Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Desarrollo en dispositivos móviles y fullstack:

One Greety

Development on mobile devices and fullstack:
One Greety

Elvis David Nogueiras Gonzalez

La Laguna, 13 de marzo de 2023

D. **Alejandro Pérez Nava**, con **N.I.F. 43821179-S** profesor asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Francisco Javier Rodríguez González**, con **N.I.F. 43.619.712-V** profesor Asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:
“Desarrollo en dispositivos móviles y fullstack: One Greety”

Ha sido realizada bajo su dirección por D. **Elvis David Nogueiras Gonzalez**, con **N.I.F. 79436893-E**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de de marzo de 2023

Agradecimientos

Primeramente, quiero agradecer a toda mi familia, especialmente a mis padres quienes me han dado todo el apoyo durante todos estos años de carrera.

Asimismo, quisiera expresar mi gratitud a todos los profesionales que me brindaron su ayuda y conocimiento en el desarrollo de este proyecto. Agradezco a mis tutores, quienes me guiaron y acompañaron en todo momento, compartiendo su experiencia y conocimiento conmigo. También agradezco a mis compañeros y amigos, quienes me brindaron su apoyo y motivación durante todo el proceso.

Licencia

Resumen

La presente memoria tiene como finalidad presentar todos los pasos en el diseño y desarrollo de One Greety, una aplicación móvil que tiene como objetivo reinventar la interacción entre los artistas y sus fans, intentando sustituir las páginas webs dedicadas a los clubs de fans y los servicios que ofrecen, unificando ambos conceptos en una sola aplicación.

One Greety además cuenta con una plataforma web únicamente disponible para los administradores o colaboradores designados por los artistas, la cual funciona a modo de back office, en donde se podrán crear espacios que representen a un club pudiendo definir varias características como una descripción del club, limitar su visibilidad por país o global y la privacidad del club la cual define la forma en cómo se debe suscribir los usuarios, siendo privado para que las peticiones sean aceptadas manualmente por el administrador o colaborador del club o pública para que las suscripciones sean automáticas.

One Greety es un proyecto desarrollado bajo el conjunto de tecnologías MERN, lo que lo hace bastante completo. Por tanto, se requiere desarrollar tanto en la parte backend como en el frontend. Este proceso va desde la creación de modelos para base de datos, una API REST gestionada por rutas y controladores creando un servidor con NodeJS y Express, hasta la implementación de componentes y cambios de estados en React y React-native para el diseño y funcionamiento de las aplicaciones web y móvil.

Palabras clave: MERN, Backend, Frontend, Full Stack, Android, MongoDB, Clubs de fans, Artistas, React, React-Native, Express, NodeJS.

Abstract

The purpose of this report is to present all the steps involved in the design and development of One Greety, a mobile application aimed at reinventing the interaction between artists and their fans. The goal is to replace fan club websites and the services they offer by unifying both concepts into a single application.

Additionally, One Greety has a web platform that is only available to administrators or collaborators designated by the artists. This platform serves as a back office where clubs can be created, and various characteristics can be defined, such as a club description, limited visibility by country or globally, and club privacy, which defines how users should subscribe. The privacy can be set to private, requiring the administrator or club collaborator to manually accept requests, or public, allowing for automatic subscription requests.

One Greety is a project developed using the MERN stack, which makes it a comprehensive platform. As such, it requires development in both the backend and frontend. This process involves creating models for the database, implementing a REST API managed by routes and controllers to create a server with NodeJS and Express, and implementing components and state changes in React and React Native for the design and operation of the web and mobile applications.

Keywords: *MERN, Backend, Frontend, Full Stack, Android, MongoDB, Fan clubs, Artists, React, React Native, Express, NodeJS.*

Índice de Figuras

Índice de Tablas

Capítulo 1
Introducción

1.1 Antecedentes y estado actual del tema

A lo largo del tiempo los clubes de fans(1) han evolucionado de la mano de la tecnología, pasando de únicamente asistir a conferencias o reuniones masivas organizadas por el artista a páginas web oficiales y no oficiales de difusión y participación online, diversos grupos dedicados en redes sociales y plataformas de pago que ofrecen contenido exclusivo, administrado por el artista para sus fans.

La evolución de la tecnología, el avance de internet y las redes sociales ha permitido que muchas personas publiquen su propio contenido y este pueda llegar a más audiencia con menos esfuerzo, en algunos casos llegando a ser “virales”, un fenómeno determinado así por el crecimiento exponencial de personas que consumen dicho contenido en corto plazo, impulsando a la fama a sus autores, debido a esto, los clubes de fans en la actualidad no solo están asociados a músicos, estrellas de cine o futbolistas como se suele pensar, si no también a una nueva generación de personajes públicos como los “streamers”, “tiktokers”, “instagramers”, “youtubers”, entre otros, nombrados así para hacer referencia a la plataforma que les dio la fama.

Hoy en día muchos clubes de fans tienen su propia página web, estos sitios suelen tener fotos e información sobre el objeto o la persona de su devoción, por ejemplo, una web de fans de un cantante puede tener vídeos suyos, foros, chats e información sobre sus próximos conciertos, por otro lado, twitter es la red social con más diversidad de club de fans, generados por personas anónimas por un bien común hacia la persona que apoyan.

Actualmente se puede encontrar dos modelos para un club de fans:

El primero es el que el club de fans está organizado por el propio artista, que se encarga de crear su propia base de datos con sus seguidores a nivel mundial a través de su propia página web y con la comunicación directa vía redes sociales. Este sistema no permite a las promotoras tener el control ni los datos del club de fans, siendo únicamente el artista quien conoce los seguidores que tiene por país y en función a eso determinar un porcentaje de entradas para poner en preventa. Además, ofrece contenido exclusivo o de acceso anticipado si estás suscrito al club.

En el segundo modelo la responsabilidad de organizar los clubes de fans está en manos de las discográficas o promotoras del artista, las cuales se encargan de buscar y agrupar fans por regiones mediante las redes sociales y usar únicamente esta vía para difundir información, crear eventos promocionales y actividades relacionadas al artista.

1.2 Objetivo

La plataforma One Greety tendrá por objetivo crear espacios en los que tanto fans como artistas puedan interactuar mediante la disposición de una serie de servicios(2) que permitan, entre otras cosas, chatear, subir archivos multimedia y participar en foros con temas relacionados al artista. Estos espacios se caracterizan por una serie de propiedades establecidas por un administrador (creador del espacio) que permiten gestionar y controlar la interacción del mismo a través de un back office en la plataforma web únicamente visible para dicho administrador o colaborador del espacio en donde le será posible gestionar peticiones de seguimiento, colaboradores y bloqueo de personas al club, además de establecer una serie de parámetros característicos del club como la región y su privacidad.

1.3 Fases de desarrollo

El desarrollo de este proyecto se divide en la siguientes fases:

- Diseñar un prototipo inicial mediante mockup.
    - Recopilación de información y servicios.
    - Prototipo de diseño y vistas para la aplicación web.
    - Prototipo de diseño y vistas para la aplicación móvil.
- Diseñar la estructura de datos para la base de datos.
    - Creación de diagramas de flujo.
    - Definición de clases y atributos principales.
    - Modelado de entidades y relaciones.
- Configuración e instalación de la base de datos.
    - Configurar el entorno de desarrollo.
    - Instalación y configuración de dependencias.
    - Creación de modelos de datos en base a las clases definidas previamente.
- Desarrollar una API REST que gestione las peticiones de la aplicación.
    - Instalación de dependencias necesarias para el servidor del Backend.
    - Configuración del servidor.
    - Definición de rutas de peticiones.
    - Definición de controladores que interactúen con los modelos y las funciones de rutas.

- Diseñar y desarrollar una aplicación web como back office para la gestión de clubes.
    - Instalación de dependencias necesarias (React)
    - Implementación del enrutamiento de páginas.
    - Implementación de la interfaz de usuario.
    - Implementación de funciones que permitan la conexión con el backend.
- Diseñar y desarrollar una aplicación móvil para la iteración de los usuarios en la plataforma.
    - Instalación de dependencias necesarias (React-native)
    - Implementación de la cola de vistas.
    - Implementación de la interfaz de usuario.
    - Implementación de funciones que permitan la conexión con el backend.
- Memoria del proyecto.
    - Corrección de errores encontrados mediante el desarrollo o uso de las aplicaciones.

1.4 Estructura del documento

- **Capítulo 2 - Tecnologías utilizadas**
  Definición de las tecnologías utilizadas para el desarrollo del proyecto.
- **Capítulo 3 - Desarrollo**
  Explicación detallada del desarrollo de la base de datos, el servidor Backend y las 2 aplicaciones que conforman el Frontend (Back office web y app móvil)
- **Capítulo 4 - Despliegue de las aplicaciones**
  Descripción del proceso de despliegue tanto del Backend como el Frontend.
- **Capítulo 5 - Conclusiones y líneas futuras**
  Resultados, conocimientos adquiridos y mejoras futuras.
- **Capítulo 6 - Summary and Conclusions**
  Results, knowledge acquired and future improvements.
- **Capítulo 7 - Presupuesto**
  Detalles de costes para el desarrollo, despliegue y mantenimiento del proyecto.

Capítulo 2
Tecnologías utilizadas

2.1 Visual Studio Code

Visual Studio Code(3) es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft para Windows, Linux y macOS. Se utiliza principalmente para escribir y depurar código en diversos lenguajes de programación, como C++, C#, Java, JavaScript, PHP, Python, entre otros.

Visual Studio Code ofrece una amplia gama de características y herramientas, como resaltado de sintaxis, autocompletado, depuración integrada, control de versiones y una gran cantidad de extensiones y complementos para personalizar la experiencia de programación.

2.2 Github

GitHub(4) es una plataforma web que ofrece servicios de alojamiento para proyectos de software utilizando el sistema de control de versiones Git. Permite la colaboración y el control de versiones de proyectos de software, así como también la gestión de problemas y solicitudes de extracción.

2.3 MongoDB

MongoDB(5) es un sistema de base de datos NoSQL (No Relacional) que permite almacenar y recuperar datos de forma flexible y escalable. Utiliza un modelo de datos de documentos JSON (JavaScript Object Notation) y permite una alta disponibilidad y rendimiento.

2.4 NodeJS

Es un entorno de tiempo de ejecución de JavaScript construido sobre el motor V8 de Google Chrome. Permite a los desarrolladores utilizar JavaScript en el servidor para construir aplicaciones web escalables y de alta velocidad.

NodeJS(6) cuenta con una gran cantidad de módulos y paquetes disponibles en su repositorio de paquetes llamado npm (Node Package Manager), que permite a los desarrolladores integrar fácilmente bibliotecas y herramientas en sus aplicaciones.

2.5 Express

Express(7) es un marco de aplicaciones web para NodeJS que ofrece una serie de funciones y herramientas para simplificar la creación de aplicaciones web. Permite el enrutamiento, el manejo de solicitudes y respuestas, y la integración de plantillas para generar vistas dinámicas.

2.6 Cloudinary

Es una plataforma de gestión de medios en la nube(8) que permite a los desarrolladores almacenar, manipular y entregar imágenes y videos en línea. Proporciona una API para realizar operaciones de manipulación de imágenes y videos, así como también una serie de herramientas para optimizar el rendimiento y la entrega de medios.

2.7 Socket.IO

Es una biblioteca de NodeJS que permite la comunicación en tiempo real bidireccional entre el servidor y el cliente a través de una conexión web en tiempo real. Socket.IO(9) proporciona una capa de abstracción sobre el protocolo de comunicación subyacente, lo que permite a los desarrolladores crear aplicaciones en tiempo real de manera más fácil.

2.8 React

React(10) es una biblioteca de JavaScript para construir interfaces de usuario. Permite a los desarrolladores construir componentes reutilizables y gestionar el estado de la aplicación de forma declarativa.

Las características más importantes de React son:

- Virtual DOM: React hace uso de un virtual DOM propio, distinto al DOM del navegador. Esto permite a la biblioteca determinar que partes del DOM han cambiado comparando el contenido entre las versiones del virtual DOM.
- Props: Son propiedades que pueden ir pasando entre los distintos niveles de componentes y contienen atributos de configuraciones.
- States: El estado de un componente se define como una representación de este en un momento concreto, es decir, una instantánea del propio componente.

2.9 React-Native

React Native(11) es un marco de desarrollo de aplicaciones móviles para construir aplicaciones nativas de iOS y Android utilizando la biblioteca de React. Permite a los desarrolladores compartir el código entre plataformas y proporciona un rendimiento cercano al de una aplicación nativa.

Capítulo 3
Desarrollo

3.1 Base de Datos

La base de datos de esta aplicación full stack fue implementada utilizando MongoDB Atlas(12), un servicio de base de datos en la nube ofrecido por MongoDB. MongoDB es una base de datos NoSQL, lo que significa que no sigue la estructura de tablas y filas que se encuentran en las bases de datos relacionales tradicionales. En su lugar, MongoDB utiliza documentos JSON para almacenar los datos, lo que permite una mayor flexibilidad en la forma en que se estructuran los datos y cómo se accede a ellos.

Una de las principales ventajas de usar MongoDB Atlas fue la facilidad de implementación y configuración. Al crear una cuenta en el sitio web de MongoDB Atlas, pude configurar una instancia de base de datos en la nube (cluster) en cuestión de minutos. Además, MongoDB Atlas ofrece una interfaz de usuario intuitiva que me permitió administrar y monitorear fácilmente la base de datos durante todo el desarrollo de la aplicación.

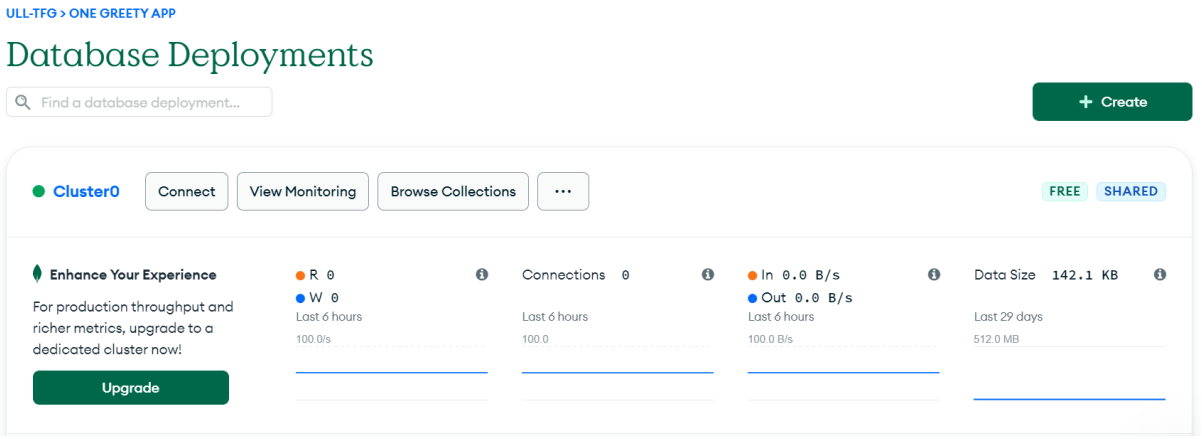

Figura 3.1: base de datos en MongoDB Atlas

También utilicé MongoDB Compass(13) para establecer una conexión con la base de datos y facilitar la edición de las tablas. MongoDB Compass es una herramienta gráfica que permite explorar y editar los datos almacenados en una base de datos de MongoDB de manera intuitiva y visual. La interfaz de usuario de MongoDB Compass es muy amigable, lo que facilitó el proceso de edición de los datos y redujo significativamente el tiempo que habría tenido que invertir en la edición manual de los documentos JSON.

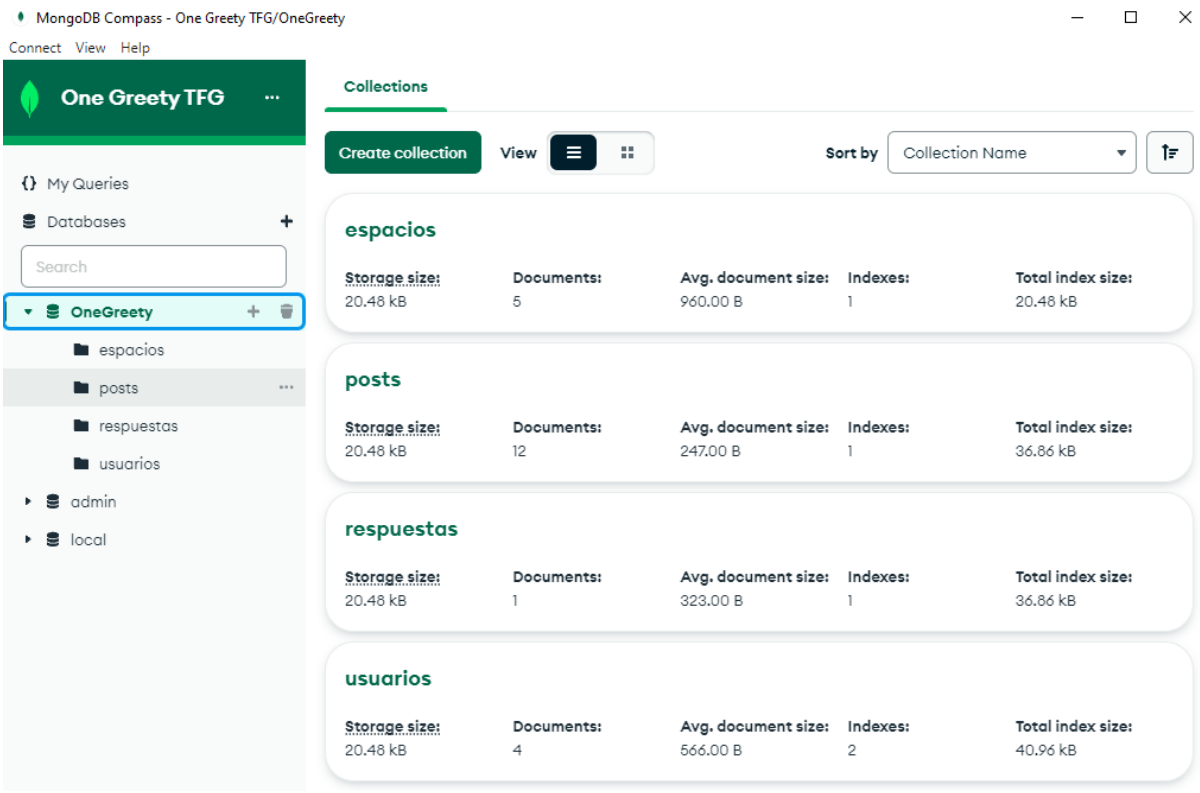

Figura 3.2: Base de datos vista desde MongoDBCompass

3.2 Backend

Para comenzar con el desarrollo del backend de la aplicación, el primer paso fue crear un proyecto de Node.js utilizando el comando npm init en la línea de comandos. Este comando inició un proceso interactivo que me permitió establecer el nombre del proyecto, la versión, una descripción, el punto de entrada del proyecto, las dependencias y otros detalles relevantes. Después de completar el proceso de configuración, se creó un archivo package.json que contiene toda la información sobre el proyecto y las dependencias instaladas.

Una vez creado el archivo package.json, pude comenzar a instalar las dependencias necesarias para el desarrollo del backend:

- **bcrypt:** Una biblioteca de cifrado de contraseñas que permite almacenar contraseñas de manera segura en la base de datos utilizando técnicas de hashing.

- **cloudinary:** El SDK de Cloudinary Node que permite integrar rápida y fácilmente el backend con Cloudinary.

- **cors:** Un paquete que permite la comunicación entre servidores en diferentes dominios para evitar errores de seguridad.

- **dotenv:** Un módulo que permite cargar variables de entorno desde un archivo .env en el servidor para evitar exponer datos confidenciales como contraseñas o claves API en el código fuente.

- **express:** Un framework de Node.js para la creación de aplicaciones web y APIs. Permite crear rutas, manejar solicitudes HTTP y más.

- **fs-extra:** Una biblioteca para trabajar con archivos y directorios en Node.js, con funciones adicionales a las que se encuentran en el módulo estándar fs.

- **jsonwebtoken:** Un paquete para crear y verificar tokens de autenticación JSON Web Tokens (JWT) utilizados para la autenticación y autorización de usuarios en la aplicación.

- **mongoose:** Una biblioteca para trabajar con bases de datos MongoDB en Node.js, proporcionando una capa de abstracción para interactuar con la base de datos y definir modelos de datos.

- **multer:** Un middleware para procesar archivos multipart/form-data en Express.

- **nodemailer:** Un módulo para enviar correos electrónicos desde Node.js.

- **socket.io:** Una biblioteca para la comunicación en tiempo real entre el servidor y el cliente a través de websockets.

- **uuid:** Un paquete para generar identificadores únicos universales (UUIDs) que se pueden utilizar para identificar objetos o entidades en la aplicación.

Además utilice **nodemon** como dependencia de desarrollo, la cual permite reiniciar automáticamente el servidor en cada cambio.

Una vez instaladas las dependencias, se configuran las conexiones tanto para la base de datos como para cloudinary definidos en los directorios “src/config/db.js” y “src/config/cloudinary.js” respectivamente.

backend > config > dB.js > ...

```
import mongoose from "mongoose";

const conectarDB = async () => {
  try {
    const conexion = await mongoose.connect(process.env.MONGO_URI, {
        useNewUrlParser: true,
        useUnifiedTopology: true
    });
    const url = `${conexion.connection.host}:${conexion.connection.port}`;
    console.log(`Base de datos conectada a ${url}`);
  } catch (error) {
    console.log(`Error al conectar la DB: ${error}`);
    process.exit(1);
  }
}

export default conectarDB;
```

Figura 3.3: Fichero db.js

backend > config > cloudinary.js > default

```
import cloudinary from "cloudinary";

const connectCloudinary = async () => {
  try {
    cloudinary.config({
      cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
      api_key: process.env.CLOUDINARY_API_KEY,
      api_secret: process.env.CLOUDINARY_API_SECRET,
    });
    console.log("Cloudinary connected");
  } catch (error) {
    console.log("Cloudinary connection error");
    console.log(error);
    process.exit(1);
  }
};

export default connectCloudinary;
```

Figura 3.4: Fichero cloudinary.js

Como se puede observar, hay variables empezadas con “process.env.”, esto es así por que son variables de entorno definidas en el fichero “.env” en la raíz del proyecto, dichas variables de entorno se utilizan para almacenar información sensible, como credenciales de bases de datos, claves de API, tokens de autenticación, entre otros. Al utilizar variables de entorno, esta información no se almacena directamente en el código fuente, lo que reduce el riesgo de que sea expuesta en caso de que el código se comparta o se publique en un repositorio público como es en este caso.

Además de configurar estas conexiones, también se definieron otras a las que llamé “helpers”, que como su nombre indica, son ficheros de código que definen funciones que ayudan a completar acciones como generar un token para el inicio de sesión y mandar correos electrónicos según sea el caso.

backend > helpers > generarJWT.js > ...

```
1  import jwt from 'jsonwebtoken';
2
3  const generarJWT = (payload) => {
4    return jwt.sign(payload, process.env.JWT_SECRET, { expiresIn: "30d" });
5  }
6
7  export default generarJWT;
```

Figura 3.5: Fichero generarJWT.js

backend > helpers > email.js > ...

```
1  import nodemailer from 'nodemailer';
2
3  export const emailRegistro = async (datos) => {
4    const { nombre, email, token } = datos;
5
6    const transport = nodemailer.createTransport({
7      host: process.env.EMAIL_HOST,
8      port: process.env.EMAIL_PORT,
9      secure: true,
10     auth: {
11       user: process.env.EMAIL_USER,
12       pass: process.env.EMAIL_PASS
13     }
14   });
15
16   const info = await transport.sendMail({
17     from: '"One Greety" <cuentas@onegreety.com>',
18     to: email,
19     subject: 'Confirmar Cuenta',
20     text: "Confirma tu cuenta en One Greety",
21     html: `
22               <h1>Hola ${nombre}</h1>
23               <p>Para confirmar tu cuenta, haz click en el siguiente enlace:</p>
24               <a href="${process.env.FRONTEND_URL}/confirmar/${token}"> Confirmar Cuenta </a>
25               <p>Si no has hecho ninguna petición, puedes ignorar este correo.</p>
26             `
27   });
28 }
```

Figura 3.6: Fichero email.js

Cabe destacar que las variables de entorno utilizadas en el fichero email.js son para una configuración SMTP y generadas por Gmail desde la cuenta onegreety@gmail.com, la cual fue creada con este fin.

Posteriormente se definieron los modelos que definen las tablas y los atributos de la base de datos y por otro lado las rutas y sus controladores.

Para este proyecto los modelos y atributos que se definieron fueron:

- Espacio
    - esp_nombre
    - esp_descripcion
    - esp_img_portada
    - esp_img_id
    - esp_administrador
    - esp_seguidores
    - esp_baneados
    - esp_peticiones
    - esp_colaboradores
    - esp_region
    - esp_acceso
    - esp_foro

- Post
    - post_creador
    - post_espacio
    - post_titulo
    - post_contenido
    - post_likes
    - post_comentarios
    - post_media_img
    - post_media_id
    - post_tags

- Respuesta
    - res_creador
    - res_post
    - res_contenido
    - res_media_img
    - res_media_id
    - res_comentarios
    - res_likes

- Usuario
    - usu_nombre
    - usu_email
    - usu_password
    - usu_token
    - usu_confirmado
    - usu_rol
    - usu_espacios
    - usu_esp_colaborador
    - usu_perfil_img
    - usu_img_id
    - usu_region
    - usu_twitter
    - usu_tiktok
    - usu_youtube
    - usu_instagram
    - usu_spotify
    - usu_soundcloud

Estos modelos están definidos por medio de un “schema” en la ruta “models/” de la cual se crearán las tablas y los atributos de la base de datos de mongoDB automáticamente.

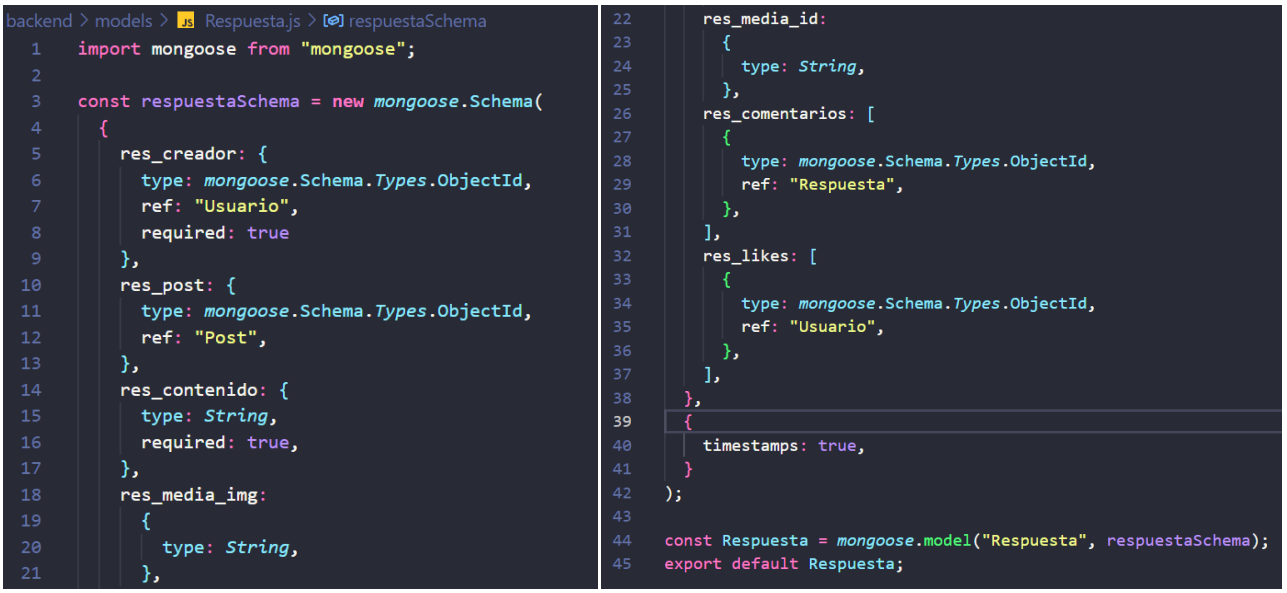

Figura 3.7: Ejemplo de definición de Schema con el modelo Respuesta.

Por otra parte, las rutas están definidas en “routes/”, divididas en diferentes ficheros que separan la lógica del proyecto:

- usuarioRoutes.js, para las rutas que impliquen datos de usuario.
- espaciosRoutes.js, para las rutas que impliquen datos de los clubes.
- filesRoutes.js, para las rutas que impliquen datos multimedia.
- postRoutes.js, para las rutas que impliquen datos del foro.

Tomaré como ejemplo la ruta de usuarios:

backend > routes > usuarioRoutes.js > ...

```
1  import express from "express";
2  import {
3    registrar,
4    autenticar,
5    confirmar,
6    recuperarPassword,
7    comprobarToken,
8    nuevoPassword,
9    perfil,
10   editarPerfil,
11   obtenerPerfilUsuario
12 } from "../controllers/usuarioController.js";
13 import checkAuth from "../middleware/checkAuth.js";
14
15 const router = express.Router();
16
17 router.post("/", registrar);
18 router.post("/login", autenticar);
19 router.get("/confirmar/:token", confirmar);
20 router.post("/recuperar-password", recuperarPassword);
21 router.route("/recuperar-password/:token").get(comprobarToken).post(nuevoPassword);
22 router.get("/perfil", checkAuth, perfil)
23 router.put("/perfil/:id", checkAuth, editarPerfil)
24 router.get("/perfil/usuario/:id", obtenerPerfilUsuario)
25
26
27 export default router;
28
```

Figura 3.8: Fichero usuarioRoutes.js

Las rutas se utilizan para especificar qué acción se debe realizar para cada petición HTTP que llega al servidor. En este caso, se utilizó el framework Express.js para crear las rutas del backend. Cada ruta se define utilizando el método correspondiente (get, post, put, delete, etc.) y se especifica la función del controlador que se encargará de procesar la petición. En este ejemplo, se importan dichas funciones desde el archivo usuarioController.js y se asocian a cada ruta mediante el método correspondiente.

En una API REST (14), los métodos HTTP se utilizan para indicar la acción que se quiere realizar sobre un recurso. En este caso, se utilizan los métodos POST, GET y PUT para registrar nuevos usuarios, autenticar usuarios, obtener y actualizar información del perfil de un usuario, entre otras funcionalidades que dependen o no del middleware “checkAuth”.

Un middleware(15) es una función que se ejecuta entre el servidor y la función del controlador de una ruta, y que tiene acceso a la petición y la respuesta. Los middleware se utilizan para realizar operaciones comunes en varias rutas, como verificar la autenticación del usuario, validar datos de entrada, establecer encabezados de respuesta, entre otros. En este caso, se utiliza el middleware checkAuth.js para verificar que el usuario esté autenticado antes de permitir el acceso a la información del perfil y la edición del mismo.

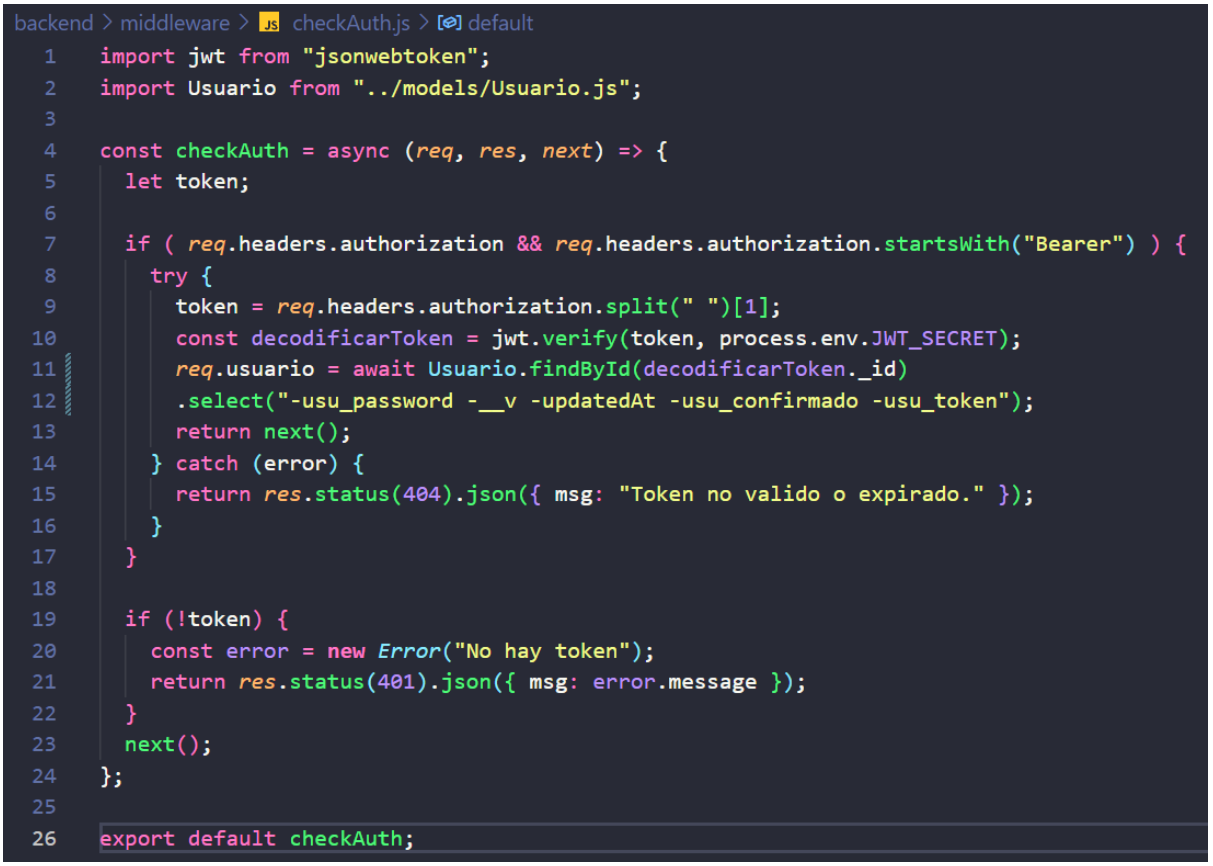

Figura 3.9: Fichero checkAuth.js

Cuando se recibe una solicitud, el middleware primero verifica si se proporciona un token de autenticación en el encabezado de autorización de la solicitud. Si existe un token, se verifica su validez utilizando la clave secreta definida en el archivo .env. Si el token es válido, se recupera el identificador del usuario (_id) codificado en el token y se busca en la base de datos el objeto Usuario correspondiente. A continuación, se agrega una propiedad usuario al objeto req para

que pueda ser utilizada por los controladores de las rutas que usen este middleware.

Si el token no es válido o no existe, se devuelve un mensaje de error con un código de estado 401 (no autorizado).

Por otra parte, los controladores están definidos en la ruta “controllers/”, como ejemplo, se muestra a continuación la función que registra a los usuarios:

Figura 3.10: Funcion de registro de usuario

Todas las funciones de los controladores tienen una lógica similar, la cual consiste en extraer la información (si la hay) de la petición, consultar la base de datos y generar una respuesta. Se puede consultar a detalle todas las funciones desde el repositorio de GitHub del Backend(16).

Para iniciar el servidor, se declaró he importo todas las rutas y librerías necesarias en el fichero index.js quedando de la siguiente manera:

backend > index.js > ...

```
import express from "express";
import conectarDB from "./config/dB.js";
import dotenv from "dotenv";
import usuarioRoutes from "./routes/usuarioRoutes.js";
import espacioRoutes from "./routes/espacioRoutes.js";
import postRoutes from "./routes/postRoutes.js";
import filesRoutes from "./routes/filesRoutes.js";
import connectCloudinary from "./config/cloudinary.js";
import cors from "cors";

const app = express();
app.use(express.json());

dotenv.config();

conectarDB();
connectCloudinary();

const whiteList = [process.env.FRONTEND_URL];
const corsOptions = {
  origin: (origin, callback) => {
    console.log(origin);
    if (whiteList.includes(origin)) {
      callback(null, true);
    } else {
      callback(new Error(`Acceso no permitido por CORS ${origin}`));
    }
  },
};

app.use(cors());
app.use("/api/usuarios", usuarioRoutes);
app.use("/api/espacios", espacioRoutes);
app.use("/api/foros", postRoutes);

app.use("/api/files", filesRoutes);
```

```
app.use("/api/files", filesRoutes);

const PORT = process.env.PORT || 4000;
const server = app.listen(PORT, () => {
  console.log(`Servidor corriendo en el puerto ${PORT}`);
});

// Socket.io

import { Server } from "socket.io";

const io = new Server(server, {
  pingTimeout: 60000,
});

io.on("connection", (socket) => {
  console.log("Usuario conectado socket.io");

  socket.on("chat room", (espacio_id) => {
    console.log('id' + espacio_id);
    socket.join(espacio_id);
  });

  socket.on("nuevo msg", (chatMsgs, esp_id) => {
    console.log(chatMsgs, esp_id);
    socket.to(esp_id).emit('msg agregado', chatMsgs, esp_id);
  });
});

```

Figura 3.11: Fichero index.js (Backend)

Este código crea una aplicación Express que utiliza una base de datos de MongoDB para crear la API REST que permite a los usuarios acceder a diferentes rutas previamente explicadas (usuarioRoutes, espacioRoutes, postRoutes y filesRoutes) para realizar diferentes operaciones en la base de datos, como crear usuarios, editar espacios, crear publicaciones y subir archivos.

La aplicación también utiliza el módulo CORS(17) para permitir que el frontend interactúe con el backend.

Además, la aplicación utiliza Socket.io para permitir la comunicación en tiempo real entre los usuarios conectados a la aplicación, permitiendo un chat en diferentes espacios.

Por último, la aplicación arranca un servidor HTTP en el puerto definido en la variable de entorno PORT que es asignado automáticamente por la plataforma de despliegue del backend la cual se hablará en el capítulo siguiente.

3.3 Frontend Web

Para el desarrollo de la página web, utilicé React, una biblioteca de JavaScript de código abierto para construir interfaces de usuario. Al utilizar React, pude dividir la interfaz de usuario en componentes reutilizables, lo que me permitió construir una aplicación modular y fácil de mantener. Además, la biblioteca proporciona un sistema de manejo de estados fácil de usar (Context)(18), lo que me permitió mantener un seguimiento de los cambios en los datos de la aplicación y actualizar la interfaz de usuario en consecuencia.

Figura 3.12: Ejemplo de definición de componente en react

El Context de React es una forma de compartir datos entre componentes sin tener que pasar props manualmente. El Context permite acceder a datos compartidos como un objeto global dentro de los componentes, lo que facilita el manejo de los estados en la aplicación. Dichos context están definidos en la ruta “src/context/”.

Para poder tener acceso a los estados definidos en los context, creé mis propios hooks personalizados los cuales están definidos en la ruta “src/hooks”. Estos Hooks hacían uso de “useContext” para acceder a los estados específicos que necesitaba en diferentes partes de mi aplicación. Luego, simplemente llamé a estos Hooks en los componentes relevantes, lo que me permitió acceder fácilmente a los estados definidos en el context que necesitaba en cada lugar de la aplicación.

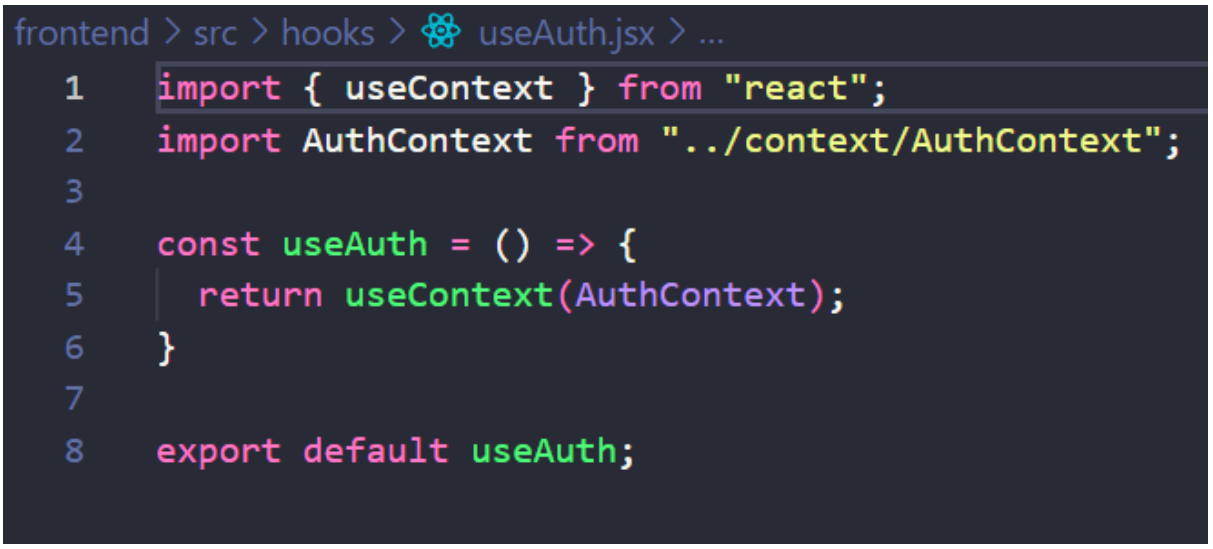

Figura 3.13: Ejemplo de definición de hook en react.

Para acelerar el proceso de desarrollo, también utilicé Vite(19), un entorno de desarrollo web que permite la compilación rápida y eficiente de aplicaciones web. Vite proporciona una serie de herramientas y características útiles, como la recarga en caliente (hot-reloading), que me permitió ver los cambios en tiempo real mientras trabajaba en la aplicación.

Por otra parte, las dependencias que utilice para esta parte del proyecto son:

- axios: una biblioteca que permite hacer solicitudes HTTP desde el navegador y Node.js.
- date-fns: una biblioteca que proporciona funciones de manipulación de fechas y horas.
- react-router-dom: una biblioteca que proporciona herramientas de enrutamiento para aplicaciones de React.
- socket.io-client: una biblioteca que proporciona una API de cliente para interactuar con servidores WebSocket.
- sweetalert2: una biblioteca que proporciona una interfaz de usuario de diálogo modal personalizable y fácil de usar.

- tailwindcss: una biblioteca de utilidades de CSS altamente personalizable utilizada para diseñar y estilizar aplicaciones de manera eficiente.

A continuación explico los componentes principales que conforman la aplicación web.

3.3.1 Inicio de sesión

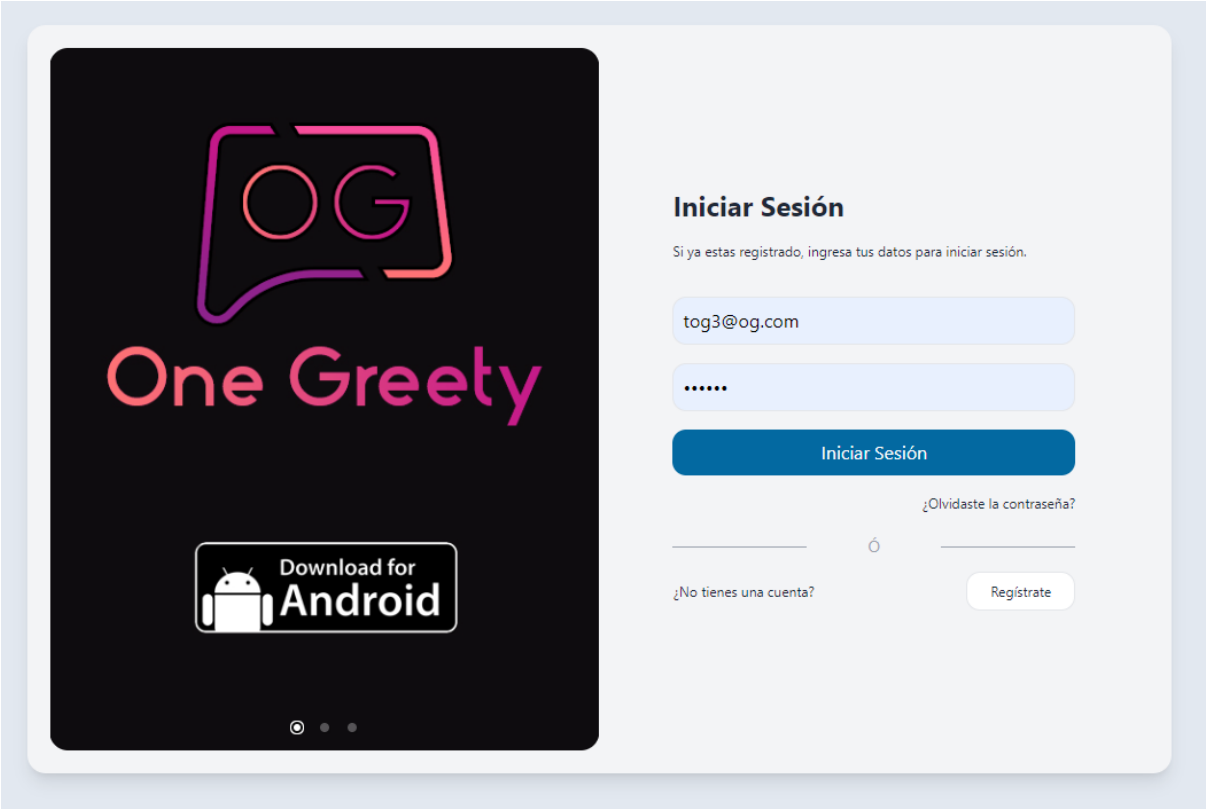

Figura 3.14: Inicio de sesión web

Este compone es el primero en mostrarse en la página web por los administradores, en donde se podrá iniciar sesión o tener acceso al registro de la cuenta o recuperar la contraseña.

3.3.2 Formulario de petición de registro como administrador de un club.

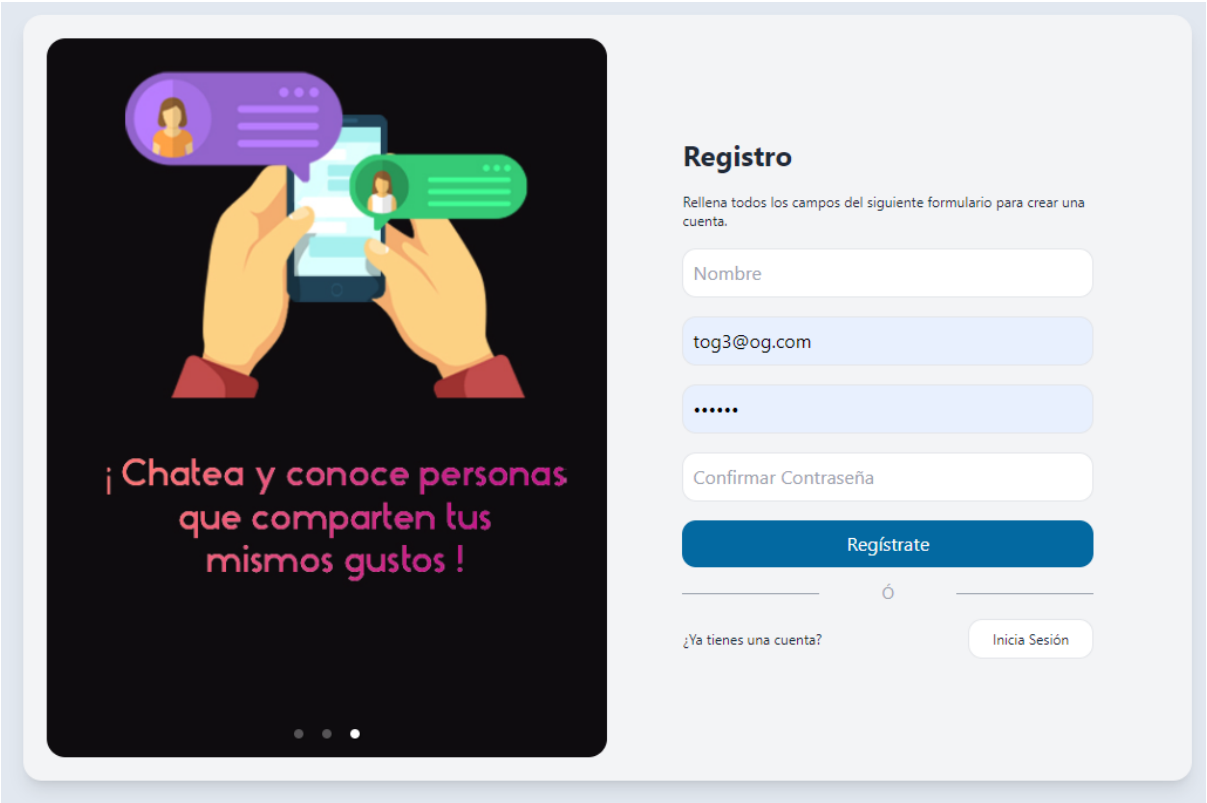

Figura 3.15: componente de registro web.

El componente de registro se muestra al presionar sobre el botón “Regístrate” de la pantalla de inicio de sesión, el cual permite el registro de los administradores luego de llenar los inputs.

3.3.3 Formulario de recuperación de contraseña

Figura 3.16: Componente de recuperación de contraseña.

Para acceder a este componente, se debe dar click en el link “¿Olvidaste la contraseña?” del componente de inicio de sesión. En este componente es posible recuperar la contraseña mediante un correo electrónico que se envía desde la ruta de usuarios del backend, verificando si existe un usuario con el correo ingresado en el input del formulario.

3.3.4 Listado de clubes.

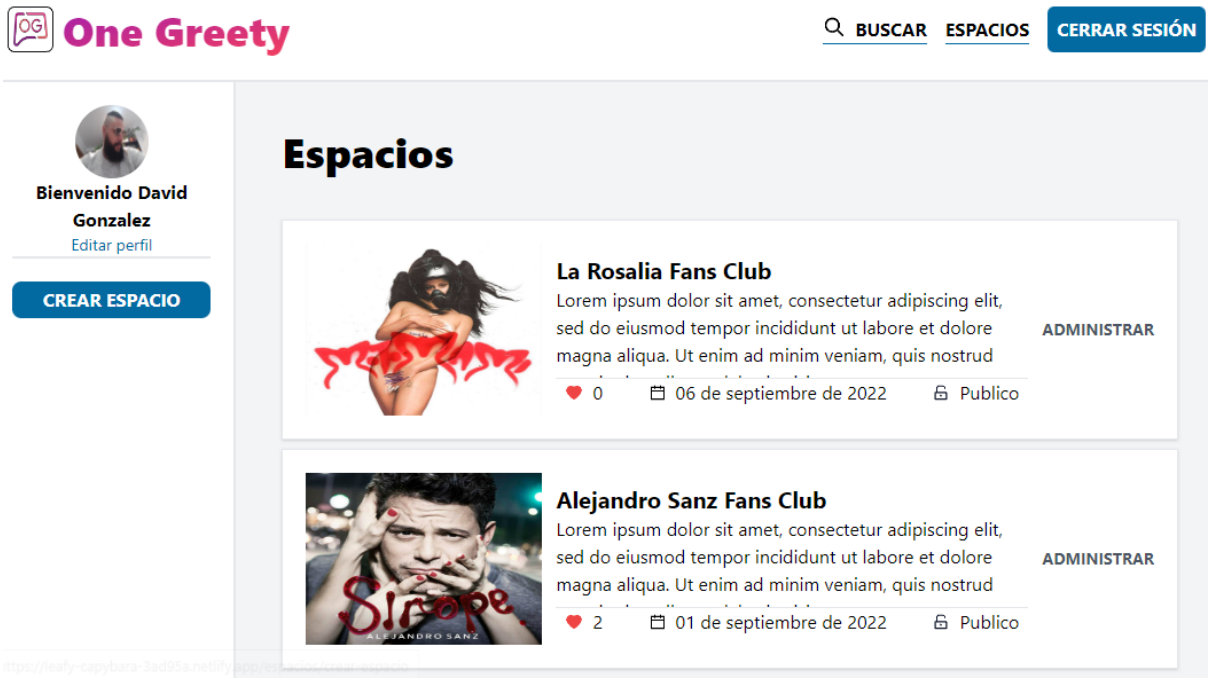

Figura 3.17: Componente de listado de clubes

Este componente es el que logra ver el administrador o colaborador una vez inicie sesión, no está demás aclarar que este componente está conformado por un conjunto de componentes como lo es el sidebar, el navbar y los items de la lista que conforman los clubes. En el solo se listaran los componentes al que el administrador puede acceder, verificado si el usuario autenticado tiene el mismo id que en el atributo esp_administrador del espacio.

3.3.5 Vista individual del club.

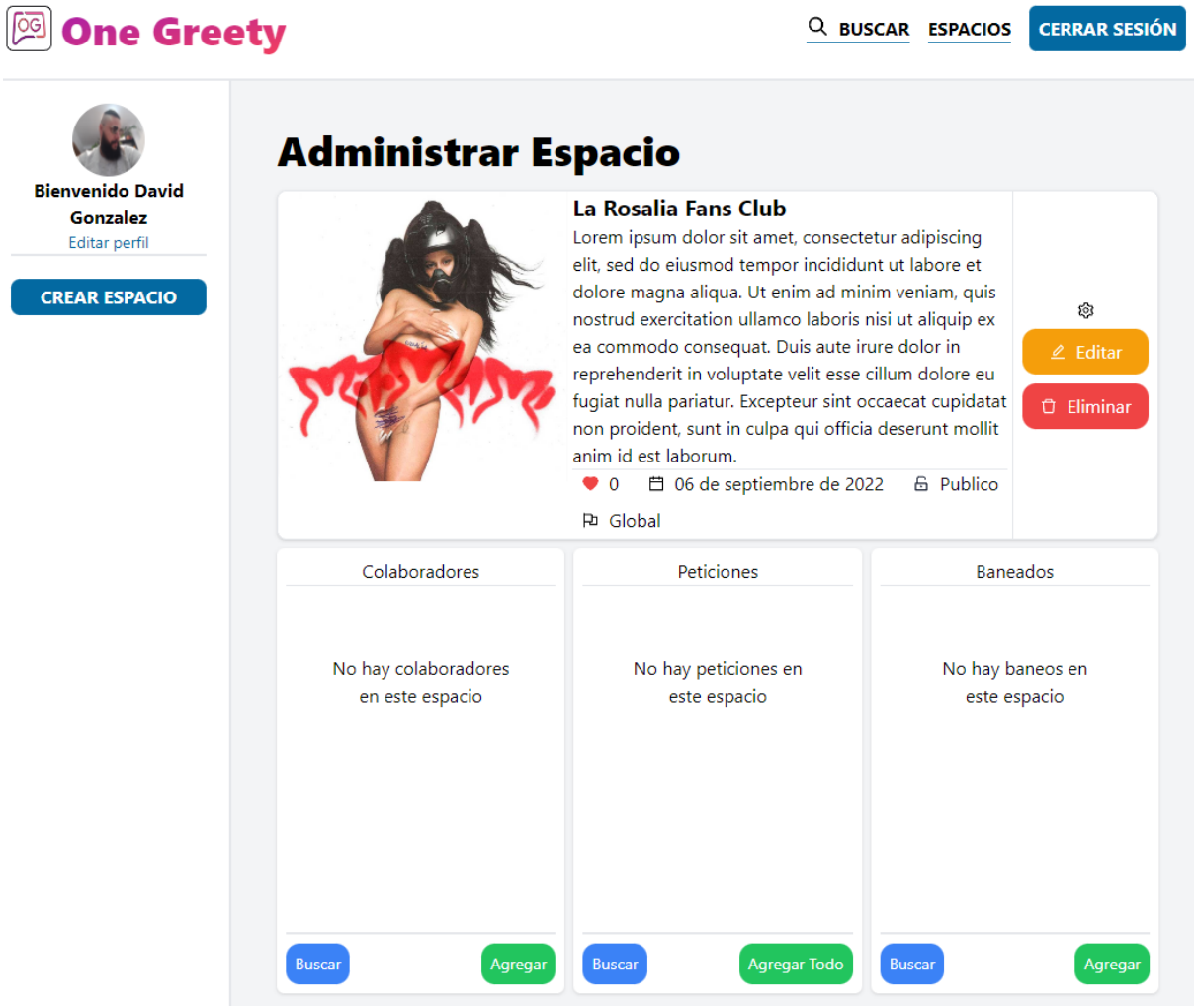

Figura 3.18: Componente de club individual

Este componente funciona como panel de administración que utilizaran los administradores o colaboradores para gestionar el club, este componente permite observar los diferentes atributos del club como el título, la descripción, los seguidores, la fecha de creación, su privacidad y la región. Además permite gestionar a los colaboradores, las peticiones de seguimiento al club y a las personas vetadas o baneadas del club, agregándolas a una lista o eliminandolas de ella según sea el caso.

3.3.6 Publicar o editar club

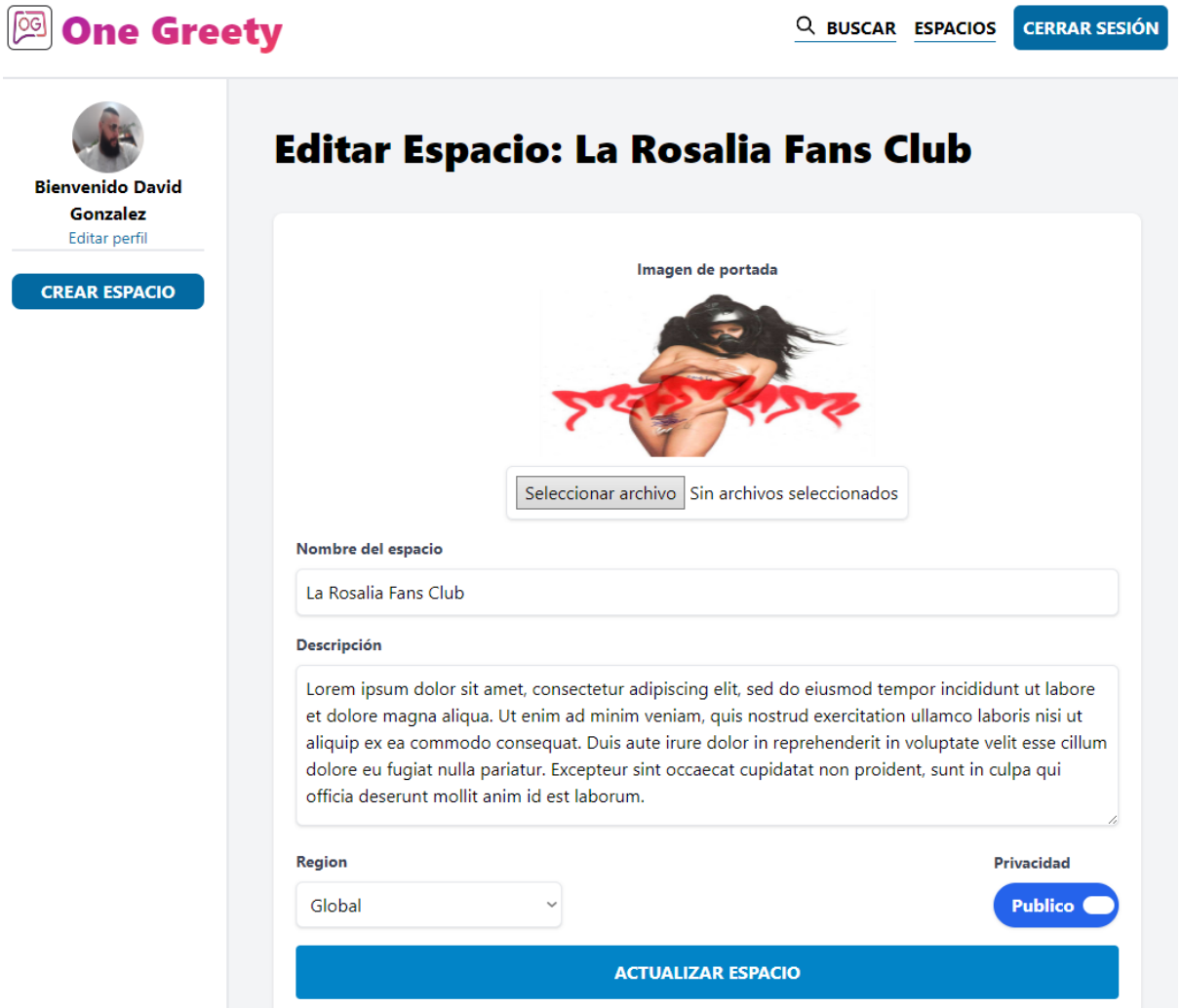

Figura 3.19: Componente para editar o publicar espacio

En este componente se permite actualizar los valores del espacio, verificando mediante una función cada uno de los inputs para evitar errores que puedan dañar la aplicación. Este componente también reutilizado para crear un nuevo club, se verifica un estado llamado “espacio” que se rellena con la información del club seleccionado al darle click en el listado de clubes y se borra al volver a él, de modo que si este estado está vacío, el componente sabe que va a crear un nuevo espacio o a editarlo en caso contrario.

3.3.7 Editar perfil de usuario

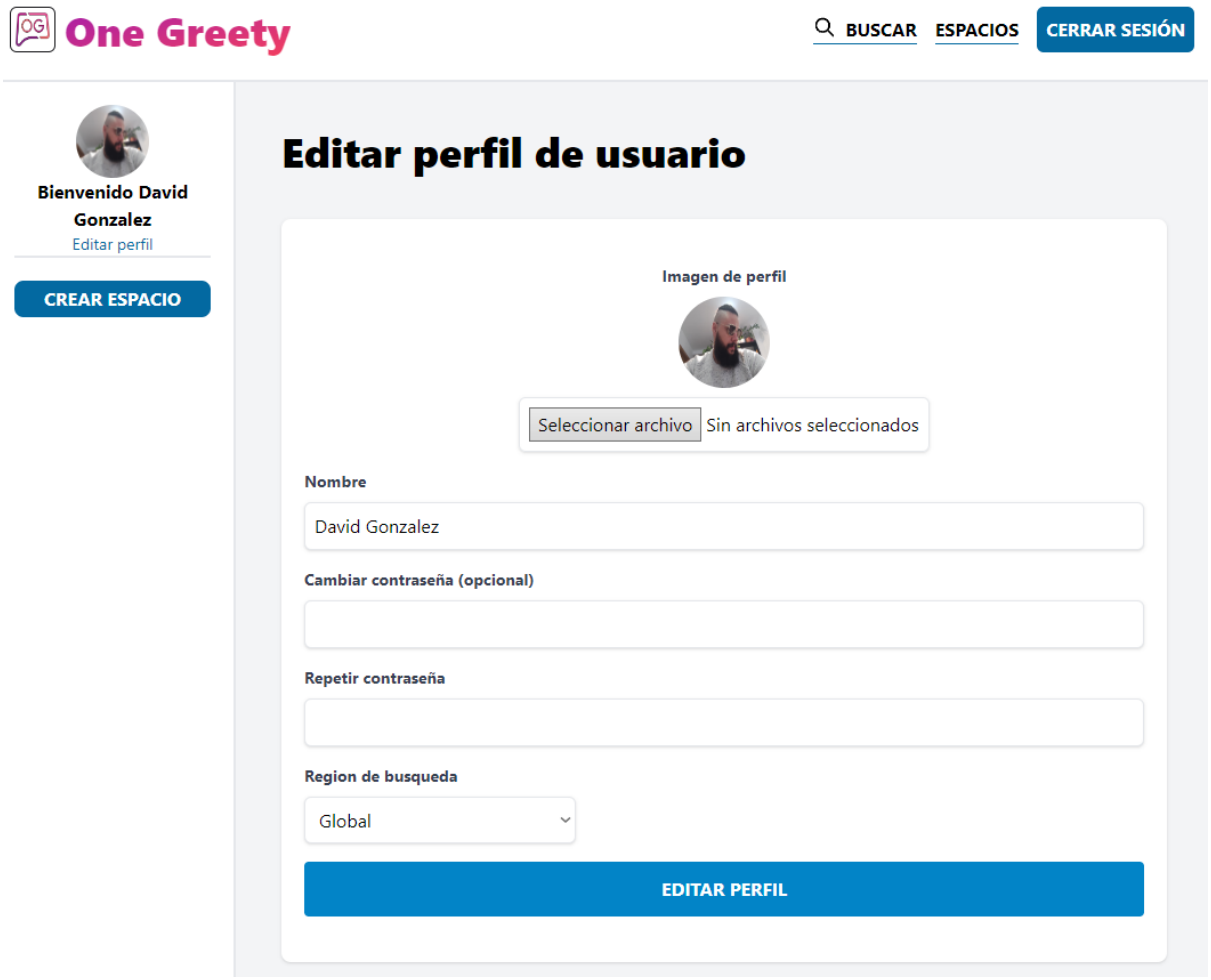

Figura 3.20: Componente para editar perfil de usuario

Parecido al componente anterior, este contiene un formulario para editar los atributos del usuario, también validados por una función que valida entre otras cosas que las contraseñas sean iguales y de mínimo 6 caracteres.

No está de más recordar que el código del desarrollo de esta plataforma web está disponible en el repositorio de GitHub frontend (20).

3.4 Frontend Mobile

React Native es un framework de desarrollo de aplicaciones móviles que permite a los desarrolladores crear aplicaciones para iOS, Android y otras plataformas móviles utilizando JavaScript y la biblioteca React. Para desarrollar la aplicación, utilicé Metro(21), que es un paquete de herramientas de JavaScript que se utiliza para compilar y empaquetar aplicaciones en React Native.

Además, para probar la aplicación durante el desarrollo, utilicé un emulador de Android Studio(22), que me permitió simular el comportamiento de una aplicación en un dispositivo Android sin tener que instalar la aplicación directamente en un dispositivo físico.

Entre las dependencias que use para el desarrollo de esta aplicación están:

- axios: Permite realizar solicitudes HTTP para recuperar datos de una API.
- react-native-document-picker: Permite seleccionar documentos de la galería o del sistema de archivos.
- react-native-paper: Proporciona componentes de IU de material design para React Native.
- react-native-reanimated: Permite crear animaciones fluidas y de alta calidad.
- react-native-vector-icons: Ofrece una amplia variedad de iconos vectoriales personalizables para su uso en la aplicación.
- socket.io-client: Permite la comunicación en tiempo real entre el cliente y el servidor mediante sockets.

A continuación explico los componentes principales que conforman la aplicación móvil.

3.4.1 Pantalla principal

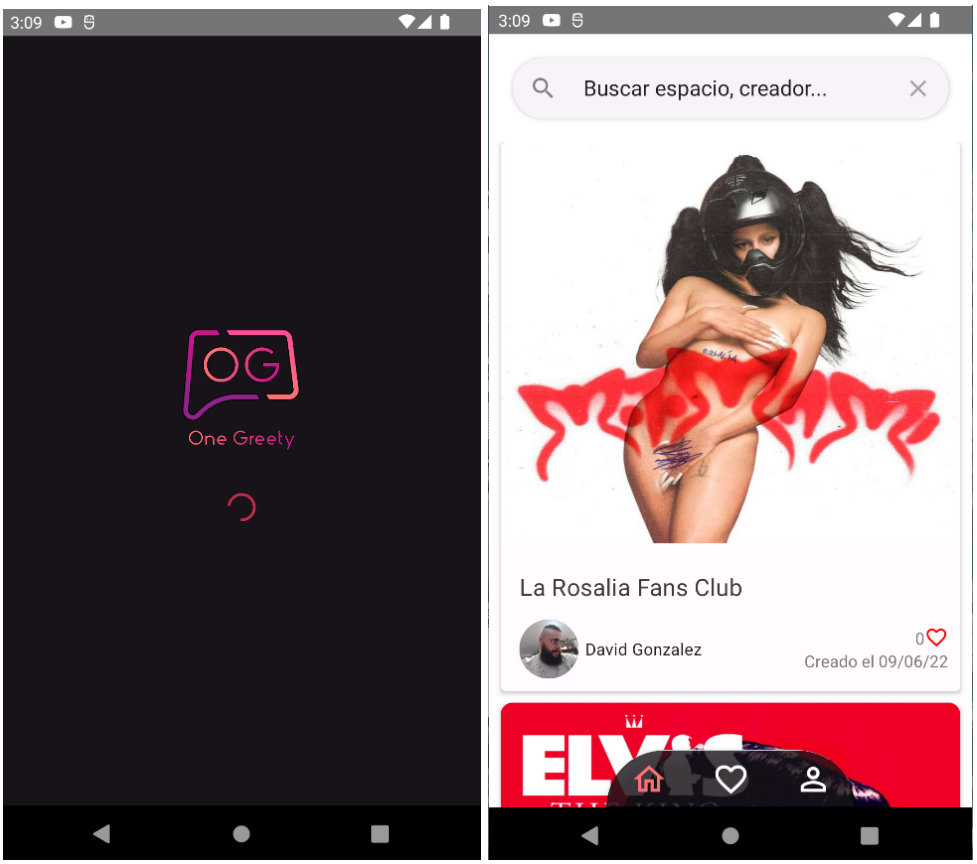

Figura 3.21: Pantalla de inicio de la aplicación

Al abrir la aplicación aparecerá esta pantalla donde se listan todos los clubes globales, esta pantalla cuenta con un buscador en la parte superior que filtra los clubes por nombre del club o el nombre del administrador. También se puede observar el menú de navegación principal, el cual permite al usuario acceder a la lista de clubes seguidos y a su perfil de usuario.

Para acceder a esta pantalla no es necesario el inicio de sesión, es posible acceder a los clubes y ver su información mas no se puede acceder a sus servicios a menos que estés registrado y seas un seguidor del club seleccionado.

3.4.2 Listado de clubes seguidos

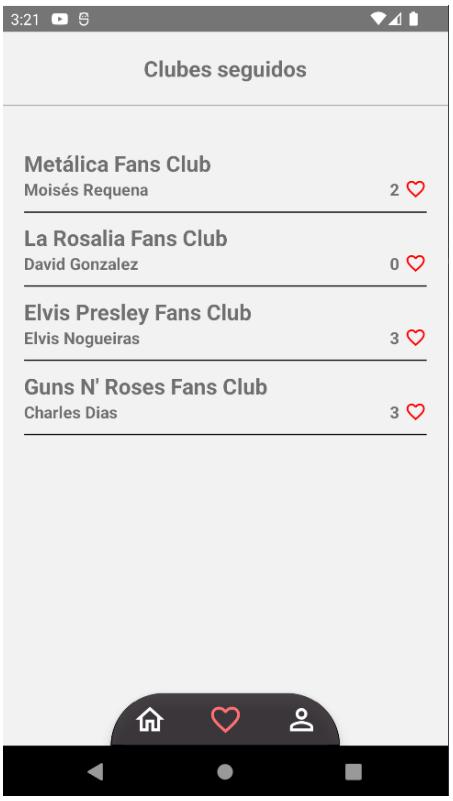

Figura 3.22: Pantalla de clubes seguidos

Esta pantalla muestra la lista de clubes al que el usuario registrado pertenece, tiene como función permitirle al usuario acceder a los clubes de una forma rápida y sencilla sin tener que buscarlo en el listado de clubes.

3.4.3 Perfil de usuario personal

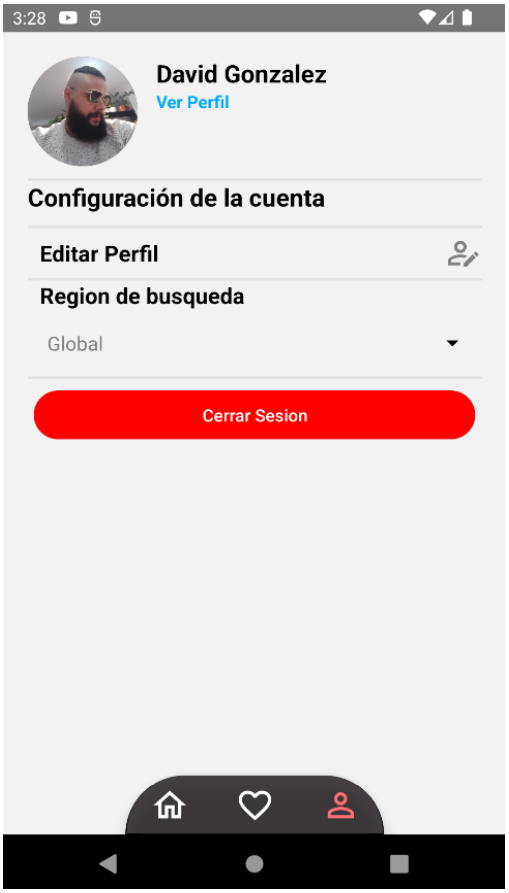

Figura 3.23: Pantalla de perfil de usuario personal

A esta pantalla solo se puede acceder una vez estés registrado y autenticado en la aplicación. En esta pantalla se le permite a los usuarios acceder a su perfil público, editar sus datos y la región de búsqueda que se toma en cuenta en el listado de clubes de la pantalla principal.

3.4.4 Inicio de sesión y registro de usuarios

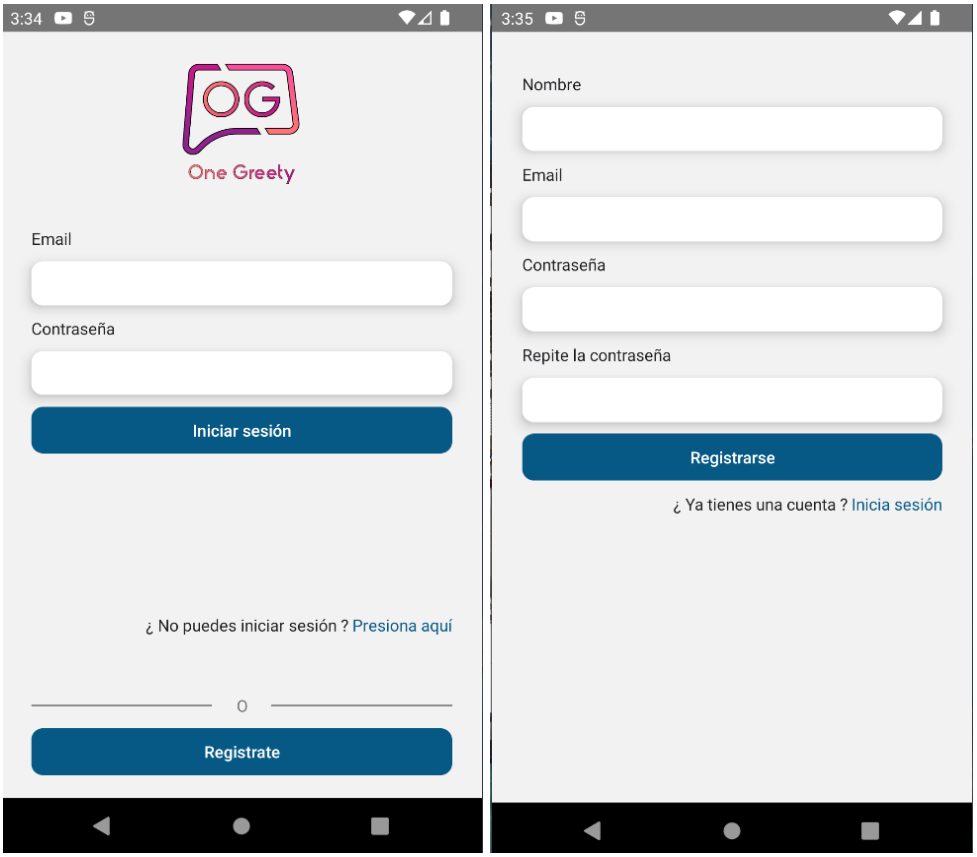

Figura 3.24: Pantalla de inicio de sesión y registro de usuarios

En esta pantalla se presentan los formularios de inicio de sesión y de registro respectivamente. Si el usuario inicia sesión, se redirigirá a la pantalla principal del listado de clubes, por otra parte si el usuario quiere registrarse deberá rellenar los campos y apretar el botón “registrarse”, de esta forma, recibirá un correo electrónico con el que puede activar su cuenta y poder iniciar sesión.

3.4.5 Club individual

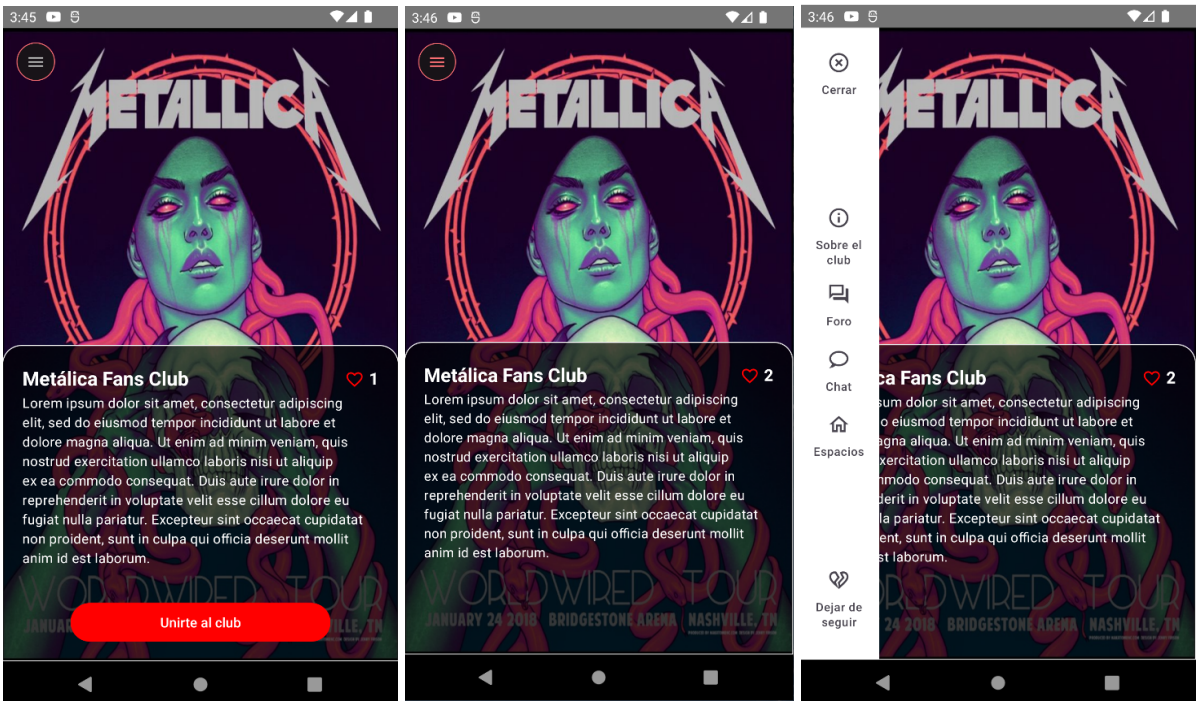

Figura 3.25: Pantalla de club seleccionado

Al seleccionar un club aparecerá esta pantalla donde se puede observar su descripción y el número de seguidores. Además hay un botón que permite al usuario poder unirse al club en el caso de que sea público o enviar una petición para unirse al club si este está como privado. Por otra parte, si el usuario ya pertenece al club, se habilita el menú lateral el cual se puede acceder desde el botón que se encuentra en la parte superior izquierda, en este menú se puede acceder a los servicios disponibles del club como el foro y el chat, también tiene la opción de dejar de seguir el club o regresar a la lista de clubes principal.

3.4.6 Foro

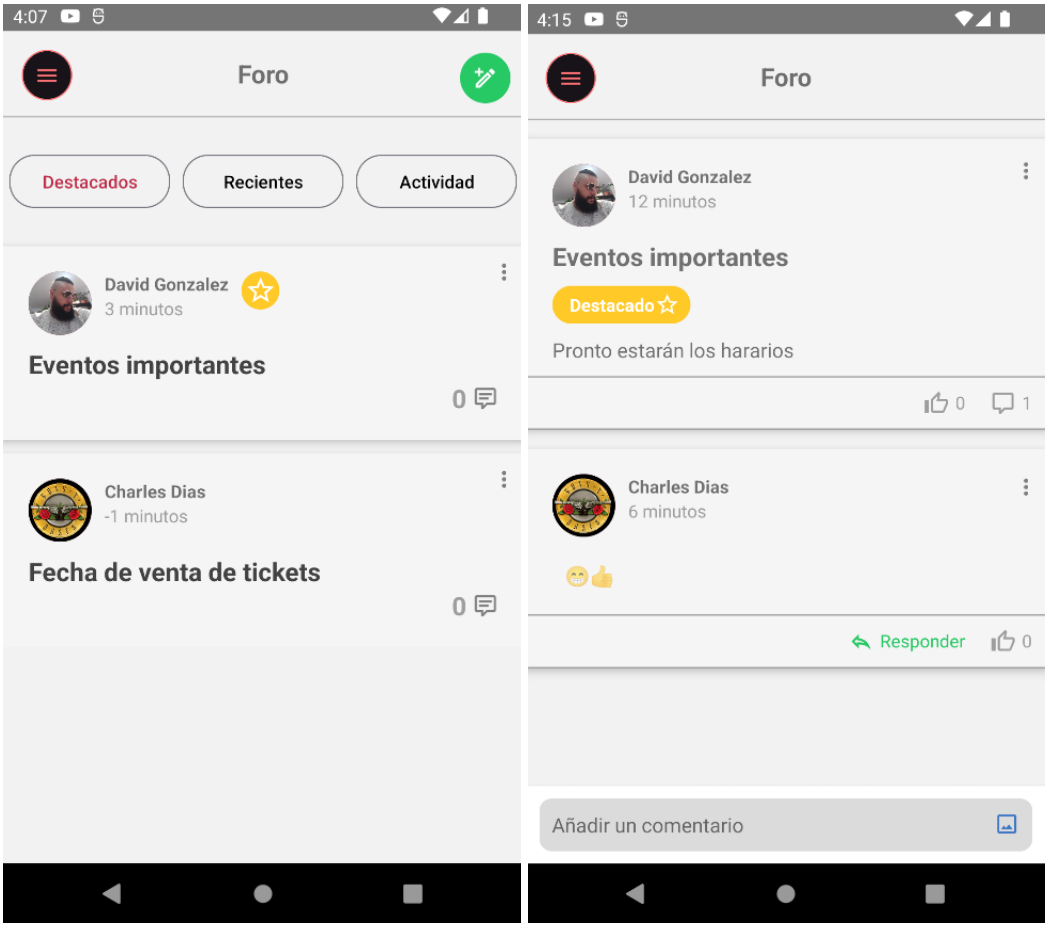

Figura 3.26: Pantallas del Foro

Cada club tiene su foro independiente, al acceder primero se mostrarán todos los posts creados los cuales se pueden filtrar por destacados, creados recientemente o de mayor actividad. En el foro se pueden subir imágenes, interactuar en los posts, crear nuevos o editarlos si se tienen los permisos necesarios.

3.4.7 Chat

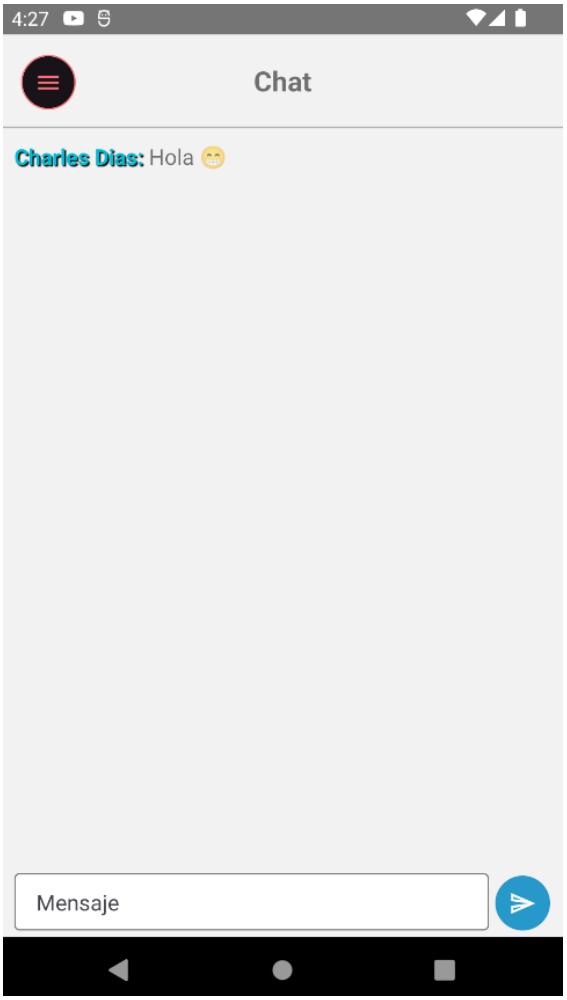

Figura 3.27: Pantalla de chat

Cada club tiene un chat individual en donde todos los miembros pueden participar. Cada persona que entra en la sala de chat se le proporciona un codigo de color con el que se dibujara su nombre en el chat, de esta forma sera mas facil ubicar sus mensajes en el caso de que haya mucha actividad. Además es posible acceder mediante el nombre de cada usuario a su perfil público.

3.4.8 Perfil público

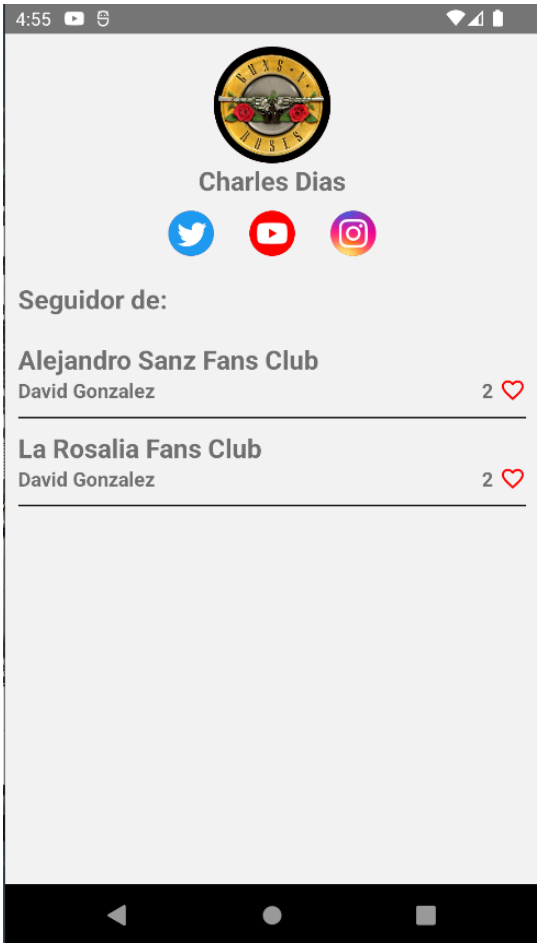

Figura 3.28: Pantalla de perfil de usuario público

Cuando un usuario toque en la foto de perfil o nombre de otro usuario, entrará en esta pantalla donde se puede ver información de dicho usuario como lo son sus redes sociales y los clubes que sigue.

No está de más recordar que el código detallado para la creación de todos estos componentes se encuentra en el repositorio de GitHub de la aplicación móvil(23).

3.5 Arquitectura del software

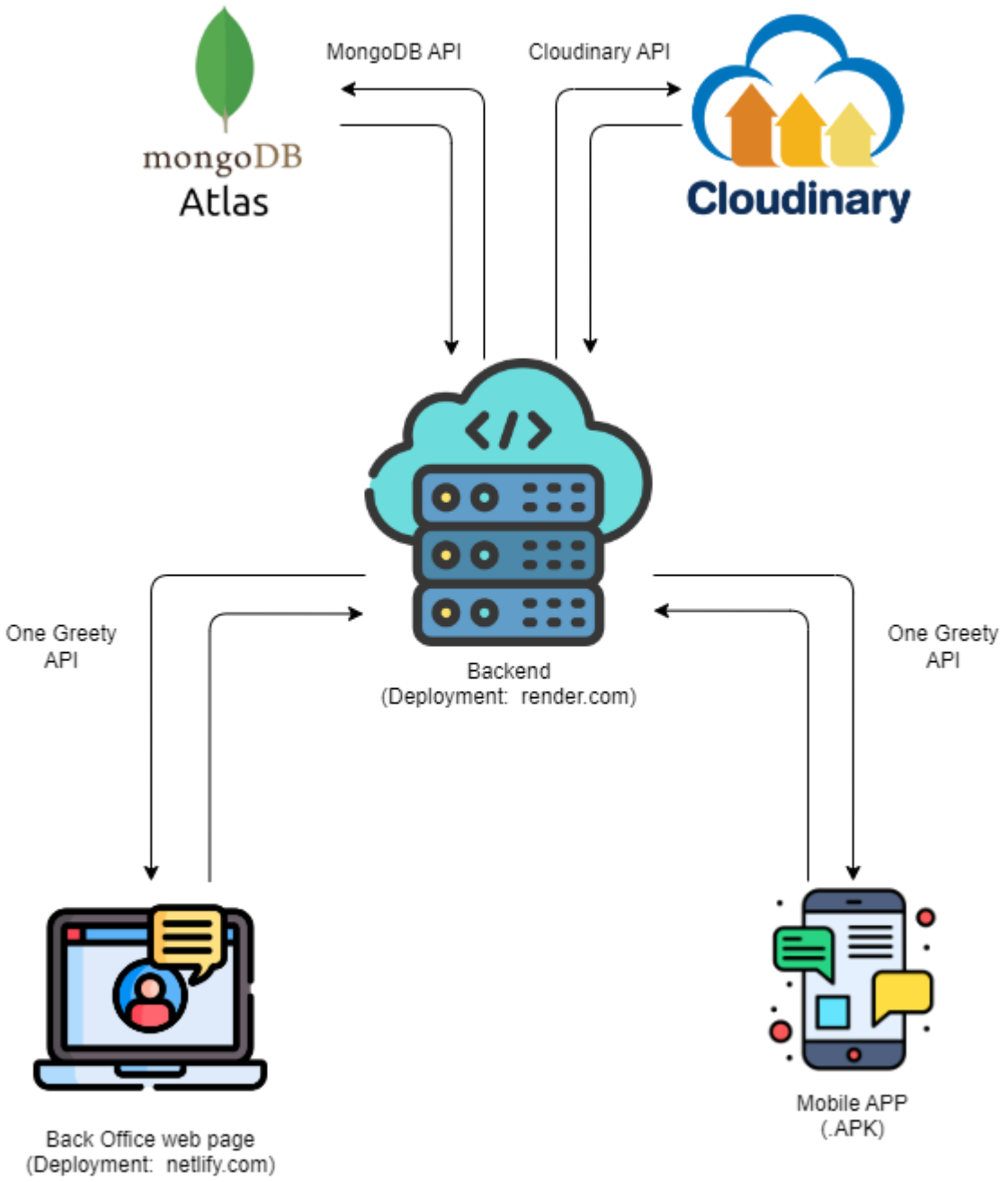

Figura 3.29: Diagrama de la arquitectura de software

En el diagrama anterior se muestra como es la arquitectura de software utilizada en este proyecto. En ella se logra observar que tanto la aplicación móvil como el back office se comunican con el backend mediante la API REST desarrollada con node y express. Por otra parte, también se observa que el backend se comunica con la base de datos de MongoDB Atlas y Cloudinary con sus respectivas API's, haciendo uso de las dependencias mongoose y cloudinary respectivamente.

Capítulo 4
Despliegue de las aplicaciones

4.1 Backend

Para el despliegue del servidor que conforma el backend utilice la plataforma y los servicios que ofrece render.com(24).

Render es una plataforma de despliegue en la nube que ofrece una solución completa para alojar y escalar aplicaciones web y servicios en línea. Render se enfoca en proporcionar una experiencia de desarrollo y despliegue simplificada y automatizada para que los desarrolladores puedan centrarse en la creación de sus aplicaciones, además ofrece un servicio de alojamiento gratuito con algunas limitaciones, pero sus características son más que suficientes para que el servidor pueda funcionar correctamente.

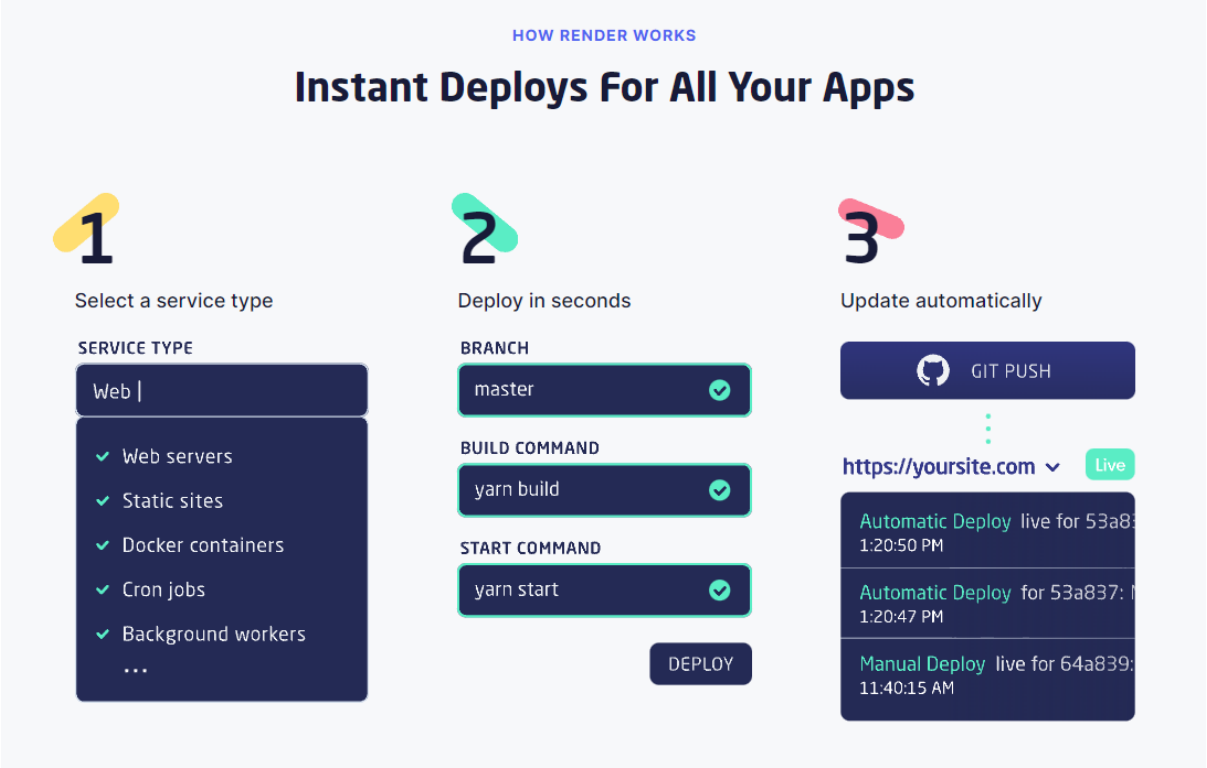

Figura 4.1: Pasos de despliegue en render.com

Los pasos a seguir son sumamente sencillos:

1. Seleccionar el tipo de servicio que necesitas, en este caso es un web server.

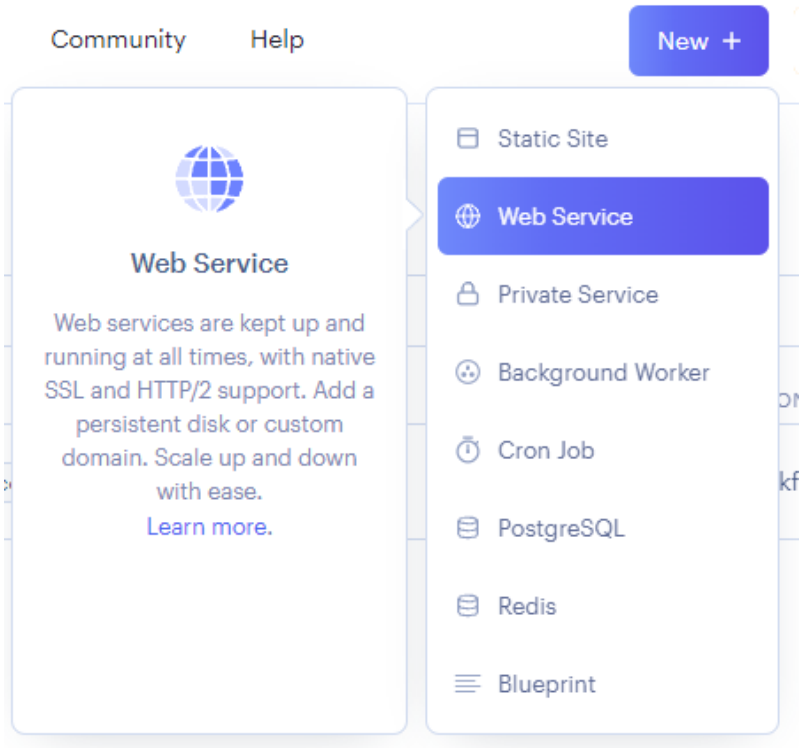

Figura 4.2: Selección de servicio en render.com

2. Vinculas tu cuenta de GitHub o GitLab y seleccionas el repositorio donde tengas el servidor del backend.

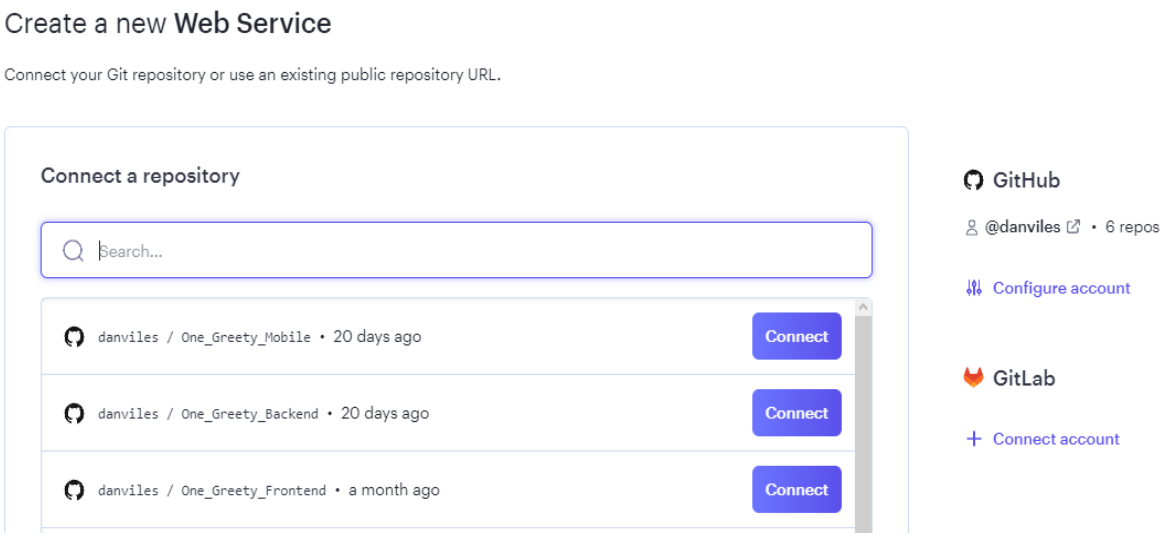

Figura 4.3: Selección de repositorio en render.com

3. Se definen las variables de entorno.

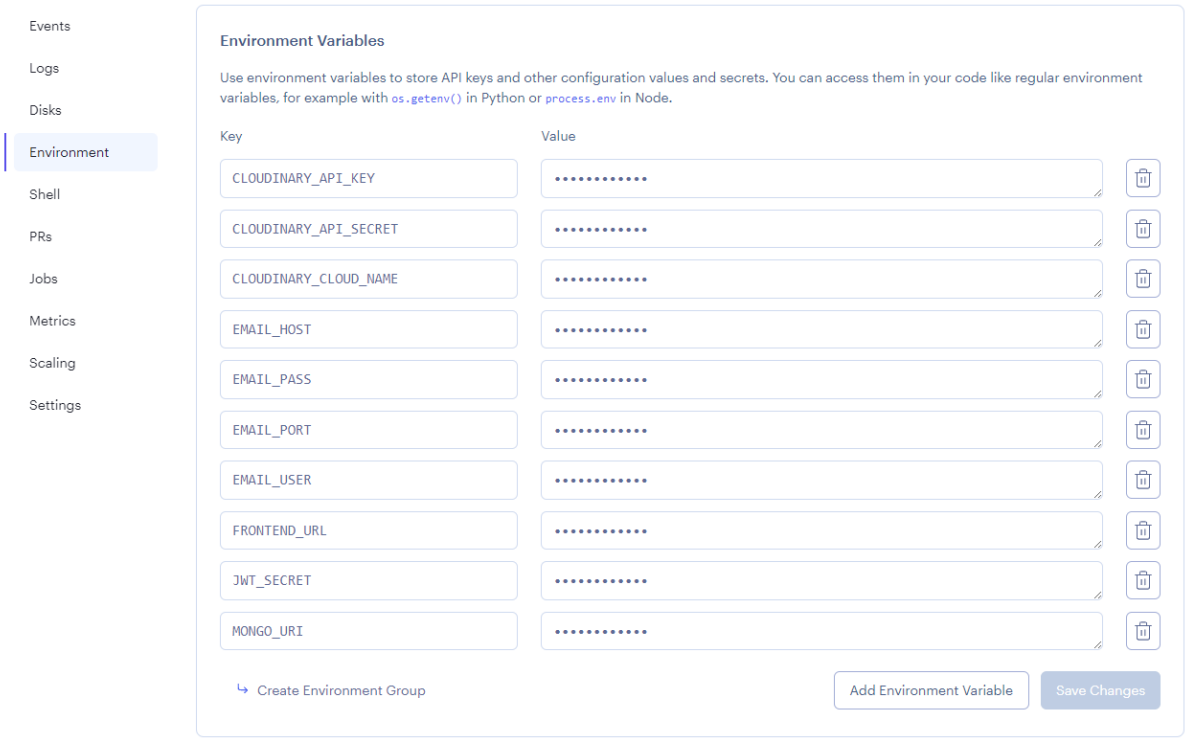

Figura 4.4: Configuración de variables de entorno en render.com

4. Se definen los comandos comandos necesarios para la ejecución del servidor. (npm install, npm build y node index.js)

Figura 4.5: Configuración de comandos en render.com

Una vez hecho los pasos anteriores, solo queda esperar a que la plataforma de render.com termine de desplegar el backend seleccionado del repositorio de Github. Se puede monitorear el proceso en la pestaña Logs la cual nos mostrará la consola del servidor y los errores si en algún momento se llega a presentar uno.

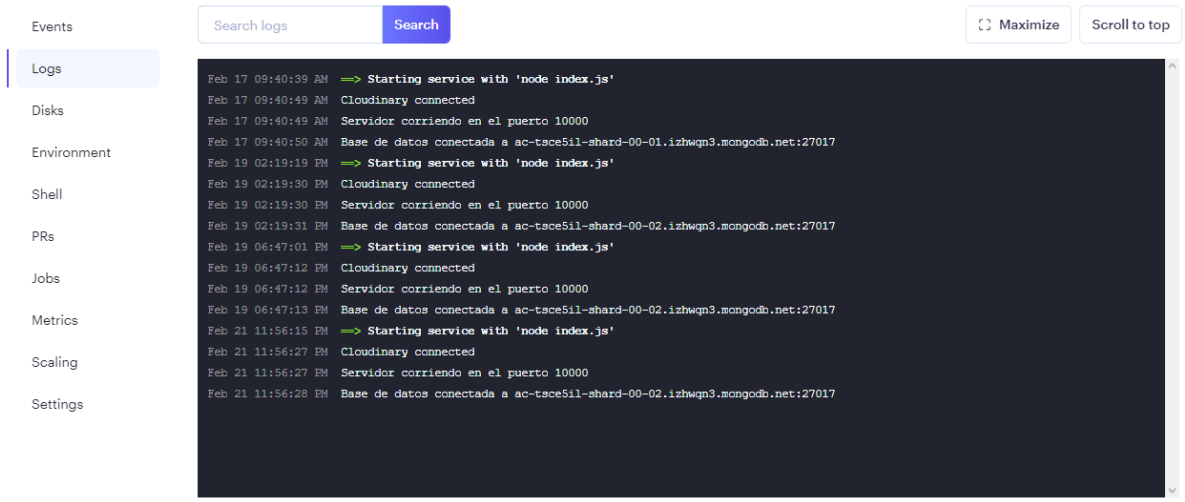

Figura 4.6: Consola en render.com

La dirección de donde se encuentra alojado el servidor siempre se muestra visible en la parte superior del menú.

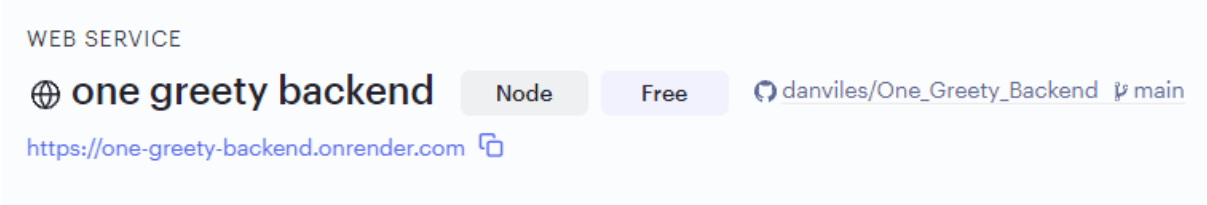

Figura 4.7: Link de alojamiento del servidor en render.com

Esta dirección se define posteriormente en las variables de entorno del frontend (tanto en la app móvil como en el back office web) para que ahora las aplicaciones accedan al servidor de la nube de render.com.

4.2 Frontend Web (Back Office)

Para el despliegue de la aplicación web, se utilizó la plataforma de Netlify(25), una solución de alojamiento en la nube que proporciona una manera fácil y rápida de alojar y publicar sitios web estáticos y dinámicos. Netlify ofrece una amplia gama de herramientas y servicios integrados para automatizar el proceso de construcción, prueba y despliegue de la aplicación web, lo que permite a los desarrolladores centrarse en la creación de su código en lugar de en la gestión de la infraestructura.

Al igual que Render, esta también posee un servicio gratuito cuyas características son suficientes para que la aplicación web pueda funcionar correctamente.

Los pasos a seguir para el despliegue de la aplicación son muy similares a los pasos anteriores con Render:

1. Registrarse en la plataforma asociando una cuenta de Github, GitLab o Bitbucket.
2. Crear un nuevo sitio y seleccionar “Import an existing project”

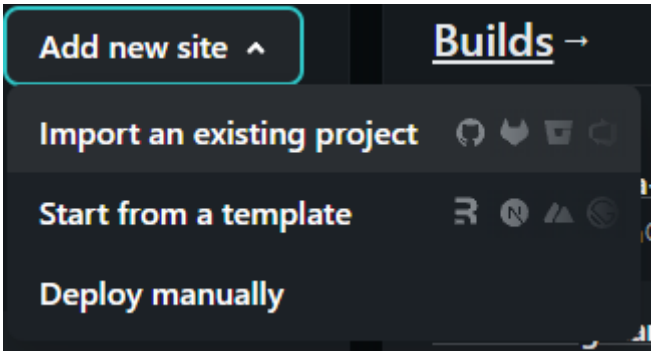

Figura 4.8: Importación de proyecto en netlify

3. Vincular una cuenta de un proveedor Git, en mi caso GitHub, y seleccionar el repositorio de la aplicación web.

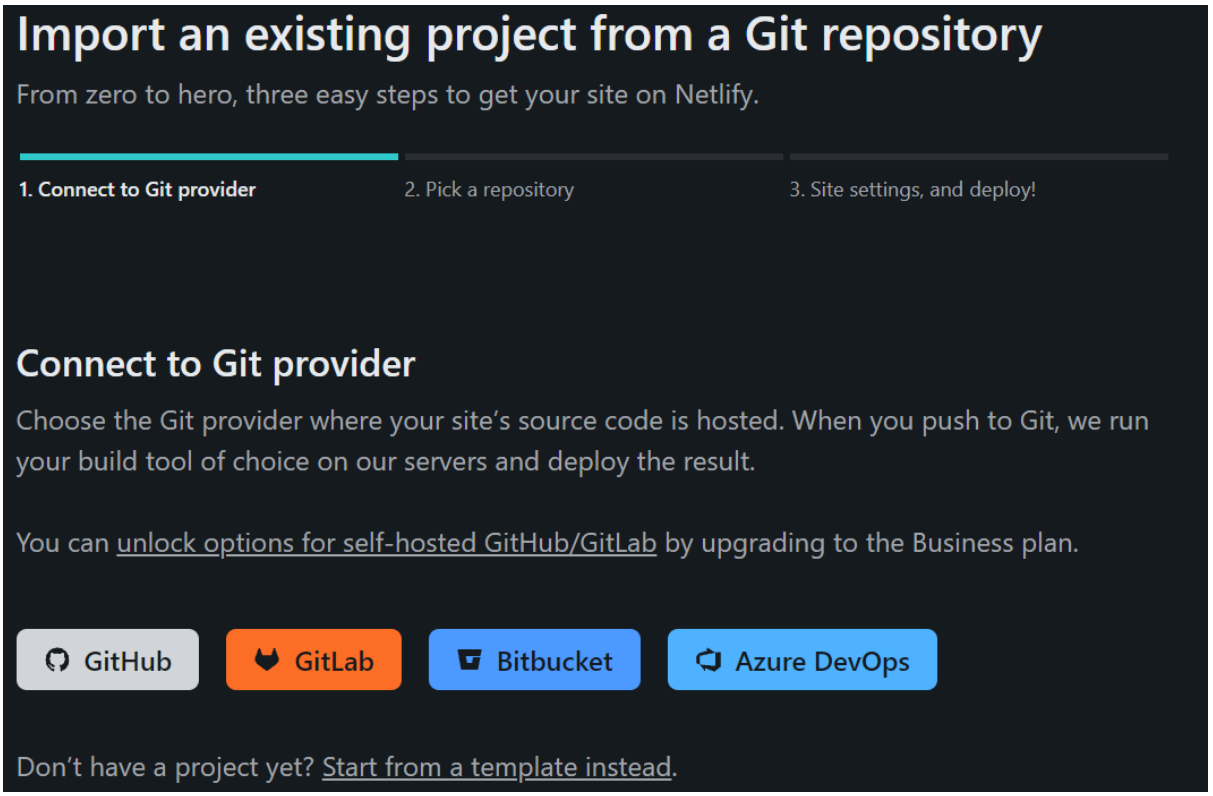

Figura 4.9: Vinculación de cuenta con github en netlify

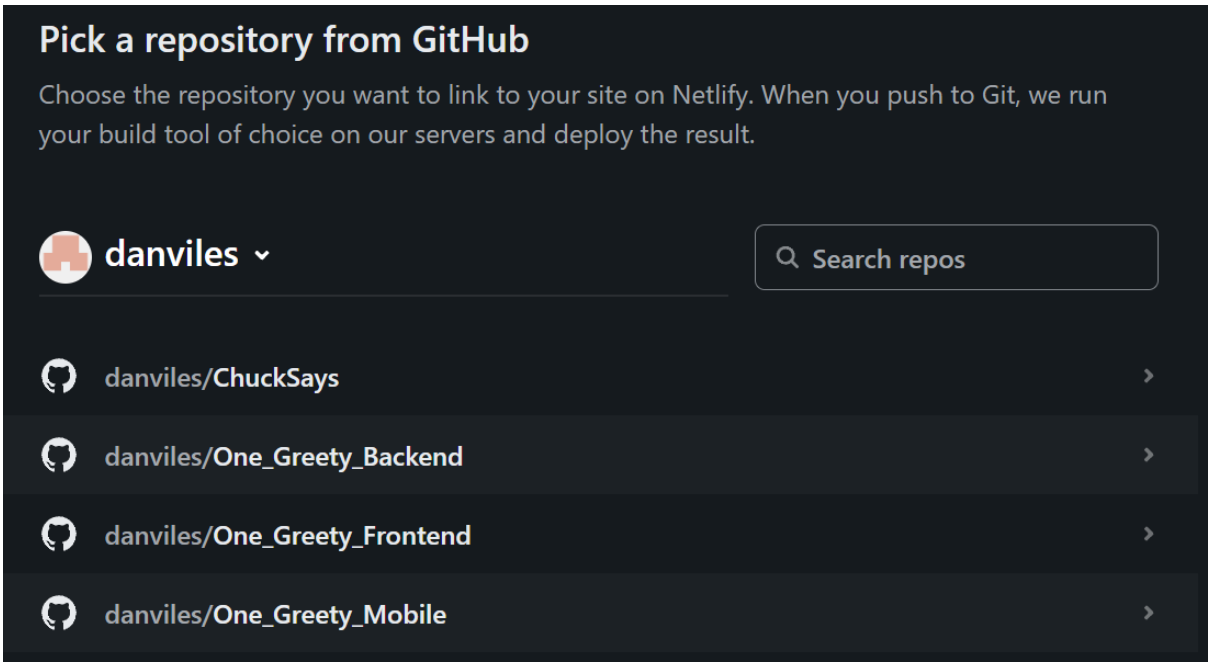

Figura 4.10: Selección de repositorio en netlify

4. Definir variables de entorno y comandos de instalación.

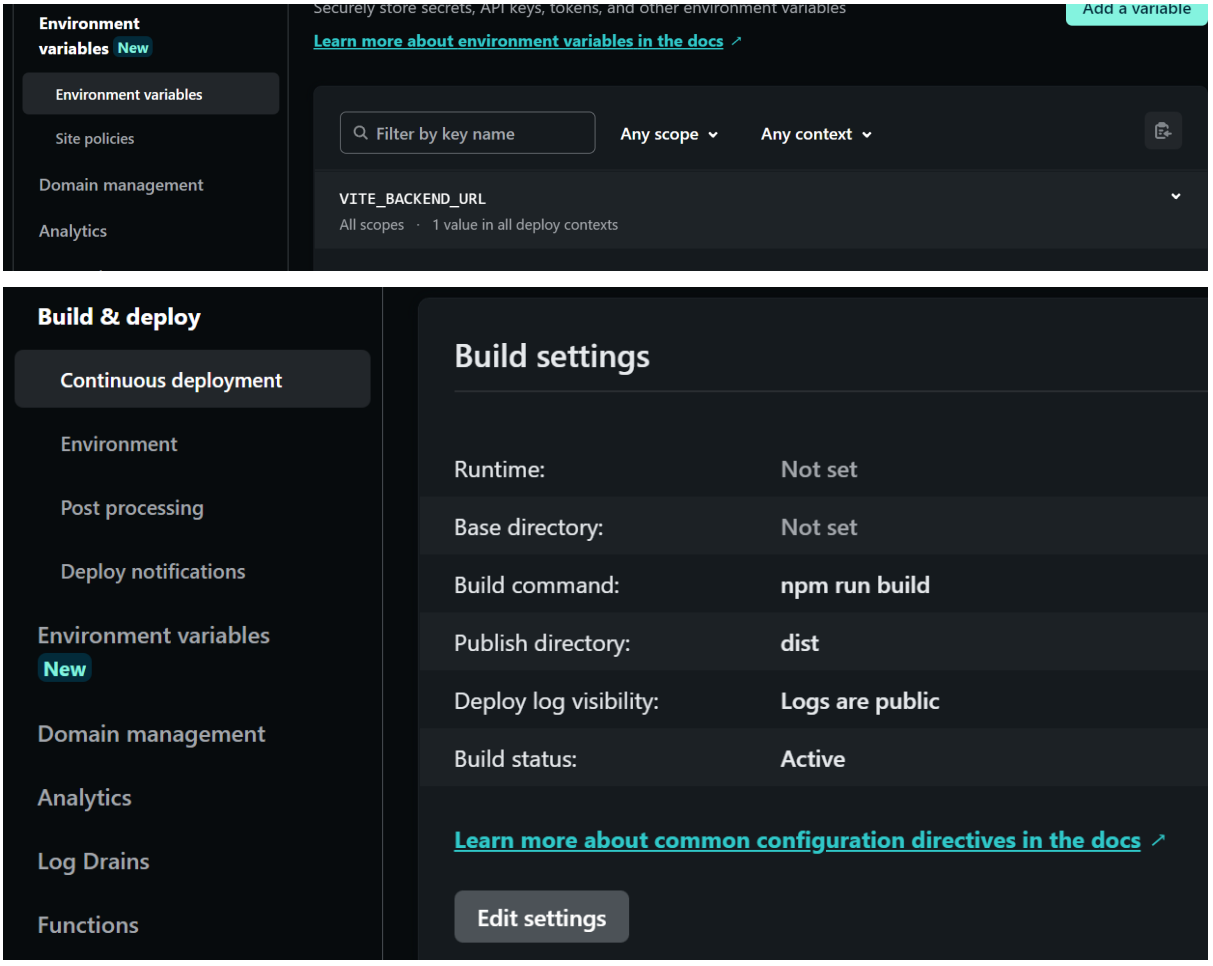

Figura 4.11: Configuración de variables de entorno y comandos en netlify

Finalmente, la dirección web en donde está alojada la aplicación se muestra en la parte superior de la página principal del proyecto:

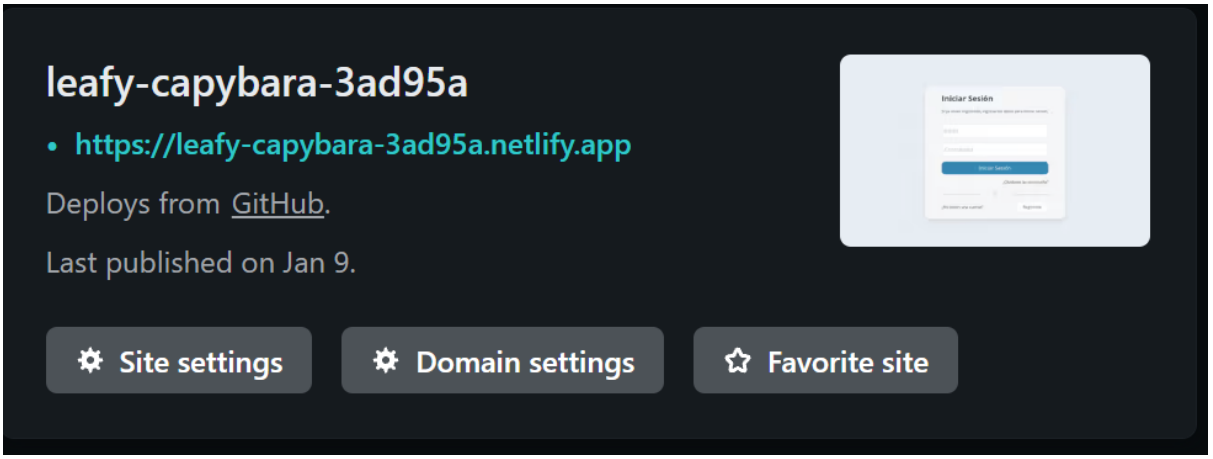

Figura 4.12: Link de alojamiento de la aplicación web en netlify

4.3 Frontend Mobile

El despliegue de la aplicación para móviles consta únicamente de la generación automática de un fichero .apk firmado mediante el uso de un workflow de github que se activa cada vez que el repositorio recibe una petición push, seguidamente, dicha apk es descargada e instalada en los dispositivos móviles.

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

En este proyecto se ha desarrollado una plataforma llamada One Greety, con el objetivo de llevar los servicios que ofrecen las páginas dedicadas a clubs de fans a una única aplicación móvil, intentando reinventar la forma en cómo los artistas y sus fans pueden interactuar brindando una experiencia única en un entorno controlado por administradores y colaboradores mediante un servicio web exclusivo para ellos.

Para el desarrollo del proyecto se utilizaron tecnologías de última generación, como Nodejs y Express para la creación de una API REST, MongoDB como base de datos, JavaScript y React para la aplicación web de gestión y React Native para la

aplicación móvil destinada al uso de los usuarios. El uso de estas tecnologías permitió el desarrollo de una plataforma robusta y escalable.

Este proyecto no solo me permitió poner en práctica mis conocimientos, sino que también me permitió aprender nuevas tecnologías y herramientas que sin duda serán de gran utilidad en mi futuro profesional.

5.2 Líneas Futuras

Considero que existen varias líneas futuras que podrían ampliar este proyecto y mejorar su funcionalidad. Una de ellas podría ser que la plataforma web tenga un apartado que permita interactuar directamente en los servicios futuros y actuales como el foro, permitiendo crear desde el back office post nuevos y administrar los post de los usuarios desde dicha plataforma.

Servicios como un sistema de pago que permita a los artistas vender sus productos (merchandising, entradas a eventos, etc.) directamente a sus fans o herramientas de gamificación, como la creación de concursos o desafíos para los usuarios que interactúan más activamente en los espacios serían buenos ejemplos de desarrollo para las siguientes versiones.

Capítulo 6
Summary and Conclusions

6.1 Conclusions

In this project, a platform called One Greety has been developed with the aim of bringing the services offered by fan club websites to a single mobile application, attempting to reinvent the way in which artists and their fans can interact by providing a unique experience in an environment controlled by administrators and collaborators through an exclusive web service for them.

For the development of the project, cutting-edge technologies were used, such as Nodejs and Express for the creation of a REST API, MongoDB as a database, JavaScript and React for the web management application, and React Native for the mobile application intended for user use. The use of these technologies allowed the development of a robust and scalable platform.

This project not only allowed me to put my knowledge into practice, but also allowed me to learn new technologies and tools that will undoubtedly be of great use in my future professional career.

6.2 Future Work

Consider that there are several future lines that could expand this project and improve its functionality. One of them could be to add a section on the web platform that allows direct interaction with current and future services, such as the forum, allowing the administrator to create new posts and manage user posts from the back office.

Services such as a payment system that allows artists to sell their products (merchandise, event tickets, etc.) directly to their fans, or gamification tools such as creating contests or challenges for users who interact more actively in the spaces would be good examples of development for future versions.

Capítulo 7
Presupuesto

7.1 Desarrollo del software

En la siguiente tabla se muestran las horas invertidas por cada una de las etapas de desarrollo, tomando en cuenta un valor de 18€/h.

Etapa	Planificación y Diseño (h)	Desarrollo (h)	Coste (€)
Prototipo Apps	16	0	288
Base de datos	16	28	792
Backend	8	112	2160
Frontend Web	2	32	612
Frontend Mobile	2	56	1044
Total	44	228	4896

Tabla 1: Coste de desarrollo del software.

7.2 Herramientas, despliegue y mantenimiento.

A continuación se muestra una tabla con los costes de las herramientas, servicios de despliegue y mantenimiento.

Nombre	Coste (€)	Coste de los servicios una vez superado las limitaciones que ofrecen (€/mes)
GitHub	0	0
Visual Studio Code	0	0
Netlify	0	18
Render	0	18
MongoDB Atlas	0	54
Cloudinary	0	84

Tabla 2: Coste de herramientas, despliegue y mantenimiento.

7.3 Coste total

Tipos	Coste plano (€)	Coste mensual de los servicios una vez superado las limitaciones que ofrecen (€)
Desarrollo del Software	4896	
Recursos, herramientas y mantenimiento		174

Tabla 3: Coste total.

Bibliografía

[1] Alvaro Osorio. “Escuela de artistas Gofar”. https://escuelagofar.com/club-de-fans/ (accessed Feb. 20, 2023)

[2] Juan Carlos. “Apps de Club de Fans | 8 Motivos para Usarlas”. https://promocionmusical.es/apps-club-fans-8-motivos-para-usarlas/ (accessed Feb. 20, 2023)

[3] Microsoft. Visual Studio Code - Code Editing. Redefined. (2023). https://code.visualstudio.com/. (accessed Feb. 20, 2023)

[4] GitHub. (2021). GitHub: Where the world builds software. Retrieved September 1, 2021, from https://github.com/ (accessed Feb. 20, 2023)

[5] MongoDB. (2021). MongoDB. https://www.mongodb.com/ (accessed Feb. 20, 2023)

[6] Node.js Foundation. (2021). Node.js. Retrieved from https://nodejs.org/ (accessed Feb. 20, 2023)

[7] Express. (2021). Express - Node.js web application framework. Retrieved from https://expressjs.com/ (accessed Feb. 20, 2023)

[8] Cloudinary. (2021). Cloudinary - Image and Video Upload, Storage, Optimization and CDN. Retrieved from https://cloudinary.com/ (accessed Feb. 20, 2023)

[9] Socket.IO. (2021). Socket.IO - Realtime application framework (Node.JS server). Retrieved from https://socket.io/ (accessed Feb. 20, 2023)

[10] Facebook. (2021). React - A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/ (accessed Feb. 20, 2023)

[11] Facebook. (2021). React Native - Build native mobile apps using JavaScript and React. Retrieved from https://reactnative.dev/ (accessed Feb. 20, 2023)

[12] MongoDB Atlas. (2022). MongoDB Atlas. https://www.mongodb.com/cloud/atlas (accessed Feb. 20, 2023)

[13] MongoDB Compass. (2022). MongoDB Compass. https://www.mongodb.com/products/compass (accessed Feb. 20, 2023)

[14] Amazon. “¿Que es una API RESTful?” https://aws.amazon.com/es/what-is/restful-api/ (accessed Feb. 20, 2023)

[15] Fernando Palacios. “Middlewares en nodeJS, el camino para jedis backend” https://eldevsin.site/los-middlewares-en-nodejs/#:~:text=Básicamente%20un%20middleware%20es%20una,de%20vida%20de%20la%20petición. (accessed Feb. 20, 2023)

[16] Elvis Nogueiras. One Greety Backend. https://github.com/danviles/One_Greety_Backend (accessed Feb. 20, 2023)

[17] Mozilla. MDN contributors. Control de acceso HTTP (CORS) https://developer.mozilla.org/es/docs/Web/HTTP/CORS (accessed Feb. 20, 2023)

[18] Facebook. (2021). React Context. Retrieved from https://reactjs.org/docs/context.html (accessed Feb. 20, 2023)

[19] Evan You & Vite Contributors. Vite. https://vitejs.dev (accessed Feb. 20, 2023)

[20] Elvis Nogueiras. One Greety Frontend. https://github.com/danviles/One_Greety_Frontend (accessed Feb. 20, 2023)

[21] Expo. Metro bundler. https://docs.expo.dev/guides/customizing-metro/ (accessed Feb. 20, 2023)

[22] Google. (2021). Android Studio. https://developer.android.com/studio (accessed Feb. 20, 2023)

[23] Elvis Nogueiras. One Greety Mobile. https://github.com/danviles/One_Greety_Mobile (accessed Feb. 20, 2023)

[24] Render. (2023). Render: Cloud Platform for Modern Apps. https://render.com/ (accessed Feb. 20, 2023)

[25] Netlify. (2023). Netlify: All-in-one platform for automating modern web projects. https://www.netlify.com/ (accessed Feb. 20, 2023)