TRABAJO FIN DE CICLO

TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Curso académico 2019/2020

Social Sports

AUTORES

Carlos Díaz Hernán

Laura Patón Campos

Sergio García Pontes

IES GASPAR MELCHOR DE Jovellanos

Profesora coordinadora

Saturnina Castro Cintas

ÍNDICE

Resumen:

Social Sports es una aplicación de móvil orientada a la realización de eventos deportivos reuniendo a una serie de personas con gustos en común para llevarlos a cabo.

Para ello hemos decidido crearla en Android, apoyándonos, por una parte, en una base de datos objeto relacional en el sistema gestor de base de datos Oracle 18c, en la cual veremos los distintos tipos y tablas creados con la aplicación SQL Developer, y por otra, con un Servicio Web RESTful, el cual nos ofrece recursos mediante URIs que se pueden usar para realizar solicitudes y recibir respuestas a través del protocolo HTTP.

Analizaremos el diagrama de casos de uso que posee nuestra interfaz, para la persona que desee utilizar la aplicación, sepa qué actividades deberá realizar para llevar a cabo algún proceso.

Abstract:

Social Sports is a mobile application oriented to the realization of sports events by bringing together a number of people with common tastes to carry them out.

For this we have decided to create it in Android, relying, on the one hand, on a relational object database in the Oracle 18c database management system, in which we will see the different types and tables created with the SQL Developer application, and on the other hand, with a RESTful Web Service, which offers us resources through URIs that can be used to make requests and receive responses through the HTTP protocol.

We will analyze the diagram of use cases that our interface has, for the person that wants to use the application, know what activities he will have to do to carry out some process.

1 PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

1.1 Tema elegido

Hemos elegido realizar una aplicación de eventos deportivos ya que es una demanda considerable que abarca muchos sectores sociales, gracias a la conexión tecnológica vigente y en aumento, en la que nos encontramos.

1.2 Objetivos planteados

Nuestro objetivo principal es elaborar una aplicación que permita brindarles la posibilidad de conocerse y organizar dichos eventos para su disfrute en conjunto y llevarlos a cabo de la forma más eficiente y concisa posible.
¿Somos capaces de unir a personas con gustos deportivos en común gracias a la tecnología?

2 DESARROLLO DE LA APLICACIÓN

2.1 Datos técnicos

2.1.1 Nombre de la aplicación

Social Sports

2.1.2 Plataforma

Android

2.1.3 Lenguaje de programación

Java

2.2 Funcionalidad

2.2.1 Características

- Interfaz amigable, intuitiva y fácil de utilizar, sin demasiados entresijos que puedan dar pie a errores.
- Visibilidad de Android, una de las dos principales plataformas actuales.
- Presenta una parte de seguridad considerable en lo referente a la protección de datos e información sensible del usuario.
- Accesibilidad, cuenta con un sistema de búsqueda sencillo, donde en unos pocos toques puedes localizar lo que desees.

2.2.2 Base de datos

La base de datos es un modelo de datos Objeto Relacional creada con el Sistema gestor de bases de datos Oracle 18c y la aplicación SQL Developer.

El término Base de Datos Objeto Relacional (BDOR), se usa para describir una base de datos que ha evolucionado desde el modelo relacional hasta otra más amplia que incorpora conceptos del paradigma orientado a objetos.

Por tanto, un Sistema de Gestión Objeto-Relacional (SGBDOR) contiene ambas tecnologías: relacional y de objetos.

Dando como resultado las siguientes tablas y tipos:

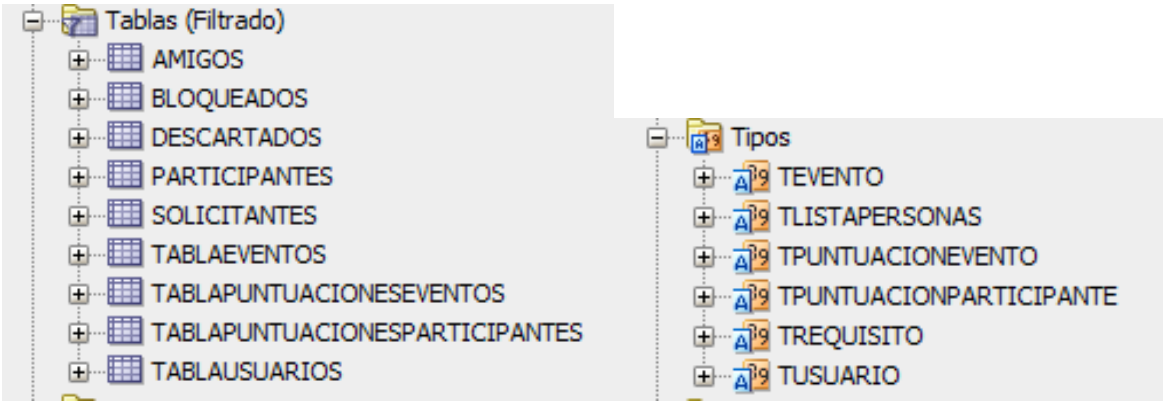


Ilustración 1:Tipos y Tablas de la BBDD

Siguiendo por esa línea de explicación, veremos algunas creaciones en concreto para recalcar una serie de datos.

2.2.2.1 Datos de usuario

Además de los datos personales de usuario y foto de perfil, se guarda información de cuándo se dio de alta en la aplicación, la opinión que tienen de ellos el resto de los usuarios y las listas de amigos que participan en sus eventos y lista de bloqueados:

```
create or replace NONEDITIONABLE TYPE TUSUARIO AS OBJECT(
    emailUsuario varchar2(50),
    passwordUsuario varchar2(128),
    usuarioSalt varchar2(40),
    nombreUsuario varchar2(50),
    apellidosUsuario varchar2(50),
    generoUsuario varchar2(6),
    direccionUsuario varchar2(60),
    fechaNacimientoUsuario date,
    fechaAltaUsuario date,
    reputacionParticipanteUsuario float,
    reputacionOrganizadorUsuario float,
    fotoPerfilUsuario blob,
    isOnlineNow number(1)
)

create or replace NONEDITIONABLE TYPE TLISTAPERSONAS
AS TABLE OF ref Tusuario;

ALTER TYPE TUSUARIO ADD ATTRIBUTE(
    listaAmigos TLISTAPERSONAS,
    listaBloqueados TLISTAPERSONAS
    )CASCADE;
```

Ilustración 2:Estructura del tipo Usuario y ListaPersonas

Si continuamos, vemos cómo creamos el tipo “TlistaPersona” como tabla de referencias de “Tusuario”, es decir, en cada fila de la tabla se va a almacenar el OID (Identificador de objeto) del objeto usuario al que hace referencia. Esto nos permite crear una lista de amigos (usuarios de la aplicación) y tener acceso a toda la información de estos a través de la referencia, manteniendo la integridad de los datos de la Base de datos y evitando la duplicidad de la información.

Una vez definido el Tipo tabla “TlistaPersonas”, añadimos a la estructura del tipo “Tusuario” los atributos “listaAmigos” y “listaBloqueados” que serán, como ya veremos en la creación de tablas, tablas anidadas dentro de la tabla usuarios.

Del atributo “usuarioSalt”, perteneciente al tipo Usuario, destacamos que es donde entran en juego las funciones hash, que son algoritmos que transforman de forma irreversible un bloque de datos en un conjunto de bytes de tamaño fijo.

Para verificar la contraseña de un usuario, se aplica la función hash a la contraseña introducida por el usuario y se comparará el resultado con el hash almacenado en la base de datos.

Aun así, existen ciertos riesgos que se pueden paliar gracias a los “Salt”, que dotan de aleatoriedad dichos hashes, de forma que cuando una misma contraseña es “hasheada” dos veces, los hashes resultantes sean distintos.
Podemos hacer que cada hash sea aleatorio añadiendo una cadena aleatoria a la contraseña antes de aplicar la función de hash. Esta cadena añadida es lo que conocemos por “salt”. Para comprobar que la contraseña es correcta, el “salt” suele ser guardado en campo junto al hash en la base de datos.

2.2.2.2 Datos de evento

Guarda la distinta información de las características de los eventos, como, por ejemplo: quién es el organizador, dónde se va a celebrar, requisitos necesarios para realizarlo, valoración por parte de los participantes que se da al evento una vez finalizado, y las listas de solicitantes, participantes y descartados.

```
create or replace NONEDITIONABLE TYPE TREQUISITO AS OBJECT (
    edadMinima number,
    edadMaxima number,
    requisitoDeGenero varchar2(6),
    reputacionNecesaria float
)

create or replace NONEDITIONABLE TYPE TEVENTO AS OBJECT (
    idEvento varchar2(100),
    organizadorEvento REF TUSUARIO,
    deporte varchar2(30),
    localidad varchar2(30),
    direccion varchar2(50),
    fechaEvento date,
    horaEvento varchar2(5),
    fechaCreacionEvento date,
    maximoParticipantes number,
    instalacionesReservadas number(1),
    costeEvento float,
    precioPorParticipante float,
    comentarios varchar2(300),
    requisitos TREQUISITO,
    terminado number(1),
    listaSolicitantes TLISTAPERSONAS,
    listaDescartados TLISTAPERSONAS,
    listaParticipantes TLISTAPERSONAS
)
```

```
create or replace NONEDITIONABLE TYPE TPUNTUACIONPARTICIPANTE AS OBJECT (
    emailUsuarioEmisor varchar2(50),
    emailUsuarioPuntuado varchar2(50),
    idEventoFinalizado varchar2(100),
    calificacion float
)

create or replace NONEDITIONABLE TYPE TPUNTUACIONEVENTO AS OBJECT (
    emailUsuarioEmisor varchar2(50),
    idEventoFinalizado varchar2(100),
    calificacion float
)
```

Ilustración 3:Estructura del tipo Evento y Puntuaciones

Primero creamos “Trequisitos”, que es un tipo objeto que almacena la información que el usuario desee filtrar para la creación del evento.

Finalmente, los dos últimos tipos, “TpuntuacionParticipante” y “TpuntuacionEvento” serán imprescindibles para la creación de tablas que servirán para almacenar la puntuación que los participantes otorgarán tanto al evento como al resto de participantes.

2.2.2.3 Creación de tablas

Para nuestra Base de datos vamos a crear 4 tablas:

La tabla de usuarios “TablaUsuarios”, que almacenará objetos del tipo Tusuario, definidos anteriormente, donde el atributo emailUsuario será clave primaria y los atributos listaAmigos y listaBloqueados serán tablas anidadas a la tabla TablaUsuarios, cuya información se almacenará en las tablas amigos y bloqueados respectivamente.

La tabla de eventos “TablaEventos”, almacenará objetos del tipo Tevento, donde el atributo idEvento será la clave primaria y los atributos listaSolicitantes, listaDescartados y listaParticipantes serán sus tablas anidadas.

La tabla puntuaciones sobre participantes “TablaPuntuacionesParticipantes” que almacenará objetos de tipo TpuntuacionParticipante, donde la clave primaria está compuesta por los atributos: emailUsuario, emailUsuarioPuntuado e idEventoFinalizado que son a su vez cada uno de ellos claves foráneas a las tablas TablaUsuario y TablaEventos respectivamente.

La tabla puntuaciones sobre eventos “TablaPuntuacionesEventos”, almacena objetos de tipo TpuntuacionEventos, donde la clave primaria está compuesta por los atributos: emailUsuarioEmisor e idEventoFinalizado que de nuevo son claves foráneas a las tablas TablaUsuario y TablaEventos.

```
create table TablaUsuarios of Tusuario
(emailUsuario primary key)
nested table listaAmigos store as amigos
nested table listaBloqueados store as bloqueados;


create table TablaEventos of Tevento
(idEvento primary key)
NESTED TABLE listaSolicitantes STORE AS solicitantes
NESTED TABLE listaDescartados STORE AS descartados
NESTED TABLE listaParticipantes STORE AS participantes;


create table TablaPuntuacionesParticipantes of TpuntuacionParticipante(
PRIMARY KEY(emailUsuarioEmisor, emailUsuarioPuntuado, idEventoFinalizado),
FOREIGN KEY(emailUsuarioEmisor) REFERENCES TablaUsuarios(emailUsuario),
FOREIGN KEY(emailUsuarioPuntuado) REFERENCES TablaUsuarios(emailUsuario),
FOREIGN KEY(idEventoFinalizado) REFERENCES TablaEventos(idEvento)
);


create table TablaPuntuacionesEventos of TpuntuacionEvento(
PRIMARY KEY(emailUsuarioEmisor, idEventoFinalizado),
FOREIGN KEY(emailUsuarioEmisor) REFERENCES TablaUsuarios(emailUsuario),
FOREIGN KEY(idEventoFinalizado) REFERENCES TablaEventos(idEvento)
```

Ilustración 4:creación de Tablas de la BBDD

2.2.2.4 Diagrama de Base de datos Objeto-Relacional

Aquí veremos las tablas y relaciones en su conjunto. Detallando para cada tabla: los nombres y tipos de sus atributos, claves primarias y foráneas en los casos de las tablas TpuntuacionesEventos y TpuntuacionesParticipantes.

Los atributos que aparecen en Unknown(desconocido), son las tablas anidadas de tipo TlistaPersonas, tablas de referencias como listaAmigos, listaBloqueados...

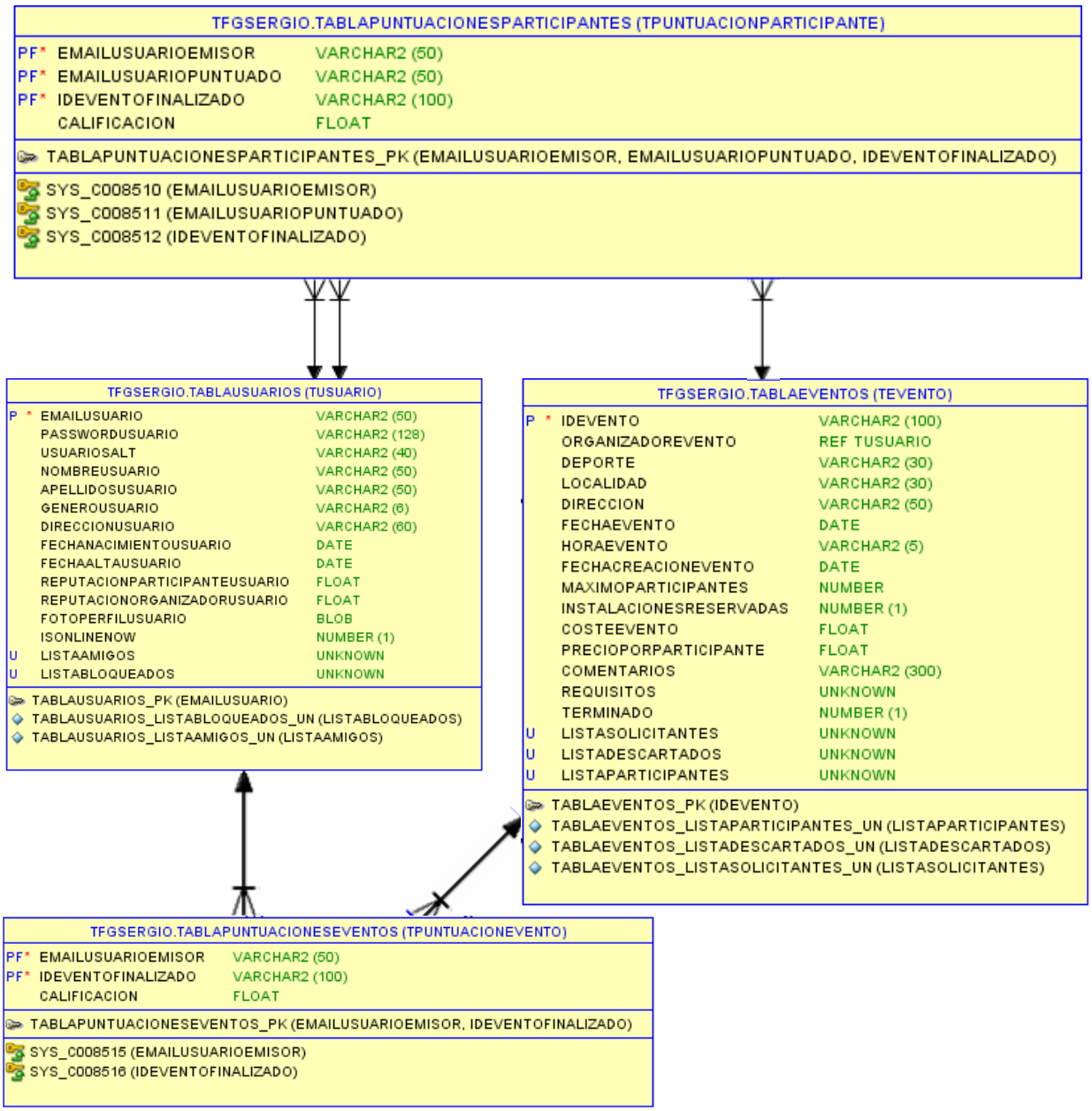


Ilustración 5:Estructura de la BBDD

2.3 Usabilidad

2.3.1 Diseño de interfaces

En este apartado explicaremos las diferentes pantallas con las que el usuario podrá interactuar dentro de la aplicación para utilizar sus diferentes funcionalidades.

Ilustración 6: Pantalla de Inicio de Sesión

Ilustración 7:Pantalla de Bienvenida

Lo primero que nos encontraremos tras abrir la aplicación, será la pantalla de inicio de sesión y registro del usuario, que les permitirá acceder a la aplicación mediante el correo electrónico y una contraseña. Una vez introducidas correctamente las credenciales, nos aparecerá la pantalla de bienvenida de inicio, en la que se muestra un carrusel de imágenes con nuestros próximos eventos en caso de tener eventos pendientes y nuestra puntuación actual como participante (existe la alternativa de que, si el usuario aún no ha indicado su género, ni su nombre y apellidos, ni su fecha de nacimiento, se abra primero la pantalla de configuración de usuario para invitarle a completar esta información). También podremos ver el menú inferior, con el cual podrá acceder a las diferentes funcionalidades de la aplicación: Crear evento, Buscar eventos, Mis eventos y Perfil.

La funcionalidad de “Crear un nuevo Evento”, que podemos seleccionamos en el menú inferior, nos ofrecerá las siguientes opciones de interacción:

Ilustración 9: Crear evento (Datos obligatorios)

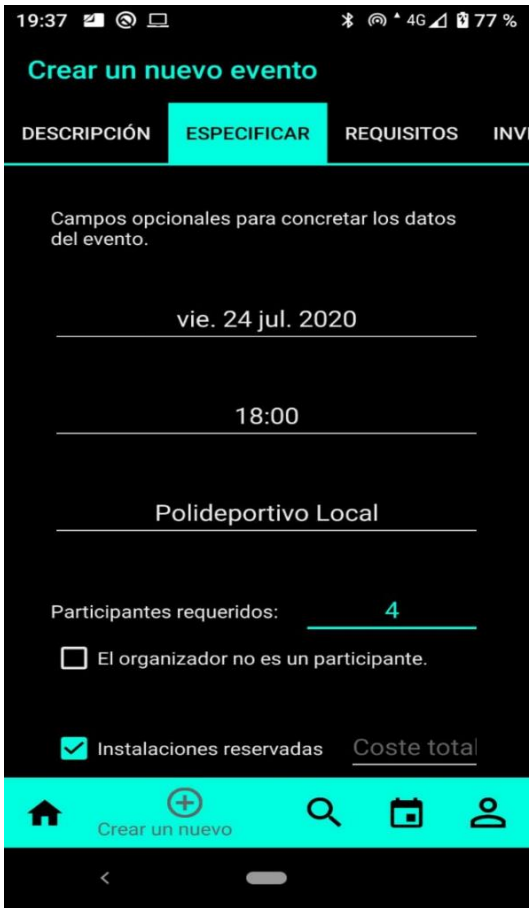


Ilustración 8: Crear evento (Datos específicos)

Cuatro pestañas: Descripción, especificar, requisitos e invitar. Siendo la primera obligatoria para crear el evento, en este caso seleccionando el nombre del deporte que se desea practicar y la localidad donde tendrá lugar el evento, las otras tres nos ofrecen la posibilidad de detallar mejor la información, como, por ejemplo: en el apartado especificar, determinar la fecha, hora, dirección, los participantes requeridos, si el organizador también participa en el evento, si se han reservado las instalaciones para la práctica deportiva, el coste que haya podido suponerle al organizador, si hay que aportar una cantidad por participante, y unos comentarios finales a criterio del organizador del evento.

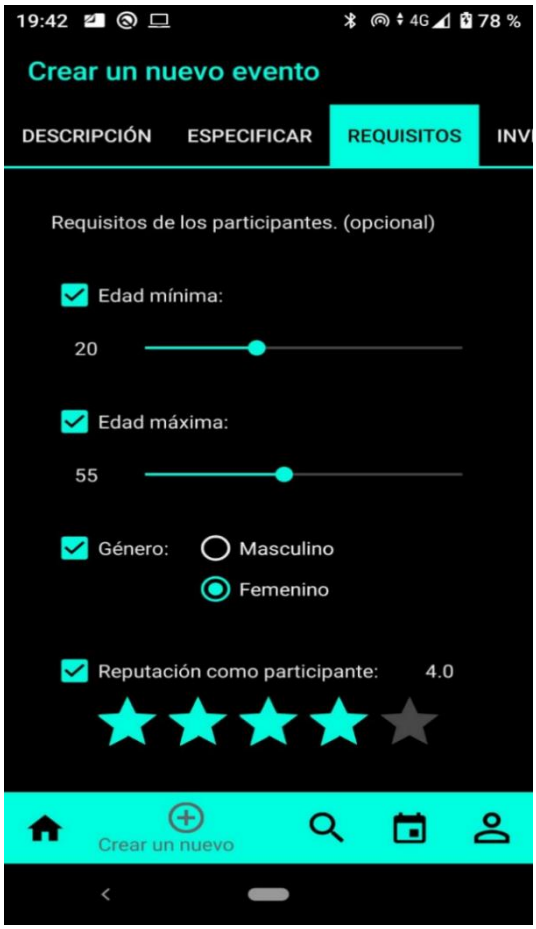


Ilustración 11: Crear evento (Requisitos)

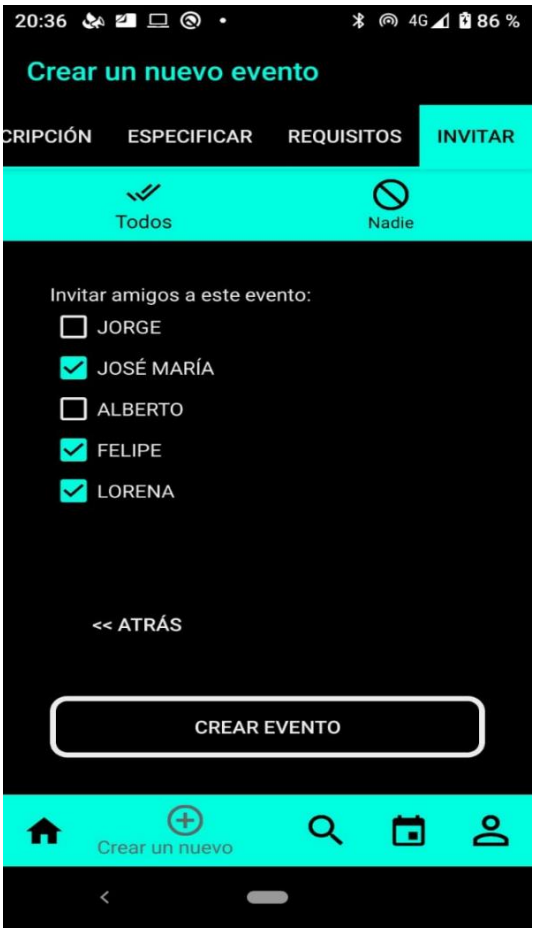


Ilustración 10: Crear evento (Invitar amigos)

En el apartado requisitos podremos determinar si establecer un mínimo y/o un máximo de edad, el género de los participantes y la reputación mínima necesaria para poder participar en el evento.

En el apartado invitar, podremos seleccionar a los usuarios de nuestra lista de amigos que queramos invitar al evento.

En cualquiera de estas cuatro pestañas pertenecientes a la opción de crear un nuevo evento, se nos mostrará el botón de “Crear Evento”, el cual podremos pulsar independientemente de la pestaña en la que nos encontremos.

Accediendo a la funcionalidad de “Buscar Evento”, mediante el menú inferior, nos encontraremos con la siguiente pantalla en la cual podremos cumplimentar cierta información por la cual queremos realizar el filtrado en la búsqueda de eventos.

Ilustración 13: Buscar evento (Filtros)

Ilustración 12: Buscar evento (Resultados)

Los filtros de búsqueda disponibles son los siguientes:

- Deporte, para buscar eventos relacionados con un deporte en concreto.
- Ubicación, para buscar eventos en una Localidad determinada.
- Fecha del evento, para buscar los eventos que hay disponibles en un día determinado.
- Hora del evento, para buscar eventos a una hora del día determinada.
- Instalaciones reservadas, para buscar eventos que ya dispongan de la reserva de instalaciones necesarias para la práctica del deporte.
- Reputación del organizador, seleccionando esta opción podremos elegir el mínimo de reputación que debe tener el organizador del evento, para que se nos muestren sus eventos en la búsqueda.

Al pulsar en la opción de “buscar”, obtendremos la lista de eventos disponibles en relación a los filtros previamente introducidos.

Accediendo a la funcionalidad de “Mis eventos”, mediante el menú inferior de la aplicación, tendremos acceso a dos pestañas en las cuales se mostrarán eventos en los que figuramos como participantes y/o organizador del evento.

Ilustración 15: Mis eventos Pendientes

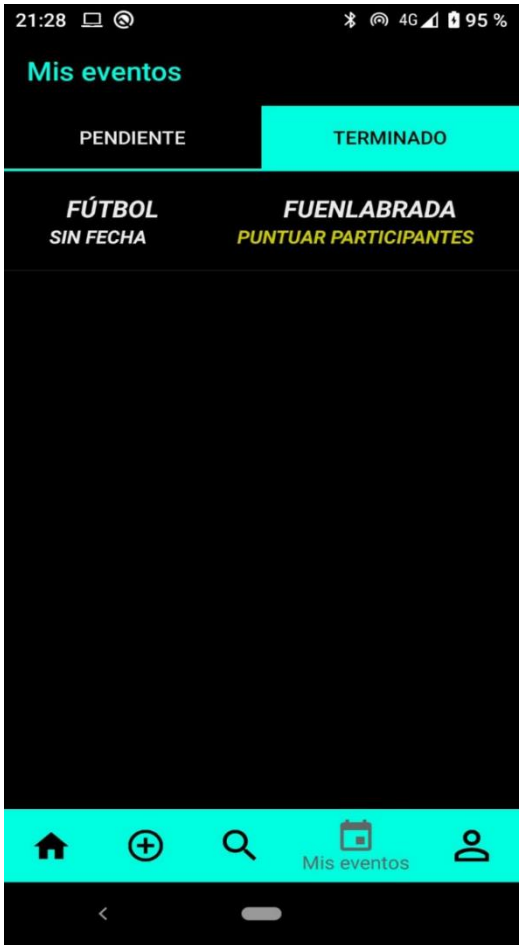


Ilustración 14: Mis eventos Finalizados

La pestaña “Pendientes”, nos mostrará una lista de eventos que aún no se han celebrado, y en los que estamos incluidos como participantes del evento y/o como organizadores de este. La pestaña “Terminados”, nos mostrará una lista de eventos ya finalizados, en los cuales participamos u organizamos.

El usuario podrá interactuar con los eventos que se muestran en las diferentes listas de eventos de la aplicación, ya sea tanto al realizar una búsqueda de eventos, como al acceder a la lista de “Mis Eventos”. Al pulsar sobre un evento concreto de la lista de eventos, podremos acceder a las pantallas de configuración de éste, y la interacción que podremos tener dentro de estas pantallas referentes al evento, dependerá de si somos el organizador, si somos un participante o solicitante, si no estamos vinculados con el evento de ninguna forma, o si el evento ha finalizado.

Ilustración 16: Ajustes del Evento

Ilustración 17: Solicitantes del Evento

Siendo el organizador del evento, podremos modificar los ajustes del evento: fecha, hora, dirección, participantes requeridos, coste del evento, precio individual, etc.

También podremos acceder a la lista de solicitudes, para poder aceptarlas/rechazarlas.

Si no somos el organizador del evento, también podremos acceder a toda esta información, pero no podremos modificar nada, y tan solo tendremos la opción de agregar como amigos a los usuarios que aparecen en la lista de solicitantes y de participantes.

También podremos acceder a la lista de participantes del evento, con los cuales podremos interactuar pulsando sobre uno en concreto y seleccionando una de las opciones que aparecen para el usuario seleccionado.

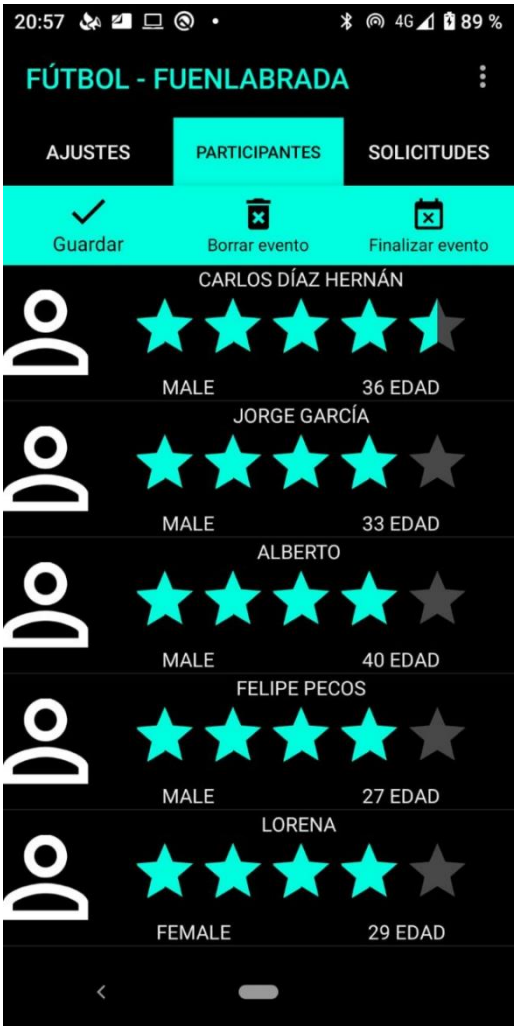


Ilustración 19: Participantes del Evento

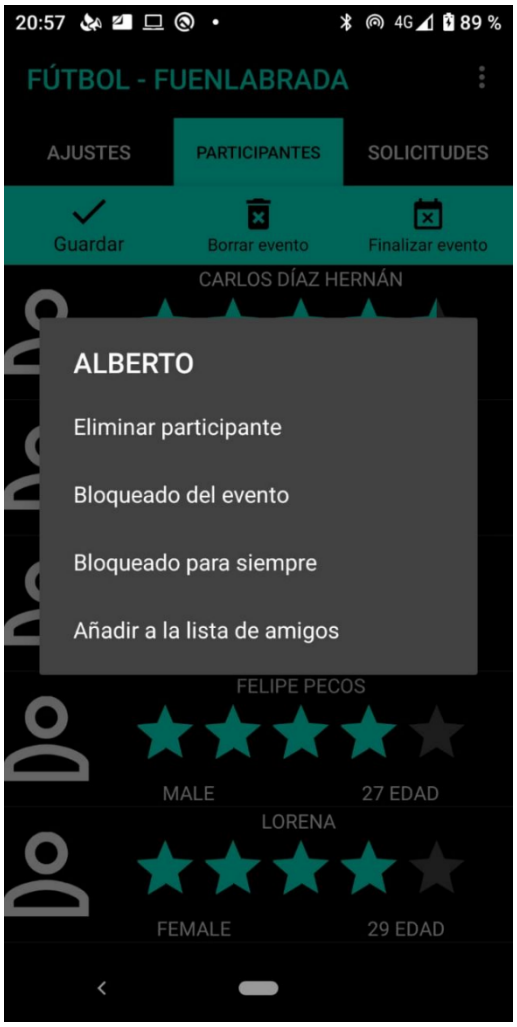


Ilustración 18: Opciones sobre Participantes

Si somos el organizador del evento, tendremos las opciones de: eliminar un participante, bloquearlo en ese evento, bloquearlo permanentemente para todos tus eventos, o añadirlo a nuestra lista de amigos.

Si no fuéramos el organizador del evento tan solo tendríamos la opción de agregar a nuestra lista de amigos a otro usuario.

Si accedemos a la pantalla de configuración de un evento que hemos seleccionado, pero no somos el organizador de dicho evento, las opciones disponibles para interactuar con el evento, cambian.

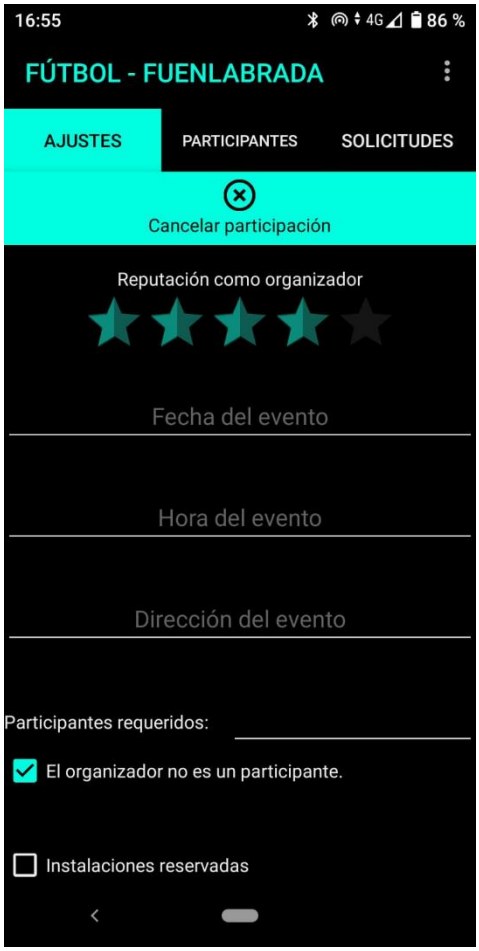


Ilustración 21: Cancelar participación Evento

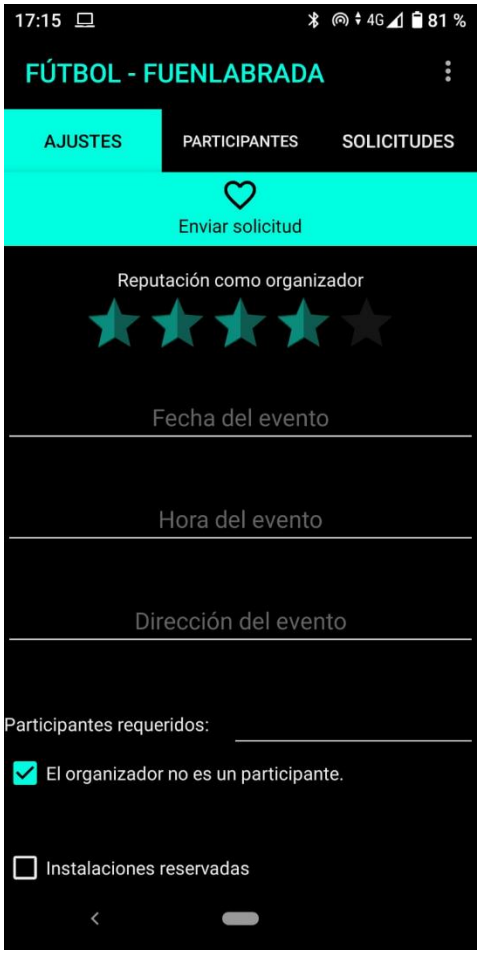


Ilustración 20: Enviar solicitud Evento

Si ya estamos registrados en el evento como participante o como solicitante, se nos dará la opción de “cancelar participación”.

Si aún no estamos vinculados con dicho evento de ninguna manera, se nos dará la opción de enviar una solicitud para participar en el evento.

Si accedemos a un evento determinado de nuestra lista de eventos que ya han finalizado, ya no nos aparecerá ninguna pantalla con la configuración del evento. Por el contrario, nos aparecerá una pantalla en la cual podremos puntuar a los diferentes participantes del evento, y si no somos el organizador, también podremos puntuar el evento en si.

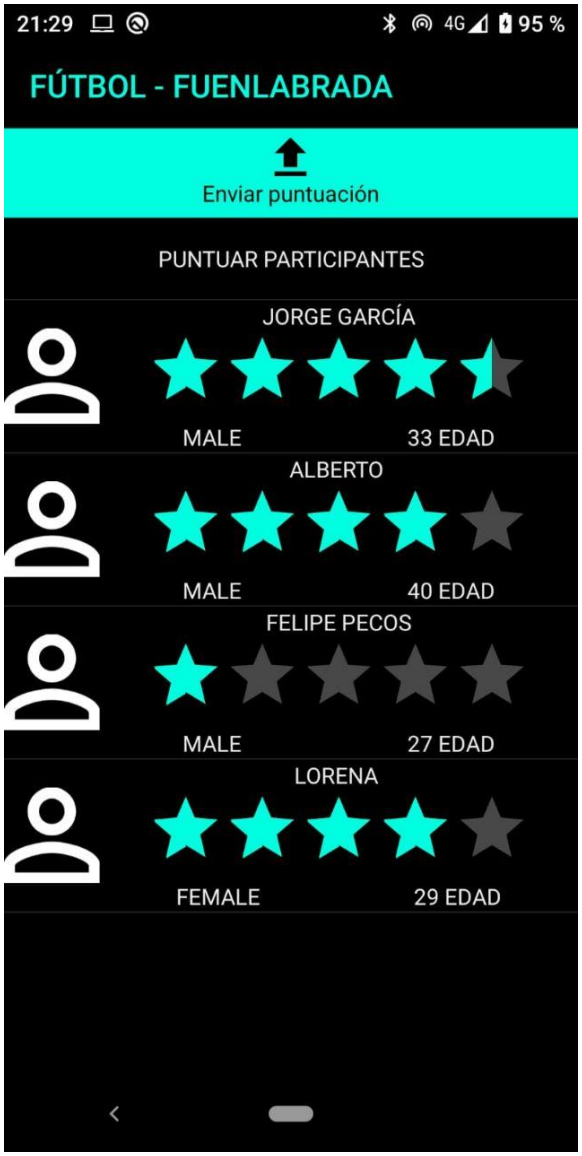


Ilustración 22: Puntuar Participantes

En la última funcionalidad del menú inferior de la aplicación, “Usuario”, podremos detallar la información de nuestro usuario introduciendo nuestros datos personales.

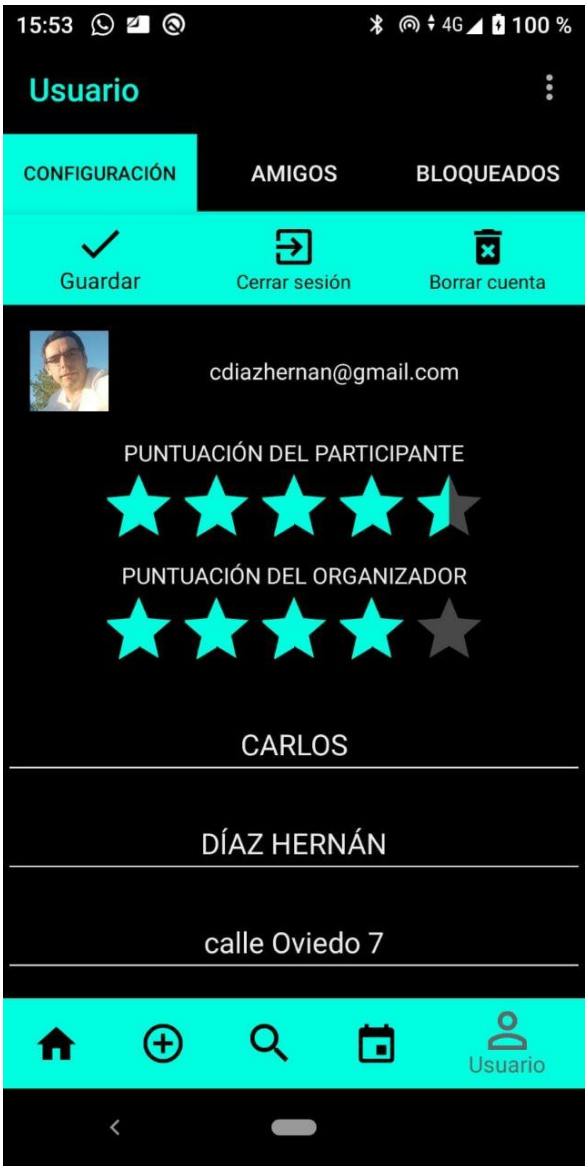


Ilustración 23: Configuración del usuario

Podremos seleccionar una foto de perfil, ver nuestra puntuación total, tanto como participante como organizador, indicar nuestro nombre, apellidos, dirección, fecha de nacimiento, género, y cambiar nuestra contraseña.

No es obligatorio la introducción de estos datos personales por parte del usuario, pero ha de tener en cuenta, que muchos eventos pueden disponer de un requisito de edad o de género, por lo que si el usuario no facilita estos datos no podrá tener acceso a dichos eventos.

En el apartado “Amigos” observamos:

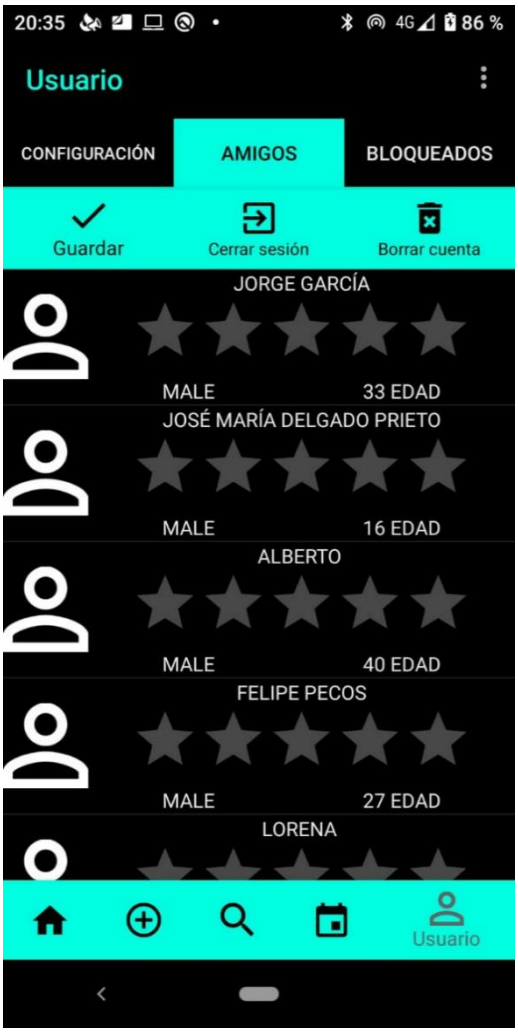


Ilustración 25: Amigos del usuario

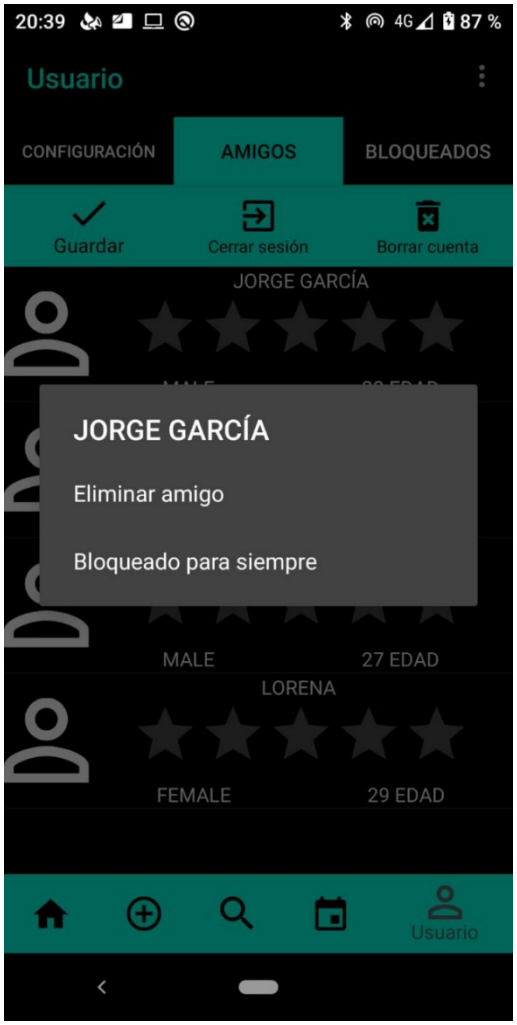


Ilustración 24: Opciones sobre los amigos

La lista de personas que hemos agregado como tal, con toda su información detallada y la posibilidad de eliminarlos o bloquearlos de la lista si se necesitase.

De igual modo, en el apartado “Bloqueados”.

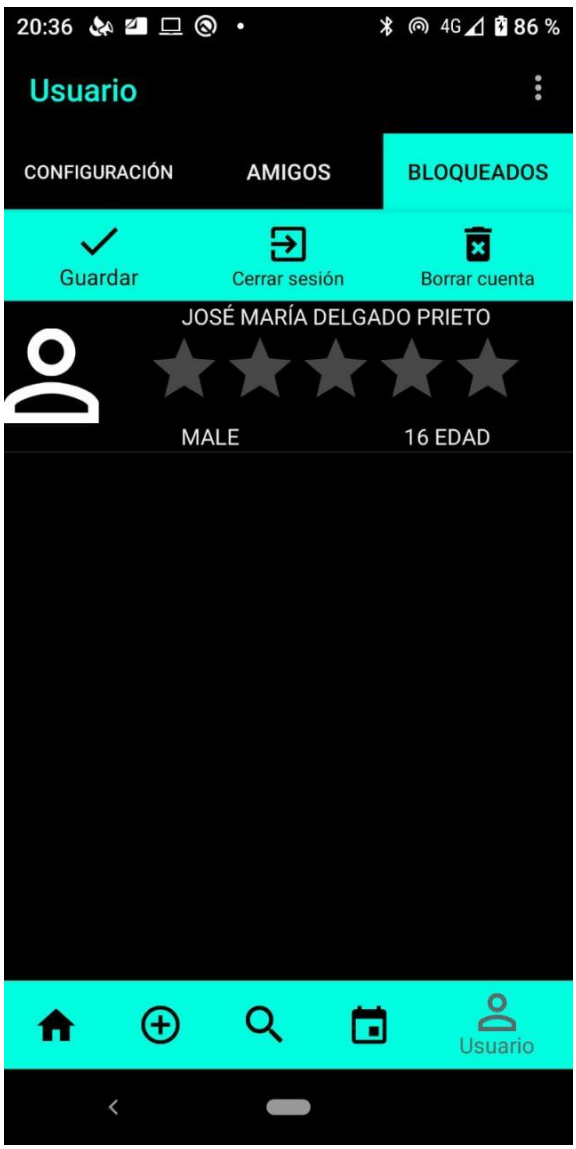


Ilustración 27: Bloqueados por el usuario

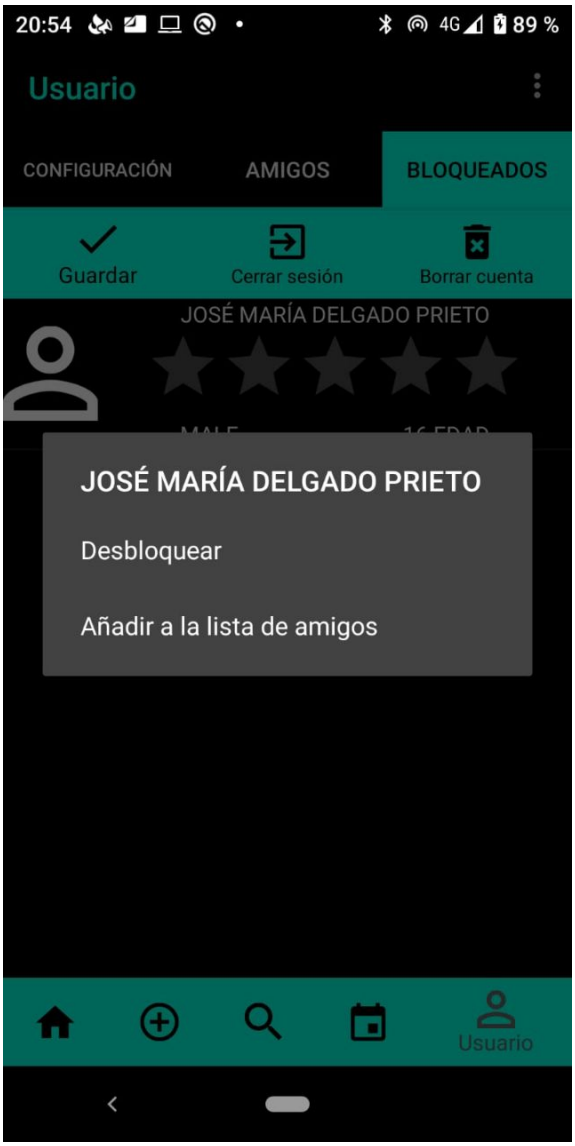


Ilustración 26: Opciones sobre bloqueados

Veremos qué usuarios tenemos en la lista de bloqueados y les daremos la posibilidad a los usuarios de poder desbloquearles o directamente llevarlos a la lista de amigos.

Una vez gestionado el perfil del usuario podemos darle a “Guardar cambios” arriba a la izquierda y la posibilidad de cerrar sesión o de borrar la cuenta.

La aplicación estará disponible tanto en inglés, como en español.

Aquí podemos ver un pequeño extracto de los Strings utilizados en ambos idiomas:

```
<!-- Strings related to login -->
<string name="prompt_email">Email</string>
<string name="prompt_password">Password</string>
<string name="action_sign_in">Sign in</string>
<string name="action_sign_in_short">Login</string>
<string name="action_register">Register</string>
<string name="usuario_incompleto">You have to enter your email for</string>
<string name="password_incompleto">You have to enter any password for</string>
<string name="login_datos_incorrectos">Incorrect login data entered.</string>
<string name="login_usuario_existe">A user with this email already exists</string>
<string name="login_creado_nuevo_usuario">The new user has been created</string>
<string name="login_error_nuevo_usuario">The new user could not be created.</string>

<string name="prompt_correo">Correo electrónico</string>
<string name="prompt_contra">Contraseñaa</string>
<string name="action_inicio">Iniciar sesión</string>
<string name="action_acceso">Acceder</string>
<string name="action_registro">Registrarse</string>
<string name="usuario_incomple">Debes introducir tu correo electrónico para</string>
<string name="password_incomple">Debes introducir una contraseña para</string>
<string name="login_datos_incorrect">Datos de acceso incorrectos</string>
<string name="login_usuario_exis">Hay otro usuario utilizando este correo electrónico</string>
<string name="login_creado_nuevo_usuar">Se ha creado un nuevo usuario</string>
<string name="login_error_nuevo_usuar">El nuevo usuario no puede ser creado.</string>
```

Ilustración 28: Idiomas soportados

2.3.2 Casos de uso

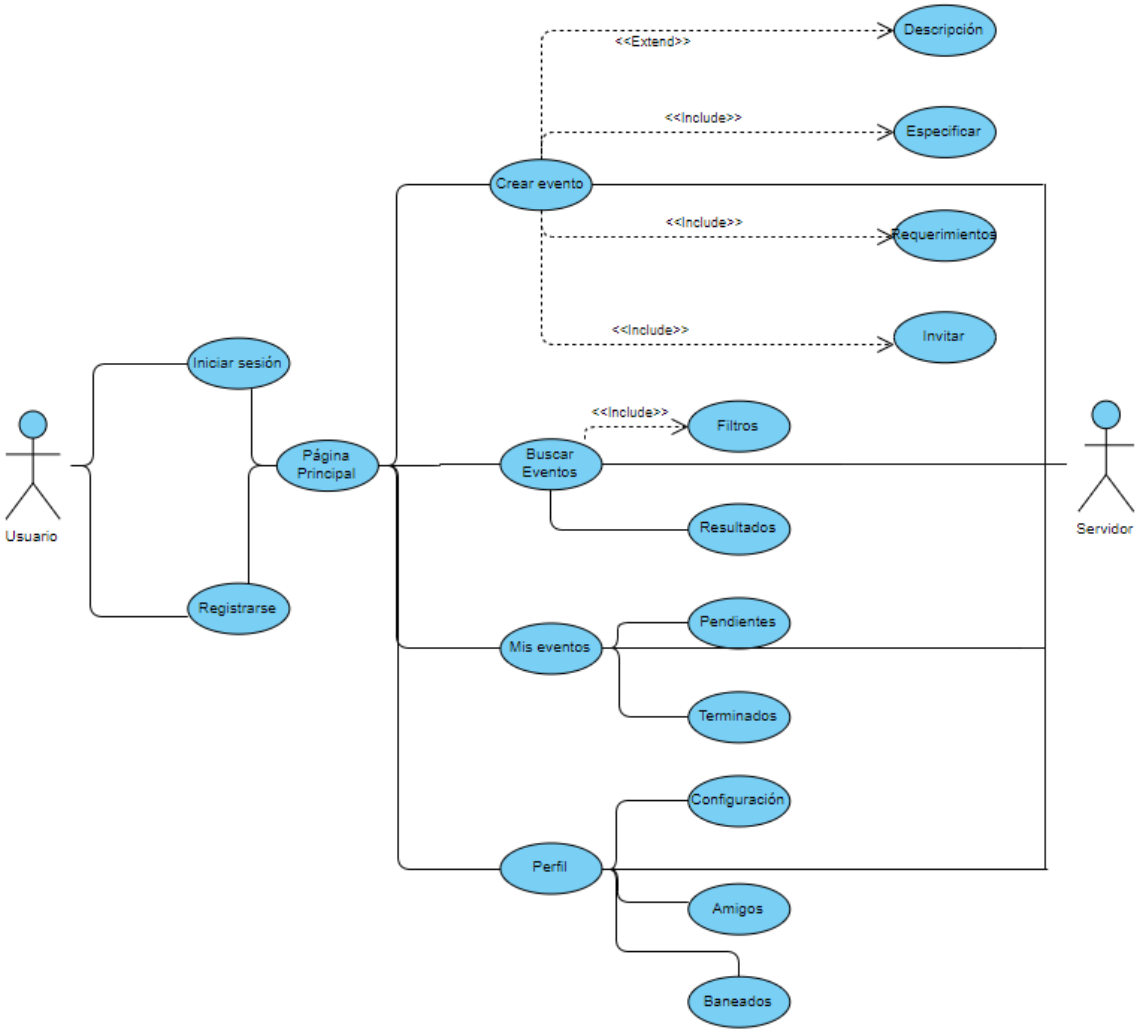


Ilustración 29: Casos de uso

2.4 Portabilidad

2.4.1 Plataformas

Android.

2.5 Rendimiento

2.5.1 Pruebas realizadas

La aplicación se ha testeado en diferentes dispositivos tales como: Nexus 6, pixel 2 XL, Redmi 7A… y en todos ellos no ha presentado ningún tipo de inconveniente más allá de las características de nuestro propio ordenador.

La aplicación es poco exigente y puede utilizarse en dispositivos con poca CPU, memoria RAM y ROM.

3 SERVIDOR API-REST

3.1 Introducción

Un Servicio Web RESTful proporciona recursos mediante URIs que se pueden usar para realizar solicitudes y recibir respuestas a través del protocolo HTTP. El formato que hemos usado para devolver y recibir los datos es JSON.

3.2 Creación

Se ha usado JAX-RS, la API de java para la creación de servicios restful y su implementación en Jersey, además de Maven para la gestión de dependencias.

JAX-RS proporciona algunas anotaciones para la creación del servicio:

@Path: Indicará la ruta de cada recurso @Path("/ruta") y se accederá a ella por medio de su URI, por ejemplo:

http://socialsports.ddns.net:8081/rest/ruta (protocolo, IP o dominio, puerto, context root, ruta de recursos rest, y ruta de recurso).

Los métodos de petición indicarán el tipo de operación a realizar:

@GET: Solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

```
@Secured
@GET
@Path("/amigos/{correo}")
@Produces({MediaType.APPLICATION_JSON})
public Response listaAmigos(@PathParam("correo") String correo) {

    usuarioDAO = new UsuarioDAO();
    ArrayList<Usuario> listaAmigos = usuarioDAO.listaAmigos(correo);

    return Response.status(Status.OK).entity(listaAmigos).build();

}
```

@PUT:Reemplaza un recurso del servidor.

```
@PUT
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Path("/actualizar/reserva")
public Response actualizarReserva(@FormParam("idEvento") String idEvento, @FormParam("reserva") boolean reserva) {
    eventoDAO = new EventoDAO();
    boolean actualizado = eventoDAO.actualizarReserva(idEvento, reserva);
    if(actualizado) {
        return Response.status(Status.NO_CONTENT).build();
    }else {
        return Response.status(Status.NOT_FOUND).build();
    }
}
```

@POST: Carga en el servidor una representación de un recurso.

```
@Secured
@POST
@Path("/amigos/agregar/{correo}/{correoAmigo}")
public Response agregarAmigo(@PathParam("correo") String correo, @PathParam("correoAmigo") String correoAmigo) {
    usuarioDAO = new UsuarioDAO();

    if(usuarioDAO.agregarAmigo(correo, correoAmigo)) {
        return Response.status(Status.NO_CONTENT).build();
    }else {
        return Response.status(Status.NOT_FOUND).build();
    }
}
```

@DELETE: Borra un recurso específico.

```
@Secured
@DELETE
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Path("/borrarusuario/{correo}")
public Response borrarUsuario(@PathParam("correo") String correo) {

    usuarioDAO = new UsuarioDAO();

    boolean borrado = usuarioDAO.borrarUsuario(correo);
    if(borrado) {
        return Response.status(Status.NO_CONTENT).build();
    }else {
        return Response.status(Status.NOT_FOUND).build();
    }

}
```

@Consumes y @Produces indicarían el tipo de datos a consumir o producir.

Y para pasarle los parámetros las anotaciones que más se han utilizado son:

@QueryParam:

Estructura de la petición: http://socialsports.ddns.net:8081/rest/eventos/buscar?deporte=futbol

@PathParam:

Estructura de la petición: http://socialsports.ddns.net:8081/rest/perfil/amigos/jose@hotmail.com

@FormParam: Debe ir acompañada del tipo APPLICATION_FORM_URLENCODED y solo puede usarse en los métodos con body como @POST y @PUT.

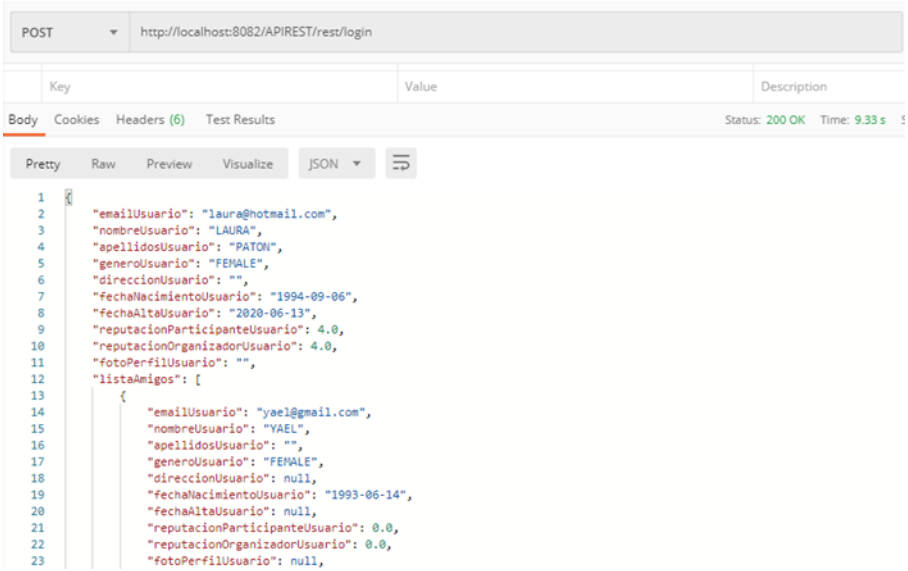
Un ejemplo de todo esto llevado a la práctica sería el método de login, que recibirá 2 parámetros que el cliente habrá enviado previamente en el body, luego comprobará con la clase que conecta a la base de datos si los datos introducidos son correctos y de ser así se genera un token que se le envia al usuario en la cabecera además de sus datos en el body en forma de Json:

```
1  package servicios;
2
3  import javax.ws.rs.Consumes;
16
17 @Path("/login")
18 public class Login {
19
20     @POST
21     @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
22     @Produces({MediaType.APPLICATION_JSON})
23     public Response login(@FormParam("emailUsuario") String emailUsuario, @FormParam("passwordUsuario") String passwordUsuario) {
24
25         UsuarioDAO usuarioDAO = new UsuarioDAO();
26
27         if(usuarioDAO.loginUsuario(emailUsuario, passwordUsuario)) {
28             JwtProvider jwt = new JwtProvider();
29             String token = jwt.generarToken(emailUsuario);
30
31             Usuario user = usuarioDAO.cogerUsuario(emailUsuario);
32             user.setListaAmigos(usuarioDAO.listaAmigos(emailUsuario));
33             user.setListaBloqueados(usuarioDAO.listaBloqueados(emailUsuario));
34
35             return Response
36                     .status(Status.OK)
37                     .header(HttpHeaders.AUTHORIZATION, "Bearer " + token)
38                     .entity(user)
39                     .build();
40         }
41
42         return Response
43                 .status(Status.UNAUTHORIZED)
44                 .build();
45     }
```

Método que consulta en la base de datos:

```java
public boolean loginUsuario(String correo, String contrasena) {

    Conexion conn = null;
    boolean valido = false;
    Validaciones validaciones = new Validaciones();

    if(validaciones.validarCorreo(correo) && validaciones.validarContrasena(contrasena)) {

        try {
            conn = new Conexion();
            PreparedStatement ps = conn.getConnection().prepareStatement("SELECT EMAILUSUARIO, PASSWORDUSUARIO FROM TABLAUSUARIOS WHERE EMAILUSUARIO = ?");
            ps.setString(1, correo);
            ResultSet rs = ps.executeQuery();

            if(rs.next()) {

                PasswordHash hash = new PasswordHash();
                String salt = getSalt(correo);
                hash.generatePassword(contrasena, salt);
                String hashedString = hash.getHash();

                if(hashedString.equals(rs.getObject(2))) {
                    valido = true;
                }
            }

            rs.close();
            ps.close();
            conn.closeConnection();

        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }

    return valido;
}
```

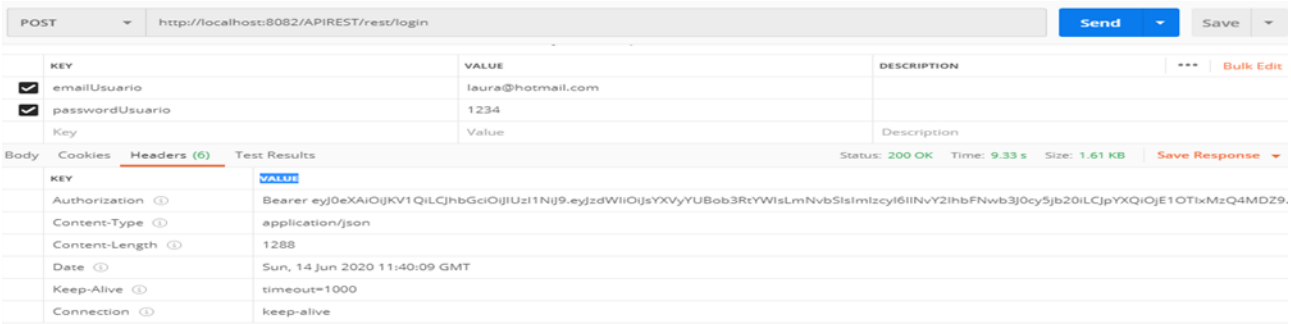
Cuerpo de la respuesta y ejemplo de la clase serializada:



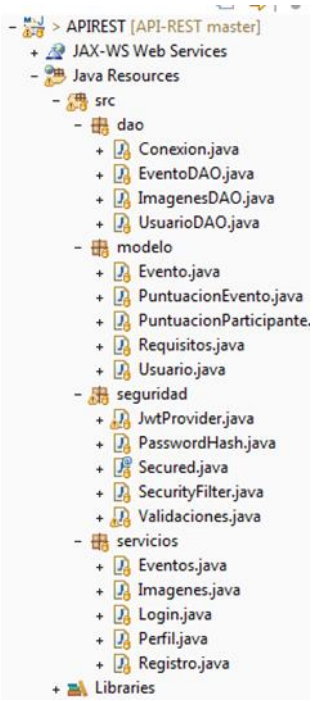

```java
package modelo;

import java.util.ArrayList;

@XmlRootElement
public class Usuario {

    private String emailUsuario;
    private String nombreUsuario;
    private String apellidosUsuario;
    private String generoUsuario;
    private String direccionUsuario;
    private String fechaNacimientoUsuario;
    private String fechaAltaUsuario;
    private float reputacionParticipanteUsuario;
    private float reputacionOrganizadorUsuario;
    private String fotoPerfilUsuario;
    private ArrayList<Usuario> listaAmigos;
    private ArrayList<Usuario> listaBloqueados;

```

Para serializar y deserializar las clases en la parte del servidor se ha usado la anotación de JAXB @XmlRootElement que permite convertir una clase modelo tanto a xml como a json.

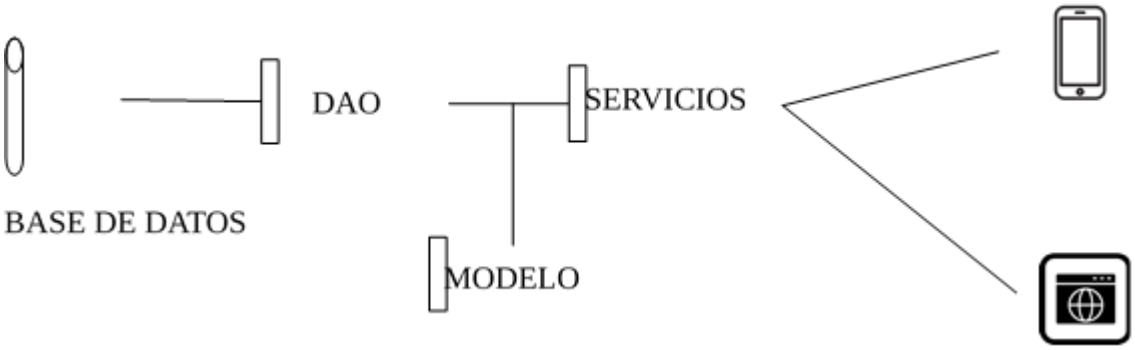
Cabecera con el token:



3.3 Diseño



- DAO: Contendrá todas las clases encargadas de conectar con la base de datos.
- Modelo: Estarán todos los objetos encargados de representar la información.
- Servicios: Será la capa encargada de recibir todas las peticiones y enviar las respuestas.



3.4 RETROFIT

Es una de las librerías de redes más usadas en android y permite hacer llamadas a un servicio web y consumir y producir datos JSON o XML de una forma muy sencilla.
Para ello ha hecho falta crear una interfaz con los métodos que harán las llamadas a las rutas del servidor.

```
86      /************************EVENTOS************************/
87
88      @GET("eventos/pendientes/{correo}")
89      Call<ArrayList<Evento>> listaEventosPendientes(@Header("Authorization") String authHeader, @Path("correo") String correo);
90
91      @GET("eventos/finalizados/{correo}")
92      Call<ArrayList<Evento>> listaEventosFinalizados(@Header("Authorization") String authHeader, @Path("correo") String correo);
93
94      @GET("eventos/buscar")
95      Call<ArrayList<Evento>> buscarEventos(@Header("Authorization") String authHeader,
96                                            @Query("deporte") String deporte, @Query("localidad") String localidad, @Query("fecha") String fecha,
97                                            @Query("hora") String hora, @Query("reservado") boolean reservado, @Query("reputacion") float reputaci
98
99      @FormUrlEncoded
100     @PUT("eventos/actualizar/fecha")
101     Call<ResponseBody> actualizarFechaEvento(@Header("Authorization") String authHeader, @Field("idEvento") String idEvento, @Field("fecha") Str
102
103     @FormUrlEncoded
104     @PUT("eventos/actualizar/hora")
105     Call<ResponseBody> actualizarHoraEvento(@Header("Authorization") String authHeader, @Field("idEvento") String idEvento, @Field("hora") Strin
106
107     @FormUrlEncoded
108     @PUT("eventos/actualizar/direccion")
109     Call<ResponseBody> actualizarDireccionEvento(@Header("Authorization") String authHeader, @Field("idEvento") String idEvento, @Field("direcci
110
111     @FormUrlEncoded
112     @PUT("eventos/actualizar/maximoparticipantes")
113     Call<ResponseBody> actualizarMaxParticipantesEvento(@Header("Authorization") String authHeader, @Field("idEvento") String idEvento, @Field("
114
```

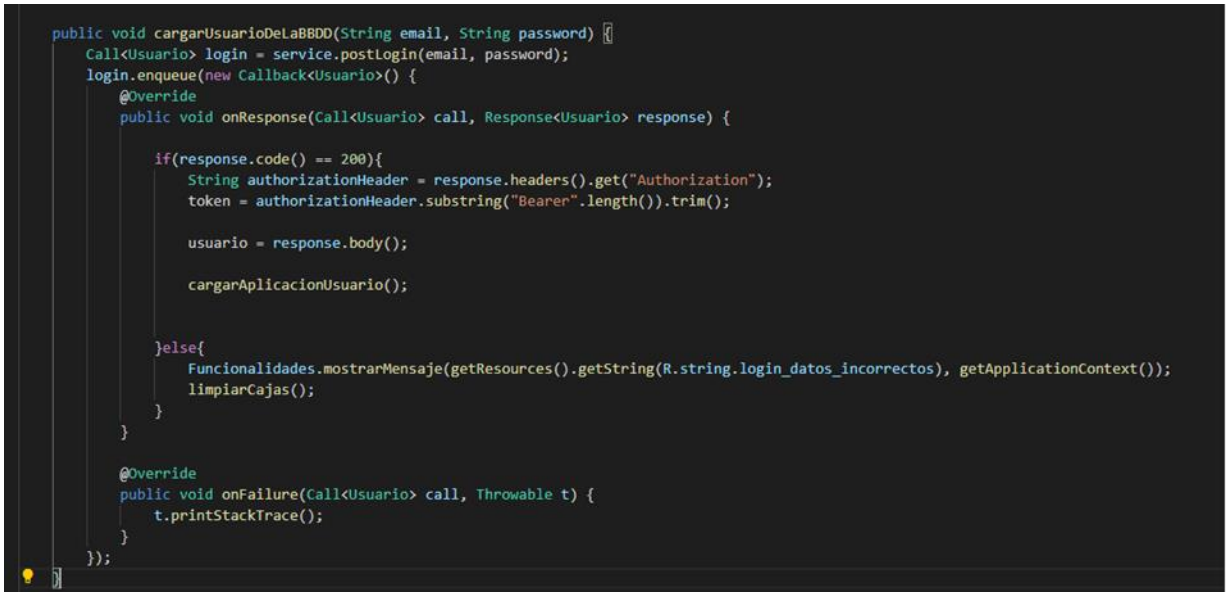
La url base y los conversores se indican al crear el objeto de retrofit:

```
import com.google.gson.GsonBuilder;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RETROFIT {

    private APIService service;

    public RETROFIT(){

        Gson gson = new GsonBuilder()
                .setDateFormat("yyyy-MM-dd")
                .create();

        Retrofit retrofit = new Retrofit.Builder()
                //.baseUrl("http://socialsports.ddns.net:8081/API-0.0.1-SNAPSHOT/rest/")
                .baseUrl("http://192.168.1.39:8082/APIREST/rest/")
                //.baseUrl("http://192.168.43.70:8080/APIREST/rest/")
                .addConverterFactory(GsonConverterFactory.create(gson))
                .build();

        service = retrofit.create(APIService.class);
    }

    public APIService getAPIService(){
        return this.service;
    }
}
```

Con esto y habiendo importado previamente las dependencias necesarias y poniendo el permiso a internet en el archivo manifest, seremos capaces de hacer las llamadas al servidor desde android.

Todas las llamadas se han realizado con “enqueue” que permite hacer llamadas asincronas; se ejecutarán de fondo y no bloquearán la ejecución de la aplicación.

Ejemplo de la llamada al método de login:



```
public void cargarUsuarioDeLaBBDD(String email, String password) {
    Call<Usuario> login = service.postLogin(email, password);
    login.enqueue(new Callback<Usuario>() {
        @Override
        public void onResponse(Call<Usuario> call, Response<Usuario> response) {

            if(response.code() == 200){
                String authorizationHeader = response.headers().get("Authorization");
                token = authorizationHeader.substring("Bearer".length()).trim();

                usuario = response.body();

                cargarAplicacionUsuario();


            }else{
                Funcionalidades.mostrarMensaje(getResources().getString(R.string.login_datos_incorrectos), getApplicationContext());
                limpiarCajas();
            }
        }

        @Override
        public void onFailure(Call<Usuario> call, Throwable t) {
            t.printStackTrace();
        }
    });
}
```

3.5 Ventajas de un servidor basado en rest

- Nos permite separar el cliente del servidor.
- Nos da escalabilidad: Podemos aumentar la capacidad de clientes y servidores por separado. Este servicio puede ser usado por distintos dispositivos como aplicaciones móviles, páginas web, cualquier cliente desarrollado con cualquier tipo de lenguaje.

4 CONCLUSIONES

¿Somos capaces de unir a personas con gustos en común gracias a la tecnología?
Retomando la demanda en cuestión, gracias a un fácil acceso a la aplicación mediante la descarga sencilla y gratuita y su interfaz de fácil manejo, podemos concretar que la gente está a un clic de realizar el deporte con quienes ellos consideren y puedan organizarse de manera sencilla y cómoda desde donde deseen, por lo tanto, se ha cumplido nuestro objetivo.
Como puntos fuertes podemos destacar varios aspectos:

- Es segura: Gracias a su cifrado de contraseña.
- Privacidad: Nunca daremos sus datos a terceros.
- Presencia de nube: Los datos quedan registrado en caso de pérdida o cambio de teléfono, desinstalación de la aplicación…
- Sin publicidad y gratuita: No tenemos pensado vender anuncios ni cuotas de suscripción.
- Está en dos idiomas: inglés y español.

La aplicación está más orientada a un público más juvenil, con capacidades más que suficientes para llevar a cabo dichos eventos, sin embargo, al poseer un amplio filtro en la creación de eventos, podemos personalizarlos hasta el punto en el que solo puedan entrar personas de cierta edad y condiciones, con lo cual, está disponible para todos los públicos.

5 REFERENCIAS BIBLIOGRÁFICAS

5.1 Páginas webs

Developer, A. (06 de Abril de 2020). *Android Developer*. Obtenido de https://developer.android.com/guide

Java, A. d. (s.f.). *Api de Java*. Obtenido de Api de Java: https://docs.oracle.com/javase/7/docs/api/

OpenWebinars. (s.f.). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/almacenar-contrasenas-bases-de-datos/

Valencia, U. P. (2017). *Universidad Politecnica de Valencia*. Obtenido de http://www.androidcurso.com/index.php/recursos/32-unidad-2-diseno-de-la-interfaz-de-usuario-vistas-y-layouts/213-uso-de-tabhost

W3Schools. (s.f.). *W3Schools*. Obtenido de SITE, THE WORLD'S LARGEST WEB DEVELOPER: https://www.w3schools.com/

5.2 Documentos

virtual, J. (s.f.). *Jovellanos virtual.* Obtenido de U.T.4 - BASES DE DATOS OBJETO-RELACIONALES Y ORIENTADAS A OBJETOS

5.3 Videos

Bottom, A. (s.f.). *Youtube*. Obtenido de Como programar una barra de navegación inferior - Android Bottom Navigation: https://www.youtube.com/watch?v=Z67AJVEAl_4&vl=en

Tutorial, A. S. (s.f.). *Youtube*. Obtenido de ViewFlipper - Android Studio Tutorial: https://www.youtube.com/watch?v=2c-GbJ-c_eA