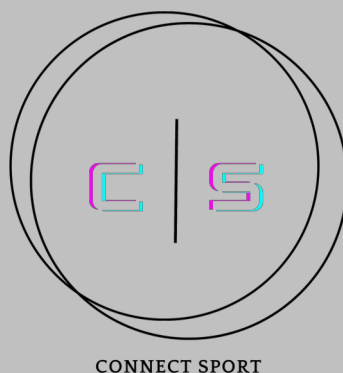


**PROYECTO**  
**CONNECT SPORT**



**CICLO FORMATIVO DE GRADO SUPERIOR:**

**Desarrollo de aplicaciones multiplataforma**

**AUTORES:**

**Jacobo Sánchez Rodríguez**

## **Licencia**

**Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.**

## RESUMEN

La empresa Connect Sport S.L. compuesta por Jacobo Sánchez Rodríguez con sede en el centro de Madrid.

Se dedica al desarrollo de software de creación de eventos deportivos .

Cubre la necesidad de los entusiastas del deporte estar conectado con otras personas de su mismo gustos deportivos. La aplicación permite a los usuarios crear y acceder a eventos mejorando su estado físico, como partidos de fútbol, baloncesto, pádel, etc., y encontrar a otros usuarios interesados en participar.

El público objetivo de Connect Sport son las personas que practican deporte de forma regular y principiantes en el mundo del deporte. La aplicación está dirigida a personas de todas las edades sin determinar su nivel de estado físico.

Esta aplicación móvil permite a los usuarios unirse y crear eventos deportivos y encontrar a otros usuarios con sus mismos intereses.

En la sección de creación de eventos pueden planificar actividades de forma sencilla y rápida guardándose los datos del creador para que previamente los demás usuarios puedan calificar el evento en base a sus gustos.

En la sección de búsqueda y filtros de eventos pueden encontrar actividades de su interés por ubicación, fecha y deporte o bien por palabras claves que faciliten encontrar lo que más le convenga.

También disponen de un mapa donde aparecen todas los eventos que hay creados mediante chinchetas, así facilitando poder encontrar actividades mas cercanas a su ubicación.

Connect Sport es una solución innovadora que satisface una necesidad importante en el mercado. La aplicación tiene el potencial de conocer a personas de todas las edades y niveles de habilidad para practicar deporte de forma organizada y divertida.

**“ConnectSport” connect with life**

## ABSTRACT

The company Connect Sport S.L. composed by Jacobo Sánchez Rodríguez based in the center of Madrid.

It is dedicated to the development of software for creating sports events.

It meets the need of sports enthusiasts to be connected with other people with the same sports interests. The application allows users to create and access events that improve their physical condition, such as football, basketball, paddle tennis matches, etc., and find other users interested in participating.

The target audience of Connect Sport is people who practice sports regularly and beginners in the world of sports. The app is aimed at people of all ages without determining their fitness level.

This mobile application allows users to join and create sporting events and find other users with the same interests.

In the event creation section you can plan activities easily and quickly, saving the creator's data so that other users can previously rate the event based on their tastes.

In the search section and event filters you can find activities of interest by location, date and sport or by keywords that make it easy to find what suits you best.

They also have a map where all the events created using pins appear, making it easier to find activities closest to your location.

Connect Sport is an innovative solution that meets an important need in the market. The application has the potential to meet people of all ages and skill levels to practice sports in an organized and fun way.

**“ConnectSport” connect with life**

## Índice de contenido

|  |           |
|--|-----------|
| <b>1. INTRODUCCIÓN.....</b>                                  | <b>8</b>  |
| <b>2. CONTEXTO FUNCIONAL Y TECNOLÓGICO .....</b>             | <b>9</b>  |
| <b>2.1 CONTEXTO FUNCIONAL.....</b>                           | <b>9</b>  |
| <b>2.2 CONTEXTO TECNOLÓGICO .....</b>                        | <b>11</b> |
| <b>3. PLANIFICACIÓN DEL PROYECTO .....</b>                   | <b>13</b> |
| <b>4. DESCRIPCIÓN DE LA SOLUCIÓN: ANÁLISIS Y DISEÑO.....</b> | <b>15</b> |
| <b>4.1 Especificación de requisitos.....</b>                 | <b>15</b> |
| <b>4.2 Selección de plataforma tecnológica .....</b>         | <b>15</b> |
| <b>4.3 Descripción del diseño de la solución .....</b>       | <b>15</b> |
| <b>5. DESCRIPCIÓN DE LA SOLUCIÓN: CONSTRUCCIÓN.....</b>      | <b>19</b> |
| <b>5.1 Documentación descriptiva.....</b>                    | <b>19</b> |
| <b>5.2 Problemas encontrados y soluciones .....</b>          | <b>39</b> |
| <b>6. FUENTES.....</b>                                       | <b>42</b> |
| <b>7. ANEXOS.....</b>  | <b>42</b> |

## Índice de figuras

|   |    |
|---|----|
| 1. API Google Maps Platform .....               | 12 |
| 2. Lista de tareas ConnectSport .....           | 13 |
| 3. Diagrama Gantt desarrollo ConnectSport ..... | 14 |
| 4. Paleta de colores ConnectSport .....         | 16 |
| 5. Diagrama de Colecciones .....                | 17 |
| 6. Diagrama de Casos de Uso .....               | 18 |
| 7. Código Splash Activity .....                 | 19 |
| 8. Animación letras in Splash Activity .....    | 20 |
| 9. Secuencia vista Splash Activity .....        | 20 |
| 10. Firebase in Splash Activity .....           | 21 |
| 11. VistaTérminos y condiciones .....           | 22 |
| 12. Código de permiso ubicación .....           | 22 |
| 13. Control ubicación AndroidManifest.xml ..... | 23 |
| 14. Código SharedPreferences .....              | 23 |
| 15. Vista MainActivity .....                    | 24 |
| 16. Código MainActivity .....                   | 25 |
| 17. Código Firebase Storage .....               | 25 |
| 18. Vista Fragment SingUp .....                 | 26 |
| 19. Método requisitos de contraseña .....       | 26 |
| 20. Vista Login .....                           | 27 |
| 21. Código reseteo de contraseña .....          | 27 |
| 22. Menú MainBn .....                           | 28 |

|  |    |
|--|----|
| 23. Código FragmentMain1 .....   | 28 |
| 24. Vista NewEvents .....  | 29 |
| 25. Código openGallery() .....   | 30 |
| 26. Código descarga de imágenes .....                                  | 30 |
| 27. Vista ítem evento .....  | 31 |
| 28. Código evento detallado .....                                      | 31 |
| 29. Código filtrado por participantes .....                            | 32 |
| 30. Código filtrado por tipo de deporte .....                          | 32 |
| 31. Permisos en AndroidManifest .....                                  | 33 |
| 32. Código recorrer base de datos .....                                | 33 |
| 33. Código tipo de mapa .....  | 34 |
| 34. Vista del mapa .....   | 34 |
| 35. Código StorageUtil .....   | 35 |
| 36. Código calcular IMC .....  | 35 |
| 37. Código editar foto de perfil .....                                 | 36 |
| 38. Archivo strings proyecto .....                                     | 36 |
| 39. Opción de AndroidStudio para internacionalizar la aplicación ..... | 37 |
| 40. Selector de idiomas para la aplicación .....                       | 37 |
| 41. Traducción de strings en los idiomas seleccionados .....           | 37 |
| 42. Archivo themes proyecto .....                                      | 38 |
| 43. Configuración Dark Theme .....                                     | 38 |
| 44. Permiso de ubicación en términos y condiciones .....               | 39 |
| 45. Submenú en nuevo archivo XML .....                                 | 40 |
| 46. Solución método CalcularImc() .....                                | 41 |

# 1 INTRODUCCIÓN

Este documento responde a la realización del **módulo de Proyecto** del CFGS en Administración de Sistemas Informáticos en Red. El módulo de Proyecto complementa, la formación establecida para el resto de los módulos profesionales que integran el título en las funciones de análisis del contexto, diseño del proyecto y organización de la ejecución.

El principal objetivo es ayudar a los organizadores a promocionar sus eventos y llegar a un público más amplio, ahorrando tiempo automatizando muchas tareas que tradicionalmente se realizarían manualmente, permite crear, planificar, promover y gestionar eventos deportivos de todo tipo.

Las características que reúne este proyecto son las siguientes:

- Creación de eventos.
- Borrar eventos.
- Unirse a eventos.
- Media de calificación.
- Listado de todos los eventos.
- Búsqueda por palabra claves y filtros.
- Mapa.
- Perfil editable.
- Calculadora de imc.
- Varias opciones de registro e inicio de sesión.
- Restablecer contraseña.



## 2.CONTEXTO FUNCIONAL Y TECNOLÓGICO

A continuación se describen los contextos funcionales y tecnológicos en los que se enmarcan el proyecto.

### 2.1.CONTEXTO FUNCIONAL

Este proyecto se centra en la creación de eventos deportivos en los que todos los usuarios puedan acceder a ellos.

Una aplicación de creación de eventos deportivos es una herramienta que permite a los organizadores crear, planificar, promocionar y gestionar eventos deportivos. Los usuarios de la aplicación son los organizadores, los participantes y los espectadores.

Los organizadores deben poder crear eventos deportivos con facilidad. La aplicación debe permitir a los organizadores definir los detalles del evento, como el nombre, la fecha, el lugar, el deporte, las categorías y los participantes.

Los participantes son los usuarios secundarios de la aplicación. Utilizan la aplicación para inscribirse en eventos deportivos y obtener información sobre los eventos.

Los participantes deben poder inscribirse en eventos deportivos de forma sencilla. La aplicación debe proporcionar a los participantes un proceso de inscripción sencillo y seguro.

Los participantes deben poder obtener información sobre los eventos deportivos. La aplicación debe proporcionar a los participantes información sobre los eventos, como el calendario, las reglas, las inscripciones y los resultados.

A parte, los participantes tendrán la oportunidad de encontrar eventos con mas facilidad buscando a sus intereses en el buscador o directamente en el mapa en el cual aparecen los eventos a través de chinchetas.

Si el usuario hace uso de esta aplicación podrá realizar las siguientes acciones (casos de uso):

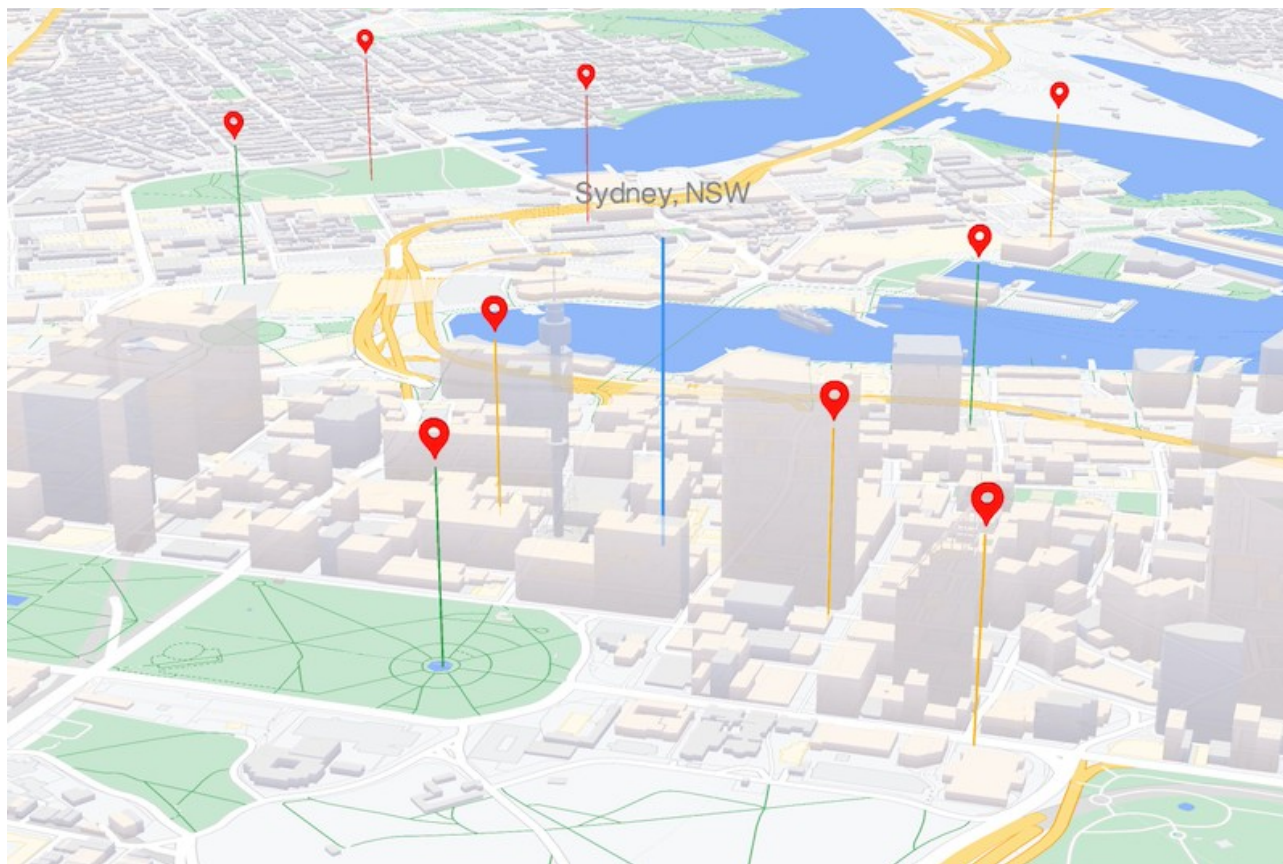
- Registrarse
- Iniciar sesión si ya se había registrado
- Cambiar la contraseña si no se recuerda
- Verificar el email cuando se registra
- Navegar entre las diferentes secciones principales de la aplicación: inicio, búsqueda, mapa y perfil.
- Navegar entre los diferentes apartados relacionados con el propio usuario y la sesión
  - editar perfil: cambiar imagen, actualizar datos(peso, altura y descripción)
  - cerrar sesión
  - calcular IMC
- Informarse sobre los eventos
- Filtrar eventos según su interés
- Consultar los detalles de cada evento
- Suscribirse al evento
- Consultar cantidad de participantes y valoración del evento

Se sugiere revisar el vídeo que demuestra la operatividad de la interfaz de la aplicación para obtener una comprensión más completa de sus funcionalidades. Anexo I.

## 2.2.CONTEXTO TECNOLÓGICO

ConnectSport, la aplicación, debe de incluir una amplia gama de funcionalidades para atender todas las necesidades de nuestros usuarios. Aquí hay una lista de características necesarias para cumplir con los diversos casos de uso de la aplicación:

- Conexión a internet mientras de usa la aplicación de forma permanente.
- Acceso a la ubicación a tiempo real del usuario para poder mostrar los eventos de su alrededor mas cercanos.
- Acceso a sus fotos del dispositivo para permitirle poder cambiar la foto de perfil o las imágenes de los eventos que cree.
- Creación y uso de base de datos [firebase](#) con las siguientes utilidades:
  - o Registrar o inicio de sesión de los usuarios y comprobaciones necesarias para el funcionamiento de la aplicación.
  - o Recoger los datos de cada usuario para el perfil.
  - o Almacenar las inscripciones de los usuarios de cada evento.
  - o Guardar todas las fotos de cada usuario.
  - o Guardar por cada usuario la valoración en caso de haber votado en los eventos.
  - o Guardar los datos de la creación de los eventos.
- Integración de la [API Google Maps Platform](#), lo que le permite incorporar mapas interactivos y funcionalidades de Googles Maps. Esta integración posibilita la visualización de eventos en el mapa, mostrando la ubicación en tiempo real del usuario y facilitando la identificación de los eventos más cercanos a su ubicación.



*API Google Maps Platform*

### 3.PLANIFICACIÓN DEL PROYECTO

Para la planificación del proyecto, las primeras tareas que se han tenido que realizar para el avance del desarrollo de la aplicación han sido, conectar el proyecto con la base de datos de firebase y crear el repositorio en GitHub para mantener un registro del historial de cambios del proyecto.

Posteriormente, se realizó la asignación de la paleta de colores correspondientes al tema y a diseñar el logotipo de ConnectSport, marcando así el estilo que tendrá la aplicación.

Una vez que la aplicación está conectada a la base de datos, almacenada en el repositorio de GitHub y habiendo asignado el estilo de la aplicación, se puede comenzar con el desarrollo de las funciones que tendrá la aplicación, mostradas a continuación en una tabla con el listado de las tareas con el tiempo empleado en cada una en implementarlas. De forma consecutiva se muestra el Diagrama de Gantt elaborado a partir de este listado.

| TAREA                      | INICIO | DURACIÓN | FIN    |
|----------------------------|--------|----------|--------|
| Repositorio GitHub         | 20-sep | 1        | 21-sep |
| Creación BBDD Firebase     | 20-sep | 7        | 10-oct |
| Estilo app                 | 27-sep | 1        | 28-sep |
| Términos y condiciones     | 28-sep | 2        | 30-sep |
| Splash                     | 29-oct | 1        | 30-sep |
| Registro                   | 02-oct | 4        | 06-oct |
| Inicio sesión              | 06-oct | 2        | 08-oct |
| Cerrar sesión              | 09-oct | 1        | 10-oct |
| Recuperación de contraseña | 09-oct | 1        | 10-oct |
| Vista perfil               | 09-oct | 3        | 12-oct |
| Vista inicio app           | 13-oct | 1        | 14-oct |
| Creación eventos           | 13-oct | 5        | 18-oct |
| Vista de eventos           | 16-oct | 6        | 20-oct |
| Vista ampliada de eventos  | 16-oct | 5        | 21-oct |
| Vista de búsqueda          | 20-oct | 4        | 24-oct |
| Filtro de búsqueda         | 23-oct | 8        | 31-oct |
| Vista mapa                 | 30-oct | 2        | 01-nov |
| Marcadores del mapa        | 02-nov | 5        | 07-nov |
| Tipo de mapa               | 03-nov | 1        | 04-nov |
| Ubicación a tiempo real    | 06-nov | 4        | 10-nov |
| Modo oscuro                | 09-nov | 1        | 10-nov |
| Cambio idioma              | 10-nov | 1        | 11-nov |
| Documentar proyecto        | 10-oct | 41       | 20-nov |

*Lista de tareas ConnectSport*

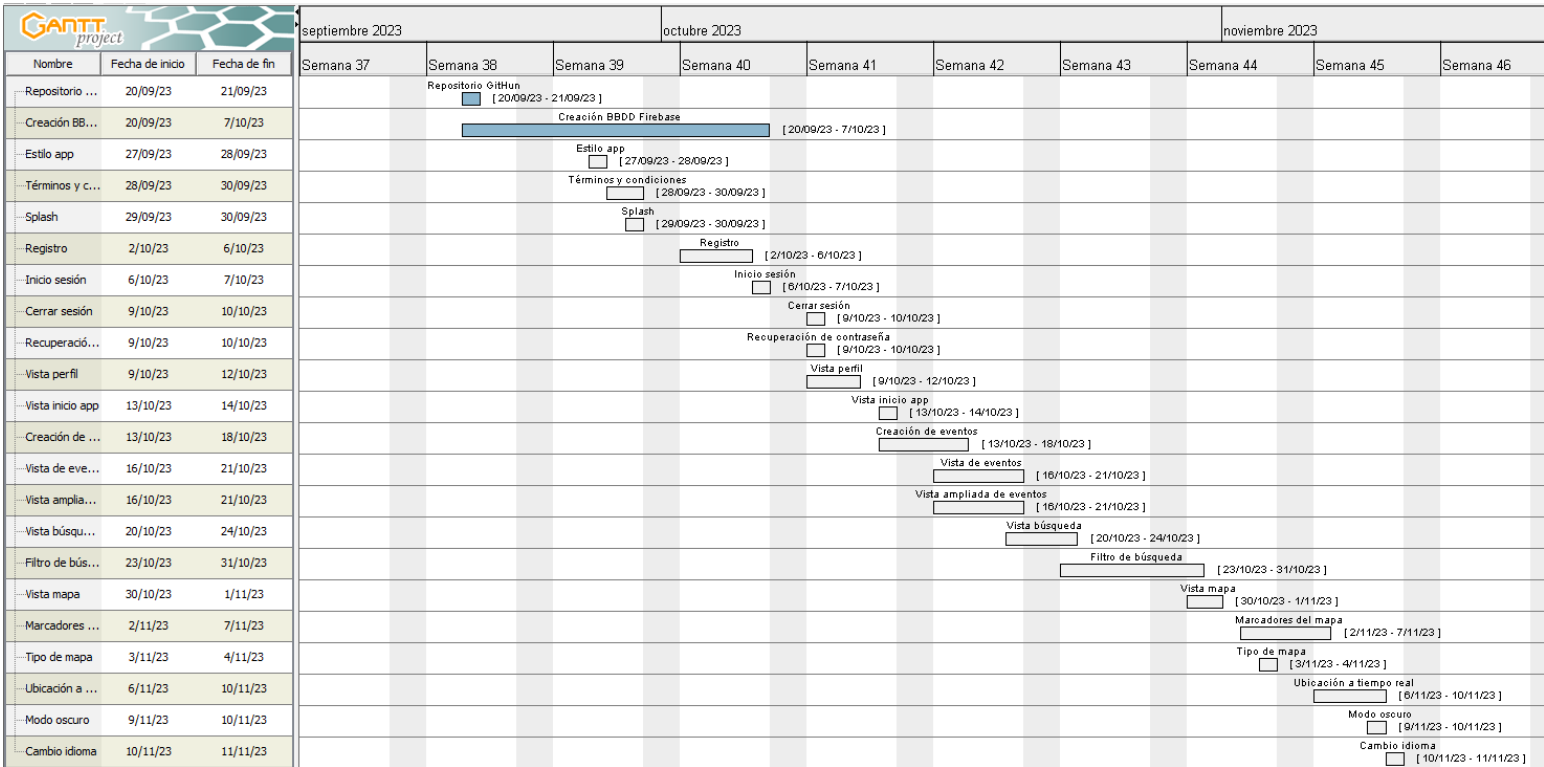


Diagrama Gantt desarrollo ConnectSport

## 4. DESCRIPCIÓN DE LA SOLUCIÓN: ANÁLISIS Y DISEÑO

### 4.1. Especificación de requisitos

Para poder llevar a cabo el desarrollo y realizar las pruebas necesarias de este proyecto, se requieren los siguientes elementos:

- Dispositivo móvil compatible con Android.
- Versión de API de Android 34 o superior.
- Conexión a internet.

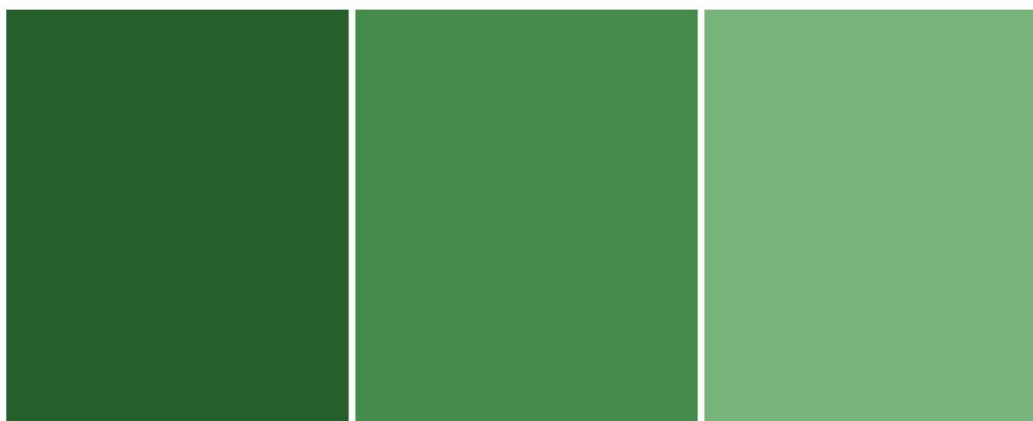
### 4.2. Selección de plataforma tecnológica

El sistema tecnológico y que respalda el cumplimiento de los requisitos previos es:

- IDE Android Studio.
- Dispositivos móviles para poder emular la aplicación.
- Firebase para comprobar la base de datos empleada.
- Repositorio GitHub.
- API Google Maps Platform.

### 4.3. Descripción del diseño de la solución

En la creación de esta aplicación móvil, se ha utilizado una paleta de colores que retransmite un estilo luminoso y vivaz acorde con la temática. Se ha experimentado con la mezcla de tonos secundarios, así con sus variaciones oscuros y claros.



#275F2C

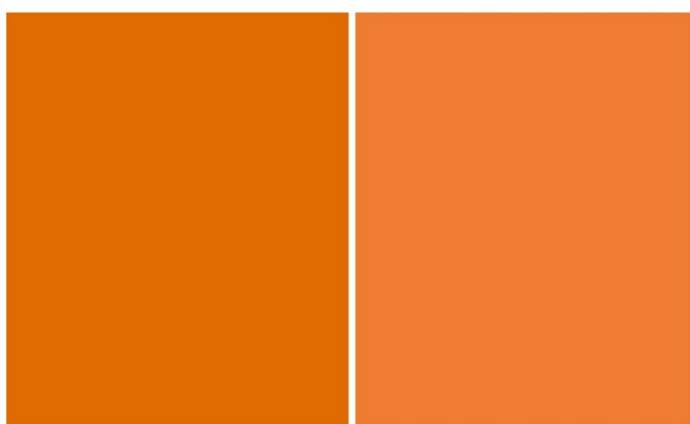
RGB 39, 95, 44

#468C4D

RGB 70, 140, 77

#79B47C

RGB 121, 180, 124



#E06B01

RGB 224, 107, 1

#F07C32

RGB 240, 124, 50

color.adobe.com

*Paleta de colores ConnectSport*



Para la concepción de la solución de este proyecto, fue esencial desarrollar un diagrama de colecciones que detallara las diversas tablas requeridas en la base de datos configurada en Firebase.

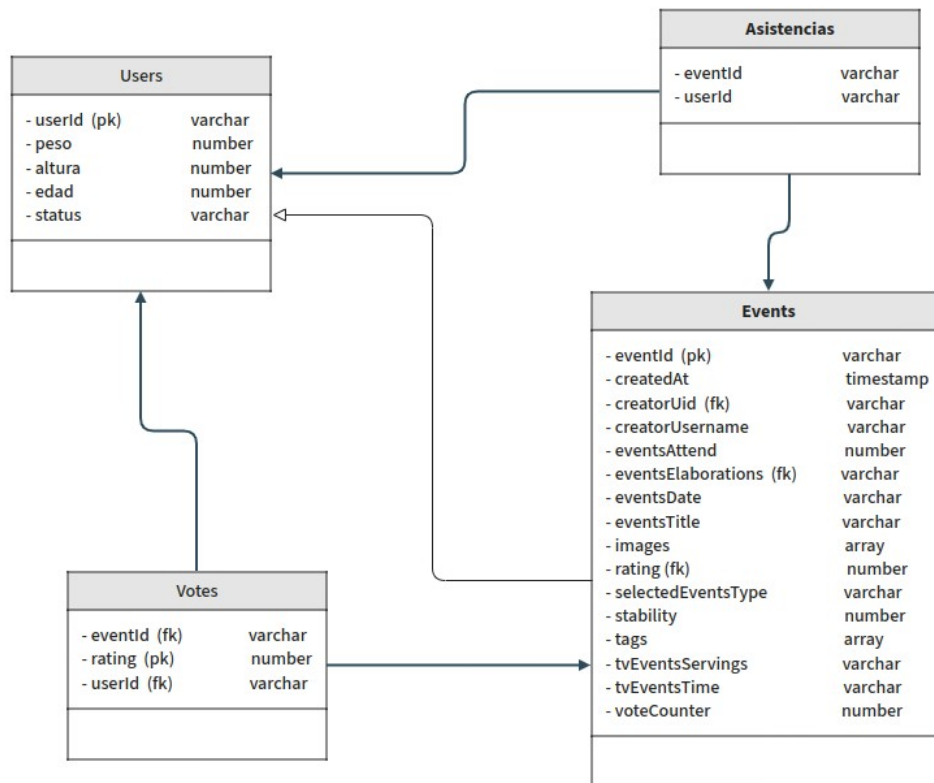


Diagrama de Colecciones

Para garantizar un diseño preciso de esta aplicación móvil, fue necesario definir los distintos estados del usuario, incluyendo “creador” y “participante”, así como sus diversas interacciones al utilizar la aplicación. Para representar visualmente este proceso, se creó un diagrama de casos de uso que ilustra todas estas situaciones.

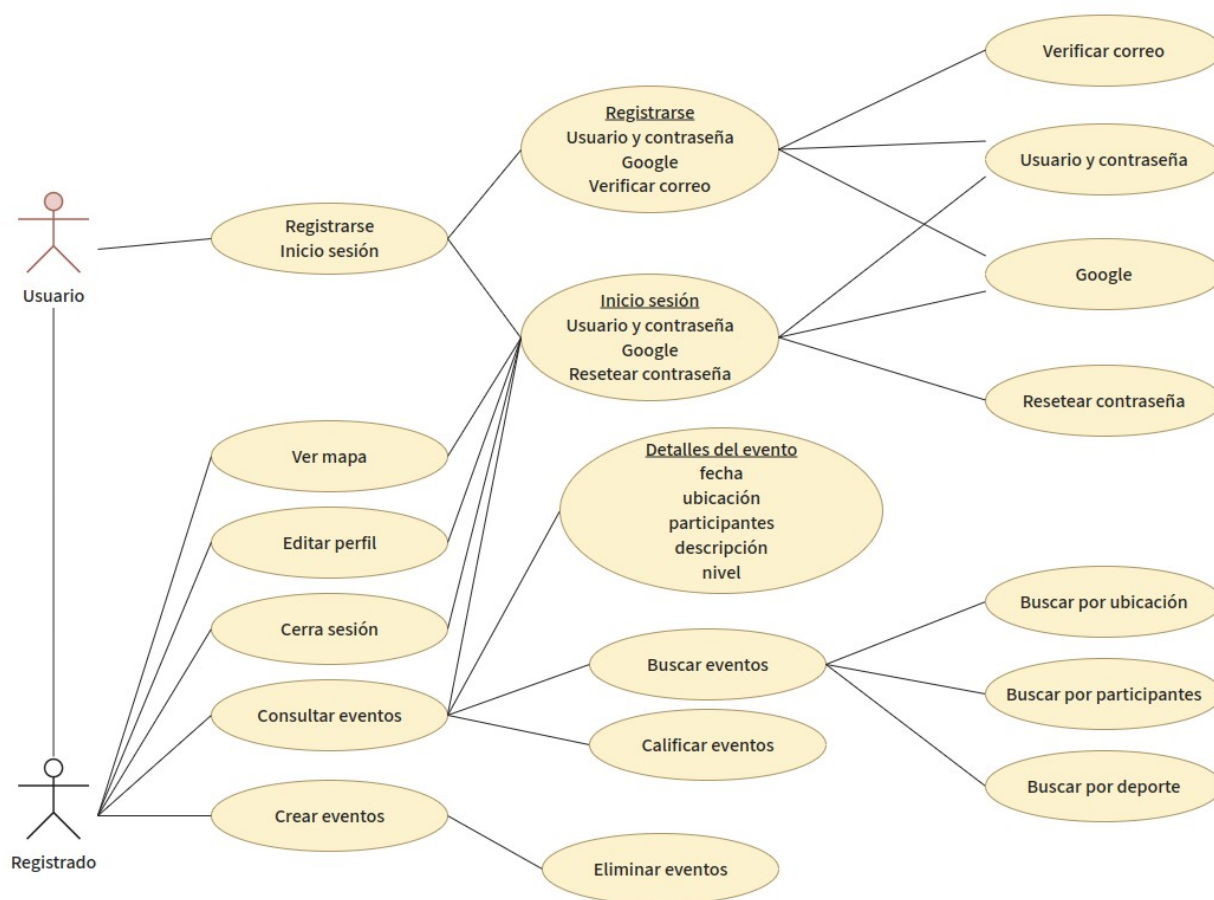


Diagrama de Casos de Uso

## 5. DESCRIPCIÓN DE LA SOLUCIÓN: CONSTRUCCIÓN

### 5.1 Documentación descriptiva

Se proporcionará una descripción detallada de cada vista y función de la aplicación, con capturas de pantalla del código implementado y su correspondiente resultado para cada actividad.

#### ➤ Splash Activity

Al iniciar la aplicación, se visualiza un fondo blanco con dos círculos intercalados entre ellos una barra central en la cual se van uniendo las dos iniciales del nombre de la aplicación Connect Sport. Tras la finalización de ambas animaciones y la visualización completa del isotipo, se procede de manera secuencial a la siguiente vista.

```
public class SplashActivity extends AppCompatActivity {  
    2 usages  
    boolean ToS;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_splash);  
  
        ToS = SharedPrefsUtil.getBoolean(context: this, key: "ToS");  
        openApp(ToS);  
  
        ImageView s = findViewById(R.id.s);  
        Animation myanim = AnimationUtils.loadAnimation(context: SplashActivity.this, R.anim.move_s);  
        s.startAnimation(myanim);  
        ImageView c = findViewById(R.id.c);  
        Animation myanim2 = AnimationUtils.loadAnimation(context: SplashActivity.this, R.anim.move_c);  
        c.startAnimation(myanim2);  
    }  
}
```

*Código Splash Activity*

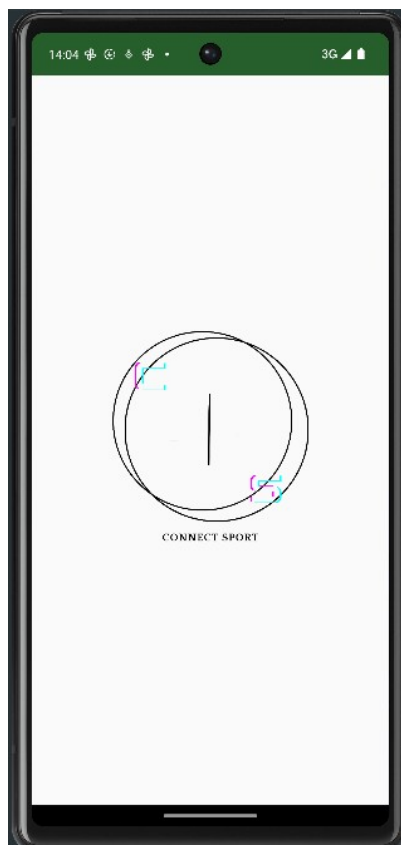
Como podemos observar, hay dos animaciones, una asociada cada letra del logotipo: una letra sale diagonalmente hasta unirse en el centro, formando así el logo de la aplicación

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate
    android:duration="3000"
    android:fromYDelta="-10%p"
    android:toYDelta="0%p"
    android:fromXDelta="-10%p"
    android:toXDelta="0%p"
  />
</set>
```

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate
    android:duration="3000"
    android:fromYDelta="10%p"
    android:toYDelta="0%p"
    android:fromXDelta="10%p"
    android:toXDelta="0%p"
  />
</set>
```

*Animación letras in Splash Activity*

El resultado final:



*Secuencia vista Splash Activity*

En el SplashActivity se sitúa el código que tiene la lógica si accede un usuario nuevo, registrado o no registrado. Estos tres estados tienen diferentes inicios de la aplicación, el nuevo tendrá que aceptar los términos y condiciones para pasar a ser un usuario no registrado, una vez que ya haya aceptado esto no volverá a tener que aceptarlos.

A continuación se muestra el código con la lógica:

```
private void openApp(boolean b) {  
  
    // Se establece un retraso de 5 segundos  
    Handler handler = new Handler();  
    handler.postDelayed(() -> {  
  
        Intent intent;  
  
        FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();  
        FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();  
  
        if (firebaseUser != null) {  
            Intent mainIntent = new Intent( packageContext: SplashActivity.this, MainBn.class);  
            startActivity(mainIntent);  
            finish();  
        } else {  
            if (b) {  
                intent = new Intent( packageContext: SplashActivity.this, MainActivity.class);  
            } else {  
                intent = new Intent( packageContext: SplashActivity.this, Fragment1.class);  
            }  
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);  
  
            startActivity(intent);  
        }  
    }, delayMillis: 5000);  
}
```

*Firebase in Splash Activity*

## ► TermsOfService

Tras ingresar por primera vez en la aplicación, se presentan los términos y condiciones que el usuario debe de aceptar para convertirse en un usuario no registrado. Hasta que no se acepten estos términos, el acceso a la aplicación permanecerá restringido.

### Eventos

Te ayudaremos a encontrar eventos que sean adecuados para ti y tus intereses. También te ayudaremos a reservar entradas y a planificar tu viaje.

Evita asistir a eventos que sean demasiado caros, que tengan lugar en lugares remotos o que no sean de tu interés. Estos eventos pueden ser una pérdida de tiempo y dinero.



Siguiente

### Planificación de horario

Te ayudamos a organizar tu horario de forma eficiente, basada en tus prioridades y compromisos.

Te ayudamos a encontrar tiempo para las cosas que más te importan, sin perderte nada.



Atrás Siguiente

### Información básica sobre Protección de Datos

**Responsable**  
CONNECTSPORT S.L.

**Finalidad**  
Envío de comunicaciones comerciales.

**Destinatarios**  
Únicamente se cederán datos en caso de obligación legal.

**Información detallada**  
Al pulsar aceptar aceptas nuestros términos y condiciones.

☐ Acepto los términos y condiciones.

ACEPTAR

Vista Términos y condiciones

También cuando inicias por primera vez pedirá permiso para acceder a la ubicación exacta de su dispositivo móvil para la funcionalidad completa de la aplicación.

```
private void requestLocationPermission() {
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        locationPermissionGranted = true;
    } else {
        ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_PERMISSION_REQUEST_CODE);
    }
}

1 usage  Jacobosr3
private void getLastKnownLocation() {
    fusedLocationClient = LocationServices.getFusedLocationProviderClient( activity: this);
    Jacobosr3
    fusedLocationClient.getLastLocation().addOnSuccessListener( activity: this, new OnSuccessListener<Location>() {
        Jacobosr3
        @Override
        public void onSuccess(Location location) {
            if (location != null) {
                double latitude = location.getLatitude();
                double longitude = location.getLongitude();
                // Usa la información de ubicación como necesites aquí
                Log.d( tag: "Location", msg: "Latitud: " + latitude + ", Longitud: " + longitude);
            }
        }
    });
}
```

Código de permiso ubicación

Para hacer estas comprobaciones es necesario que en el archivo AndroidManifest.xml del proyecto se añada las siguientes línea de código.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyDSkoTwYe_TII5yYyjh5JS_wxLIjPURWtE"
/>
```

*Control ubicación AndroidManifest.xml*

Para comprobar si ya se había marcado anteriormente los términos se ha utilizado la utilidad de SharedPreferences que se utiliza para almacenar y recuperar pequeños conjuntos de datos de forma persistente. Se puede acceder fácilmente desde cualquier parte de la aplicación y ofrece una forma conveniente de mantener el estado de la aplicación entre sesiones.

Almacenando un valor booleano en SharedPreferences que indique si el usuario ha aceptado los términos y condiciones o no. Si el valor está establecido como falso se seguirá mostrando los términos hasta que el valor cambie a verdadero.

```
public class SharedPrefsUtil {

    4 usages
    private static final String PREFS_NAME = "VALUES";

    1 usage  🧑 Jacobosr3
    public static void saveBoolean(Context context, String key, boolean value) {
        SharedPreferences prefs = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putBoolean(key, value);
        editor.apply();
    }

    1 usage  🧑 Jacobosr3
    public static boolean getBoolean(Context context, String key) {
        SharedPreferences prefs = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        return prefs.getBoolean(key, b: false);
    }
}
```

*Código SharedPreferences*



### ➤ MainActivity

En esta actividad se muestra el inicio de la aplicación. La pantalla presenta el nombre y el eslogan con dos posibles maneras de entrar a la siguiente actividad, “Crear cuenta” y “Accede con:”. La posibilidad de “Crear cuenta” nos pedirá nuestras credenciales que a posteriori tendremos que verificar mediante la dirección de correo que hemos ingresado, a parte de nuestras de algunos datos personales como, el peso, la altura y la edad. A continuación de haber verificado el correo podremos “Acceder con” pulsando en el icono del sobre e indicando las credenciales o bien con nuestra cuenta de Google pulsando en su logo correspondiente. Una vez que ya hemos accedido a la aplicación correctamente no nos volverá a pedir nuestras credenciales, pese a que cambiemos de dispositivo o hayamos cerrado sesión.



Vista MainActivity

En el código de esta actividad inicia la carga de una imagen que se encuentra en la carpeta de recursos drawable utilizando la biblioteca Glide utilizando la función `.centerCrop()` para ajustar la imagen al tamaño. Otro aspecto a detallar, es la configuración de inicio de sesión con Google, solicitando un token de identificación del cliente web predeterminado, que especifica en los recursos de la aplicación y con la implementación `requestEmail()` solicita acceso al correo electrónico del usuario durante el inicio de sesión.



```
// Cargamos el fondo
Glide.with( activity: this) RequestManager
    .load(R.drawable.deportes) RequestBuilder<Drawable>
    .transition(DrawableTransitionOptions.withCrossFade( duration: 100))
    .centerCrop()
    .into(fondo);

// Configure Google Sign In
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken("11920694656-qvojalmet2ptjla0g4cpttjo2au1kjf2.apps.googl...")
    .requestEmail()
    .build();
mGoogleSignInClient = GoogleSignIn.getClient( activity: this, gso);

googleAccess_btn.setOnClickListener(view -> signInWithGoogle());
```

Código MainActivity

## ➤ SingUp









La creación de una cuenta a través de credenciales esta configurada en 4 fragment en los que se van recopilando los datos mencionados anteriormente. Una vez que el usuario ha rellenado todos los datos del registro y han sido almacenados este método se encarga de guardarlos en un archivo JSON en Firebase Storage, en una ruta específica basada en el UID del usuario.

```
public static void createStorageUser(FirebaseUser user, String peso, String edad, String altura, String status) {
    String uid = user.getId();
    FirebaseStorage storage = FirebaseStorage.getInstance();
    StorageReference rootRef = storage.getReference();
    StorageReference folderRef = rootRef.child( pathString: "documents/users/" + uid + "/userInformation.json");

    Map<String, String> userInformation = new HashMap<>();
    userInformation.put( k: "edad", peso);
    userInformation.put( k: "peso", edad);
    userInformation.put( k: "altura", altura);
    userInformation.put( k: "status", status);

    folderRef.putBytes(new Gson().toJson(userInformation).getBytes(StandardCharsets.UTF_8)) UploadTask
        .addOnSuccessListener(taskSnapshot -> Log.d(TAG, msg: "User information file written successfully")) StorageTask<UploadTask.TaskSnapshot>
        .addOnFailureListener(exception -> Log.w(TAG, msg: "Error writing user information file", exception));
}
```

Código Firebase Storage

|                   |   | (Opcional)  |   |                   |  |
|-------------------|---|-------------|---|-------------------|--|
| Nombre de usuario |  | Edad        |  | Contraseña        |   |
| Email             |  | Altura (cm) |  | Repite contraseña | <br> |
|                   |   | Peso (kg)   |  |                   |  |
| SIGUIENTE         |   | SIGUIENTE   |   | SIGUIENTE         |  |

**Confirma tus datos**

Nombre:

Email:

Altura:

Peso:

Edad:

CANCELAR CONFIRMAR

Vista Fragment SingUp

Para restringir la creación de contraseñas durante el proceso de creación de una cuenta, el método `isPasswordValid(String password)` comprueba que las contraseñas cumplan con los requisitos solicitados.

```
private boolean isPasswordValid(String password) {
    return password.length() > 5 && password.matches( regex: ".*[A-Z].*[0-9].*[@#$$%^&*+=_\\-\\\\[\\\\]{}|;:'\\\\\",.<>/?.*");
}
```

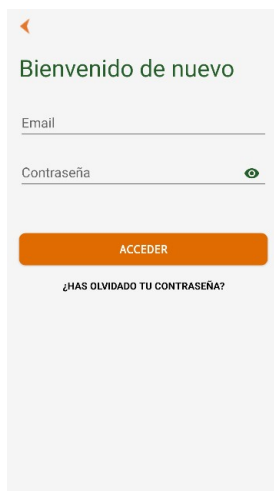
### Requisitos de contraseña

- Al menos 6 caracteres
- Al menos 1 mayúscula
- Al menos 1 número
- Al menos 1 carácter especial

Método requisitos de contraseña

## ➤ Login

Si el registro ha sido correcto podrá acceder mediante la actividad de login la cual ingresando sus credenciales tendrá el acceso completo a la aplicación con todas un funcionalidades.

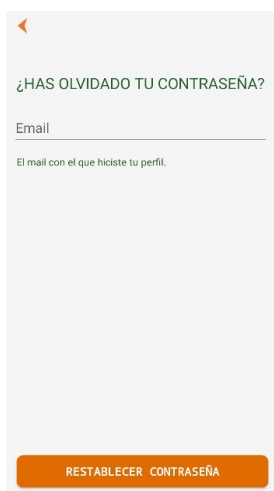


Mockup of the login screen. It features a header 'Bienvenido de nuevo' in green. Below it are two input fields: 'Email' and 'Contraseña' (password), with a toggle icon for the password field. An orange button labeled 'ACCEDER' is centered below the inputs. At the bottom, there is a link '¿HAS OLVIDADO TU CONTRASEÑA?'.

```
mAuth.signInWithEmailAndPassword(editmail.getText().toString(), editpass.getText().toString())
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            FirebaseAuth user = mAuth.getCurrentUser();
            if (user.isEmailVerified()) {
                Toast.makeText(context: Login.this, "Inicio de sesión correcto", Toast.LENGTH_SHORT).show();
                limpiar();
                updateUI(user);
            } else {
                Toast.makeText(context: Login.this, "El email no ha sido verificado", Toast.LENGTH_SHORT).show();
            }
        } else {
            try {
                throw task.getException();
            } catch (FirebaseAuthInvalidCredentialsException e) {
                Toast.makeText(context: Login.this, "Contraseña o email incorrectos", Toast.LENGTH_SHORT).show();
            } catch (FirebaseAuthInvalidUserException e) {
                Toast.makeText(context: Login.this, "Usuario no registrado", Toast.LENGTH_SHORT).show();
            } catch (Exception e) {
                Log.e(TAG, e.toString());
                Toast.makeText(context: Login.this, "Error al iniciar sesión", Toast.LENGTH_SHORT).show();
            }
        }
        updateUI( user: null);
    });
});
```

Vista Login

También ofrece la oportunidad de restablecer la contraseña que habíamos indicado en el registro. Este proceso se efectuara con un correo en el cuál podremos hacer el proceso del reseteo.



Mockup of the reset password screen. It features a header '¿HAS OLVIDADO TU CONTRASEÑA?' in green. Below it is an 'Email' input field. A small text label 'El mail con el que hiciste tu perfil.' is positioned below the input field. An orange button labeled 'RESTABLECER CONTRASEÑA' is centered at the bottom.

```
reset.setOnClickListener(view -> {
    if (editmail.getText().toString().isEmpty()) {
        lMail.setError("Este campo es obligatorio");
    } else if (!validarEmail(editmail.getText().toString())) {
        lMail.setError("Introduce un correo válido");
    } else {
        FirebaseAuth auth = FirebaseAuth.getInstance();
        auth.sendPasswordResetEmail(editmail.getText().toString())
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Intent i = new Intent(packageContext: ResetPassword.this, ResetPassword.class);
                        startActivity(i);
                        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        Toast.makeText(context: ResetPassword.this, "Email enviado", Toast.LENGTH_LONG).show();
                        limpiar();
                    } else {
                        try {
                            throw task.getException();
                        } catch (FirebaseAuthInvalidUserException e) {
                            Toast.makeText(context: ResetPassword.this, "El email introducido no está registrado", Toast.LENGTH_LONG).show();
                        } catch (Exception e) {
                            Log.e(TAG, e.toString());
                            Toast.makeText(context: ResetPassword.this, text: "Error!", Toast.LENGTH_LONG).show();
                        }
                    }
                }
            });
    }
});
```

Código reseteo de contraseña

## ➤ MainBn

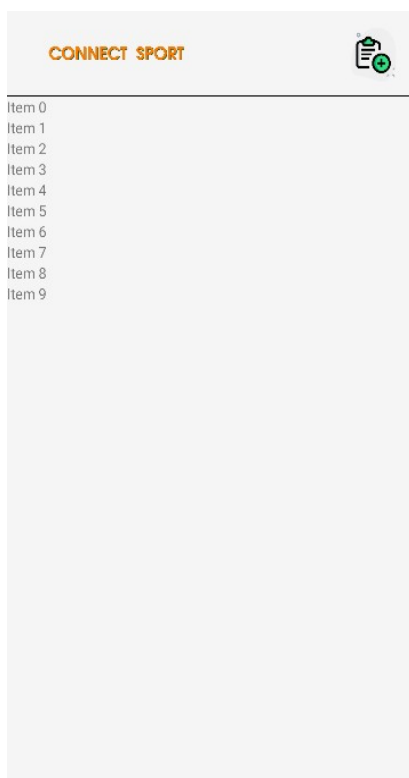
Dentro de la aplicación, las diversas funciones están organizadas en cuatro secciones accesibles a través de un menú: inicio, búsqueda, mapa y perfil.



Menú MainBn

## ➤ FragmentMain1 (Inicio)

En Inicio se sitúa la vista de todos los eventos creados, listados con un RecyclerView ordenado por fecha de creación. En ella hay un botón donde nos lleva a la actividad de creación de eventos, aparte del nombre de la aplicación.



```
// Traemos los eventos por orden de creación ascendente
FirebaseFirestore firestore = FirebaseFirestore.getInstance();
CollectionReference eventsRef = firestore.collection( collectionPath: "events");
Query query = eventsRef.orderBy( field: "createdAt", Query.Direction.DESENDING);
FirestoreRecyclerOptions<Events> options = new FirestoreRecyclerOptions.Builder<Events>()
    .setQuery(query, Events.class)
    .build();

// Configurar el RecyclerView con un LinearLayoutManager
LinearLayoutManager layoutManager = new LinearLayoutManager(getActivity());
layoutManager.setOrientation(RecyclerView.VERTICAL);
feed_events.setLayoutManager(layoutManager);

// Configuramos el firestore recycler adapter
@Jacobos3*
FirestoreRecyclerAdapter<Events, EventsViewHolder> adapter = new FirestoreRecyclerAdapter<>(options) {
    @Jacobos3
    @Override
    protected void onBindViewHolder(@NonNull EventsViewHolder holder, int position, @NonNull Events model) {
        String eventId = this.getSnapshots().getSnapshot(position).getId();
        DocumentReference eventRef = firestore.collection( collectionPath: "events").document(eventId);
        model.setRef(eventRef);
        // Bind the event data to the view holder
        holder.bind(model);
    }
    @Jacobos3
    @NonNull
    @Override
    public EventsViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // Create a new view holder for the event items
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.events_item, parent, attachToRoot: false);
        return new EventsViewHolder(view, listener: FragmentMain1.this, longListener: FragmentMain1.this);
    }
};
```

Código FragmentMain1

En la actividad que nos ha redirigido el botón de creación de eventos podemos poner las características de nuestro propio evento. Así podemos elegir tanto como la ubicación, imágenes, día y hora, una pequeña descripción, la duración de la actividad, las personas que recomendamos que se apunten y el nivel determinado que deberían de tener los participantes.



Vista NewEvents

Puntos a destacar:

- El método `openGallery()` se llama cuando el `ImageButton`. Esto abre la galería del usuario. Cuando el usuario selecciona una imagen y presiona el botón "Aceptar", el sistema operativo devuelve un `Intent` con la ruta de la imagen seleccionada. El método `onActivityResult()` se encarga de obtener la ruta de la imagen y asignarla al `ImagenButton`.

```
// Método que se llama cuando se presiona un ImageButton
3 usages  🐞 Jacobosr3
private void openGallery() {
    Intent intent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, PICK_IMAGE_REQUEST);
}

// Método que se llama cuando el usuario selecciona una imagen de la galería
🐞 Jacobosr3
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK && data != null && data.getData() != null) {
        Uri selectedImageUri = data.getData();
        selectedImageButton.setImageURI(selectedImageUri);
    }
}
```

Código openGallery()

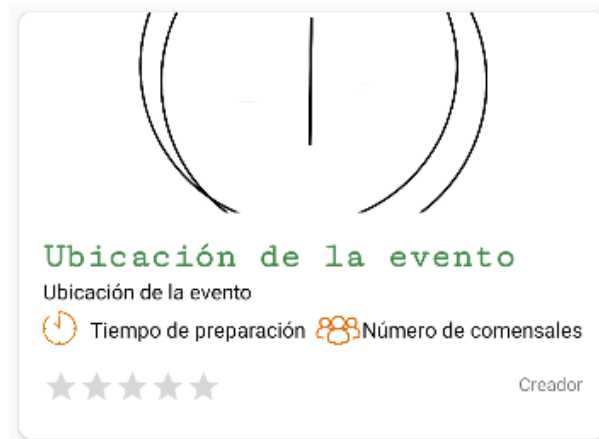
- Descargas de imagen, empleando el método *Tasks.whenAllComplete()* para esperar que todas las tareas de *urlTasks* se completen. Una vez que se hayan agregado todas las URLs de las imágenes descargadas a *imageUrlBuilder*, se convierte en *StringBuilder* a una cadena de texto.

```
// Combinar todas las tareas en una sola tarea
Tasks.whenAllComplete(urlTasks).addOnCompleteListener(task -> {
    if (task.isSuccessful()) {
        StringBuilder imageUrlBuilder = new StringBuilder();
        for (Task<Uri> urlTask : urlTasks) {
            if (urlTask.isSuccessful()) {
                Uri downloadUri = urlTask.getResult();
                if (downloadUri != null) {
                    imageUrlBuilder.append(downloadUri.toString());
                    imageUrlBuilder.append(",");
                }
            } else {
                taskCompletionSource.setException(urlTask.getException());
                return;
            }
        }

        // Devolver una cadena de texto con todas las URLs de las imágenes
        String imageUrlString = imageUrlBuilder.toString();
        if (imageUrlString.length() > 0) {
            imageUrlString = imageUrlString.substring(0, imageUrlString.length() - 1);
        }
        taskCompletionSource.setResult(imageUrlString);
    } else {
        taskCompletionSource.setException(task.getException());
    }
});
```

Código descarga de imágenes

El listado de RecyclerView muestra la vista de los eventos a través de ítems, cada evento se muestra en un ítem diferente. Ejemplo:



Vista ítem evento

Cuando pulsamos sobre el ítem se abre una actividad, mostrando una vista más detallada. Dependiendo si somos los creadores o no, nos mostrara el botón de asistir o borrar el evento. También si pulsamos sobre la ubicación se nos abrirá la aplicación de GoogleMaps.



```
// Cargar ubicación del evento
eventTitle.setText(mEvents.getEventsTitle());

// Redirrección de la ubicación a maps
eventTitle.setOnClickListener(view -> {
    String searchQuery = mEvents.getEventsTitle();
    String url = "https://www.google.com/maps/place/" + Uri.encode(searchQuery);
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(url));
    startActivity(intent);
});
```

Código evento detallado



## ➤ FragmentMain2 (Búsqueda)

Esta actividad está diseñada para ayudar a los usuarios a buscar y filtrar eventos según sus intereses y preferencias. La interfaz de usuario se divide en dos secciones principales: la barra de Búsqueda por ubicación y la lista de eventos.

La lista de eventos es un menú el cual puedes filtrar por más o menos participantes, o bien, marcando el tipo de deporte el cuál que quieres encontrar. Integrado está funcionalidad en un ImageButton con forma de lupa.

```
case R.id.menu_option1:
    // Cargamos los eventos por orden de participantes ascendente
    firestore = FirebaseFirestore.getInstance();
    eventsRef = firestore.collection( collectionPath: "events");
    query = eventsRef.orderBy( field: "eventsAttend", Query.Direction.DESENDING);
    options = new FirestoreRecyclerOptions.Builder<Events>()
        .setQuery(query, Events.class)
        .build();
```

*Código filtrado por participantes*

Cuando pulsamos en “Tipo de deporte” se despliega un submenú con todos los deportes que hay:



```
case R.id.menu_option3:
    androidx.appcompat.widget.PopupMenu popupMenu = new androidx.appcompat.widget.PopupMenu(getContext(), imageView);
    popupMenu.getMenuInflater().inflate(R.menu.submenu, popupMenu.getMenu());
    popupMenu.setOnMenuItemClickListener(subMenuItem -> {
        switch (subMenuItem.getItemId()) {
            case R.id.submenu_option1:
                // Cargamos los eventos
                firestore = FirebaseFirestore.getInstance();
                eventsRef = firestore.collection( collectionPath: "events");

                query = eventsRef.whereEqualTo( field: "selectedEventsType", value: "Fútbol");

                options = new FirestoreRecyclerOptions.Builder<Events>()
                    .setQuery(query, Events.class)
                    .build();
```

*Código filtrado por tipo de deporte*



### ➤ FragmentMain3 (Mapa)

Para la implementación del mapa en nuestra aplicación habrá que implementar la correspondiente dependencia de Google Maps "implementation ("com.google.android.gms:play-services-maps:18.1.0")" en el build.gradle a módulo de app. Aparte de los permisos en AndroidManifest y la clave API que obtendremos previamente en la *Consola de Google Cloud*.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyDSk[REDACTED]S_wxLIjPURWtE"
/>
```

*Permisos en AndroidManifest*

En la configuración del mapa, se recorre la base de datos para guardar las localidades en un arraylist. Una vez que se han recorrido todos los documentos de la colección, se utiliza un bucle for para recorrer el arraylist. En cada iteración del bucle, se utiliza el método *addMarkerByLocationName()* para añadir un marcador en el mapa utilizando la ubicación del evento, mostrando a la vez un mensaje por consola de que se ha añadido correctamente.

```
//Recorrer la base datos para guardar las localidades en un arraylist
@Jacobosr3
eventsRef.get().addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
    @Jacobosr3
    @Override
    public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
        for (QueryDocumentSnapshot documentSnapshot : queryDocumentSnapshots) {
            String campo = documentSnapshot.getString( field: "eventsTitle");
            if (!arrayList.contains(campo)) {
                arrayList.add(campo);
            }
        }
        for (String event : arrayList) {
            Log.d( tag: "AÑADIDO.....", msg: "Event: " + event);
            // Añadir marcador utilizando la ubicación del evento
            addMarkerByLocationName(event);
        }
    }
});
```

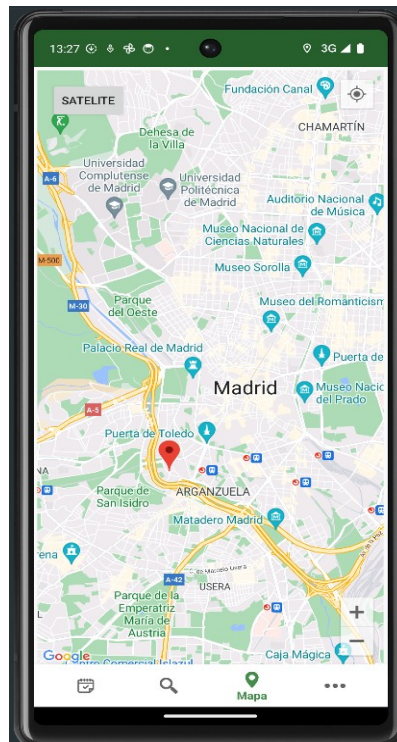
*Código recorrer base de datos*

Para cambiar la vista del mapa hemos implementado un botón cambiando su valor cada vez que pulsamos en el. Es una variable booleana que indica si el mapa está en modo híbrido. El modo híbrido muestra una combinación de imágenes de satélite y mapas de calles.

```
mTypeBtn.setOnClickListener(new View.OnClickListener() {  
    3 usages  
    boolean isHybrid = true;  
    // Jacobosr3  
    @Override  
    public void onClick(View v) {  
        if (isHybrid) {  
            mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);  
        } else {  
            mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);  
        }  
        isHybrid = !isHybrid;  
    }  
});
```

*Código tipo de mapa*

El método `mMap.setMyLocationEnabled(true)` se utiliza para habilitar la capa de “Mi ubicación” en el mapa, lo que significa que muestra la ubicación actual del usuario en el mapa.



*Vista del mapa*

### ➤ FragmentMain4 (Perfil)

Cargamos los datos guardados del JSON en firebase mediante listener con interfaz en StorageUtil y actualizar la interfaz del usuario una vez que los datos ya estén disponibles.

```
StorageUtil.OnReadUserInformationListener listener = userInformation -> {  
    userStatus.setText(userInformation.get("status"));  
    icm_weight.setText(userInformation.get("peso"));  
    icm_height.setText(userInformation.get("altura"));  
};  
StorageUtil.readStorageUser(user, listener);  
  
// Cargar datos del objeto user de firebase  
userTextView.setText(user.getDisplayName());  
mailTextView.setText(user.getEmail());  
Glide.with(fragment: this).load(user.getPhotoUrl()).into(profilePicture);
```

Código StorageUtil

A través de los datos cargados, podemos calcular el IMC(índice de masa corporal) mediante una formula matemática “  $\text{Peso(Kg)}/\text{Altura(m)}^2$  “

```
public static double calcularIMC(double pesoKg, double alturaCm) {  
    // Convertir altura a metros  
    double alturaM = alturaCm / 100;  
  
    // Calcular el IMC  
    double imc = pesoKg / (alturaM * alturaM);  
  
    return imc;  
}
```

Código calcular IMC

Los datos del usuario como peso, altura, descripción e imagen se pueden modificar. Para modificar la foto de perfil se desplegará un menú emergente, cuando el usuario hace clic en la imagen de perfil, ofreciendo opciones para actualizar o eliminar.



```
// Cambiar foto de perfil
profilePicture.setOnClickListener(view1 -> {
    PopupMenu popup = new PopupMenu(getContext(), profilePicture);
    popup.getMenuInflater().inflate(R.menu.updateordelete_profile_pic, popup.getMenu());
    popup.setOnMenuItemClickListener(item -> {
        switch (item.getItemId()) {
            case R.id.update:
                // Code to update the picture
                selectImageFromGallery();
                return true;
            case R.id.delete:
                // Code to delete the picture
                if (user.getPhotoUrl() == null) {
                    //Picture is already null, what u gonna delete
                    Toast.makeText(getActivity(), text: "No hay ninguna foto!", Toast.LENGTH_SHORT).show();
                } else {
                    profilePicture.setImageBitmap(null);
                    updateUserProfileWithImageUri(null);
                }
                return true;
            default:
                return false;
        }
    });
    popup.show();
});
```

*Código editar foto de perfil*

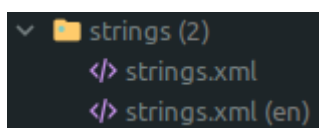
En resumen, la sección de perfil ofrece una visión general de la información que proporcionaremos durante el registro, permitiéndonos realizar ediciones si es necesario.

Además presenta un contador de los eventos que hemos creado. En esta área, encontramos dos botones adicionales: uno destinado a brindar información detallada sobre los términos y condiciones que previamente aceptamos, y otro para cerrar sesión en la plataforma.

### ✖ Internacionalización de la aplicación (Inglés/Español)

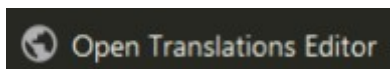
Con el objetivo de garantizar la disponibilidad de la aplicación en diversos idiomas y su actualización conforme al idioma del dispositivo móvil, se han implementado los siguientes pasos:

- Creación de recursos: Carpeta res ➡ values ➡ strings



*Archivo strings proyecto*

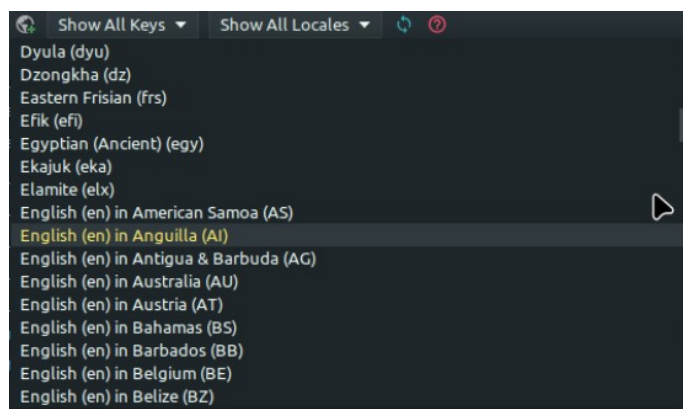
- Botón derecho con el ratón y seleccionamos en el nuevo strings



*Opción de AndroidStudio para internacionalizar la aplicación*

- Se eligen los idiomas de la aplicación según la configuración de idioma del dispositivo.

Para modificarlos , toca el icono del globo terráqueo seguido de la tecla de adicción (“+”)



*Selector de idiomas para la aplicación*

- En ConnectSport, hemos establecido el español como idioma predeterminado, pero la aplicación está también disponible el inglés. Esto significa que si el dispositivo esta configurado en uno distinto al español o inglés, la aplicación se mostrara en español.

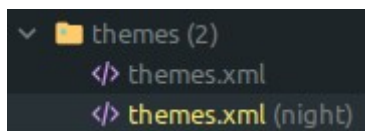
|             |                  |                          |                                |                                |
|-------------|------------------|--------------------------|--------------------------------|--------------------------------|
| app_name    | app/src/main/res | <input type="checkbox"/> | ConnectSport                   | ConnectSport                   |
| siguiente   | app/src/main/res | <input type="checkbox"/> | Siguiente                      | Next                           |
| atras       | app/src/main/res | <input type="checkbox"/> | Atrás                          | Back                           |
| aceptar     | app/src/main/res | <input type="checkbox"/> | ACEPTAR                        | ACCEPT                         |
| eventos     | app/src/main/res | <input type="checkbox"/> | Eventos                        | Events                         |
| evento      | app/src/main/res | <input type="checkbox"/> | Te ayudaremos a encontrar e    | We'll help you find events tha |
| evento2     | app/src/main/res | <input type="checkbox"/> | Evita asistir a eventos que se | Avoid attending events that a  |
| horarios    | app/src/main/res | <input type="checkbox"/> | Planificación de horario       | Schedule planning              |
| horario     | app/src/main/res | <input type="checkbox"/> | Te ayudamos a organizar tu h   | We help you organize your sc   |
| horario2    | app/src/main/res | <input type="checkbox"/> | Te ayudamos a encontrar tien   | We help you find time for the  |
| datos       | app/src/main/res | <input type="checkbox"/> | Información básica sobre Prot  | Basic information on Data Pro  |
| responsable | app/src/main/res | <input type="checkbox"/> | Responsable                    | Responsible                    |

*Traducción de strings en los idiomas seleccionados*

## ✖ Modo Oscuro

La implementación del modo oscuro en la aplicación ConnectSport es parecida a la internacionalización de la misma. Guía paso a paso para implementar el modo oscuro:

- Creación de recursos: Carpeta res  values  themes



*Archivo themes proyecto*

- Ya creado el archivo, es necesario ajustar el tema para que los colores seleccionados se apliquen automáticamente cuando se cambie la configuración del dispositivo a modo oscuro.

*parent="Theme.AppCompat.DayNight.NoActionBar"*

```
<style name="Theme.ConnectSport." parent="Theme.AppCompat.DayNight.NoActionBar">
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/greend</item>
    <item name="colorPrimaryVariant">@color/green2d</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/oranged</item>
    <item name="colorSecondaryVariant">@color/purple2d</item>
    <item name="colorOnSecondary">@color/black</item>
    <!-- Status bar color. -->
    <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
    <!-- Customize your theme here. -->
    <item name="colorSurface">@color/white</item>
    <item name="colorOnSurface">@color/black</item>

    <item name="backgroundColor">@color/fadeBlack</item>
    <item name="popupMenuBackground">@color/moreFadeBlack</item>
</style>
```

*Configuración Dark Theme*

Para obtener un análisis más detallado de secciones del código, se recomienda consultar el Anexo II.



## 5.2 Problemas encontrados y soluciones

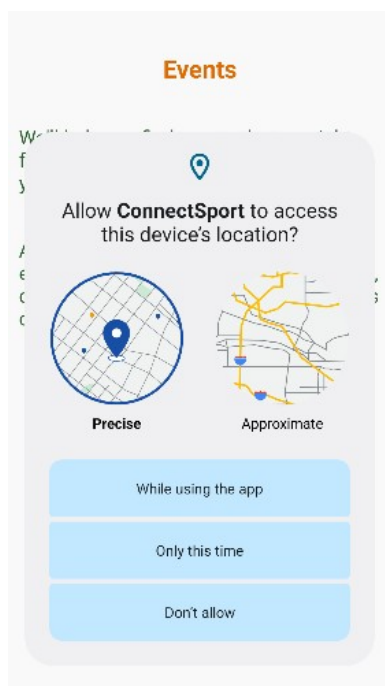
Se expondrán los problemas que han ido surgiendo en el desarrollo de la aplicación y las soluciones implementadas de los mismos.

### Permiso de ubicación

La solicitud de permisos para acceder a la ubicación exacta del usuario se realizaba al ingresar en la vista del mapa. Sin embargo, esto generaba al posibilidad de que la ubicación no se mostraba la primera vez que el usuario accedía, ya que aún no se le había otorgado el permiso hasta después. Para visualizar la ubicación a tiempo real, era necesario cerrar la aplicación y volver abrirla, a partir de ese punto el funcionamiento se normalizaba correctamente.

Como solución, se implementó la solicitud de permisos antes de ingresar a la vista del mapa. De esta manera, al acceder por primera vez, todas las funciones estaban disponibles desde el inicio. La solicitud de permisos se incorporó inmediatamente después de abrir la aplicación, dentro de los términos y condiciones, antes de cualquier otra funcionalidad.

```
1 usage  Jacobosr3
private void requestLocationPermission() {
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        LocationPermissionGranted = true;
    } else {
        ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_PERMISSION_REQUEST_CODE);
    }
}
```



*Permiso de ubicación en términos y condiciones*

## Menú y submenú

En la vista de búsqueda, inicialmente se diseñó un solo archivo XML, colocando un submenú con opciones adicionales en el tercer ítem. Esto resultó en la necesidad de realizar dos pulsaciones al seleccionar alguna de esas opciones para que se ejecutaran la función correspondiente.

Se implementó una solución al inconveniente mediante la incorporación de un archivo XML adicional que contiene las configuraciones del submenú previo. De esta manera, al optar por la tercera opción, se procederá a la apertura del nuevo menú teniendo que pulsar solo una vez para que se realice la función.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/submenu_option1"
        android:title="Fútbol" />
    <item
        android:id="@+id/submenu_option2"
        android:title="Baloncesto" />
    <item
        android:id="@+id/submenu_option3"
        android:title="Padel" />
    <item
        android:id="@+id/submenu_option4"
        android:title="Tenis" />
    <item
        android:id="@+id/submenu_option5"
        android:title="Voleibol" />
    <item
        android:id="@+id/submenu_option6"
        android:title="Balonmano" />
</menu>
```

Submenú en nuevo archivo XML



## Calcular IMC

Los parámetros del método *CalcularImc(double pesoKg, double alturaCm)* estaban intercambiados, pasando la altura como peso y viceversa. Para resolver el problema se cambió de posición los parámetros y ahora la altura se proporciona como altura y el peso como peso.

```
1 usage  🧑 Jacobosr3
public static double calcularIMC(double pesoKg, double alturaCm) {
    // Convertir altura a metros
    double alturaM = alturaCm / 100;

    // Calcular el IMC
    double imc = pesoKg / (alturaM * alturaM);

    return imc;
}
```

*Solución método CalcularImc()*

Para verificar la solidez de la solución propuesta en el desarrollo, se ha sometido a exhaustivas pruebas de rendimiento y funcionalidad. Estas pruebas se llevaron a cabo mediante un emulador que simulaba la ejecución de la aplicación en diversas condiciones.

Se llevaron a cabo pruebas de robustez para identificar posibles vulnerabilidades y asegurar la fiabilidad del sistema frente a situaciones inesperadas. Se prestó especial atención a los posibles cuellos de botella y se realizaron ajustes en consecuencia para optimizar el rendimiento general.

Las pruebas exhaustivas realizadas con el emulador brindaron una validación completa de la solución propuesta, asegurando que la aplicación sea capaz de cumplir con los requisitos esperados y ofrecer una experiencia confiable a los usuarios finales.

## 6. FUENTES

Adobe. (s.f.). Color Contrast Analyzer. <https://color.adobe.com/es/create/color-wheel>

Flaticon. (s.f.). <https://www.flaticon.es/>

Calcula tu IMC (Índice de Masa Corporal) – UCOM (s.f.). <https://cirugiayobesidad.es/calcular-imc/>

Documentación de firebase (s.f.). <https://firebase.google.com/docs/auth/android/start?hl=es>

API Google Maps Platform (s.f.). <https://developers.google.com/maps/documentation/android-sdk?hl=es-419>

## 7. ANEXOS

### **Anexo I**

Vídeo demostrativo de la funcionalidad. *Video(.mp4)*

### **Anexo II**

*Se trata de un archivo .zip que contiene el código completo de la aplicación. Código fuente(.zip)*