UNIVERSIDAD EUROPEA DE MADRID

DESARROLLO DE APLICACIONES WEB

RECIPES

TFC: Recipe Realm

ARIS MAXIMILIAM KUHS

DANI GARRIDO NUÑEZ

JUAN DIEGO MOTTA MONCADA

DIRIGIDO POR

IRENE DEL RINCÓN

SARA VILLANUEVA

Junio 2024

ÍNDICE

1. INTRODUCCIÓN

Recipe Realm es una aplicación web diseñada para mejorar la búsqueda de recetas y alimentos basados en las preferencias del usuario. La idea es que Recipe Realm ayude a optimizar el tiempo en la cocina, reduciendo el desperdicio de alimentos y simplificando el proceso de preparación de comidas.

1.1 Motivación

La motivación principal detrás de este proyecto es proporcionar una herramienta que no solo recomiende recetas basadas en preferencias, alérgenos, tipos de dieta e ingredientes disponibles, sino que también facilite la creación de nuevas recetas y la compra de platos saludables.

1.2 Objetivos generales

Los objetivos generales de Recipe Realm son:

- **Facilitar la creación y descubrimiento de recetas:** Proporcionando una plataforma intuitiva donde los usuarios pueden crear, y descubrir nuevas recetas.

- **Promover una alimentación saludable y personalizada:** Ofreciendo recomendaciones de recetas que se ajusten a las preferencias y necesidades dietéticas de los usuarios.

- **Optimizar la gestión del inventario de alimentos:** Permitiendo a los usuarios llevar un control detallado de los ingredientes disponibles en su cocina.

- **Permitir la compra de recetas saludables:** Ofreciendo un surtido de platos cuidadosamente seleccionados, asegurando así que los usuarios tengan acceso fácil y directo a una alimentación equilibrada y saludable.

2. MÓDULOS APLICADOS

En el proceso de desarrollo de la plataforma, se logró implementar la gran mayoría de los conocimientos adquiridos a lo largo del curso, abarcando todas las asignaturas correspondientes, que incluyen Entorno Servidor, Entorno Cliente y Diseño de Interfaces Web. En esta sección, se proporcionará un esquema que detalla qué módulos y conocimientos específicos se han utilizado en el proyecto:

- **Entorno Cliente**
    - Conceptos esenciales
    - Propiedades y métodos
    - Funciones
    - Manejo de ventanas
    - Formularios
    - Eventos
    - Validación de formularios
    - JSON y Fetch
- **Entorno Servidor**
    - Conceptos esenciales
    - Estructuras de control
    - Funciones
    - Datos compuestos
    - Formularios
    - Directorios
    - Sesiones
    - Integración con Fetch y JSON
- **Diseño de Interfaces web**
    - Investigación de la competencia
    - SiteMap
    - Wireframes y Mockups
    - Kit de componentes

Estas secciones proporcionan una visión concisa de los diversos módulos que se han implementado a lo largo del curso. Será en la sección de Fases del proyecto donde se profundizará más en cada uno de estos módulos, brindando una explicación detallada de cómo se llevaron a cabo las tareas y qué herramientas se utilizaron en cada caso. La integración de todos estos conocimientos y secciones fue esencial para poder desarrollar la aplicación con todas las funcionalidades necesarias.

2.1 Entorno cliente

En el desarrollo del proyecto de recetas, se han implementado varios módulos formativos relacionados con el entorno cliente utilizando JavaScript. Estos módulos han permitido dominar diversas habilidades y construir una interfaz de usuario interactiva y eficiente. A continuación, se presenta una breve explicación de cada módulo.

2.1.1 Conceptos Esenciales

En este módulo, se cubrieron los fundamentos de JavaScript, incluyendo variables, tipos de datos, operadores y estructuras de control (bucles y condicionales). Estos conceptos forman la base sobre la cual se construyeron el resto de las funcionalidades del proyecto de recetas. El entendimiento de estos fundamentos es crucial para cualquier programación efectiva en JavaScript.

2.1.2 Propiedades y Métodos

Este módulo se centró en el manejo de objetos en JavaScript. Se estudiaron las formas de definir propiedades y métodos dentro de un objeto, lo que permitió una gestión más estructurada y eficiente de los datos de las recetas. La utilización correcta de objetos es esencial para la organización del código y la reutilización de funcionalidades.

2.1.3 Funciones

El módulo de funciones abordó la creación y uso de funciones en JavaScript, incluyendo funciones declarativas, anónimas y flecha. Las funciones son vitales para modularizar y reutilizar el código. En el contexto del proyecto, las funciones se utilizaron para operaciones repetitivas como el cálculo de ingredientes, el manejo de eventos y la validación de datos.

2.1.4 Manejo de Ventanas

En este módulo, se aprendió a controlar las ventanas del navegador, incluyendo la apertura, cierre y manipulación de nuevas ventanas y pestañas. Este conocimiento es útil para la gestión de interacciones avanzadas, como abrir páginas adicionales para visualizar detalles de recetas o realizar acciones específicas sin recargar la página principal, mejorando así la experiencia del usuario.

2.1.5 Formularios

Se profundizó en la creación y manipulación de formularios HTML mediante JavaScript. Se aprendió a capturar datos de entrada del usuario, una funcionalidad fundamental para agregar nuevas recetas, buscar recetas por ingredientes o categorías, y permitir a los usuarios ver todos los contenidos de todas las secciones de forma customizable. La correcta gestión de formularios es crucial para la interacción y la recolección de datos en la aplicación.

2.1.6 Eventos

El manejo de eventos en JavaScript fue abordado en este módulo, incluyendo eventos de clic, cambios de formulario y otros eventos de usuario. Este conocimiento fue clave para hacer la aplicación interactiva. Por ejemplo, se utilizaron eventos para actualizar dinámicamente la lista de recetas en función de los filtros aplicados por el usuario, o para mostrar u ocultar secciones de la página según las acciones del usuario.

2.1.7 Validación de Formularios

La validación de formularios es esencial para asegurar que los datos ingresados por los usuarios sean correctos y seguros. En este módulo, se implementaron técnicas de validación tanto del lado del cliente (con JavaScript) como del lado del servidor. Esto incluyó verificar que los campos obligatorios estuvieran completos, que los datos tuvieran el formato correcto y prevenir la inserción de datos maliciosos.

2.1.8 JSON y Fetch

Finalmente, se aprendió a trabajar con JSON (JavaScript Object Notation) y la API Fetch para realizar solicitudes HTTP. Este módulo permitió integrar el proyecto con APIs externas y gestionar datos de manera asíncrona. Por ejemplo, se utilizó Fetch para obtener recetas de una base de datos en línea y mostrar los resultados en la página de manera dinámica. Además, se manejó el almacenamiento y recuperación de datos del usuario utilizando JSON, lo que mantuvo una experiencia de usuario coherente y fluida.

2.2 Entorno Servidor

En el desarrollo del proyecto de recetas, se han implementado varios módulos formativos relacionados con el backend utilizando PHP. Estos módulos han permitido dominar diversas

habilidades y construir una aplicación robusta siguiendo la arquitectura MVC (Modelo-Vista-Controlador). A continuación, se presenta una breve explicación de cada módulo.

2.2.1 Conceptos Esenciales

Se cubrieron los conceptos básicos de PHP, como variables, tipos de datos y operadores. Este módulo sentó las bases para el desarrollo en PHP, permitiendo escribir scripts básicos y comprender cómo PHP se integra con HTML para generar contenido dinámico.

2.2.2 Estructuras de Control

Este módulo abarcó las estructuras de control en PHP, incluyendo condicionales (if, else, switch) y bucles (for, while, foreach). Estas estructuras son fundamentales para controlar el flujo de ejecución del código y gestionar la lógica de negocio en el backend del proyecto.

2.2.3 Funciones

Se exploró la creación y uso de funciones en PHP. Las funciones permiten modularizar el código y hacerlo más reutilizable. En el proyecto, las funciones se utilizaron para tareas repetitivas como la validación de datos, el manejo de sesiones y la interacción con la base de datos.

2.2.4 Datos Compuestos

Este módulo se centró en el manejo de arrays y objetos en PHP. Los datos compuestos son esenciales para gestionar colecciones de datos, como las listas de recetas y los detalles de los usuarios. Se aprendió a manipular estos datos de manera eficiente, lo cual es crucial para el funcionamiento de la aplicación.

2.2.5 Formularios

Se aprendió a crear y procesar formularios HTML utilizando PHP. Este módulo incluyó la validación de datos de entrada del usuario y la gestión de formularios de registro, inicio de sesión y búsqueda de recetas. La correcta gestión de formularios es fundamental para la interacción del usuario con la aplicación.

2.2.6 Directorios

En este módulo, se estudiaron las funciones de PHP para manejar directorios y archivos. Esto incluye la creación, lectura, escritura y eliminación de archivos, así como la navegación por el sistema de archivos del servidor. Esta habilidad es útil para gestionar imágenes de recetas y archivos de configuración.

2.2.7 Sesiones

Se profundizó en el manejo de sesiones en PHP, una característica esencial para mantener el estado del usuario en la aplicación. Se aprendió a iniciar, gestionar y destruir sesiones, lo que es crucial para funcionalidades como el inicio de sesión, la gestión de favoritos y el mantenimiento de preferencias del usuario.

2.2.8 Integración con Fetch y JSON

Finalmente, se exploró cómo integrar PHP con Fetch y JSON para realizar solicitudes HTTP y manejar datos asíncronos. Este módulo permitió conectar el frontend con el backend de manera eficiente. Se utilizaron endpoints en la carpeta api para manejar solicitudes AJAX, como agregar recetas a favoritos o procesar registros de usuario. Por ejemplo, el endpoint para agregar recetas a favoritos maneja la solicitud del cliente, interactúa con la base de datos y retorna una respuesta en formato JSON.

2.3 Diseño de Interfaces Web

En el desarrollo del proyecto de recetas, se han implementado varios módulos formativos relacionados con el diseño de interfaces web. Estos módulos han permitido realizar una investigación exhaustiva de la competencia y desarrollar una interfaz de usuario atractiva y funcional. A continuación, se presenta una breve explicación de cada módulo.

2.3.1 Investigación de la Competencia

En este módulo, se llevó a cabo una investigación detallada de la competencia, analizando aplicaciones como Uber Eats, Glovo y Just Eat. Además, se examinaron funcionalidades específicas de otras aplicaciones como MyFitnessPal y Ekilu. Este análisis permitió identificar las mejores prácticas y funcionalidades clave que podrían ser incorporadas en nuestro proyecto para mejorar la experiencia del usuario.

2.3.2 SiteMap

Se desarrolló un SiteMap para la aplicación de recetas, que es una representación visual de la estructura del sitio web. Este mapa del sitio facilitó la planificación de la navegación y la organización de las diferentes secciones y páginas, asegurando que la interfaz fuera intuitiva y fácil de usar para los usuarios.

2.3.3 Wireframes y Mockups

En este módulo, se crearon wireframes y mockups de todas las vistas que tendrá el usuario utilizando la herramienta Figma. Los wireframes proporcionaron una estructura básica de las páginas, mientras que los mockups ofrecieron una representación más detallada y estilizada de la interfaz. Este proceso fue crucial para visualizar y refinar el diseño antes de la implementación.

2.3.4 Kit de Componentes

Se diseñó un kit de componentes que incluye todos los elementos necesarios para la interfaz de usuario, como botones, colores, logotipos y otros elementos visuales. Este kit de componentes aseguró la coherencia visual en toda la aplicación y facilitó el proceso de desarrollo al proporcionar un conjunto estándar de recursos de diseño.

3. HERRAMIENTAS Y LENGUAJES UTILIZADOS

El desarrollo del proyecto ha involucrado una cuidadosa selección de herramientas y tecnologías, con el objetivo de garantizar un entorno de desarrollo eficiente y una experiencia óptima para los usuarios finales. A continuación, se detallan las principales herramientas y lenguajes utilizados en la implementación de este proyecto:

3.1 Entorno de Desarrollo Integrado (IDE): Visual Studio Code (VS Code)

VisualStudio Code ha sido elegido como el IDE principal debido a su interfaz intuitiva, extensibilidad, y soporte robusto para una variedad de lenguajes de programación.

3.2 Navegador Web: Google Chrome

Google Chrome se ha utilizado como navegador principal para el desarrollo y pruebas, aprovechando sus herramientas de desarrollo integradas para depuración y optimización de la interfaz de usuario.

3.3 Control de Versiones: Git y GitHub

La combinación de Git para el control de versiones y GitHub como plataforma de alojamiento ha proporcionado una estructura organizada para el trabajo colaborativo, seguimiento de cambios y gestión eficiente del código fuente.

3.4 Gestión de Base de Datos: PHPMyAdmin

Justificación: PHPMyAdmin ha sido la herramienta elegida para administrar la base de datos MySQL de manera visual, facilitando la creación, modificación y gestión de tablas y datos.

3.5 Lenguaje de Programación del Lado del Servidor: PHP con MVC

PHP se ha utilizado para el desarrollo del lado del servidor debido a su versatilidad, amplia adopción en el desarrollo web y capacidades para interactuar con bases de datos. Y se ha utilizado una estructura Modelo Vista Controlador para gestionar todos los recursos de la aplicación.

3.6 Lenguajes de Marcado y Estilos Frontend: HTML, CSS

HTML y CSS son los pilares fundamentales para la estructura y diseño de la interfaz de usuario, garantizando una presentación coherente y atractiva de la información.

3.7 Lenguaje de Programación del Lado del Cliente: JavaScript (JS) y jQuery

JavaScript y jQuery se han empleado para mejorar la interactividad y dinamismo en el lado del cliente, proporcionando una experiencia de usuario fluida y enriquecida.

3.8 Base de Datos: Se ha utilizado MySQL para la gestión de la base de datos

MySQL es una opción robusta y ampliamente utilizada que ofrece eficiencia en la manipulación de datos y es compatible con el desarrollo web.

3.9 API Spoonacular

En esta aplicación se utilizaron los datos de esta API ya que ofrece el rendimiento y los datos necesarios para llevar este proyecto adelante. Ofrece información sobre recetas, ingredientes, valoraciones, etc…

4. COMPONENTES DEL EQUIPO Y APORTACIONES

En el desarrollo de **Recipe Realm**, cada miembro del equipo ha tenido un rol crucial y ha contribuido con la creación de funcionalidades específicas y el diseño de varias pantallas de la aplicación. A continuación, se detalla la participación y las aportaciones realizadas por cada integrante del equipo:

4.1 Aris

Aris ha sido responsable de desarrollar y diseñar las siguientes funcionalidades y pantallas clave:

- **Home**: Aris ha creado la página de inicio de la aplicación, que ofrece una visión general de las funcionalidades principales y facilita la navegación para los usuarios. Esta pantalla es crucial ya que representa la primera impresión de la aplicación y establece la estructura y el acceso a las diferentes secciones.
- **Recipe Discovery**: Aris desarrolló esta funcionalidad para permitir a los usuarios explorar nuevas recetas basadas en sus preferencias y los ingredientes disponibles en su inventario. La pantalla ofrece opciones de búsqueda avanzada y filtrado para personalizar la experiencia de descubrimiento de recetas.
- **Recipe Details**: Esta pantalla muestra la información detallada de una receta seleccionada, incluyendo ingredientes, instrucciones, valor nutricional y más. Aris se encargó de diseñar una interfaz clara y atractiva que presenta toda la información relevante de manera accesible y organizada.

4.2 Juan

Juan ha desempeñado un papel fundamental en la creación de funcionalidades relacionadas con la gestión de usuarios y la base de datos:

- **Register**: Juan desarrolló la pantalla de registro de usuarios, donde los nuevos usuarios pueden crear una cuenta ingresando su correo electrónico, nombre de usuario y contraseña. Esta pantalla incluye validaciones para asegurar la integridad de los datos y mejorar la seguridad del proceso de registro.
- **Login**: Esta pantalla permite a los usuarios existentes iniciar sesión en la aplicación utilizando su correo electrónico o nombre de usuario y contraseña. Juan implementó

la lógica de autenticación y manejo de sesiones para asegurar un acceso seguro y eficiente.

- **My Recipes (My Kitchen)**: Juan fue responsable de la pantalla de "Mis Recetas" o "Mi Cocina", que permite a los usuarios ver y gestionar sus recetas guardadas y creadas. Esta pantalla facilita la organización y acceso a las recetas personales de los usuarios.
- **Base de datos (Creación y conexión)**: Juan también fue responsable de la creación y gestión de la base de datos, incluyendo la definición de las tablas y sus relaciones, así como la conexión de la base de datos con la aplicación. Este rol es fundamental para asegurar la integridad y accesibilidad de los datos.

4.3 Dani

Dani ha trabajado en la integración de funcionalidades relacionadas con las compras y la interfaz de usuario:

- **Shopping Cart**: Dani diseñó la funcionalidad del carrito de compras, que permite a los usuarios agregar y gestionar los ingredientes que desean comprar. Esta pantalla facilita el proceso de compra y se integra con otras funcionalidades de la aplicación.
- **Shopping List**: Esta pantalla permite a los usuarios generar listas de compras basadas en las recetas seleccionadas y los ingredientes necesarios. Dani implementó una interfaz que facilita la creación y gestión de listas de compras, mejorando la experiencia de planificación de comidas.
- **Payment Gateway (Stripe)**: Dani se encargó de integrar la pasarela de pago Stripe, que permite a los usuarios realizar compras de ingredientes directamente desde la aplicación. Esta funcionalidad asegura un proceso de pago seguro y eficiente.
- **Food Market**: Dani desarrolló la pantalla del mercado de alimentos, donde los usuarios pueden explorar y comprar ingredientes disponibles en línea. Esta funcionalidad se integra con la lista de compras y el carrito de compras, proporcionando una experiencia de compra completa.
- **Header**: Dani también fue responsable del diseño del encabezado de la aplicación, que incluye la navegación principal y facilita el acceso a las diferentes secciones de la aplicación.

- **Footer**: Finalmente, Dani diseñó el pie de página de la aplicación, que incluye enlaces importantes y facilita la navegación y la accesibilidad a la información y funcionalidades clave.

5. FASES DEL PROYECTO

5.1. Estudio de Mercado

El mercado de las aplicaciones de gestión de recetas y alimentos ha experimentado un crecimiento significativo en los últimos años, impulsado por la tendencia creciente hacia la digitalización y la búsqueda de soluciones que faciliten la vida cotidiana. Las aplicaciones que permiten gestionar recetas, planificar comidas y realizar compras en línea han ganado popularidad, especialmente entre aquellos que buscan optimizar su tiempo y reducir el desperdicio de alimentos.

5.1.1 Análisis de la Demanda:

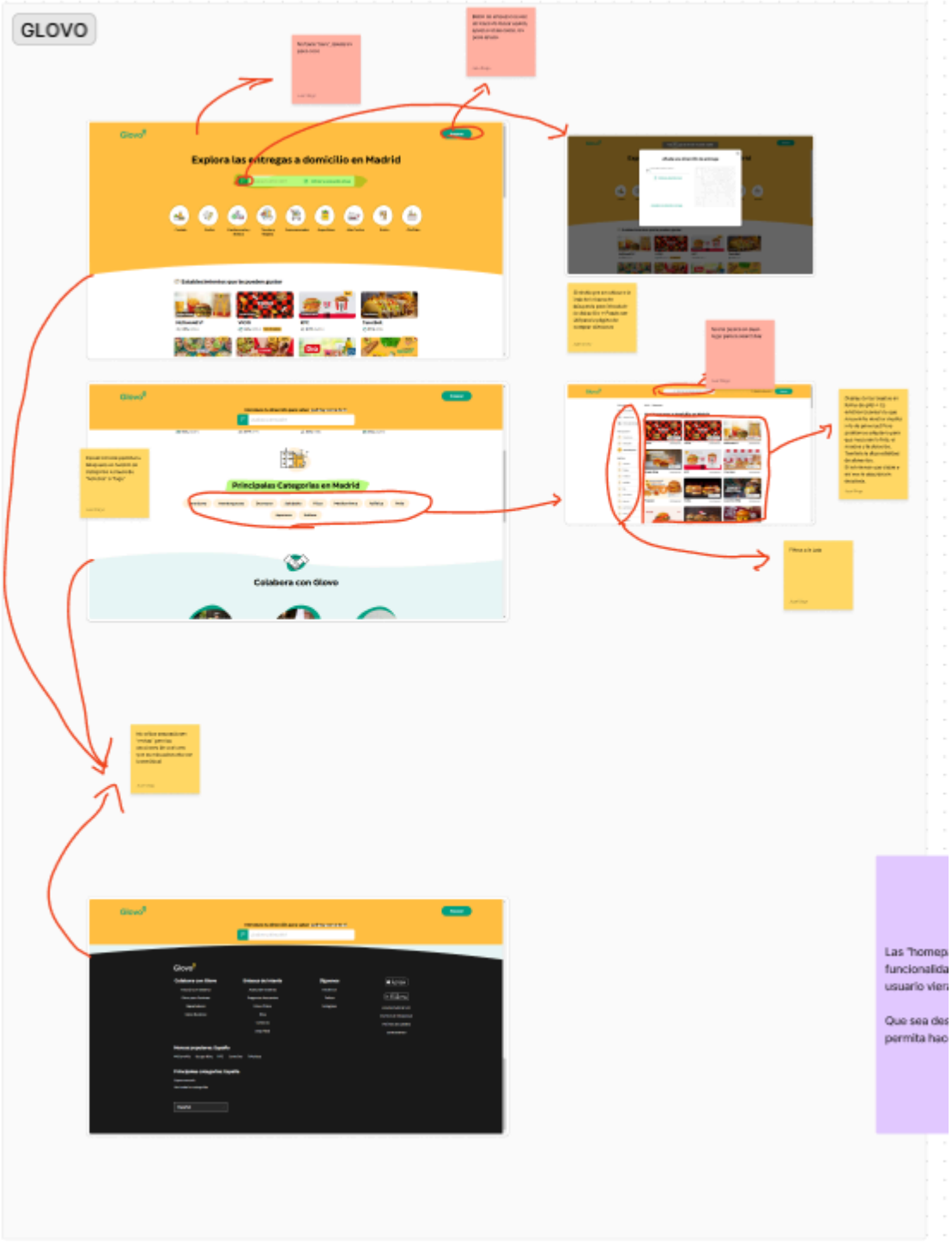
La demanda de aplicaciones de gestión de recetas está impulsada por varios factores, entre ellos la creciente conciencia sobre la alimentación saludable, la necesidad de ahorrar tiempo en la planificación de comidas y la preferencia por soluciones digitales que simplifiquen las tareas diarias.

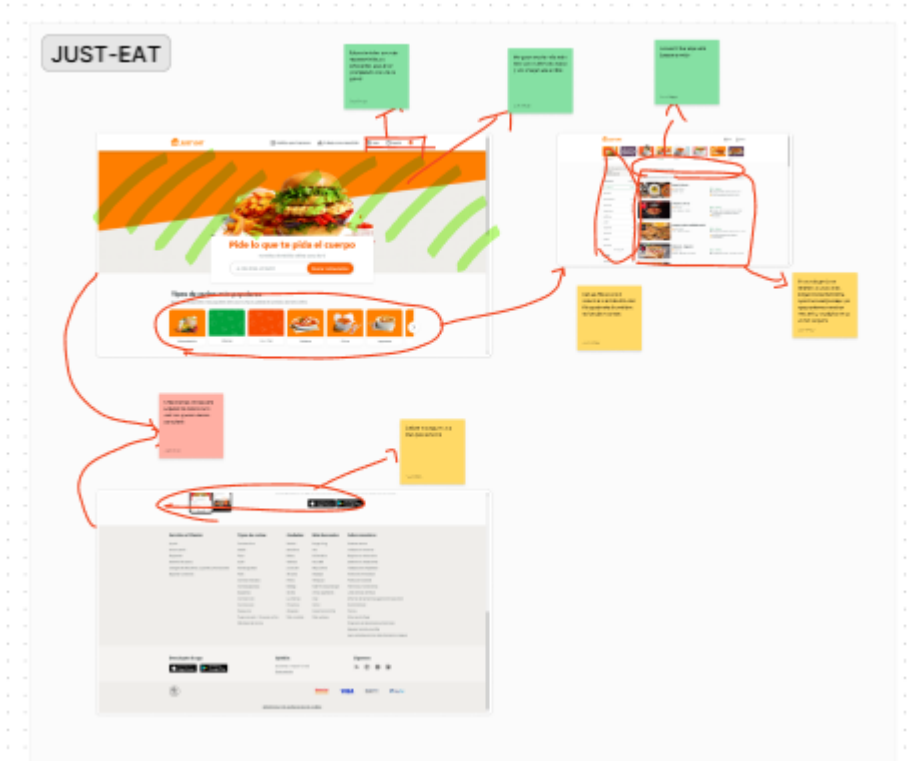
5.1.2 Competencia:

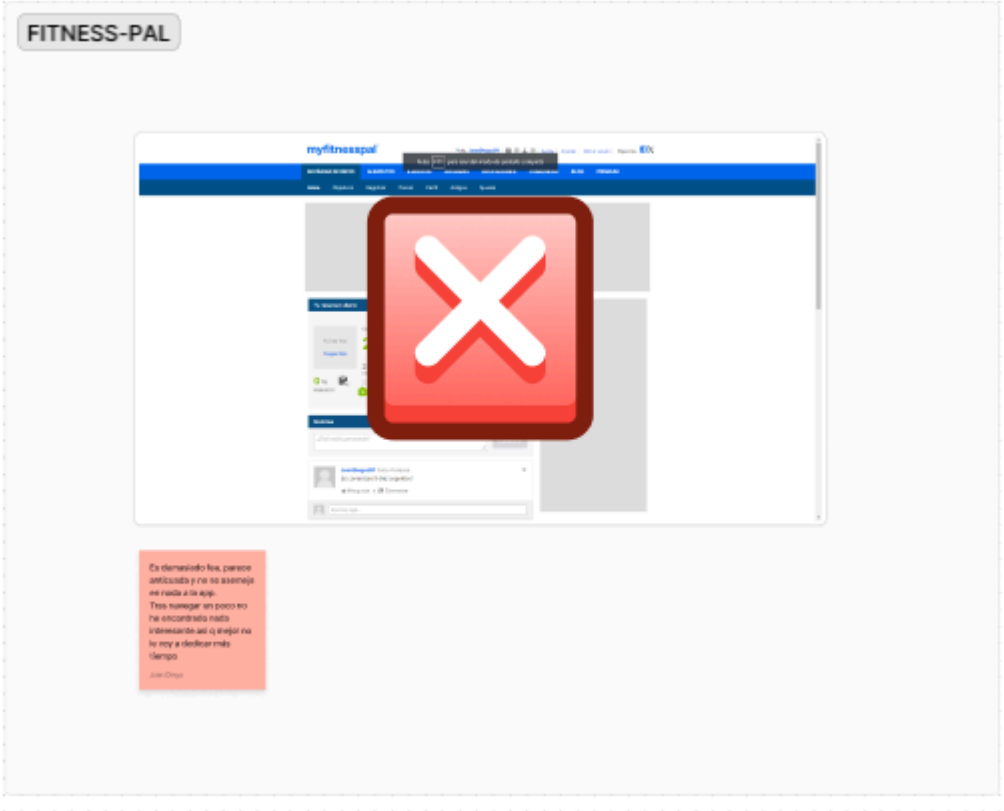
Existen varias aplicaciones en el mercado que ofrecen funcionalidades similares a Recipe, a la hora de realizar el análisis de la competencia, nos hemos centrado en aplicaciones de comida como Glovo y JustEat. También hemos tomado en cuenta aplicaciones de conteo de calorías como MyFitnesspal y aplicaciones de recetas como Ekilu app.

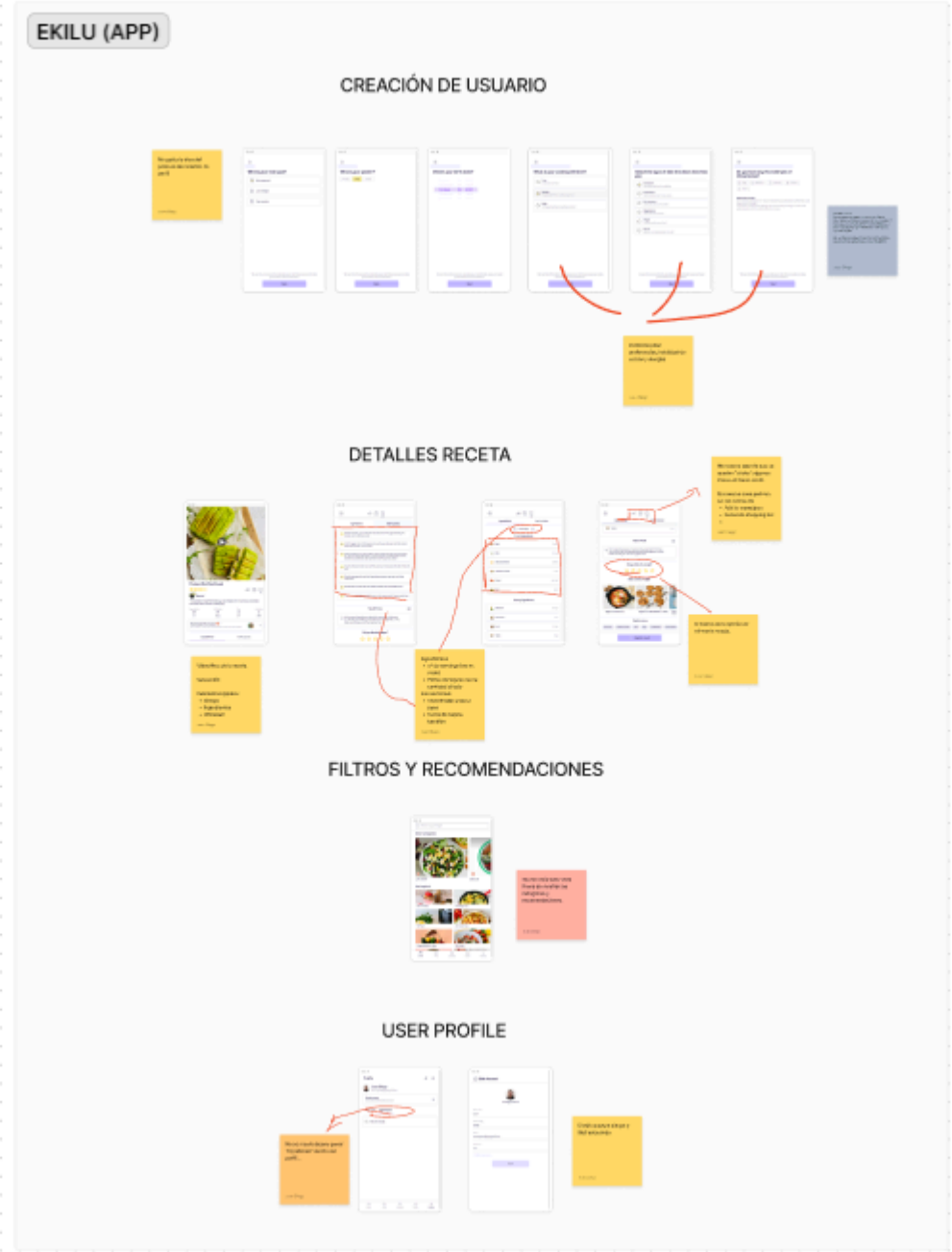
Adjuntamos el panel figma utilizado para el análisis de la competencia:

https://www.figma.com/board/bJ10y3tNV5QzfvCQhHPaLo/FigJam-Board?node-id=0-1&t=R7weU8DcCp5vg72f-1


GLOVO
Explora las entregas a domicilio en Madrid
Principales Categorías en Madrid
Colabora con Glovo


JUST-EAT
Pide lo que te pida el cuerpo


FITNESS-PAL
myfitnesspal


EKILU (APP)
CREACIÓN DE USUARIO
DETALLES RECETA
FILTROS Y RECOMENDACIONES
USER PROFILE

5.1.3 Mejoras Propuestas

A diferencia de la competencia, Recipe Realm no solo ofrece recomendaciones de recetas, sino que también permite a los usuarios gestionar su inventario de alimentos, crear recetas personalizadas y realizar compras directamente desde la aplicación.

También incorpora filtros mucho más detallados que permiten a los usuarios encontrar recetas que se ajusten a sus preferencias dietéticas y necesidades nutricionales.

En términos de diseño, Recipe Realm destaca con una estética minimalista y un diseño poco cargado que se integra en todas las pantallas de la aplicación causando una sinergia que el usuario puede experimentar mientras usa la aplicación.
Otras aplicaciones de recetas, en cambio, suelen tener diseños mucho más cargados y menos consistentes a lo largo de su flujo de navegación.

5.2 Análisis de la Aplicación

A continuación, se detallan los requisitos y funcionalidades clave de la aplicación:

5.2.1 Requisitos Funcionales:

- Recomendación de recetas:
    - Sugerir recetas basadas en ingredientes disponibles.
    - Ofrecer recomendaciones personalizadas según preferencias dietéticas.
- Creación y gestión de recetas:
    - Crear y editar recetas propias.
    - Publicar recetas o mantenerlas privadas.
    - Buscar recetas creadas y guardadas.
- Compra en línea:
    - Integrar con servicios de compra para adquirir ingredientes.
    - Facilitar la adición de artículos al carrito y el pago.
- Gestión de inventario de alimentos:
    - Agregar, editar y eliminar ingredientes.
    - Buscar ingredientes dentro del inventario.

5.2.2 Requisitos No Funcionales:

- Usabilidad:
    - Interfaz intuitiva y fácil de usar.
    - Compatibilidad con desktop, tablet y smartphone.
- Rendimiento:
    - Búsqueda y procesamiento de datos rápida y eficiente.
    - Capacidad para manejar grandes volúmenes de datos sin pérdida de rendimiento.
- Seguridad:
    - Protección de la información del usuario, incluyendo datos personales como contraseñas.
    - Uso de sentencias preparadas para evitar la inyección de código
- Mantenimiento y Escalabilidad:
    - Estructura de la aplicación que facilite el mantenimiento y las actualizaciones.
    - Capacidad de escalar para soportar un aumento en el número de usuarios y datos.

5.2.3 Funcionalidades Clave:

Interfaz de Usuario Intuitiva: Diseñada para facilitar la navegación y el acceso a todas las funcionalidades de la aplicación.

Gestión Integral de Recetas e Ingredientes: Combina la recomendación y creación de recetas con la gestión del inventario de alimentos.

Integración de Servicios de Compra en Línea: Permite a los usuarios adquirir platos saludables sin salir de la aplicación.

5.3 Modelo de Datos

La base de datos de Recipe Realm se ha diseñado para gestionar de manera eficiente los datos relacionados con usuarios y recetas. Empleamos un enfoque híbrido, combinando una base de datos relacional para la gestión de usuarios y datos locales, y una API externa, Spoonacular API, para obtener y gestionar recetas. Este enfoque permite mantener un equilibrio entre la gestión interna de datos y el acceso a una amplia base de datos externa de recetas.

5.3.1 Diseño de la Base de Datos

Para garantizar una experiencia de usuario sin fricciones, la base de datos se ha estructurado con un enfoque en la simplicidad y eficiencia:

Gestión de Registro y Login de Usuarios:
La base de datos incluye tablas específicas para manejar el registro y la autenticación de los usuarios. Para minimizar la fricción, los únicos campos necesarios para el registro y el inicio de sesión son el correo electrónico, el nombre de usuario y la contraseña.

Perfiles de Usuario:
La información adicional sobre los usuarios, como preferencias dietéticas y alergias, se almacena en una tabla de perfiles (USERS_PROFILES). Esta tabla incluye campos opcionales que permiten personalizar la experiencia del usuario en la aplicación.

Gestión de Recetas:
La funcionalidad principal de la aplicación se gestiona a través de dos tablas:

- USER_RECIPES: Esta tabla relaciona cada usuario con las recetas que han guardado, obtenidas de la Spoonacular API.
- USER_MYRECIPES: Esta tabla almacena las recetas creadas por los usuarios dentro de la aplicación, permitiendo una gestión completa de sus propias creaciones culinarias.

5.3.2 Tablas y Relaciones

- USERS
    - id: Identificador único del usuario.
    - email: Correo electrónico, único.
    - username: Nombre de usuario, único.
    - name: Nombre del usuario.
    - surname: Apellido del usuario.

- USERS_SECURITY
    - user_id: Identificador del usuario, FK de USERS.
    - pwd: Contraseña encriptada del usuario.

- USERS_PROFILES
    - user_id: Identificador del usuario, FK de USERS.
    - photo: URL de la foto del perfil.
    - gender: Género del usuario.
    - birth_date: Fecha de nacimiento.
    - diet_type: Tipo de dieta.
    - food_allergies: Alergias alimentarias.
    - other_allergies: Otras alergias.
    - height: Altura del usuario.
    - weight: Peso del usuario.
    - activity: Nivel de actividad física.
    - estimated_calories: Calorías estimadas diarias.
    - goal: Objetivo de salud o fitness.

- USER_RECIPES
    -
    - id: Identificador único de la relación.
    - user_id: Identificador del usuario, FK de USERS.
    - api_id: Identificador de la receta en la API de Spoonacular.

- USER_MYRECIPES
    - recipe_id: Identificador único de la receta.
    - user_id: Identificador del usuario, FK de USERS.
    - name: Nombre de la receta.
    - description: Descripción de la receta.
    - instructions: Instrucciones de preparación.
    - ingredients: Ingredientes utilizados.
    - allergens: Alérgenos presentes.
    - photo: URL de la foto de la receta.
    - prep_time: Tiempo de preparación.
    - calories: Calorías totales.
    - serves: Porciones que rinde la receta.

Relaciones entre las Tablas:

Relación Usuarios-Perfiles: La tabla USERS se relaciona con USERS_PROFILES a través del campo user_id, formando una relación uno a uno. Cada usuario tiene un perfil asociado con información adicional que mejora la experiencia personalizada.

Relación Usuarios-Recetas Guardadas: La tabla USER_RECIPES relaciona a cada usuario con las recetas que ha guardado, obtenidas de la Spoonacular API.

Relación Usuarios-Recetas Propias: La tabla USER_MYRECIPES permite a cada usuario gestionar sus propias recetas creadas dentro de la aplicación, facilitando la organización y acceso a sus creaciones culinarias.

5.3.3 Spoonacular API

Utilizamos Spoonacular API como una pieza clave para gestionar y proporcionar acceso a una vasta base de datos de recetas y alimentos. Esta API permite a la aplicación extender sus capacidades más allá de los datos locales almacenados en la base de datos interna, ofreciendo

a los usuarios una experiencia enriquecida y diversificada en la búsqueda y gestión de recetas.

Papel de la API en la Aplicación

1. Acceso a una Amplia Base de Datos de Recetas:

La API proporciona acceso a más de 380,000 recetas, lo que permite a los usuarios explorar una variedad de opciones culinarias sin límites.

2. Gestión de Ingredientes y Alimentos:

Con la Spoonacular API, la aplicación puede obtener información detallada sobre miles de ingredientes y productos alimenticios. Esto incluye datos nutricionales, sustitutos de ingredientes y detalles sobre alérgenos, lo que facilita la personalización de la experiencia del usuario según sus necesidades y restricciones dietéticas.

4. Búsqueda y Filtrado Avanzado de Recetas:

La API soporta búsquedas avanzadas y filtrados por múltiples criterios como contenido nutricional, tipo de dieta, y preferencias personales. Los usuarios pueden buscar recetas que cumplan con ciertos requisitos dietéticos, como bajo contenido en carbohidratos o alto en proteínas, permitiendo una planificación de comidas más eficiente y adaptada.

Para más detalles sobre las capacidades de la Spoonacular API, adjuntamos su documentación: (https://spoonacular.com/food-api).

5.4 Diseño

En el tema de diseño, se ha trabajado principalmente con el programa Figma. Este software de diseño colaborativo ha sido fundamental para la creación y perfeccionamiento de la interfaz de usuario del proyecto de recetas. En Figma, se han llevado a cabo diversas actividades esenciales, incluyendo la investigación de la competencia, la elaboración de wireframes y mockups, y el desarrollo de un kit de componentes. Estas tareas han permitido definir y visualizar cada aspecto del diseño de la aplicación, asegurando una experiencia de usuario coherente y atractiva. A continuación, se detallan los componentes clave del proceso de diseño. Podemos separar esta sección en distintas categorías:

5.4.1 Paleta de colores

Color verde (#71A463)

El color #71A463 es un tono de verde fresco y natural. Este color también ofrece varias ventajas para su uso en un sitio web de recetas:

1. **Asociación con Naturaleza y Salud**:
    - Los tonos verdes están fuertemente asociados con la naturaleza, la frescura y la salud. Esto es muy adecuado para un sitio de recetas que promueve la alimentación saludable y el uso de ingredientes frescos.
2. **Calma y Relajación**:
    - El verde es un color que transmite calma y relajación, lo cual puede mejorar la experiencia del usuario al navegar por la página, haciendo que se sienta tranquilo y cómodo.
3. **Equilibrio Visual**:
    - Este tono de verde proporciona un equilibrio visual agradable cuando se combina con colores oscuros como #242E3B. La combinación de colores oscuros y verdes claros puede crear un diseño visualmente atractivo y armonioso.

Color Azul Oscuro(#242E3B)

El color #242E3B es un tono oscuro que se encuentra dentro de la gama de los azules oscuros o grises azulados. Este color tiene varias ventajas y beneficios para su uso en páginas web, especialmente en un proyecto de recetas:

1. **Elegancia y Profesionalidad**:

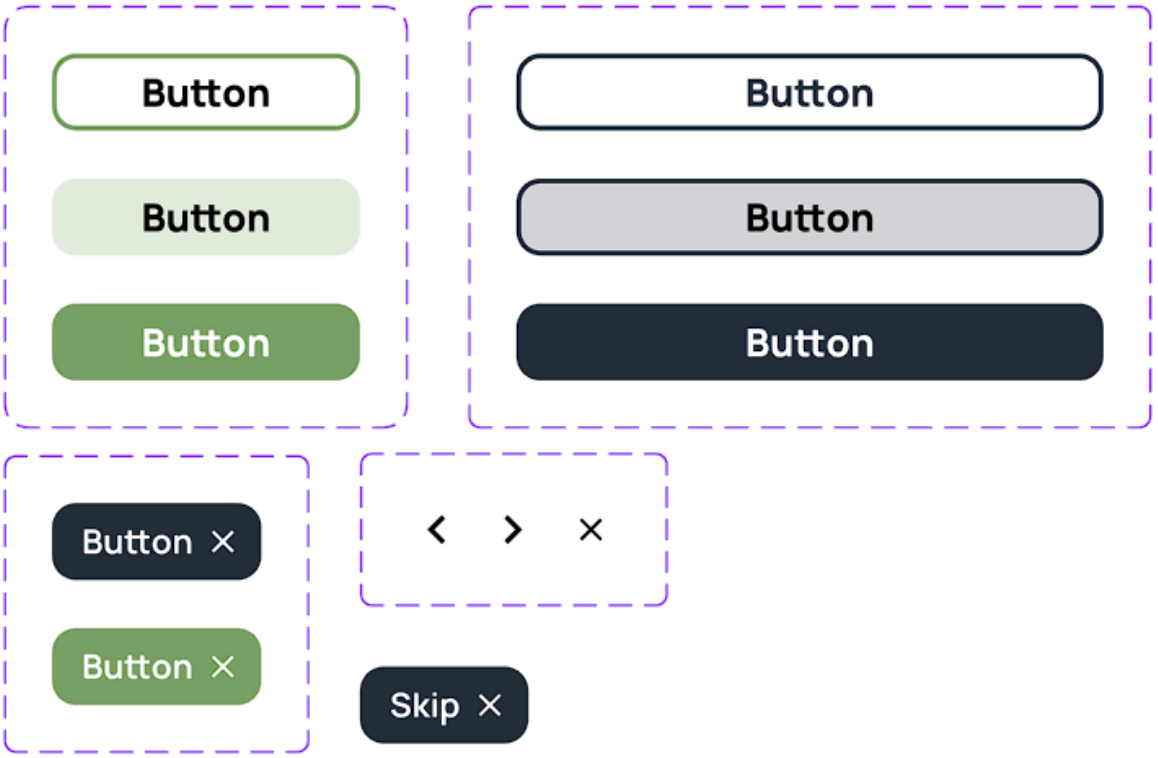
- Los tonos oscuros como el #242E3B transmiten una sensación de elegancia y profesionalidad. Esto es especialmente importante para páginas que desean proyectar una imagen sofisticada y confiable.

2. **Contraste y Legibilidad**:
   - Este color oscuro proporciona un excelente contraste con colores claros, especialmente con el texto blanco (#FFFFFF) o tonos claros. Esto mejora la legibilidad y hace que el contenido sea más fácil de leer para los usuarios.
3. **Reducción de Fatiga Visual**:
   - Los colores oscuros son menos fatigantes para la vista en condiciones de poca luz o para sesiones de navegación prolongadas. Esto puede hacer que la experiencia del usuario sea más cómoda y agradable.

5.4.2 Kit de componentes

1. **Botones**

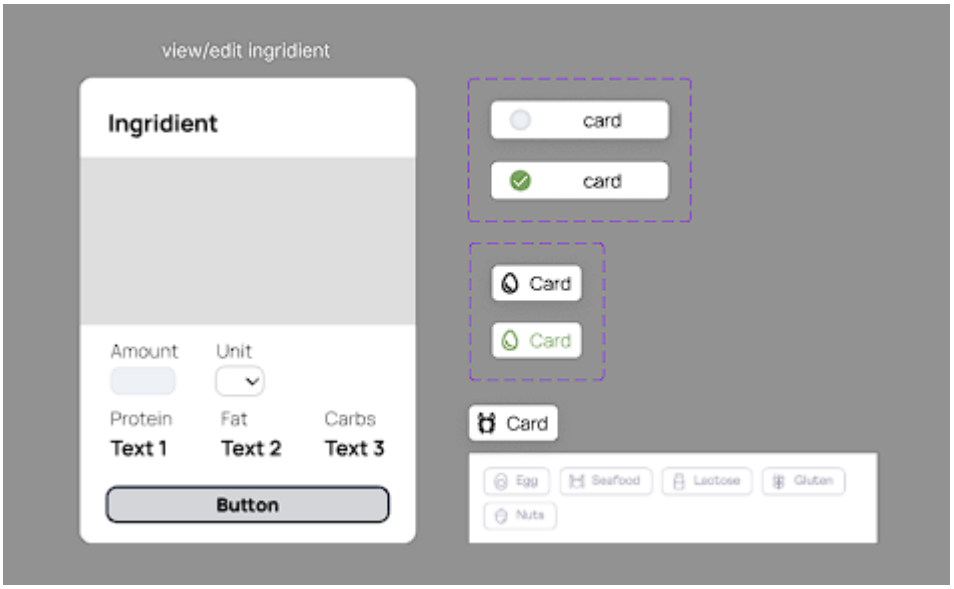
El kit de componentes incluye diversos botones estilizados según la paleta de colores #242E3B y #71A463, diseñados para mantener la coherencia visual y mejorar la experiencia del usuario. Los botones están categorizados para diferentes funciones como acciones primarias, secundarias, cancelación y navegación, asegurando claridad y efectividad en cada interacción del usuario.



2. **Tarjetas**

El kit de componentes incluye tarjetas diseñadas para mostrar y editar información, utilizando los colores #242E3B y #71A463. Las tarjetas contienen elementos

interactivos como botones, iconos y etiquetas, proporcionando una experiencia de usuario clara e intuitiva.



3. **DropDown/Select**

El kit de componentes incluye menús desplegables (DropDown/Select) diseñados para facilitar la selección de opciones por parte del usuario. Estos componentes permiten una interacción intuitiva y eficiente, asegurando una experiencia de usuario clara y coherente.

4. **Iconos**

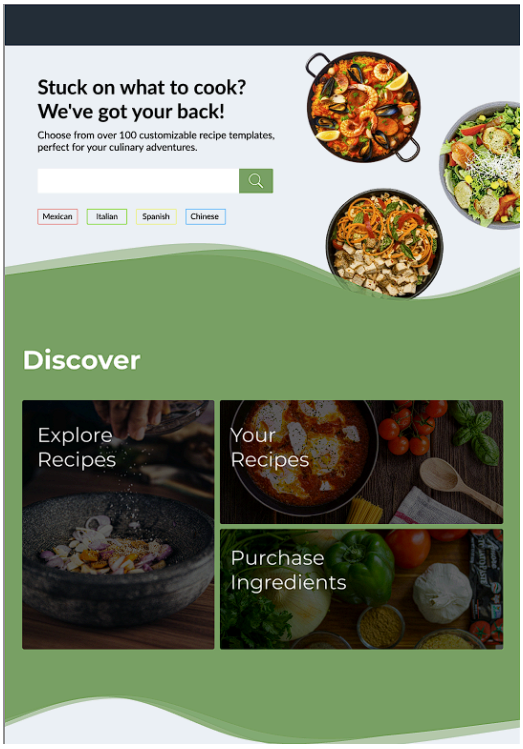
El kit de componentes incluye una serie de iconos diseñados para representar diversas funciones y categorías dentro de la aplicación. Estos iconos son intuitivos y fáciles de entender, facilitando la navegación y mejorando la experiencia del usuario al proporcionar señales visuales claras y reconocibles. Los iconos abarcan categorías como alergias y perfil, ayudando a los usuarios a identificar rápidamente secciones y acciones importantes en la aplicación.

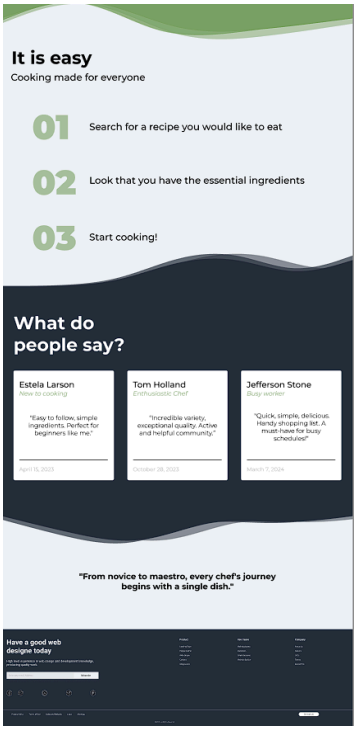
5.4.3 Mockups

En esta categoría se muestran los distintos diseños de las distintas vistas que se han hecho para el proyecto. Las imágenes vendrán acompañadas de una breve explicación de qué es y las acciones que se pueden hacer en las vistas.

1. **Home**

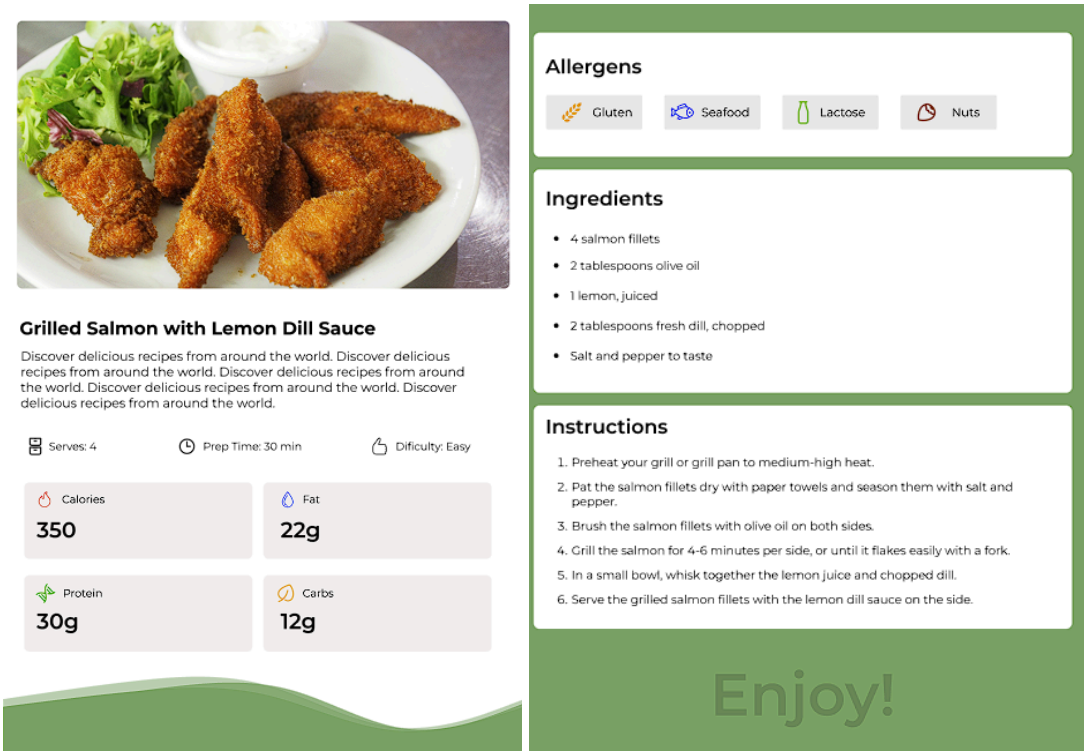
   La home como el propio nombre indica es la landing page en donde el usuario aterriza en primer lugar. Aquí se ofrece un poco de información de que es lo que hacemos y accesos directos a otras páginas de interés para los usuarios.





2. Recipe-Discovery

En Recipe discovery el usuario puede buscar por palabras clave o por filtros recetas. En cada receta pone información relevante para el usuario como las alergias, tiempo, porciones… Al dar llevaría al usuarios recipe que veremos más adelante.

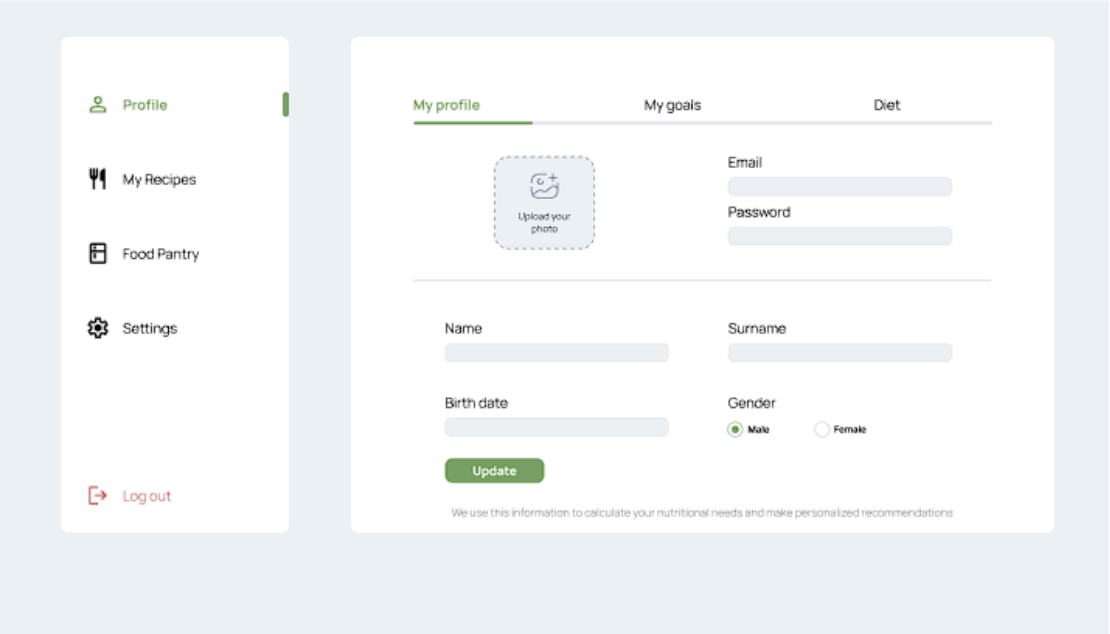


3. Register

En registro como el propio nombre indica es en donde el usuario se podrá registrar. En este caso también se añaden datos de alergias, calorías entre otros datos. El proceso de registro está dividido en 4 pasos que se muestran en la imagen.

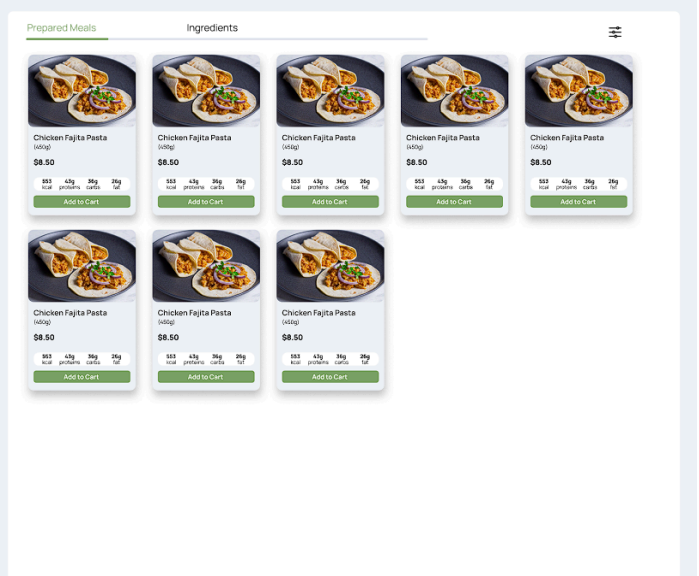
4. Profile

En profile se encuentran todos los datos del usuario logueado. Se encontrará información sobre los nombres, apellidos, imagen, etc… Hay más secciones que se llaman My Recipes, Settings y Logout. En la sección de My Recipes se encuentran las recetas favoritas, creadas por el usuario y la opción de crear una. La página de settings gestiona algunos cambios de la configuración del propio usuario. Y el logout cierra la sesión del usuario.



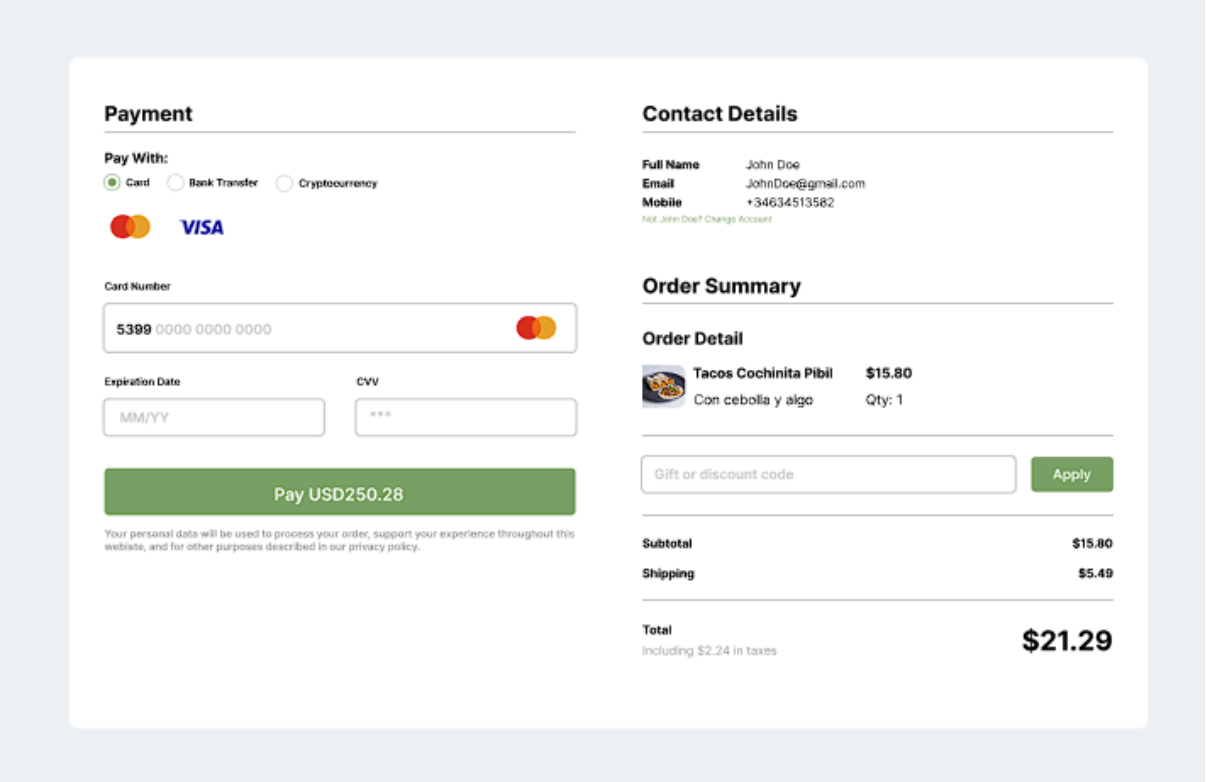
5. Food market

Es la página en donde se ofrecen platos ya hechos para pedir a domicilio. Se pueden agregar y eliminar platos de la cesta. También muestra información relevante sobre cada plato.



6. **Pasarela de pago**

Esta página representa la pasarela de pago para comprar los platos con la tarjeta de crédito.



5.5 Planificación del desarrollo

5.5.1 Diagrama de Gantt

En el proyecto de recetas, se ha utilizado el diagrama de Gantt como una herramienta fundamental para organizar y gestionar todo el trabajo. Este tipo de diagrama ha permitido planificar, coordinar y seguir el progreso de las diversas tareas y actividades del proyecto de manera eficiente y visualmente clara.

Organización y Planificación

El equipo de desarrollo ha empleado el diagrama de Gantt para detallar el cronograma completo del proyecto. Cada tarea, desde la investigación inicial hasta el lanzamiento final de la aplicación, ha sido representada mediante barras horizontales que muestran su duración y sus fechas de inicio y finalización. Este nivel de detalle ha permitido una planificación

meticulosa, asegurando que cada fase del proyecto esté claramente definida y temporalmente alineada con el resto del trabajo.

Coordinación de Tareas

Gracias al diagrama de Gantt, el equipo ha podido coordinar de manera efectiva las tareas y actividades. Las dependencias entre tareas, como la necesidad de completar el diseño antes de comenzar con el desarrollo, se han indicado con líneas y flechas, facilitando la gestión de las interdependencias. Esto ha ayudado a evitar retrasos y a asegurar que cada parte del proyecto avance según lo previsto.

Seguimiento del Progreso

El diagrama de Gantt ha sido una herramienta invaluable para el seguimiento del progreso del proyecto. El equipo ha podido actualizar el diagrama regularmente, comparando el progreso real con el planificado. Esto ha permitido identificar rápidamente cualquier desviación del cronograma y tomar medidas correctivas oportunas, manteniendo el proyecto en el buen camino.

Comunicación y Transparencia

El uso del diagrama de Gantt también ha mejorado la comunicación y la transparencia dentro del equipo y con las partes interesadas. Al proporcionar una representación visual del cronograma del proyecto, todos los miembros del equipo han tenido una comprensión clara de sus responsabilidades y del estado actual del proyecto. Esto ha facilitado una colaboración más efectiva y ha asegurado que todos estén alineados con los objetivos y plazos del proyecto.

5.5.2 Uso de herramientas de planificación

En el proyecto de recetas, se han utilizado diversas herramientas de planificación para organizar y gestionar eficientemente todas las tareas y actividades. Las principales herramientas empleadas fueron

Trello, GitHub y Discord, las cuales facilitaron la colaboración, el seguimiento del progreso y la comunicación del equipo.

Trello

Trello fue utilizado como la principal herramienta de gestión de tareas. Con su interfaz basada en tableros y tarjetas, Trello permitió al equipo visualizar y organizar todas las tareas del proyecto de manera clara y estructurada. Cada tarea o actividad se representó con una tarjeta en el tablero, categorizada por fases del proyecto (por ejemplo, "Investigación", "Desarrollo", "Pruebas"). Las tarjetas incluían detalles específicos, como descripciones, fechas de vencimiento, listas de verificación y asignaciones de miembros del equipo. Esta herramienta facilitó la asignación de responsabilidades, el seguimiento del progreso y la priorización de tareas.

GitHub

GitHub se utilizó para la gestión del código y la colaboración en el desarrollo del proyecto. A través de GitHub, el equipo pudo gestionar versiones del código fuente, realizar revisiones de código y fusionar cambios de manera eficiente. Los "issues" y "pull requests" de GitHub permitieron a los desarrolladores reportar y solucionar problemas, así como discutir y revisar cambios antes de integrarlos en la base de código principal. Esta plataforma aseguró que todos los miembros del equipo trabajaran con la versión más actualizada del código y que cualquier conflicto o error se resolviera rápidamente.

Discord

Discord fue la herramienta elegida para la comunicación y las reuniones del equipo. Se realizaron reuniones dos veces a la semana, los miércoles y domingos, para discutir el progreso, establecer pautas y definir metas para la semana siguiente. Estas reuniones permitieron una comunicación fluida y efectiva entre los miembros del equipo, facilitando la resolución de dudas y la toma de decisiones en tiempo real. Además, los canales de texto y voz de Discord sirvieron como un espacio constante para la colaboración y el intercambio de ideas.

6. CONCLUSIONES Y MEJORAS

En conclusión, teniendo en cuenta el tiempo reducido de este proyecto comparado con el integrador, las prácticas y la complejidad del mismo, estamos muy satisfechos con el resultado.

Fuimos capaces de trabajar con una API bastante extensa y compleja como la de Spoonacular, consiguiendo usar los datos requeridos y mostrarlos de forma exitosa.

También el uso de la API de Stripe la cual es usada para realizar pagos en línea. Tras leer cuidadosamente y entender la documentación, fue implementada de forma exitosa y podría funcionar perfectamente en un proyecto real en producción.

Otro de los retos a los que nos hemos enfrentado ha sido el trabajar con una arquitectura MVC (Model View Controller), la cuál decidimos utilizar por lo potente que es, todo más organizado y mejor estructurado. Una de las principales razones también fue debido a que es la arquitectura más usada a nivel empresarial a la hora de construir proyectos reales.

6.1 MEJORAS

Como mejores a futuro / con más tiempo, añadiríamos la página de “Food Pantry” así como una recomendación de recetas en función de los ingredientes que tengas, refactorizar el código para hacerlo más legible y eficiente, incluir una estructura de almacenamiento de datos del usuario para poder hacer análisis de datos de nuestros usuarios y poder tomar decisiones en base a los datos recolectados.

7. ANEXOS

7.1 Código fuente

En esta sección, se presentarán ejemplos de las partes más interesantes y relevantes del código fuente del proyecto de recetas. Estos fragmentos de código destacan las implementaciones clave y las soluciones técnicas utilizadas para desarrollar las funcionalidades principales de la aplicación. A continuación, se detallarán las técnicas y metodologías empleadas, así como las estructuras de datos y algoritmos que forman la base del proyecto. Estos ejemplos ilustran cómo se han abordado los desafíos del desarrollo y cómo se han aplicado las mejores prácticas de programación para crear una aplicación robusta y eficiente.

7.2 Recipe Discovery

Este código JavaScript se ejecuta cuando la página web ha terminado de cargarse. Configura el formulario de búsqueda y los filtros de una aplicación de recetas. Guarda y restaura filtros desde localStorage, actualiza los resultados de búsqueda basados en los filtros aplicados, y muestra iconos de intolerancias alimentarias en las recetas. Además, maneja la interacción con íconos de corazón para marcar recetas como favoritas, y establece eventos de clic para navegar a los detalles de una receta al hacer clic en una tarjeta de receta.

```
function updateResults(data) {
    const resultsDiv = document.getElementById('results');
    resultsDiv.innerHTML = '';

    if (data.results) {
        data.results.forEach(recipe => {
            const recipeDiv = document.createElement('div');
            recipeDiv.classList.add('recipe_card');
            recipeDiv.setAttribute('data-id', recipe.id);

            recipeDiv.innerHTML = `
                <div class='card_image'>
                <img src="${recipe.image}" alt="${recipe.title}">
                </div>
                `;
            resultsDiv.appendChild(recipeDiv);

            recipeDiv.addEventListener('click', (event) => {
                if (!event.target.classList.contains('heart-icon')) {
                    window.location.href = `/recipe?id=${recipe.id}`;
                }
            });

            fetch(`${window.location.origin}/recipe-discovery?id=${recipe.id}`)
                .then(response => response.json())
                .then(details => {
                    let textoLargo = details.summary;
                    let textoCorto = stripHTML(textoLargo).substring(0, 150) + "...";
                    const calories = details.nutrition?.nutrients.find(nutrient => nutrient.name === 'Calories')?.amount || 0;

                    recipeDiv.innerHTML += `
                        <div id='heart_container'><span class="material-icons heart-icon" id="heart-${recipe.id}">favorite_border</span></div>
                        <div class='card_details'>
                            <div class='inner_card_details1'>
                                <h4>${recipe.title}</h4>
                                <p>${textoCorto}</p>
                                <div class="intolerances" id="intolerances-${recipe.id}">
                                </div>
                            </div>
```

```
                        <div class='inner_card_details2'>
                            <div class='box'>
                                <span><i class="material-icons">access_time</i> <br> <span id="readyInMinutes">${details.readyInMinutes}</span> min</span>
                            </div>
                            <div class='box'>
                                <span><i class="material-icons">restaurant</i> <br> <span id="servings">${details.servings}</span> servings</span>
                            </div>
                            <div class='box'>
                                <span><i class="material-icons">flash_on</i> <br> <span id="calories">${calories}</span> kcal</span>
                            </div>
                        </div>
                    </div>
                    `;

                displayIntolerances(details.extendedIngredients, `intolerances-${recipe.id}`);
                heartIconListeners(); // Agregar los listeners de los corazones después de actualizar los detalles

            })
            .catch(error => {
                console.error('Error fetching recipe details:', error);
            });
        });

    } else {
        resultsDiv.innerHTML = '<p>No recipes found.</p>';
    }
}

function displayIntolerances(ingredients, containerId) {
    const intolerancesDiv = document.getElementById(containerId);
    const foundIntolerances = [];

    ingredients.forEach(ingredient => {
        for (const [intolerance, keywords] of Object.entries(intoleranceKeywords)) {
            if (keywords.some(keyword => ingredient.name.toLowerCase().includes(keyword.toLowerCase()))) {
                if (!foundIntolerances.includes(intolerance)) {
                    foundIntolerances.push(intolerance);
                }
            }
        }
    });

    foundIntolerances.forEach(intolerance => {
        if (intoleranceIcons[intolerance]) {
            const intoleranceElement = document.createElement('div');
            intoleranceElement.className = 'intolerance';
            intoleranceElement.innerHTML = `<i class="material-icons">${intoleranceIcons[intolerance]}</i> ${intolerance}`;
            intolerancesDiv.appendChild(intoleranceElement);
        }
    });
}

function handleFilterChange() {
    const query = document.getElementById('query').value;
    const sort = document.querySelector('input[name="sort"]:checked')?.value || '';
    const types = Array.from(document.querySelectorAll('input[name="type"]:checked')).map(el => el.value);
    const cuisines = Array.from(document.querySelectorAll('input[name="cuisine"]:checked')).map(el => el.value);
    const diets = Array.from(document.querySelectorAll('input[name="diet"]:checked')).map(el => el.value);
    const intolerances = Array.from(document.querySelectorAll('input[name="intolerances"]:checked')).map(el => el.value);
    const includeIngredients = Array.from(document.querySelectorAll('input[name="includeIngredients"]:checked')).map(el => el.value);

    const params = new URLSearchParams({
        query,
        sort,
        type: types.join(','),
        cuisine: cuisines.join(','),
        diet: diets.join(','),
        intolerances: intolerances.join(','),
        includeIngredients: includeIngredients.join(',')
    });

    const url = `${window.location.origin}/recipe-discovery?${params.toString()}`;

    fetch(url)
        .then(response => response.json())
        .then(data => {
            updateResults(data);
        })
        .catch(error => {
            console.error('Error:', error);
```

11.1.2 Register

Este código, ejecutado al cargar la página, gestiona un formulario de registro en múltiples pasos. Establece escuchadores de eventos para alternar la visibilidad de contraseñas, seleccionar alergias, validar formularios de cuenta, perfil, dieta y calorías, y calcular calorías estimadas. Además, maneja la navegación entre formularios y actualiza una barra de progreso, asegurando una interacción fluida y validación tanto del lado del cliente como del servidor.

```php
require 'database.php';

$conn = conectarDb();

session_start();
header('Content-Type: application/json');

// Obtener los datos de la solicitud
$data = json_decode(file_get_contents('php://input'), true);
$formId = $data['formId'];

$response = ['success' => false, 'message' => '', 'errorField' => ''];

switch ($formId) {
    case 'account_form':      Dani Garrido, last week via PR #1 • Integración registro de Juan Diego
        $email = $data['email'];
        $username = $data['username'];
        $password = $data['password'];



        // Validar unicidad del email y username
        $email_check = $conn->prepare("SELECT * FROM USERS WHERE email = ?");
        $email_check->bind_param("s", $email);
        $email_check->execute();
        $email_check_result = $email_check->get_result();

        $username_check = $conn->prepare("SELECT * FROM USERS WHERE username = ?");
        $username_check->bind_param("s", $username);
        $username_check->execute();
        $username_check_result = $username_check->get_result();

        if ($email_check_result->num_rows > 0) {
            $response['message'] = 'Email already exists';
            $response['errorField'] = 'email';
            echo json_encode($response);
            exit();
        }

        if ($username_check_result->num_rows > 0) {
            $response['message'] = 'Username already exists';
```

```php
            $response['message'] = 'Email already exists';
            $response['errorField'] = 'email';
            echo json_encode($response);
            exit();
        }

        if ($username_check_result->num_rows > 0) {
            $response['message'] = 'Username already exists';
            $response['errorField'] = 'username';
            echo json_encode($response);
            exit();
        }

        // Iniciar transacción
        $conn->begin_transaction();
        try {
            // Insertar el nuevo usuario en la tabla USERS
            $insert_user = $conn->prepare("INSERT INTO USERS (email, username) VALUES (?, ?)");
            $insert_user->bind_param("ss", $email, $username);
            $insert_user->execute();

            // Obtener el ID del usuario recién insertado
            $user_id = $conn->insert_id;

            // Insertar la contraseña en la tabla USERS_SECURITY
            $password_hashed = password_hash($password, PASSWORD_BCRYPT);
            $insert_security = $conn->prepare("INSERT INTO USERS_SECURITY (user_id, pwd) VALUES (?, ?)");
            $insert_security->bind_param("is", $user_id, $password_hashed);
            $insert_security->execute();

            // Guardar el ID del usuario en la sesión
            $_SESSION['user_id'] = $user_id;

            // Confirmar la transacción
            $conn->commit();

            $response['success'] = true;
            $response['message'] = 'User created successfully';
        } catch (Exception $e) {
            // Revertir la transacción si hay un error
```

```
        $conn->rollback();
        $response['message'] = 'Error creating user';
    }

    break;

case 'profile_form':
    // Obtener el ID del usuario de la sesión
    if (!isset($_SESSION['user_id'])) {
        $response['message'] = 'User not logged in';
        echo json_encode($response);
        exit();
    }

    $user_id = $_SESSION['user_id'];
    $name = $data['name'];
    $surname = $data['surname'];
    $birthdate = $data['birthdate'];
    $gender = $data['gender'];

    // Actualizar los datos del perfil del usuario en la tabla USERS_PROFILES
    $update_profile = $conn->prepare("INSERT INTO USERS_PROFILES (user_id, birth_date, gender) VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE birth_date = VALUES(birth_date),
    $update_profile->bind_param("iss", $user_id, $birthdate, $gender);

    // Actualizar los datos del usuario en la tabla USERS
    $update_user = $conn->prepare("UPDATE USERS SET name = ?, surname = ? WHERE id = ?");
    $update_user->bind_param("ssi", $name, $surname, $user_id);

    if ($update_profile->execute() && $update_user->execute()) {
        $response['success'] = true;
        $response['message'] = 'Profile updated successfully';
    } else {
        $response['message'] = 'Error updating profile';
    }

    break;

case 'diet_form':
    if (!isset($_SESSION['user_id'])) {
        $response['message'] = 'User not logged in';
        echo json_encode($response);
        exit();
```

```
case 'diet_form':
    if (!isset($_SESSION['user_id'])) {
        $response['message'] = 'User not logged in';
        echo json_encode($response);
        exit();
    }
    $user_id = $_SESSION['user_id'];
    $diet = $data['diet'];
    $allergies = implode(',', $data['allergies']);
    $goal = $data['goal'];

    // Actualizar los datos de la dieta del usuario en la tabla USERS_PROFILES
    $update_diet = $conn->prepare("UPDATE USERS_PROFILES SET diet_type = ?, food_allergies = ?, goal = ? WHERE user_id = ?");
    $update_diet->bind_param("ssii", $diet, $allergies, $goal, $user_id);

    if ($update_diet->execute()) {
        $response['success'] = true;
        $response['message'] = 'Diet information updated successfully';
    } else {
        $response['message'] = 'Error updating diet information';
    }
    break;

case 'calories_form':
    if (!isset($_SESSION['user_id'])) {
        $response['message'] = 'User not logged in';
        echo json_encode($response);
        exit();
    }

    $user_id = $_SESSION['user_id'];
    $height = floatval($data['height']);
    $weight = floatval($data['weight']);
    $activity = intval($data['activity']);
    $estimated_calories = intval($data['estimated_calories']);

    $stmt = $conn->prepare("UPDATE USERS_PROFILES SET height = ?, weight = ?, activity = ?, estimated_calories = ? WHERE user_id = ?");
    $stmt->bind_param("ddiii", $height, $weight, $activity, $estimated_calories, $user_id);
    if ($stmt->execute()) {
        $response['success'] = true;
```

11.1.3 Checkout.js

Este código gestiona un carrito de compras en una página web. Inicializa varios elementos del DOM y agrega escuchadores de eventos para manejar la interacción del usuario. Al cargarse la página, se llaman a funciones para imprimir el contenido del carrito y calcular el precio total. Los usuarios pueden eliminar elementos del carrito y cerrar un modal de confirmación. También se actualiza el precio total y se maneja la lógica para iniciar el

proceso de compra mediante un botón de compra que envía los datos del carrito al servidor para crear una sesión de compra.

```
const items_container = document.querySelector('.checkout__left');
const checkoutPrice = document.querySelector('.checkout__price');
const buyButton = document.querySelector('.checkout__buy__button');
const modalCancel = document.querySelector('.modal__cancel');
const btnCloseModal = document.querySelector('.btn__close__modal');

window.addEventListener('DOMContentLoaded', () => {
    eventListeners2();
});


function eventListeners2() {

    printCheckout();
    calculatePrice();

    items_container.addEventListener('click', (e) => {
        deleteCart(e, 'checkout');
    });

    modalCancel.addEventListener('click', (e) => {
        if(!e.target.classList.contains('btn__close__modal')){
            closeModal();
            removeCanceledParam();
        }
    });

    btnCloseModal.addEventListener('click', () => {
        closeModal();
        removeCanceledParam()
    });
}

function closeModal() {      Dani Garrido, 2 days ago • Todo sobre el checkout + páginas de error y suc…
    modalCancel.classList.add('none');
}

function removeCanceledParam() {
    const url = new URL(window.location);
    url.searchParams.delete('canceled');
    window.history.replaceState({}, document.title, url.pathname + url.search);
}
```

```
// Empties the cart html
function emptyCheckout() {
    while (items_container.firstChild) {
        items_container.removeChild(items_container.firstChild);
    }
}

function printCheckout() {

    emptyCheckout();

    if (cart_items.length === 0) {
        items_container.innerHTML = '<p class="empty__cart">Your cart is empty</p>';
    }
    cart_items.forEach(item => {
        const { id, name, price, quantity, img } = item;

        // Create the article element
        const itemContainer = document.createElement('article');
        itemContainer.classList.add('item__container');

        // Create the image container
        const itemImage = document.createElement('div');
        itemImage.classList.add('item__image');
        const imgElement = document.createElement('img');
        imgElement.src = img;
        imgElement.alt = `${name} image`;
        itemImage.appendChild(imgElement);

        // Create the info container
        const itemInfo = document.createElement('main');
        itemInfo.classList.add('item__info');

        // Create the item name and description container
        const itemInfoTop = document.createElement('div');
        const itemName = document.createElement('h2');
        itemName.textContent = name;
        const itemDesc = document.createElement('p');
        itemDesc.textContent = 'This is a description of the item.';
        itemInfoTop.appendChild(itemName);
        itemInfoTop.appendChild(itemDesc);
```

```
    // Create the bottom info container
    const itemInfoBottom = document.createElement('div');
    itemInfoBottom.classList.add('item__info-bottom');
    const itemPrice = document.createElement('p');
    itemPrice.textContent = price;

    const itemMisc = document.createElement('div');
    itemMisc.classList.add('item__misc');

    // Create the quantity select element
    const selectElement = document.createElement('select');
    selectElement.name = 'quantity';
    selectElement.id = 'quantity';
    for (let i = 1; i <= 4; i++) {
        const option = document.createElement('option');
        option.value = i;
        option.textContent = i;
        if (i === quantity) {
            option.selected = true;
        }
        selectElement.appendChild(option);
    }

    // Create the SVG icons
    const favoriteIcon = document.createElementNS('http://www.w3.org/2000/svg', 'svg');
    favoriteIcon.setAttribute('height', '24px');
    favoriteIcon.setAttribute('viewBox', '0 -960 960 960');
    favoriteIcon.setAttribute('width', '24px');
    favoriteIcon.setAttribute('fill', '#5f6368');
    favoriteIcon.innerHTML = '<path d="m480-120-58-52q-101-91-167-157T150-447.5Q111-500 95.5-544T80-634q0-94 63-157t157-63q52 0 99 22t81 62q34-40 81-62t99-22q94 0 157 63
```

```
        // Append elements
        itemMisc.appendChild(selectElement);
        itemMisc.appendChild(favoriteIcon);
        itemMisc.appendChild(deleteIcon);

        itemInfoBottom.appendChild(itemPrice);
        itemInfoBottom.appendChild(itemMisc);

        itemInfo.appendChild(itemInfoTop);
        itemInfo.appendChild(itemInfoBottom);

        itemContainer.appendChild(itemImage);
        itemContainer.appendChild(itemInfo);

        // Append the item container to the items container
        items_container.appendChild(itemContainer);
    });

    syncStorage();
}

function calculatePrice() {
    let total = 0;
    cart_items.forEach(item => {
        total += parseFloat(item.price) * item.quantity;
    });
    total = total.toFixed(2); // Limit the float to 2 digits
    checkoutPrice.textContent = `$${total}`;
}


/* BUY BUTTON CODE */
document.addEventListener('DOMContentLoaded', () => {
    buyButton.addEventListener('click', function() {
        fetch('/checkout/createSession', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(cart_items)
        })
```

11.2 Scripts instalación/ejecución

El único script que hay que ejecutar sería el de la creación de la base de datos. Este script es muy útil, ya que engloba la funcionalidad de crear la bbdd si no existe, borrar las tablas, crearlas de nuevo y añadir datos. De esta manera, cada vez que se requiere inicializar la bbdd desde cero en un contexto de pruebas, bastaría con ejecutar este script una sola vez.

```sql
-- Crear la base de datos si no existe
CREATE DATABASE IF NOT EXISTS tfg;

-- Seleccionar la base de datos
USE tfg;

-- Deshabilitar temporalmente las restricciones de claves foráneas
SET FOREIGN_KEY_CHECKS = 0;

-- Generar y ejecutar las sentencias DROP TABLE para cada tabla en la base de datos
SET @tables = NULL;
SELECT GROUP_CONCAT('`', table_name, '`') INTO @tables
FROM information_schema.tables
WHERE table_schema = 'tfg';

SET @tables = CONCAT('DROP TABLE IF EXISTS ', @tables);
PREPARE stmt FROM @tables;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Habilitar nuevamente las restricciones de claves foráneas
SET FOREIGN_KEY_CHECKS = 1;

-- Crear las tablas
CREATE TABLE USERS(
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    surname VARCHAR(50),
    email VARCHAR(100) NOT NULL UNIQUE,
    username VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE USERS_SECURITY(
    user_id INT NOT NULL,
    pwd VARCHAR(255) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES USERS(id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE USERS_PROFILES(
  user_id INT NOT NULL,
  photo VARCHAR(255),
  gender CHAR(1) CHECK (gender IN ('M', 'F') OR gender IS NULL), -- M=Male, F=Female, allows NULL
  birth_date DATE,
  diet_type VARCHAR(50) CHECK (diet_type IN ('omnivore', 'pescetarian', 'vegetarian', 'vegan') OR diet_type IS NULL), -- allows NULL
  food_allergies VARCHAR(255),
  other_allergies VARCHAR(255),
  height DECIMAL(5,2), -- e.g., 175.50 cm
  weight DECIMAL(5,2), -- e.g., 70.50 kg
  activity INT(1) CHECK (activity IN (0, 1, 2, 3) OR activity IS NULL), -- allows NULL
  estimated_calories INT,
  goal INT(1) CHECK (goal IN (0, 1, 2) OR goal IS NULL), -- allows NULL
  FOREIGN KEY (user_id) REFERENCES USERS(id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE USER_RECIPES (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  api_id VARCHAR(255) NOT NULL,
  FOREIGN KEY (user_id) REFERENCES USERS(id) ON DELETE CASCADE
);
```

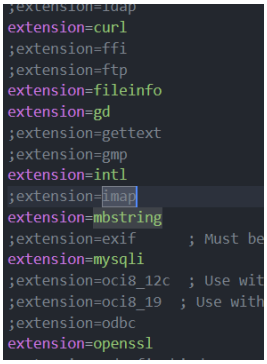
```sql
CREATE TABLE USER_MYRECIPES (
    recipe_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    instructions TEXT,
    ingredients TEXT,
    allergens TEXT,
    photo VARCHAR(255),
    prep_time INT,
    calories INT,
    serves INT,
    FOREIGN KEY (user_id) REFERENCES USERS(id) ON DELETE CASCADE
);
```

```sql
-- Insertar datos en la tabla USERS
INSERT INTO USERS (name, surname, email, username) VALUES
('John', 'Doe', 'john.doe@example.com', 'johndoe'),
('Jane', 'Smith', 'jane.smith@example.com', 'janesmith'),
('Alice', 'Johnson', 'alice.johnson@example.com', 'alicejohnson'),
('Bob', 'Brown', 'bob.brown@example.com', 'bobbrown'),
('Charlie', 'Davis', 'charlie.davis@example.com', 'charliedavis');

-- Insertar datos en la tabla USERS_PROFILES
INSERT INTO USERS_PROFILES (user_id, photo, gender, birth_date, diet_type, food_allergies, other_allergies, height, weight, activity, estimated_calories, goal) VALUES
(1, '/build/img/profile_photos/photo1.jpg', 'M', '1980-01-01', NULL, 'nuts,gluten', 'pollen,dust', 175.50, 70.50, 0, 2500, 0),
(2, '/build/img/profile_photos/photo2.jpg', 'F', '1990-02-02', 'vegan', 'lactose', 'cats', 165.00, 60.00, 3, 2000, 1),
(3, '/build/img/profile_photos/photo3.jpg', 'F', '1985-03-03', 'omnivore', 'seafood', 'penicillin', 170.00, 65.00, 1, 2200, 2),
(4, '/build/img/profile_photos/photo4.jpg', 'M', '1995-04-04', 'pescetarian', 'egg,gluten', NULL, 180.00, 75.00, 3, 2700, 0),
(5, '/build/img/profile_photos/photo5.jpg', 'M', '2000-05-05', 'vegetarian', 'nuts,seafood', 'latex', 160.00, 55.00, 1, 1800, 1);

-- Insertar datos en la tabla USERS_SECURITY
INSERT INTO USERS_SECURITY (user_id, pwd) VALUES
(1, 'password1'),
(2, 'password2'),
(3, 'password3'),
(4, 'password4'),
(5, 'password5');

-- Insertar datos en la tabla USER_RECIPES
INSERT INTO USER_RECIPES (user_id, api_id) VALUES
(1, '156992'),
(2, '102141'),
(3, '204305'),
(4, '284867'),
(5, '368990');

-- Insertar datos en la tabla USER_MYRECIPES
INSERT INTO USER_MYRECIPES (user_id, name, description, instructions, ingredients, allergens, photo, prep_time, calories, serves) VALUES
(1, 'Pasta Primavera', 'A delightful mix of fresh veggies and pasta.', '1. Cook pasta. 2. Sauté vegetables. 3. Mix together.', 'pasta, tomatoes, bell peppers, onions', 'gluten', '
(2, 'Vegan Tacos', 'Delicious plant-based tacos with a variety of toppings.', '1. Prepare toppings. 2. Heat tortillas. 3. Assemble tacos.', 'tortillas, beans, avocado, salsa', 'no
(3, 'Grilled Chicken Salad', 'A healthy and light chicken salad.', '1. Grill chicken. 2. Prepare salad base. 3. Combine and serve.', 'chicken breast, lettuce, tomatoes, cucumber',
(4, 'Quinoa Bowl', 'Nutritious quinoa bowl with mixed veggies.', '1. Cook quinoa. 2. Sauté vegetables. 3. Combine ingredients.', 'quinoa, carrots, bell peppers, broccoli', 'none',
(5, 'Spaghetti Carbonara', 'Classic Italian pasta dish.', '1. Cook spaghetti. 2. Prepare sauce. 3. Mix together.', 'spaghetti, eggs, pancetta, parmesan cheese', 'gluten, dairy', '
```

7.3 Manual de uso

7.3.1 Requisitos:

- **PHP**
- **Composer**
- **Habilitar en php.ini las extensiones “curl, mbstring, intl”**



- **IDE a elección (VSCode en nuestro caso)**

7. 3.2 Instalación:

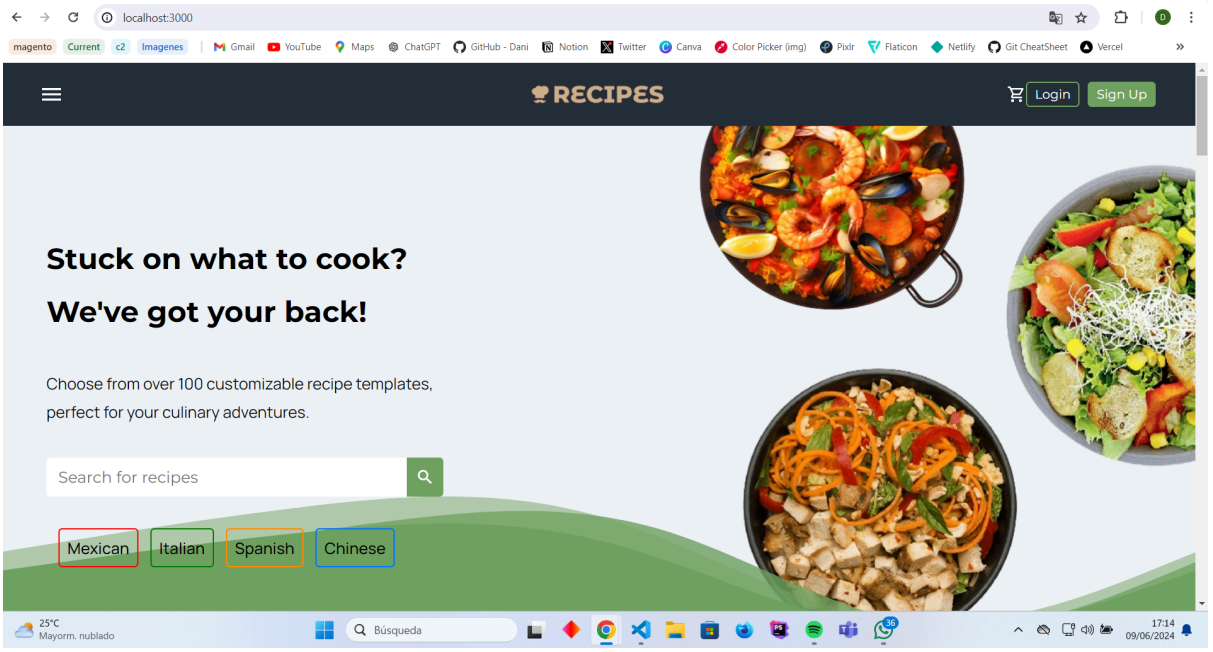
- composer install (para tener la carpeta *vendor*)
- Ejecutar manualmente el código SQL que se encuentra en el archivo “base_datos_sql” dentro de MySql Workbench o el administrador de BD que uses (DBeaver, TablePlus, etc)

7.3.3 Deployment:

- Situarse en la carpeta public/ en la terminal
- Ejecutar php -S localhost:3000
- Abrir Chrome y dirigirse a localhost:3000

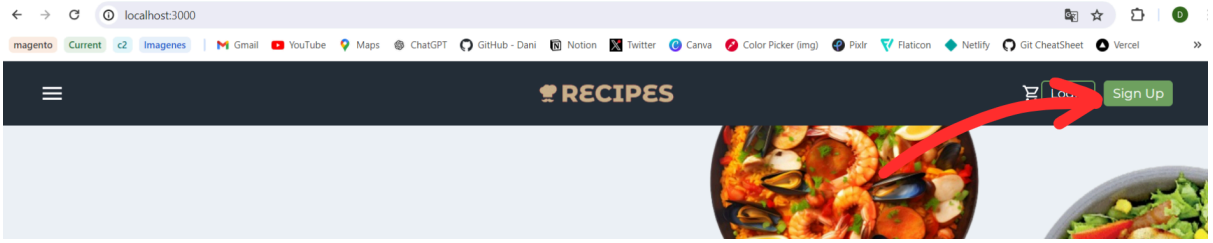
7.4 Guía de uso:

Una vez hayas hecho todos los pasos anteriores, estarás situad@ en una ventana como esta:

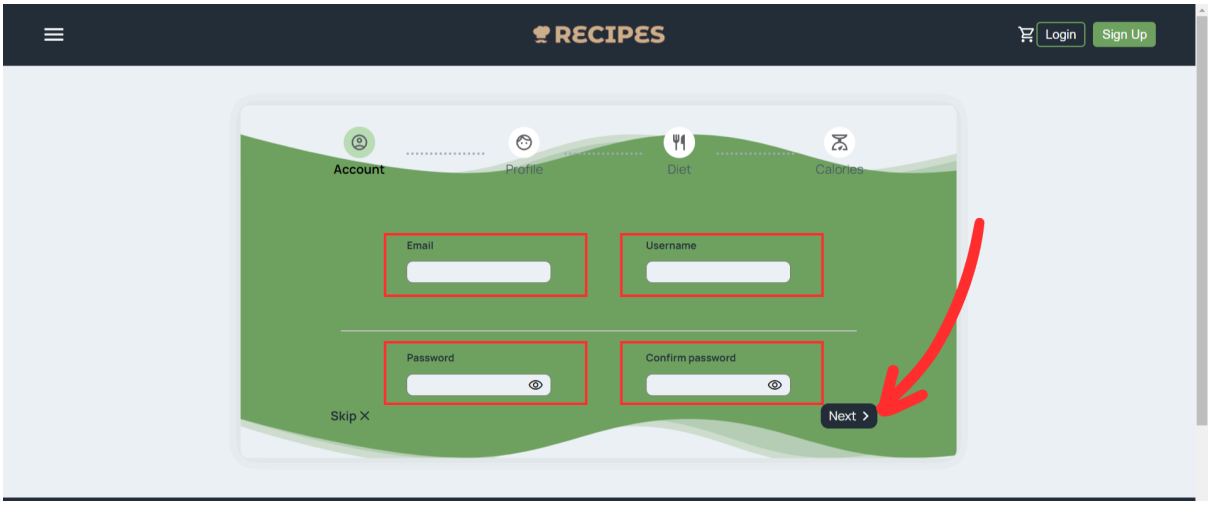


REGISTRO

En caso de querer registrarse hacer click aquí:



Te encontrarás en esta página, rellena los campos marcados y haz click en “Next”:

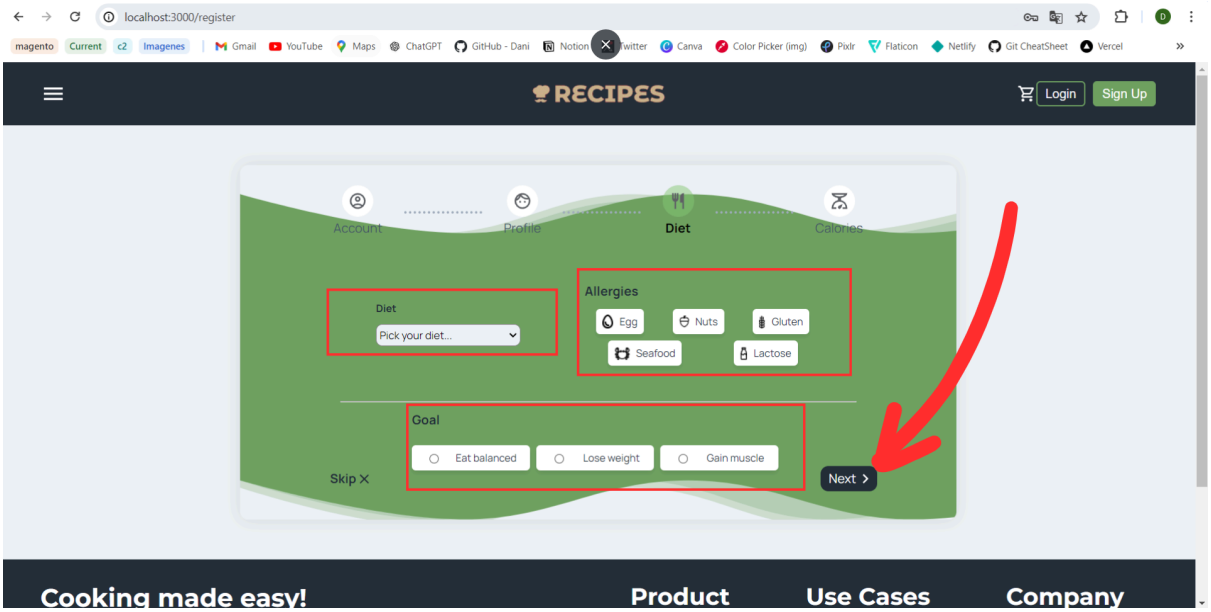


Serás enviado al siguiente paso, donde llenarás la siguiente info:

Al terminar clickar en Next (o puedes saltarte esta parte y clickar en Skip)

Y ya llegarás a la siguiente parte del registro:

Aquí indicarás el tipo de dieta que quieres, posibles alergías y tu objetivo.



Y aquí en el último paso, ingresar datos como la altura, peso y tu nivel de actividad. Una vez rellenado los datos, hacer click en “Show” y se mostrarán tu ingesta calórica diaria en función de lo indicado.

Después hacer click en “Finish” y terminarás el registro.

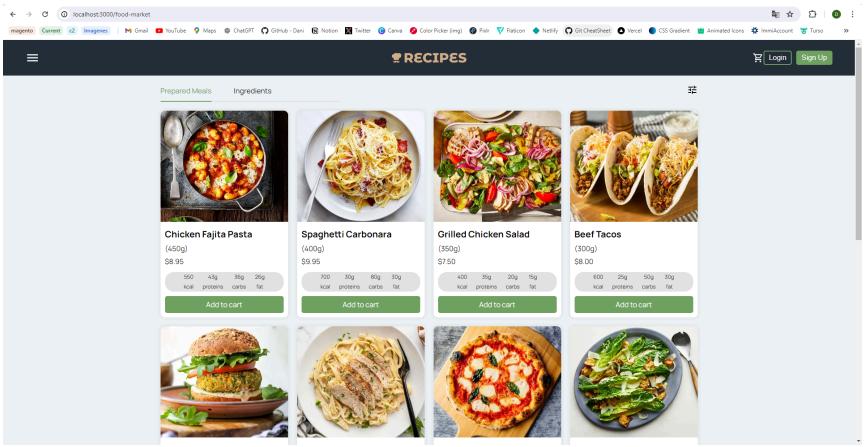
NAVEGACIÓN

En la imagen de abajo tenemos enumerados los elementos clickables que redirigen a X página

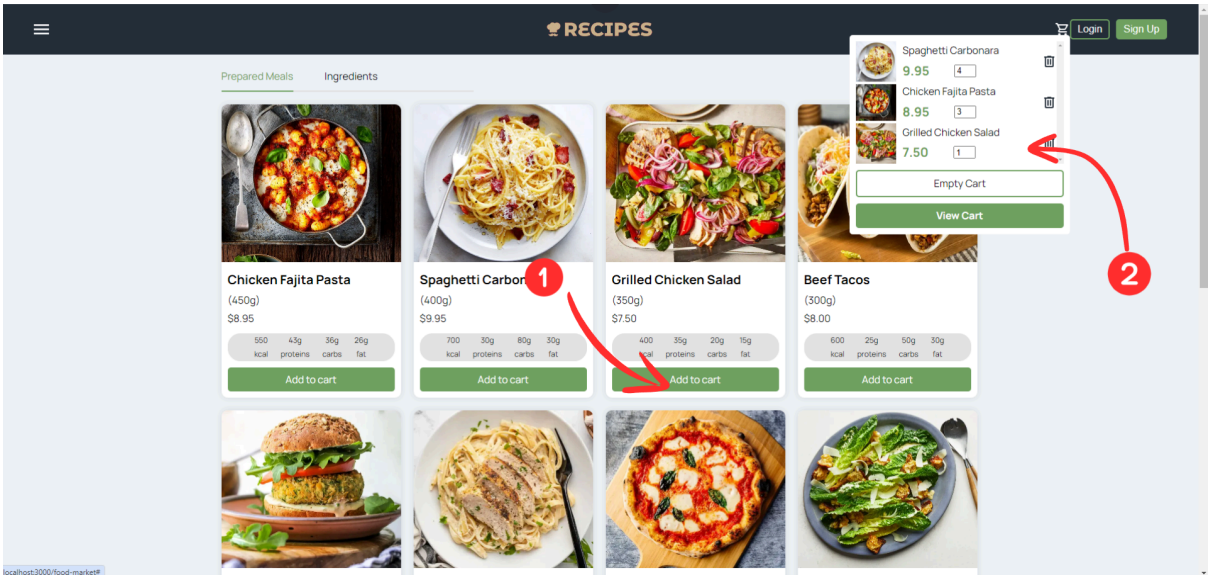
1- Home
2- Food Market (Página en la que se puede comprar nuestros platos a la venta)
3- My Kitchen
4- Recipe Discovery (Página en la que puedes buscar la receta que quieras)
5- Redes Sociales (al hacer click en cualquiera de ellas, te redirige)
6- Login (Redirige a la página para iniciar sesión)
7- Register (Te manda a la página en la que te vas a registrar)

FOOD MARKET

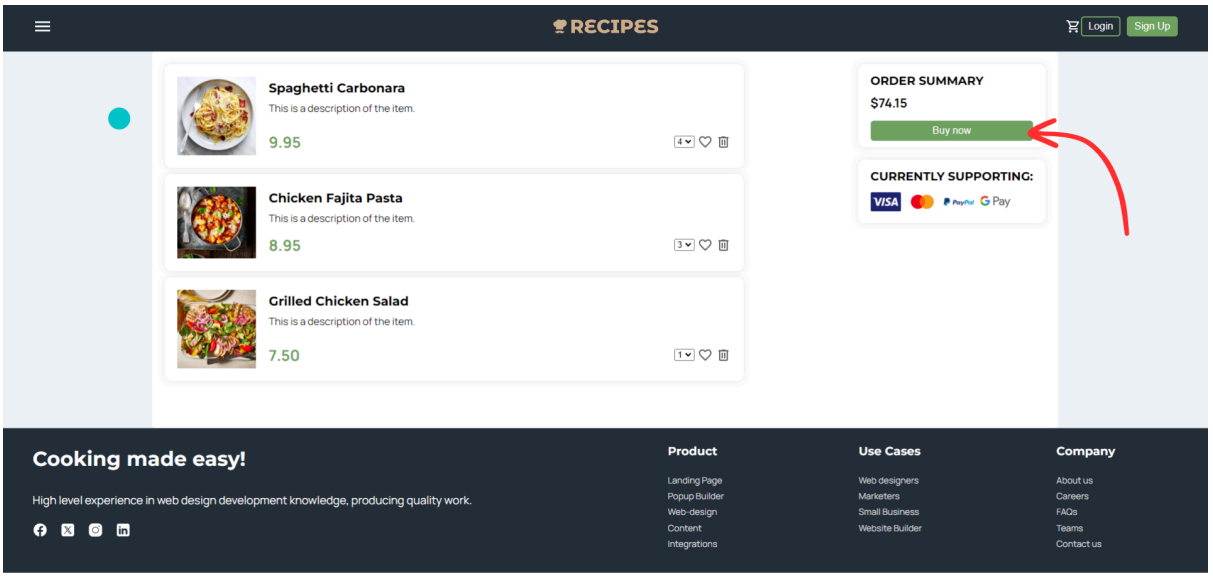
Una vez en la página deberías estar aquí:



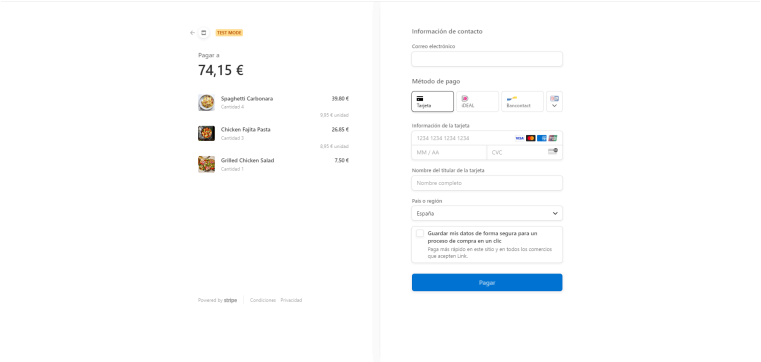
Para agregar una receta simplemente hacer click en “Add to cart” y observar en el carrito que se ha agregado.



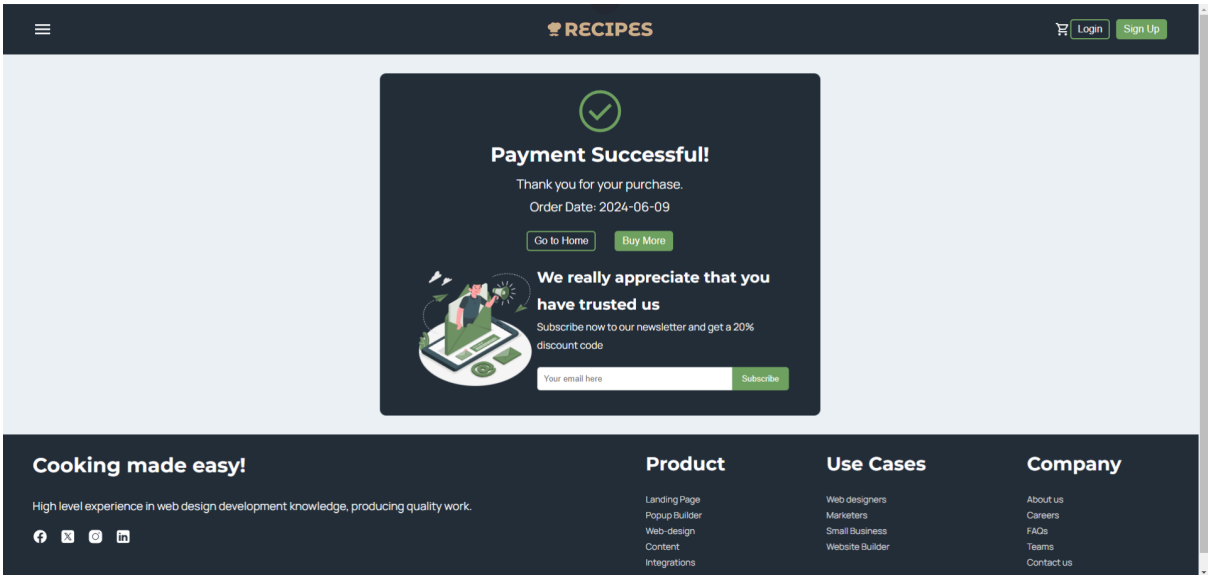
Si haces click en “Add to cart” te llevará al Checkout donde podrás ver de forma más detallada lo que vas a comprar:



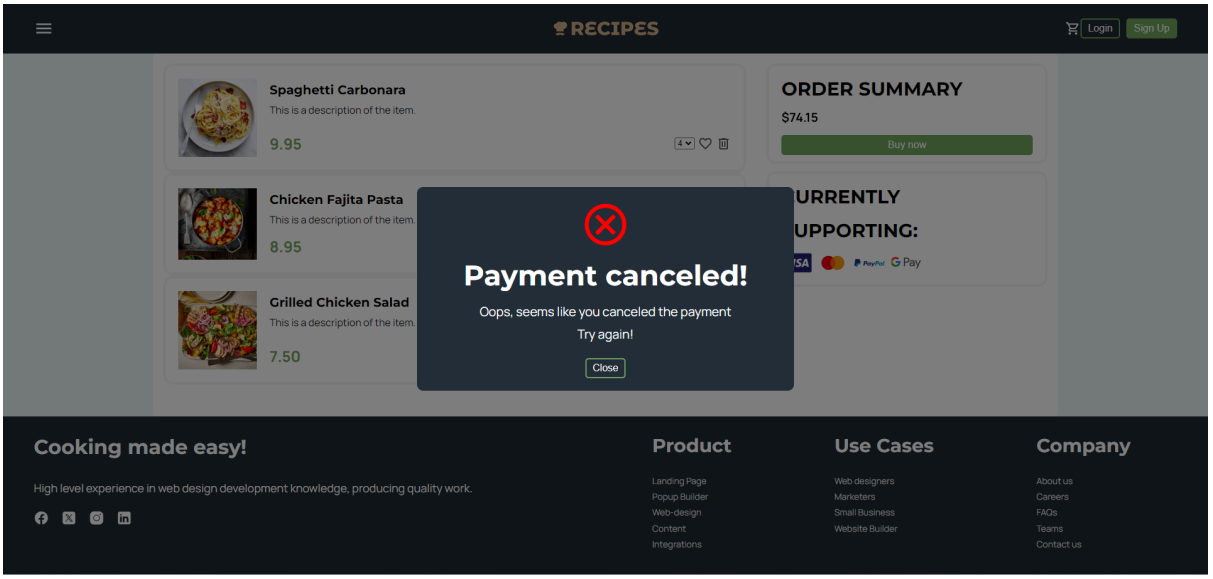
Al darle a comprar serás redirigido a la pasarela de pago:



Una vez compres con éxito serás enviado a la siguiente página:



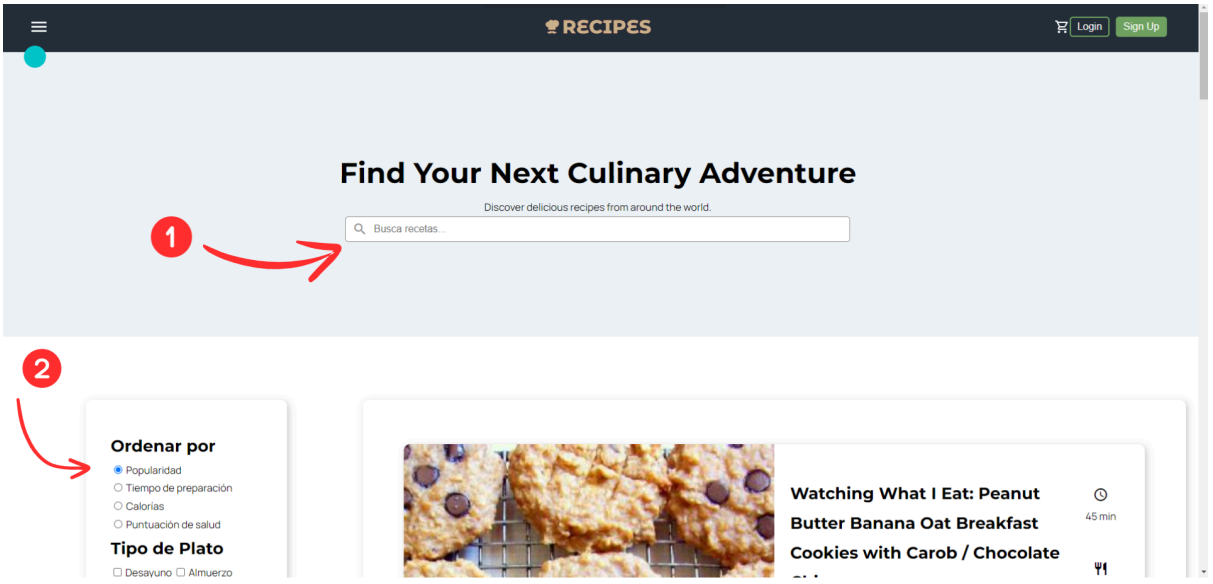
En caso de cancelar el pago:



Cualquier tipo de error como número de tarjeta incorrecto, saldo insuficiente en la cuenta, etc… Se encarga Stripe directamente, ya que estamos usando su propia página de Checkout que tienen integrada.

RECIPE DISCOVERY

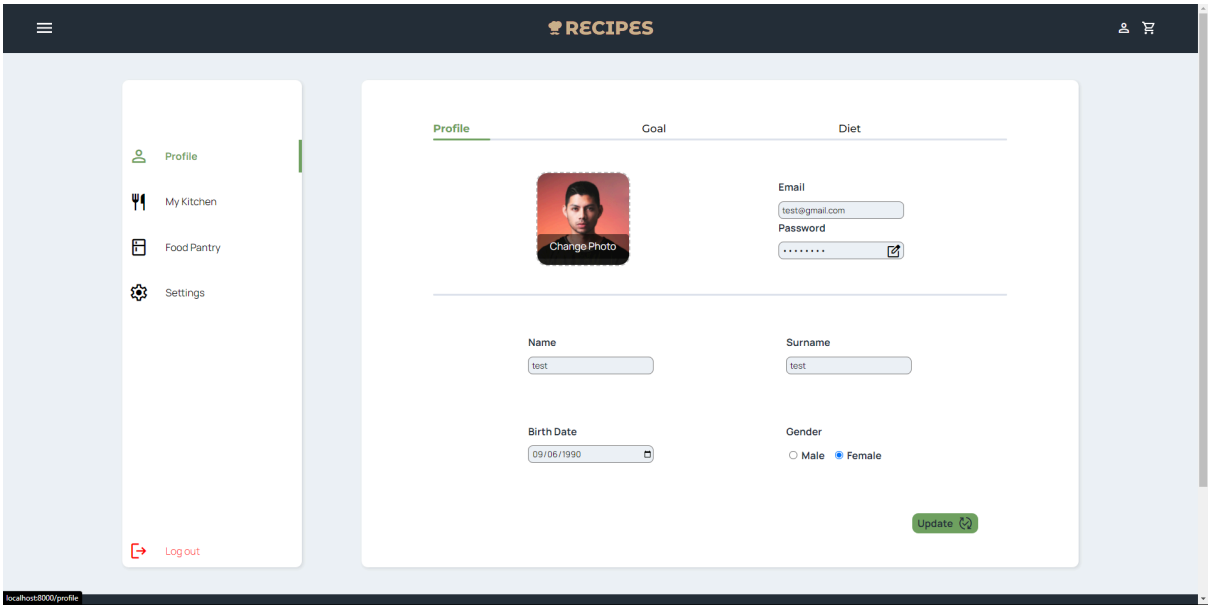
Una vez dentro puedes buscar por nombre directamente en la barra de búsqueda (1) o puedes usar los filtros (2) y se te actualizarán los resultados.



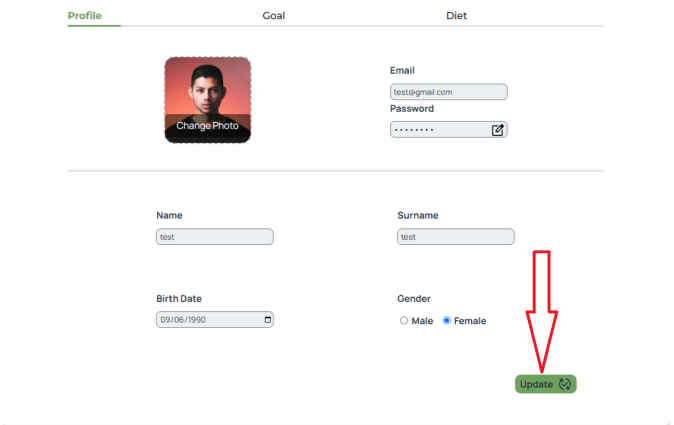
PROFILE

Una vez vas a tu perfil clicando en el header al icono:

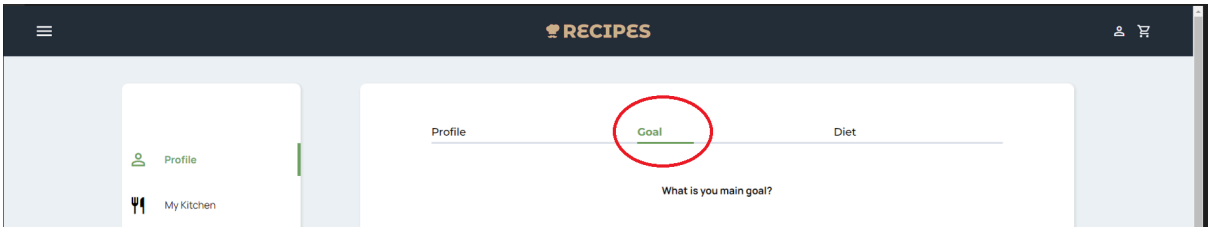
Serás redirigido a esta página:



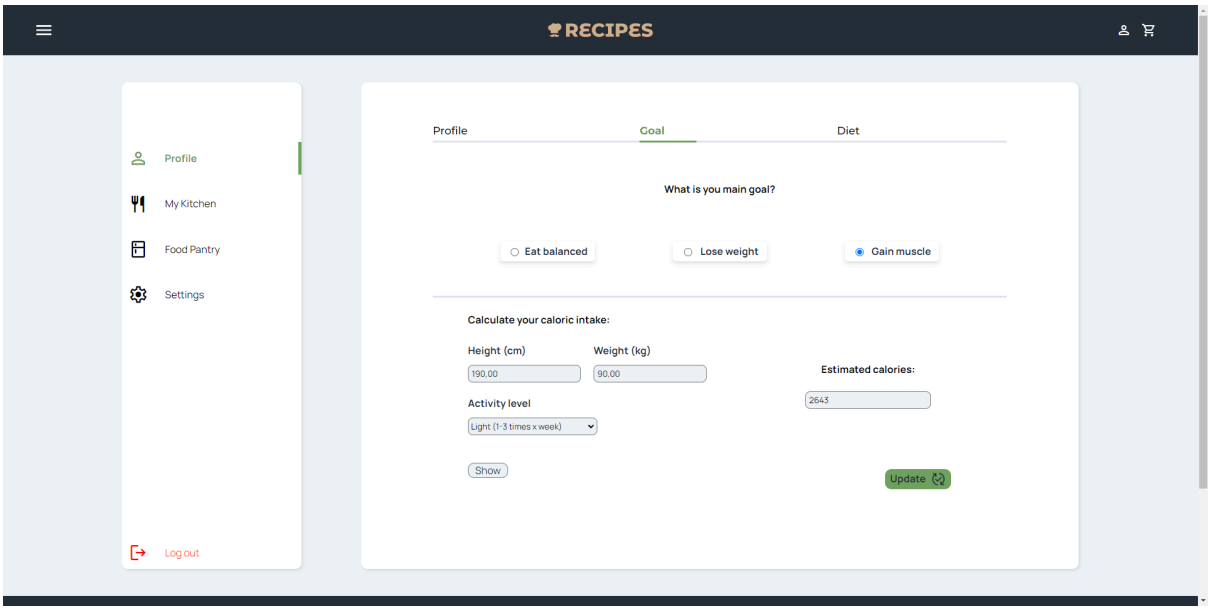
En la página de arriba, puedes ver todos los campos que puedes modificar, una vez modificas los cambios que quieres, le das al botón de “Update” para actualizar los cambios, el botón es el que te enseño aquí debajo:



En la siguiente pestaña:



Tendrás los campos para modificar tus objetivos. Se guardan de la misma manera que el paso anterior



Y por último, en el paso 3, el cual es “Diet”:

Aquí puedes cambiar tus alérgenos y el tipo de dieta que sigues.