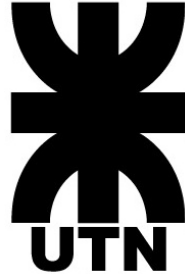


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL SANTA FE



SEMINARIO INTEGRADOR
INFORME FINAL DE PROYECTO

***“Rediseño y optimización del módulo de Clientes de
SantaFood: una WebApp para una empresa de delivery”***

Integrantes:

- Beltramino, Jeremías
jerebeltra2@gmail.com
L.U. N° 27666
- Kraft, Juan Cruz
kraft.juancruz@gmail.com
L.U. N° 28480
- Marchese, Luciano Albertino
luciano.marchese25@gmail.com
L.U. N° 27488

Docentes:

- Ballejos, Luciana
- Yanotti, Paula

Fecha de presentación: 19 de septiembre de 2025

ÍNDICE

1. Introducción	2
1.1 Contexto, problema y fundamentación	2
1.2 Objetivos	2
1.2.1 Objetivo general	2
1.2.2 Objetivos específicos	2
1.3 Alcance	3
1.4 Población objetivo	3
2. Metodología y herramientas	3
2.1 Proceso de desarrollo y gestión de tareas	3
2.2 Herramientas utilizadas	4
3. Requerimientos	6
3.1 Requerimientos funcionales	6
3.2 Requerimientos no funcionales	12
4. Casos de uso	13
4.1 Diagrama de casos de uso	13
5. Diseño del sistema	14
5.1 Arquitectura	14
5.2 Modelos de diseño	14
5.2.1 Diagrama de clases	14
5.2.2 Diagrama de Entidad Relación	15
5.2.3 Diagrama de tablas	17
5.3 Observaciones.	19
6. Workflow del sistema	19
7. Conclusiones y trabajo futuro	25
8. Referencias	26
9. Glosario	27

1. Introducción

1.1 Contexto, problema y fundamentación

La empresa posee un desarrollo de un sistema previo desarrollado en Java y Thymeleaf, el cual se encuentra en funcionamiento pero se comporta de manera irregular, y no es amigable para el usuario, entre otros problemas. Esto tiene como consecuencia que la empresa esté perdiendo clientes debido a que muchos usuarios reportan errores múltiples en el sistema. Se necesitan acciones de rediseño, depuración y optimización de este sistema que permita a los clientes tener una experiencia satisfactoria al momento de realizar sus pedidos.

1.2 Objetivos

Se establece cuales son los objetivos que definen la ruta a tomar para llevar a cabo el proyecto mencionado.

1.2.1 Objetivo general

Rediseñar la aplicación web de la empresa de delivery, denominada SantaFood, enfocándose principalmente en el módulo de Clientes, mejorando funciones existentes, agregando nuevos servicios, optimizando el sistema y modernizando su interfaz para ofrecer una experiencia amigable, con la intención de atraer más cantidad de usuarios y de esa forma generar más valor a la empresa.

1.2.2 Objetivos específicos

- Migrar el sistema a nuevas tecnologías.
- Optimizar el sistema para que mejore la gestión de pedidos.
- Implementar medidas de seguridad en el sistema.
- Rediseñar el sistema para que sea amigable e intuitivo a los usuarios.
- Definir rol de cliente mediante registro e inicio de sesión de usuario.
- Desarrollar el módulo de cliente.
- Permitir a los clientes visualizar los restaurantes cercanos.
- Permitir a los clientes realizar pedidos de manera sencilla, seleccionando ítems desde un catálogo digital de cada vendedor.
- Proteger los datos personales de los usuarios y de las transacciones de pago
- Reposicionar a la empresa en el mercado atrayendo nuevamente a los usuarios.

- Gestionar la información del estado de los pedidos realizados por un cliente, facilitando su control desde la compra hasta la entrega.

1.3 Alcance

El alcance del problema se restringe al módulo Cliente del sistema de delivery. Se focaliza en las deficiencias que afectan la experiencia del usuario durante la interacción con la plataforma, incluyendo la visualización de vendedores, la selección de productos, la gestión del carrito, la confirmación del pedido y el seguimiento de su estado. No se abordan problemáticas relacionadas con los módulos de Vendedor, Repartidor o Administración, ni funcionalidades internas como la logística de entrega o la gestión de pagos externos.

1.4 Población objetivo

El sistema propuesto estará enfocado en suplir necesidades de usuarios que:

- Se encuentren en la ciudad de Santa Fe.
- Tengan entre 13 y 75 años.
- Cuenten con acceso a internet y a un dispositivo compatible con navegador web.
- Dispongan de poco tiempo para preparar su propia comida.
- Que no tengan vehículo propio y quieran disfrutar un plato de un negocio en particular.
- Personas que simplemente quieran que la comida sea enviada hacia su casa por comodidad y ahorrarse el viaje pagando un monto extra.

2. Metodología y herramientas

2.1 Proceso de desarrollo y gestión de tareas

Para mantener una comunicación fluida y un avance organizado dentro del grupo de trabajo, se establece un proceso de desarrollo basado en reuniones constantes, tanto presenciales como virtuales. En dichas instancias se discuten los aspectos a implementar, se realizan bosquejos y diagramas, y se trabaja de manera conjunta en la programación de funcionalidades del sistema.

En cuanto a la gestión de versiones, se utiliza GitHub [1] como repositorio central. Cada integrante trabaja sobre una rama personal, lo que permite un desarrollo

paralelo y seguro, mientras que la rama “*main*” se mantuvo como la versión estable del proyecto.

Para la gestión de tareas se emplea la herramienta *Trello*, la cual facilita la organización del flujo de trabajo mediante un tablero con columnas que representan las tareas pendientes, en proceso y finalizadas; además de documentar los errores que se encuentran durante el desarrollo. Esta metodología permitió tener una visión clara del estado del proyecto en todo momento, mejorando la coordinación entre los integrantes del equipo.

2.2 Herramientas utilizadas

La aplicación SantaFood se desarrolla con las siguientes herramientas:

- **Frontend:**

- Next.js [3] (basado en React, utilizando TypeScript): Se elige por ser un framework moderno que combina la flexibilidad de React con funcionalidades adicionales de interés para nuestro desarrollo. El uso de TypeScript permite contar con tipado estático, reduciendo errores y mejorando el análisis del código.
- Node.js: Se utiliza para ejecutar Next.js en el lado del servidor y para gestionar el entorno de desarrollo, facilitando la instalación de dependencias.

- **Backend:**

- Spring Boot (Java versión 21) [4]: Se elige porque proporciona un marco robusto para el desarrollo de aplicaciones, simplificando la configuración y ofreciendo soporte para arquitectura REST. La versión de Java garantiza compatibilidad con las últimas mejoras del lenguaje y mayor rendimiento.
- Maven: Nos permite organizar, descargar y gestionar todas las librerías necesarias para el backend.

- **Base de datos:**

- MySQL: Se utiliza por ser un motor relacional ampliamente probado, confiable y con buena integración tanto con Spring Boot como con herramientas de administración (PhpMyAdmin). Su modelo relacional facilita la consistencia de datos.

- PhpMyAdmin: Se emplea como herramienta gráfica para la administración de la base de datos, permitiendo ejecutar consultas, gestionar tablas y verificar datos de manera sencilla.
- **Control de versiones:**
 - Git y GitHub: Permite llevar un control detallado de los cambios en el código, gestionar ramas de desarrollo y colaborar de forma ordenada.
- **Otras herramientas utilizadas:**
 - JPA e Hibernate: Se utiliza para implementar la capa de persistencia mediante mapeo objeto-relacional (ORM), reduciendo la complejidad de las consultas SQL y facilitando la interacción entre entidades del modelo y la base de datos.
 - Spring Security Crypto: se integra para garantizar seguridad en la aplicación, específicamente para encriptar datos sensibles almacenados en la base de datos, en nuestro caso las contraseñas de los clientes, cumpliendo con los requerimientos solicitados. Se implementó el algoritmo de encriptado “*Bcrypt*” con 10 rondas de *hashing* que es un balance entre eficacia y seguridad.
 - Lombok: se utiliza porque reduce significativamente el código repetitivo en las clases del modelo, permitiendo centrarse en la lógica de negocio y mantener un código más claro y legible.
 - iText [5]: se utiliza porque necesitamos generar comprobantes en formato PDF de manera automática desde el backend, brindando al usuario la posibilidad de descargarlos.
 - Geoapify [6]: plataforma de localización que proporciona un conjunto de APIs para integrar funciones basadas en la ubicación. En este proyecto se utilizan las siguientes APIs:
 - Autocomplete API: se utiliza cuando el usuario ingresa su dirección. Esta API devuelve una lista de sugerencias de direcciones posibles en tiempo real, facilitando la selección y reduciendo errores de escritura, además de brindar el formato de texto esperado para que Geocoding Api procese la misma. Estas sugerencias que se muestran son personalizadas para coincidir con direcciones dentro de la ciudad de Santa Fe
 - Geocoding API: se utiliza para convertir la dirección seleccionada en coordenadas geográficas (latitud y longitud), lo que permite guardarla en la base de datos y realizar posteriormente operaciones logísticas.

3. Requerimientos

3.1 Requerimientos funcionales

1. El sistema debe permitir a los usuarios registrarse proporcionando: nombre, apellido, identificador de usuario, dirección de entrega, correo electrónico, contraseña, confirmación de contraseña, cuil/cuit, número de teléfono.
2. El sistema debe validar que los datos ingresados por el usuario en el formulario de registro cumplan con los siguientes criterios:
 - Todos los campos son obligatorios, deben estar completos.
 - El correo electrónico debe coincidir con el siguiente formato:
 - Longitud máxima de 254 caracteres.
 - Tener una sola arroba (@).
 - Tener al menos un carácter antes y después del @.
 - Tener un dominio con al menos un punto (.) y una extensión válida.
 - No contener espacios ni caracteres especiales ilegales.
 - No comenzar ni terminar con @ o con punto.
 - La contraseña debe cumplir con los siguientes requisitos:
 - La contraseña debe contener una longitud mínima de 8 caracteres.
 - La contraseña debe contener al menos una letra mayúscula.
 - La contraseña debe contener al menos un dígito.
 - El CUIL/CUIT debe coincidir con el siguiente formato:
 - Debe tener una longitud fija de 11 dígitos.
3. El sistema debe validar en el registro de un nuevo usuario que el identificador de usuario y el CUIT/CUIL proporcionados no hayan sido previamente registrados en un cliente activo.
4. El sistema debe validar en el registro de un nuevo usuario que el campo contraseña y el campo confirmación de la contraseña sean idénticos.
5. El sistema debe permitir al cliente iniciar sesión proporcionando identificador de usuario y contraseña.
6. El sistema debe validar que el identificador de usuario y la contraseña ingresadas por el usuario coincidan con las de un cliente registrado.
7. El sistema debe permitir a un usuario no autenticado registrarse.
8. El sistema debe permitir a un usuario no autenticado iniciar sesión.

9. El sistema debe listar vendedores disponibles, calculando y mostrando para cada vendedor: nombre del local, dirección del local, distancia entre el vendedor y el cliente, costo del envío, y tiempo estimado de entrega.
10. El sistema debe calcular y ordenar por distancia al cliente los vendedores pertenecientes al listado de vendedores disponibles.
11. El sistema debe permitir a un cliente visualizar el menú de un vendedor.
12. El sistema debe informar al cliente si no hay vendedores disponibles.
13. El sistema debe listar todos los productos disponibles del menú de un vendedor.
14. El sistema debe mostrar, por cada producto disponible del menú de un vendedor:
 - Nombre del producto.
 - Descripción del producto.
 - Precio unitario en pesos.
 - Categorías.
 - Peso expresado en gramos.
 - Stock disponible.

Además, si:

- El producto pertenece a la categoría “Plato”:
 - Calorías expresadas en KiloCalorías.
 - El producto pertenece a la categoría “Bebida”:
 - Tamaño en mililitros.
 - Graduación alcohólica porcentual.
15. El sistema debe permitir ordenar los productos por precio o por nombre, ascendente o descendente.
 16. El sistema debe permitir al cliente filtrar según las categorías existentes en los ítems del menú de un vendedor.
 17. El sistema debe permitir al cliente agregar uno o más ítems del menú de un vendedor a su carrito, indicando la cantidad de ítems deseada.
 18. El sistema debe verificar que el cliente no tenga un carrito activo con otro vendedor antes de permitir la adición de productos. Si no existe un carrito del cliente con el vendedor actual, el sistema debe crear uno.
 19. El sistema debe validar que el stock del producto sea mayor o igual a la cantidad solicitada antes de permitir su agregado al carrito.
 20. El sistema muestra un mensaje si el stock disponible no cubre la cantidad solicitada, informando la cantidad disponible.
 21. El sistema debe actualizar el subtotal del carrito cada vez que se modifica, agrega o elimina un ítem al mismo.

22. El sistema debe actualizar la cantidad solicitada si se agrega un ítem que ya estaba presente en el carrito.
23. El sistema debe crear un pedido asociado a un cliente y a un vendedor, almacenando los siguientes datos: identificador del cliente, identificador del vendedor, listado de productos (inicialmente vacía) y estado “en carrito”.
24. El sistema debe actualizar el estado de un pedido, los posibles estados son: “en carrito”, “confirmado”, “en envío” y “entregado”. Los vendedores y administradores son los encargados de modificar estos estados.
25. El sistema debe permitir la visualización del carrito de compras al cliente, si este existe, mostrando:
 - Subtotal del carrito.
 - Dirección de entrega.
 - Tiempo de entrega.
 - Costo de envío.Además, para cada producto del carrito, el sistema debe mostrar:
 - Nombre de producto.
 - Precio unitario.
 - Cantidad solicitada.
26. El sistema debe permitir a un cliente confirmar el pedido correspondiente al carrito.
27. El sistema debe permitir a un cliente eliminar el carrito mientras aún no haya sido confirmado.
28. El sistema debe permitir eliminar un ítem del carrito si el cliente lo solicita.
29. El sistema debe devolver al stock del vendedor la cantidad correspondiente del ítem, una vez confirmada la eliminación del carrito.
30. El sistema debe permitir al cliente aumentar o disminuir la cantidad de un producto en su carrito.
31. El sistema debe restringir la disminución de la cantidad de un ítem del carrito cuando se desea modificar la cantidad del mismo, a un mínimo de una (1) unidad.
32. El sistema debe permitir la eliminación de un carrito asociado a un cliente con un vendedor, eliminando todos los ítems contenidos en el carrito y, por cada ítem, se devolverá al stock del vendedor la cantidad correspondiente del ítem.
33. El sistema debe permitir a los clientes buscar vendedores disponibles utilizando uno de los siguientes criterios: nombre del local o nombre de un producto ofrecido. El sistema debe ejecutar la búsqueda y mostrar los vendedores que cumplan con lo solicitado. Si no se encuentra ningún vendedor que cumpla con los criterios ingresados, el sistema debe notificar al cliente que no existen vendedores que cumplan con el criterio de búsqueda.

34. El sistema debe confirmar un pedido de un cliente luego de que el pago asociado al mismo se confirme. En el momento que se confirma el pago del pedido, el pedido transiciona del estado “en carrito” a “confirmado”.
35. El sistema debe recibir confirmación del pago de un pedido desde un sistema de pagos externo.
36. El sistema debe permitir al cliente visualizar su historial de pedidos realizados. Por cada pedido realizado y en estado “entregado”, se mostrará:
 - Nombre del vendedor.
 - Fecha de transacción.
 - Monto pagado.
 - Costo de envío.Además, para cada ítem del pedido, se deberá mostrar:
 - Nombre del producto.
 - Cantidad adquirida del producto.
 - Precio unitario del producto .
37. El sistema debe mostrar al cliente las opciones de pago disponibles junto a sus porcentajes de recargo, antes de confirmar su compra.
38. El sistema debe mostrar el precio total del pedido acorde al método de pago seleccionado. Los métodos de pagos y sus respectivos porcentajes de recargo son:
 - Transferencia bancaria: 0% recargo
 - Tarjeta de débito: 0% de recargo.
 - Tarjeta de crédito: 10% de recargo.Estos porcentajes se calcularán en base al monto total del carrito incluyendo su costo de envío.
39. El sistema debe solicitar confirmación al cliente de la operación de pago una vez elegido el método de pago. Cuando el cliente confirma, el sistema se debe comunicar con un sistema de pagos externo que efectúa la transacción en base al método de pago seleccionado. Una vez que el sistema de pagos externo envía confirmación de la transacción de pago, el sistema marca el pedido al estado “confirmado”.
40. Si el pago de un pedido en estado “en carrito” se realiza en manera exitosa, el sistema debe cambiar el estado del pedido a “Confirmado”
41. El sistema debe mostrar, luego de la confirmación del pago de un pedido, un resumen del pago al cliente, incluyendo:
 - Número de ticket
 - Fecha y hora de la operación
 - Monto total pagado

y la opción para descargar la factura correspondiente

42. El sistema debe permitir la generación de una factura para cada pedido pagado, la cual puede ser descargada por el cliente. En la factura se deberá incluir:

- Número de factura
- Fecha y hora del pago
- Dirección de entrega
- Método de pago
- Monto total
- Datos del cliente (nombre, apellido, CUIL/CUIT,email)
- Detalle del pedido (por cada ítem del producto: nombre del producto, cantidad adquirida, precio unitario).
- Costo de envío.

43. El sistema debe permitir al cliente visualizar su perfil, mostrando los siguientes datos:

- Nombre
- Apellido
- Dirección de entrega
- CUIL/CUIT
- Email
- Identificador de usuario

44. El sistema no debe permitir al cliente modificar los siguientes campos de su perfil: CUIL/CUIT e identificador de usuario.

45. El sistema debe permitir al cliente calificar a un vendedor en base a algún pedido realizado. El sistema deberá mostrar un formulario con: escala de puntuación (1 a 5) y un campo para un comentario opcional. Una vez que el cliente envía la calificación el sistema debe guardarla y marcar el pedido como calificado.

46. El sistema debe permitir al cliente modificar los siguientes campos de su perfil: nombre, apellido, dirección y email.

47. El sistema no debe permitir a un cliente calificar a un vendedor en base a algún pedido si dicho pedido ya fue previamente calificado.

48. El sistema debe permitir la visualización de las calificaciones de un vendedor.

49. El sistema debe calcular y mostrar para un vendedor:

- Promedio general de calificación.
- Cantidad total de calificaciones.

Además, el sistema debe mostrar para cada calificación del vendedor:

- Nombre del cliente que calificó.
- Puntaje.

- Comentario (si existe).

50. El sistema debe permitir al cliente eliminar su perfil.
51. El sistema debe solicitar confirmación al cliente si este desea eliminar su perfil.
52. El sistema debe permitir al cliente ingresar su dirección mediante un campo de texto, a partir de esta dirección provista por el cliente, el sistema debe obtener las coordenadas (latitud y longitud), desde un sistema de geolocalización externo.
53. El sistema debe solicitar confirmación de que la dirección del cliente es correcta, posteriormente a su conversión a coordenadas por el sistema externo de geolocalización.
54. El sistema debe enviar la dirección de un cliente al servicio externo de geolocalización para obtener la dirección dada en un formato de coordenadas: longitud y latitud.
55. El sistema debe notificar al cliente si no se pudieron obtener las coordenadas geográficas a partir de la dirección que proveyó.
56. El sistema debe calcular la distancia en kilómetros entre las coordenadas del cliente y las del vendedor.
57. El sistema debe calcular el tiempo estimado de entrega en base a la distancia entre el vendedor y el cliente.
58. El sistema debe calcular el costo de envío multiplicando el valor de la distancia en kilómetros por un factor de precio provisto por el administrador del sistema.
59. El sistema deberá permitir al cliente cerrar sesión invalidando la sesión activa.
60. El sistema debe permitir al cliente la visualización de sus pedidos en curso, es decir aquellos cuyo estado sea “confirmado” o “en envío”. Para cada pedido en curso, el sistema debe mostrar:
 - Nombre del vendedor.
 - Fecha y hora de creación.
 - Estado actual.
 - Costo total.
61. El sistema debe permitir al cliente, cuando este esté visualizando sus pedidos en curso, seleccionar un pedido para obtener los siguientes datos:
 - Método de pago
 - Tiempo estimado de entrega
 - Costo de envío
 - Detalle de productos: nombre, cantidad, precio unitario.

3.2 Requerimientos no funcionales

- **RNF #1:** El sistema debe almacenar las contraseñas de los usuarios en la base de datos utilizando “BCRYPT”, que es un algoritmo de hash seguro.
- **RNF #2:** El sistema debe validar las credenciales en cada intento de inicio de sesión, y rechazar accesos incorrectos con un mensaje explicativo.
- **RNF #3:** El sistema deberá mostrar mensajes de error que indiquen al usuario la causa de la falla en lenguaje sencillo y sin tecnicismos.
- **RNF #4:** El tiempo de respuesta para el cálculo logístico (distancia entre cliente y vendedor, estimación de tiempo de entrega y costo de envío) de un cliente respecto a un vendedor no debe superar los 500 ms.
- **RNF #5:** El acceso a la modificación de datos del perfil y a la confirmación de pedidos solo estará disponible para usuarios autenticados.
- **RNF #6:** Si se detecta un intento de agregar un ítem con stock insuficiente, el sistema deberá cancelar la operación y notificar al usuario sin afectar el resto del carrito.
- **RNF #7:** Cuando se solicite la conversión de una dirección en formato de texto a coordenadas geográficas (latitud y longitud), el sistema deberá recibir una respuesta del sistema externo de geolocalización en un tiempo máximo de 10 segundos.
- **RNF #8:** El sistema externo encargado de los pagos debe procesar y confirmar al sistema la transacción en un tiempo máximo de 10 segundos desde el envío de los datos.
- **RNF #9:** Luego de que el cliente confirme la compra, el sistema debe recibir la confirmación del pago exitoso por parte del sistema externo de pagos en un tiempo máximo de 7 minutos. Luego de este tiempo, el sistema asumirá que el pago no fue realizado con éxito.
- **RNF #10:** Los datos del cliente no deberán ser compartidos con terceros sin consentimiento explícito según la Ley de Protección de Datos Personales (Ley 25.326).

4. Casos de uso

4.1 Diagrama de casos de uso

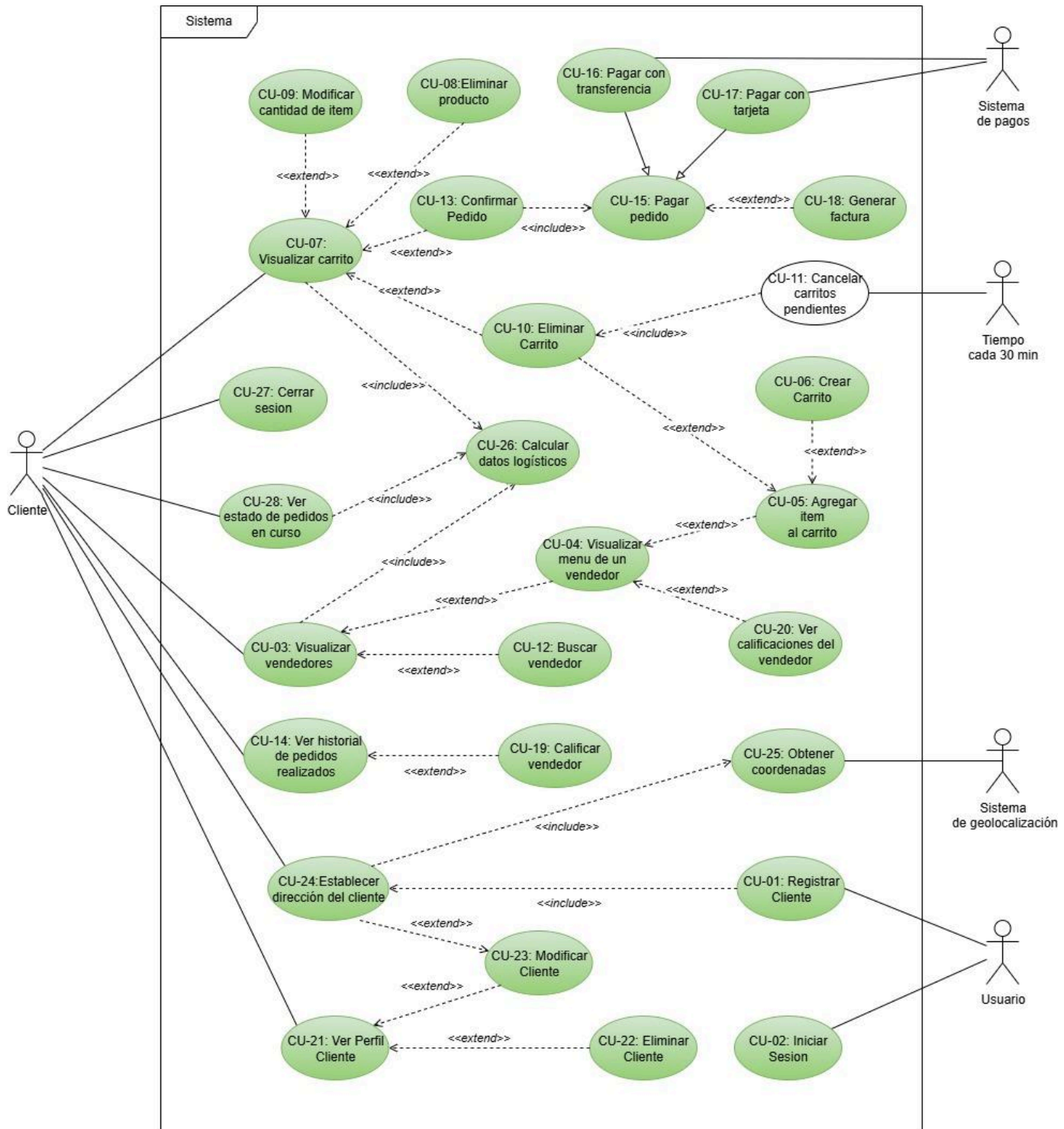


Imagen 4.1: Casos de uso desarrollados en color verde

La planilla de cada caso de uso se encuentra en el **“Anexo 1 - Fichas de especificación de casos de uso”**.

5. Diseño del sistema

5.1 Arquitectura

El sistema se implementa llevando a cabo una estructura del tipo Cliente-Servidor. El proceso de análisis y los modelos de diseño generados en este proyecto tienen como alcance la arquitectura del lado del servidor.

El sistema propuesto se desarrolla siguiendo una arquitectura cliente-servidor, en la cual el cliente (usuario) realiza peticiones al servidor, y este se encarga de procesar la información y devolver una respuesta.

Internamente, el servidor (backend) fue diseñado siguiendo una **arquitectura de tres capas**, promoviendo la separación de responsabilidades, el mantenimiento y la escalabilidad del sistema. Las capas son:

1. Capa de presentación: responsable de recibir las solicitudes del cliente y devolver una respuesta. Está compuesta por los controladores, que actúan como punto de entrada del sistema y delegan el procesamiento a la capa de negocio.
 2. Capa de negocio: Contiene la lógica del sistema, o sea, las reglas que definen el comportamiento del sistema según los requerimientos funcionales. Está compuesta por clases de servicio (services) que coordinan operaciones sobre las entidades, validan datos y gestionan el acceso a los datos a través de los repositorios.
 3. Capa de acceso a datos: Encargada de la persistencia, recuperación y actualización de la información en la base de datos. Se implementa a través de los repositorios (repositories), los cuales actúan como interfaces de acceso a los datos.
- Se agregan además las entidades del modelo, las cuales son manipuladas directamente por la capa de acceso a datos y la capa de negocio. De este modo, se ilustra la interacción entre controladores, servicios, repositorios y entidades, evidenciando la organización modular del sistema.

5.2 Modelos de diseño

5.2.1 Diagrama de clases

El diagrama de clases presentado refleja la arquitectura descrita en la sección 5.1, mostrando cómo se estructuran los distintos componentes del servidor y de qué manera interactúan para dar soporte a los procesos definidos en el sistema.

El diagrama de clases se encuentra en “**Anexo 2 - Diagramas de Clases**”.

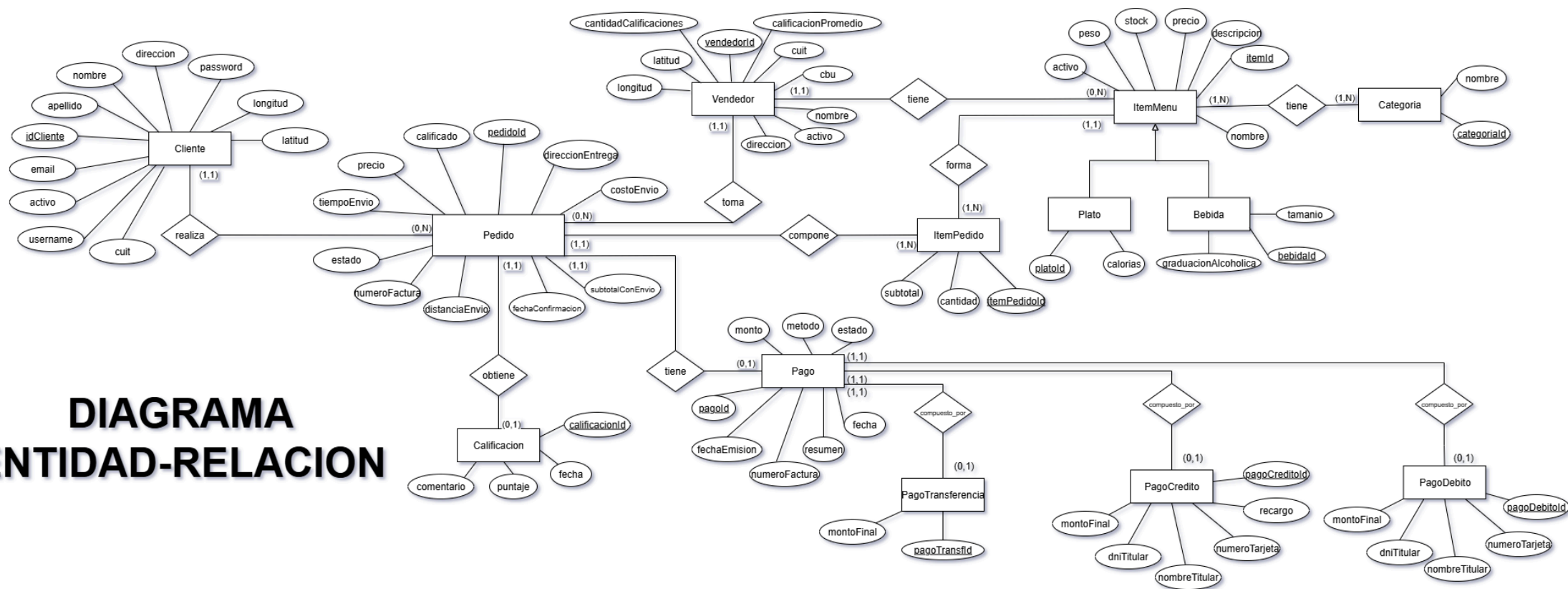
5.2.2 Diagrama de Entidad Relación

El Diagrama de Entidad-Relación se diseña posteriormente a la realización del diagrama de clases, considerando las relaciones definidas en este, además de los requisitos del sistema, y busca asegurar una correcta persistencia de los datos.

Las entidades Cliente y Vendedor incorporan de manera directa los atributos de localización (latitud y longitud), evitando la necesidad de una entidad independiente para coordenadas y simplificando el modelo sin perder precisión en el cálculo de distancias de envío.

Se representan las relaciones opcionales, como es el caso de la entidad Calificación, asociada a Pedido con una relación 0.1, indicando que no todos los pedidos necesariamente serán calificados.

Se modeló la relación entre pedido y pago de forma opcional debido a que un pedido puede tener (o no) a lo sumo un pago asociado. Si el pedido está en estado “en carrito”, aún no tiene un pago asociado, cuando el pedido avanza al estado “confirmado” obligatoriamente se le asocia un pago.



5.2.3 Diagrama de tablas

El Diagrama de Tablas representa la transformación del modelo conceptual (DER) al modelo lógico relacional. Cada entidad fue convertida en una tabla, respetando claves primarias, claves foráneas, tipos de datos apropiados y restricciones de integridad.

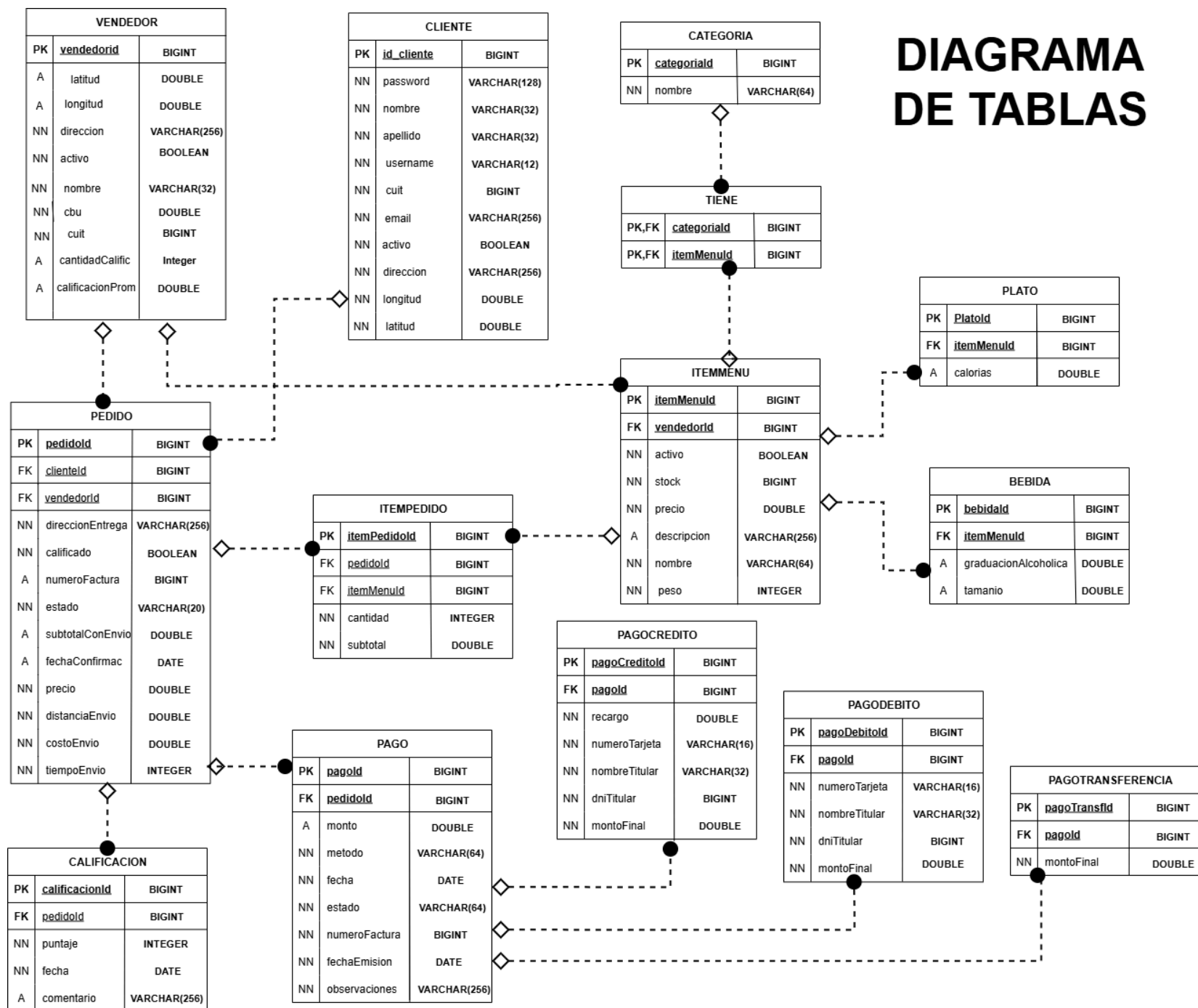
La jerarquía entre ItemMenu, Plato y Bebida se implementa de la siguiente manera:

- La tabla ItemMenu contiene los atributos comunes a todos los ítems del menú.
- Las tablas Plato y Bebida almacenan los atributos específicos de cada tipo y contienen una clave foránea que referencia a ItemMenu.

Las tablas Cliente y Vendedor incorporan directamente los atributos de localización (latitud y longitud), eliminando la necesidad de una entidad separada para coordenadas y evitando así la duplicación de información.

De esta forma, el modelo lógico garantiza integridad referencial, evita redundancias y facilita la escalabilidad futura del sistema.

DIAGRAMA DE TABLAS



5.3 Observaciones.

Observaciones respecto a la capa de acceso a datos implementada con repositories: se utiliza la técnica de Mapeo Objeto-Relacional (ORM) con la herramienta Hibernate, que implementa la especificación JPA (Java Persistence API). Esto permite convertir las clases del modelo en tablas relacionales de forma automática, sin necesidad de escribir consultas SQL explícitas. Debido a esta decisión, en el diagrama de clases los repositorios aparecen representados como interfaces, ya que es Hibernate quien se encarga de generar dinámicamente la implementación de las operaciones CRUD básicas.

6. Workflow del sistema

Esta sección mostrará el flujo y las interfaces del sistema desarrollado a medida que un usuario nuevo ingresa en el sistema para crear un carrito y confirmarlo.

Imagen 6.1: Un usuario ingresa a la página de SantaFood y la primer pantalla que se encuentra será la de inicio de sesión, con opción de registro para nuevos usuarios.

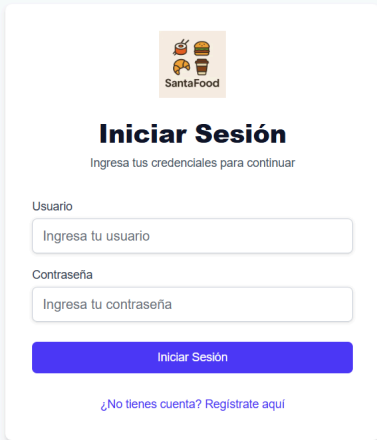
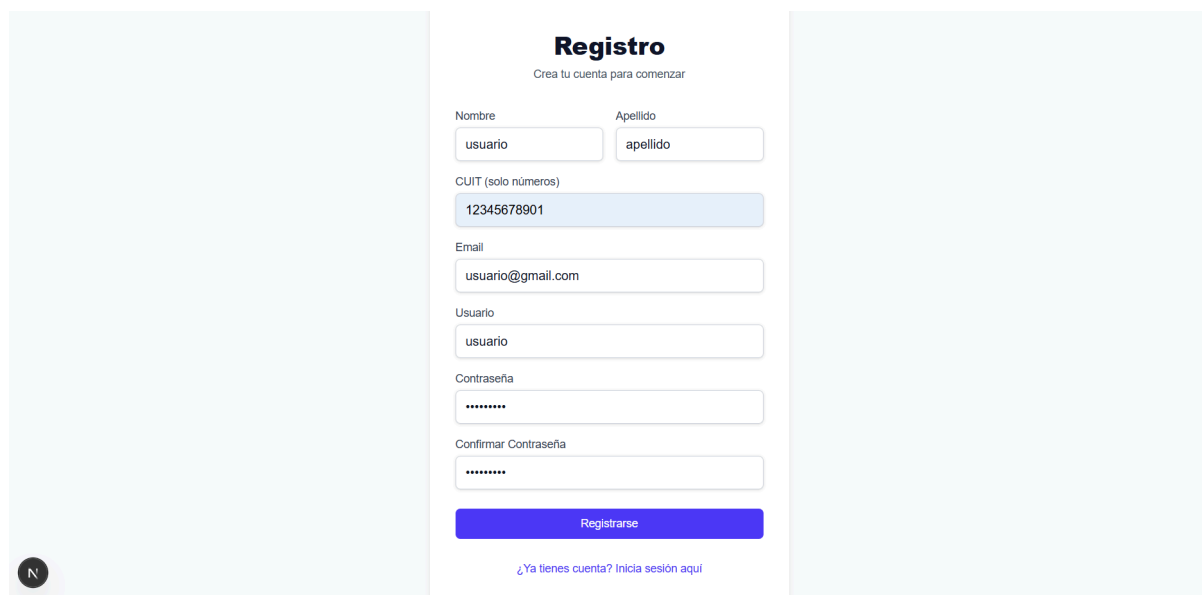


Imagen 6.1 - Inicio de sesión o registro de nuevo usuario.

Como el usuario no se encuentra previamente registrado presiona el botón para llevar a cabo el registro. En la **imagen 6.2** se observa los datos completados por el usuario.



Registro
Crea tu cuenta para comenzar

Nombre: Apellido:

CUIT (solo números):

Email:

Usuario:

Contraseña:

Confirmar Contraseña:

[Registrarse](#)

[¿Ya tienes cuenta? Inicia sesión aquí](#)

Imagen 6.2 - El usuario rellena los campos para su registro.

Imagen 6.3: Posterior a su correcto registro, el usuario debe especificar su dirección, seleccionando la que coincida con la que ingresa.



Configurar Dirección
¡Bienvenido usuario! Para completar tu registro, configura tu dirección de entrega.

Dirección completa:

- ☐ Lavalse 610, República Los Hornos, S3000 Santa Fe, Argentina
- ☐ Lavalse 610, San Pantaleón, S3000 Santa Fe, Argentina
- ☐ Lavalse 610, Soils, 3016 Municipio de Santo Tomé, Argentina

Imagen 6.3 - El usuario ingresa su dirección en el sistema.

Una vez el registro haya sido exitoso, el usuario ingresa a la página principal del sistema, donde puede ver información de los restaurantes, historial de pedidos y carritos pendientes. Así como también puede modificar su perfil, confirmar pagos o cancelar carritos pendientes. En la **imagen 6.4** se observa la interfaz mencionada.

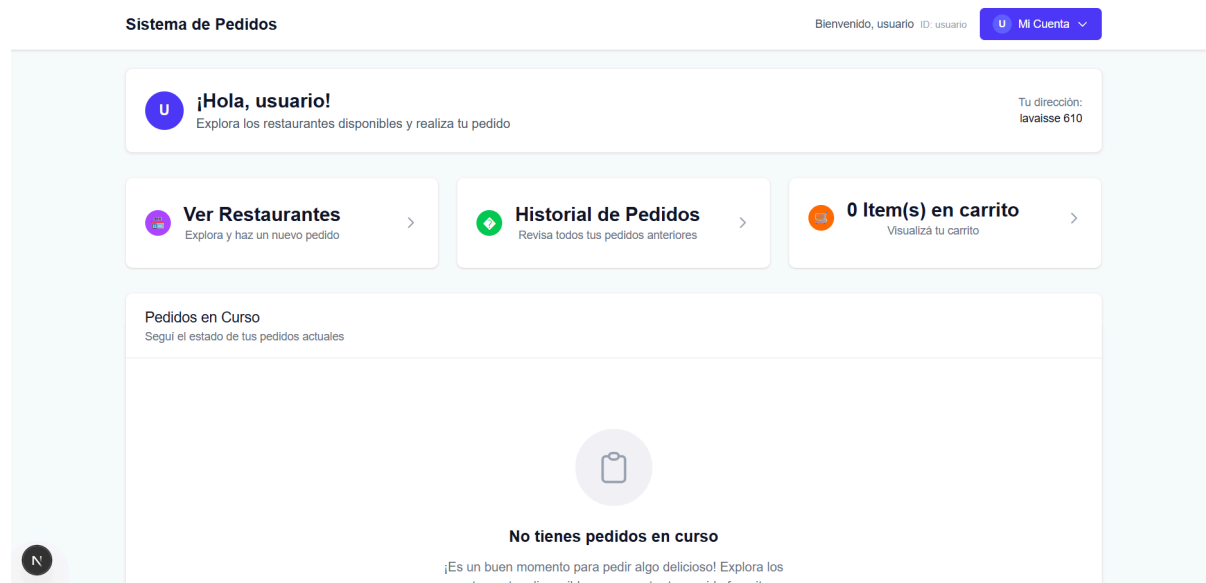


Imagen 6.4 - Página inicial del sistema.

Imagen 6.5: Para poder ver los vendedores, el usuario pulsa en el botón de “Ver Restaurantes”.

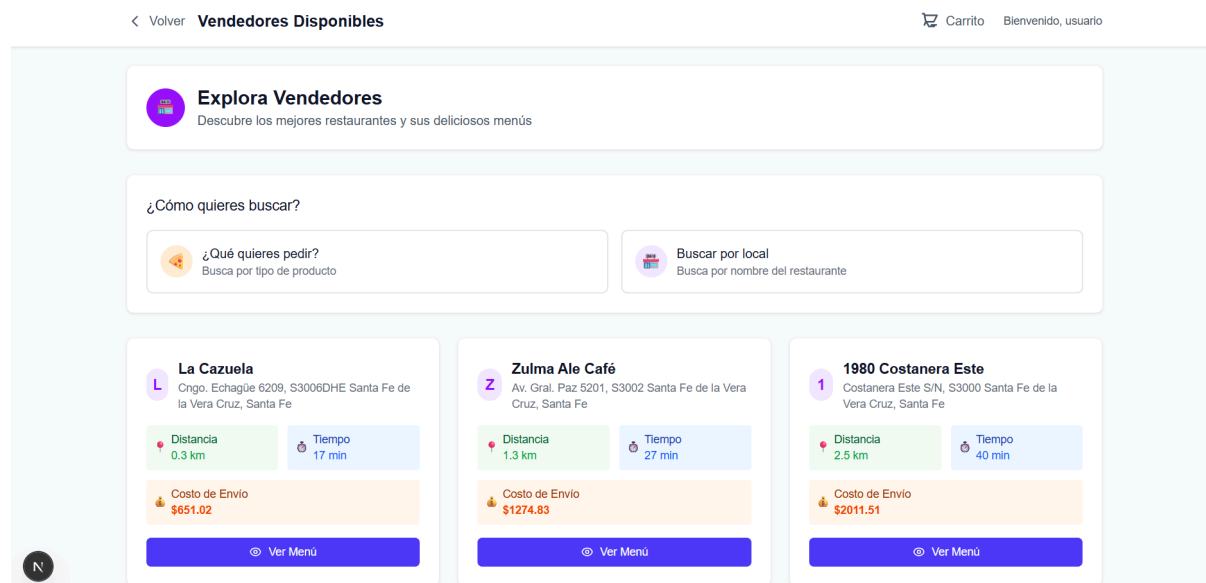


Imagen 6.5 - El usuario observa los Vendedores Disponibles.

En la **imagen 6.6** se observa al usuario ingresando en el buscador la palabra “hamburguesa” para obtener todos los vendedores asociados que venden dicho producto.

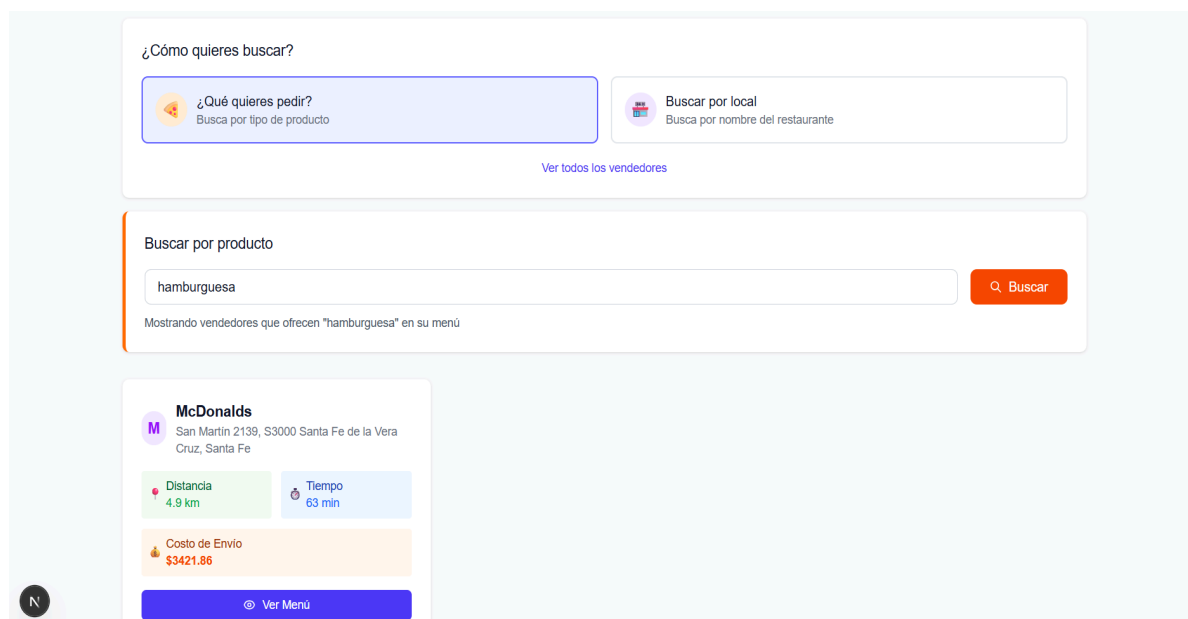


Imagen 6.6 - Usuario busca vendedores que vendan hamburguesas.

Luego de pulsar el botón “Ver Menú”, el usuario es llevado a la interfaz donde se muestran todos los productos que ofrece el vendedor seleccionado. El usuario selecciona dos “Big mac”, dos “Chicken McNuggets 10u” y cuatro “Coca-Cola 500ml”. Esto se observa en la **imagen 6.7**.

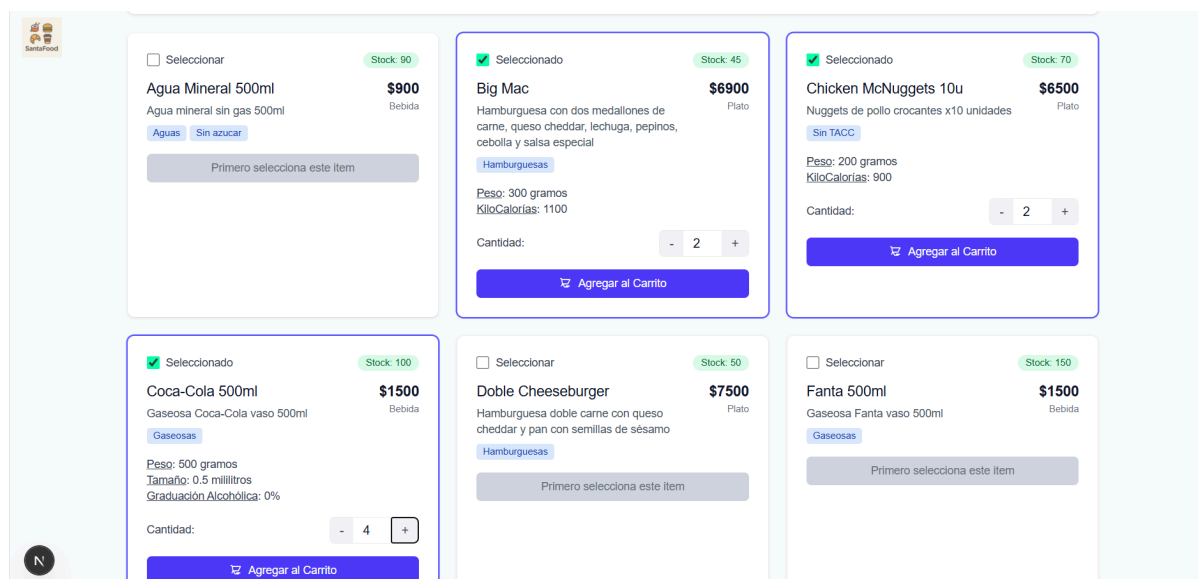


Imagen 6.7 - Usuario visualiza menú del vendedor “McDonalds”.

Luego de agregar dichos productos al carrito, el usuario se dirige al apartado de “Ver Carrito” para visualizar los precios parciales y totales del carrito juntos con el tiempo estimado de la entrega del mismo y la dirección de entrega. Además de tener la posibilidad de editar las cantidades de los productos ingresados. Esto se visualiza en la **Imagen 6.8**.

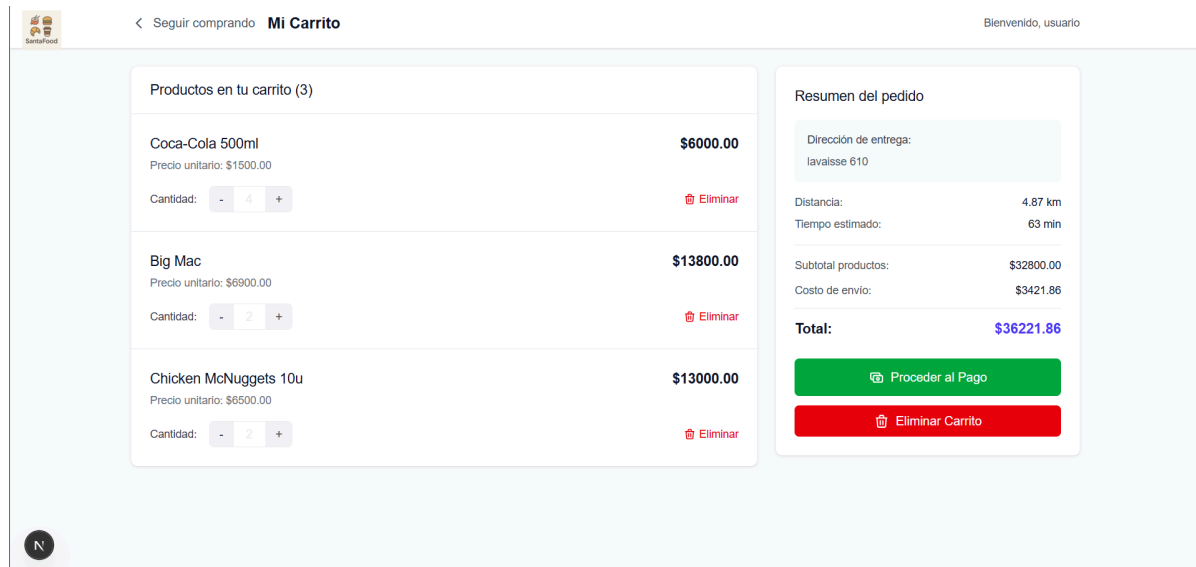


Imagen 6.8 - Información del carrito pendiente del usuario.

Cuando el usuario decide proceder con el pago, el sistema le muestra las opciones de pago disponibles, las cuales se pueden observar en la **imagen 6.9**.

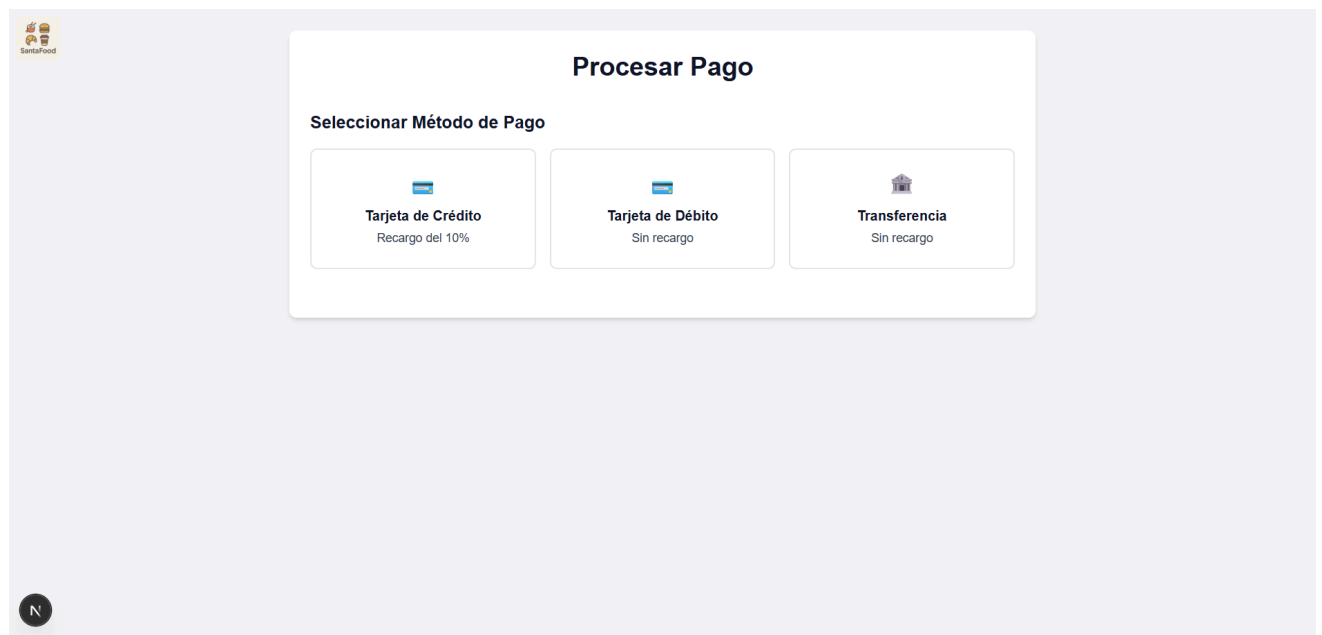
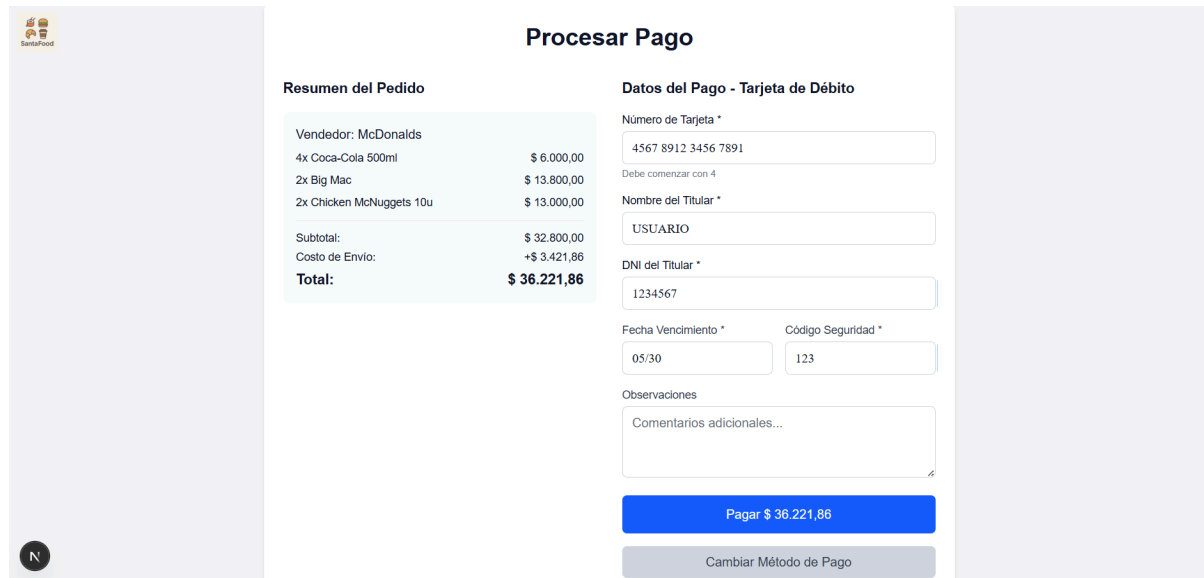


Imagen 6.9 - Métodos de pago habilitados por el sistema.

El usuario selecciona el método de pago “Tarjeta de Débito” y prosigue a rellenar los campos solicitados en la **imagen 6.10**.



Procesar Pago

Resumen del Pedido

Vendedor: McDonalds	
4x Coca-Cola 500ml	\$ 6.000,00
2x Big Mac	\$ 13.800,00
2x Chicken McNuggets 10u	\$ 13.000,00
Subtotal:	\$ 32.800,00
Costo de Envío:	+\$ 3.421,86
Total:	\$ 36.221,86

Datos del Pago - Tarjeta de Débito

Número de Tarjeta *

Debe comenzar con 4

Nombre del Titular *

DNI del Titular *

Fecha Vencimiento *

Código Seguridad *

Observaciones

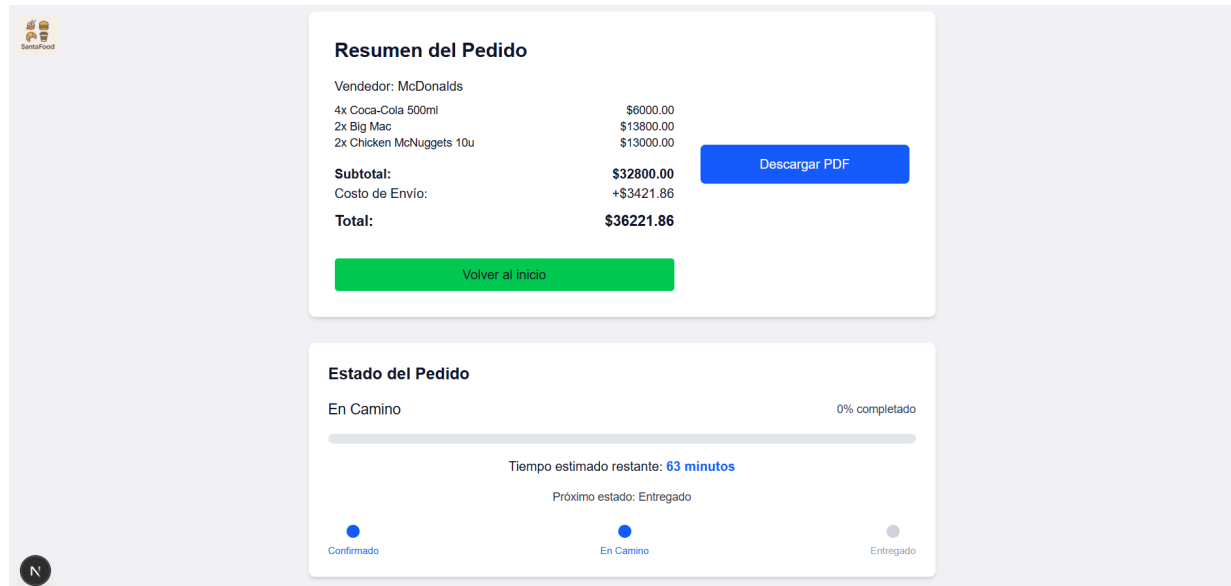
Pagar \$ 36.221,86

Cambiar Método de Pago

Imagen 6.10 - El usuario rellena los campos solicitados para el pago del pedido.

Una vez realizado el pago con éxito, el sistema redirige al usuario a una pantalla donde le muestra el resumen del pedido, facilitando la opción de descargar un archivo PDF con la factura del mismo.

Debajo, se observa el estado actual del pedido, que comienza “En preparación”, luego transiciona a “En camino” y finaliza en “Entregado”. Se puede apreciar en la **Imagen 6.11**.



Resumen de compra del pedido

Resumen del Pedido

Vendedor: McDonalds	
4x Coca-Cola 500ml	\$6000.00
2x Big Mac	\$13800.00
2x Chicken McNuggets 10u	\$13000.00
Subtotal:	\$32800.00
Costo de Envío:	+\$3421.86
Total:	\$36221.86

Descargar PDF

Volver al inicio

Estado del Pedido

En Camino 0% completado

Tiempo estimado restante: **63 minutos**

Próximo estado: Entregado

Confirmado

En Camino

Entregado

Imagen 6.11 - Resumen de compra del pedido.

7. Conclusiones y trabajo futuro

El proyecto permite rediseñar y optimizar el módulo de clientes de SantaFood, alcanzando una solución más robusta, moderna y escalable que responde a las necesidades detectadas en la versión previa. A lo largo del desarrollo se logró una migración tecnológica hacia herramientas actuales, como Next.js en el frontend y Spring Boot en el backend, lo que garantiza un mejor rendimiento y una mayor mantenibilidad del sistema. Esta actualización no solo moderniza la plataforma, sino que también sienta bases sólidas para la incorporación de nuevas funcionalidades en el futuro.

Otro aporte fundamental es la definición de una arquitectura en capas, que separa de manera clara la presentación, la lógica de negocio y la persistencia de datos. Esta organización permite una mayor escalabilidad y facilita el mantenimiento a largo plazo, dado que cada capa puede evolucionar de forma independiente sin afectar al resto del sistema.

Asimismo, se implementan los requerimientos funcionales y no funcionales definidos en la etapa inicial, lo que se traduce en una experiencia de usuario más intuitiva, segura y confiable. Junto con ello, la construcción de los diagramas de clases, entidad-relación y tablas permitió validar la consistencia del modelo de datos, establecer relaciones claras entre entidades y garantizar la integridad del sistema.

Estos artefactos de diseño resultan clave para fundamentar las decisiones técnicas y para asegurar la correcta implementación del sistema. Entre los logros alcanzados se destaca la incorporación de un flujo de interacción completo que abarca registro, gestión del carrito, procesos de pago y confirmación de pedidos, cubriendo de manera integral las principales necesidades del cliente. Estas mejoras permiten superar los problemas detectados en la versión anterior del sistema, donde la experiencia de usuario se veía limitada por errores frecuentes y una interfaz poco amigable.

A pesar de los avances logrados, el sistema aún cuenta con un amplio margen de crecimiento. Como líneas de trabajo a futuro se proyecta la extensión del alcance hacia otros módulos, como vendedor, repartidor y administración, lo que brindaría una cobertura integral de la aplicación y permitiría mejorar la coordinación de todo el ecosistema de delivery. También se considera prioritaria la incorporación de notificaciones en tiempo real que informen al cliente sobre el estado de su pedido, desde la preparación hasta la entrega, fortaleciendo la comunicación y la confianza en el servicio.

Otra posible mejora es la implementación de sistemas de recomendación personalizados, basados en el historial de compras y en las preferencias del usuario, con el objetivo de incrementar la fidelización y generar una experiencia más atractiva.

También se considera a futuro el desarrollo de una aplicación móvil nativa, que permita ampliar el alcance de SantaFood más allá del entorno web. Esta iniciativa busca ofrecer una experiencia optimizada para dispositivos móviles, facilitando el acceso rápido, las notificaciones instantáneas y una mayor usabilidad para los clientes. A su vez, la aplicación móvil representa una oportunidad para expandir la cobertura geográfica de la plataforma, comenzando por las provincias limítrofes a Santa Fe y proyectando un crecimiento escalable hacia otras regiones del país.

8. Referencias

- [1] [Repositorio de GitHub – Seminario Integrador.](#)
- [2] [Grupo de trabajo Seminario Integrador. Repositorio en Google Drive con documentación.](#)
- [3].[Next.js Documentation](#)
- [4] [Spring. Spring Boot Reference Guide.](#)
- [5] [iText Suite Java](#)
- [6] [Geoapify](#)

9. Glosario

Pedido: conjunto de ítems de pedido asociados a un cliente y a un vendedor. Representa la solicitud de productos realizada al vendedor.

Ítem de Menú: producto disponible en la carta de un vendedor. Un ítem del menú pertenece a solo un vendedor. Ejemplo: ItemMenu= Coca Cola.

Ítem de Pedido: producto solicitado por el cliente, compuesto por un ítem de menú y la **cantidad** deseada, como por ejemplo: ItemPedido = Coca Cola, 2 unidades

Estados de un Pedido: los pedidos pueden transitar por los siguientes estados en orden secuencial:

- **EN_CARRITO:** pedido creado por el cliente en el cual se agregan, modifican o eliminan ítems. En esta instancia el pedido no está confirmado ni pago, y por lo tanto puede ser modificado o eliminado.
- **CONFIRMADO:** estado alcanzado una vez que el sistema valida el pago exitoso del pedido. A partir de aquí, el pedido queda listo para ser procesado por el vendedor.
- **EN_ENVÍO:** el pedido fue despachado para su entrega al cliente.
- **ENTREGADO:** el ciclo de vida del pedido ha finalizado, ya que el cliente recibió el pedido.

Dirección del Cliente: la dirección registrada por el cliente es única y cumple simultáneamente la función de dirección de entrega y dirección de facturación.

Pedido en Curso: pedido cuyo estado es *CONFIRMADO* o *EN_ENVÍO*. Un pedido deja de estar en curso cuando pasa al estado *ENTREGADO*.

Carrito: instancia de un pedido en estado *EN_CARRITO*, donde el cliente puede gestionar los productos antes de confirmar la compra.

Categoría: categorías de productos existentes en el sistema.

Vendedor: hace referencia a un negocio gastronómico. No representa a una persona física individual, sino al comercio en sí.