



Universitat de Lleida

TREBALL FINAL DE GRAU



ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Tiberiu Ionut Paiu

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: DeliverEat: diseño e implementación de una aplicación de reparto de comida a domicilio

Director/a: David Sarrat González

Presentació

Mes: Setembre

Any: 2024

Índice de contenidos

Introducción	4
Presentación del TFG	4
Objetivos del Proyecto	4
Investigación Previa	5
Análisis de diferentes frameworks web (Django, Spring Boot, CodeIgniter)	5
Django	5
Spring Boot	6
CodeIgniter	6
Definición del Proyecto	7
Objetivos del Proyecto	7
Objetivo Principal	7
Objetivos Específicos	7
Modelo de negocio	8
Análisis de Riesgos	9
Identificación de Riesgos	9
Evaluación de Riesgos	10
Plan de Contingencia	11
Planificación del Proyecto	12
	12
Conclusión sobre la Rentabilidad y Viabilidad del Proyecto	13
Análisis de Requisitos	13
Método de Obtención de Requisitos	13
Definición de Perfiles de Usuario y Necesidades	13
Análisis de Aplicaciones Similares	14
Identificación de Requisitos de Plataforma	15
Integraciones de Sistemas Externos	16
Identificación de Requisitos	17
Requisitos Funcionales	17
No Funcionales	19
Definición de Casos de Uso	22
Casos de Uso del perfil del Partner	25
Casos de Uso del perfil del Cliente	29
Casos de Uso del perfil del Responsable de cocina	37

Casos de Uso del perfil del Repartidor	38
Diseño de la Aplicación	40
Guía de Estilo	40
Logo	40
Colores Principales	40
Componentes de Interfaz	41
Botones	41
Formularios	41
Tipos de Mensajes:	41
Menú Lateral	41
Iconografía	42
Imágenes	42
Navegación	42
Creación de Prototipos de Interfaz de Usuario	42
Diseño de Relaciones de la Base de Datos	43
Diagramas de Secuencia del Sistema	44
Desarrollo del sistema web	46
Elección de tecnología	46
Elección del Lenguaje	46
Instalar Django y configurar el entorno de desarrollo	46
Interfaz de Usuario	47
Funcionalidades clave de la aplicación	47
Funcionalidad del Carrito de Compras	47
Propósito del Carrito de Compras	47
Descripción de las Funciones de la Clase	48
Uso de la Clase CarritoDeCompras en las Vistas	49
Justificación del Uso de la Clase CarritoDeCompras	50
Funcionalidad de Procesamiento de Pedidos	51
Descripción del Proceso de Pedidos	51
Obtención de Coordenadas Geográficas	53
Funcionalidad de Sistema de Pago	54
Integración del Sistema de Pago con Stripe	54
Justificación	56
Funcionalidad de Validación de Pedido	57
Descripción de Validación de Pedido	57
Justificación del Uso de la funcionalidad de validación de pedido	59

Funcionalidad de Seguimiento del Estado del Pedido	59
Implementación funcionalidad de Seguimiento del Estado del Pedido	59
Justificación del Uso de AJAX	60
Vistas Genéricas para Detalles	60
Implementación de la Vista DetalleGenericoView	60
Justificación	62
Pruebas	63
Evaluación de los Requerimientos	63
Criterios de evaluación de Quality Gateway	63
Tabla Inicial de Quality Gateway	64
Ajustes Realizados	65
Tabla Final de Quality Gateway	67
Evaluación heurística de usabilidad	68
Criterios de evaluación heurística de usabilidad	68
Ejecución de la evaluación heurística	69
Resultados evaluación heurística	69
Conclusiones después de realizar la evaluación heurística	71
Pruebas BDD	72
Herramienta Utilizada: Django Behave	72
Estructura de las Pruebas BDD	72
Ejecución de las Pruebas	82
Beneficios de BDD	82
Repository del Proyecto	82
Resultados del Sistema	83
Pantalla de Inicio de Sesión	83
Pantalla de Registro de Partners	84
Pantalla de Registro de Cliente	84
Pantalla de listado de Restaurantes	85
Pantalla de listado de Platos	86
Funcionalidades de Búsqueda y Filtros	86
Pantallas de la gestión del carrito de compras	87
Pantallas del proceso de pedido	88
Pantallas de gestión del pedido para el cliente	89
Pantallas de gestión del pedido en el restaurante por el responsable de cocina	91
Pantallas de Gestión del Pedido por el Repartidor	92
Pantallas de las funcionalidades para el Partner	93

Gestión de Restaurantes	94
Gestión de Platos	95
Gestión de Personal	96
Pantalla de la Gestión de Reseñas	96
Licencia	98
Licencia del Documento	98
Licencia del Programa	98
Planes Futuros para el Desarrollo del Sistema	99
Notificaciones en Tiempo Real	99
Pago a Restaurantes	99
Estadísticas de Ventas por Restaurante	99
Conclusiones	99
Referencias Bibliográficas	100

Introducción

Presentación del TFG

Este **Trabajo de Fin de Grado (TFG)** tiene como objetivo principal el desarrollo de "**DeliverEat**", un sistema web para la gestión de pedidos de comida a domicilio, abordando todas las etapas del ciclo de vida del software. Este proyecto busca adquirir un conocimiento profundo sobre el desarrollo de aplicaciones web funcionales y con un diseño atractivo para los usuarios, utilizando Django como framework principal. Además, integra diversos conocimientos adquiridos a lo largo de mi formación en Ingeniería Informática en la Universidad de Lleida (UDL), en la rama de software. A través de este trabajo, se ha buscado integrar múltiples disciplinas aprendidas en diferentes asignaturas, como ingeniería de software, desarrollo web, ingeniería de requisitos, bases de datos, interacción persona-ordenador, gestión de la calidad y gestión de proyectos. Lo que trato con este trabajo no es solo la creación de una aplicación web funcional, sino que también demostró la capacidad de llevar a cabo un análisis exhaustivo de los requisitos, un diseño meticuloso y una implementación efectiva de soluciones tecnológicas entre otras cosas.

Objetivos del Proyecto

Objetivo Principal del trabajo:

El objetivo principal de este Trabajo de Fin de Grado (TFG) es adquirir un conocimiento profundo sobre un lenguaje de programación web y las tecnologías necesarias para desarrollar un sistema de pedidos de comida a domicilio, utilizando Django como framework principal. Este objetivo implica no solo la creación de una aplicación funcional, sino también la comprensión y aplicación de las mejores prácticas en el desarrollo web.

Objetivos Específicos del trabajo:

- Realizar un análisis de requisitos detallado para identificar las funcionalidades clave que deben incluirse en la aplicación.
- Diseñar una interfaz de usuario intuitiva y atractiva mediante la creación de prototipos. La interfaz debe ser fácil de usar y visualmente agradable, asegurando una experiencia de usuario positiva.
- Seleccionar y justificar las tecnologías más apropiadas para el desarrollo del sistema.

- Desarrollar el frontend, asegurándose de que este refleje fielmente los prototipos diseñados. Esto implica una atención meticulosa a la estética, la usabilidad.
- Desarrollar un backend sólido que satisfaga de manera efectiva tanto los requisitos funcionales como los no funcionales establecidos durante la fase de análisis. Esto incluye la implementación de lógica de negocio eficiente, la gestión adecuada de la base de datos y la integración efectiva con otros sistemas si es necesario.
- Realizar pruebas exhaustivas para garantizar la funcionalidad del sistema.

Investigación Previa

Análisis de diferentes frameworks web (Django, Spring Boot, CodeIgniter)

Esta sección presenta un análisis comparativo de tres frameworks web populares: Django, Spring Boot y CodeIgniter. El objetivo es evaluar sus características, ventajas y desventajas para seleccionar el más adecuado para el desarrollo de un sistema web de reparto de comida.

Django

- **Lenguaje:** Python
- **Arquitectura:** MVT (Model-View-Template)
- **ORM (Object-Relational Mapping):** Django ORM. Facilita el manejo de la base de datos usando objetos Python, lo que permite realizar consultas y operaciones CRUD sin necesidad de escribir SQL.
- **Soporte de plantillas:** Django Template Language. Permite la separación clara de la lógica y la presentación, facilitando la creación de vistas reutilizables y mantenibles.
- **Pruebas BDD (Behavior Driven Development):** Behave.
- **Soporte para REST:** Django REST Framework. Ofrece un conjunto completo de herramientas para construir APIs RESTful de forma rápida y fácil.
- **Facilidad de Uso:** Alta. Django tiene una curva de aprendizaje moderada y una documentación extensa.
- **Ventajas:**
 - Desarrollo rápido gracias a muchas funcionalidades integradas.
 - Abundante documentación.
 - Buen soporte para pruebas y desarrollo ágil.
- **Desventajas:**
 - Puede ser excesivo para proyectos pequeños.
 - El rendimiento puede ser un problema en aplicaciones muy grandes y complejas sin la optimización adecuada.
- **Puntuación:** 9/10

Spring Boot

- **Lenguaje:** Java
- **Arquitectura:** Basado en el principio de "convention over configuration"
- **ORM (Object-Relational Mapping):** Hibernate (JPA). Ofrece una capa de abstracción robusta y potente para la gestión de bases de datos relacionales, facilitando el manejo de datos con Java.
- **Soporte de plantillas:** Thymeleaf, JSP. Permite crear vistas dinámicas con soporte completo de HTML5 y plantillas reutilizables.
- **Pruebas BDD (Behavior Driven Development):** Cucumber
- **Soporte para REST:** Spring MVC / Spring REST. Proporciona un marco completo para construir APIs RESTful de forma rápida y fácil.
- **Facilidad de Uso:** Moderada. Spring Boot requiere conocimientos previos de Java y del ecosistema de Spring, pero simplifica el desarrollo con configuraciones por defecto y una amplia gama de herramientas.
- **Ventajas:**
 - Ideal para arquitecturas de microservicios.
 - Amplio soporte de integración con otros sistemas y servicios.
- **Desventajas:**
 - Curva de aprendizaje pronunciada si no se tiene experiencia con Java o Spring.
 - Requiere más configuración inicial comparado con otros frameworks.
- **Puntuación:** 8/10

Codelgniter

- **Lenguaje:** PHP
- **Arquitectura:** MVC (Model-View-Controller)
- **ORM (Object-Relational Mapping):** No tiene un ORM integrado, pero se pueden usar bibliotecas como Eloquent de Laravel.
- **Soporte de plantillas:** Básico, pero se puede usar cualquier motor de plantillas como Twig o Smarty
- **Pruebas BDD (Behavior Driven Development):** Aunque no tiene una integración directa, se pueden escribir pruebas BDD para Codelgniter usando Codeception.
- **Soporte para REST:** Codelgniter RESTful API Library. Aunque no es muy potente comparado con los de Django o Sprint Boot.
- **Facilidad de Uso:** Alta. Codelgniter es fácil de aprender y usar, ideal para proyectos pequeños a medianos.
- **Ventajas:**
 - Simplicidad y rapidez de desarrollo.
 - Buena documentación y fácil de entender.
 - Bajo consumo de recursos, ideal para servidores con capacidades limitadas.
- **Desventajas:**
 - Menos funcionalidades integradas comparado con Django o Spring Boot.
 - No es tan adecuado para aplicaciones muy grandes y complejas.
- **Puntuación:** 7/10

Definición del Proyecto

En esta etapa me he dedicado a establecer los objetivos, modelo de negocio, análisis de riesgos y la planificación para el desarrollo de "DeliverEat", que son elementos fundamentales para evaluar la viabilidad y rentabilidad del proyecto, ayudando a determinar si es factible llevarlo a cabo.

Objetivos del Proyecto

Objetivo Principal

Desarrollar una sistema web de entrega de comida a domicilio denominada "DeliverEat" y que pueda hacer posible el proceso de entrega de comida a domicilio.

Objetivos Específicos

Objetivo 1: Permitir Pedidos de Múltiples Restaurantes

Descripción: Implementar un sistema que permita a los clientes realizar pedidos de diferentes platos de múltiples restaurantes en una sola transacción.

Propósito: Simplificar el proceso de pedido al permitir a los usuarios pedir de diferentes sitios sin necesidad de realizar múltiples pedidos.

Ventajas:

- Ahorro de tiempo al realizar un solo pedido en lugar de varios.
- Mayor conveniencia para los usuarios al combinar opciones de distintos restaurantes.
- Posibilidad de ofrecer una mayor variedad de opciones en una sola transacción.

Objetivo 2: Implementar un Carrito de Compras

Descripción: Desarrollar un carrito de compras para facilitar la gestión de múltiples pedidos en una sola transacción.

Propósito: Simplificar el proceso de pedido y aumentar la conveniencia para el usuario.

Ventajas:

- Incrementa la satisfacción del usuario mediante una experiencia de compra más fluida.

Objetivo 3: Mejorar la Experiencia de Búsqueda y Pedidos

Descripción: Diseñar e implementar una interfaz de usuario intuitiva para la búsqueda y realización de pedidos.

Propósito: Optimizar la experiencia del usuario al buscar platos y realizar transacciones.

Ventajas:

- Mayor eficiencia en la navegación y selección de platos.
- Reducción de la fricción en el proceso de pedido.

Objetivo 4: Implementar Rastreo

Descripción: Integrar tecnologías de seguimiento de pedidos.

Propósito: Ofrecer a los clientes la capacidad de rastrear sus pedidos.

Ventajas:

- Mayor transparencia en el proceso de entrega.
- Flexibilidad para los usuarios al programar entregas.

Objetivo 5: Facilitar el Feedback de los Usuarios

Descripción: Integrar un sistema de calificación y comentarios para que los usuarios proporcionen feedback sobre los platos consumidos o los restaurantes.

Propósito: Recopilar comentarios valiosos para mejorar la calidad de los platos y la experiencia general.

Ventajas:

- Retroalimentación directa para la mejora continua para los restaurantes.

Objetivo 6: Desarrollar un panel de administrador para los propietarios de restaurantes

Descripción: Crear un panel de administrador que permita gestionar los platos, precios y actividades para sus propios locales en el sistema.

Propósito: Tener un lugar para la gestión de contenidos para simplificar la administración y mantener la coherencia en la aplicación.

Ventajas:

- Mayor eficiencia en la actualización de menús y precios.
- Visualizar el Feedback de los Usuarios que han realizado el pedido en el restaurante.

Modelo de negocio

El modelo de negocio que elegí para "DeliverEat" se basa en la estrategia de "Comisión por Transacción". Esto implica aplicar una comisión del 10% más la comisión del sistema de pagos a los restaurantes por cada transacción que se realiza a través de la aplicación "DeliverEat". Este modelo lo he elegido debido a su potencial para generar considerables beneficios financieros en caso de un crecimiento significativo de usuarios y también a que tengo capacidad de manipular las tarifas de comisión.

Análisis de Riesgos

Identificación de Riesgos

Riesgo	Descripción del Riesgo
Falta de experiencia técnica	Falta de experiencia en la tecnología del desarrollo como Django o la seleccionada.
Falta de personal	Falta de personal al no disponer de un equipo de desarrollo o de un grupo para realizar el proyecto.
Tiempo limitado	Escasez de recursos como el tiempo para cumplir con los plazos del proyecto.
Cambios en los requisitos	Establecer proceso formal de cambios, evaluación de impacto
Riesgo de Dependencia de Terceros	Dependencia de servicios externos que pueden cambiar o tener interrupciones.
Riesgo de Compatibilidad de Dispositivos	Problemas de compatibilidad con diferentes versiones de dispositivos móviles y navegadores web.

Evaluación de Riesgos

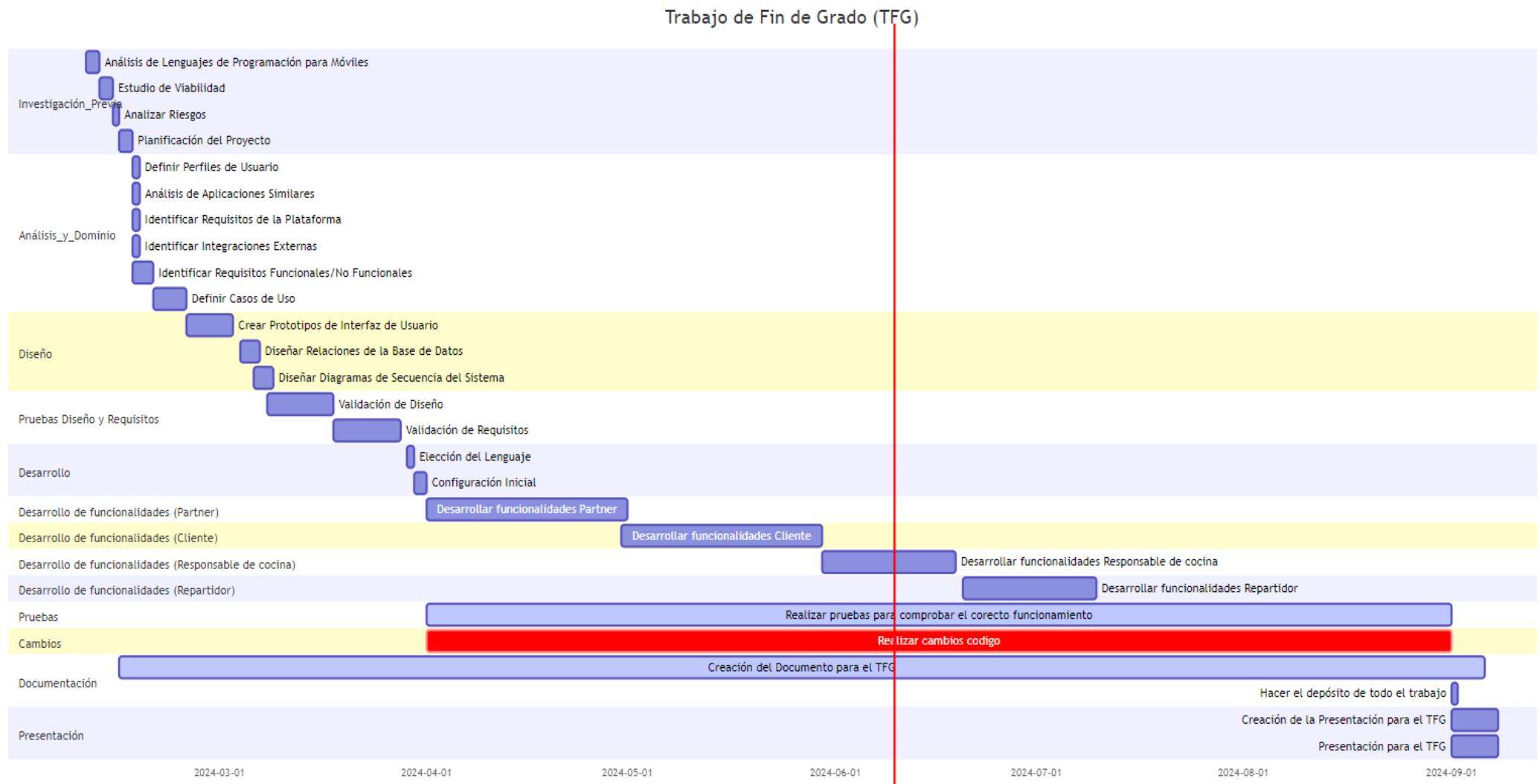
Riesgo	Probabilidad	Impacto	Prioridad
Falta de experiencia técnica	Muy Alta	Alta	Muy Alta
Falta de personal	Media	Baja	Alta
Tiempo limitado	Baja	Baja	Baja
Cambios en los requisitos	Baja	Alto	Medio
Riesgo de Dependencia de Terceros	Moderado	Moderado	Baja
Riesgo de Compatibilidad de Dispositivos	Baja	Moderado	Baja

Plan de Contingencia

Riesgo	Mitigación
Falta de experiencia técnica	En el caso de la falta de experiencia técnica, asignar tiempo adicional para aprender y practicar.
Riesgo de recursos	Planificación cuidadosa del tiempo, asignación eficiente de tareas y tener un plan de contingencia para situaciones imprevistas.
Tiempo limitado	Planificación cuidadosa del tiempo, asignación eficiente de tareas y tener un plan de contingencia para situaciones imprevistas.
Cambios en los requisitos	Establecer revisiones regulares de los requisitos y obtener aprobaciones de los stakeholders.
Riesgo de Dependencia de Terceros	Identificar servicios de respaldo. Tener un plan de contingencia para cambios en servicios externos.
Riesgo de Compatibilidad de Dispositivos	Realizar pruebas exhaustivas en diferentes dispositivos y navegadores web.

Planificación del Proyecto

Para la planificación del proyecto, he desarrollado un diagrama de Gantt que organiza las fases clave del desarrollo del sistema. Este diagrama incluye las etapas de Análisis y Dominio, Diseño, Pruebas de Diseño y Requisitos, Desarrollo de Funcionalidades para Partner, Cliente, Responsable de Cocina y Repartidor, Pruebas, Cambios, Documentación y Presentación. Esta estructura permite una gestión eficiente y un seguimiento claro del progreso en cada fase del proyecto, garantizando que se cumplan los requisitos y objetivos establecidos.



Conclusión sobre la Rentabilidad y Viabilidad del Proyecto

Tras el análisis detallado anteriormente de los objetivos, el modelo de negocio, los riesgos y la planificación del proyecto "DeliverEat", se puede concluir que el proyecto es viable y rentable por lo que se procederá con su implementación. El modelo de negocio basado en comisiones por transacción es sólido y escalable, garantizando una fuente de ingresos continua. Los objetivos definidos son alcanzables, lo que asegura una experiencia de usuario óptima. La planificación del proyecto está alineada con los plazos de entrega y permite una gestión eficiente del tiempo. Además, aunque se han identificado algunos riesgos, se han propuesto medidas de mitigación que aseguran que estos desafíos pueden ser superados. El proyecto tiene un alto potencial de éxito.

Análisis de Requisitos

Método de Obtención de Requisitos

Se realizó un análisis detallado de las aplicaciones más destacadas en España, como Glovo, Uber Eats y Just Eat. Este análisis permitió identificar los perfiles de usuarios, las plataformas en las que utilizarán la aplicación, los sistemas externos necesarios, así como las funcionalidades comunes y distintivas que podrían ser relevantes para nuestra propia aplicación.

Definición de Perfiles de Usuario y Necesidades

Perfil del Cliente:

Descripción: Personas que utilizan la aplicación para realizar pedidos de comida de restaurantes locales.

Franja de Edad: En su mayoría jóvenes adultos de 18 a 35 años

Experiencia con Aplicaciones: Puede variar desde usuarios experimentados hasta principiantes.

Necesidades:

- Visualizar opciones y precios ofrecidos por diversos restaurantes locales.
- Realizar búsquedas y pedidos de los platos de forma rápida y fácil.
- Rastrear el estado de los pedidos en tiempo real.
- Puntuar y dar una opinión sobre la comida.
- Programar la entrega de un pedido.

Perfil del Repartidor:

Descripción: Personas contratadas para entregar pedidos a los clientes desde el restaurante.

Franja de Edad: Diversas edades, énfasis en adultos jóvenes entre 25 a 45 años.

Experiencia con Aplicaciones: Se espera que la mayoría tenga experiencia en el uso de aplicaciones móviles.

Necesidades:

- Recibir notificaciones inmediatas de nuevos pedidos.
- Acceder a la información de entrega y la ubicación del cliente.
- Actualizar el estado de entrega en la aplicación.

Perfil del Partner:

Descripción: Personas encargadas de gestionar el restaurante y sus menús en la plataforma.

Franja de Edad: Diversas edades, generalmente adultos.

Experiencia con Aplicaciones: Variada, desde principiantes hasta experimentados.

Necesidades:

- Agregar, editar o eliminar platos del menú y las ofertas.
- Monitorear los pedidos de restaurantes.
- Dar de alta a los trabajadores a la aplicación
- Dar alta de sus locales.

Perfil del Encargado de Cocina:

Descripción: Personal de cocina encargado de preparar los pedidos recibidos.

Franja de Edad: Puede variar

Experiencia con Aplicaciones: Varía, algunos pueden tener experiencia, mientras que otros pueden no tener experiencia en uso de tecnologías.

Necesidades:

- Visualizar pedidos entrantes y detalles de los platos.
- Marcar los pedidos como preparados para el envío.
- Acceder a información sobre cambios en los pedidos.

Análisis de Aplicaciones Similares

Llevé a cabo un análisis detallado de tres de las aplicaciones más destacadas en España: Glovo, Uber Eats y Just Eat. Este análisis nos permitió identificar funcionalidades comunes y características que las diferencian.

Funcionalidades Comunes:

- Proporcionan una función de búsqueda para localizar restaurantes cercanos basándose en la ubicación del usuario o preferencias de comida.
- Presentan los menús completos de los restaurantes asociados, con descripciones detalladas de los platos para ayudar a los usuarios a tomar una decisión.

- Permiten seleccionar platos, personalizar pedidos y realizar compras directamente a través de la aplicación.
- Ofrecen métodos seguros de pago en línea.
- Permiten a los usuarios guardar sus restaurantes favoritos.
- Permiten a los usuarios acceder a su historial de pedidos.
- Permiten a los usuarios calificar y dejar reseñas sobre restaurantes y platos.

Funcionalidades Distintivas:

Glovo

- Glovo ofrece servicios de entrega no solo de comida, sino también de otros productos, como medicamentos y flores.
- Se permite a los usuarios recoger sus pedidos en ubicaciones específicas.

Uber Eats

- Ofrecen a los usuarios un sistema de puntos de Uber Rewards al utilizar Uber Eats.
- Permite el pago en efectivo aparte de poder pagar de manera online.

Just Eat

- Los usuarios tienen la opción de programar pedidos con antelación.

Este análisis proporcionó una visión completa del panorama actual de las aplicaciones de entrega de comida en España, sirviendo como base esencial para identificar requisitos y funcionalidades clave para nuestra propia aplicación.

Identificación de Requisitos de Plataforma

Clients (Aplicación Móvil - iOS y Android):

Descripción: Los clientes tendrán la opción de utilizar la aplicación en dispositivos iOS (iPhone) o Android.

Justificación: Al ofrecer la aplicación en ambas plataformas principales (iOS y Android), se maximiza la audiencia potencial, permitiendo que la aplicación sea accesible para una gama más amplia de usuarios.

Repartidores (Aplicación Móvil - Android):

Descripción: Los repartidores accederán a una aplicación móvil diseñada en principio para dispositivos Android.

Justificación: La inversión para el restaurante en un dispositivo móvil con android es menor y cumple con las necesidades.

Administradores (Aplicación de Escritorio o Web):

Descripción: Los administradores accederán a la plataforma a través de una aplicación de escritorio o una interfaz web normalmente.

Justificación: Una aplicación de escritorio o web proporciona mayor funcionalidad y comodidad para las tareas de administración

Encargado de Cocina a (Aplicación en Tablet - Android)

Descripción: Los encargados de cocina accederán a una aplicación específica diseñada para tabletas con sistema Android.

Justificación: Las tabletas proporcionan una pantalla más grande, facilitando la visualización de detalles de pedidos y mejorando la usabilidad para el personal de cocina. Adicionalmente, la inversión para el restaurante en un tablet con Android es menor y cumple con las necesidades.

Integraciones de Sistemas Externos

Integración con Servicio de Mapas

Se requiere la integración con el servicio de mapas tiene como objetivo principal proporcionar a los repartidores la capacidad de visualizar la ubicación de los clientes. De esta manera facilitamos a los repartidores la localización de las entregas a domicilio que tienen que realizar.

Nombre del Sistema Integrado: **Leaflet**

Beneficios de usar Leaflet:

Sencillo y Rápido de Aprender

- Su interfaz intuitiva y diseño amigable ofrecen una experiencia de usuario sin complicaciones.
- Permite la personalización del aspecto con elementos como botones y layer switcher para proporcionar una experiencia visual única.
- Ofrece un sólido soporte para dispositivos móviles.
- Ofrece un alto rendimiento.

Integración con Sistema de Pagos en Línea

Se requiere la integración con el sistema de pagos en línea tiene como objetivo facilitar a los usuarios la realización de transacciones seguras y eficientes. De esta manera los pagos en línea facilitarán transacciones seguras y sin complicaciones, mejorando la experiencia del usuario y brindando una plataforma confiable.

Nombre del Sistema Integrado: **Stripe**

Beneficios de usar Stripe:

Fácil de Usar e Implementar

- Destaca por su interfaz fácil de usar y de implementar, simplificando el proceso para usuarios y desarrolladores.
- Garantiza transacciones seguras y rápidas, priorizando la seguridad de los datos del usuario.
- No requiere cuotas de membresía ni contratos a largo plazo.

- Permite aceptar pagos de diferentes partes del mundo y en diversas monedas.
- Ofrece soporte técnico las 24 horas del día.
- Stripe es compatible con una amplia variedad de plataformas y frameworks de programación de diferentes lenguajes.
- La comisión en Europa es de 1,4% que es la más baja de todos los servicios de pagos.
- Es muy utilizada en muchos comercios online.

Al integrar estos sistemas externos, se logra mantener la comercialización de la aplicación sin afectar el modelo de negocio establecido. Mientras atribuyas a Leaflet en mi aplicación y la citó en los créditos dentro de sus servicios de mapas y por parte de Stripe mientras cumple con todas las leyes y regulaciones aplicables en relación con el comercio electrónico y pago la comisión establecida por ellos.

Identificación de Requisitos

Requisitos Funcionales

Perfil del Partners:

1. El sistema permitirá registrar nuevos restaurantes y gestionar su información, incluyendo detalles específicos y permitiendo la creación de una cuenta de administrador para los restaurantes creados.
2. Los partners tendrán la capacidad de gestionar los platos que ofrece el restaurante.
3. Los partners tendrán la capacidad de visualizar y modificar la lista de platos disponibles y poder aplicar descuentos si lo desean.
4. Los partners tendrán la capacidad de poder visualizar las reseñas y puntuación de los restaurantes y de sus platos.
5. Los partners tendrán la capacidad de poder visualizar un historial de los pedidos realizados en el restaurante.
6. Los partners tendrán la capacidad de poder dar acceso al sistema a los repartidores de comida y al responsable de cocina.

Perfil del Cliente:

7. El sistema permitirá a los clientes registrarse proporcionando un nombre de usuario y una contraseña.
 - 7.1. Los clientes tendrán que establecer una ubicación predeterminada para la entrega de pedidos.
 - 7.2. Los clientes tendrán que establecer sus datos de contacto como el correo electrónico y el número de teléfono.
8. Los clientes podrán visualizar los restaurantes disponibles, más cercanos.
9. Los clientes podrán explorar los platos ofrecidos por cada restaurante disponible, incluyendo fotos, precios, lista de ingredientes y puntuación.
10. Los clientes podrán filtrar los platos en función de nombre, la categoría de comida, precios o puntuación media.
11. Las listas de platos estarán posicionadas en función a la puntuación más alta.

12. Los clientes podrán seleccionar diferentes platos de diferentes restaurantes que ofrece el sistema y añadirlos al carrito de compra.
13. Los clientes podrán ver dentro del carrito la cantidad de platos y su montón acumulado y el precio total. También en el caso de que hay platos de diferentes restaurantes agruparlos y mostrar el total de cada restaurante.
14. Los clientes podrán realizar el pago del pedido realizado mediante pagos con tarjeta de crédito.
15. Los clientes podrán elegir si realizará el pedido a la dirección por defecto o especificará la nueva dirección.
16. Los clientes podrán cancelar un pedido de un restaurante que no ha empezado con su proceso de elaboración del plato.
17. Los clientes podrán dejar una puntuación y/o una reseña para valoraciones para el restaurante en general.
18. Los clientes podrán dejar una puntuación y/o una reseña y valoraciones para platos específicos ofrecidos por el restaurante.
19. Los clientes tendrán acceso a información detallada sobre el estado actual de su pedido, incluyendo la fase de preparación en la cocina y el estado de entrega por parte del repartidor.
20. El sistema realizará el pago del pedido al restaurante una vez el cliente haya confirmado que su pedido ha sido entregado.
21. El cliente podrá recibir una factura que la podrá descargar en formato electrónico o imprimir.

Perfil del Repartidor:

22. El sistema notificará a los repartidores del restaurante de los pedidos que tiene que realizar.
23. El sistema le proporcionará al repartidor la información de entrega y la ubicación del cliente.
24. El repartidor confirmará que realizó el pedido correctamente.

Perfil del Encargado de Cocina:

25. El sistema notificará al encargado de la cocina sobre el nuevo pedido.
26. El sistema le mostrará con detalles los encargos que tenga que preparar y los ingredientes.
27. El Encargado de Cocina podrá cambiar el estado de los pedidos según como se encuentra la preparación del pedido.
28. El Encargado de la cocina asignará un repartidor que estará disponible.

No Funcionales

Nº Req. N.F.	1
Descripción	El sistema debe estar protegido contra el acceso no autorizado.
Justificación	Evitando accesos no autorizados que podrían comprometer la seguridad.
Fit Criterion	La tasa de éxito en la autenticación debe ser del 100%.
Categoría	Seguridad

Nº Req. N.F.	2
Descripción	El sistema no debe permitir realizar funcionalidades que no corresponden a su rol.
Justificación	Evitar que los usuarios realicen acciones para las cuales no tienen permisos, reduciendo así los riesgos de manipulación indebida de datos o funciones críticas del sistema.
Fit Criterion	Las funciones no autorizadas deben ser rechazadas en el 100% de los casos.
Categoría	Seguridad

Nº Req. N.F.	3
Descripción	La interfaz de usuario debe ser amigable y fácil de usar.
Justificación	Mejorar la experiencia del usuario y facilitar la interacción con el sistema.
Fit Criterion	Se debe realizar una prueba heurística de usabilidad con un experto en usabilidad y el tiempo promedio para completar una tarea en la interfaz de usuario es inferior a 2 minutos.
Categoría	Usabilidad

Nº Req. N.F.	4
Descripción	Que el sistema sea capaz de responder en el momento de hacer una petición a un sistema externo y este falle.
Justificación	Garantizar la disponibilidad y la robustez del sistema frente a posibles fallos en los sistemas externos.
Fit Criterion	La aplicación es capaz de responder adecuadamente a un fallo de sistema externo en menos de 5 segundos, mostrando un mensaje de error al usuario.
Categoría	Fiabilidad

Nº Req. N.F.	5
Descripción	Diseño responsive
Justificación	Garantiza que el diseño sea responsive y se adapte adecuadamente a diferentes dispositivos y a una variedad de tamaños de pantalla, para poder mejorar la experiencia del usuario en todas las plataformas posibles.
Fit Criterion	La aplicación se prueba en al menos 3 tamaños de pantalla diferentes y se verifica que se adapta correctamente en todos ellos.
Categoría	Usabilidad

Nº Req. N.F.	6
Descripción	Servicio de mapas incorporado en la aplicación para los repartidores.
Justificación	La inclusión de un servicio de mapas facilita a los repartidores la búsqueda y navegación hacia las direcciones de entrega, mejorando la eficiencia y precisión en la entrega de pedidos.
Fit Criterion	Intentar que con la inclusión del servicio de mapas integrado para mejorar la eficiencia y precisión en la entrega de pedidos en un 60%.
Categoría	Usabilidad

Nº Req. N.F.	7
Descripción	Integración con Sistema de Pagos en Línea
Justificación	La integración con un servicio de pagos en línea confiable y seguro como Stripe garantiza que las transacciones bancarias realizadas a través de la aplicación sean seguras y eficientes.
Fit Criterion	Probar que la aplicación pueda realizar más de 100 peticiones al sistema de pagos en línea Stripe de forma segura y que el tiempo de procesamiento promedio sea de menos de 5 segundos por transacción y si no responder con un mensaje de error al usuario.
Categoría	Seguridad, Fiabilidad

Nº Req. N.F.	8
Descripción	Mensajes de error informativos y orientados al usuario final
Justificación	Proporcionar mensajes de error informativos y orientados al usuario final mejora la usabilidad reduciendo la frustración del usuario y facilita la resolución de problemas.
Fit Criterion	Hacer preguntas en la evaluación heurística que tenga en cuenta este tema y probar en más de 10 personas para asegurar que los mensajes de error son claros y ayudar al usuario.
Categoría	Usabilidad

Nº Req. N.F.	9
Descripción	Autenticación de entrega
Justificación	La autenticación de entrega asegura que los productos sean entregados al destinatario correcto y asegurar que los ha recibido, lo que garantiza la integridad del proceso de entrega y la satisfacción del cliente.
Fit Criterion	El sistema confirma esta autenticación en menos de 1 minuto. Los pasos para confirmar esta autenticación no deben tardar más de 5 segundos en realizarse.
Categoría	Seguridad

Nº Req. N.F.	10
Descripción	Protección de datos
Justificación	Garantizar la protección de los datos del usuario y cumplir con las normas de privacidad y seguridad GDPR.
Fit Criterion	El sistema debe garantizar la protección de los datos del usuario, cumpliendo con las normativas de privacidad y seguridad GDPR al menos un 90%
Categoría	Seguridad / Privacidad

Definición de Casos de Uso

CU nº	CU-001 Login
Actores Principales:	Usuario (Cliente, Administrador, Repartidor, Encargado de Cocina)
Descripción:	El proceso mediante el cual un usuario accede al sistema después de autenticarse con sus credenciales.
Dependencia:	CU-003 Registro de Clientes
Precondiciones:	El usuario debe tener una cuenta registrada en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El usuario accede a la página de inicio de sesión del sistema. 2. El sistema presenta un formulario de inicio de sesión solicitando al usuario que ingrese su dirección de correo electrónico y contraseña. 3. El usuario ingresa su dirección de correo electrónico y contraseña en los campos correspondientes. 4. El sistema valida las credenciales ingresadas por el usuario y lo dirigirá a la página que le corresponde a su rol.
Poscondición :	El usuario ha accedido con éxito al sistema y tiene acceso a las funcionalidades correspondientes a su rol.
Excepciones:	<p>Validación de Datos Fallida: El sistema muestra un mensaje de error en los siguientes casos:</p> <ul style="list-style-type: none"> • No existe ninguna cuenta creada con este correo electrónico. • La contraseña no es correcta. • El sistema no permitirá el acceso y no dirigirá al usuario a la página inicial.
Flujo Alternativo:	<p>Recuperación de contraseña: Si el usuario no recuerda su contraseña, puede hacer clic en "¿Olvidaste tu contraseña?" para iniciar el flujo de recuperación de contraseña.</p>

CU nrº	CU-002 Registrar Partners en el Sistema
Actores Principales:	Partner (Socio)
Descripción:	Como el partner sigue para registrarse en el sistema de entrega de comida.
Dependencia:	
Precondiciones:	El partner debe acceder vía navegador a la página de registro del sistema de entrega de comida para utilizarla por primera vez si no tiene una cuenta registrada en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El partner accede a la página de registrarse para partners. 2. El sistema muestra un formulario de registro. 3. El partner completa el formulario proporcionando nombre de usuario y contraseña. 4. El sistema valida los datos introducidos por el partner. 5. El sistema solicita al partner ingresar sus datos personales (Nombre, Apellido) y sus datos de contacto (Teléfono, Correo electrónico). 6. El sistema valida los datos introducidos por el partner. 7. El sistema solicita al partner ingresar su nombre de negocio y sus documentos (Licencias comerciales, permisos de salud y registros sanitarios) en formato PDF. 8. El sistema valida los datos de la dirección. 9. El sistema muestra un mensaje de confirmación de registro y redirige al partner a la página de inicio de sesión.
Poscondición :	El partner ha creado con éxito una cuenta en el sistema de entrega de comida y puede iniciar sesión en cualquier momento.
Excepciones:	<p>Validación de Datos Fallida (Nombre de usuario y Contraseña): El sistema muestra un mensaje de error si:</p> <ul style="list-style-type: none"> • Una cuenta ya existe con este nombre de usuario. • El formato del nombre de usuario o la contraseña no es válido. <p>Validación de Datos de Negocio: El sistema muestra un mensaje de error si:</p> <ul style="list-style-type: none"> • El campo de nombre de negocio está vacío. • No se añade un documento en formato PDF. <p>Validación de Datos de Contacto: El sistema muestra un mensaje de error si los datos de contacto no son válidos.</p> <p>Validación Final Fallida: El sistema muestra un mensaje de error si el cliente no selecciona al menos una categoría de comida favorita.</p>
Flujo Alternativo:	

CU nrº	CU-003 Registro de Clientes
Actores Principales:	Cliente
Descripción:	Este caso de uso describe el proceso que un cliente sigue para registrarse en el sistema de entrega de comida.
Dependencia:	
Precondiciones:	El cliente debe acceder vía navegador a la página de registro del sistema de entrega de comida para utilizarla por primera vez si no tiene una cuenta registrada en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la página de registrarse para cliente. 2. El sistema muestra un formulario de registro. 3. El cliente completa el formulario proporcionando nombre de usuario y contraseña. 4. El sistema valida los datos introducidos por el cliente. 5. El sistema solicita al cliente ingresar sus datos personales (Nombre, Apellido) y sus datos de contacto (Teléfono, Correo electrónico). 6. El sistema valida los datos introducidos por el cliente. 7. El sistema solicita al cliente ingresar su ubicación por defecto. 8. El cliente introduce la dirección de entrega preferida. 9. El sistema valida los datos de la dirección. 10. El sistema muestra un mensaje de confirmación de registro y redirige al cliente a la página de inicio de sesión.
Poscondición :	El cliente ha creado con éxito una cuenta en el sistema de entrega de comida y puede iniciar sesión en cualquier momento.
Excepciones:	<p>Validación de Datos Fallida (Nombre de usuario y Contraseña): El sistema muestra un mensaje de error si:</p> <ul style="list-style-type: none"> • Una cuenta ya existe con este nombre de usuario. • El formato del nombre de usuario o la contraseña no es válido. <p>Validación de Datos de Dirección Fallida: El sistema muestra un mensaje de error si:</p> <ul style="list-style-type: none"> • El formato de la dirección ingresada no es válido. • La dirección no se encuentra en el área de servicio de entrega. • La dirección es inválida o no reconocida por el sistema de geolocalización. <p>Validación de Datos de Contacto: El sistema muestra un mensaje de error si los datos de contacto no son válidos.</p> <p>Validación Final Fallida: El sistema muestra un mensaje de error si el cliente no selecciona al menos una categoría de comida favorita.</p>
Flujo Alternativo:	

Casos de Uso del perfil del Partner

CU nrº	CU-004 Agregar Restaurante en el Sistema
Actores Principales:	Partner
Descripción:	Como el partner puede agregar un nuevo restaurante al sistema.
Dependencia:	
Precondiciones:	El partner debe haber iniciado sesión en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El partner accede a la sección de gestión de restaurantes en el sistema. 2. El sistema muestra la opción de dar de alta un nuevo restaurante. 3. El partner selecciona la opción para dar de alta un nuevo restaurante. 4. El partner completa el formulario de registro del restaurante, proporcionando detalles como nombre del local, dirección, descripción, categoría de comida que ofrece, información de contacto, horarios de funcionamiento, etc. 5. El sistema valida los datos ingresados por el partner. 6. Si los datos son válidos, el sistema registra el nuevo restaurante en el sistema. 7. El partner confirma la creación exitosa del restaurante.
Poscondición :	El restaurante se registra correctamente en el sistema y está disponible para añadir los platos.
Excepciones:	<p>Validación de Datos Fallida:</p> <ul style="list-style-type: none"> • El usuario no ha respetado el formato del formulario al ingresar el nombre del local, dirección, descripción, etc. • No se ha seleccionado al menos una categoría de comida que ofrece el negocio. • La imagen ingresada en el formulario no cumple con el formato JPG o PNG. • El sistema muestra un mensaje claro y conciso de error y no permite continuar con la creación del restaurante hasta que los datos sean corregidos.
Flujo Alternativo:	

CU nrº	CU-005 Añadir Platos a un Restaurante en el Sistema
Actores Principales:	Partner
Descripción:	El proceso mediante el cual un partner añade nuevos platos al menú de un restaurante registrado en el sistema.
Dependencia:	
Precondiciones:	Debe existir al menos un restaurante registrado en el sistema, asociado a la cuenta del partner. El partner debe haber iniciado sesión en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El partner accede a la sección de gestión de platos de un restaurante específico. 2. El sistema muestra una lista de platos actualmente disponibles en el menú del restaurante. 3. El partner selecciona la opción para añadir un nuevo plato. 4. El sistema presenta un formulario para ingresar los detalles del nuevo plato, como nombre, descripción, categoría, precio, ingredientes, etc. 5. El partner completa el formulario con la información del nuevo plato. 6. El partner confirma la creación del nuevo plato. 7. El sistema valida la información proporcionada por el partner. 8. Si la validación es exitosa, el sistema añade el nuevo plato al menú del restaurante y lo hace visible para los clientes. 9. El sistema muestra un mensaje de confirmación al administrador indicando que el plato ha sido añadido con éxito.
Poscondición :	El nuevo plato está añadido al menú del restaurante y es visible para los clientes que accedan a la aplicación.
Excepciones:	<p>Validación de Datos Fallida en la creación del plato:</p> <ul style="list-style-type: none"> • El usuario no ha respetado el formato del formulario al ingresar datos como nombre, descripción, categoría, precio, ingredientes. • El sistema muestra un mensaje claro y conciso de error y no permite continuar con la creación del plato hasta que los datos sean corregidos.
Flujo Alternativo:	

CU nrº	CU-006 Crear Usuarios para Responsables de Cocina y Repartidores del Restaurante
Actores Principales:	Partner
Descripción:	Los Partner podrán crear usuarios para responsables de cocina y repartidores del restaurante el sistema de entrega de comida.
Dependencia:	
Precondiciones:	Debe existir al menos un restaurante registrado en el sistema, asociado a la cuenta del partner. El partner debe haber iniciado sesión en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El partner accede a la sección de gestión de usuarios de un restaurante específico. 2. El sistema muestra las opciones disponibles para crear nuevos usuarios. 3. El partner elige la opción de crear un nuevo usuario. 4. El partner completa el formulario de registro del nuevo usuario, proporcionando detalles como nombre, dirección de correo electrónico, contraseña, número de teléfono, rol (responsable de cocina o repartidor). 5. El sistema valida los datos ingresados por el partner. 6. Si los datos son válidos, el sistema registra al nuevo usuario en el sistema con el rol correspondiente. 7. El partner confirma la creación exitosa del nuevo usuario.
Poscondición :	El nuevo usuario con el rol de responsable de cocina o repartidor podrá acceder a sus funcionalidades específicas en el sistema.
Excepciones:	<p>Validación de Datos Fallida en la creación de usuarios:</p> <ul style="list-style-type: none"> • Una cuenta ya existe con este correo electrónico. • El usuario no ha respetado el formato del formulario al ingresar datos como correo electrónico, contraseña, etc. • El usuario no selecciona el rol que desea asignar. • El sistema muestra un mensaje claro y conciso de error y no permite continuar con la creación del usuario hasta que los datos sean corregidos.
Flujo Alternativo:	

CU nº	CU-007 Historial de Pedidos de un Restaurante
Actores Principales:	Partner
Descripción:	Como partner del restaurante, quiero ver un historial de todos los pedidos realizados en mi restaurante para poder gestionar eficientemente las operaciones.
Dependencia:	
Precondiciones:	El partner debe haber iniciado sesión en el sistema.
Flujo Principal:	<ol style="list-style-type: none"> 1. El partner accede a la sección de historial de pedidos del restaurante en el sistema. 2. El sistema muestra una tabla con los pedidos ordenados ascendentemente por la fecha de creación del pedido. 3. La tabla contiene los siguientes campos: Código del pedido, Cliente, Estado, Platos, Total, Repartidor, Fecha. 4. El partner selecciona un pedido específico de la tabla. 5. El partner presiona el botón de "Ver más detalles". 6. El sistema muestra información detallada del pedido seleccionado, incluyendo todos los campos mencionados más cualquier información adicional relevante. 7. El partner tiene la opción de imprimir o descargar los detalles del pedido en su PC.
Poscondición :	El partner puede ver la historia de los pedidos que se realizaron en el restaurante y podrá ver todos los detalles del pedido deseado.
Excepciones:	
Flujo Alternativo:	

Casos de Uso del perfil del Cliente

CU nº	CU-008 Realizar Pedido
Actores Principales:	Cliente
Descripción:	Este caso de uso describe el proceso que sigue el cliente para realizar un pedido de comida a través de la aplicación.
Dependencia:	CU-00X Login
Precondiciones:	El cliente ha iniciado sesión en la aplicación y ha seleccionado la opción de realizar un pedido.
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la sección de pedidos en la aplicación. 2. El sistema muestra una lista de restaurantes disponibles cercanos a su dirección, ordenados por puntuación media de mayor a menor, proporcionando el nombre, la foto, la descripción, la categoría de comidas que ofrecen, la puntuación y las reseñas de otros usuarios. 3. El cliente selecciona un restaurante de la lista. 4. El sistema muestra el listado de platos del restaurante seleccionado, ordenados por puntuación media de mayor a menor, con imagen, precio, ingredientes, puntuación y valoraciones de otros usuarios. 5. El cliente puede buscar platos dentro del restaurante utilizando filtros por precio, tipo de comida y nombre. 6. El cliente puede ordenar los platos por precio o por puntuación media. 7. El sistema verifica que el restaurante seleccionado tiene un responsable de cocina y al menos un repartidor dados de alta. 8. Para cada plato que cumple con esta condición, el sistema muestra un botón de "Añadir al carrito". 9. El cliente selecciona el plato deseado y lo añade al carrito de compras. <p>El sistema añade el plato seleccionado al carrito de compras del cliente.</p>
Poscondición :	El plato seleccionado por el cliente se añade al carrito de compra finalmente.
Excepciones:	Si el usuario realiza una búsqueda con un rango de precios incorrecto (por ejemplo, Precio desde mayor que Precio hasta), el sistema muestra un mensaje de error indicando que los filtros de búsqueda de precios son inválidos.
Flujo Alternativo:	

CU nrº	CU-009 Gestionar el Carrito de Compras
Actores Principales:	Cliente
Descripción:	Cómo el cliente gestiona los platos seleccionados para comprar antes de finalizar el pedido.
Dependencia:	CU-00X Login
Precondiciones:	El cliente ha iniciado sesión en la aplicación y ha seleccionado uno o varios platos para comprar.
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la sección de carrito de compras en la aplicación. 2. El sistema muestra una lista de los platos seleccionados por el cliente, incluyendo detalles como nombre, precio, cantidad seleccionada y subtotal. 3. El cliente tiene la opción de editar la cantidad de cada plato en el carrito. 4. El cliente puede eliminar un plato específico del carrito si ya no desea comprarlo. 5. El cliente puede eliminar todo el contenido del carrito de compra si lo desea. 6. El sistema recalcula automáticamente el total de la compra en función de las modificaciones realizadas por el cliente. 7. El cliente revisa la lista de platos en el carrito y confirma que está listo para finalizar el pedido.
Poscondición :	El cliente ha gestionado con éxito los platos en su carrito de compras y podrá realizar la finalización del pedido.
Excepciones:	Si el cliente intenta finalizar el pedido pero el total del pedido es inferior a 10 euros, el sistema no mostrará el botón para la finalización del pedido.
Flujo Alternativo:	

CU nrº	CU-010 Finalizar Pedido
Actores Principales:	Cliente
Descripción:	Cómo el cliente finaliza su pedido proporcionando la dirección de entrega y los detalles de pago, y cómo el sistema procesa el pedido.
Dependencia:	CU-00X Login
Precondiciones:	El cliente ha iniciado sesión en la aplicación y ha gestionado su carrito de compras y está listo para finalizar el pedido.
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la opción de finalizar pedido desde el carrito de compras. 2. El sistema muestra un formulario con la dirección por defecto del cliente y los detalles de la tarjeta de crédito. 3. El cliente tiene la opción de cambiar la dirección de entrega. 4. El cliente introduce los detalles de la tarjeta de crédito y confirma el pago. 5. El sistema procesa el pago y verifica que se ha realizado correctamente. 6. El sistema agrupa los platos por cada restaurante en pedidos diferentes. 7. El sistema envía los pedidos correspondientes a cada restaurante.
Poscondición :	El pedido ha sido procesado correctamente y enviado a los restaurantes correspondientes para su preparación y entrega.
Excepciones:	Error en el pago: El sistema muestra un mensaje de error y solicita al cliente que revise los detalles de la tarjeta de crédito y no se realizará el pedido.
Flujo Alternativo:	

CU nrº	CU-011: Consultar Pedidos Realizados
Actores Principales:	Cliente
Descripción:	Cómo el cliente puede ver un listado de todos los pedidos que ha realizado, con la opción de cancelar pedidos en ciertos estados y generar o imprimir una factura en formato PDF para un pedido seleccionado.
Dependencia:	CU-003: Finalizar Pedido
Precondiciones:	El cliente ha iniciado sesión en la aplicación y ha realizado al menos un pedido
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la sección de "Mis Pedidos" en la aplicación. 2. El sistema muestra una lista de todos los pedidos realizados por el cliente con los siguientes detalles: Código del Pedido, Estado, Platos, Total, Restaurante, Fecha y Código de Validación. 3. El cliente puede seleccionar un pedido de la lista para ver más detalles como los platos y las cantidad y el total y los impuestos que le aplica la plataforma de envío de comida. 4. Si el pedido seleccionado no está en el estado "En preparación", el cliente tiene la opción de cancelarlo. 5. El cliente tiene la opción de descargar o imprimir la factura con los detalles mostrados en formato PDF para el pedido seleccionado. 6. El sistema genera la factura y según lo que seleccionó el cliente la imprimiera o descargara un archivo en formato PDF.
Poscondición :	El cliente ha visualizado los detalles de sus pedidos y a realizado las opciones que a tenido convenientes como la cancelación de pedido o la descargado e imprimido de una factura
Excepciones:	<p>Error al cancelar el pedido: El sistema muestra un mensaje de error si no se puede cancelar el pedido debido a un cambio de estado inesperado.</p> <p>Error al generar la factura: El sistema muestra un mensaje de error si ocurre un problema al generar la factura en formato PDF.</p>
Flujo Alternativo:	Si el cliente selecciona un pedido en estado "En preparación", el sistema deshabilita la opción de cancelación

CU nrº	CU-012 Agregar Puntuación y/o Reseña para un Plato
Actores Principales:	Cliente
Descripción:	El cliente desea agregar una puntuación y/o una reseña para un plato específico que ha pedido anteriormente.
Dependencia:	CU-003: Finalizar Pedido
Precondiciones:	El cliente ha iniciado sesión en la aplicación y ha completado al menos un pedido.
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la sección de historial de pedidos en la aplicación. 2. El sistema muestra una lista de todos los pedidos realizados por el cliente. 3. El cliente selecciona el pedido que incluye el plato al que desea dejar la puntuación y/o reseña. 4. El sistema muestra los detalles del pedido, incluyendo los platos solicitados. 5. El cliente selecciona el plato al que desea dejar la puntuación y/o reseña. 6. El sistema presenta un formulario donde el cliente puede ingresar la puntuación y/o escribir una reseña para el plato seleccionado. 7. El cliente completa el formulario y confirma el envío de la puntuación y/o reseña. 8. El sistema registra la puntuación y/o reseña asociada al plato seleccionado.
Poscondición :	La puntuación y/o reseña ha sido agregada correctamente al plato seleccionado y queda registrada en el sistema.
Excepciones:	<p>El sistema no permite añadir una reseña sin una puntuación.</p> <p>El sistema no permite añadir una reseña de un plato que no ha sido pedido.</p> <p>El sistema no permite realizar reseñas sin puntuación y sin comentarios.</p>
Flujo Alternativo:	Si el cliente intenta añadir una reseña sin una puntuación, el sistema muestra un mensaje de error solicitando que se incluya una puntuación.

CU nrº	CU-013 Agregar Puntuación y/o Reseña para un Restaurante
Actores Principales:	Cliente
Descripción:	El cliente desea agregar una puntuación y/o una reseña para un restaurante específico basado en su experiencia general.
Dependencia:	CU-003: Finalizar Pedido
Precondiciones:	El cliente ha iniciado sesión en la aplicación y ha completado al menos un pedido.
Flujo Principal:	<ol style="list-style-type: none"> 1. El cliente accede a la sección de historial de pedidos en la aplicación. 2. El sistema muestra una lista de todos los pedidos realizados por el cliente. 3. El cliente selecciona el pedido que incluye el restaurante al que desea dejar la puntuación y/o reseña. 4. El sistema muestra los detalles del pedido, incluyendo el restaurante asociado. 5. El cliente selecciona el restaurante al que desea dejar la puntuación y/o reseña. 6. El sistema presenta un formulario donde el cliente puede ingresar la puntuación y/o escribir una reseña para el restaurante seleccionado. 7. El cliente completa el formulario y confirma el envío de la puntuación y/o reseña. 8. El sistema registra la puntuación y/o reseña asociada al restaurante seleccionado.
Poscondición :	La puntuación y/o reseña ha sido agregada correctamente al restaurante seleccionado y queda registrada en el sistema.
Excepciones:	<p>El sistema no permite añadir una reseña sin una puntuación.</p> <p>El sistema no permite añadir una reseña de un restaurante si no se ha realizado un pedido en él.</p> <p>El sistema no permite realizar reseñas sin puntuación y sin comentarios.</p>
Flujo Alternativo:	Si el cliente intenta añadir una reseña sin una puntuación, el sistema muestra un mensaje de error solicitando que se incluya una puntuación.

CU nrº	CU-014 Mirar Reseñas para un Restaurante
Actores Principales:	Usuario (Cliente, Partner)
Descripción:	El usuario desea mirar las reseñas y puntuaciones dejadas por otros usuarios para un restaurante específico.
Dependencia:	El restaurante debe estar registrado en el sistema y debe haber al menos una reseña disponible para el restaurante
Precondiciones:	El usuario ha iniciado sesión en la aplicación. Que exista al menos un restaurante dado de alta en el sistema en el cual se ha realizado alguna reseña.
Flujo Principal:	<ol style="list-style-type: none"> 1. El usuario accede a la sección de restaurantes en el sistema. 2. El usuario accede a la sección de búsqueda que ofrece la aplicación para buscar restaurantes en la aplicación. 3. El usuario introduce el nombre del restaurante que desea buscar. 4. El sistema muestra una lista de restaurantes que coinciden con la búsqueda del usuario. 5. El cliente selecciona el restaurante específico para el que desea ver las reseñas. 6. El sistema muestra las reseñas y puntuaciones dejadas por otros usuarios para el restaurante seleccionado.
Poscondición :	El usuario ha visualizado las reseñas y puntuaciones para el restaurante seleccionado.
Excepciones:	Si no hay reseñas disponibles para el restaurante seleccionado, el sistema mostrará un mensaje indicando que no hay reseñas disponibles.
Flujo Alternativo:	

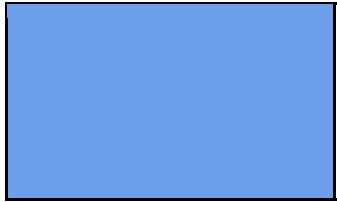
CU nrº	CU-015 Mirar Reseñas para un Plato
Actores Principales:	Usuario (Cliente, Partner)
Descripción:	El cliente desea mirar las reseñas y puntuaciones dejadas por otros usuarios para un plato específico.
Dependencia:	
Precondiciones:	El usuario ha iniciado sesión en la aplicación. Que exista al menos un restaurante dado de alta en el sistema en el cual se ha realizado alguna reseña.
Flujo Principal:	<ol style="list-style-type: none"> 1. El usuario accede a la sección de restaurantes en el sistema. 2. El usuario accede a la sección de búsqueda que ofrece la aplicación para buscar restaurantes en la aplicación. 3. El usuario introduce el nombre del restaurante que desea buscar. 4. El sistema muestra una lista de restaurantes que coinciden con la búsqueda del usuario. 5. El usuario selecciona el restaurante específico y entrará en los menús de los restaurantes. 6. El cliente accede a la sección de búsqueda de platos en la aplicación. 7. El cliente introduce el nombre o la categoría del plato que desea buscar. 8. El sistema muestra una lista de platos que coinciden con la búsqueda del cliente. 9. El cliente selecciona el plato específico para el que desea ver las reseñas. 10. El sistema muestra las reseñas y puntuaciones dejadas por otros usuarios para el plato seleccionado.
Poscondición :	El usuario ha visualizado las reseñas y puntuaciones para el restaurante seleccionado.
Excepciones:	Si no hay reseñas disponibles para el restaurante seleccionado, el sistema mostrará un mensaje indicando que no hay reseñas disponibles.
Flujo Alternativo:	

Casos de Uso del perfil del Responsable de cocina

CU nrº	CU-016 Gestionar Pedidos en la Cocina
Actores Principales:	Responsable de cocina
Descripción:	Cómo el responsable de cocina gestiona los pedidos recibidos y supervisa su preparación en la cocina.
Dependencia:	
Precondiciones:	El responsable de cocina ha iniciado sesión en la aplicación y tiene acceso a la sección de gestión de pedidos.
Flujo Principal:	<ol style="list-style-type: none"> 1. El responsable de cocina accede a la sección de gestión de pedidos en la aplicación. 2. El sistema muestra una lista de todos los pedidos recibidos, ordenados por hora de recepción. 3. Para cada pedido, el sistema muestra los platos solicitados y sus respectivos ingredientes. 4. El responsable de cocina revisa los detalles del pedido y cambia su estado a "En preparación". 5. El sistema actualiza el estado del pedido. 6. Una vez que el pedido está preparado, el responsable de cocina cambia su estado a "Listo para entrega". 7. El sistema actualiza el estado del pedido. 8. El responsable de cocina selecciona un repartidor disponible para entregar el pedido.
Poscondición :	El responsable de cocina ha gestionado con éxito los pedidos recibidos y entrega el pedido al repartidor correspondiente.
Excepciones:	
Flujo Alternativo:	

Casos de Uso del perfil del Repartidor

CU nrº	CU-017 Entregar Pedidos como Repartidor
Actores Principales:	Repartidor
Descripción:	Cómo un repartidor entrega los pedidos asignados a través de la aplicación de reparto de comida.
Dependencia:	El repartidor debe haber iniciado sesión en la aplicación y tener acceso a la sección de gestión de pedidos asignados.
Precondiciones:	El repartidor ha iniciado sesión en la aplicación y tiene acceso a la sección de gestión de pedidos asignados.
Flujo Principal:	<ol style="list-style-type: none"> 1. El sistema le notifica al repartidor que tiene un nuevo pedido para entregar. 2. El repartidor accede a la sección de gestión de pedidos asignados en la aplicación. 3. El sistema muestra una lista de todos los pedidos asignados al repartidor, incluyendo los detalles como la dirección de entrega, la ubicación en el mapa y los platos solicitados. 4. El repartidor selecciona el primer pedido de la lista para entregar. 5. El sistema muestra la dirección de entrega en un mapa incorporado en la aplicación. 6. Una vez en la ubicación, el repartidor se asegura de que el pedido sea entregado al destinatario correcto. 7. El repartidor autentica la entrega solicitando al destinatario que confirme su usuario y proporcionando un código de entrega único. 8. El repartidor marca el código de entrega único. 9. El sistema confirma la autenticación. 10. El repartidor marca el pedido como entregado en la aplicación. 11. El sistema actualiza el estado del pedido y lo elimina de la lista de pedidos asignados al repartidor.
Poscondición :	El repartidor ha entregado con éxito el pedido asignado y ha actualizado su estado en la aplicación.
Excepciones:	<p>Falla en la autenticación de entrega:</p> <ul style="list-style-type: none"> • Si el código de entrega no coincide, el repartidor se comunica con el restaurante y el cliente para coordinar una solución. • El sistema muestra un mensaje de error indicando que la autenticación ha fallado y solicita una nueva verificación o una acción correctiva. <p>Pedido no entregable:</p> <ul style="list-style-type: none"> • Si un pedido no puede ser entregado por algún motivo (dirección incorrecta, cliente no disponible, etc.), el repartidor debe informar al restaurante y al cliente para coordinar una solución. • El repartidor puede marcar el pedido como "No entregado" y proporcionar una razón en la aplicación. • El sistema actualiza el estado del pedido a "No entregado" y notifica al restaurante y al cliente.
Flujo Alternativo:	<p>Entrega sin autenticación por error técnico:</p> <ol style="list-style-type: none"> 1. Si el sistema tiene un fallo técnico y no puede autenticar la entrega mediante el código, el repartidor debe

- 
- tomar una fotografía del cliente recibiendo el pedido y comunicarlo al restaurante.
 - 2. El partner marca el pedido como entregado y el sistema actualiza el estado del pedido con la foto adjunta como prueba de entrega.

Diseño de la Aplicación

Guía de Estilo

Esta guía de estilo tiene como objetivo proporcionar directrices claras para el diseño visual del sistema web de reparto de comida. Su propósito es asegurar la coherencia interfaces.

Logo



Colores Principales

#F28C28	#FFFFFF	#333333
Este color se utiliza para botones principales, encabezados, y otros elementos interactivos clave.	Fondo principal de la aplicación, utilizado para crear un contraste limpio y moderno.	Para texto principal y elementos de navegación.

Componentes de Interfaz

Botones

- **Botón Principal:**
 - Color de fondo: #F28C28 ()
 - Color de texto: #FFFFFF ()
 - Bordes: Redondeados con 5px de radio.
 - Tamaño del texto: 16px, en negrita.
- **Botón Secundario:**
 - Color de fondo: #FFFFFF ()
 - Color de texto: #F28C28 ()
 - Borde: 2px de grosor, color #F28C28.

Formularios

- **Cajas de texto:**
 - Bordes: 1px de grosor,
 - Color #CCCCCC ().
 - Tamaño 14px.
- **Etiquetas de campo:**
 - Fuente: Roboto Regular, 14px, color #333333 ().
 - Alineación: Izquierda, encima del campo correspondiente.

Tipos de Mensajes:

- **Mensajes de Éxito:**
 - **Propósito:** Informar al usuario que una acción se ha completado correctamente.
 - **Estilo:**
 - **Color de fondo:** Verde (#28a745 o similar).
 - **Texto:** Los mensajes que se le proporciona al usuario tiene que ser sencillos de entender y coherentes.
- **Mensajes de Error:**
 - **Propósito:** Informar al usuario que ha ocurrido un problema que ha impedido completar una acción.
 - **Estilo:**
 - **Color de fondo:** Rojo (#dc3545 o similar).
 - **Texto:** Los mensajes que se le proporciona al usuario tiene que ser sencillos de entender y coherentes.

Menú Lateral

- **Fondo del Menú:**
 - Color: #F5F5F5. ()
 - Texto: #333333 ()
 - iconos: #F28C28. ()
- **Elementos del Menú:**
 - Color de fondo al seleccionar: #E0E0E0. ()
 - Iconos: A la izquierda del texto, tamaño 20px.

Iconografía

Usar un icono de advertencia de Font Awesome.

Imágenes

- **Imágenes de los platos y de los restaurantes:**
 - Tamaño máximo: 150x150px.
 - Formato: .png o .jpg

Navegación

- **Estructura de Navegación:**
 - Menú lateral fijo para facilitar la navegación entre las secciones principales.
- **Enlaces de Páginas:**

Los enlaces deben estar subrayados al pasar el ratón y cambiar su color a #F28C28 ()

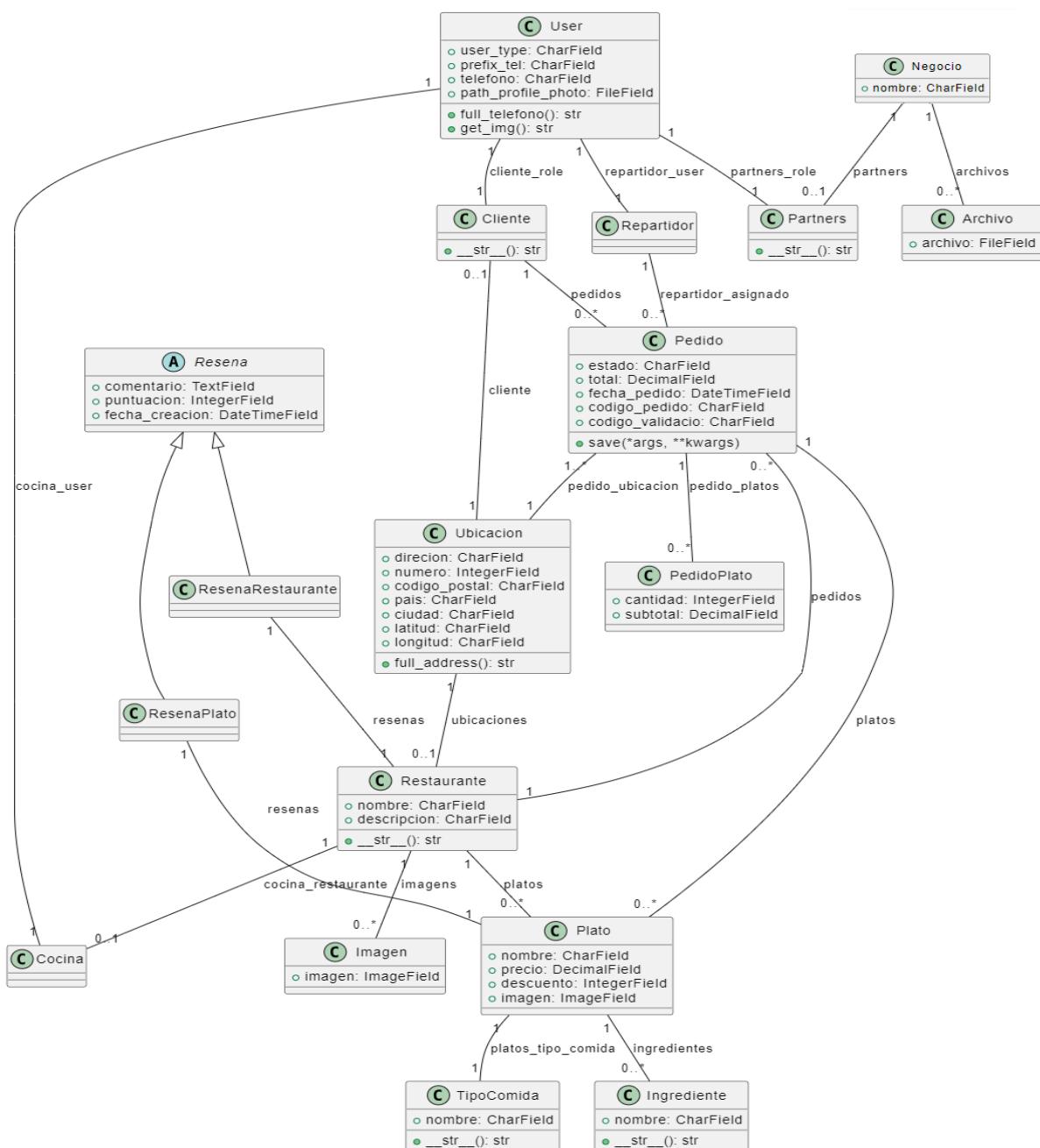
Creación de Prototipos de Interfaz de Usuario

Para la creación de los prototipos de las diferentes interfaces de usuarios del sistema he utilizando la herramienta [Marvel](#), para diseñar los prototipos interactivos que reflejen la experiencia del usuario esperada.

- **Para el prototipo de interfaz del Partner:**
<https://marvelapp.com/prototype/c97aee3>
- **Para el prototipo de interfaz del Encargado de Cocina:**
<https://marvelapp.com/prototype/31bi278g>
- **Para el prototipo de interfaz del Repartidor:**
<https://marvelapp.com/prototype/7e035a2>
- **Para el prototipo de interfaz del Cliente:**
<https://marvelapp.com/prototype/10adg92a>

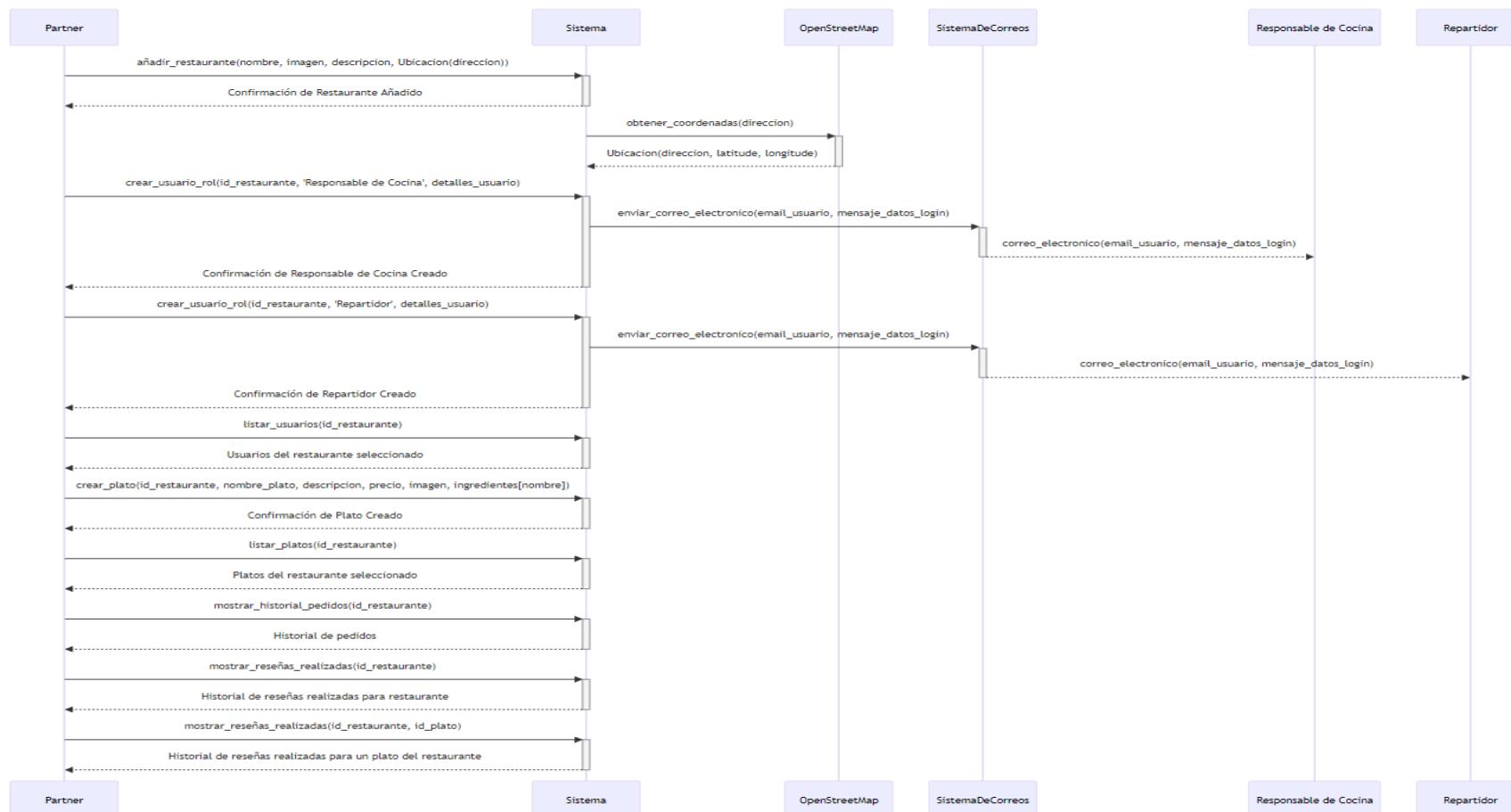
Diseño de Relaciones de la Base de Datos

En el diseño del modelo de datos para la aplicación, se destacan varias características clave que optimizan la funcionalidad y flexibilidad del sistema. Primero, el modelo abstracto **Resena** se utiliza para facilitar la creación y gestión de reseñas en diferentes contextos, como restaurantes y platos, permitiendo una reutilización eficiente de la funcionalidad de reseñas sin necesidad de duplicar código. Además, los modelos **Pedido** y **PedidoPlato** son fundamentales para manejar las órdenes de manera efectiva. **Pedido** gestiona el estado, cliente, ubicación y total de cada pedido, mientras que **PedidoPlato** establece una relación entre pedidos y platos, permitiendo la asociación de múltiples platos a cada pedido y el cálculo del total correspondiente. Esto asegura una gestión precisa de los elementos del pedido y su impacto en la facturación. En cuanto a los usuarios, se extiende el modelo **AbstractUser** para incluir un campo adicional de tipo de usuario, con opciones como Partner, Repartidor, Cocina, Cliente y Administrador. Esta extensión permite una gestión más específica y flexible de los roles dentro del sistema, con la posibilidad de añadir campos o funciones adicionales en el futuro según sea necesario, adaptándose así a los requisitos cambiantes del negocio y mejorando la personalización del sistema.

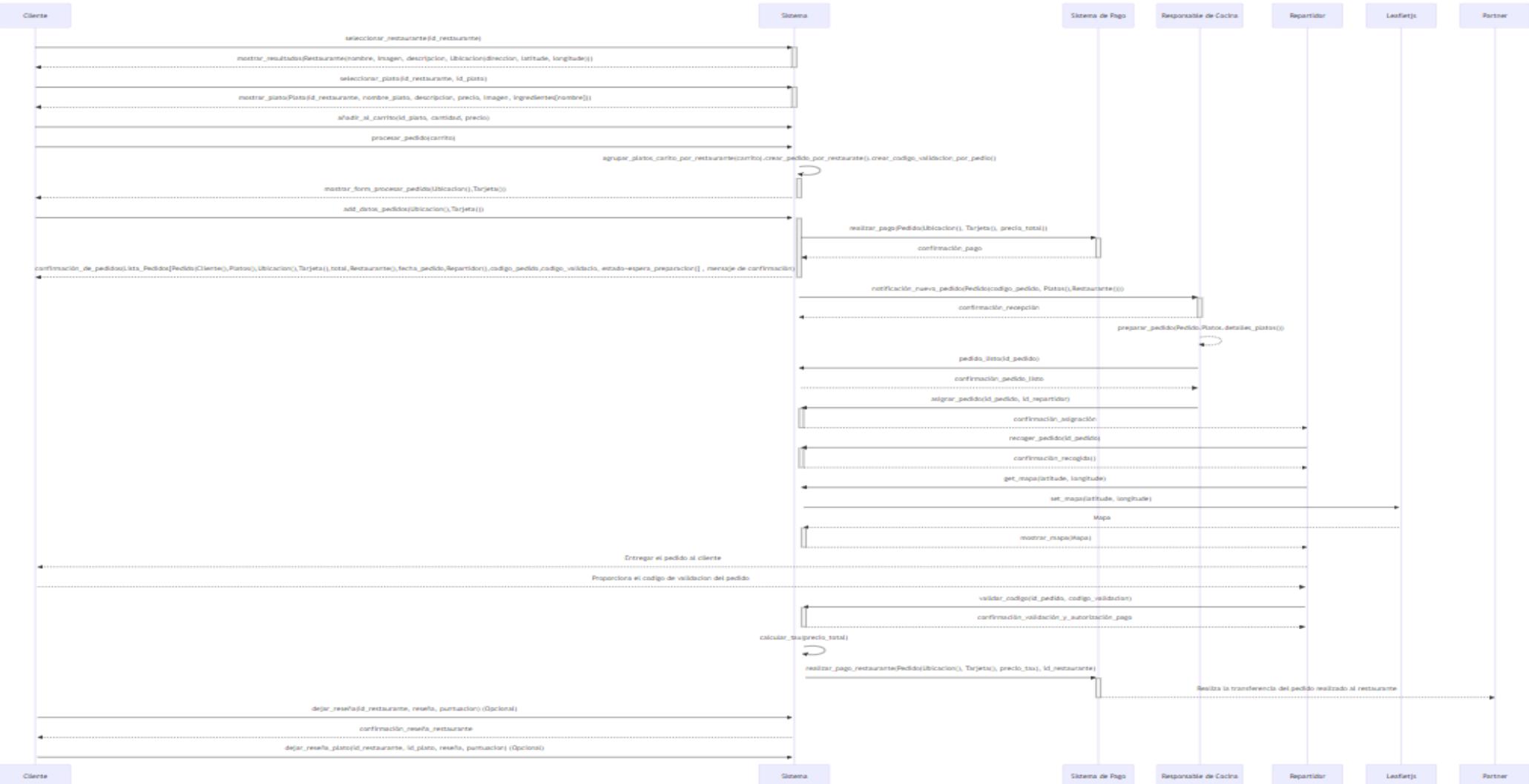


Diagramas de Secuencia del Sistema

La iteración entre Partner y el sistema para administrar los restaurantes, los trabajadores, los diferentes platos, pedidos y reseñas de cada restaurante.



Todos los pasos clave del proceso de pedido y entrega de comida del sistema DeliverEat.



Desarrollo del sistema web

Elección de tecnología

He optado por desarrollar un sistema web porque es accesible desde cualquier dispositivo con un navegador, incluyendo computadoras, tabletas y teléfonos móviles, lo que elimina la necesidad de desarrollar y mantener aplicaciones separadas para cada plataforma.

Además, esto permite que el sistema sea utilizado por una base de usuarios más amplia, independientemente del sistema operativo que utilicen, como iOS, Android o Windows. El desarrollo web también es más rápido y rentable, evitando los procesos de aprobación en las tiendas de aplicaciones y facilitando actualizaciones más eficientes. Esta elección estratégica reduce los costos y acelera el tiempo de implementación, haciendo del sistema web una solución más escalable y flexible.

Elección del Lenguaje

Elegí Django para desarrollar este sistema por varias razones clave. Django facilita el manejo de la base de datos a través de su ORM (Object-Relational Mapping), permitiendo realizar consultas y operaciones CRUD usando objetos Python en lugar de escribir SQL, lo que simplifica el desarrollo y el mantenimiento. Además, cuenta con vistas genéricas que agilizan la creación de funcionalidades comunes, reduciendo la cantidad de código que se debe escribir. Django también ofrece una curva de aprendizaje moderada y una documentación extensa, lo que facilita el proceso de desarrollo. Su sistema de autenticación integrado (Django auth) garantiza la seguridad y simplifica la implementación de características como la gestión de usuarios. Además, al ser compatible con Stripe, permite la integración de sistemas de pago directamente en el código Python. Finalmente, Django facilita la realización de pruebas BDD (Behavior Driven Development) utilizando herramientas como Behave, lo que mejora la calidad y confiabilidad del sistema.

Instalar Django y configurar el entorno de desarrollo

Para comenzar a desarrollar el sistema web de pedidos de comida a domicilio, se requirió la instalación de python en nuestro sistema y luego se creó un entorno virtual utilizando el comando “**python -m venv venv**”.

Luego, se activó el entorno virtual mediante “**.\venv\Scripts\activate**” y se instaló Django mediante “**pip install django**”.

Después de verificar la instalación de Django, se creó un nuevo proyecto Django con “**django-admin startproject sistemdeliverat**”.

El entorno virtual lo utilice para poder instalar todos los paquetes y librerías necesarias para mi proyecto de manera independiente. Esto permitió instalar y gestionar las dependencias del proyecto de manera independiente del entorno del sistema operativo.

Interfaz de Usuario

He decidido utilizar la plantilla AdminLTE para dar estilo a mi sistema web, ya que proporciona un diseño limpio, moderno y completamente adaptable a diferentes dispositivos. AdminLTE incluye una variedad de componentes listos para usar, como menús, tablas y formularios, lo que acelera significativamente el desarrollo del frontend.

Para estructurar mi proyecto de manera eficiente, implementé una plantilla base llamada **base_adminlte.html** en la cual se incluyen todos los elementos comunes, como la barra de navegación, el encabezado y el pie de página. Esto me permitió reutilizar código en diferentes partes del sistema, manteniendo la consistencia en la interfaz de usuario y facilitando el mantenimiento a largo plazo.

También al usar una plantilla base asegura que cualquier cambio en el diseño global se apliquen de manera uniforme en todo el sistema.

Además, el uso de bloques (`{% block %}`) en la plantilla base permitió a cada vista especificar su propio contenido sin alterar la estructura general del sitio.

El bloque `{% block content %}` se utiliza para injectar el contenido específico de cada vista, mientras que otros bloques como `{% block style %}` y `{% block script %}` permiten personalizar el estilo y los scripts en cada página. Este enfoque modularizado facilita enormemente la escalabilidad del proyecto y mantiene el código organizado.

Funcionalidades clave de la aplicación

Funcionalidad del Carrito de Compras

El carrito de compras es una parte esencial del sistema de reparto de comida, permitiendo a los usuarios gestionar los platos que desean pedir antes de realizar la compra. La clase **CarritoDeCompras** fue desarrollada para realizar esta funcionalidad de manera eficiente y se integra directamente con las vistas que gestionan las interacciones del usuario con el carrito.

Propósito del Carrito de Compras

El carrito de compras sirve como una estructura temporal para cada usuario logueado en la que los usuarios pueden agregar, visualizar, modificar y eliminar los platos que desean pedir mientras navegan por las ofertas de los restaurantes.

El principal objetivo al crear la clase **CarritoDeCompras** fue encapsular la lógica relacionada con la gestión del carrito dentro de una clase dedicada, asegurando una separación clara entre la lógica de negocio. Esta separación facilita el mantenimiento, la prueba y la escalabilidad del código.

Descripción de las Funciones de la Clase

La clase **CarritoDeCompras** incluye las siguientes funciones clave:

- **__init__(self, request)**: Este constructor inicializa el carrito de compras utilizando la sesión del usuario. Si el carrito no está presente en la sesión, se crea un nuevo diccionario vacío que almacena los platos y sus cantidades.
- **agregar_plato(self, plato_id)**: Permite agregar un plato al carrito. Si el plato ya existe, incrementa su cantidad; de lo contrario, lo añade al carrito con una cantidad inicial de uno.
- **quitar_plato(self, plato_id)**: Reduce la cantidad de un plato específico en el carrito. Si la cantidad del plato es uno, lo elimina completamente del carrito.
- **eliminar_plato(self, plato_id)**: Elimina un plato del carrito sin importar su cantidad actual. Esto es útil cuando el usuario decide no incluir un plato en su pedido final.
- **guardar_carrito(self)**: Guarda el estado actual del carrito en la sesión del usuario, asegurando que cualquier modificación se refleje inmediatamente.
- **obtener_carrito(self)**: Devuelve el contenido actual del carrito, lo que permite a otras partes del sistema acceder a la información de los platos seleccionados por el usuario.
- **borrar_carrito(self)**: Elimina todo el contenido del carrito, dejando al usuario con un carrito vacío. Es útil cuando el usuario decide cancelar o reiniciar su pedido.

```
class CarritoDeCompras:

    def __init__(self, request):
        # Inicializamos el carrito de compras

        self.session = request.session
        # Obtenemos el carrito de la sesión, si no existe lo inicializamos como un diccionario vacío
        if 'carrito' not in self.session:
            self.session['carrito'] = {}
        self.carrito = self.session['carrito']


    def agregar_plato(self, plato_id):
        # Agregamos un plato al carrito
        if str(plato_id) not in self.carrito:
            # Si el plato no está en el carrito, lo agregamos y establecemos la cantidad en 1
            self.carrito[plato_id] = {'cantidad': 1}

        else:
            # Si el plato ya está en el carrito, incrementamos la cantidad en 1
            self.carrito[plato_id]['cantidad'] += 1
        self.guardar_carrito()


    def quitar_plato(self, plato_id):
        plato_id = str(plato_id)
        if self.carrito[plato_id]['cantidad'] <= 1:
            self.eliminar_plato(plato_id)
        else:
            self.carrito[plato_id]['cantidad'] -= 1
        self.guardar_carrito()

    def eliminar_plato(self, plato_id):
        if plato_id in self.carrito:
            del self.carrito[plato_id]
        self.guardar_carrito()

    def guardar_carrito(self):
        self.session.modified = True

    def obtener_carrito(self):
        return self.carrito

    def borrar_carrito(self):
        self.carrito = {}
        self.guardar_carrito()
```

```

def eliminar_plato(self, plato_id):
    # Eliminamos un plato del carrito
    if str(plato_id) in self.carrito:
        # Si el plato está en el carrito, lo eliminamos
        del self.carrito[plato_id]
    self.guardar_carrito()

def guardar_carrito(self):
    # Guardamos el carrito en la sesión
    self.session['carrito'] = self.carrito
    # Marcamos la sesión como modificada para asegurarnos de que se guarde
    self.session.modified = True

def obtener_carrito(self):
    # Obtenemos el carrito actual
    return self.carrito

def borrar_carrito(self):
    self.session['carrito'] = {}
    self.session.modified = True

```

Uso de la Clase **CarritoDeCompras** en las Vistas

La clase **CarritoDeCompras** se utiliza en diversas vistas del sistema para permitir la interacción del usuario con el carrito de compras. A continuación, se explica cómo se integra la clase con las vistas correspondientes:

- **agregar_al_carrito(request, plato_id)**: Esta vista permite al usuario agregar un plato al carrito. Utiliza la clase CarritoDeCompras para agregar el plato identificado por plato_id y luego redirige al usuario a la página anterior (HTTP_REFERER), lo que proporciona una experiencia fluida sin interrupciones.

```

@login_required
@web_access_type_required("cliente")
def agregar_al_carrito(request, plato_id):
    carrito = CarritoDeCompras(request)
    carrito.agregar_plato(str(plato_id))
    return redirect(request.META.get('HTTP_REFERER', '/'))

```

- **carrito_lista(request)**: Esta vista muestra al usuario el contenido actual del carrito. La clase CarritoDeCompras se utiliza para obtener los platos en el carrito, calcular los totales por restaurante y el total general, que luego se presentan en la página de carrito.

```

@login_required
@web_access_type_required("cliente")
def carrito_lista(request):
    carrito = CarritoDeCompras(request).obtener_carrito()
    platos_en_carrito = Plato.objects.filter(id__in=carrito.keys())
    return render(request, 'cliente/carrito/carrito_lista.html', {
        'platos': platos_en_carrito,
    })

```

- **eliminar_plato_carrito(request, plato_id)**: Esta vista permite eliminar un plato específico del carrito. Una vez eliminado, redirige al usuario de nuevo a la página del carrito.

```
@login_required
@web_access_type_required("cliente")
def eliminar_plato_carrito(request, plato_id):
    carrito = CarritoDeCompras(request)
    carrito.eliminar_plato(str(plato_id))
    return redirect('myapp:página_del_carrito')
```

- **quitar_plato_carrito(request, plato_id)**: Esta vista reduce la cantidad de un plato específico en el carrito. Si la cantidad llega a cero, el plato se elimina del carrito, y luego redirige a la página del carrito.

```
@login_required
@web_access_type_required("cliente")
def quitar_plato_carrito(request, plato_id):
    carrito = CarritoDeCompras(request)
    carrito.quitar_plato(str(plato_id))
    return redirect('myapp:página_del_carrito')
```

- **borrar_carrito(request)**: Esta vista vacía por completo el carrito de compras del usuario y redirige a la página del carrito, mostrando que ya no contiene platos.

```
@login_required
@web_access_type_required("cliente")
def borrar_carrito(request):
    carrito = CarritoDeCompras(request)
    carrito.borrar_carrito()
    return redirect('myapp:página_del_carrito')
```

Justificación del Uso de la Clase CarritoDeCompras

La decisión de utilizar la sesión del usuario en la clase **CarritoDeCompras**, en lugar de implementar un modelo que almacene el carrito en la base de datos, se basa en varias ventajas importantes:

- Permite a los usuarios agregar, quitar y eliminar platos del carrito de manera sencilla.
- Garantiza que el carrito se guarde en la sesión del usuario para que esté disponible en cualquier momento.
- Permite el controlar y personalizar la lógica de negocio del carrito de compras.
- No tener que guardar datos en la base de datos. Al almacenar el carrito de compras en la sesión del usuario, no es necesario guardar cada operación del carrito en la base de datos.
- Reducir la carga en la base de datos. Al no tener que guardar cada cambio en el carrito de compras en la base de datos, se reduce significativamente la carga en la base de datos

Funcionalidad de Procesamiento de Pedidos

En el sistema de reparto de comida, los clientes pueden seleccionar y añadir al carrito de compras platos de distintos restaurantes. De esta manera ofrece una experiencia flexible y cómoda para los usuarios, permitiéndoles realizar pedidos a múltiples restaurantes en una sola operación.

Cada vez que un cliente decide finalizar su compra y realizar el pedido, el sistema agrupa automáticamente los platos en función del restaurante al que pertenecen y genera un pedido separado para cada uno. Este proceso se realiza de manera automática y transparente para el usuario, quien solo necesita realizar una confirmación y un único pago.

Descripción del Proceso de Pedidos

A continuación, se detalla el funcionamiento interno de la función **procesar_pedido_from()**, que maneja este proceso.

El primer paso en el procesamiento de pedidos es revisar los datos de ubicación proporcionados por el cliente. Si se pueden obtener las coordenadas geográficas de la dirección mediante el sistema Geocoder, el proceso continúa; de lo contrario, se cancela y el cliente es redirigido al formulario con un mensaje de error. Si la verificación de los datos enviados por POST es correcta, el siguiente paso es recuperar la sesión del carrito de compras asociada al usuario que ha iniciado sesión. Esta sesión contiene los platos seleccionados por el usuario junto con la cantidad de cada uno.

```
carrito = CarritoDeCompras(request).obtener_carrito()
```

Aquí, el carrito de compras es una instancia que contiene toda la información relacionada con los productos que el usuario ha decidido comprar.

Una vez que se obtiene la información del carrito, el sistema procede a buscar los platos seleccionados en la base de datos para obtener los detalles completos de cada uno. Posteriormente, estos platos se agrupan en función del restaurante al que pertenecen. Este paso es crucial, ya que permite crear pedidos individuales para cada restaurante, lo que facilita la gestión y el seguimiento de los mismos.

```
platos_por_restaurante = {} # Diccionario para agrupar los platos
                             # por restaurante

# Agrupar los platos por restaurante
plato_ids = carrito.keys()
platos = Plato.objects.filter(id__in=plato_ids)

for plato in platos:
    restaurante_id = plato.restaurante.id
    if restaurante_id not in platos_por_restaurante:
        platos_por_restaurante[restaurante_id] = {'platos': [], 'total': 0}
    platos_por_restaurante[restaurante_id]['platos'].append(plato)
    platos_por_restaurante[restaurante_id]['total'] += descuento(plato) * carrito[str(plato.id)][
        'cantidad']
```

En este fragmento de código, se recorre cada plato para determinar a qué restaurante pertenece. Los platos se agrupan en un diccionario, donde cada clave representa un restaurante y cada valor es una lista de platos pertenecientes a ese restaurante, junto con el total acumulado para dicho pedido.

Después de agrupar los platos, el sistema procede a la creación de los pedidos correspondientes. Dado que un cliente puede realizar pedidos a múltiples restaurantes en una sola operación, es fundamental que este proceso se realice de manera conjunta y segura. Para garantizar la integridad de los datos, se utiliza una transacción atómica. Esto significa que, si ocurre algún error durante la creación de uno de los pedidos, ningún pedido será creado, lo que permite manejar adecuadamente cualquier error.

```
# Crear los pedidos dentro de una transacción
with transaction.atomic():
    for restaurante_id, data in platos_por_restaurante.items():
        restaurante = Restaurante.objects.get(id=restaurante_id)
        cliente = Cliente.objects.get(user=request.user)
        platos_en_pedido = Plato.objects.filter(id__in=data['platos'])

        total_pedido = data['total']
        total_trasferencia += tax(total_pedido)
        restaurantes=restaurantes+" "+ restaurante.nombre

        # Crear una nueva ubicación para cada pedido
        ubicacion_pedido = Ubicacion.objects.create(
            direccion=request.POST.get('direccion'),
            numero=request.POST.get('numero'),
            codigo_postal=request.POST.get('codigo_postal'),
            pais=request.POST.get('pais'),
            ciudad=request.POST.get('ciudad'),
            latitud=latitud,
            longitud=longitud
        )

        # Crear el pedido con la ubicación única
        pedido = Pedido.objects.create(
            cliente=cliente,
            ubicacion=ubicacion_pedido,
            total=total_pedido,
            restaurante=restaurante
        )

        for plato in platos_en_pedido:
            cantidad = carrito[str(plato.id)]['cantidad']
            subtotal = descuento(plato) * cantidad
            PedidoPlato.objects.create(
                pedido=pedido,
                plato=plato,
                cantidad=cantidad,
                subtotal=subtotal
            )
```

En este fragmento de código, se crea un pedido separado para cada restaurante, agrupando los platos seleccionados por el usuario y calculando el total correspondiente. Además, se establece una transacción atómica para garantizar que todos los pedidos se creen de manera segura y conjunta; si ocurre algún error, la transacción se revierte, y no se crea ningún pedido. Durante este proceso, también se genera una ubicación

para cada pedido, utilizando la dirección proporcionada por el usuario y obteniendo las coordenadas geográficas a través de una clase personalizada llamada **Geocoder**.

Una vez que se ha completado con éxito el proceso de creación de los pedidos, el carrito de compras del usuario se limpia para asegurar que no queden elementos residuales. Esto se logra vaciando el contenido del carrito y actualizando la sesión con el carrito vacío, marcando la sesión como modificada para reflejar este cambio. Despues de limpiar el carrito, el usuario es redirigido a una página que muestra los pedidos realizados, acompañado de un mensaje de éxito que indica que el proceso de pedido se ha completado exitosamente.

```
# Limpiar el carrito de compras después de realizar los pedidos
carrito.clear()

# Actualizar la sesión con el carrito vacío
request.session['carrito'] = {}
request.session.modified = True

messages.success(request, "El proceso de pedido se realizó exitosamente.")
return redirect('myapp:pedidos_realizados')
```

Obtención de Coordenadas Geográficas

Para mejorar la precisión en la entrega, se utiliza la clase **Geocoder** para convertir la dirección ingresada por el usuario en coordenadas geográficas (latitud y longitud). Esto facilita la localización exacta del lugar de entrega.

```
from geopy.geocoders import Nominatim

class Geocoder:
    def __init__(self):
        self.geolocator = Nominatim(user_agent="sistemadelivereat")
        self.error = None

    def obtener_coordenadas(self, direccion):
        try:
            location = self.geolocator.geocode(direccion)
            if location:
                return (location.latitude, location.longitude)
            else:
                return None
        except Exception as e:
            self.error = str(e)
            return None
```

La clase Geocoder utiliza la API de geocodificación de Nominatim para convertir la dirección proporcionada en coordenadas. Si la dirección no puede ser encontrada, se **retorna None** y se registra un error.

Funcionalidad de Sistema de Pago

La funcionalidad de pago es una parte fundamental del sistema, ya que permite procesar transacciones de manera segura y eficiente. En este proyecto, se ha implementado un sistema de pagos utilizando **Stripe**, una plataforma que facilita la gestión de pagos online. Durante el desarrollo del Trabajo de Fin de Grado (TFG), se ha utilizado el modo de prueba de Stripe, lo que permite realizar transacciones simuladas sin incurrir en gastos reales. Para las pruebas, se han utilizado tarjetas de prueba proporcionadas por Stripe.

Integración del Sistema de Pago con Stripe

Para integrar Stripe en la aplicación, fue necesario instalar su librería mediante el siguiente comando: `pip install stripe`

Luego, configuré las claves de API de Stripe en el archivo de configuración `settings.py` de Django:

```
STRIPE_SECRET_KEY = "la_clave_secreta" # Clave secreta proporcionada por Stripe para pruebas  
STRIPE_PUBLIC_KEY = "tu_clave_publica" # Clave pública proporcionada por Stripe para pruebas
```

El formulario de pago se encuentra en el archivo `hacer_pedido.html`, donde el usuario introduce su tarjeta de crédito y la dirección de envío. Para el campo de la tarjeta, se utilizó el siguiente HTML proporcionado por la API de Stripe:

```
<div class="form-group row">  
    <label for="card-element" class="col-sm-3 col-form-label">Tarjeta de crédito o débito</label>  
    <div class="col-sm-9">  
        <div id="card-element" class="form-control">  
            <!-- Stripe.js will create this element -->  
        </div>  
    </div>  
    <div class="col-sm-3 col-form-label"></div>  
    <small id="emailHelp" class="form-text text-danger">  
        <div id="card-errors" role="alert"></div>  
    </small>  
</div>
```

Además, se añadió el siguiente script en el mismo archivo HTML para manejar la creación del token de pago mediante **Stripe.js**:

```
<script src="https://js.stripe.com/v3/"></script>  
<script>  
    var stripe =  
        Stripe('pk_test_51PMYfiGF2SGr9v2EBwtrcVsJN6QWLD6hEi07dNc4vVHpDMaxLX41hthGjo92MKkTHmYhB4IXKhXqSt2iimRC4OX1  
        001stOu6Pm');  
    var elements = stripe.elements();  
    var card = elements.create('card');  
    card.mount('#card-element');  
  
    card.addEventListener('change', function(event) {  
        var displayError = document.getElementById('card-errors');  
        if (event.error) {  
            displayError.textContent = event.error.message;
```

```

    } else {
        displayError.textContent = '';
    }
});

var form = document.getElementById('payment-form');
form.addEventListener('submit', function(event) {
    event.preventDefault();
    stripe.createToken(card).then(function(result) {
        if (result.error) {
            var errorElement = document.getElementById('card-errors');
            errorElement.textContent = result.error.message;
        } else {
            stripeTokenHandler(result.token);
        }
    });
});

function stripeTokenHandler(token) {
    var form = document.getElementById('payment-form');
    var hiddenInput = document.createElement('input');
    hiddenInput.setAttribute('type', 'hidden');
    hiddenInput.setAttribute('name', 'stripeToken');
    hiddenInput.setAttribute('value', token.id);
    form.appendChild(hiddenInput);
    form.submit();
}
</script>

```

Este script se encarga de gestionar la validación de la tarjeta y de crear un token de pago seguro que se envía al servidor para procesar el pago.

La lógica de procesamiento de pedidos y pagos está implementada en la vista **procesar_pedido_form**. En esta función, el cliente realiza el pago después de que se han generado los pedidos correspondientes a cada restaurante.

Primero, se aplica una tasa al total del pedido usando la función **tax**:

```

def tax(precio):
    precio_con_tax = precio + ((precio * 10 ) / 100 ) + ((precio * 3 ) / 100 )
    return precio_con_tax.quantize(Decimal('0.00') + Decimal('0.50'))

```

Se verifica si el token de Stripe está presente en la solicitud POST. Si no lo está, se muestra un mensaje de error:

```

stripe_token = request.POST.get('stripeToken')
if not stripe_token:
    messages.error(request, "Error con el token de Stripe. Inténtalo de nuevo.")
    return redirect('myapp:procesar_pedido_form')

```

Si el token es válido, se procede a crear un cargo en Stripe:

```
try:
    charge = stripe.Charge.create(
        amount=int(total_trasferencia * 100), # Stripe maneja las cantidades en centavos
        currency='eur',
        description=f'Pago del pedido',
        source=stripe_token
    )
except stripe.error.StripeError as e:
    messages.error(request, f"Error en el procesamiento del pago para los restaurantes: {restaurantes}:\n{e.error.message}")
    return redirect('myapp:procesar_pedido_form')
```

Si el cargo es exitoso, se procede con la creación de los pedidos y los registros correspondientes en la base de datos.

Justificación

La integración de una API de pagos como Stripe hace que las transferencias sean más seguras y confiables. Además, Stripe ofrece un modo de prueba que es extremadamente útil para el desarrollo del TFG, ya que permite realizar pruebas sin incurrir en costos reales, asegurando que el sistema funcione correctamente antes de ponerlo en producción.

Funcionalidad de Validación de Pedido

Una de las funcionalidades clave implementadas en el sistema es la **validación de pedido**. Esta característica garantiza que el pedido sea entregado al cliente correcto, agregando una capa adicional de seguridad y confiabilidad al proceso de entrega.

Cuando el repartidor llega a la dirección de entrega, solicita al cliente un código de validación que solo el cliente tiene visible en su perfil. Este código debe ser introducido en el sistema por el repartidor a través de un formulario. Una vez que el código es validado correctamente, el repartidor puede proceder con la entrega del pedido.

Descripción de Validación de Pedido

En el momento en que se crea un pedido, el sistema genera dos códigos únicos:

- **codigo_pedido**: Un identificador único que se utiliza para rastrear el pedido en el sistema.
- **codigo_validacion**: Un código único que solo el cliente puede ver y que debe proporcionar al repartidor al momento de la entrega para validar la autenticidad del pedido.

Ambos códigos se generan de manera aleatoria y son únicos, lo que significa que no hay posibilidad de duplicación. Estos códigos son almacenados en la base de datos como parte del modelo Pedido. A continuación, se muestra cómo se implementa este proceso en la clase Pedido:

```
class Pedido(models.Model):  
    # Otros campos relevantes del modelo  
    codigo_pedido = models.CharField(max_length=50, unique=True)  
    codigo_validacion = models.CharField(max_length=50, unique=True, null=True)  
  
    def save(self, *args, **kwargs):  
        if not self.codigo_pedido:  
            # Generar un código único para el pedido de 12 caracteres  
            unique_id_pedido = uuid.uuid4().hex[:12]  
            unique_id_validacion = uuid.uuid4().hex[:12]  
            self.codigo_pedido = unique_id_pedido  
            self.codigo_validacion = unique_id_validacion  
        super(Pedido, self).save(*args, **kwargs)
```

En este código, la función `save` se sobrescribe para asegurarse de que, al guardar un nuevo pedido, se generen estos códigos únicos si no se han generado previamente. La función `uuid.uuid4().hex[:12]` se utiliza para crear un identificador de 12 caracteres, garantizando la unicidad de cada código.

```
@login_required  
@web_access_type_required("repartidor")  
def validar_pedido(request, pedido_id):  
    pedido = get_object_or_404(Pedido, id=pedido_id)  
  
    if request.method == 'POST':  
        # Procesar los datos del formulario si son válidos  
        codigo_validacion = request.POST.get('codigo')
```

```

if codigo_validacion == pedido.codigo_validacion:
    # Código de validación correcto, puedes marcar el pedido como validado
    try:
        messages.success(request,"Transferencia realizada correctamente:")
        return redirect('myapp:validar_pedido', pedido_id=pedido_id)

    pedido.estado = 'entregado'
    pedido.save()

    return redirect('myapp:pedidos_repartidor', tipo_objeto='entregado')

except Exception as e:
    messages.error(request, "Error aqui"+ str(e))
    return redirect('myapp:validar_pedido', pedido_id=pedido_id)

else:
    messages.error(request, "El codigo de validación es incorrecto. Vuelva a intentarlo o
                           ponte en contacto con el restaurante.")
    return redirect('myapp:validar_pedido', pedido_id=pedido_id)

ruta_pagina = [
{
    'text': "Lista de pedidos para entregar",
    'link': "myapp:list_restaurantes",
},
{
    'text': "Validar Pedido " + pedido.codigo_pedido,
    'link': "",
}
]

title_pagina = [
{
    'label_title': "Validar Pedido",
    'title_card': "Validar Pedido con el cliente "+pedido.cliente.user.get_full_name(),
}
]

context = {
    'ruta_pagina': ruta_pagina,
    'title_pagina': title_pagina
}

return render(request, 'repartidor/from_validacion_pedido.html', context)

```

El proceso de validación se lleva a cabo por el repartidor cuando llega a la dirección de entrega. Utilizando la vista **validar_pedido**, el repartidor ingresa el código de validación proporcionado por el cliente. El sistema verifica este código y, si coincide con el código generado y almacenado en el modelo **Pedido**, el estado del pedido se actualiza a "entregado".

En esta función se realiza lo siguiente:

- **Validación del Código:** Si el método de la solicitud es **POST**, se verifica el código ingresado con el código de validación almacenado en el pedido. Si coinciden, el pedido se marca como "entregado" y se guarda el cambio.
- **Manejo de Errores:** Si el código no coincide o ocurre algún error al guardar el pedido, se muestra un mensaje de error al repartidor.
- **Redirección:** Tras la validación correcta o un error, el repartidor es redirigido a la lista de pedidos o a la misma página para intentar de nuevo.

Justificación del Uso de la funcionalidad de validación de pedido

- Garantiza que el pedido se entrega al cliente correcto, previniendo fraudes o errores en la entrega.
- El uso de códigos únicos para cada pedido y su validación permite un rastreo y manejo eficiente de los pedidos dentro del sistema.

Funcionalidad de Seguimiento del Estado del Pedido

La funcionalidad de seguimiento del estado del pedido es una característica clave que permite tanto al cliente como al partners visualizar en tiempo real cómo se encuentra el pedido sin necesidad de recargar la página. Esto mejora significativamente la experiencia del usuario, proporcionando actualizaciones constantes.

Implementación funcionalidad de Seguimiento del Estado del Pedido

Para lograr un diseño visualmente atractivo y organizado, utilicé la librería **bs-stepper** en la plantilla **estado_pedido_actualizado.html**. Esta herramienta proporciona un estilo elegante y moderno para mostrar los diferentes estados del pedido en un formato de pasos o "stepper". Cada paso representa un estado del pedido, y el estado actual se resalta visualmente para que los usuarios puedan identificar fácilmente el progreso.

Para actualizar el estado del pedido en tiempo real, se utiliza **AJAX** junto con jQuery. Esta técnica permite que la página web consulte el servidor de forma asíncrona para obtener el estado más reciente del pedido, actualizando la interfaz del usuario sin recargar la página.

```
<script src="https://cdn.jsdelivr.net/npm/bs-stepper/dist/js/bs-stepper.min.js"></script>

<script>
    var x_segundo=20000
    function actualizarEstado () {
        $.ajax({
            url: '{% url 'myapp:estado_pedido_actualizado' pedido_id=pedido.id %}',
            method: 'GET',
            success: function(response) {
                var estadoActual = response.estado;
```

```

        // Actualiza el stepper visualmente
        $('.bs-stepper-circle').removeClass('active');
        $('#circle-'+estadoActual).addClass('active');
    }
}

// Actualiza el estado cada x segundos
setInterval(actualizarEstado, x_segundo);
</script>

```

La vista **EstadoPedidoActualizado** maneja la solicitud AJAX y devuelve el estado actual del pedido en formato JSON, lo que permite que la interfaz de usuario se actualice en tiempo real.

```

class EstadoPedidoActualizado(View):

    def get(self, request, pedido_id, *args, **kwargs):
        pedido = get_object_or_404(Pedido, pk=pedido_id)

        if request.headers.get('x-requested-with') == 'XMLHttpRequest':
            return JsonResponse({'estado': pedido.estado})

        else:
            return render(request, 'generico/estado_pedido_actualizado.html', {'pedido': pedido})

```

Justificación del Uso de AJAX

Con AJAX, el cliente puede ver el progreso del pedido en tiempo real y la implementación es fácil de mantener, permitiendo futuras mejoras sin afectar la funcionalidad básica del sistema. Permite actualizar el estado del pedido sin recargar la página esto proporcionando una experiencia más fluida para el usuario.

Vistas Genéricas para Detalles

Para optimizar la estructura del código y evitar la repetición innecesaria en la creación de vistas y plantillas, he implementado una vista genérica, **DetalleGenericoView**, que permite mostrar los detalles de diferentes tipos de objetos (como restaurantes, platos, usuarios y pedidos) utilizando una única plantilla.

Implementación de la Vista DetalleGenericoView

La clase **DetalleGenericoView** extiende **DetailView**. Esto asegura que solo los usuarios autenticados y con el rol adecuado puedan acceder a los detalles.

```

class DetalleGenericoView(LoginRequiredMixin, RolRequiredMixin, DetailView):
    template_name = 'admin/detalle_generico.html'
    user_type_required = ['partners', 'cliente']

    def get_object(self):
        # Obtener el tipo de objeto y su ID desde la URL
        tipo_objeto = self.kwargs['tipo_objeto']
        id_objeto = self.kwargs['id_objeto']
        # Determinar qué modelo se está solicitando y obtener el
        # objeto correspondiente
        if tipo_objeto == 'restaurante':
            return Restaurante.objects.get(pk=id_objeto)
        elif tipo_objeto == 'plato':
            return Plato.objects.get(pk=id_objeto)

        elif tipo_objeto == 'usuario':
            return User.objects.get(pk=id_objeto)
        elif tipo_objeto == 'pedido':
            return Pedido.objects.get(id=id_objeto)
        # Puedes agregar más tipos de objetos aquí según sea
        # necesario

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['tipo_objeto'] = self.kwargs['tipo_objeto']
        return context

```

De esta manera identifica el tipo de objeto (**restaurante, plato, usuario, pedido o los que necesite**) y obtiene la instancia correspondiente del modelo. Esto permite que la misma vista se utilice para diferentes tipos de objetos.

He configurado una URL genérica en **urls.py** que permite acceder a los detalles de diferentes objetos utilizando una única vista:

```

# DetalleGenerico
path('detalle/<str:tipo_objeto>/<int:id_objeto>', login_required(DetalleGenericoView.as_view())),
name='detalle_generico'),

```

La plantilla **detalle_generico.html** es responsable de mostrar los detalles del objeto. Como se maneja de manera genérica, puede adaptarse a cualquier tipo de objeto al verificar el **tipo_objeto** pasado en el contexto.

```

{%- block content %}
<div class="container">
    <h2>Detalles del {{ tipo_objeto|capfirst }}</h2>

    {%- if tipo_objeto == 'restaurante' %}
        <p>Nombre: {{ object.nombre }}</p>
        <p>Dirección: {{ object.direccion }}</p>
        <!-- Otros campos específicos de Restaurante -->
    {%- elif tipo_objeto == 'plato' %}

```

```

<p>Nombre: {{ object.nombre }}</p>
<p>Precio: {{ object.precio }}</p>
<!-- Otros campos específicos de Plato --&gt;
{%- elif tipo_objeto == 'usuario' %}
    &lt;p&gt;Nombre de usuario: {{ object.username }}&lt;/p&gt;
    &lt;p&gt;Email: {{ object.email }}&lt;/p&gt;
    <!-- Otros campos específicos de Usuario --&gt;
{%- elif tipo_objeto == 'pedido' %}
    &lt;p&gt;Código de Pedido: {{ object.codigo_pedido }}&lt;/p&gt;
    &lt;p&gt;Total: {{ object.total }}&lt;/p&gt;
    <!-- Otros campos específicos de Pedido --&gt;
{%- endif %}
&lt;/div&gt;
{%- endblock %}
</pre>

```

Justificación

La implementación de una vista genérica para mostrar detalles de diferentes tipos de objetos permite reducir significativamente el código repetido, centralizando la lógica en un solo lugar. Esto facilita el mantenimiento, ya que cualquier modificación se realiza en un único punto, mejorando la consistencia y escalabilidad del proyecto. Además, este enfoque ofrece flexibilidad para manejar nuevos tipos de objetos en el futuro, simplemente extendiendo la lógica existente.

Pruebas

Evaluación de los Requerimientos

Para asegurar la calidad y el cumplimiento de los objetivos del sistema, se realizó una evaluación exhaustiva de los requerimientos mediante una tabla de Quality Gateway. Que define un conjunto de criterios de evaluación para aplicar a los casos de uso mediante la **Quality Gateway**, asegurando que cada funcionalidad del sistema. Después de realizar esta evaluación inicial, la compartí con mi tutor, quien revisó los casos de uso conmigo. Durante esta revisión conjunta, validamos las decisiones tomadas y ajustamos aquellos casos de uso que requerían mejoras. Esta colaboración fue clave para optimizar el sistema, asegurando que cumpliera con los objetivos del proyecto y las expectativas del cliente.

Criterios de evaluación de Quality Gateway

Para garantizar que los requerimientos del proyecto estén correctamente definidos, alineados con los objetivos del proyecto, y que su implementación sea viable y eficiente, se aplicaron los siguientes procesos de evaluación:

Evaluaciones:	Descripción de la evaluación	Procedimiento de Evaluación:
Within Scope (Dentro del Alcance)	Se evalúa si los requisitos del proyecto están claramente definidos y comprendidos por todas las partes interesadas. También se verifica si el trabajo realizado está alineado con los objetivos y alcances establecidos inicialmente para el proyecto.	La persona encargada de la evaluación debe revisar los objetivos y alcance del proyecto para determinar si el elemento contribuye directamente a ellos. Si es así, se marca como "si" en la columna de Evaluación; de lo contrario, se marca como "no".
Relevancy (Relevancia):	Evalúa si un requisito específico es fundamental para el éxito del proyecto o es indispensable para su funcionamiento.	La persona encargada de la evaluación debe asignar un valor (alto, medio o bajo) según la importancia del elemento para alcanzar los objetivos del proyecto. Se considera "alto" si el elemento es esencial para el éxito del proyecto, "medio" si contribuye pero no es crítico, y "bajo" si tiene poco impacto en el proyecto.
Fit Criterion (Criterio de Ajuste):	Se evalúa si los requisitos del sistema tienen definidos criterios adecuados.	La persona encargada de la evaluación debe revisar si tiene definidos los criterios adecuados. Se marca como "si" si están definidos y como "no" si no están definidos.
Consistent Terminology (Terminología Consistente):	Se verifica si se ha utilizado una terminología consistente y clara en la documentación de los requisitos.	La persona encargada de la evaluación debe revisar la documentación de los requisitos y casos de uso, para identificar si se utilizan términos de manera consistente. Se marca como "si" si la terminología es coherente en toda la documentación, y como "no" si se encuentran inconsistencias.
Viable Within Constraints (Viable Dentro de las Restricciones):	Se evalúa si la solución es viable y factible dentro de las limitaciones y restricciones del proyecto, como el tiempo, el personal disponible.	La persona encargada de la evaluación debe revisar los requisitos para determinar si son factibles de implementar dentro de las restricciones del proyecto. Se marca como "si" si son viables dentro de las restricciones, y como "no" si no lo son.
Requirement or Solution (Requisito o Solución):	Se verifica si los requisitos del proyecto se han convertido en una solución viable y efectiva que cumple con las necesidades del cliente y los objetivos del proyecto. Se busca garantizar que la solución propuesta sea adecuada para abordar los problemas identificados inicialmente.	La persona encargada de la evaluación debe comparar la solución propuesta con los requisitos del proyecto para determinar si se cumplen. Se marca como "si" si la solución cumple con los requisitos, y como "no" si no los cumple.
Requirement Value (Valor del Requisito):	Se evalúa el valor y la importancia de cada requisito del proyecto en relación con los objetivos y prioridades del cliente. Se busca priorizar los requisitos que aportan el mayor valor al cliente y al proyecto en su conjunto.	La persona encargada de la evaluación debe considerar la contribución de cada requisito al logro de los objetivos del proyecto y prioridades del cliente. Se asigna un valor del 1 al 5, donde 5 representa un valor alto y 1 un valor bajo, según su importancia.

Gold Plating:	Se verifica si se ha realizado "gold plating", si se han añadido funcionalidades o características adicionales que no aportan valor real al cliente.	La persona encargada de la evaluación determinará si incluye funcionalidades o características adicionales que no están alineadas con los requisitos del proyecto. Se marca como "si" si no hay gold plating, y como "no" si se detectan funcionalidades innecesarias.
Requirements Creep:	Se evalúa que no se generen conflictos entre los requisitos actuales.	La persona encargada de la evaluación debe revisar si se podría generar conflictos entre los diferentes requisitos definidos. Se marca como "si" si con si se detecta un conflicto, y como "no" si no se detectan conflictos.

Tabla Inicial de Quality Gateway

A continuación, se presenta la tabla inicial de Quality Gateway del [data], donde se evaluaron todos los casos de uso identificados en el proyecto. En esta tabla se destacan los elementos que requieren ajustes, ya sea por ser *gold plating* o por otros factores que requieren mejoras o eliminaciones.

Caso de Uso (CU)	Within Scope	Relevancy	Fit Criterion	Consistent Terminology	Viable Within Constraints	Requirement or Solution	Requirement Value	Gold Plating	Requirements Creep	Comentario
CU-001: Login	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-002: Registrar Partners en el Sistema	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-003: Registro de Clientes	Sí	Alto	No	Sí	Sí	Requisito	4	No	No	Criterios de ajuste no definidos claramente
CU-004: Agregar Restaurante en el Sistema	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-005: Añadir Platos a un Restaurante	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-006: Crear Usuarios para Responsables de Cocina y Repartidores	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-007: Historial de Pedidos de un Restaurante	Sí	Medio	No	No	Sí	Requisito	3	No	No	Terminología inconsistente
CU-008: Realizar Pedido	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-009: Gestionar el Carrito de Compras	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-010: Finalizar Pedido	No	Alto	No	No	Sí	Requisito	4	No	No	Necesita incluir opción de dirección y tarjeta de crédito
CU-011: Enviar Notificaciones por SMS	No	Bajo	No	Sí	No	Solución	2	Sí	No	Fuera del alcance del proyecto (<i>gold plating</i>)
CU-012: Agregar Puntuación y/o Reseña	Sí	Medio	Sí	Sí	Sí	Requisito	3	No	No	N/A

para un Plato										
CU-013: Mirar Reseñas para un Restaurante	Sí	Medio	Sí	Sí	Sí	Requisito	3	No	No	Revisar necesidad de reseñas separadas por plato y restaurante
CU-014: Gestionar Pedidos en la Cocina	Sí	Alto	Sí	Sí	No	Requisito	5	No	No	Viabilidad cuestionable dentro de las restricciones
CU-015: Mirar Reseñas para un Plato	Sí	Medio	Sí	Sí	Sí	Requisito	3	No	No	Possible duplicidad con CU-013
CU-016: Ofertas y Descuentos Especiales	No	Bajo	No	Sí	No	Solución	2	Sí	No	Prolongaría innecesariamente el proyecto (relevancia baja)
CU-017: Uso de GPS por Repartidores	No	Medio	No	Sí	No	Solución	3	Sí	No	Requerimiento no necesario en el alcance del proyecto (gold plating)

Ajustes Realizados

Después de la evaluación inicial de los casos de uso en la tabla de Quality Gateway, se identificaron y ejecutaron varios ajustes para alinear mejor el sistema con los objetivos del proyecto y eliminar funcionalidades innecesarias o duplicadas. A continuación, se describen los ajustes realizados:

1. Eliminación de "Enviar Notificaciones por SMS" (CU-011)

Motivo del ajuste: Este caso de uso fue considerado *gold plating* ya que añadir notificaciones por SMS no es fundamental para el funcionamiento del sistema y se encuentra fuera del alcance del proyecto.

Acción: Se eliminó CU-011 del proyecto para concentrar los recursos en funcionalidades esenciales.

2. Eliminación de "Uso de GPS por Repartidores" (CU-017)

Motivo del ajuste: Aunque la relevancia de este caso de uso se consideró media, fue identificado como *gold plating* ya que la funcionalidad de GPS no es indispensable para el éxito del proyecto en su fase actual.

Acción: Se eliminó CU-017 para simplificar el desarrollo y enfocarse en los requerimientos más críticos.

3. Separación de Reseñas para Platos y Restaurantes

Motivo del ajuste: Se identificó la necesidad de mantener reseñas separadas para platos y restaurantes para evitar confusión y mejorar la claridad para los usuarios.

Acción: Se decidió mantener los casos de uso CU-012 y CU-015 (reseñas para platos) separados de CU-013 y CU-014 (reseñas para restaurantes), asegurando una categorización clara en la plataforma.

4. Incorporación de Funcionalidad Adicional en "Finalizar Pedido" (CU-010)

Motivo del ajuste: Durante la evaluación se observó que el caso de uso "Finalizar Pedido" no incluía la funcionalidad de ingresar dirección de entrega y método de pago, lo cual es esencial para completar una transacción de pedido.

Acción: Se amplió CU-010 para incluir la opción de ingresar la dirección de entrega y los detalles de pago, garantizando que los pedidos puedan completarse de manera efectiva.

5. Eliminación de "Ofertas y Descuentos Especiales" (CU-016)

Motivo del ajuste: Este caso de uso fue considerado de baja relevancia y se determinó que añadir ofertas y descuentos prolongaría innecesariamente el desarrollo del proyecto.

Acción: Se eliminó CU-016 para evitar sobrecargar el sistema con funcionalidades adicionales que no son críticas en la fase actual del proyecto.

6. **Corrección de Terminología y Criterios de Ajuste**

Motivo del ajuste: Se detectaron inconsistencias en la terminología y la falta de criterios de ajuste claros en algunos casos de uso, especialmente en CU-003 y CU-007.

Acción: Se revisaron y corrigieron las inconsistencias terminológicas, y se definieron claramente los criterios de ajuste para todos los casos de uso, asegurando la coherencia en la documentación.

Tabla Final de Quality Gateway

Después de aplicar los ajustes necesarios, la tabla de Quality Gateway se actualizó para reflejar el estado final de los casos de uso en el sistema. A continuación, se muestra la tabla con las evaluaciones y decisiones finales:

Caso de Uso (CU)	Within Scope	Relevancy	Fit Criterion	Consistent Terminology	Viable Within Constraints	Requirement or Solution	Requirement Value	Gold Plating	Requirements Creep	Comentario
CU-001: Login	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-002: Registrar Partners en el Sistema	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-003: Registro de Clientes	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-004: Agregar Restaurante en el Sistema	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-005: Añadir Platos a un Restaurante	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-006: Crear Usuarios para Responsables de Cocina y Repartidores	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-007: Historial de Pedidos de un Restaurante	Sí	Medio	Sí	Sí	Sí	Requisito	4	No	No	N/A
CU-008: Realizar Pedido	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-009: Gestionar el Carrito de Compras	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-010: Finalizar Pedido	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-012: Agregar Puntuación y/o Reseña para un Plato	Sí	Medio	Sí	Sí	Sí	Requisito	4	No	No	N/A
CU-013: Mirar Reseñas para un Restaurante	Sí	Medio	Sí	Sí	Sí	Requisito	4	No	No	N/A
CU-014: Gestionar Pedidos en la Cocina	Sí	Alto	Sí	Sí	Sí	Requisito	5	No	No	N/A
CU-015: Mirar Reseñas para un Plato	Sí	Medio	Sí	Sí	Sí	Requisito	4	No	No	N/A

Esta tabla final refleja un conjunto optimizado de casos de uso, alineado con los objetivos del proyecto y dentro de sus restricciones.

Evaluación heurística de usabilidad

Para garantizar que la interfaz de usuario de la aplicación sea intuitiva y eficiente, se llevará a cabo una **Evaluación Heurística de Usabilidad** basada en los principios establecidos por expertos en el campo del diseño de interfaces de usuario, Jakob Nielsen y Bruce Tognazzini. Esta metodología está diseñada para identificar y corregir problemas de usabilidad mediante la aplicación de una serie de principios heurísticos que ayudan a evaluar la calidad y la efectividad de la interfaz.

Criterios de evaluación heurística de usabilidad

Se utilizaron los siguientes criterios para la evaluación heurística:

- **Visibilidad del Estado del Sistema:** Asegurarse de que los usuarios siempre estén informados sobre lo que está sucediendo en la aplicación.
- **Coincidencia entre el Sistema y el Mundo Real:** Garantizar que la información y los comandos sean comprensibles y coherentes con las expectativas y conocimientos previos de los usuarios.
- **Control y Libertad del Usuario:** Ofrecer opciones claras para deshacer acciones y regresar a estados anteriores sin dificultad.
- **Consistencia y Estándares:** Mantener una interfaz coherente en términos de terminología, iconos y comportamiento para facilitar la navegación y el aprendizaje.
- **Prevención de Errores:** Minimizar las posibilidades de error y proporcionar mensajes de error claros y útiles.
- **Reconocimiento antes que Recuerdo:** Hacer que la información y las acciones sean fácilmente accesibles sin requerir que los usuarios recuerden detalles previos.
- **Flexibilidad y Eficiencia de Uso:** Permitir a los usuarios experimentar con diferentes métodos para realizar tareas y proporcionar accesos directos cuando sea necesario.
- **Diseño Estético y Minimalista:** Utilizar un diseño claro y libre de redundancia para facilitar la comprensión y la navegación.
- **Ayuda a los Usuarios a Reconocer, Diagnosticar y Recuperarse de Errores:** Proporcionar mensajes de error claros y ayuda contextual para la recuperación de errores.
- **Color y Legibilidad:** Asegurar que el texto sea legible y el contraste sea adecuado para facilitar la lectura.
- **Autonomía y Valores por Defecto:** Mantener a los usuarios informados y proporcionar valores por defecto cuando sea apropiado.
- **Reducción de la Latencia:** Garantizar que las tareas pesadas se ejecuten de manera eficiente y transparente para el usuario.

Ejecución de la evaluación heurística

Esta evaluación será realizada en colaboración con mi tutor del Trabajo de Fin de Grado (TFG). Durante esta evaluación, se examinarán las diferentes áreas de la interfaz utilizando los principios heurísticos mencionados. Se analizará la aplicación en cada uno de los principios heurísticos para identificar posibles problemas de usabilidad. Se obtendrán observaciones y comentarios detallados sobre las áreas que requieren mejorarse. Finalmente utilizar los resultados de la evaluación para realizar ajustes en la interfaz, asegurando que se resuelvan los problemas identificados y se optimice la experiencia del usuario.

Resultados evaluación heurística

Evaluación heurística de usabilidad

Aplicación a evaluar: Sistema DeliverEat

Nombre / Name: David Sarrat González

Perfil/Profile: Director del TFG

Fecha/Date: 10/08/2023

Heurística	Pregunta	Respuesta (Sí/No)	Comentario
Visibilidad del Estado del Sistema	¿La aplicación incluye de forma visible el título de la página, de la sección o del sitio?	Sí	Títulos claramente visibles en todas las páginas.
	¿El usuario sabe en todo momento dónde está?	Sí	N/A
	¿El usuario sabe en todo momento qué está haciendo el sistema o aplicación?	Sí	N/A
	¿Los enlaces están claramente definidos?	Sí	N/A
	¿Todas las acciones pueden verse directamente? (Sin requerir acciones adicionales)	Sí	Las opciones y acciones están visibles sin necesidad de navegación adicional.
Coincidencia entre el Sistema y el Mundo Real	¿La información aparece en un orden lógico para el usuario?	Sí	La disposición de la información sigue un flujo intuitivo.
	¿El diseño de los iconos se corresponde con objetos cotidianos?	Sí	Los iconos que se muestran representan claramente los objetos y acciones.
	¿Cada ícono realiza la acción que el usuario espera?	Sí	N/A
	¿Se utilizan frases y conceptos familiares para el usuario?	Sí	El lenguaje es accesible y comprensible para todos los usuarios.
Control y Libertad del Usuario	¿Existe un vínculo para volver al estado inicial o a la página de inicio?	Sí	Cuenta con un breadcrumb en todas las páginas.

	¿Es fácil volver a un estado anterior de la aplicación?	Sí	La navegación hacia atrás es sencilla debido al uso de breadcrumb
Consistencia y Estándares	¿Las etiquetas de los vínculos tienen los mismos nombres que sus destinos?	No	En algunos casos, los nombres de los vínculos no coinciden con sus destinos, se requiere revisión.
	¿Las mismas acciones siempre conducen a los mismos resultados?	Sí	La consistencia en las acciones es mantenida a lo largo del sistema.
	¿Un mismo ícono tiene el mismo significado en todo el sistema?	Sí	Los iconos tienen los mismos significados en toda la interfaz.
	¿La información se muestra de forma consistente en todo el sistema?	Sí	N/A
	¿Los colores de los enlaces son estándares o, si no, adecuados para su uso?	Sí	Los colores de los enlaces cumplen y son apropiados.
	¿Los elementos de navegación siguen los estándares? (botones, checkboxes, etc.)	Sí	N/A
Prevención de Errores	¿Aparece un mensaje de confirmación antes de realizar las acciones?	No	No se proporciona confirmación antes de realizar acciones críticas.
	¿Queda claro qué hay que introducir en cada campo de un formulario?	Sí	Las instrucciones de los formularios son claras.
Reconocimiento antes que Recuerdo	¿Es sencillo de utilizar por primera vez?	Sí	La aplicación es intuitiva para nuevos usuarios.
	¿Es fácil localizar información que ya ha sido buscada con anterioridad?	Sí	N/A
	¿En todo momento puedes utilizar el sistema sin necesidad de recordar pantallas anteriores?	Sí	N/A
	¿Todo el contenido necesario para la navegación o para las diferentes tareas está en la "pantalla actual"?	Sí	N/A
	¿La información está organizada según la lógica familiar de los usuarios "tipo"?	Sí	La organización de la información es lógica y familiar tanto para los clientes como para los partners, responsables de cocina y repartidores.
Flexibilidad y Eficiencia de Uso	¿Existencia de atajos del teclado para las acciones frecuentes?	No evaluado	No se realizó una evaluación específica sobre este aspecto.
	Si existen, ¿queda claro cómo usarlas?	No evaluado	No se realizó una evaluación específica sobre este aspecto.
	¿Es posible realizar de manera sencilla una acción realizada anteriormente?	Sí	N/A
	¿El diseño se adapta al cambiar la resolución de la pantalla?	Sí	El diseño es responsive, aunque se detectaron algún problema en el pie de página en listas de pedidos y con la librería bs-stepper
	¿Es visible el uso de aceleradores para el usuario habitual?	No evaluado	No se realizó una evaluación específica sobre este aspecto.

	¿Se mantiene siempre ocupado al usuario? (sin tiempos de espera innecesarios)	Sí	La aplicación no presenta tiempos de espera innecesarios.
Diseño Estético y Minimalista	¿Se ha usado un diseño sin redundancia de información?	Sí	La información presentada es concisa y relevante.
	¿La información es corta, concisa y precisa?	Sí	N/A
	¿Cada elemento de información se diferencia del resto y no se confunde?	Sí	Los elementos de información están bien definidos. Algunas mejoras menores podrían ser beneficiosas.
	¿El texto está bien organizado, con frases cortas y de interpretación rápida?	Sí	La mayoría de las frases están bien estructuradas. Algunas mejoras menores podrían ser beneficiosas.
Ayuda a los Usuarios a Reconocer, Diagnosticar y Recuperarse de Errores	¿Se muestra un mensaje antes de tomar acciones irreversibles?	No	No se proporcionan mensajes de confirmación para acciones irreversibles.
	¿Los errores cometidos se muestran en tiempo real?	Sí	Los errores se presentan de manera inmediata.
	¿El mensaje de error que aparece es fácilmente interpretable?	Sí	Los mensajes de error son claros en la mayoría de los casos.
	¿Se usa, además, algún código para referenciar el error?	No	No se utilizan códigos para identificar errores específicos.
Color y Legibilidad	¿Las fuentes del texto tienen un tamaño adecuado?	Sí	El tamaño de la fuente es apropiado.
	¿Las fuentes del texto utilizan colores con suficiente contraste con el fondo?	Sí	El contraste es adecuado para la lectura en todas las pantallas.
	¿Las imágenes o patrones del fondo no impiden la lectura del contenido?	No	No se identificaron problemas relacionados con imágenes o patrones de fondo.
	¿Se tiene en cuenta a los usuarios con visión reducida?	Sí	El tamaño del texto y el contraste son adecuados para usuarios con visión reducida.
Autonomía	¿Se mantiene en todo momento informado al usuario del estado del sistema?	Sí	N/A
	Además, ¿el estado del sistema es visible y actualizado?	Sí	N/A
Valores por Defecto	¿El sistema proporciona valores por defecto al mostrarse un formulario?	Sí	Los valores por defecto se proporcionan cuando es necesario.
Reducción de la Latencia	¿La ejecución de tareas pesadas es transparente al usuario?	No	La aplicación no informa al usuario sobre la ejecución de tareas intensivas.

Conclusiones después de realizar la evaluación heurística

Una vez realicé la evaluación heurística junto con mi tutor de TFG, se revelaron tanto fortalezas como áreas para mejorar en la interfaz de usuario del sistema de entrega de comida. Se identificaron varios aspectos positivos en términos de visibilidad, consistencia y diseño estético, mientras que se detectaron oportunidades para mejorar la confirmación de acciones críticas, la referencia de errores y la transparencia durante la ejecución de tareas pesadas. Estos hallazgos servirán como base para futuros ajustes en la interfaz y mejoras de la usabilidad del sistema.

Pruebas BDD

En el desarrollo del sistema de reparto de comida, se implementó el enfoque de **Desarrollo Guiado por Comportamiento (BDD)** para asegurar que las funcionalidades del sistema se comporten conforme a lo esperado. Para ello, se utilizó la herramienta **Django Behave**, que integra el marco de trabajo Behave con Django, facilitando la ejecución de pruebas BDD en el entorno del proyecto. En este proyecto, se realizaron pruebas básicas utilizando este enfoque.

Herramienta Utilizada: Django Behave

Django Behave es una herramienta que integra el framework de pruebas Behave con Django, permitiendo escribir pruebas en forma de características (features) que contienen escenarios descriptivos del comportamiento esperado del sistema. Cada escenario se define utilizando el formato "Given" (Dado), "When" (Cuando), y "Then" (Entonces), que describe el estado inicial, la acción realizada y el resultado esperado, respectivamente.

Los **steps** (pasos) son los componentes de código que implementan las acciones descritas en los escenarios, permitiendo que las pruebas se ejecuten en el entorno de Django. Estos steps están escritos en Python y permiten la interacción con los modelos, vistas, y otros componentes del sistema.

Estructura de las Pruebas BDD

Las pruebas BDD se organizan en los archivos **.feature**, donde cada archivo contiene una o más características de la aplicación, y cada característica contiene múltiples escenarios. Estos archivos se encuentran en la carpeta **features** dentro del directorio de pruebas del proyecto Django. A continuación se mostrará uno de los archivos de features que se encuentra en el proyecto como ejemplo

```
Feature: Login de Usuario
  Quiero iniciar sesión en mi apartado correspondiente a mi rol
  Scenario: Login exitoso de usuario registrado
    Given estoy en la página de inicio de sesión
    Given existen las siguientes cuentas registradas en el sistema
    | rol   | username| password|
    | cliente | tibi_cliente_test| Tiberiu.1234 |
    | partners | tibi_partners_test| Tiberiu.1234 |
    | cocina | tibi_cocina_test| Tiberiu.1234 |
    | repartidor | tibi_repartidor_test| Tiberiu.1234 |
    When ingreso mis credenciales
    | username | password |
    | tibi_cliente_test| Tiberiu.1234 |
```

```
| tibi_partners_test| Tiberiu.1234 |
| tibi_cocina_test| Tiberiu.1234 |
| tibi_repartidor_test| Tiberiu.1234 |
```

And presiono el botón de iniciar sesión

Then debería ser redirigido a la página correspondiente a mi rol

rol	url
cliente	restaurantes_list_cliente
partners	restaurantes_list
cocina	pedidos_actualizados
repartidor	pedidos_repartidor

Scenario: el nombre de usuario no registrado no puede iniciar sesión

When inicio sesión como usuario "inexistente" con contraseña "contraseña"

And presiono el botón de iniciar sesión

Then debería ver la página de login con el siguiente mensaje "Por favor, introduzca un nombre de usuario y clave correctos. Observe que ambos campos pueden ser sensibles a mayúsculas."

Scenario: error tipográfico incorrecto, el usuario registrado no puede iniciar sesión

Given existe un usuario "nombredelusuario" con contraseña "contraseña"

When inicio sesión como usuario "nombredelusuario" con contraseña "contraseña incorrecta"

And presiono el botón de iniciar sesión

Then debería ver la página de login con el siguiente mensaje "Por favor, introduzca un nombre de usuario y clave correctos. Observe que ambos campos pueden ser sensibles a mayúsculas."

Las Pruebas BDD se organizan en archivos .py de implementación de los pasos (**steps**) escritos en Python, que definen cómo se ejecutan los escenarios descritos en los archivos .feature. A continuación se mostrará uno de los archivos de features que se encuentra en el proyecto como ejemplo.

```
import os

# Importamos los modelos que usaremos en las pruebas

from myapp.models import User, Archivo, Partners, Negocio

# Importamos los módulos necesarios para hacer pruebas BDD con behave y Django

from behave import given, when, then

from django.urls import reverse

from django.test import Client

@given(u'estoy en la página de registro de clientes')

def visit_registration_page(context):

    context.client = Client() # Simulamos un cliente web para hacer peticiones

    context.response = context.client.get(reverse('myapp:hacer_registro_cliente')) # Petición GET a la URL de registro de clientes

    assert context.response.status_code == 200 # Verificamos que la respuesta sea exitosa (200 OK)

@when(u'lleno el formulario con los siguientes datos')

def fill_registration_form(context):

    for row in context.table: # Recorremos las filas de datos proporcionadas en la tabla del escenario

        form = {

            'first_name': row['nombre'],

            'last_name': row['apellidos'],

            'email': row['email'],

            'telefono': row['telefono'],

            'prefix_tel': row['prefijo'],

            'username': row['username'],
```

```

'password': row['password'],
'pais': row['pais'],
'ciudad': row['ciudad'],
'codigo_postal': row['codigo_postal'],
'direccion': row['direccion'],
'numero': row['numero'],

}

context.data = form # Guardamos el formulario en el contexto para posibles validaciones

context.response = context.client.post(reverse('myapp:hacer_registro_cliente'), form, follow=True)
# Enviamos el formulario con una petición POST

# Verificaciones adicionales

print("Datos del formulario enviados:", form)

print("Código de estado de la respuesta:", context.response.status_code)

print("Contenido de la respuesta:", context.response.content.decode('utf-8')) # Imprimimos el
contenido de la respuesta para ver resultados

@when(u'envío el formulario')

def submit_registration_form(context):

    pass # No es necesario implementar nada aquí, ya se hace en el paso anterior

@then(u'debería ver la página de inicio de sesión')

def see_login_page(context):

    assert context.response.status_code == 200 # Verificamos que la respuesta sea exitosa

    assert 'Inicia sesión dentro del sistema de DeliveEreat' in context.response.content.decode('utf-8')
# Verificamos que se muestre el mensaje esperado

```

```

@then(u'debería ver la página de registro con el siguiente mensaje "{mesg_error}"')
def see_registration_page_with_errors(context, mesg_error):
    content = context.response.content.decode('utf-8')

    assert mesg_error in content, "No se encontraron mensajes de error en el contenido de la respuesta."
# Validamos que el mensaje de error aparezca


@then(u'debería haber un username registrado con el username "{username}" con el siguiente rol "{rol}"')
def check_user_registered(context, username, rol):
    assert User.objects.filter(username=username).exists() # Verificamos que el usuario existe en la base
de datos

    user = User.objects.get(username=username)

    assert user.user_type == rol, f"El rol del usuario es {user.user_type}, pero se esperaba {rol}" #
Comprobamos que el rol sea el esperado


@then(u'debería de no existir el usuario con el username "{username}"')
def check_user_not_registered(context, username):
    assert not User.objects.filter(username=username).exists() # Verificamos que no exista un usuario con
ese username


@then(u'debería existir solo "{namber_registered}" usuarios')
def check_namber_user_registered(context, namber_registered):
    assert int(namber_registered) == User.objects.all().count() # Comprobamos que el número total de
usuarios coincida


@given(u'estoy en la página de registro de partners')
def visit_registration_page(context):

    context.client = Client() # Simulamos un cliente

```

```

    context.response = context.client.get(reverse('myapp:hacer_registro')) # Petición GET a la URL de
registro de partners

    assert context.response.status_code == 200 # Verificamos que la página se cargue correctamente (200
OK)

@when(u'llenó el formulario de partners con los siguientes datos')

def fill_registration_form(context):
    for row in context.table:

        # Definimos la ruta a los archivos que queremos adjuntar
        archivos_path = os.path.join(os.path.dirname(__file__), 'test_files')
        archivos = [
            ('archivos', open(os.path.join(archivos_path, 'file1.pdf'), 'rb')),
            ('archivos', open(os.path.join(archivos_path, 'file2.pdf'), 'rb'))
        ]

        # Datos del formulario
        form = {
            'username': row['username'],
            'email': row['email'],
            'password': row['password'],
            'prefix_tel': row['prefix_tel'],
            'telefono': row['telefono'],
            'first_name': row['first_name'],
            'last_name': row['last_name'],
            'nombre_negocio': row['nombre_negocio'],
            'archivos': archivos, # Adjuntamos los archivos
        }

```

```

context.data = form # Guardamos el formulario en el contexto

context.response = context.client.post(reverse('myapp:hacer_registro'), form, format='multipart',
follow=True) # Enviamos el formulario con los archivos adjuntos

@when(u'llenó el formulario de partners con los siguientes datos pero sin archivos')

def not_fill_registration_form(context):

    for row in context.table:

        # Formulario sin archivos

        form = {

            'username': row['username'],

            'email': row['email'],

            'password': row['password'],

            'prefix_tel': row['prefix_tel'],

            'telefono': row['telefono'],

            'first_name': row['first_name'],

            'last_name': row['last_name'],

            'nombre_negocio': row['nombre_negocio'],

        }

        context.data = form # Guardamos los datos en el contexto

        context.response = context.client.post(reverse('myapp:hacer_registro'), form, files=None,
follow=True) # Enviamos el formulario sin archivos

@then('debería ver que los archivos fueron subidos correctamente para el usuario "{username}"')

def check_files_uploaded(context, username):

    try:

        user = User.objects.get(username=username) # Buscamos el usuario por su username

        partner = Partners.objects.get(user=user) # Obtenemos el partner asociado al usuario

        negocio = partner.negocio # Obtenemos el negocio asociado al partner

```

```

# Buscamos los archivos asociados al negocio

archivos_negocio = Archivo.objects.filter(negocio=negocio)

assert archivos_negocio.exists(), f"No se encontraron archivos asociados al negocio de {partner}"


# Verificamos que los archivos específicos estén presentes

assert archivos_negocio.filter(archivo__icontains='file1.pdf').exists(), "No se encontró file1.pdf"

assert archivos_negocio.filter(archivo__icontains='file2.pdf').exists(), "No se encontró file2.pdf"


except User.DoesNotExist:

    raise AssertionError(f"El usuario con username '{username}' no existe en la base de datos")

except Partners.DoesNotExist:

    raise AssertionError(f"No se encontró un partner asociado al usuario '{username}'")

except Negocio.DoesNotExist:

    raise AssertionError(f"No se encontró un negocio asociado al partner de '{username}'")

except Archivo.DoesNotExist:

    raise AssertionError(f"No se encontraron archivos asociados al negocio de {partner}")

@given(u'estoy en la página de inicio de sesión')

def visit_login_page(context):

    context.client = Client() # Simulamos un cliente web

    context.response = context.client.get(reverse('myapp:login')) # Hacemos una petición GET a la página de login

@given(u'existen las siguientes cuentas registradas en el sistema')

def create_test_users(context):

    for row in context.table: # Recorremos las filas de la tabla con usuarios de prueba

        user = User.objects.create_user(username=row['username'], password=row['password']) # Creamos el usuario

```

```

user.user_type = row['rol'] # Asignamos el rol al usuario

user.save() # Guardamos el usuario en la base de datos

@when(u'ingreso mis credenciales')

def fill_login_form(context):
    for row in context.table:

        form = {

            'username': row['username'],
            'password': row['password'],
        }

        context.response = context.client.post(reverse('myapp:login'), form, follow=True) # Enviamos el formulario de login

@then(u'debería ser redirigido a la página correspondiente a mi rol')

def verify_redirection(context):

    assert context.client.session.get('_auth_user_id') is not None # Verificamos que el usuario esté autenticado

    user = User.objects.get(id=context.client.session['_auth_user_id']) # Obtenemos el usuario autenticado

    for row in context.table:

        if user.user_type == row['rol']: # Verificamos que la redirección corresponda al rol del usuario

            if row['rol'] == 'repartidor': # Caso especial para repartidores

                expected_url = reverse('myapp:' + str(row['url']), kwargs={'tipo_objeto': 'recoger'})

            else:

                expected_url = reverse('myapp:' + str(row['url']))

            print(f"Expected URL: {expected_url}")

            print(f"Redirect chain: {context.response.redirect_chain}") # Mostramos la cadena de redirecciones

            assert expected_url in context.response.redirect_chain[-1][0] # Verificamos que la URL esperada esté en la cadena de redirecciones

```

```

@given(u'existe un usuario "{username}" con contraseña "{password}"')
def create_user(context, username, password):
    User.objects.create_user(username=username, password=password) # Creamos un usuario en la base de datos

@when(u'inicio sesión como usuario "{username}" con contraseña "{password}"')
def login_user(context, username, password):
    context.client = Client() # Simulamos un cliente web
    form = {
        'username': username,
        'password': password,
    }
    context.response = context.client.post(reverse('myapp:login'), form) # Enviamos el formulario de login

@then(u'debería ver la página de login con el siguiente mensaje "{message}"')
def verify_login_error_message(context, message):
    assert context.response.status_code == 200 # Verificamos que la página se cargue correctamente
    assert message in context.response.content.decode() # Comprobamos que el mensaje de error esté presente

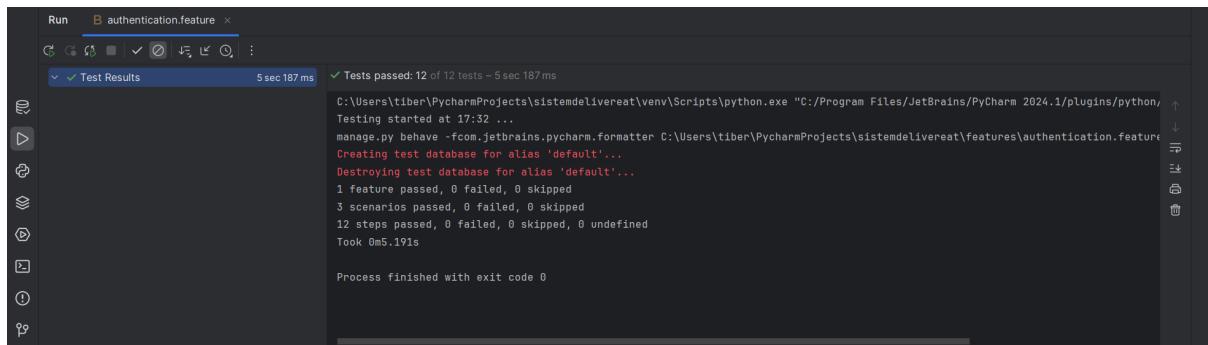
```

Ejecución de las Pruebas

Para ejecutar las pruebas, se utilizó el comando de Django Behave:

```
python manage.py behave authentication.feature
```

Este comando ejecuta los escenarios definidos y genera un resultado, indicando cuáles pruebas pasaron, fallaron o quedaron pendientes.



```
Run authentication.feature x
Run Test Results 5 sec 187 ms
Tests passed: 12 of 12 tests – 5 sec 187 ms
C:\Users\tiber\PycharmProjects\sistemdelivereat\venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm 2024.1/plugins/python/
Testing started at 17:32 ...
manage.py behave -fcom.jetbrains.pycharm.formatter C:\Users\tiber\PycharmProjects\sistemdelivereat\features\authentication.feature
Creating test database for alias 'default'...
Destroying test database for alias 'default'...
1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
12 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m5.191s

Process finished with exit code 0
```

Beneficios de BDD

Los escenarios de prueba en BDD son fáciles de mantener gracias a su formato simple y legible, lo que facilita su actualización conforme el sistema evoluciona. Además, BDD promueve una colaboración más estrecha entre los desarrolladores y los stakeholders o usuarios del sistema.

Repositorio del Proyecto

En el proceso de desarrollo de este proyecto, se ha utilizado la herramienta de gestión de código Git para mantener el control de versiones del proyecto y facilitar la revisión del código fuente. Para este propósito, se creó un repositorio en GitHub, donde se encuentran todas las versiones y actualizaciones del proyecto.

El repositorio está disponible en el siguiente enlace:

<https://github.com/TiberiuPaiu/sistemdelivereat>

Resultados del Sistema

Este apartado presenta una serie de capturas de pantalla del sistema para ilustrar su interfaz y funcionamiento. A través de estas imágenes, se puede visualizar cómo los usuarios interactúan con el sistema y cómo se presentan las diferentes funcionalidades.

Pantalla de Inicio de Sesión



La pantalla de inicio de sesión permite a los usuarios acceder al sistema utilizando sus credenciales.

Pantalla de Registro de Partners

The registration form for partners is a "Linear Stepper" with four steps:

- Step 1: Información personal**
Fields: Nombre (Tibi), Apellidos (Pau). Buttons: Iniciar sesión, Siguiente.
- Step 2: Información de contacto**
Fields: Dirección de correo electrónico (tibipaiu719@gmail.com), Teléfono (+34 61200022). Buttons: Previo, Siguiente.

El formulario de registro para los socios comerciales (partners) está organizado en un formato de "Linear Stepper", que guía al usuario a través de las siguientes secciones Información Personal, Información de Contacto, Información de la Cuenta: Configuración de la cuenta de usuario e Información del Negocio.

Pantalla de Registro de Cliente

The registration form for clients is a "Linear Stepper" with four steps:

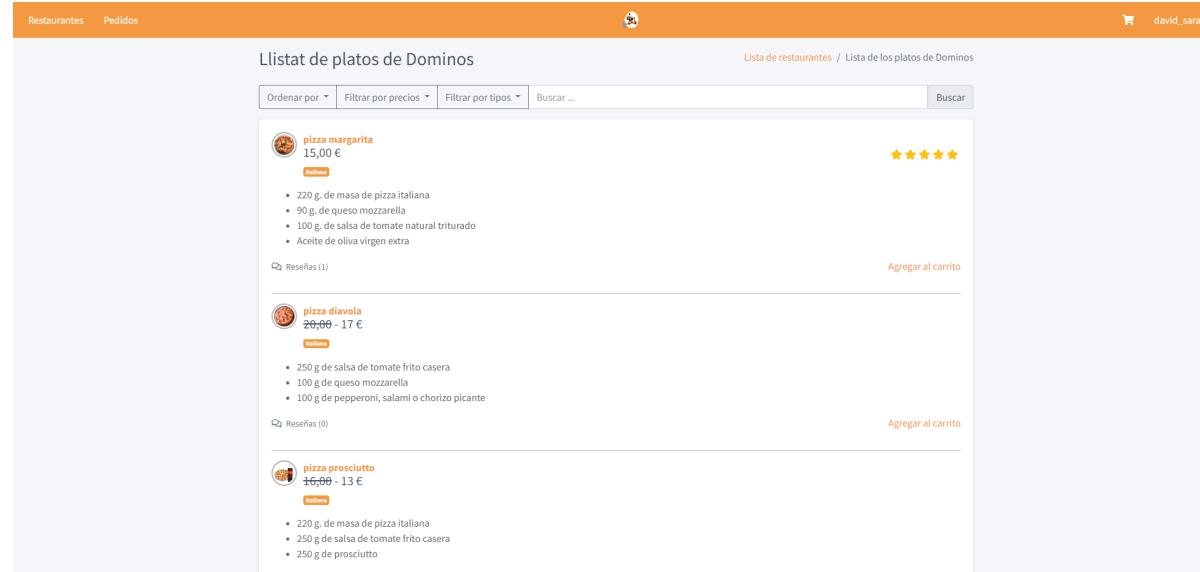
- Step 1: Información personal**
Fields: Nombre de usuario (test_tiberiu), Contraseña (.....). Buttons: Previo, Siguiente.
- Step 2: Información de contacto**
Fields: Nombre del Negocio (Test Business N). Buttons: Elegir archivos (requisitos_2.pdf), Register.

Similar al registro de partners, el formulario de registro para clientes también sigue un formato de "Linear Stepper" e incluye la información Personal, la información de Contacto, información de la Cuenta y la dirección principal para la entrega de pedidos.

Pantalla de listado de Restaurantes

Una vez registrado, al cliente se le muestran los restaurantes disponibles en la ciudad correspondiente a su dirección y ordenado por el promedio de la puntuación más alta. El cliente puede seleccionar un restaurante de su preferencia.

Pantalla de listado de Platos

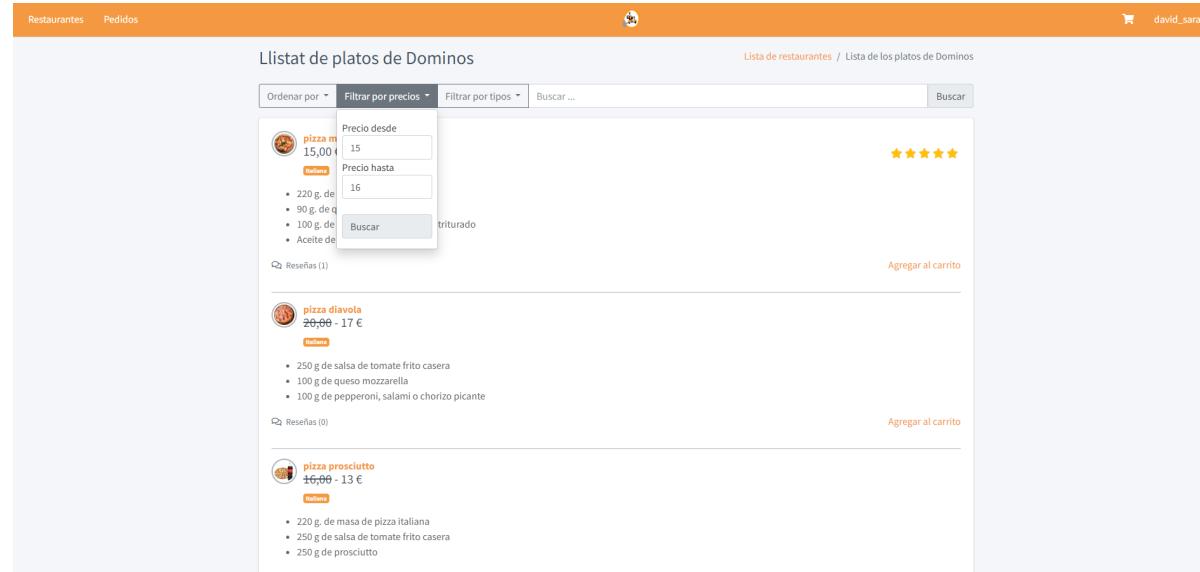


The screenshot shows a list of three dishes from Dominos:

- pizza margarita** 15,00 €
Italiana
• 220 g. de masa de pizza italiana
• 90 g. de queso mozzarella
• 100 g. de salsa de tomate natural triturado
• Aceite de oliva virgen extra
- pizza diavola** 20,90 - 17 €
Italiana
• 250 g de salsa de tomate frito casera
• 100 g de queso mozzarella
• 100 g de pepperoni, salami o chorizo picante
- pizza prosciutto** 16,90 - 13 €
Italiana
• 220 g. de masa de pizza italiana
• 250 g de salsa de tomate frito casera
• 250 g de prosciutto

Los clientes pueden elegir cualquier plato del menú del restaurante seleccionado. Los platos se añaden automáticamente al carrito de compras. El cliente puede añadir platos de diferentes restaurantes al carrito. El número total de platos seleccionados se muestra en el menú superior lateral.

Funcionalidades de Búsqueda y Filtros



El sistema permite a los clientes ordenar los platos por precio, puntuación, y tipo de comida. También ofrece la opción de buscar platos por nombre utilizando la barra de búsqueda.

Pantallas de la gestión del carrito de compras

Nombre	Restaurante	Precio	Cantidad	Opción
pizza margarita	Dominos	15,00 €	<input type="button" value="-"/> <input type="button" value="2"/> <input type="button" value="+"/>	<input type="button" value="Eliminar"/>
ANGUS GRILL	mcdonalds	15,00 €	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	<input type="button" value="Eliminar"/>

Restaurante : Dominos

Subtotal: 30,00 €

Impuesto Plataforma (10%): 3,00 €

Impuesto Traferencias (3% + 0,50€): 0,90 €

Total: 33,90 €

Restaurante : mcdonalds

Subtotal: 15,00 €

Impuesto Plataforma (10%): 1,50 €

Impuesto Traferencias (3% + 0,50€): 0,45 €

Total: 16,95 €

Total : 50,85 €

El cliente puede acceder al carrito en cualquier momento para revisar los platos seleccionados, añadir más cantidad o eliminar elementos. Además, se muestra el restaurante al que pertenece cada plato y el coste total desglosado, incluyendo las comisiones de la plataforma por cada transacción.

Nombre	Restaurante	Precio	Cantidad	Opción
pizza margarita	Dominos	15,00 €	<input type="button" value="-"/> <input type="button" value="2"/> <input type="button" value="+"/>	<input type="button" value="Eliminar"/>
pizza diavola	Dominos	20,00 - 17,00 € Descuento : 15%	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	<input type="button" value="Eliminar"/>
ANGUS GRILL	mcdonalds	15,00 €	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	<input type="button" value="Eliminar"/>

Restaurante : Dominos

Subtotal: 47,00 €

Impuesto Plataforma (10%): 4,70 €

Impuesto Traferencias (3% + 0,50€): 1,41 €

Total: 53,11 €

Restaurante : mcdonalds

Subtotal: 15,00 €

Impuesto Plataforma (10%): 1,50 €

Impuesto Traferencias (3% + 0,50€): 0,45 €

Total: 16,95 €

Total : 70,06 €

Si un plato añadido en el carrito tiene un descuento, el sistema lo aplica automáticamente al total y lo muestra al cliente.

Pantallas del proceso de pedido

Realizar el pedido

Nombre: David

Apellido: Sarat

Teléfono: +34 61200022

País: España

Ciudad: Valladolid

Código postal: 47003

Calle: Santo Domingo de Guzmán

Número Calle: 8

Tarjeta de crédito o débito: VISA 4242 4242 4242 4242 12 / 26 111 47003

Enviar Pago

Sobre Nosotros: Somos una plataforma dedicada a ofrecer la mejor experiencia en entrega de comida. Nuestro objetivo es conectar a los usuarios con sus restaurantes cercanos de manera rápida y eficiente.

Enlaces Útiles: Términos y Condiciones | Política de Privacidad

Contacto: Calle 123, Lleida, España | +34 123 456 789 | info@deliverreat.com

Síguenos: Facebook | Twitter | Instagram | LinkedIn

Una vez que el cliente está listo para realizar el pedido, presiona el botón "Realizar Pedido". El sistema muestra un formulario donde se confirma la dirección de entrega y se solicita la información de la tarjeta de crédito.

Modo de prueba

Estás usando datos de prueba. Para aceptar pagos, completa el perfil de tu empresa.

70,06 € EUR Exitoso ✓

Última actualización: 27 ago. 2024 2:45 | Cliente: Ninguno | Método de pago: VISA **** 4242 | Evaluación de riesgos: Normal

Cronograma

Pago efectuado con éxito: 27 ago. 2024 2:45

Pago iniciado: 27 ago. 2024 2:45

Desglose del pago

Importe del pago	70,06 € EUR
Comisiones de procesamiento de Stripe: Más información	- 2,53 € EUR
Importe neto	67,53 € EUR

Datos del pago

Descripción del extracto: Stripe
Descripción: Pago realizado por los siguientes pedidos: 47af4181f47f, ca6f933c1d58 | [Modificar](#)

Una vez confirmado el pago, el sistema ingresa el dinero correspondiente al pedido.

Restaurants Pedidos

Llistat de pedidos realizados

success
El proceso de pedido se realizó exitosamente.

Código del pedido	Estado	Platos	Total	Restaurante	Fecha	Opciones
ca6f933c1d58	Estado Actual	• ANGUS GRILL	15,00 €	mcdonalds	27 de agosto de 2024 a las 02:45	<button>Cancelar Pedido</button>
47af4181f47f	Estado Actual	• pizza margarita • pizza diavola	47,00 €	Dominos	27 de agosto de 2024 a las 02:45	<button>Cancelar Pedido</button>

Lista de restaurantes / Llistat de pedidos realizados



Sobre Nosotros

Somos una plataforma dedicada a ofrecer la mejor experiencia en entrega de comida. Nuestro objetivo es conectar a los usuarios con sus restaurantes cercanos de manera rápida y eficiente.

Enlaces Útiles

[Términos y Condiciones](#)
[Política de Privacidad](#)

Contacto

Calle 123, Lleida,
España
+34 123 456 789
info@delivereat.com

Síguenos

© 2024.
Desarrollado por [Tiberiu Pau](#).

Tras completar el formulario, el sistema procesa el pago y, una vez confirmado, el pedido aparece en la lista de pedidos del cliente en estado de "Procesamiento".

Pantallas de gestión del pedido para el cliente

Restaurants Pedidos

Llistat de pedidos realizados

success
El pedido se ha cancelado exitosamente.

Código del pedido	Estado	Platos	Total	Restaurante	Fecha	Opciones
ca6f933c1d58	Estado Actual	• ANGUS GRILL	15,00 €	mcdonalds	27 de agosto de 2024 a las 02:45	
47af4181f47f	Estado Actual	• pizza margarita • pizza diavola	47,00 €	Dominos	27 de agosto de 2024 a las 02:45	<button>Cancelar Pedido</button>

Lista de restaurantes / Llistat de pedidos realizados

El cliente tiene la opción de cancelar el pedido desde la lista de pedidos.



El cliente puede ver el estado actual de su pedido.

Detalles del pedido

DeliverEat

mcdonalds
Paseo del Hospital Militar 11
España Valladolid, 47011
Teléfono: +34 61200022
Correo electrónico:tiberiuip97@gmail.com

David Sarat
Santo Domingo de Guzmán 8
España Valladolid, 47011
Teléfono: +34 61200022
Correo electrónico:tibigame6521@gmail.com

Pedido #ca6f933c1d58
Código de confirmación : 3ec50760e1a0
Fecha de pago:
Cuenta:

Unidades	Product	Imagen	Precio unidad	Subtotal
1	ANGUS GRILL		15,00 €	15,00 €

Fecha del pedido 27/08/2024

Subtotal:	15,00 €
Impuesto Plataforma (10%)	1,50 €
Impuesto Traferencias (3% + 0,50€):	0,45 €
Total:	16,95 €

Imprimir **Generar PDF**

El cliente puede acceder a los detalles completos del mismo.

Restaurants Pedidos

Llistat de pedidos realizados

Mostrar todo Filtrar por estado Filtrar por repartidores Filtrar por fecha Buscar ...

Código del pedido	Estado	Platos	Fecha	Opciones
ca6f933c1d58	Estado Actual	• ANGUS GRILL	27/08/2024 a las 02:45	
47af4181f47f	Estado Actual	• pizza margarita • pizza diavola	27/08/2024 a las 02:45	

El cliente puede filtrar los pedidos por estado, restaurante y fecha, entre otros.

Pantallas de gestión del pedido en el restaurante por el responsable de cocina

The screenshot shows a web-based application interface for managing restaurant orders. The title bar indicates the URL is `localhost:8000/pedidos_actualizados/?estado=espera_preparacion&page=2`. The main header says "Llistat de pedidos". There are three order cards displayed:

- Pedido - 3de72484c768**
 - pizza margarita
 - 220 g. de masa de pizza italiana
 - 90 g. de queso mozzarella
 - 100 g. de salsa de tomate natural triturado
 - Aceite de oliva virgen extra
 - pizza diavola
 - 250 g de salsa de tomate frito casera
 - 100 g de queso mozzarella
 - 100 g de pepperoni, salami o chorizo picante

Empezar la elaboración

- Pedido - bf81552bdc41**
 - pizza margarita
 - 220 g. de masa de pizza italiana
 - 90 g. de queso mozzarella
 - 100 g. de salsa de tomate natural triturado
 - Aceite de oliva virgen extra
 - pizza diavola
 - 250 g de salsa de tomate frito casera
 - 100 g de queso mozzarella
 - 100 g de pepperoni, salami o chorizo picante

Empezar la elaboración

- Pedido - 47af4181f47f**
 - pizza margarita
 - 220 g. de masa de pizza italiana
 - 90 g. de queso mozzarella
 - 100 g. de salsa de tomate natural triturado
 - Aceite de oliva virgen extra
 - pizza diavola
 - 250 g de salsa de tomate frito casera
 - 100 g de queso mozzarella
 - 100 g de pepperoni, salami o chorizo picante

Empezar la elaboración

El pedido llega al responsable de la cocina, quien es el encargado de marcar el inicio de la preparación del pedido, el sistema le permite buscar por identificador de pedido o filtrar por estado de espera de la preparación, en preparación. Los pedidos que no han empezado con la elaboración están en un card gris.

The screenshot shows the same web-based application interface for managing restaurant orders. The title bar indicates the URL is `localhost:8000/pedidos_actualizados/?estado=preparacion`. The main header says "Llistat de pedidos". There are three order cards displayed, all of which are now marked as prepared:

- Pedido - 23a4fb9c7795**
 - pizza margarita
 - 220 g. de masa de pizza italiana
 - 90 g. de queso mozzarella
 - 100 g. de salsa de tomate natural triturado
 - Aceite de oliva virgen extra
 - Listo para la entrega
- Pedido - 4c221c4e29b9**
 - pizza margarita
 - 220 g. de masa de pizza italiana
 - 90 g. de queso mozzarella
 - 100 g. de salsa de tomate natural triturado
 - Aceite de oliva virgen extra
 - Listo para la entrega
- Pedido - 47af4181f47f**
 - pizza margarita
 - 220 g. de masa de pizza italiana
 - 90 g. de queso mozzarella
 - 100 g. de salsa de tomate natural triturado
 - Aceite de oliva virgen extra
 - pizza diavola
 - 250 g de salsa de tomate frito casera
 - 100 g de queso mozzarella
 - 100 g de pepperoni, salami o chorizo picante
 - Listo para la entrega

Una vez se finaliza la preparación, se marcan como listos para el envío aquellos pedidos que están de esta forma, señalizados con un card naranja.

Asignar repartidor para la entrega de pedido
47af4181f47f

Seleccionar un repartidor
Yoyo Repciuc

Realizar

También es responsable de asignar un repartidor disponible en el restaurante.

Pantallas de Gestión del Pedido por el Repartidor

Llistat de pedidos para recoger

Escribe la dirección que desea buscar Buscar

Pedidos para recoger Pedidos para entregar Pedidos entregados Pedida devolución

Lobato
no Sep
Santo Tomás
Pedido : 47af4181f47f - David Sarat
Santo Domingo de Guzmán 8, 47003, Valladolid, España
pizza margarita
Mec Leaflet Pizza diavola

Recojer pedido

El repartidor recibe por parte del sistema los pedidos que tiene pendientes para recoger.

Ubicación pedido

Ubicación del Pedido-47af4181f47f

Cuando el repartidor clica en el mapa, el sistema le muestra la ubicación del pedido a ser entregado.

Llistat de pedidos para entregar

Escribe la dirección que desea buscar Buscar

Pedidos para recoger Pedidos para entregar Pedidos entregados Pedida devolución

Pedido	Dirección	Acciones
5c28239d5fde	Tibi Volostiucl Calle del Universo 3, 47011, Valladolid, España pizza margarita Pizza diavola	Entregar
b7cf7683830e	Tibi Volostiucl Calle del Universo 3, 47011, Valladolid, España pizza margarita Pizza diavola	Entregar
47af4181f47f	Santo Domingo de Guzmán 8, 47003, Valladolid, España pizza margarita Pizza diavola	Entregar

Una vez que el repartidor recoge el pedido, comienza su trayecto hacia el destino. Al llegar, presiona el botón de "Entregar".

Validar Pedido con el cliente David Sarat

Lista de pedidos para entregar / Validar Pedido 47af4181f47f

Código de validación de pedido

Validar

El cliente proporciona un código de validación único al repartidor. El sistema verifica este código y marca el pedido como entregado.

Llistat de pedidos entregados

repartidor12

Escribe la dirección que desea buscar Buscar

Pedidos para recoger Pedidos para entregar Pedidos entregados Pedida devolución

Código del pedido	Cliente	Dirección Cliente	Contenido del pedido	Fecha
47af4181f47f	David Sarat	Santo Domingo de Guzmán 8, 47003, Valladolid, España	• pizza margarita • pizza diavola	27 de agosto de 2024 a las 02:45

El sistema verifica este código y marca el pedido como entregado.

Pantallas de las funcionalidades para el Partner

Gestión de Restaurantes:

The screenshot shows a dashboard titled "Llistado de restaurants". It displays two restaurant entries: "Dominos" and "mcdonalds". Each entry includes a thumbnail, the restaurant name, its rating (4 stars for Dominos, 3 stars for mcdonalds), address, and a sidebar with various management options like "Datos Restaurante", "Añadir Plato", "Plato", etc. A navigation bar at the bottom shows page 1 of 1.

Una vez autenticado, el partner puede ver el listado de los restaurantes que tiene en su propiedad. Además, puede visualizar la **puntuación media** basada en todas las reseñas recibidas para cada restaurante, una vez que se han creado platos, añadido repartidores y responsables de cocina, y se han entregado pedidos.

The screenshot shows a form titled "Añadir el restaurante". It includes fields for "Nombre", "País", "Ciudad", "Código postal", "Calle", "Número Calle", "Descripción" (with a text area for input), and "Imagenes" (with a file upload field showing "Ningún archivo seleccionado"). A "Enviar" button is at the bottom. The top right corner shows a link to "Lista de restaurantes / Añadir restaurante".

El partner tiene la capacidad de **dar de alta varios restaurantes**, ya sea en la misma ciudad o en diferentes ciudades, lo que le permite expandir su red de restaurantes fácilmente.

Gestión de Platos:

Añadir un plato para Dominos

Agregar Plato

Nombre del plato:	pizza champiñones
Precio:	12
Tipo de comida:	Italiana
Descuento (%):	20
Imagen:	Seleccionar archivo Logo_4.png
Ingredientes:	Agregar Ingrediente champiñones 20g Eliminar masa Eliminar

Guardar

El partner puede seleccionar un restaurante específico y **añadir nuevos platos** al menú.

Llistat de platos de Dominos

Filtrar por tipos ▾ Escribe la dirección que deseas buscar Buscar

Llistat de platos

Nombre	Precio	Descuento	Ingredientes	Puntuacion	Opciones
pizza margarita	15,00 €	0 %	<ul style="list-style-type: none"> 220 g. de masa de pizza italiana 90 g. de queso mozzarella 100 g. de salsa de tomate natural triturado Aceite de oliva virgen extra 		Reseñas (1)
pizza diavola	20,00 - 17,00 €	15 %	<ul style="list-style-type: none"> 250 g de salsa de tomate frito casera 100 g de queso mozzarella 100 g de pepperoni, salami o chorizo picante 		Reseñas (0)
pizza prosciutto	16,00 - 12,80 €	20 %	<ul style="list-style-type: none"> 220 g. de masa de pizza italiana 250 g de salsa de tomate frito casera 250 g de prosciutto 		Reseñas (0)

También puede ver el **listado de platos** añadidos para un restaurante seleccionado, junto con la **puntuación media** de todas las reseñas recibidas para cada plato.

Gestión de Personal:

Añadir un repartidor Dominos

Nombre:

Apellidos:

Dirección de correo electrónico:

Teléfono: +34

Nombre de usuario:

Envía

El partner puede crear y otorgar acceso en el sistema a usuarios con el rol de **repartidor** o **responsable de cocina** para un restaurante seleccionado. Esto facilita la gestión eficiente del personal.

Llistat de trabajadores de Dominos

ale_yoyo_12 cocina

repartidor12 repartidor

repartidor14 repartidor

Restear Contraseña

Denegar el acceso

Ver los datos

Restear Contraseña

Denegar el acceso

Ver los datos

Restear Contraseña

Denegar el acceso

Ver los datos

Además, el partner tiene acceso a la lista de trabajadores en cada restaurante y puede ver los detalles de sus cuentas, **denegar acceso** o **restablecer contraseñas** según sea necesario.

Pantalla de la Gestión de Reseñas

Reseñas del restaurante Dominos

david_sarat 2 de septiembre de 2024 a las 03:12

Poco espacio , mucho frio

★★★☆☆

El partner puede visualizar las **reseñas** que los clientes han dejado para un restaurante en particular o para un plato específico. Esta función permite al partner monitorear la satisfacción del cliente y responder adecuadamente.

The screenshot shows a web interface for leaving a review. At the top, there are navigation links for 'Restaurantes' and 'Pedidos'. On the right, there's a user profile icon for 'david_sarat'. The main title is 'Reseñas del restaurante Dominos'. Below it, a review by 'david_sarat' from September 2, 2024, at 03:12, rated 4 stars, states: 'Poco espacio , mucho frio'. There's a trash icon next to the review. Below the review form, there's a message box with placeholder text 'Deja tu comentario para la reseña' and an orange 'Enviar' button.

Los clientes también pueden ver las reseñas que han realizado sobre un plato o un restaurante en concreto.

This screenshot shows the same review interface as above, but with a prominent red error message box at the top stating 'error' with a checkmark and 'Ya has dejado una reseña para este restaurante.' Below the message, the review entry is identical to the one in the first screenshot.

Los clientes solo pueden hacer una reseña por restaurante o plato. Para ello, deben asignar una puntuación, aunque no están obligados a escribir un comentario. Sin embargo, esta opción solo está disponible si el cliente ha realizado un pedido anteriormente.

Licencia

Licencia del Documento

Para este documento, he elegido la licencia [Creative Commons Attribution - NonCommercial - ShareAlike 4.0 International](#). Esta licencia permite que otros copien, distribuyan y modifiquen el contenido, siempre y cuando se atribuya correctamente al autor, no se utilice con fines comerciales y se comparta bajo la misma licencia. Hemos escogido Creative Commons porque es una licencia pública, gratuita, robusta y fácil de entender. Además, es ampliamente reconocida y utilizada en motores de búsqueda y aplicaciones web, lo que garantiza una difusión eficiente y segura de nuestro contenido.

Licencia del Programa

Para el sistema de entrega de comida, he optado por una **Licencia de Software Propietario**, es un tipo de licencia en la que el creador retiene todos los derechos sobre el código fuente y la distribución de la misma. He elegido la Licencia de Software Propietario porque me permite garantizar la **rentabilidad** del modelo de negocio, que se basa en comisiones generadas por el uso de la plataforma. Si el software fuera libre, podría ser utilizado o modificado por terceros sin mi autorización, lo que podría reducir o eliminar las fuentes de ingresos. Con esta licencia, controlamos completamente quién puede utilizar nuestro software y cómo puede ser utilizado, protegiendo así nuestra inversión.

Además, una licencia propietaria otorga la **flexibilidad** para realizar actualizaciones, agregar nuevas características o ajustar la plataforma según las necesidades del mercado sin depender de terceros. Esto permitirá adaptar rápidamente las demandas de los usuarios, mejorando continuamente el sistema. Al tener control total sobre el software también permite que el desarrollo siga alineado con los objetivos iniciales.

Planes Futuros para el Desarrollo del Sistema

Notificaciones en Tiempo Real

En futuras iteraciones del sistema, me gustaría implementar un sistema de notificaciones en tiempo real para mejorar la comunicación entre los clientes y los restaurantes. Esto permitiría a los usuarios recibir actualizaciones instantáneas sobre el estado de sus pedidos, como la confirmación de recepción, el inicio de la preparación y la estimación de la entrega. Las notificaciones podrían enviarse a través de correos electrónicos, mensajes de texto o notificaciones push en la web, proporcionando una experiencia más dinámica y transparente.

Pago a Restaurantes

Actualmente, el sistema está configurado para procesar pagos del cliente al sistema, pero en el futuro, planeo desarrollar una funcionalidad para manejar los pagos a los restaurantes correspondientes. Este desarrollo implica la integración de un sistema de pagos que permita transferir automáticamente los fondos a los restaurantes una vez que se complete el pedido. Esto no solo simplificará el proceso de liquidación para los restaurantes, sino que también aumentará la eficiencia del sistema al reducir la necesidad de intervención manual en las transacciones financieras.

Estadísticas de Ventas por Restaurante

Otra mejora considerada es la implementación de un módulo de estadísticas de ventas por restaurante y uso por parte de los distintos usuarios de la app, permitiendo obtener conocimiento adicional sobre qué aspectos de mejora existen en el servicio de los restaurantes, en el diseño de la carta de los restaurantes, nuevas oportunidades de negocio, etc.

Conclusiones

El desarrollo de este Trabajo de Fin de Grado (TFG) ha sido una experiencia enriquecedora que me ha permitido consolidar y aplicar los conocimientos adquiridos en diferentes asignaturas a lo largo de la carrera, especialmente en la especialización de software. Además, me ha ayudado a profundizar en el uso de Django, así como en las otras etapas del ciclo de vida del software. A través de este trabajo, he aprendido a enfrentar y resolver problemas técnicos complejos, a gestionar el tiempo y los recursos de manera eficiente, y a garantizar una experiencia de usuario óptima, entre otros. Considero que este proyecto refleja fielmente las competencias adquiridas durante mi formación, y estoy convencido de que los conocimientos y habilidades desarrollados durante la realización de este TFG serán de gran utilidad en mi futura carrera profesional.

Referencias Bibliográficas

- How to use Q objects for complex queries? — Django ORM Cookbook 2.0 documentation.* (s. f.). https://books.agiliq.com/projects/django-orm-cookbook/en/latest/query_relatedtool.html
- Akpara, O. J. N. (2019). Requirements Specification. *ResearchGate*. https://www.researchgate.net/publication/332763217_Requirements_Specification
- AlgoriSoft. (2022, 28 enero). *Curso de Django Avanzado | Introducción | Video 1* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=3y26ur9HbBg>
- behave-django — behave-django 1.4.0 documentation.* (s. f.). <https://behave-django.readthedocs.io/en/stable/>
- Bluuweb. (2020, 13 noviembre). *DOM - Curso JavaScript Moderno - #01 ¿Qué es el DOM?* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=11MEBKljhFc>
- Class-based views | Django documentation.* (s. f.). Django Project. <https://docs.djangoproject.com/en/5.1/topics/class-based-views/>
- Contributors, M. o. J. T. A. B. (s. f.). *Introduction.* <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
- Curso de Interacción Persona-Ordenador. (2023, 20 junio). *Modelo de proceso de la usabilidad y la accesibilidad.* <https://mpiua.invid.udl.cat>
- Database transactions | Django documentation.* (s. f.). Django Project. <https://docs.djangoproject.com/en/5.1/topics/db/transactions/>
- Developer. pe. (2020, 17 mayo). *61.- Curso Django 2 | Peticiones AJAX con Django: Listado Parte 1* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=KN7un1aqluo>
- Django. (s. f.). Django Project. <https://www.djangoproject.com/django-filter/24.3/documentation>. (s. f.). https://django-filter.readthedocs.io/en/stable/django-renderpdf—django_renderpdf/4.0.0/documentation. (s. f.). <https://django-renderpdf.readthedocs.io/en/stable/>
- How to manage static files (e.g. images, JavaScript, CSS) | Django documentation.* (s. f.). Django Project. <https://docs.djangoproject.com/en/5.1/howto/static-files/introduction>
- (s. f.). AdminLTE v3.2 Documentation. <https://adminlte.io/docs/3.2/>
- Johann-S. (s. f.). *GitHub - Johann-S/bs-stepper: A stepper for Bootstrap 4.x.* GitHub. <https://github.com/Johann-S/bs-stepper>
- SINERGIA. (2023, 16 febrero). *Puntuación con estrellas con HTML y CSS [Muy fácil - Tutorial]* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=tR5-3RY1kJg>
- Tutorial de Django Parte 7: Framework de sesiones - Aprende desarrollo web | MDN.* (s. f.). MDN Web Docs. <https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Sessions>
- UML 2.5. (s. f.). Google Books. https://books.google.com.pe/books?id=sCU_bpeIECAC
- Using the Django authentication system | Django documentation.* (s. f.). Django Project. <https://docs.djangoproject.com/en/5.1/topics/auth/default/>
- W3Schools.com. (s. f.). <https://www.w3schools.com/>
- Zepeda, E. (2024, 4 septiembre). Django Annotate y aggregate explicados. *Coffee bytes*. <https://coffeebytes.dev/es/django-annotate-y-aggregate-explicados/>