Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Chefier: Aplicación web de recetas de cocina

Chefier: Recipe web application

Marc Carbonell González de Chaves

La Laguna, 24 de mayo de 2023

D. **Vicente José Blanco Pérez**, con N.I.F. 42.171.808-C profesor Titular de Universidad adscrito al Departamento de Estadística, Investigación Operativa y Computación de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Chefier: Aplicación web de recetas de cocina"

ha sido realizada bajo su dirección por D. **Marc Carbonell González de Chaves**, con N.I.F. 79.151.774-B.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 24 de mayo de 2023

Agradecimientos

Agradecimientos especiales a mis padres David y Marta y a mi hermano Daniel que siempre han estado ahí apoyándome, tanto en los buenos como en los malos momentos.

Licencia

Resumen

En la actualidad existen numerosas vías para compartir recetas de cocina en internet, ya sea a través de aplicaciones dedicadas específicamente para ello, que funcionan de forma similar a un blog, o mediante redes sociales, que priman la interacción entre usuarios. La idea detrás de Chefier es unir estos dos conceptos para obtener las ventajas de ambos.

El propósito de este proyecto es desarrollar una aplicación web orientada al ámbito culinario que, a modo de red social, permita al usuario compartir recetas de cocina propias, así como consultar y dar valoraciones o reseñas a las compartidas por la comunidad.

Esta aplicación contará con un modelo de base de datos centrado en cuentas de usuarios y recetas. Entre las funcionalidades principales se encuentran registro, visualización y publicación de recetas, búsqueda y filtrado de resultados, e interacción con el contenido.

Para el desarrollo de este proyecto web, se ha prestado especial atención al seguimiento de buenas prácticas de gestión de proyectos, testing de backend y frontend, e integración y despliegue continuo en servidores en la nube.

El objetivo principal del proyecto es aplicar los conocimientos sobre desarrollo software, gestión de proyectos y buenas prácticas adquiridos durante la carrera, así como conocer y familiarizarse con algunas de las tecnologías más importantes y más utilizadas en la actualidad en el ámbito del desarrollo web.

Palabras clave: Comida, Receta, Cocina, Aplicación, Web, Full Stack, TypeScript, GitHub, Next.js, React, Node, Koa, MongoDB Atlas, Vercel.

Abstract

Nowadays, there are numerous ways to share recipes online, either through dedicated applications, which work similarly to a blog, or through social networks, which prioritise user interaction. The idea behind Chefier is to combine these two concepts to get both advantages.

This project aims to develop a culinary-oriented web application that, as a social network, allows users to share their recipes and to consult and give ratings or reviews to those shared by the community.

This application will have a database model centred on user accounts and recipes. The main functionalities include registration, visualisation and publication of recipes, search and filtering of results, and interaction with the content.

For the development of this web project, special attention has been paid to following good project management practices, backend and frontend testing, and continuous integration and deployment on cloud servers.

The project's main objective is to apply the knowledge of software development, project management and best practices acquired during the degree, as well as to get to know and become familiar with some of the most important and widely used technologies in web development.

Keywords: Food, Recipe, Kitchen, Application, Web, Full Stack, TypeScript, GitHub, Next.js, React, Node, Koa, MongoDB Atlas, Vercel.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Este capítulo sirve de introducción para el proyecto, incluyendo los antecedentes y estado actual del tema, así como las principales tecnologías utilizadas y la planificación de tareas seguida durante el desarrollo.

1.1. Antecedentes y estado actual del tema

El tipo de aplicación que se pretende desarrollar cuenta con numerosos antecedentes diseñados e implementados para distintas plataformas y sistemas, lo cuál deja claro que se trata de una idea útil que suscita interés en el ámbito culinario.

La mayoría de herramientas diseñadas con este propósito proveen un servicio similar, permitiendo compartir y valorar recetas, aunque también existen algunas que aportan funcionalidades extra que enriquecen la aplicación, como sistemas de búsqueda o de recomendación. Por lo general, este tipo de aplicaciones son completamente gratuitas y solo requieren un registro para poder ser utilizadas.

Entre las aplicaciones de recetas más populares se encuentra MyRealFood [1], la cual se enfoca principalmente en la alimentación saludable, y que posee además, un sistema de escaneo de alimentos y de registro de comidas. Esta está dirigida especialmente a usuarios de móvil, pero también cuenta con interfaces para ordenador y tableta.

También es interesante el enfoque de la idea aplicado por Cookpad [2], centrándose más específicamente en recetas caseras, y teniendo como objetivo convertir la cocina diaria en un proceso ameno y divertido, compartiendo técnicas y trucos culinarios poco conocidos, además de las propias recetas. Esta aplicación posee interfaces para web, móvil y tableta.

Así mismo, se pueden encontrar otros muchos ejemplos de aplicaciones web para la compartición de recetas que únicamente ofrecen los servicios básicos pero que sirven de apoyo para la retroalimentación de la comunidad, como Funcook [3] o PetitChef [4].

La idea con este proyecto es desarrollar una aplicación de cocina desde un enfoque diferente al habitual, que suele asemejarse más a un blog, y en su lugar crear algo más parecido a una red social, donde los usuarios puedan publicar recetas en su canal o perfil para compartirlas con otros amantes de la cocina, seguir a otros creadores, interactuar a través de me gustas o comentarios, guardar sus publicaciones favoritas y explorar todo el contenido generado por la comunidad.

Este enfoque ya es adoptado por muchas cuentas de otras redes sociales como Instagram o Twitter, sin embargo, la ventaja de esta aplicación radicaría en poder dirigir el contenido a un público mucho más específico, en este caso, interesado en la cocina.

1.2. Tecnologías utilizadas

Para el desarrollo de la aplicación, al igual que en casi cualquier otro proyecto de desarrollo de software, se ha utilizado Git [5] como sistema de control de versiones y GitHub [6] como plataforma donde almacenar el código fuente de forma remota. Así mismo, en cuanto al lenguaje de programación, se ha optado por utilizar Typescript [7] en lugar de Javascript en la mayor parte del código debido a las facilidades que ofrece el tipado estático.

En cuanto a las tecnologías de desarrollo web utilizadas, Chefier [8] es una aplicación web basada en el stack MERN (MongoDB [9], Express [10], React [11], Node [12]), sin embargo algunas de estas tecnologías han sido sustituidas por alternativas similares que se adaptan mejor al proyecto.

En cuanto al frontend, se ha decidido utilizar el framework de React, Next.js [13] frente a otras soluciones debido a que esta ofrece ciertas ventajas como el pre-renderizado de páginas. Además, se trata de uno de los frameworks más populares del momento, y está considerado por muchos expertos como el futuro del desarrollo frontend.

Por otro lado, las principales tecnologías utilizadas en el backend son Node.js como entorno de ejecución de Javascript, y el framework web diseñado por el equipo detrás de Express, Koa.js [14], como una alternativa más enfocada al desarrollo backend dividido por capas o niveles.

Por último, el sistema de base de datos elegido es MongoDB debido a las facilidades que ofrecen las bases de datos no relacionales y la posibilidad de desplegar dicha base de datos en la nube de forma gratuita gracias a MongoDB Atlas [15].

Todas las herramientas nombradas anteriormente será analizadas más en profundidad en el siguiente capítulo de la memoria.

1.3. Planificación del Desarrollo

El desarrollo de la aplicación ha sido planificado basándose en metodologías ágiles de desarrollo software. De esta manera, se ha llevado a cabo un desarrollo iterativo e incremental dividido en bloques temporales con el que se pretende dar un resultado más completo del producto final.

Siguiendo esta metodología, el desarrollo estará dividido en tareas de preparación o inicialización del proyecto y tareas que se llevarán a cabo en cada iteración. Para llevar un registro de estas tareas se ha utilizado la herramienta de proyectos proporcionada por GitHub, GitHub Projects [16].

Preparación	Iteraciones
Prototipado	Desarrollo Frontend
Setup Infraestructura	Desarrollo Backend
Setup Frontend	Testing
Setup Backend	Despliegue

Tabla 1.1: Planificación de tareas

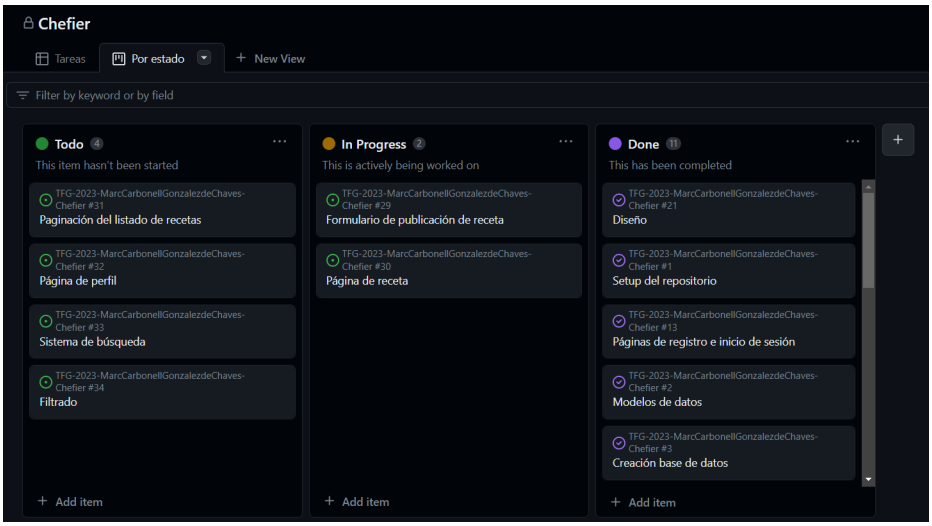

Figura 1.1: Registro de tareas del proyecto en GitHub Projects

Capítulo 2

Tecnologías y Herramientas

Como en cualquier otro proyecto de desarrollo web, para construir la aplicación será necesario emplear una serie de tecnologías y recursos. Estas herramientas se utilizan con el propósito de obtener funcionalidades específicas, simplificar el código y agilizar el proceso de desarrollo. En el siguiente apartado, se proporcionará una breve descripción y una justificación para el uso de las principales librerías de terceros, frameworks y otros elementos similares utilizados en este proyecto.

2.1. General

- **Git** El sistema de control de versiones Git, como su nombre indica, permite llevar a cabo un control de versiones del código desarrollado a través de repositorios. De esta manera es posible realizar un seguimiento preciso de los cambios realizados en el código a lo largo del tiempo, mejorando la eficiencia, y la organización del mismo.

- **GitHub** GitHub es una plataforma que permite alojar proyectos utilizando el sistema de control de versiones Git. Ha sido utilizada para alojar el código del proyecto en un repositorio, así como para llevar a cabo la integración/despliegue continuo mediante GitHub Actions, y la gestión del proyecto mediante Github Projects.

  En cuánto al flujo de trabajo, se han utilizado dos ramas principales, la rama *main* o rama de producción y la rama *develop* o rama de desarrollo. Las nuevas funcionalidades a implementar han sido desarrolladas sobre las ramas *feature* creadas a partir de la rama de desarrollo y fusionadas de nuevo con esta. Al alcanzarse cierto número de funcionalidades consideradas como una *release* la rama de desarrollo es fusionada con la rama de producción.

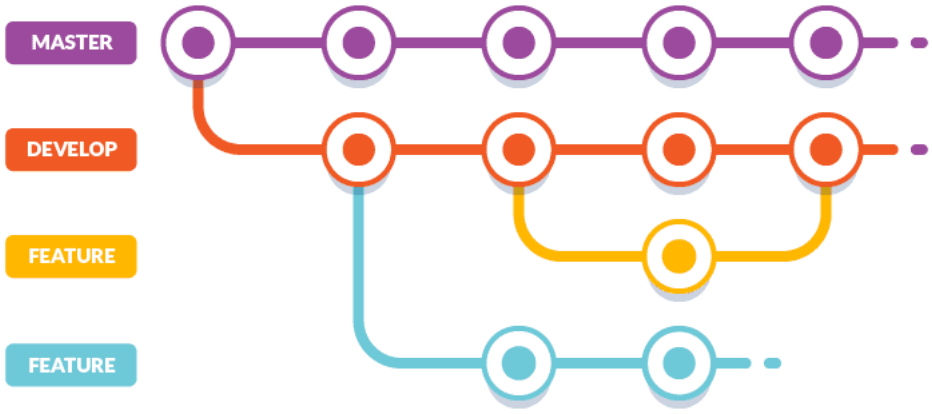

Figura 2.1: Flujo de trabajo aplicado en el proyecto

- **Typescript** Typescript ha sido el lenguaje de programación utilizado para el desarrollo de la mayor parte del proyecto debido a que su uso mejora la calidad, la productividad y la mantenibilidad del código

  Al ser un superset de JavaScript, Typescript permite aprovechar todas las funcionalidades del lenguaje base mientras agrega características adicionales, como el sistema de tipos estáticos. Esto ayuda a detectar errores y problemas potenciales durante la fase de desarrollo, lo que hace que el código sea más robusto y menos propenso a errores.

- **Vercel** [17] Vercel es la plataforma web creadora de Next.js, esta permite realizar el despliegue en la nube de aplicaciones web de forma gratuita. Ha sido utilizada para desplegar tanto el frontend como el backend de la aplicación.

  Entre las ventajas de esta plataforma se encuentran el despliegue automático, que se realiza con cada *push* a cualquier rama del repositorio, y el hecho de que cada rama pueda desplegarse de forma independiente, lo cuál es muy útil para diferenciar entre el despliegue de desarrollo y el de producción.

2.2. Frontend

- **Next.js** Next.js es un marco web de desarrollo frontend de React de código abierto que habilita, entre otras muchas funcionalidades, el pre-renderizado de páginas, la representación del lado del servidor (Server Side Rendering) y la generación de sitios web estáticos (Static Generation) para aplicaciones web basadas en React.

  Se trata de uno de los framework de desarrollo de frontend más importantes del momento, considerado por muchos como el futuro líder del desarrollo web. Los mismos desarrolladores de React mencionan a Next.js entre sus herramientas recomendadas, describiéndolo como una solución versátil que permite crear aplicaciones de cualquier tamaño, desde un blog estático hasta una aplicación dinámica compleja.

- **SWR** [18] SWR es una librería desarrollada por Vercel para gestionar el *fetching* (obtención) de datos de una aplicación. Esta sigue una estrategia de *stale-while-revalidate* [19], que consiste en devolver primero los datos de la caché (obsoletos), enviar después la petición de obtención (revalidación) y, por último, retornar los datos actualizados.

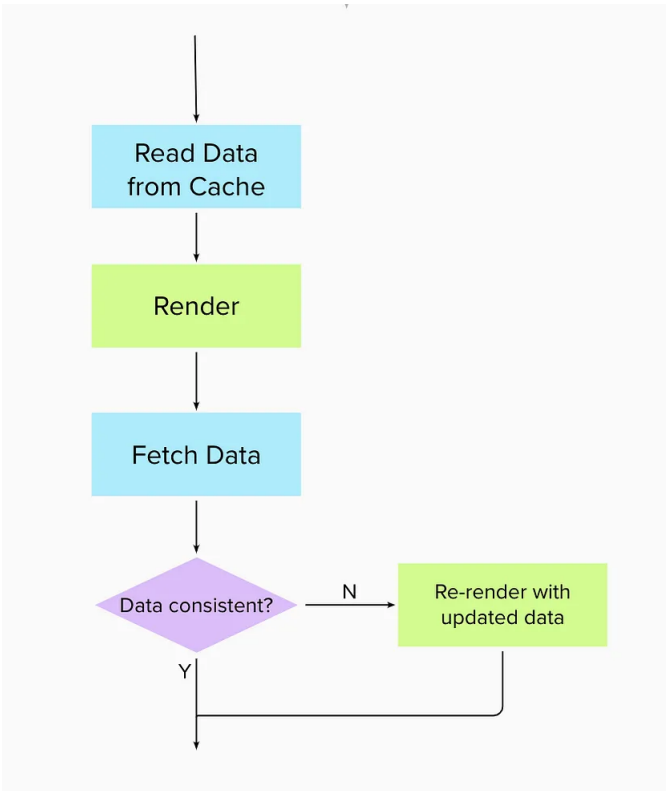

Figura 2.2: Estrategia *stale-while-revalidate*

Esta librería proporciona una forma sencilla y eficiente de administrar el estado y los datos de una aplicación, permitiendo realizar solicitudes HTTP, cachear los resultados y mantener el estado de manera automática, lo que reduce la cantidad de código necesario para gestionar el flujo de datos. Esto mejora la experiencia del usuario al mostrar datos rápidamente y mantenerlos actualizados en tiempo real.

Debido a todo esto y a su simplicidad, eficiencia y capacidad para mantener los datos actualizados de manera automática, se ha optado por SWR como alternativa a otras librerías de manejo de estado en React.

- **NextAuth.js** [20] Esta librería ofrece una solución completa para el control de sesión de aplicaciones desarrolladas con Next.js. Se ha utilizado para implementar la autenticación mediante credenciales y a través Google y GitHub utilizando el estándar Oauth2. En el siguiente capítulo de la memoria se profundizará sobre la implementación de dicho sistema de autenticación.

2.3. Backend

- **Node.js** Node.js es el mejor entorno de ejecución para JavaScript y el más utilizado por la comunidad. Su principal ventaja es que permite gestionar múltiples conexiones al mismo tiempo, evitando el bloqueo de procesos.

- **Koa** Koa es un nuevo framework web diseñado por el equipo detrás de Express, que pretende ser una versión más pequeña, expresiva y robusta para aplicaciones web y APIs, proporcionando un elegante conjunto de métodos que hacen que escribir servidores sea rápido y agradable.

  Además, Koa fomenta la implementación de un estilo de arquitectura de software por capas, que divide la aplicación en 3 tres diferentes niveles o regiones, mejorando la abstracción de sus diferentes partes. En el siguiente capítulo de la memoria se profundizará sobre este tipo de arquitectura.

2.4. Base de datos

- **MongoDB** MongoDB es un sistema de base de datos no relacional, orientado a documentos y de código abierto. Este guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

- **MongoDB Atlas** Esta plataforma permite desplegar bases de datos de MongoDB en la nube de forma gratuita, estas pueden ser gestionadas fácilmente a través de la herramienta de interfaz de usuario que proporciona la compañía, llamada MongoDB Compass.

- **Imagekit** [21] Imagekit es una plataforma web que permite almacenar archivos multimedia. Esta ofrece la posibilidad de agregar puntos de entrada a la API para poder gestionar la base de datos. Ha sido utilizada para almacenar las imágenes de la aplicación, es decir, los avatares de los usuarios y las fotos incluidas en las publicaciones/recetas.

2.5. Testing

- **Vitest** [22] Vitest es un framework de tests unitarios basado en Jest que, además de ser bastante rápido, ofrece entre otras funcionalidades una interfaz de usuario que permite realizar tests de forma rápida y sencilla. Además permite calcular el cubrimiento de los tests del proyecto de forma global, es decir, de tanto el frontend como del backend de una sola vez, facilitando la integración continua.

- **Cypress** [23] Cypress es una herramienta de testing de frontend para aplicaciones web que permite realizar pruebas *end-to-end* o extremo a extremo para simular la experiencia del usuario final, sin necesidad de configuraciones y de manera rápida, completa y segura.

Capítulo 3

Desarrollo de la aplicación

En este capítulo se abordarán los aspectos principales del desarrollo de la aplicación, desde su diseño y prototipado hasta su despliegue. Se describirá la estructura del código fuente y la arquitectura de las diferentes partes de la aplicación, así como aspectos generales del desarrollo como el proceso de autenticación o la integración continua de testing, cubrimiento y análisis de calidad de código.

3.1. Diseño y prototipado

El primer paso del desarrollo del proyecto es realizar el diseño y prototipado de la aplicación con el objetivo de obtener una aproximación inicial de como será la interfaz de usuario.

3.1.1. Mockups

Previo al comienzo del desarrollo de la aplicación, se crearon una serie de *mockups* (maquetas de diseño) con el objetivo de definir la distribución de los distintos elementos de la interfaz de usuario, dejando por el momento de lado el apartado estético, es decir, logotipo, paleta de colores, etc. Estos diseños son una aproximación inicial, por lo tanto, han podido estar sujetos a cambios durante el desarrollo del proyecto.

La idea inicial es que todas las páginas cuenten con una barra de navegación y un camino de hormigas en la parte superior con enlaces que permitirán navegar por las diferentes páginas del sitio.

- **Inicio**

  Página principal de la aplicación en la que se mostrarán recetas y que contará con un sistema de búsqueda con filtros aplicables.

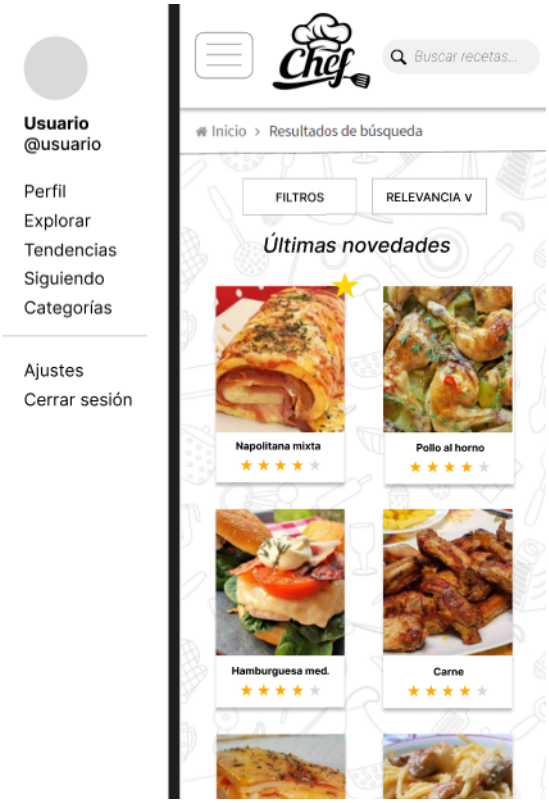

Figura 3.1: *Mockup* de la página de inicio en móvil

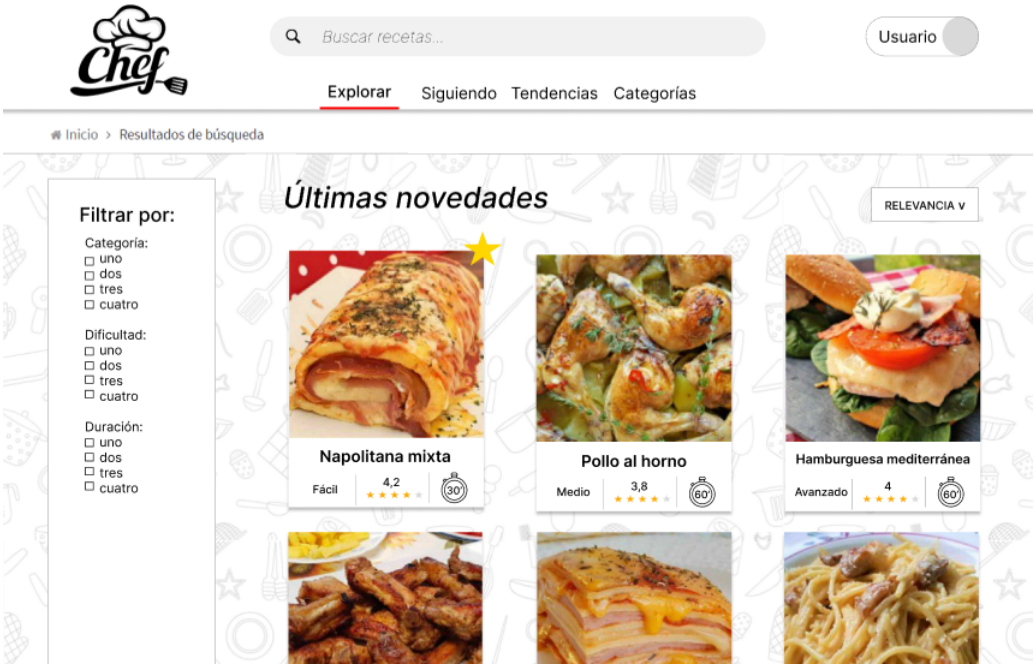

Figura 3.2: *Mockup* de la página de inicio en ordenador

- **Inicio de sesión/Registro**

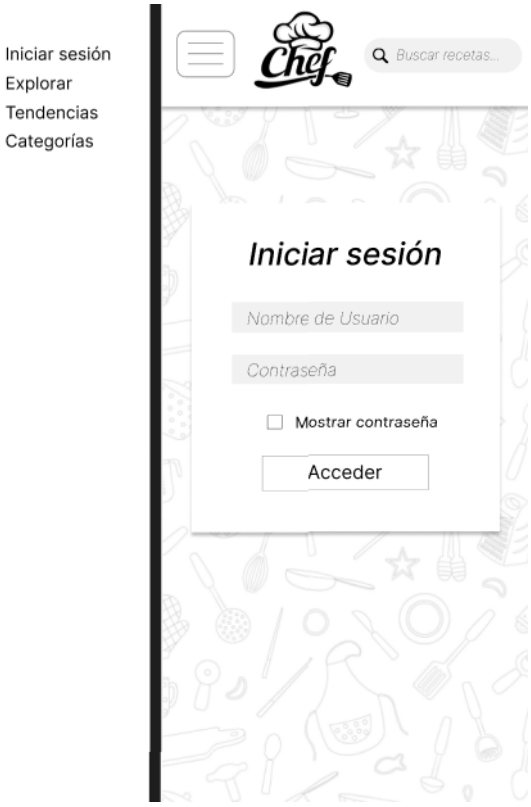

Figura 3.3: *Mockup* de la página de inicio de sesión en móvil

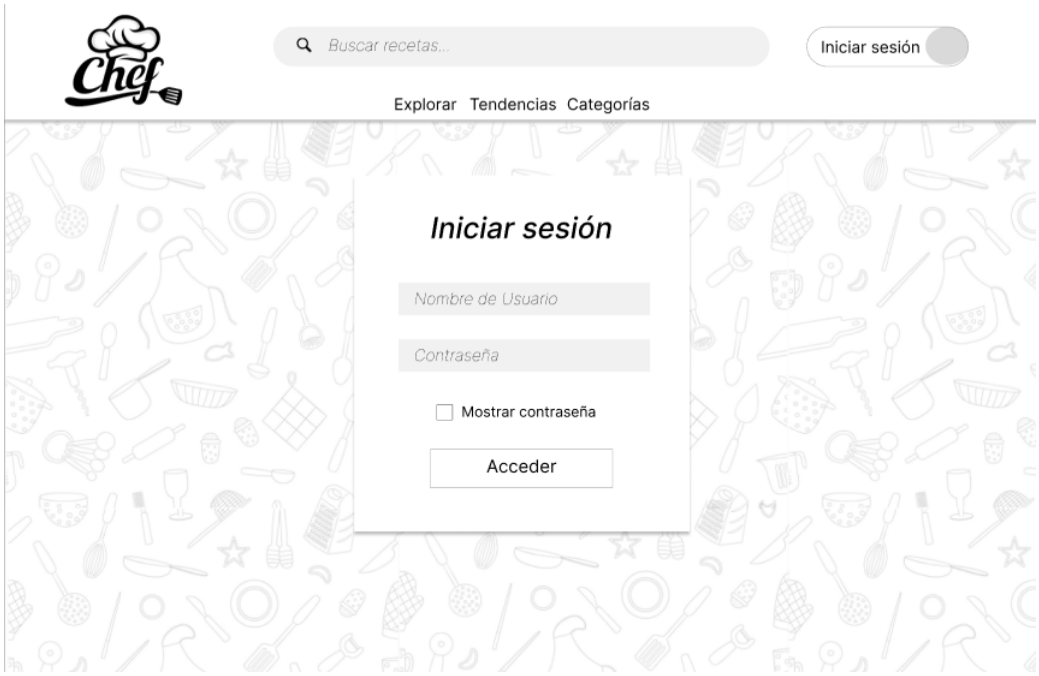

Figura 3.4: *Mockup* de la página de inicio de sesión en ordenador

- **Perfil de usuario**

  En esta página el usuario podrá configurar los datos de su perfil, así como ver sus recetas, publicaciones a las que ha dado me gusta y las que ha guardado.

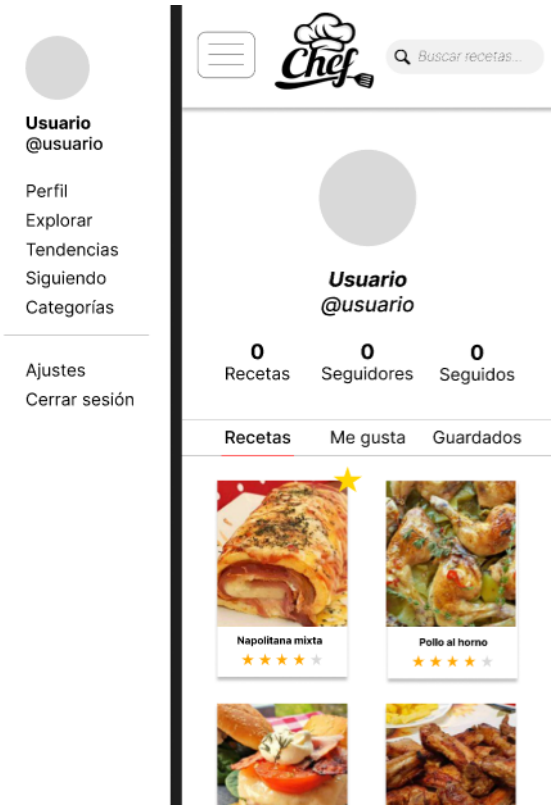

Figura 3.5: *Mockup* de la página de perfil de usuario en móvil

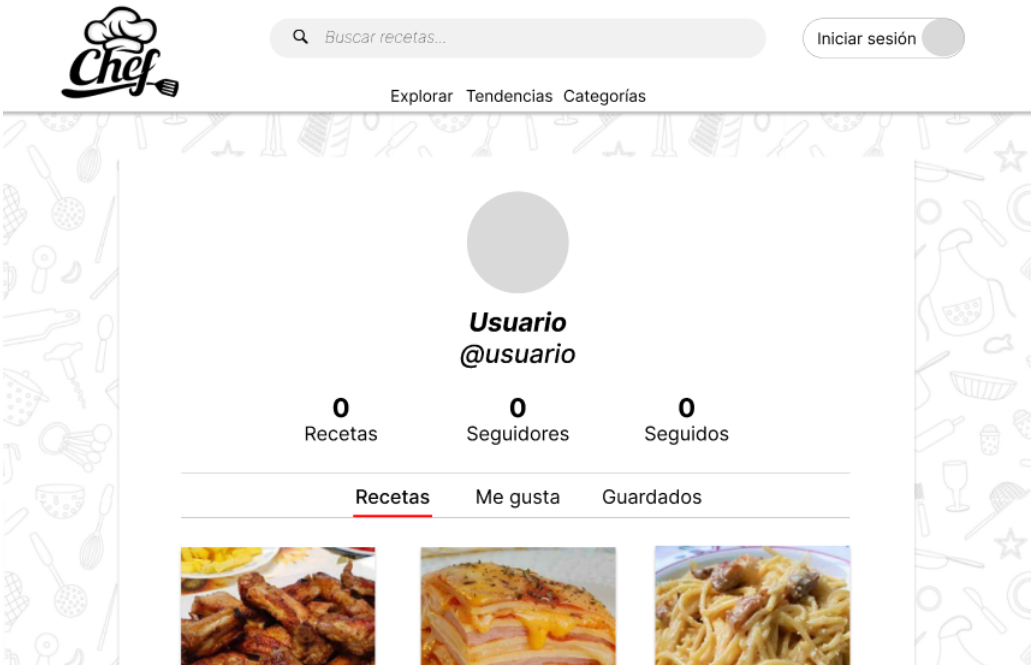

Figura 3.6: *Mockup* de la página de perfil de usuario en ordenador

- **Publicación/Receta**

  Página que mostrará una publicación junto con datos e información relevante sobre esta, y que permitirá al usuario realizar una serie de interacciones, como calificar, dar me gusta o guardar la publicación.

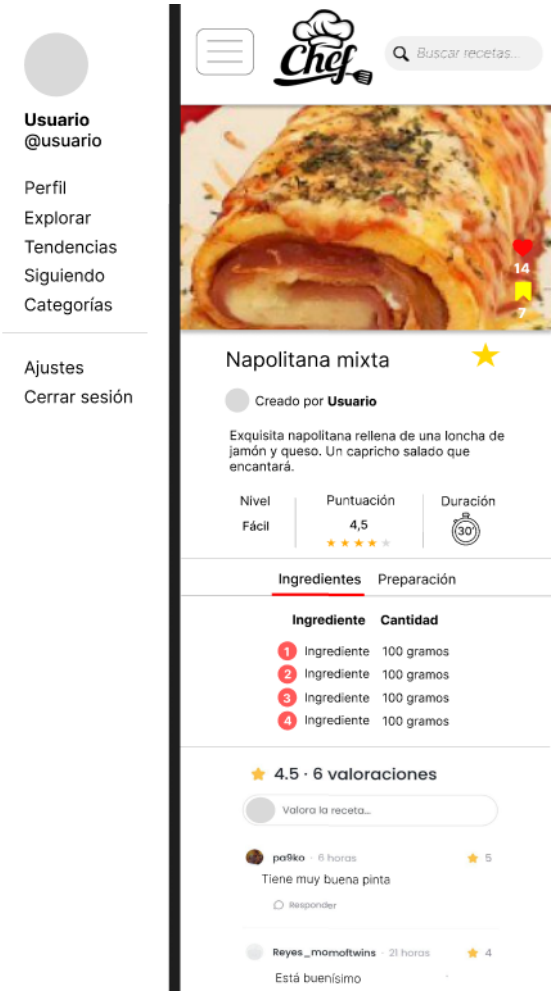

Figura 3.7: *Mockup* de la página de publicación en móvil

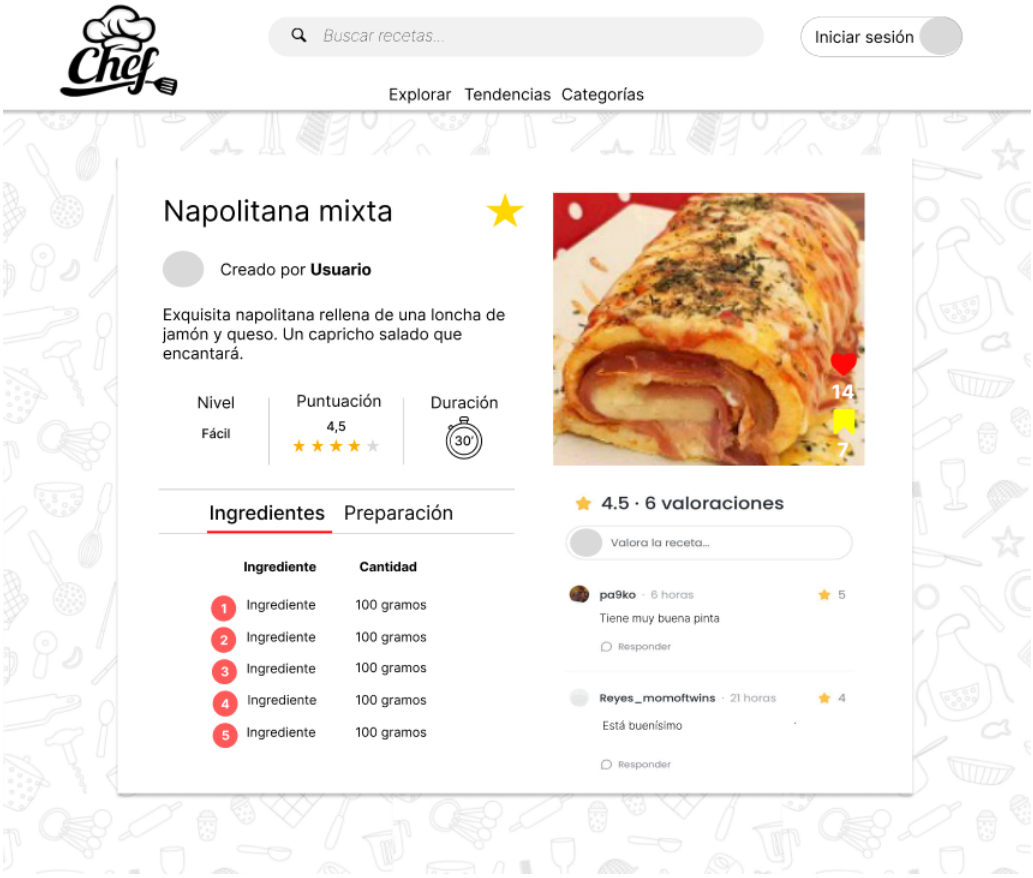

Figura 3.8: *Mockup* de la página de publicación en ordenador

3.1.2. Logotipo y paleta de colores

Para este proyecto elegí el color rojo pastel como color principal y los colores blanco y negro como colores secundarios dependiendo del tema, claro u oscuro. Buscando una idea de un logo original decidí optar por un animal como figura principal, y, habiendo escogido el rojo como color principal, terminé decidiéndome por representar un panda rojo como elemento principal del logotipo. Dicho logotipo fue diseñado utilizando Adobe Illustrator [24] y Photoshop [25].

Figura 3.9: Paleta de colores de la aplicación

Figura 3.10: Logotipo de la aplicación

3.2. Configuración del repositorio

El repositorio del proyecto [26], alojado en la plataforma GitHub, ha sido configurado a modo de mono-repo, de tal forma que dicho repositorio contiene tanto el código fuente correspondiente al frontend, en el directorio raíz, como el correspondiente al backend, en del directorio "server". La estructura de directorios es la siguiente:

```
/ (Frontend)
├── .github
│   └── workflows (Configuración Github Actions)
├── public (Directorio público)
├── __tests__ (Tests del frontend)
├── src (Código fuente del frontend)
└── server (Backend)
    ├── __tests__ (Tests del backend)
    └── src (Código fuente del backend)
```

3.3. Autenticación

Para el sistema de autenticación se ha utilizado la librería NextAuth [20] en el frontend, la cuál facilita el uso de OAuth 2.0, un estándar abierto y simple para la autenticación segura de APIs. Esta posibilita el acceso a a la aplicación a través de proveedores de otros servicios como Google, Facebook, GitHub, etc., así como mediante credenciales. Y , por otro lado, en el backend se han utilizado las librerías jasonwebtoken [27] y bcrypt [28] para la generación de tokens de acceso y la encriptación de contraseñas respectivamente.

En el forntend, NextAuth se encarga de gestionar la sesión, generando de forma automática el token de acceso (por defecto JSON Web Token) y proporcionando funcionalidades para comprobar su estado, pudiendo así restringir el acceso a rutas, páginas o contenido a usuarios no identificados, entre otras muchas posibilidades.

En este caso, se ha implementado el inicio de sesión mediante credenciales, gestionadas de forma interna por el backend, y mediante los servicios de Google y GitHub.

3.3.1. Credenciales

La librería NextAuth proporciona un proveedor para gestionar el inicio de sesión de forma interna mediante credenciales. Este sistema se basa en la comunicación del cliente con el servidor backend, que es el encargado de verificar la identidad del usuario para permitir su acceso a la aplicación.

De esta forma, para la autenticación, en primer lugar el usuario deberá registrarse en la aplicación, proporcionando un nombre de usuario, correo y contraseña, que serán validados y almacenados en la base de datos. Una vez registrado, el usuario podrá iniciar sesión mediante dichas credenciales, las cuáles serán verificadas por el backend. En el proceso de inicio de sesión, el backend generará un token de acceso que será integrado dentro del token creado de forma automática por NextAuth. La sesión será gestionada mediante dicho token, que será necesario para acceder a ciertos "endpoints" o puntos de entrada de la API.

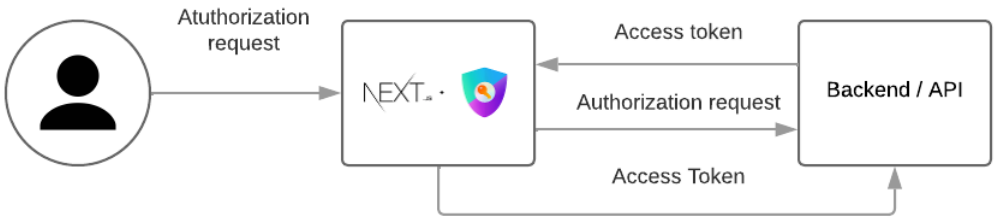

Figura 3.11: Proceso de autenticación con credenciales

3.3.2. Google y GitHub

El proceso de autenticación mediante Google y GitHub es un tanto diferente. En este caso, serán dichos proveedores los que generen el token de acceso que nuevamente será integrado dentro del token generado por NextAuth. Para la verificación de dichos token en el backend se han utilizado los procesos recomendados por cada proveedor, en el caso de GitHub mediante una llamada a su API REST, y en el caso de Google utilizando una librería oficial desarrollada por la compañía.

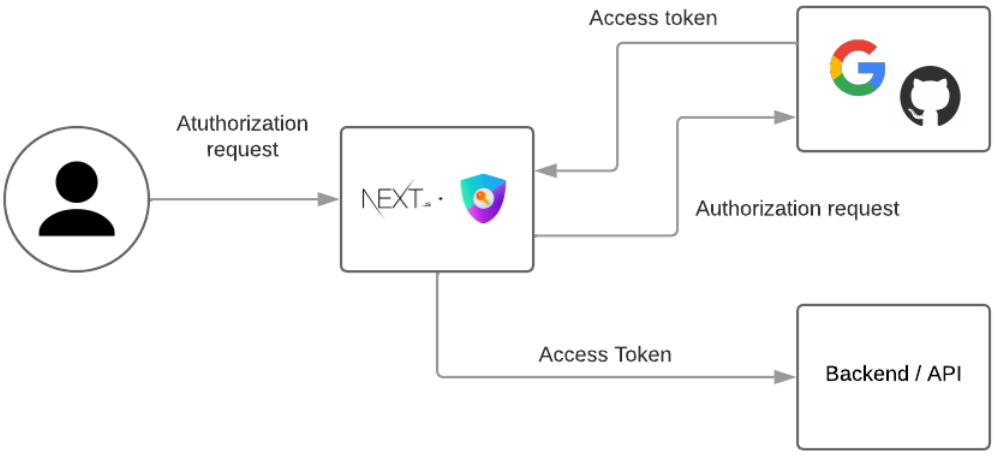

Figura 3.12: Proceso de autenticación con Google y GitHub

3.4. Integración continua

Los procesos de integración continua han sido implementados con GitHub Actions, de manera que las pruebas de testing, cubrimiento, y calidad de código son ejecutadas con cada *push* y *pull-request* sobre las ramas de desarrollo (*develop*) y producción (*main*).

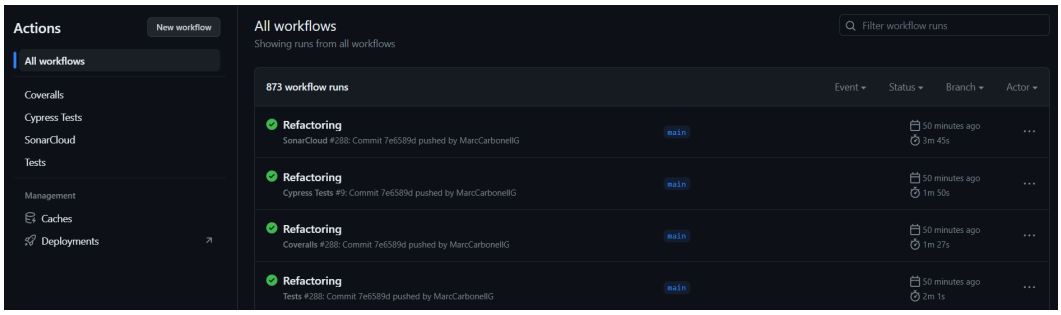

Figura 3.13: Resultados de la ejecución de las GitHub Actions

3.4.1. Testing

Para el testing de la aplicación se ha diseñado una batería de tests con el fin de asegurar el correcto funcionamiento de la mayor parte del código desarrollado. Para ambas partes de la aplicación, frontend y backend, se ha utilizado el framework de testing Vitest, nombrado anteriormente. Asi mismo, ha sido necesario utilizar la librería supertest para testear las llamadas a la API en el backend.

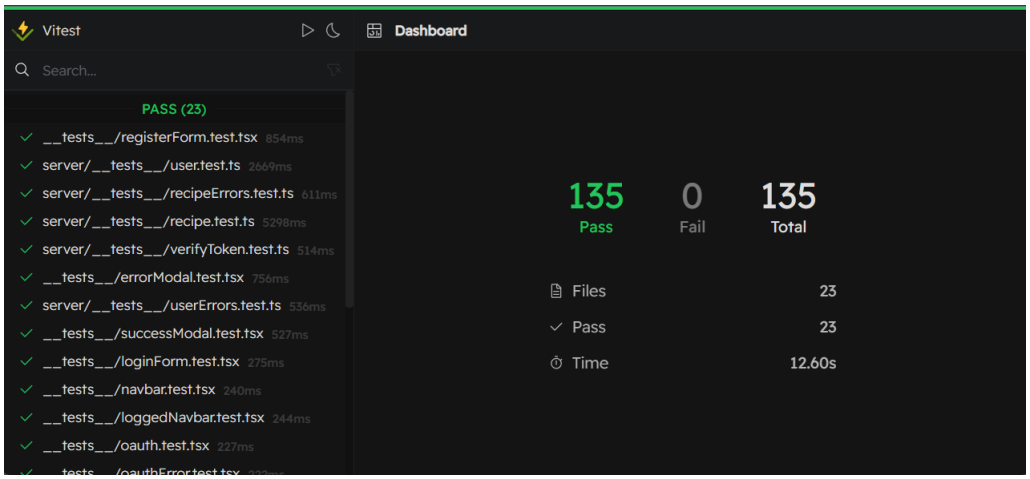

Figura 3.14: Ejecución de los tests en la interfaz de usuario de Vitest

Además, también se han implementado una serie de tests extremo a extremo o *end-to-end* con el framework Cypress para comprobar el correcto funcionamiento de algunas funcionalidades de la interfaz de usuario.

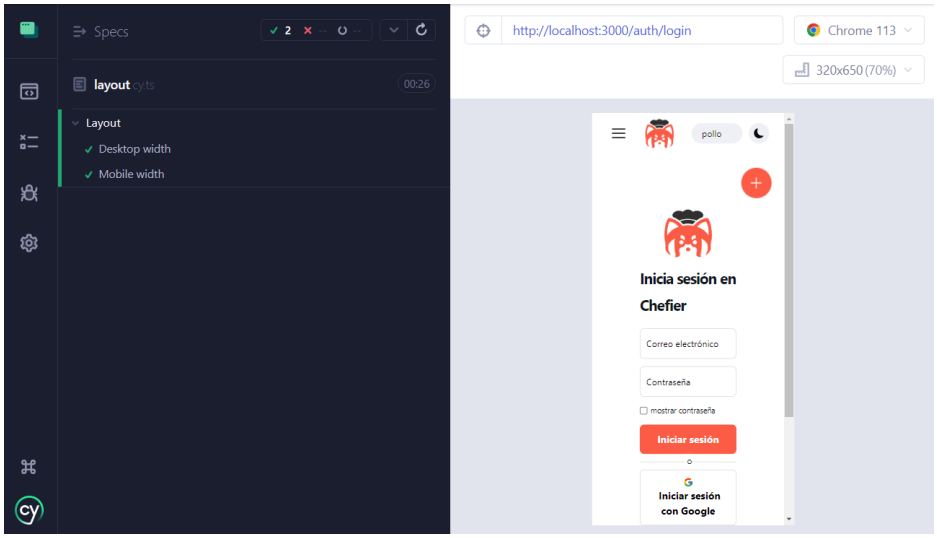

Figura 3.15: Ejecución de los tests *end-to-end* en la interfaz de usuario de Cypress

3.4.2. Cubrimiento

Los tests son implementados con la idea de cubrir la mayor parte posible del código, para asegurar esto, se ha utilizado la plataforma Coveralls [29], que proporciona un análisis del cubrimiento de los tests sobre el mismo.

Figura 3.16: Resultados de cubrimiento en Coveralls

3.4.3. Calidad de código

SonarCloud [30] ha sido la herramienta utilizada para asegurar que el código cumple unos estándares mínimos de calidad. Esta plataforma realiza un exhaustivo análisis de la calidad del código, permitiendo detectar errores, duplicidades, etc.

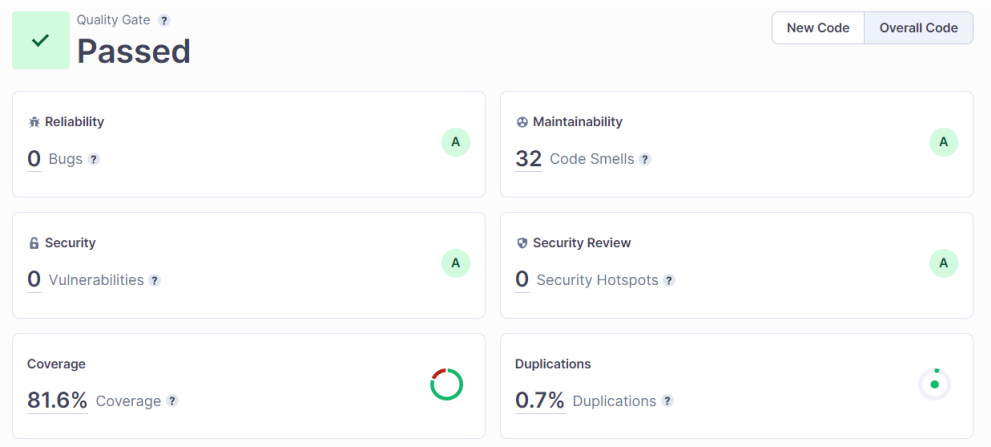

Figura 3.17: Resultados del análisis de calidad de código en SonarCloud

3.5. Frontend

Como se indicó en el capítulo anterior, se ha utilizado Next.js, para el desarrollo frontend. Este framework se encarga de gestionar las herramientas y la configuración necesarias para React, y proporciona estructura, características y optimizaciones adicionales para la aplicación.

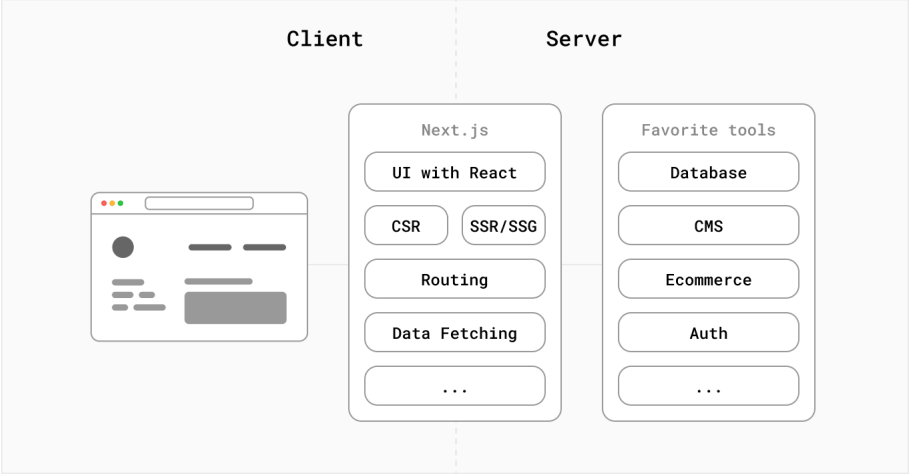

Figura 3.18: Arquitectura de una aplicación desarrollada con Next.js

Una de las ventajas que tiene Next.js, es que optimiza e integra automáticamente los archivos JavaScript y CSS para la producción. Mediante este proceso se eliminan el formato y los comentarios innecesarios del código sin cambiar su funcionalidad, lo que mejora el rendimiento de la aplicación al reducir el tamaño de los archivos.

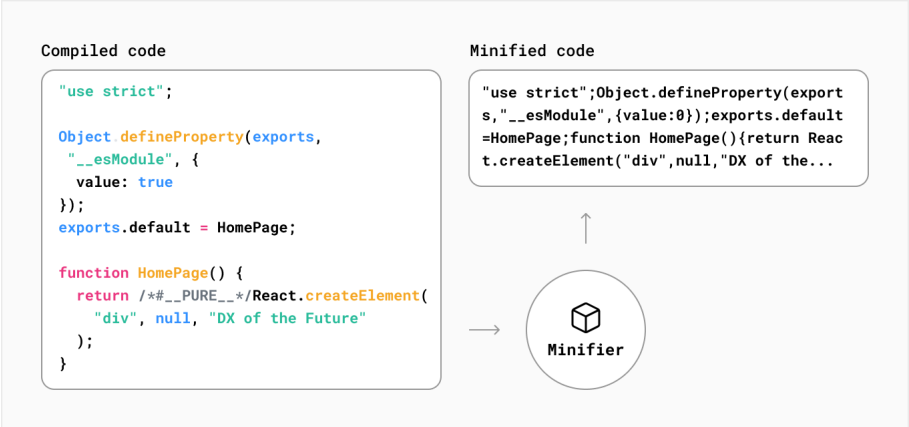

Figura 3.19: Proceso de *minificación* (optimización e integración de archivos)

Así mismo, mientras que en una aplicación React estándar, el navegador recibe un HTML vacío del servidor junto con las instrucciones JavaScript para construir la interfaz de usuario, Next.js pre-renderiza cada página por defecto. Pre-renderizar significa generar el HTML de antemano, en un servidor, en lugar de generarlo totalmente en el cliente.

En la práctica, esto significa que para una aplicación completamente renderizada en el lado del cliente, el usuario verá una página en blanco mientras se realiza el trabajo de renderizado. En comparación con una aplicación pre-renderizada, en la que el usuario verá el HTML construido.

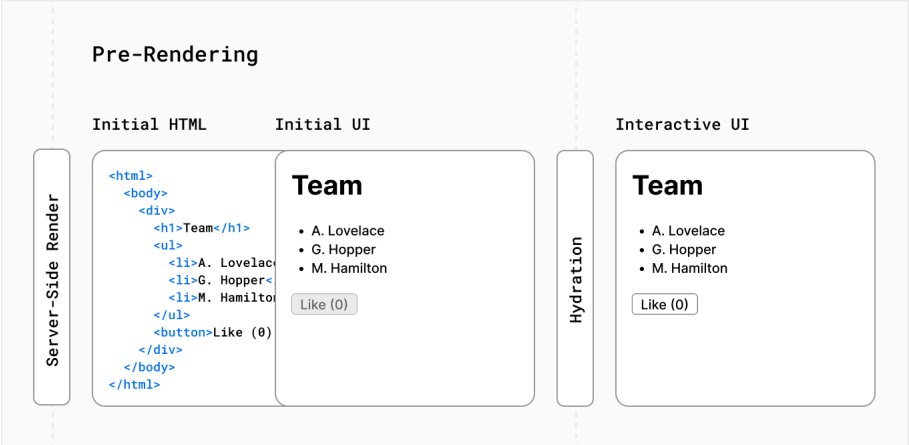

Figura 3.20: Proceso de pre-renderizado de páginas

Además Next.js por sí solo resuelve requisitos habituales de las aplicaciones, como el enrutamiento, la obtención de datos y las integraciones, al tiempo que mejora la experiencia del desarrollador y del usuario final, permitiendo crear aplicaciones web totalmente interactivas, muy dinámicas y de gran rendimiento.

3.5.1. Estructura código fuente

La estructura de directorios del código fuente del frontend, localizada en la raíz del repositorio del proyecto, es la siguiente:

```
src
├── components (Componentes)
├── hooks (Hooks personalizados)
├── pages (Páginas)
├── services (Servicios)
├── styles (Estilos CSS)
├── themes (Temas)
├── types (Tipos)
└── utils (Útiles)
```

- **/components** Contiene los componentes que conforman la interfaz de usuario. Estos actúan como piezas reutilizables de código, encapsulando la lógica de una parte específica de la aplicación, ya sea un botón, un título, un formulario, etc.
- **/hooks** Contiene los *hooks* personalizados. Un *hook* es una función que permite "enganchar" el estado y otras características de React desde componentes funcionales. Gracias a los *hooks* personalizados es posible combinar los *hooks* básicos de React o de otras librerías para extraer la lógica de un componente en funciones reutilizables.
- **/pages** Contiene las páginas de la aplicación. Se trata de un grupo de componentes complejos que están conformados por otros más básicos y que representan la interfaz de usuario de una página de la aplicación.
- **/services** Contiene los servicios que encargados de la comunicación entre el frontend y el backend. El papel de estas funciones o métodos es gestionar las llamadas a la API.
- **/styles** Contiene los estilos CSS que definen el diseño o aspecto visual de los componentes de la interfaz de usuario.
- **/themes** Contiene la definición de los temas claro y oscuro de la interfaz de usuario de la aplicación, es decir, paleta de colores, espaciados, tamaños de la fuente, etc.
- **/types** Contiene las definiciones de tipos. Al ser una aplicación desarrollada con Typescript, en ocasiones es necesario definir tipos o interfaces personalizadas, o sobrescribir las de algunas librerías.
- **/utils** Contiene funciones o métodos que proporcionan utilidades genéricas empleadas en diferentes partes del código de la aplicación.

3.5.2. Despliegue

El frontend de la aplicación ha sido desplegado en la plataforma Vercel, la cuál, al ser la desarrolladora del framework Next.js, ofrece muchas facilidades.

En primer lugar ha sido necesario crear un nuevo equipo en el sitio web de Vercel, y un nuevo proyecto al que se le ha asociado el repositorio de GitHub del proyecto. Una vez importado el proyecto, simplemente ha sido necesario seleccionar la configuración preestablecida para el framework Next.js y definir las variables de entorno necesarias para el funcionamiento del proyecto. Tras la configuración, la aplicación ha sido desplegada de forma automática.

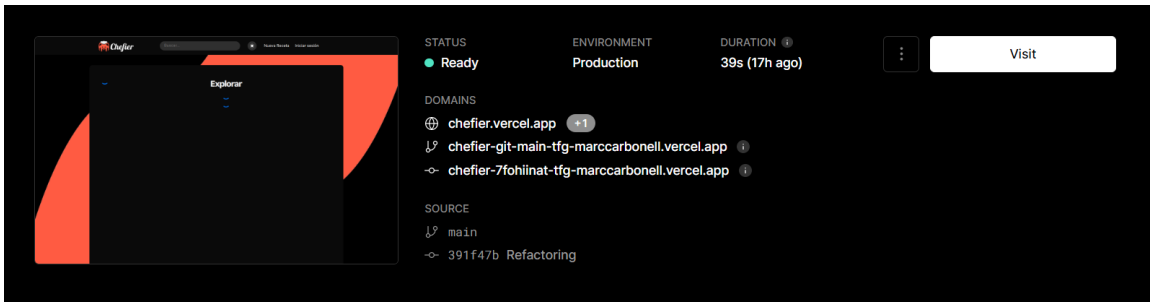

Figura 3.21: Despliegue del frontend de la aplicación en Vercel

3.6. Backend

El backend ha sido implementado sobre una arquitectura de tres capas con el objetivo principal de separar la aplicación en distintos niveles o regiones con su propia responsabilidad. De esta manera, se consigue desacoplar las capas de la aplicación, permitiendo que evolucionen de manera aislada y mejorando la mantenibilidad del código.

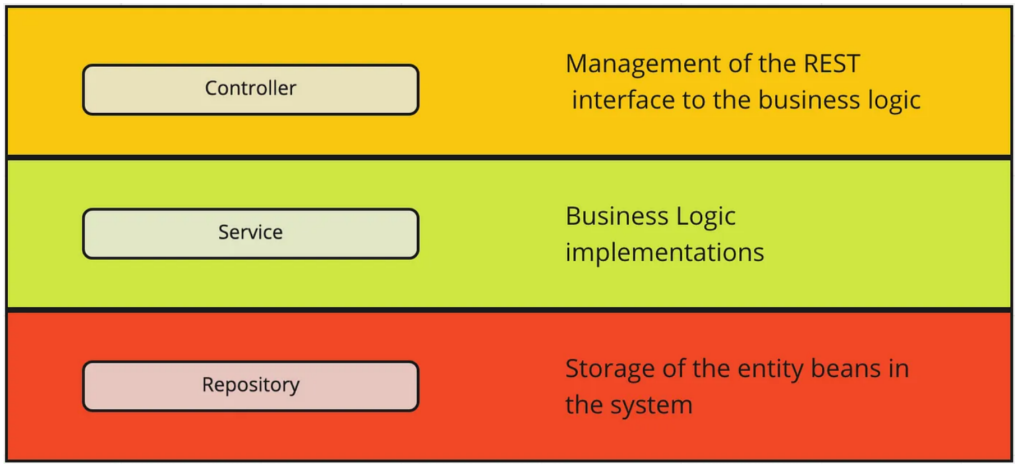

Figura 3.22: Arquitectura Controller-Service-Repository [31]

Esta arquitectura divide la aplicación en tres capas:

- Controlador: responsable de exponer las funcionalidades de la aplicación, en este caso desde los puntos de entrada de la API, y de recibir las peticiones del frontend.
- Servicio: responsable de toda la lógica de negocio de la aplicación. Desde ella se accede a la capa repositorio y sus servicios son invocados desde la capa controlador.
- Repositorio: contiene los modelos de datos de la aplicación y es responsable de almacenar y recuperar datos de la base de datos.

3.6.1. Estructura código fuente

La estructura de directorios del código fuente del backend, localizada en el directorio "server" del proyecto, es la siguiente:

```
src
├── db (Conexión con la base de datos)
├── middlewares (Middlewares)
├── models (Modelos de datos)
├── routers (Rutas de la API)
├── services (Servicios)
├── types (Tipos)
└── utils (Útiles)
```

- **/db** Contiene la configuración de la conexión con la base de datos.
- **/middlewares** Contiene los *middlewares* del servidor backend. Los *middlewares* son funciones que sirven para administrar la entrada y salida de datos de la API. Estos puede acceder y manipular tanto la solicitud entrante (*request*) como la respuesta saliente (*response*) antes de que sean manejadas por la ruta final o el controlador correspondiente.
- **/models** Correspondiente a la capa repositorio, contiene la definición de los modelos de datos de la aplicación.
- **/routers** Correspondiente a la capa controlador, contiene los puntos de entrada de la API.
- **/services** Correspondiente a la capa servicio, contiene toda la lógica de negocio de la API. En este directorio se definen las funciones o métodos encargados de interactuar con la base de datos y realizar las llamadas operaciones CRUD (Create, read, update and delete).
- **/types** Al igual que en el frontend, este directorio contiene las definiciones de tipos o interfaces personalizadas necesarios para el desarrollo con Typescript.
- **/utils** De nuevo, de forma similar al frontend, este directorio contiene funciones o métodos que proporcionan utilidades genéricas empleadas en diferentes partes del código.

3.6.2. API

La API (Application Programming Interface) de Chefier ha sido implementada con el objetivo de poder administrar los datos de la aplicación de forma segura desde el servidor backend. Esta ofrece rutas para manejar los datos de los usuarios y las recetas de la base de datos.

Chefier es una aplicación que permite el acceso de forma libre al contenido, pero que requiere de iniciar sesión para interactuar con el mismo, por lo tanto, por cuestiones de seguridad, se ha optado por proteger las rutas que permiten modificar o eliminar datos de la base de datos. De esta forma, para acceder a ellas es necesario proporcionar un token de acceso que únicamente puede obtenerse a través del inicio de sesión.

Para ofrecer una descripción detallada de cada una de las rutas de acceso y los esquemas de peticiones y respuestas de la API, se ha creado una documentación [32] que ofrece toda la información necesaria para la correcta utilización de la misma, empleando la herramienta para documentación de APIs Swagger [33].

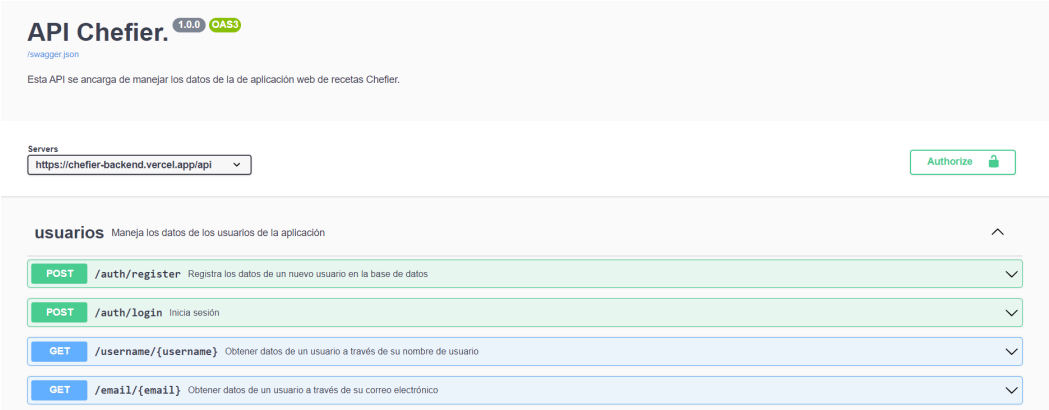

Figura 3.23: Documentación de la API de Chefier en Swagger

3.6.3. Despliegue

Al igual que el frontend, el backend de la aplicación ha sido desplegado en la plataforma Vercel debido a las facilidades que ofrece, sin embargo, en este caso el proceso ha sido algo diferente.

En primer lugar, no ha sido necesario crear un nuevo equipo en el sitio web de Vercel si no que se ha utilizado el mismo que para el frontend.

En cuanto a la configuración se ha seleccionado la opción "otro" como framework, ya que Koa.js no está disponible por el momento, y el directorio "server" como raíz del proyecto, y se han configurado las variables de entorno. El resto de la configuración, en la que se establece como ejecutar el proyecto, se ha realizado en un fichero "vercel.json" dentro del código fuente del proyecto.

```
{
  "version": 2,
  "builds": [
    {
      "src": "./dist/index.js",
      "use": "@vercel/node"
    }
  ],
  "rewrites": [
    {
      "source": "/(.*)",
      "destination": "/dist/index.js"
    }
  ],
  "headers": [
    {
      "source": "/(.*)",
      "headers": [
        {
          "key": "Access-Control-Allow-Credentials",
          "value": "true"
        },
        {
          "key": "Access-Control-Allow-Origin",
          "value": "*"
        },
        {
          "key": "Access-Control-Allow-Methods",
          "value": "GET,OPTIONS,PATCH,DELETE,POST,PUT"
        },
        {
          "key": "Access-Control-Allow-Headers",
          "value": "X-CSRF-Token, X-Requested-With, Accept, Accept-Version, Content-Length, Content-MD5, Content-Type, Date,
              X-Api-Version"
        }
      ]
    }
  ]
}
```

Listado 3.1: Archivo de configuración del despliegue "vercel.json"

Tras dicha configuración, el backend de la aplicación ha sido desplegado de forma automática. Cabe señalar que ha sido necesario incluir el directorio "dist", generado de la *transpliación* (traducción fuente a fuente) del código typescript a javascript, en el repositorio para que el despliegue pueda realizarse correctamente.

3.7. Base de datos

Como base de datos de la aplicación, se ha optado por MongoDB, una base de datos *NoSQL* o no relacional en la que la información no se almacena en tablas sino mediante documentos. Dicha elección se debe a que este tipo de bases de datos ofrece una mayor flexibilidad a la hora de crear esquemas de datos y una mayor escalabilidad que las bases de datos relacionales.

3.7.1. Despliegue

El despliegue de la base de datos de la aplicación se ha realizado en MongoDB Atlas. Para ello se han seguido una serie de pasos. En primer lugar, una vez registrado en el sitio web de la herramienta, se ha creado una organización llamada TFG-2223 y un proyecto llamado Chefier.

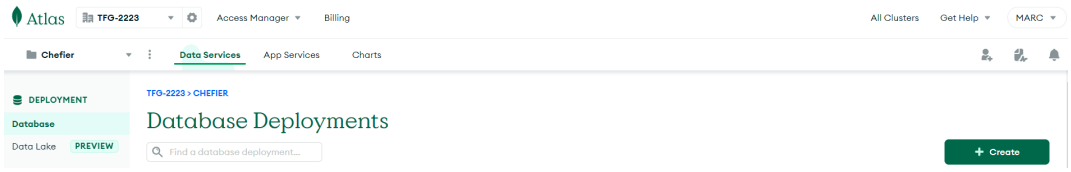

Figura 3.24: Página del proyecto en MongoDB Atlas

A continuación, se ha creado un clúster, haciendo clic en el botón “Create” y se ha seleccionado el clúster compartido (que es la opción gratuita), dejando por defecto el resto de opciones de configuración.

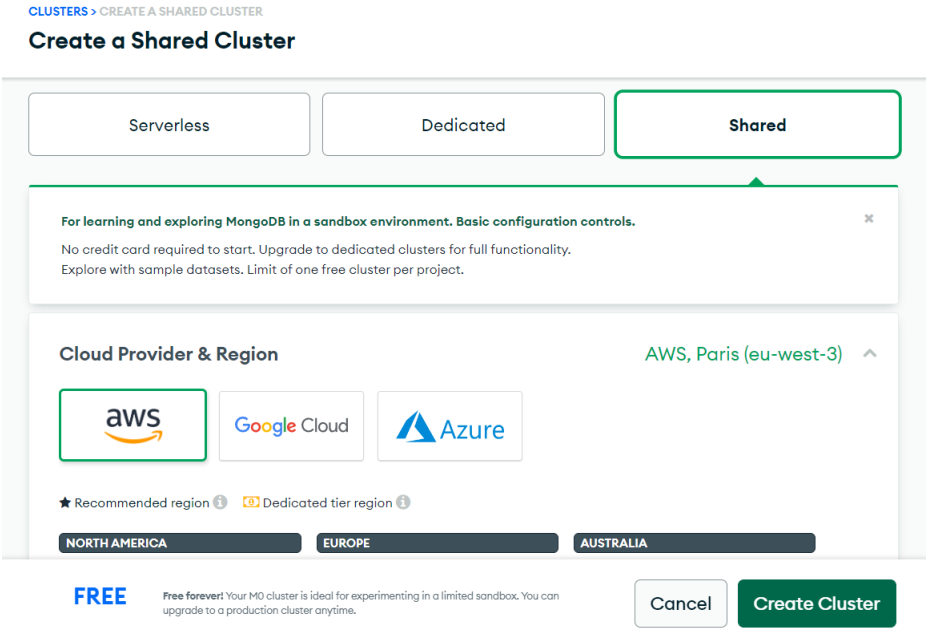

Figura 3.25: Menú de creación de clúster

Tras esto se ha seleccionado nombre de usuario y contraseña como método de autenticación, especificando un usuario y una contraseña, y respecto a la IP de conexión, se ha indicado el valor 0.0.0.0/0, que representa a cualquier IP válida. Finalmente se ha creado el clúster pulsando sobre “Finish and Close”.

Para obtener la URL de conexión a la base de datos se ha pulsado sobre el botón “Connect” y “Connect your application”.

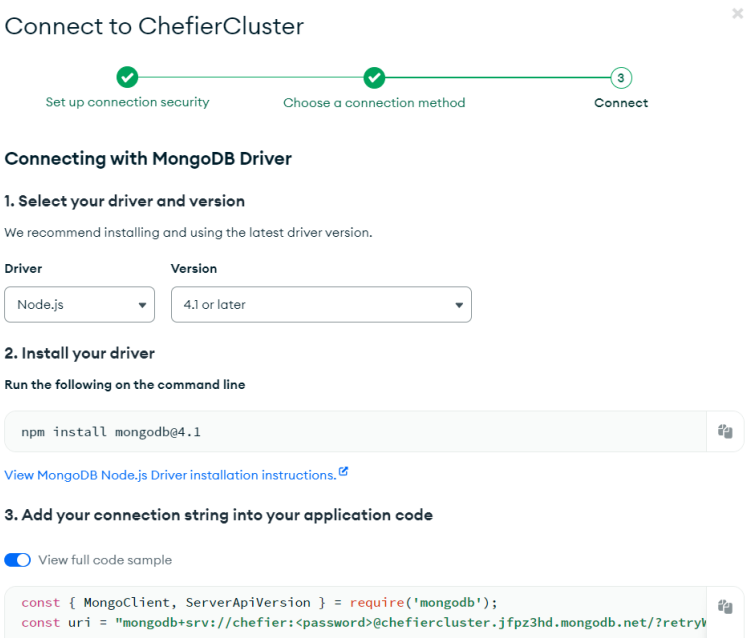

Figura 3.26: Obtención de url de conexión a la base de datos

Capítulo 4

Descripción de la aplicación

En este capítulo se describirán y mostrarán todas las funcionalidades que ofrece la aplicación en su estado final, entre ellas se encuentran registro e inicio de sesión, visualización y publicación de recetas, búsqueda y filtrado de resultados e interacción con el contenido.

- **Registro e inicio de sesión**

  En primer lugar, tenemos las páginas de registro e inicio de sesión en las que el usuario puede autenticarse a través de credenciales o mediante los proveedores de Google o GitHub.

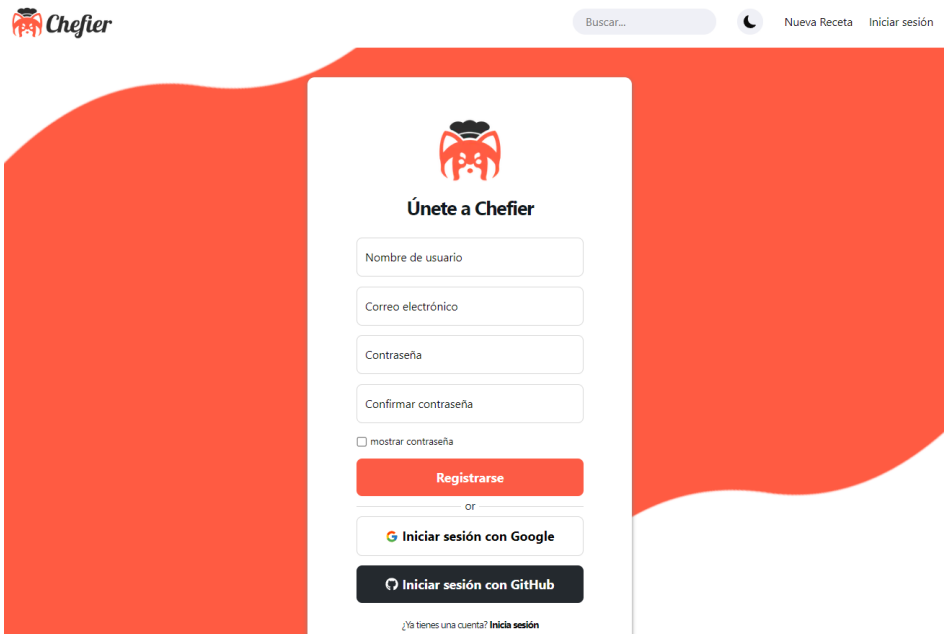

Figura 4.1: Página de registro

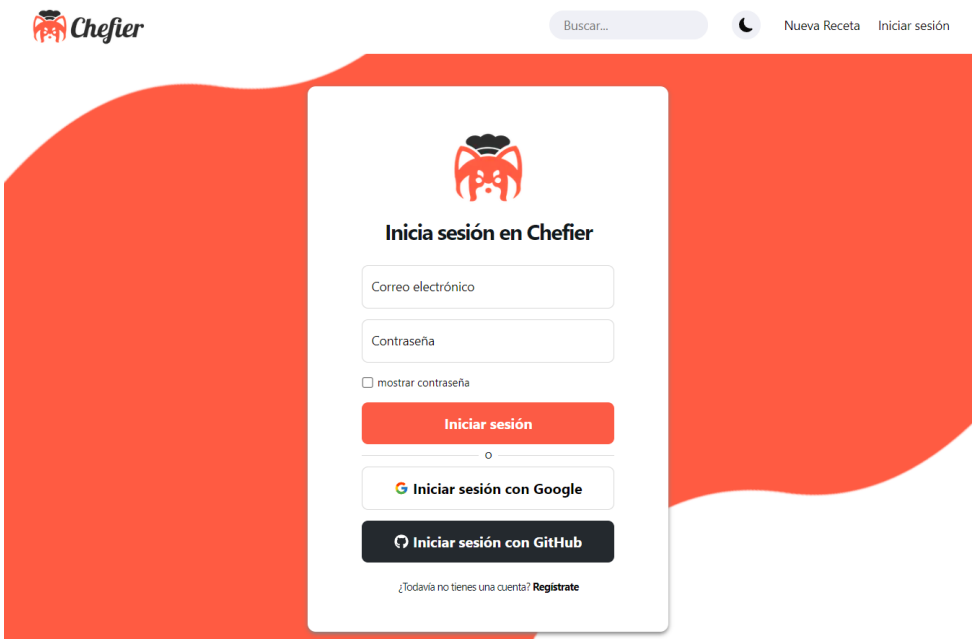

Figura 4.2: Página de inicio de sesión

El contenido de la aplicación es visualizable para cualquier usuario, pero para poder acceder a las funcionalidades de interacción con el mismo es necesario iniciar sesión.

- **Inicio**

  En la página de inicio se muestra una lista con las recetas publicadas en la aplicación, el usuario podrá explorar dicha lista mediante la barra de búsqueda situada en la barra de navegación y los filtros de resultados localizados en la zona izquierda de la pantalla.

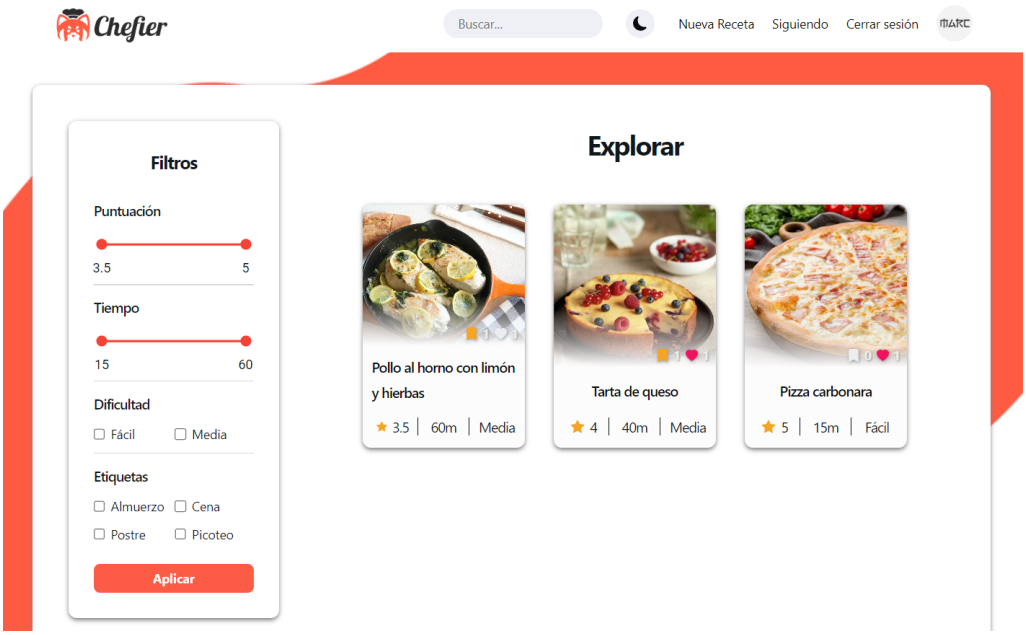

Figura 4.3: Página de inicio

Además, en caso de haber iniciado sesión, es posible cambiar a la sección de 'Siguiendo' desde la barra de navegación para visualizar las recetas publicadas por usuarios seguidos, así como dar me gusta o guardar una publicación desde la misma pantalla de inicio, sin tener la necesidad de acceder a la página de la propia receta.

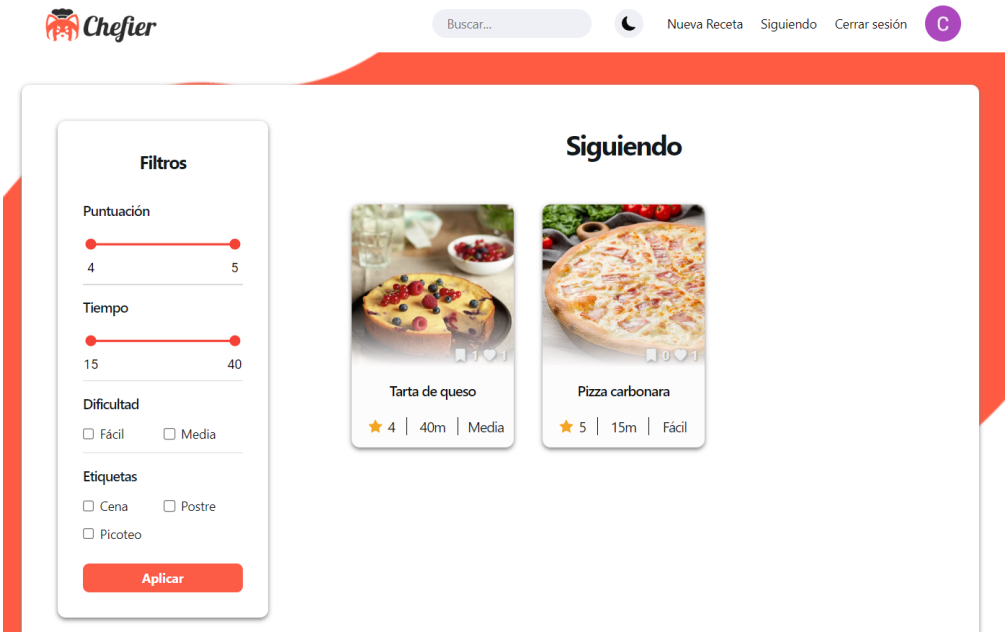

Figura 4.4: Sección de "siguiendo" de la página de inicio

- **Creación de recetas**

  Desde esta página es posible crear y publicar una receta. Esta funcionalidad requiere también de haber iniciado sesión en la aplicación.

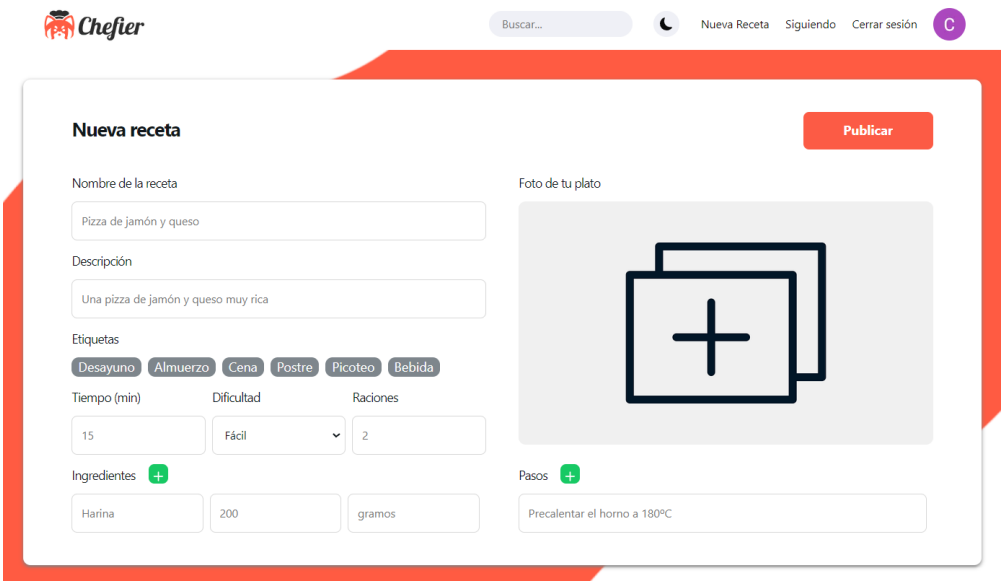

Figura 4.5: Página de creación de recetas

Para crear una receta es necesario introducir campos como el nombre de la propia receta, una breve descripción, una imagen, al menos un ingrediente y un paso de la elaboración, etc.

- **Publicación/receta**

  En la página de visualización de cada receta se muestra información de la misma como el nombre, la puntuación media, el tiempo de preparación, los ingredientes o los pasos para la elaboración.

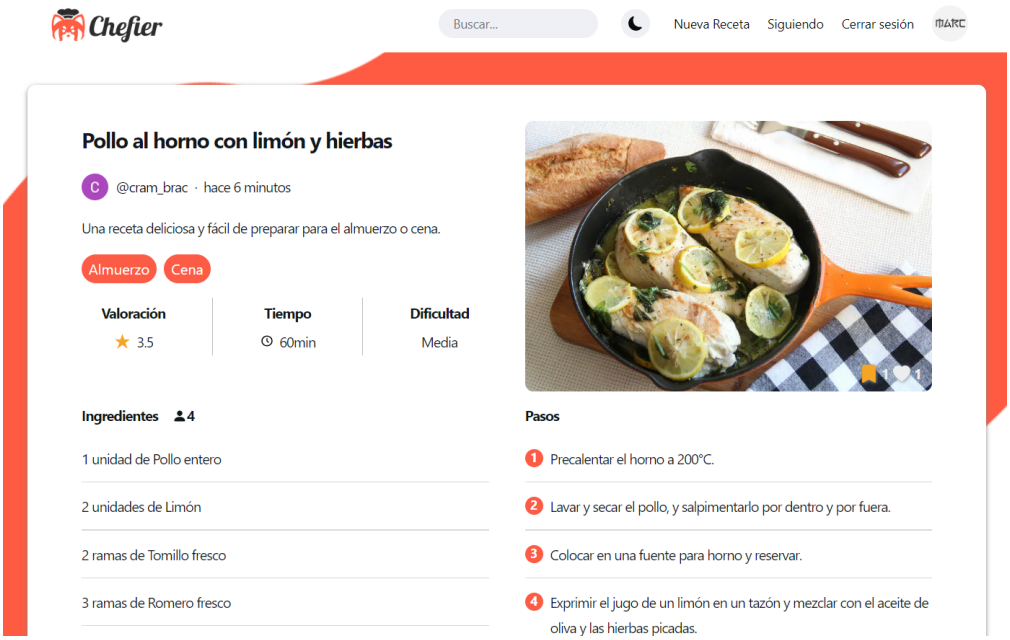

Figura 4.6: Página de publicación/receta

Además, en caso de haber iniciado sesión, es posible dar me gusta a la receta, guardarla o incluso dar una valoración, puntuándola bajo un nivel de 5 estrellas y haciendo una reseña de la misma.

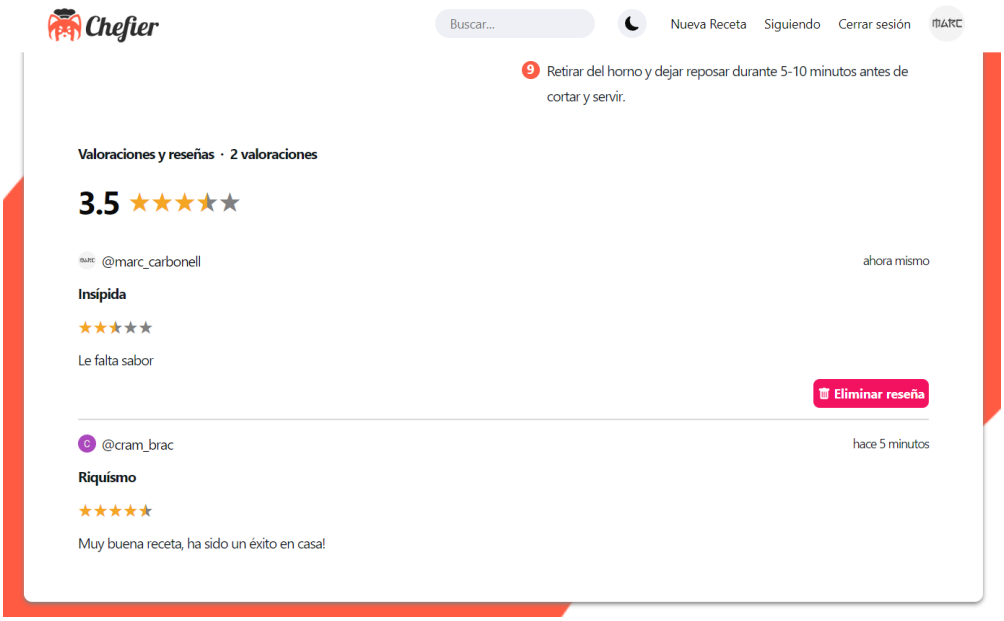

Figura 4.7: Valoraciones y reseñas de una receta

- **Perfil de usuario**

  En esta página se podrá visualizar información básica de un usuario como su imagen de perfil, nombre de usuario y arroba, algunas estadísticas como el número recetas publicadas, seguidores y seguidos, y las listas de recetas, me gustas y guardados del usuario en cuestión.

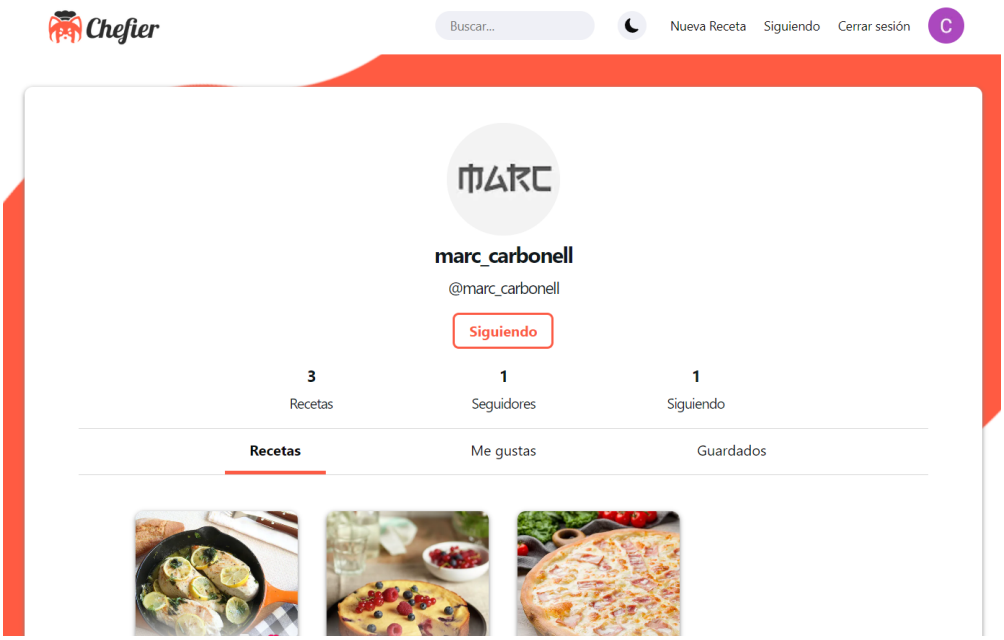

Figura 4.8: Perfil de usuario

Así mismo, desde esta página es posible seguir o dejar de seguir a otro usuario en caso de haber iniciado sesión.

- **Página no encontrada**

  Esta página se mostrará en caso de intentar acceder a una página que no exista o que no se encuentre disponible.

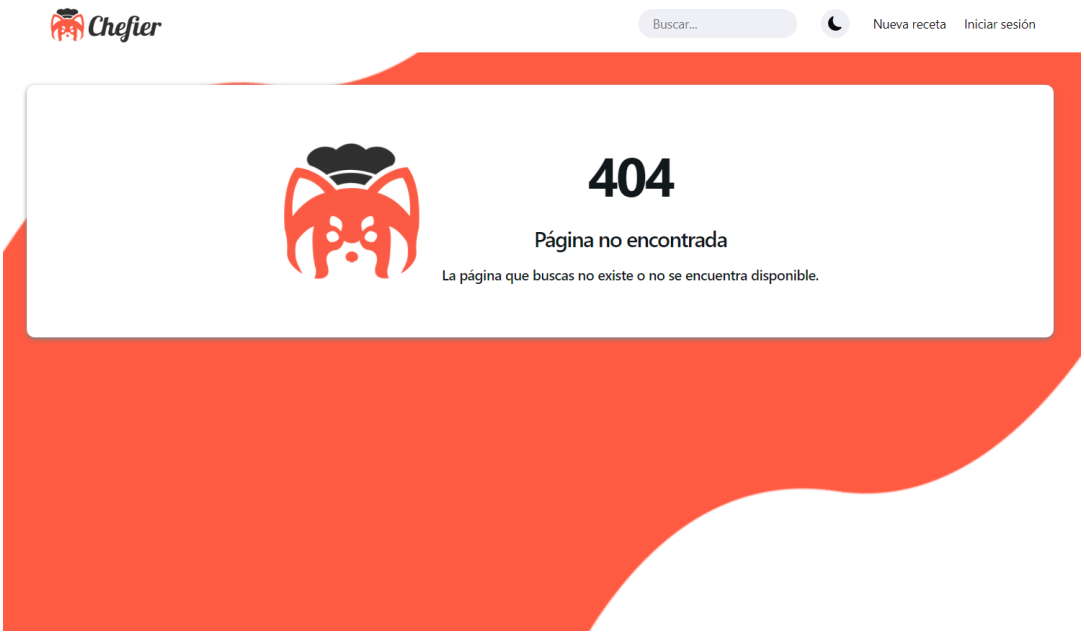

Figura 4.9: Página no encontrada

Cabe destacar que Chefier es una aplicación con diseño responsivo, por lo que su interfaz es capaz de adaptarse a cualquier dispositivo, sea ordenador, móvil o tableta. Además, es posible personalizar el aspecto de dicha interfaz, pudiendo seleccionar entre los temas claro y oscuro.

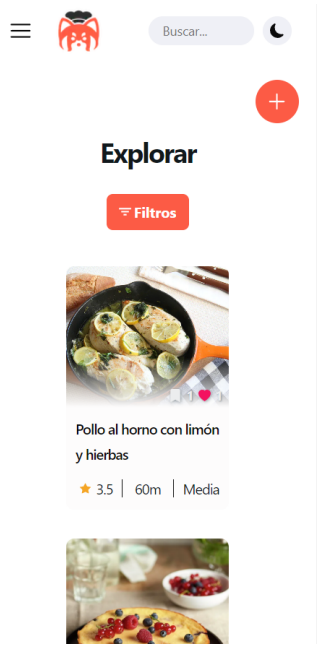

Figura 4.10: Página de inicio en un dispositivo móvil

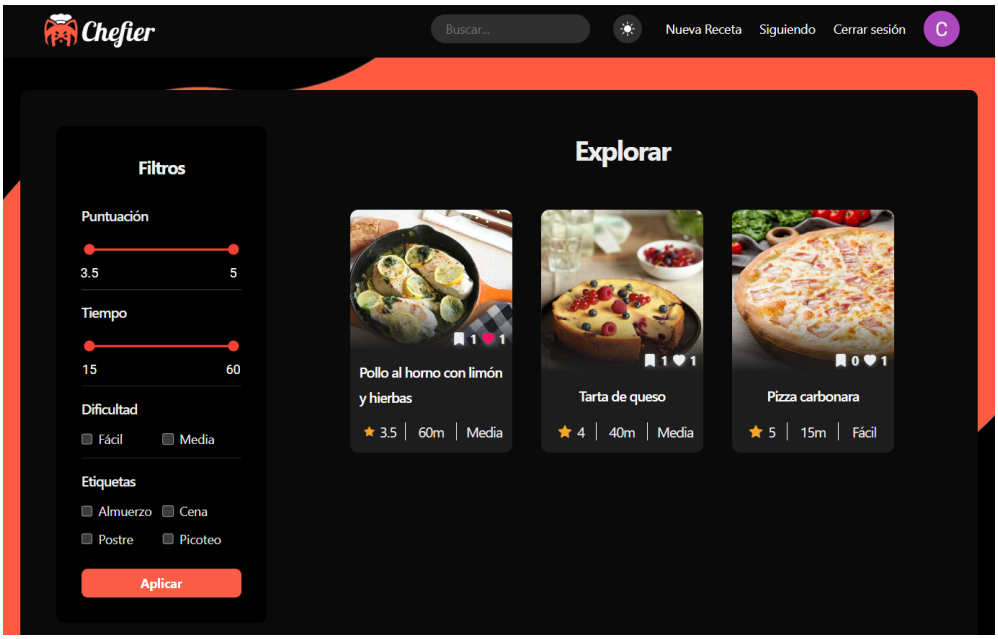

Figura 4.11: Página de inicio con el tema oscuro

Capítulo 5

Conclusiones y líneas futuras

Como conclusiones, este trabajo me ha permitido descubrir lo que supone llevar a cabo un proyecto full-stack de forma individual, con los numerosos factores que intervienen en el desarrollo del mismo, y con los añadidos de tener un tiempo limitado y de ser la primera vez que se afronta un proyecto de estas características. Este tipo de proyectos te obliga a mantener un orden en la ejecución de tareas y prestar especial atención a la gestión del proyecto y la metodología de trabajo aplicadas, pues esta es la mejor manera de poder llevar a cabo el desarrollo de forma organizada y así poder asegurar el éxito del mismo.

Además, este ha sido un proyecto que no solo ha servido para demostrar todos los conocimientos adquiridos durante la carrera, si no que también me ha permitido asentar algunos conocimientos y adquirir otros nuevos en el ámbito del desarrollo de aplicaciones web, como el uso de nuevos frameworks y tecnologías, o la aplicación de nuevas metodologías de trabajo. Esto ha resultado en una experiencia muy enriquecedora que ha servido para seguir formándome en el perfil profesional de desarrollador full-stack, al que quiero dirigir mi futura carrera profesional.

En cuanto a posibles líneas futuras del proyecto, si tenemos en cuenta el anteproyecto del trabajo, podemos ver como el estado actual de la aplicación está bastante completo, pero han quedado ciertas funcionalidades planteadas en el inicio por implementar. Si bien, sí que posee la gran mayoría de las funcionalidades básicas, queda pendiente la configuración de los datos de perfil y la implementación de migas de pan para la navegación entre pantallas de la aplicación.

La razón principal de no haber podido implementar todas las funcionalidades planteadas en un inicio ha sido la falta de tiempo. Al comienzo del proyecto fue necesario dedicar una gran parte del tiempo a la selección de las tecnologías y herramientas a utilizar. Durante este período realicé varios cursos y estudié la documentación disponible de algunas tecnologías como React y Next.js para aprender cómo funcionan y cómo utilizarlas. Esto, junto a la fase inicial de diseño de la aplicación, hicieron que dispusiera de menos tiempo para el desarrollo de la aplicación en sí.

Por otro lado, algunas posibles mejoras a futuro de la aplicación podrían ser la implementación de un chat directo entre usuarios, la posibilidad de responder o comentar las valoraciones y reseñas de las recetas, compartir publicaciones o brindar la posibilidad a los usuarios de incluir más imágenes o incluso vídeos del proceso de elaboración en cada receta. Lo que está claro es que, debido a la naturaleza del proyecto, este permite un sinfín de actualizaciones o mejoras futuras que no se han implementado por falta de tiempo, pero que podrían dar lugar a años de desarrollo y mejora continua de la aplicación para poder dar si quiera por concluido el proyecto.

Capítulo 6

Summary and Conclusions

In conclusion, this work has allowed me to discover what it means to carry out a full-stack project individually, with the numerous factors that intervene in its development, and with the bonus of having limited time and being the first time that a project of these characteristics has been undertaken. This type of project forces you to maintain order in the execution of tasks and pay special attention to the project management and the work methodology applied, as this is the best way to carry out the development in an organised manner and thus ensure its success.

In addition, this has been a project that has not only served to demonstrate all the knowledge acquired during the degree but has also allowed me to consolidate some knowledge and acquire new ones in the field of web application development, such as the use of new frameworks and technologies, or the application of new work methodologies. This has resulted in a very enriching experience that has continued training me in the professional profile of a full-stack developer, to which I want to direct my future career.

As for possible future lines of the project, if we consider the preliminary draft of the work, we can see how the current state of the application is quite complete, but there are still some functionalities to be implemented. Although it does have most of the basic functionalities, the configuration of the profile data and the implementation of breadcrumbs for navigation between screens of the application are still pending.

The lack of time was the main reason for not being able to implement all the functionalities initially proposed. At the beginning of the project, it was necessary to spend a great deal of time selecting the technologies and tools to be used. During this period, I took several courses and studied the available documentation of some technologies, such as React and Next.js, to learn how they work and how to use them. This, together with the initial design phase of the application, meant that I had less time to develop the application itself.

On the other hand, some possible future improvements of the application could be the implementation of a direct chat between users, the possibility of responding or commenting on the ratings and reviews of the recipes, sharing publications or giving users the option of including more images or even videos of the elaboration process in each recipe. What is clear is that the nature of the project allows for an endless number of future updates or improvements that have not been implemented due to lack of time but could lead to years of development and continuous application improvement before the project can even be considered complete.

Capítulo 7

Presupuesto

En este capítulo se hará una estimación del presupuesto del proyecto para un desarrollo de 6 meses, teniendo en cuenta el alojamiento de la aplicación en la nube, el almacenamiento de datos y las horas de trabajo computadas.

En primer lugar, para el despliegue y alojamiento de los servidores frontend y backend, será necesario contratar como mínimo el plan pro de Vercel, que proporciona hasta 1 TB de ancho de banda, solicitudes ilimitadas y 1000 GB/hora de ejecución, y que tiene un precio de 20 dólares mensuales.

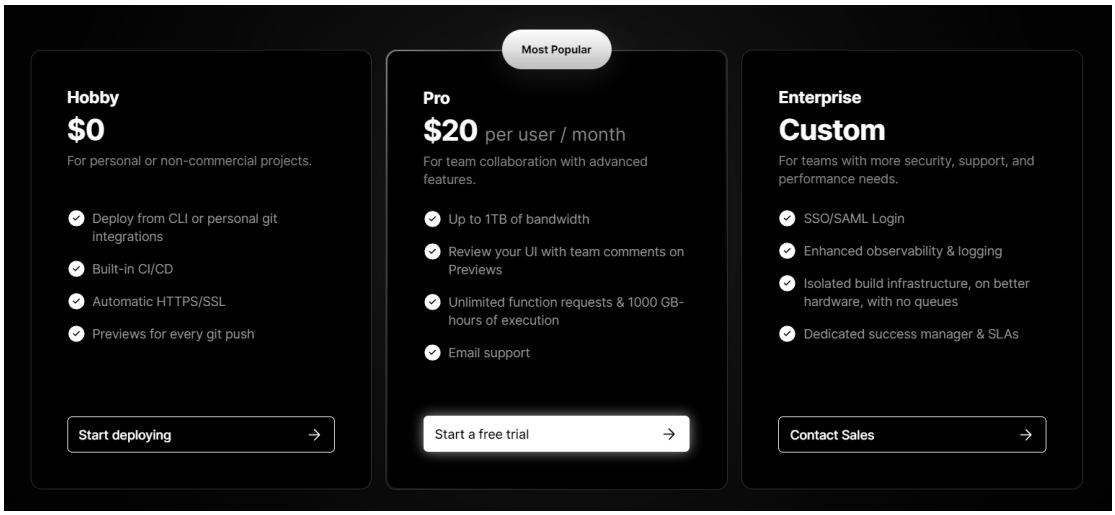

Figura 7.1: Planes disponibles en Vercel

Por otro lado, para poder desplegar y alojar la base de datos en MongoDB Atlas será necesario contratar el plan dedicado, que por 57 dolares al mes proporciona de 10 GB a 4 TB de almacenamiento, de 2 GB a 768 GB de RAM, aislamiento de red con controles de acceso detallados y opciones de multirregión y multicloud.

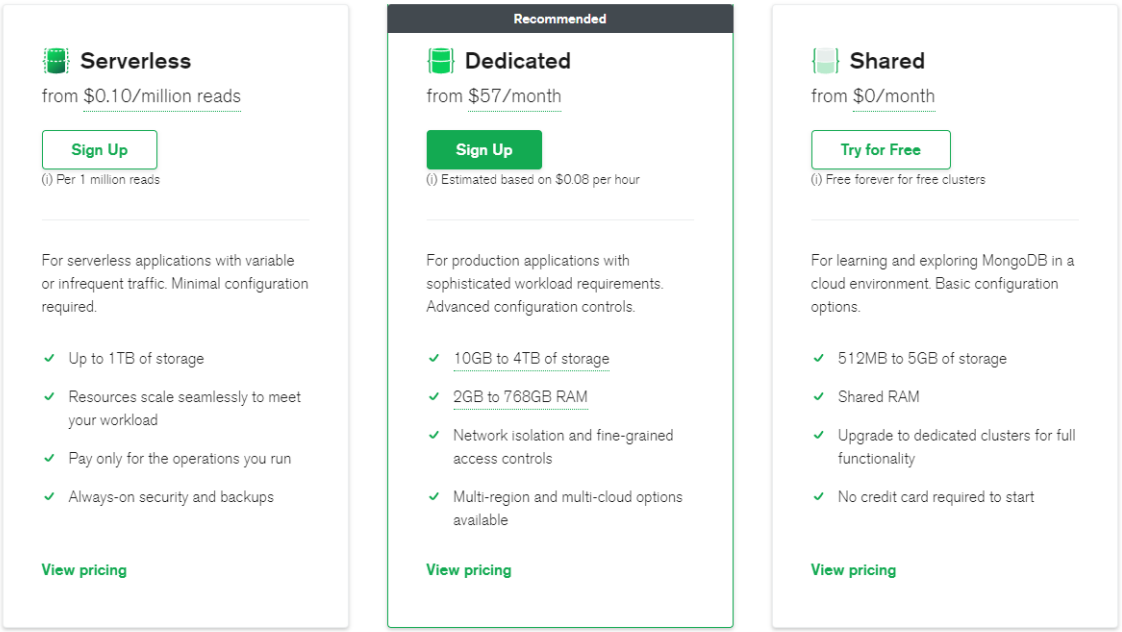

Figura 7.2: Planes disponibles en MongoDB Atlas

Por último, suponiendo un sueldo para un desarrollador junior de unos 1200€/mes, durante 6 meses de desarrollo, la estimación del coste total sería la siguiente:

Concepto	Conste mensual	Coste total
Desarrollo de la aplicación	1200€	7200€
Despliegue y alojamiento	18,53€	111,18€
Almacenamiento de datos	52,80	316,80€
Total	1271,33€	7627,98€

Tabla 7.1: Estimación del presupuesto del proyecto

Bibliografía

[1] Myrealfood, la app para mejorar tu estilo de vida. https://myrealfood.app/, 2023. Accessed: 2023-02-14.

[2] Cookpad, cocina casera más divertida. https://cookpad.com/es/home, 2023. Accessed: 2023-02-14.

[3] Funcook, descubre y comparte recetas de cocina. https://funcook.com/, 2023. Accessed: 2023-02-14.

[4] Petitchef, recetas de cocina. https://www.petitchef.es/, 2023. Accessed: 2023-02-14.

[5] Git, a free and open source distributed version control system. https://git-scm.com/, 2023. Accessed: 2023-02-14.

[6] Github, let's build from here. https://github.com/, 2023. Accessed: 2023-02-14.

[7] Typescript, javascript with syntax for types. https://www.typescriptlang.org/, 2023. Accessed: 2023-03-01.

[8] Chefier. https://chefier.vercel.app, 2023. Accessed: 2023-05-23.

[9] Mongoose, the developer data platform. https://www.mongodb.com/, 2023. Accessed: 2023-04-24.

[10] Express, infraestructura web rápida, minimalista y flexible para node.js. https://expressjs.com/es/, 2023. Accessed: 2023-04-24.

[11] React, la biblioteca para interfaces de usuario web y nativas. https://es.react.dev/, 2023. Accessed: 2023-04-24.

[12] Node.js, un entorno de ejecución para javascript construido con v8, motor de javascript de chrome. https://nodejs.org/es/, 2023. Accessed: 2023-02-14.

[13] Next.js, the react framework for the web. https://nextjs.org/, 2023. Accessed: 2023-02-14.

[14] Koa, next generation web framework for node.js. https://koajs.com/, 2023. Accessed: 2023-02-14.

[15] Mongodb atlas, database. deploy a multi-cloud database. https://www.mongodb.com/atlas/database, 2023. Accessed: 2023-02-14.

[16] Github projects, an adaptable, flexible tool for planning and tracking work on github. https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects, 2023. Accessed: 2023-03-01.

[17] Vercel, develop. preview. ship. https://next-auth.js.org/, 2023. Accessed: 2023-02-14.

[18] Swr, biblioteca react hooks para la obtención de datos. https://swr.vercel.app/es-ES, 2023. Accessed: 2023-02-14.

[19] Sunny yang, swr: Frontend data fetching and caching. https://medium.com/nerd-for-tech/swr-frontend-data-fetching-and-caching-ca0313239d6f, 2021. Accessed: 2023-02-14.

[20] Nextauth.js, authentication for next.js. https://next-auth.js.org/, 2023. Accessed: 2023-02-14.

[21] Imagekit, an integrated media library and aws cdn. https://imagekit.io/, 2023. Accessed: 2023-04-24.

[22] Vitest, authentication for next.js. BlazingFastUnitTestFramework, 2023. Accessed: 2023-02-14.

[23] Cypress, test. automate. accelerate. https://www.cypress.io/, 2023. Accessed: 2023-02-14.

[24] Adobe illustrator, diseña unos gráficos espectaculares. https://www.adobe.com/es/products/illustrator.html?promoid=RYGDN24L&mv=other, 2023. Accessed: 2023-02-14.

[25] Photoshop, para todos. https://www.adobe.com/es/products/photoshop.html, 2023. Accessed: 2023-02-14.

[26] Repositorio del proyecto. https://github.com/ULL-TFGyMs-vblanco/TFG-2023-MarcCarbonellGonzalezdeChaves-Chefier, 2023. Accessed: 2023-05-23.

[27] Jsonwebtoken. https://github.com/auth0/node-jsonwebtoken#readme, 2023. Accessed: 2023-02-14.

[28] Bcrypt. https://github.com/kelektiv/node.bcrypt.js#readme, 2023. Accessed: 2023-02-14.

[29] Coveralls, deliver better code. https://coveralls.io/, 2023. Accessed: 2023-02-14.

[30] Sonarcloud, clean code in your cloud workflow with SonarCloud. https://www.sonarsource.com/products/sonarcloud/, 2023. Accessed: 2023-02-14.

[31] Tom collings, controller-service-repository. https://tom-collings.medium.com/controller-service-repository-16e29a4684e5, 2021. Accessed: 2023-02-14.

[32] Documentación api chefier. https://chefier-backend.vercel.app/docs, 2023. Accessed: 2023-05-20.

[33] Swagger,api development for everyone. https://swagger.io/, 2023. Accessed: 2023-05-20.