Trabajo de Fin de Grado

PLATAFORMA DE BÚSQUEDA Y RESERVA DE RESTAURANTES EN LÍNEA

TITULACIÓN: Grado en Ingeniería Informática

AUTORA: Kiowa del Carmen Andueza Cárdenes

TUTORIZADO POR:

Dr. Luis Miguel Hernández Acosta

1 de septiembre de 2023

Agradecimientos

Quisiera expresar mi más profundo agradecimiento a mi familia, en particular a mi querida abuela María. A pesar de que ya no está con nosotros, su apoyo constante durante mi trayectoria académica ha sido indispensable para llegar hasta aquí. Más que nadie, ella merece este reconocimiento.

Además, quiero agradecer a los amigos que me han acompañado durante mi carrera, siempre dispuestos a ayudarme cuando más lo necesitaba. En especial, a Alexis, quien nunca dejó de confiar en mí.

A mi pareja, quiero agradecerle su motivación constante, su paciencia y su gran ayuda durante estos últimos años. Su apoyo ha sido fundamental.

Por último, pero no menos importante, deseo expresar mi gratitud hacia mi tutor, Luis Miguel Hernández Acosta. Siempre ha estado dispuesto a ayudarme y aconsejarme cuando lo necesitaba. Su orientación ha sido esencial para este proyecto.

Gracias a todos ustedes por haber formado parte de esta etapa tan importante en mi vida.

Resumen

En los últimos años, España ha enfrentado el impacto de una pandemia global que llevó al cierre de numerosos negocios, incluyendo restaurantes. A esta situación se suma un conflicto bélico entre Rusia y Ucrania que ha desencadenado una gran inflación en los precios de los alimentos, generando presiones tanto para los consumidores como para los propietarios de restaurantes, viéndose obligados a tener que reducir gastos.

Por esa razón, surge GastroNet, una plataforma de búsqueda y reserva de restaurantes en línea que no solo ayuda a los restaurantes a aumentar su visibilidad y atraer a más clientes, sino que también ofrece a los usuarios la posibilidad de comparar precios adaptados a su economía, además de facilitar la planificación de sus comidas diarias.

GastroNet intenta ayudar a impulsar la economía de los restaurantes que hacen uso de nuestra plataforma y ofrece una gran variedad de establecimientos a nuestros clientes.

Abstract

In recent years, Spain has faced the impact of a global pandemic that led to the closure of numerous businesses, including restaurants. To this situation is added a military conflict between Russia and Ukraine that has triggered significant inflation in food prices, creating pressures for both consumers and restaurant owners, who find themselves compelled to reduce expenses.

For this reason, GastroNet emerges, an online restaurant search and reservation platform that not only helps restaurants increase their visibility and attract more customers but also offers users the ability to compare prices tailored to their economy, in addition to facilitating the planning of their daily meals.

GastroNet aims to assist in boosting the economy of the restaurants that make use of our platform and provides a wide variety of establishment options to our customers.

Índice general

Índice de figuras

1. Introducción

1.1. Contexto

Actualmente, España, como muchos otros países, sufre las consecuencias de la pandemia mundial. La salud de los ciudadanos se ha visto afectada por la enfermedad, y el impacto económico también está teniendo graves consecuencias. La imposición de medidas restrictivas por parte del gobierno, especialmente en el sector de la hostelería, obligaron a muchas empresas, en su mayoría dirigidas por pequeños empresarios, a cerrar sus puertas.

A este problema, también se han sumado las consecuencias de los conflictos bélicos entre Rusia y Ucrania. El aumento del precio del combustible ha tenido un gran impacto en la subida de los alimentos, lo cual representa otro desafío significativo para el sector de la restauración.

Por otro lado, los consumidores no han logrado eludir las consecuencias de los problemas mencionados anteriormente. Asimismo, algunos trabajadores se han visto afectados por las medidas adoptadas en el sector hostelero. Los empresarios, forzados a reducir personal para hacer frente a las pérdidas durante la pandemia, también enfrentan dificultades financieras adicionales. Muchos consumidores, además, han tenido que ajustarse el cinturón recortando gastos innecesarios, incluyendo los relacionados con la cesta de la compra, para hacer frente al impacto económico actual.

Este 2023, España lo ha iniciado con desafíos económicos significativos que fueron predichos a principios de año y han afectado a diversos aspectos de la vida cotidiana. En un artículo de RTVE publicado el 1 de enero de 2023, se aborda detalladamente el panorama del incremento de los alimentos, gasolina e hipotecas [1]. Según la fuente,

> "Este año arranca marcado por un importante encarecimiento de la cesta de la compra. Hasta mediados de enero no se conocerán los incrementos definitivos de diciembre con

> respecto a los alimentos, pero tomando como referencia el dato definitivo del INE de noviembre de 2022, el pan ha subido casi un 15% y un *brick* de leche entera, un 30,9%. A la lista se añaden el aceite de oliva, con un incremento del 25,9%; las verduras frescas, un 14,6%; y la carne, pescado y marisco fresco, con una subida por encima del 11%".

En este documento se presenta una posible solución tecnológica que podría servir como medida para ayudar tanto al consumidor a seleccionar establecimientos acordes a su economía como a los empresarios que utilicen la herramienta para aumentar su visibilidad.

1.2. Motivación

La motivación principal para abordar una solución tecnológica que busque ayudar al sector de restauración surge especialmente de experiencias personales vividas en mi entorno cercano. Tanto empresarios hosteleros como camareros en mi círculo familiar y social han experimentado dificultades económicas. He presenciado cómo los empresarios se han visto obligados a tomar medidas duras para mantener el negocio a flote, desde reducciones salariales hasta préstamos bancarios para enfrentar la situación. Al mismo tiempo, los camareros se has visto afectados directamente por estas decisiones, sintiendo el impacto en su estabilidad financiera.

Esta especial conexión con la industria hostelera me ha impulsado a contribuir con este proyecto para abordar los problemas que enfrentan en su día a día.

2. Objetivos

El objetivo principal de este trabajo de fin de grado es desarrollar una solución tecnológica que aumente la visibilidad de los restaurantes españoles y estimule la economía en este sector. Este proyecto está especialmente enfocado en pequeñas empresas, ya que son las principales afectadas.

La aplicación web permite que cualquier restaurante registrado reciba reservas y reseñas de parte de los clientes. Además, se busca facilitar la elección de los usuarios al proporcionar información detallada sobre precios y calidad, basada en las experiencias de otros clientes.

Para los restaurantes que se registren en la aplicación, se establece la condición de publicar únicamente un negocio por cuenta. Deben proporcionar información sobre el establecimiento, incluyendo dirección, precio medio por comensal, tipos de alérgenos, tipos de dietas, precios de la carta, y añadir imágenes para atraer al cliente.

En el caso de los usuarios registrados como clientes, pueden realizar búsquedas personalizadas que se adapten a sus preferencias, compartir reseñas para orientar a otros usuarios, realizar reservas y agregar restaurantes a su lista de favoritos.

3. Competencias específicas y aportaciones del trabajo

3.1. Competencias específicas cubiertas

CII01 - "Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente".

Para asegurar el cumplimiento de la competencia **CII01**, se lleva a cabo un estudio detallado del mercado tecnológico y de la competencia directa de otras aplicaciones. Además, se emplean servicios que garantizan la integridad de la privacidad de los datos. Asimismo, se implementan versiones estables de las tecnologías utilizadas para proporcionar un entorno seguro y confiable.

TI02 - "Capacidad para seleccionar, diseñar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados".

La competencia **TI02** se logra mediante una evaluación previa de los requisitos necesarios para el desarrollo de la aplicación. Se emplean herramientas de diseño para crear modelos de la interfaz web, y otras herramientas para la gestión y la organización del proyecto. Este enfoque permite integrar de manera efectiva las etapas de diseño y desarrollo, garantizando que la implementación cumple con los requisitos establecidos y se gestiona eficientemente a lo largo del ciclo del proyecto.

TI07 - "Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos".

Para asegurar el cumplimiento de la competencia **TI07** en el proyecto, se implementa una metodología de desarrollo iterativa que

permite revisar y mejorar continuamente la seguridad del sistema durante todo el proceso de desarrollo. Además, en la aplicación se incorpora un sistema de gestión de acceso que utiliza métodos seguros como correo electrónico y contraseña, así como identificadores únicos de identidad para garantizar un acceso seguro y controlado a la plataforma. Estas medidas contribuyen a comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos en el contexto del proyecto.

3.2. Aportaciones del trabajo

Este proyecto busca impulsar la economía en las provincias españolas al destacar y promover la diversidad de los restaurantes locales. Al atraer más clientes a estos establecimientos, se busca incrementar sus ingresos, generando un impacto positivo en su estabilidad financiera. Esta mejora económica no solo contribuiría a evitar despidos, sino que también podría impulsar la creación de nuevos empleos para satisfacer la creciente demanda de servicios en el sector de la restauración.

Apoyar a los negocios locales no solo beneficia a los restaurantes, sino que también genera un impacto positivo en el tejido social. Favorece la diversidad gastronómica tanto local como internacional. El proyecto alienta a los usuarios a apoyar al negocio español e impulsa el crecimiento del sector hostelero, contribuyendo al bienestar general de los ciudadanos españoles.

4. Desarrollo

4.1. Metodología

La metodología aplicada durante el desarrollo de esta aplicación web es el desarrollo iterativo e incremental. En cada iteración, el proceso comienza con un análisis detallado de los requisitos del *software*. Posteriormente, se procede a la implementación y prueba de nuevas funcionalidades.

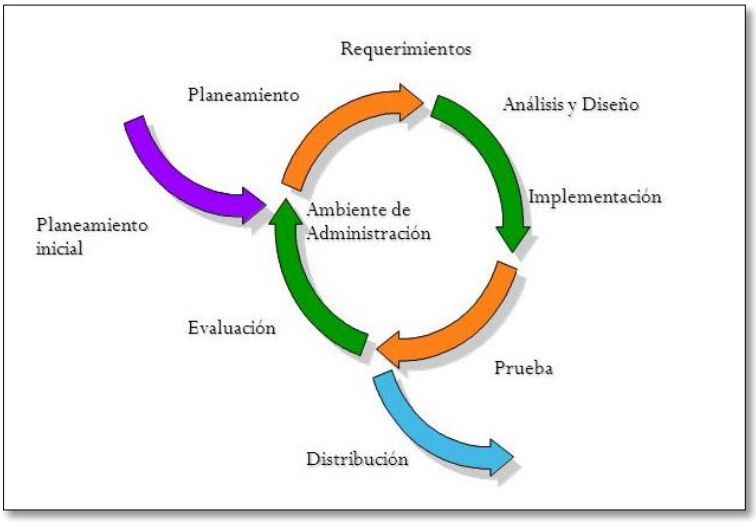

Ilustración 1: Gráfico del proceso iterativo e incremental.

A medida que avanza cada iteración, la aplicación incrementa y adquiere más funcionalidades, generando así un mayor valor para el cliente. Además, en cada ciclo iterativo se vuelven a examinar y, si fuese necesario, mejorar aspectos del producto ya desarrollados, garantizando así su calidad y eficiencia en cada etapa del desarrollo.

La elección de esta metodología se debe a su capacidad para adaptarse fácilmente a los cambios del proyecto. Permite ajustar y evolucionar los requisitos durante el proceso de desarrollo, facilitando la incorporación de cambios de manera sencilla sin comprometer la calidad del producto final. Esto garantiza que el resultado cumpla con las expectativas del cliente.

4.2. Tecnologías

4.2.1. Herramientas de desarrollo

- **Django:**

Este *framework* se utiliza para desarrollar la aplicación web y gestionar las conexiones y peticiones con la base de datos. El lenguaje de programación que utiliza esta herramienta es Python. A través de este marco de trabajo, se implementa la arquitectura de Modelo-Vista-Controlador (MVC), en la cual, tal como indica el nombre, existen tres componentes principales encargados de administrar la aplicación. En este contexto, el modelo gestiona los datos, la vista es la interfaz de usuario, y el controlador maneja las solicitudes entre el modelo y la vista.

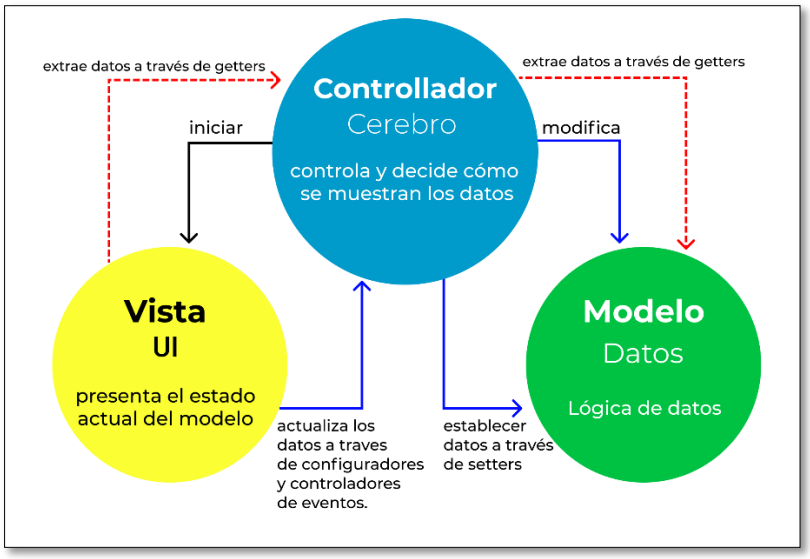

Ilustración 2: Arquitectura de MVC

- **React JS:**

La interfaz gráfica de usuario se implementa a través de React, una herramienta basada en JavaScript. Gracias a ella, se logra desarrollar una aplicación web intuitiva, fácil de utilizar y adaptable a diferentes dispositivos.

- **Firebase Authentication:**

Firebase Authentication es un servicio en la nube de Google que brinda seguridad a los usuarios durante el proceso de registro o inicio de sesión. Se encarga de gestionar sistema de autenticación que ofrece el sitio web.

- **Cloud Firestore:**

Cloud Firestore es otra herramienta en la nube de Google que se utiliza como base de datos no relacional para almacenar información personal de los usuarios.

- **Firebase Storage:**

Firebase Storage se utiliza en la aplicación para asegurar el almacenamiento de imágenes en la nube. Esta herramienta simplifica la administración de imágenes y permite a los usuarios compartir este tipo de archivos a través de la aplicación de manera sencilla y segura.

- **Github:**

GitHub sirve como un alojamiento seguro para guardar el código fuente de una aplicación. En este caso, se utiliza para hospedar el código del proyecto en un repositorio con el fin de tener una copia de seguridad. Sin embargo, en el futuro, podría utilizarse como un sistema de control de versiones.

4.2.2. Herramientas de diseño

- **Figma:**

El diseño de *mockups* para el desarrollo de la interfaz de usuario se realiza con Figma antes de la implementación. Esta herramienta permite modelar un prototipo antes de iniciar la fase de desarrollo del código fuente.

- **StarUML:**

StarUML es una herramienta que se utiliza en el proyecto para el diseño y desarrollo del *software*. Se emplea para la creación de diagramas de flujo y para desarrollar los casos de uso, es decir, se utiliza para indicar el flujo de datos en una funcionalidad y para la descripción de tareas que puede realizar cada usuario.

4.2.3. Librerías

- **Bootstrap:**

Bootstrap es una librería que ofrece estilos personalizados y componentes predefinidos, facilitando así el desarrollo de interfaces de usuario en aplicaciones web. En este proyecto, se utiliza principalmente para la adaptación del diseño a dispositivos de diversos tamaños. Además, se emplea en la creación de formularios y sistemas de paginación para mejorar la apariencia y la experiencia de usuario.

- **Leaflet:**

Leaflet es una biblioteca de JavaScript empleada para la visualización de mapas en la aplicación, encargada de mostrar la ubicación aproximada de los restaurantes. Entre sus diversas herramientas, destacan la capacidad de acercar o alejar la vista del mapa, el marcador que indica la ubicación exacta y la posibilidad de desplazarse por el mapa para explorar la zona.

- **Nominatim:**

Nominatim es una biblioteca de Python que se encarga de realizar búsquedas de direcciones en OpenStreetMap. En el *backend* de la aplicación, se utiliza para obtener la ubicación de cada restaurante. Además, esta herramienta cuenta con una funcionalidad que permite restringir la búsqueda de direcciones a un territorio específico, en este caso, se limita la búsqueda exclusivamente al territorio español.

- **SweetAlert2:**

SweetAlert2 es una librería de JavaScript que se utiliza para mostrar mensajes de error, confirmación o éxito en la aplicación. Estos mensajes se pueden personalizar, lo que mejora significativamente la experiencia del usuario. Además, aporta un diseño atractivo a la aplicación a la hora de informar al usuario.

4.3. Plan de trabajo

En la implementación, tal como se describe en la sección 4.1, se emplea la metodología de desarrollo iterativo e incremental. Cada iteración se muestra en la siguiente ilustración.

Fases	***Duración Estimada*** *(horas)*	***Tareas*** *(nombre y descripción, obligatorio al menos una por fase)*
Estudio previo / Análisis	35	Tarea 1.1: Estudiar el estado del arte
		Tarea 1.2: Establecer los requisitos
		Tarea 1.3: Diseñar *mockups*
Diseño / Desarrollo / Implementación	195	Tarea 2.1: Preparar el entorno
		Tarea 2.2: Implementar la autenticación de usuarios
		Tarea 2.3: Implementar gestión del establecimiento
		Tarea 2.4: Integración de mapas
		Tarea 2.5: Implementar el filtrado de búsquedas
		Tarea 2.6: Implementar el sistema de reservas
		Tarea 2.7: Implementar la gestión de favoritos
		Tarea 2.8: Implementar el sistema de calificación y reseñas
Evaluación / Validación / Prueba	20	Tarea 3.1: Pruebas de funcionalidad
		Tarea 3.2: Pruebas de interfaz
		Tarea 3.3: Pruebas de rendimiento
		Tarea 3.4: Pruebas de seguridad
Documentación / Presentación	50	Tarea 4.1: Elaborar la memoria
		Tarea 4.2: Elaborar el manual de usuario
		Tarea 4.3: Preparar y desarrollar la presentación

Ilustración 3: Organización del proyecto

En la primera iteración, se lleva a cabo un análisis detallado del mercado de la competencia con el objetivo de conocer en detalle las funcionalidades que ofrecen a los usuarios. Al completar esta fase, se definen los requisitos de la aplicación, manteniéndolos flexibles para posibles cambios a lo largo del desarrollo. Finalmente, se diseñan los *mockups* para obtener una representación gráfica de la interfaz de usuario con las diferentes funcionalidades de la aplicación.

En la siguiente iteración, se procede con el desarrollo del código fuente de la aplicación. Este proceso abarca la preparación del entorno, que incluye la implementación de un entorno virtual de Python, la instalación de Django, la configuración de la conexión con la base de datos y la actualización de los paquetes de React, considerando que este último ya se encuentra instalado en el equipo. Posteriormente, se abordan las funcionalidades del sitio web.

Cabe destacar que, durante este proceso, se ajusta el orden de desarrollo de algunas tareas en la fase de la implementación. Estas modificaciones han sido reflejadas correctamente en la ilustración 3. Enumerando cada una de ellas en el orden adecuado y añadiendo otras funcionalidades que surgieron durante el desarrollo, se incluyen: autenticación de usuario, gestión del establecimiento, integración de mapas, filtrado de búsqueda, sistema de reservas, administración de favoritos y sistema de calificación y reseñas.

En la cuarta iteración, se llevan a cabo pruebas exhaustivas de operatividad, interfaz, rendimiento y seguridad. Es importante señalar que, además, estas pruebas se realizaron otras cada vez que se implementaba una nueva funcionalidad en la aplicación.

Por último, la fase de preparación de la documentación del proyecto tanto la del manual de usuario [2] como la de la memoria del trabajo de fin de grado.

5. Análisis preliminar

5.1. Descripción del problema

Los restaurantes españoles, como se menciona anteriormente, están enfrentando las consecuencias tanto de la pandemia mundial como de los conflictos bélicos entre Ucrania y Rusia. Sin embargo, no solo los empresarios del sector hostelero se ven afectados, sino también los trabajadores, como resultado de las medidas adoptadas por los dueños para afrontar la situación. Además, se suma el incremento en el costo de los alimentos, lo que impacta directamente tanto a los consumidores como a los restaurantes, llevándolos a realizar recortes en sus gastos.

5.2. Historia del arte

En la actualidad, diversas aplicaciones web brindan servicios de búsqueda y reserva de restaurantes en línea. Este proyecto se fundamenta en ofrecer, sin ánimo de lucro, herramientas a los establecimientos españoles para impulsar la economía de sus negocios.

En el mercado actual, existen aplicaciones como TheFork y Opentable, que comparten características similares con este proyecto. TheFork ofrece a los restaurantes publicar su negocio sin costos, los tres primeros meses. Después de este tiempo, deben elegir un plan de pago mensual, ya sea básico o *premium*, el que mejor se adapte a sus necesidades. Además, TheFork cobra a los restaurantes una comisión por cada reserva realizada a través de la plataforma.

Por otro lado, Opentable requiere pagos desde el primer mes de uso y también implementa un sistema de comisiones para los restaurantes en cada reserva. Este enfoque permite a Opentable generar ingresos tanto a través de suscripciones y comisiones.

Sin embargo, es importante destacar que, a diferencia de este proyecto, tanto TheFork como Opentable tienen un gran alcance mundial. Cabe resaltar que no comparten el mismo propósito que este proyecto, el cual está centrado específicamente en impulsar la economía del sector hostelero español.

5.3. Fortalezas

Esta es una aplicación que se destaca por ofrecer una interfaz de usuario intuitiva y fácil de utilizar, lo que facilita tanto a los empresarios como a los clientes una buena experiencia de usuario. En el caso de los empresarios, el proyecto les brinda la oportunidad de dar a conocer su negocio de forma gratuita, aumentando así su visibilidad en el mercado. Además, simplifica la gestión de reservas del establecimiento de manera eficiente.

Por otro lado, para los clientes, la aplicación les ofrece la opción de buscar restaurantes cercanos a su ubicación mediante la herramienta de geolocalización. No obstante, esta búsqueda no se limita solo a la proximidad, también pueden seleccionar cualquier otra provincia de España o incluso ajustar la búsqueda según otras preferencias, como el rango de precios, tipos de alérgenos o dietas. Esto garantiza que el usuario obtenga resultados acordes a sus preferencias culinarias.

Además de simplificar la búsqueda, esta herramienta facilita al cliente la organización de sus comidas diarias mediante el sistema de reserva *online*. Esta funcionalidad facilita el proceso y permite a los usuarios planificar su tiempo de ocio cómodamente desde cualquier lugar. En resumen, con este proyecto no solo se benefician los empresarios hosteleros al promover sus negocios, sino que también mejora la experiencia de los clientes proporcionándoles herramientas útiles y accesibles para encontrar establecimientos adaptados a sus gustos.

5.4. Debilidades

Como cualquier aplicación, esta también presenta algunas debilidades y aspectos que se proponen a continuación para mejorar en futuras versiones. En primer lugar, la principal vulnerabilidad consiste en que los clientes podrían realizar reservas y luego no asistir al establecimiento. A pesar de que nuestras políticas detallan que este tipo de comportamientos pueden llevar a la suspensión de la cuenta, es difícil controlar en su totalidad que algunas personas utilicen esta herramienta de manera fraudulenta para perjudicar a los establecimientos.

Otro aspecto que podría ser mejorado es la implementación de un sistema de mensajería. Esto facilitaría la comunicación directa entre los clientes y los restaurantes a través de la aplicación, permitiendo resolver dudas o atender peticiones especiales relacionadas con las reservas. Integrar esta funcionalidad contribuiría a mejorar la experiencia del usuario, ya que les brinda la posibilidad de comunicarse a través del canal para resolver cualquier inconveniente.

5.5. Solución propuesta

Este proyecto surge, como se menciona anteriormente, con el propósito principal de ayudar al sector hostelero a enfrentar las consecuencias derivadas tanto de la postpandemia como de los conflictos bélicos entre Rusia y Ucrania. Este último evento ha ocasionado un aumento significativo en el costo de los productos alimentarios, impactando directamente en los empresarios y consumidores.

Como solución, se propone que cada empresario que lo desee pueda publicar su negocio con el objetivo de aumentar sus ganancias y mejorar las condiciones laborales de sus empleados. El uso de la herramienta tiene como consecuencia directa la posibilidad de ofrecer estabilidad laboral y promover la creación de empleo para hacer frente

a las altas demandas.

Es importante destacar que la presencia de todos los establecimientos en esta herramienta depende de los restaurantes que decidan registrarse. Al ofrecer esta oportunidad de manera gratuita, se motiva a los restaurantes a probar la aplicación sin compromiso, lo cual podría resultar en una mejora significativa en su rendimiento empresarial. Cuanto mayor sea la diversidad, más beneficios se proporcionarán al consumidor. A medida que se incrementan las opciones, hay más posibilidades de que la oferta disponible se ajuste a las preferencias del cliente.

6. Requisitos

6.1. Requisitos de usuario

La aplicación consta de tres tipos de usuarios posibles: no registrado, restaurante y cliente. A continuación, se detallan las historias de usuario para cada uno de los roles.

6.1.1. No registrado

- Como usuario no registrado, deseo registrarme como cliente.
- Como usuario no registrado, deseo registrarme como restaurante.
- Como usuario no registrado, deseo buscar restaurantes por provincia/ciudad, fecha y número de comensales.
- Como usuario no registrado, deseo filtrar la búsqueda mediante geolocalización.
- Como usuario no registrado, deseo filtrar la búsqueda por alérgenos.
- Como usuario no registrado, deseo filtrar la búsqueda por tipo de dieta.
- Como usuario no registrado, deseo seleccionar un restaurante para ver sus detalles.
- Como usuario no registrado, deseo ver la ubicación del restaurante en el mapa.
- Como usuario no registrado, deseo acceder a la carta del restaurante.
- Como usuario no registrado, deseo ver imágenes del establecimiento.
- Como usuario no registrado, deseo leer las reseñas de los clientes sobre el restaurante.
- Como usuario no registrado, deseo contactar con el administrador del sitio web.

6.1.2. Restaurante

- Como restaurante, deseo iniciar sesión con mis credenciales.
- Como restaurante, deseo modificar la imagen de mi perfil.
- Como restaurante, deseo actualizar mis datos personales en el perfil (nombre y apellidos, nombre del restaurante, dirección, correo electrónico o contraseña).
- Como restaurante, deseo publicar mi establecimiento.
- Como restaurante, deseo modificar la información de mi establecimiento (teléfono de contacto, tipos de alérgenos, tipo de dieta, carta, horario, aforo, imágenes y precio medio por comensal).
- Como restaurante, deseo leer las reseñas de mi establecimiento.
- Como restaurante, quiero buscar restaurantes por provincia/ciudad, fecha y número de comensales.
- Como restaurante, deseo filtrar la búsqueda mediante geolocalización.
- Como restaurante, deseo filtrar la búsqueda por alérgenos.
- Como restaurante, quiero deseo la búsqueda por tipo de dieta.
- Como restaurante, deseo seleccionar un restaurante para ver sus detalles.
- Como restaurante, deseo ver la ubicación de un restaurante en el mapa.
- Como restaurante, deseo acceder a la carta de un restaurante.
- Como restaurante, deseo ver imágenes del establecimiento.
- Como restaurante, deseo ver las reservas realizadas en mi establecimiento.
- Como restaurante, deseo modificar el aforo de las reservas en una fecha y hora específica.
- Como restaurante, deseo cancelar una reserva.
- Como restaurante, deseo contactar con el administrador del sitio web.

6.1.3. Cliente

- Como cliente, deseo iniciar sesión con mis credenciales.
- Como cliente, deseo modificar la imagen de mi perfil.
- Como cliente, deseo actualizar mis datos personales en el perfil (nombre y apellidos, correo electrónico o contraseña).
- Como cliente, deseo buscar restaurantes por provincia/ciudad, fecha y número de comensales.
- Como cliente, deseo filtrar la búsqueda mediante geolocalización.
- Como cliente, deseo filtrar la búsqueda por alérgenos.
- Como cliente, deseo filtrar la búsqueda por tipo de dieta.
- Como cliente, deseo seleccionar un restaurante para ver sus detalles.
- Como cliente, deseo ver la ubicación de un restaurante en el mapa.
- Como cliente, deseo acceder a la carta de un restaurante.
- Como cliente, deseo ver imágenes del establecimiento.
- Como cliente, deseo añadir reseñas a los establecimientos.
- Como cliente, deseo eliminar mi reseña.
- Como cliente, deseo ver mis reseñas.
- Como cliente, deseo leer las reseñas de clientes sobre el restaurante.
- Como cliente, deseo ver las reservas actuales, pasadas y canceladas.
- Como cliente, deseo cancelar una reserva actual.
- Como cliente, deseo modificar mi lista de favoritos.
- Como cliente, deseo ver mi lista de favoritos.
- Como cliente, deseo contactar con el administrador del sitio web.

6.2. Requisitos del sistema

Los requisitos del sistema comprenden tanto los requisitos funcionales, que están directamente relacionados con las funcionalidades proporcionadas por la aplicación, como los requisitos no funcionales, que influyen en cómo se manejan estas funcionalidades. A continuación, se exponen cada uno de ellos.

6.2.1. Requisitos no funcionales

En este apartado se detallan los requisitos no funcionales, los cuales describen el comportamiento del sistema en diversas condiciones. Estos requisitos son fundamentales para asegurar la eficiencia, seguridad y usabilidad de la aplicación.

Código	Nombre	Descripción	Validación
RNF01	**Conectividad a Internet**	El sistema debe contar con una conexión a Internet óptima.	Se visualizan las provincias/ciudades en la búsqueda de restaurantes.
RNF02	**Disponibilidad del repositorio**	El código fuente de la aplicación debe estar disponible en un repositorio público.	Se tiene acceso al siguiente enlace: https://github.com/kiowaAndueza/tft
RNF03	**Privacidad de datos**	El sistema debe utilizar los datos de los usuarios exclusivamente para fines internos de la aplicación.	Se dispone de acceso a la privacidad de datos a través de los términos de uso y política de la privacidad en la aplicación.
RNF04	**Documentación**	La aplicación debe disponer de información detallada sobre la configuración y manejo de la aplicación.	Se dispone de acceso al manual de usuario a través del siguiente enlace.

RNF05	**Tiempo de respuesta**	El sistema debe ser capaz de mostrar información en 9 segundos como máximo.	La búsqueda de los restaurantes no debe tardar más del tiempo de respuesta establecido.

6.2.2. Requisitos funcionales

Los requisitos funcionales definen los datos necesarios para el correcto funcionamiento de una funcionalidad, así como el comportamiento esperado en base a dichos datos. Los requisitos de la aplicación se detallan en la siguiente tabla.

Código	Nombre	Descripción	Validación
RF01	**Registro**	El sistema debe permitir que el usuario se registre seleccionando el tipo de rol deseado (restaurante o cliente).	Se muestra un mensaje de éxito confirmando que el registro se ha completado correctamente.
RF02	**Registro de restaurante**	El sistema debe permitir que un restaurante se registre ingresando sus datos personales, incluyendo nombre y apellidos, correo electrónico, nombre del restaurante, nombre de usuario, dirección y contraseña.	En caso de que el formulario contenga errores, se indica el error en cada campo con un mensaje de color rojo.
RF03	**Registro de cliente**	El sistema debe permitir que un cliente se registre ingresando sus datos personales, incluyendo nombre y apellidos, correo electrónico, nombre de usuario y contraseña.	La misma validación que **RF02**.
RF04	**Inicio de sesión**	El sistema debe permitir que el usuario registrado inicie sesión mediante correo y contraseña.	Se muestra un mensaje de éxito confirmando que se ha iniciado sesión correctamente.
RF05	**Acceso a los datos**	El sistema debe permitir que el usuario registrado acceda	Se muestran los datos personales en la pestaña

	personales	a los datos personales de su perfil.	de su perfil.
RF06	**Modificar datos personales**	El sistema debe permitir que el usuario registrado modifique sus datos personales.	Se muestra un mensaje de éxito confirmando que se han modificado correctamente.
RF07	**Modificar foto de perfil**	El sistema deberá permitir que el usuario registrado actualice su foto de perfil.	La misma validación que **RF06**.
RF08	**Publicación del restaurante**	El sistema debe permitir que el restaurante publique su establecimiento.	Se muestra un mensaje de éxito confirmando la acción.
RF09	**Modificar la publicación del restaurante.**	El sistema debe permitir que el restaurante modifique los datos de su establecimiento.	Se muestra un mensaje de éxito confirmando que se han modificado correctamente.
RF10	**Búsqueda de restaurantes**	El sistema debe permitir que el usuario realice búsquedas de restaurantes ingresando la provincia/ciudad, fecha, hora y el número de comensales.	Se muestra la lista de coincidencias. En caso de no haber ninguna, debe de aparecer un mensaje indicando que no se encontraron resultados.
RF11	**Filtrar mediante geolocalización**	El sistema debe permitir que el usuario busque restaurantes cercanos a través de su ubicación.	Se activa la ubicación y se aplica el filtro correspondiente.
RF12	**Filtrar mediante tipo de alérgenos**	El sistema debe permitir que el usuario aplique filtros de alérgenos durante la búsqueda.	Se seleccionan uno o varios tipos y se aplica el filtro para ver los resultados.
RF13	**Filtrar mediante tipo de dieta**	El sistema debe permitir que el usuario aplique filtros de diferentes tipos de dietas durante la búsqueda.	La misma validación que **RF12**.
RF14	**Visualizar el listado de restaurantes**	El sistema debe mostrar al usuario el listado resultante de la búsqueda.	La misma validación que **RF10**.
RF15	**Mostar detalles del restaurante**	El sistema debe mostrar al usuario la información detallada del restaurante tal como teléfono de contacto, dirección, imágenes, horario de apertura, tipos de alérgenos y dietas.	Se muestran los detalles del restaurante al seleccionar un establecimiento de la lista de resultados de búsqueda.
RF16	**Mostrar ubicación del restaurante en el mapa**	El sistema debe mostrar al usuario la ubicación aproximada del restaurante en el mapa.	La misma validación que **RF15**.

RF17	**Mostrar carta del restaurante**	El sistema debe mostrar al usuario la carta del restaurante con sus respectivos precios.	La misma validación que **RF15**.
RF18	**Mostrar reseñas clientes**	El sistema debe mostrar al usuario las reseñas asociadas al restaurante seleccionado.	La misma validación que **RF15**.
RF19	**Agregar nueva reseña**	El sistema debe permitir a al cliente autenticado realizar una reseña sobre el establecimiento seleccionado.	Se muestra un botón 'Calificar' al final de la sección de detalles, que abre una ventana para agregar una reseña. Tras completar y enviar correctamente el formulario, aparece un mensaje de éxito informando que la reseña ha sido añadida.
RF20	**Eliminar reseña**	El sistema debe permitir a al cliente autenticado eliminar una reseña existente.	Se muestra un mensaje de éxito si se elimina correctamente la reseña.
RF21	**Mostrar mis reseñas**	El sistema debe permitir a al cliente autenticado visualizar sus reseñas.	Se muestra la lista en el apartado de reseñas y, en caso de no haberlas, debe de aparecer un mensaje que lo indique.
RF22	**Mostrar reseñas de mi restaurante**	El sistema debe permitir al restaurante autenticado visualizar las reseñas sobre su restaurante.	La misma validación que **RF21**.
RF23	**Modificar lista de favoritos**	El sistema debe permitir a al cliente autenticado modificar su lista de favoritos.	Se muestra un mensaje de éxito cada vez que se modifica la lista de favoritos.
RF24	**Agregar nueva reserva**	El sistema debe permitir al cliente autenticado realizar una reserva en el establecimiento introduciendo un horario válido.	Se muestra un mensaje de éxito al realizar la reserva, siempre y cuando la hora seleccionada y el aforo sean compatibles con la disponibilidad del establecimiento.
RF25	**Visualizar reservas de los clientes**	El sistema debe permitir al restaurante autenticado visualizar las reservas de los clientes seleccionando una fecha en el calendario.	Se muestra la lista de reservas en caso de que existan para la fecha indicada. Si no las hubiese para la fecha seleccionada, aparece un mensaje indicándolo.

RF26	**Visualizar reservas en los restaurantes**	El sistema debe permitir al cliente autenticado visualizar sus reservas actuales, pasadas y canceladas.	Se proporciona una lista de opciones que permita al cliente seleccionar el tipo de reserva que desea ver. En caso de que existan reservas de ese tipo, se muestran. Si no hay reservas de ese tipo, aparece mensaje indicándolo.
RF27	**Cancelar reserva por parte del restaurante**	El sistema debe permitir al restaurante autenticado cancelar la reserva de un cliente seleccionando un día concreto en el calendario y haciendo clic en el botón de 'Cancelar'.	Se muestra un mensaje de confirmación antes de cancelar la reserva y luego un mensaje de éxito en caso de cancelarla.
RF28	**Cancelar reserva por parte del cliente**	El sistema debe permitir al cliente autenticado cancelar una reserva actual haciendo clic en el botón de 'Cancelar'.	La misma validación que **RF27**.
RF29	**Ajustar aforo diario del restaurante**	El sistema debe permitir al restaurante autenticado modificar el aforo del establecimiento seleccionando un día concreto en el calendario y una franja horaria.	Se muestra un mensaje de éxito cuando el aforo se modifica correctamente.
RF30	**Contactar al administrador del sitio web**	El sistema debe permitir al usuario enviar un correo electrónico al administrador del sitio web para informar sobre dudas o incidencias.	Se muestra un mensaje de éxito cuando se envía el correo correctamente.

7. Análisis y diseño

7.1. Diagramas de casos de uso

La aplicación consta de tres tipos de usuarios: no registrado, cliente y restaurante. Estos actores se definen en el siguiente diagrama de casos de uso.

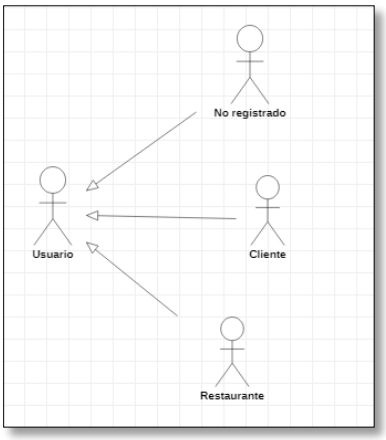

Ilustración 4: Diagrama de casos de uso de los tipos de usuario.

Cada uno de estos roles dispone de diferentes permisos en el sitio web. A continuación, se detalla el diagrama de cada uno de ellos.

7.1.1. No Registrado

Tal como se observa en la ilustración 5, al usuario no registrado se le permite realizar búsquedas de restaurantes, especificando los criterios necesarios: provincia, fecha, hora y número de comensales.

A esta búsqueda se le pueden aplicar opcionalmente diferentes tipos de filtros, que incluyen opciones como tipos de alérgenos, dietas, precio medio por comensal y restaurantes cercanos a la ubicación. Para listar los restaurantes próximos a la zona, es

necesario habilitar el permiso para obtener la ubicación actual del usuario. Una vez completada la búsqueda, este usuario puede seleccionar un restaurante y visualizar su información detallada.

Dentro de los detalles del restaurante, se proporciona información sobre la dirección, la cual se encuentra marcada aproximadamente en el mapa. También se incluye información exhaustiva sobre los tipos de dieta que ofrecen, y opcionalmente, alérgenos. Además, aparece una breve descripción del restaurante, seguida de la carta con sus respectivos precios. Por último, se encuentra la sección de reseñas, donde todos los usuarios pueden leer las experiencias compartidas por otros clientes.

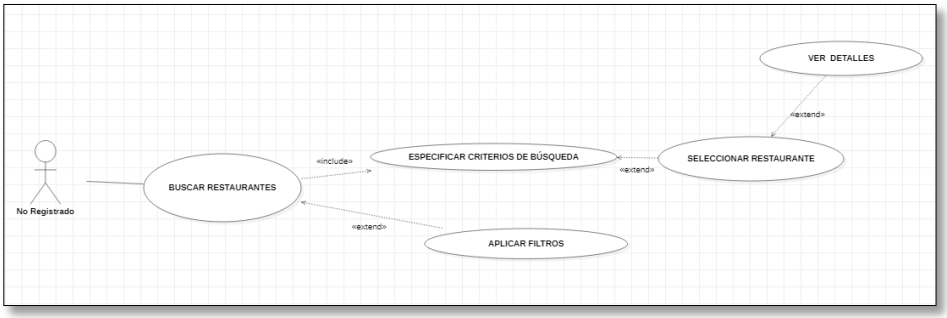

Ilustración 5: Diagrama de casos de uso del usuario no registrado.

7.1.2. Restaurante

El restaurante cuenta con diversas funcionalidades, representadas en la ilustración 6. En primer lugar, tiene la posibilidad de publicar o actualizar la información de su establecimiento, incluyendo obligatoriamente campos como teléfono de contacto, horario de apertura, aforo máximo, tipos de dieta, precio medio por comensal, descripción, dirección, menú con sus respectivos precios e imágenes, y opcionalmente, alérgenos. Es necesario publicar el establecimiento para que el usuario reciba reservas y reseñas, no es suficiente con registrarse en la aplicación.

Además, el restaurante tiene la capacidad para realizar búsquedas de restaurantes de la misma forma que los usuarios no registrados. Por otro lado, en la sección de reservas, el restaurante debe seleccionar una fecha específica para ver los usuarios que disponen reserva en su establecimiento. Desde esa sección, también tienen la opción de ajustar el aforo de su establecimiento al seleccionar una fecha y franja horaria específicas, en caso de que clientes externos a la aplicación busquen reservar a través de otros medios. Asimismo, desde esta página, el restaurante tiene la posibilidad de cancelar reservas realizadas por otros usuarios.

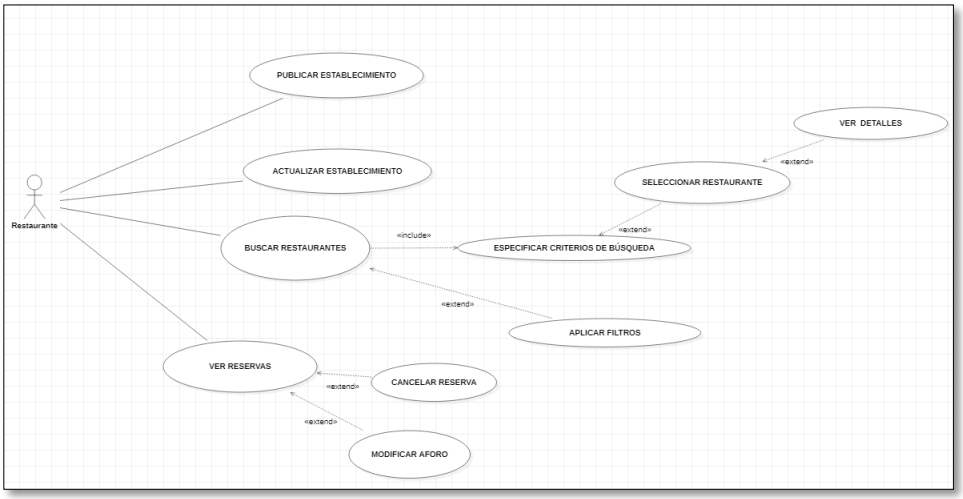

Ilustración 6: Diagrama de casos de uso del restaurante.

7.1.3. Cliente

El cliente disfruta de varios privilegios, los cuales se exponen en la ilustración 7. En primer lugar, comparte la funcionalidad genérica de todos los usuarios: la búsqueda de restaurantes. Además, tiene la capacidad de llevar a cabo operaciones CRUD (Crear, Leer, Actualizar y Eliminar) tanto para las reservas como para la lista de restaurantes favoritos. También puede visualizar y eliminar las reseñas realizadas en cada establecimiento.

Dentro de la sección de reservas, el cliente puede seleccionar el tipo de reserva que desea ver: actuales, pasadas o canceladas. Es

relevante señalar que solo puede cancelar las reservas actuales. Por otro lado, las reservas solo se pueden crear después de realizar la búsqueda y seleccionar el restaurante deseado. Al entrar, aparece la sección de reservas, donde el cliente, seleccionando el número de comensales y eligiendo fecha y hora, puede realizar la reserva en el establecimiento, siempre y cuando este disponga de aforo suficiente para ejecutarla.

Para la gestión de la lista de favoritos, el cliente debe buscar el restaurante que desea añadir a favoritos y seleccionar el icono del corazón. Luego, dispone de una sección propia donde aparecen todos los restaurantes agregados, y desde allí también puede eliminar cualquier establecimiento que desee quitar de la lista.

Finalmente, el cliente, a través del apartado de detalles del restaurante, puede añadir su reseña al establecimiento, así como eliminar la existente. Además, dispone de un apartado propio desde donde también puede ver todas las reseñas añadidas a los establecimientos y eliminar aquellas que desee.

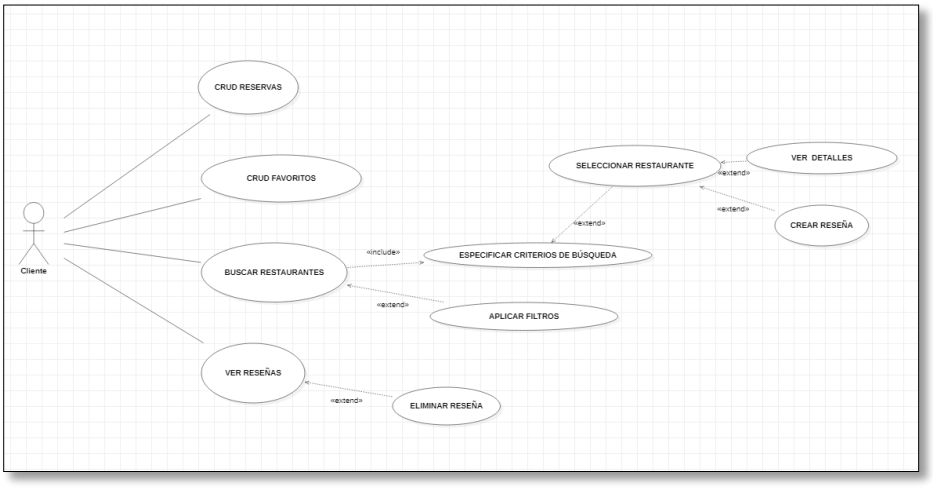

Ilustración 7: Diagrama de casos de uso del cliente.

7.2. *Mockups*

Los *mockups* se diseñan durante la fase inicial del proyecto para obtener una visión preliminar del desarrollo de la interfaz de usuario. Durante la creación de estos diseños, aún no se había seleccionado la paleta de colores que se utiliza en el sitio web. Por lo tanto, en este prototipado se aplican tonos neutrales que no corresponden con los utilizados en la aplicación. A continuación, se presentan algunos de estos diseños.

7.2.1. *Formularios de registro*

El proceso de registro varía en función del tipo de rol, ya que se requieren más datos personales para registrar a un restaurante que para registrar a un cliente. En los *mockups* mostrados en la ilustración 8, no se especifica el nombre de usuario, así como otros campos relacionados con la dirección. Sin embargo, durante la fase de desarrollo se considera añadirlos.

Ilustración 8: Mockups de los formularios de registro.

7.2.2. *Formulario de inicio de sesión*

Como se puede observar en la ilustración 9, los usuarios que ya están registrados tienen la opción de autenticarse mediante el siguiente formulario utilizando su correo electrónico y contraseña.

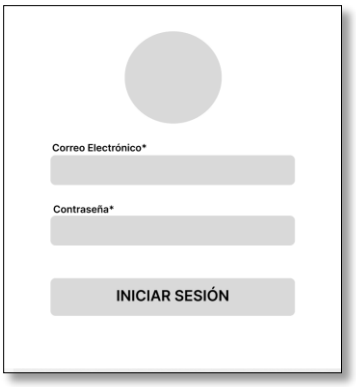

Ilustración 9: Mockup de inicio de sesión.

7.2.3. *Home*

Desde la página de inicio, los usuarios pueden realizar una búsqueda de restaurantes especificando los criterios, que en este caso son la localización, fecha, hora y número de comensales.

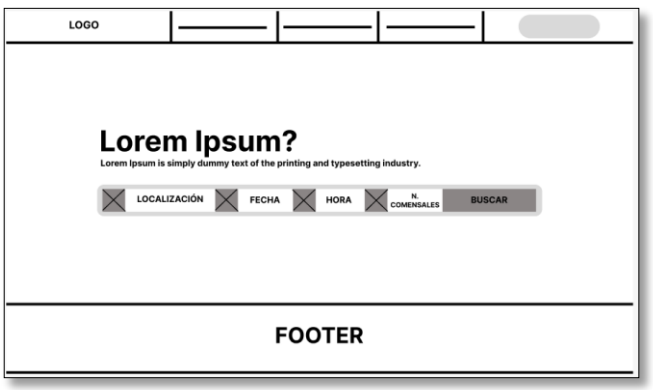

Ilustración 10: Mockup de la página de inicio.

7.2.4. Resultado de la búsqueda

Después de realizar la búsqueda, si existen coincidencias con los criterios introducidos, el resultado debería verse algo similar a la ilustración 11. A la izquierda, se permite realizar una nueva consulta sin tener que volver al inicio. Justo debajo, aparecen los filtros, los cuales no fueron los definitivos, ya que más tarde se añade el filtrado mediante geolocalización a los requisitos. Por último, a la derecha aparecen los restaurantes que coinciden con los criterios especificados. En la imagen de cada uno de ellos, hay un icono de corazón, que solo aparece para los clientes autenticados.

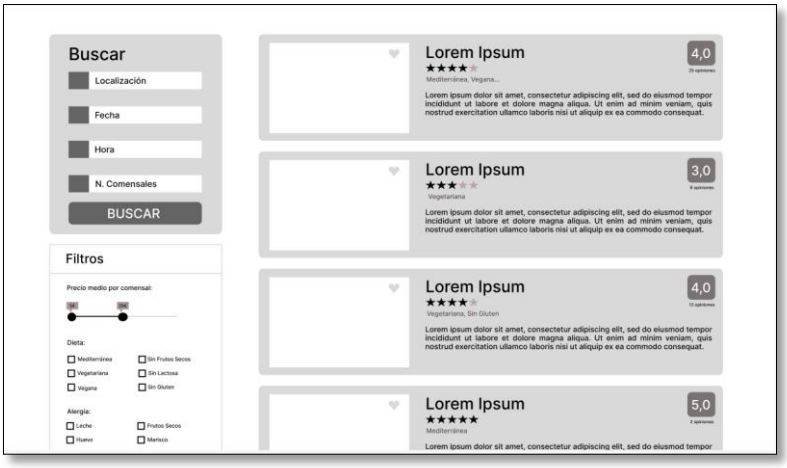

Ilustración 11: Mockup del listado de restaurantes.

7.2.5 Vista del restaurante

Los *mockups* de las Ilustraciones 12 y 13, relacionados con la vista del establecimiento, se muestran en la aplicación cuando se selecciona un restaurante específico a través del listado de búsqueda. Desde ahí, se puede ver su información detallada, reservar mesa e incluso consultar las reseñas al final de la página, tal y como muestra la ilustración 13. Cabe destacar que, aunque este *mockup* es muy similar al desarrollo real, no es exactamente igual.

Ilustración 12: Mockup vista del restaurante 1.

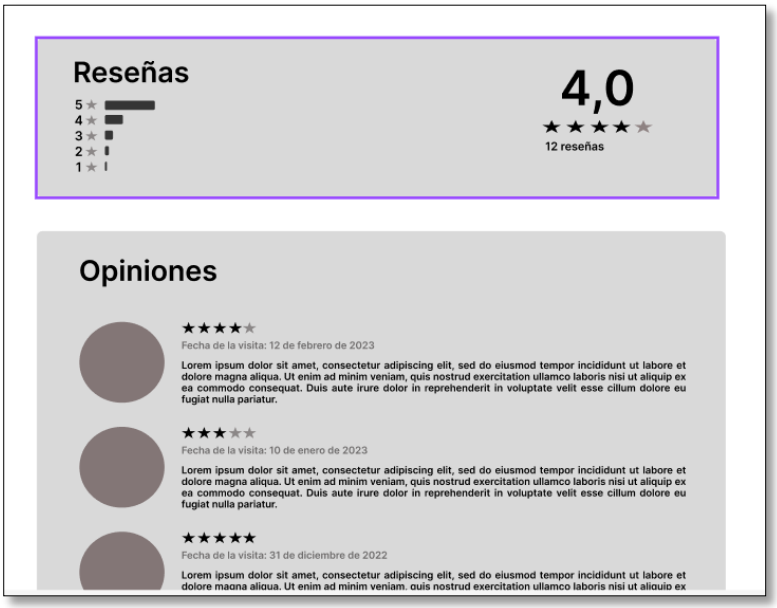

Ilustración 13: Mockup vista del restaurante 2.

7.3. *Flujos de trabajo*

Se han elaborado dos diagramas de flujo con el objetivo de explicar la búsqueda y modificación de la dirección de un establecimiento, ya que se considera que esta funcionalidad es la más compleja de comprender en la aplicación.

7.3.1. Flujo de búsqueda de dirección en el registro

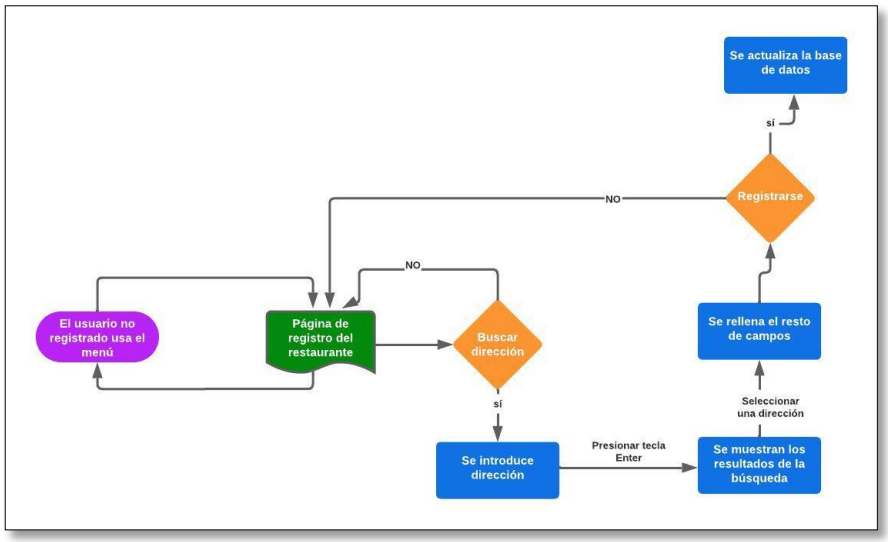

Ilustración 14: Flujo para la búsqueda de dirección.

En esta ilustración, se describe el proceso de búsqueda de la dirección de un establecimiento cuando un usuario desea registrarse como restaurante. Para llevar a cabo esta acción, el usuario debe acceder a la página de registro de restaurantes, donde se le solicita añadir la dirección. Una vez la introduce, debe presionar la tecla "Enter" para visualizar los resultados disponibles. A continuación, el usuario selecciona una de las opciones de la lista de resultados y termina de completar los campos restantes. Finalmente, se debe enviar el formulario para guardar la información correspondiente en la base de datos.

7.3.2. Flujo de modificación de dirección en el perfil

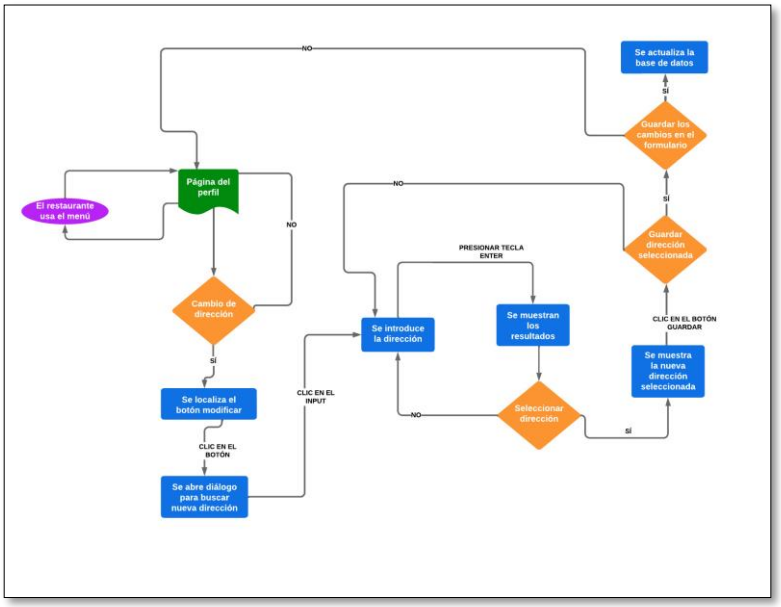

Ilustración 15: Flujo para modificar la dirección.

En la ilustración 15, se detalla el procedimiento para modificar la dirección de un establecimiento. Primeramente, el usuario debe iniciar sesión en la aplicación utilizando sus credenciales de restaurante. Posteriormente, debe acceder a la página de perfil y seleccionar el botón de "Modificar" ubicado en la sección de dirección.

Al hacerlo, se abre un diálogo que permite al usuario buscar y seleccionar la nueva dirección. Después de ingresar la nueva dirección, el usuario debe presionar la tecla ‘Enter’, lo que genera una lista de resultados. A continuación, el usuario debe seleccionar la dirección deseada y guardar el cambio para actualizar el formulario.

Completado este proceso, el usuario procede a llenar cualquier espacio en blanco restante en la sección de dirección y envía el formulario para actualizar la información en la base de datos.

7.4. *Arquitectura*

El tipo de arquitectura que se utiliza para desarrollar la aplicación se explica a continuación. En primer lugar, se implementa el *Model-View-Controller* (MVC). Tal y como se explica en la ilustración 16, el usuario solicita información al controlador a través de la interfaz de usuario mediante una petición, ya sea *GET*, *POST*, *PUT* o *DELETE*. Luego, el controlador se encarga de transmitir dicha petición al modelo para gestionarla y obtener los datos necesarios. Finalmente, estos parámetros son enviados a la vista para que el usuario pueda visualizar la respuesta.

- ***GET*:** Solicita al servidor información específica de la base de datos.
- ***POST*:** Envía al servidor nueva información para ser agregada a la base de datos.
- ***PUT*:** Envía al servidor datos para modificar la información existente en la base de datos.
- ***DELETE*:** Envía al servidor datos para eliminar registros de la base de datos.

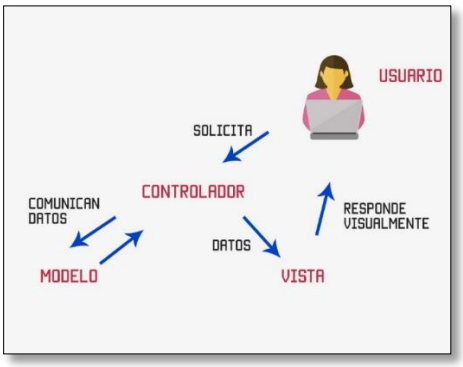

Ilustración 16: Explicación básica del patrón de diseño MVC.

Por otro lado, el patrón MVC se apoya en la *Clean Architecture*. Esta metodología se utiliza para estructurar el código por capas, separando la interfaz de usuario, los casos de uso que controlan la lógica de negocio, las vistas que gestionan las peticiones y las

entidades. Al aplicarla, se definen claramente las responsabilidades, estableciendo que la capa más externa se comunica con la siguiente en la jerarquía, es decir que la dependencia está centrada. Esto se ilustra gráficamente en la ilustración 17.

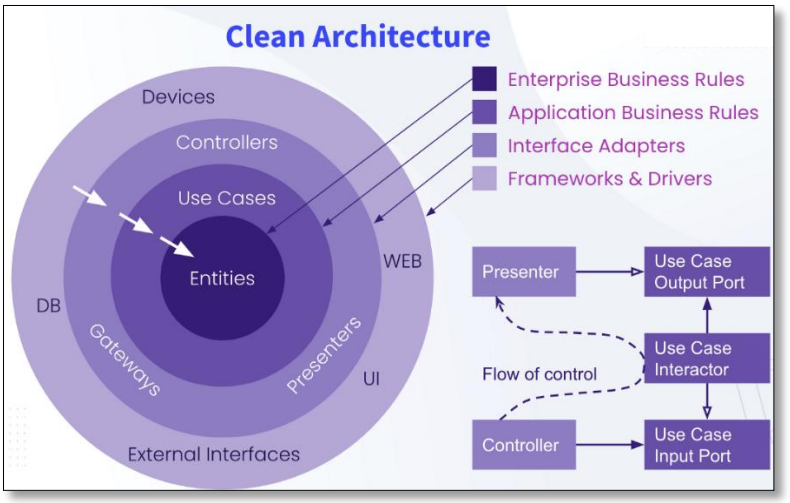

Ilustración 17: Metodología Clean Architecture

Por último, se utiliza un sistema de archivos almacenado dentro de un repositorio llamado "*database*" que se encarga de gestionar la persistencia de los datos. A esta práctica se le denomina patrón *Repository*, el cual separa la lógica de datos de la lógica de negocio.

8. Implementación

8.1. Persistencia

Para lograr la persistencia en la aplicación, se decide utilizar una base de datos no relacional. Google proporciona servicios que permiten almacenar información de forma gratuita en la nube. Este tipo de sistema aporta flexibilidad en la gestión y estructuración de los datos del sitio web.

En este caso, se utilizan tres servicios fundamentales proporcionados por Firebase para guardar y administrar la información del proyecto: Firestore para la base de datos no relacional, Authentication para la autenticación de usuarios y Storage para el almacenamiento de imágenes en la nube.

8.1.1. Configuración

Para configurar la base de datos en el *backend*, es necesario tener instaladas las siguientes bibliotecas mediante el comando "pip install": firebase-admin, google-auth, firebase, google-cloud-firestore y google-cloud-storage.

Una vez instaladas, se copian las credenciales de la base de datos desde la consola del proyecto en Firebase [3], seleccionando el apartado de “Descripción general” y accediendo luego a la configuración del proyecto. Posteriormente, se crea un archivo ".json" para insertar dichas credenciales, como se ilustra en la figura 18.

Ilustración 18: Claves de acceso de Firebase.

Y, por último, se genera una variable de entorno a partir de dicho archivo, la cual se utiliza posteriormente en otra clase para inicializar automáticamente las credenciales a través del fichero "settings.py".

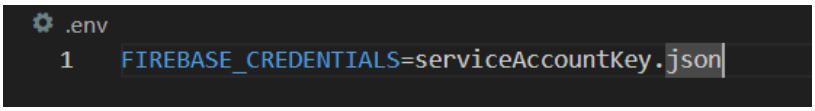

Ilustración 19: Variable de entorno de las credenciales.

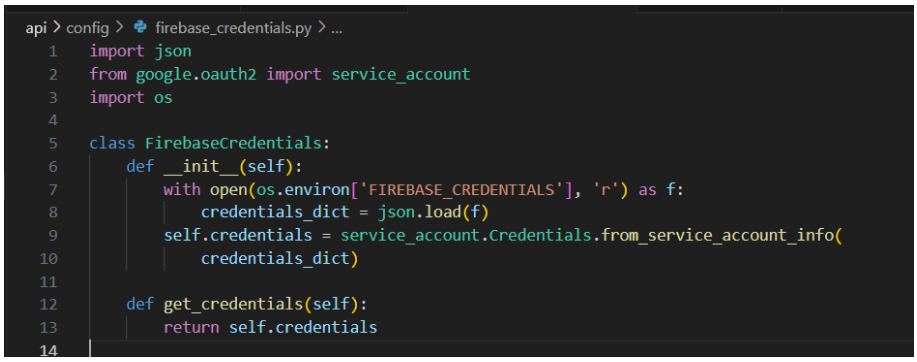

Ilustración 20: Clase encargada de leer las credenciales.

Ilustración 21: Inicialización de la base de datos en setting.py

8.1.2. *Firebase Authentication*

Este servicio se utiliza para validar a los usuarios. A través de él, se emplea el proveedor de correo electrónico y contraseña para crear y confirmar las identidades de los usuarios. Una vez se ha creado la cuenta mediante el sistema de autenticación, los demás datos personales se almacenan en la base de datos de Firestore.

- **Datos almacenados en el sistema de autenticación:**

  - **Correo electrónico:** *Email* proporcionado por el usuario durante el proceso de registro.

  - **Contraseña:** Cadena de caracteres que debe cumplir con ciertos requisitos mínimos de seguridad y se utiliza para acceder a la cuenta.

  - **UID de usuario:** Identificador único asignado aleatoriamente por el sistema a cada usuario registrado.

Identificador	Proveedores	Fecha de creación ↓	Fecha de acceso	UID de usuario
mariachibar@gmail.com	✉	19 nov 2023	19 nov 2023	2YwFk0DN9kVNErwYZBLozmhQO...
rincongastronomico@gmai...	✉	19 nov 2023	19 nov 2023	kB6oTMoNbvbWTGcyvTvpLDhOk9...
lacucharacasera@gmail.co...	✉	19 nov 2023	19 nov 2023	UfMHZ739mIgPHII2ZMHSNQEsky...
manuel123@gmail.com	✉	26 oct 2023	19 nov 2023	uOI4Vh9pZMOkovISP9D112Wa4io2
gastrobar@gmail.com	✉	17 oct 2023	27 oct 2023	Wzr97I0FkqcZ2EWOeXIJmH4thdF2
luisalonsoclavijo@gmail.c...	✉	29 ago 2023	26 nov 2023	DHRxAKR5RFVrICHzRZRZec3eWH...
k.andueza96@gmail.com	✉	29 ago 2023	28 nov 2023	9C0K0MPMZ7dBlqFZbol8lx62iuu2

Ilustración 22: Tabla de autenticación.

8.1.3. Firestore

Firestore es la base de datos no relacional que se utiliza como medio de almacenamiento en la nube para guardar la información que gestiona la aplicación. La estructura de esta base de datos consta de un total de diez colecciones que se detallan a continuación.

- **Colecciones:**
    - "**allergens**": Encargada de enumerar los diferentes tipos de alérgenos.
    - "**cuisines**": Almacena los distintos tipos de dietas.
    - "**myFavoriteList**": Contiene la lista de favoritos de cada cliente.
    - "**myReservations**": Dispone de información sobre las reservas realizadas por el cliente, facilitando el acceso a su historial.
    - "**post**": Almacena los detalles de la publicación de cada restaurante.
    - "**provinces**": Contiene el listado de todas las provincias españolas.
    - "**reservation**": Almacena las reservas de un restaurante. La información está estructurada de manera que cada documento representa una fecha de reserva diferente y está asociada a un único establecimiento.
    - "**reviewsClient**": Alberga el historial de reseñas de cada cliente.
    - "**reviewsRestaurant**": Guarda la lista de reseñas de cada restaurante.

- "**users**": Dentro de esta colección se encuentran dos subcolecciones, "**restaurants**" y "**clients**". En ellas se almacenan los datos personales de cada tipo de usuario.

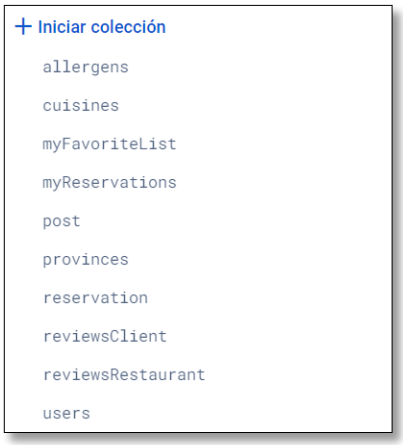

Ilustración 23: Colecciones de la base de datos.

8.1.4. *Firebase Storage*

La gestión del almacenamiento de imágenes de la aplicación se lleva a cabo mediante Firebase Storage. Esta estructura consta de un contenedor de archivos, conocido como "bucket", que abarca dos tipos de directorios. Uno está destinado a alojar las fotos de perfil de los usuarios, mientras que el otro almacena las imágenes de cada establecimiento. Este último directorio incluye subcarpetas declaradas con el UID de cada restaurante, donde se guardan las imágenes correspondientes.

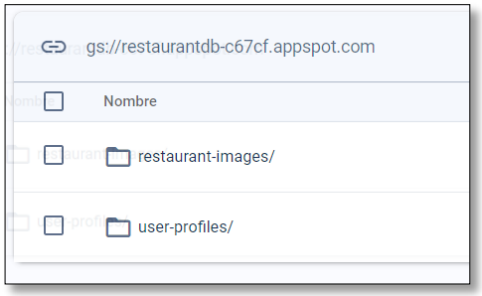

Ilustración 24: Estructura del contenedor.

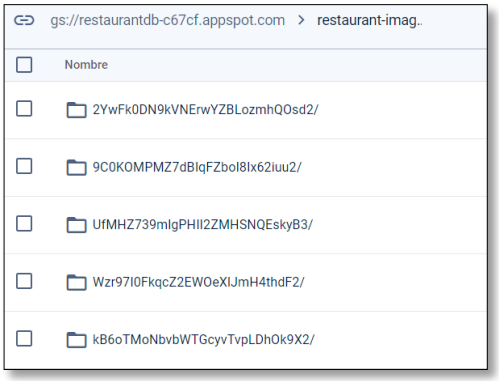

Ilustración 25: Estructura del directorio de las imágenes de los establecimientos.

8.2. Lógica de negocio

En esta sección se detallan los aspectos más importantes del *backend*. El desarrollo de esta parte del proyecto se lleva a cabo utilizando el *framework* Django, basado en el lenguaje de programación Python.

Además, para facilitar la compatibilidad en cualquier equipo, todas las dependencias de este lenguaje se han instalado dentro de un entorno virtual, tal y como se menciona en el manual de usuario [2]. Esto evita la necesidad de modificar o instalar cualquier otra versión del lenguaje.

8.2.1. *Entorno virtual*

Para la instalación del entorno virtual denominado ".env", se sigue como guía el tutorial del apartado 12, "*Virtual Environments and Packages*", que aparece en la documentación oficial de Python [4].

Explicado brevemente, "env" es utilizado para crear entornos virtuales de Python. Al configurar el entorno, se instala por defecto la versión más reciente del lenguaje disponible en el equipo. Esta herramienta se utiliza en el proyecto para facilitar la instalación de

las dependencias que incluye la aplicación.

Para habilitarlo es necesario abrir el terminar y navegar hasta el directorio “Scripts” que se encuentra dentro del entorno virtual. Una vez ahí, se ejecuta el comando "activate" para activarlo.

```
cd /entorno-virtual/Scripts
activate
```

Seguidamente, para incorporar los paquetes dentro del entorno, se ejecuta el siguiente comando.

```
pip install nombre_del_paquete
```

Tras añadir las dependencias deseadas, se guardan los cambios en un archivo, el cual se denomina “requeriments.txt” por convención. Este fichero se utiliza para gestionar el control de versiones de todas las librerías empleadas en el proyecto. Esto asegura que, al implementar la aplicación en otros entornos, se utilicen las mismas versiones.

Para crear o actualizar el fichero con los nuevos paquetes basta con ejecutar:

```
pip freeze > requirements.txt
```

Por último, para desactivar el entorno, es necesario ejecutar la acción que aparece a continuación.

```
deactivate
```

8.2.2. *Geolocalización*

La funcionalidad de filtrado de búsqueda de restaurantes basada en la geolocalización del usuario es una de las principales herramientas de la aplicación. Esta característica tiene la capacidad de ajustar la búsqueda a un rango limitado entre 5 y 50 kilómetros, habilitando la ubicación actual y seleccionando una distancia máxima.

En esta sección, se explica el cálculo de la distancia entre dos puntos, específicamente entre el usuario y el restaurante. La información necesaria para obtener estos resultados proviene de datos proporcionados en un artículo de Geeksforgeeks [6], donde se encuentra la solución a este problema. En dicha página, se explica brevemente la fórmula de Haversine, la cual traza una línea geodésica entre los dos puntos, señalada en ilustración 26 de color rojo. Esta ruta se considera la más corta para calcular distancias.

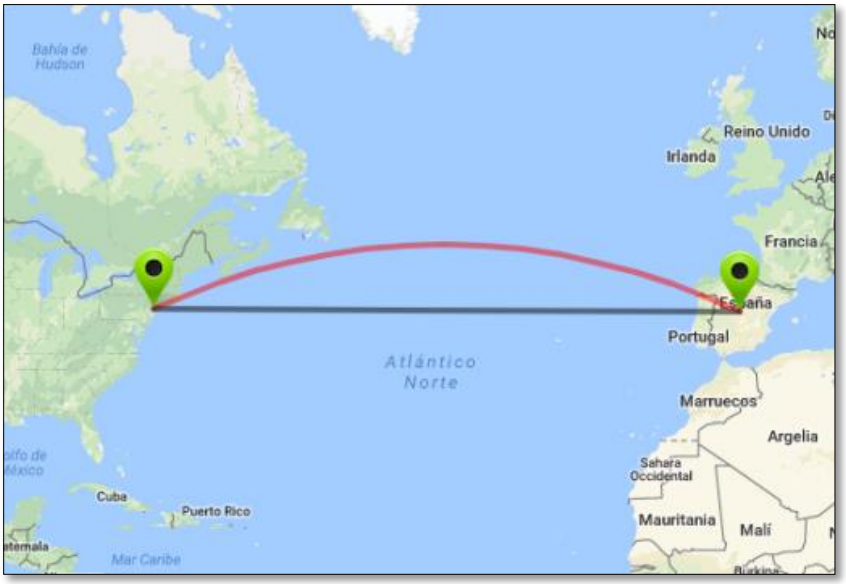

Ilustración 26: Representación de la línea geodésica.

Según la información obtenida, se concluye que la forma más eficiente de calcular la distancia entre dos puntos en la Tierra es utilizando la línea geodésica, como destaca la Sociedad Española de Astronomía (SEA) en su definición de línea geodésica [7]:

> "Sobre una superficie, es la línea de mínima longitud que une dos puntos. El caso más simple correspondería a una superficie plana en la que las geodésicas son líneas rectas. Para superficies más complejas la definición depende de la métrica (es decir, de la forma de la superficie); así, en una superficie esférica la geodésicas son los arcos de los círculos máximos. Por último, en la teoría de la relatividad general las trayectorias de las partículas corresponden a geodésicas trazadas en un espacio-tiempo curvo."

Después de verificar que la solución proporcionada por Geeksforgeeks es la más adecuada para calcular la distancia entre el cliente y el restaurante, se aplica el método correspondiente. En este caso, también se ha tomado como referencia el código proporcionado por el autor, adaptándolo a las necesidades específicas del proyecto.

La fórmula de Haversine utiliza las coordenadas de dos puntos para calcular la distancia. La latitud y la longitud desempeñan un papel fundamental en este proceso. A continuación, se explica matemáticamente dicho cálculo.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Ilustración 27: Fórmula de Haversine.

Donde:

r = Radio de la Tierra = 6371 km

Φ_1 = Latitud en radianes del punto 1

Φ_2 = Latitud en radianes del punto 2

λ_1 = Longitud en radianes del punto 1

λ_2 = Longitud en radianes del punto 2

$\Phi_2 - \Phi_1$ = Distancia de latitud de radianes entre los dos puntos

$\lambda_2 - \lambda_1$ = Distancia de longitud de radianes entre los dos puntos

Una vez comprendida la fórmula matemática para calcular esta distancia, se procede a su implementación en el entorno de programación. Para ello, se utiliza la librería "math" de Python, la cual simplifica el procedimiento.

Posteriormente, se especifica el origen de las coordenadas de ambos usuarios.

```python
import math


def calculate_distance(lat1, lon1, lat2, lon2):
    R = 6371.0

    lat1 = math.radians(lat1)
    lon1 = math.radians(lon1)
    lat2 = math.radians(lat2)
    lon2 = math.radians(lon2)

    distance_lat = lat2 - lat1
    distance_lon = lon2 - lon1

    a = math.sin(distance_lat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(distance_lon/2)**2
    c = 2 * math.asin(math.sqrt(a))

    distance = R * c

    return distance
```

Ilustración 28: Cálculo de distancia en el entorno de desarrollo.

8.2.3. *Gestión de correos*

La aplicación dispone de una funcionalidad que permite a los usuarios enviar correos electrónicos al administrador del sitio web. Esta herramienta se considera útil para plantear dudas o incidencias relacionadas con la plataforma.

Esta característica se implementa utilizando un módulo específico de Django, "django.core.email", que permite la configuración y transmisión de correos electrónico a través de un servidor SMTP (Protocolo Simple de Transferencia de Correo), un protocolo de red que permite gestionar el envío y recepción de mensajes, actuando como intermediario entre el remitente y el destinatario.

El proceso de envío de mensajes se realiza de la siguiente manera:

1. El usuario accede a la página de "Contacto" y completa el formulario con el asunto y el mensaje.
2. Esta información se envía al *backend* a través del controlador.
3. Al recibir los datos, el correo se envía al servidor SMPT, que ha sido configurado previamente en el proyecto, más adelante se explica con detalle.
4. El servidor SMPT se encarga de entregar el mensaje al administrador del sitio web.

De esta forma se facilita la comunicación entre el usuario y el administrador. A continuación, se detalla cómo se configura e implementa este proceso en la aplicación.

En primer lugar, se debe generar la contraseña de aplicación en el correo para que permita la recepción de mensajes a través del sitio web. Esta contraseña es necesaria para luego configurar el archivo "settings.py" con el correo del administrador encargado de recibir los mensajes de los usuarios.

```
43  EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
44  EMAIL_HOST = 'smtp.gmail.com'
45  EMAIL_USE_TLS = True
46  EMAIL_PORT = 587
47  EMAIL_HOST_USER = 'info.gastronet@gmail.com'
48  EMAIL_HOST_PASSWORD = '[redacted]'
```

Ilustración 29: Configuración del correo electrónico.

Los parámetros de configuración de la ilustración anterior tienen las siguientes funciones:

- **EMAIL_BACKEND**: Indica la clase que utiliza Django para transmitir el correo.

- **EMAIL_HOST:** Especifica el servidor SMPT que se utiliza para enviar el mensaje.

- **EMAIL_PORT**: El puerto mediante el cual se transmite el mensaje.

- **EMAIL_HOST_USER**: El correo utilizado para la recepción del mensaje.

- **EMAIL_HOST_PASSWORD**: Contraseña de aplicación que se genera a través del correo.

Por otro lado, la clase "ContactManager" se encarga de enviar el mensaje al correo electrónico que se añade al archivo de configuración del proyecto. Para lograr esto, se utiliza la función "send_email" del módulo mencionado anteriormente. Este método requiere ciertos parámetros, como el asunto, el mensaje, y los correos electrónicos tanto del destinatario como del remitente. El destinatario del correo es **EMAIL_HOST_USER**, configurado previamente.

```python
from django.core.mail import send_mail
from django.conf import settings

class ContactManager:
    def send_email(self, origin, subject, message):
        try:
            sent = send_mail(
                subject,
                f'Mensaje de {origin}:\n\n{message}',
                origin,
                [settings.EMAIL_HOST_USER],
                fail_silently=False,
            )

            if sent > 0:
                return True
            else:
                return False

        except Exception as e:
            error_message = f"Error al enviar el correo: {e}"
            print(error_message)
            return False, error_message, None

```

Ilustración 30: Clase encargada de enviar correos.

8.2.4. *Búsqueda de direcciones*

La búsqueda de direcciones juega un papel esencial en la aplicación, ya que esta herramienta permite a los usuarios localizar un restaurante basándose en la ubicación seleccionada. Por esta razón, se utiliza la API de Nominatim. Según su página oficial [8], esta API se define como un servicio de geolocalización de código abierto que recopila datos de OpenStreetMap y permite encontrar ubicaciones por nombre y dirección.

La documentación de Nominatim señala que, al enviar una petición, la API genera un GeoJSON en respuesta. Esta estructura de datos, basada en JSON (JavaScript Object Notation), proporciona una lista de espacios geográficos donde cada uno es representado con las siguientes características: ciudad, provincia, país, código postal, calle, número, latitud, longitud, etc. A continuación, se explican los parámetros empleados en el diseño de la aplicación y el método para obtenerlos.

- Los parámetros utilizados en la fase de desarrollo son los siguientes:

    - ***address***: Utilizado para extraer una dirección específica a partir de un texto.

    - ***road***: Empleado para obtener el nombre de una calle.

    - ***house_number***: Se utiliza para identificar el número de la vivienda en una calle.

    - ***postcode***: Este parámetro corresponde al código postal de la ubicación.

    - ***state_district***: Se emplea para obtener el nombre de una ciudad.

    - ***city***: En ocasiones, la ciudad se representa con este parámetro o con ***state_district***.

    - ***province***: Utilizado para obtener información sobre la provincia correspondiente.

    - ***latitude*** y ***longitude***: Se utilizan para obtener las coordenadas geográficas precisas de una ubicación.

Antes de desarrollar la lógica de esta funcionalidad, es necesario instalar la librería que permite realizar consultas a Nominatim.

```
pip install geopy
```

Una vez instalado Geopy, se implementa la clase "MapService", la cual tiene como tarea principal obtener un listado de direcciones

coincidentes con un texto de búsqueda proporcionado. En la clase, primero se inicializa un objeto de Nominatim, el cual se utiliza para buscar información geográfica. Luego, se desarrolla el método "get_places", cuya función principal es devolver los lugares que coinciden con el texto ingresado en la búsqueda.

Durante la ejecución de esta función, se realiza una consulta utilizando el objeto Nominatim, limitando la búsqueda exclusivamente al territorio español. Los resultados obtenidos se almacenan en una estructura de datos en formato JSON para facilitar su manejo. Finalmente, el listado obtenido se envía a la interfaz de usuario.

```python
from geopy.geocoders import Nominatim

class MapService:
    def __init__(self):
        self.geolocator = Nominatim(user_agent="my_app", timeout=10)

    def get_places(self, text):
        places_address = self.geolocator.geocode(text, addressdetails=True, country_codes="ES", exactly_one=False)
        print(places_address)

        results = places_address
        data = []

        for place in results:
            if place:
                address_data = {}
                address = place.raw.get('address', {})

                address_data['street'] = address.get('road')
                address_data['number'] = address.get('house_number')
                address_data['cp'] = address.get('postcode')
                address_data['city'] = address.get('city')
                address_data['province'] = address.get('state_district') or address.get('city') or address.get('province')
                address_data['latitude'] = place.latitude
                address_data['longitude'] = place.longitude

                data.append(address_data)
        return data
```

Ilustración 31: Clase encargada de enviar peticiones a Nominatim.

8.2.5. *Implementación del diccionario*

Durante el desarrollo del proyecto, surge un problema al realizar la búsqueda de restaurantes por fecha y hora. El horario de apertura de los establecimientos se almacena en español, mientras que Python transforma la fecha a día de la semana en inglés. Para abordar este problema, se implementa un diccionario que traduce los días al español.

En primer lugar, se convierte la fecha a día de la semana en inglés con "strftime".

```
def check_filters(self, restaurant_id, day, hour, numGuests):
    date_obj = datetime.strptime(str(day), '%Y-%m-%d')

    day_of_week_english = date_obj.strftime('%A')
    day_of_week_spanish = get_day_spanish(day_of_week_english)
```

Ilustración 32: Consulta al diccionario.

Una vez transformada, se realiza una consulta al diccionario encargado de traducirlo, tal como se muestra en la siguiente ilustración.

```
def get_day_spanish(day):
    days_week = {
        'Monday': 'Lunes',
        'Tuesday': 'Martes',
        'Wednesday': 'Miércoles',
        'Thursday': 'Jueves',
        'Friday': 'Viernes',
        'Saturday': 'Sábado',
        'Sunday': 'Domingo'
    }

    return days_week.get(day, day)
```

Ilustración 33: Diccionario para la conversión de días de la semana de inglés a español.

8.3. Interfaz de usuario

La interfaz de usuario se desarrolla utilizando el *framework* React, donde se implementan varias de las funcionalidades esenciales de la aplicación. A continuación, se describe el desarrollo de algunos de los componentes ofrecidos por este sitio web.

8.3.1. *Sistema de autenticación*

En el apartado 8.2 se explica cómo se configuran los servicios de Firebase para la persistencia de datos. A pesar de que el *backend* es responsable de registrar y obtener los datos del usuario, el sistema de autenticación se lleva a cabo a través del *frontend* por motivos de seguridad, evitando así la transmisión de contraseñas a través de diferentes medios.

En primer lugar, es necesario instalar Firebase mediante el siguiente comando:

```
npm install firebase
```

Una vez instalado el módulo, se crea un archivo de configuración que contiene las claves de acceso a la base de datos. La variable "authConfig" almacena las credenciales necesarias. A continuación, 'firebaseApp' se define para inicializar la configuración de Firebase y se genera una instancia del servicio de autenticación llamada "*authentication*". El resultado final se muestra en la ilustración siguiente:

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";


export const authConfig = {
  apiKey: "AIzaSyDXUjaLXojOILuEIShtzTmWR79eApMLGyY",
  authDomain: "restaurantdb-c67cf.firebaseapp.com",
  projectId: "restaurantdb-c67cf",
  storageBucket: "restaurantdb-c67cf.appspot.com",
  messagingSenderId: "519650639726",
  appId: "1:519650639726:web:f9a1843c21101fc8c68b7f",
  measurementId: "G-48C2GNH4G7"
};

export const firebaseApp = initializeApp(authConfig);

export const authentication = getAuth();

export default firebaseApp;
```

Ilustración 34: Fichero de configuración de Firebase.

A continuación, se crea un nuevo archivo responsable de la lógica del sistema de autenticación. En este archivo, se importa la instancia "*authentication*" creada en el paso anterior, junto con los métodos que el módulo "auth" de Firebase proporciona, permitiendo al usuario iniciar y cerrar sesión, así como modificar su contraseña. Además, en esta etapa, se instala el módulo "crypto-js" para realizar la encriptación de contraseñas mediante el algoritmo SHA256.

```
import CryptoJS from "crypto-js";
import { authentication } from "../firebase/firebaseConfig";
import { signInWithEmailAndPassword, signOut, updatePassword } from "firebase/auth";

const hashPassword = (password) => {
  return CryptoJS.SHA256(password).toString();
};
```

Ilustración 35: Importaciones en el sistema de autenticación.

Las funciones proporcionadas por Firebase para la autenticación de credenciales utilizadas en el proyecto son las siguientes:

- **signInWithEmailAndPassword(authentication, email, password)**: Este método permite que el usuario se autentique mediante su dirección de correo electrónico y contraseña. Para utilizarlo de forma adecuada, se debe enviar "*authentication*", junto al *email* y a la *password*.

- **signOut(*authentication*)**: Este método se utiliza para cerrar sesión, y solamente necesita recibir como parámetro la instancia de la autenticación.

- **updatePassword(user, newPassword)**: Este método se utiliza para modificar la contraseña. Para llevar a cabo esta acción, es necesario enviar el usuario que actualmente está identificado. Este dato se obtiene accediendo a la instancia de autenticación de la siguiente manera:

  authentication.currentUser

  También, se debe enviar como segundo parámetro la nueva contraseña.

```
export const login = async (email, password) => {
  try {
    const response = await signInWithEmailAndPassword(authentication, email, hashPassword(password));
    return response.user;
  } catch (error) {
    console.log(error)
  }
};

export const logout = async () => {
  try {
    await signOut(authentication);
    return { status: 200 };
  } catch (error) {
    throw error;
  }
};


export const changePassword = async (newPassword) => {
  const user = authentication.currentUser;
  try {
    await updatePassword(user, hashPassword(newPassword));
    return newPassword;
  } catch (error) {
    console.log(error)
  };
}
```

Ilustración 36: Implementación de los métodos de autenticación.

Tras la implementación de estas funciones, quedan listas para ser utilizadas por otros componentes.

8.3.2. *Componente de mapa*

Para mejorar la experiencia del usuario, se integra en el proyecto la biblioteca Leaflet. Leaflet es una herramienta de JavaScript de código abierto que facilita la visualización de mapas en la aplicación. Para instalar la dependencia:

```
npm install react-leaflet
```

Por otro lado, la documentación oficial [9] de este módulo explica cómo se crea y personaliza este componente. Los pasos realizados en este proyecto son los siguientes:

1. Instalar la dependencia anterior.

2. Cargar la hoja de estilos que facilita la visualización del mapa en la aplicación.

```
import “leaflet/dist/leaflet.css”;
```

3. Importar los componentes que se van a utilizar.

```
import { MapContainer, TileLayer,
CicleMarker, Popup } from “react-leaflet”;
```

   - **MapContainer**: Componente principal del mapa que envuelve a otros componentes relacionados con la visualización.
   - **TileLayer**: Componente utilizado para cargar y mostrar las capas del mapa.
   - **CicleMarker**: Componente encargado de marcar con un círculo la zona por la que se encuentra la ubicación.
   - **Popup**: Componente interactivo que muestra el nombre, en este caso, del establecimiento.

4. Crear la función que devuelve el componente.
   - Esta función debe recibir las coordenadas y el nombre del restaurante. Según la sección 8.2.4, el método utiliza latitud y longitud provenientes de Nominatim. En el *backend*, esta información se almacena en la base de datos y luego se extraen de ahí para enviarlos al *frontend*.

5. Definir la variable de posición que incluye las coordenadas recibidas por parámetros.

6. Declarar el contenedor del mapa.
   - Se define el contendor utilizando la siguiente etiqueta y se establecen sus propiedades: *zoom*, posición inicial, tamaño del marco, etc.

```
<MapContainer
      center={position}
      zoom={13}
      minZoom={8}
      maxZoom={15}
      scroolWheelZoom={false}
      style={{ width: “100%”, height: “100%”}}
      >

      Contenido

</MapContainer>
```

7. Integrar dentro del contenedor el resto de los componentes.
   - Se integran “TileLayer”, “CicleMarker” y “Popup” dentro del contendedor del mapa.

Después de seguir correctamente estos pasos, el resultado final del componente se muestra en la ilustración 37.

```jsx
import React from "react";
import { MapContainer, TileLayer, CircleMarker, Popup } from "react-leaflet";
import "leaflet/dist/leaflet.css";

export const MapComponent = ({ latitude, longitude, name }) => {
  if (latitude === null || longitude === null) {
    return <div>No hay coordenadas disponibles</div>;
  }

  const position = [latitude, longitude];

  return (
    <div>
      <div style={{ width: "100%", height: "20em" }}>
        <MapContainer
          center={position}
          zoom={13}
          minZoom={8}
          maxZoom={15}
          scrollWheelZoom={false}
          style={{ width: "100%", height: "100%" }}
        >
          <TileLayer
            attribution='© <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          />
          <CircleMarker
            center={position}
            color="red"
            radius={30}
          >
            <Popup>{name}</Popup>
          </CircleMarker>
        </MapContainer>
      </div>
    </div>
```

Ilustración 37: Componente de Mapa.

8.3.3. *Componente de reseña*

El componente de reseña permite a los clientes autenticados agregar su valoración a un establecimiento específico. Al final de la sección de detalles del restaurante, los clientes tienen la opción de utilizar esta funcionalidad a través de un botón que abre una ventana para añadir la reseña.

Para implementar este componente, se deben instalar dos bibliotecas: "react-bootstrap" y "react-rating-stars-component". La primera se utiliza para emplear el componente "Modal", que posibilita la apertura del cuadro donde se agrega la calificación y el comentario. La segunda se utiliza para valorar el restaurante mediante un sistema de calificación de estrellas.

A continuación, se proporcionan los comandos para instalar ambas bibliotecas.

```
npm install react-bootstrap
```

```
npm install react-rating-stars-component
```

La información necesaria para implementar el componente modal se obtiene del sitio web de React Bootstrap [10]. En él se especifica la estructura que debe seguir este componente, el cual consta de una cabecera, un título, un cuerpo y un pie de página.

En este caso, se establece que, por defecto, el contenido dentro de la etiqueta "Modal" debe estar oculto hasta que se presione el botón de calificar. La cabecera incluye el título y el botón para cerrar el modal. En el cuerpo, se implementa el sistema de calificación de estrellas y el comentario del usuario. Finalmente, en el pie de página se encuentra el botón "Guardar".

```
<Modal show={show} onHide={handleClose}>
      <Modal.Header closeButton>
        <Modal.Title>Reseña</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <div>
          <div>
            <label>Calificación (*)</label>
            <div>
              <Rating
                count={5}
                size={30}
                value={rating}
                onChange={handleRatingChange}
              />
            </div>
          </div>
        </div>
        <div className="errorMessage">{errorMessages.rating}</div>

        <div className='row mt-3'>
          <label className='col-12'>Comentario (*)</label>
          <textarea
            className='col-11 ms-2 mt-3 rating-textarea'
            placeholder="Escribe tu comentario..."
            rows="8"
            value={comment}
            onChange={handleCommentChange}
          />
          <div className="errorMessage">{errorMessages.comment}</div>
        </div>
      </Modal.Body>
      <Modal.Footer>
        <button type="button" className="btn btn-primary col-12 btn-
rating" onClick={handleSave}>
          Guardar
        </button>
      </Modal.Footer>
    </Modal>
```

Ilustración 38: Componente de reseña.

En la ilustración anterior se muestra el componente Rating, el cual es fácil de comprender con el respaldo de la documentación de Material UI [11]. Dentro de la etiqueta "Rating", se encuentran los parámetros que se enumeran a continuación:

- ***count***: Número de estrellas.

- ***size***: Tamaño del componente.

- ***value***: Número de estrellas seleccionadas.

- ***onChange***: Evento encargado de modificar el valor de la calificación.

Con ambos componentes configurados, se consigue implementar el sistema de reseñas, facilitando a los usuarios la posibilidad de compartir sus experiencias en los establecimientos.

8.3.4. *Componente de filtrado*

En la interfaz de usuario, el desarrollo del filtrado de búsqueda es una de las tareas más complejas, ya que consta de cuatro componentes que mejoran la experiencia del usuario.

En primer lugar, se encuentra el filtrado basado en el precio medio por comensal. Para implementarlo, se utiliza un campo en el formulario de tipo rango que abarca desde 5 hasta 200 euros (€).

```
<div className="mb-1 title-filter-text">Precio medio por comensal:</div>
<div className="input-group mb-3">
  <div className="range-label mf-1 me-2">5€</div>
  <input
    type="range"
    className="form-range custom-range"
    id="customRange"
    min="5"
    max="200"
    step="1"
    value={rangeValue}
    onChange={handleRangeChange}
    title={`${rangeValue}€`}
  />

  <div className="range-label ms-2">200€</div>
</div>
```

Ilustración 39: Filtrado por el precio medio.

En segundo lugar, se encuentra el filtrado de búsqueda mediante la ubicación del usuario. Para obtenerla, se utiliza la API de *Geolocalization* [12], la cual permite al usuario compartir su ubicación. Por motivos de privacidad, se solicita su permiso antes de obtenerla. Estas coordenadas se utilizan junto al rango de distancia en kilómetros que establece el usuario para mostrar los restaurantes cercanos. El rango se puede limitar entre 5 y 50 kilómetros.

La función 'requestGeolocationPermission' gestiona la obtención del permiso y la ubicación del usuario a través de la API. Cuando la geolocalización está desactivada, se muestra un mensaje solicitando al usuario su activación. Una vez activada, se utiliza 'navigator.geolocation.getCurrentPosition' para obtener la posición actual. Estas coordenadas se emplean para actualizar el estado y notificar al componente mediante 'locationToParent' sobre la modificación. Por el contrario, si la geolocalización ya está habilitada, se le ofrece al usuario la posibilidad de desactivarla.

```
const requestGeolocationPermission = () => {
    if (!isGeolocationEnabled) {
      confirmationMessage("¿Desea habilitar la geolocalización?").then(
        (result) => {
          if (result.isConfirmed) {
            if ("geolocation" in navigator) {
              navigator.geolocation.getCurrentPosition(
                function (position) {
                  const latitude = position.coords.latitude;
                  const longitude = position.coords.longitude;
                  setLocation({ latitude, longitude, rangeValueKm });
                  locationToParent();
                },
                function (error) {
                  console.error("Error al obtener la ubicación:", error);
                }
              );
              setIsGeolocationEnabled(true);
            } else {
              console.error(
                "Geolocalización no está disponible en este navegador."
              );
            }
          }
        }
      );
    } else {
      confirmationMessage("¿Desea desactivar la geolocalización?").then(
        (result) => {
          if (result.isConfirmed) {
            setIsGeolocationEnabled(false);
            setLocation(null);
            locationToParent();
          }
        }
      );
    }
  };
```

Ilustración 40: Función que habilita/deshabilita la ubicación del usuario.

La lógica de la ilustración 40 se utiliza en el componente encargado de mostrar al usuario los establecimientos próximos a sus coordenadas. El componente dispone de un icono que llama a esa función. En caso de activar la localización, se muestra el rango anteriormente mencionado para definir la distancia máxima a la que quiere tener el restaurante.

```
<div>
      <div className="row">
        <div className="row">
          <div className="col-3 d-flex align-items-center title-filter-
text">
            Ubicación:
          </div>
          <div className="col-9">
            <FaMapMarkerAlt
              className={`location-icon ${
                isGeolocationEnabled ? "enabled-icon" : "disabled-icon"
              }`}
              onClick={requestGeolocationPermission}
            />
          </div>
        </div>

        {isGeolocationEnabled ? (
          <div>
            <div className="input-group mb-4">
              <div className="range-label mf-1 me-2">5km</div>
              <input
                type="range"
                className="form-range custom-range"
                id="customRange"
                min="5"
                max="50"
                step="1"
                value={rangeValueKm}
                onChange={handleRangeChangeKm}
                title={`${rangeValueKm}km`}
              />

              <div className="range-label ms-2">50km</div>
            </div>
          </div>
        ) : (
          <div className="col-11 mb-3 text-location-filter">
            Haga clic en el ícono para habilitar la geolocalización.
          </div>
        )}
      </div>
    </div>
```

Ilustración 41: Filtrado por geolocalización.

Por último, se encuentra el filtrado de alérgenos y el filtrado de dietas, ambos funcionan de la misma forma. Se extrae el listado de la base de datos y se genera una lista de opciones que permite al usuario seleccionar una o varias de ellas.

```
<div className="mt-3 mb-1 title-filter-text">Cocina:</div>
        <div className="mb-4 types-filter row">
          {availableCuisines.map((cuisine, index) => (
            <div className="col-6" key={index}>
              <div className="form-check">
                <input
                  className="form-check-input"
                  type="checkbox"
                  id={`cuisine-${index}`}
                  value={cuisine}
                  onChange={handleCuisineChange}
                />
                <label
                  className="form-check-label"
                  htmlFor={`cuisine-${index}`}
                >
                  {cuisine}
                </label>
              </div>
            </div>
          ))}
        </div>

        <div className="mb-1 title-filter-text">Alérgeno:</div>
        <div className="mb-4 types-filter row">
          {availableAllergens.map((allergen, index) => (
            <div className="col-6" key={index}>
              <div className="form-check">
                <input
                  className="form-check-input"
                  type="checkbox"
                  id={`allergen-${index}`}
                  value={allergen}
                  onChange={handleAllergenChange}
                />
                <label
                  className="form-check-label"
                  htmlFor={`allergen-${index}`}
                >
                  {allergen}
                </label>
              </div>
            </div>
          ))}
        </div>
```

Ilustración 42: Filtrado por tipos de alérgenos y dietas.

Los componentes anteriores conforman el sistema de filtrado de búsqueda, donde el usuario puede seleccionar uno o varios para personalizar los resultados según sus necesidades.

8.3.5. *Diseño responsivo*

La aplicación cuenta con un diseño responsivo que facilita la adaptación del contenido a diversos dispositivos de distintos tamaños. Tanto Bootstrap como CSS han desempeñado un papel fundamental en el desarrollo de este diseño.

En Bootstrap, la adaptación de las columnas en diferentes dispositivos se realiza mediante el sistema de cuadrícula (*grid*). Este sistema utiliza clases para definir la estructura de las columnas en distintos tamaños de pantalla.

A continuación, se presentan los prefijos de las clases junto con los tamaños máximos de pantalla. Es importante destacar que Bootstrap siempre divide el contenedor web en 12 columnas, independientemente de su tamaño.

	Dispositivo extrapequeño <576px	**Dispositivo pequeño ≥576px**	**Dispositivo mediano ≥768px**	**Dispositivo grande ≥992px**	**Dispositivo extragrande ≥1200px**
Prefijo de la clase	col-	col-sm-	col-md-	col-lg-	col-xl-
Ancho máximo del contenedor	Ninguno (automático)	540px	720px	960px	1140px

Los prefijos de cada clase deben ir acompañados por la cantidad de columnas que se desean ocupar en la web, con un máximo de 12 columnas. Este sistema *grid*, combinado con el diseño responsivo que ofrece CSS, permite ofrece una buena experiencia de usuario.

8.3.6. *Interfaz resultante*

En esta sección se muestra el resultado final de la interfaz de usuario de algunas de las páginas, tanto en dispositivos móviles como en ordenadores.

Ilustración 43: Home en dispositivos grandes.

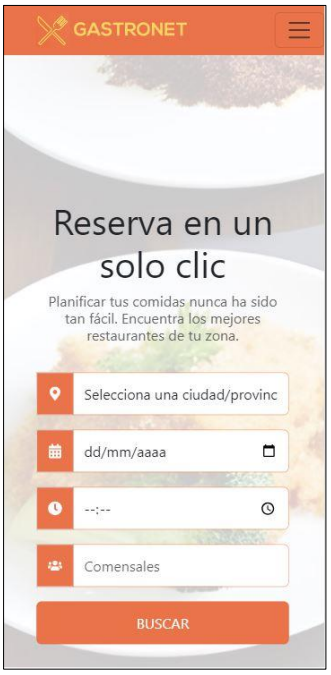

Ilustración 44: Home en dispositivos pequeños.

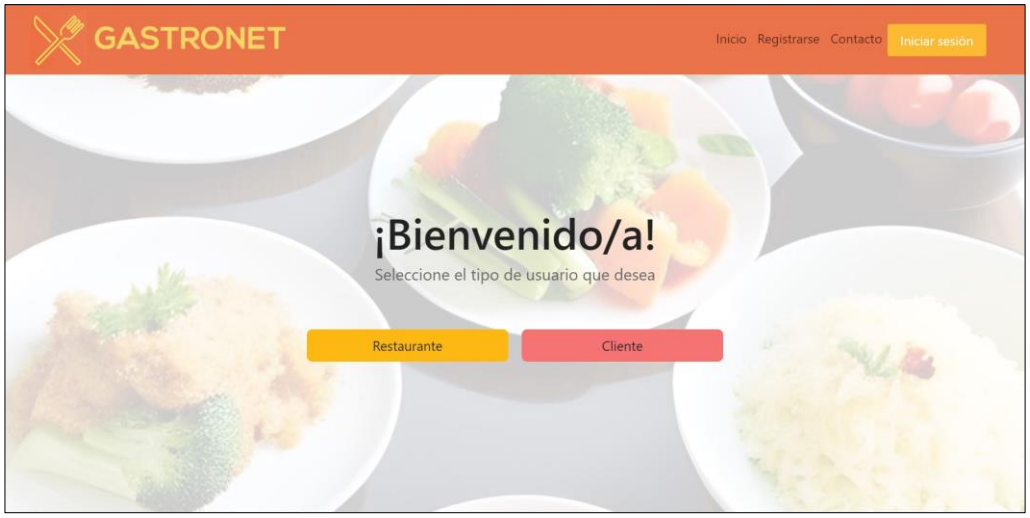

Ilustración 45: Seleccionar tipo de registro en dispositivos grandes.

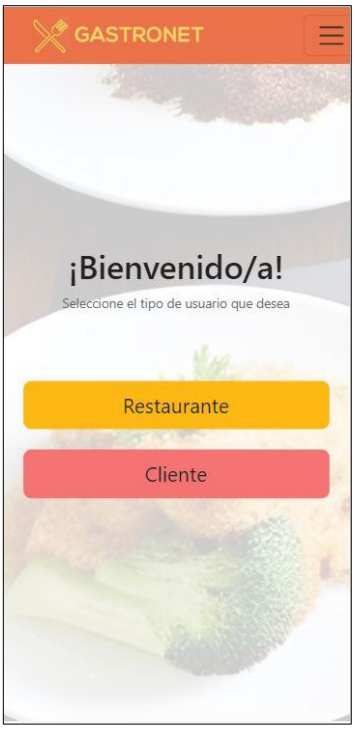

Ilustración 46: Seleccionar el tipo de registro en dispositivos pequeños.

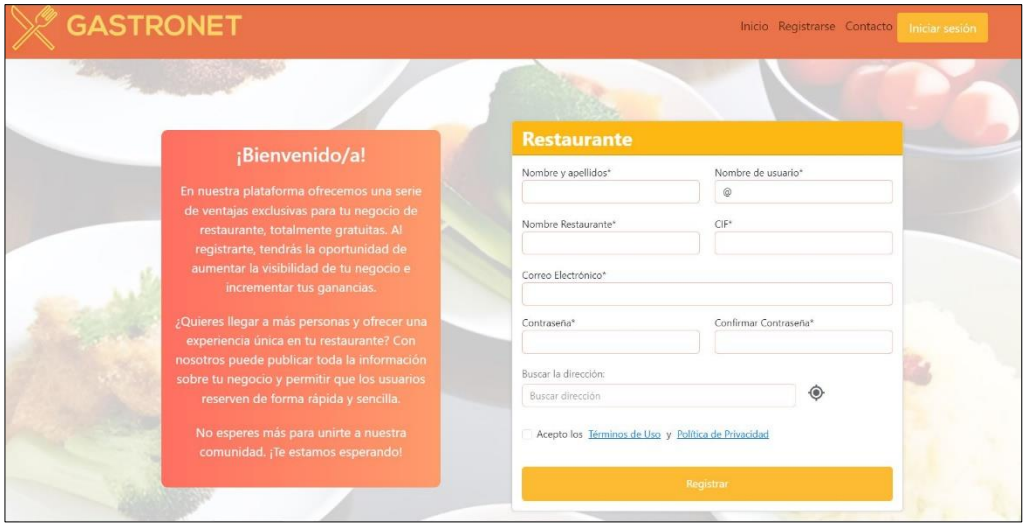

Ilustración 47: Formulario de registro del restaurante en dispositivos grandes.

Ilustración 48: Formulario de registro del restaurante en dispositivos pequeños.

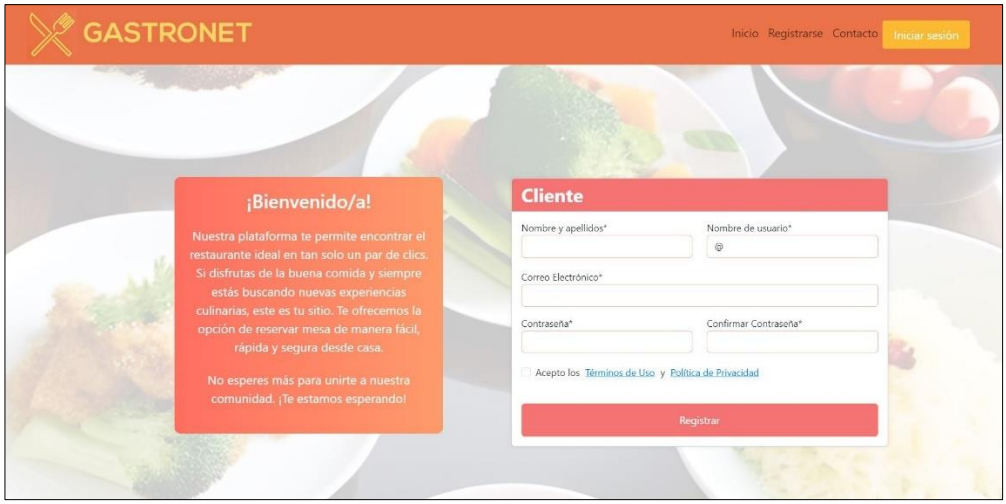

Ilustración 49: Formulario de registro del cliente en dispositivos grandes.

Ilustración 50: Formulario de registro del cliente en dispositivos pequeños.

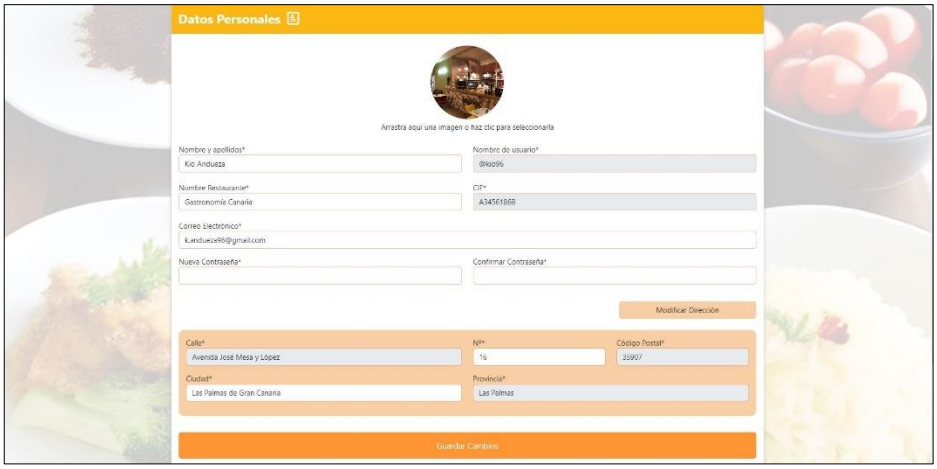

Ilustración 51: Perfil del restaurante en dispositivos grandes.

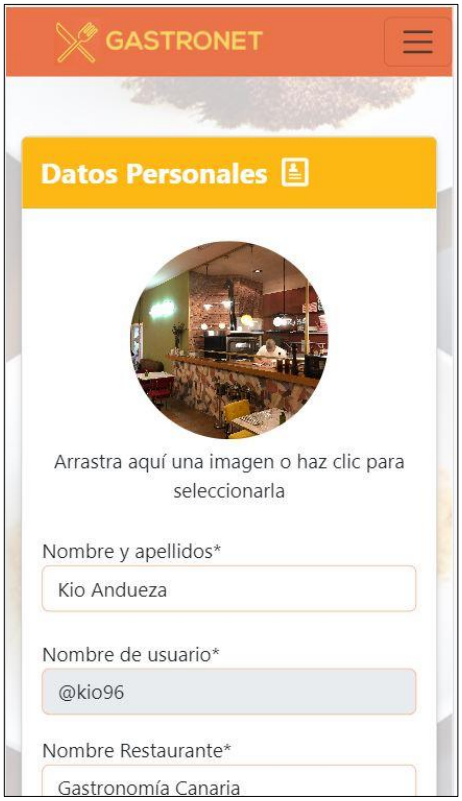

Ilustración 52: Perfil del restaurante en dispositivos pequeños.

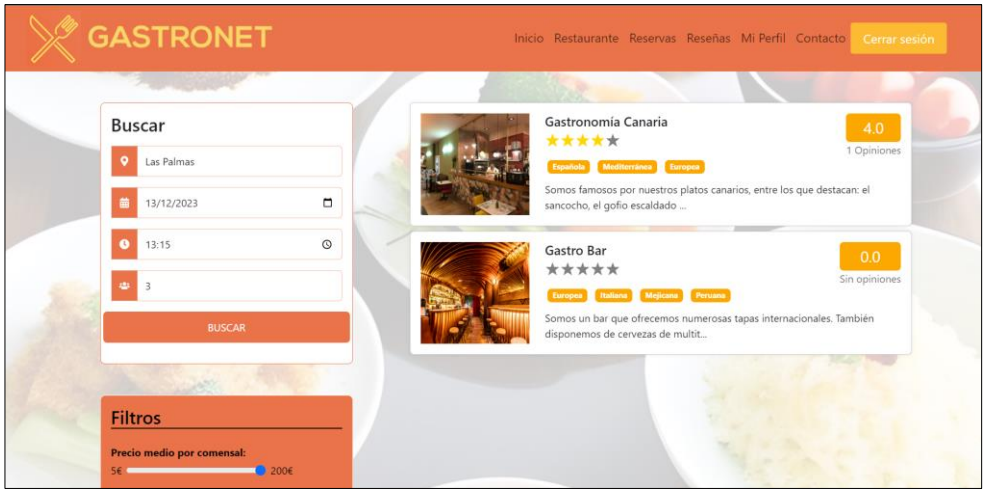

Ilustración 53: Listado de restaurantes en dispositivos grandes.

Ilustración 54: Listado de restaurantes en dispositivos pequeños.

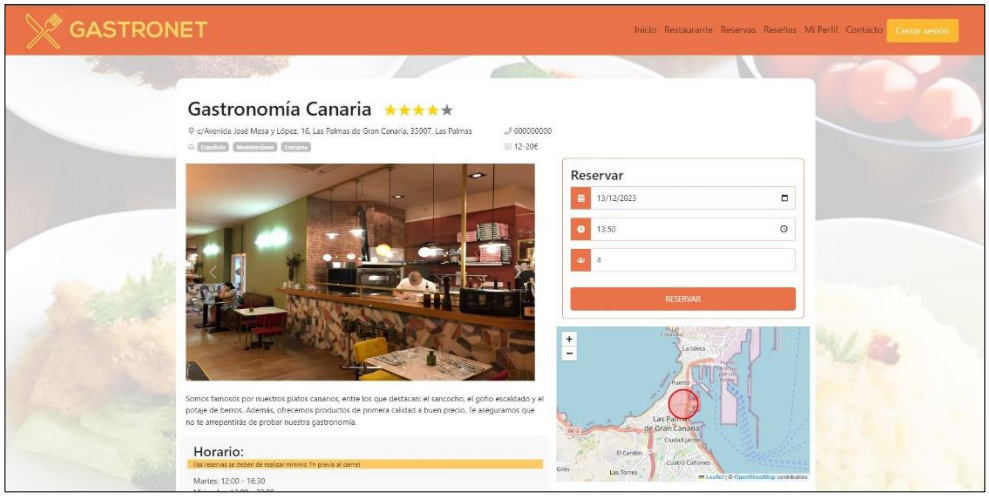

Ilustración 55: Información del restaurante en dispositivos grandes.

Ilustración 56: Información del restaurante en dispositivos pequeños.

9. Perspectivas de mejora

Desde el principio, se supo que las funcionalidades de la aplicación estarían limitadas por los recursos disponibles. Además del tiempo, las APIs y herramientas utilizadas han sido totalmente gratuitas. Esto implica que la aplicación tiene cierto margen de mejora en cuanto a las funcionalidades. A continuación, se proponen algunas de ellas.

- **Mejorar la búsqueda de direcciones**: La API de Nominatim, en algunos casos, no dispone de ciertas direcciones o no proporciona una exactitud óptima. Por esta razón, se propone la API de Google Maps como alternativa para mejorar esta funcionalidad, ya que por ser herramienta de pago no se integra en el proyecto.

- **Implementación de un sistema de mensajería**: Como se menciona en secciones anteriores, la incorporación de un sistema de mensajería facilitaría la comunicación directa entre los clientes y los restaurantes. A través de este sistema, los usuarios podrían resolver dudas o realizar solicitudes especiales relacionadas con las reservas. Esta funcionalidad mejoraría la experiencia de usuario, ya que la aplicación serviría como canal de comunicación.

- **Mejorar el sistema de reserva**: La aplicación actualmente cuenta con un sistema de reserva en el que los clientes no tienen la capacidad de modificar una reserva existente. La propuesta para una mejora futura incluye la posibilidad de realizar modificaciones en reservas ya existentes, en lugar de tener que cancelar y crear una nueva. Además, se sugiere la implementación de un sistema de verificación que brinde mayor seguridad a las reservas realizadas por los clientes y proporcione mayor confianza a los restaurantes.

- **Desplegar la aplicación**: Durante el desarrollo del proyecto, se intenta realizar el despliegue de la aplicación, sin embargo, surgieron inconvenientes, ya que muchas plataformas de servicios en la nube son de pago. No obstante, con la disponibilidad de recursos económicos se puede lograr.

10. Conclusiones

Desde la fase inicial del proyecto, he adquirido conocimientos fundamentales en la gestión y organización de recursos. Ha sido necesario planificar cada tarea previamente para ajustarse al tiempo disponible. Además de este aprendizaje, el proyecto me ha brindado la oportunidad de adquirir conocimientos en el ámbito tecnológico, donde he podido darme cuenta de las grandes ventajas que ofrece utilizar una base de datos no relacional. También, he podido familiarizarme más con la documentación oficial de las herramientas implementadas en el proyecto y he mejorado mi habilidad en la resolución de problemas, lo que ha hecho que le dé más importancia a la depuración cuando aparecen errores de ejecución en el código.

La creación de un sitio web centrada en el usuario en un plazo tan ajustado ha sido un desafío interesante para mí, dada la amplitud de aspectos a considerar en la aplicación. La fase de pruebas, por ejemplo, es un elemento destacado en este proceso. Aunque se han llevado a cabo varias de ellas, siempre existe la posibilidad de profundizar más en esta área. Como menciono anteriormente, el tiempo disponible limitaba la cantidad de tareas que se podrían haber abordado.

En conclusión, pese a las circunstancias, he logrado cumplir los objetivos marcados inicialmente. Ha sido una experiencia gratificante que no solo me ha proporcionado una gran cantidad de conocimientos, sino que también he podido desarrollar y mejorar aptitudes y habilidades.

11. Bibliografía

[1] RTVE (2023). Alimentos, gasolina, hipotecas y pensiones: ¿cuáles son las subidas que llegan en 2023? https://www.rtve.es/noticias/20230101/sube-2023-espana-alimentos-peajes-energia/2413073.shtml

[2] Andueza, K (2023). Manual de usuario. https://drive.google.com/file/d/165XrYfF4l-6S72W78O4tudWxbcydVzXg/view?usp=drive_link

[3] Firebase. Documentación oficial de Firebase. https://firebase.google.com/?hl=es

[4] Python. Documentación oficial de Python. https://docs.python.org

[5] Django. Documentación oficial de Django. https://docs.djangoproject.com/en/4.2/

[6] Geeksforgeeks. (2022). Haversine *formula to find distance between two points on a sphere*. Sitio oficial de Geeksforgeeks. https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/

[7] Sociedad Española de Astronomía. Geodésica. Sitio oficial de la SEA. https://www.sea-astronomia.es/glosario/geodesica

[8] Nominatim. Documentación oficial de Nominatim. https://www.nominatim.org/

[9] Leaflet. Documentación oficial de React Leaflet. https://react-leaflet.js.org/

[10] React Bootstrap. Documentación oficial de React Bootstrap. https://www.react-bootstrap.netlify.app/

[11] Material UI. Documentación oficial de Material UI.
https://mui.com/

[12] Mozilla *Developer Network*. *Geolocation* API. Documentación oficial de Mozilla.
https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API

[13] Andueza, K. (2023). Código fuente de la aplicación. Sitio oficial de GitHub.
https://github.com/kiowaAndueza/tft