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

*Development on mobile devices and fullstack*

José Daniel Fuentes Marra

La Laguna, 5 de julio de 2022

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

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

Que la presente memoria titulada:

*"Desarrollo en dispositivos móviles y fullstack"*

ha sido realizada bajo su dirección por D. **José Daniel Fuentes Marra**, con N.I.F. 79074267-Z.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de julio de 2022

# Agradecimientos

En primer lugar, quiero agradecer a toda mi familia y amigos por todo el apoyo que me han dado a lo largo de estos años, así como a algunos profesores de las dos universidades en las que he cursado esta carrera por fomentar en mi la curiosidad y la pasión por la programación que me ha convertido en el profesional que soy hoy.

# Licencia

**Resumen**

*El objetivo de este trabajo ha sido el diseño y desarrollo de una aplicación móvil enfocada en el crowdfunding el cual es un método de financiación colectiva mediante donaciones o preventas de algún producto. En este caso se realiza mediante la venta de boletos para conciertos de artistas que están empezando en el mundo de la música y por lo tanto no tienen los recursos necesarios para permitirse el coste de este tipo de eventos.*

*La aplicación se llama "FundMyMusic" y ofrece una web dedicada solo para los artistas donde podrán crearse una cuenta y desde ella poder configurar, publicar y supervisar los conciertos que estén dispuestos a realizar. Por otro lado, la aplicación móvil para Android ofrece a los usuarios la posibilidad de comprar los boletos para los conciertos que estén publicados en la aplicación y de esta forma estarán ayudando al artista a obtener los fondos necesarios para realizar ese concierto y también poco a poco ir dándose a conocer para así llegar a más gente he ir creciendo en el mundo de la música.*

*Para el correcto funcionamiento de la aplicación se han implementado los siguientes componentes:*

- *Frontend Web – React*
- *Frontend Mobile – React Native*
- *Backend – NodeJS y Express*
- *Base de Datos – MongoDB*

**Palabras clave:** Aplicación, Android, Crowdfunding, Base de datos, Frontend, Backend.

**Abstract**

*The objective of this work has been the design and development of a mobile application focused on crowdfunding, which is a method of collective financing through donations or pre-sales of a product. In this case, it is made through selling tickets for artists' concerts that are just starting out in the world of music and therefore they do not have the necessary resources to afford the cost of this type of event.*

*The application is called "FundMyMusic" and it offers a website dedicated only to artists in which they can create an account and through it be able to configure, publish and supervise the concerts that they are willing to make. On the other hand, the mobile application for Android offers users the possibility of buying tickets for the concerts that are published in the application and in this way, they will be helping the artist to obtain the necessary funds to carry out that concert and also helping them step by step to reach more people and to grow in the world of music.*

*For the proper working of the application, the following components have been implemented components:*

- *Frontend Web – React*
- *Frontend Mobile – React Native*
- *Backend – NodeJS and Express*
- *Data Base – MongoDB*

**Keywords:** Application, Android, Crowdfunding, Data Base, Frontend, Backend.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1
# Introducción

## 1.1 Antecedentes y estado actual del tema

El crowdfunding es un término inglés que hace referencia a un tipo de financiación colectiva (también llamado micromecenazgo), generalmente realizada a través de plataformas online, en la que a través de pequeñas aportaciones se financia un determinado proyecto o iniciativa.

Para un artista que se encuentra empezando su carrera musical puede llegar a ser muy difícil encontrar formas de financiar sus proyectos por lo que el crowdfunding puede ser la solución perfecta a este problema. Con este tipo de plataformas es posible ayudar al crecimiento de estos artistas noveles pudiendo llevar a cabo proyectos que de otra forma nunca hubiesen sido posible sin la ayuda de un productor, patrocinadores o de un banco.

Existen varios tipos de crowdfunding, en primer lugar, tenemos el crowdfunding de recompensas que es una colecta de dinero para un proyecto, por ejemplo, la grabación de un disco o la realización de un concierto. Los participantes reciben a cambio una pequeña compensación como por ejemplo el ejemplar del disco, que la banda incluya tu nombre en el libreto del disco o asistiendo a parte de la grabación del disco, etc.

En segundo lugar, tenemos el crowdfunding de donaciones que es una colecta de dinero para una causa, por ejemplo, humanitaria.

En tercer lugar, el crowdfunding de capital que consiste en recoger dinero y con ese capital fundar una empresa. Para ello el emprendedor inicia una campaña y la gente que confía en su negocio, invierte y si tiene éxito, se convierten en accionistas.

Y por último el crowdfunding de préstamos: Se recoge dinero, pero en este caso es como un préstamo. El participante espera que le devuelvan ese dinero. A esto se le conoce como crowdlending.

Uno de los pioneros del crowdfunding en la industria de la música ha sido el grupo británico de rock “Marillion”. En 1997, los fans estadounidenses financiaron su gira por EE.UU, que costó 60.000 dólares, mediante donaciones que consiguieron a través de internet. Y en España el grupo “Extremoduro’’ financió su primer disco en 1989 mediante crowdfunding, recaudando 250.000 pesetas que les permitió ir a Madrid para grabarlo.

Actualmente existen varias plataformas de crowdfunding como por ejemplo Kickstarter(1), Indiegogo(2) o KissKissBankBank(3) en los cuales un artista podría publicar su concierto e iniciar una campaña de crowdfunding pero estas plataformas no se centran en la música, están diseñadas para cualquier tipo de proyecto como por ejemplo el desarrollo de un video juego o el lanzamiento de un libro lo que hace que existan demasiadas categorías y la música no sea el plano principal o no se le haga demasiada publicidad en estas plataformas perdiendo publico y no

pudiendo llegar a tener el impacto que podría necesitar el artista. Y es en esto en lo que se centra este proyecto, una plataforma enfocada únicamente al mundo musical donde los artistas tengan la visibilidad que necesitan y se puedan concentrar en una sola plataforma todo el publico interesado en la música pudiendo así aumentar el éxito que podrían tener estos artistas noveles.

## 1.2 Objetivo

El objetivo de este trabajo ha sido el diseño y desarrollo de una aplicación móvil enfocada en el crowdfunding para artistas noveles con el fin de ofrecer una plataforma donde se busca ayudar e impulsar el crecimiento de dichos artistas en el mundo de la música dando a conocer su talento a miles de personas las cuales pueden ayudar con su dinero a financiar los conciertos y eventos de estos artistas haciendo crecer la popularidad de estos y por consiguiente poder evolucionar y crecer su carrera como artista musical.

## 1.3 Fases de desarrollo

Para el desarrollo de este trabajo se ha seguido el siguiente flujo de trabajo:

- Realizar un diseño inicial o mockup.
    - Documentación y análisis de este tipo de plataformas
    - Diseño y vistas de la aplicación móvil
    - Diseño y vistas de la aplicación web
- Desarrollar y configurar la base de datos.
    - Análisis de los datos que se necesitan almacenar
    - Encontrar la mejor forma de relacionarlos
    - Creación del modelo para las tablas
- Desarrollar una API REST capaz de gestionar la interacción y comunicación con la base de datos.
    - Documentación y configuración
    - Configuración del servidor Backend
    - Configuración de las rutas para las peticiones
    - Configuración de los controladores que interactúan con los modelos de la base de datos
- Desarrollar la aplicación web
    - Documentación y configuración
    - Implementación de formularios y menús
    - Implementación de conexiones con el Backend
    - Muestreo de datos
- Desarrollar la aplicación móvil

    - Documentación y configuración
    - Implementación de formularios y menús
    - Implementación de conexiones con el Backend
    - Muestreo de datos
- Redacción de Memoria (Tarea paralela a las anteriores)
    - Retoques y correcciones

## 1.4 Estructura del documento

- **Capítulo 2** – Tecnologías

  Explicación detallada de todas las tecnologías utilizadas para el desarrollo de este proyecto.

- **Capítulo 3** – Desarrollo

  Descripción y explicación de todos los pasos realizados en orden cronológico de los 4 grandes apartados en los que se dividen este capítulo: Base de datos, Backend, Frontend Web y Frontend Mobile.

- **Capítulo 4** – Despliegue

  Descripción de cómo se configuró el despliegue de los distintos módulos de este proyecto.

- **Capítulo 5** – Conclusiones y líneas futuras

  Resultados obtenidos luego del desarrollo de este proyecto, así como mejoras que se pueden implementar en el futuro para mejorar la experiencia de usuario.

- **Capítulo 6** – Summary and Conclusions

  Capitulo anterior adaptado en ingles

- **Capítulo 7** – Presupuesto

  Contiene todos los costes del proyecto tanto de desarrollo como de despliegue y alojamiento.

# Capítulo 2
# Tecnologías

## 2.1 React

React(6) o React.js es una biblioteca Javascript de código abierto que se utiliza para crear interfaces de usuario y desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre.

Está basado en el uso de componentes y es muy útil para el desarrollo de aplicaciones que usan datos que cambian todo el tiempo. React es la Vista en un contexto en el que se use el patrón MVC (Modelo-Vista-Controlador) y también puede ser utilizado con las extensiones de React-based que se encargan de las partes no-UI (que no forman parte de la interfaz de usuario) de una aplicación web.

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.

## 2.2 React Native

React Native(7) es un framework JavaScript para crear aplicaciones nativas para iOS y Android, basado en la librearía React antes mencionada pero en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas, en este caso iOS y Andorid. Es decir, en lugar de desarrollar una aplicación web híbrida o en HTML5, lo que se obtiene al final como resultado es una aplicación real nativa, indistinguible de la que podrías desarrollar con tu código en Objective-C o Java.

Las características más importantes de React Native son:

- Compatibilidad Cross-Platform

- Funcionalidad nativa
- Actualizaciones instantáneas (para desarrollo y/o test)

## 2.3 NodeJS y Express

NodeJS(8) es un entorno que trabaja en tiempo de ejecución, multi-plataforma, de código abierto, que permite crear toda clase de herramientas del lado del servidor y aplicaciones en JavaScript. Añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros.

Express es el framework web más popular de NodeJS y nos proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL.
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto.

## 2.4 MongoDB

MongoDB(9) es un sistema de base de datos NoSQL orientado a documentos, de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico.

Las principales características de MongoDB son:

- Indexación
- Replicación
- Balanceo de carga
- Almacenamiento de archivos
- Ejecución de JavaScript del lado del servidor

## 2.5 Cloudinary

Cloudinary(10) es una empresa de tecnología SaaS con sede en Santa Clara, California. La empresa ofrece servicios de gestión de imágenes y vídeos basados en la nube, lo que será muy útil ya que se deben almacenar las imágenes de los conciertos y de esta forma no ocuparan espacio en la base de datos.

Por otro lado, cuenta con una API propia lo que facilita su uso del lado del Backend a la hora de subir, consultar, actualizar y eliminar las imágenes necesarias desde el propio código del Backend.

## 2.6 VS Code

Visual Studio Code(11) es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias.

También dispone de una gran tienda de extensiones gratuitas que son muy útiles a la hora de desarrollar en distintos lenguajes, así como de personalizar y generar sintaxis de lenguajes en específico.

## 2.7 GitHub

Github(12) es una plataforma donde se pueden alojar proyectos utilizando el control de versiones de Git. Permite la automatización de tareas como el despliegue o las pruebas del código y cuenta con una gran cantidad de características como:

- Issues
- Pull requests
- Debates
- Actions
- Wiki
- GitHub Pages

# Capítulo 3
# Desarrollo

## 3.1 Base de Datos

Lo primero ha sido configurar la base de datos en MongoDB Atlas(13) ya que es necesario tener un cliente de MongoDB ejecutándose para poder conectarse a él y probar los modelos que serán implementados más adelante en el Backend.

Para esto se ha creado una cuenta en MongoDB Atlas y luego es necesario crear y configurar un "cluster", esto es una especie de máquina virtual que ofrece MongoDB Atlas donde se estará ejecutando un cliente de MongoDB al cual nos conectaremos desde el código del Backend.

Una vez creado el cluster solo queda crear un usuario con su contraseña dentro de este cluster y darle permisos de lectura y escritura en la base de datos, estas son las credenciales que se usaran más adelante en el código para acceder y gestionar la base de datos.

Luego de tener todo esto configurado ya estaría lista la base de datos para empezar a crear los modelos desde el Backend.

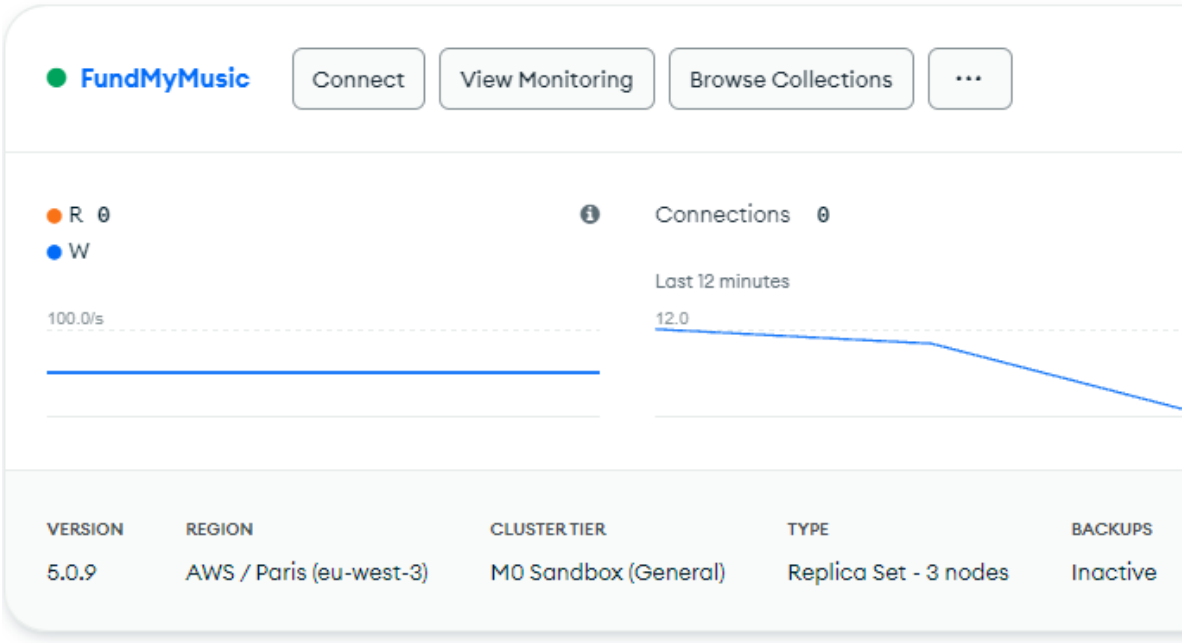



Figura 3.1: Cluster de la base de datos en MondoDB Atlas

## 3.2 Backend

Para empezar con el Backend primero se tuvo que crear un proyecto de Node.JS con el comando “npm init” e instalar todas las dependencias que hacen falta para este proyecto:

- Express

  Es un framework para que crear API’s en Node.js de una forma fácil y sencilla.

- Bcryptjs

  Es una dependencia para encriptar las contraseñas que almacenaremos en la base de datos.

- Bloudinary

  Es una dependencia que nos facilita conectarnos a la API de Cloudinary en el cual se subirán las imágenes de los conciertos para no ocupar espacio en la base de datos.

- Jsonwebtoken

  Es una dependencia para generar tokens que se usarán para el control de sesión de los usuarios.

- Mongoose

  Es una dependencia que nos facilita la conexión y comunicación con la base de datos de MongoDB.

- Multer

  Es un middleware que hace que sea más fácil manipular “multipart/form-data” cuando un usuario sube la imagen de un concierto.

- Nodemailer

  Es una dependencia que nos permite enviar emails desde el código del Backend.

Luego de tener todas estas dependencias instaladas se pasó a configurar algunas conexiones con servicios externos como la base de datos y Cloudinary.

Empezando con la base de datos se ha creado el directorio “src/config” que es donde se almacenaran todas las configuraciones necesarias, luego de esto se crea el archivo “db.js” y haciendo uso de la dependencia de Mongoose hacemos la conexión con la base de datos de la siguiente manera:

```
import mongoose from "mongoose";

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGODB_URI, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log("MongoDB connected");
  } catch (error) {
    console.log("MongoDB connection error");
    console.log(error);
    process.exit(1);
  }
};

export default connectDB;
```

Figura 3.2: Conexión del Backend a la base de datos

Como se puede ver, la variable de entorno MONGODB_URI contiene la dirección, el usuario y la contraseña que antes fue creada en MongoDB Atlas. Todas las variables de entorno que se usan de ahora en adelante están declaradas en ".env" el cual solo sirve para un entorno de desarrollo, cuando se despliegue en algún servicio en la nube, todas estas variables tendrán que ser configuradas en dicho servicio.

Luego de esto se ha pasado a crear el archivo "cloudinary.js" en el cual configuraremos la conexión con el servicio de Cloudinary:

```
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 3.3: Conexión del Backend a Cloudinary

Aquí también se pueden apreciar las variables de entorno utilizadas, estas fueron sacadas de la cuenta de Cloudinary cuando fue creada.

Por último, antes de empezar con las rutas y los modelos, se han configurado dos dependencias más, "jsonwebtoken" para generar los tokens a los usuarios al iniciar sesión y "nodemailer" para enviar los correos necesarios.

Empezando por jsonwebtoken, se creó el siguiente archivo "src/helpers/generateJWT.js" que contiene lo siguiente:

```
import jwt from "jsonwebtoken";

const generateJWT = (user) => {
  const payload = {
    user: {
      id: user.id,
    },
  };
  return jwt.sign(payload, process.env.SECURE_KEY, {
    expiresIn: "15d",
  });
};

export default generateJWT;
```

Figura 3.4: Configuración de los Tokens de inicio de sesión

Como se puede ver es una función que hace uso de la dependencia antes instalada el cual genera un token único pasándole la información del usuario y una clave secreta que ha sido declarada nuevamente como variable de entorno, también se puede ver que el tiempo de duración del token es de 15 días lo cual es aceptable ya que también servirá para la aplicación móvil el cual no es agradable que pida iniciar sesión cada día.

Por otro lado, tenemos la configuración para enviar los correos en "src/helpers.email.js" el cual contiene lo siguiente:

```
import nodemailer from "nodemailer";

export const emailRegister = async (data) => {
  const { email, name, token } = data;

  const transport = nodemailer.createTransport({
    host: process.env.SMTP_HOST,
    port: process.env.SMTP_PORT,
    secure: true,
    auth: {
      user: process.env.SMTP_USER,
      pass: process.env.SMTP_PASS,
    },
  });
```

Figura 3.5: Configuración para Nodemailer

En este caso la dependencia de Nodemailer necesita la configuración de un servicio SMTP para enviar correos por lo que se ha usado el servicio que ofrece Gmail para esto.

Se ha creado una cuenta de Gmail para este proyecto y en las configuraciones de la cuenta se obtienen las credenciales necesarias para hacer uso de su servicio SMTP las cuales nuevamente fueron declaradas como variables de entorno ya que es información sensible.

Una vez configurado todo esto se pasó a codificar las rutas y los modelos necesarios para este proyecto.

Empezando con los modelos y analizando la información que se requiere almacenar para el correcto funcionamiento de este proyecto, se llegó a la conclusión que se necesita almacenar lo siguiente:

Para las cuentas de los usuarios:

- id
- email
- nombre artístico (Solo para las cuentas de artistas)
- nombre
- apellido
- teléfono
- contraseña
- rol

- saldo
- conciertos guardados
- boletos comprados
- token (Esto solo sirve a la hora de reestablecer la contraseña)

Y para almacenar los conciertos:

- artista (id del artista que lo creó)
- titulo
- url de la imagen del concierto
- id de la imagen del concierto
- tamaño de la imagen del concierto
- genero
- lugar
- fecha
- descripción
- capacidad
- entradas mínimas para celebración
- regalo
- número de entradas vendidas
- precio
- status (si se encuentra abierta o cerrada la fecha)
- agotado (Booleano)
-

Con toda esta información se pasó a crear los modelos que en este caso han sido definidos en “src/models/userModel.js” y “src/models/concertModel.js”, dentro de estos archivos se crea un “Schema” de Mongoose el cual sirve para definir la estructura que tendrán los datos en la base de datos de MongoDB. Un pequeño ejemplo de cómo se definen estos esquemas es el siguiente:

```
import mongoose from "mongoose";

const concertSchema = mongoose.Schema(
  {
    artist: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Users",
      required: true,
    },
    title: {
      type: String,
      required: true,
      trim: true,
    },
    FlyerURL: {
      type: String,
      required: true,
    },
```

Figura 3.6: Schema de Mongoose para conciertos

Y al hacer uso de estos schemas y almacenar datos en la base de datos, se puede observar cómo quedan almacenados:

```
_id: ObjectId("62aa5a4167eebcfeac2ac630")
email: "alu0101166247@ull.edu.es"
stageName: ""
name: "Jose Daniel"
surname: "Fuentes Marra"
phone: 683529959
password: "$2a$10$ug3SDAfpU1GoF7J5ALc9DOVyUtQoDY4SuSm6NDa7E0LkKKtwLyBUi"
token: "Confirmed"
confirmed: true
role: "User"
balance: 6416
savedConcerts: Array
  0: ObjectId("62aa555a67eebcfeac2ac604")
  1: ObjectId("62aa4ecb67eebcfeac2ac5b5")
  2: ObjectId("62aa548967eebcfeac2ac5ea")
  3: ObjectId("62aa4c1c67eebcfeac2ac588")
purchasedTickets: Array
  0: Object
    concert: ObjectId("62aa4b4767eebcfeac2ac572")
    quantity: 1
    _id: ObjectId("62aa5d2b67eebcfeac2ac67f")
  1: Object
  2: Object
  3: Object
  4: Object
  5: Object
  6: Object
createdAt: 2022-06-15T22:16:33.145+00:00
updatedAt: 2022-06-23T15:01:55.020+00:00
__v: 0
```

Figura 3.7: Datos de un usuario en la base de datos

Se puede observar cómo se almacenan los datos y la contraseña se almacena encriptada, por otro lado, se ve como el nombre artístico no está definido porque este usuario es de tipo “User” lo que significa que solo puede iniciar sesión en la aplicación móvil, en el caso de los artistas si se le permite iniciar sesión en la aplicación web y si cuentan con nombre artístico.

Luego se pasó a configurar las rutas, analizando el flujo de información que tiene una aplicación como esta, se llegó a la conclusión de que se necesitan dos rutas principales las cuales derivaron en más rutas cada una, estas son “user” y “concert” y estas se configuraron en “src/routes/user.routes.js” y “src/routes/concert.routes.js” respectivamente.

En estos archivos se le especifica al backend que función tiene que ejecutar dependiendo de que tipo de petición le llegue y por cual ruta.

En el caso de las rutas para “users” se definieron las siguientes:

```
const router = express.Router();

// SingUp
// api/user
router.post("/", singUp);

// SingIn
// api/user/auth
router.post("/auth", singIn);

// Get the authenticated user
router.get("/auth", checkAuth, authenticatedUser);

// Confirm account
router.get("/confirm/:token", confirmIdToken);

// Reset password
router.post("/reset-password", resetPasswordResetToken);
router
  .route("/reset-password/:token")
  .get(resetPasswordCheckToken)
  .post(resetPasswordNewPass);

// Delete User
// api/user
router.delete("/", checkAuth, deleteUser);

// Profile
router.get("/profile", checkAuth, profile);
router.put("/profile", checkAuth, editProfile);

export default router;
```

Figura 3.8: Rutas para usuarios

En este caso empezando de arriba para abajo se definió un POST a “/api/user/” el cual es para cuando un usuario quiere registrarse, como se puede ver se hace una llamada a una función llamada “singUp” la cual contiene la lógica necesaria para registrar al usuario en la base de datos y generar un token para que pueda acceder a la aplicación, más adelante se explicaran dichas funciones.

Luego se definió un POST a “/api/user/auth” lo cual llama a “singIn” y es para el inicio de sesión de los usuarios.

A continuación, tenemos varios GET y POST a “/api/user/reset-password” los cuales sirven para reestablecer la contraseña y por último un par de GET y PUT a “/api/user/profile” que sirven para editar y consultar el perfil de los usuarios.

Por otro lado, se crearon las rutas para los conciertos las cuales son las siguientes:

```
const router = express.Router();

// Artist
router.get("/artist", checkAuth, getArtistConcerts);
router.post("/artist", checkAuth, createArtistConcert);
router.get("/artist/:id", checkAuth, getArtistConcert);
router.put("/artist/:id", checkAuth, editArtistConcert);
router.delete("/artist/:id", checkAuth, deleteArtistConcert);

// Upload and update images
router.post("/", checkAuth, multerUpload.single("file"), uploadImage);
router.put("/", checkAuth, multerUpload.single("file"), editImage);

// Upload APK
router.post("/apk", multerUploadAPK.single("file"), uploadAPK);

// User
router.post("/user-saved-concerts", checkAuth, setUserSavedConcerts);
router.post("/purchased-tickets", checkAuth, setpurchasedTickets);

// All
router.get("/", getConcerts);
router.get("/:id", getConcert);

export default router;
```

Figura 3.9: Rutas para conciertos

Estas rutas son utilizadas para todo lo referente a los conciertos, en el primer apartado podemos ver que están dedicadas a los artistas y son las rutas para crear, ver, modificar y eliminar conciertos desde el menú que tienen los artistas en la web.

El siguiente apartado es para subir o actualizar una imagen de un concierto seguido de la ruta que guarda el APK en el Backend a la hora del despliegue.

Luego de esto se definieron las rutas usadas por los usuarios que son para guardar conciertos en favoritos y cuando compran boletos y por último se definieron dos rutas más que son generales de la aplicación y sirven para ver todos los conciertos de la base de datos o uno solo en específico conociendo su id.

Como se puede ver, en algunas de las rutas, antes de llamar a la función correspondiente que contiene la lógica para dicha función, se ha creado un middleware que sirve para verificar que el usuario este autenticado y tenga un token valido, esto solo se usa en las acciones que requieren este tipo de validación y se ha configurado en "src/middleware/checkAuth.js" y contiene lo siguiente:

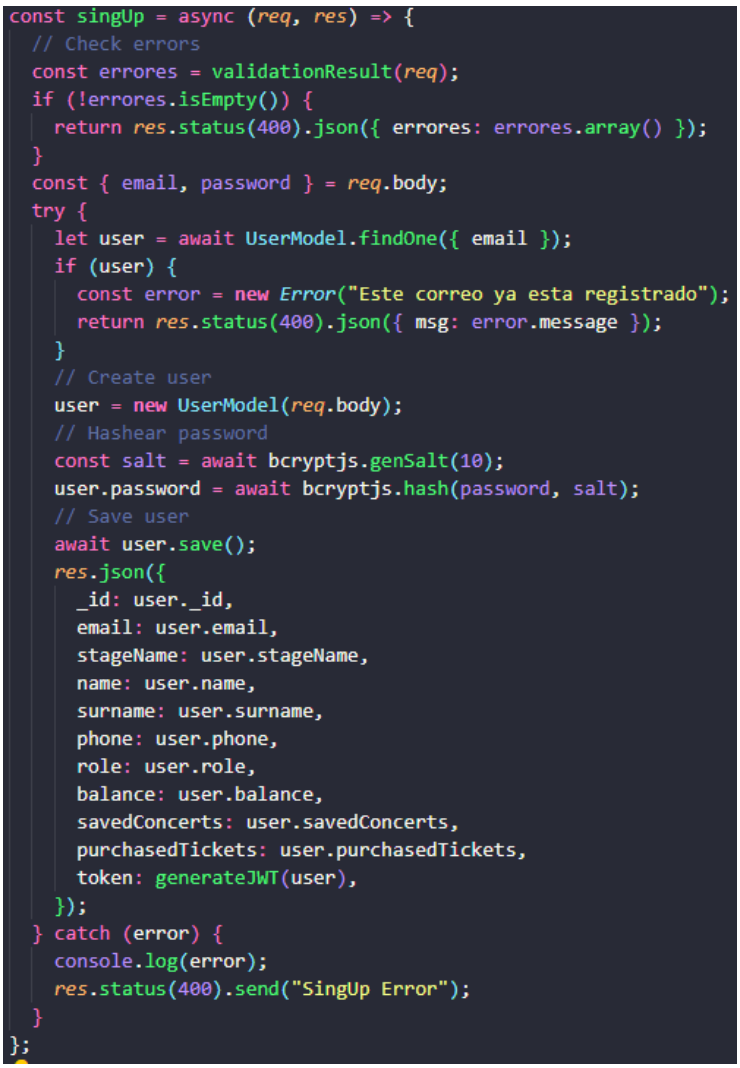
```
const auth = async (req, res, next) => {
  let token;
  if (
    req.headers.authorization &&
    req.headers.authorization.startsWith("Bearer")
  ) {
    try {
      token = req.headers.authorization.split(" ")[1];
      const decoded = jwt.verify(token, process.env.SECURE_KEY);
      req.user = await UserModel.findById(decoded.user.id).select(
        "-password -confirmed -token -createdAt -updatedAt -__v"
      );
      return next();
    } catch (error) {
      res.status(404).json({ msg: "Token no válido" });
    }
  }
  if (!token) {
    const error = new Error("Token no válido");
    return res.status(401).json({ msg: error.message });
  }
  next();
};

export default auth;
```

Figura 3.10: Middleware para autenticación

Podemos observar que su función es intentar tomar de la cabecera de la petición que llega al Backend algún token almacenado y verificar si ese token es válido y si pertenece al usuario que está haciendo la petición, en caso positivo se continúan ejecutando las demás funciones de las rutas y en caso negativo la petición es rechazada con un error de "Token no válido".

Todas estas funciones definidas en las rutas y que contienen la lógica para cada función se encuentran en "src/controllers/user.controllers.js" y "src/controllers/concert.controllers.js", aquí es donde se hacen las consultas a la base de datos y se devuelven las respuestas al Frontend.

Un ejemplo de una de estas funciones es el registro de usuarios que es la siguiente:



```
const singUp = async (req, res) => {
  // Check errors
  const errores = validationResult(req);
  if (!errores.isEmpty()) {
    return res.status(400).json({ errores: errores.array() });
  }
  const { email, password } = req.body;
  try {
    let user = await UserModel.findOne({ email });
    if (user) {
      const error = new Error("Este correo ya esta registrado");
      return res.status(400).json({ msg: error.message });
    }
    // Create user
    user = new UserModel(req.body);
    // Hashear password
    const salt = await bcryptjs.genSalt(10);
    user.password = await bcryptjs.hash(password, salt);
    // Save user
    await user.save();
    res.json({
      _id: user._id,
      email: user.email,
      stageName: user.stageName,
      name: user.name,
      surname: user.surname,
      phone: user.phone,
      role: user.role,
      balance: user.balance,
      savedConcerts: user.savedConcerts,
      purchasedTickets: user.purchasedTickets,
      token: generateJWT(user),
    });
  } catch (error) {
    console.log(error);
    res.status(400).send("SingUp Error");
  }
};
```

Figura 3.11: Función para registro de usuario

Aquí podemos observar que lo primero que se hace es extraer del cuerpo de la petición el email para preguntar a la base de datos si ya ese email existe, si no existe se procede a extraer todos los datos del usuario del cuerpo de la petición aun “Schema” de tipo usuario antes definido y luego de eso se encripta la contraseña para almacenarla encriptada. Por último, se guarda el usuario en la base de datos y se devuelve una respuesta con toda la información que se ha almacenado y un token generado con la función antes creadas que será almacenada en el “local-storage” del navegador o del móvil donde sea ejecutada la aplicación.

El resto de las funciones tienen una lógica parecida de extraer datos de la petición, consultar a la base de datos y generar una respuesta, para el apartado de conciertos se sigue la misma lógica.

Todas estas funciones se pueden encontrar en el repositorio(14) de GitHub del Backend.

Por último, se configuró en el index.js todo esto antes creado para poder ejecutar la aplicación y que se inicie el servidor que escuchara las peticiones, establecer conexión con la base de datos, con Cloudinary y especificar las rutas que llegaran al Backend:

```
import express from "express";
import path from "path";
import cors from "cors";
import { fileURLToPath } from "url";

import connectDB from "./config/db.js";
import connectCloudinary from "./config/cloudinary.js";

import userRoutes from "./routes/user.routes.js";
import concertRoutes from "./routes/concert.routes.js";

import dotenv from "dotenv";
dotenv.config();

const app = express();
const PORT = process.env.PORT || 4000;

// Connections
connectDB();
connectCloudinary();

// Enabling cors
app.use(cors());

// Enabling express.json
app.use(express.json({ extended: true }));

// Routing
app.use("/api/user", userRoutes);
app.use("/api/concerts", concertRoutes);
app.use("/api/files", concertRoutes);

// Landing Page
app.get("/", (req, res) => {
  res.sendFile(
    path.join(
      path.dirname(fileURLToPath(import.meta.url)),
      "../public/index.html"
    )
  );
});

// Download APK
app.get("/api/fundmymusic", (req, res) => {
  res.sendFile(
    path.join(
      path.dirname(fileURLToPath(import.meta.url)),
      "../public/app-release.apk"
    )
  );
});

// Turn on the server
app.listen(PORT, () => console.log(`Server listening on port ${PORT}`));
```

Figura 3.12: Index del Backend

Como se puede ver al inicio se hacen las importaciones necesarias, luego se llaman a las funciones que conectan la base de datos y Cloudinary y luego se habilitan funciones para que el servidor pueda procesar JSONs, más debajo se definen las rutas de nuestro Backend que van dirigidas a todas las que se crearon anteriormente y por ultima se inicia el servidor para escuchar las peticiones.

## 3.3 Frontend Web

Para el desarrollo de la aplicación web se ha utilizado React mediante VSCode que es un editor de texto muy completo, React se caracteriza por el uso de componentes para mostrar la interfaz al usuario por lo que las imágenes que se mostraran a continuación están conformadas por componentes que se irán explicando según sea el caso.

Las dependencias usadas junto con React para esta aplicación web son:

- Tailwindcss(15) para los estilos
- Axios(16) para las peticiones al Backend
- Sweetalert2 para las alertas

### 3.3.1 Inicio de sesión

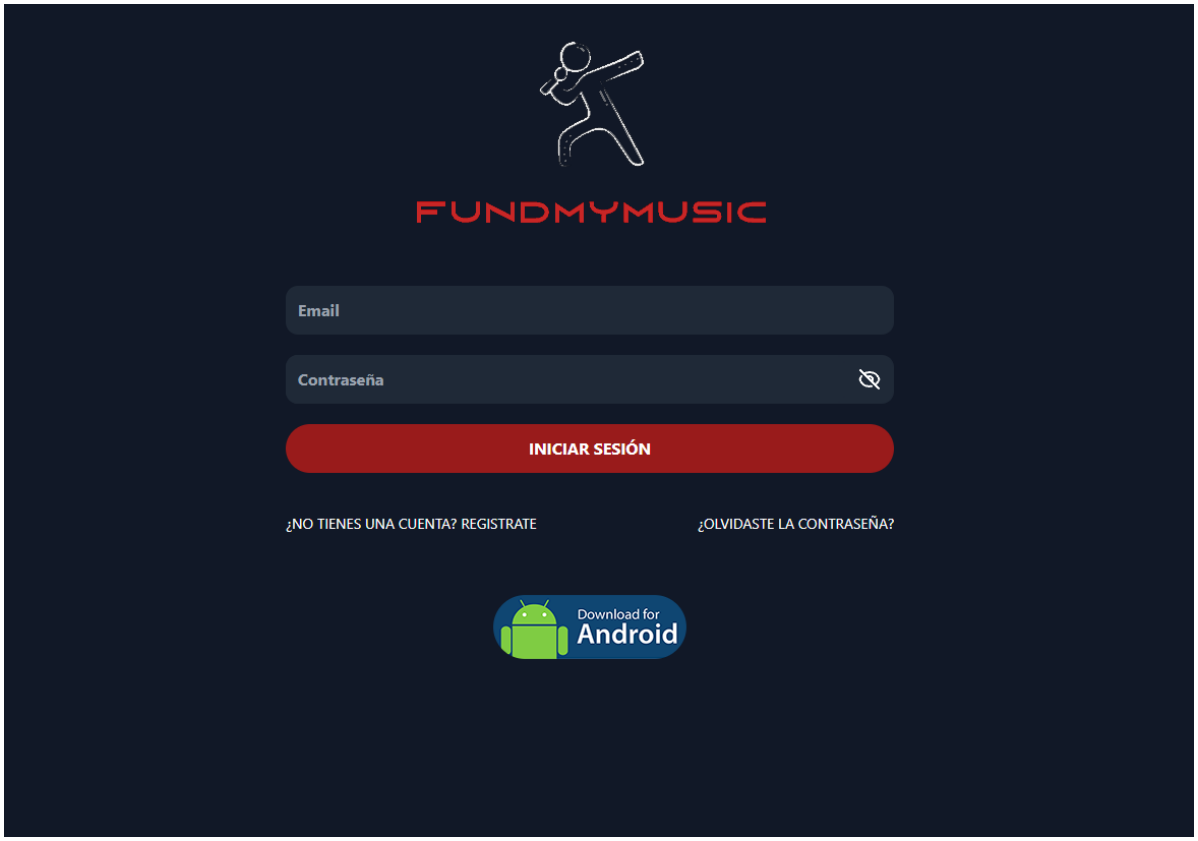



Figura 3.13: Inicio de sesión web

Lo primero que ve el artista al ingresar a fundmumusic.es(17) es la pantalla para iniciar sesión, todo esto esta creado con React y en este caso el logo de la aplicación es un componente separado de los inputs junto con el botón que son otro componente.

En este apartado se puede iniciar sesión o si no se posee una cuenta se puede crear una en los enlaces que se encuentran debajo del botón, por otro lado, hay un botón azul de donde se puede descargar el APK de la app móvil que se encuentra alojada en el Backend.

### 3.3.2 Registro de usuario

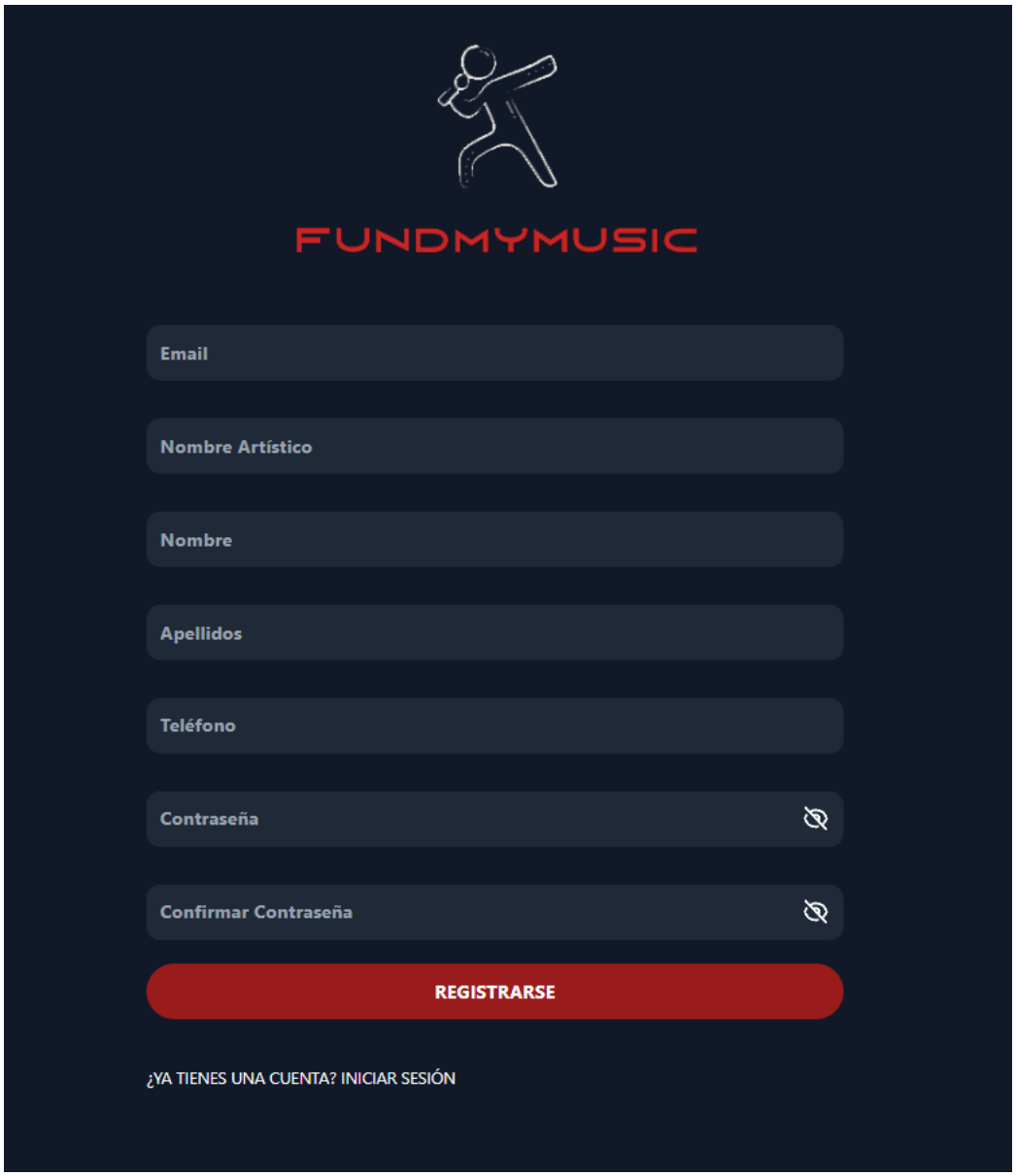



Figura 3.14: Registro web

Al pinchar en registrarse aparece esta vista la cual permite al artista crearse una cuenta con todos los datos necesarios para la base de datos, todos los campos de la aplicación se encuentran validados para que no se puede ingresar datos erróneos al sistema.

Si se rellenan todos los datos y todo es válido el usuario se registra y se pasa al panel principal de la aplicación web.

### 3.3.3 Panel principal

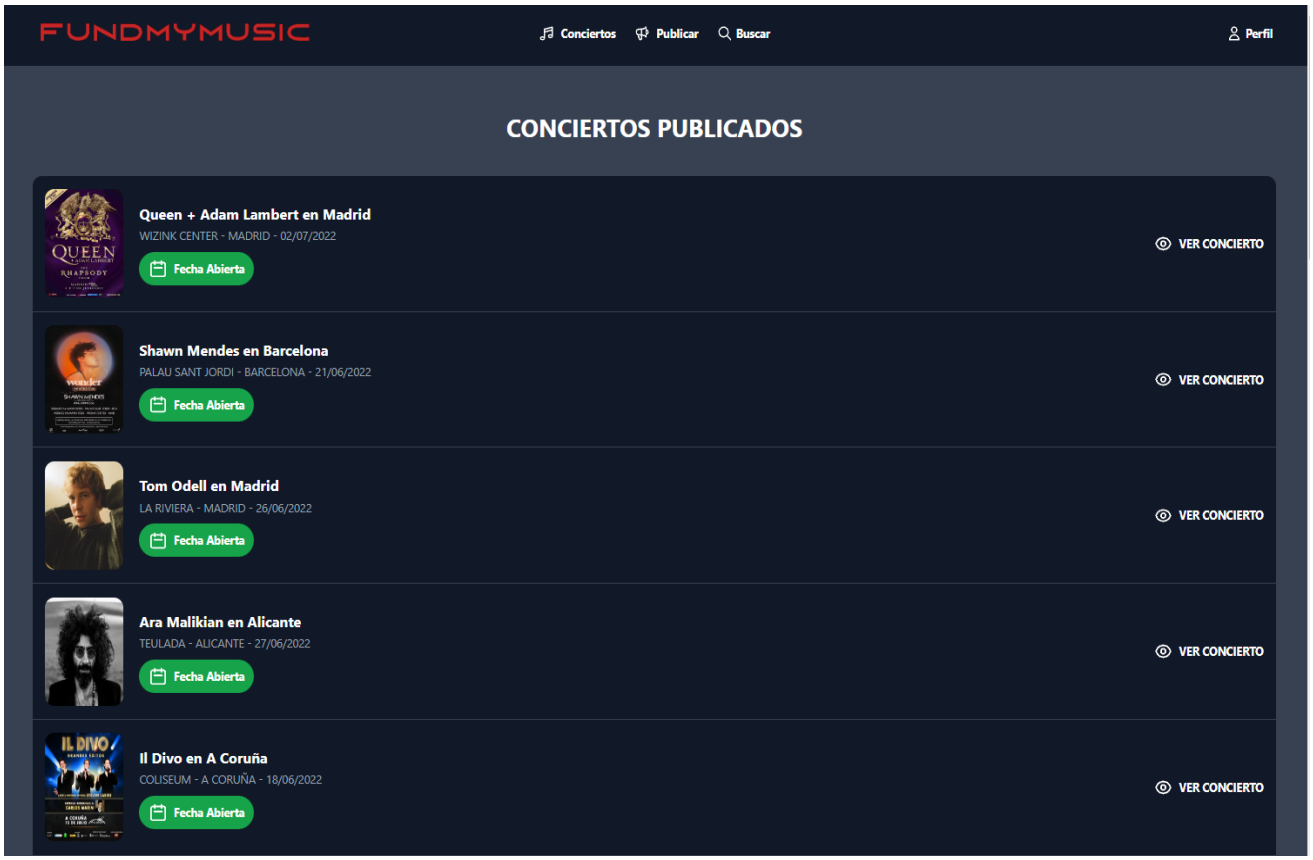



Figura 3.15: Panel principal

En este panel el artista tiene una cabecera donde tiene en la parte central un menú de 3 botones donde puede ver los conciertos publicados, puede publicar uno nuevo o puede buscar alguno en concreto por su título con un buscador, al lado izquierdo se encuentra el logo de la aplicación y del lado derecho un botón para ir al perfil del usuario.

En la parte central el artista tiene una lista de todos sus conciertos publicados y tiene información importante de cada uno como puede ser el título, la fecha, el lugar, la imagen del concierto y si está abierta o cerrada la fecha, también se puede ver si está o no agotado el concierto.

A la derecha de cada ítem de la lista hay un botón el cual permite al artista ver con detalle cada concierto por separado.

### 3.3.4 Vista individual de un concierto


Figura 3.16: Vista individual de un concierto web

Si se pincha en “ver concierto” en la vista anterior, se accede a esta pantalla donde se puede ver detalladamente toda la información de un concierto que se encuentre publicado, incluyendo las entradas vendidas y en la parte superior derecha se encuentran dos botones que son para eliminar el concierto o para editarlo.

En caso de que la fecha del concierto este cerrada porque se han vendido todas las entradas mínimas requeridas por el artista, el botón de editar desaparece porque ya no se puede editar el concierto.

### 3.3.5 Publicar o editar un concierto

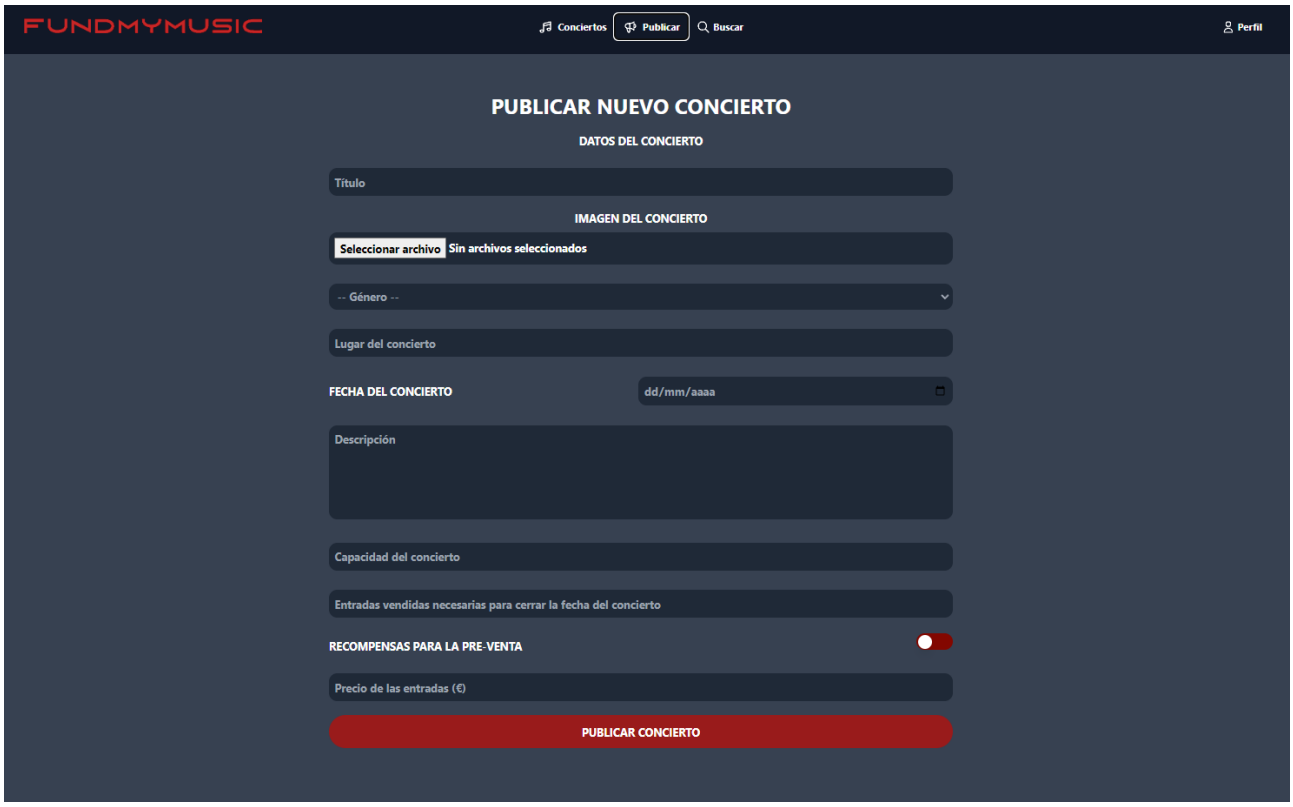



Figura 3.17: Publicar concierto

Este apartado es el de publicar concierto, para acceder a él se tiene el menú superior antes mencionado que contiene un botón para publicar, como se puede ver se pide al artista toda la información necesaria para la creación de un concierto.

Todos los campos de este formulario se encuentran validados para evitar ingresar datos erróneos como una fecha que ya pasó, una imagen que no es de formato imagen valido, que las entradas mínimas no sean mayores que la capacidad del concierto entre otras validaciones.

Por último, este mismo formulario es el que se muestra y llena automáticamente cuando se pincha en editar un concierto en la vista individual de cada concierto y el botón cambia a actualizar concierto.

Todos los componentes y el código de estos apartados se encuentran en el repositorio(18) del Frontend Web.

### 3.3.6 Buscador

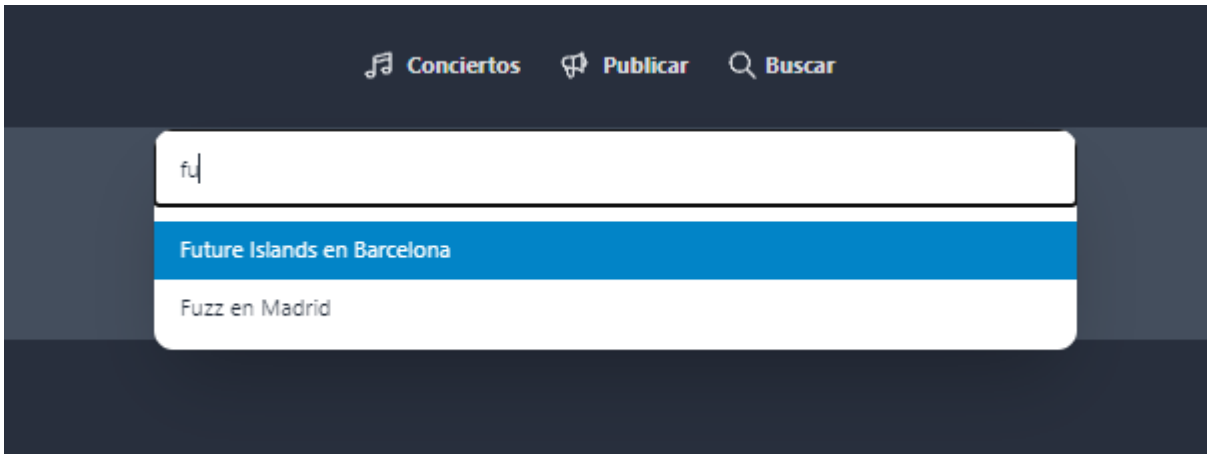



Figura 3.18: Buscador web

Para el botón del buscador simplemente se despliega un buscador que al ir escribiendo en el se van filtrando los conciertos que tengan esas letras en el nombre y al pinchar en uno de ellos se vuelve a mostrar la vista del concierto individual, con el concierto que se ha buscado y pinchado.

### 3.3.7 Perfil

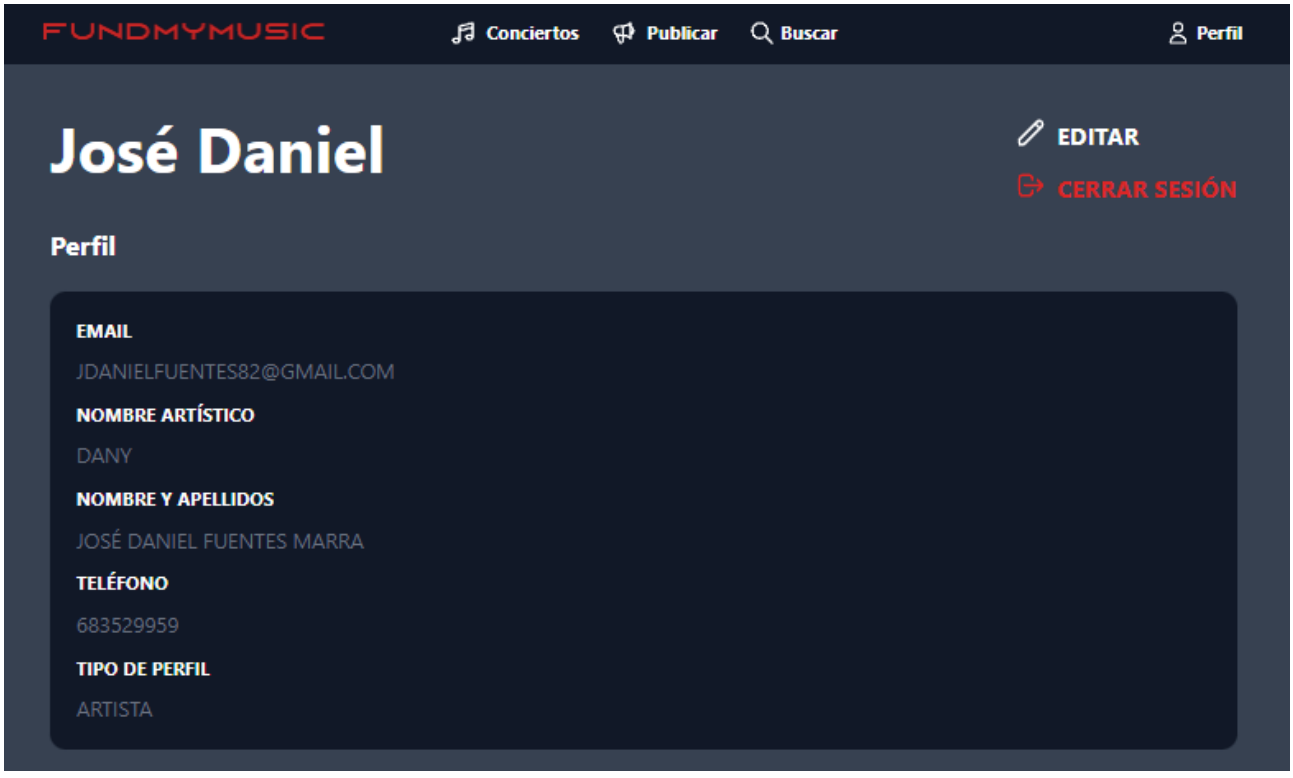



Figura 3.19: Perfil web

Si el artista pincha en perfil en el menú superior a la derecha se le muestra esta vista el cual le muestra la información del su perfil, aquí puede editarlo o cerrar sesión en los botones que se encuentran en la parte superior derecha.

## 3.4 Frontend Mobile

La aplicación móvil fue creada en React Native, usando VSCode como editor de texto y un emulador de Android conectado a través de Metro(19) al modo desarrollador que ofrece React Native para poder ver los cambios en tiempo real en el emulador de Android.

Las dependencias adicionales usadas para la aplicación móvil son:

- Async-storage para usar el almacenamiento interno del dispositivo
- Axios para las peticiones al Backend
- Anchor-carousel para el carousel de la pantalla principal
- Credit-card para las animaciones de la tarjeta al agregar saldo
- React-native-paper(20) para los estilos
- Qrcode-svg para generar los códigos QR de los boletos
- Sweet-alert para las alertas
- Vector-icons para los iconos

### 3.4.1 Pantalla principal

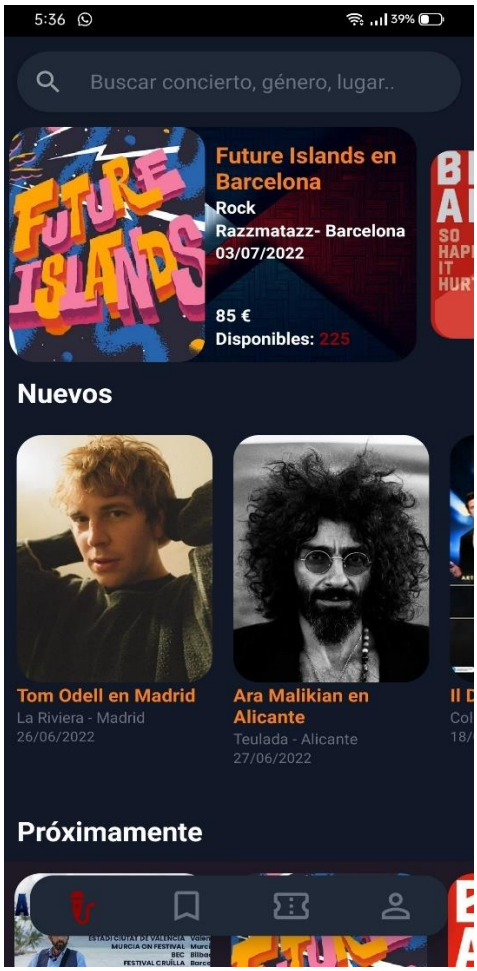



Figura 3.20: Pantalla principal móvil

Como se puede ver al abrir la aplicación al usuario se le muestra la pantalla principal donde se pueden ver arriba del todo un buscador, en el centro podemos ver varias secciones que son listas horizontales con los conciertos que se dividen en los últimos que se han publicado, los próximos que son los que tienen lugar de la fecha actual a 3 meses y debajo se encuentra otra sección de más vendidos donde se muestran los 4 conciertos que más entradas han vendido últimamente.

En la parte inferior el usuario tiene una barra de navegación que es un menú con 4 opciones que son los conciertos publicados, los conciertos guardados, los boletos comprados y el perfil del usuario.

### 3.4.2 Vista individual de un concierto


Figura 3.21: Vista individual de un concierto móvil

Al pinchar en un concierto se muestra esta vista donde se puede ver la información completa del concierto y al final se puede comprar un boleto para ese concierto. En la parte superior derecha se encuentra el botón para añadir el concierto a guardados.

La cantidad en € en el botón de comprar cambiar según el número de entradas que se pongan y las entradas están validadas para no comprar más de las que hay disponibles, en

caso de no tener la sesión iniciada o estar agotadas las entradas, el botón de comprar desaparece y aparece un botón de iniciar sesión o que las entradas están agotadas.

### 3.4.3 Conciertos guardados

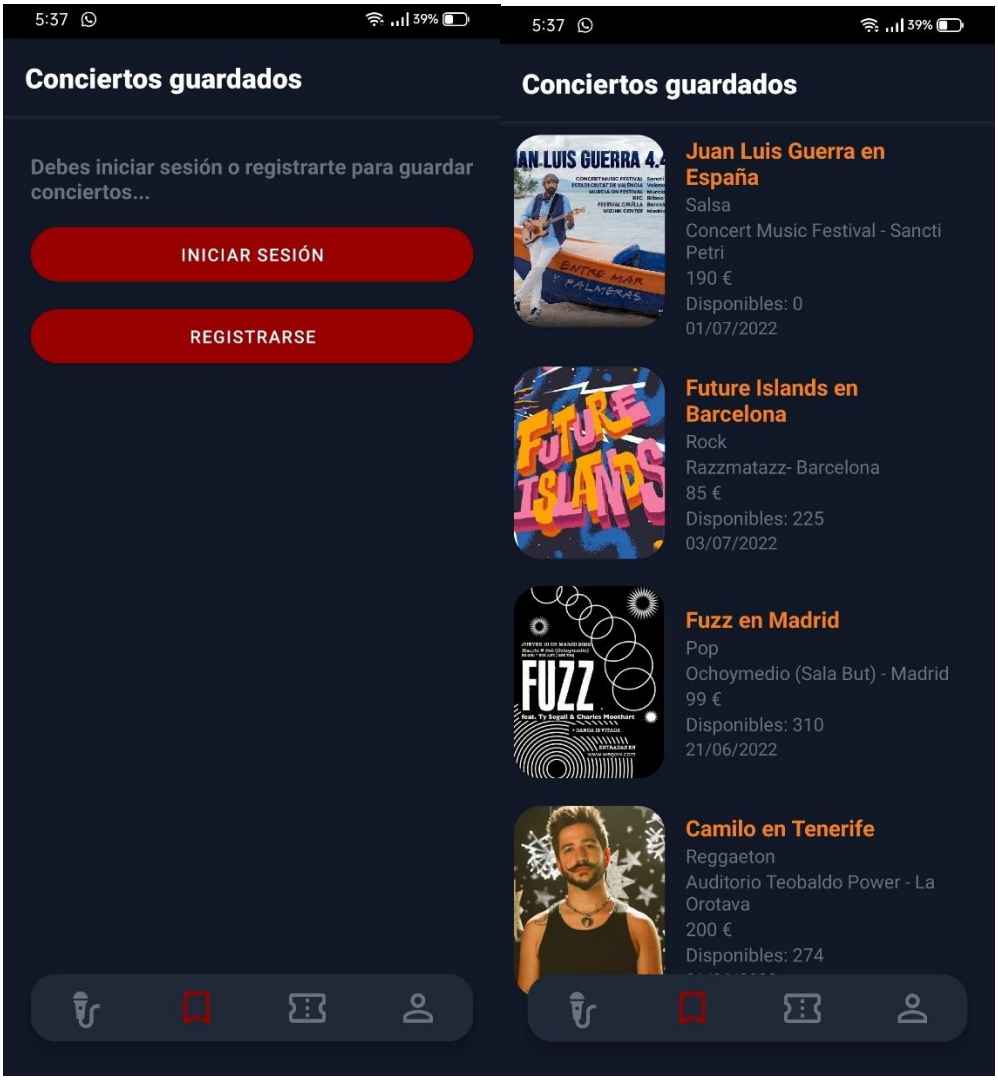



Figura 3.22: Conciertos guardados

Mediante el segundo botón de la barra inferior de navegación el usuario puede acceder a sus conciertos guardados, si no se encuentra la sesión iniciada aparecerá como en la imagen de la derecha, si se tiene la sesión iniciada se verán en una lista los conciertos que se tienen guardados con información adicional como el precio de la entrada o la cantidad de entradas que quedan disponibles.

Al pichar en cualquiera de estos conciertos se accedería a la vista anterior donde se pueden comparar las entradas.

### 3.4.4 Boletos comprados

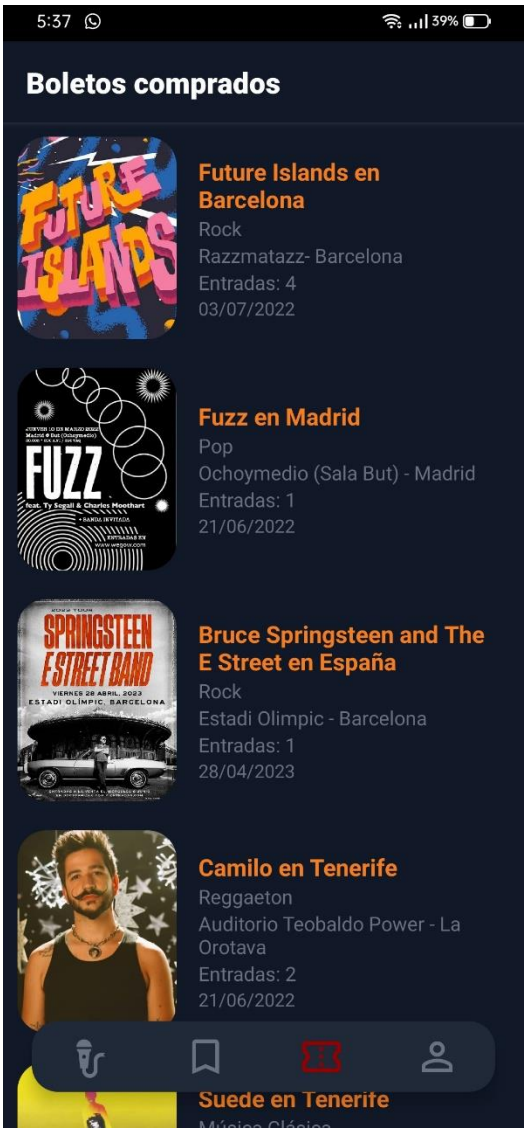



Figura 3.23: Boletos comprados

El siguiente apartado en la barra de navegación inferior es para ver los boletos que se han comprado hasta el momento, nuevamente si la sesión no está iniciada solo se ven los botones para iniciar sesión o registrarse.

En esta vista el usuario puede ver una lista como todas las entradas que ha comprado agrupadas por conciertos ya que se pueden comprar varias entradas y le aparece cuantas entradas tiene compradas de cada concierto en la lista.

### 3.4.5 Vista de boleto

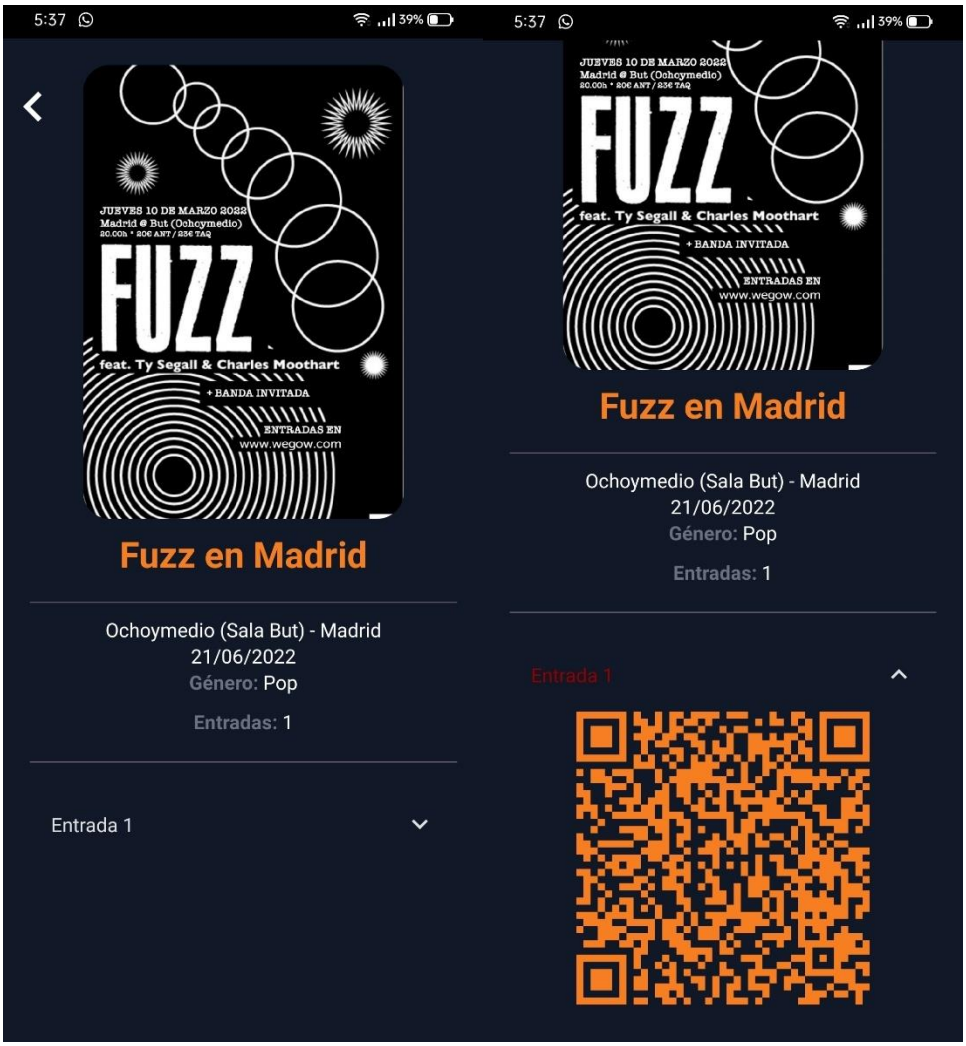



Figura 3.24: Vista de boleto

Al pinchar en cualquier boleto de la lista de la vista anterior, se le muestra al usuario la información del concierto seguido de las entradas que haya comprado, en este caso solo tiene una y al desplegar la entrada se puede ver el código QR generado que debe mostrar en el concierto.

### 3.4.6 Perfil

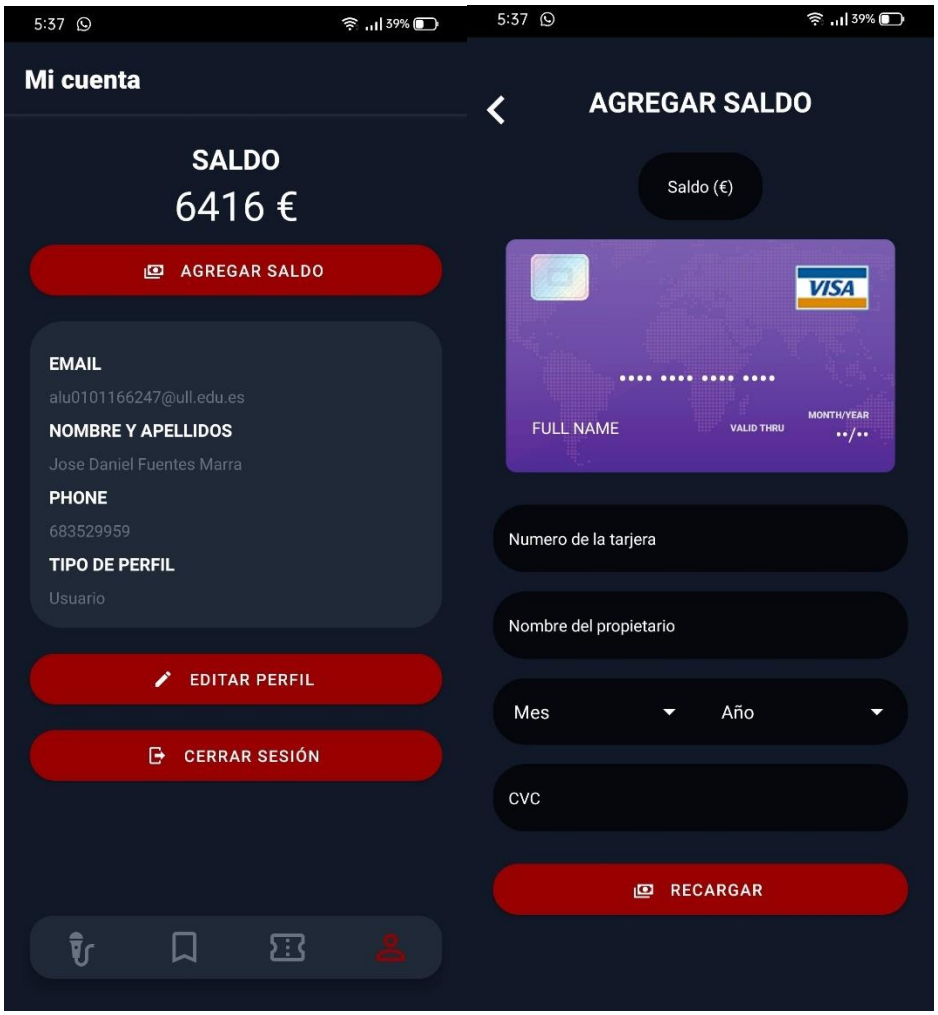



Figura 3.25: Perfil móvil

El último elemento en la barra de navegación inferior lleva al usuario a su perfil donde puede agregar saldo para comprar boletos, editar los datos de su perfil, o cerrar sesión.

Como para este proyecto no se cuenta como una pasarela de pago implementada, los campos de la tarjeta no son requeridos para recargar saldo. Basta con poner la cantidad en € en la parte inferior y pinchar en “Recargar” lo que automáticamente nos agrega saldo a la cuenta.

### 3.4.7 Iniciar sesión y registrarse

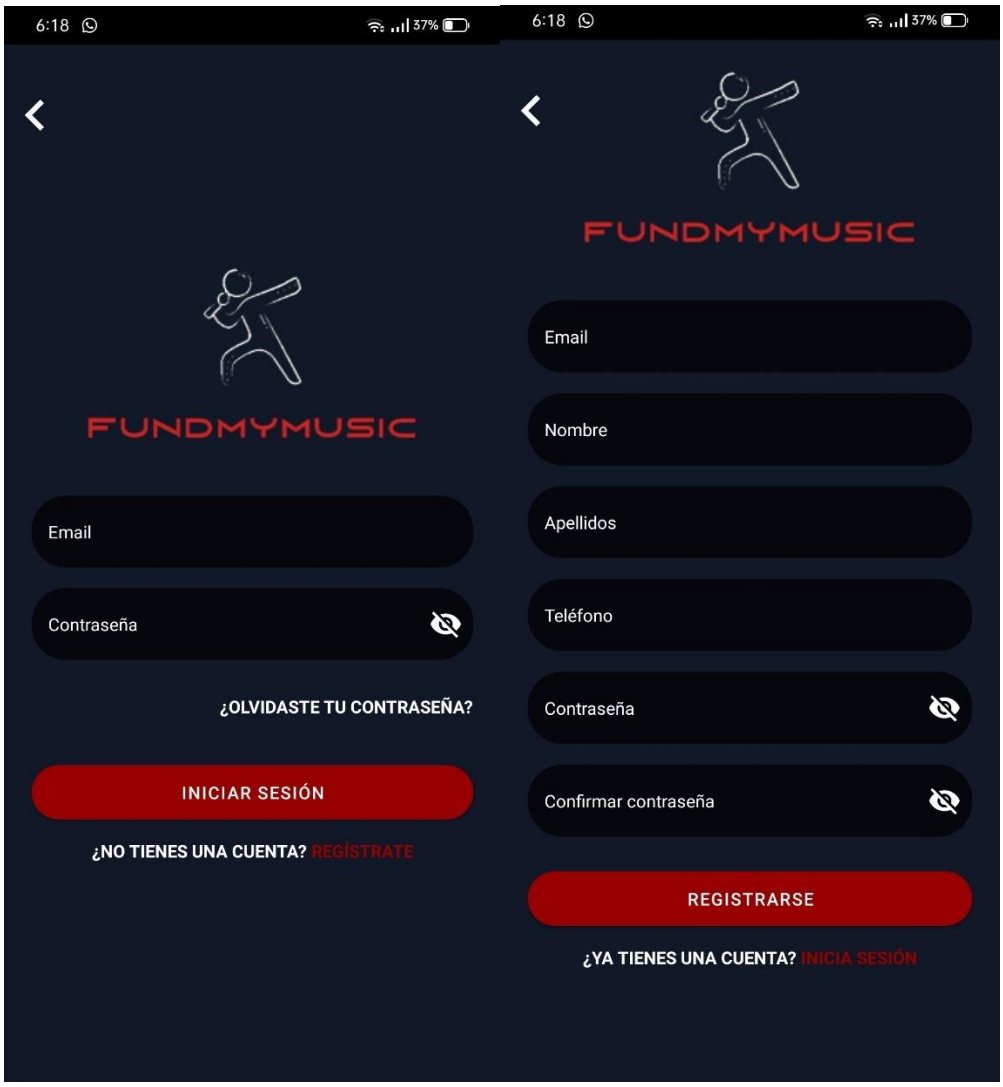



Figura 3.26: Inicio de sesión y registro móvil

Por último, estas son las vistas de las pantallas de iniciar sesión y registrarse si en alguna de las ocasiones anteriores el usuario no está registrado y pincha en iniciar sesión o registrarse.

Al registrarse o iniciar sesión el usuario es devuelto a la pantalla principal de la aplicación donde se pueden ver todos los conciertos.

Para finalizar mencionar que todos los campos en donde se puede ingresar información en la aplicación móvil están validados igual que en la aplicación web.

Todos los componentes así como el código de este apartado se puede ver en el repositorio(21) del Frontend Mobile.

# Capítulo 4
# Despliegue

## 4.1 Backend

Para el despliegue automático se ha utilizado el flujo de trabajo de Github Actions, en los archivos del Backend se ha creado un directorio ".github/workflow" donde se encuentra un archivo llamado "deploy.yml" el cual contiene la configuración necesaria para que al realizar un "push" del código al repositorio de GitHub automáticamente se despliegue en Heroku*(22).*

Heroku es un servicio el cual permite ejecutar el Backend en uno de sus servidores proporcionándonos una dirección a donde podremos hacer las peticiones desde el Frontend logrando así tener el Backend ejecutándose en la nube y poder acceder a él desde cualquier lugar.

Para esto se ha creado una cuenta en Heroku, luego se ha creado un proyecto nuevo dentro de Heroku el cual nos proporciona el "app-name" y el "api-key" que fueron configurados como "secrets" en el repositorio de Github del Backend para que este se pueda conectar a Heroku y desplegar correctamente el Backend.

Este es el contenido del archivo de configuración para el Action del Backend:

```yaml
name: Deploy

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${{secrets.HEROKU_API_KEY}}
          heroku_app_name: "fundmymusic-backend"
```

Figura 4.1: Configuración para el Actions del Backend

## 4.2 Frontend Web

Para el despliegue de la aplicación web también se ha hecho uso de los Actions de Github, en este caso la página web esta alojada en Ionos(23) el cual es un hosting que tiene una funcionalidad llamada DeployNow que nos permite mediante las Actions generar el build del proyecto de React y publicarlo en una URL especifica.

Para ello se ha creado una cuenta en Ionos y dentro del panel de control se ha comprado el dominio “fundmymusic.es” para desplegar nuestro sitio web, luego de esto en el apartado de DeployNow se creó un proyecto nuevo y al igual que el apartado anterior nos proporciona las “keys” necesarias para configurar el workflow.

Nuevamente se ha creado un directorio “.github/workflow” y dentro un archivo llamado “deploy-now.yaml” el cual contiene una configuración parecida al del Backend pero para Ionos.

Con esto se encuentra configurado el despliegue de la aplicación web por lo que al pushear nuevas versiones del código, automáticamente se genera el build y se envía a Ionos donde se publica en el dominio comprado.

## 4.3 Frontend Mobile

Para el despliegue de la versión para móvil es diferente ya que lo que se genera en este caso es un .apk que tiene que ser instalado en los dispositivos móviles donde se quiera usar la aplicación por lo tanto lo que se ha hecho es un workflow que al pushear el código de la aplicación, se genera el APK, se firma y se sube a un apartado de Artifacts que tiene Github Actions del cual mediante la página principal de fundmymusic.es se puede descargar el APK para ser instalado en cualquier dispositivo.

También se valoró configurar una cuenta de Google Play Store para distribuir el APK, pero se necesitaban muchas licencias y requisitos que dieron por descartada esta opción.

# Capítulo 5
# Conclusiones y líneas futuras

## 5.1 Conclusiones

En conclusión, se ha desarrollado una aplicación que mediante su versión web permite a los artistas publicar sus conciertos y vender los boletos de dichos conciertos con una interfaz fácil de usar e intuitiva y mediante la versión móvil permite a usuarios conocer los conciertos que se publican y da la posibilidad de comprar boletos para asistir a dichos conciertos, todo esto con una base de datos central donde se almacenan todos los datos necesarios como las cuentas de los usuarios o los conciertos.

Una aplicación muy completa desarrollada en poco tiempo y utilizando diversos lenguajes y herramientas para lograrlo, cumple muy bien sus objetivos iniciales y tiene un gran potencial para escalar y adaptarse a distintos mercados.

En lo personal he aprendido mucho sobre React y React Native que son unas herramientas increíbles para este tipo de desarrollo, también he aprendido bastante sobre como desplegar este tipo de servicios ya que mi conocimiento en esto era muy básico.

## 5.2 Líneas Futuras

En mi opinión creo que a la aplicación todavía le queda mucho margen de crecimiento sobre todo en la versión web donde se podrían implementar por ejemplo un apartado donde se lleven estadísticas sobre la venta de los boletos de los conciertos, así como datos estadísticos sobre el tipo de público, la edad o los gustos de las personas que compra los boletos.

Por otro lado, el apartado de recargar saldo se encuentra en fase de simulación ya que no se cuenta con una pasarela de pago implementada lo cual se puede hacer en un futuro agregando también más métodos de pago.

Y por último quizás también agregar un sistema de valoraciones de los conciertos para tener un ranking de estrellas para los artistas y así poder tener más feedback por parte de los usuarios que utilizan la aplicación y asisten a los conciertos.

# Capítulo 6
# Summary and Conclusions

## 6.1 Conclusions

In conclusion, an application has been developed that through, its web version allows artists to publish their concerts and sell tickets for said concerts with an easy-to-use and intuitive interface, and through its mobile version allows users to know the concerts that are published and gives them the possibility of buying tickets to attend said concerts, all this with a central database where all the necessary data such as user accounts or concerts are stored.

A very complete application developed in a short period of time and using several languages and tools to achieve it, it meets its initial objectives very well and has great potential to scale and adapt to different markets.

Personally, I have learned a lot about React and React Native, which are incredible tools for this type of development. I have also learned a lot about how to deploy this type of service, since my knowledge of this was very basic.

## 6.2 Future Work

In my opinion, I believe that the application still has a lot of room for growth, especially in the web version where, for example, a section could be implemented where statistics are kept on the sale of concert tickets, as well as statistical data on the type of public, the age or the tastes of the people who buy the tickets.

On the other hand, the balance recharge section is in the simulation phase since there is no payment gateway implemented, which can be done in the future and as well adding more payment methods.

And finally, perhaps also adding a rating system for the concerts to have a star ranking for the artists and thus be able to have more feedback from the users who use the application and attend the concerts.

# Capítulo 7
# Presupuesto

## 7.1 Desarrollo del software

En la siguiente tabla se especifican las horas invertidas en cada etapa del desarrollo, tomando en cuenta un valor de 26€/h.

| Tipos | Planificación y Diseño (h) | Desarrollo (h) | Coste (€) |
|---|---|---|---|
| Base de datos | 5 | 10 | 390 |
| Backend | 5 | 30 | 910 |
| Frontend Web | 10 | 40 | 1300 |
| Frontend Mobile | 10 | 50 | 1560 |
| Total | 30 | 130 | 4160 |

**Tabla 1:** Coste de desarrollo del software del proyecto.

## 7.2 Herramientas, despliegue y alojamiento

En este caso las herramientas de desarrollo que se han utilizado son gratis, por otro lado, las herramientas como Heroku, Ionos, Cloudinary y MongoDB Atlas que son para desplegar y alojar toda la aplicación ofrecen servicios gratuitos pero limitados.

Para MongoDB Atlas que es donde se encuentra alojada la base de datos, se ha utilizado el plan gratuito que ofrece solo 500mb de almacenamiento, pero es suficiente para este proyecto por lo que no genera gasto adicional.

Para Heroku que es donde se aloja y ejecuta el Backend también cuenta con un plan gratuito de 500 horas de uso por mes por lo que es más que suficiente para este proyecto y tampoco genera gasto adicional.

Para Cloudinary que es donde se almacenan las imágenes de los conciertos también se puede utilizar con un plan gratuito que es suficiente para este proyecto y nuevamente no genera gastos adicionales.

Por último, en Ionos si se necesita pagar una mensualidad para tener un hosting con capacidad de soporte para DNS y configurar el dominio personalizado para la aplicación web por lo que son unos 15€ al mes más 12 al año por el dominio “fundmymusic.es”.

| Tipos | Coste (€/mes) |
|---|---|
| VS Code | 0 |
| GitHub | 0 |
| Heroku | 0 |
| Cloudinary | 0 |
| MongoDB Atlas | 0 |
| Ionos (Hosting) | 15 |
| Dominio “fundmymusic.es” | 1 |

**Tabla 2:** Coste de herramientas, despliegue y alojamiento del proyecto.

## 7.3 Total

| Tipos | Coste plano (€) | Coste mensual (€) |
|---|---|---|
| Desarrollo del software | 4160 | |
| Recursos, herramientas y alojamiento | | 16 |

**Tabla 3:** Gastos totales del proyecto.

# Bibliografía


[1] Kickstarter,

https://www.kickstarter.com/

[2] Indiegogo,

https://www.indiegogo.com/

[3] KissKissBankBank,

https://www.kisskissbankbank.com/

[4] Mola.fm,

http://www.molatv.cat

[5] My Major Company,

https://www.mymajorcompany.com

[6] React,

https://es.reactjs.org

[7] React Native,

https://reactnative.dev

[8] NodeJS,

https://nodejs.org/es

[9] MongoDB,

https://www.mongodb.com

[10] Cloudinary,

https://cloudinary.com

[11] Visual Studio Code,

https://code.visualstudio.com

[12] GitHub,

https://github.com

[13] MongoDB Atlas,

https://www.mongodb.com/es/cloud/atlas/efficiency

[14] Repositorio GitHub Backend,

https://github.com/JDany94/FundMyMusic-Backend

[15] Tailwindcss,

https://tailwindcss.com

[16] Axios,

https://axios-http.com/es/docs/intro

[17] FundMyMusic,

https://fundmymusic.es

[18] Repositorio GitHub Frontend Web,

https://github.com/JDany94/FundMyMusic-FrontendWeb

[19] Metro,

https://facebook.github.io/metro

[20] React Native Paper,

https://reactnativepaper.com

[21] Repositorio GitHub Frontend Mobile,

https://github.com/JDany94/FundMyMusic-FrontendMobile

[22] Heroku,

https://id.heroku.com

[23] Ionos,

https://www.ionos.es