Domicilios Hermez

Diana Marcela Gutiérrez Ramírez

Jesús Donato Silva Figueroa

Asesor

Javier Hernán Jiménez Beltrán

Universidad Nacional Abierta y a Distancia UNAD

Escuela Ciencias Básicas, Tecnología e Ingeniería ECBTI,

Tecnología en Desarrollo de Software

2025

Dedicatoria

A mi padre y a mi madre, por ser el pilar fundamental en cada paso de mi camino. Su fe inquebrantable en mí, su apoyo y amor constantes, sin condiciones han sido la luz que ha guiado mi formación y crecimiento. Gracias por preocuparse por mi educación, por enseñarme con paciencia y por impulsarme siempre hacia adelante. Les agradezco cada consejo, cada gesto, su presencia ha sido mi mayor fortaleza.

Jesús Donato Silva Figueroa

A mi familia, mi madre, mis hermanos, y en especial a mi padre, que me apoya siempre, que ha sido siempre un ejemplo de una buena persona y un honesto ciudadano, un hombre trabajador, inteligente, y que logra superar todos los obstáculos con la mejor mentalidad, gracias por inspirarme a no rendirme, a dar lo mejor de mí, me siento muy orgullosa de llamarlo papá.

Diana Marcela Gutierrez Ramírez

Agradecimientos

Expresamos nuestro más sincero agradecimiento al Ingeniero Javier Hernán Jiménez Beltrán por haber guiado con compromiso y sabiduría la realización de este proyecto, su experiencia, empatía y enfoque practico fueron fundamentales en cada etapa del proceso. Mas allá de sus valiosos consejos durante el desarrollo del proyecto. Valoramos no solo su conocimiento técnico sino también su interés desde el principio por ayudarnos.

Resumen

El proyecto nace ante la ausencia de plataformas de domicilios en Barichara, Santander. Su objetivo general es desarrollar una aplicación web que gestione integralmente el registro, seguimiento y asignación de servicios de entrega; los objetivos específicos son analizar requisitos, codificar la solución y validarla con pruebas funcionales y de rendimiento. Se aplicó benchmarking a Rappi, Uber Envíos, Tadda, Rapigo y Didi Food para extraer buenas prácticas y definir requisitos funcionales y no funcionales. La arquitectura cliente-servidor separa Astro/React en el frontend, Django REST Framework en el backend y SQLite como base de datos; la comunicación se realiza mediante API REST y JSON. Los resultados incluyen: 13 requisitos funcionales implementados (registro, login, selección de domiciliario/tarifa, estados del pedido, pago efectivo/transferencia, calificaciones, historial, diseño responsive) y 7 requisitos no funcionales. Se ejecutaron 16 pruebas unitarias con pytest y pruebas manuales end-to-end que demostraron flujo completo sin fallos. Se concluye que la solución satisface las necesidades locales, elimina intermediarios, reduce tiempos de entrega y moderniza la economía del municipio sin perder la lógica comunitaria. El código abierto bajo licencia MIT permite replicarla en otros territorios rurales.

***Palabras Clave**:* Servicios domiciliarios, arquitectura cliente-servidor, aplicación web, gestión de envíos.

Abstract

The project originated from the absence of delivery-service platforms in Barichara, Santander. Its general objective was to develop a web application capable of comprehensively managing the registration, tracking, and assignment of delivery services; specific objectives were to analyze requirements, implement software solution, and validate it through functional and performance testing. Benchmarking was conducted on Rappi, Uber Envíos, Tadda, Rapigo, and Didi Food to extract best practices and define both functional and non-functional requirements. Client–server architecture separates Astro/React in the frontend, Django REST Framework in the backend, and SQLite as the database; communication is handled through a REST API using JSON. Results include 13 implemented functional requirements (user registration, login, delivery-person/tariff selection, order status management, cash/transfer payment options, ratings, history, and responsive design) and 7 non-functional requirements. 16 tests were executed using pytest, along with manual end-to-end tests and it demonstrated a complete and error-free operational flow. It is concluded that the solution meets local needs, removes intermediaries, reduces delivery times, and modernizes the municipality's economy without disregarding its community-based logic. The open-source code under the MIT license enables replication in other rural territories.

Keywords**:** delivery services, client–server architecture, web application, delivery management.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Introducción

En el presente proyecto se va a diseñar y construir la solución para una carencia muy particular que se ha encontrado en un municipio de Colombia, en Barichara, ciudad turística, y como se sabe el turismo además de función social, contribuye al desarrollo social, económico y cultural del país y la generación de empleo(Garcia-Capdevilla, Diana Alí; Velasquez-Valencia, Alexander And Hernandez-Gil, Cristian., 2024), en donde no existe actualmente plataforma para el manejo de los domicilios ni el transporte de personas como existe en la ciudad por ejemplo Uber o Indriver, por esta razón y ante la carencia se ha pensado en una opción que ayude en la gestión de los domicilios del pueblo.

En la sociedad actual es muy importante el transporte de mercancías, todo el tiempo tenemos acceso rápido a diferentes opciones para transportar aquello que se requiere. En los municipios pequeños de Colombia, el dinero en efectivo y las transacciones en línea son las principales opciones para realizar pagos; esto es una característica muy importante para este proyecto, ya que tiene en cuenta las particularidades del municipio para ofrecer una solución realista. Las empresas pequeñas mantienen alto niveles flujo de caja (David Yepes & Diego Restrepo-Tobon, 2016), la economía en los pueblos se mueve principalmente por pequeños negocios familiares, es por esto que se ha decidido a desarrollar una web, acorde con las necesidades de las personas de esta zona.

Nuestra iniciativa surge como nuestro granito de arena para dar opciones a las personas y a los emprendedores locales para optimizar sus entregas, vamos a ver los objetivos, y todo el desarrollo de este proyecto, para demostrar como queremos apoyar nuestra comunidad.

Justificación

En el municipio de Barichara, turístico por excelencia siendo monumento nacional desde 1978 como en muchos otros municipios, no tienen presencia de plataformas de transporte como Uber o Cabify, en este pueblo para el transporte de personas se encuentra una asociación las motos de tres llantas llamadas “tuc tuc”, pero para el transporte de domicilios no existe empresa o asociación. Se hace necesario entonces para el municipio de Barichara crear una opción de comunicación entre los clientes y las personas que prestan el servicio de domicilios en el municipio con el fin de otorgar una mayor organización y registro de los domicilios.

El desarrollo de una aplicación que funcione como canal entre clientes y domiciliarios en Barichara responde a una necesidad de los habitantes y turistas, brindando a la comunidad en general múltiples ventajas:

Eliminar intermediarios: el usuario o empresa se conecta directamente con el domiciliario sin depender de una oficina o de una persona que reciba el pedido.

Reducción del tiempo de respuesta: Al reducir un tercero en el proceso de solicitud de domicilios se reduce el tiempo de espera.

Mayor comodidad para el cliente: El usuario solicita el servicio desde cualquier dispositivo, sin necesidad de conocer números telefónicos o desplazarse de su ubicación.

Mayor eficiencia para el domiciliario: Ver en tiempo real todas las solicitudes y decidir cual aceptar según su ubicación.

Reducir errores de comunicación: La aplicación web permite pormenorizar detalles relevantes para garantizar obtener lo que se desea por medio del servicio.

Mayor control por parte del Cliente: la aplicación muestra diferentes estados del domicilio permitiendo que el cliente visualice de forma clara si su servicio está en proceso o ya se entregó.

Seguimiento y calificación: tanto clientes como domiciliarios podrán calificar, cual asegura que se promueva un mejor servicio y trato entre ellos.

Acceso equitativo: Más personas pueden ofrecer sus servicios sin depender de estar afiliado a una empresa local.

Modernización del territorio: Democratizar el acceso a herramientas digitales, modernizando principalmente a Barichara que no tiene este tipo de plataformas.

Presentación del Proyecto

Planteamiento del Problema

Durante una visita a Barichara, considerado Monumento Nacional (Consejo de Monumentos Nacionales,1978), un turista intenta solicitar comida a domicilio mediante plataformas digitales. Sin embargo, ninguna de ellas opera en la zona. Ante esta limitación, acude directamente a un restaurante, el cual tampoco dispone de medios tecnológicos para gestionar entregas, evidenciando la ausencia de soluciones digitales en el área. No hay un canal de comunicación como una app o un website para enviar un domicilio, Barichara carece de plataformas tecnológicas que faciliten la gestión de servicios de transporte. Los domicilios se realizan principalmente mediante motocicletas, gestionados de forma desorganizada por negocios locales o repartidores independientes, si bien, esta modalidad es funcional, presenta dificultades en la organización y trazabilidad de las entregas y la comunicación entre cliente y proveedor. Si bien Barichara no es un municipio muy grande solo cuenta con 7.647 habitantes (DANE, 2019) la alta cantidad de turistas que acoge cada año hace necesaria una alternativa para esta carencia.

Tener opciones es importante al momento de enfrentar una necesidad, de tal forma que si una opción no funciona se pueda probar con otra, esta es la necesidad en estos momentos en el municipio de Barichara, crear una opción para sus habitantes en donde se puedan comunicar no con uno solo sino con varias personas que les puedan ayudar para el transporte de su mercancía. Posiblemente la inmensa extensión de sus veredas o la poca cantidad de carros puedan ser las razones por las cuales no existe una plataforma online para transporte. Actualmente cuando se realiza la solicitud de domicilios solo por teléfono genera varias oportunidades de mejora. Primero es necesario contar con el número del

teléfono de cada una de las personas que prestan el servicio de domicilio, con el fin de identificar cual se encuentra más cercana y disponible para realizar el envío. Segundo, todo este proceso busca optimizar el tiempo de entrega y, en consecuencia, garantizar una mayor satisfacción del consumidor final.

Se observa que es necesario una herramienta que permita a los negocios locales gestionar los domicilios de manera que ofrezca a los usuarios una experiencia organizada, accesible y confiable, una aplicación web, centrada en las necesidades específicas de este entorno para fomentar la inclusión digital en esta comunidad.

Todo lo anterior nos lleva a plantearnos, ¿podemos crear una opción, un canal diferente de comunicación entre las empresas y los encargados de los domicilios?

Objetivos

Objetivo General

Desarrollar una aplicación de software para gestionar integralmente los servicios domiciliarios permitiendo el registro, seguimiento y asignación de solicitudes dentro del territorio previamente definido por la organización.

Objetivos Específicos

Analizar los requisitos funcionales y no funcionales del sistema, verificando su alineación con las necesidades de los usuarios y los objetivos estratégicos del negocio.

Desarrollar la aplicación de software de acuerdo con los requerimientos y el lenguaje de programación definido.

Implementar la aplicación de software en un entorno controlado, realizando pruebas funcionales y de rendimiento que permitan validar su correcto funcionamiento antes de la puesta en producción.

Marco Conceptual y Teórico

Servicios Domiciliarios

Definición y Características

Los servicios domiciliarios han atraído el interés de la industria desde aproximadamente 1993, se necesitaba un método para validar la información durante el domicilio, la novedad fue estudiada y registrada en el mundo académico en 1991, específicamente se estudió el domicilio de una tienda (Ghajargar et al., 2016). Los servicios domiciliarios se podrían entender como la coordinación entre los consumidores, el vendedor(es) y una tercera parte que se encarga de la entrega, teniendo un único punto de contacto, pago y entrega para los consumidores (Duthie et al., 2023). Desde el 2001 existen estudios para identificar los factores de éxito de los domicilios y compara los nuevos servicios ofrecidos por los proveedore en Estados Unidos. Se investigó también las necesidades y los comportamientos de los clientes ante los domicilios usando la herramienta QFD, en especial para el sector de domicilios de comida, los resultados arrojaron que los clientes hacían énfasis en la seguridad de su información personal y el mecanismo de intercambio, también la velocidad del domicilio y la rapidez en la que la empresa respondía ante cualquier inconveniente (Ghajargar et al., 2016). En Colombia, el trabajo por aplicaciones o plataformas digitales empezó a comienzos del 2010, cuando empresas como Domicilios.com y Uber iniciaron el proceso de concertar con empresas para brindar el servicio de domicilios o repartidores (Elicabide et al., 2024). En este modelo, el valor agregado radica en la conveniencia, la rapidez y la accesibilidad.

Un servicio de domicilio es un servicio destinado a entregar un envío a un destino; es similar al servicio postal. En comparación con este último, existe una diferencia significativa en la

velocidad del transporte, que suele ser más rápido, el método de carga y el precio. Todo lo anterior permite una considerable flexibilidad ante cambios en la dirección de entrega. El servicio de domicilios consiste en la recogida inmediata del envío del remitente y su posterior entrega al destinatario (Semerádová & Weinlich, 2022). Las aplicaciones de domicilios se han nutrido de la adaptación y la capacidad de innovación, y se adaptan a cada contexto de cada país, redefiniendo lo que es un trabajo en la era digital, generando un vacío legal. En estas plataformas, el algoritmo se encarga de organizar el trabajo, donde se monitorea, y se toman decisiones basadas en datos (Elicabide et al., 2024).

En una encuesta (Ghajargar et al., 2016) se preguntó a los participantes cuáles factores caracterizan un buen servicio a domicilio. El orden de importancia fue: rapidez en la entrega, servicio de calidad, posibilidad de escoger el tiempo y el lugar de entrega, precio, opción de seleccionar el empaque, eco-sostenibilidad del servicio y seguimiento en tiempo real del domicilio.

En este mismo estudio, al probar otras opciones, las personas también dieron gran importancia a que tan confiable es el domiciliario, y la relación de confianza con la empresa repartidora, también luego prefirieron un servicio más lento pero económico. Así que no está muy claro si al final, quieren un servicio rápido o están dispuestos a esperar más, pero con mejores tarifas.

Importancia en el Contexto Actual: Se puede ver ejemplos como los de Uber, que continua siendo una crónica de nuestros tiempos, del poder transformador de la tecnología, de lo perecedero del empleo a largo plazo, y de la oportunidad de convertir la improvisación y el ingenio en algo positivo (Lashisnky, 2018) . Se siguió un incremento en la demanda de servicio a domicilio para la comida, creando la necesidad de leyes y regulaciones para estos servicios y

también para los bienes que transportan (Duthie et al., 2023) ., Actualmente los servicios domiciliarios en Colombia no cuentan con un marco regulatorio, no tiene una protección adecuada para los trabajadores de plataformas digitales. En otros lugares como España o California donde ya hay leyes para estos trabajadores y las plataformas, Colombia aún necesita un marco legal que equilibre la innovación y los derechos de los trabajadores (Elicabide et al., 2024). Los servicios domiciliarios constituyen un elemento central de las economías urbanas y rurales por varias razones: comodidad para el usuario, impulso al pequeño comercio local en la industria y turismo, así como inclusión digital y logística.

En pueblos pequeños, donde los canales digitales aún son limitados, el domicilio pagado en efectivo cobra relevancia, pues responde a realidades como la baja bancarización y la preferencia por transacciones tradicionales, y se crean relaciones complejas entre los actores involucrados en el servicio a domicilio de comida (Shroff et al., 2022).

Evolución con el uso de Tecnologías Digitales: Las raíces de la transformación digital pueden rastrearse hasta la creación de los computadores y la digitalización de los procesos, en las etapas tempranas esto implicaba pasar de análogo a digital. El presente fenómeno sin embargo se puede atribuir a Alemania, que se propuso una estrategia altamente tecnológica, emergiendo as la cuarta revolución industrial y la evolución de la economía digital (Omol, 2023). La globalización y el rápido auge de nueva tecnología han hecho que diversas opciones lleguen a las personas, entre esas surge a principios de este siglo XXI una creación que ofrece una alternativa a los taxis, que es Uber, plataformas digitales que progresivamente remplazan a intermediarios tradicionales en la economía, introduciendo nuevos modelos de negocios (Weng, S. Goo, Zailani.S & Iranmanesh. M, 2017). Las plataformas gestionan el reparto por medio de algoritmos, es decir, conjuntos

automatizados de instrucciones que transforman datos en resultados deseados, la gestión algorítmica implica una multiplicidad de funciones productivas (Ballestrin, 2024).

Antes, la gestión se hacía por vía telefónica o directamente en el establecimiento; hoy, las aplicaciones móviles y páginas web permiten centralizar la oferta de diferentes comercios, facilitar el rastreo de pedidos y tiempos de entrega, recolectar datos de consumo útiles para la mejora de la experiencia del usuario y ampliar la visibilidad de negocios pequeños en entornos digitales.

En esta era digital los usuarios demandan una interacción personalizada, continua y consistente durante el proceso de domicilio (Omol, 2023), y mediante un análisis de los datos se pueden llenar estas expectativas, creando interfaces intuitivas, tiempos de respuesta rápidos, dando como resultado la satisfacción del cliente. Sin embargo, en territorios con menor acceso a internet o menor adopción de medios de pago electrónicos, la adaptación tecnológica requiere un modelo híbrido: plataformas digitales sencillas que mantengan la opción de pago en efectivo. Este enfoque es clave para pequeñas comunidades, pues asegura la inclusión y la sostenibilidad del sistema de domicilios.

Gestión de Solicitudes en Servicios Domiciliarios

En los servicios domiciliarios la gestión corresponde al conjunto de procesos que permiten organizar, registrar y dar respuesta a las demandas de los usuarios en tiempo real. De acuerdo con estudios en logística y comercio electrónico (Chopra, S. & Meindl, P., 2019), esta gestión es esencial dentro de la cadena de suministro de última milla, el ultimo trayecto antes que el producto llegue al destino final, ya que enlaza la oferta del proveedor con la demanda del consumidor mediante flujos de información y distribución eficientes. Existen varios estudios, como, por ejemplo: Zhou, L., Wang, X., Ni, L., & Lin, Y. (2016) Location-Routing Problem with

Simultaneous Home Delivery and Customer's Pickup for City Distribution of Online Shopping Purchases, referentes a la optimización en la gestión de los domicilios donde analizan paso a paso y con modelos las fases del pedido, la recogida y la entrega y como seleccionar rutas optimas que minimicen los costos, basados en correctas asignaciones y la evaluación correcta de la eficiencia de los algoritmos, mostrando ventajas a la hora de reducir el costo de la operación y los vehículos usados, mejorando la red de distribución en el área.

Registro de Pedidos: La primera etapa del proceso es el registro de pedidos, el cual implica la entrada de la solicitud del cliente a través de diferentes canales: llamadas telefónicas, mensajería instantánea, aplicaciones móviles o plataformas web. La digitalización de este proceso garantiza mayor precisión en la información, reducción de errores en la transcripción manual y generación de bases de datos útiles para su posterior análisis (Turban et al., 2018). Tener un registro de pedido, permite guardar un historial de los mismos y hace fácil dar seguimiento y manejar todo el proceso.

Se debe planear efectivamente este primer paso, crear y controlar el flujo del envío, diseñando una opción que pueda captar todos los requerimientos del cliente. En esta sección el algoritmo capta cual es la ruta y vehículo con tiempo y más cercano al punto de recogida del domicilio.

Seguimiento de Solicitudes: El seguimiento consiste en monitorear el estado del servicio desde su aceptación hasta la entrega final. Esto incluye etapas como confirmación del pedido, preparación, despacho y entrega al cliente. En este despacho del pedido, se otorga especial relevancia al responsable de efectuar la entrega al cliente final, dado que dicha fase constituye el último eslabón del proceso operativo. Esta etapa reviste especial importancia, ya que representa un

punto de contacto directo con el cliente, cuya experiencia de compra puede verse significativamente influenciada por la calidad de dicha interacción.

En poblaciones pequeñas, el seguimiento puede simplificarse mediante mecanismos básicos como notificaciones en la aplicación, mensajes de texto o mensajes a través de redes como WhatsApp. Hacer el seguimiento de un pedido, saber la ubicación del domiciliario es una de las más útiles aplicaciones de la tecnología en la industria del transporte, permite manejar y optimizar los movimientos de los vehículos e incrementar la seguridad y la eficiencia. Además otro de los resultados es dar a los clientes información en tiempo real del progreso y el tiempo estimado para su pedido (Prado et al., 2010). También es importante destacar que esta tecnología para dar seguimiento a los vehículos está disponible no solo para grandes empresas sino para cualquiera, proporciona herramientas para coordinar y sacar estrategias para el mejoramiento.

Asignación de Entregas: La asignación de entregas consiste en distribuir los pedidos entre los domiciliarios disponibles, buscando optimizar tiempos, costos y recursos. Con respecto a la asignación una opción es asignar el trabajo a los vehículos cercanos aleatoriamente, así esta se basa en la ubicación del cliente. Otra opción es asignar el domicilio al vehículo más cercano (Benarbia & Kyamakya, 2022).

Se debe considerar: la ubicación geográfica del cliente, la urgencia del pedido, la disponibilidad del repartidor, y el tiempo estimado de entrega. En aplicaciones modernas, esta tarea suele automatizarse mediante algoritmos de optimización.

Aplicaciones Web y Móviles

Conceptos y diferencias: las aplicaciones web son una aplicación cliente/servidor que usa un navegador web como su programa cliente, interactúa conectándose con los servidores por

internet. Una aplicación web presenta un contenido dinámico adaptado y basado en los parámetros requeridos (Shklar & Rosen, 2003). Mientras que las aplicaciones móviles son programas diseñados para los sistemas operativos móviles, el usuario debe descargar e instalar la aplicación y ejecutarla en su aparato móvil antes de usarlas (Cuello & Vittone, 2013).

Beneficios en la Prestación de Servicios: la extensión y la complejidad del transporte en el mundo lo hace una excelente área dentro de la logística para el uso de las tecnologías de la información. El uso de apps o website empezó con la determinación de rutas de transporte, que minimiza los costos y cumple con las restricciones de la entrega (Prado et al., 2010).

Ejemplos en el Sector Domiciliario: En Colombia se encuentran Uber eats, Rappi, Didi Food, Glovo y Tada. La logística y transporte están experimentando un rápido crecimiento, que esta soportado por una gran evolución en la innovación, y con esto nuevas empresas de reparto.

Geolocalización en Aplicaciones

Principios Básicos de GPS y Mapas Digitales: el GPS funciona con una red de satélites que transmiten señales de la posición. El sistema provee acertada y continua información de la posición tridimensional y la velocidad. Existen diverson satélites siviendo diferentes gobiernos, como el GPS para USA, el GNSS para Europa, el GLONASS para rusia y el chino BNTS. El acceso a toda esta información de la posición y todos los datos relacionados(sensores auxiliares) permiten crear un mapa digital que puede ser usado para guiar para vehículos en tierra, aviación, marítima e incluso espacial (Kaplan & Hegarty, 2006).

Aplicaciones en Logística de Última Milla: el uso de los celulares y las apps, hizo posible que surgiera un nuevo tipo de servicio denominado “crowd delivery”, esto permite a los particulares usar su propio vehículo para entregar paquetes a cambio de una tarifa que el conductor

recibe, Cuando un cliente realiza el pedido, un conductor recibe una notificación y se compromete a entregarlo según las condiciones del cliente (Ghajargar et al., 2016). Con esta clase de servicios los particulares pueden obtener ingresos adicionales y la calidad del servicio aumenta, ya que los clientes reciben la entrega según sus condiciones. Ejemplo de esto son Fryat, Goshare, roadie, dispatch, veo, amazon flex, doordash, instacart, en Colombia existe Rappi, Glovo, Uber eats.

Ventajas para el Usuario y la Empresa: las ventajas para las empresas de los sistemas de posicionamiento global es que se puede rastrear la ubicación de los vehículos en tiempo real, esto en países como Colombia también significa seguridad, y la notificación del tiempo de llegada (Chopra, S. & Meindl, P., 2019). Con esta información en tiempo real, permite la optimización dinámica de las rutas y las entregas, haciendo el proceso más confiable (Prado et al., 2010).

Experiencia del Usuario (UX/UI)

Concepto de Usabilidad: según la norma ISO 9241-11, (2018) es el grado en que un producto puede ser utilizado por usuarios específicos para alcanzar objetivos concretos (*ISO 9241-11:2018(en)*, 2018). Se debe establecer los requerimientos y restricciones del sistema, tener en cuenta el uso regular, aprendizaje, uso por parte de personas con diferentes capacidades, minimizar errores, fácil mantenimiento.

Diseño Centrado en el Usuario: significa diseñar para las personas, tener en cuenta que todo en el proceso de crear se centra en el usuario. Se debe conocer al usuario, que es lo que quiere, además también el contexto en que va a utilizar el software, todo lo anterior garantiza que los objetivos están claros y así el producto funcionara. Esta es la mejor manera, entender y satisfacer al cliente (Pedraza-Gutiérrez et al., 2023).

Relevancia en la Satisfacción del Consumidor Final: es un factor fundamental para la adopción y el éxito del sistema, buscar que el sistema tenga usabilidad, que sea practico (Pedraza-Gutiérrez et al., 2023). Al comprender a los usuarios, sus necesidades y requerimientos, y a estos, aplicar principios de ergonomía, usabilidad y el factor humano, con todo esto se garantiza que el sistema sea exitoso.

Calidad del Servicio

Concepto y Dimensiones: se refiere al resultado del sistema, a la percepción del cliente y su satisfacción, está basado en diversos factores, sin embargo, se sabe que el usuario puede estar afectado por su condición psicológica y física (Parasuraman et al., 1988). Por lo tanto, puede haber un grado de discrepancia entre las expectativas del cliente y su percepción al momento de evaluar. Incorpora múltiplos atributos o dimensiones que son tangibilidad, confiabilidad, capacidad de respuesta, seguridad y empatía.

Indicadores de Medición Aplicables: se puede aplicar el CSAT o satisfacción del usuario, el NPS o puntuación neta del promotor, mide cuanto recomendaría el producto, y finalmente el CES o puntuación del esfuerzo del cliente.

Marco Teórico

Modelos de Desarrollo de Software

Metodologías Tradicionales (Cascada): este modelo es uno de los más antiguos y estructurados, se caracteriza por el proceso secuencial y lineal, en el cual cada fase debe completarse para poder pasar a la siguiente. Este modelo toma lo fundamental de las actividades como especificidad, desarrollo validación y evolución, el resultado de cada fase en este modelo es

uno o más documentos que son aprobados. Las fases son: análisis de requisitos, diseño, implementación, pruebas, despliegue e integración, y mantenimiento (Sommerville, 2016).

Metodología Design Thinking: Thinking es una metodología de innovación centrada en comprender profundamente a los usuarios para resolver problemas de manera creativa. Se caracteriza por ser iterativa, colaborativa y orientada a la experimentación. Su enfoque combina empatía, ideación y prueba continua de soluciones, permitiendo desarrollar productos o servicios más ajustados a las necesidades reales de las personas.

El proceso suele incluir cinco etapas: empatizar, definir, idear, prototipar y evaluar, las cuales pueden repetirse según los hallazgos y la retroalimentación(Plattner, H., Meinel, C., & Leifer, L., 2011).

KANBAN: este método se centra en la visualización del flujo de trabajo a través de tableros con columnas donde se visualiza el flujo de trabajo, se destaca por su simplicidad y adaptabilidad. Este método tiene un doble objetivo, informas las fases del proceso de desarrollo en el momento presente y agilizar el proceso de liberación de producción. Informa a las etapas anteriores del proceso de desarrollo sobre la demanda que ocurre en la etapa final, entonces actúa como enlace entre las etapas del proceso y la parte final. Este método es un sistema de control visual y permite tomar decisiones sobre las necesidades de desarrollo si necesidad de preparar ni consultar informes (Contador et al., 2005).

Requisitos de Software

Los requisitos para un sistema describen los servicios que el sistema debería proveer y las restricciones en su operabilidad (Sommerville, 2016). Estos requisitos reflejan las necesidades de

los usuarios del sistema, hay una distinción entre los requisitos de usuario y los del sistema. Diferentes clases de requisitos son necesarios para comunicar información de un sistema a diferentes tipos de lectores. Es necesario escribir estos requisitos con diferentes niveles de detalles porque personas diferentes van a leerlos y deben entender y podrían usarlo de diferentes formas para desarrollar el sistema.

Funcionales: se trata de la declaración de los servicios que el sistema debe prestar o cumplir, como debe reaccionar a entradas específicas y también en que debe hacer en situaciones particulares. Algunas veces también pueden indicar expresamente que no debe hacer el sistema (Sommerville, 2016).

No funcionales: estos requisitos muestran las restricciones en los servicios o las funciones que debe tener el sistema. Suele aplicarse a todo el sistema, no solo a una característica o parte de él. Estos influyen en la experiencia del usuario y en la operatividad del sistema.

Ejemplos Aplicados al Sistema: ejemplo de funcional: el sistema debe permitir el registro de los domiciliarios y también de los clientes, el sistema debe ser capaz de mostrar el valor del domicilio. Ejemplos de no funcionales: el sistema debe ser accesible para los navegadores comunes, el sistema debe estar operativo 97% del día.

Arquitectura de Software para Aplicaciones en Línea

Muestra cómo debemos entender y como el sistema debería estar organizado y diseñando la estructura general del sistema, y también como se dividen y se relacionan los componentes para cumplir con los requisitos solicitados (Sommerville, 2016). El diseño arquitectónico es el primer paso en el proceso de desarrollo de software. Revela como el sistema se organiza en componentes que se comunican.

Modelo Cliente-Servidor: esta arquitectura presenta al sistema como un conjunto de servicios cada uno de los cuales es proporcionado por un servidor independiente. Los clientes son usuarios de estos servicios y acceden a los servidores para usarlos. La principal ventaja es que al distribuir el sistema en diferentes servidores, una funcionalidad puede estar disponible para todos los clientes y no necesita implementarse en todos los servicios. Una desventaja es que el rendimiento es poco predecible ya que depende del network y también del sistema.

Microservicios y API REST: los microservicios son un diseño del sistema de característica tradicional, básicamente divide la aplicación en módulos pequeños, independientes y especializados que se comunican entre si a través de APIs. **REST** significa transferencia de estado representacional, es un estilo de arquitectura y está basado en la transferencia de representaciones de recursos de un servidor a un cliente (Sommerville, 2016). Es un método sencillo. Así un API REST es una API que sigue los principios REST.

Escalabilidad y Mantenimiento: en la escalabilidad: los sistemas distribuidos son escalables ya que sus capacidades pueden incrementarse al añadir nuevos recursos para satisfacer las nuevas demandas del sistema, así que el sistema se adapta al crecimiento del mismo. En realidad, la red que conecta las computadoras individuales del sistema puede limitar esta escalabilidad.

En cuanto al mantenimiento, el hecho de que este constituida en módulos permite realizar cambios sin modificar todo el sistema reduciendo el impacto de fallos también ya que se limita a cierta parte, mejora la adaptabilidad del software a largo plazo permaneciendo útil.

Para el mantenimiento hay 3 clases (Sommerville, 2016) de mantenimiento de software:

Reparación de fallos para corregir errores y vulnerabilidades, los errores en el código suelen ser relativamente económicos de corregir, los errores de diseño son más costosos, los errores de requisitos son más costosos de reparar, ya que puede ser necesario un rediseño exhaustivo del sistema.

Adaptación del software a nuevas plataformas y entornos, este tipo de mantenimiento es necesario cuando cambia algo en el entorno del sistema, por ejemplo, el hardware.

Adición de funcionalidad, cuando los requisitos del sistema cambian una respuesta a cambio organizacionales o empresariales. Los cambios requeridos en este tipo son mucho mayores que para otros tipos de mantenimiento.

Bases de Datos para Sistemas Transaccionales.

Una transacción es una secuencia de operaciones que se ejecutan como una sola unidad de trabajo, garantizando la integridad de los datos (Silberschatz et al., 2011). Un sistema transaccional es la plataforma de software diseñada para gestionar y procesar transacciones, actualmente son clave en las empresas, ya que ayudan a controlar y administrar las operaciones financieros (Obrien y Marakas, 2011). Entonces las bases de datos para estos sistemas permiten almacenar información de transacciones, así como garantizar la integridad y la precisión de los datos en tiempo real. Los sistemas transaccionales pueden integrarse con otros sistemas como de contabilidad, CRM, RR HH, esta integración da una visión completa del sistema.

Importancia de la Persistencia de Datos: la persistencia es la capacidad de un sistema para guardar los datos, con o sin la ejecución del programa. El almacenamiento persistente de objetos de programa y estructuras de datos es algo fundamental en las bases de datos, los sistema de bases de datos tradicionales solían tener un problema de desajuste llamado impedancia, ya que había una interrupción entre la base de datos y el lenguaje de programación, esto se presentaba porque eran incompatibles en la estructura (Elmasri & Navathe, 2016). Es importante la

persistencia porque almacenan los datos de manera confiable, existiendo también integridad y consistencia en la información aun con fallos en el sistema.

Bases de Datos Relacionales vs. no Relacionales: las bases de datos relacionales o SQL, representa la base de datos como una colección de relaciones. Informalmente cada relación parece una tabla de valores o a un archivo plano de registros. Utilizan el lenguaje SQL para los datos. Estas bases ofrecen integridad para transacciones complejas. En el modelo relacional una fila se denomina tupla, un encabezado de columna se denomina atributo y la tablase denomina relación (Elmasri & Navathe, 2016).

Las bases de datos no relacionales o NoSQL, el nombre NoSQL intenta transmitir que muchas aplicaciones necesitan sistemas distintos a los sistemas SQL relacionales para poder complementar la gestión de datos. Estos NoSQL son bases de datos distribuidas, centradas en el almacenamiento de datos semiestructurados, el alto rendimiento, la alta disponibilidad y escalabilidad, no se concentra en la consistencia inmediata de los datos. Para organizaciones que almacenan gran cantidad de datos, para esto un sistema SQL no es adecuado ya que no podría ser muy restrictivo.

Aplicación en la Trazabilidad de Pedidos: la trazabilidad es poder seguir todo el ciclo de una transacción, en este caso sería seguir todo el proceso de pedido. Una base de datos relacional puede registrar los datos como clientes, pedidos, domiciliarios.

Teorías de Satisfacción del Cliente

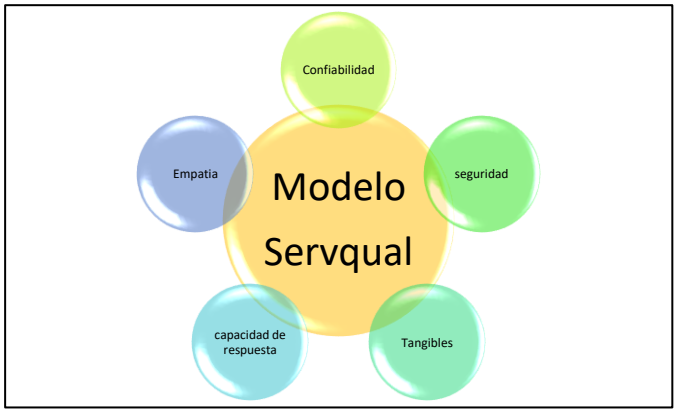
La satisfacción ese un estado psicológico que resulta cuando la emoción que rodea a las expectativas no confirmadas se combina con los sentimientos previos del consumidor sobre la experiencia de consumo. Existe una distinción entre calidad del servicio y satisfacción, la calidad

del servicio es un juicio global, mientras la satisfacción se relaciona con una transacción especifica. Así, una persona puede estar satisfecha con el servicio, pero no cree en la calidad de los productos (Parasuraman et al., 1988). Sin embargo, la satisfacción se terminará transformando en la actitud general hacia la compra de productos.

Modelos como SERVQUAL: La calidad percibida, medida mediante el modelo Servqual, representa el juicio que realiza el consumidor sobre la excelencia de una entidad. Esta noción difiere de la calidad objetiva, ya que surge de la comparación entre las expectativas del usuario y sus percepciones de rendimiento. El modelo Servqual ha sido ampliamente adoptado para evaluar entornos digitales, permitiendo medir la calidad percibida de aplicaciones web. Evidencia de lo anterior mencionado se puede observar en la Figura 1, que ilustra las dimensiones fundamentales de este modelo.

Figura 1

Modelo Servqual



Nota. Partes de modelo Servqual. *Obtenido de.* Autores

Variables que Influyen en la Percepción de Calidad: Con el modelo Servqual creado en 19884 se identifican las variables: tangibilidad, fiabilidad, capacidad de respuesta, seguridad y

empatía cuantificar la calidad que el cliente percibe de un servicio no es fácil, se requieren herramientas que ayuden a las empresas a comprender que es valioso para los clientes. Con el modelo Servqual hay un cuestionario de 22 preguntas (Parasuraman et al., 1988). Para el servicio a domicilio vemos las necesidades personales, el servicio esperado, el servicio percibido, experiencias pasadas.

Seguridad Informática en Aplicaciones Web

La ciberseguridad son las medidas que buscan proteger las redes informáticas y los sistemas de amenazas y ataques. Se hace necesario la adopción de mecanismos para enfrentar los problemas de seguridad en aplicaciones web, la etapa de despliegue del software es momento decisivo ya que se ejecutan actividades que permiten la disponibilidad de la aplicación web para su uso (Yeja & Rubier, 2016). La seguridad de la información protege los activos informáticos, que estos puedan conservar su confidencialidad, integridad y disponibilidad, independientemente si están almacenados, procesados o transmitidos.

Protección de Datos Personales y Financieros: las personas tienen el derecho fundamental a controlar su información, en Colombia el marco legal para los datos personales son la Ley 1581 de 2012 y el Decreto 1377 de 2013. Los datos personales son la información asociada a una persona y estos datos permiten la identificación de la persona (Jimenez et al., 2023). Para la protección de datos financieros se tiene el Habeas Data que es un derecho a conocer, actualizar y rectificar la información financiera recopilada sobre ellos.

Normativas Aplicables: en Colombia el Habeas Data, ISO/IEC 27001, Ley 1581 de 2012 y el Decreto 1377 de 2013, ley 1978 de 2019, Decreto 1008/2018.

Logística y Última Milla

La logística es entregar los bienes o servicios correctos en el lugar y el tiempo acordado y bajo las condiciones determinadas, también hace parte el almacenamiento, distribución y transporte, inventario, y los sistemas de gestión e información. Con todo esto la logística busca asegurar un menor costo operativo, suministrar oportunamente los productos, y ser un factor competitivo antes otras empresas. La logística es un método de dirección y gestión.

La última milla es una de las etapas más costosa de toda la cadena de logística, es crucial para el éxito de la cadena de suministro, existen numerosas opciones de entregas.

Conceptos Básicos de Distribución: la distribución se refiere a cada fase que se debe pasar para trasladar y almacenar un producto desde que sale del productor hasta que llega al cliente. La distribución se hace en las materias primas y en el producto terminado, una buena distribución afecta el costo de la cadena de suministro (Chopra, S. & Meindl, P., 2019). Para evaluar la distribución se debe tener en cuenta dos dimensiones: el valor que se le da al cliente y el costo de cubrir las necesidades del cliente. Se deben tener en cuenta también factores como tiempo de respuesta, variedad de productos, disponibilidad del producto, experiencia del cliente, tiempo al mercado, trazabilidad del pedido y facilidad para devolverlo.

Importancia en servicios de entrega: prestar atención a esta última etapa de la cadena de suministro que es cuando el cliente recibe el producto o el servicio ayudara a reducir los costos y esto aumenta la competitividad, además de impactar positivamente en la satisfacción del cliente. En un municipio pequeño, la coordinación es clave, ya que las distancias son más cortas.

Estrategias de Optimización de Tiempos: trazabilidad del pedido con el uso de GPS too esto en tiempo real permitiendo a las partes involucradas tener acceso, crear algoritmos que permitan asignar el pedido al domiciliario más cercano y que esté disponible.

Antecedentes de Aplicaciones Similares

En el siglo XXI, un antecedente se podría nombrar a Just-Eat creado en el 2001 en Dinamarca, para distribuir la comida de los restaurantes. Y en 2011 se empezó a expandir como takeaway.com. Y ha continuado creciendo con presencia en 17 países (Just eat, s. f.).

Rappi: comienza en 2015 en Bogotá, al principio ofrecía solo domicilios de comida, y luego añadió muchas otras opciones como farmacias, supermercados, pagos. La plataforma virtual mediante aliados comerciales ofrece y comercializan productos o servicios. Ahora Rappi tiene presencia en México, Brasil, Perú, Ecuador, Uruguay, Chile y Argentina. En su mayoría se transportan en bicicletas o motos para transitar más rápidamente (Gabriel Marcelo Duran Sanchez, Laura Marulanda Grisales, 2024).

Uber Eats: lanzado en 2014 en USA es una adición a la plataforma de Uber, de entrega de comida a domicilio. Uber eats se crea para lograr mayor crecimiento y expansión para recibir domicilios de diferentes comercios, se encuentra en 45 países, sin embargo, Colombia no es uno de ellos, dejo de estar disponible en octubre de 2020 tal y como aparece en la página web de la empresa.

Domicilios.com: Fundada en 2007 en Colombia fue pionera en el país en ofrecer un portal de pedidos en línea. Su objetivo fue digitalizar los perdidos a restaurantes locales y ofrecer una alternativa tecnológica ante el servicio telefónico tradicional. El éxito de esta plataforma creo un mercado para otros negocios similares. Llego a tener presencia en varios países, el dueño era

Delivery Hero, una empresa alemana. Se fusiono con ifood y dejó de funcionar (Periodico el colombiano, 2019). Todo lo anterior lo podemos ver condensado en la Tabla 1.

Tabla 1

Aplicaciones Similares en Colombia

Nombre	Año Creación	Lugar Creación	Activa
Domicilios.com	2007	Colombia	No activa
Uber Eats	2014 (uber en 2009)	Estados Unidos	Activa (sin cobertura en Colombia)
Rappi	2015	Colombia	Activa
Didi Food	2012	China	Activa
Pedidos Ya	2009	Uruguay	No activa

Nota. Esta tabla muestra las diferentes aplicaciones que ofrecen un servicio de transporte de alimentos en Colombia. *Obtenido de.* Autores

Diseño Metodológico

Enfoque y Tipo de Estudio

Con este proyecto se ha considerado un enfoque cualitativo, se busca entender las dinámicas, necesidades y características de los domicilios en este entorno de un municipio pequeño. Se realizó un análisis comparativo mediante la técnica de benchmarking, se identificaron referentes operacionales para orientar el diseño de esta solución tecnológica contextualizada en el municipio de Barichara.

De acuerdo con el alcance el estudio realizado es descriptivo ya que se enfoca en las características, procesos y practicas actuales que se hacen en Barichara, pero revisando las funcionalidades de plataformas similares que prestan servicios en otros municipios del país.

En cuanto al control de variables este es un estudio no experimental, ya que no se manipulan intencionadamente las condiciones del entorno para realizar el análisis, solo se observan tal y como son. En cuanto al momento de la medición el estudio es transversal pues se hace solamente en un momento especifico, donde se compara y se contextualiza las variables que se necesitan para desarrollar la aplicación web.

Metodología Design Thinking

Design Thinking es una metodología de innovación centrada en las personas. Su objetivo es entender profundamente las necesidades de los usuarios y generar soluciones creativas a problemas complejos. Se basa en un enfoque iterativo, colaborativo y orientado a la experimentación.

El proceso de Design Thinking se divide en cases fases: empatizar, para comprender a los usuarios, sus necesidades y contexto; definir, con el fin de sintetizar la información y plantear un problema claro y significativo; idear, con el propósito de generar la mayor cantidad de ideas y alternativas de solución; prototipar, a través de la creación de representaciones simples y rápidas de las ideas seleccionadas; y evaluar, al probar los prototipos con usuarios y obtener retroalimentación que permita mejorar la solución.El valor de esta metodología radica en su capacidad de combinar creatividad, comprensión humana y pensamiento estratégico, permitiendo desarrollar soluciones innovadoras y efectivas.

En la tabla 2 se detalla la correspondencia entre los objetivos del proyecto y las etapas de la metodología aplicada.

Tabla 2

Relación entre Objetivos y Etapas del Método

Objetivo	Metodología	Relación	Fase del Design Thinking	Descripción de la Articulación con Design Thinking
Analizar los requisitos funcionales y no funcionales del sistema, verificando	Análisis (Benchmarking)	Con un estudio cualitativo se identifican y analizan las características	**Empatizar / Definir**	El análisis competitivo y la recolección de requisitos permiten comprender a los

Objetivo	Metodología	Relación	Fase del Design Thinking	Descripción de la Articulación con Design Thinking
su alineación con las necesidades de los usuarios y los objetivos estratégicos del negocio.		funcionales y no funcionales para deducir los criterios de diseño útiles para Barichara, permitiendo crear los requisitos.		usuarios, su contexto y sus necesidades. A partir de este análisis se formula el problema y se definen criterios claros para orientar las soluciones.
Desarrollar la aplicación de software de acuerdo con los requerimientos y el lenguaje de programación definido.	Etapa de diseño y desarrollo	Con base en la etapa anterior, se empieza a estructurar y aplicar lo analizado en el código de la aplicación web, con los lenguajes seleccionados.	**Idear / Prototipar**	Se diseñan y construyen las primeras versiones funcionales del sistema, convirtiendo las ideas en soluciones tangibles (prototipos funcionales)

Objetivo	Metodología	Relación	Fase del Design Thinking	Descripción de la Articulación con Design Thinking
				alineadas con los requisitos definidos.
Implementar la aplicación de software en un entorno controlado, realizando pruebas funcionales y de rendimiento que permitan validar su correcto funcionamiento antes de la puesta en producción.	Etapa de pruebas	Se despliega la implementación de la aplicación y se aplican técnicas para evaluar rendimiento y funcionalidades, comparando lo obtenido con lo proyectado mediante benchmarking.	**Probar (Test)**	Se valida el prototipo con pruebas estructuradas, verificando que cumpla las expectativas del usuario y los criterios de calidad. La retroalimentación obtenida orienta ajustes y mejoras antes de su liberación final.

Nota. Esta tabla muestra la relación entre objetivos y etapas de la metodología. *Obtenido de.* Autores

Variables

En este proyecto las variables no son objeto de manipulación sino de observación y análisis para el desarrollo de la aplicación web.

Variables Independientes: El diseño y desarrollo de la aplicación web: acá se tiene en cuenta el componente tecnológico que es el resultado del análisis realizado como primer paso. Se incluye los diferentes lenguajes de programación, la arquitectura del sistema y la interfaz de diseño.

Variables Dependientes:

- Cumplimiento con las necesidades de los usuarios, todos los requisitos funcionales y no funcionales encontrados, que se pueda aplicar al caso específico de Barichara.
- Funcionamiento del sistema, eficiencia en los procesos, estabilidad, velocidad de respuesta y capacidad de adaptación al público objetivo.
- Satisfacción del usuario, como experimenta la interfaz, navegación intuitiva y la satisfacción a nivel general mientras usa la aplicación web.

Instrumentos para el Desarrollo del Benchmarking

El benchmarking inicialmente pensado como herramienta de gestión empresarial, ha demostrado ser útil en el ámbito del desarrollo de software, sobre todo siendo muy importante en el levantamiento de los requerimiento funcionales y no funcionales (Ore Quiroz, H., Aldana Juárez, W., Salazar Sandoval, C., & Pantoja-Tirado, L., 2021). Esta

técnica permite realizar una comparación entre las tecnólogas existentes y poder ver que es relevante para el proyecto y que es usable, mejorable y novedoso en el proyecto. Así, se facilita la alineación de los requisitos del nuevo sistema con parámetros ya conocidos y buenas prácticas en este sector (Jiménez Beltrán, J. H., 2015). Los instrumentos son: 1. Instrumento para elección de aplicaciones de software relacionados, 2. Instrumento para estudio de las aplicaciones de aplicaciones de software elegidas, 3. Instrumento de Elicitación de Requerimientos y 4. Instrumento para Consolidar la Propuesta de innovación. A continuación de desarrolla cada uno de ellos.

1. Instrumento para elección de aplicaciones de software relacionados

La tabla 3 presenta una evaluación de la aplicación Rappi basada en nueve parámetros de análisis, que abarcan desde cobertura de funciones principales hasta disponibilidad de soporte técnico y documentación.

Tabla 3

Características de Interes de la Aplicación de Rappi

Software para Evaluar				
Nombre de la Aplicación	Parámetro de Evaluación	Valor Cualitativo Total, Parcial, Nulo	Escala Cuantitativo (1-10)	URL

Rappi	Cobertura de funciones principales (registro, asignación, seguimiento, notificaciones)	Total	10	www.rapppi.com Rappi App
	Opciones adicionales (geolocalización, historial, chat en vivo, cancelaciones).	Total (permite seguimiento GPS)	10	https://www.rappi.com.co/account/orders
	Flexibilidad para distintos tipos de servicio domiciliario.	Nulo (no permite cambiar un destino, pero lo justifica)	3	www.rapppi.com
	Facilidad de uso (interfaz intuitiva, accesibilidad).	Total	10	www.rapppi.com
	Experiencia del usuario (fluidez en navegación, pasos mínimos para realizar acciones).	parcial	10	www.rapppi.com

Diseño visual y adaptabilidad a distintos dispositivos (responsive, móvil, escritorio).	Total	10	www.rapppi.com
Disponibilidad de soporte técnico (canales, tiempo de respuesta).	Total	10	www.rapppi.com
Documentación y manuales de usuario.	Total	10	www.rapppi.com
	Total	73	

Nota. Esta tabla contiene la evaluación de características de interés para el proyecto de la aplicación Rappi. *Obtenido de.* Autores

La tabla 4 muestra la evaluación del servicio Uber Envíos mediante los mismos ocho parámetros de análisis, implementando un sistema dual de valoración cualitativa.

Tabla 4

Características de Interés de la Aplicación de Uber.

Software para Evaluar				
Nombre de la	Parámetro de Evaluación	Valor Cualitativo Total, Parcial, Nulo	Escala Cuantitat	URL

Aplicación			ivo (1-10)	
Uber Envios (parte del servicio de Uber)	Cobertura de funciones principales (registro, asignación, seguimiento, notificaciones)	Total (este servicio es solo envíos, recoge en un lugar y deja en otro)	10	www.uber.com/co/es Uber app
	Opciones adicionales (geolocalización, historial, chat en vivo, cancelaciones).	Total (En algunos casos puede haber un cobro si cancelo muy tarde)	10	www.uber.com/co/es Uber app
	Flexibilidad para distintos tipos de servicio domiciliario.	Parcial (No permite para domicilios de comida en específico)	7	www.uber.com/co/es Uber app

Facilidad de uso (interfaz intuitiva, accesibilidad).	Parcial (en el website es ingles solamente)	7	www.uber.com/co/es Uber app
Experiencia del usuario (fluidez en navegación, pasos mínimos para realizar acciones).	Parcial (ya que en el sitio web no todo el proceso está en español)	7	www.uber.com/co/es Uber app
Diseño visual y adaptabilidad a distintos dispositivos (responsive, móvil, escritorio).	Total	10	www.uber.com/co/es Uber app
Disponibilidad de soporte técnico (canales, tiempo de respuesta).	Total	10	www.uber.com/co/es Uber app
Documentación y manuales de usuario.	Total	10	www.uber.com/co/es Uber app

Total	81

Nota. Esta tabla contiene la evaluación de características de interés para el proyecto de la aplicación Uber envíos. *Obtenido de.* Autores

La tabla 5 presenta la evaluación de la aplicación Didi App (Didi Food) mediante los ocho parámetros estándar, empleando un sistema dual de valoración cualitativo.

Tabla 5

Características de Interes de la Aplicación de Didi Food

Software para Evaluar				
Nombre de la Aplicación	Parámetro de Evaluación	Valor Cualitativo Total, Parcial, Nulo	Escala Cuantitativo (1-10)	URL
Didi Food	Cobertura de funciones principales (registro, asignación, seguimiento, notificaciones)	Total	10	Didi App
	Opciones adicionales (geolocalización,	Total	10	Didi App

historial, chat en vivo, cancelaciones).			
Flexibilidad para distintos tipos de servicio domiciliario.	Total	10	Didi App
Facilidad de uso (interfaz intuitiva, accesibilidad).	Total	10	Didi App
Experiencia del usuario (fluidez en navegación, pasos mínimos para realizar acciones).	Total	10	Didi App
Diseño visual y adaptabilidad a distintos dispositivos (responsive, móvil, escritorio).	Parcial (el website es solo información general)	5	Didi App
Disponibilidad de soporte técnico (canales, tiempo de respuesta).	Total	10	Didi App

Documentación y manuales de usuario.	Nulo (no tienen un manual como tal, solo un par de preguntas)	2	Didi App
	Total	77	

Nota. Esta tabla contiene la evaluación de características de interés para el proyecto de la aplicación Didi Food. *Obtenido de.* Autores

La tabla 6 evalúa el servicio de fletes de Indriver mediante los ocho parámetros establecidos, aplicando un sistema dual de valoración cualitativo.

Tabla 6

Características de Interes de la Aplicación de Indriver

Software para Evaluar				
Nombre de la Aplicación	Parámetro de Evaluación	Valor Cualitativo Total, Parcial, Nulo	Escala Cuantitativo (1-10)	URL
Indriver (flete)	Cobertura de funciones principales (registro,	Total	10	Indriver App

asignación, seguimiento, notificaciones)			
Opciones adicionales (geolocalización, historial, chat en vivo, cancelaciones).	Total	10	Indriver App
Flexibilidad para distintos tipos de servicio domiciliario.	Total	10	Indriver App
Facilidad de uso (interfaz intuitiva, accesibilidad).	Parcial (podría mejorar, especialmente respecto a la parte del pago y la cambian mucho)	8	Indriver App
Experiencia del usuario (fluidez en	Total	10	Indriver App

navegación, pasos mínimos para realizar acciones).			
Diseño visual y adaptabilidad a distintos dispositivos (responsive, móvil, escritorio).	Parcial (el website es solo para información general)	5	Indriver App Indriver.com/es-co
Disponibilidad de soporte técnico (canales, tiempo de respuesta).	Parcial (el soporte técnico no tiene chat)	7	Indriver App Indriver.com/es-co
Documentación y manuales de usuario.	Nulo (no tienen un manual como tal, solo un par de preguntas)	2	Indriver App
	Total	62	

Nota. Esta tabla contiene la evaluación de características de interés para el proyecto de la aplicación Indriver. *Obtenido de.* Autores

La tabla 7 evalúa la plataforma Rapigo empleando los ocho parámetros estándar con un sistema dual cualitativo-cuantitativo, alcanzando un puntaje total de **46 puntos.**

Tabla 7

Características de Interes de la Aplicación de Rapigo.

Software para Evaluar				
Nombre de la Aplicación	Parámetro de Evaluación	Valor Cualitativo Total, Parcial, Nulo	Escala Cuantitativo (1-10)	URL
Rapigo	Cobertura de funciones principales (registro, asignación, seguimiento, notificaciones)	Total	10	Rapigo.co
	Opciones adicionales (geolocalización, historial, chat en vivo, cancelaciones).	Parcial (no hay historial)	8	Rapigo.co

Flexibilidad para distintos tipos de servicio domiciliario.	Nulo (El único servicio es el envío de paquetes, con un límite)	2	Rapigo.co
Facilidad de uso (interfaz intuitiva, accesibilidad).	Parcial (podría mejorar, especialmente respecto al gps)	7	Rapigo.co
Experiencia del usuario (fluidez en navegación, pasos mínimos para realizar acciones).	Parcial (en el gps, y en general del sitio web, podría ser más específica y atractiva visualmente)	7	Rapigo.co
Diseño visual y adaptabilidad a distintos dispositivos (responsive, móvil, escritorio).	Parcial (al cambiar el tamaño de la pantalla pierde elementos)	5	Rapigo.co

Disponibilidad de soporte técnico (canales, tiempo de respuesta).	Parcial (el soporte técnico es exclusivamente a través de whatsapp)	7	Rapigo.co
Documentación y manuales de usuario.	Nulo (no tienen un manual como tal o preguntas frecuentes)	0	Rapigo.co
	Total	46	

Nota. Esta tabla contiene la evaluación de características de interés para el proyecto de la aplicación Rapigo. *Obtenido de.* Autores

La tabla 8 evalúa Tadda Delivery mediante los ocho parámetros estándar, alcanzando el puntaje más bajo: 40 puntos.

Tabla 8

Características de Interés de la Aplicación de Tadda Delivery.

Software para Evaluar				
Nombre de la Aplicación	Parámetro de Evaluación	Valor Cualitativo Total, Parcial, Nulo	Escala Cuantitativo (1-10)	URL

Tadda Delivery	Cobertura de funciones principales (registro, asignación, seguimiento, notificaciones)	Parcial (limitado solo a productos de o relacionados a Bavaria)	7	https://tadadelivery.com.co/ Tadda app
	Opciones adicionales (geolocalización, historial, chat en vivo, cancelaciones).	Parcial (no hay chat en vivo para soporte)	8	https://tadadelivery.com.co/ Tadda app
	Flexibilidad para distintos tipos de servicio domiciliario.	Nulo (El único servicio es el envío de productos Bavaria)	2	https://tadadelivery.com.co/ Tadda app
	Facilidad de uso (interfaz intuitiva, accesibilidad).	Parcial (tiene muy malas opiniones sobre mal	4	https://tadadelivery.com.co/ Tadda app

	funcionamiento de la app, y sin soporte)		
Experiencia del usuario (fluidez en navegación, pasos mínimos para realizar acciones).	Parcial (tiene muy malas opiniones sobre mal funcionamiento de la app, y sin soporte)	7	https://tadadelivery.com.co/ Tadda app
Diseño visual y adaptabilidad a distintos dispositivos (responsive, móvil, escritorio).	Parcial (solo funciona en la app, no en el sitio web)	7	https://tadadelivery.com.co/ Tadda app
Disponibilidad de soporte técnico (canales, tiempo de respuesta).	Parcial (el soporte técnico aparece solo un numero para llamar y un email)	5	https://tadadelivery.com.co/ Tadda app

	Documentación y manuales de usuario.	Nulo (no tienen un manual como tal o preguntas frecuentes)	0	https://tadadelivery.com.co/ Tadda app
		Total	40	

Nota. Esta tabla contiene la evaluación de características de interés para el proyecto de la aplicación Tadda Delivery. *Obtenido de.* Autores

2. Instrumento para estudio de las aplicaciones de aplicaciones de software elegidas

Para fundamentar el desarrollo del proyecto Domicilios Hermez, se analizaron tres plataformas representativas del sector de logística urbana en Colombia: Rappi, referente local con amplia cobertura de servicios; Uber Envíos, solución integrada dentro de una app global de movilidad; y Didi Food, servicio de delivery adyacente a su plataforma de transporte. El resultado de este análisis se muestra en la tabla 9.

Tabla 9

Descripción General de Plataformas de Domicilios y Mensajería

Sinopsis de las Aplicaciones	
Rappi	Fundada en 2015, en Bogotá, opera como una plataforma de intermediación entre los consumidores, los comercios y los repartidores. Tiene diferentes opciones de domicilios. Se orienta hacia las personas que buscan rapidez y variedad en cuanto a los domicilios, personas que viven en ciudades y es difícil desplazarse. También a comercios que quieren

	ampliar la visibilidad. Y a los que necesitan generar ingresos como repartidores. Tiene un amplio catálogo de servicios, diferentes opciones de rapidez en la entrega, una interfaz amigable, opción de geolocalización del repartidor, políticas de protección de datos, y soporte a los diferentes usuarios.
Uber envíos	Es una empresa global, fundada en 2009 en USA, conecta pasajeros y conductores, y en Colombia ofrece servicio de envíos (mensajería, paquetería) este fue lanzado en 2020. En Uber envíos es para las personas que necesitan enviar un artículo puntual, dependiendo del tamaño del paquete entonces se puede enviar por moto o vehículo tradicional o una camioneta. Este se oferta solo en ciudades principales. La interfaz es amigable y está en la misma app para viajes.
Didi Food	Es una adición al servicio de Didi, empezó en Colombia en 2021, tiene presencia en las ciudades principales. Su público objetivo es los socios repartidores, los comercios y los consumidores. Ha ido mejorando la interfaz, tiene canales de contacto para los repartidores, sin embargo, no tiene tantos repartidores como otras apps.

Nota. Sinopsis de plataformas de domicilios y envíos analizadas. *Obtenido de.* Autores.

Instrumento benchmarking de software

Para el desarrollo de la aplicación Hermez, se identificaron cinco características técnicas fundamentales presentes en las soluciones líderes del mercado. La tabla 10 compara cómo Rappi, Uber, Didi y el proyecto Hermez abordan estos requisitos esenciales.

Tabla 10

Matriz de Atributos Técnicos: Plataformas de Domicilios

Aplicación de domicilios - Características				
Aplicación/ Característica	Rappi	Uber	Didi	Hermez
Escalabilidad: capacidad de crecer en usuarios, datos y procesos.	X	X	X	X
Compatibilidad multiplataforma: servidores, navegadores y sistemas operativos soportados.	X	X	X	X
Seguridad integrada: protección contra SQL injection, XSS, CSRF, manejo de sesiones y cifrado.	X	X	X	X
Mantenimiento y facilidad de actualización.	X	X	X	X

Tolerancia a Fallos	X	X	X	X

Nota. Comparativa de requisitos técnicos críticos para plataformas de domicilios. *Obtenido de.* Autores.

Para definir el alcance funcional del proyecto Domicilios Hermez, se analizaron las capacidades principales de las plataformas líderes del sector. La tabla 11 establece una comparativa de siete características funcionales críticas presentes en Rappi, Uber, Didi y la solución propuesta.

Tabla 11

Matriz Comparativa de Características Funcionales de Plataformas de Domicilios

Benchmarking de Software de Domicilios - Características Funcionales				
Aplicación/ Característica	Rappi	Uber	Didi	Hermez
Gestión de usuarios y roles.	X	X	X	X
Soporte para autenticación y autorización.	X	X	X	X
Capacidad de personalización	X	X	X	X
Escalabilidad Funcional	X	X	X	X
Diferentes opciones para envío	X	X	X	X
Integración con APIs	X	X	X	X
Diseño responsive	X	X	X	X

Nota. Comparativa de requisitos funcionales en plataformas de domicilios. *Obtenido de.* Autores.

En la tabla 12 se evaluaron las características de usabilidad y desarrollo presentes en las plataformas líderes del mercado y en el proyecto Hermez.

Tabla 12

Benchmarking de Usabilidad y Desarrollo de Software Domiciliario

Benchmarking de Software de Domicilios - Características de Usabilidad y Desarrollo

Aplicación/ Característica	Rappi	Uber	Didi	Hermez
Interfaz de usuario (UI): fácil de usar	X	X	X	X
Experiencia de usuario (UX)	X	X	X	X
Compatibilidad	X	X	X	X
Cifrado de datos personales	X	X	X	X
Accesibilidad	X	X	X	X
Velocidad de respuesta	X	X	X	X

Nota. Comparación entre la usabilidad y desarrollo de Software Domiciliario. *Obtenido de.* Autores

El soporte técnico y la comunidad de usuarios son factores determinantes para la sostenibilidad y evolución de una plataforma de domicilios. En la tabla 13, se contrastan

cuatro aspectos clave de acompañamiento post-lanzamiento presentes en las principales aplicaciones del mercado.

Tabla 13

Benchmarking de Software de Domicilios - Características de Soporte Y Comunidad

Benchmarking de Software de Domicilios - Características de Soporte Y Comunidad				
Aplicación/ Característica	Rappi	Uber	Didi	Hermez
Documentación oficial y guías de uso.	X	X	X	X
Soporte de la comunidad (foros, GitHub, Stack Overflow).	X	X	X	X
Frecuencia de actualizaciones	X	X	X	X
Casos de éxito o adopción en proyectos reales.	X	X	X	X

Nota. Benchmarking de soporte técnico y comunidad en plataformas de domicilios.

Obtenido de. Autores

3. Instrumento de Elicitación de Requerimientos

Para diferenciar la plataforma Domicilios Hermez del panorama existente, el equipo de desarrollo diseñó dos características específicas que responden directamente al contexto

local y a las necesidades de los usuarios. La tabla 14 describe estas funcionalidades personalizadas y la motivación que fundamenta su implementación.

Tabla 14

Funcionalidades Personalizadas y Fundamento de Implementación

Características Ideadas por el Equipo de Desarrollo	
Característica Propia	Motivación
Aceptación de efectivo o transferencia para el pago solamente	Esta decisión responde a la necesidad del contexto local del proyecto ya que es un municipio pequeño, y también porque el proyecto tiene restricción de recursos.
El proyecto incluirá herramientas avanzadas de calificación y retroalimentación.	Con esto aumentar la calidad del servicio, esta retroalimentación, esto fomentando la mejora continua y la confianza entre ambas partes.

Nota. Características personalizadas y fundamentos de implementación. *Obtenido de.* Autores

4. Instrumento para Consolidar la Propuesta de innovación.

Para estructurar el desarrollo de Domicilios Hermez con enfoque en la innovación y viabilidad, se priorizó un conjunto de características técnicas y funcionales clave. La siguiente tabla establece un orden de importancia (donde 1 representa la máxima prioridad

y 10 la mínima), fundamentando cada decisión en las necesidades del contexto local, cumplimiento normativo y sostenibilidad del sistema.

Tabla 15

Matriz de Innovación: Características Priorizadas para Domicilios Hermez

Aportes a la Innovación del Proyecto de Domicilios (Domicilios Hermez)		
Características para tener en cuenta en el Desarrollo	Prioridad Orden (1-10)	Motivación
Gestión de usuarios y roles	1	Es fundamental para garantizar un control adecuado del acceso y las funcionalidades dentro de la aplicación.
Cifrado de datos personales	2	Por ley es muy importante proteger la información sensible de los usuarios.
Manejo de permisos y datos de GPS	2	Con esto se optimiza la precisión de las entregas y el seguimiento de los pedidos. Con los permisos se garantiza la protección a la privacidad.
Documentación técnica	6	Este componente garantiza la claridad, mantenibilidad y escalabilidad del sistema.

Compatibilidad multiplataforma	3	Hoy con tantos dispositivos tecnológicos, es importante que la aplicación web pueda funcionar en estos y con diferentes sistemas operativos.
Escalabilidad del sistema	3	Con esto se garantiza la opción de actualización y mejoramiento de la aplicación en cualquier momento.
Integración con APIs	2	La integración con APIs es clave, para poder ampliar las funcionalidades de la aplicación.
Gestión del pedido	1	Este el centro de la aplicación controla y coordina todo el ciclo del pedido.
Gestión historial	10	Con el historial se busca poder llevar un registro y análisis de los domicilios.
Base de datos	4	Sin bases de datos no hay como recolectar y gestionar de manera más segura la información

Nota. Contiene la priorización de características técnicas para Domicilios Hermez.

Obtenido de. Autores

Consideraciones Éticas

Algunos señalan que, dentro del ámbito del desarrollo de software, los principales valores éticos incluyen la privacidad, la equidad, la sostenibilidad y la responsabilidad. Se

resalta también que estos principios éticos deben incorporarse desde las etapas iniciales del diseño, no solo tener el punto de vista de contención ante posibles inconvenientes (Alidoosti, R., Lago, P., Razavian, M., & Tang, A., 2022). Teniendo en cuenta todo lo anterior se ha planeado y desarrollado la aplicación web, para que siempre haya transparencia, y confianza, se tendrá una documentación clara, sin practicas engañosas, así como el cuidado en la seguridad y la privacidad de todos los usuarios.

Alcances y Limitaciones

Alcances

Se busca ofrecer una alternativa para los domicilios, ya que no existe ninguna opción en el municipio de Barichara. Se ofrecerá especialmente a los establecimientos comerciales pero cualquier persona tendrá acceso.

Limitaciones

En este proyecto por limitación en el dinero disponible, solo se tuvo en cuenta los pagos en efectivo y por transferencia, sin embargo, siempre está la opción de escalar la aplicación para agregar otros medios de pago como tarjetas débito o crédito.

Las pruebas se realizaron en entornos estandarizados, controlados.

Licenciamiento

La aplicación web se desarrollará bajo un modelo de código abierto u open source, con esto se quiere que cualquier persona tenga acceso y pueda mejorar su código fuente. La decisión se tomó para fomentar la colaboración, y el aprendizaje colectivo, permitiendo que

otros desarrolladores contribuyan y puedan mejorar o adaptar esta aplicación de acuerdo con distintas necesidades.

El proyecto será distribuido bajo la licencia MIT, una de las licencias de software libre más usadas y flexibles, las licencias MIT permiten el uso, copia, modificación, fusión, publicación y distribución del código sin ninguna restricción, siempre y cuando mantenga el aviso de copyright y la nota de la licencia original. Así se puede reutilizar e innovar al mismo tiempo.

Requerimientos del sistema

Esta aplicación web se adapta a diferentes dimensiones de viewport donde se puede agregar clientes y domiciliarios, registrarse dependiendo del rol. Se realiza luego la solicitud del domicilio que tiene varios estados (inicio, preparado, terminado) y se guarda en el historial.

Requerimientos Funcionales

Los requerimientos funcionales definidos para el sistema Domicilios Hermez establecen las capacidades operativas que guiarán su desarrollo. Estas especificaciones abarcan el ciclo completo de gestión de pedidos, desde el registro de usuarios y selección de direcciones, hasta la asignación de domiciliarios, confirmación de pagos y retroalimentación del servicio, priorizadas según su impacto en la experiencia de usuario y viabilidad técnica su respectivo análisis se puede ver en las tablas 29 a 34.

Tabla 16

Requerimiento Funcional RF1: Registro de Usuarios

# ID Requerimiento	ReqF1	Tipo de Requerimiento	Funcional
Descripción	ReqF1. Registro de usuario (clientes y domiciliarios) exigido por el sistema para ingresar, con datos básicos: correo electrónico, y luego en el perfil el nombre y teléfono.		

Justificación	Para el uso del website es necesario tener información básica, para esto es necesario el registro y así poder proceder con el pedido.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	01/10/2025	**Fecha Modificación**	01/10/2025
Porcentaje Implementación	90%		
Origen	Grupo técnico		
Prioridad	Alta.		
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”		

Nota. Especificaciones del RF1: Registro de Usuario. *Obtenido de.* Autores.

Tabla 17

Requerimiento Funcional RF2: Inicio de sesión

# ID Requerimiento	ReqF2	Tipo de Requerimiento	Funcional
Descripción	ReqF2. Inicio de sesión con validación por correo electrónico.		
Justificación	Como usuario (cliente o domiciliario) para la aplicación web se usará el correo electrónico.		

Autor	Equipo	**Versión**	0.1
Fecha Creación	01/10/2025	**Fecha Modificación**	01/10/2025
Porcentaje Implementación	90%		
Origen	Grupo técnico		
Prioridad	Alta.		
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”		

Nota. Especificaciones del RF2: Inicio de sesión. *Obtenido de.* Autores.

Tabla 18

Requerimiento Funcional RF3: Recuperación de contraseña

# ID Requerimiento	ReqF3	**Tipo de Requerimiento**	Funcional
Descripción	ReqF3. Recuperación de contraseña mediante correo electrónico.		
Justificación	Cualquier usuario pueda recuperar o crear una nueva contraseña en cualquier momento.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	01/10/2025	**Fecha Modificación**	01/10/2025

Porcentaje Implementación	90%
Origen	Grupo técnico
Prioridad	Baja
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificaciones del RF3: Recuperación de contraseña mediante correo. *Obtenido de.* Autores.

Tabla 19

Requerimiento Funcional RF4: Opciones de pago permitidos

# ID Requerimiento	ReqF4	Tipo de Requerimiento	Funcional
Descripción	ReqF4. En cuanto al pago, se darán dos opciones un pago inmediato por transferencia o efectivo y la otra diferirlo y pagarlo en el próximo servicio.		
Justificación	En el contexto de inseguridad del país, esta opción permite dar algo más de seguridad al usuario,		
Autor	Equipo	**Versión**	0.1
Fecha Creación	01/10/2025	**Fecha Modificación**	01/10/2025

Porcentaje Implementación	90%
Origen	Grupo técnico
Prioridad	Media.
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificaciones del RF4: Opciones de pagos permitidos, con opción de aplazamiento para el próximo servicio. *Obtenido de.* Autores.

Tabla 20

Requerimiento Funcional RF5: Gestión de Direcciones con Detalles

# ID Requerimiento	ReqF5	**Tipo de Requerimiento**	Funcional
Descripción	ReqF5. Selección de opción y espacios para digitar la dirección y detalles adicionales para una correcta ubicación para la recogida y entrega del pedido		
Justificación	Algunas direcciones presentan cierta dificultad, por esta razón se crea un espacio extra para que el cliente pueda escribir esto.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025

Porcentaje Implementación	90%
Origen	Grupo técnico
Prioridad	Alta.
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificaciones del RF5: Detalles complementarios de ubicación en la aplicación para mejorar la experiencia. *Obtenido de.* Autores.

Tabla 21

Requerimiento Funcional RF6: Visualización de Estados del Pedido

# ID Requerimiento	ReqF6	Tipo de Requerimiento	Funcional
Descripción	ReqF6. Permitir ver el estado del pedido: aceptado, en proceso, pago, entregado.		
Justificación	Las dos partes podrán ver el estado del pedido, para mayor tranquilidad.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025

Porcentaje Implementación	90%
Origen	Grupo técnico
Prioridad	Baja
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF6: Consulta del estado del pedido.

Obtenido de. Autores.

Tabla 22

Requerimiento Funcional RF1: Confirmación de Pago por el Domiciliario

# ID Requerimiento	ReqF7	Tipo de Requerimiento	Funcional
Descripción	ReqF7. Confirmación de pago por parte del domiciliario.		
Justificación	El domiciliario marcará el pedido como pagado, una vez verifique el pago, ya sea efectivo o transferencia.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		

Origen	Grupo técnico
Prioridad	Alta.
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificaciones del RF1: Verificación y registro de pagos. *Obtenido de.* Autores

Tabla 23

Requerimiento Funcional RF8: Asignación de Domiciliario

# ID Requerimiento	ReqF8	Tipo de Requerimiento	Funcional
Descripción	ReqF8. Asignación de domiciliario según la selección del cliente, al tener la opción de elegir entre diferentes tarifas ofrecidas por el domiciliario.		
Justificación	Es importante por rapidez, eficiencia y costos que el domiciliario más cercano sea quien recoja el pedido		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		
Origen	Grupo técnico		

Prioridad	Alta.
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF8: Asignación de domiciliario según la selección del cliente. *Obtenido de.* Autores

Tabla 24

Requerimiento Funcional RF9: Historial y Calificación de Domicilios.

# ID Requerimiento	ReqF9	Tipo de Requerimiento	Funcional
Descripción	ReqF9. El sistema permitirá llevar un control de todos los domicilios realizados y medir la calidad de los mismos por medio de calificaciones.		
Justificación	Es necesario que se pueda consultar el historial en el sistema y además por calidad que los clientes puedan evaluar a los domiciliarios y al sitio web, con esto se espera ofrecer siempre un buen servicio y mejorar.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		

Origen	Grupo técnico
Prioridad	Media
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF9: El sistema permitirá llevar un control de todos los domicilios realizados y medir la calidad. *Obtenido de.* Autores

Tabla 25

Requerimiento Funcional RF10: Diseño Responsive Multiplataforma

# ID Requerimiento	ReqF10	**Tipo de Requerimiento**	Funcional
Descripción	ReqF10. Integración con dispositivos móviles, permitiendo abrir la página en tabletas, portátiles, celulares.		
Justificación	La aplicación web deberá ajustarse a los dispositivos móviles también, conservando un diseño que sea entendible y funcional para el cliente como lo seria en un computador.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		

Origen	Grupo técnico
Prioridad	Alta.
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF10: Integración con dispositivos móviles, permitiendo abrir la página en diferentes tamaños. *Obtenido de.* Autores

Tabla 26

Requerimiento Funcional RF11: Arquitectura Modular del Sistema

# ID Requerimiento	ReqF11	Tipo de Requerimiento	Funcional
Descripción	ReqF11. Diseñado en módulos.		
Justificación	Para permitir escalar y agregar nuevos módulos de ser necesario o nuevas funcionalidades de manera eficiente, así como integrar todos los cambios con lo ya existente.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		
Origen	Grupo técnico		

Prioridad	Alta.
Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF11: Diseñado en módulos. *Obtenido de.* Autores

Tabla 27

Requerimiento Funcional RF12: Cancelación de Pedidos por Error del Cliente

# ID Requerimiento	ReqF12	Tipo de Requerimiento	Funcional
Descripción	ReqF12. Poder cancelar el pedido si el cliente tuvo un error.		
Justificación	Se espera siempre que pueda existir un error de parte de alguno de los usuarios, así que el sistema permitirá ajustar la dirección.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		
Origen	Grupo técnico		
Prioridad	Alta.		

Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF12: Poder cancelar el pedido si el cliente tuvo un error. *Obtenido de.* Autores

Tabla 28

Requerimiento Funcional RF13: Visualización en Tiempo Real de Cambios en Pedidos

# ID Requerimiento	ReqF13	Tipo de Requerimiento	Funcional
Descripción	ReqF12. La aplicación mostrara en tiempo real los cambios hechos en el pedido.		
Justificación	Una característica esencial es que cualquier cambio en el pedido debe poder ser visualizado por las partes interesadas, para estar seguros de que el pedido se recogerá y entregará correctamente.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	03/10/2025	**Fecha Modificación**	03/10/2025
Porcentaje Implementación	90%		
Origen	Grupo técnico		
Prioridad	Alta.		

Verificación	Nuevo instrumento generado con un nuevo número de versión y cuyo estado es “borrador”

Nota. Especificación del requerimiento funcional RF13: Visualización en Tiempo Real de Cambios en Pedidos. *Obtenido de.* Autores

Requerimientos no Funcionales

Los requerimientos no funcionales establecen los estándares de calidad, seguridad y rendimiento que garantizarán la robustez y confiabilidad del sistema Domicilios Hermez. Estas especificaciones definen criterios técnicos obligatorios para la protección de datos, control de errores, autenticación de usuarios, adaptabilidad multiplataforma, mantenibilidad y velocidad de respuesta, asegurando una experiencia segura y eficiente para todos los actores del servicio (tablas 29-34).

Tabla 29

Requerimiento No Funcional NF001: Encriptación de Datos Sensibles

# ID Requerimiento	REQ-NF001	Tipo de Requerimiento	No funcional
Descripción	REQ-NF001_Encriptacion		
Justificación	Almacenamiento de contraseñas y datos sensibles encriptadas, para proteger la información del usuario.		
Autor	Equipo	**Versión**	0.1

Fecha Creación	02/10/2025	**Fecha Modificación**	02/10/2025
Porcentaje Implementación	100%		
Origen	Grupo técnico		
Prioridad	Alta.		
Verificación			

Nota. Especificación del requerimiento no funcional NF001: Almacenamiento de contraseñas y datos sensibles encriptadas, para proteger la información del usuario.

Obtenido de. Autores

Tabla 30

Requerimiento No Funcional NF002: Control de Errores

# ID Requerimiento	REQ-NF002	Tipo de Requerimiento	No funcional
Descripción	REQ-NF002_Control errores		
Justificación	En la app web los errores se mostrarán al usuario en unas pantallas personalizadas, tratando de controlar el error, en caso de que ocurran, para no permitir que el sistema muestre secciones del código que puedan poner en riesgo la seguridad del software		
Autor	Equipo	**Versión**	0.1
Fecha Creación	02/10/2025	**Fecha Modificación**	02/10/2025

Porcentaje Implementación	100%
Origen	Grupo técnico
Prioridad	Alta.
Verificación	

Nota. Especificación del requerimiento no funcional NF002: Manejo de errores con pantallas personalizadas que evitan exponer código del sistema. *Obtenido de.* Autores

Tabla 31

Requerimiento No Funcional NF003: Confidencialidad y Autenticación de Usuarios

# ID Requerimiento	REQ-NF003	Tipo de Requerimiento	No funcional
Descripción	REQ-NF003_Confidencialidad		
Justificación	Antes de realizar cualquier actividad en el sistema los usuarios deberán autenticarse, con un registro previo usando el correo electrónico, Facebook o Google, y tendrá asociado una contraseña.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	02/10/2025	**Fecha Modificación**	02/10/2025
Porcentaje Implementación	100%		

Origen	Grupo técnico
Prioridad	Alta.
Verificación	

Nota. Especificación del requerimiento no funcional NF003: El sistema valida que el usuario hubiera iniciado sesión en cada solicitud. *Obtenido de.* Autores

Tabla 32

Requerimiento No Funcional NF004: Interfaz Responsive y Accesible

# ID Requerimiento	REQ-NF004	Tipo de Requerimiento	No funcional
Descripción	REQ-NF004_Interfaz ajustable		
Justificación	La web puede adaptarse a pantallas sin pérdida de funcionalidad (diseño responsive). La interfaz se ajusta a diferentes tamaños y tiene contraste de color para una comprensión fácil de los diferentes usuarios.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	02/10/2025	**Fecha Modificación**	02/10/2025
Porcentaje Implementación	100%		
Origen	Grupo técnico		
Prioridad	Alta.		

Verificación

Nota. Especificación del requerimiento no funcional NF004 Interfaz adaptable a diferentes tamaños de pantalla sin pérdida de funcionalidad, con contraste de color para fácil comprensión de los usuarios. *Obtenido de.* Autores

Tabla 33

Requerimiento No Funcional NF005: Mantenibilidad del Sistema

# ID Requerimiento	REQ-NF005	Tipo de Requerimiento	No funcional
Descripción	REQ-NF005_Mantenibilidad		
Justificación	El website será fácil de mantener, actualizar y corregir para garantizar que continue en el tiempo y con los cambios exigidos		
Autor	Equipo	**Versión**	0.1
Fecha Creación	02/10/2025	**Fecha Modificación**	02/10/2025
Porcentaje Implementación	100%		
Origen	Grupo técnico		
Prioridad	Alta.		
Verificación			

Nota. Especificación del requerimiento no funcional NF005: la aplicación web será fácil de mantener, actualizar y corregir. *Obtenido de.* Autores

Tabla 34

Requerimiento No Funcional NF006: Rendimiento

# ID Requerimiento	REQ-NF006	Tipo de Requerimiento	No funcional
Descripción	REQ-NF006_Rendimiento		
Justificación	El tiempo de respuesta promedio de la página principal debe ser ≤ 2 segundos ante una petición del usuario. La aplicación web debe estar disponible para su uso un mínimo del 99.9 % al mes.		
Autor	Equipo	**Versión**	0.1
Fecha Creación	02/10/2025	**Fecha Modificación**	02/10/2025
Porcentaje Implementación	100%		
Origen	Grupo técnico		
Prioridad	Alta.		
Verificación			

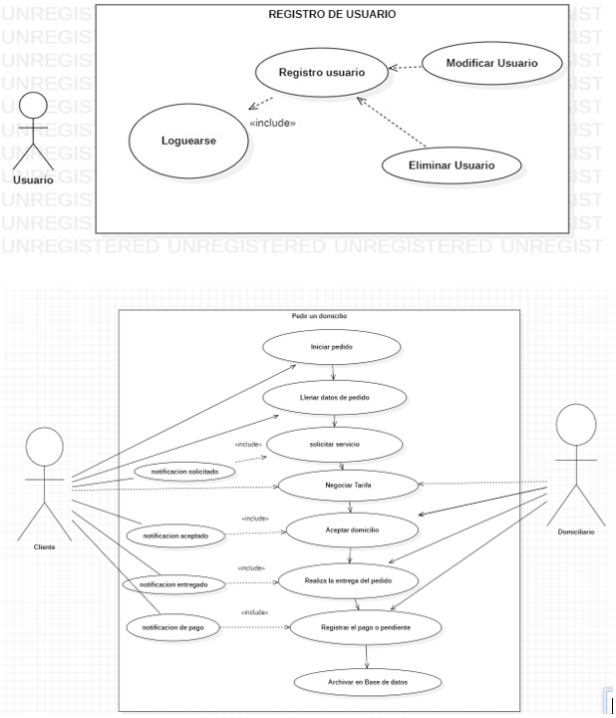
Nota. Especificación del requerimiento no funcional NF006: Tiempo de respuesta ≤2 segundos y disponibilidad del 99.9% mensual *Obtenido de.* Autores

Casos de Usuario

Para el proyecto Domicilios Hermez, se presentó dos cosas de uso que representa como los actores interactúan en los casos de registro de usuario y solicitud de un domicilio, visibles en la Figura 2, analizados en las tablas 35 y 36.

Figura 2

Diagrama de Caso de Uso: Registro y Gestión de Usuarios



Nota. Los autores representan en esta nota los casos de uso Registro, Modificar Usuario, Eliminar Usuario y Loguearse, con los actores Usuario y UNREGISTERED, documentando el flujo completo desde el pedido del cliente hasta el archivado en base de datos, incluyendo negociación de tarifa, notificaciones y registro de pago. *Obtenido des.* Autores

Tabla 35

Caso de Uso: Registro y Gestión de Usuarios

Caso de Uso	Registro Usuario
Actores	Domiciliario y Cliente
Propósito	Permitir la creación de cuentas a los diferentes usuarios, modificarlas o eliminarlas.
Resumen	1.El usuario que puede ser un domiciliario o cliente, entra en el sitio web y selecciona registrarse. 2.Ingresa todos los datos solicitados. 3.Luego de eso podrá ingresar en la aplicación web. Podrá modificar los datos de registro o darse de baja.

Nota. Caso de uso: Registro y gestión de cuentas de usuario, contiene el ciclo de registro, modificación y eliminación de cuentas. *Obtenido de.* Autores

Tabla 36

Proceso de Solicitud y Ejecución de Domicilio

Caso de Uso	Realizar pedido domicilio
Actores	Domiciliario, Cliente
Propósito	Obtener todos los datos necesarios para realizar correctamente el domicilio
Resumen	1.El usuario cliente ingresa en la aplicación web y en primer lugar ingresa la dirección de recogida y luego la dirección de entrega.

	2.Selecciona la clase de vehículo que quiere para esto, por ahora está solo la moto 3.Elige el tipo de domicilio que necesita y puede colocar observaciones especiales para su transporte 4.Luego de esto el sistema mostrará un resumen con todos estos datos ya ingresado y sugerirá un valor para el domicilio, el cliente puede dar más si lo desea. 5.Por su parte a los domiciliarios les llegara la solicitud del servicio y pueden aceptarlo o dejarlo pasar. 6.El domiciliario realiza la entrega 7.Registra el pago ya sea efectivo, transferencia o queda pendiente para la próxima entrega. 8.El servicio queda registrado y archivado en la base de datos para su posterior consulta en la sección historial.

Nota. Caso de uso: Realizar pedido domicilio, contiene el ciclo completo desde la solicitud del cliente, asignación del domiciliario, entrega, registro de pago y archivado en el historial. *Obtenido de.* Autores

Arquitectura del software

Introducción

La arquitectura de software es un elemento crítico para una ingeniería de software efectiva. Se puede ver la arquitectura de software como el puente entre los requerimientos del sistema y la implementación, proporcionando un alto nivel en el diseño del sistema y satisfaciendo los requerimientos. Es como un mapa en el cual está la guía para tomar las decisiones técnicas y también articula los distintos equipos en el desarrollo del software.

Concepto de Arquitectura de Software

La arquitectura de software se puede definir como un conjunto de estructuras necesarias para razonar acerca del sistema, comprendiendo los elementos del software, las relaciones entre ellos y sus propiedades. Al desarrollar un sistema excelente según su arquitectura, se puede mejorar la calidad del sistema, su integridad y coherencia, asegurar que el sistema tendrá las propiedades requeridas por el diseño, así como manejar la complejidad esencial y la complejidad accidental (David Garlan, Mary Shaw, 1993).

En la arquitectura de software también se encuentran los conectores los cuales permiten el intercambio de información y control entre los componentes, los conectores comunican los módulos, servicios o capas dentro de la arquitectura.

Las interfaces en la arquitectura de software definen el conjunto de los métodos, las operaciones o los servicios que un componente da a otro especificando las funcionalidades de cada uno.

Como se configuran todos estos componentes, los módulos los servicios, la forma en la que el sistema organiza, conecta y despliega las distintas funciones, dando soporte, todo esto definido en el diseño. En otras palabras, la configuración de la arquitectura de software da como resultado la estructura lógica del sistema, su distribución física y tecnológica de todos los elementos.

Requerimientos que Influyen en la Arquitectura

El diseño arquitectónico del software está totalmente condicionado por sus requerimientos, aquellos que deben identificarse al realizarse el análisis del problema. Con la selección de la arquitectura se deben tener en cuenta todos los requerimientos y además garantizar el equilibrio entre lo requerido y los atributos del sistema,

Requerimientos Funcionales

Dicen las capacidades del sistema, determina aquello que debe ser útil al usuario, como la gestión de pedidos, la autenticación, o los pagos. Estos requisitos dicen que módulos deben ser los principales y también las interacciones entre los componentes y el flujo de los datos (Pressman, R. S., & Maxim, B. R., 2020). El Paradigma que tiene la arquitectura en islas de Astro busca optimizar el rendimiento, haciendo que las secciones interactivas se carguen solo cuando sean necesarias, mejorando la eficiencia del código. Y con Django en el backend, se establece una conexión por medio de API REST. Otra cualidad es que favorece la modularidad, permitiendo que cada componente sea desarrollado, probado y desplegado de manera independiente.

Requerimientos no Funcionales

Con estos requerimientos se puede delimitar el sistema, ayudan a la toma de decisiones técnicas, pero también estructurales, se revisan aspectos como escalabilidad, disponibilidad, rendimiento, seguridad, mantenibilidad, y usabilidad. Al usar React con Django en el front y back end respectivamente, se garantizan todos los aspectos anteriores,

Estilo de la Arquitectura

Los tipos más comunes de arquitectura son la monolítica, donde toda la funcionalidad se concentra en un único programa que integra interfaz, lógica de negocio y acceso a datos; la cliente-servidor, que separa responsabilidades entre quien solicita los servicios y quien gestiona datos o procesos; la arquitectura en capas, que organiza los componentes en niveles; la de microservicios, que divide el sistema en servicios independientes con funciones específicas; la vista-controlador, que distingue lógica, interfaz y control de interacciones para facilitar la organización del código; y la dirigida a eventos, donde distintas partes del sistema reaccionan ante eventos conforme se presentan.

Dada la naturaleza del proyecto, se puede decir que es una arquitectura cliente-servidor, en la que se puede ver muy diferenciado el frontend (cliente) y el backend (servidor). Este tipo de arquitectura separa las partes del sistema que deben usar una función específica (clientes) de las otras partes que proporcionan esas funciones, la lógica (servidores), es el núcleo del sistema al procesar todas las solicitudes provenientes del cliente, gestionando las bases de datos y aplicando los requerimientos. Con esto se permite trabajar en cada una de las partes por separado (Rod Stephens, 2015).

Justificación de la Arquitectura

Se ha seleccionado la arquitectura cliente-servidor e implementada para con esto garantizar tener una arquitectura moderna, que puede ofrecer un buen rendimiento y facilidad de mantenimiento optimizando los tiempos de carga. Estas partes diferenciadas, permite tener componentes interactivos, sin comprometer la velocidad general del sitio, se carga únicamente el código que cada sección necesita.

Esta arquitectura facilita la incorporación de componentes reutilizables y dinámicos, permitiendo la escalabilidad en una parte especifica sin necesidad de alterar todo el código. Finalmente, la arquitectura que se ha elegido mejora la experiencia de usuario, permitiendo que con ella se alcance los objetivos propuestos.

Diseño de la Arquitectura

En esta arquitectura modelo cliente-servidor se tiene:

En el **frontend** se usa el framework Astro, que ayuda en la rapidez de la aplicación web con su paradigma de arquitecturas de islas, y el uso de Javascript. Esto configura la interfaz de usuario y consume la API del backend. Con el uso de Astro se obtiene un renderizado parcial o completo dependiendo de la configuración. Entre las tareas esta el manejo de la autenticación y el envío de peticiones a la API. La integración de Astro es fácil con componentes desarrollados con React.

En el **backend** se utiliza Django REST framework y Phyton, desde aquí se manejan los datos y la lógica del sistema, se reciben y devuelven los datos en formato Json, respondiendo así todas las solicitudes del frontend, valida, guarda y actualiza los datos, la

persistencia de la información, así como se encarga de la seguridad y los permisos del sistema.

Para la comunicación se usa **API REST**, este comunica Astro y Django a través de peticiones HTTP. Y entonces el backend responde desde Json, y el frontend interpreta esos datos y los muestra al usuario. La comunicación de esta forma mejora la escalabilidad y el mantenimiento del sistema.

En cuanto a la base de datos se tiene Slq Lite.

Vista General del Sistema

Al implementar una arquitectura cliente-servidor, se permite la interacción entre el cliente que solicita el servicio de domicilio y el repartidor o domiciliario que lo ejecuta.

El flujo de la información sería así:

Solicitud del servicio: la información inicia al cliente acceder a la aplicación web (Astro y Javascript), hace el registro en el sistema, aquí coloca su nombre, teléfono, correo electrónico, crea una contraseña y luego solicita el envío, en esta interfaz el cliente proporciona los datos como dirección de recogida, dirección de entrega, cantidad y método de pago. Acá el estado es “Solicitado”. Luego se envían estos datos al backend a través de una API REST, donde se valida y genera un registro de solicitud en la base de datos SQL lite.

Asignación: el backend toma la información y gestiona todas las solicitudes, el domiciliario va a ver (luego de registrarse e ingresar) todos los pedidos activos, al negociar con el cliente la tarifa y cuando el cliente acepta la tarifa, el domicilio el estado nuevo seria

“Aceptado” el backend cambia el estado para no mostrarlo a otro repartidor, y en cambio notificar al domiciliario que el cliente acepto la tarifa y él puede iniciar el transporte. Al realizar todo esto garantizamos la sincronización de la información para todos los actores envueltos en el proceso.

Ejecución del servicio: el domiciliario utiliza su modulo y ve todos los detalles del pedido, durante la entrega el estado del servicio cambia de “En Proceso”. Al actualizar los estados, el backend mantiene la trazabilidad completa del servicio, y actualiza la información visible para el cliente.

Cierre: una vez el domiciliario haya entregado el pedido, dependiendo de lo que el cliente decida, está la opción de realizar el pago de una vez o pagarlo en el siguiente pedido, este estado se muestra como “Pago” y luego “Entregado” que ya indicaría que no hay nada más pendiente en este domicilio. Cuando el domicilio ya está terminado, el sistema registra todos los detalles, todo esto el backend lo almacena en la base de datos, y se puede consultar después como un historial, para cada cliente y cada domiciliario.

Al seguir la información este flujo, se espera una comunicación eficiente, segura y estructurada, así como abierta a cambios de ser necesario.

Componentes Principales

Para Backend:

Python: es un leguaje de propósito muy general, es un lenguaje de muy alto nivel, dinámico, orientado a objetos y multiplataforma también. Se puede ejecutar en las principales plataformas de hardware. Es de código abierto.

Django: es un framework basado en Python, permite crear soluciones web con pocas líneas de código,

Para Frontend:

Astro: es el framework para construir sitios web, estos están orientados al contenido, páginas que tengan mucho contenido, como por ejemplo algo de marketing.

REACT: es una biblioteca JavaScript, diseñada para crear interfaces. Es de código abierto, para aplicaciones web y móviles.

HTML: Lenguaje de Marcado de Hipertexto, es un lenguaje fundamental para trabajar en la web, usa etiquetas para organizar.

CSS: Hojas de estilo en cascada, es un poderoso lenguaje de programación que transforma la presentación de un documento o la colección de documentos. Css describe la presentación de un contenido web en la pantalla, manejando o controlando el color, márgenes, y todo lo referente al aspecto del diseño visual.

JavaScript: es un lenguaje de scripts para dar funcionalidades interactivas o dinámicas en las páginas web como carruseles de imágenes o controlar multimedia.

API: Django + Django REST Framework

Diagramas Arquitectónicos

Los diagramas arquitectónicos constituyen la base documental para la comprensión, diseño e implementación del sistema Domicilios Hermez. Esta sección presenta cuatro modelos visuales fundamentales: el diagrama de arquitectura, que establece la estructura

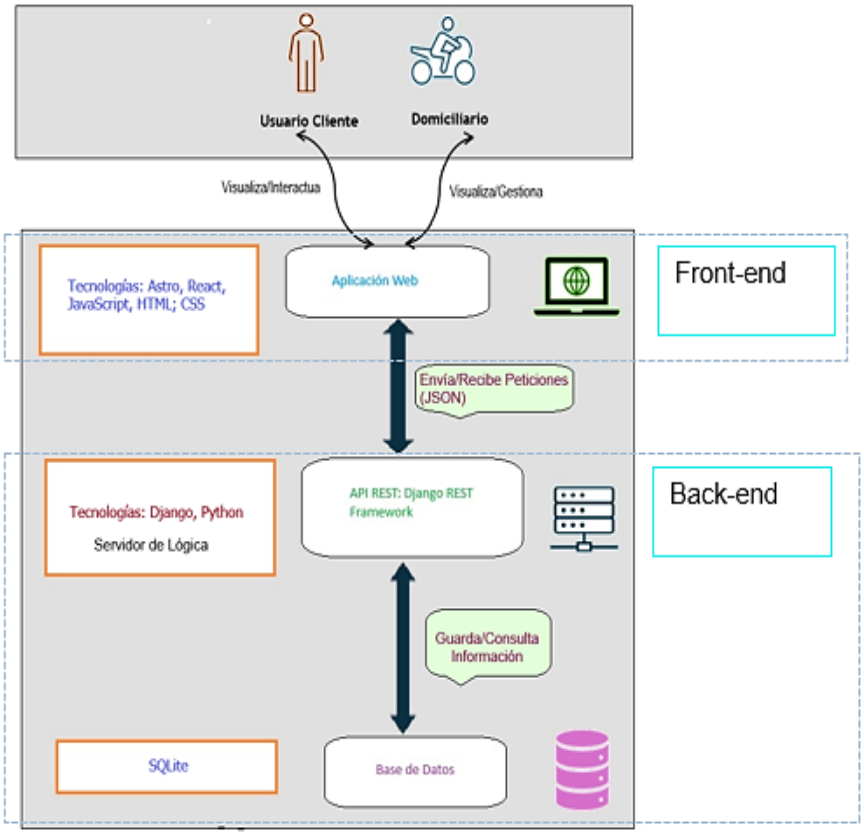
técnica de capas y tecnologías; el diagrama de actividades, que modela el flujo de procesos del negocio; el diagrama de secuencia, que detalla la interacción temporal entre componentes; y el diagrama entidad-relación, que define el modelo de datos subyacente. Juntos, estos diagramas proporcionan una visión integral y coherente del sistema, facilitando la toma de decisiones técnicas y garantizando la alineación entre los requisitos funcionales y la solución propuesta.

Diagrama de Arquitectura

La Figura 3 expone la arquitectura global de la aplicación web, mostrando la interacción entre los actores principales (usuario cliente y repartidor), la parte de la interfaz de usuario construida con astro y JavaScript, HTML y CSS, que son tecnologías contemporáneas y como lógica en el backend desarrollado en Django junto con Django REST Framework y la base de datos SQLite. La representación gráfica detalla el flujo de comunicación que se realiza a través de Json tanto el frontend como en el backend. Y la base de datos SQLite que permite el proceso de consulta y persistencia de datos, garantizando el funcionamiento integral del sistema.

Figura 3

Diagrama de Arquitectura



Nota. Representa la estructura de tres capas: frontend (Astro, React, JavaScript, HTML, CSS), backend (API REST Django) y base de datos (SQLite). *Obtenido de.* Autores.

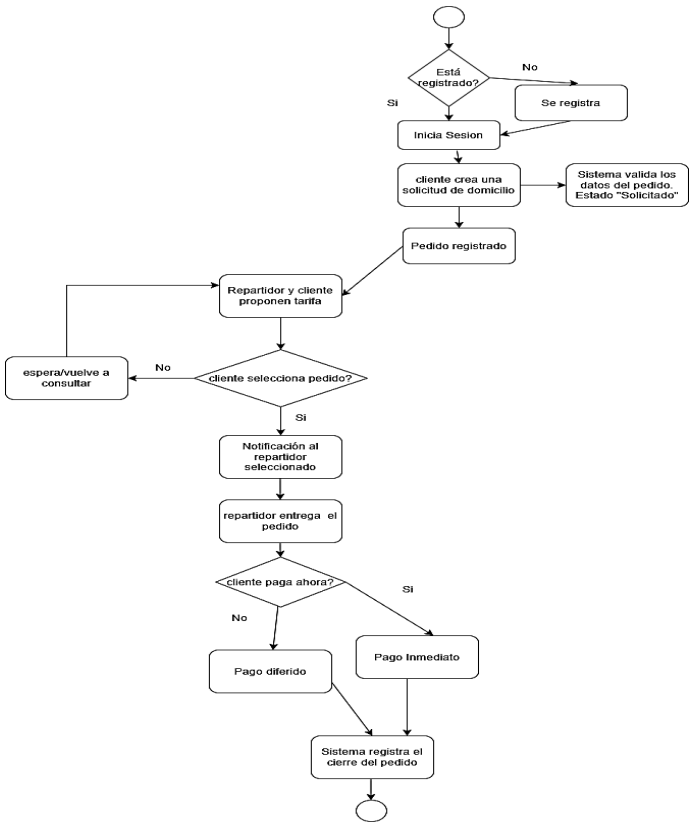
Diagrama de Actividades

La Figura 4 muestra el ciclo completo de actividades de un pedido dentro de la aplicación web. El proceso se inicia con el acceso del usuario. Quien debe contar con registro previo para continuar, una vez autenticado, el cliente formula la solicitud y ofrece una tarifa, el domiciliario recibe la notificación y puede aceptar esa tarifa o pedir una

diferente, cuando el cliente acepte la tarifa negociada, entonces se le notificara al domiciliario para que pueda empezar el desplazamiento necesario para el pedido. Posterior a la llegada al destino del domiciliario, entonces sigue el pago que puede ser en efectivo o transferencia inmediato o la oportunidad de diferirlo para pagarlo con el siguiente pedido, y una vez que el domiciliario registre esto en el sistema, el sistema clausura el pedido y el ciclo se da por concluido.

Figura 4

Diagrama de Flujo del Proceso de Solicitud y Ejecución de Domicilio.



Nota. Diagrama de flujo que representa el proceso completo desde registro de cliente, creación de solicitud, validación, propuesta de tarifas, selección de repartidor, entrega y registro de pago, hasta cierre del pedido. *Obtenido de.* Autores

Diagrama de Secuencia UML

La Figura 5 muestra de manera detallada el flujo dinámico de interacción entre los actores y los componentes principales del sistema de gestión de domicilios. Este modelo UML permite visualizar el comportamiento temporal del sistema desde la solicitud inicial del servicio hasta su finalización y registro en el historial, evidenciando la comunicación entre el cliente, el domiciliario y las distintas capas de la arquitectura tecnológica.

El proceso inicia cuando el Cliente accede a la aplicación web, se registra o inicia sesión y proporciona los datos necesarios para generar una solicitud de domicilio. La Aplicación Web envía dicha información al Backend que valida los datos y registra la solicitud en la Base de Datos SQLite, estableciendo el estado inicial "Solicitado".

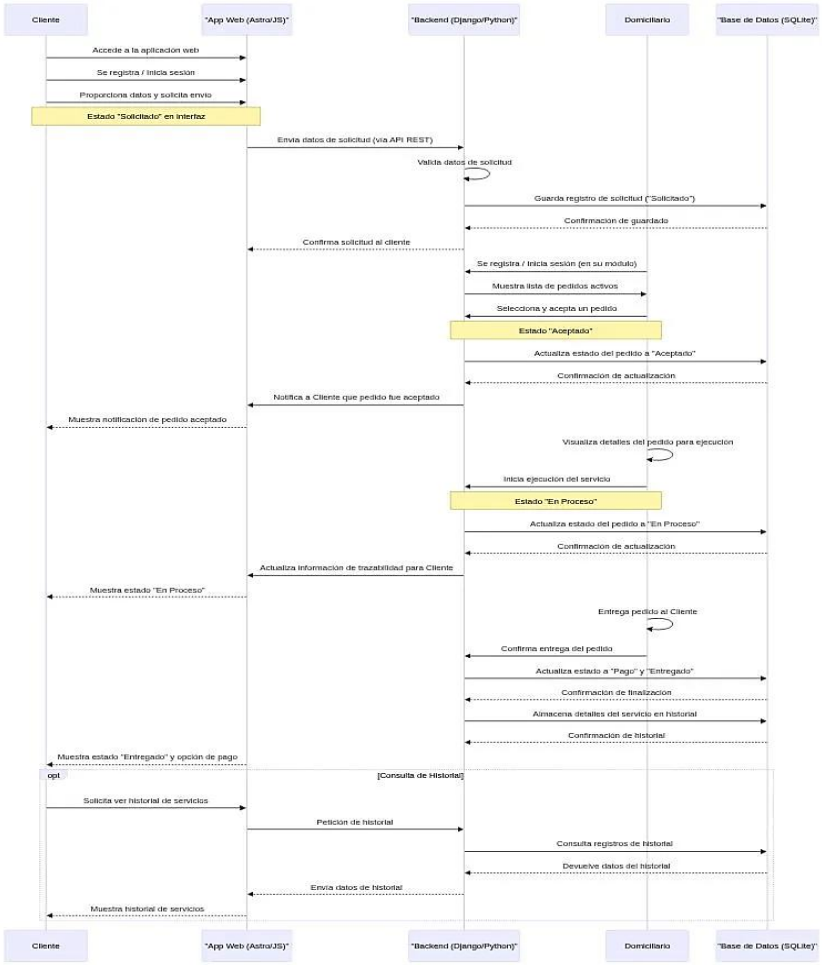
Paralelamente, el domiciliario accede a su módulo dentro de la aplicación, donde visualiza la lista de solicitudes activas. Tras negociar la tarifa y el cliente aceptar una de las tarifas, el backend actualiza el estado a “Aceptado” y notifica al cliente que su solicitud ya fue tomada. A continuación, el domiciliario visualiza los detalles del servicio e inicia la operación correspondiente, momento en el cual el estado cambia a “En Proceso”.

Durante la ejecución, el backend realiza actualizaciones periódicas de la información asociada al pedido, asegurando la trazabilidad en tiempo real para el cliente. Una vez completada la entrega, el domiciliario confirma la finalización del servicio y del

pago, y el backend registra los estados “Entregado” y “Pagado” según corresponda. Finalmente, el sistema almacena la información del proceso en el historial del cliente y del domiciliario.

Figura 5

Diagrama Secuencia UML



Nota. Representación de la interacción entre los componentes del sistema. Mostrando el orden de casos y sus actores dando a comprender la línea de vida de la aplicación.

Obtenido de. Autores.

Diagrama de Entidad Relación

El presente Diagrama Entidad–Relación (DER) representa la estructura lógica de la base de datos diseñada para el sistema de gestión de domicilios. El modelo incluye las entidades fundamentales, sus atributos, llaves primarias, llaves foráneas y las relaciones entre los diferentes elementos que intervienen en los procesos de solicitud, asignación, ejecución y registro histórico de los servicios de entrega.

La entidad central del modelo es deliveries_delivery, la cual almacena la información principal de cada servicio de entrega, incluyendo datos como descripción, peso estimado, direcciones, estados, vehículo asignado, domiciliario responsable y timestamps relevantes. Esta entidad se relaciona con deliveries_deliveryoffer, que registra las ofertas o propuestas de servicio realizadas por los domiciliarios, y con deliveries_deliveryhistory, que almacena el historial de estados y eventos asociados a cada entrega.

El sistema define además la entidad deliveries_deliveryquote, donde se administran las cotizaciones generadas para un servicio, incluyendo valores estimados, tipo de vehículo requerido, método de pago y observaciones. La integración con la gestión de usuarios se realiza mediante la entidad users_user, que representa tanto al cliente como al domiciliario y que se relaciona con el resto del modelo mediante llaves foráneas para vincular solicitudes, ofertas, entregas e historiales.

En cuanto a la administración de vehículos, el diagrama incluye las entidades vehicles_vehicle, que almacena la información detallada de cada vehículo registrado en el sistema, y vehicles_vehicletype, que define los tipos de vehículos disponibles junto con sus capacidades. La relación entre ambos se refuerza con la tabla asociativa

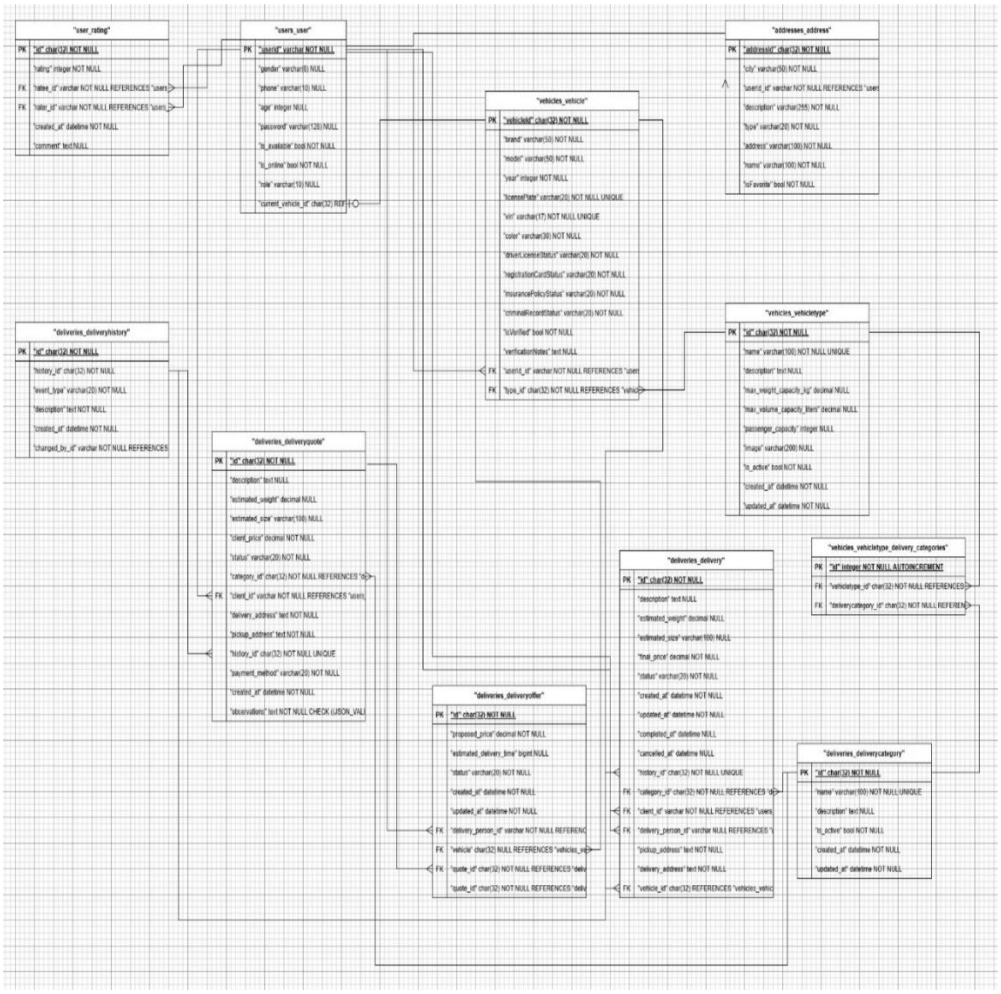
vehicles_vehicletype_delivery_categories, utilizada para categorizar los tipos de entrega compatibles con cada tipo de vehículo. Asimismo, la entidad deliveries_deliverycategory define categorías específicas de entrega y se incorpora como referencia dentro de las solicitudes de domicilio.

Adicionalmente, la entidad addresses_address permite almacenar y asociar direcciones específicas para clientes y entregas, garantizando consistencia en la información geográfica. La entidad user_rating, por su parte, registra evaluaciones entre clientes y domiciliarios, fortaleciendo el componente de reputación dentro del sistema.

En conjunto, este DER refleja una base de datos altamente estructurada, normalizada y orientada a soportar los diferentes procesos del sistema: registro de usuarios, administración de vehículos, gestión de solicitudes, seguimiento operativo, actualización de estados, registro histórico y calificaciones. La claridad de las relaciones y la distribución de responsabilidades en las entidades aseguran integridad referencial, consistencia de datos y capacidad de crecimiento para futuras extensiones del sistema. Lo anterior se puede validar en la Figura 6.

Figura 6

Diagrama Entidad Relación.



Nota. Diagrama de las relaciones entre las tablas presentes en la base de datos del proyecto Domicilios Hermez *Obtenido de.* Autores

Consideraciones de Seguridad y Escalabilidad

La seguridad del sistema tiene en cuanta la separación entre el frontend y el backend, entonces Django actúa como el único acceso a la base de datos, y se gestionan

todas las solicitudes del frontend por medio de API REST seguras, ya que han aplicado las políticas de autenticación previamente.

La tecnología de Django ofrece protecciones adicionales contra amenazas comunes: evita ataques XSS al impedir la inyección de código malicioso en campos de entrada, bloquea CSRF para evitar que un atacante ejecute acciones en nombre del usuario y aplica validación de datos en el servidor que impide la manipulación de solicitudes mediante SQL.

Y por este diseño también, y su arquitectura, facilita grandemente la escalabilidad horizontal y verticalmente. El backend desarrollado con Django permite atender un mayor número de solicitudes en cualquier momento dado, así como el aumento en la cantidad de recursos. También permitiría en cualquier momento su despliegue en la nube de ser necesario, y la base de datos permitiría su optimización ante un aumento en los usuarios y las solicitudes.

Reflexiones Acerca de la Arquitectura

La arquitectura que se ha propuesto para esta aplicación web de domicilios constituye un elemento fundamental para el cumplimiento de los requerimientos del proyecto, trata de ofrecer una estructura que sea sólida, escalable y segura, siendo capaz de soportar y contener todas las funcionalidades requeridas.

Al elegir la opción cliente-servidor se separan las responsabilidades y se hace más fácil la actualización y cambio que se requieran en cualquier momento, sin necesidad de cambiar todo, solo aquella parte necesaria, se logra separar la capa de presentación de la

lógica, y la capa de persistencia, asegurando modularidad, escalabilidad y un mantenimiento más eficiente del sistema.

Por medio de los diferentes diagramas podemos visualizar la forma integral como fluye la información y como interactúan los diversos actores. También podemos ver los procesos internos del sistema, pudiendo ver una lógica bien sincronizada para llegar al cierre del ciclo. Las medidas contempladas de seguridad más las validaciones que hace el sistema, garantizan que las interacciones entre cliente, domiciliario y servidor y bases de datos sean siempre seguras.

Diseño del Sistema

Diseño de Interfaz de Usuario

Durante esta fase de diseño del sistema, se realizó un proceso de ideación visual centrado en el usuario y en dotar de identidad al software, optando por crear Wireframes interactivos de alta fidelidad en Figma. Esto permitió definir la arquitectura de la interfaz y la experiencia de navegación, enfocándonos en mapear los diferentes flujos desde los principales roles o actores (cliente y domiciliario).

El diseño siguió los principios de usabilidad ISO 9241-11 y diseño centrado en el usuario (Pedraza-Gutiérrez et al., 2023) establecidos en el marco conceptual, priorizando:

Claridad visual: Interfaz limpia con contrastes de color accesibles, íconos intuitivos y jerarquía tipográfica que facilite la comprensión para usuarios con diversas capacidades digitales.

Flujo mínimo: Reducción de pasos para completar acciones críticas (solicitar domicilio en máximo 5 interacciones).

Adaptabilidad responsive: Diseño que conserva funcionalidad en pantallas de 320px (celulares básicos) hasta 1920px (desktop).

Lenguaje coloquial: Uso de términos familiares en español santandereano para generar cercanía cultural.

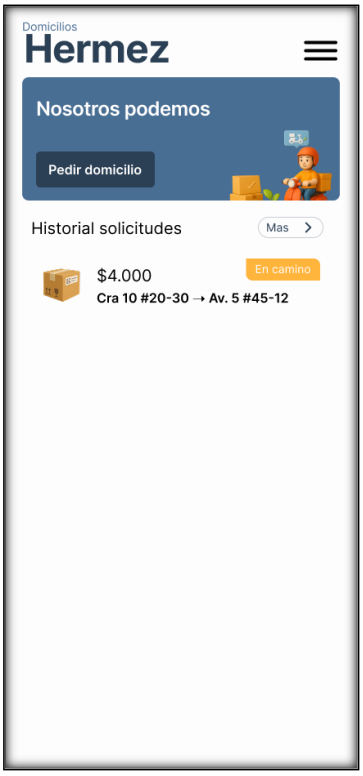
La estructura en **Figma** incluye pantallas estáticas para: inicio de sesión, creación de cuenta, vista del inicio, configuración, menú, formulario de solicitud del domicilio,

página de historial, sección de direcciones, sección de vehículos, vista del rol de domiciliario, entre otras. Esta fase de diseño previo al desarrollo sirvió como guía de referencia para una codificación más ágil, logrando validar rápidamente los componentes y su visión frente a diferentes casos de uso.

La Figura 7 representa el mockup de la pantalla principal de la app móvil Domicilios Hermez, con opciones de pedido, historial activo y navegación.

Figura 7

Mockup de la Pantalla Principal de la Aplicación Móvil Domicilios Hermez

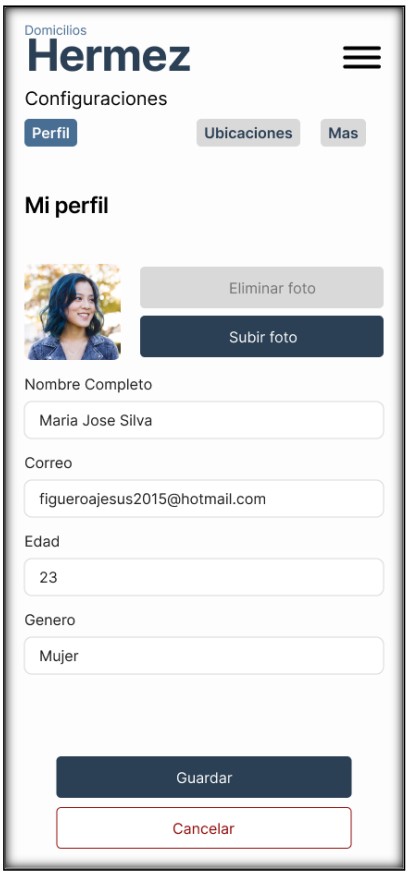


Nota. Representa la interfaz principal de usuario del aplicativo móvil, mostrando las funcionalidades básicas de pedido, historial y navegación. Tomado de Prototipo de diseño Domicilios Hermez. *Obtenido de.* Autores.

La Figura 8 representa el Mockup de la pantalla de configuración de perfil de la aplicación móvil Domicilios Hermez, con formulario editable de datos personales y opciones de gestión.

Figura 8

Mockup de la Pantalla de Configuración de Perfil de Usuario

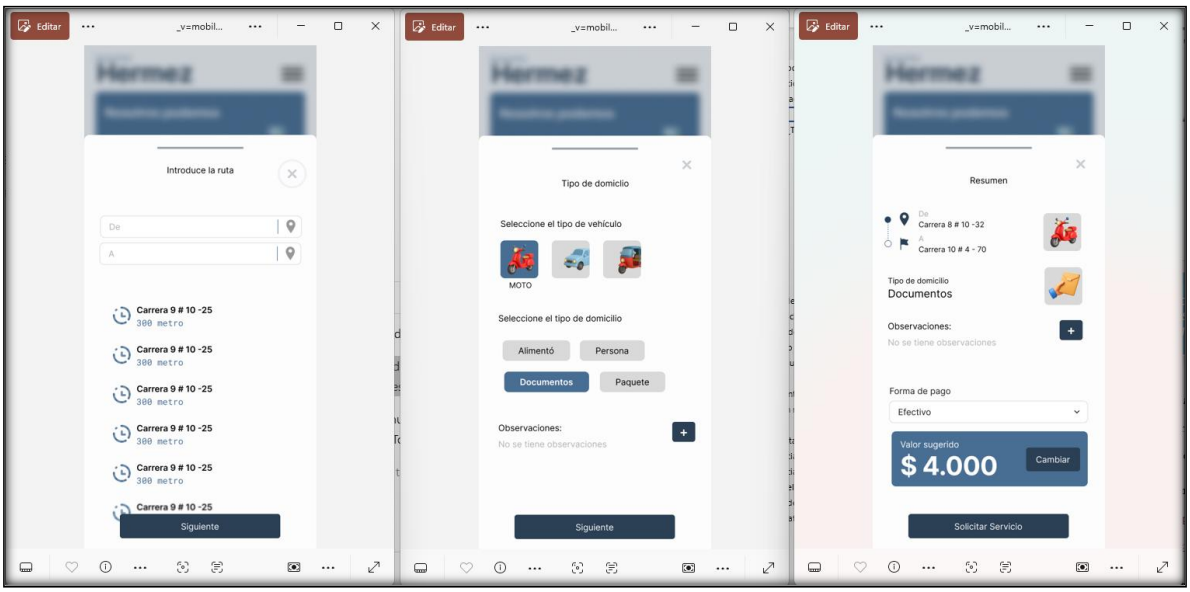


Nota. Pantalla de configuración de perfil con formulario de datos personales (nombre, correo, edad, género), gestión de foto de perfil y menú lateral. Tomado de Prototipo de diseño Domicilios Hermez. *Obtenido de.* Autores.

La Figura 9 representas los Mockups del flujo de solicitud de domicilio mostrando las pantallas secuenciales de ingreso de direcciones, selección de servicio y resumen final.

Figura 9

Secuencia de Tres Pantallas del Proceso de Solicitud



Nota. Secuencia de tres pantallas del proceso de solicitud: 1) ingreso de direcciones origen y destino, 2) selección de tipo de vehículo y categoría de domicilio, 3) resumen del pedido con confirmación de tarifa y pago. Tomado de Prototipo de diseño Domicilios Hermez. *Obtenido de.* Autores.

La Figura 10 representa el Mockup de la pantalla donde el cliente recibe y evalúa las ofertas de los domiciliarios disponibles para su solicitud.

Figura 10

Mockup de Pantalla de Selección de Propuestas de Domiciliarios

Nota. Pantalla de espera y selección de propuestas de domiciliarios, mostrando temporizador, calificaciones, tarifas y botones de aceptar/rechazar. Tomado de Prototipo de diseño Domicilios Hermez. *Obtenido de.* Autores.

Enlace al proyecto en Figma: Domicilios Hermez - Figma

Manual de Usuario

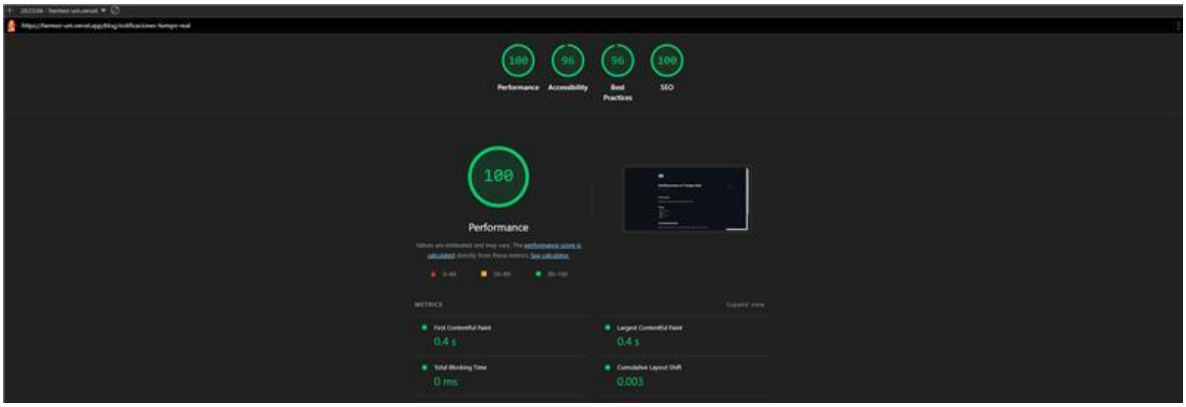
Se desarrolló un manual de usuario interactivo y accesible que documenta de manera integral el funcionamiento de la aplicación web para los diferentes perfiles de usuario. Este material se concibió como una guía autocontenida, diseñada bajo los principios de claridad pedagógica y minimalismo cognitivo, priorizando la comprensión inmediata sobre la sobrecarga de información.

El manual combina secciones transversales con vistas diferenciadas por rol. Las secciones como Bienvenida, Configuraciones, Historial, Consejos y Mejores Prácticas son comunes a todos los usuarios, mientras que las secciones de Inicio ofrecen flujos específicos según el rol activo (cliente o domiciliario). Cada sección incluye instrucciones paso a paso, capturas de pantalla contextualizadas y ejemplos de casos de uso reales del municipio de Barichara. Se implementó un sistema de navegación no lineal que permite acceder directamente a la información requerida sin recorrer el documento secuencialmente, optimizando el tiempo de consulta en situaciones operativas críticas.

Enfoque particular recibió la sección de Consejos y Mejores Prácticas, donde se documentan los escenarios más comunes durante la ejecución de domicilios, con protocolos de contingencia y contactos de soporte local. La interfaz del manual responde a criterios de accesibilidad WCAG 2.1, garantizando su uso en dispositivos con conectividad limitada y condiciones de visualización adversas, como evidencia a que se cumple lo anterior mencionado se puede ver la Figura 11, realizada con la herramienta de Lighthouse del navegador Chrome.

Figura 11

Resultados de Auditoría de Accesibilidad del Manual.



Nota. Imagen del resultado de la herramienta de Lighthouse. *Obtenido de.* Autores

Este diseño documental, alineado con los requisitos de usabilidad ISO 9241-11, permite reducir la curva de aprendizaje en los usuarios.

Acceso al manual interactivo: https://hermez-um.vercel.app/

Pruebas del Sistema

Estrategia de Pruebas

Se aplicó un enfoque de pruebas en dos niveles para validar la funcionalidad y robustez del sistema:

- **Pruebas Unitarias (Backend):** Validación de lógica de negocio usando pytest en Django, cubriendo modelos (User, Delivery, Vehicle, Address) y funciones utilitarias (generación de PIN, cálculo de búsqueda por cercanía).
- **Pruebas Unitarias (Backend):** Validación de lógica de negocio usando pytest en Django, cubriendo modelos (User, Pedido) y funciones utilitarias (generación de PIN, cálculo de búsqueda por cercanía).

Prueba Unitarias (Backend)

Se ejecutaron 16 casos unitarios validando lo siguiente:

Módulo Deliveries (Archivo: deliveries/tests/test_delivery_models.py)

Valida que, al crear una entrega, los campos status y delivery_person se establezcan con sus valores por defecto (“assigned" y None respectivamente).

Comprueba que cuando un vehículo asociado a una entrega es eliminado, el campo vehicle en la entrega se establece a None (comportamiento SET_NULL).

Verifica que el campo status de una entrega solo acepta valores válidos definidos en sus choices, lanzando una ValidationError si se intenta asignar un valor inválido.

Asegura que el campo final_price de tipo Decimal persiste correctamente el valor asignado sin pérdida de precisión.

Valida que las relaciones inversas (related_name) entre User (como cliente), Vehicle y Delivery funcionan correctamente, permitiendo contar las entregas asociadas a un cliente y a un vehículo.

Confirma que cuando el estado de una entrega cambia a "delivered" , el campo completed_at se establece automáticamente (no es None), y cancelled_at permanece None.

Confirma que cuando el estado de una entrega cambia a "cancelled" , el campo cancelled_at se establece automáticamente (no es None), y completed_at permanece None.

Módulo vehicles (Archivo: vehicles/tests/test_vehicle_models.py)

Valida que, al crear un vehículo, los campos vehicleId y isVerified se establezcan con sus valores por defecto (no es None y False respectivamente).

Comprueba que el campo licensePlate es único, lanzando una IntegrityError si se intenta crear dos vehículos con la misma placa.

Comprueba que el campo vin (Número de Identificación del Vehículo) es único, lanzando una IntegrityError si se intenta crear dos vehículos con el mismo VIN.

Verifica que no se puede eliminar un VehicleType si hay vehículos asociados a él, lanzando una ProtectedError (comportamiento PROTECT).

Valida que la relación inversa (related_name) entre User (como conductor) y Vehicle funciona correctamente, permitiendo contar los vehículos asociados a un usuario.

Módulo Addresses (Archivo: addresses/tests/test_address_models.py)

Valida que, al crear una dirección, los campos addressId y isFavorite se establezcan con sus valores por defecto (no es None y False respectivamente).

Verifica que el campo type de una dirección solo acepta valores válidos definidos en sus choices, lanzando una ValidationError si se intenta asignar un valor inválido.

Comprueba que el campo isFavorite funciona correctamente y que se pueden filtrar las direcciones de un usuario basándose en este campo.

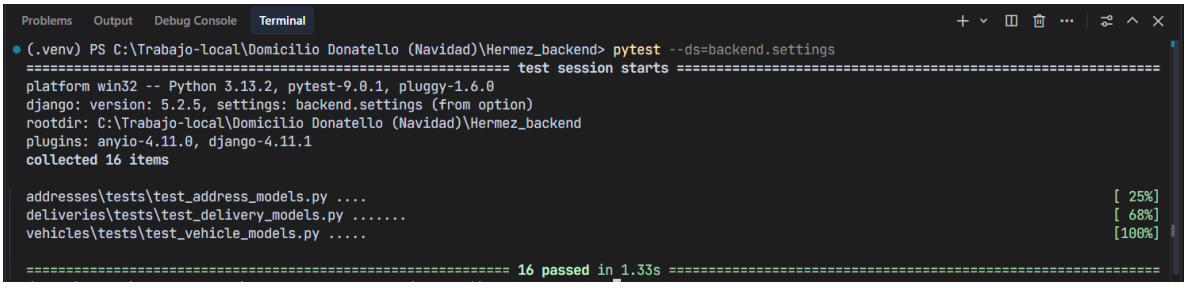
Valida que la relación inversa (related_name) entre User y Address funciona correctamente, permitiendo contar las direcciones asociadas a un usuario.

Las pruebas unitarias se realizaron en el entorno virtual de Python, ejecutando el comando y teniendo el resultado de la Figura 12:

- pytest --ds=backend.settings

Figura 12

Captura de Pantalla de la Consola, donde se Valida que se Pasaron las Pruebas



Nota. Captura de la consola donde se pasan con éxito las 16 pruebas. *Obtenido de.* Autores

Despliegue del Sistema

Guía de Despliegue para el Backend de Hermez

Este documento describe los pasos necesarios para desplegar el proyecto en una plataforma de nube moderna que soporte aplicaciones Python y servicios de larga duración (para WebSockets).

Resumen de la Arquitectura en Producción

Un servidor de aplicaciones ejecutará Django con gunicorn para atender peticiones HTTP; un trabajador de fondo, en proceso separado, lanzará los consumidores de django-channels encargados de las conexiones WebSocket; la plataforma gestionará una base de datos PostgreSQL y un servicio de Redis actuará como caché y broker de mensajes para la comunicación entre Django y los trabajadores de channels.

Preparar el Código para Producción

Antes de desplegar, actualiza tu repositorio local incluyendo en requirements.txt todas las librerías que se indican en la Figura 13.

Figura 13

Dependencias Necesarias para Producción

```
# Servidor de aplicación para producción
gunicorn

# Adaptador de Python para PostgreSQL
psycopg2-binary

# Servir archivos estáticos (para el admin de Django)
whitenoise

# Para leer variables de entorno de un archivo .env en desarrollo local
python-dotenv

# Para parsear la URL de la base de datos
dj-database-url
```

Nota. Archivo de requerimientos con paquetes esenciales para producción: servidor WSGI (gunicorn), adaptador PostgreSQL (psycopg2-binary), manejo de archivos estáticos (whitenoise), carga de variables de entorno (python-dotenv) y parseo de URL de base de datos (dj-database-url). *Obtenido de.* Autores

Modificar backend/settings.py para Producción

Para evitar exponer secretos como SECRET_KEY en el código, se gestionarán mediante variables de entorno; por ello, al inicio del archivo se importan las librerías os y dotenv, tal como se ilustra en la Figura 14, con el fin de acceder de forma segura a dichas variables.

Figura 14

Configuración de Carga de Variables de Entorno en Django

```python
import os
from pathlib import Path
# Opcional: para cargar un archivo .env en desarrollo
from dotenv import load_dotenv

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Cargar variables de entorno desde .env (solo para desarrollo local)
load_dotenv(os.path.join(BASE_DIR, ".env"))
```

Nota. Fragmento de código Python que importa módulos os y pathlib, define la ruta base del proyecto (BASE_DIR) y carga variables de entorno desde un archivo. *Obtenido de.* Autores

Configurar SECRET_KEY y DEBUG:

Reemplaza la configuración actual de SECRET_KEY y DEBUG, por lo de la Figura 16:

Figura 15

Configuración de Variables de Entorno en Django

```python
# Lee la SECRET_KEY desde una variable de entorno.
SECRET_KEY = os.environ.get('SECRET_KEY', 'una-clave-secreta-por-defecto-para-desarrollo')

# El modo DEBUG debe ser False en producción.
# La variable de entorno se debe establecer como 'True' o 'False' (texto).
DEBUG = os.environ.get('DEBUG', 'False').lower() == 'true'
```

Nota: Fragmento de código que lee la SECRET_KEY y el modo DEBUG desde variables de entorno para la seguridad y configuración del entorno. *Obtenido de.* Autores

Configurar ALLOWED_HOSTS:

La configuración presente en la Figura 16 es vital para la seguridad. Permite que solo tu dominio de producción acceda a la aplicación.

Figura 16

Configuración de Hosts Permitidos en Django.

```
# Obtiene el dominio de una variable de entorno.
# En la plataforma de despliegue, deberás configurar la variable DEPLOYMENT_HOST con el dominio que te asignen
ALLOWED_HOSTS = []
DEPLOYMENT_HOST = os.environ.get('DEPLOYMENT_HOST')
if DEPLOYMENT_HOST:
    ALLOWED_HOSTS.append(DEPLOYMENT_HOST)
```

Nota: Código que lee el dominio de despliegue desde una variable de entorno (DEPLOYMENT_HOST) y lo agrega dinámicamente a la lista de hosts permitidos.

Obtenido de. Autores.

Configurar la Base de Datos (PostgreSQL):

Reemplaza la configuración de DATABASES para que use PostgreSQL. La URL de conexión se leerá de una variable de entorno. Ejemplo visual Figura 17.

Figura 17

Configuración de Base de Datos en Django

```
import dj_database_url

DATABASES = {
    'default': dj_database_url.config(
        # La variable de entorno DATABASE_URL será proporcionada por la plataforma de despliegue.
        default=f"sqlite:///{BASE_DIR / 'db.sqlite3'}",
        conn_max_age=600
    )
}
```

Notas. Código que configura la base de datos mediante dj_database_url, leyendo la URL desde una variable de entorno (proporcionada por la plataforma de despliegue). *Obtenido de.* Autores

Configurar Archivos Estáticos con WhiteNoise (Recomendado):

Esto es necesario para que el panel de administrador de Django (/admin) funcione correctamente. Añade whitenoise.middleware.WhiteNoiseMiddleware a tu lista de **MIDDLEWARE**, justo después de SecurityMiddleware, como se muestra en la Figura 18:

Figura 18

Configuración de Middleware para Archivos Estáticos.

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    # ... el resto de tus middlewares
]
```

Nota: Fragmento de código que muestra la configuración de middlewares en Django, incluyendo SecurityMiddleware y WhiteNoiseMiddleware. *Obtenido de.* Autores.

Al final del archivo, añade la configuración para los archivos estáticos, como la Figura 19:

Figura 19

Configuración de Archivos Estáticos en Django

```
# Directorio donde `collectstatic` reunirá todos los archivos estáticos.
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATIC_URL = '/static/'
# Almacenamiento para los archivos estáticos
STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

Nota. Código que define el directorio de archivos estáticos (STATIC_ROOT), URL (STATIC_URL) y el backend de almacenamiento (WhiteNoise). *Obtenido de.* Autores.

Configurar django-channels con Redis:

Reemplaza la configuración de CHANNEL_LAYERS como esta en la Figura 20 para que use Redis en producción.

Figura 20

Configuración de Capas de Canales en Django

```
CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            # La variable de entorno REDIS_URL será proporcionada por la plataforma.
            "hosts": [os.environ.get('REDIS_URL', 'redis://localhost:6379')],
        },
    },
}
```

Nota. Configuración de Django Channels usando Redis como backend para WebSockets y comunicación en tiempo real. Lee la URL de Redis desde una variable de entorno con fallback a localhost:6379. *Obtenido de.* Autores.

Crear un Archivo .env para Desarrollo Local (Opcional)

En la raíz de tu proyecto, crea un archivo llamado .env (y añádelo a tu .gitignore). Este archivo te permitirá simular las variables de entorno en tu máquina local, ejemplo Figura 21.

Figura 21

Archivo de Configuración de Variables de Entorno

```
# .env
SECRET_KEY=tu-clave-secreta-local
DEBUG=True
# No necesitas DATABASE_URL si usas sqlite localmente
# No necesitas REDIS_URL si usas InMemoryChannelLayer localmente
```

Nota: Ejemplo de archivo .env para desarrollo local con SECRET_KEY y modo DEBUG activado, incluyendo comentarios sobre variables opcionales para base de datos y Redis *Obtenido de.* Autores.

Configuración en la Plataforma de Despliegue

Estos son los pasos generales que seguirías en la interfaz de la plataforma de despliegue.

Crear los Servicios.

Necesitarás crear tres servicios. En primer lugar, una base de datos PostgreSQL, la cual debes generar desde el panel de la plataforma; una vez creada, este te proporcionará una URL de conexión interna. En segundo lugar, un servicio de Redis, que también se crea

desde el panel y te entregará una URL específica para Redis. Finalmente, un servicio web (Web Service) destinado a alojar la aplicación Django.

Configurar el Servicio Web.

Conectar el repositorio: vincula tu proyecto desde GitHub, GitLab o similar. En la configuración básica selecciona Python como entorno de ejecución y main (o tu rama activa); luego introduce el comando de build: pip install -r requirements.txt && python manage.py collectstatic --no-input && python manage.py migrate. El primer fragmento instala las dependencias, el segundo agrupa los archivos estáticos para que WhiteNoise los sirva y el tercero aplica las migraciones en la base de datos de producción. Como comando de inicio usa: gunicorn backend.wsgi:application.

Para las variables de entorno accede a la sección “Environment” y define: SECRET_KEY con una clave segura generada para producción; DATABASE_URL con la cadena de conexión interna de PostgreSQL; REDIS_URL con la dirección del servicio Redis; DEPLOYMENT_HOST con el dominio asignado (p. ej. hermez-backend.onrender.com); PYTHON_VERSION: 3.13.2; CLERK_WEBHOOK_SIGNING_SECRET con el valor proporcionado por Clerk; y CORS_ALLOWED_ORIGINS con la URL de tu frontend (p. ej. https://mi-frontend.vercel.app).

Configurar el Trabajador de Fondo (Background Worker).

Crea un nuevo “Background Worker” asociado a la misma aplicación; heredará la configuración de build y las variables de entorno definidas en el Servicio Web. Como comando de inicio establece: python manage.py runworker -v 2, orden que instruye a django-channels para que escuche los mensajes en la capa de Redis y ejecute los consumidores correspondientes.

Despliegue Final.

Una vez que todo esté configurado, guarda los cambios. La plataforma debería empezar a construir y desplegar tus servicios automáticamente. Si todo va bien, tu API estará en línea en la URL proporcionada, y los WebSockets estarán funcionando a través del trabajador de fondo.

Guía de Despliegue del Proyecto Hermez (Frontend).

Se describe el paso a paso para llevar a producción el frontend de Hermez: una aplicación web desarrollada con Astro 5.15.3 como framework principal, React 19.2.0 como librería de interfaz, Clerk para autenticación, TailwindCSS 4.1.16 para estilos y Bun como gestor de paquetes.

Requerimientos Previos

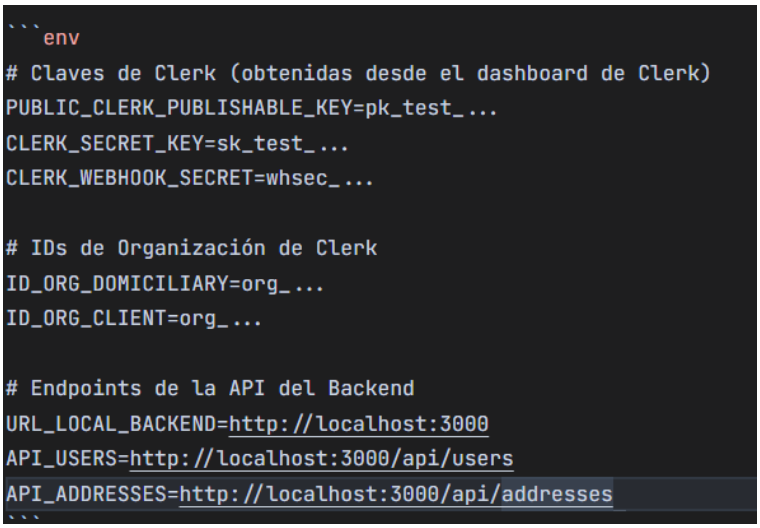
Asegúrate de tener instalado Node.js versión 20.x o superior y Bun versión 1.x o superior; este último puede obtenerse desde la web oficial https.

Configuración de Variables de Entorno

Esta parte permite que se comporte adecuadamente, es aquí donde se ponen las llaves privadas o constantes que dependen de donde se ejecuta, para darse una idea de las que necesita el proyecto hay un archivo en la raíz del proyecto frontend llamado **.env.template** el cual es nombre de las variables y donde se pueden obtener, variables de entorno para la versión actual de este proyecto se pueden observar en la Figura 22.

Figura 22

Archivo de Configuración de Variables de Entorno para Autenticación y API.



Nota. Fragmento del archivo .env con las claves de autenticación de Clerk (PUBLIC_CLERK_PUBLISHABLE_KEY, CLERK_SECRET_KEY, CLERK_WEBHOOK_SECRET), identificadores de organización para domiciliarios y clientes, y endpoints de la API backend local. *Obtenido des.* Autores

Construcción del Proyecto

Una vez definidas las variables de entorno en el entorno de ejecución, ejecuta localmente el comando bun run build; este generará la carpeta .vercel/output con los artefactos de construcción listos para desplegar.

Despliegue

Astro es un framework versátil que permite desplegar tu proyecto en una variedad de plataformas, gracias a su arquitectura de islas y la capacidad de usar adaptadores para diferentes entornos de ejecución (SSR - Server-Side Rendering). Esto significa que, aunque este proyecto está configurado para Vercel, puedes adaptarlo fácilmente a otros servicios.

Para más información sobre los adaptadores de SSR y las opciones de despliegue en Astro, consulta la documentación oficial: [Astro Deployment] (https://docs.astro.build/es/guides/deploy/)

Opción Recomendada: Vercel

El proyecto ya está preconfigurado para desplegarse sin complicaciones en Vercel con el adaptador @astrojs/vercel. Primero, importa tu repositorio Git en el panel de Vercel; después, define las variables de entorno tal como se indica en el paso 8.2.2. Por último, inicia el despliegue: Vercel detectará automáticamente la configuración de Astro y publicará la aplicación.

Opción Genérica (Otros Proveedores)

Si prefieres desplegar en otra plataforma compatible con Node.js —Netlify, Render o un VPS—, el procedimiento general consiste en cambiar el adaptador de Astro. Primero,

reemplaza en astro.config.mjs el adaptador de Vercel por el que corresponda a tu proveedor, por ejemplo @astrojs/netlify o @astrojs/node; esta elección es crucial, ya que indica a Astro cómo empaquetar y ejecutar la aplicación en el entorno destino. Ejemplo en la figura 23

Figura 23

Configuración de Despliegue SSR en Astro

```javascript
// astro.config.mjs
import { defineConfig } from 'astro/config';
import node from '@astrojs/node'; // Ejemplo para
despliegue en Node.js

export default defineConfig({
  output: 'server', // Asegúrate de que el output sea
  'server' para SSR
  adapter: node({ // Usa el adaptador adecuado para tu
  proveedor
    mode: 'standalone'
  }),
  // ... resto de la configuración
});
```

Nota. Código JavaScript de configuración de Astro para despliegue en Node.js con SSR, definiendo el adaptador en modo standalone. *Obtenido de.* Autores.

A continuación, instala el nuevo adaptador con bun add @astrojs/node. Por último, ajusta el comando de inicio según el adaptador y el proveedor: con @astrojs/node podrías usar node ./dist/server/entry.mjs, pero siempre verifica la documentación específica del adaptador y del servicio de hosting para obtener la instrucción exacta.

Scripts Útiles

El archivo package.json define los scripts siguientes: bun run dev, que inicia el servidor de desarrollo en http://localhost:4321; bun run build, que compila el proyecto para producción; bun run preview, que levanta un servidor local para previsualizar la versión final; y bun run lint, que ejecuta ESLint para revisar y corregir el estilo del código.

Trabajos Futuros

Considerando que todo debe evolucionar siempre para mejorar, se entiende que algunas cosas en este caso la aplicación web contiene diversas oportunidades de mejora que podrían implementarse en un futuro con más disponibilidad de recursos económicos y también de tiempo. Estas recomendaciones buscan fortalecer la funcionalidad y competitividad de la plataforma, teniendo en cuenta donde se tenga mayor tiempo, presupuesto y capacidad operativa.

Integración de Métodos de Pagos Electrónicos

Algo que se pensó, a pesar de que no es el método más común de pago en el municipio de Barichara, es la incorporación de una pasarela de pago como mercadopago o PayU, que permitiera procesar transacciones con tarjetas débito o crédito o el pago PSE. Ciertamente para este avance de deberá seguir las normativas internacionales como PCI DSS e implementar un cifrado para datos sensibles, garantizando un flujo seguro de la información y de las transacciones.

Optimización de la Interfaz de Usuario

Se quería también por elección personal y por hacer un diferencia en el producto de este proyecto, que la aplicación contara con más animaciones, que fuese más atractiva para los usuarios, mostrando una diferencia con otras opciones, algo más colombiano, más autóctono, sin embargo el tiempo fue muy limitado, a futuro definitivamente se trabajara en contar con más recursos creativos, refinar la estética de la plataforma, implementar

micro interacciones y quizás aprovechar librerías de animación modernas para obtener una interfaz aún más dinámica, intuitiva y agradable para los usuarios.

Persistencia de Datos

Reconocemos que la elección de SQLite para desarrollo, aunque pragmática debido a su facilidad de integración y bajo requerimiento de configuración, generará deuda técnica al escalar, cuando el volumen de transacciones, usuarios concurrentes y operaciones aumente significativamente. No obstante, el diseño de modelos con abstracciones compatibles con PostgreSQL (usando django-db-geventpool) facilitará la migración futura sin necesidad de reestructurar toda la lógica del negocio ni la arquitectura del sistema.

Seguridad de Dos Pasos PIN

Una mejora que también se proyecta es una verificación en dos pasos para seguridad tanto del cliente como del domiciliario escogido. Esta funcionalidad busca generar mayor confianza y seguridad en el proceso, especialmente en donde la identificación del repartidor puede generar incertidumbre. Así que, una vez confirmada la solicitud del domicilio, el sistema genera un código PIN único que se muestra al cliente y que deberá proporcionar al domiciliario para poder realizar el servicio.

Con estos trabajos futuros se permitirá que la aplicación web continue evolucionando hacia un estado más completo, seguro y competitivo, con la opción de ofrecerse en otros municipios, consolidando la sostenibilidad tecnológica del proyecto.

Integración de Geolocalización en Tiempo Real con Google Maps API

Una mejora estratégica para optimizar la eficiencia operativa y la experiencia de usuario sería la implementación de geolocalización activa mediante Google Maps API. Esta funcionalidad permitiría al domiciliario compartir su ubicación en tiempo real durante el trayecto, mientras el cliente visualiza el progreso del pedido sobre un mapa interactivo dentro de la aplicación. Además, la API calcularía automáticamente rutas óptimas considerando las características topográficas de Barichara (callejones, veredas, desniveles), reduciendo tiempos de entrega estimados hasta en un 30% según estudios de logística urbana en zonas colombianas (Prado et al., 2010). Desde el lado del cliente, esta transparencia visual aumenta la percepción de control y confianza, factores críticos en la satisfacción del servicio (Parasuraman et al., 1988). La integración también habilitaría funciones predictivas como alertas de llegada estimada y notificaciones automáticas cuando el domiciliario está a menos de 500 metros del destino, mejorando significativamente la comunicación sin necesidad de llamadas telefónicas. Técnicamente, esto requiere implementar un servicio de WebSockets persistente, actualizar los modelos de Delivery para almacenar coordenadas geográficas, y gestionar permisos de ubicación según normativas de privacidad colombiana (Ley 1581 de 2012).

Conclusiones

El desarrollo de la aplicación web para la gestión de domicilios en el municipio de Barichara responde a una necesidad concreta de modernización y optimización de entregas de domicilios en una zona geográfica muy turística, con economía basada en negocios familiares y sin presencia de plataformas tecnológicas en cuanto al transporte se refiere. El desarrollo de la aplicación web para gestión de domicilios en Barichara constituye una demostración práctica de cómo la tecnología, cuando se diseña con profundo anclaje en el contexto, puede transformar procesos informales en sistemas eficientes sin perder la esencia de las relaciones comunitarias. A lo largo de este proyecto, se encontraron desafíos que excedieron la mera programación, pensando en las relaciones sociales y técnicas que reconfiguraron la comprensión de la "innovación apropiada" para el municipio. La tecnología verdaderamente inclusiva no impone paradigmas, sino que toma practicas existentes para optimizarlas.

Para esta tarea algo esencial era el contexto, ya que se quería desarrollar algo muy específico, pensando en las experiencias del cliente y del repartidor, así como en la ventaja competitiva, pero que a su vez permitiera luego adaptaciones para poder tenerlo en otros municipios si se requería; se realizó un análisis de los requerimientos en los cuales se compendio las necesidades reales de los usuarios finales, tanto clientes como repartidores y alinear todo esto con los objetivos estratégicos del sistema. Con esto se aseguró que cada componente diseñado respondiera efectivamente a una necesidad concreta, garantizando pertinencia tecnológica y viabilidad operativa del proyecto de la aplicación web.

Al usar los lenguajes y framework definidos durante la planificación. La selección de Astro para el frontend permitió una interfaz ligera e intuitiva, a través de sus islas de interactividad, mientras que Django REST aporto robustez a la lógica, también seguridad y eficiencia. La separación que se hizo a su vez, entre la interfaz gráfica, la lógica y los datos, favoreció la mantenibilidad y escalabilidad estructural del sistema.

La aplicación web fue implementada en un entorno controlado donde se llevaron a cabo pruebas fe integración, funcionales y de rendimiento. Como resultado se permitió verificar que las funcionalidades principales como el registro de solicitudes, la negociación de la tarifa, la actualización de los estados del pedido y el cierre, operaran de manera correcta, estable y segura.

Este proyecto muestra como la innovación tecnológica en territorios no metropolitanos no requiere trascender con algoritmos complejos, sino respetar y potenciar la lógica existente en esos sitios. Requirió observar la dinámica del municipio, su contexto particular, poder coordinar domicilios en algo tan colonial, establecer confianza en estas opciones digitales, la posibilidad de la negociación tal y como se hace en estos momentos de la tarifa. El aporte fue con código, pero también con pensamiento empático en aquellas necesidades enraizadas de las comunidades.

Referencias Bibliográficas

Alidoosti, R., Lago, P., Razavian, M., & Tang, A. (2022). *Ethics in Software Engineering: A Systematic Literature Review*.

Ballestrin, J. B. (2024). Trabajo y alienación en plataformas de reparto: Libertad, dinero, culpa y aburrimiento. *Trabajo y sociedad*, *25*(42), 345-366.

Benarbia, T., & Kyamakya, K. (2022). A Literature Review of Drone-Based Package Delivery Logistics Systems and Their Implementation Feasibility. *Sustainability*, *14*(1), 360. https://doi.org/10.3390/su14010360

Chopra, S. & Meindl, P. (2019). *Supply Chain Management: Strategy, Planning, and Operation* (6.ª ed.). Pearson.

Contador, J. L., Contador, J. C., Carvalho, M. F. H. de, & Neto, P. L. de O. C. (2005). Sistema Kanban Para Fábrica De Tintas. *RAI - Revista de Administração e Inovação*, *2*(1), 68-77.

Cuello, J., & Vittone, J. (2013). *Diseñando apps para móviles*. José Vittone — Javier Cuello.

DANE. (2019). *Censo Nacional*. www.dane.gov.co

David Garlan, Mary Shaw. (1993, diciembre 31). *An Introduction to Software Architecture*. https://www.sei.cmu.edu/library/an-introduction-to-software-architecture/

David Yepes & Diego Restrepo-Tobon. (2016). *Determinantes del nivel de efectivo de las compañías colombianas. Lecturas de Economía*. https://doi.org/10.17533/udea.le.n85a08

Duthie, C., Pocock, T., Curl, A., Clark, E., Norriss, D., Bidwell, S., McKerchar, C., & Crossin, R. (2023). Online on-demand delivery services of food and alcohol: A scoping review of public health impacts. *SSM - Population Health*, *21*, 101349. https://doi.org/10.1016/j.ssmph.2023.101349

Elicabide, L. C. M., Reyes, N. A. M., & Osorio, S. A. (2024). La Precarización laboral de la innovación, el caso de Rappi en Colombia. *Estudios Sociológicos de El Colegio de México*, *42*, 1-18. https://doi.org/10.24201/es.2024v42.e2707

Elmasri, R., & Navathe, S. (2016). *Fundamentals of database systems* (Seventh edition). Pearson.

Gabriel Marcelo Duran Sanchez, Laura Marulanda Grisales. (2024). *Impacto del Marketing de Contenidos en Rappi: Estrategias para Mejorar la Experiencia del Usuario y Aumentar las Ventas* [Javeriana]. https://apidspace.javeriana.edu.co/server/api/core/bitstreams/89adbffe-1860-451f-89c4-00cb3a988652/content

GARCIA-CAPDEVILLA, Diana Alí; VELASQUEZ-VALENCIA, Alexander and HERNANDEZ-GIL, Cristian. (2024). Revista Investig. Desarro. Innov. [Online]. *Turismo de naturaleza y educación ambiental: perspectivas de las políticas públicas en Colombia.*, 220.

Ghajargar, M., Zenezini, G., & Montanaro, T. (2016). *Home delivery services: Innovations and emerging needs*. *49*(12), 1371-1376. Scopus. https://doi.org/10.1016/j.ifacol.2016.07.755

ISO 9241-11:2018(en). (2018). https://www.iso.org. https://www.iso.org/obp/ui/en/#iso:std:iso:9241:-11:ed-2:v1:en

Jiménez Beltrán, J. H. (2015). *Software libre para la gestión de peticiones, quejas, reclamos y felicitaciones con énfasis en la cocreación* [UNAB]. http://hdl.handle.net/20.500.12749/3388

Jimenez et al. (2023). *Ciberseguridad y Seguridad Integral: Un análisis reflexivo sobre el avance normativo en Colombia - ProQuest*. https://www.proquest.com/openview/9ec42ec894a0cf7606f7a0fbedcdfac4/1?pq-origsite=gscholar&cbl=1006393

Just eat. (s. f.). *Con-nect-ing con-sumers to local partners*. https://www.justeattakeaway.com/about/our-story/default.aspx

Kaplan, E. D., & Hegarty, C. J. (Eds.). (2006). *Understanding GPS: Principles and applications* (Second edition). Artech House.

Lashisnky, A. (2018). *La travesía de Uber: Una mirada al interior de una de las empresas de mayor crecimiento y expansión a*. CONECTA.

Obrien y Marakas. (2011). *Management Information Systems*.

Omol, E. J. (2023). Organizational digital transformation: From evolution to future trends. *Digital Transformation and Society*, *3*(3), 240-256. https://doi.org/10.1108/DTS-08-2023-0061

Ore Quiroz, H., Aldana Juárez, W., Salazar Sandoval, C., & Pantoja-Tirado, L. (2021). *Benchmarking como herramienta gerencial en las empresas: Revisión bibliográfica.* 54-65.

Parasuraman et al. (1988). Servqual a multiple-item scale for measuring consumer. *Perception of service quality*, 12-40.

Pedraza-Gutiérrez, S. I., Romero-González, J. F., Güiza-Rodríguez, J. C., & Giraldo-Henao, E. W. (2023). Diseño centrado en el usuario y experiencia de usuario en el sistema de control de acceso de la Universidad Libre. *Revista Científica de Sistemas e Informática*, *3*(1), e426-e426. https://doi.org/10.51252/rcsi.v3i1.426

Periodico el colombiano. (2019). *Domicilios.com se le mete al negocio de droguerías, supermercados y mascotas." El Colombiano.*

Plattner, H., Meinel, C., & Leifer, L. (2011). *Design Thinking: Understand – Improve – Apply.Springer.*

Prado, J., Peinado, J., & Graeml, A. R. (2010). Perception of the benefits of using tracking systems by trucking companies. *BBR - Brazilian Business Review*, *7*(2), 1-18.

Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9.ª ed.). McGraw-Hill Education.

Ricardo Celi, Miguel Bone, Aldo Mora. (2023). *Programación web del frontend al backend*. Grupo AEA.

Rod Stephens. (2015). *Beginning Software Engineering*. John Wiley & Sons.

Semerádová, T., & Weinlich, P. (2022). Moving Businesses Online and Embracing E-Commerce: Impact and Opportunities Caused by COVID-19. En *Https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-7998-8294-7*. IGI Global Scientific Publishing. https://www.igi-global.com/book/moving-businesses-online-embracing-commerce/www.igi-global.com/book/moving-businesses-online-embracing-commerce/269293

Shklar, L., & Rosen, R. (2003). *Web application architecture: Principles, protocols and practices*. Wiley.

Shroff, A., Shah, B. J., & Gajjar, H. (2022). Online food delivery research: A systematic literature review. *International Journal of Contemporary Hospitality Management*, *34*(8), 2852-2883. https://doi.org/10.1108/IJCHM-10-2021-1273

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *Database system concepts* (6th ed). McGraw-Hill.

Sommerville, I. (2016). *Software engineering* (Tenth edition). Pearson.

Turban, E., Pollard, C., & Wood, G. (2018). *Information Technology for Management: On-Demand Strategies for Performance, Growth and Sustainability*. John Wiley & Sons.

Yeja, A. H., & Rubier, J. P. (2016). *Procedimiento para la seguridad del proceso de despliegue de aplicaciones web*. *10*(2).