {
                            Toast.makeText(MainActivity.this,
                                "Error de autenticación", Toast.LENGTH_LONG).show();
                        }
                    }
                });
    }
});
```

El método para inciar aplicación lo que hace es iniciar un activity distinto

```
private void iniciarAplicacion(String user) {
    Intent intent = new Intent(this, MainApplication.class);
    intent.putExtra("usuario", user);
    startActivity(intent);
}
```

MainApplication



El layout de esta activity es un recycler y un FAB

Estas son las variables para esta activity

```
private FirebaseFirestore db;
private FirebaseStorage storage;
```

```
private RecyclerView recycler;
private Adapter adapter;
private FloatingActionButton fab;
private int COD_EDITAR = 2;
private int COD_ADD = 3;
```

Importante, al igual que en el activity de login, hay que hacer *Override* de los métodos `onStart` y `onStop`, en circunstancias normales, habría que llamar en `onStart` al método `startListening` pero no es así por que la carga de recycler (en mi caso) es asíncrona, por lo tanto el método `onStart` se ejecutará antes de que el adaptador haya sido creado. Es por esto que yo llamo al método `onStart` tras la creación del adaptador

De todas maneras, hay que colocalo por si se ha detenido la aplicación, por ejemplo al ir a otra activity, ya que de esa manera no pasaría por el método `onCreate` otra vez, pero si por el `onStart`

```
@Override
protected void onStart() {
    super.onStart();
    if (adapter != null) {
        adapter.startListening();
    }
}

@Override
protected void onStop() {
    super.onStop();
    adapter.stopListening();
}
```

Al iniciar la aplication, en el método `onCreate` se inicializan los elementos de layout y se le asigna un *Listener* al FAB para que inicia la activity para añadir más datos, además se llama al método `cargarDatos` que es el que hace una *query* para obtener todos los datos de la base de datos y construir el recycler a partir de esos datos

```
private void cargarRecycler(Query q) {
FirestoreRecyclerOptions<Ciudad> firestoreRecyclerOptions = new FirestoreRecyclerOptions.Builder<Ciudad>()
        .setQuery(q, Ciudad.class).build();
adapter = new Adapter(firestoreRecyclerOptions);
adapter.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        editarCiudad(recycler.getChildAdapterPosition(v));
    }
});
adapter.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        eliminarCiudad(recycler.getChildAdapterPosition(v));
        return false;
    }
});
adapter.startListening();
recycler.setAdapter(adapter);
recycler.setLayoutManager(new LinearLayoutManager(this));
}
```

Al adaptador también se le asigna un escuchador para poder editar una ciudad cuando se haga click sobre una vista del recycler y para poder eliminarla si se hace un click largo

```
private void cargarDatos() {
CollectionReference ref = db.collection("ciudades");
ref.get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
```

```
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()) {
            cargarRecycler(task.getResult().getQuery());
        } else {
            Toast.makeText(MainApplication.this, "Fallo al leer datos", Toast.LENGTH_LONG).show();
        }
    }
});
}
```

Para eliminar una ciudad simplemente pregunto al adaptador por el ID del elemento que corresponde a la vista que se ha pulsado gracias a la funcionalidad del adaptador por extender de `FirestoreRecyclerAdapter`

```
// Index i
adapter.getSnapshots().getSnapshot(i).getId()
```

Tras eso simplemente le digo a la base de datos que elimine el documento que corresponda con dicho ID

```
private void eliminarCiudad(int i) {
    String key = adapter.getSnapshots().getSnapshot(i).getId();
    db.collection("ciudades").document(key).delete().addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()) {
                Toast.makeText(MainApplication.this, "Se ha elimnado el registro", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(MainApplication.this, "Fallo al eliminar", Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Para editar una ciudad, inicio una activity que al final me devolverá la ciudad editada y el ID para poder localizar el objeto a editar

```
private void editarCiudad(int item) {
    Intent i = new Intent(this, EditCiudad.class);
    Ciudad c = adapter.getItem(item);
    i.putExtra("ciudad", c);
    i.putExtra("key", adapter.getSnapshots().getSnapshot(item).getId());
    startActivityForResult(i, COD_EDITAR);
}
```

El método que recibe los datos de otras activities

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == COD_EDITAR && resultCode == RESULT_OK) {
        Ciudad c = (Ciudad) data.getExtras().get("ciudad");
        String key = (String) data.getExtras().get("key");
        uploadImage(c, key);
    } else if (requestCode == COD_ADD & resultCode == RESULT_OK) {
        Ciudad c = (Ciudad) data.getExtras().get("ciudad");
        uploadImage(c, getKey());
    } else if (resultCode == RESULT_CANCELED) {
        Toast.makeText(this, "Se ha cancelado la operación", Toast.LENGTH_SHORT).show();
    }
}
```

```
private String getKey(){
    Random r = new Random();
    StringBuilder builder = new StringBuilder();
    for (int i = 0; i < 10; i++) {
        builder.append(r.nextInt(96) + 32);
    }
    return builder.toString();
}
```

Cundo recibo un dato de otra activity, llamo al método `uploadImage` que subirá la imagen al `Storage` de la base de datos, y después llama al método `actualizarRegistro` que editará o añadira un registro. Si es una edición entonces se sustituirá el registro asociado al ID por el nuevo objeto que se ha recibido, si se trata de una adición, entonces se genera un nuevo ID con el método `getKey` y se añadira a la base de datos con el ID nuevo generado

```
private void uploadImage(final Ciudad c, final String key) {
    if (c.getImagen() == null || c.getImagen().equals("")) {
        actualizarRegistro(c, key, "");
    } else {
        Uri uri = Uri.parse(c.getImagen());
        String fileName = getFileNameFromUri(uri);
        final String downloadURL = "images/" + fileName;
        StorageReference refSubida = storage.getReference().child(downloadURL);
        refSubida.putFile(uri).addOnCompleteListener(new OnCompleteListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<UploadTask.TaskSnapshot> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(MainApplication.this,
                        "Se ha subido la imagen", Toast.LENGTH_SHORT).show();
                    actualizarRegistro(c, key, downloadURL);
                } else {
                    Toast.makeText(MainApplication.this,
                        "No se ha podido subir la imagen", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

```
private void actualizarRegistro(Ciudad c, String key, String downloadURL) {
    HashMap<String, Object> hashMap = new HashMap<>();
    hashMap.put("ciudad", c.getCiudad());
    hashMap.put("pais", c.getPais());
    hashMap.put("imagen", downloadURL);
    db.collection("ciudades").document(key).set(hashMap);
}
```

> La cadena que tiene el campo imagen es la ruta de un archivo local, hasta que se sube la imagen y se obtiene la URL de descarga para cargarla en el adaptador/holder

Intents

Los intents para editar y añadir registros, son muy parcecidos

En el intent de editar simplemente se obtienen los datos del objeto y se colocan en sus correspondientes elementos del layout, además, si la imagen no es nula o no es cadena vacía, se carga desde la base de datos; en cambio, si lo es, se carga una imagen por defecto

```
public class EditCiudad extends AppCompatActivity {

    private final String DEFAULT_IMAGE = "/default.jpg";

    private Ciudad c;
    private EditText editCiudad, editPais;
    private ImageView imagen;
    private Button aceptar;
    private String key;
    private int COD_ELEGIR_IMAGEN = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_ciudad);
        key = (String) getIntent().getExtras().get("key");
        c = getIntent().getParcelableExtra("ciudad");

        editCiudad = findViewById(R.id.editCiudad);
        editPais = findViewById(R.id.editPais);
        imagen = findViewById(R.id.imageView);
        aceptar = findViewById(R.id.buttonAceptar);

        if (c.getImagen() == null || c.getImagen().equals("")) {
            StorageReference reference = FirebaseStorage.getInstance().getReference(DEFAULT_IMAGE);
            reference.getDownloadUrl().addOnCompleteListener(new OnCompleteListener<Uri>() {
                @Override
                public void onComplete(@NonNull Task<Uri> task) {
                    Picasso.get().load(task.getResult()).into(imagen);
                }
            });
        } else {
            StorageReference reference = FirebaseStorage.getInstance().getReference(c.getImagen());
            reference.getDownloadUrl().addOnCompleteListener(new OnCompleteListener<Uri>() {
                @Override
                public void onComplete(@NonNull Task<Uri> task) {
                    Picasso.get().load(task.getResult()).into(imagen);
                }
            });
        }
        editPais.setText(c.getPais());
        editCiudad.setText(c.getCiudad());

        imagen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
                intent.setType("image/*");
                if (intent.resolveActivity(EditCiudad.this.getPackageManager()) != null) {
                    startActivityForResult(intent, COD_ELEGIR_IMAGEN);
                }
            }
        });

        aceptar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                c.setCiudad(editCiudad.getText().toString());
                c.setPais(editPais.getText().toString());
                Intent i = new Intent();
                i.putExtra("ciudad", c);
                i.putExtra("key", key);
                setResult(RESULT_OK, i);
                finish();
            }
        });
```

```
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == COD_ELEGIR_IMAGEN && resultCode == RESULT_OK) {
            Uri rutaImagen = data.getData();
            c.setImagen(rutaImagen.toString());
            imagen.setImageURI(rutaImagen);

        } else if (resultCode == RESULT_CANCELED) {
            Toast.makeText(this, "Se ha cancelado la operación", Toast.LENGTH_SHORT).show();
        }
    }
}
```

El de añadir es más de lo mismo solo que no se comprueba que exista una imagen ya que nunca hay una imagen al tratarse de un objeto nuevo y se carga una imagen por defecto

```
public class AddCiudad extends AppCompatActivity {
    Ciudad c = new Ciudad();
    private EditText editCiudad, editPais;
    private ImageView imagen;
    private Button aceptar;
    private int COD_ELEGIR_IMAGEN = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_ciudad);

        editCiudad = findViewById(R.id.editCiudad);
        editPais = findViewById(R.id.editPais);
        imagen = findViewById(R.id.imageView);
        aceptar = findViewById(R.id.buttonAceptar);

        imagen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
                intent.setType("image/*");
                if (intent.resolveActivity(AddCiudad.this.getPackageManager()) != null) {
                    startActivityForResult(intent, COD_ELEGIR_IMAGEN);
                }
            }
        });

        aceptar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                c.setCiudad(editCiudad.getText().toString());
                c.setPais(editPais.getText().toString());
                Intent i = new Intent();
                i.putExtra("ciudad", c);
                setResult(RESULT_OK, i);
                finish();
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == COD_ELEGIR_IMAGEN && resultCode == RESULT_OK) {
```

```
            Uri rutaImagen = data.getData();
            c.setImagen(rutaImagen.toString());
            imagen.setImageURI(rutaImagen);
        } else if (resultCode == RESULT_CANCELED) {
            Toast.makeText(this, "Se ha cancelado la operación", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Adaptador

El adaptador no es más que un adaptador genérico que extiende de la clase `FirestoreRecyclerAdapter`

```
public class Adapter extends FirestoreRecyclerAdapter<Ciudad, Holder>
        implements View.OnClickListener, View.OnLongClickListener {
    private View.OnClickListener clickListener;
    private View.OnLongClickListener longClickListener;

    Adapter(@NonNull FirestoreRecyclerOptions<Ciudad> options) {
        super(options);
    }

    @Override
    protected void onBindViewHolder(@NonNull Holder holder, int position, @NonNull Ciudad model) {
        holder.bind(model);
    }

    @NonNull
    @Override
    public Holder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.holder, viewGroup, false);
        view.setOnClickListener(this);
        view.setOnLongClickListener(this);
```

```
        return new Holder(view);
    }

    void setOnClickListener(View.OnClickListener listener) {
        this.clickListener = listener;
    }

    void setOnLongClickListener(View.OnLongClickListener listener) {
        this.longClickListener = listener;
    }

    @Override
    public void onClick(View v) {
        if (clickListener != null) clickListener.onClick(v);
    }

    @Override
    public boolean onLongClick(View v) {
        if (longClickListener != null) {
            longClickListener.onLongClick(v);
        }
        return true;
    }
```

Holder

Lo único especial que tiene este holder es la carga de la imagen en la creación a través de `FirebaseStorage` y utilzando la librería de *Picasso* que permite cargar una imagen en un `ImageView` con solo una `URL`

```
public class Holder extends RecyclerView.ViewHolder {

    private final String DEFAULT_IMAGE = "/default.jpg";

    private TextView text;
    private ImageView imageView;

    public Holder(View v) {
        super(v);
        text = v.findViewById(R.id.text);
        imageView = v.findViewById(R.id.imageView);
    }

    public void bind(Ciudad item) {
        text.setText(String.format("%s/%s", item.getPais(), item.getCiudad()));
        if (item.getImagen() == null || item.getImagen().equals("")) {
            StorageReference reference = FirebaseStorage.getInstance().getReference(DEFAULT_IMAGE);
            reference.getDownloadUrl().addOnCompleteListener(new OnCompleteListener<Uri>() {
                @Override
                public void onComplete(@NonNull Task<Uri> task) {
                    Picasso.get().load(task.getResult()).into(imageView);
                }
            });
        } else {
            StorageReference reference = FirebaseStorage.getInstance().getReference(item.getImagen());
            reference.getDownloadUrl().addOnCompleteListener(new OnCompleteListener<Uri>() {
                @Override
                public void onComplete(@NonNull Task<Uri> task) {
                    Picasso.get().load(task.getResult()).into(imageView);
                }
            });
        }
    }
}
```