



CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

EventSports

Iván Hernán, Sebastián Huete, Álvaro Tuñón

CURSO 2020-21

TÍTULO : EventSports

AUTORES: Iván Hernán Pérez

Sebastián Huete Londoño

Álvaro Tuñón Berlanga

TUTOR DEL PROYECTO: Ernesto Ramiro Córdoba

FECHA DE LECTURA: 7 de Junio de 2021

En Madrid

Ernesto Ramiro Córdoba
Tutor del PFC

RESUMEN:

Muchas veces tenemos ganas de realizar actividades deportivas que requieren un mínimo de participantes, pero lamentablemente algunos de nuestros amigos no están disponibles en ese momento y no somos los suficientes para realizar ese deporte que tantas ganas teníamos de hacer. También puede existir la posibilidad de que quieras conocer gente nueva y compartir con ellos los mismos gustos deportivos o tener la pasión de ofrecer tus conocimientos que has adquirido en algún momento de tu vida a gente que lo deseé y que está dispuesta a escucharte y a seguir tus pasos.

Este proyecto va dirigido a todas aquellas personas que se encuentran en estos casos, y por supuesto, quieran disfrutar en grupo de la magia del deporte sin límites.

No existirá ningún tipo de restricción de edad, ya que pensamos que independientemente de los años que tengas, cualquiera puede aportar cosas beneficiosas a la gran comunidad de EventSports.

¿Cómo conseguiremos todo lo mencionado anteriormente? De una forma clara y concisa. La aplicación tendrá dos tipos de usuario, y tú puedes elegir cuál y cuándo serlo, incluso de manera simultánea. El primero de ellos te dará la posibilidad de crear un evento con el objetivo de ofrecer clases de un deporte cualquiera o simplemente convocar una quedada para realizar cualquier tipo de actividad física. El segundo tipo de usuario mostrará todos aquellos eventos que nuestra comunidad ha creado para que todos podamos pasar un rato agradable y disfrutarlo en compañía. Para apuntarte a esa actividad que tanto deseas, únicamente tendrás que hacer clic en el evento e inscribirse.

¿A qué esperas? Los límites los pones tú.

ABSTRACT:

Many times we want to do sports activities that require a minimum of participants, but unfortunately, some of our friends are not available at the time and we are not enough to perform that sport that we so much wanted to do. There may also be the possibility that you want to meet new people and share with them the same sporting tastes or have the passion to offer the knowledge that you have acquired at some point of your life to people who want it and who are willing to listen to you and follow your steps.

This project is aimed at all those people who are in such cases, and of course, want to enjoy in group the magic of sport without limits. There will be no age restriction whatsoever, as we believe that regardless of how old you are, you can contribute beneficially to the great EventSports community.

How will we achieve all of the above? In a clear and concise way. The app will have two types of users, and you can choose which and when to use it, even simultaneously. The first of them will give you the possibility to create an event with the aim of offering classes of any sport or simply call a meeting to perform any type of physical activity. The second type of user will show you all those events that our community has created so that we can all have a nice time and enjoy it in company. To sign up for that activity you want so much, you just have to click on the event and sign up.

What are you waiting for? You set the limits.

AGRADECIMIENTOS

Nuestro agradecimiento va dirigido a todos aquellos profesores que nos han formado estos dos años como técnicos superiores. Nos habéis ayudado a conocer y abrirnos un hueco en el mundo de la programación, y a muchos de nosotros a descubrir la pasión que teníamos dentro ya que a algunos nos hacía falta un empujón para encontrarla. También queríamos mencionar a aquellas personas (amigos/as, pareja, familia) que han estado apoyándonos en esos momentos tan duros que nos hicieron plantear si lo correcto era seguir avanzando en nuestro deseo de ser grandes programadores o quedarnos estancados. Esos momentos son muy difíciles para todos, pero gracias a esas personas se ha podido lograr.

Sin ellos, no podríamos desarrollar este gran proyecto que puede aportar un granito de arena en la salud física de cada uno de nosotros. Muchas gracias.



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia original completa(jurídicamente válida) que puede encontrarse en:

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>

RESUMEN:	1
ABSTRACT:	3
AGRADECIMIENTOS	3
INTRODUCCIÓN	8
Objetivos	8
Motivación	8
Antecedentes	9
DESARROLLO DEL PROYECTO	10
Herramientas tecnológicas	11
Firebase	11
Java	14
Android Studio	14
Planificación	18
Descripción del trabajo realizado	20
Paleta de colores y Tipografía	20
Mock Up	21
Splash Screen	24
Inicio de Sesión (Log In)	25
Nueva Cuenta (Sign Up)	27
Deportes (Main)	28
Mapa	29
Nuevo Evento o Clase	30
Lista Usuarios	32
Perfil	34
Navigation Bar	35
Chat	36
Specify Category	38
Imágenes de Galería y Permisos	40
Resultados y validación	41
CONCLUSIONES	42
Innovación	43
Trabajo Futuro	43
BIBLIOGRAFÍA Y WEBGRAFÍA	44
ANEXOS	45
A.1 Comprobación Formularios	45
A.2 Layouts SpecifyCategory	46
A.2.1 Activities SpecifyCategory	51

1. INTRODUCCIÓN

El propósito de EventSports es fomentar el deporte a cualquier tipo de público, dando la posibilidad de generar eventos deportivos por y para sus usuarios.

1.1. Objetivos

- Diferenciar y distinguir los posibles eventos generados por los usuarios (clases y quedadas deportivas).
- Localizar el evento propuesto.
- Permitir a los usuarios inscribirse en los eventos creados.
- Establecer una comunicación entre usuarios.

El primer objetivo es encontrar una manera de poder dejar claro qué tipo de evento se trata para que no haya confusiones a la hora de concluir la inscripción. Al realizar este paso, el usuario tiene que tener la facilidad de localizar los eventos propuestos. Como se ha mencionado anteriormente, cada uno tiene la posibilidad de inscribirse a cualquier clase o quedada deportiva. Para asegurar que no haya ningún tipo de confusión, se establecerá comunicaciones entre usuarios mediante un intercambio de mensajes por chat.

1.2. Motivación

Se ha elegido este proyecto ya que hoy en día se le da poca importancia a la salud física y mental de cada individuo. La falta de éste puede afectar de forma perjudicial al estado de ánimo y, por lo tanto, al rendimiento en las actividades a lo largo del día. Por ello, vemos la necesidad de fomentar la actividad física mediante el proyecto EventSports creando eventos deportivos en los que cualquier persona puede participar. Cualquier ejercicio físico, por poco que sea, es importante para mejorar la salud física y mental.

1.3. Antecedentes

Se quiere recalcar que no hay antecedentes establecidos que traten sobre la idea del proyecto EventSports pero sí algunos que se asemejan:

En estos últimos años se ha visto un gran avance tecnológico y, por lo tanto, una gran cantidad de nuevas apps para dispositivos móviles. Esto ha dificultado de cierta manera el encontrar una aplicación que tenga la oportunidad de resaltar entre las demás. El equipo de EventSports ha sacado la conclusión de que hay pocas apps que permitan la creación de eventos a sus usuarios pero, en cambio, las que existen tienen un gran éxito como las de quedadas de coches deportivos. A partir de esto, se ha tomado como idea principal la creación de eventos. También EventSports ha querido darle un toque personal, dirigir la aplicación al mundo del deporte pues hay cierta tendencia en la sociedad a tener más en cuenta la salud física.

La elección de la distribución de las interfaces y la aparición de algunos elementos se ha realizado estudiando algunas de las aplicaciones más conocidas como "Instagram", "Youtube", "Carmeets"...

2. DESARROLLO DEL PROYECTO

EventSports es una aplicación dirigida a los eventos deportivos y clases deportivas. La realización de este trabajo ha sido simple y bien organizada.

¿Qué es lo que se ha hecho? En este trabajo se ha realizado una aplicación en la cual nada más entrar te pide registrarte o iniciar sesión para saber quien eres y tener un nombre de usuario para poder comunicarte en la aplicación. También se ha creado una lista de deportes en los cuales podrás elegir entre si quieres participar en un evento o dar una clase de dicho deporte. No solo eso, podrás crear tus propios eventos o tus propias clases. Si te apuntas en ellas, te dará permiso para entrar en un chat global en el que te podrás comunicar con aquellos que estén apuntados a esa clase o evento.

Se ha decidido usar unos colores diferentes al logo para contrastar dichos temas, ¿Por qué se ha hecho así? porque buscábamos algo simple en cuanto a colores y temas para que la gente se centre en lo importante de la aplicación que es el deporte.

Para esta aplicación usa diversas aplicaciones para la creación del logo por ejemplo o diferentes sitios web como firebase para bases de datos, búsqueda de información, librerías ... etc. Cuando se terminaba una interfaz o funcionalidad de la aplicación, se le mostraba al resto del equipo y se buscaban distintos fallos y las aprobaciones para añadir eso al proyecto, todas las ideas son bienvenidas.

Destacar la plataforma de GitHub que ha sido donde se ha ido subiendo paso por paso todo lo que se ha ido haciendo hasta llegar a la finalización del trabajo.

Todo esto está más detallado en las siguiente páginas.

2.1. Herramientas tecnológicas

Para la realización de este trabajo, Se ha usado diversas herramientas tecnológicas que nos han sido útiles y de vital importancia para crear este proyecto de fin de ciclo. Estas herramientas suelen ser conocidas en el sector de la programación e incluso algunas sin saber de programación.

Las tecnologías usadas son Firebase, la cual ha servido de base de datos para guardar la información que recogía la aplicación. Java, es el lenguaje en el cual se ha escrito todo el código de la aplicación y, Android studio, “plataforma” o IDE (Integrated Drive Electronics, Entorno de Desarrollo Integrado) donde se ha ido creando y generando interfaces de la aplicación.

• **Firebase**

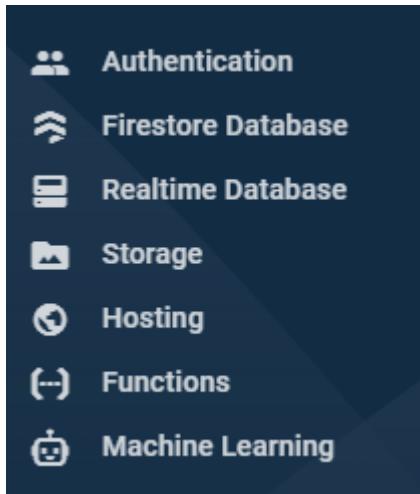
Firebase se trata de una plataforma móvil creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida, con el fin de que se pueda aumentar la base de usuarios y ganar más dinero.



Logo Firebase

Para comunicarte en Firebase se utiliza un único SDK (Software Development Kit), el cual es un grupo de herramientas que ayudan en el entorno de programación Android, IOS, etc.

La utilización de esta plataforma es muy simple e intuitiva para poder conectarse a ella simplemente poniendo este link <https://firebase.google.com/> y registrándose con tu cuenta Google, se comenzará creando un proyecto para futuras aplicaciones. Una vez dentro, se encontrará un menú en la parte de la izquierda en el cual se pueden realizar diversas Funcionalidades, de las cuales tres han sido de gran importancia para la realización del proyecto.



Menú de Firebase

El primero es el “Authentication”. En este apartado, encontrarás la funcionalidad de los proveedores de acceso, es decir, “canales” o “formas” por las cuales los usuarios se van a registrar o iniciar sesión en la aplicación como, por ejemplo, Google, Facebook, Correo Electrónico, etc...

Proveedores de acceso	
Proveedor	Estado
✉ Correo electrónico/contraseña	Habilitada
📞 Teléfono	Inhabilitado
🇬 Google	Habilitada
▶ Play Juegos	Inhabilitado
🎮 Game Center	Inhabilitado
🇫 Facebook	Inhabilitado

Proveedores de acceso

En la aplicación se han usado dos, las cuales son Google y Correo electrónico, esta parte estará más detallada más abajo.

Otra de las funcionalidades usadas de Firebase “Realtime Database”, en la cual se encuentra la base de datos, es decir, toda la información que quieras recoger en la aplicación, como ejemplo: los datos de los usuarios, de eventos etc.



La tercera funcionalidad de Firebase utilizada en la aplicación es el “Storage”, la zona donde se van a guardar todas las imágenes recogidas en la aplicación. En este apartado se puede generar tantas carpetas de imágenes como se necesiten.

Su funcionalidad es muy buena y muy útil a la hora de usarla en una aplicación.



• Java

Java es un lenguaje de programación y una plataforma informática rápida, segura y fiable que se comercializó hace más de 25 años por Microsystem. Java se utiliza desde portátiles hasta centros de datos, desde juegos de móvil hasta en supercomputadores, incluso en internet. Java está en todas partes.



Logo de Java

Se ha utilizado Java para el desarrollo del proyecto de fin de ciclo. Se ha usado este lenguaje no solo por lo intuitivo y seguro que es, sino también por las grandes funcionalidades y plataformas que se pueden utilizar con dicho lenguaje.

• Android Studio

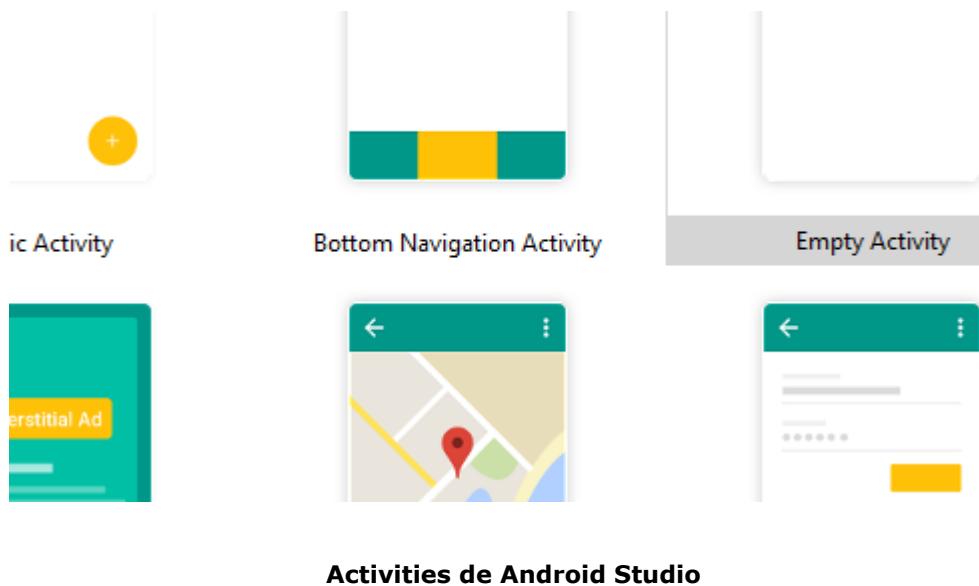
Como bien se ha dicho antes, Android studio es un entorno de desarrollo integrado (IDE) que sirve para el desarrollo de aplicaciones en los dispositivos Android.



Logo de Android Studio

Esta Herramienta ha sido la principal para la realización del proyecto de fin de

ciclo, ya que su utilización es muy simple y rápida a la hora de trabajar. Su funcionalidad consiste en crear un nuevo proyecto y elegir una “actividad”, “interfaz” o “ventana” para trabajar y programar en ella.



Una vez seleccionada, en la parte de la izquierda se encontrará un esquema o menú en el cual se irá viendo la estructura y organización de la aplicación. En la imagen (Figura 1) se ve las diversas clases Java, es decir, donde se irá programando en el lenguaje utilizado (Java,Kotlin,etc.) y en la (Figura 2) se observa las diversas “Interfaces” descritas anteriormente.

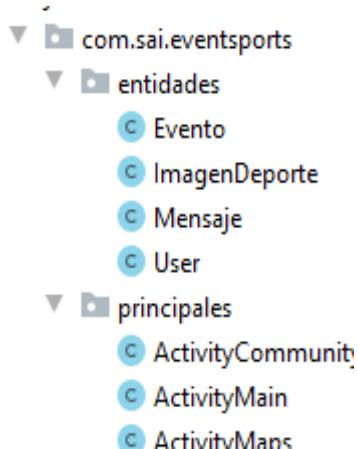


Figura 1: Clases Java

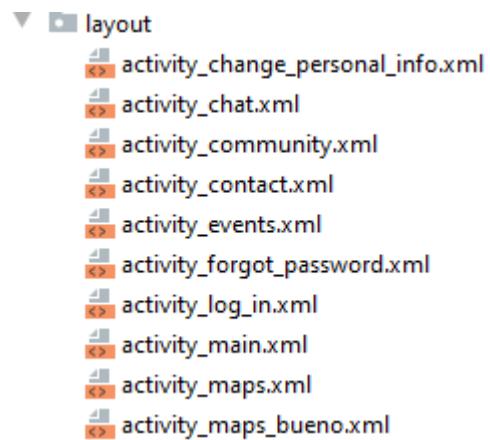


Figura 2: "Activities".xml

Estas “Activities” serán un xml y su funcionalidad y creación puede ser de dos formas muy diversas: La primera, arrastrando “Widgets” hacia una pantalla donde se irán mostrando su colocación (Figura 3) o la segunda, que sería mediante el código (Figura 4).

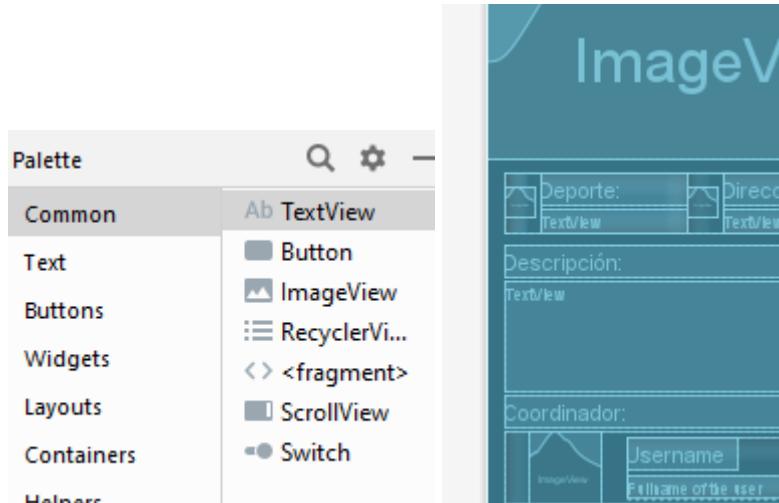


Figura 3: Arrastrando los “Widgets”

```
<androidx.constraintlayout.widget.Const
    xmlns:app="http://schemas.android.c
    xmlns:tools="http://schemas.android
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".ActivityEvents">
    <androidx.core.widget.NestedScrollV
        android:layout_width="match_par
        android:layout_height="match_pa
        <LinearLayout
            android:layout_width="match
```



Figura 4: Programando “Widgets”

Destacar que cada vez que se elige un Activity se genera una clase Java automáticamente.

Por último añadir otra funcionalidad importante de Android studio, la cual es que al crear una aplicación se puede observar en un simulador, que ofrece la propia plataforma de un dispositivo android al igual que un teléfono móvil. Para arrancar dicho simulador se debe crear en la parte superior derecha (Figura 5).



Figura 5

Después, se dará a crear y se elegirá un dispositivo (Figura 6) y para arrancarlo simplemente dando a la flecha verde (Figura 7).

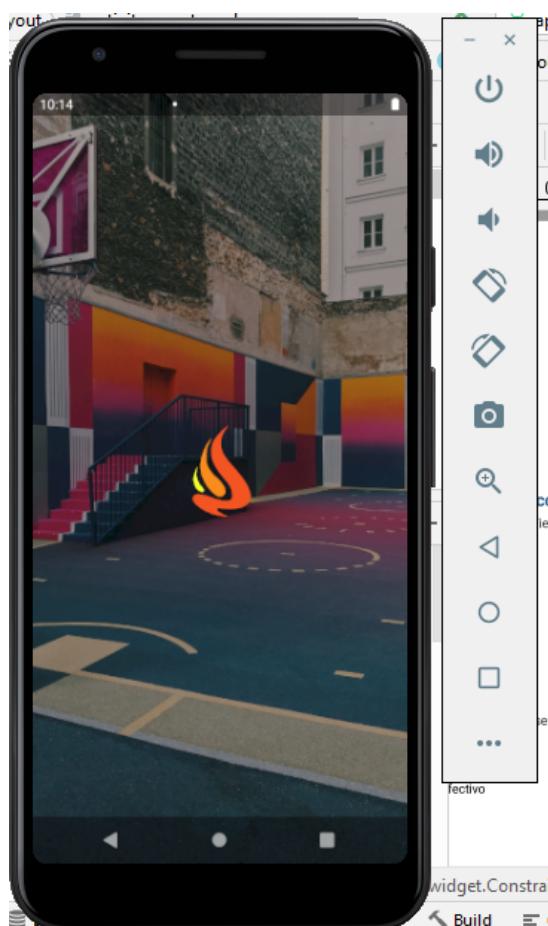


Figura 7

Category	Name
TV	Pixel XL
Phone	Pixel 4 XL
Wear OS	Pixel 4
Tablet	Pixel 3a XL
Automotive	Pixel 3a

Figura 6: Elección de Dispositivo

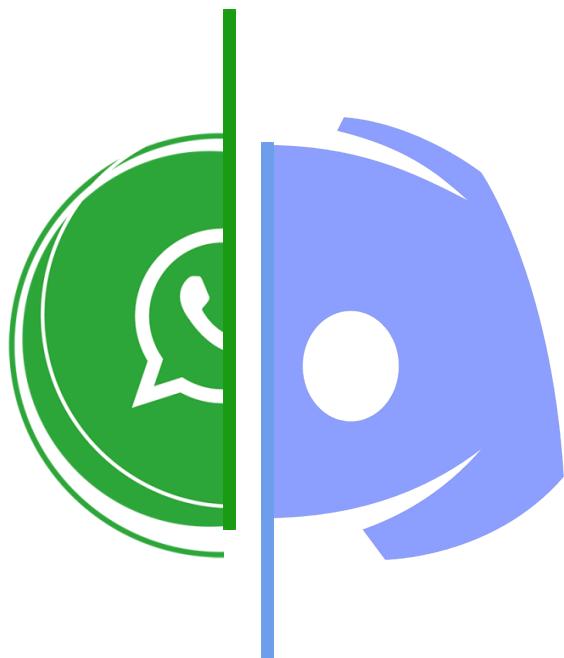
Para terminar, en la siguiente imagen se verá el dispositivo con la aplicación arrancada:



Dispositivo con la aplicación

2.2. Planificación

Como se ha mencionado anteriormente, el equipo está compuesto por tres miembros. La comunicación entre ellos se ha realizado mediante varias herramientas. La principal ha sido la aplicación de chat “Whatsapp” ya que cada uno tenía sus actividades a lo largo del día, por lo que lo más sencillo era intercambiar mensajes por esta app. Cada domingo, se ha realizado videollamada por la aplicación “Discord” para que cada uno enseñase al resto de los miembros del equipo los avances de la tarea correspondiente del proyecto. En alguna ocasión, también se ha utilizado el “Blackboard” de la Universidad Europea con el mismo objetivo de la app “Discord”.

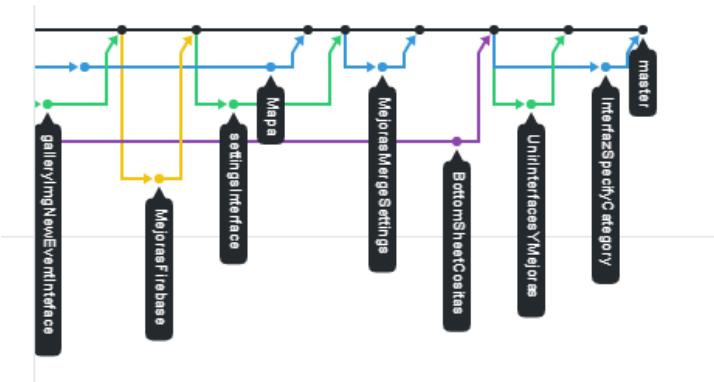


Logos de Whatsapp y Discord

Otra herramienta que ha sido útil para observar el desarrollo de las tareas ha sido “Github”. Con ella, cada uno realizaba sus respectivos “commits” y dejaba registrado todo lo que había avanzado del proyecto. Después, se podía observar todos los “commits” en las ramas que aparecían al entrar en el apartado “Network”.



Logo de GitHub



Network de la Aplicación

La planificación respecto a las actividades para realizar la app EventSports se ha dividido en tres partes. Una de ellas ha consistido en desarrollar la parte visual de la aplicación, es decir, el diseño de las interfaces ya que se considera de gran importancia para llamar la atención de los futuros usuarios. Otro punto de la planificación sería la de base de datos, que consistía en implementar la funcionalidad de la aplicación. Por último, se dejó los días finales para repasar de forma global el proyecto y realizar algunos ajustes para perfeccionar la app EventSports.

Cada punto de la planificación se ha dividido de forma equitativa para que todos los miembros del equipo trabajen por igual y no hayan malentendidos.

2.3. Descripción del trabajo realizado

Aquí se desarrollará y se aplicará lo más importante de las interfaces implementadas en el proyecto de fin de ciclo EventSports.

Paleta de colores y Tipografía

La paleta de colores que utiliza la aplicación EventSports se basa en una escala de azules como principal y en un segundo plano una paleta de colores relacionada con el logo de la aplicación (Figura 1 y 2).

Este tono de azules se podrá observar a lo largo de la aplicación en muchas de las interfaces implementadas.

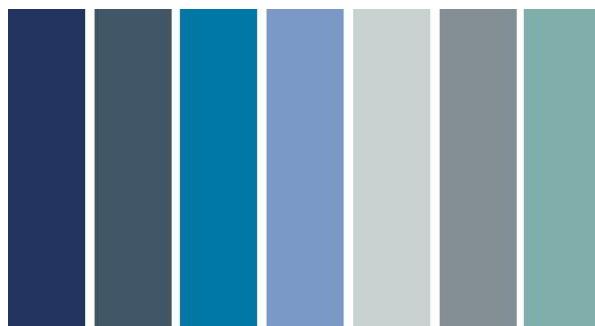


Figura 1: Paleta Principal

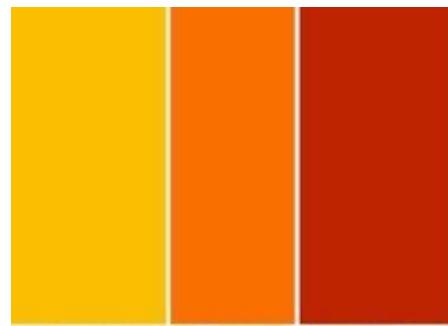


Figura 2: Paleta Secundaria

En cuanto a la tipografía se ha usado una poco corriente pero que le da una perspectiva bastante buena a los títulos y textos de la aplicación.

Esa tipografía es la de Roboto es la tipografía más usada en android studio y elegida para ser usada en la aplicación, es una tipografía limpia y sencilla, muy visible para los usuarios.

Roboto

SUNGLASSES

Self-driving robot ice cream truck

Fudgesicles only 25¢

ICE CREAM

Marshmallows & almonds

#9876543210

Music around the block

Summer heat rising up from the boardwalk

Tipografía

Mock Up

Ya que la prioridad del equipo es la de aportar algo nuevo que se adapte a las necesidades de las personas, antes de la aparición de la idea del proyecto EventSports, algunos miembros del grupo de trabajo preguntaron por redes sociales qué era lo que más echaban en falta para facilitar su día a día. Entre todos, sacaron la conclusión de desarrollar la aplicación EventSports.

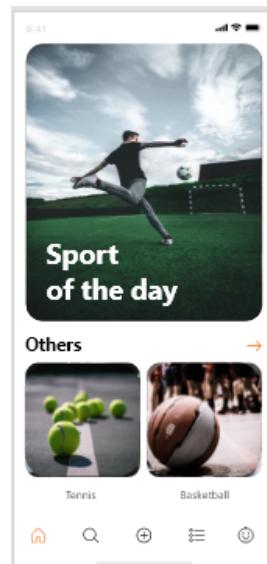
Como buenos desarrolladores, el primer paso era analizar de forma global cómo podría ser el diseño y el orden de las interfaces.

Analizando diversas aplicaciones y plantillas se llegó a establecer un primer prototipo de EventSports.

Para crear el “Mockup” se utilizó la herramienta Adobe XD. Este prototipo no tiene todas las interfaces que aparecen en EventSport ya que su objetivo es mostrar de forma física una idea, y no el proyecto completo. A continuación, se puede visualizar algunas interfaces con su respectivo nombre:



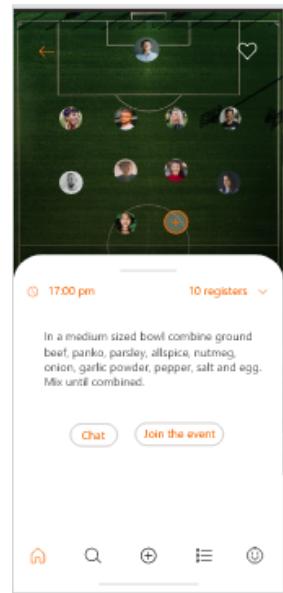
Interfaz 1: Splash Screen



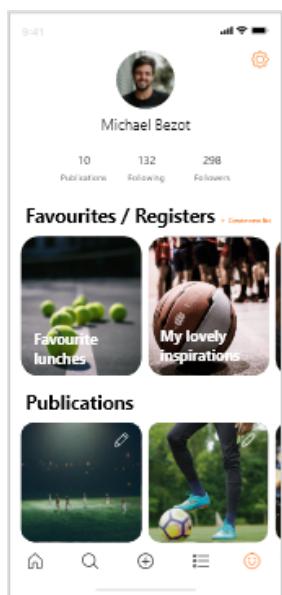
Interfaz 2: Main



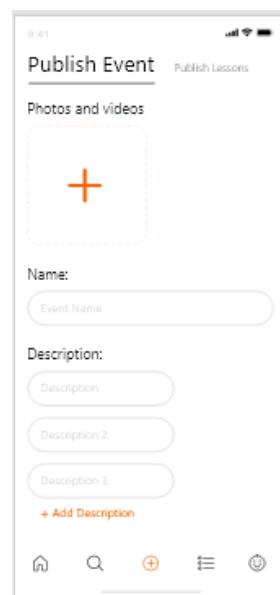
Interfaz 3: Specific Category Page



Interfaz 4: Event page



Interfaz 5: Profile



Interfaz 6: Post Events / Lessons

Después del “Mock Up”, se puso en marcha la creación del logo EventSports. Para que resultara más sencillo, se utilizó la ayuda de páginas web que muestran diversos prototipos. Se eligió uno de ellos y se le dió un toque más personal para adaptarlo de mejor manera al proyecto.

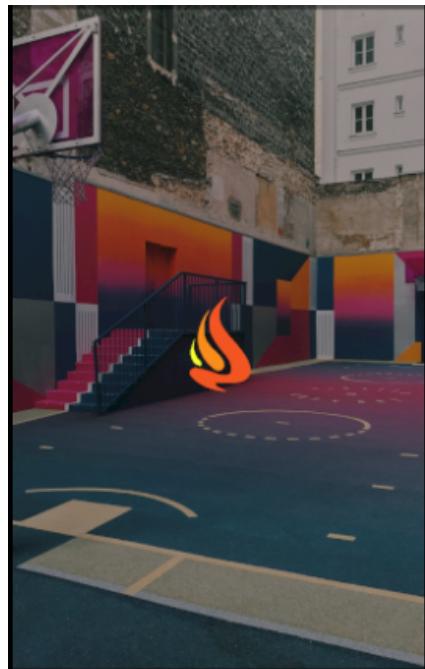
Solo se ha desarrollado un prototipo para el logo pues posee todas las características necesarias para transmitir el poder de EventSports.



Logo: EventSports

Splash Screen

Una buena primera impresión es importante, ya que dependiendo de ello puede hacer que el usuario continúe navegando por la app o no. Por esta razón se ha implementado una interfaz que tiene la función de presentación breve. En ella se muestra una imagen de fondo que representa el tipo de actividades que ofrece la aplicación, el deporte. En la zona central de la interfaz también aparece el logo de EventSports ya que es la identidad del proyecto.



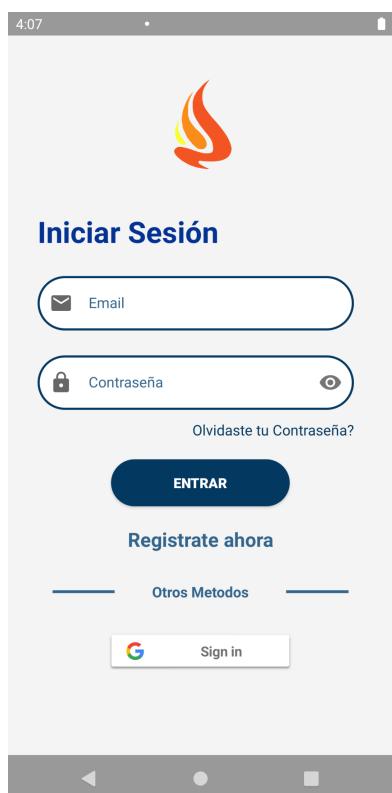
Interfaz "Splash Screen"

Inicio de Sesión (Log In)

En esta interfaz será donde los usuarios que quieran acceder a la aplicación deberán iniciar sesión con el correo y la contraseña, si ya tienen cuenta creada anteriormente, también tendrán la posibilidad de registrarse mediante Google.

Cuando se comenzó a diseñar inicio sesión se tuvo clara la idea de que se quería algo sencillo e intuitivo para un público abierto. Es por esto por lo que esta interfaz tiene un diseño simple sin demasiados colores y remarcando las formas de iniciar sesión.

Esta Interfaz será la segunda que se vea al entrar en la aplicación después del Splash Screen.



Como se puede observar en la imagen esta interfaz resalta las partes importantes de ella, estas partes son:

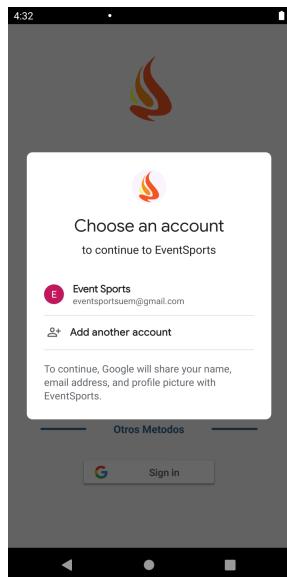
- El logo de EventSports que es lo primero que se encuentra en la interfaz.
- El título de la interfaz en la que se encontraría el usuario.
- Las celdas y el botón de inicio de sesión. Será aquí donde el usuario deberá introducir el correo y la contraseña, posteriormente deberá pulsar el botón para comprobar si la cuenta existe y acceder a la aplicación.

En segundo plano se observa el "Regístrate ahora" donde el usuario podrá hacer click y llegar a la interfaz de Sign Up o Registro para crearse una nueva cuenta (Detallado más abajo).

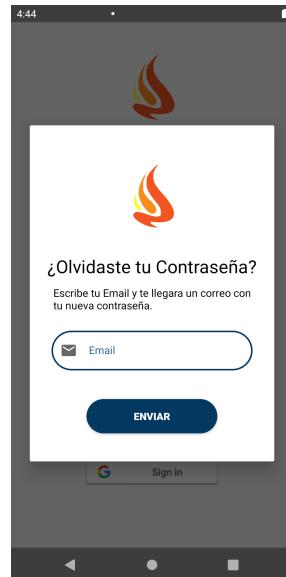
Interfaz Inicio Sesión

También se puede ver otro método de iniciar sesión, el cual es mediante Google como bien se ha mencionado antes (Figura 1). En donde al hacer click se deberá seleccionar el correo electrónico.

Destacar de inicio sesión el "¿Olvidaste tu contraseña?" donde el usuario podrá recibir un correo electrónico para poder recuperar la contraseña (Figura 2).



Inicio con Google



Recuperar Contraseña

Por último, lo más importante de esta interfaz es el pequeño código que usará la aplicación en caso de que ya se haya iniciado sesión. Este pequeño código con no más de 10 líneas de código, comprobará si ya se ha iniciado sesión, si es así recogerá el id del usuario e irá a la interfaz principal de la aplicación.

```
@Override  
protected void onStart() {  
    super.onStart();  
    FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();  
    if (firebaseUser != null){  
        USERUID = firebaseUser.getUid();  
        Intent accessIntent = new Intent(getApplicationContext(), ActivityMain.class);  
        accessIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
        accessIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);  
        startActivity(accessIntent);  
    }  
}
```

Código: Comprobar si ya se ha iniciado sesión

Parte del código de esta Interfaz se encontrará más abajo en la parte de Anexos A.1.

Nueva Cuenta (Sign Up)

En esta interfaz se busco la misma sencillez que en el Inicio sesion y se propuso un fondo relacionado con el deporte, este fondo es una imagen de una persona pegando una patada (Figura 1).

Para crear una nueva cuenta el usuario deberá llenar los campos marcados en la interfaz, siendo estos:

- **Nombre de Usuario.**
- **Email.**
- **Contraseña.**

Posteriormente para acceder a la aplicación se deberá pulsar el botón de "Regístrate" (Figura 2).



Figura 1: Fondo Sign Up

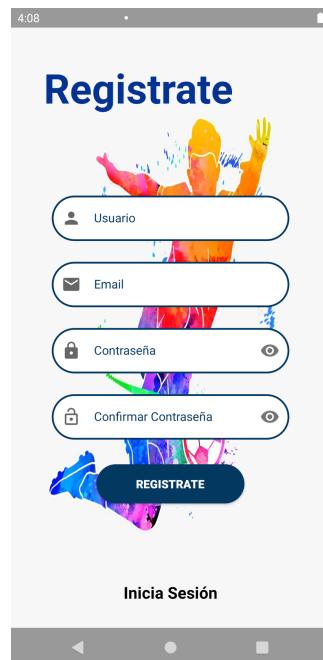


Figura 2: Interfaz Sign Up

Parte del código de esta Interfaz se encontrará más abajo en la parte de Anexos A.1.

Deportes (Main)

Como en cualquier aplicación es primordial establecer una interfaz en la que se muestra la información principal de la misma para que el usuario tenga claro en qué consiste, pues si no es así, le puede resultar complicado el resto de la navegación por las demás interfaces. En EventSports la información principal es la de mostrar los diversos eventos deportivos que han sido generados por los usuarios. Esta interfaz se divide en dos partes:

- **Deporte del día:** se encuentra en la parte superior de la interfaz y tiene el objetivo de mostrar mediante una imagen el deporte que es más practicado por los usuarios.
- **Resto de deportes:** se localiza en la parte inferior de la interfaz y tiene la función de exponer los deportes de los eventos creados en la app.



Interfaz "Main"

Mapa

Esta Interfaz consiste en la visualización de un mapa del mundo. En este mapa irán saliendo unos marcadores con diferentes colores dependiendo de si es un evento o una clase o bien si quieres el usuario quiere saber cual es la ubicación actual.



Estos colores son los mismos que se utilizan en el propio logo de EventSports, el Rojo, el Amarillo y el Naranja.

Cada uno significa una cosa como bien se ha mencionado antes. El Naranja marca los eventos, el Amarillo marca las clases y la ubicacion saldrá en el marcador de color Rojo.

Todo esto se encuentra en la propia interfaz arriba a la izquierda, para recordar a los usuarios que significa cada cosa (Figura 1).

También en esta Interfaz está implementado un "Navigation Bar" en la cual el usuario podrá moverse

entre interfaces de forma intuitiva y rápida.

Este "Navigation Bar" contiene cinco simples botones por los que se puede navegar , explicado con detalle más abajo(Figura 2).

Por último destacar el botón flotante que se encuentra en la parte inferior izquierda, el cual va a ser el encargado de que al clickar busque tu ubicación actual (Figura 3).



Figura 1



Figura 2: "Navigation Bar"

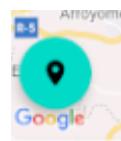


Figura 3

Nuevo Evento o Clase

Esta interfaz es un formulario para crear una nueva clase o evento dependiendo de lo que desee el usuario. Es muy intuitiva y simple.

The screenshot shows a mobile application interface titled "Nuevo Evento o Clase". At the top, there is a placeholder text "Escoja una imagen para su Evento/Clase:" with a camera icon. Below it is a circular profile picture placeholder. The main form area has a header "DATOS". It contains several input fields: "Título", "Dirección" (with a "Nº" button), "Localidad", "Deporte:" (set to "Futbol"), and a large "Descripción" text area. Below these is a "Tipo(Clase/Evento)" dropdown menu with options like "Carrera", "Entrenamiento", "Competencia", and "Otro". At the bottom of the form is a "Nuevo Even..." button. The footer of the screen includes standard navigation icons (home, search, etc.) and a "CREAR" button.

Para empezar en esta interfaz se necesitará añadir todos los datos que pide para poder crear ese evento o clase.

El primer dato que pide es el de la elección de una imagen, esto está más detallado más abajo, consiste en elegir una imagen de forma rápida de tu galería simplemente haciendo click en el ícono de la persona.

Lo segundo que te encuentras en la interfaz son campos a llenar. En estos campos se puede observar que dicen lo que hay que escribir en ellos.

Excepto en dos que se deberá elegir entre varias opciones.

- El primer campo de elección que encuentras es el del deporte a que pertenece esa clase o evento que se desea crear.
- El segundo es la elección de la clase o evento.

Interfaz Nuevo Evento/Clase

Después de esa última elección se encontrará un botón (Figura 1), en el cual pone crear y es el que guardará los datos de ese evento/clase creado con los datos que el usuario ha introducido.

CREAR

Figura 1: Botón Crear

Destacar un pequeño código importante en esta interfaz, el cual consiste en generar la latitud y la longitud de una dirección escrita, es decir, como se puede observar en la interfaz el usuario añade la dirección (la calle/avenida) con su número (si tiene) y la localidad.

Es muy sencillo y muy útil para el proyecto.

```
public static double[] takeCoordenadas(Context context, String direc, String num, String loca) {  
    double[] coords = new double[2];  
    String total = direc + "," + num + "," + loca;  
    Geocoder coder = new Geocoder(context);  
    try {  
        ArrayList<Address> addresses = (ArrayList<Address>)  
            coder.getFromLocationName(total, maxResults: 50);  
        for(Address add : addresses){  
            coords[0] = add.getLatitude();  
            coords[1] = add.getLongitude();  
            //Log.d("Coordenadas", " " + coords[0] + " " + coords[1]);  
        }  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
    return coords;  
}
```

Código: Para generar Latitud y Longitud

El código es muy simple es un método, el cual consiste en pasar por parámetros un contexto para saber de qué interfaz se llama al método, la dirección, el número y la localidad. Este método devolverá un Array de doubles, aquí irá la latitud y la longitud, y consiste en que se introduce al Geocoder (una clase para transformar direcciones en latitud y longitud o viceversa) la dirección y este devuelve datos en un ArrayList de donde se recogerá la latitud y la longitud.

Por último señalar que esta interfaz también contiene el “Navigation Bar” con cinco simples botones por los que se puede navegar , explicado con detalle más abajo.

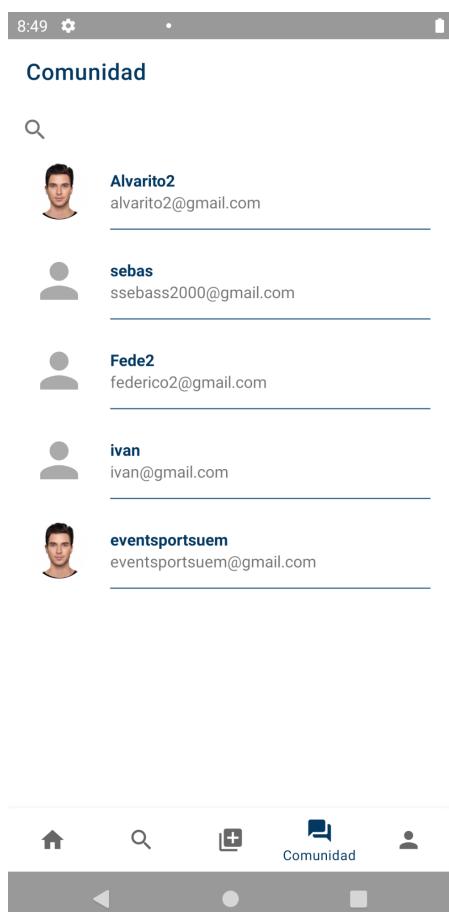
Parte del código de esta Interfaz se encontrará más abajo en la parte de Anexos A.1.

Lista Usuarios

Esta interfaz es una de las más simples del proyecto de fin de ciclo y consiste en ver una lista de toda la “comunidad”, es decir, cargar una lista de todos los usuarios registrados en la aplicación.

Se irán mostrando los usuarios con su foto, nombre de usuario y email correspondientes.

Lo único complejo de esta interfaz es el buscador que contiene, en este se podrá escribir el nombre de usuario que se desee buscar y se buscará de forma automática (Figura 1).



Interfaz Listado Usuarios



Figura 1: Usuario buscado

El código sería el siguiente:

```
public void filter(final String strSearch) {  
    if (strSearch.length() == 0) {  
        mData.clear();  
        mData.addAll(originalItems);  
    }  
    else {  
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {  
            mData.clear();  
            List<User> collect = originalItems.stream()  
                .filter(i -> i.getUser().toLowerCase().contains(strSearch))  
                .collect(Collectors.toList());  
  
            mData.addAll(collect);  
        }  
        else {  
            mData.clear();  
            for (User i : originalItems) {  
                if (i.getUser().toLowerCase().contains(strSearch)) {  
                    mData.add(i);  
                }  
            }  
        }  
    }  
    notifyDataSetChanged();  
}
```

Código: Para filtrar por nombre de usuario

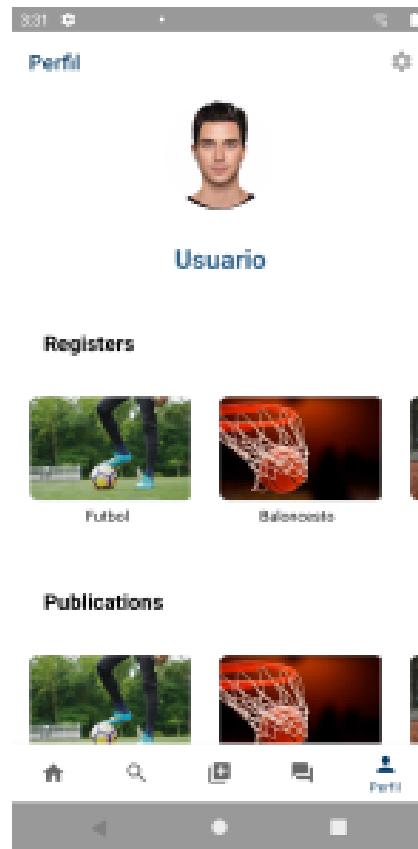
Este método consiste en preguntar si no se está haciendo una búsqueda, si es así que limpie la lista y guarde la lista de firebase en otra creada.

Si se está haciendo una búsqueda se comprobará si tiene más de una solución, si es así devolverá una lista de usuarios y esta se añadirá a la creada para mostrarla. Si resulta que solo devuelve un usuario este será recogido añadido a la lista y mostrado en la interfaz.

Perfil

Es importante poder visualizar de forma genérica todas las interacciones que el usuario ha tenido en la aplicación, en este caso son las publicaciones creadas por él y los eventos en los que se ha registrado. También es imprescindible poder cambiar algunos de los datos personales de la cuenta.

Por ello, es necesario el desarrollo de una interfaz que tenga todas estas características. En este caso, está implementada en EventSports y tiene el de nombre de "Perfil". Su diseño es el siguiente:



Interfaz "Perfil"

Navigation Bar

En todas las interfaces está implementado un “Navigation Bar” en la cual el usuario podrá moverse entre interfaces de forma intuitiva y rápida.

Este “Navigation Bar” contiene cinco simples botones por los que se puede navegar.

- El primero es el “**home**”, que al hacer clic muestra la interfaz de deportes.
- El segundo es el icono “**lupa**”, que pertenece a la interfaz del mapa.
- El tercero es la interfaz “**Añadir evento o clase**”. Este botón llevará al usuario a la interfaz Nuevo Evento o Clase.
- El cuarto icono es “**Listado Usuarios**”. Este muestra la interfaz Listado Usuario.
- El último es el de ajustes de perfil que corresponde a la interfaz “**Perfil**”.

(Todas las interfaces has sido explicadas anteriormente)



“Navigation Bar”

Chat

Esta Interfaz es una de las más complejas porque utiliza dos .xml y dependiendo de quien envíe el mensaje usará uno u otro, es decir, como un chat normal si el usuario habla en el chat le saldrá el mensaje de un color u posición diferente al que le responde (Figura 1 y 2).



Figura 1: Mensaje Receptor

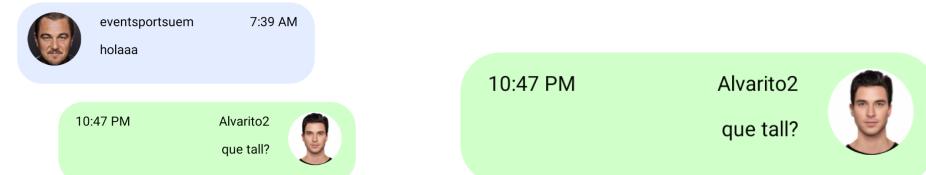


Figura 2: Mensaje Emisor



Interfaz Chat

Para poder realizar ese cambio de .xml se utilizan dos métodos muy simples los cuales comprueban quién está mandando el mensaje y el otro controla cuándo debe poner un .xml u otro (Figura 3 y 4).

```
@Override  
public int getItemViewType(int position) {  
    if (lista.get(position).getUid_user().equals(ActivityLogIn.USERUID)) {  
        return 1;  
    } else {  
        return -1;  
    }  
}
```

Figura 3: Método Comprobación Emisor o Receptor

En el método anterior se empieza comprobando si en la lista de mensajes que se muestran el id de ese mensaje es igual al del usuario conectado si es así devolverá 1 si el id no es igual devolverá -1. Método simple y muy útil.

```
@Override  
public MiAdaptadorChat.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
    if (viewType == 1) {  
        vista = LayoutInflater.from(parent.getContext()).inflate(R.layout.recycler_view_chat2, parent, attachToRoot: false);  
    } else {  
        vista = LayoutInflater.from(parent.getContext()).inflate(R.layout.recycler_view_chat, parent, attachToRoot: false);  
    }  
    return new MiAdaptadorChat.ViewHolder(vista);  
}
```

Figura 4: Método de elección de .xml (Mensaje Emisor/Mensaje Receptor)

En el método anterior se le pasará por parámetro el viewType que es el 1/-1 comentado en el método de la “Figura 3” y se comprobará, si es 1 vista, variable que dirá que .xml que se mostrará, será igual a chat Emisor y si no vista será igual a chat Receptor.

Destacar también otro pequeño método que va a ser el que enseñe el último mensaje del chat (Figura 5).

```
recyclerViewChat.addOnLayoutChangeListener(new View.OnLayoutChangeListener() {  
    @Override  
    public void onLayoutChange(View v, int left, int top, int right, int bottom, int oldLeft, int oldTop, int oldRight,  
        if (bottom < oldBottom) {  
            recyclerViewChat.postDelayed(new Runnable() {  
                @Override  
                public void run() {  
                    recyclerViewChat.smoothScrollToPosition(listAdapter.getItemCount() - 1);  
                }  
            }, delayMillis: 100);  
        }  
    }  
});
```

Figura 5: Método Visualizar último mensaje.

Es un “oyente” del recycler para saber si en la lista que rellena dicho recycler se ha añadido otro mensaje, si es así coloca la vista en él.

Specify Category

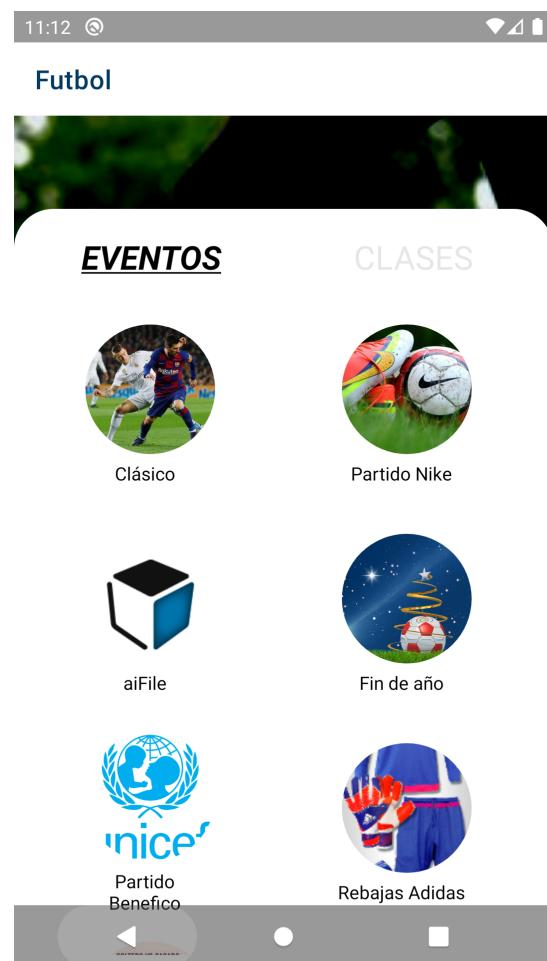
Specify Category es una interfaz en la cual podrás encontrar todos los eventos y clases relacionados con el deporte previamente seleccionado.

Esta interfaz es muy intuitiva y fácil de utilizar debido a la implementación de una parte deslizable que se expande y carga los datos que se deseaba ver, ya sean las clases o los eventos.

Una vez que seleccionan las clases o eventos se expandirá y mostrará el tipo que se ha seleccionado dejando marcado la opción, y en segundo plano la opción restante. A su vez oscurece el fondo para que nos da una sensación de superposición sobre el fondo.



Deslizable Cerrado



Deslizable Abierto

Dentro del deslizable se puede ver todos los eventos o clases disponibles, en esta vista se puede interactuar con ellos pulsando sobre ellos, y nos llevará a la siguiente interfaz donde se verá más información acerca del evento o clase seleccionado y se podrá apuntarse al evento/clase.

Ahora se observará cómo se ha creado la interfaz, esta interfaz se compone de tres layout y dos activities.

La interfaz la se divide en tres partes o layout:

- ❖ **Desplegable**, también llamado “**BottomSheet**”. Es la parte de abajo de la interfaz que se despliega hacia arriba mostrando las diferentes clases o eventos que contiene el deporte seleccionado previamente.
- ❖ **Lista de eventos o clases**, técnicamente llamado como “**RecyclerView**”, es una vista o contenedor que va dentro del desplegable y se encarga de mostrar los datos, para que en caso de tener muchos eventos o clases se pueda deslizar para verlos. En este caso los datos los se cargarán de la base de datos “**Firebase**”.
- ❖ **Interfaz principal**, en esta parte se pondrá el fondo de la interfaz y se juntan las otras dos partes.

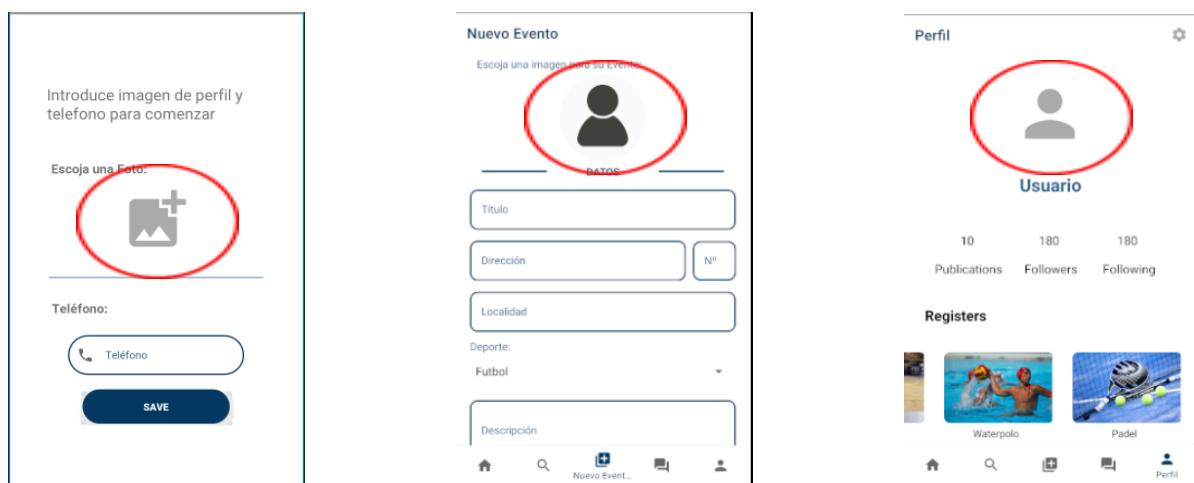
Las dos activities que se utiliza para esta interfaz son:

- ❖ **SpecifyCategory**, que es la encargada de darle funcionalidad a la interfaz, como que se despliegue el contenedor de las clases o se oculte, llamar a clases o a eventos. También se encarga de recoger el deporte del que es llamado para colocar su correspondiente fondo y cabecera, según el deporte seleccionado.
- ❖ **MiAdaptadorSC**, es un activity que sirve para coger los eventos o clases de la base de datos y cargarlos en el RecyclerView explicado anteriormente, o para que se entienda mejor cuál es la actividad encargada de mostrar los datos de la base de datos en el desplegable.

La explicación del código de esta interfaz estará más en el anexo A.2 y A.2.1

Imágenes de Galería y Permisos

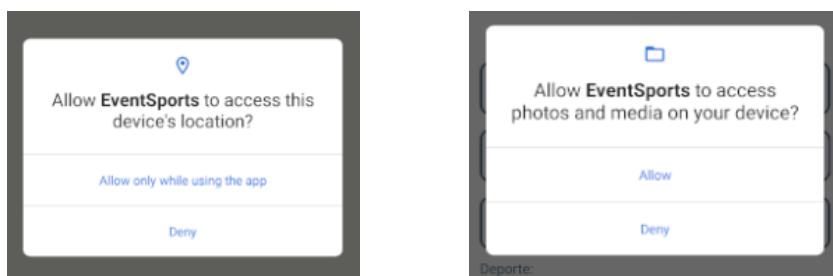
En la aplicación EventSports, es necesario el registro de usuario para poder utilizar todos los servicios que ofrece. Para darle un toque personalizado a la persona registrada, cada usuario tiene la opción de añadir una imagen a su nueva cuenta de EventSports y por supuesto, cambiarla cuando se desee. También es necesario elegir una foto y agregarla cada vez que se genere un tipo de evento en la aplicación.



Añadir imagen en “registro” Añadir imagen en “crear evento” Modificar imagen en “perfil”

Para que esta funcionalidad se haga realidad, hay que implementar cierto código y además, pedir permisos al usuario. Al aceptar estos términos de uso, la aplicación tiene la autorización de utilizar las imágenes de la galería del usuario y así, llevar a cabo la funcionalidad que se ha mencionado anteriormente.

La app utiliza la localización del dispositivo para generar los eventos, por lo que también es necesario los permisos de ubicación para dicha función.



Permisos de “localización”

Permisos para “imágenes”

2.4. Resultados y validación

Respecto al funcionamiento de EventSports, se considera exitoso pues todo ha resultado como se esperaba.

Al iniciar la aplicación se puede observar como los nuevos registros de la app aparecen almacenados en la base de datos y las cuentas ya existentes pueden acceder a EventSports sin ningún tipo de falla.

Al utilizar la “**Navigation Bar**”, las interfaces que aparecen corresponden con su respectivo ícono. El mapa muestra las localizaciones de cada evento creado por los usuarios.

Una de las cosas más importantes respecto a la visualización de las interfaces, es la aparición de las imágenes, pues es la base de todas ellas. Teniendo en cuenta lo anterior, se puede observar que al navegar por la app la base de datos almacena y muestra correctamente las fotos de los diversos deportes.

Esta funcionalidad permite que el usuario tenga una gratificante experiencia al usar EventSports y que pueda disfrutar de la comunidad deportiva de este gran proyecto.

3. CONCLUSIONES

La intención del equipo era la de desarrollar la aplicación lo más parecido al “Mockup” que se explicó en los puntos anteriores, es decir, adaptar el diseño de EventSports a esa plantilla. Pensándolo más detenidamente, se cambió la paleta de colores a un color azul con sus respectivas tonalidades.

Algunas interfaces resultaban algo complejas para el tiempo de entrega del proyecto que se concretó por lo que se modificaron para que el equipo se adapte a la fecha establecida. Esto no impidió conseguir los hitos propuestos, que eran desarrollar el diseño y funcionalidad de un grupo de interfaces necesarias para que la navegación por la aplicación fuese lo más cómoda posible. La funcionalidad era la de poder crear los eventos, la visualización y localización de estos y la comunicación entre los usuarios.

El método utilizado para desarrollar EventSports fue, en primer lugar, realizar el diseño de las interfaces teniendo como prioridad un diseño limpio y sencillo. Despues de finalizar lo anterior, se puso en marcha la creación de la base de datos mediante “Firebase”. Luego se dejó un margen de varios días para perfeccionar lo trabajado antes de la entrega final.

Por la experiencia en otros proyectos, es recomendable optar por este tipo de método de desarrollo, pues evita complicaciones a la hora de establecer la funcionalidad completa en la aplicación, es decir, diseño y base de datos unidos.

3.1. Innovación

En EventSports, se ha tenido como prioridad un diseño que sea atractivo para el usuario y además de sencillo. Algo que es considerado innovador respecto a otros proyectos desarrollados por este equipo es establecer una forma rectangular con bordes redondeados a las imágenes que se muestra en la app consiguiendo lo mencionado anteriormente. También se implementaron los “bottom sheets” para darle una funcionalidad distinta.

También se quiere recalcar que gracias a las experiencias realizando otros proyectos, el código que se ha desarrollado para EventSports se ha creado de manera más optimizada, pues se han encontrado otras formas de realizar lo mismo pero requiriendo menos codificación.

3.2. Trabajo Futuro

Los creadores de este proyecto tienen la intención de continuar el desarrollo de este, por ello, han pensado en ciertos puntos que se podrán llevar a cabo más adelante:

- **Pago mediante la aplicación:** consiste en que los creadores de eventos puedan llevarse una comisión por contribuir en app.
- **Inicio de sesión con facebook y apple:** para que la aplicación llegue a todos los rincones del mundo, es necesario que haya varios métodos de registro.
- **La configuración de los idiomas:** teniendo en cuenta lo anterior, para que cualquier persona pueda entender el mundo del deporte es necesario ofrecerle la opción de configurar el idioma de EventSports a su gusto.
- **Mejorar la funcionalidad de la interfaz ajustes:** añadir nuevas funcionalidades para mejorar la experiencia como usuario.
- **Implementar un chat individual:** Es interesante establecer una comunicación para consultas con el creador del evento o clase.
- **Permitir roles para profesores, creadores de eventos y usuarios:** las funciones de cada rol se pensarán más adelante pero se quiere implementar esto para que cada tipo de usuario se vea importante en la comunidad de EventSports.

4. BIBLIOGRAFÍA Y WEBGRAFÍA

Estas son las referencias por la cual se ha desarrollado este proyecto:

- MATERIALDESIGN. (2021) www.materialdesign.com Fecha de consulta: 17:21, abril 18, 2021 de <https://material.io/components/sheets-bottom>

Se quiere recalcar que la plataforma “Youtube” se ha utilizado numerosas veces para consultar implementaciones de código. Estos son algunos ejemplos:

Sheets-bottom:

- YOUTUBE. (2021) www.youtube.com Fecha de consulta: 19:32, mayo 3, 2021 de <https://www.youtube.com/watch?v=q-jRdJHtrvg>

Imagenes para galería:

- YOUTUBE. (2021) www.youtube.com Fecha de consulta: 18:56, abril 21, 2021 de https://www.youtube.com/watch?v=dA_T1IHxcMg

Formas para imágenes:

- YOUTUBE. (2021) www.youtube.com Fecha de consulta: 15:44, abril 23, 2021 de <https://www.youtube.com/watch?v=jihLJ0oVmGo&t=305s>

En el proyecto se han usado diversas librerías para el uso de código:

→ Para fotos y Recycler: librerías Glide y Recyclerview.

- Ejemplo:

```
implementation 'com.github.bumptech.glide:glide:4.11.0'  
implementation 'androidx.recyclerview:recyclerview:1.2.0'
```

→ Para “Log In”: librerías Loading Button.

- Ejemplo:

```
implementation 'br.com.simplepass:loading-button-android:1.14.0'
```

→ Para registrarse con Google: librerías Google Firebase.

- Ejemplo:

```
implementation 'com.google.firebase:firebase-auth:20.0.4'
```

5. ANEXOS

A.1 Comprobación Formularios

Este anexo explica el método que contiene varias interfaces más o menos en común cambiando algunos campos.

El método es muy simple e intuitivo y se utiliza en la aplicación para la comprobación de si un usuario ha llenado todos los apartados requeridos tanto para iniciar sesión, registrarse o crear un nuevo evento o clase.

Este método es usado en las siguientes interfaces:

- Interfaz Inicio de Sesión
- Nueva Cuenta
- Nuevo Evento/Clase

```
if (TextUtils.isEmpty(correo)){
    email.setError("Enter your email");
    return;
} else if (TextUtils.isEmpty(pwd)) {
    passwd.setError("Enter your password");
    return;
} else if (pwd.length() < 6) {
    passwd.setError("Minimum length of password should be 6");
    return;
} else if (TextUtils.isEmpty(usuario)) {
    username.setError("Enter your username");
    return;
} else if (TextUtils.isEmpty(confirmPwd)) {
    confirmPasswd.setError("Confirm your password");
    return;
} else if (!pwd.equals(confirmPwd)) {
    confirmPasswd.setError("Passwords are different");
    return;
} else if (!isValidEmail(correo)) {
    email.setError("This is not a valid email");
    return;
} else {
    //Lo que quieras que haga tu Formulario
}
```

Método para comprobar Formulario

Cuando el usuario intente hacer alguna de las funcionalidades marcadas anteriormente y se haya dejado un espacio vacío saltará un mensaje diciendo que error ha cometido.

A.2 Layouts SpecifyCategory

Este es el layout principal de la interfaz SpecifyCategory en él hay varios elementos todos ellos comentados con sus funciones.

```
<androidx.constraintlayout.widget.ConstraintLayout
    app:layout_collapseMode="parallax"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:ignore="ExtraText">

    //Barra de arriba donde se muestra el nombre del deporte, el nombre del deporte lo
    //estableceremos desde Java según el deporte seleccionado
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar_specify_category"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/white"
        app:title="Cambiar en java"
        app:titleTextColor="@color/colorAccent"
        tools:ignore="MissingConstraints" />

    //Imagen de fondo desde java estableceremos un fondo u otro dependiendo del deporte
    //seleccionado
    <ImageView
        android:id="@+id/fondo"
        android:layout_width="match_parent"
        android:layout_height="600dp"
        android:scaleType="centerCrop"
        android:src="@drawable/fondocategorialibro"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar_specify_category" />
</androidx.constraintlayout.widget.ConstraintLayout>

    //Incluimos el layout BottomSheet este elemento es el que posteriormente desde java
    //configuraremos para que se despliege hacia arriba.
    <include
        layout="@layout/bottom_sheet"
        android:background="@drawable/drawablebg"
        android:gravity="bottom"
        app:layout_behavior="com.google.android.material.bottomsheet.BottomSheetBehavior" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

activity_specify_category.xml (código)



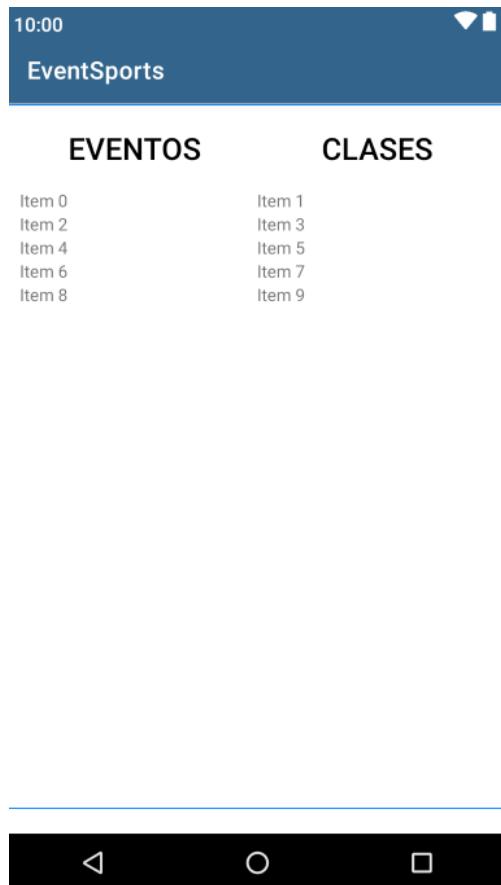
activity_specify_category.xml (diseño)

Ahora se mostrará el layout de **BottomSheet** que estará incluido en la parte de abajo de la interfaz **SpecifyCategory**. Este layout es creado para poder mostrar los eventos o clases en otra interfaz que se sobreponer sobre la de **SpecifyCategory** cuando la se despliega y se rellena con los datos de la base de datos.

En el código solo tendrá 3 elementos a destacar que son 2 botones y un recycler, en la figura de abajo se podrá ver el código y su diseño.

```
<LinearLayout
    android:layout_width="match_parent"
    android:backgroundTint="@color/white"
    app:layout_behavior="com.google.android.material.bottomsheet.BottomSheetBehavior"
    android:layout_height="60dp"
    android:layout_gravity="center_vertical"
    android:orientation="horizontal"
    android:weightSum="4">
    //Botón de Eventos
    <Button
        android:id="@+id/eventos"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="170dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginBottom="10dp"
        android:text="Eventos"
        android:textColor="@color/black"
        android:textSize="25sp"/>
    //Botón de Clases
    <Button
        android:id="@+id/clases"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="170dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginBottom="10dp"
        android:text="Clases"
        android:textColor="#000000"
        android:textSize="25sp"/>
</LinearLayout>
//Añadimos el RecyclerView y le establecemos 2 columnas de ancho, posteriormente
//crearemos una clase en java para adaptar los datos de la base de datos en este
//contenedor
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerEvents"
    android:layout_width="match_parent"
    android:layout_height="500dp"
    app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
    app:spanCount="2"/>
```

BottomSheet.xml (código)



BottomSheet.xml (diseño)

Aquí se mostrará el layout que se ha creado para configurar el diseño del recycler, por cada clase o evento solo se necesita mostrar su foto y nombre, por lo tanto se usará un textView y un shapeable imageView de la librería google material, y estos dos elementos se incluirán dentro de un linearlayout que reducirá su tamaño a tipo "**cardview**".

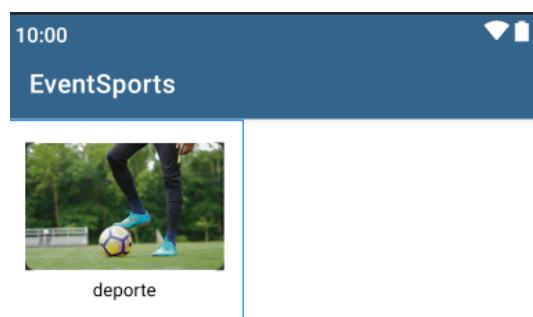
```

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="180dp"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="1dp"
    android:layout_height="160dp">

    //Contenedor tipo cardview para colocar la imagen y el nombre del evento o clase
    <LinearLayout
        android:id="@+id/cardview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        android:padding="2dp">
        //Imagen del evento o clase
        <com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/imageViewDeporte"
            android:layout_width="154dp"
            android:layout_height="99dp"
            android:src="@drawable/bota"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.301"
            app:shapeAppearanceOverlay="@style/RoundedSquare" />
        //Nombre del evento o clase
        <TextView
            android:layout_marginTop="5dp"
            android:id="@+id/textoDeporte"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:text="deporte"
            android:textAlignment="center"
            android:textColor="@color/black"
            app:layout_constraintVertical_bias="0.0" />
    </LinearLayout>

```

vistaRecycler (código)



vistaRecycler (diseño)

A.2.1 Activities SpecifyCategory

En esta sección del anexo se explicará las dos activities que dan funcionalidad a la pantalla de SpecifyCategory, aquí se parte del punto en el que se accede a la interfaz tras seleccionar un deporte en la pantalla del main, y se reciben dos objetos mediante esta acción, uno es el Nombre del deporte y el otro es la Imagen del deporte.

Sabiendo esto verá el código por bloques, cada bloque está comentado para que sirve y cómo funciona.

Primero se mostrará la definición de las variables y posteriormente se verá la utilidad y el uso de cada una de ellas en el código de la aplicación.

```
//Botones clases y eventos
private Button clases;
private Button eventos;
//Booleanos donde estableceremos si estan activos o no, clases y eventos
private boolean clas = false;
private boolean event = false;
//Imagen de fondo y el nombre del deporte seleccionado
private ImageView fondo;
private String titulo="";
//A través de esta variable llamaremos a un método para coger los datos de Firebase
private CollectData.Comunicacion comunicacion = this;
//RecyclerView y adaptador para el RecyclerView
private RecyclerView recyclerView;
private MiAdaptadorSC listaAdapter;
//Toolbar, la barra de arriba
private Toolbar nombre;
//Estas dos variables son para el bottomsheet (desplegable de la parte de abajo)
private LinearLayout linearLayout;
private BottomSheetBehavior bottomSheetBehavoir;
```

Definición de variables

Ahora se ve la parte del código donde se inicializan las variables al principio del método onCreate

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_specify_category);

    //Sacamos el nombre del deporte seleccionado y lo establecemos en los elementos de esta interfaz (fondo y el título del toolbar)
    Intent intent = getIntent();
    titulo = intent.getStringExtra( name: "NombreEvento");
    Bundle extras = getIntent().getExtras();
    if (extras == null) {
        return;
    }
    int res = extras.getInt( key: "resourceInt");
    nombre=findViewById(R.id.toolbar_specify_category);
    nombre.setTitle(titulo);
    recyclerView=findViewById(R.id.recyclerEvents);
    fondo=findViewById(R.id.fondo);
    Bitmap bitmap = intent.getParcelableExtra( name: "Fondo");
    fondo.setImageBitmap(bitmap);

    //Inicializamos variables
    clases = findViewById(R.id.clases);
    eventos = findViewById(R.id.eventos);
    linearLayout= findViewById(R.id.bottomSheet);
    bottomSheetBehavoir=BottomSheetBehavior.from(linearLayout);

    //Establecemos los estilos del fondo y de los textos de clases y eventos subrayados para cuando estén seleccionados que se cambie
    //el estilo y se oscurezca el fondo
    SpannableString classess = new SpannableString( source: "Clases");
    classess.setSpan(new UnderlineSpan(), start: 0, classess.length(), flags: 0);
    SpannableString eventoss = new SpannableString( source: "Eventos");
    eventoss.setSpan(new UnderlineSpan(), start: 0, eventoss.length(), flags: 0);
    Drawable drawable = fondo.getDrawable();
```

Método onCreate (SpecifyCategory.java)

La implementación del BottomSheet o desplegable es largo de código pero muy fácil de entender, aquí abajo se muestra el código comentado y bien explicado

```
//Implementamos el bottomsheet y le establecemos el método onStateChanged que se ejecuta cada vez
//que desplegamos o plegamos el desplegable
bottomSheetBehavoir.setBottomSheetCallback(new BottomSheetBehavior.BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
        //Hacemos un switch según si el estado es Expandido (abierto) o Collapsado (cerrado)
        switch(newState){
            //Si está abierto, oscurece el fondo, hace el recycler visible y activa que
            //el desplegable lo podamos cerrar arrastrando hacia abajo
            case BottomSheetBehavior.STATE_EXPANDED:
                drawable.setColorFilter(Color.BLACK,PorterDuff.Mode.OVERLAY);
                fondo.setImageDrawable(drawable);
                recyclerView.setVisibility(View.VISIBLE);
                bottomSheetBehavoir.setDraggable(true);
                break;
            //Si está cerrado, reestablece el color del fondo, vuelve invisible el recycler,
            //pone los textos de los botones con estilo normal, y desactiva que se pueda
            //desplegar el bottomsheet arrastrando
            case BottomSheetBehavior.STATE_COLLAPSED:
                drawable.clearColorFilter();
                clas=false;
                recyclerView.setVisibility(View.INVISIBLE);
                clases.setText("Clases");
                clases.setTypeface( tf: null, style: 0);
                clases.setAlpha(1f);
                event=false;
                eventos.setText("Eventos");
                eventos.setAlpha(1f);
                eventos.setTypeface( tf: null, style: 0);
                bottomSheetBehavoir.setDraggable(false);
                break;
        }
        //Llama al método RecogerEventos que es el encargado de sacar los datos de Firebase
        CollectData.recogerEventos(comunicacion);
    }
})
```

Implementación del BottomSheet (SpecifyCategory.java)

Después de establecer funcionalidad al BottomSheet se tiene que configurar los activadores de los botones. El código de los botones es similar así que sólo se muestra el del botón de Clases.

```

//Establecemos la funcionalidad al clicar el botón de clases
clases.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Cada vez que pulsamos en el botón comprueba si el botón ha sido pulsado o no
        //Esta comprobación la hacemos para ver está pulsado o no, por defecto empiezan
        //sin pulsar.
        //Si esta pulsado le cambiamos el estilo para marcarlo, sino le ponemos normal
        if (!clas){
            clas=true;
            clases.setText(clases);
            eventos.setAlpha(0.1f);
            clases.setTypeface(tf, style: 3);
        }else{
            clas=false;
            clases.setText("Clases");
            eventos.setAlpha(1f);
            clases.setTypeface(tf, style: 0);
        }
        //Ahora comprobamos si esta pulsado, expandiremos el bottomsheet, si hemos
        //desactivado el botón cerramos el bottomsheet
        if(bottomSheetBehavoir.getState()!=BottomSheetBehavior.STATE_EXPANDED){
            bottomSheetBehavoir.setState(BottomSheetBehavior.STATE_EXPANDED);
        }else{
            bottomSheetBehavoir.setState(4);
        }
    }
});
```

funcionalidad botones eventos y clases (SpecifyCategory.java)

Por último se ha implementado el método mandar Eventos para recoger los datos del Firebase, y su funcionamiento consiste en coger todos los datos de firebase y filtrar según si son de tipo clases o eventos, y después comprueba si son del deporte seleccionado o no, y devuelve la lista de datos según su deporte y tipo.

```

@Override
public void mandarEventos(List<Evento> eventos) {
    //Establecemos el array que vamos a utilizar despues para mostrar
    //Y dos arrays, uno para clases y otro para eventos
    listAdapter=new MiAdaptadorSC( evento: null);
    List<Evento> ents = new ArrayList<>();
    List<Evento> clases = new ArrayList<>();

    //Aqui rellenamos cada array segun el tipo que sean los datos si son clases o eventos
    for (Evento e:eventos) {
        if(e.getDeporte().equals(titulo)) {
            if (e.getTipo().equals("Evento")) {
                ents.add(e);
            } else {
                clases.add(e);
            }
        }
    }
    //Ahora depende del boton clicado si es eventos o clases establecemos el array que utilizaremos
    //para el recycler
    if(event) {
        listAdapter = new MiAdaptadorSC(ents);
    }else{
        listAdapter = new MiAdaptadorSC(clases);
    }

    //Ahora asignamos el array a nuestro recycler, y le establecemos 2 columnas
    RecyclerView.LayoutManager mLayoutManager = new GridLayoutManager( context: this, spanCount: 2);
    recyclerView.setLayoutManager(mLayoutManager);
    recyclerView.setAdapter(listAdapter);
}

```

metodo mandarEventos (SpecifyCategory.java)

Para terminar la sección de “Activities SpecifyCategory” mostraremos la clase que utiliza como adaptador para el Recycler y como extrae los datos de la base de datos firebase a la interfaz.

```

//Extendemos la clase de RecyclerView.Adapter
public class MiAdaptadorSC extends RecyclerView.Adapter<MiAdaptadorSC.ViewHolder>{
    //Atributos de la clase, Array de Eventos y la View
    private List<Evento> evento;
    View view;

    //Constructor de la clase con el atributo del array de Eventos
    public MiAdaptadorSC(List<Evento> evento) { this.evento = evento; }

    @NonNull
    @Override
    public MiAdaptadorSC.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        //Asignamos el layout que hemos diseñado para colocar los datos dentro del Recycler
        view = LayoutInflater.from(parent.getContext()).inflate(R.layout.vista_imagenes_main, parent, attachToRoot: false);
        return new MiAdaptadorSC.ViewHolder(view);
    }
}

```

creación de la clase (MiAdaptadorSC.java)

Ahora tenemos que implementar los métodos de la clase que se ha extendido

```

//Implementamos los métodos de la clase que hemos extendido
@Override
public void onBindViewHolder(@NonNull MiAdaptadorSC.ViewHolder holder, int position) {
    //Creamos el evento seleccionado
    Evento e = evento.get(position);
    //Le colocamos la imagen
    Glide.with(view) RequestManager
        .load(Uri.parse("https://firebasestorage.googleapis.com/v0/b/stellar-operand-305716.appspot.com/o/FotosEvent%2E"
            + e.getUserid() + "-" + e.getNombre() + ".jpg?alt=media&token=2df21feb-3d9a-4e6a-8e53-770ce46c6740")) RequestManager
        .centerCrop()
        .transition(DrawableTransitionOptions.withCrossFade(300))
        .circleCrop()
        .into(holder.imgDeporte);
    //Le colocamos el texto
    holder.nombreEvento.setText(e.getNombre());
    //Y le agregamos la interactividad para que al clicar le pase el nombre del deporte, nombre del evento y el id del usuario
    holder.cardview.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(view.getContext(), ActivityEvents.class);
            intent.putExtra( name: "Deporte", e.getDeporte());
            intent.putExtra( name: "NombreEvento", e.getNombre());
            intent.putExtra( name: "UserIdEvento", e.getUserid());
            v.getContext().startActivity(intent);
        }
    });
}
public void setItems(List<Evento> items) { evento = items; }
@Override
public int getItemCount() { return evento.size(); }

```

implementación métodos Recycler (MiAdaptadorSC.java)

Ahora creamos la clase ViewHolder y se le asigna los atributos que necesita para adaptar los datos en el Recycler View.

```

//Creamos la clase ViewHolder en la que establecemos los componentes que tenemos en el layout
//para adaptar el recyclerview
public static class ViewHolder extends RecyclerView.ViewHolder{
    //Creamos las variables necesarias
    ImageView imgDeporte;
    TextView nombreEvento;
    LinearLayout cardview;

    ViewHolder(View itemView){
        super(itemView);
        //Asignamos las variables con sus ids del layout
        this.imgDeporte = itemView.findViewById(R.id.imageViewDeporte);
        this.nombreEvento = itemView.findViewById(R.id.textoDeporte);
        this.cardview = itemView.findViewById(R.id.cardview);
    }
}

```

creación de la clase ViewHolder (MiAdaptadorSC.java)