Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Aplicación web para la visualización de eventos

Web application for the visualization of events

Carla María Mendoza Domínguez

La Laguna, 27 de junio de 2019

D. **José Luis González Ávila**, con N.I.F. 78.677.390-W 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

Que la presente memoria titulada:

”*Aplicación web para la visualización de eventos*”

ha sido realizada bajo su dirección por D. **Carla María Mendoza Domínguez**, con N.I.F. 78.858.539-A.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 27 de junio de 2019

Agradecimientos

A todos aquellos que me han apoyado durante mi recorrido.

Licencia

Resumen

Dada la cantidad actual de eventos que se celebran a diario, se ha creado un sistema para poder gestionarlos y visualizarlos. Tras estudiar las diferentes páginas web disponibles actualmente para localizar y gestionar los acontecimientos, se ha realizado una investigación sobre las diversas formas de organizar los eventos y las tecnologías con la que realizar los sistemas apropiados.

De este análisis se ha obtenido Mytfg, una aplicación web, que puede también ser usada en móviles, desarrollada con las tecnologías más punteras para desarrollo web como React, Node,Express y Mongo.

Mytfg permite al usuario la búsqueda y creación de eventos en cualquier lugar del mundo de forma geolocalizada. Esta aplicación pretende ser un referente para que empresas y particulares puedan darse a conocer promocionando sus eventos en la página.

Palabras clave: eventos, web, geolocalizada, React, Node, Express, Mongo

Abstract

Given the current number of events that take place on a daily , you have created a system for managing and displaying them. After studying the different websites currently available for locating and managing events, research has been carried out into the various ways of organising events and the technologies with which to make the appropriate systems.

From this analysis we have obtained Mytfg, a web application, which can also be used on mobile phones, developed with the latest technologies for web development such as React, Node, Express and Mongo.

Mytfg allows the user to search and create events anywhere in the world in a geolocalized way. This application is intended to be a reference for companies and individuals to make themselves known by promoting their events on the page.

Keywords: events, web, geolocalized, React, Node, Express, Mongo

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Introducción

Las tecnologías durante años han ayudado al ser humano, desde el descubrimiento de la rueda, hasta el aparato más complejo que pueda existir a día de hoy, han permitido al hombre desarrollarse. Tanto es así, que hoy en día mucho de lo que nos rodea es tecnología, hasta los pasos más elementales de nuestro día tienen algo que ver con los nuevos aparatos, y que éstos son ya inseparables partes de nuestro ecosistema.

La principal finalidad de las tecnologías es mejorar nuestra calidad de vida, por ello el objetivo principal de este trabajo es ayudar para simplificarnos la vida.

¿Cuántas veces nos hemos quedado en casa porque no sabíamos que hacer? o ¿cuántas veces nos hemos perdido un evento porque no sabíamos que se estaba celebrando ese mismo fin de semana? El objetivo de esta aplicación web es que esto no vuelva a suceder.

Mytfg es una página web diseñada para todos, un lugar donde consultar eventos próximos de forma centralizada y sencilla. Tiene una interfaz simple, intuitiva y llamativa, centrada únicamente en los eventos y su visualización. Además posee un sistema de filtrado donde poder obtener los eventos por categoría, por un rango de fechas, nombre o lugar, sin tener que, obligatoriamente recorrer todos eventos que haya.

Es una plataforma donde negocios, pueblos, ciudades y otras entidades podrán dar a conocer acontecimientos que celebren con tan solo rellenar un pequeño formulario.

Una aplicación por y para todos. Creada con las últimas tecnologías. En el lado del servidor (backend) se ha utilizado Node.js con Express y una base de datos no relacional, MongoDB. En el lado cliente (frontend) se hace uso de React con Redux, que permite crear una interfaz basada en componentes con un estado global. Para apoyar el desarrollo, se han utilizado herramientas como Jest, Gitlab, Docker, etc.

Para mantener los estándares de calidad de este proyecto se han llevado a cabo test unitarios y End-to-end, herramientas de validación de calidad de código e integración y despliegue continuo.

1.2. Antecedentes

Actualmente cuando navegamos por internet y buscamos páginas de eventos podemos encontrar varias opciones, sin embargo, ninguna de ellas cubre el 100 % de nuestras necesidades.

Uno de los grandes competidores es Google. Cuando introducimos en su campo de búsqueda “eventos en Tenerife” se nos muestra un listado de eventos como podemos ver en la imagen 1.1. La lista que nos expone Google, carece de orden aparente, y, además no posee ningún tipo de filtrado, por lo que hace difícil la búsqueda en la misma.

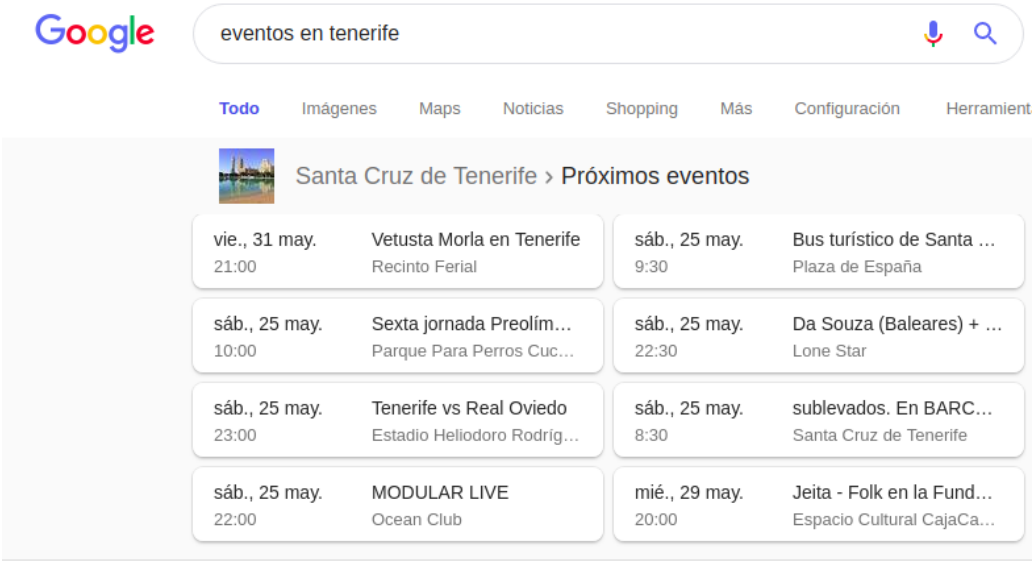

Figura 1.1: Eventos en Google

Por otro lado existen páginas como Tomaticket o Ticketea donde podemos informarnos de numerosos eventos, con la negativa de que estos son únicamente de pago, por lo que aquellos eventos que son sin ánimo de lucro no aparecen en este tipo de páginas.

Otras páginas como Webtenerife, La Agenda o La Opinión, nos informan de los acontecimientos en las Islas Canarias, por lo que esta información es útil solamente para personas que visiten o viven en esta ubicación.

Como podemos ver existen multitud de páginas donde informarnos sobre eventos, sin embargo, o carecen de una interfaz, usabilidad y experiencia de usuario adecuada , o no hay una información centralizada, es decir, encontrar eventos de cualquier parte del mundo de forma unificada, sin tener que acceder a diferentes páginas.

Capítulo 2

Objetivos

El objetivo principal del proyecto es crear una aplicación web limpia, que proporcione una buena experiencia de usuario (UX), donde poder crear y visualizar diferentes eventos en cualquier lugar del mundo.

2.1. Subobjetivos

El objetivo principal del trabajo se puede desgranar en los siguientes subobjetivos:

- **OBJ-1:** Proporcionar un mapa para localizar los eventos de forma visual.
- **OBJ-2**: Creación de un perfil personal del usuario para visualizar y editar sus eventos creados.
- **OBJ-3**: Proporcionar una forma sencilla de crear eventos.
- **OBJ-4**: Crear filtros para personalizar la búsqueda de los eventos.

2.2. Tareas para cumplir los objetivos

- **TA-1:** Implementar un mapa y listado donde visualizar los eventos.
- **TA-2**: Crear un backend con una API REST simple para que pueda ser reutilizado y ampliado si fuera necesario.
- **TA-3**: Configurar los sistemas de integración y despliegue continuo.
- **TA-4**: Configurar entornos de desarrollo.
- **TA-5**: Implementación de un formulario para la creación de eventos.
- **TA-6**: Desarrollar un sistema de autenticación y gestión de perfiles de usuarios.

Capítulo 3

Planificación

Este proyecto estará compuesto por 3 fases fundamentales: análisis, diseño y desarrollo. Se desglosan de la siguiente manera:

- Análisis:
  - Investigación de las actuales aplicaciones relacionadas.
  - Especificación de requisitos.
  - Estudio del presupuesto.
- Diseño:
  - Diseño de la interfaz.
  - Diseño de la infraestructura.
- Desarrollo:
  - Configuración de entornos.
  - Desarrollo del back.
  - Desarrollo del front.
  - Testing.
  - Memoria.

En la imagen 3.1 encontramos un diagrama Gantt con una planificación de tiempo de las tareas:

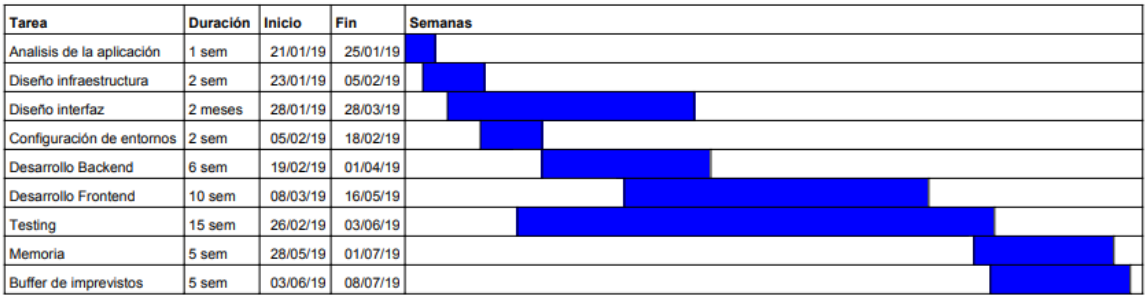

Figura 3.1: Diagrama Gantt

Las estimaciones de tiempo son trabajando seis horas diaria, excluyendo fines de semana. Como podemos observar existen tareas que se realizan al mismo tiempo, con lo cual la dedicación a cada una de ellas se hará por porcentaje, en orden de dependencias entre ellas, o en su defecto por prioridades.

Capítulo 4

Análisis

En este capítulo haremos un análisis de la aplicación. Primero analizaremos la aplicación y recogeremos tanto los requisitos funcionales como los no funcionales. Luego se hará diseñará de un diagrama de caso de uso para estudiar la intención que tiene el actor (el usuario) al interactuar con la web y así centrarnos en sus necesidades.

4.1. Especificación de requisitos

4.1.1. Requisitos funcionales

1. Gestión de usuarios:
    - Autenticación de usuarios.
2. CRUD de eventos:
    - Creación de eventos.
    - Listado de los eventos.
    - Edición de los eventos creados.
    - Borrarlos eventos creados

4.1.2. Requisitos no funcionales

- El diseño de la aplicación debe ser simple e intuitiva.
- Aplicación adaptada a daltónicos.
- Rapidez al mostrar los eventos, es decir, un tiempo inferior a 5 segundos.
- Mapa para visualizar eventos de forma geolocalizada.
- La aplicación debe poder ser usada en móvil.

4.2. Diagrama de caso de uso

El funcionamiento de la aplicación se muestra en la imagen 4.1 a través de un diagrama caso de uso, cuyo actor es el usuario que interactúa con la aplicación.

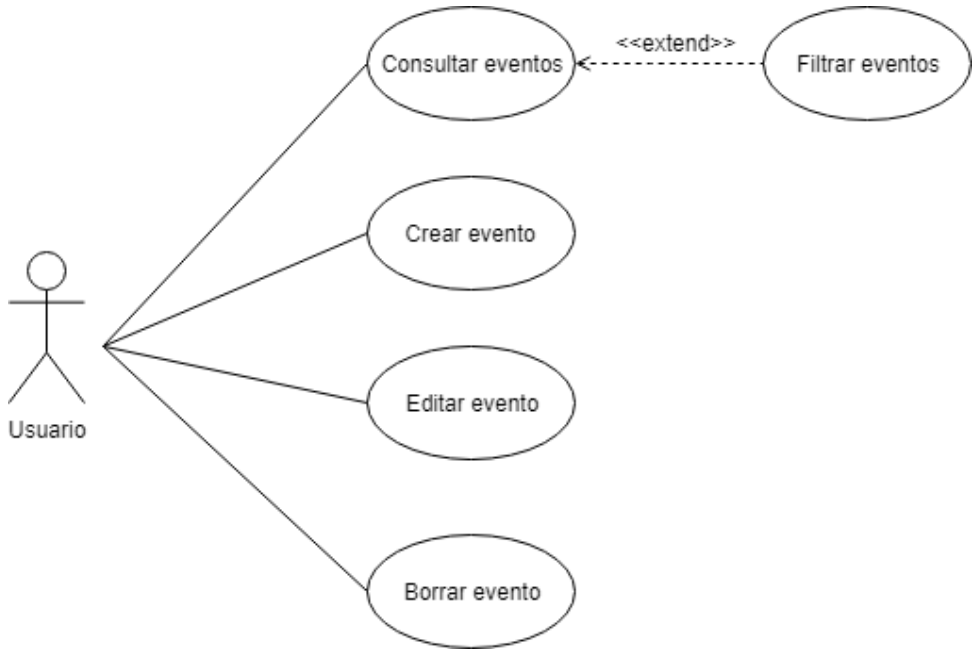

Figura 4.1: Diagrama caso de uso

Capítulo 5

Presupuesto

5.1. Presupuesto

En la siguiente tabla se muestra el presupuesto de un año de la aplicación, suponiendo un registro de alrededor de cinco mil usuarios, y un tráfico de diez mil personas consultando eventos al mes.

Concepto	Precio en euros por mes	Precio en euros por año
Desarrollador	2.100 €	25.000 €
AWS	182 €	2.184 €
Auth0	54 €	648 €
Cloudinary	80 €	960 €
Autocomplete Google	16 €	192 €
Geocoding Google	5 €	60 €
Total	**2.437 €**	**29.244 €**

Tabla 5.1: Tabla de presupuesto

Se ha calculado un salario de dos mil cien euros al mes para el desarrollador. Se ha contemplado el pago de los servicios externos, comprobando el precio en sus páginas webs oficiales, aunque este presupuesto podría variar, si alguno de los precios establecidos cambiara.

Dado el tráfico de personas usando la web se ha calculado mediante la calculadora de Amazon Web Services un servidor de ciento ochenta y dos euros al mes.

Teniendo en cuenta el salario del desarrollador, el precio del servidor y de los servicios externos, el precio de la aplicación al mes sería de un aproximado de dos mil cuatrocientos treinta y siete euros. Al año se calcula un total del veintinueve mil doscientos cuarenta y cuatro euros.

5.2. Visión de negocio

Teniendo en cuenta el gasto que supone el mantenimiento y desarrollo de la aplicación, a continuación se dan varias ideas para rentabilizarla:

- Introducir publicidad en la aplicación.
- Limitar la creación de eventos a usuarios específicos que paguen una cuota mensual o un paquete de eventos a crear.
- Realizar además ventas de entradas a comisión.
- Eventos promocionados.

Capítulo 6

Diseño

En este capítulo hablaremos tanto del diseño de la interfaz de la aplicación como el de su infraestructura sin profundizar en las tecnologías utilizadas.

6.1. Diseño aplicación

El diseño de la aplicación se ha llevado a cabo siguiendo un proceso iterativo e incremental, es decir, se ha diseñado un boceto inicial, y poco a poco ha ido evolucionando en función de las opiniones de los usuarios. Este proceso se ha llevado a cabo para proporcionar un resultado completo sobre el producto final.

Como podemos ver en la imagen 6.1 al principio se diseñó un prototipo a mano alzada, donde se podía observar la funcionalidad principal de la aplicación, visualizar eventos. A lo largo de las iteraciones, el diseño y funcionalidades se fueron concretando hasta obtener el resultado final de la aplicación como podemos ver en la imagen 6.2 y la imagen 6.3, logrando así cumplir con los requisitos establecidos.

1. Iteración inicial del diseño:

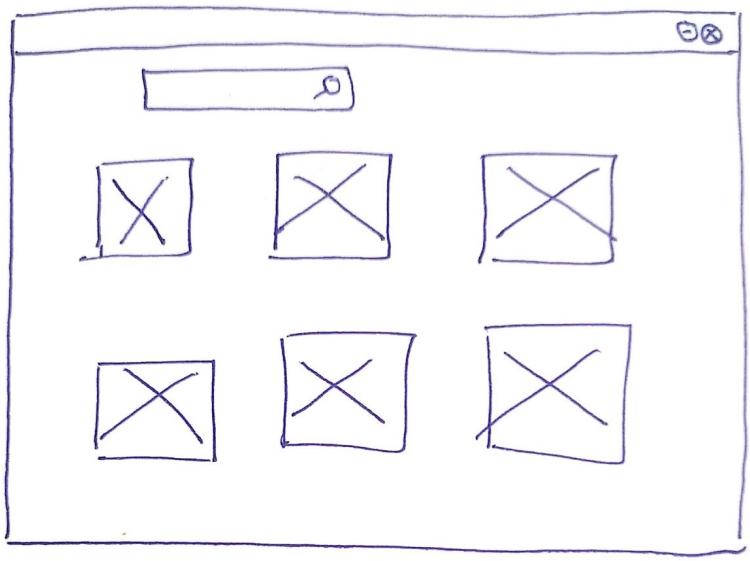

Figura 6.1: Diseño inicial de la aplicación

2. Iteración intermedia del diseño:

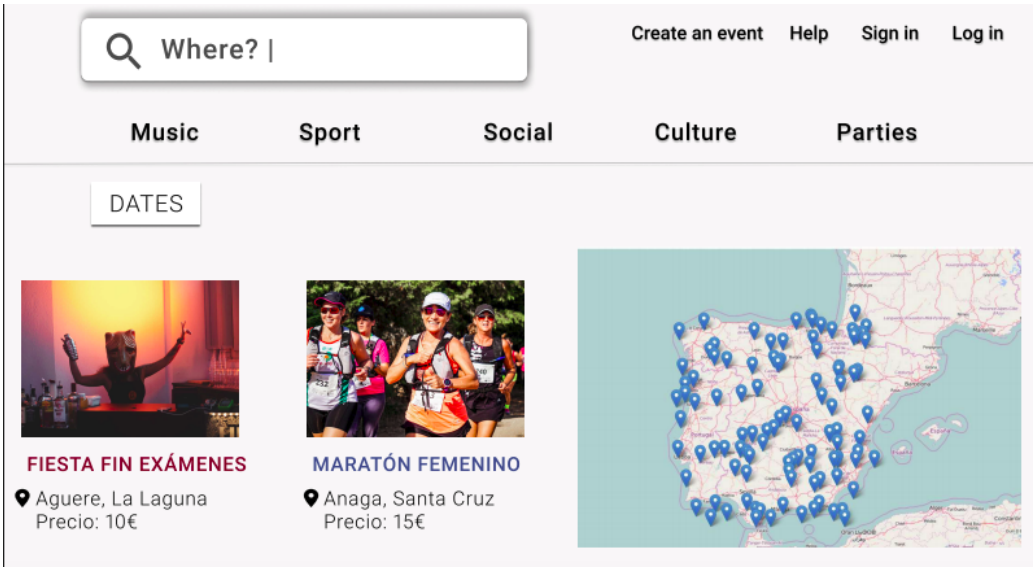

Figura 6.2: Diseño intermedio de la aplicación

3. Iteración final del diseño:

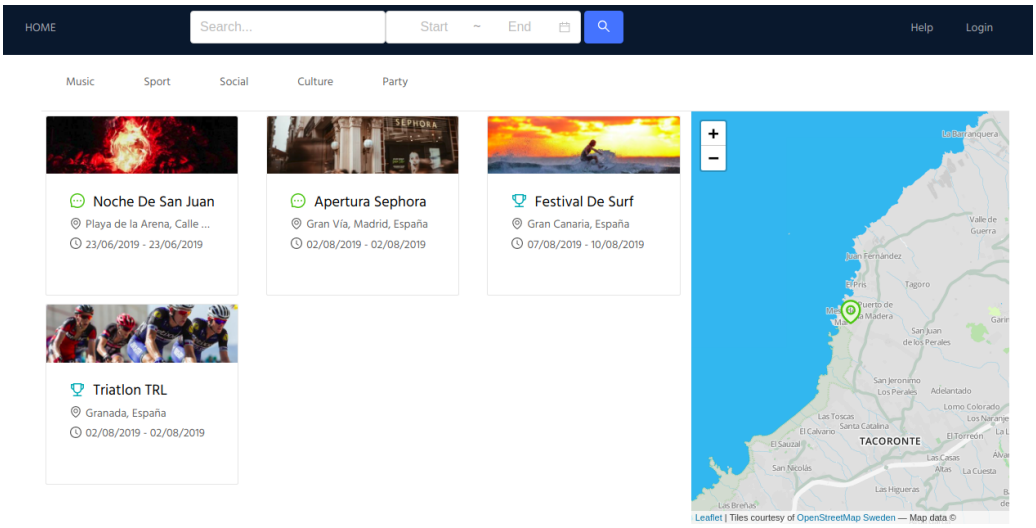

Figura 6.3: Diseño final de la aplicación

Además del diseño, se ha estudiado cómo tener una óptima experiencia de usuario. Para ello la aplicación sigue los siguientes principios:

- Confianza: Para el uso de la aplicación no es necesario registrarse a la misma, ya que en ciertas ocasiones el usuario es reacio a dar sus datos cuando no son necesarios.
- Utilidad: La aplicación cubre satisfactoriamente las necesidades de los usuarios.
- Usabilidad: La aplicación tiene un comportamiento responsive para que pueda ser utilizada desde cualquier dispositivo.
- Accesible para daltónicos: cumple con las pautas WCAG [1] necesarias para que la página pueda ser usada por daltónicos.

- La presentación visual de texto tiene una relación de contraste de más de 4.5:1 para todas las secciones.
- No se utiliza el color como único medio visual para transmitir información. Por ejemplo, la temática de los eventos no se indica solo a través de colores, sino también con diferentes iconos. Otro ejemplo es el formulario de creación de un evento, cuando hay un error no se indica únicamente con un color sino además nos muestra un mensaje.

6.2. Diseño de la infraestructura de la aplicación

La imagen 6.4 muestra de forma gráfica la intraestructura de la aplicación

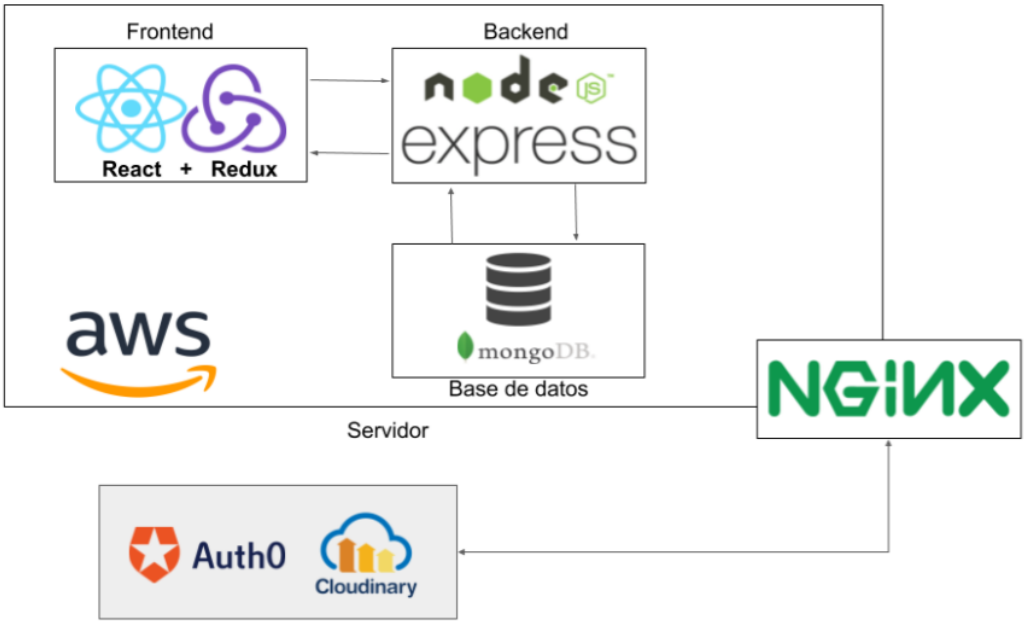

Figura 6.4: Infraestructura de la aplicación

Todo el conjunto de la aplicación hace uso del lenguaje TypeScript. Para el frontend se ha escogido el uso de React con Redux, para desarrollar una interfaz basada en componentes que nos permite la reutilización de código. El front hará peticiones a una API REST implementada en Node.js con Express.

El backend estará conectado a mongoDB, una base de datos no relacional, donde alojaremos todos los datos de nuestros eventos. Tanto el front como el back estarán corriendo en contenedores de Docker y todo ello se encontrará detrás de Nginx funcionando como un proxy inverso en un servidor de Amazon(AWS) [2].

Nuestra aplicación hará uso a su vez de dos servicios externos:

- **Auth0:** se ha usado este servicio de autenticación para garantizar que los datos de usuario se almacenan de forma segura, ya que la gestión de estos datos y la autenticación se trata de un aspecto sensible, debido a que día a día se encuentran nuevas vulnerabilidades en los sistemas, y además, estos desarrollos deben seguir numerosas leyes como la GDPR. Por esas razones se ha decido usar Auth0 como servicio externo para la autenticación de usuarios.

- **Cloudinary:** este servicio se utiliza para el almacenamiento y tratamiento de las imágenes que contendrá la aplicación.

Capítulo 7

Implementación

Durante este capítulo abarcaremos todo lo relativo a la implementación de la aplicación, que se ha desarrollado con el stack MERN. "MERN"es el acrónimo de cuatro tecnologías: MongoDB, Express, React y Node.js. Explicaremos el porqué se ha escogido cada una de ellas y como se han usado.

7.1. Control de versiones

Como herramienta para el control de versiones de nuestro código se ha utilizado Git y seguido un flujo de trabajo llamado GitFlow que especifica como deben tratarse las diferentes ramas en un repositorio. Durante el desarrollo se ha trabajado de forma diferenciada entre el backend y frontend con Gitlab [10], cada uno de ellos cuenta con dos ramas base, Máster, donde se aloja el código estable y con características completas que es lo que se despliega a producción, y Develop, en la cual se encuentra la última versión en pruebas de la aplicación.

Cuando desarrollamos una nueva tarea o funcionalidad se crea una rama con el nombre feature/nombreTarea a partir de la rama Develop, cuando el desarrollo esté finalizado se mezcla con Develop.

En la imagen 7.1 podemos ver de forma resumida la estructura de ramas llevada a cabo para el desarrollo.

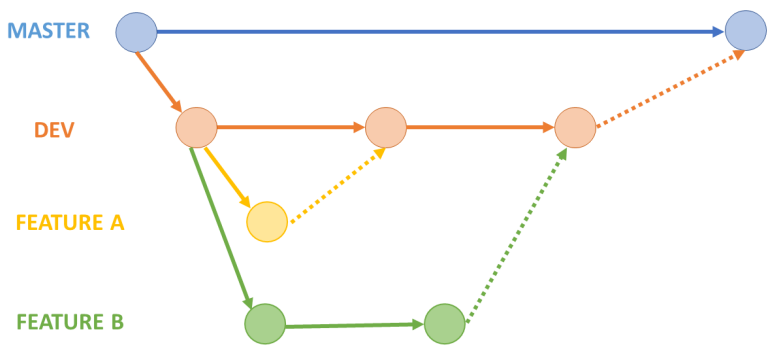

Figura 7.1: Estructura en átomos

7.2. Base de datos

Para el almacenamiento de datos se ha utilizado una base de datos no relacional, MongoDB. Las bases de datos MongoDB[3] almacenan documentos en colecciones con un formato similar a JSON, llamado BSON, y no registros como sucede en las tablas de las bases de datos relacionales. Para mapear los datos de la base de datos a entidades de la aplicación se ha utilizado como ODM (Object Data Transfer) Moongose[4], esto da la posibilidad de crear consultas complejas de forma sencilla e hidratar entidades de forma automática.

Se ha escogido MongoDB con Mongoose por los siguientes motivos:

- Su gran comunidad y nivel de participación. Lo que significa una mayor documentación y una resolución de problemas más rápida al tener más información y foros.
- Su gran velocidad. Necesitamos una base de datos que nos permita guardar y consultar eventos por geolocalizacion en el menor tiempo posible.
- Su flexibilidad. En las bases de datos relacionales el esquema de la información está minuciosamente escogido de antemano y puede llegar a ser dificil de modificar, por ello, dado que el esquema de los datos de la aplicación podría ir evolucionando, se ha escogido una base de datos no relacional.

7.2.1. Esquema de los datos en Mongo

A continuación se muestra una tabla de como se han almacenado los eventos:

Nombre	**Tipo de dato**
eventName	String
update	Date
created	Date
eventAddress	String
eventStartDate	Date
eventEndDate	Date
eventDescription	String
eventLocation	String
eventTopic	EventTopicType
eventImages	Array<string>
userId	String

Tabla 7.1: Esquema de los datos en Mongo

Cada uno de los datos tiene una validación realizada a nivel esquema.

7.3. Backend

El backend de la aplicación está programado en su totalidad en TypeScript. Esta parte de la aplicación se encarga del manejo y almacenamiento de los datos dados por el front.

Este backend consiste en una API REST para almacenar y extraer información de los eventos, creada sobre Node.js con el framework Express que nos facilita la creación de APIs. Estas dos tecnologías han sido escogidas principalmente por su gran comunidad y su rendimiento.

El funcionamiento del backend es simple. Tenemos un servidor Node.js[5] corriendo con Express[6]. Este se conecta con la base de datos. Cuando se recibe una petición, según la ruta y el verbo HTTP, accedemos a un controlador u otro, que se encargará de llamar a las funciones y realizar las consultas pertinentes a la base de datos, y este nos devolverá los datos pedidos.

Nuestra API REST se divide en varios endpoints que se muestran en la siguiente tabla:

Método	**Path**	**Descripción**
GET		
	/events/	Obtenemos todos los eventos de la aplicación
	/events/:id	Obtenemos el evento cuyo id coincida con el pasado por parámetro
	/topics/	Obtenemos todos los topics
POST		
	/events/	Registra un nuevo evento
	/events/bulk	Registra un conjunto de eventos
PUT		
	/events/:id	Se actualiza el evento cuya id coincida con la pasada por parámetro
DELETE		
	/events/:id	Elimina el evento cuyo id coincida con el pasado por parámetro

Tabla 7.2: Endpoints de la API REST

7.4. Frontend

Nuestro frontend es una SPA (Single Page Application), que se ha desarrollado en el lenguaje TypeScript con React y, además se ha hecho uso de diversos paquetes adicionales, destacando Redux.

En el front disponemos de un fichero HTML que contiene la llamada a un fichero Javascript donde estará definida toda la aplicación, y según la ruta se inyectarán diferentes componentes.

La aplicación mantiene un estado global del que leen todos los componentes, según la filosofía Redux.

7.4.1. React

React [7] es una biblioteca escrita en JavaScript, desarrollada por Facebook para facilitar la creación de componentes reutilizables. Algunos de los beneficios más destacados del uso de React son los siguientes:

- Uso de componentes que nos permite reutilizar código y hacer de forma sencilla una escalabilidad de la aplicación
- DOM virtual. La principal ventaja de React es poder generar el DOM (estructura de los elementos que se generan en el navegador web al cargar una página) de forma dinámica. Esto permite que para poder visualizar los cambios de los datos, no es necesario renderizar toda la página de nuevo, sino solamente el componente que haya sido actualizado. Lo que permite una mayor rapidez de carga y con ello una mejor experiencia de usuario.
- Gran comunidad.

En la aplicación hemos utilizado dos tipo de componentes StateFull, que son los componentes conocedores del estado de la aplicación y que contienen lógica, y los React Stateless que solamente se encargan de recibir datos y mostrarlos.

Se ha llevado a cabo un diseño atómico, este se encarga de dividir la página por elementos pequeños e individualizados que no tienen sentido por sí solo. Estos se acoplarán entre sí para crear elementos más complejos. La jerarquía que se sigue es la siguiente: Átomo >Molécula >Organismo >Página.

Enmarcado en verde en la imagen 7.2 se muestran alguno de los átomos de la pantalla principal.

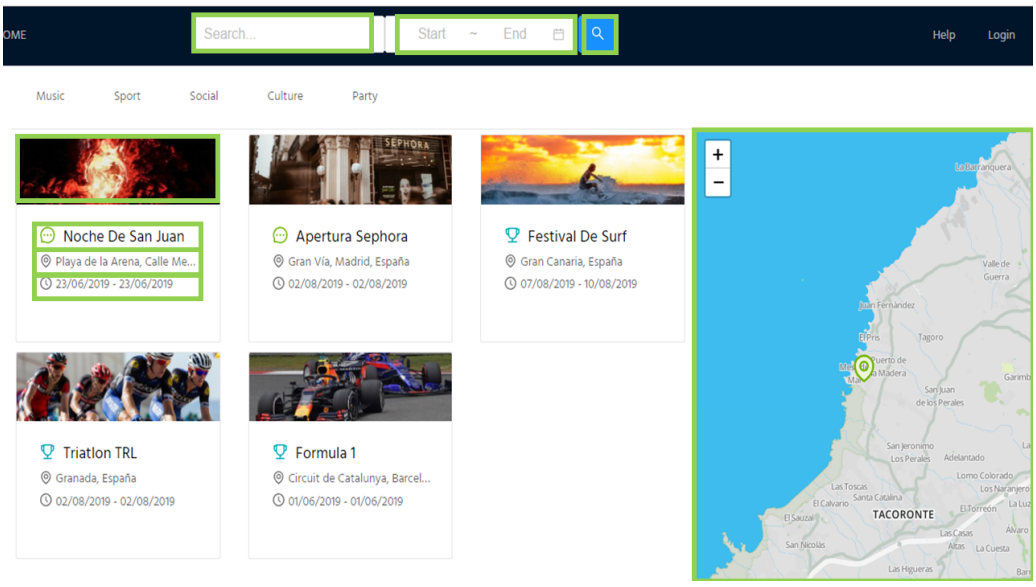

Figura 7.2: Estructura en átomos

Gracias a la unión de varios átomos creamos moléculas que es las que están resaltadas en la imagen 7.3 en color violeta.

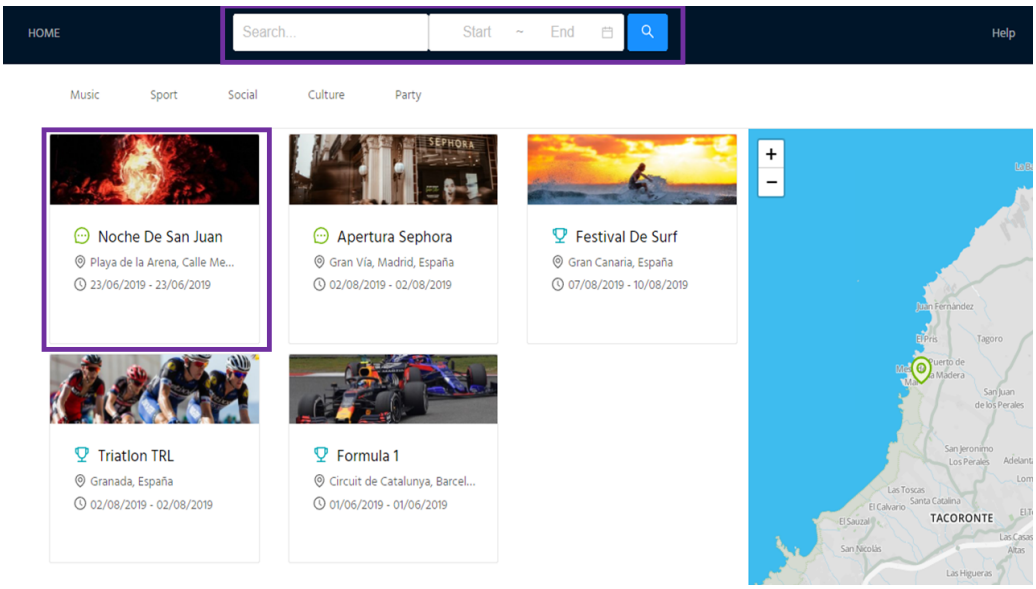

Figura 7.3: Estructura en moléculas

Con la unión de varias moléculas formamos organismos, que son los que se encuentran de color azul en la imagen 7.4.

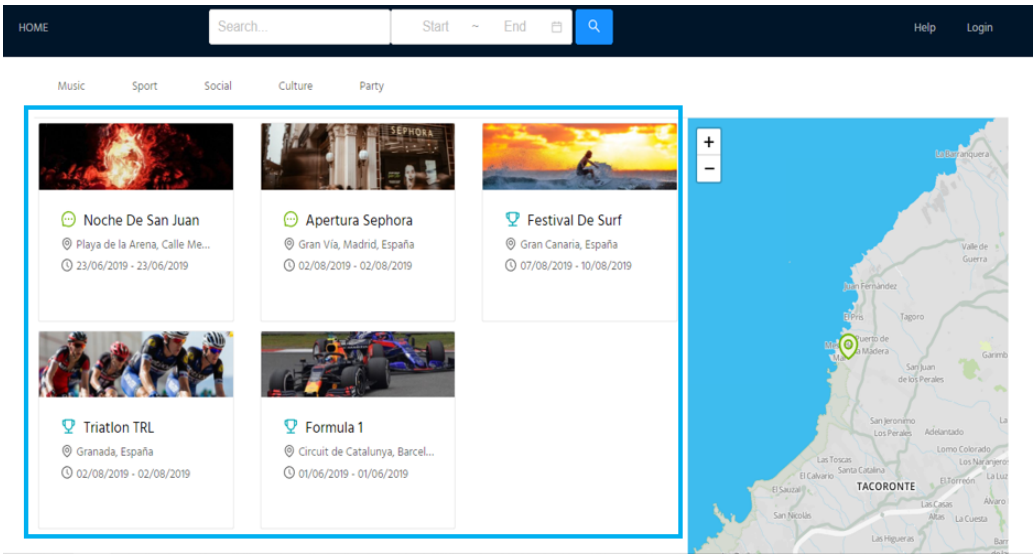

Figura 7.4: Estructura en organismos

Por último tendremos las páginas con la unión de todos los elementos creados como podemos ver en rojo en la imagen 7.5.

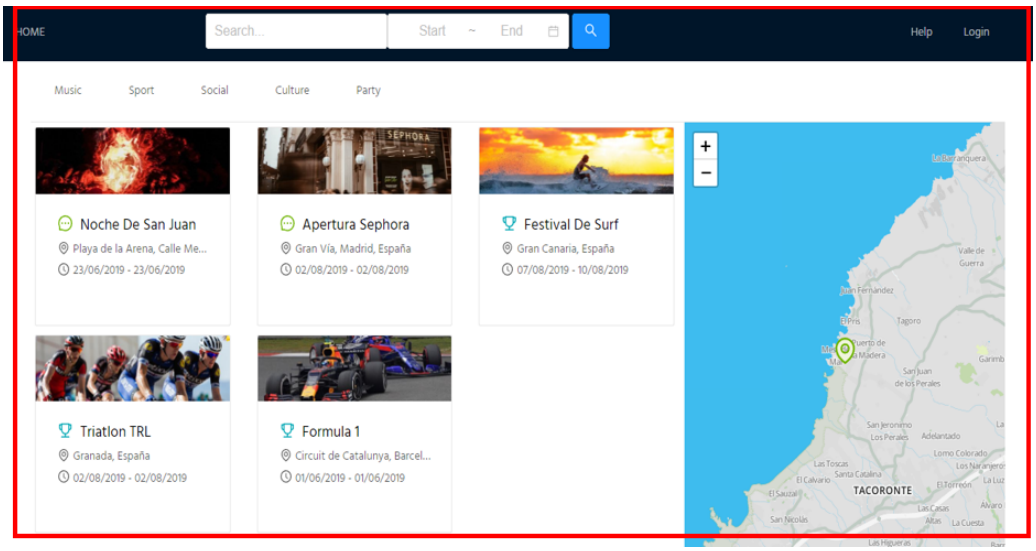

Figura 7.5: Estructura en páginas

7.4.2. Redux

Al tratarse React de una biblioteca, podemos agruparla junto con otras como Redux. Tanto Redux como React trabajan con los estados de la aplicación. Aunque los componentes de React pueden tener su propio estado y que este se vaya modificando a lo largo de su ciclo de vida, utilizaremos Redux para mantener un estado global y permitir una comunicación sencilla entre los diferentes componentes.

Redux[8] se basa en que sólo tenemos una fuente de verdad,el store, y que además el estado que almacena ese store es inmutable, si necesitamos realizar algún cambio, creamos una nueva instancia del mismo.

Para realizar cualquier cambio en el estado de la aplicación, debemos lanzar una acción. Dicha acción es recogida por un reducer, que haciendo uso de los datos proporcionados por la acción y el estado anterior, genera un nuevo estado con el que se actualiza el store.

De esta manera tendremos claro de dónde debemos extraer la información de los estados en los componentes de React, y así evitamos duplicidades, efectos colaterales indeseados y facilitamos la depuración.

El flujo de Redux es el que se muestra en la imagen 7.6.

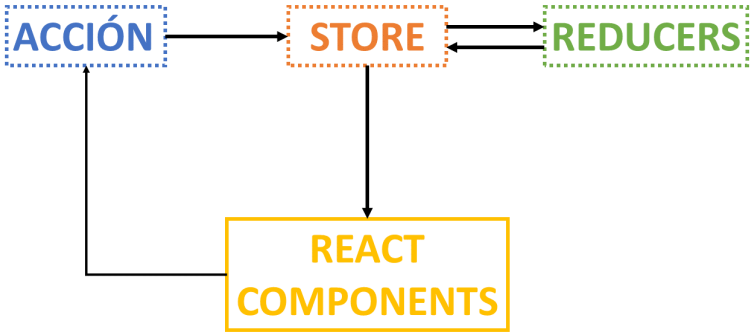

Figura 7.6: Flujo Redux

1. El componente recibe un evento y emite una acción.
2. Esta acción, se pasa a el store, que es donde se guarda el estado único.
3. El store comunica la acción junto con el estado actual a los reducers.
4. Los reducers, devuelven un nuevo estado, probablemente modificado en base a la acción.
5. Los componentes se actualizarán en base al nuevo estado.

7.5. Integración continua y despliegue continuo

7.5.1. Integración continua

La integración continua, también conocida por sus siglas CI (Continuos Integration) consiste en hacer verificaciones automáticas de un proyecto cada vez que realizamos cambios en el código para garantizar que una modificación no rompe el comportamiento de la aplicación. Esto es especialmente útil e importante cuando trabajan varios desarrolladores al mismo tiempo en la aplicación, ya que se garantiza el correcto funcionamiento del código, de forma rápida, lo que nos permitirá un desarrollo ágil.

El procedimiento para llevar a cabo la integración continua se ejecutará a través del propio Gitlab, añadiendo un archivo de configuración `gitlab-ci.yml`. Cada vez que se realice una subida de código o una mezcla de ramas en nuestra aplicación, este será el encargado de que se ejecuten los test y el análisis de código, y verificar así que la aplicación sigue funcionando.Esto nos evitará mezclar código con errores y bugs en producción.

7.5.2. Despliegue Continuo

El despliegue continuo, también conocido por las siglas CD (Continuos Deploy) se encarga de proporcionar una manera ágil y fiable de poder entregar o desplegar los nuevos cambios en el servidor para que lleguen a los clientes.

Este paso va ligado al de la integración continua. Una vez se ha pasado con éxito la fase de integración continua, llega el momento de desplegarla. En este caso el despliegue se realiza de forma automática en el servidor cuando el código que haya pasado la integración continua sea de la rama máster. Esta configuración se ha indicado en el mismo fichero de configuración `gitlab-ci.yml`.

Capítulo 8

Calidad del código y pruebas

Para mantener la integridad del funcionamiento de de la aplicación, ya que esta está en continuo desarrollo, es imprescindible realizar pruebas automáticas en cada versión, para garantizar así el correcto funcionamiento de la aplicación y asegurarnos una adecuada calidad del código.

8.1. Test unitarios

Los test unitarios son es un nivel de pruebas de software en el que se prueban las unidades individuales de un código. El propósito aislar cada parte del programa y validar que las partes individuales son funcionan correctamente.

Para realizar los test unitarios se ha utilizado el framework Jest[20] que nos permite testear sin previa configuración. En las siguientes imágenes se muestran los test unitarios realizados tanto para el front 8.1 como para el back 8.2.

```
PASS  src/test/ActionTypes.test.tsx
PASS  src/test/molecules/AddEvent.test.tsx
PASS  src/test/molecules/SearchForm.test.tsx
PASS  src/test/organisms/EventDetailsContainer.test.tsx
PASS  src/test/molecules/SelectTopic.test.tsx
PASS  src/test/atoms/ScrollToTop.test.tsx
PASS  src/test/pages/ProfilePage.test.tsx
PASS  src/test/atoms/EventImagenes.test.tsx
PASS  src/test/atoms/SearchButton.test.tsx
PASS  src/test/molecules/UserProfile.test.tsx
PASS  src/test/atoms/SearchBar.test.tsx
PASS  src/test/atoms/LocationSearchInput.test.tsx
PASS  src/test/molecules/EditEventForm.test.tsx
PASS  src/test/molecules/EventDetailsText.test.tsx
PASS  src/test/App.test.tsx
PASS  src/test/molecules/CardEvent.test.tsx
PASS  src/test/organisms/TopBar.test.tsx

Test Suites: 17 passed, 17 total
Tests:       64 passed, 64 total
Snapshots:   0 total
Time:        11.446s
Ran all test suites.
```

Figura 8.1: Tests unitarios front

```
PASS  __tests__/Events.test.ts
  Events
    ✓ should insert a doc into collection (99ms)
    ✓ should POST an event (1316ms)
    ✓ should UPDATE (PUT) an event (43ms)
    ✓ should DELETE an event (48ms)
    ✓ should GET all events (76ms)
    ✓ should GET all topics (23ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.484s, estimated 4s
Ran all test suites.
```

Figura 8.2: Tests unitarios back

8.2. Test End-to-end

Los test End-to-end o E2E son unas pruebas que nos permiten testear la aplicación de principio a fin. Simula el comportamiento de un usuario al utilizar la aplicación, comprobando que todas las funcionalidades trabajan correctamente. Para la implementación de estos test se ha utilizado el framework Cypress[21] que permite escribir tests de forma fácil y rápida y ver los test en tiempo real. En la imagen 8.3.se muestran los distintos tests realizados en la aplicación:

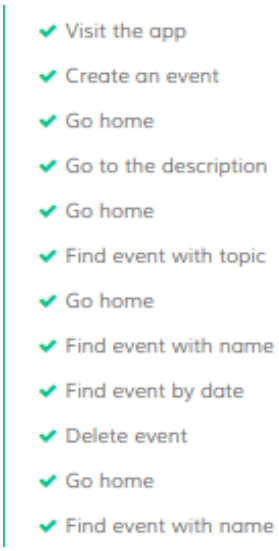

Figura 8.3: Tests con Cypress

8.3. SonarQube

SonarQube[22] es una plataforma para evaluar código fuente gracias a un conjunto de requisitos que indican si una nueva versión de un proyecto cumple unos estándares de calidad y por lo tanto puede pasar a producción o no. Gracias a SonarQube podremos mejorar la calidad de nuestro código logrando menor porcentaje de vulnerabilidades y bugs.

Tras finalizar el desarrollo se introdujo Sonar en el proyecto, dándonos los siguientes resultados iniciales en el front 8.4 y en el back 8.5:

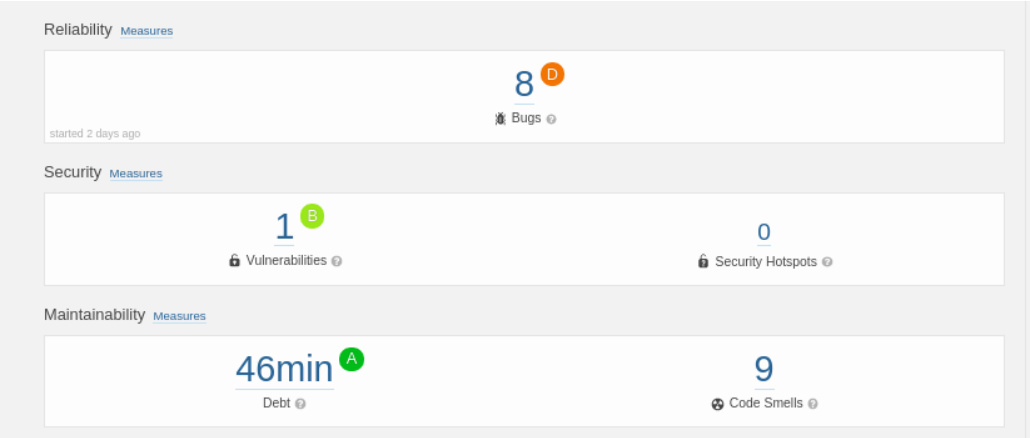

Figura 8.4: Resultados Sonar iniciales para el front

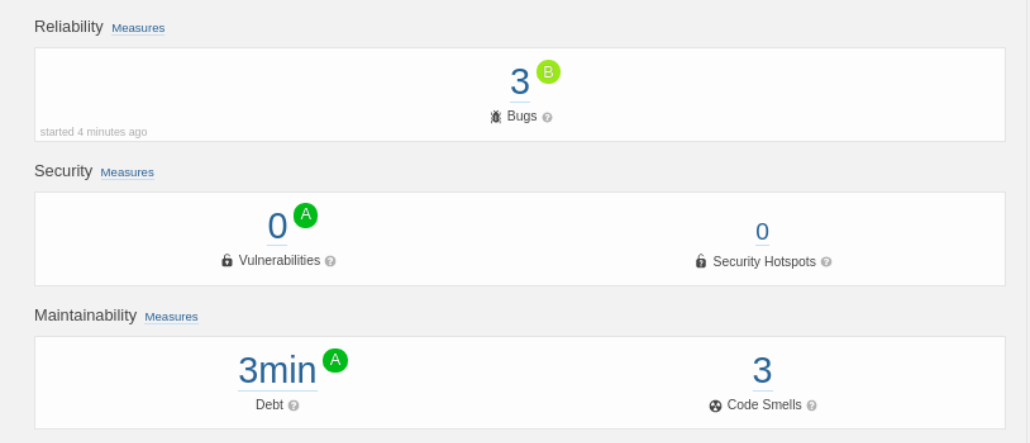

Figura 8.5: Resultados Sonar iniciales para el back

Tras modificar el código hemos logrado eliminar casi en su totalidad los bugs y vunerabilidades del front 8.6 y del back 8.7 como se muestra a continuacón:

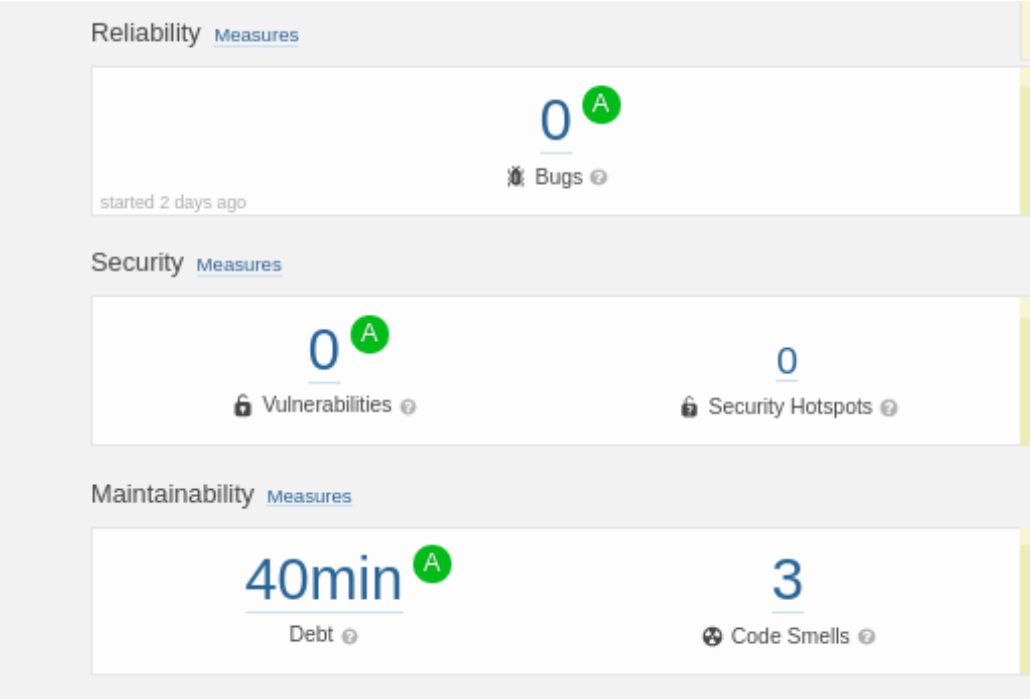

Figura 8.6: Resultados Sonar finales para el front

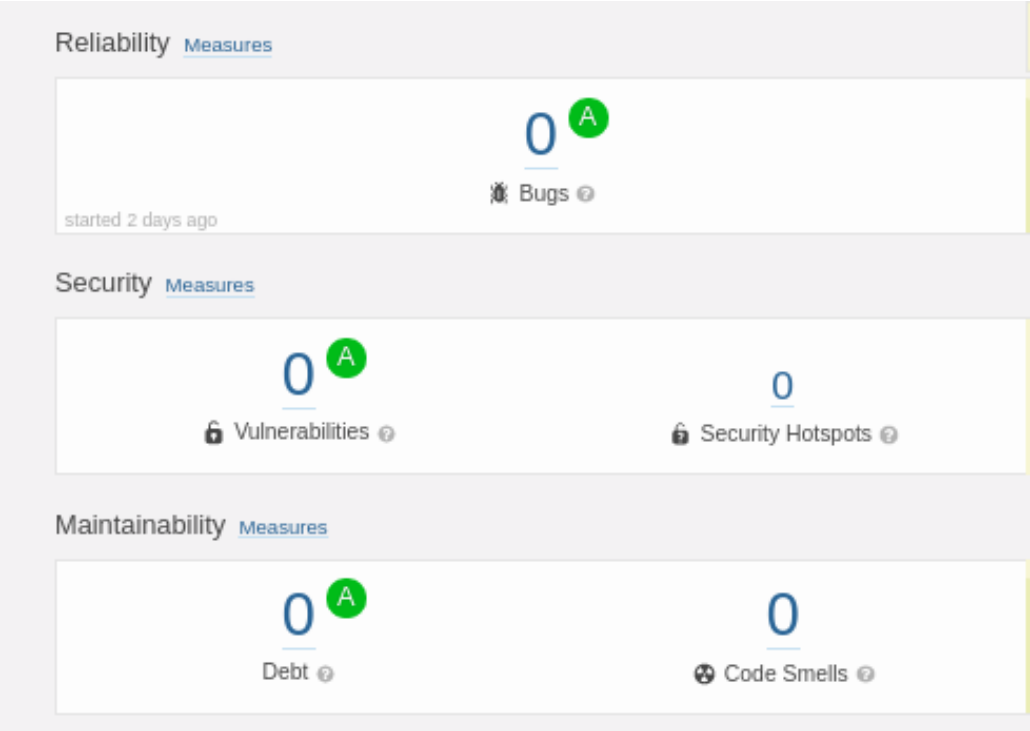

Figura 8.7: Resultados Sonar finales para el back

8.4. Linter

Un linter es un programa que comprueba si un código se ajusta a una serie de reglas que se fijan, señalando errores y posibles bugs, mejorando así la calidad del código. En nuestro proyecto se ha utilizado ESLint[24] que es un linter para JavaScript.

8.5. Husky

Husky [23] permite lanzar los hooks de Git de forma sencilla. Los hooks de Git son scripts que se ejecutan automáticamente cuando suceden ciertas acciones importantes. En este este proyecto se ha configurado para que cuando se realice un commit, ponga el código en el formato correcto gracias a prettier.

Capítulo 9

Herramientas

En ese capítulo se explicarán las herramientas utilizadas en el desarrollo.

- **Ant design.**[11] Es un paquete que proporciona una serie de componentes con estilos que ayudan a la creación de una aplicación con un estilo uniforme. Nos ha permitido tener un página limpia y clara con un buen diseño.
- **React-Router-dom.** [12] Es un paquete que mapea las URL a los distintos componentes de la aplicación.
- **OpenStreetMap.**[13] Es un mapa del mundo de uso libre bajo una licencia abierta.
- **Leaflet.**[14] Es una biblioteca de código fuente abierto de JavaScript que se utiliza para crear aplicaciones de mapas interactivos. En nuestra aplicación se le ha añadido una capa de OpenStreetMap.
- **Create React App.**[15] Es una herramienta que nos permite crear aplicaciones React de una sola pagina sin configuración. Hace uso de Webpack, Babel, EsLint, entre otras herramientas.
- **Auth0.**[16] Es un servicio de autenticación que nos permite proteger nuestra página, sin tener que convertirnos en expertos en seguridad. Cada vez que un usuario intente autenticarse en nuestra página, Auth0 verificará su identidad y enviará la información requerida a la aplicación.
- **Cloudinary.**[17] Es un servicio para la gestión de imágenes y vídeos basada en la nube. Permite cargar, almacenar, administrar y manipular imágenes. Además provee funcionalidades avanzadas como el uso de inteligencia artificial para recortar la imagen por los puntos que nos proporcionan información.
- **Docker.**[18] Es una tecnología para la gestión y creación de contenedores. La finalidad de Docker es crear contenedores ligeros y portables para que las aplicaciones software puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues. (https://www.docker.com/)
- **Nodemon.**[19] Servicio que monitoriza los cambios de los archivos y reinicia el servidor de Node.js automáticamente.

Capítulo 10

Problemas durante el desarrollo

Durante el desarrollo del proyecto, este se vio truncado por varias problemáticas. Que se explican durante este capítulo.

10.1. Planificación

Se realizaron planificaciones no ajustadas a la realidad debido a estimaciones erróneas de tiempo o imprevistos. Cuando se realizó la estimación de horas del proyecto se calculó seis horas de trabajo diarias, pero debido al comienzo de un trabajo como desarrolladora, se tuvo que hacer una nueva planificación, impidiendo que se acabara en el tiempo calculado.

10.2. Falta de experiencia

La falta de conocimiento en las tecnologías utilizadas, y sobre todo en programación asíncrona (necesaria por la naturaleza de Node.js) hizo que se retrasara mucho el comienzo del desarrollo de la aplicación. Y que la implementación de este fuera lenta. Por lo que se tuvo que, primero leer la documentación oficial de cada una de las tecnologías antes de ponerse a planificar.

10.3. Autenticación de usuarios

Al comienzo se optó por la creación de un sistema de autenticación propio. Una vez comenzó el desarrollo de este surgieron dudas sobre la seguridad por el almacenamiento de contraseñas o tratar los datos de los usuarios. Tras valorar la complejidad de implementación y la sensibilidad de la información se decidió externalizar la autenticación a Auth0.

10.4. Multitud de paquetes NPM

React por si solo no proporciona todas las características necesarias para llevar a cabo la aplicación que teníamos como objetivo, por lo que conforme hemos ido realizando la aplicación se han añadido nuevos paquetes para aumentar la funcionalidad. A pesar de ser una gran ayuda, existen múltiples paquetes npm [25] y numerosas versiones de los

mismos, lo cual supone una gran dificultad a la hora de elegir el que mejor se adaptaba a tu problema, y además contemplar las incompatibilidades.

10.5. Los estados en React

Cuando una aplicación se hace muy grande como en este caso, llegamos a un punto, en el que un componente debe modificar el estado de varios componentes, lo que hace un código complejo de entender, y que la gestión de un estado se vuelva una tarea compleja. Esto hizo, entre otras razones, que se decidiera hacer uso de Redux y así evitar todos estos problemas.

Capítulo 11

Resultados finales

En el siguiente capítulo se mostrará el resultado final de la página mediante imágenes, explicando las funcionalidades de la aplicación.

Para acceder a la aplicación lo podemos hacer mediante la siguiente url `www.mytfg.me` o `mytfg.me`.

11.1. Pantalla principal

Cuando accedemos a la aplicación la pantalla principal es la que se muestra en la imagen 11.1.

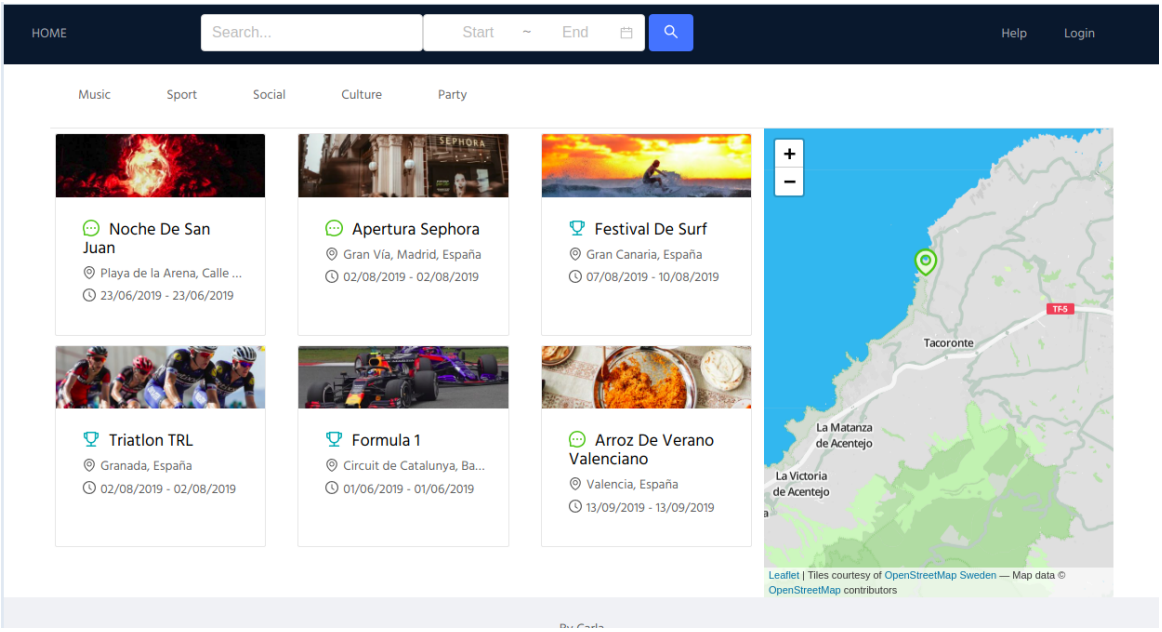

Figura 11.1: Pantalla principal de la aplicación

En la parte superior se encuentra la barra de búsquedas para poder hacer el filtrado de eventos. A la izquierda está el botón de HOME, que es para que nos redirija a esta pantalla independientemente de donde estemos. Por el lado derecho encontramos el botón de login que nos permite registrarnos en la aplicación para poder empezar a crear eventos. Bajo esta barra se muestra un listado de temáticas, que nos permite filtrar los eventos por las mismas.

En la parte central de la página vemos unas cartas, cada una de ellas contiene diferentes eventos con una imagen descriptiva del evento, su nombre, ubicación y fecha, además delante del nombre hay un pequeño símbolo que indica en color y forma de que temática trata.

A la derecha de la página toma protagonismo un mapa en el cual vemos de forma localizada todos los eventos creados y si clickamos en la chincheta se desplegará un pop-up con la imagen del evento como vemos en la imagen 11.2.

Figura 11.2: Pop-up evento

Según nos vamos alejando del mapa se van agrupando las chinchetas mostrándonos el número de eventos por esa zona, a medida que nos acercamos esta agrupación se va disolviendo dándonos una localización cada vez más exacta. Lo podemos ver de forma visual en la imagen 11.3.

Figura 11.3: Agrupación de eventos

Los eventos se podrán filtrar, permitiéndonos ver solo aquellos eventos que nos interesan. Las tres formas de filtrado, pudiendo ser de manera simultánea son:

- Nombre.
- Lugar.
- Rango de fechas.
- Temática.

11.2. Descripción del evento

Al seccionar uno de los eventos de la pantalla principal, este nos llevará a una página donde encontramos una descripción detallada del evento, la imagen e información que mostraban en la pantalla anterior y un mapa para poder ver con exactitud la localización del evento clickado. En la imagen 11.4 se muestra la pantalla mencionada.

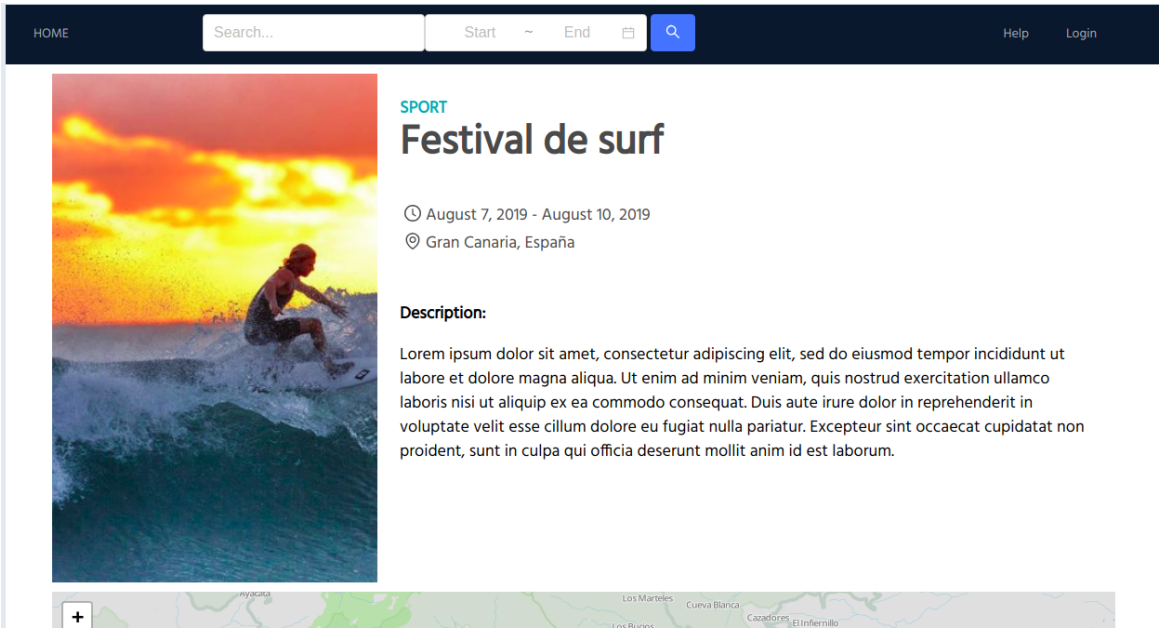

Figura 11.4: Página descripción evento

11.3. Perfil de usuario

Cuando nos logueamos en la página podremos acceder a nuestro perfil de usuario. En este se encuentra una breve información del usuario y los eventos creados por el, para poder editarlos o eliminarlos. Una vez logueados podremos crear eventos, pues se nos mostrará en la barra superior un botón de crear evento. En la siguiente imagen 11.5 se muestra la página del perfil de usuario:

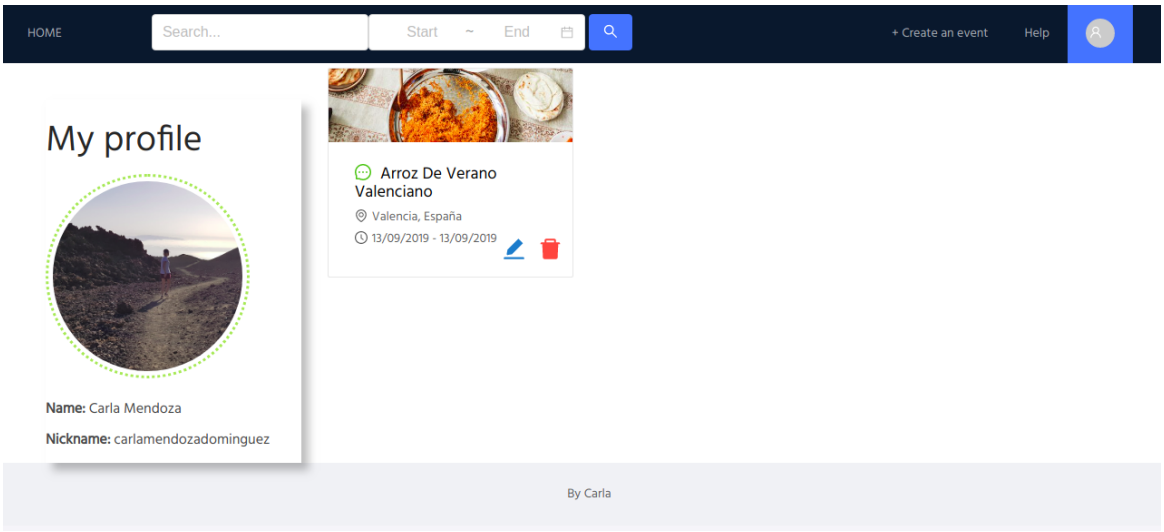

Figura 11.5: Página de perfil de usuario

11.4. Creación de eventos

Tanto para crear, como para editar eventos dispondremos de un formulario como el de la imagen 11.6.

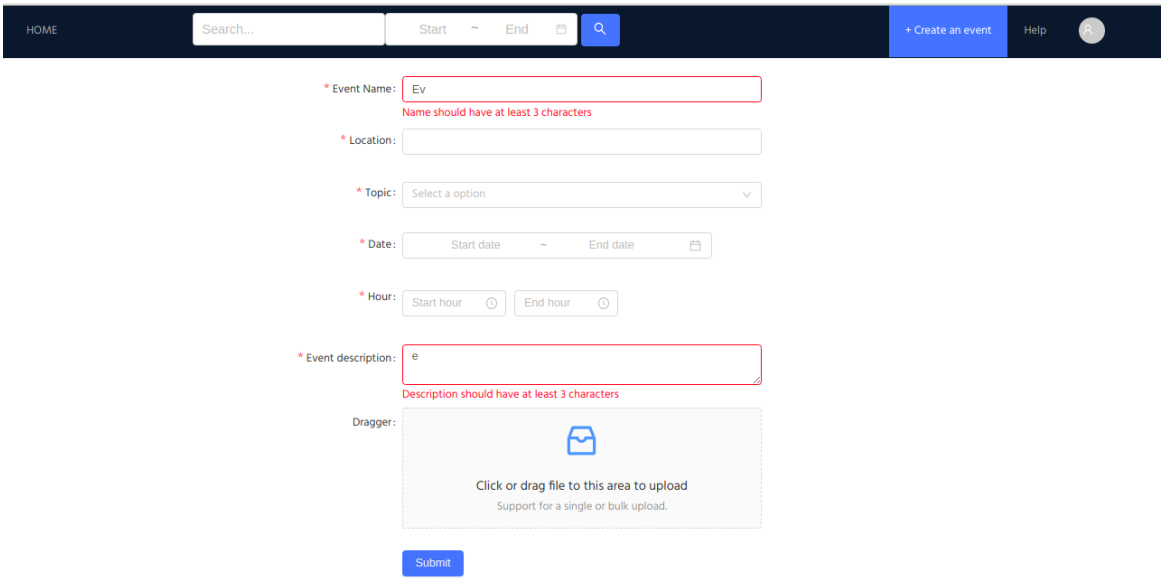

Figura 11.6: Formulario creación de evento

Capítulo 12

Conclusiones y líneas futuras

Este proyecto a sido todo un reto personal dado a mi poca experiencia en la programación web. Tras la realización del mismo, he obtenido un aprendizaje y comprensión más profunda del desarrollo FullStack, desde la configuración de los entornos necesarios como CI/CD, Docker, servidores, entre otros, hasta la implementación de la aplicación.

También se han adquirido conocimentos en programación asíncrona, debido el comportamiento de NodeJs y las peticiones HTTP.

Cabe destacar que a pesar de ser una aplicación web, se ha adaptado para poder ser usada en móviles dado a que hoy en día el manejo del teléfono móvil es mucho más frecuente que el de un ordenador.

El futuro de la aplicación tiene los siguientes frentes claros:

- Mejorar el comportamiento responsive de la aplicación.
- Resolver los conflictos entre la adición de mismos eventos.
- Mejorar la cobertura de los test.
- Realizar un diseño más personal.

Capítulo 13

Conclusion and summary

This project has been quite a personal challenge given my little experience in web programming. After the realization of the same one, I have obtained a learning and deeper understanding of the FullStack development, from the configuration of the necessary environments as CI/CD, Docker, servers, among others, until the implementation of the application.

Also have acquired knowledge in asynchronous programming, due to the behavior of NodeJs and HTTP requests.

It should be noted that despite being a web application, it has been adapted to be used on mobile phones because today the handling of the mobile phone is much more frequent than that of a computer.

The future of the application has the following clear fronts:

- Improve the responsive behavior of the application.
- Resolve conflicts between the addition of the same events.
- Improve coverage tests.
- Make a more personal design.

Apéndice A

Código Gitlab

Dada la extensión del código se indica el link donde poder visualizarlo:

- Gitlab

Bibliografía

[1] WCAG. http://www.sidar.org/traducciones/wcag20/es/

[2] Amazon Web Service. https://aws.amazon.com/es/

[3] MongoDB. https://docs.mongodb.com/

[4] Mongoose. https://mongoosejs.com/docs/guide.html

[5] Node.js. https://nodejs.org/api/

[6] Express.js. http://expressjs.com/es/

[7] React.js. https://reactjs.org/

[8] Redux. https://es.redux.js.org/

[9] Git. https://git-scm.com/

[10] Gitlab. https://gitlab.com/gitlab-org

[11] Ant design. https://ant.design/

[12] React-Router. https://reacttraining.com/react-router

[13] OpenStreetMap. https://www.openstreetmap.org/#map=5/41.541/33.860

[14] Leaflet. https://leafletjs.com/reference-1.5.0.html

[15] Create-React-App. https://github.com/facebook/create-react-app

[16] Auth0. https://auth0.com/docs

[17] Cloudinary. https://cloudinary.com/documentation

[18] Docker. https://www.docker.com/get-started

[19] Nodemon. https://nodemon.io/

[20] Jest. https://facebook.github.io/jest/

[21] Cypress. https://docs.cypress.io/guides/overview/why-cypress.html

[22] SonarQube. https://docs.sonarqube.org/latest/

[23] Husky. https://github.com/typicode/husky

[24] ESlint. https://eslint.org/docs/user-guide/getting-started

[25] NPM. https://docs.npmjs.com/