



**DESARROLLO DE UN SISTEMA INTEGRAL
ORIENTADO
A LA OPTIMIZACIÓN DE RUTAS EN EL
PROCESO DE TRANSPORTE**

Autores:

Daniel Ignacio Lamas y Joaquín Macaroff Pérez

Facultad de Ingeniería, Universidad Católica de Salta

Ingeniería en Informática

Profesor Guía:

Lic. Patricio Alejandro Arredes

2025

Tesis para la obtención del título de:

Ingeniero en Informática

PRESIDENTE DEL TRIBUNAL EVALUADOR

**MIEMBRO DEL TRIBUNAL
EVALUADOR**

**MIEMBRO DEL TRIBUNAL
EVALUADOR**

PROFESOR GUÍA

Lic. Patricio Alejandro Arredes

AUTOR

Daniel Ignacio Lamas

AUTOR

Joaquín Macaroff Pérez

Fecha de exposición

DEDICATORIAS

A mis padres, Ignacio y Celia, por su esfuerzo constante y acompañamiento incondicional a lo largo de mi desarrollo académico y personal.

A mis hermanos, Gisela y Ezequiel, por ser referentes y por motivarme a alcanzar este objetivo personal.

A mis tías, Rosana y María, por confiar plenamente en mis padres y mantener viva la unidad familiar.

A mi prima Lucía, por estar presente con cariño y acompañar con su apoyo a mi familia en esta última etapa.

A mi tío Manuel, por transmitirme siempre su buen humor, especialmente en los momentos difíciles.

A mis amigos, por estar presentes cuando más los necesité, compartiendo momentos importantes.

Y, por último, a mi amigo y compañero Joaquín, que gracias a su paciencia y perseverancia logramos cumplir con este objetivo.

Daniel Ignacio Lamas

A mis padres, Isabel y Ariel, por su incondicional amor, confianza y constante apoyo, no solo durante el transcurso de mi formación académica, sino a lo largo de toda mi vida. ¡Los amo infinito!

A mi hermana Tatiana, por el cariño y aliento desde la distancia para alcanzar este objetivo. ¡Sin dudas pronto vamos a concretar ese viaje a Miami!

A mis compañeros y amigos de la facultad, por la buena onda, por ayudarme siempre y por hacer que los años en la universidad fueran inolvidables.

Y a mi amigo y compañero Daniel. Tu compromiso, dedicación, apoyo y esas ganas de seguir aprendiendo y mejorando cada día fueron clave para que alcanzáramos nuestras metas.

Joaquín Macaroff Pérez

AGRADECIMIENTOS

Queremos expresar nuestro más sincero agradecimiento a la Facultad de Ingeniería de la Universidad Católica de Salta, por brindarnos la formación académica y profesional que hizo posible la realización de este trabajo.

Agradecemos especialmente al Lic. Patricio Alejandro Arredes, nuestro director de tesis, por su acompañamiento, orientación y compromiso durante cada etapa del proyecto. Su profesionalismo y calidad humana han sido un factor clave en la culminación del presente proyecto.

Extendemos nuestro reconocimiento a los docentes y compañeros de carrera que nos apoyaron directa o indirectamente, aportando valiosas sugerencias y conocimientos a lo largo del proceso.

También queremos reconocer el apoyo constante de nuestras familias, cuyo acompañamiento emocional y aliento fueron clave a lo largo de nuestra formación académica.

Finalmente, agradecemos a quienes, de una u otra forma, contribuyeron a que esta tesis se convierta en una realidad.

Daniel Ignacio Lamas y Joaquín Macaroff Pérez

RESUMEN

La presente tesis expone el desarrollo de un sistema informático integral orientado a la optimización de rutas en el proceso de transporte, adaptable a distintos contextos y casos de uso. Con el propósito de mejorar la eficiencia operativa, se diseñó y desarrolló una aplicación web moderna que automatiza parcialmente el proceso de optimización de rutas.

La solución se validó en dos contextos distintos, utilizando para ello un caso de uso representativo en cada uno, con el fin de demostrar su aplicación práctica y efectividad en escenarios reales. En su desarrollo se contemplaron aspectos técnicos clave como escalabilidad, accesibilidad, optimización para motores de búsqueda (SEO), rendimiento, e integración con servicios externos. Además, se empleó un enfoque metodológico híbrido, combinando el modelo de proceso incremental, el marco ágil Kanban y el desarrollo basado en componentes.

La solución incorpora funcionalidades adicionales como estadísticas básicas, notificaciones en tiempo real, asistencia inteligente y programación de tareas asíncronas. Como resultado, se obtuvo un sistema funcional, que conforma un producto mínimo viable (MVP) validable, que cumple satisfactoriamente con los requerimientos definidos y establece las bases técnicas y metodológicas para futuras líneas de investigación y desarrollo del problema en cuestión.

ÍNDICE

Resumen	ix
Índice de tablas	xvii
Índice de figuras	xix
1. Introducción.....	1
1.1 Descripción del problema y su importancia.....	1
1.2 Motivación	2
1.3 Criterios de éxito.....	2
1.4 Pasos a realizar.....	3
2. Estado de la cuestión	5
2.1 Introducción	5
2.2 Antecedentes	5
2.3 Fundamentos teóricos	10
2.3.1 Teoría general del transporte y movilidad.....	10
2.3.2 Optimización de rutas y sus aplicaciones.....	11
2.3.3 Planificador de rutas vs optimizador de rutas	12
2.3.4 Problemas de optimización de rutas.....	12
2.3.5 Desarrollo de software	13
2.3.6 Modelo de proceso incremental	14
2.3.7 Desarrollo basado en componentes	16
2.3.8 Desarrollo de software ágil	18
2.3.9 Kanban	19
2.3.10 Desarrollo web	20
2.3.11 Arquitectura cliente-servidor	23
2.3.12 Protocolos en la capa de transporte y aplicación.....	24
2.3.13 Diseño web adaptable	27
2.3.14 Accesibilidad web	27

2.3.15 Arquitectura monolítica	29
2.3.16 Transferencia de estado representacional (REST)	31
2.3.17 Interfaz de programación de aplicaciones (API).....	32
2.3.18 Servicio web.....	34
2.3.19 Arquitectura peer-to-peer (P2P)	34
2.3.20 Comunicación en tiempo real (RTC)	36
2.3.21 Manejo de tareas asincrónicas y programación de trabajos	36
2.3.22 Modelos de inteligencia artificial.....	37
2.4. Soluciones similares	37
2.4.1 Google Maps	37
2.4.2 Waze.....	38
2.4.3 YPF Ruta.....	38
2.5 Tecnologías requeridas para el tratamiento del problema	39
2.5.1 Lenguajes de programación, frameworks y librerías	39
2.5.2 Sistemas de gestión de paquetes	40
2.5.3 Sistemas de control de versiones.....	41
2.5.4 Autenticación y autorización.....	41
2.5.5 Bases de datos	42
2.5.6 Plataforma como servicio (PAAS).....	43
2.5.7 WebRTC	44
2.5.8 WebSocket	44
2.5.9 Asistentes de inteligencia artificial	45
2.5.10 Sistemas de gestión de tareas programadas	45
2.5.11 Sistemas de gestión de proyectos.....	46
2.6 Temas pendientes de resolución	46
3. Definición del problema	47
3.1 Objetivos.....	47

3.1.1 Objetivos generales	47
3.1.2 Objetivos específicos	47
3.2 Contextualización del problema	48
3.3 Alcance	48
3.3.1 Descripción del alcance del producto.....	48
3.3.2 Entregables	51
3.3.3 Criterios de aceptación	52
3.3.4 Exclusiones del proyecto.....	52
3.3.5 Restricciones o límites del trabajo	53
3.3.6 Supuestos.....	54
3.4 Explicación funcional de los casos de uso	55
3.4.1 Planificación de viajes asistidos	55
3.4.2 Gestión de flotas en servicios a domicilio.....	56
3.5 Alternativas tecnológicas de solución al problema.....	56
3.5.1 Tecnología del lado del cliente.....	56
3.5.2 Tecnología del lado del servidor	57
3.5.3 Plataformas de alojamiento basadas en Git.....	58
3.5.4 Sistemas de gestión de paquetes	58
3.5.5 Sistemas de gestión de proyectos	58
3.5.6 IDEs, editores de textos, terminales de comando y gestores de código.....	58
3.5.7 Herramientas de despliegues y plataformas como servicio	58
4. Solución propuesta	61
4.1 Presentación	61
4.2 Beneficios	62
4.3 Tecnologías utilizadas.....	63
4.3.1 Frontend	63
4.3.2 Backend.....	63

4.3.3 Bases de datos y tareas programadas	64
4.3.4 Servicios y SDK de terceros	65
4.3.5 Despliegue.....	66
4.3.5 Herramientas de gestión.....	67
4.4 Proceso de desarrollo	67
4.4.1 Ciclo de vida del desarrollo.....	67
4.4.2 Enfoque metodológico aplicado.....	73
4.4.3 Flujo de trabajo y división de tareas	74
4.4.4 Recolección de requerimientos	81
4.4.5 Incrementos	86
4.4.6 Historias de usuario.....	86
4.4.7 Correcciones de errores y mejoras menores.....	90
4.4.8 Cronograma del desarrollo.....	91
4.5 Desarrollo informatizado de los casos de uso.....	95
4.5.1 Planificación de viajes asistidos.....	95
4.5.2 Gestión de flotas en servicios a domicilio.....	103
4.6 Modelado de datos	107
4.7 Arquitectura de la aplicación	110
4.8 Cronograma	111
4.9 Evaluación económica-financiera.....	115
4.9.1 Estimación de costos	115
4.9.2 Plan de suscripciones	117
4.9.3 Cálculo de indicadores	117
4.10 Evaluación técnica	119
4.11 Evaluación operativa.....	119
4.12 Evaluación legal.....	119
4.13 Gestión de riesgos	121

4.13.1 Plan gestión de riesgos	121
4.13.2 Identificación de riesgos.....	125
4.13.3 Análisis cualitativo de riesgos.....	128
4.13.4 Estrategia de respuesta a riesgos	128
4.14 Dificultades encontradas y soluciones	131
4.14.1 Configuración para autenticar bibliotecas del cliente	131
4.14.2 Exceso de consumo de los servicios de Google Maps Platform.....	131
4.14.3 Despliegue del sistema en producción	132
4.14.4 Respuesta de error de las transacciones debido a la falta de recursos asociados al sistema y/o la configuración de estas en el ORM	133
4.14.5 Organización de registros en cache.....	133
5. Resultados.....	135
5.1 Resultados obtenidos	135
5.1.1 Generales.....	136
5.1.2 Planificación de viajes asistidos.....	136
5.1.3 Gestión de flotas en servicios a domicilio.....	141
5.2 Calidad de los resultados	147
6. Conclusiones y futuras líneas de trabajo	155
6.1 Conclusiones.....	155
6.2 Futuras líneas de trabajo	156
Bibliografía.....	157
Anexos.....	163
Anexo 1: Conceptos relacionados.....	163
1.1 Paradigmas de programación	163
1.2 Patrones de diseño.....	164
1.3 Principios SOLID.....	165
1.4 Arquitectura Limpia	166

1.5 Aplicaciones web de acuerdo con su estructura y comportamiento.....	167
1.6 Estrategias de renderizado y generación de páginas web	171
Anexo 2: Otras tecnologías implicadas en el desarrollo	174
2.1 Lenguajes	174
2.2 Frontend	175
2.3 Backend.....	175
2.4 Entorno de desarrollo	176
2.5 Herramientas de gestión	176
Anexo 3: Instalación local del sistema.....	176
3.1 Instalación del backend	176
3.2 Instalación del frontend.....	177

ÍNDICE DE TABLAS

Tabla 1. Niveles de conformidad WCAG	29
Tabla 2. División de tareas - Parte 1.....	79
Tabla 3. División de tareas - Parte 2.....	80
Tabla 4. Requerimientos funcionales del sistema	82
Tabla 5. Requerimientos funcionales del sistema: Planificación de viajes asistidos	83
Tabla 6. Requerimientos funcionales del sistema: Gestión de flotas en servicios a domicilio.....	84
Tabla 7. Requerimientos no funcionales del sistema	85
Tabla 8. Incrementos del sistema	86
Tabla 9. Historia de usuario - Log In con Credenciales - Parte 1.....	87
Tabla 10. Historia de usuario - Log In con Credenciales - Parte 2.....	88
Tabla 11. Historia de usuario - Definir criterios básicos de optimización (viajes personales).....	89
Tabla 12. Historia de usuario - Definir criterios avanzados de optimización (viajes personales).....	89
Tabla 13. Historia de usuario - Registrar viaje.....	89
Tabla 14. Historia de usuario - Establecer configuración de optimización (gestión de flotas).....	89
Tabla 15. Historia de usuario - Registrar hoja de ruta.....	90
Tabla 16. Correcciones de errores y mejoras: Backend	91
Tabla 17. Correcciones de errores y mejoras: Frontend.....	91
Tabla 18. Costos en dólares de talento humano	115
Tabla 19. Costos en dólares de infraestructura y servicio	116
Tabla 20. Detalle de costos en dólares de Google Maps Platform.....	116
Tabla 21. Planes de suscripción en dólares	117
Tabla 22. ROI Simple.....	117
Tabla 23. Tasas por considerar para el cálculo de VAN y TIR.....	118
Tabla 24. Proyección de flujo de caja a 5 años – Parte 1	118
Tabla 25. Proyección de flujo de caja a 5 años - Parte 2.....	118
Tabla 26. Cálculo de VAN y TIR.....	119
Tabla 27. Extracto de la Estructura de Desglose de los Riesgos (RBS).....	122
Tabla 28. Definiciones para probabilidad e impactos	123

Tabla 29. Registro de riesgos – Parte 1	126
Tabla 30. Registros de riesgos - Parte 2	127
Tabla 31. Análisis cualitativo de los riesgos identificados.....	128
Tabla 32. Estrategia de respuesta a los riesgos identificados – Parte 1	129
Tabla 33. Estrategias de respuesta a los riesgos identificados – Parte 2	130

ÍNDICE DE FIGURAS

Figura 1. Modelo de proceso incremental [16]	16
Figura 2. Procedimiento detallado del modelo de proceso incremental.....	16
Figura 3. Anatomía básica de una tarjeta Kanban	19
Figura 4. Alternativa de división interna para el desarrollo web.....	22
Figura 5. Arquitectura cliente-servidor	23
Figura 6. Comunicación solicitud-respuesta en la arquitectura cliente-servidor.....	24
Figura 7. Flujo HTTP en la comunicación navegador web – servidor web	25
Figura 8. Representación del flujo HTTPS y certificado TLS	26
Figura 9. Dispositivos comunes en el diseño web adaptable	27
Figura 10. Arquitectura monolítica con organización modular en el backend.....	31
Figura 11. Relación cliente-servidor en un contexto de API REST	33
Figura 12. Composición de solicitudes y respuestas HTTP	33
Figura 13. Estructura de una solicitud y respuesta HTTP en la comunicación con una API REST	33
Figura 14. Arquitectura de red peer-to-peer (P2P)	34
Figura 15. Clasificación de la arquitectura P2P	35
Figura 16. Arquitectura P2P Híbrida.....	35
Figura 17. Modelo de proceso incremental aplicado al desarrollo del sistema.....	71
Figura 18. GitFlow adoptado en el proyecto. Se observan las ramas principales (development, production), las ramas de características (feature/*), la rama de lanzamiento (release/v1.0.0) y los prefijos de confirmación utilizados.	76
Figura 19. Cronograma de desarrollo de módulos del sistema organizado por trimestres	92
Figura 20. Cronograma de desarrollo de módulos del sistema organizado por meses – Parte 1	92
Figura 21. Cronograma de desarrollo de módulos del sistema organizado por meses – Parte 2	93
Figura 22. Cronograma de desarrollo de módulos del sistema organizado por meses – Parte 3	93
Figura 23. Cronograma de ciclo de vida del desarrollo.....	94
Figura 24. Diagrama BPMN del proceso de planificación de viajes asistidos – Parte 1	98
Figura 25. Diagrama BPMN del proceso de planificación de viajes asistidos – Parte 2	99

Figura 26. Diagrama BPMN del subproceso reusable de manejo de intentos de optimización (máximo 5 intentos).....	100
Figura 27. Diagrama BPMN del proceso de asistencia inteligente de guía de viaje – Parte 1	101
Figura 28. Diagrama BPMN del proceso de asistencia inteligente de guía de viaje – Parte 2	102
Figura 29. Diagrama BPMN del proceso de gestión de flotas en servicios a domicilio – Parte 1	105
Figura 30. Diagrama BPMN del proceso de gestión de flotas en servicios a domicilio – Parte 2	106
Figura 31. DER para la planificación de viajes asistidos	108
Figura 32. DER para la gestión de flotas en servicios a domicilio.....	109
Figura 33. Arquitectura de la aplicación	110
Figura 34. Cronograma del proyecto - Parte 1	112
Figura 35. Cronograma del proyecto – Parte 2.....	113
Figura 36. Cronograma del proyecto – Parte 3.....	114
Figura 37. Matriz de probabilidad e impacto con esquema de puntuación [40]	124
Figura 38. Logo de Journeys	135
Figura 39. Pantalla principal del sistema.....	136
Figura 40. Planificación de viajes asistidos – Pantalla de optimización – Criterios básicos	137
Figura 41. Planificación de viajes asistidos – Pantalla de optimización – Criterios avanzados	138
Figura 42. Planificación de viajes asistidos – Pantalla de optimización – Resultados.	139
Figura 43. Planificación de viajes asistidos – Pantalla de inicio	139
Figura 44. Planificación de viajes asistidos – Pantalla de viajes.....	140
Figura 45. Planificación de viajes asistidos – Pantalla de guía de viaje.....	141
Figura 46. Gestión de flotas en servicios a domicilio – Pantalla de optimización – Primera Etapa	142
Figura 47. Gestión de flotas en servicios a domicilio – Pantalla de optimización – Segunda Etapa	143
Figura 48. Gestión de flotas en servicios a domicilio – Pantalla de optimización.....	143
Figura 49. Gestión de flotas en servicios a domicilio – Pantalla de optimización.....	144
Figura 50. Gestión de flotas en servicios a domicilio – Pantalla de panel de control ..	145

Figura 51. Gestión de flotas en servicios a domicilio – Pantalla de hojas de ruta	146
Figura 52. Gestión de flotas en servicios a domicilio – Pantalla detalles de hoja de ruta	146
Figura 53. Informe de rendimiento de la página de optimización en la planificación de viajes asistidos – Parte 1	148
Figura 54. Informe de rendimiento de la página de optimización en la planificación de viajes asistidos – Parte 2	149
Figura 55. Informe de accesibilidad de la página de optimización en la planificación de viajes asistidos – Parte 1	150
Figura 56. Informe de accesibilidad de la página de optimización en la planificación de viajes asistidos – Parte 2	151
Figura 57. Informe de buenas prácticas de la página de optimización en la planificación de viajes asistidos	152
Figura 58. Informe de optimización para motores de búsqueda de la página de optimización en la planificación de viajes asistidos	153
Figura 59. Puntuación obtenida en la página de inicio de planificación de viajes asistidos	153
Figura 60. Puntuación obtenida en la página de viajes de planificación de viajes asistidos	154
Figura 61. Puntuación obtenida en la página de optimización de gestión de flotas en servicios a domicilio	154
Figura 62. Puntuación obtenida en la página de panel de control de gestión de flotas en servicios a domicilio	154
Figura 63. Puntuación obtenida en la página de hojas de ruta de gestión de flotas en servicios a domicilio	154
Figura 64. Funcionamiento de una Progressive Web Application	171
Figura 65. Client Side Rendering	172
Figura 66. Server Side Rendering	173
Figura 67. Static Site Generation	174
Figura 68. Incremental Site Regeneration	174

1. INTRODUCCIÓN

1.1 Descripción del problema y su importancia

La movilidad eficiente es un aspecto fundamental en el proceso de transporte, ya que permite abordar una gran variedad de contextos y casos de uso. En un escenario donde el tiempo, los recursos y la sostenibilidad son factores críticos, lograr dicha eficiencia requiere atravesar un proceso de optimización de rutas, el cual se presenta como una necesidad tanto para las empresas como para las personas.

A pesar de los avances tecnológicos, en países como Argentina aún persiste una carencia de soluciones informáticas integrales que permitan automatizar este proceso y adaptarse a los distintos casos de uso que pueden presentarse en diversos contextos. Las herramientas existentes suelen enfocarse en aspectos aislados —como la distancia, el tiempo o condiciones de tráfico— sin integrar aspectos relevantes como las restricciones de desplazamiento, los costos asociados, la asistencia inteligente, las sugerencias contextuales o preferencias del usuario, entre otros.

La ausencia de esta optimización puede derivar en una serie de consecuencias negativas, como el aumento de accidentes, demoras en las entregas, pérdida de recursos logísticos, mayores costos operativos y un impacto ambiental considerable. Incluso, en casos extremos, puede comprometer la salud humana, la viabilidad económica de una empresa o el equilibrio ecológico de determinadas regiones.

Frente a este escenario, se propone desarrollar una solución informática que permita integrar y automatizar el proceso de optimización de rutas, con la capacidad de adaptarse a múltiples contextos y resolver diversos casos de uso de manera flexible. Si bien se ilustran dos situaciones concretas —una empresarial y otra personal— el objetivo no radica en resolver un problema puntual, sino en demostrar el potencial de una solución genérica, extensible y aplicable a distintas realidades.

En relación con esto, no solo se trata de resolver un problema concreto mediante tecnología, sino también de abrir nuevas posibilidades para el análisis y la adaptación de soluciones personalizadas en función de distintas variables y casos de uso. En consecuencia, se pretende sentar las bases para sistemas más robustos, escalables y sostenibles, capaces de responder a los desafíos actuales y futuros en el proceso de transporte.

1.2 Motivación

En un mundo donde los recursos son limitados y las demandas crecen constantemente, resulta imprescindible desarrollar soluciones tecnológicas que permitan optimizar su uso de forma eficiente y sostenible. Bajo este contexto, abordar un trabajo centrado en la optimización de rutas representa un esfuerzo orientado a equilibrar esta balanza, integrando tecnología, contexto y necesidad.

Una optimización adecuada no solo contribuye a la eficiencia operativa, sino que también permite reducir significativamente los riesgos que puedan surgir para las personas, las empresas involucradas y el medio ambiente. Elegir rutas más cortas, seguras y sostenibles puede tener un impacto tangible en la economía, la seguridad y la ecología.

Desde una perspectiva local, y particularmente en Argentina, son pocas las soluciones informáticas integrales que abordan la optimización de rutas desde un enfoque personalizado, adaptativo y contextualizado. Esto representa una oportunidad de innovación y un desafío para el desarrollo de tecnologías más flexibles.

Analizando el problema, por un lado, se busca garantizar el uso eficiente de los recursos, reducir los riesgos asociados y brindar funcionalidades útiles. Por otro lado, se espera que se colabore en menor medida con el impacto ambiental.

Finalmente, una solución de este estilo permite establecer las bases para concretar una optimización adecuada, independientemente del nivel de personalización que se requiera. Esto posibilitará la apertura a nuevas herramientas que impacten positivamente en las vidas de las personas, con el propósito de contribuir al desarrollo de sistemas más inteligentes, humanos y sostenibles.

1.3 Criterios de éxito

Los aspectos por considerar para el éxito del trabajo son:

1. El sistema informático debe reflejar un avance funcional que cumpla con las características y especificaciones planteadas.
2. El sistema desarrollado debe servir como base para análisis, adaptación y desarrollo de futuras soluciones personalizadas que estén relacionadas con el problema.
3. Cumplir con los objetivos especificados.

4. Uso adecuado de las metodologías y procedimientos propuestos.
5. Emplear tecnología web avanzada en el desarrollo del avance funcional que permita a los usuarios acceder desde cualquier dispositivo con acceso a internet.
6. Demostrar que es posible el desarrollo de un sistema integral que involucre diferentes contextos y casos de uso.

1.4 Pasos a realizar

El desarrollo de la presente tesis se organiza conforme a los siguientes capítulos y títulos:

1. Capítulo 2: Estado de la cuestión
Se presenta el análisis del estado actual del conocimiento relacionado con la optimización de rutas, abarcando antecedentes, fundamentos teóricos, soluciones similares, tecnologías requeridas para el tratamiento del problema y temas pendientes de resolución.
2. Capítulo 3: Definición del problema
Se realiza la formulación del problema, estableciendo los objetivos generales y específicos, contextualizando el problema, definiendo el alcance del producto y presentando las alternativas tecnológicas consideradas para su solución.
3. Capítulo 4: Solución propuesta
Se describe la solución informática desarrollada, incluyendo su presentación, los beneficios esperados, las tecnologías empleadas, el proceso de desarrollo, los entregables, el cronograma del proyecto, las evaluaciones correspondientes, la gestión de riesgos y las dificultades encontradas.
4. Capítulo 5: Resultados
Se exponen los resultados obtenidos a partir de la implementación de la solución propuesta, incorporando las principales pantallas del sistema y analizando la calidad del desarrollo en términos de rendimiento, buenas prácticas de programación, accesibilidad y optimización para motores de búsqueda (SEO).
5. Capítulo 6: Conclusiones y futuras líneas de trabajo
Se presentan las conclusiones generales derivadas del trabajo realizado y se proponen líneas de investigación y desarrollo futuras orientadas a la mejora y expansión del sistema.

6. Bibliografía

Se enumeran las referencias bibliográficas utilizadas como sustento teórico y metodológico a lo largo del trabajo.

7. Anexos

Se incorporan documentos y materiales complementarios que respaldan la información desarrollada en los capítulos anteriores.

2. ESTADO DE LA CUESTIÓN

2.1 Introducción

En este capítulo se presenta un análisis en detalle de la investigación previa y relevante para sustentar el problema y su tratamiento. Se explica de manera breve información perteneciente al tema central, que nos permite tener una base de conocimientos más amplia en la investigación.

En primer lugar, como investigadores nos es de suma utilidad comprender los antecedentes e investigaciones previas relacionados con el problema debido a que nos ayuda a tener un panorama más focalizado sobre los puntos clave del tema principal.

Por otro lado, al presentar los aportes de diferentes autores, los fundamentos teóricos y las limitaciones en las soluciones existentes establecemos la base para construir nuestra propia argumentación. Con el propósito de limitar esta argumentación, se exponen las tecnologías requeridas para tratar el problema y los temas pendientes a resolver.

Por último, a lo largo del capítulo exploramos muchas definiciones obtenidas de diferentes fuentes. También se introducen acotaciones y conceptos desde nuestro enfoque. Evitamos, en lo posible, entrar en profundidad sobre los subtemas relacionados con la problemática para que usted como lector pueda enfocarse en el tema principal.

2.2 Antecedentes

La optimización representa un factor clave que permite obtener la mejor solución posible entre diferentes alternativas independientemente sea el contexto en el que se encuentre involucrada.

Cuando se habla de optimización en el contexto del transporte y la movilidad, por lo general nos referimos a su aplicación en la determinación de rutas basada en distintos criterios, o simplemente optimización de rutas.

En el camino hacia la comprensión de la optimización de rutas se tuvieron en cuenta varias teorías y modelos. Desde la teoría de grafos, hacia un estudio más específico como la teoría de redes o el modelo de transporte, entre otras. De acuerdo con determinadas perspectivas dicho campo se ha alimentado de diversos problemas, algoritmos y técnicas avanzadas de optimización.

A lo largo del tiempo, la optimización de rutas fue experimentando cambios debido a los avances tecnológicos, los requerimientos de la sociedad y las necesidades comerciales impulsadas por las empresas.

Por un lado, el uso de algoritmos avanzados y sistemas de información geográficos nos permiten determinar rutas más precisas. Además, con la inclusión de dispositivos GPS se fomentó a incorporar el seguimiento en tiempo real en los vehículos y así ajustar las optimizaciones sobre la marcha. Por otro lado, las expectativas de los usuarios en cuanto a precisión y preferencia nos llevaron a tener sistemas con mayor personalización en la determinación de rutas. Adicionalmente, las preocupaciones ambientales hicieron que se empiece para tener en cuenta variables como la huella de carbono y la consideración de rutas ecológicas. Por último, la demanda de productos o servicios fue creciendo con la aparición del comercio electrónico, y se vio en la necesidad de adaptar las optimizaciones. Para cada caso, el propio desplazamiento realizado requiere que se tengan en cuenta regulaciones, condiciones y eventos imprevistos en el transcurso.

El boom de la informática y los avances tecnológicos han facilitado el problema a las personas y empresas, proporcionándoles comodidades y asistencia continua. En base a esto, “el desarrollo de nuevas tecnologías partiendo de sistemas inteligentes para la planeación de la operación de sistemas de transporte son herramientas útiles para la optimización de funciones y costos operativos de un sistema de transporte” [1].

Actualmente los medios de transporte son relevantes en la sociedad ya que permiten la circulación de las personas y sus bienes, logrando una integración social que favorece al desarrollo; y que debido a esto con el pasar del tiempo veremos una mejora en la eficiencia de los medios de transporte, los servicios aplicados y el uso de recursos necesarios [2]. En consecuencia, tendremos que lidiar con sistemas que incorporen variables o criterios que se ajusten a las nuevas necesidades que surgen con el tiempo.

Las investigaciones que encontramos se centran en resolver el problema desde los fundamentos teóricos y prácticos aplicando herramientas y técnicas informáticas. A continuación, se presentan los siguientes:

1. El trabajo final “Bidaiatzen. Aplicación web para viajar” de la Universitat Oberta de Catalunya ubicada en Barcelona, España realizado en el año 2017, describe como solucionar el problema en la organización de un viaje grupal mediante la implementación de un planificador de viaje y guías basadas en la experiencia. El

objetivo específico del trabajo fue ayudar a los integrantes del viaje a organizarse y hacer un seguimiento de este en detalle. El autor concluyó que fue posible planificar un viaje, que incluya la definición de tareas, responsables y detalles de este, mediante la implementación del ciclo de vida en cascada y el uso de tecnologías web [3].

2. El proyecto de fin de grado titulado “Desarrollo de un sistema multimodal para la planificación de viajes” de la Universidad Carlos III de Madrid, España realizado en el año 2013, plantea implementar una aplicación web cuya interfaz sea multimodal facilitando el acceso a usuarios con impedimentos físicos o visuales. Dicha temática se llevó a cabo mediante el desarrollo de un planificador de viajes. La autora nos mostró la implementación de un planificador de viajes con interfaz multimodal donde los usuarios pueden consultar diferente información como destino, medios de transportes, alojamiento, atracciones turísticas de interés y condiciones meteorológicas [4].
3. El trabajo de fin de grado “Planificador colaborativo de viajes” de la Universitat Oberta de Catalunya ubicada en Barcelona, España realizado en el año 2015, describe la creación de una aplicación web en la que los usuarios pueden, valiéndose de las opiniones de otros, planificar viajes. Uno de los objetivos pretende que la aplicación sea completamente funcional, utilizable en distintas plataformas y de uso colaborativo. El autor finalmente nos presentó el desarrollo de un proyecto dividido en dos etapas, por una parte, elaboración de la idea y análisis del mercado y, por otra parte, desarrollo y programación. También mostró el uso del stack MEAN en el ámbito de desarrollo web [5].
4. El trabajo de fin de grado “TouristApp: Planificador automático de viajes turísticos” de la Universidad de Alicante ubicada en San Vicente del Raspeig, España realizado en el año 2023, plantea una aplicación web que tiene como fin encontrar los lugares de interés en base a datos proporcionados por el usuario. El objetivo fue desarrollar una aplicación accesible y gratuita que proporcione al usuario una lista de lugares basados en sus características y gustos implementando tanto elementos gráficos como textuales. El autor concluyó que se cumplieron con la mayoría de los objetivos propuestos resaltando la importancia del proceso del proyecto, la implementación de herramientas para procesamiento de lenguaje natural y el potencial uso comercial [6].

5. El trabajo de fin de grado titulado “Aplicación web para la creación de itinerarios turísticos personalizados” de la Universidad de La Laguna ubicada en San Cristóbal de La Laguna, España realizado en el año 2022, plantea la realización de una aplicación web capaz de generar itinerarios turísticos personalizados. El trabajo deja en claro que estos itinerarios están basados en preferencias y puntos de interés, además la aplicación permite gestionar información acerca de turistas. El autor hizo la inclusión de mapas interactivos para que los usuarios puedan usarlos e indicadores para la visualización de diferentes tiempos en el itinerario. En relación con esto obtuvo como conclusión la creación de una aplicación que cumple con el objetivo, dejando en apertura la posibilidad de incorporar mejoras [7].
6. La tesis “Aplicación web utilizando geolocalización en tiempo real y aplicando la teoría de redes, para mejorar el proceso de trazabilidad de rutas y la distribución de insumos” de la Universidad Católica Santo Toribio de Mogrovejo de Perú del año 2021, presenta la implementación de una aplicación web utilizando geolocalización en tiempo real y bajo la filosofía de la teoría de redes para hacer más eficiente el proceso de trazabilidad de rutas y la distribución de insumos de panadería en la empresa DIPROPAN SAC. El autor concluyó que el uso de una aplicación web en combinación con el algoritmo de la ruta más corta resolvió el problema de la trazabilidad de ruta mejorando tiempos, reduciendo gastos y aumentando el nivel de satisfacción de los clientes [2].
7. La tesis “Optimización de rutas para la gestión de pedidos y entregas aplicado a las empresas distribuidoras de productos” de la Universidad Técnica Estatal de Quevedo de Ecuador del año 2016, propone estimular el uso de aplicaciones móviles para obtener una mejora en la distribución de productos, de tal manera que se pueda planificar las rutas según las restricciones de cargas, de horarios y de recursos. El autor concluyó que el uso de una aplicación móvil mejoró el proceso de entregas de productos sobre los pedidos que realizan los clientes a los vendedores [8].
8. La tesis “Asistencia inteligente en la planificación personalizada de la movilidad urbana” de la Universidad Nacional del Centro de la Provincia de Buenos Aires de Argentina del año 2016, plantea la creación de un asistente para el traslado anticipado que permita ayudar en los traslados habituales de un usuario teniendo en cuenta las diferentes eventualidades del tránsito diario garantizando aliviar las

congestiones y el caos. Los autores concluyeron que con el asistente existe una gran mejora debido a que se incrementa el porcentaje de aciertos de predicción de actividades [9].

9. El artículo “Optimización de flotas de vehículos, una herramienta para incrementar la eficiencia” publicado en la revista académica en línea *Universia Business Review* en el año 2007, analiza las aplicaciones y beneficios de la optimización de flotas y expone la experiencia de IBM en este tipo de proyectos. Los autores concluyeron que para el éxito de este tipo de proyectos a veces es necesario cambiar los procesos de negocio para obtener así el máximo beneficio de la optimización. Debido a esto, la información obtenida tiene que ser filtrada y utilizada de forma óptima para detectar anomalías o desviaciones de la planificación. Por otro lado, resaltaron la importancia de contar con un socio tecnológico que no solo tenga el conocimiento experto para llevar a cabo este tipo de solución, sino que disponga de la capacidad para efectuar el despliegue en las áreas de aprovisionamiento, logística e instalación de dispositivos y sistemas [10].

Estos antecedentes especifican los criterios considerados en la optimización, así como diferentes características adicionales. A su vez, las implementaciones se llevaron a cabo mediante estrategias y tecnologías informáticas, que van desde un desarrollo teórico-conceptual o de proyecto, hasta la programación y el desarrollo efectivo de un software.

En cada caso, la optimización de rutas se aborda desde una perspectiva diferente —planificación de viajes, turismo, gestión de flotas o distribución y logística—, lo cual es beneficioso, ya que ayuda a entender el problema no solo a nivel general, sino también desde un contexto en particular.

Asimismo, resulta relevante destacar el papel que cumple la tecnología en el desarrollo de sistemas informáticos de este estilo —específicamente en el desarrollo de sistemas web—, ya que la inclusión de nuevas tendencias como la inteligencia artificial, la programación de tareas, la accesibilidad, la optimización para los motores de búsqueda, entre otras, contribuyen significativamente a la mejora funcional y estratégica de las soluciones desarrolladas.

2.3 Fundamentos teóricos

2.3.1 Teoría general del transporte y movilidad

El transporte es el desplazamiento de personas o bienes desde un lugar a otro, utilizando algún medio y modo para concretarlo. En sentido literal, facilita la movilidad y dota accesibilidad a diversas regiones [11].

Al estudiar el transporte necesitamos hacer énfasis en los modos en los que esté puede ser realizado. Estas modalidades nos permiten definir la siguiente clasificación de transporte:

1. De acuerdo con el objeto transportado: si en el desplazamiento se ven involucradas personas y sus equipajes, se conoce como transporte de viajeros. Por otra parte, si lo que está involucrado tiene que ver con objetos o mercancías sean de la clase que sean, se conoce como transporte de mercancías.
2. De acuerdo con el ámbito territorial: si se presta en el exterior se denomina transporte internacional. En caso de que se realice dentro del mismo país, se conoce como transporte nacional.
3. De acuerdo con el modo de transporte: dependiendo del transporte que se utilice, puede ser transporte terrestre (carretera, cable y ferrocarril), aéreo o acuático (fluvial, lacustre y marítimo). En caso de que se utilice y coordine los distintos modos de transportes mencionados puede ser multimodal, intermodal o combinado.
4. De acuerdo con su naturaleza: si se realiza por un tercero e incluye un precio, se conoce como transporte de servicio público. En caso de ser realizado de manera independiente para satisfacer necesidades particulares o con fines individuales, se denomina transporte privado.

La movilidad hace referencia a todos los desplazamientos, ya sea de personas y bienes, que se efectúan en un espacio físico y que involucran distintos sistemas de transporte incluyendo el simple caminar o andar en bicicleta. En su esencia el objetivo es el de salvar la distancia que nos separa de nuestro destino facilitando así la accesibilidad a determinados lugares [12].

Si bien ambos conceptos son parecidos, cabe aclarar que el transporte es el medio y modo que utilizamos para movernos, mientras que la movilidad es la capacidad de las personas para moverse de un lugar a otro.

En el marco de esta tesis, hacemos alusión al transporte terrestre siendo más específico al transporte por carreteras tanto de mercancías como de viajeros en un ámbito nacional con naturaleza pública o privada, dando por supuesto que la capacidad de movimiento se ve limitada por diversos criterios.

2.3.2 Optimización de rutas y sus aplicaciones

Desde un punto de vista abstracto se entiende por optimización de rutas a todas aquellas acciones necesarias, en un contexto de transporte o movilidad, que contribuyen en términos de calidad.

En el ámbito empresarial la optimización de rutas se involucra con el aspecto estratégico, táctico y operativo con el propósito de gestionar la toma de decisiones en la planificación y gestión de rutas cortas y baratas.

Por consiguiente, la optimización de rutas es el proceso de encontrar el mejor camino o trayecto teniendo en cuenta diversos criterios. El propósito aquí es reducir costos y aumentar la eficiencia.

El concepto de optimización de rutas se aplica a una variedad de ámbitos en especial a aquellos que involucran situaciones en las que se requiera planificar y gestionar de manera óptima el desplazamiento.

A continuación, se mencionan algunas de sus aplicaciones:

1. Gestión de flotas: para incrementar la eficiencia en todos los procesos empresariales en los que participen los vehículos.
2. Planificación de viajes: para gestionar de forma eficiente un trayecto con diversas recomendaciones y optimizaciones cuando se requiera una movilización por turismo, negocios u otra necesidad.
3. Servicios de emergencia: para identificar caminos críticos y llegar rápido a una ubicación específica.
4. Servicios de entrega de paquetería o correo: para proporcionar entregas rápidas y precisas de paquetes.
5. Servicios de recolección de residuos: para armar un circuito de recorrido de forma eficiente.
6. Distribución de productos: para realizar entregas de los productos en negocios, distribuidoras o consumidores finales.

7. Transporte público: para armar un circuito de viaje optimizando horarios y rutas del recorrido en base a los usuarios.

Además de las mencionadas, existen múltiples aplicaciones de la optimización de rutas. Cada una de ellas varía según el contexto en el que se aplique y puede dar lugar a distintos casos de uso en función de necesidades específicas. Por ejemplo, en un contexto de gestión de flotas, es posible encontrar casos aplicados a servicios a domicilio, así como también a operaciones de entrega y retiro de productos.

2.3.3 Planificador de rutas vs optimizador de rutas

La optimización se refiere a un proceso de encontrar la mejor solución entre diferentes alternativas ya sea minimizando o maximizando nuestro objetivo de acuerdo con diferentes limitaciones, por otro lado, la planificación hace énfasis en la definición de actividades y estrategias para lograr ese objetivo, por lo que en el afán de llevar a cabo una herramienta que trabaje con estos conceptos es esencial diferenciar entre planificador de rutas y optimizador de rutas.

Un planificador de rutas se centra en determinar la ruta de acuerdo con lugares de interés o paradas con el objetivo de unirlos de manera lógica y con una estrategia preestablecida. Por lo general los planificadores de rutas no buscan la optimización en los criterios básicos como distancia o tiempo, sino que buscan armar la secuencia más eficiente de estos puntos a visitar. Por otra parte, un optimizador de rutas va más allá de la determinación de la ruta agregándole rentabilidad y eficiencia, ya sea minimizando o maximizando los diversos criterios (distancia, tiempo, costos u otro recurso) basado en nuestro objetivo. Por lo general los optimizadores de rutas requieren el uso de algoritmos y técnicas más avanzadas los cuales pueden llevar a redefinir nuestra ruta planificada asignando y ajustando los recursos de manera adecuada.

2.3.4 Problemas de optimización de rutas

La optimización de rutas es un campo amplio que incluye varios problemas. La mayoría de estos problemas tienen una relación directa con la temática y formulan las condiciones a resolver mediante técnicas, algoritmos y métodos avanzados.

En nuestro caso nos centraremos en explicar los problemas que consideramos de suma relevancia para nuestro trabajo:

1. Problema del agente viajero (TSP): TSP es un problema donde un vendedor necesita encontrar la ruta más corta para un conjunto de ciudades. El planteo formula la condición de que el vendedor debe pasar por cada ciudad una sola vez y finalmente regresar a la ciudad de partida [13].
2. Problema del agente viajero con ventanas de tiempo (TSP con T): en este caso tratamos con una variante del problema TSP que introduce restricciones de tiempo para visitar las ciudades por lo que cada ciudad tiene un intervalo de tiempo durante la cual puede ser visitada. El problema TSP con T sigue teniendo el mismo objetivo que el problema TSP, pero sujeto a restricciones de tiempo. De manera alternativa se podría generalizar este problema para cualquier restricción y no solamente el tiempo.
3. Problema de enrutamiento de vehículos (VRP): es un problema formulado para la optimización de rutas y vehículos asociados a ellas. Este problema fue propuesto por primera vez en 1959 [14] y es importante en la distribución de artículos cuando, mediante una flota de vehículos, se requiere minimizar el costo total al efectuar una entrega. VRP es una extensión de TSP, pero más complicado de resolver. VRP presenta diferentes variantes, por ejemplo, cuando los clientes no sean conocidos o las entregas deban ser realizadas en una ventana de tiempo, entre otras [15].
4. Problema de rutas con múltiples criterios: para cada problema existen sus variantes basadas en diferentes restricciones. Este problema de optimización en lugar de tener en cuenta solo una métrica tiene en cuenta las de nuestra necesidad, es decir, en lugar de considerar solamente el tiempo podremos considerar también distancia, costo y cualquiera de nuestra necesidad.

2.3.5 Desarrollo de software

El software es el conjunto de instrucciones que entregan al usuario la funcionalidad y el desempeño adecuado. En su aspecto contiene estructuras de datos que permiten que estas instrucciones trabajen de forma apropiada la información. Además, en dicha definición queda incluida la documentación que describe la operación y uso del software [16].

Un buen software, debe ser sustentable, confiable y utilizable. Por otra parte, los productos de software se desarrollan para un cliente en particular o para un mercado en general [17].

Existen diferentes dominios de aplicación del software, yendo desde software creado para dar servicios a otros programas como son los sistemas operativos a un software que incluye tendencias actuales como aplicaciones que hacen uso de inteligencia artificial o que están centradas en las redes [16].

Por lo tanto, para llevar a cabo un producto de este estilo se requiere atravesar un proceso conocido como desarrollo de software.

El desarrollo de software consiste en realizar una serie de actividades que permiten crear, diseñar, desplegar y mantener software en relación con ciertos objetivos y necesidades. Cuando el software se realiza para propósitos de negocios específicos o está destinado para uso en dispositivos, no solo por su desarrollador sino también por otros usuarios, y además incluye documentación e información pertinente para su configuración, el proceso pasa a llamarse desarrollo profesional de software [17]. Es interesante hacer énfasis en lo anterior ya que nos plantea las principales diferencias con desarrollar software como aficionado.

Por su parte, al desarrollar software profesional se trata de proporcionar una solución estándar que cubra en cierto sentido las características básicas en un ámbito específico. Suele pasar que en ciertas ocasiones esta solución tiene que adaptarse de acuerdo con los procesos de negocio de la empresa o simples solicitudes del cliente en cuestión, es aquí donde aparece el desarrollo de software a medida para solventar dicha situación. El desarrollo de software a medida trata que el software sea lo más parecido y preciso posible con las necesidades de nuestro cliente, proporcionando funcionalidades únicas o adaptando funcionalidades preexistentes.

2.3.6 Modelo de proceso incremental

Cuando trabajamos en la construcción de un software es importante seguir una serie de actividades predecibles para obtener a tiempo un resultado de alta calidad. Esto se conoce como proceso del software. Además, el software recorre una serie de estados sufriendo transformaciones hasta llegar al producto final, este conjunto de estados se conoce como ciclo de vida del software [16].

Existen diversos modelos de proceso que dependen del software que se esté por elaborar. De esta manera un modelo que puede ser apropiado para crear un sitio web puede que no sea el conveniente para la creación de un sistema operativo. En relación con esto haremos énfasis en el modelo de proceso incremental el cual se usa para complementar el proceso de desarrollo.

“El modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades. Cada secuencia lineal produce incrementos de software susceptibles de entregarse de manera parecida a los incrementos producidos en un flujo de proceso evolutivo” [16, p. 35].

Cuando empleamos este modelo frecuentemente el primer incremento se centra en las funcionalidades más importantes que incorporará el producto. En otras palabras, se atienden los requisitos fundamentales sin tener en cuenta cualidades suplementarias. Este primer incremento se lo suele llamar producto fundamental o en algunos casos producto mínimo viable.

Una vez realizado el producto fundamental, el cliente lo utiliza y evalúa detalladamente. En relación con esta experiencia el equipo de trabajo formula un plan para atacar el próximo incremento. Este plan implica ajustar el producto fundamental en base a la retroalimentación proporcionada por el cliente, así como incorporar a la entrega características adicionales potenciando la funcionalidad. Este proceso se repite al entregar cada incremento, hasta concluir con el producto final.

“El modelo de proceso incremental se centra en que en cada incremento se entrega un producto que ya opera” [16, p. 36]. En base a esto, cada incremento representa una versión básica del producto final con capacidad suficiente como para servirle al usuario y permitirle evaluar. El modelo de proceso incremental en relación con el calendario del proyecto y las funcionalidades del software se muestran en la Figura 1. La Figura 2 muestra el procedimiento detallado del modelo incremental desde que se lleva a cabo el relevamiento de datos inicial hasta que se completa el sistema final.

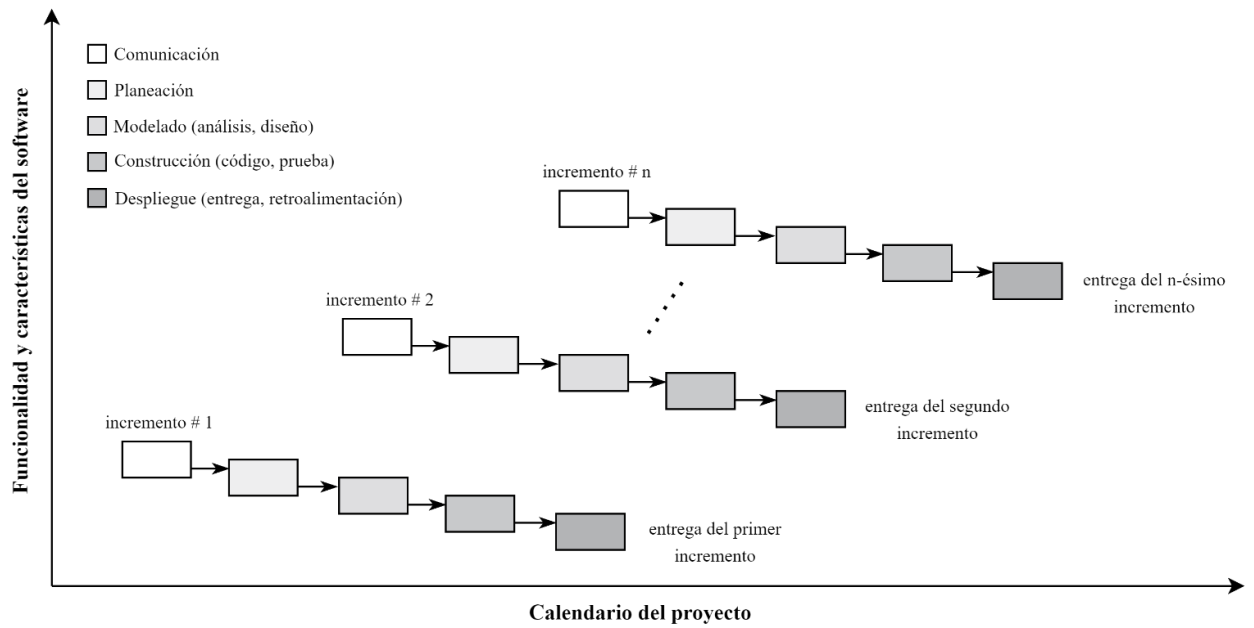


Figura 1. Modelo de proceso incremental [16]

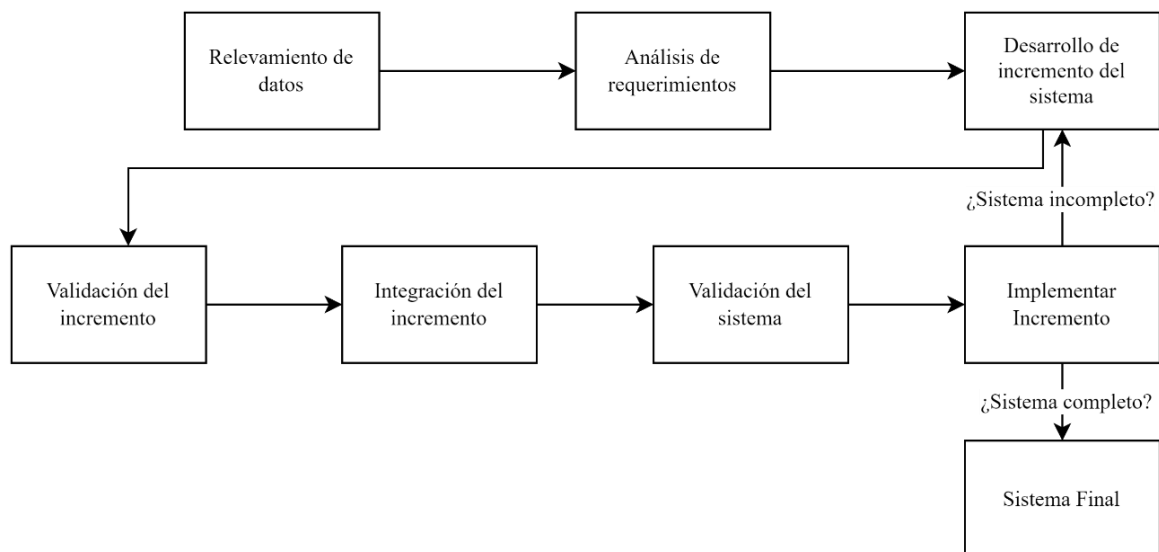


Figura 2. Procedimiento detallado del modelo de proceso incremental

2.3.7 Desarrollo basado en componentes

La reutilización de software es un proceso que implica el uso recurrente de activos de software en cada una de las etapas del proceso de construcción del software [18].

Por lo general cuando se construye software durante su desarrollo existe un gran porcentaje de código fuente utilizado que nos servirá para otro software similar. En

detalle, existen algoritmos, diseños, esquemas, patrones, requerimientos, arquitecturas y casos de prueba que pueden ser empleados de manera recurrente en otro software. Cada elemento que pueda ser reutilizado se denomina como activo de software reutilizable.

A medida que se fue desarrollando software, como resultado de reducir tiempo y costos en el proceso manteniendo la calidad del producto, surgió un nuevo enfoque de desarrollo el cual estaba basado en la reutilización de activos. Estos activos utilizados para desarrollar software fueron denominados componentes de software.

Un componente software es una parte casi independiente y reemplazable de un software que puede ser reutilizado en otro software manteniendo su propia funcionalidad y soportando adaptaciones. En términos bibliográficos se define como “una parte modular, desplegable y sustituible de un sistema, que incluye la implantación y expone un conjunto de interfaces” [16, p. 235].

Como requisitos adicionales, un componente de software debe tener una identificación clara, características genéricas y cuasi independencia de otros componentes para garantizar fácil accesibilidad y reutilización. También debe adaptarse a una interfaz para que cuando se requiera pueda ser reemplazado por otro componente que cumpla con la misma interfaz. Por último, se le debe realizar mantenimiento continuo para introducir mejoras y garantizar nuevas versiones con optimizaciones y nuevas características. Además, acompañado de todo esto se debe proporcionar documentación adecuada que facilite uso e integración del componente.

Los componentes tienen dos interfaces relacionadas:

1. La interfaz denominada proporciona que define los servicios que ofrece el componente.
2. La interfaz denominada requiere que especifica que servicios deben ofrecer otros componentes en el sistema para que un componente opere correctamente [17].

El desarrollo de software basado en componentes es un enfoque del desarrollo de software que consiste en realizar software a partir de componentes prefabricados utilizando las buenas prácticas de la ingeniería de software. El objetivo es proporcionar ventajas durante el desarrollo disminuyendo el tiempo y costos en los procesos.

Existen dos procesos de desarrollo de software basado en componentes. El primer proceso habla del desarrollo para reutilización, es decir, desarrollar componentes o servicios que

se reutilizaran en otras aplicaciones que por lo regular implica la generalización de los componentes existentes. El segundo proceso trata del desarrollo con reutilización, es decir, desarrollar nuevas aplicaciones utilizando los componentes y servicios que ya existen [17].

Estos procesos tienen diferentes objetivos. Por un lado, cuando hablamos del desarrollo para reutilización intentamos crear uno o más componentes que sean reutilizables, por lo que conocemos el catálogo de componentes con los que trabajaremos. Por el contrario, en el desarrollo con reutilización, desconocemos este catálogo ya que buscamos producir un sistema basado en los mismos.

2.3.8 Desarrollo de software ágil

El desarrollo de software ágil es un enfoque del desarrollo de software que está basado en la filosofía de satisfacer al cliente mediante la entrega de resultados de forma incremental implementando métodos informales y sencillez en el desarrollo [16].

Según lo planteado en el manifiesto para desarrollo ágil de software en el año 2001 por un grupo de profesionales, la agilidad propone:

1. construir un equipo que en base a sus necesidades cree su propio entorno. Según estos profesionales las personas son el eje principal para el éxito de un proyecto por lo que hay que valorar a las personas y las interacciones entre ellas por sobre los procesos y las herramientas.
2. producir documentación cuando sea necesaria de forma inmediata, por ejemplo, cuando se tomen decisiones importantes. Se debería valorar el funcionamiento del software por sobre la documentación detallada.
3. interacción constante entre el cliente y el equipo de desarrollo ya que nos permitirá avanzar en el proyecto y asegurar nuestro éxito, lo cual implica valorar la colaboración con el cliente por sobre la negociación de contratos.
4. planificación flexible y abierta con el propósito de responder a cambios que puedan surgir a lo largo del proyecto. Para estos profesionales no hay que apegarse estrictamente a un plan y valorar más la respuesta a los cambios [19].

Uno de los propósitos del desarrollo de software ágil es introducir flexibilidad, colaboración y entrega continua de software.

Existen muchas metodologías o marcos para desarrollo de software ágil, para nuestro propósito nos centraremos en Kanban una subcategoría de gestión ágil de proyectos.

2.3.9 Kanban

Kanban es un método visual de gestión de proyectos implementado mediante tableros que permite a los equipos visualizar el flujo y carga de trabajo [20].

Tanto el flujo de trabajo como la carga de trabajo se visualizan en una barra de tareas a través de tarjetas en un tablero. Estas tarjetas se distribuyen en una serie de columnas, que, por lo general, representan una etapa del trabajo. El tablero Kanban más básico puede ser representado por las siguientes columnas: Trabajo Pendientes, En Curso y Finalizado.

Hacer una división de las tareas en columnas de acuerdo con el progreso del proyecto permite visualizar el flujo de trabajo, evitar que se nos olvide alguna tarea e inspeccionar el trayecto dando la posibilidad de adaptarse.

Kanban trabaja con tableros físicos o digitales construido en base a columnas que permiten una visualización clara de las tareas a realizar, en qué etapa esta cada una y quien es el encargado de estas. En dicho tablero se ubican tarjetas donde cada una de estas representa una tarea. La anatomía básica de una tarjeta se representa en la Figura 3.

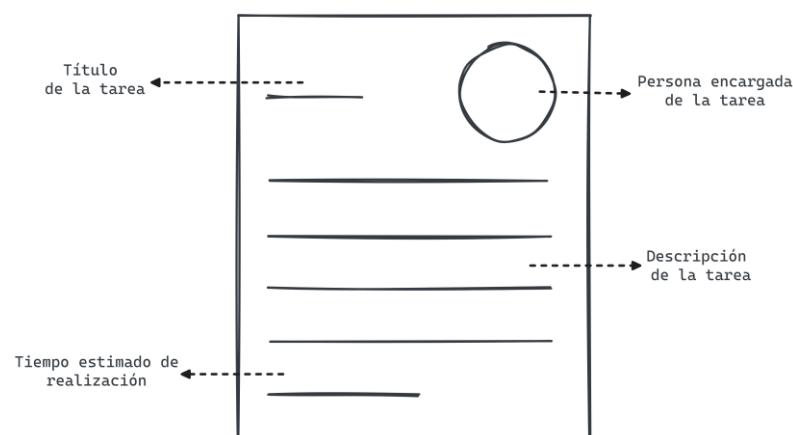


Figura 3. Anatomía básica de una tarjeta Kanban

Esta subcategoría de gestión ágil se puede implementar en cualquier flujo de trabajo siendo lo suficientemente flexible como para adaptarse a procesos y actividades del equipo que forma parte del proyecto. Al no tener roles integrados evita que modifiquemos de forma radical nuestro sistema de trabajo. Por un lado, fomenta el cambio progresivo con el propósito de introducir mejoras continuas en el proyecto, es decir, en lugar de buscar una evolución repentina se busca que los procesos evolucionen con el tiempo. Por

otro lado, promueve el cambio desde diferentes direcciones introduciendo participación continua del equipo y proponiendo nuevas formas para evolucionar los procesos.

Con Kanban podemos visualizar como el trabajo avanza a través de las etapas permitiéndonos obtener información en tiempo real y valorar el estado del proyecto. Es esencial entender que este marco está pensado para la entrega temprana de resultados lo que implica que las tareas deben moverse rápidamente de una columna a otra. Debemos tener en cuenta que, aunque no existe un requisito sobre cuantas tareas pueden estar en progreso hay que limitar el trabajo para animar al equipo a terminar las tareas y evitar acumulaciones por retrasos. La mejor manera para limitar el trabajo es implementando gestión en el flujo de manera que se pueda controlar el tiempo predestinado y reducir el tiempo de entrega. Para lograr esta gestión es importante que se hayan establecido e innovado convenciones con participación de todos los integrantes del equipo. Además, con el propósito de introducir mejoras continuas en el proceso es importante recopilar comentarios ya sea para obtener retroalimentación de la calidad y eficiencia de la solución por parte del cliente o para medir la satisfacción del equipo sobre la ejecución del proceso. Por último, hay que tener en cuenta que Kanban puede funcionar con otros sistemas, marcos o metodologías del desarrollo de software permitiéndonos experimentar y desarrollar nuestros procesos.

Combinar este marco para desarrollo de software ágil con alguna metodología de desarrollo de software nos permitirá establecer un procedimiento para la gestión del proyecto basado en principios de la agilidad y etapas de un modelo en concreto.

2.3.10 Desarrollo web

El desarrollo web es una subdisciplina del desarrollo de software que consiste en crear y mantener aplicaciones web funcionales a Internet teniendo en cuenta un diseño y una arquitectura sofisticada. Cada aplicación cuenta con un dominio único que la distingue de las demás en la red informática mundial.

Al crear una aplicación web, hay tres principios fundamentales para tener en cuenta:

1. Desde el punto de vista del cliente, la aplicación debe ser simple, estéticamente agradable y abordar la mayoría de sus problemas.
2. Teniendo en cuenta el aspecto empresarial, la aplicación debe estar alineada con su producto o mercado.
3. Desde la perspectiva de un ingeniero de software, la aplicación debe ser escalable, funcional y capaz de soportar altas cargas de tráfico.

Haciendo énfasis en el aspecto técnico, una de las formas de llevar a cabo el desarrollo de una aplicación web es realizando la siguiente subdivisión:

1. Frontend: incluye la creación de interfaces intuitivas y agradables, así como el almacenamiento, presentación y actualización eficiente de los datos recibidos desde el backend. Está relacionado con la interfaz que es la capa de presentación de una aplicación web sujeto a todo lo que ve el usuario y que puede ofrecer contenido estático o dinámico, en la mayoría de los casos es una combinación de ambos.
2. Backend: es el responsable de la lógica interna de una aplicación web. Los desarrolladores backend programan la capa empresarial y la capa de datos de una aplicación. Cuando se habla de backend hacemos foco en el servidor y la forma en la que se define el código de programación oculto que le da vida a una aplicación. Este código de programación incluye lo necesario para mantener la accesibilidad, privacidad y seguridad de una aplicación web.

En conclusión, los desarrolladores frontend programan la apariencia de una aplicación, mientras que los desarrolladores backend se centran en los aspectos funcionales del desarrollo de aplicaciones web.

Si ambas partes son realizadas por un mismo programador se dice que esté lleva a cabo un desarrollo full stack, en términos generales, la palabra full stack se utiliza para referirse al conjunto frontend y backend. La Figura 4 presenta un esquema que vincula y muestra la composición de ambas partes [21].

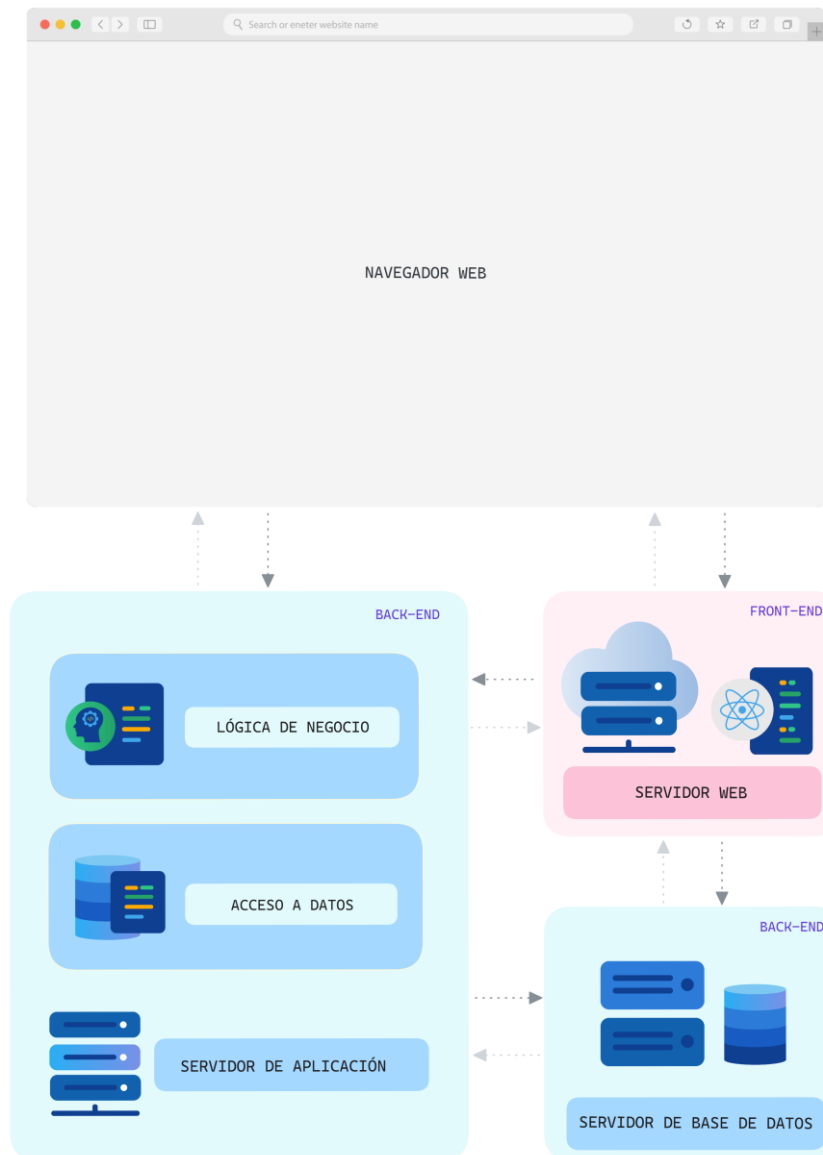


Figura 4. Alternativa de división interna para el desarrollo web

Durante esta sección se mencionaron los conceptos de desarrollo web ligados a las aplicaciones web, ¿pero es lo mismo hablar de una aplicación web, un sitio web y una página web? En el habla cotidiano puede que los usuarios utilicen estos tres términos como equivalentes para referirse a un sistema basado en la web, pero para la informática y el mundo de los programadores no es correcto utilizarlos como equivalentes. Por lo que:

1. Página web: es un documento con diseño y contenido que representa una pantalla de un sitio web o aplicación web.

2. Sitio Web: es un conjunto de páginas web estáticas estructuras en un dominio que por lo general son de uso informativo, tienen interactividad escasa o nula, son menos costosas y requieren un uso minimalista en cuanto a tecnología para su desarrollo.
3. Aplicación Web: es un conjunto de páginas web, pero con alta interactividad y funcionalidades avanzada como la vinculación con servicios de compra, facturación, autenticación e incorporación de nuevas tendencias como la IA, entre otros. Por lo general este estilo de aplicaciones reemplazan a las aplicaciones de escritorio, suelen ser más costosas que los sitios web y requieren el uso tecnología más compleja para su desarrollo.

2.3.11 Arquitectura cliente-servidor

La arquitectura cliente-servidor es un enfoque arquitectónico distribuido compuesto por un proveedor (servidor) y consumidores (clientes). Los clientes se comunican con el servidor para recuperar todos los recursos necesarios para funcionar y el servidor se encarga de facilitarle estos recursos. El cliente solo es una capa de representación de datos que detona acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado [22]. En base a esto, en la Figura 5 podemos ver un esquema que presenta la arquitectura cliente-servidor en un contexto donde hay un proveedor y cuatro consumidores.

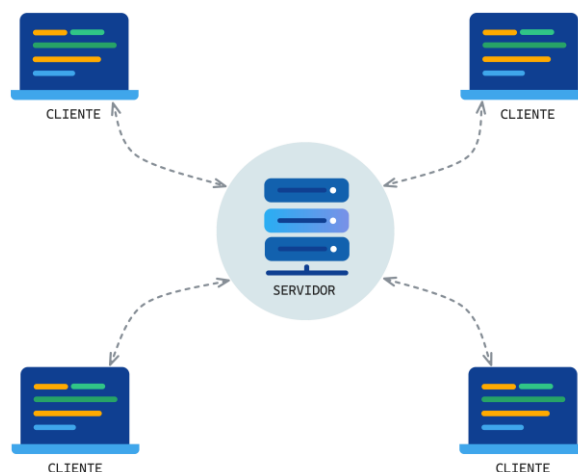


Figura 5. Arquitectura cliente-servidor

El protocolo TCP/IP permite la comunicación continua y bidireccional de tal manera que el cliente puede enviar y recibir datos del servidor mediante la red [22]. La Figura 6 muestra un panorama de esta comunicación.

Esta arquitectura se aplica comúnmente en las aplicaciones web permitiendo dividir las funciones entre cliente-servidor.

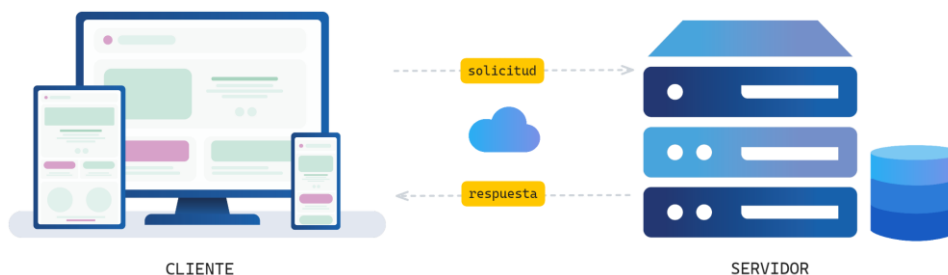


Figura 6. Comunicación solicitud-respuesta en la arquitectura cliente-servidor

¿Cómo es que funciona?

Cuando un usuario accede a la aplicación, el navegador actúa como cliente y solicita recursos al servidor web, que responde enviando los archivos necesarios para renderizar o mostrar la página.

Dependiendo de nuestras necesidades los datos para la representación del contenido de cada página pueden ser obtenidos en el servidor web durante el renderizado o generación de la página o mediante solicitudes posteriores comunicándose directamente con el servidor de aplicación.

Por lo tanto, podemos tener un servidor que aloje el servidor de aplicación y el servidor web o bien servidores distintos que establezcan comunicación con nuestro cliente. Siendo más específico, el cliente vendría a ser el navegador web y el servidor la máquina que almacena, procesa y entrega los recursos solicitados por el navegador. Este último ejemplo no es el único que podríamos considerar ya que todo va a depender de como asignemos el rol de consumidor y proveedor.

Por último, se utilizan protocolos en la capa de aplicación y transporte para la comunicación y transferencia de información entre el cliente y el servidor.

2.3.12 Protocolos en la capa de transporte y aplicación

Tanto la comunicación como la seguridad en las aplicaciones web se ve reflejada mediante la implementación de protocolos en la capa de transporte y aplicación. Estos protocolos permiten la transferencia de información entre un cliente y un servidor garantizando la confidencialidad, integridad y cifrado de datos durante la transmisión, así como la protección contra ataques.

HyperText Transfer Protocol (HTTP) es un protocolo de la capa de aplicación que permite la transferencia de información entre dispositivos conectados a la red, generalmente para la transmisión de documentos hipermedia, como HTML. Se creó para que los navegadores y los servidores web se comuniquen, pero también se puede usar para otros propósitos. Sigue la arquitectura clásica de cliente-servidor, explicada en la sección anterior. Se trata de un protocolo sin estado, lo que significa que el servidor no almacena información entre dos solicitudes [23].

Un flujo típico sobre HTTP implica una maquina cliente que realiza una solicitud a un servidor, que a continuación envía un mensaje de respuesta. La Figura 7 muestra un ejemplo de este flujo donde el navegador web se comunica con el servidor web para obtener recursos.

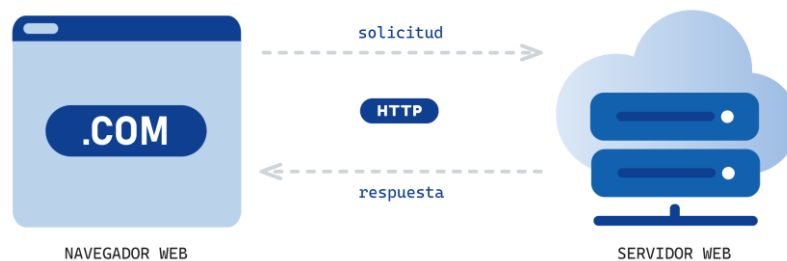


Figura 7. Flujo HTTP en la comunicación navegador web – servidor web

Secure Sockets Layer (SSL) es un protocolo de seguridad de internet de la capa de transporte desarrollado por Netscape en 1995 con el propósito de garantizar la privacidad, la autenticación y la integridad de los datos en las comunicaciones de Internet [24]. Dicho protocolo:

1. Brinda un alto grado de privacidad, cifrando los datos que se transmiten por la web. Por ello, cualquiera que intente interceptar estos datos se encontrará con una mezcla confusa de caracteres, que será muy difícil de descifrar.
2. Asegura que los dispositivos sean los que aparentan utilizando un proceso de autenticación conocido como establecimiento de comunicación.
3. Firma digitalmente los datos para proporcionar integridad y verificar que no se hayan alterado antes de alcanzar al destinatario designado.

Cada una de las diferentes versiones de SSL ofrece una mayor seguridad que la anterior. A partir de 1999 se actualizó para convertirse en el protocolo Transport Layer Security (TLS). Al igual que SSL, Transport Layer Security (TLS) es un protocolo de seguridad

desarrollado para facilitar la privacidad y la seguridad de los datos en las comunicaciones de Internet. Un caso de uso típico de TLS es la encriptación de las comunicaciones entre aplicaciones web y servidores, pero puede emplearse para encriptar otras comunicaciones como el correo electrónico, los mensajes y la voz sobre IP [25].

Al ser SSL el predecesor de TLS, están estrechamente relacionados, es por eso por lo que ambos términos se suelen utilizar indistintamente. De hecho, en la práctica se usa SSL para referirse a TLS o directamente se dice cifrado SSL/TLS. La única diferencia que podemos resaltar es que al ser TLS una tecnología actualizada utiliza algoritmos más robustos y suites de cifrados más seguros [26].

El protocolo SSL se considera obsoleto y en la actualidad dispone de muchas vulnerabilidades conocidas, por lo que se recomienda usar TLS como protocolo de cifrado.

Para que una aplicación o sitio web use TLS, debe tener un certificado TLS instalado en su servidor de origen. Estos certificados son emitidos por una autoridad de certificación para la persona o empresa propietaria de un dominio. El certificado contiene información importante para validar la identidad del servidor [25].

En consecuencia, cuando en la comunicación mediante HTTP se implementa la encriptación mediante el protocolo TLS o SSL (no recomendado) se dice que estamos trabajando con el protocolo HyperText Transport Protocol Secure (HTTPS). Entonces, HTTPS es la versión segura de HTTP. La Figura 8 muestra la implementación del certificado TLS en el servidor web con el objetivo que se trabaje con un flujo HTTPS.

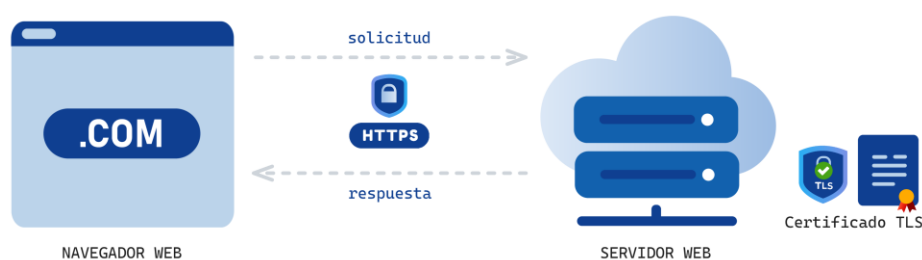


Figura 8. Representación del flujo HTTPS y certificado TLS

Al implementar seguridad en las aplicaciones o sitios web además de las ventajas mencionadas anteriormente, genera confianza en los usuarios y mejora la optimización para los motores de búsqueda ayudando a obtener un mejor posicionamiento en la web.

2.3.13 Diseño web adaptable

El diseño web adaptable, es una técnica de diseño y maquetación web que nos permite hacer que las páginas web de una aplicación web o sitio web sean adaptables a las condiciones del dispositivo, es decir, crear paginas flexibles que se adapten a varios tamaños u orientaciones de la ventana gráfica. Por ende, al implementar diseño web adaptable no es necesario crear varios diseños [27].

El objetivo de la técnica es conseguir que el contenido de cada página web se adapte al tamaño de la pantalla, de modo que el usuario final pueda leer de forma cómoda, sin la necesidad de que tenga que hacer movimientos de scroll.

La Figura 9 muestra los dispositivos más comunes para los que se suele adaptar una página web.



Figura 9. Dispositivos comunes en el diseño web adaptable

2.3.14 Accesibilidad web

La accesibilidad web es “la posibilidad de que un producto o servicio web pueda ser accedido y usado por el mayor número posible de personas, indiferentemente de las limitaciones propias del individuo o de las derivadas del contexto de uso” [28]. Esto significa que los sitios web, las herramientas y las tecnologías deben estar diseñados y desarrollados para que las personas con discapacidad o en un contexto desfavorable puedan usarlos. Lo que se busca es que las personas puedan percibir, comprender, navegar e interactuar con la web, y también que puedan contribuir con ella [29].

Entre las discapacidades que afectan la accesibilidad web están incluidas:

1. Auditivas
2. Cognitivas

3. Neurológicas
4. Físicas
5. Visuales
6. Del habla

“La accesibilidad no solo implica la necesidad de facilitar acceso, sino también la de facilitar el uso” [28]. Esto no solo mejora la inclusión digital, sino que además optimiza la experiencia de usuario general, favorece la optimización para motores de búsqueda (SEO), y extiende el alcance del sistema a una audiencia más amplia. En algunos casos, también puede ser un requisito legal.

La accesibilidad se rige principalmente por los principios establecidos en las Pautas de Accesibilidad para el Contenido Web (WCAG, por sus siglas en inglés), desarrolladas por la Iniciativa de Accesibilidad Web (WAI) del W3C. Los principios son:

1. Perceptible: los usuarios deben poder percibir la información que se presenta.
2. Operable: los usuarios deben poder operar la interfaz.
3. Comprensible: los usuarios deben poder comprender la información, así como el funcionamiento de la interfaz de usuario.
4. Robusto: los usuarios deben poder acceder al contenido a medida que avanzan las tecnologías.

Las WCAG abarcan una amplia gama de recomendaciones para mejorar la accesibilidad del contenido web. Seguir estas pautas hace que el contenido sea más accesible para un mayor número de personas con discapacidad, incluyendo adaptaciones para ceguera y baja visión, sordera y pérdida auditiva, movilidad limitada, discapacidades del habla, fotosensibilidad y combinaciones de estas, así como algunas adaptaciones para dificultades de aprendizaje y limitaciones cognitivas. Sin embargo, no se aborda todas las necesidades de los usuarios con estas discapacidades. Estas pautas abordan la accesibilidad del contenido web en cualquier tipo de dispositivo. Seguir estas pautas también suele hacer que el contenido web sea más usable para los usuarios en general [30].

Las WCAG han tenido varias versiones. La versión más utilizada actualmente es la WCAG 2.1, aunque en 2023 se ha publicado la versión 2.2. Cada versión define niveles de conformidad:

Tabla 1. Niveles de conformidad WCAG

Nivel	Qué significa	Qué exige	Para quien es útil
A	Accesibilidad básica	1. Lo mínimo indispensable. 2. Que el contenido no sea inaccesible.	Persona con discapacidad visual, auditiva o motriz leve.
AA	Accesibilidad estándar	1. Buen contraste entre texto y fondo. 2. Navegación clara. 3. Textos comprensibles.	La mayoría de los usuarios con discapacidades. Es el nivel recomendado.
AAA	Accesibilidad avanzada	1. Alto contraste. 2. Lectura simplificada. 3. Traducción en lengua de señas.	Usuarios con discapacidades severas o necesidades cognitivas específicas.

Adicionalmente, existen legislaciones nacionales e internacionales que adoptan estas pautas, como la Ley 26653 de Argentina, la Sección 508 en Estados Unidos, o el EN 301 549 en la unión europea.

2.3.15 Arquitectura monolítica

La arquitectura monolítica busca crear una aplicación autosuficiente que contenga toda la funcionalidad necesaria para cumplir como solución del problema para el cual fue diseñada [22].

Este estilo arquitectónico, se construye como una sola unidad cohesiva de software compuesta por diferentes componentes que pueden o no estar organizados de manera modular. Aunque su organización pueda ser modular, al momento de compilarse todo se empaqueta como una sola pieza de software. Esto significa que la arquitectura monolítica se define por su despliegue como una unidad única, y no por su organización interna.

Si bien en sus inicios todas las aplicaciones nacían con este estilo arquitectónico, con el tiempo fueron apareciendo nuevas arquitecturas aplicables –como microservicios o en capas– según las necesidades del contexto. A pesar de esto, la realidad es que, hasta el día de hoy, este estilo arquitectónico sigue teniendo un protagonismo muy importante sobre todo en casos donde hay un nivel de complejidad bajo, tenemos un equipo reducido, existen tiempos ajustados de entrega, no se requiere una gran escalabilidad independiente por módulos y no se precisa de una infraestructura distribuida.

Entre las ventajas de la arquitectura encontramos:

1. Al ser una única pieza de software, es fácil de desarrollar y ponerlo en producción.
2. Puede replicarse fácilmente en múltiples servidores, por lo que es fácil de escalar de forma horizontal.
3. Al ser autosuficientes, mitigan gran parte de los errores por lo que hay menos puntos de fallo.
4. Al ser autónomas, permiten el funcionamiento independiente con respecto a otras aplicaciones.
5. Tienen alto rendimiento debido a que la mayor parte del procesamiento lo realizan localmente.
6. Al ser una única pieza de software, es fácil de probar.

Entre las desventajas de la arquitectura tenemos:

1. Al ser una pieza de software, se debe usar un solo Stack tecnológico lo que impide que aprovechemos otras tecnologías.
2. Ante la necesidad de escalar, se debe hacer en toda la aplicación lo que gasta recursos en funcionalidades que tal vez no necesitan ser escaladas.
3. A medida que la aplicación crece se vuelve cada vez más complicado dividir el trabajo sin afectar funcionalidad.
4. Cualquier cambio en la aplicación implicara realizar una compilación de todo y con ello una nueva versión.
5. Si falla el sistema, falla todo.
6. Es fácil caer en malas prácticas de programación y separación de responsabilidades.
7. Cuando el sistema crece puede ser abrumador hacer cambios.

La Figura 10 representa visualmente un ejemplo de la arquitectura monolítica organizada de forma modular en el backend.

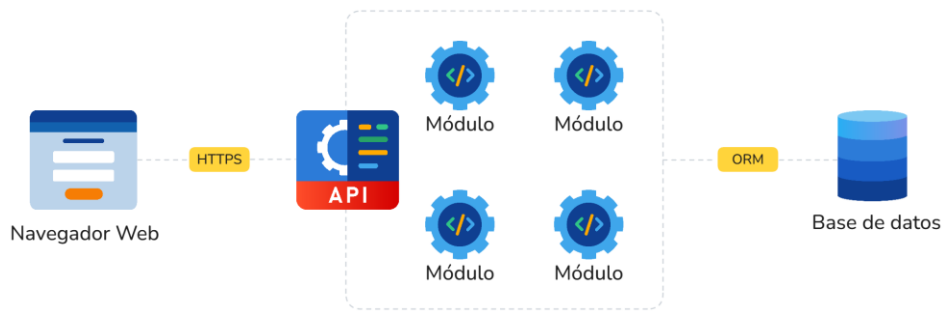


Figura 10. Arquitectura monolítica con organización modular en el backend

2.3.16 Transferencia de estado representacional (REST)

La transferencia de estado representacional (REST) es una arquitectura de software que impone condiciones y principios sobre cómo debe funcionar una API o servicio web [31].

En la práctica, se puede emplear una estructura fundamentada en REST para respaldar interacciones confiables y eficientes a gran escala.

A continuación, se presentan los principios del estilo arquitectónico REST:

1. Interfaz uniforme: todas las solicitudes para el mismo recurso deben tener el mismo aspecto, sin importar de donde provenga la solicitud. Se debe garantizar que el mismo dato pertenezca a un solo identificador uniforme de recurso (URI). Los recursos no deben ser demasiados grandes, pero deben contener toda la información que el cliente necesite.
2. Desacoplamiento cliente-servidor: las aplicaciones de cliente y servidor deben mantener una total independencia entre sí. La única información que la aplicación del cliente debe conocer es el identificador uniforme de recurso (URI) del recurso solicitado, sin la capacidad de interactuar de ninguna otra manera con el servidor. Del mismo modo una aplicación de servidor no debería modificar la aplicación del cliente además de pasarle los datos solicitados a través de HTTP.
3. Tecnología sin estado: La API o servicio web no debe mantener un estado, lo que implica que cada solicitud debe incluir toda la información necesaria para su comprensión y procesamiento. Además, no se necesita de ninguna sesión en el lado del servidor, y las aplicaciones del servidor no tienen autorización para almacenar datos relacionados con las solicitudes de los clientes.
4. Sistema por capas: las solicitudes y respuestas pasan por diferentes capas, por lo que puede haber varios intermediarios en la comunicación. El diseño de la API debe de ser de tal manera que ni el cliente ni el servidor puedan saber si se están

comunicando con la aplicación final o un intermediario. Como regla general no hay que asumir que las aplicaciones clientes y servidor se comunican directamente entre sí.

5. Almacenamiento en caché: cuando sea posible los recursos deben almacenarse en la memoria cache en el lado del cliente o del servidor. Las respuestas del servidor también deben contener información sobre si se permite el almacenamiento en cache para el recurso entregado. El objetivo es mejorar el rendimiento en el lado del cliente, mientras aumenta la escalabilidad en el lado del servidor.
6. Código bajo demanda: las respuestas también tienen la posibilidad de incorporar código ejecutable, el cual solo debería efectuarse cuando sea requerido [32].

2.3.17 Interfaz de programación de aplicaciones (API)

Una API, o interfaz de programación de aplicaciones es un conjunto de reglas que definen la forma en la que diferentes sistemas de software pueden contactarse y comunicarse entre sí. En cierto sentido permite que una aplicación o servicio acceda a un recurso dentro de otra aplicación o servicio. La aplicación o servicio que realiza el acceso se llama cliente y la aplicación o servicio que contiene el recurso se llama servidor [33].

Los desarrolladores podemos diseñar APIs por medio de varias arquitecturas. Una API diseñada bajo la arquitectura REST se denomina API REST, si resulta que está desarrollada bajo el contexto de la web suele conocerse como API RESTful, en la práctica ambos términos se suelen utilizar de manera intercambiable [31].

Las API REST se pueden desarrollar utilizando prácticamente cualquier lenguaje de programación y admiten una variedad de formatos de datos. El único requisito para que una API sea REST es que esté diseñada bajo los principios mencionados en la sección anterior.

La comunicación entre el cliente y el servidor se realiza mediante solicitudes y respuestas HTTP. Con este enfoque, el cliente es el navegador o servidor web y el servidor es el servidor de aplicación. El cliente solicita o envía datos al servidor, y el servidor le responde con tal información. La Figura 11 muestra la relación cliente-servidor en un contexto de API REST y como esta es afectada por los conceptos de solicitud y respuesta HTTP.

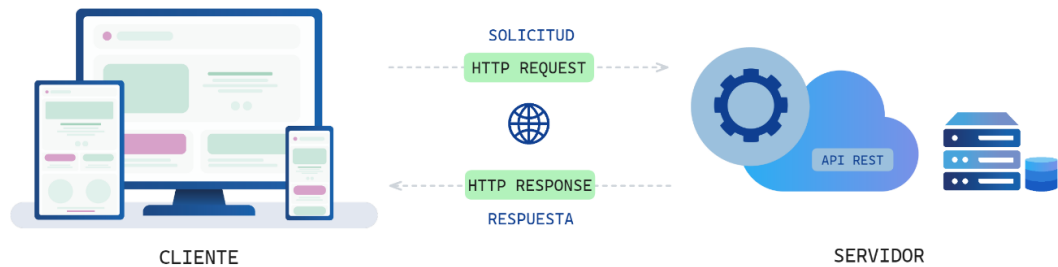


Figura 11. Relación cliente-servidor en un contexto de API REST

Adicionalmente, la Figura 12 muestra un análisis detallado sobre las solicitudes y respuestas HTTP.

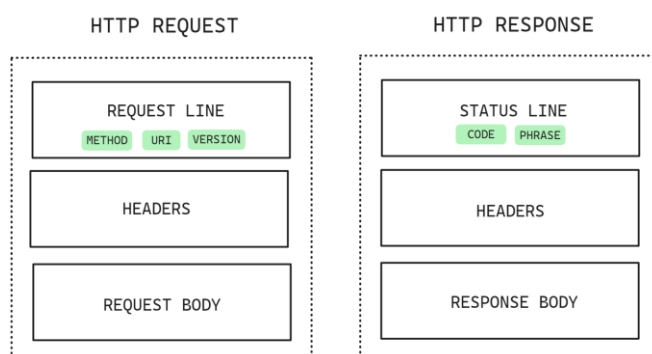


Figura 12. Composición de solicitudes y respuestas HTTP

Por consiguiente, desde un punto de vista práctico la estructura que conlleva la comunicación entre un cliente y un servidor con la posibilidad de obtener o enviar datos basado en una API construida bajo la arquitectura REST que representada por la Figura 13.

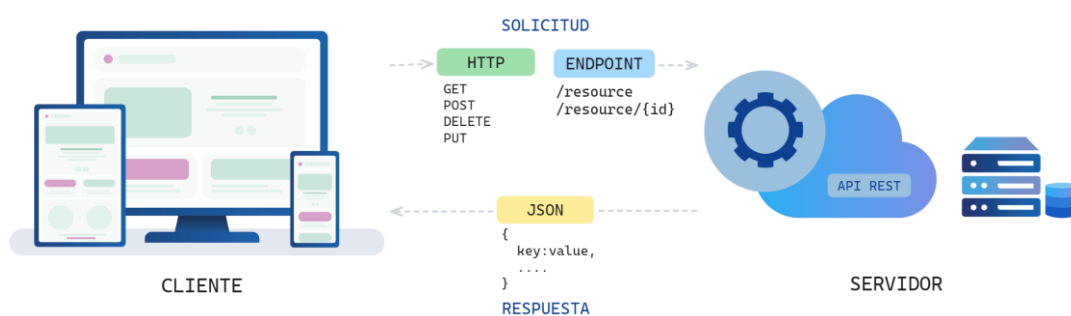


Figura 13. Estructura de una solicitud y respuesta HTTP en la comunicación con una API REST

2.3.18 Servicio web

Un servicio web se puede entender como una tecnología que emplea el conjunto de protocolos y estándares necesarios para permitir la interoperabilidad de aplicaciones, a través de internet, independientemente del lenguaje o sistema operativo que utilicen.

Conceptualmente, un servicio web es una API accesible mediante una red (normalmente Internet) que permite a aplicaciones o sistemas intercambiar información usando protocolos estándar como HTTP o HTTPS. Por ejemplo, Google Maps Platform ofrece diversos servicios web que permiten a aplicaciones obtener rutas, mapas y datos geográficos en tiempo real.

Si bien todo servicio web implementa una API, no todas las APIs son servicios web, ya que una API puede ser también local (como una librería de programación o una interfaz interna del sistema operativo).

2.3.19 Arquitectura peer-to-peer (P2P)

La arquitectura P2P es modelo de red donde cada nodo actúa como consumidor (cliente) y proveedor (servidor) al mismo tiempo. En este estilo arquitectónico no es necesario un servidor central que administre la red, aunque puede existir, sino que todos los nodos pueden comunicarse entre sí [22]. La Figura 14 representa la arquitectura de red P2P y un ejemplo de comunicación entre sus nodos.

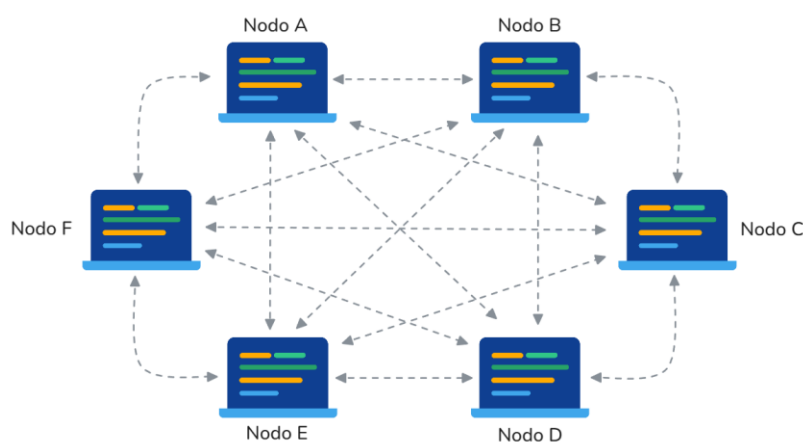


Figura 14. Arquitectura de red peer-to-peer (P2P)

El funcionamiento radica en que cada nodo consume los recursos que otros nodos exponen y a su vez permite que otros nodos consuman de él, haciendo que el software se presente como cliente y servidor al mismo tiempo. Para conectar todos estos nodos de la red, utiliza los mismos conceptos que la arquitectura cliente-servidor. Aunque utilicen los

misimos conceptos, la arquitectura cliente-servidor tiene como objetivo ofrecer centralización, mientras que la arquitectura P2P busca lo contrario.

En consecuencia, entre más nodos tengamos en la red más poder de procesamiento se agrega. Esto significa que la arquitectura tiene una increíble capacidad de escalamiento.

Aunque mencionamos que la arquitectura P2P busca la descentralización y no es necesario de un servidor central para su funcionamiento, existen variantes que lo requieren. Esto depende de lo que estemos buscando resolver, es por ello por lo que podemos establecer la siguiente clasificación:

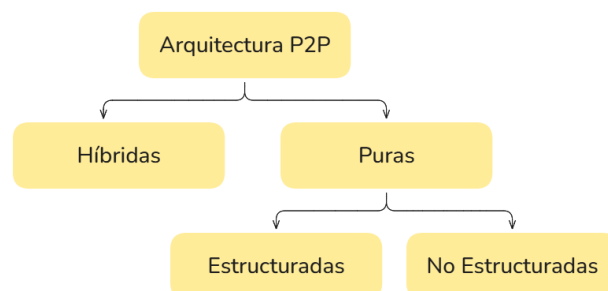


Figura 15. Clasificación de la arquitectura P2P

Arquitectura P2P Híbrida

Se caracteriza por tener un servidor central que facilita la conexión entre los nodos de la red. Este servidor no provee los recursos, sino que solo sirve de enlace para conectar los nodos [22]. La Figura 16 representa la arquitectura P2P híbrida con los nodos que se comunican entre sí para el intercambio de datos y un servidor central que actúa como enlace.

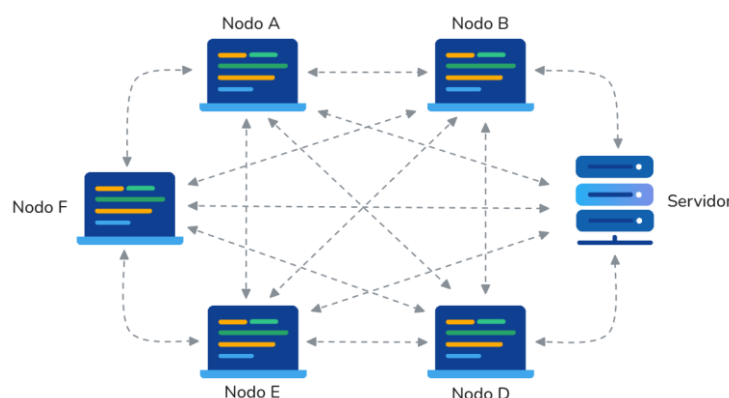


Figura 16. Arquitectura P2P Híbrida

Arquitectura P2P Pura No Estructurada

Plantea un modelo de red totalmente descentralizado, donde no existe un servidor central ni roles que den más privilegios o responsabilidades a algún nodo. En esta arquitectura un nodo solo se puede comunicar con los nodos que conozca. A estos nodos se los conoce como vecinos, por lo que cada nodo puede tener un número de vecinos diferente. La búsqueda de recursos se hace por inundación [22].

Arquitectura P2P Pura Estructurada

Es una variante de la arquitectura P2P pura no estructurada solo que cambia la forma de buscar los recursos. En lugar de usar el método por inundación, busca los recursos basados en Tabla Hash Distribuidas (DHT por sus siglas en inglés) [22].

2.3.20 Comunicación en tiempo real (RTC)

La comunicación en tiempo real (RTC, por sus siglas en inglés) hace referencia a la transmisión instantánea de datos entre dos o más puntos, permitiendo una interacción sin demoras perceptibles para los usuarios. Este tipo de comunicación es fundamental en sistemas modernos que requieren inmediatez, como asistentes virtuales, chats interactivos, videollamadas, transmisiones en vivo o monitoreo en tiempo real.

RTC no se limita a audio o video: también abarca la sincronización de estados, comandos o información contextual entre cliente y servidor. Para lograrlo, se utilizan mecanismos como conexiones persistentes, intercambio de mensajes asíncronos o protocolos que minimicen la latencia.

2.3.21 Manejo de tareas asincrónicas y programación de trabajos

El manejo de tareas asincrónicas permite que los sistemas informáticos ejecuten operaciones sin necesidad de esperar a que una finalice para iniciar la siguiente, lo cual mejora significativamente el rendimiento y la escalabilidad. En entornos donde los procesos dependen de múltiples eventos (como respuestas de APIs, tareas programadas o cálculos intensivos), esta capacidad es esencial.

Por su parte, la programación de trabajos (job scheduling) permite organizar tareas que deben ejecutarse en momentos específicos, intervalos recurrentes o en función de condiciones del sistema. Este enfoque es ampliamente usado en automatizaciones, optimización de procesos, y coordinación de múltiples servicios.

2.3.22 Modelos de inteligencia artificial

Los modelos de inteligencia artificial (IA) son sistemas informáticos capaces de simular capacidades humanas como el razonamiento, el aprendizaje, la comprensión del lenguaje o la toma de decisiones. Estos modelos se basan en algoritmos que permiten analizar datos, detectar patrones y generar respuestas en función del contexto.

Existen diversos tipos de modelos de IA, desde técnicas de aprendizaje automático (machine learning) hasta modelos de aprendizaje profundo (deep learning). En particular, los modelos de lenguaje natural permiten interpretar y generar texto de forma coherente, permitiendo su aplicación en asistentes inteligentes, chatbots, análisis semántico y más.

2.4. Soluciones similares

2.4.1 Google Maps

Google Maps es la aplicación de mapas y navegación más usada en el mundo, un hecho innegable gracias a todas sus características:

1. Es gratuita.
2. Está disponible en móviles y en la web.
3. Ofrece mapas de más de 220 países y territorios.
4. Brinda la ubicación de millones de lugares y empresas.
5. Permite descubrir nuevos lugares y explorarlos como si estuvieras allí.
6. Brinda distintos modos de viaje y muestra la duración aproximada del mismo.
7. Ofrece información del tráfico en tiempo real.
8. Ofrece las rutas más rápidas y ecológicas.
9. Brinda un aproximado del costo por peaje.

Exprimiendo todo esto y siguiendo algunas guías, Google Maps se puede convertir en un excelente planificador de viajes por carretera para viajantes y turistas. Sin embargo, no es su función principal y se echan de menos varias características avanzadas que pueden enriquecer la experiencia de la planificación como la integración con un agente inteligente, la posibilidad de crear plantillas de actividades personalizadas y poder visualizar estadísticas básicas.

2.4.2 Waze

Es una aplicación de navegación de participación colectiva que ayuda a los conductores a evitar algunos imprevistos que se puedan dar en la ruta.

Se encuentra impulsada principalmente por su comunidad, que la suele utilizar para los viajes y trayectos que recorren en su día a día. Mientras la usan, los conductores comparten información en tiempo real y pueden informar sobre los imprevistos presentes en su recorrido tales como el estado del tráfico, accidentes, controles policiales, condiciones meteorológicas, entre otros.

Estos datos aportados por la comunidad son recopilados y analizados al instante por Waze para brindar a otros conductores sugerencias de la ruta óptima para llegar a sus destinos.

Al ser clave la participación de la comunidad, la aplicación funciona mejor teniendo acceso a internet y recorriendo las rutas que sugiere.

Además, brinda algunos servicios adicionales como:

1. Añadir paradas intermedias
2. Consultar estaciones de servicio y precio de combustibles
3. Compartir ubicación
4. Evitar peajes

Si bien Waze prioriza brindar la ruta más rápida con la actualización en tiempo real gracias a su comunidad y puede ser utilizada como un buen planificador de rutas, también se echa de menos el contar con características avanzadas para mejorar la experiencia de la planificación del viaje.

2.4.3 YPF Ruta

Es una plataforma de gestión y optimización de flotas creada por la empresa argentina YPF (Yacimientos Petrolíferos Fiscales, S.A). Cuenta con cobertura nacional en todo el país y ofrece planes que se ajustan a la necesidad de cada tipo de flota.

Se puede aplicar a cualquier tipo de unidad, ya sea liviana o pesada, y se adapta a cualquier tipo de operación como el transporte de carga, transporte de pasajeros o equipos pesados permitiendo entre otras cosas:

1. Tener una cuenta para consolidar todo el consumo, habilitar vehículos, conductores, definir reglas de consumo y límites, haciendo que la gestión del

abastecimiento y el control sea una tarea sencilla, además de acceder a descuentos por tus consumos y un plazo de pago

2. Obtener la ubicación en tiempo real de las unidades, para programar recorridos, hacer seguimiento de las rutas planificadas, medir horarios de retiro, entrega o paradas, frecuencias de visitas a clientes, y hasta los tiempos que pasan las unidades dentro o fuera de un sector específico de una planta.
3. Disponer de un scoring de manejo por unidad o conductor, para corregir hábitos imprudentes en la conducción, detectar actitudes imprudentes y excesos de consumo, para alcanzar un índice de manejo icono-eficiente y seguro que minimice las posibilidades de siniestros e infracciones.
4. Utilizar la analítica avanzada que genera la operación de tu flota te permitirá conocer el tiempo de uso, los kilómetros recorridos, las zonas y rutas por las que transitan, la huella de carbono, el rendimiento, y otros indicadores más. Con esa información en tu mano podrás tomar decisiones más precisas y confiables, respaldadas en información sólida.
5. Programar vencimientos de documentación, controles o reparaciones a través de un completo módulo de mantenimiento para gestionar los planes de mantenimiento y registrar toda intervención a las unidades. Esto permite controlar los gastos de mantenimiento, conocer el TCO (Costo total operativo) y lograr implementar un esquema preventivo y predictivo para las reparaciones, que te permitirá ahorrar dinero y evitar paradas imprevistas.

Esto es posible tras conectar los vehículos con tecnología y utilizar telemática vehicular avanzada para recolectar, interpretar y presentar de forma fácil e intuitiva toda la información generada por la flota en tiempo real permitiendo que los gestores puedan tomar mejores decisiones basadas en esta información confiable.

2.5 Tecnologías requeridas para el tratamiento del problema

2.5.1 Lenguajes de programación, frameworks y librerías

Desarrollar una aplicación web implica que como programadores se realice una elección sobre lenguajes de programación, frameworks y librerías.

Un lenguaje de programación es un conjunto de reglas sintácticas y semánticas que nos permiten construir código fuente. Son utilizados con el objetivo de expresar instrucciones y lógica en los diferentes algoritmos que formen parte de nuestros procesos. La selección

de un lenguaje de programación va a depender de las necesidades de negocio, el área específica de aplicación y las capacidades técnicas del equipo de desarrollo. También puede influir la comunidad y el soporte.

Un framework, en el contexto de programación, es un marco que nos proporciona una estructura base para desarrollar una aplicación. Los frameworks incorporan un mecanismo de abstracción que mediante librerías, módulos y buenas prácticas facilitan la construcción y organización del código fuente de la aplicación. Además, nos ayudan a eliminar código repetido mediante la reutilización y a implementar nuevos estándares en la arquitectura de la aplicación, lo cual a su vez hace que los programadores sean más eficientes y productivos. La selección de un framework va a depender del lenguaje de programación seleccionado para el desarrollo, el tipo de aplicación a desarrollar, soporte y tamaño de la comunidad.

Una librería, también conocido como biblioteca o dependencia, es un conjunto de utilidades escritas en un lenguaje de programación que proporcionan funcionalidad extra para la aplicación. Utilizan una interfaz para facilitar la compatibilidad con el lenguaje de programación utilizado en la aplicación. Las librerías permiten reutilizar código y ayudan con el mantenimiento y comprensión del código fuente. La selección de las librerías adecuadas va a depender de las necesidades del proyecto, compatibilidad con el lenguaje de programación seleccionado, valoración de la comunidad, soporte y documentación de estas.

La exclusiva elección de los lenguajes de programación, frameworks y librerías requeridas tanto para backend como para front-end en nuestro caso, además de tener en cuenta lo anteriormente mencionado, dependerá adicionalmente de las posibilidades que estos puedan ofrecernos en la actualidad en términos de escalabilidad, abstracción, sencillez y eficiencia en el desarrollo.

2.5.2 Sistemas de gestión de paquetes

Un sistema de control de paquetes es un sistema que nos permite instalar, remover o actualizar las librerías o dependencias de terceros necesarias para el correcto funcionamiento de una aplicación. Actúa como gestor mediante comandos facilitando la instalación de versiones personalizadas y la eliminación de código deprecado de terceros. Adicionalmente, permite a los desarrolladores trabajar con ciertos recursos sin la necesidad estricta de que conozcan cómo se realizaron internamente. Esto último

proporciona una capa de abstracción de tal forma que al programador solo le basta con conocer para que sirve y como se aplica.

Para la construcción de la aplicación es necesario el uso de sistemas de gestión de paquetes, de tal forma que podamos instalar, remover y actualizar todas aquellas dependencias que nos faciliten resolver el problema.

2.5.3 Sistemas de control de versiones

Un sistema de control de versiones es un sistema que registra los cambios que se realizan sobre un archivo o conjunto de archivos a lo largo del tiempo, en nuestro caso sobre un proyecto de desarrollo o código fuente, de tal manera que permita establecer diferentes versiones para que ante una necesidad puedan ser recuperadas[45]. En el ámbito del desarrollo de software es una práctica que hoy en día es obligatoria ya que nos permite tener un control sobre el código de la aplicación y nos facilita rastrear y administrar los cambios. Adicionalmente, esta herramienta ayuda a los equipos a trabajar de manera más rápida y a dividir las tareas eliminando en mayor medida las preocupaciones ocasionadas por problemas de integración.

Tener un sistema de control de versiones para realizar un seguimiento en la aplicación web facilitará su desarrollo ya que si se presenta un error tendremos la posibilidad de volver a versiones anteriores para comparar y proporcionar una solución adecuada.

2.5.4 Autenticación y autorización

Json Web Token.

O bien JWT es un estándar para compartir información de forma segura entre partes, a través de objetos JSON. Esta información es codificada y firmada digitalmente para garantizar su autenticidad [46].

En los desarrollos actuales es muy común encontrar JWT en los flujos de autenticación y autorización de varias APIs como así también en la transferencia de datos entre clientes y servidores.

JWT se compone de 3 secciones:

1. Cabecera “*header*”: Indica el algoritmo y el tipo de token
2. Carga útil “*payload*”: Podemos encontrar datos del usuario, privilegios y cualquier otro dato que consideremos conveniente.

3. Firma “*Signature*”: Permite verificar la validez del token.

De forma representativa, un JWT tiene la siguiente estructura:

``xxxxx.yyyyyy.zzzzz``

Donde tanto la cabecera “x” como la carga útil “y” se encuentran codificados en Base64 y la firma “z” es un hash generado a partir del algoritmo de cifrado indicado en la cabecera, tomando como parámetros “x”, “y” y una clave secreta compartida entre el emisor y el receptor del token.

Para verificar la autenticidad de un JWT, el destinatario genera una nueva firma con la clave secreta provista por el emisor. Si la firma generada coincide con la recibida en el JWT, entonces podemos asegurar que el JWT no fue alterado.

OAuth.

Es un marco de autorización, ampliamente utilizado en la industria, que permite acceder a recursos alojados por aplicaciones de terceros en nombre del usuario sin compartir sus credenciales. Este acceso tiene el consentimiento del usuario y restringe las acciones que la aplicación de origen puede realizar con los recursos disponibles en la aplicación externa.

Los JWT suelen utilizarse para emitir y verificar tokens de acceso y actualización en OAuth. Su versión más reciente es OAuth 2.0.

2.5.5 Bases de datos

En el desarrollo de software, una base de datos es una recopilación de datos sistemática y almacenada electrónicamente. Puede usarse un software denominado sistema de administración de base de datos (DBMS) para almacenar, recuperar y editar los datos.

Relacionales.

Son bases de datos, inspiradas en el modelo relacional, que almacenan y proporcionan acceso a datos relacionados entre sí. Para almacenarse, los datos se organizan en filas y columnas, que forman una tabla.

Los datos normalmente se estructuran en varias tablas, que se pueden unir a través de una clave principal o una clave externa.

Estos identificadores únicos demuestran las relaciones entre las tablas, y se ilustran con diferentes tipos de modelos de datos.

No relacionales.

Las bases de datos no relacionales, llamadas generalmente NoSQL (Not Only SQL), se ocupan de manejar y almacenar datos de manera flexible y escalable.

A diferencia de las bases de datos relacionales, las bases de datos NoSQL permiten almacenar y recuperar datos de forma más libre y menos estructurada. Se encuentran diseñadas para manejar grandes volúmenes de datos no estructurados o semi estructurados, como documentos, gráficos, clave-valor o columnas.

Las bases de datos NoSQL se utilizan en escenarios donde la escalabilidad, la velocidad y la flexibilidad son prioritarias. Son especialmente adecuadas para aplicaciones web y móviles con grandes volúmenes de datos y elevados requerimientos de performance, como redes sociales, sistemas de gestión de contenido y aplicaciones en tiempo real.

2.5.6 Plataforma como servicio (PAAS)

Una PaaS “es un entorno en la nube integral que incluye todo lo que los desarrolladores necesitan para crear, ejecutar y gestionar aplicaciones, desde servidores y sistemas operativos hasta redes, almacenamiento, middleware, herramientas y más” [47].

A diferencia de otros sistemas, una PaaS es particular en el desarrollo de aplicaciones, proporcionando diversas herramientas para que los desarrolladores puedan trabajar y colaborar sin complicaciones. Es habitual que las aplicaciones se creen en la plataforma y se puedan implementar de inmediato cuando se terminen.

Una PaaS tiene las siguientes ventajas:

1. Requiere menos esfuerzo por lo que permite que el equipo de desarrollo pueda dedicar más tiempo a desarrollar y desplegar la aplicación.
2. No hay que preocuparse por el mantenimiento ya que el proveedor se encarga de mantener todo actualizado.
3. Ofrece un conjunto de herramientas de desarrollo avanzadas disponibles bajo demanda, es decir, se solo se paga por lo que se utilice.
4. Permiten escalar verticalmente para hacer frente a aumentos de demanda.

5. Provee acceso a herramientas y servicios compartidos desde cualquier lugar y en cualquier dispositivo.
6. Ofrece las practicas recomendadas para desarrollar en la plataforma garantizando que sea el proveedor el encargado de proteger la infraestructura [47].

Disponer de una infraestructura en la nube, kits de desarrollo, frameworks, bibliotecas, interfaz gráfica de usuario, línea de comandos, y muchas otras herramientas que proporcionan una PaaS facilitara el desarrollo de la aplicación permitiendo que toda la configuración tanto en equipos de red y servidores como en sistemas operativos y entorno necesarios para que la aplicación funcione sea totalmente abstracto y transparente. Además, permitirá que como desarrolladores nos centremos específicamente en el análisis y codificación de la aplicación asegurando un tiempo de lanzamiento más rápido con menos esfuerzo en el proceso.

2.5.7 WebRTC

WebRTC (Web Real-Time Communication, por sus siglas en inglés), es un conjunto de interfaces estandarizadas que permiten desarrollar aplicaciones web con capacidades de transmisión en tiempo real de audio, video y datos arbitrarios entre navegadores. Su principal ventaja radica en que no requiere la instalación de complementos, extensiones ni software adicional, ya que está nativamente soportada en los navegadores modernos mediante APIs de JavaScript [37]-[38].

Esta tecnología permite establecer conexiones peer-to-peer (P2P) que posibilitan la comunicación directa entre dispositivos, reduciendo la latencia e incrementando la eficiencia en la transmisión de información. Esta tecnología se encuentra disponible tanto en navegadores como en clientes nativos para las principales plataformas de escritorio y móviles, lo que facilita su integración en una amplia variedad de entornos [39].

El funcionamiento de WebRTC se basa en tres fases fundamentales:

1. Captura de medios
2. Establecimiento de conexión
3. Negociación de sesión y transmisión

2.5.8 WebSocket

WebSocket es un protocolo que nos permite establecer una conexión bidireccional en tiempo real entre un servidor y un cliente a través de un único canal de conexión. Esto

facilita el envío y la recepción simultánea de los datos sin sobrecargar la red con múltiples solicitudes HTTP.

La conexión por WebSocket se establece, conceptualmente, de la siguiente manera:

1. El cliente envía una petición HTTP para iniciar la conexión.
2. El servidor responde con el código 101, confirmando el cambio de protocolo.
3. Se establece un canal de comunicación full-duplex sobre un único socket TCP.
4. Establecido el canal, tanto el cliente como el servidor pueden intercambiar datos en tiempo real sin la necesidad de realizar nuevas peticiones.

Gracias a su eficiencia y baja latencia para compartir información en tiempo real, esta tecnología es ampliamente utilizada en la creación de chats en línea, dashboards que muestran estadísticas en vivo, juegos multijugador y sistemas de notificaciones.

2.5.9 Asistentes de inteligencia artificial

Los asistentes de inteligencia artificial son sistemas diseñados para interactuar con los usuarios de manera natural, comprender sus necesidades y brindar respuestas o acciones útiles en función del contexto. Ejemplos populares incluyen asistentes como Siri, Alexa, o ChatGPT.

Estos asistentes no funcionan como una tecnología única, sino que se construyen sobre modelos de inteligencia artificial avanzados, especialmente aquellos orientados al procesamiento de lenguaje natural (PLN).

2.5.10 Sistemas de gestión de tareas programadas

Un sistema de gestión de tareas programadas es un sistema que permite automatizar la ejecución de tareas en momentos específicos o bajo ciertas condiciones, sin requerir intervención humana. Estas tareas pueden ser de ejecución única (cuando son diferidas) o de ejecución periódica (cuando se repiten en intervalos definidos).

Una de las principales ventajas de estos sistemas es que permiten configurar aspectos como la prioridad de ejecución, los reintentos en caso de error y las dependencias entre tareas. Además, es posible establecer flujos de trabajo que contemplen dependencias transitivas, facilitando así la coordinación de procesos más complejos.

Estos sistemas se utilizan habitualmente para tareas como el envío de correos programados, la ejecución de procesos en segundo plano, la realización de respaldos

automáticos de bases de datos, y la gestión de notificaciones o recordatorios, entre otros casos de uso frecuentes.

2.5.11 Sistemas de gestión de proyectos

Un sistema de gestión de proyectos es un sistema que permite dar seguimiento a proyectos en términos de análisis funcional, codificación de aplicación, pruebas y documentación. Permite que el equipo de trabajo, es decir, analistas, desarrolladores, QA, entre otros, puedan ver el avance del proyecto de acuerdo con las funcionalidades y marco de trabajo utilizado. Además, permite ver las incidencias del equipo dentro del proyecto facilitando la definición y asignación de actividades independientemente de la etapa en la que nos encontremos.

Estos sistemas por lo general incluyen herramientas y características que nos permiten definir el esquema de trabajo, dar seguimiento a las actividades, integrar la aplicación y escribir documentación. Adicionalmente proporcionan flexibilidad y capacidad de personalización para adaptarlo a nuestro flujo de trabajo.

El uso de un sistema para gestionar el trabajo permitirá tener mayor productividad y visibilidad a medida que se avanza a la implementación del software.

2.6 Temas pendientes de resolución

Con la necesidad de expandir el conocimiento y superar la adversidad de los distintos conceptos teóricos desarrollados en este capítulo, quedan pendientes los siguientes temas:

1. Teoría de grafos y el modelo de transporte.
2. Técnicas avanzadas de optimización.
3. Factores influyentes en la optimización de rutas.
4. Aplicaciones de la optimización de rutas.
5. Elementos del transporte.
6. Integración de sistemas informáticos.
7. Técnicas para mejorar el rendimiento de los sistemas.

3. DEFINICIÓN DEL PROBLEMA

3.1 Objetivos

3.1.1 Objetivos generales

Diseñar y desarrollar un sistema informático que integre y automatice parcialmente el proceso de optimización de rutas, adaptable a distintos contextos y casos de uso, con el propósito de mejorar la eficiencia operativa en el proceso de transporte.

3.1.2 Objetivos específicos

1. Analizar los requerimientos del sistema, considerando las particularidades de distintos contextos y casos de uso dentro del proceso de transporte.
2. Investigar y seleccionar algoritmos, librerías y servicios existentes que ayuden en el proceso de optimización de rutas.
3. Diseñar una arquitectura de software sólida, robusta y escalable, basada en buenas prácticas, que facilite la integración de múltiples contextos y casos de uso sin modificar la lógica base del sistema.
4. Desarrollar un sistema web moderno que automatice el proceso de optimización de rutas, contemplando funcionalidades específicas de cada contexto y caso de uso seleccionados como casos de validación práctica.
5. Implementar criterios de optimización parametrizables, que puedan adaptarse a las necesidades de cada tipo de usuario (tradicional o empresarial), asegurando flexibilidad y personalización en la optimización de rutas.
6. Aplicar técnicas de desarrollo web moderno, priorizando la calidad del código, la optimización para motores de búsqueda (SEO), la accesibilidad y la experiencia de usuario.
7. Validar el funcionamiento del sistema mediante pruebas funcionales, verificando su desempeño en los dos contextos seleccionados, y documentando los resultados obtenidos.
8. Evaluar la escalabilidad del sistema y su potencial de extensión, analizando cómo podría adaptarse a otros contextos y casos de uso del transporte sin alterar su estructura principal.
9. Utilizar metodologías ágiles y herramientas de trabajo colaborativo, aplicando buenas prácticas de gestión de proyectos durante todo el proceso de desarrollo.

3.2 Contextualización del problema

En el marco de esta tesis, se aborda el problema de la optimización de rutas de manera general, sin restringirse a un único contexto y caso de uso, sino proponiendo una solución adaptable.

Desarrollar un sistema informático integral que cumpla con lo planteado requiere contemplar las particularidades de cada contexto y las necesidades asociadas a sus casos de uso, asegurando que ambos puedan integrarse adecuadamente en una solución unificada.

En líneas generales, cada contexto dispone de características generales, y cada uno de los casos de uso hacen que estas varíen dependiendo de las necesidades en el proceso de transporte.

Para el desarrollo del sistema y validación de los objetivos, se han tomado como referencia dos contextos representativos —planificación de viajes y gestión de flotas—, y en cada uno de ellos se ha elegido un caso de uso específico: la planificación de viajes asistidos, y la gestión de flotas en servicios a domicilio, ambas con optimizaciones programadas. El análisis detallado de cada contexto y su respectivo caso de uso se presenta en las siguientes secciones.

3.3 Alcance

3.3.1 Descripción del alcance del producto

El presente proyecto consiste en el diseño y desarrollo de un sistema web moderno integral orientado a la optimización de rutas en el proceso de transporte. La solución respeta una arquitectura modular y escalable de manera que permite la adaptación de diferentes contextos y casos de uso.

La validación del sistema se realiza en dos casos específicos: la planificación de viajes asistidos y la gestión de flotas en servicios a domicilio, ambas con optimizaciones programadas. En este marco, el sistema contempla dos tipos de usuario:

1. Usuario personal, enfocado en la planificación de viajes asistidos mediante inteligencia artificial y con optimizaciones programadas.
2. Usuario empresarial, orientado a la gestión de flotas en servicios a domicilio con optimizaciones programadas.

La interfaz del sistema está compuesta por un conjunto de secciones y páginas diseñadas para responder a las necesidades específicas de cada caso de uso, en función del tipo de usuario. Además, se contemplan prácticas de accesibilidad web alineadas con los principios establecidos por las WCAG, con el propósito de garantizar la usabilidad por parte de personas con distintos niveles de discapacidad.

Con el propósito de mejorar la visibilidad del sistema y su contenido en entornos públicos, se consideran estrategias básicas de optimización para motores de búsqueda.

Si bien el sistema ha sido concebido con la capacidad para gestionar suscripciones, esta funcionalidad no ha sido desarrollada en esta versión, pero se tienen en cuenta en lo funcional como parte de una versión futura.

Los módulos del sistema son:

1. Módulos Generales

- a. Módulo de autenticación: Este módulo permite a los usuarios, mediante los datos solicitados, ingresar y darse de alta en el sistema. Además, facilita el ingreso incorporando un inicio de sesión mediante Google y permite la recuperación de las credenciales de acceso y la verificación de la cuenta.
- b. Módulo de configuración: Este módulo permite a los usuarios configurar su cuenta mediante la modificación o agregado de información faltante. Además, permite modificar las credenciales de acceso.
- c. Módulo de actividades: Este módulo permite gestionar plantillas de actividades que posteriormente podrán ser usadas en los criterios avanzados o como servicios a domicilio en el optimizador. Entre sus funciones principales se incluye: el alta, baja y modificación de plantillas, así como la visualización de estas. Adicionalmente, incorpora el alta y baja de actividades en las plantillas.
- d. Módulo de notificaciones: Este módulo facilita la visualización de notificaciones dentro del sistema. También permite marcar las notificaciones como leídas y eliminarlas. Adicionalmente, automatiza las notificaciones ante eventualidades en el proceso de optimización.
- e. Módulo de suscripciones: Este módulo permite a los usuarios la gestión de sus suscripciones. Entre sus funciones principales se incluye: el alta y baja

de suscripciones mediante una pasarela de pagos y la visualización del historial de suscripciones, así como de los beneficios de cada plan.

2. Contexto: Planificación de viajes

Caso de uso: Planificación de viajes asistidos

- a. Módulo de direcciones favoritas: Este módulo se encarga de gestionar todas las direcciones que los usuarios suelen frecuentar. Entre sus principales funcionalidades están: alta, baja y modificación de direcciones favoritas y su visualización.
- b. Módulo de lugares favoritos: Este módulo permite a los usuarios administrar sus lugares de favoritos: Un lugar es cualquier ubicación relevante para el usuario como restaurantes, hoteles, museos, parques, puntos turísticos, estaciones de servicio, hospitales o cualquier otro lugar que los usuarios puedan querer visitar o necesitar durante su viaje. Entre sus funcionalidades principales se encuentra: el alta, baja y modificación de lugares favoritos como así también la visualización de estos.
- c. Módulo de optimización: Este módulo permite a los usuarios trazar la ruta más eficiente sobre un conjunto de criterios preestablecidos. Entre sus funciones principales se incluye: la definición de criterios básicos y avanzados, y la visualización de los resultados de la optimización.
- d. Módulo de viajes: Este módulo permite a los usuarios gestionar sus viajes dentro del sistema. Entre sus funciones principales se incluye: el alta, baja y visualización de viajes, así como la modificación del alias y de la condición asociada a cada uno.

Al dar de alta un nuevo viaje, el sistema crea automáticamente un flujo de trabajo que se encarga de optimizarla y marcarlo como usado de forma secuencial y automatizada, según corresponda.

Además, el módulo permite replicar viajes ya completados o con eventos imprevistos, y visualizar la guía del viaje con los resultados optimizados, complementada por un asistente inteligente que brinda información contextual al usuario.

3. Contexto: Gestión de flotas

Caso de uso: Gestión de flotas en servicios a domicilios

- a. Módulo de flotas. Este módulo permite administrar las flotas de la empresa. Entre sus funcionalidades principales encontramos: alta, baja y

modificación de flotas, vehículos y conductores, así como la visualización de estos. Adicionalmente, permitirá la vinculación y desvinculación de los vehículos y los conductores a una flota.

- b. Módulo de optimización. Este módulo permite a los usuarios trazar la ruta más eficiente sobre una configuración preestablecida. Entre sus funciones principales se incluye: la configuración de las etapas del proceso y la visualización de los resultados de la optimización.
- c. Módulo de hojas de ruta. Este módulo permite gestionar las hojas de ruta generadas por la empresa para los servicios de entrega o atención a domicilio. Entre sus funciones principales se incluye: el alta, baja y visualización de las hojas de ruta, así como la modificación de su alias y estado operativo.

Al dar de alta una nueva hoja de ruta, el sistema crea automáticamente un flujo de trabajo que se encarga de optimizarla, iniciarla y finalizarla de forma secuencial y automatizada, según corresponda.

Además, el módulo permite consultar en detalle cada hoja de ruta, incluyendo información sobre su configuración, métricas, asignaciones de recursos, visitas, transiciones y otros parámetros relevantes.

3.3.2 Entregables

De acuerdo con los objetivos definidos y el alcance del producto, los entregables son los siguientes:

1. Especificación de requerimientos:
 - a. Requerimientos funcionales.
 - b. Requerimientos no funcionales.
2. Diseño de la arquitectura del sistema.
3. Modelado de datos y diagramas de proceso de negocio informatizados.
4. Historias de usuario por incremento.
5. Código fuente.
6. Pruebas de calidad técnica del sistema.

3.3.3 Criterios de aceptación

Los criterios de aceptación que dan validez a los entregables son:

1. Cada módulo, a excepción de exclusiones, cumple con los requisitos funcionales y no funcionales definidos en el capítulo 4 sección 4.4.4.
2. El proceso de optimización se lleva a cabo para cada uno de los casos considerados, solventando errores producidos en los flujos de trabajo.
3. La interfaz es accesible para personas que presentan discapacidad visual.
4. La interfaz es adaptable asegurando la compatibilidad con dispositivos móviles y tablets.
5. Las páginas principales (optimización, viajes y hojas de ruta) obtienen en Google Lighthouse un puntaje mínimo de:
 - a. Rendimiento ≥ 80
 - b. Accesibilidad ≥ 90
 - c. Mejores prácticas ≥ 90
 - d. SEO ≥ 90
6. La documentación técnica y de usuario esta completa y disponible.
7. Las pruebas de aceptación son superadas en un entorno de producción.

3.3.4 Exclusiones del proyecto

Queda excluido del proyecto:

1. El desarrollo de una aplicación móvil nativa.
2. El desarrollo del módulo de suscripciones, y por lo tanto la integración con pasarelas de pago.
3. El control de métricas en tiempo real para generar estadísticas avanzadas.
4. El desarrollo del sistema de navegación.
5. La modificación de los viajes y las hojas de ruta sobre la marcha.
6. El seguimiento en tiempo real para los casos de uso.
7. La posibilidad de implementarse fuera de Argentina.
8. El cálculo de rutas seguras y ecológicas.
9. La documentación correspondiente a la migración de los servicios de tercero.
10. El ingreso al sistema como invitado sin necesidad de registrarse para dar un vistazo a todo lo que se ofrece de forma limitada.

3.3.5 Restricciones o límites del trabajo

El trabajo está sujeto al siguiente conjunto de restricciones o límites que condicionan la implementación de la solución:

1. Restricciones de alcance
 - a. El proceso de optimización se enfoca exclusivamente en el transporte por carretera en vehículos de dos o cuatro ruedas, limitándose a otras modalidades.
 - b. El sistema está pensado para uso exclusivo en territorio argentino, lo cual limita su gestión en otras regiones.
2. Restricciones legales y normativas
 - a. El sistema hace uso de servicios de terceros para el cálculo de optimizaciones, la asistencia inteligente y la gestión de mapas, de direcciones, de lugares e imágenes. Estos servicios están sujetos a políticas y condiciones de uso que exigen su cumplimiento para una implementación con fines comerciales.
 - b. No se cuentan con acuerdos a nivel de servicio (SLA) con los proveedores, por lo tanto, no hay garantía de disponibilidad, soporte técnico y mantenimiento continuo.
 - c. El sistema no implementa actualmente políticas específicas para el tratamiento, consentimiento y resguardo de datos personales, conforme a la Ley 25.326 de Protección de Datos Personales. Por tanto, no está habilitado para recolectar ni almacenar información sensible en entornos productivos.
 - d. El código fuente y los recursos utilizados están protegidos por la Ley 11.723 de Propiedad Intelectual. Asimismo, el uso de contenido y servicios de terceros está sujeto a sus respectivas licencias, lo que limita su redistribución o explotación comercial sin autorización.
 - e. El sistema no contempla condiciones legales exigidas por la Ley 24.240 de Defensa del Consumidor, tales como términos y condiciones, política de privacidad o mecanismos formales de atención al usuario, por lo cual no puede ser utilizado como una plataforma comercial sin adaptaciones legales previas.

- f. Aunque se utilizan servicios de terceros confiables para la generación de rutas, el sistema no valida normativas de tránsito locales (como zonas restringidas, horarios especiales o permisos). Por ende, no se garantiza que cumplan con la legislación vial vigente, quedando bajo responsabilidad del usuario su revisión y cumplimiento.
- 3. Restricciones de tiempo
 - a. El sistema se desarrolla en el marco de un proyecto de grado, por lo cual los plazos están condicionados por el calendario académico y la disponibilidad horaria de los autores.
- 4. Restricciones técnicas
 - a. Los servicios utilizados para el despliegue de las partes del sistema en un entorno de producción –como base de datos, API, cache e interfaz de usuario– limitan el rendimiento, procesamiento de datos y concurrencia.
 - b. Los servicios utilizados para el desarrollo del sistema limitan la manipulación, almacenamiento y redistribución de los datos, así como la posibilidad de una excesiva carga de peticiones.
- 5. Restricciones de recursos
 - a. El desarrollo se lleva a cabo por los autores, limitándose a la realización de tareas complejas en paralelo.
- 6. Restricciones de presupuesto
 - a. El sistema, en un entorno de producción, está limitado por los recursos que ofrecen los planes adquiridos, detallado en el capítulo 4 sección 4.6.

3.3.6 Supuestos

Se asume que:

- 1. Todos los usuarios del sistema cuentan con dispositivos con conexión estable a internet.
- 2. Todos los dispositivos están actualizados y cuentan con la última versión de su navegador web de preferencia.
- 3. Los potenciales usuarios cuentan con un correo electrónico verificable para darse de alta en el sistema.
- 4. Los servicios de terceros están disponibles a modo de prueba cerrada en un entorno de desarrollo y a modo de prueba abierta en un entorno de producción.
- 5. Los usuarios diferencian entre las particularidades de cada rol.

6. No se necesita un soporte multilingüe en una versión inicial.
7. Los dispositivos cuentan con los recursos necesarios (audio y entrada de voz) para recibir asistencia inteligente.
8. El uso del sistema se realiza en su correspondiente contexto.
9. No se requiere del módulo de suscripciones para analizar la factibilidad de la solución.

3.4 Explicación funcional de los casos de uso

3.4.1 Planificación de viajes asistidos

Este caso de uso, perteneciente al contexto de planificación de viajes, atiende la necesidad de organizar y optimizar un trayecto antes de su realización, considerando una serie de condiciones particulares definidas.

Cuando una persona necesita trasladarse, debe determinar la ruta más conveniente teniendo en cuenta distintos criterios, tales como: origen, destino, fecha y hora de salida, puntos de interés, condiciones de tráfico, presencia o ausencia de peajes, autopistas o traspasadores, costos estimados, tipo de vehículo y tipos de ruta, entre otros factores relevantes. Actualmente, esta actividad puede realizarse mediante herramientas generalistas como Google Maps o, en su defecto, de forma manual a través de mapas físicos, aunque esta última alternativa no es recomendable.

En el afán de lograr una planificación eficiente, la persona requiere contar con una guía optimizada, recibir asistencia contextual sobre la misma y, en caso de ser necesario, reutilizar los criterios previamente definidos para planificar nuevos trayectos. Sin embargo, al no existir un mecanismo que permita conservar y consultar dicha información de manera integrada, cada nuevo viaje implica repetir el proceso de análisis desde cero, incrementando el esfuerzo y el tiempo necesarios para tomar una decisión adecuada.

Cabe destacar que, si bien el caso de uso se denomina planificación de viajes asistidos, en su funcionamiento interno contempla un proceso de optimización de rutas. Esta optimización se realiza en función de los criterios definidos previamente, permitiendo que la planificación no solo contemple el armado de un trayecto, sino que proponga la alternativa más eficiente dentro de las posibilidades disponibles.

3.4.2 Gestión de flotas en servicios a domicilio

Este caso de uso, perteneciente a la gestión de flotas, atiende la necesidad en la que una empresa desea optimizar y organizar hojas de rutas para la realización de servicios a domicilio.

Cuando una empresa de logística, servicios técnicos u otro rubro similar debe organizar sus operaciones, necesita obtener el trayecto más eficiente posible, junto con la asignación de los servicios a ejecutar. Para ello, agrupa múltiples servicios dentro de una misma hoja de ruta, considerando diversos aspectos como: disponibilidades de recursos (flota, vehículo y conductor), horario de inicio y fin del recorrido, inclusión o exclusión de peajes, autopistas o trasbordadores, condiciones de tráfico, ubicaciones de los servicios y sus respectivas duraciones, perfiles de costos y restricciones asociadas a la gestión de flotas, como límites de tiempo o distancia.

Para lograr una gestión de flotas eficiente, es necesario que la asignación de servicios esté coordinada con el horario del recorrido y la ruta óptima para su ejecución. En este sentido, la empresa requiere contar con hojas de ruta que integren todos estos detalles, indiquen su estado y sirvan como guía operativa para el personal encargado de llevar a cabo el recorrido.

3.5 Alternativas tecnológicas de solución al problema

Para solucionar el problema se tienen en cuenta las siguientes alternativas tecnológicas:

3.5.1 Tecnología del lado del cliente

1. Lenguajes y preprocesadores.
 - a. JavaScript
 - b. TypeScript
 - c. CSS3
 - d. SASS
 - e. LESS
 - f. HTML5
 - g. Pug
2. Librerías de UI, frameworks y metaframeworks.
 - a. Angular
 - b. Next.js

- c. React
- d. Astro
- 3. Librerías de componentes y utilidades.
 - a. Bootstrap 5
 - b. Material UI
 - c. Material Design
 - d. Tailwind
 - e. Prime
 - f. Shadcn/ui
 - g. Mantine

3.5.2 Tecnología del lado del servidor

- 1. Lenguajes de programación.
 - a. C#
 - b. JavaScript
 - c. TypeScript
 - d. Python
- 2. Frameworks y metaframeworks.
 - a. .NET
 - b. NestJS
 - c. Next.js
 - d. FastAPI
 - e. Express.js
- 3. Base de datos.
 - a. MongoDB
 - b. PostgreSQL
 - c. MySQL
 - d. Redis
- 4. Servidores.
 - a. Apache
 - b. Nginx
 - c. Microsoft IIS
 - d. Vercel Edge Network

5. Sistemas Operativos.
 - a. Linux
 - b. Windows Server
 - c. FreeBSD

3.5.3 Plataformas de alojamiento basadas en Git

1. GitHub
2. GitLab

3.5.4 Sistemas de gestión de paquetes

1. Node Package Manager (NPM)
2. Performant Node Package Manager (PNPM)
3. Yarn

3.5.5 Sistemas de gestión de proyectos

1. Asana
2. Azure DevOps
3. Jira Software

3.5.6 IDEs, editores de textos, terminales de comando y gestores de código

1. Visual Studio
2. Visual Studio Code
3. Table Plus
4. Redis Insight
5. DBeaver
6. Sublime Text
7. Windows Powershell
8. Terminal (Mac)
9. Warp

3.5.7 Herramientas de despliegues y plataformas como servicio

1. Vercel
2. Heroku
3. Netlify
4. Amazon Web Service (AWS)

5. Render

6. Docker

4. SOLUCIÓN PROPUESTA

4.1 Presentación

En este capítulo se describe la solución aportada para abordar el problema de la optimización de rutas en el proceso de transporte. La solución consiste en el desarrollo de un sistema informático concebido como una aplicación web progresiva (PWA) modular, escalable y adaptable, capaz de integrar diversos contextos y casos de uso.

El sistema permite automatizar parcialmente el proceso de optimización de rutas en función de las particularidades del usuario, considerando restricciones y necesidades propias del contexto y caso de uso.

Tal como se expuso en el capítulo anterior, la validación de los objetivos se llevó a cabo mediante la implementación del sistema en dos casos de uso pertenecientes a contextos representativos: por un lado, la planificación de viajes asistidos con optimizaciones programadas, destinada a las personas; y por otro, la gestión de flotas en servicios a domicilio con optimizaciones programadas, dirigida a las empresas.

En consecuencia, el sistema se desarrolla bajo un mismo esquema funcional, diferenciando los módulos y funcionalidades según el tipo de usuario y el caso de uso abordado. Asimismo, la escalabilidad y adaptabilidad se lograron mediante el uso de patrones de diseño, principios y otras buenas prácticas de programación. La estructura general, además, sigue un diseño web adaptable, incorporando prácticas de accesibilidad orientadas a usuarios con algún tipo de discapacidad y contemplando lineamientos fundamentales de optimización para motores de búsqueda (SEO).

La funcionalidad principal del sistema es obtener una o más rutas optimizadas y presentar al usuario métricas relevantes según el caso de uso. En esta primera versión, no se busca replicar el funcionamiento de un sistema de navegación en tiempo real ni ofrecer funcionalidades de control de estadísticas avanzadas, sino proporcionar una guía de viaje o una hoja de ruta orientativa, dirigida a personas o empresas, respectivamente. Por ello, el sistema genera una optimización estática: no contempla actualizaciones dinámicas ni permite modificaciones sobre la marcha una vez generada la optimización. No obstante, estas capacidades podrían ser consideradas en versiones futuras, con el fin de ampliar el circuito funcional y brindar mayor adaptabilidad en escenarios operativos en tiempo real.

En resumen, la solución propuesta constituye un punto de partida sólido para abordar la optimización de rutas en el proceso de transporte. Sus características permiten cubrir diversas necesidades y facilitar su extensión sin comprometer la operabilidad del sistema. Asimismo, sienta las bases para el desarrollo de soluciones personalizadas que respondan a otras variantes del mismo problema.

4.2 Beneficios

La solución propuesta aporta una serie de beneficios significativos no solo en el plano técnico, sino también desde una perspectiva ambiental y económica. A continuación, se detallan los principales aportes para los distintos actores involucrados:

1. Mejora en la eficiencia del transporte. Al automatizar el proceso de optimización de rutas se reducen tiempos de desplazamiento y mejor utilización de recursos, lo que se traduce en operaciones más ágiles y económicas tanto para personas como para empresas.
2. Reducción del impacto ambiental. Al optimizar rutas y evitar trayectos innecesarios, se promueve un menor consumo de combustible y por ende una disminución de las emisiones de gases contaminantes.
3. Accesibilidad y equidad tecnológica. Al estar diseñado bajo prácticas de accesibilidad facilita el acceso a personas con diferentes capacidades ampliando el alcance de la solución y fomentando la inclusión tecnológica.
4. Autonomía del usuario en la toma de decisiones. Al brindar información detallada sobre las rutas, métricas y criterios aplicados permite que el usuario tenga mayor control y conocimiento sobre sus decisiones.
5. Potencial para mejorar la calidad de vida. Al automatizar el proceso de optimización se reduce la incertidumbre sobre las posibles rutas y trayectos que el usuario pueda tomar en un viaje o realización de servicio.
6. Base para futuras soluciones. Al tener un diseño modular y escalable, permite su extensión hacia nuevas funcionalidades, así como su utilización como base para el desarrollo de otros sistemas personalizados que aborden variantes del mismo problema.

4.3 Tecnologías utilizadas

Esta sección describe las tecnologías que fueron utilizadas en el desarrollo de la solución informática. Dado que la infraestructura fue delegada a servicios de terceros, no se especifican aspectos relacionados con hardware o redes; sin embargo, para mayor trazabilidad en la sección 4.9.1 se detallan estos aspectos.

Cabe destacar que la mayor parte de la funcionalidad fue implementada mediante los servicios de Google y OpenAI, los cuales resultaron fundamentales para dar solución al problema.

Con esto aclarado, a continuación, se describen las tecnologías de software implicadas en el desarrollo del sistema.

4.3.1 Frontend

1. Next.js

Next.js es un metaframework de React que proporciona bloques de construcción para crear aplicaciones web. Se utiliza para crear aplicaciones web totalmente interactivas, muy dinámicas, con un gran rendimiento y que brindan una excelente experiencia de usuario.

Es una tecnología utilizada por las mayores compañías del sector como Nike, Spotify, OpenAI, TikTok, Stripe, Notion, HBO Max, por mencionar algunos.

2. Shadcn/ui

Conjunto de componentes de interfaz accesibles y estilizados, utilizados para acelerar la creación de vistas modernas y accesibles conforme a WCAG 2.2.

4.3.2 Backend

1. NestJS

NestJS es un framework para construir aplicaciones Node.js eficientes y escalables del lado del servidor. Internamente Nest hace uso de sólidos framework de HTTP como Express, de forma nativa, y opcionalmente, puede configurarse para utilizar también Fastify.

Se lo utiliza para diseñar arquitectura basadas en monolitos como así también en microservicios. También permite el desarrollo de REST API, aplicaciones GraphQL, Web Socket, aplicaciones de consola y tareas programadas.

Entre sus colaboradores más importantes se destacan empresas de renombre como AUTODESK, IBM, JetBrains, BMW Mercedes Benz y GitLab.

4.3.3 Bases de datos y tareas programadas

El sistema utiliza tecnologías robustas para el almacenamiento estructurado de datos, el caching y la ejecución de tareas asincrónicas.

1. PostgreSQL

PostgreSQL también llamado "Postgres", es un avanzado sistema de gestión de bases de datos objeto-relacionales de código abierto.

Aunque principalmente es una base de datos relacional, Postgres va más allá de los modelos relacionales tradicionales al incorporar características orientadas a objetos, como la compatibilidad con JSON y los tipos de datos personalizados, lo que le permite gestionar los datos con mayor complejidad y flexibilidad.

Su fiabilidad, escalabilidad, manejo de altos volúmenes de datos, facilidad de uso y gran compatibilidad lo vuelven una opción sólida para utilizarlo en cualquier proyecto.

Empresas como Apple, Netflix, Uber, Instagram, Spotify y BMW, por mencionar algunas, ya están aprovechando todo lo que Postgres le puede ofrecer.

2. Redis

Redis acrónimo de Remote Dictionary Server (servidor de diccionario remoto), es un sistema de base de datos NoSQL de clave-valor y de código abierto.

Es compatible con cualquier lenguaje de programación lo que la hace muy fácil de usar y se destaca por ser extremadamente rápida debido a su almacenamiento en memoria. Esto la hace ideal para trabajar con sistemas de transmisión de eventos, la gestión de trabajos en cola y el almacenamiento en cache.

Empresas como Adobe, Uber, Intel, IBM, Mercedes-Benz y Jet Blue son algunas de la que ya confían en su uso.

Para el proyecto, la usamos como cache remota para almacenar la información proporcionada por Google Maps API y cumplir con las normativas que exige sobre él.

3. BullMQ

BullMQ es una biblioteca de Node que implementa un sistema de colas rápido y eficiente sobre Redis.

BullMQ está diseñado para cumplir con los siguientes objetivos:

- a. Semántica de cola de una sola vez: intenta entregar cada mensaje exactamente una vez, pero entregará al menos una vez en el peor de los casos.
- b. Fácil escalabilidad horizontal: permite añadir más trabajadores para procesar trabajos en paralelo.
- c. Consistencia.
- d. Alto rendimiento: consigue el máximo rendimiento posible de Redis combinando scripts eficientes escritos en LUA y una comunicación y sincronización constante entre procesos.

Nos permite crear la automatización de las optimizaciones de los viajes y las hojas de rutas, así como las notificaciones relacionadas.

4.3.4 Servicios y SDK de terceros

Se integraron APIs y servicios externos que ampliaron las capacidades funcionales del sistema, especialmente en lo referente a mapas, inteligencia artificial y gestión de medios.

1. Google Maps Platform

Google Maps Platform es una de las opciones más sólidas para integrar servicios de mapas y navegación en aplicaciones. Empresas como Uber, Tesla y Mercado Libre las utilizan en sus operaciones diarias.

Las APIs de Google Maps Platform permiten agregar de forma sencilla mapas interactivos, buscar ubicaciones, calcular rutas y obtener información de geolocalización, mejorando la experiencia del usuario con datos en tiempo real y visualizaciones dinámicas.

En el proyecto, se utilizan estos servicios para mostrar mapas, recuperar lugares y direcciones, obtener información de geolocalización y para llevar a cabo la optimización de la ruta.

2. Cloudinary

Cloudinary es una plataforma especializada en el almacenamiento, optimización y entrega de archivos multimedia con potentes APIs, widgets intuitivos y una interfaz amigable con el usuario.

Cuenta con la confianza de compañías como: Puma, Adidas, Hilton y Levis; y se integra fácilmente con Google, AWS, Salesforce, SAP y Shopify.

Lo usamos como servicio web para gestionar todo el contenido multimedia de la aplicación.

3. OpenAI

OpenAI es la empresa líder en la investigación e implementación de la inteligencia artificial multipropósito para crear herramientas que se beneficien de sus aplicaciones creando experiencias excepcionales.

Permite elegir entre diferentes modelos de razonamientos y ofrece un panel para desarrolladores que facilita la integración con cualquier aplicación a través de sus APIs.

Su capa gratuita es más que suficiente para poder experimentar y ofrece diferentes planes de pago según la necesidad.

La utilizamos como un servicio web para para la creación e interacción con el asistente inteligente de viaje.

4. Socket.IO

Socket.IO es una biblioteca que permite una comunicación bidireccional, de baja latencia y basada en eventos entre un cliente y un servidor. Brinda características muy demandadas como la reconexión, los reconocimientos y la difusión.

Es muy sencilla tanto del lado del cliente como del servidor y su documentación brinda varios ejemplos de cómo usarla en diferentes situaciones.

Con Socket.IO logramos implementar de forma sencilla WebSockets en nuestro proyecto, tanto del lado del cliente como del servidor para interactuar con las notificaciones en tiempo real.

4.3.5 Despliegue

Se utilizaron herramientas que facilitaron la configuración de entornos consistentes, el control de versiones, el despliegue continuo y el aseguramiento de calidad del código.

1. Git y GitHub

Sistema de control de versiones distribuido y plataforma de colaboración empleada para el versionado del código fuente, seguimiento de cambios y trabajo en equipo.

2. Render

Render es una plataforma como servicio que ayuda a los equipos de desarrollo a entregar productos rápidamente y a cualquier escala. Es fácil de usar, su plan gratuito es bastante accesible y soporta múltiples lenguajes de programación.

La utilizamos principalmente para alojar nuestra REST API en Nest en conjunto con las bases de datos de Postgres y Redis.

3. Vercel

Vercel es una plataforma como servicio que ofrece una solución robusta diseñada para simplificar el despliegue, la colaboración y el escalado de las aplicaciones web modernas.

Ofrece una rápida y fácil manera de desplegar proyectos web, con escalamiento automático y soporte de tecnologías web modernas. Cuenta con un generoso plan gratuito lo que lo hace ideal para proyectos pequeños y personales.

Vercel también tiene una interfaz fácil de usar e integraciones con herramientas de desarrollo populares como GitHub y GitLab.

La utilizamos principalmente para alojar nuestra aplicación web.

4.3.5 Herramientas de gestión

Durante el desarrollo se emplearon herramientas complementarias para la gestión ágil de tareas, modelado de procesos y diagramación.

1. Jira Software

Plataforma de gestión de proyectos ágil utilizada para planificar y organizar tareas bajo la metodología Kanban, con seguimiento de historias de usuario.

2. Bizagi Modeler

Herramienta utilizada para el modelado de procesos BPMN, clave para documentar y validar visualmente los flujos operativos del sistema.

3. Excalidraw

Aplicación de diagramación colaborativa empleada para realizar esquemas preliminares y prototipos visuales en etapas tempranas del diseño.

4.4 Proceso de desarrollo

Esta sección describe los distintos aspectos que complementan al proceso de desarrollo llevado a cabo para la construcción de la solución informática propuesta, el cual se vincula con la fase de ejecución del proyecto.

4.4.1 Ciclo de vida del desarrollo

Como se mencionó en la sección 2.3.9 del capítulo 2, el software atraviesa un conjunto de estados sufriendo transformaciones hasta llegar al producto final. Este proceso se conoce como ciclo de vida del software, el cual define las fases fundamentales del desarrollo.

En lo que respecta al proyecto, dicho ciclo se tradujo en un conjunto de fases organizadas que estructuraron el desarrollo del sistema. Este conjunto de fases es una modificación del procedimiento detallado del modelo de proceso incremental representado en la Figura 2 de la sección 2.3.9 del capítulo 2. En función de dicha adaptación, las fases que atravesó el desarrollo del sistema fueron las siguientes:

1. Investigación y comprensión del dominio.

Consiste en la búsqueda, análisis y comparación de la información necesaria para completar el desarrollo. Esta información incluye: el negocio, lo que existe, lo que se puede mejorar y lo que no existe o posiblemente no está implementado. Los temas tratados incluyen:

- a. Análisis del dominio.
- b. Restricciones y supuestos del mercado.
- c. Servicios de terceros y algoritmos de optimización de rutas.
- d. Buenas prácticas en el desarrollo de software.

Esta fase se llevó a cabo de forma única durante la etapa inicial del proyecto.

2. Recolección y análisis de requerimientos.

Consiste en determinar y documentar las necesidades particulares de los contextos y casos de uso que integrará el sistema, así como las necesidades generales del dominio. La documentación obtenida en esta fase proporciona una base para la definición de los incrementos del sistema y para el refinamiento de requerimientos funcionales de cada incremento, así como también nos da una visión del cronograma y división de tareas.

Esta fase se llevó a cabo de forma única durante la etapa inicial del proyecto.

3. Definición de incrementos del sistema.

Consiste en determinar la cantidad de incrementos que tendrá el sistema. Esta cantidad estará dada por el nivel de relación entre los requerimientos funcionales y su importancia para la operabilidad. En esta fase también se decide si el incremento contendrá requerimientos de cualquier módulo o módulos completos.

Esta fase se llevó a cabo de forma única durante la etapa inicial del proyecto.

4. Desarrollo de los incrementos del sistema.

Consiste en la ejecución de las actividades necesarias para cumplir con los incrementos del sistema. Las actividades planteadas son las siguientes:

- a. Refinamiento de requerimientos funcionales.

Consiste en desmenuzar los requerimientos funcionales del incremento en historias de usuarios, indicando la necesidad, el motivo de negocio y los criterios de aceptación. Adicionalmente, en esta actividad es posible la incorporación o eliminación de requerimientos funcionales.

b. Diseño del incremento.

Incluye el diseño de la arquitectura backend y frontend, la actualización del modelo de datos si corresponde, y la elaboración o modificación de la interfaz de usuario.

c. Codificación del incremento.

Consiste en la realización del código de acuerdo con los requerimientos refinados y el diseño previamente establecido. Se siguen buenas prácticas de desarrollo garantizando la escalabilidad y mantenibilidad del sistema.

d. Prueba del incremento.

Incluye la ejecución de pruebas manuales sobre las historias de usuario realizadas, con el objetivo de detectar errores y validar el cumplimiento de los criterios de aceptación.

e. Integración del incremento.

Consiste en la incorporación del incremento al sistema principal. Se realiza mediante la fusión de ramas en los repositorios correspondientes y la publicación de versiones alineadas con el esquema de versionado semántico adoptado.

f. Validación de la integración.

Se valida que las funcionalidades integradas operen correctamente dentro del sistema completo y que no generen conflictos con versiones anteriores. La validación se da por cumplida cuando se confirma el cumplimiento funcional.

Esta fase se repite para cada uno de los incrementos definidos.

5. Entrega del producto funcional.

Consiste en la preparación y presentación del producto funcional cumpliendo con lo redactado en la sección 3.3 del capítulo 3 y considerando los entregables propuestos.

Esta fase se llevó a cabo de forma única durante la etapa final del proyecto.

6. Mantenimiento del producto funcional.

Consiste en la corrección de errores e introducción de mejoras menores al producto funcional que ya opera.

Esta fase se llevó a cabo de forma única durante la etapa final del proyecto.

A continuación, se presenta la representación gráfica del modelo modificado:

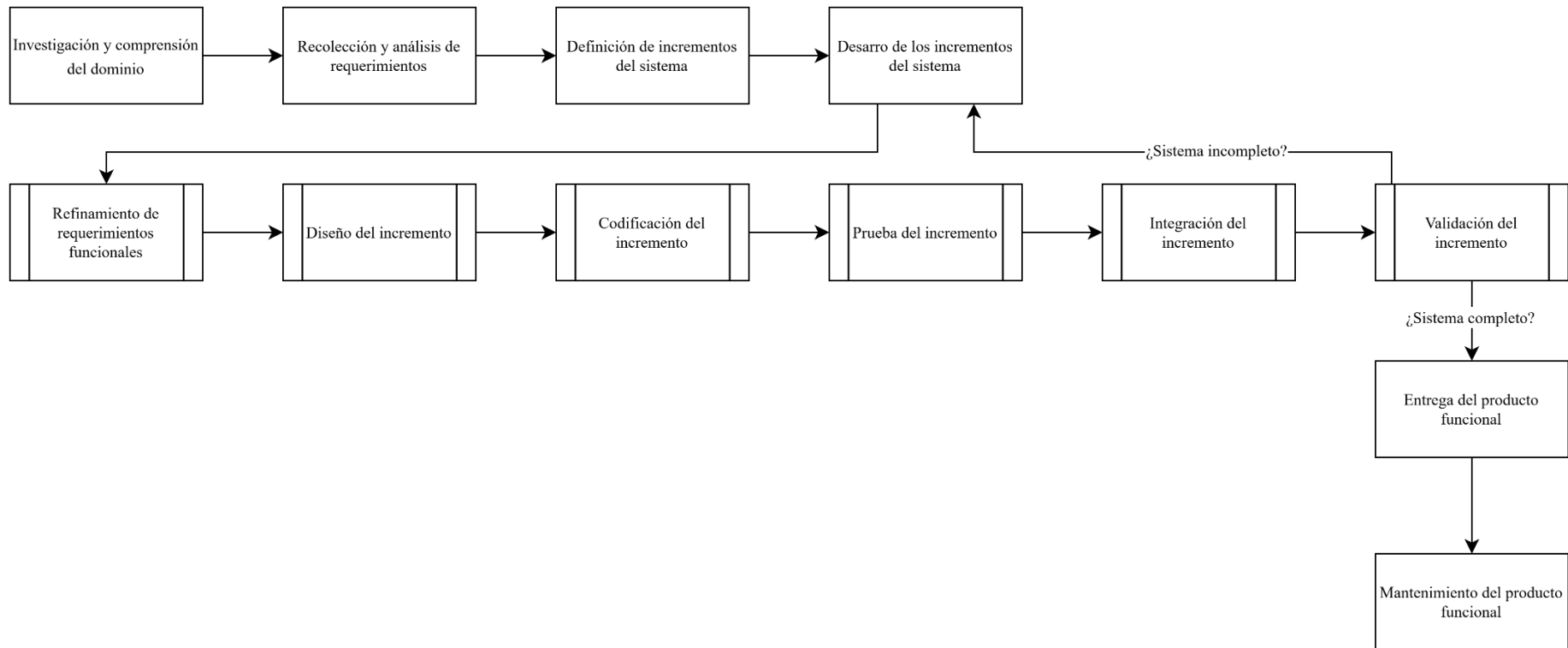


Figura 17. Modelo de proceso incremental aplicado al desarrollo del sistema

Con el fin de brindar trazabilidad del ciclo de vida del desarrollo y facilitar la consulta del lector, se presenta una relación de cada una de estas fases con acotaciones o secciones de la presente tesis:

1. La investigación y comprensión del dominio se desarrolla en parte del estudio realizado en el capítulo 1 y 2, y se complementa con la contextualización del problema y la explicación funcional de los casos de uso abordados en las secciones 3.2 y 3.4 del capítulo 3 respectivamente.
2. La determinación de requerimientos a grandes rasgos se logra con el estudio realizado en el capítulo 3 y la documentación obtenida se encuentra en la sección 4.4.4 del presente capítulo.
3. La cantidad de incrementos y la forma de implementación se encuentra en la sección 4.4.5 del presente capítulo.
4. El refinamiento de los requerimientos funcionales de cada incremento se encuentra incluido en las historias de usuario redactadas en la sección 4.4.6 del presente capítulo. En las secciones 4.5, 4.6 y 4.7 se presentan los detalles correspondientes al diseño del sistema, los cuales se abordan de manera global y no segmentada por incrementos, con el fin de proporcionar utilidad y facilitar la comprensión integral de la solución. La codificación de cada incremento se encuentra en los repositorios de la aplicación. La prueba del incremento se realizó de forma manual en todos los entornos, si bien no se documentaron casos de prueba se verificó el cumplimiento de cada historia de usuario en base a los criterios de aceptación definidos. La integración del incremento se llevó a cabo mediante la fusión de ramas y versionado, descrito en el flujo de trabajo redactado en la sección 4.4.3 del presente capítulo. La validación de la integración se asumió como cumplida al probar el correcto funcionamiento del incremento con el sistema principal.
5. El producto funcional corresponde al código fuente ubicado en los repositorios de la aplicación y los demás entregables detallados en las secciones 4.5, 4.6 y 4.7 del presente capítulo.
6. La corrección de errores e introducción de mejoras menores en el producto funcional se encuentran en la sección 4.4.7 del presente capítulo.

Si bien este ciclo de vida define las fases que atraviesa el desarrollo del sistema desde su concepción hasta el mantenimiento, no especifica cómo se gestionan, organizan o

ejecutan dichas fases. Para ello, se aplicó un enfoque metodológico y se estableció un flujo de trabajo, el cual se detalla en las siguientes secciones.

4.4.2 Enfoque metodológico aplicado

El sistema fue desarrollado mediante un enfoque híbrido que integra prácticas y principios del modelo de proceso incremental, el marco ágil Kanban y el desarrollo basado en componentes. Este enfoque permitió adaptar el flujo de trabajo a las particularidades del proyecto, organizar las actividades dentro de cada fase del ciclo de vida, optimizar el tiempo y garantizar entregas funcionales progresivas.

El enfoque propuesto integró las siguientes prácticas y principios:

1. Modelo de proceso incremental. El desarrollo del sistema se estructuró por incrementos, cada uno con un conjunto definido de funcionalidades. Esto facilitó la validación temprana de las funcionalidades mediante la entrega de versiones y permitió refinar los requerimientos de manera progresiva.
2. Marco ágil Kanban. El trabajo se gestionó mediante un tablero Kanban. Esto facilitó la organización y división de tareas necesarias para cada funcionalidad, así como la entrega de versiones.
3. Desarrollo basado en componentes. La codificación adoptó la reutilización de componentes visuales y funcionales integrados de librerías externas o creados de forma personalizada para el sistema. Esto incorporó escalabilidad y mantenibilidad.

El enfoque propuesto estructura el sistema en tres incrementos principales, cada uno incorporando módulos y funcionalidades autosuficientes como para operar. Para gestionar el trabajo se utilizó un tablero Kanban en Jira Software donde cada issue (épica, historia, bug, etc.) representa una tarjeta Kanban. Cada módulo se representó como una épica, cuyos requerimientos funcionales se desmenuzaron en historias de usuario. Estas historias se desagregaron en tareas específicas de backend y frontend. El tablero cuenta con tres columnas:

1. Por hacer: módulo, funcionalidad o tarea a realizar.
2. En curso: módulo, funcionalidad o tarea en progreso.
3. Listo: módulo, funcionalidad o tarea terminada.

Además, se incorporó un backlog donde se agruparon todos los requerimientos funcionales del sistema por versión. Cada versión menor representa un incremento del sistema. También se definieron etiquetas para diferenciar las tareas de backend, frontend o las desestimadas.

4.4.3 Flujo de trabajo y división de tareas

El flujo de trabajo seguido durante el desarrollo del sistema incorporó actividades individuales y grupales en cada una de las fases del ciclo de vida del software, descrito en la sección 4.4.1 de este capítulo. A través del uso de herramientas colaborativas como OneDrive, Discord, Meet y Office 365, junto con una estrategia de control de versiones mediante Git y una gestión del trabajo a través de Jira Software, se garantizó la continuidad y trazabilidad de cada actividad dentro de cada fase.

Fase de investigación y comprensión del negocio.

Durante esta etapa, el equipo trabajó en conjunto en la búsqueda, análisis y comparación de la información necesaria para el desarrollo del sistema. Para ello, se elaboraron documentos colaborativos. Cada integrante se enfocó en investigar y exponer un tema específico correspondiente a esta fase, con el objetivo de construir una visión compartida del dominio abordado. Si bien las investigaciones se realizaron de forma individual, se llevaron a cabo reuniones virtuales, para consensuar la redacción y el contenido final de los temas. En esta etapa también se definieron y desarrollaron los contextos y casos de uso a integrar con fines prácticos.

Fase de recolección y análisis de requerimientos

La determinación de las necesidades y su correspondiente documentación se realizaron de manera conjunta mediante reuniones virtuales elaborando documentos colaborativos. Mediante la implementación de la técnica brainstorming (lluvia de ideas), cada integrante expuso los módulos y funcionalidades con los que debería contar el sistema. Además, cada integrante dio su perspectiva técnica a considerar en términos de calidad. En esta etapa se debatió el objetivo, alcance y las alternativas tecnológicas definidas con el propósito de establecer a grandes rasgos los requerimientos del sistema. Estos requerimientos se categorizaron en funcionales a nivel módulo y no funcionales a nivel sistema.

Fase de definición de los incrementos del sistema.

Una vez definidos los requerimientos funcionales del sistema, el equipo determinó de manera conjunta la cantidad de incrementos necesarios, así como la estrategia de trabajo para cada uno de ellos; ya sea integrando requerimientos de diferentes módulos o abarcando módulos completos. Esta decisión se fundamentó en la relevancia funcional de cada requerimiento, considerando además que el sistema constituye una solución de producto y no una solución orientada a un cliente específico, lo que permitió una mayor flexibilidad en la estrategia de trabajo y en la priorización técnica del desarrollo.

Fase de desarrollo de los incrementos.

Teniendo definido los incrementos el equipo procedió a trabajar de manera colaborativa para cumplir con el objetivo. Para ello, de forma iterativa se realizó lo siguiente:

1. Se refinaron los requerimientos funcionales del incremento mediante la construcción de historias de usuario y la descomposición de estas en tareas de frontend y backend. Este procedimiento forma parte del enfoque metodológico aplicado. La construcción de historias de usuarios y tareas que la componen se realizó de manera conjunta, en donde cada integrante expuso sus conocimientos de acuerdo con su especialidad. La incorporación o eliminación de requerimientos funcionales se logró bajo el análisis de disponibilidad de tiempos y las capacidades de cada integrante.
2. El diseño del incremento fue creciendo en cuanto a arquitectura, modelo de datos e interfaz de usuario. Para obtener los diseños cada integrante intervino de acuerdo con su competencia. Si bien se tuvieron en cuenta las capacidades técnicas de cada integrante, el resultado es una combinación de los aportes de ambos.
3. Las actividades de codificación, prueba, integración y validación del incremento se llevaron a cabo de manera conjunta estableciendo una división de tareas de acuerdo con la especialidad de cada integrante. Para lograr esto, se definió una estrategia de control de versiones basada en el modelo GitFlow, cuya implementación se muestra en la Figura 18.

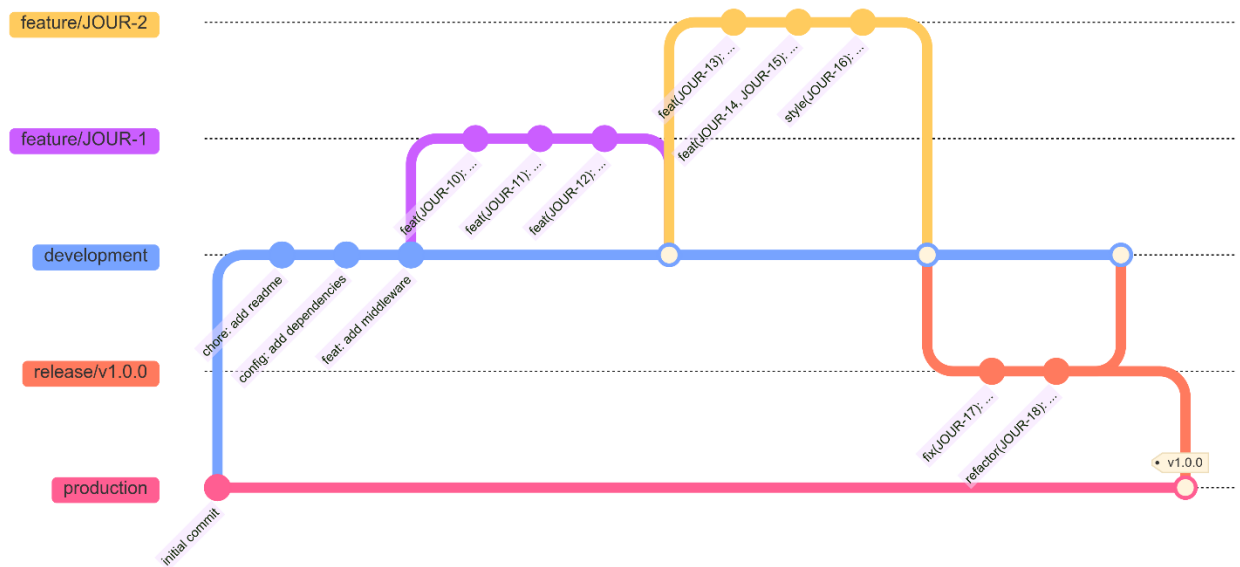


Figura 18. GitFlow adoptado en el proyecto. Se observan las ramas principales (development, production), las ramas de características (feature/*), la rama de lanzamiento (release/v1.0.0) y los prefijos de confirmación utilizados.

Esta estrategia siguió el siguiente estándar:

- a. Se establecieron dos entornos:
 - i. development: representa el entorno de desarrollo y es la rama que actúa como punto de partida de las nuevas características a desarrollar o liberar y como espacio de integración de las características del sistema desarrolladas antes de ser liberadas.
 - ii. production: representa el entorno de producción y es la rama principal que contiene las versiones estables del sistema.
- b. Para cada historia de usuario se crearon ramas features nacidas desde la rama development bajo el siguiente apartado:
 - i. El nombre de la rama debe ser feature/[identificador de historia en Jira Software].
 - ii. Las confirmaciones de cambios deben respetar una convención de prefijos que permiten identificar rápidamente la naturaleza de cada una de ellas:
 1. feat: incorporación de nuevas características.
 2. fix: corrección de errores.
 3. chore: cambios en la documentación y mantenimiento general.
 4. config: cambios en configuración o dependencias.

5. style: cambios de estilo, presentación o contenido.
6. refactor: reestructuración o mejoras en el código sin alterar funcionalidad.

Cada confirmación puede o no incluir un identificador de tarea definida en Jira Software. En caso de incluir múltiples tareas, se listan separadas por coma. Todos los mensajes de confirmación se deben redactar en inglés para obtener consistencia técnica.

- c. Cada historia de usuario completada debe ser fusionarse a la rama development para mantener el código actualizado.
- d. Para obtener cada versión se crearon ramas releases nacidas desde la rama development bajo el siguiente apartado:
 - i. El nombre de la rama debe ser: release/versión.
 - ii. Queda abierta la posibilidad de solucionar errores o introducir mejoras menores en la rama antes de enviar las características a producción.
 - iii. Se debe incrementar la versión menor.
 - iv. Para liberar las características, se debe fusionar la rama release a development y a production.
 - v. Como paso final se crea un tag en la rama production con el número de versión correspondiente empaquetando el código del incremento.
4. La codificación se trabajó en conjunto tomando como referencia la descomposición de tareas de las historias de usuarios definidas en el tablero Kanban. Salvo en ciertas situaciones se recurrió al uso de programación conjunta para esclarecer aspectos técnicos.
5. En la actividad de prueba del incremento, se utilizaron herramientas como Postman, para la validación de endpoints, y Google Lighthouse, para el análisis de rendimiento, accesibilidad y buenas prácticas en el frontend. Las pruebas se realizaron de forma manual e individual por cada integrante del equipo, tomando como referencia los criterios de aceptación definidos en las historias de usuario. Los resultados obtenidos permitieron detectar errores menores y aplicar los ajustes necesarios antes del empaquetado final y la generación de la versión correspondiente.

6. La actividad de integración y validación del incremento se realizó de manera conjunta mediante reuniones virtuales.

Fase de entrega del producto funcional.

Una vez concluido el desarrollo de todos los incrementos se preparó los entregables de la versión final del sistema. Esta etapa se realizó en conjunto mediante reuniones virtuales.

Fase de mantenimiento del producto funcional.

Una vez obtenida la versión final del sistema, se llevaron a cabo pruebas manuales con el objetivo de detectar errores y proponer mejoras menores. Estas pruebas fueron realizadas de forma individual por los integrantes del equipo. La corrección de errores y la incorporación de ajustes se efectuaron también de manera individual, generando nuevas versiones del sistema mediante la aplicación de parches. Dichas actualizaciones siguieron la misma estrategia de versionado adoptada durante la fase de desarrollo de los incrementos. Además, como resultado se actualizaron los entregables del proyecto.

En consecuencia, la ejecución del proyecto implicó una división de tareas basada en la especialidad, disponibilidad y experiencia técnica de cada integrante. Si bien muchas actividades se realizaron de forma colaborativa, se asignaron responsabilidades específicas para garantizar una ejecución eficiente y organizada. A continuación, se presenta una tabla resumen con la distribución general de las actividades a lo largo del ciclo de vida del desarrollo del sistema:

Tabla 2. División de tareas - Parte 1

Fase / Actividad	Daniel Ignacio Lamas	Joaquín Macaroff Pérez
Investigación y comprensión del dominio	Investigación y exposición del problema, antecedentes relacionados con el problema, contextos y casos de uso, fundamentos teóricos y buenas prácticas en el desarrollo de software.	Investigación y exposición de soluciones similares y supuestos del mercado, servicios y algoritmos de optimización de rutas existentes, tecnologías requeridas para el tratamiento del problema y temas pendientes de resolución.
Recolección y análisis de requerimientos	Exposición de módulos y funcionalidades. Análisis en conjunto del objetivo, alcance y alternativas tecnológicas con la intención de determinar y documentar los requerimientos del sistema. Redacción de consideraciones en términos de calidad relacionadas al SEO, accesibilidad, usabilidad, compatibilidad, portabilidad y rendimiento del sistema sobre todo desde una perspectiva de desarrollador frontend.	Exposición de módulos y funcionalidades. Análisis en conjunto del objetivo, alcance y alternativas tecnológicas con la intención de determinar y documentar los requerimientos del sistema. Redacción de consideraciones en términos de calidad relacionadas a la seguridad, resiliencia, escalabilidad, mantenibilidad, disponibilidad y rendimiento del sistema sobre todo desde una perspectiva de desarrollador backend.
Definición de incrementos del sistema	Determinación en conjunto de los incrementos necesarios y su correspondiente estrategia de trabajo.	Determinación en conjunto de los incrementos necesarios y su correspondiente estrategia de trabajo.
Refinamiento de requerimientos funcionales	Redacción de historias de usuario y división en tareas frontend.	Redacción de historias de usuario y división en tareas backend.
Diseño del incremento	Participación en el diseño de la arquitectura con perspectiva de desarrollador frontend. Diseño de la interfaz de usuario.	Participación en el diseño de la arquitectura con perspectiva de desarrollador backend. Modelado de los datos.

Tabla 3. División de tareas - Parte 2

Fase / Actividad	Daniel Ignacio Lamas	Joaquín Macaroff Pérez
Codificación del incremento	Resolución de tareas de frontend y backend. Creación y mantenimiento de la interfaz de usuario, componentes y acciones del lado del servidor. Integración mediante SDK de los servicios de terceros necesarios. Integración de cache y flujos de trabajo. Preparación de los entornos desarrollo y producción frontend.	Resolución de tareas de backend. Creación y mantenimiento de la base de datos y semilla. Creación y mantenimiento de módulos, servicios y controladores de la aplicación. Creación de vistas para las estadísticas. Preparación de los entornos de desarrollo y producción backend.
Prueba del incremento	Pruebas de rendimiento, SEO, accesibilidad y buenas prácticas. Pruebas manuales de navegación e interacción.	Pruebas de endpoints. Pruebas manuales de navegación e interacción.
Integración del incremento	Creación en conjunto de releases y tags en frontend.	Creación en conjunto de releases y tags en backend.
Validación del incremento	Validación en conjunto de cumplimiento funcional de las funcionalidades integradas.	Validación en conjunto de cumplimiento funcional de las funcionalidades integradas.
Entrega del producto funcional	Preparación y presentación de la interfaz de usuario, código fuente backend y frontend, y arquitectura de la aplicación.	Preparación y presentación del modelo de datos, código fuente backend y arquitectura de la aplicación.
Mantenimiento del producto funcional	Correcciones visuales e incorporación de mejoras menores en el frontend.	Correcciones de errores funcionales e incorporación de mejoras menores en el backend.

4.4.4 Recolección de requerimientos

La recolección de requerimientos constituye una etapa esencial dentro del proceso de desarrollo, ya que permite establecer con claridad qué debe ofrecer el sistema, tanto en términos de funcionalidades como de comportamiento técnico.

Dicha recolección se realiza a partir del análisis de los objetivos planteados, las características de los casos de uso considerados y la experiencia de los autores en el desarrollo de sistemas web. Este proceso permite delimitar el alcance del sistema y asegurar que las funcionalidades respondan a las necesidades concretas de cada caso de uso.

En este contexto, los requerimientos se agrupan en dos categorías principales:

1. Requerimientos funcionales, que describen las acciones y características operativas del sistema, es decir, lo que el sistema hace desde la perspectiva del usuario.
2. Requerimientos no funcionales, que hacen énfasis en las propiedades de calidad del sistema, tales como el rendimiento, la usabilidad, la accesibilidad, la seguridad y la portabilidad.

A continuación, se enumeran los requerimientos considerados para el desarrollo del sistema:

Tabla 4. Requerimientos funcionales del sistema

Identificador	Módulo	Requerimiento
JOUR-5	Autenticación	Log In con Credenciales
JOUR-87	Autenticación	Forgot Password
JOUR-8	Autenticación	Sign Up
JOUR-9	Autenticación	Log In con Google
JOUR-10	Autenticación	Log Out
JOUR-21	Configuración	Modificar credenciales de acceso
JOUR-55	Actividades	Registrar plantilla de actividades
JOUR-58	Actividades	Visualizar plantilla de actividades
JOUR-57	Actividades	Modificar plantilla de actividades
JOUR-56	Actividades	Eliminar plantilla de actividades
JOUR-238	Actividades	Gestionar actividades
JOUR-329	Notificaciones	Crear notificación
JOUR-63	Notificaciones	Visualizar notificaciones dentro del sistema
JOUR-65	Notificaciones	Eliminar notificación
JOUR-67	Notificaciones	Marcar notificaciones como leídas

Tabla 5. Requerimientos funcionales del sistema: Planificación de viajes asistidos

Identificador	Módulo	Requerimiento
JOUR-4	Configuración	Modificar datos del perfil
JOUR-26	Direcciones favoritas	Agregar dirección a favoritas
JOUR-28	Direcciones favoritas	Visualizar direcciones favoritas
JOUR-31	Direcciones favoritas	Modificar dirección favorita
JOUR-29	Direcciones favoritas	Eliminar dirección de favoritas
JOUR-35	Lugares favoritos	Agregar lugar a favoritos
JOUR-38	Lugares favoritos	Visualizar lugares favoritos
JOUR-37	Lugares favoritos	Modificar lugar favorito
JOUR-36	Lugares favoritos	Eliminar lugar de favoritos
JOUR-20	Optimización	Definir criterios básicos de optimización
JOUR-25	Optimización	Definir criterios avanzados de optimización
JOUR-316	Optimización	Visualizar los resultados
JOUR-12	Viajes	Registrar viaje
JOUR-15	Viajes	Visualizar viajes
JOUR-14	Viajes	Cambiar condición del viaje
JOUR-16	Viajes	Visualizar guía de viaje
JOUR-17	Viajes	Modificar datos de viaje
JOUR-326	Viajes	Eliminar viaje
JOUR-340	Viajes	Replicar viaje
JOUR-354	Estadísticas	Visualizar estadísticas de viajes

Tabla 6. Requerimientos funcionales del sistema: Gestión de flotas en servicios a domicilio

Identificador	Módulo	Requerimiento
JOUR-119	Configuración	Modificar datos del perfil
JOUR-64	Flotas	Registrar flota
JOUR-70	Flotas	Visualizar flotas
JOUR-68	Flotas	Modificar flota
JOUR-66	Flotas	Eliminar flota
JOUR-71	Flotas	Registrar vehículo
JOUR-74	Flotas	Visualizar vehículos
JOUR-73	Flotas	Modificar vehículo
JOUR-72	Flotas	Eliminar vehículo
JOUR-75	Flotas	Relacionar vehículos con una flota
JOUR-81	Flotas	Registrar conductor
JOUR-83	Flotas	Visualizar conductores
JOUR-84	Flotas	Modificar conductor
JOUR-82	Flotas	Eliminar conductor
JOUR-76	Flotas	Relacionar conductores con una flota
JOUR-30	Optimización	Establecer configuración de optimización
JOUR-358	Optimización	Visualizar los resultados
JOUR-297	Hojas de ruta	Registrar hoja de ruta
JOUR-298	Hojas de ruta	Visualizar hojas de ruta
JOUR-299	Hojas de ruta	Modificar datos de hoja de ruta
JOUR-300	Hojas de ruta	Cambiar estado de hoja de ruta
JOUR-301	Hojas de ruta	Eliminar hoja de ruta
JOUR-337	Hojas de ruta	Visualizar detalles de la hoja de ruta
JOUR-350	Estadísticas	Visualizar estadísticas de hojas de ruta

Tabla 7. Requerimientos no funcionales del sistema

Identificador	Categoría	Requerimiento
RNF-01	Rendimiento	El sistema debe permitir generar una ruta optimizada en menos de 20 segundos.
RNF-02	Usabilidad	La interfaz debe ser intuitiva, accesible y permitir a un usuario sin conocimientos técnicos completar una optimización en menos de 4 pasos.
RNF-03	Accesibilidad	El sistema debe cumplir con las pautas WCAG 2.2 en nivel AA para garantizar el acceso a personas con discapacidades visuales o motrices.
RNF-04	Compatibilidad	El sistema debe ser compatible con los principales navegadores modernos: Chrome, Firefox, Edge y Safari en sus últimas 2 versiones.
RNF-05	Portabilidad	La aplicación debe poder ejecutarse en dispositivos móviles, tablets, laptops y equipos de escritorio, adaptando el diseño según el tamaño de pantalla.
RNF-06	Seguridad	Las contraseñas deben almacenarse utilizando hash seguro.
RNF-07	Seguridad	Las sesiones deben contar con validación JWT.
RNF-08	SEO	Las páginas públicas deben contar con etiquetas meta, estructura semántica y optimización para motores de búsqueda.
RNF-09	Escalabilidad	El sistema debe permitir incorporar nuevos contextos y/o casos de uso, así como extender en funcionalidades sin necesidad de refactorizar los módulos existentes.
RNF-10	Mantenibilidad	El código fuente debe seguir principios SOLID, Clean Architecture y estar documentado para facilitar su evolución.
RNF-11	Disponibilidad	El sistema debe estar disponible al menos el 95% del tiempo, considerando el entorno actual de producción en Render y Vercel.
RNF-12	Resiliencia	El sistema debe notificar al usuario cuando haya fallas ya sea por caídas de los servicios externos o por algún otro motivo propio del mismo.
RNF-13	Legalidad	El sistema no debe recolectar ni almacenar datos sensibles sin consentimiento expreso del usuario, cumpliendo la Ley 25.326 de Protección de Datos Personales.
RNF-14	Internacionalización	El sistema debe permitir, en versiones futuras, la incorporación de soporte para múltiples idiomas sin alterar su estructura base.

4.4.5 Incrementos

El desarrollo del sistema se organizó en tres incrementos principales, alineados con versiones semánticas publicadas en los repositorios oficiales del proyecto. Para mantener una arquitectura desacoplada, el backend y el frontend fueron desarrollados en repositorios independientes, cada uno con su propio historial de versiones.

A pesar de estar separados, ambos repositorios respetaron un esquema de versiones sincronizadas a nivel de versión mayor y menor, mientras que los cambios relacionados a corrección de errores o mejoras menores pueden variar según el repositorio.

A continuación, se detallan los incrementos establecidos.

Tabla 8. Incrementos del sistema

Incremento	Versión	Descripción
Incremento 1	v1.0.0	Versión base del sistema. Incluye el módulo de autenticación, estructura del sistema, configuración inicial del entorno de desarrollo y despliegue en producción.
Incremento 2	v1.1.0	Versión que incorpora los módulos de configuración, direcciones favoritas, lugares favoritos, actividades y flotas.
Incremento 3	v1.2.0	Versión que incorpora los módulos de optimización, viajes, hojas de ruta y estadísticas, además de mejoras generales en el rendimiento, accesibilidad, SEO e interfaz de usuario.

4.4.6 Historias de usuario

Ante la necesidad de refinar progresivamente los requerimientos funcionales de cada incremento, se emplearon historias de usuario.

Una historia de usuario es una especificación funcional que invita a la conversación entre sus participantes con el propósito de generar una comunicación directa y sólida, favoreciendo que la definición de los aspectos funcionales se consolide progresivamente a partir del diálogo colaborativo [19].

En este sentido, y como se mencionó anteriormente, cada requerimiento funcional fue representado en Jira Software como una issue de tipo historia, en la cual se detalló como mínimo los siguientes elementos básicos:

1. Título de la historia.

Nombre del requerimiento funcional

2. Descripción.

Redacción de acuerdo con el siguiente formato:

Como (rol), necesito (funcionalidad), para (beneficio)

3. Criterios de aceptación con sus notas.

Condiciones específicas, claras y verificables que debe cumplir la historia, con la inclusión de notas aclarativas e informativas.

Además de los elementos básicos, cada historia de usuario podía complementarse con información adicional relevante para su correcta ejecución, como el informador, la versión, el módulo, el estado, las dependencias con otras historias, y una lista de subtarefas de backend y frontend asociadas. En particular, la inclusión de subtarefas fue obligatoria, ya que permitía descomponer la historia en unidades técnicas más manejables y asignables, facilitando su planificación y seguimiento en el tablero de trabajo.

El conjunto completo de historias de usuario se encuentra documentado en la herramienta Jira Software. La Tabla 9 y Tabla 10 representan un ejemplo completo y real de una historia de usuario teniendo en cuenta solamente los elementos básicos.

Tabla 9. Historia de usuario - Log In con Credenciales - Parte 1

Campo	Contenido
Identificador	JOUR-05
Título	Log In con Credenciales
Descripción	Como usuario personal o empresa, necesito autenticar mi usuario en el sistema, para ser identificado de manera única y tener acceso a las funcionalidades correspondiente de mi rol.

Tabla 10. Historia de usuario - Log In con Credenciales - Parte 2

Campo	Contenido
Criterios de aceptación	1. Debe existir una pantalla de inicio de sesión que contemple un formulario para el ingreso de las credenciales de acceso a la aplicación. 2. Se debe hacer que la pantalla siga un diseño adaptable de manera que se pueda abrir en dispositivos móviles, tablets, laptops y desktops. 3. Las credenciales de acceso al sistema son: a. Email: campo obligatorio de tipo email. b. Contraseña: campo obligatorio de tipo password. 4. No se deben validar que las credenciales de acceso sigan algún otro patrón o criterio además del especificado en el inciso 3. 5. Al enviar el formulario, el sistema debe verificar que las credenciales de acceso coincidan con las de algún usuario registrado. 6. En caso de que las credenciales de acceso sean correctas se debe redireccionar al usuario al INICIO o PANEL DE CONTROL de la aplicación. 7. En caso de que las credenciales de acceso sean incorrectas se debe mostrar una notificación al usuario indicando que los datos ingresados son inválidos. 8. La aplicación debe ser capaz de hacer la distinción entre usuarios personales y usuarios empresa permitiendo el acceso a ambos tipos. 9. La aplicación no debe permitir acceder a las funcionalidades correspondientes del rol si no se ha autenticado. 10. En caso de encontrarse una sesión activa, no se debe permitir al usuario navegar a esta funcionalidad y se debe redireccionar al usuario automáticamente al INICIO o PANEL DE CONTROL de la aplicación. 11. Las sesiones por defecto deben caducar pasado 1 día. 12. En la pantalla de inicio de sesión se debe proporcionar al usuario la posibilidad de recuperar la contraseña y generar sesiones largas por periodos de hasta un mes.

Para favorecer la claridad del contenido y una estructura más ordenada, se presentan a continuación algunas historias de usuario representativas, sin detallar sus criterios de aceptación.

Tabla 11. Historia de usuario - Definir criterios básicos de optimización (viajes personales)

Campo	Contenido
Identificador	JOUR-20
Título	Definir criterios básicos de optimización (viajes personales)
Descripción	Como usuario personal, necesito definir los criterios básicos que se usarán en la optimización de mi viaje.

Tabla 12. Historia de usuario - Definir criterios avanzados de optimización (viajes personales)

Campo	Contenido
Identificador	JOUR-25
Título	Definir criterios avanzados de optimización (viajes personales)
Descripción	Como usuario personal, necesito definir los criterios avanzados que se usarán en la optimización de mi viaje.

Tabla 13. Historia de usuario - Registrar viaje

Campo	Contenido
Identificador	JOUR-12
Título	Registrar viaje
Descripción	Como usuario personal, necesito guardar mi viaje, para que el sistema automáticamente lo optimice antes de su salida y pueda recibir asistencia durante el mismo.

Tabla 14. Historia de usuario - Establecer configuración de optimización (gestión de flotas)

Campo	Contenido
Identificador	JOUR-30
Título	Establecer configuración de optimización (gestión de flotas)
Descripción	Como usuario empresa, necesito establecer la configuración que se usará, para la optimización de una hoja de ruta.

Tabla 15. Historia de usuario - Registrar hoja de ruta

Campo	Contenido
Identificador	JOUR-297
Título	Registrar hoja de ruta
Descripción	Como usuario empresa, necesito registrar una hoja de ruta, para que el sistema automáticamente la optimice en el momento de la salida y cambie el estado dependiendo del momento en el que se encuentre.

Debido a que Jira Software no permite la visualización pública de los tableros del proyecto, la totalidad de las historias de usuario se encuentra documentada y disponible por incremento en:

<https://journeys-user-stories.vercel.app/>

4.4.7 Correcciones de errores y mejoras menores

Como se mencionó en la sección 4.4.5 los cambios correspondientes a correcciones de errores y mejoras menores pueden variar según el repositorio. Estos cambios se realizaron al finalizar el ultimo incremento, creando nuevas versiones del sistema.

A continuación, se detallan los cambios por repositorio:

Tabla 16. Correcciones de errores y mejoras: Backend

Versión	Descripción
v1.2.1	Se incorporó correcciones de lógica en el servicio de hojas de rutas y se eliminó el control de errores para Redis debido a que producía caídas y reinicios del sistema. Se refactorizó el esquema de validación de las variables de entorno eliminando las no utilizadas y agregando las faltantes. Se removió código deprecado.
v1.2.2	Se integró el servicio de optimización en los servicios de hojas de ruta y viajes para calcular la optimización en sus procesadores de trabajo de manera que el código sea más eficiente. Se reescribieron la forma de calcular los tiempos de ejecución en las tareas definidas para realizar las optimizaciones, cambio de estado y cambio de condición haciendo que el código sea más mantenible.
v1.2.3	Se corrigió el método de replicar viajes y se aplicó para que los viajes no se puedan replicar en el momento sino en 30 minutos. Se agrego validación de parte del backend para controlar el límite de conductores y vehículos a vincular en una flota.

Tabla 17. Correcciones de errores y mejoras: Frontend

Versión	Descripción
v1.2.1	Se cambiaron algunos textos y estilos de la interfaz de usuario a nivel general de manera que las notificaciones, campos y otros elementos tengan coherencia con el contexto y caso de uso.
v1.2.2	Se agrego lógica para la visualización de la guía de viaje, haciendo que esta solo se muestre 10 minutos antes de la salida. Se corrigió la visualización de la cantidad de conductores asignados en una flota. Se modificó el modal de replicación de viaje para que muestre una salida en 30 minutos y no al momento.

4.4.8 Cronograma del desarrollo

Esta sección expone la planificación y el seguimiento temporal realizados durante el desarrollo, considerando tanto las fases del ciclo de vida adoptado como los distintos incrementos implementados. Su elaboración facilitó la gestión del trabajo en equipo, el control ágil del avance, la priorización de funcionalidades y el ajuste de la carga de trabajo.

Con el propósito de garantizar la trazabilidad del desarrollo y documentar la ejecución de cada fase e incremento, a continuación, se presentan los cronogramas elaborados mediante la herramienta Jira Software y Excel.

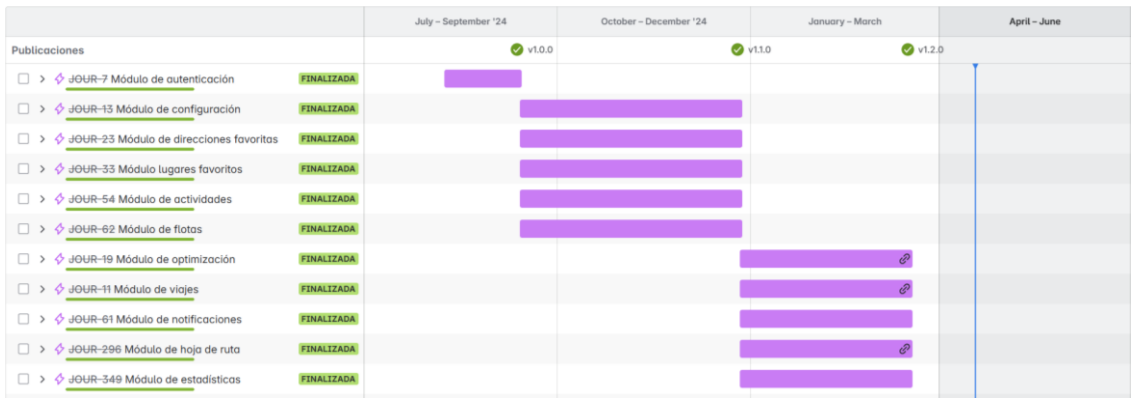


Figura 19. Cronograma de desarrollo de módulos del sistema organizado por trimestres

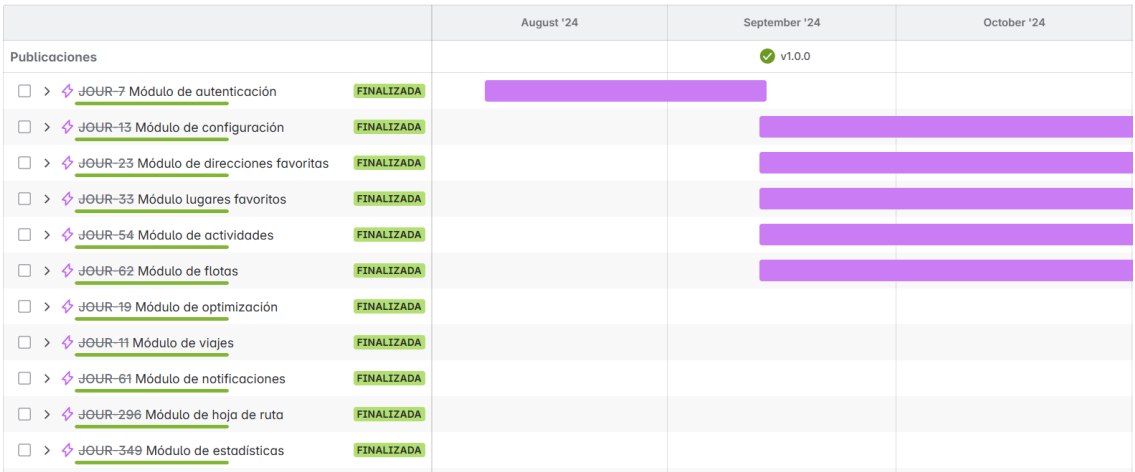


Figura 20. Cronograma de desarrollo de módulos del sistema organizado por meses – Parte 1

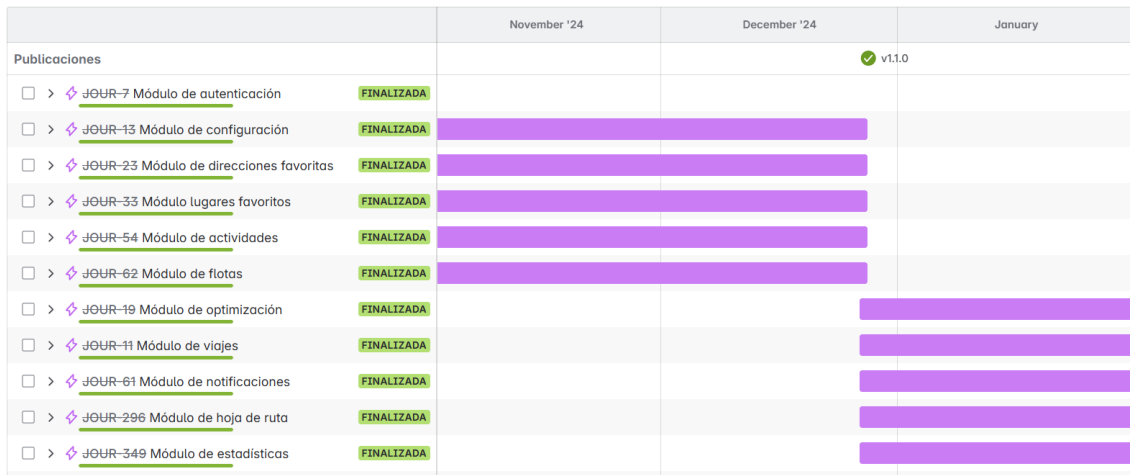


Figura 21. Cronograma de desarrollo de módulos del sistema organizado por meses – Parte 2

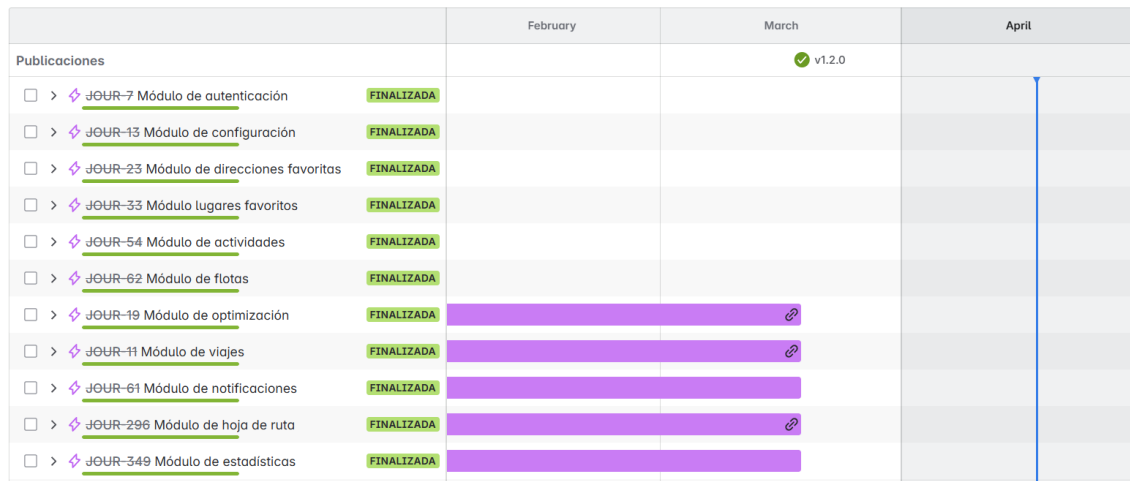


Figura 22. Cronograma de desarrollo de módulos del sistema organizado por meses – Parte 3

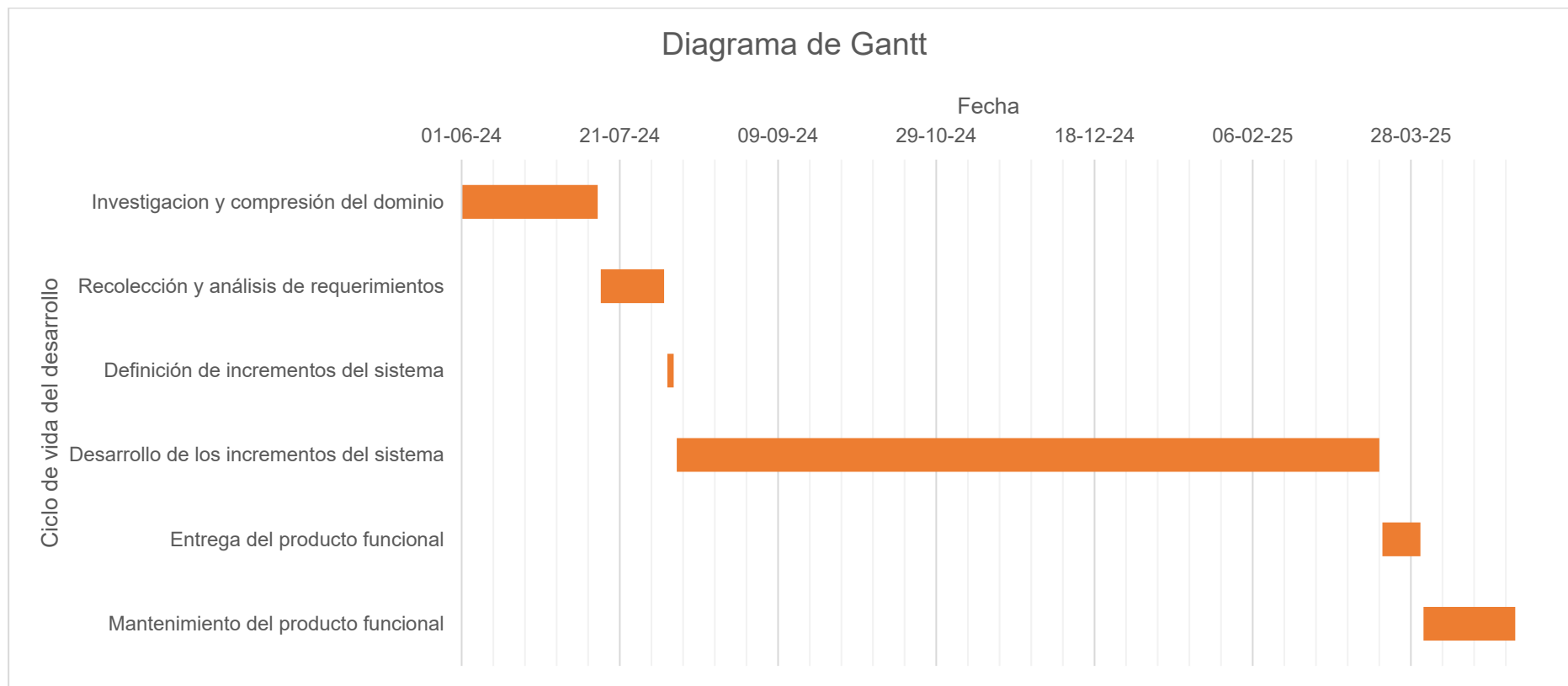


Figura 23. Cronograma de ciclo de vida del desarrollo

4.5 Desarrollo informatizado de los casos de uso

Con el propósito de esclarecer el funcionamiento de los casos de uso y su adaptación al sistema informático, esta sección describe los procesos más relevantes desarrollados durante la fase de ejecución del proyecto, detallando el flujo de actividades, actores involucrados y decisiones tomadas. Para ello, se emplea la notación gráfica BPMN (Business Process Model and Notation), que permite modelar visualmente estos elementos y facilita su comprensión tanto desde una perspectiva técnica como funcional.

4.5.1 Planificación de viajes asistidos

Los procesos relevantes involucrados en la planificación de viajes asistidos son:

1. Proceso de planificación de viajes asistidos
2. Subproceso reusable de manejo de intentos de optimización (máximo 5 intentos)
3. Proceso de asistencia inteligente de guía de viaje

El proceso de planificación de viajes asistidos, representado por las Figura 24 y Figura 25, es un proceso general que muestra desde una perspectiva funcional como se planifican los viajes en fases.

La primera fase, representada en la Figura 24, muestra la creación inicial del viaje y lo importante a destacar son las actividades de configuración de criterios, generación de optimización y presentación de resultados. Para entender que es lo que conlleva estas actividades se presenta el siguiente desglose:

1. Configuración de criterios

La configuración de criterios corresponde a dos actividades dentro del proceso que permiten configurar:

- a. Criterios básicos
 - i. Origen
 - ii. Destino
 - iii. Fecha de salida
 - iv. Hora de salida
 - v. Puntos de interés (hasta 5)
 - vi. Tipo de vehículo (automóvil o motocicleta)
 - vii. Consideraciones de tráfico
 - viii. Otras consideraciones (peajes, autopistas y transbordadores)

b. Criterios avanzados (opcional)

- i. Cálculos adicionales (costos de peaje, tráfico en tiempo real e instrucciones de navegación)
- ii. Ruta de referencia (distancia más corta)
- iii. Rutas alternativas
- iv. Optimización de orden en puntos de interés
- v. Inclusión de actividades en puntos de interés con duraciones.
- vi. Modelo de tráfico.

2. Generación de optimización

La generación de optimización es una actividad que mediante cálculos y la consulta a servicios externos obtiene los resultados de la optimización. Es aquí donde se hace uso de la biblioteca del cliente de Google Maps Platform para la implementación de Routes.

3. Presentación de resultados

La presentación de resultados es la actividad que complementa la generación de optimización formateando los resultados para una correcta visualización por parte del usuario.

La segunda fase, representada en la Figura 25, muestra la obtención de la guía de viaje y lo importante a destacar son las actividades de creación de evento programado de optimización y el almacenamiento de los resultados. Adicionalmente, en esta fase se implementa el subproceso reusable de manejo de reintentos de optimización (máximo 5 intentos). Para entender el funcionamiento de cada actividad se presenta la siguiente descomposición:

1. Creación de evento programado de optimización

La creación de evento programado de optimización es la actividad que crea un evento que se ejecuta 10 minutos antes de la hora de salida del viaje.

2. Almacenamiento de los resultados

El almacenamiento de los resultados es la actividad que guarda los resultados de la optimización para ser utilizados como guía de viaje.

El subproceso reusable de manejo de reintentos de optimización (máximo 5 intentos) es un proceso que realiza lo mismo que la actividad de generación de optimización de la

primera fase solo que incorpora cinco reintentos y un control de errores. La Figura 26 muestra el diagrama BPMN de este subproceso.

El proceso de asistencia inteligente de guía de viaje, representado en las Figura 27 y Figura 28, es un proceso que muestra desde una perspectiva funcional y técnica como se consulta la guía de viaje y como se interactúa mediante el asistente de IA con ella, es decir, que cuenta de dos fases. El proceso de por si es sencillo de entender por lo que no es de suma importancia el explicar que hace cada actividad, solo basta con esclarecer que en la fase de interacción con el asistente de IA se utiliza OpenAI RealTime y la información de la guía de viaje para crear una sesión de asistencia inteligente. La Figura 27 muestra la fase de consulta de la guía y la Figura 28 presenta la fase de interacción con el asistente de IA.

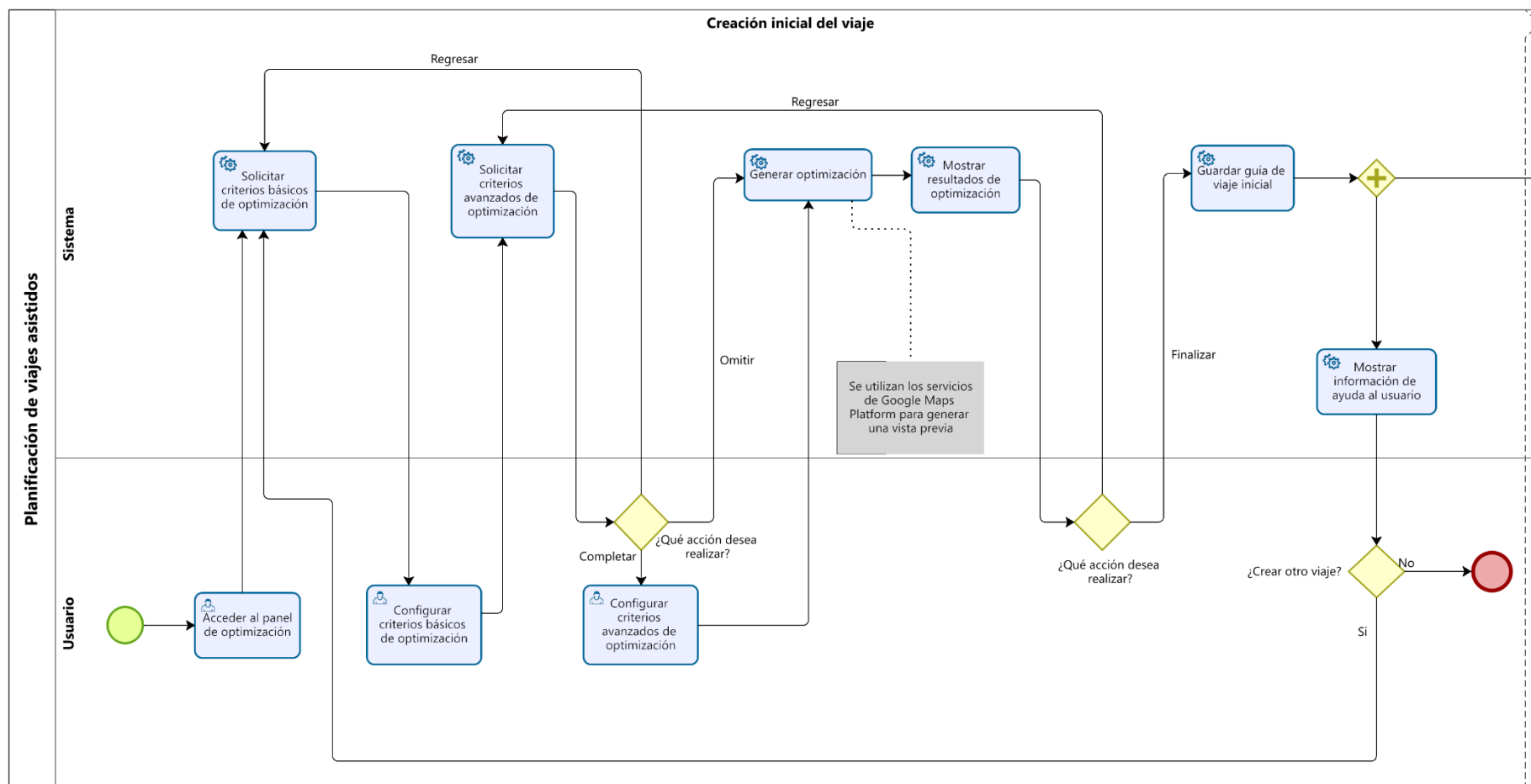


Figura 24. Diagrama BPMN del proceso de planificación de viajes asistidos – Parte 1

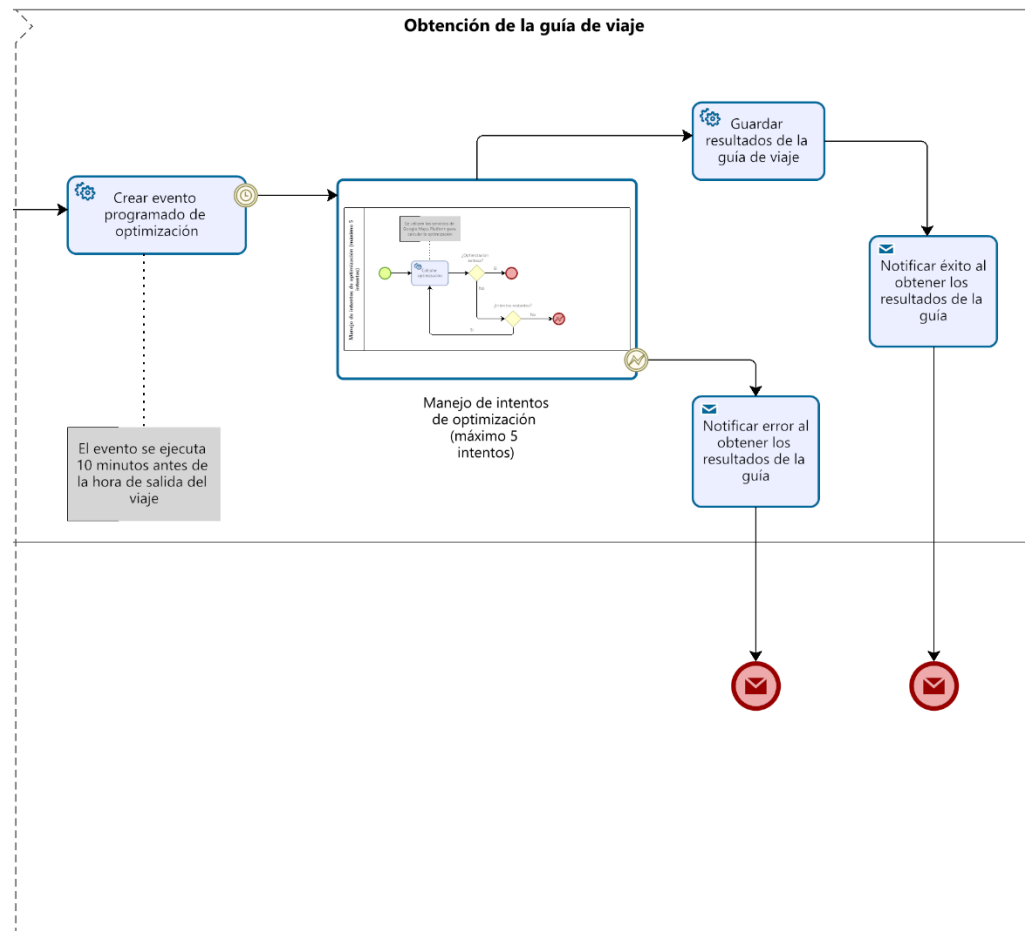


Figura 25. Diagrama BPMN del proceso de planificación de viajes asistidos – Parte 2

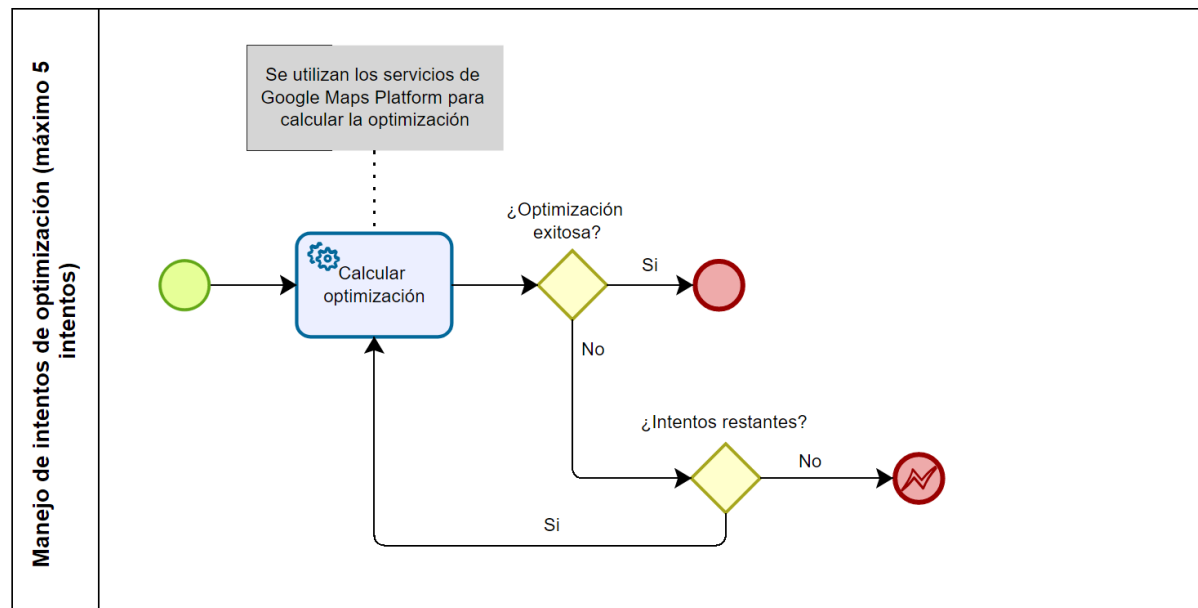


Figura 26. Diagrama BPMN del subproceso reusable de manejo de intentos de optimización (máximo 5 intentos)

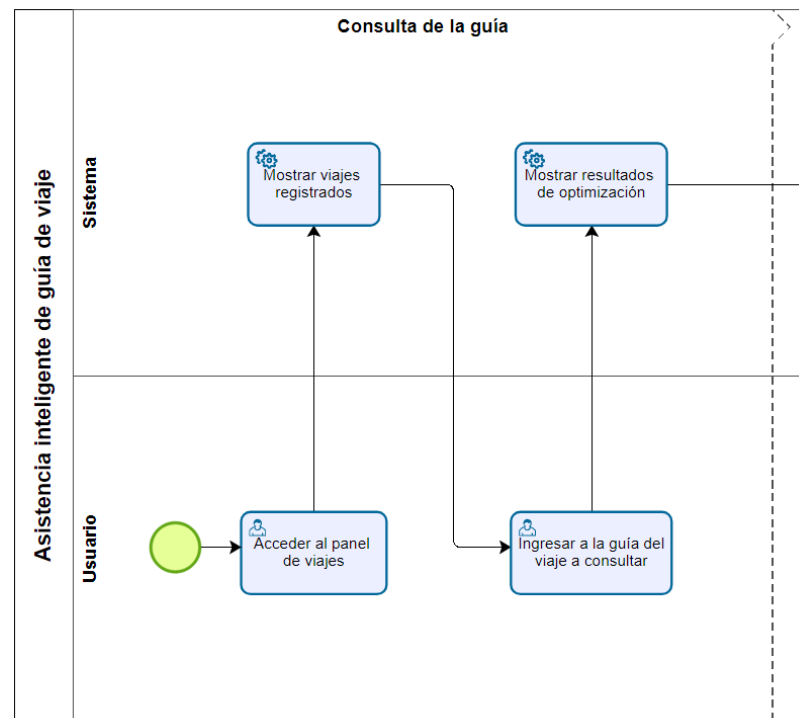


Figura 27. Diagrama BPMN del proceso de asistencia inteligente de guía de viaje – Parte 1

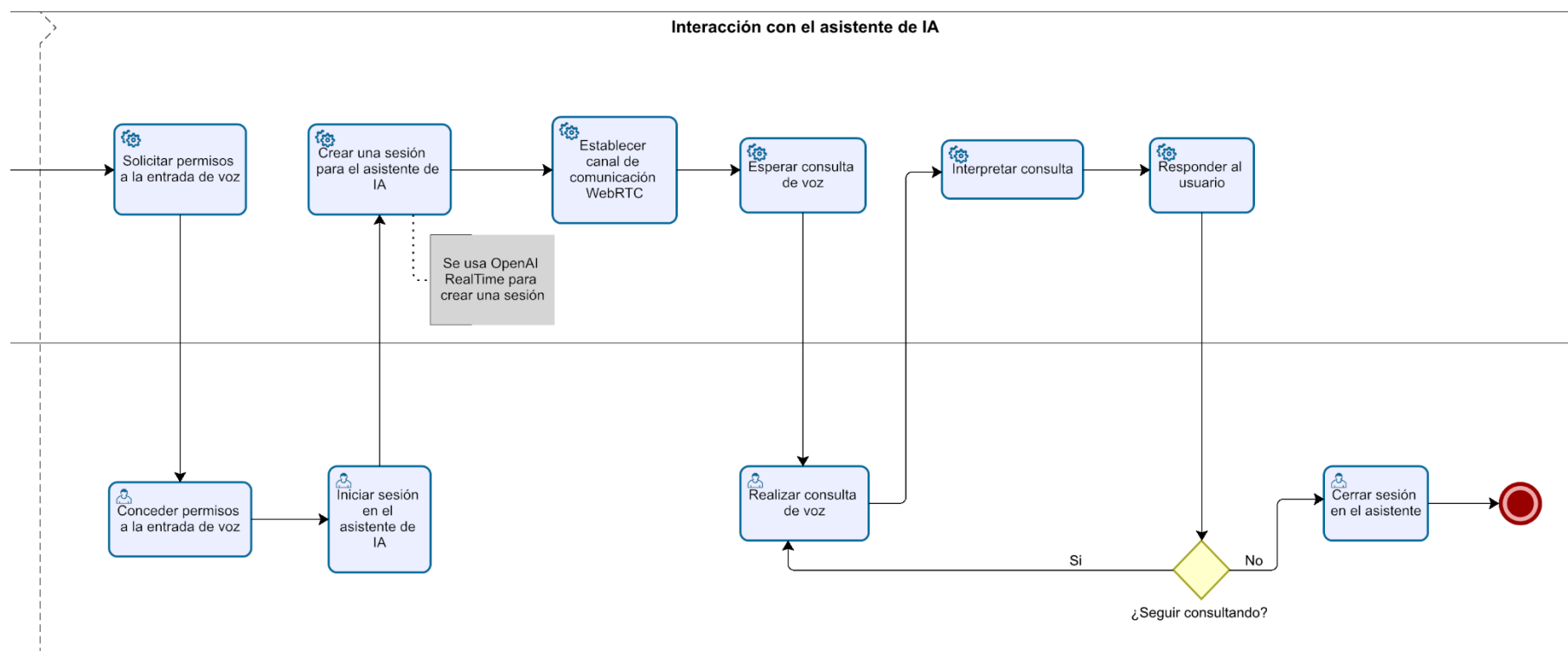


Figura 28. Diagrama BPMN del proceso de asistencia inteligente de guía de viaje – Parte 2

4.5.2 Gestión de flotas en servicios a domicilio

Los procesos relevantes involucrados en la gestión de flotas en servicios a domicilio son:

1. Proceso de gestión de flotas en servicios a domicilio
2. Subproceso reusable de manejo de intentos de optimización (máximo 5 intentos)

El proceso de gestión de flotas en servicios a domicilio, representado en las Figura 29 y Figura 30, es un proceso general que muestra desde una perspectiva funcional como se gestiona una flota mediante fases. Para entender la optimización y administración del uso de los recursos y del recorrido se necesita explicar cada una de estas fases.

La primera fase, representada en la Figura 29, muestra la creación inicial de la hoja de ruta que permite optimizar y administrar el uso de los recursos y del recorrido en la realización de un conjunto de servicios a domicilio para un determinado día. Lo importante a destacar en esta fase son las actividades de configuración por etapas, la generación de optimización y la presentación de los resultados. Para comprender que es lo que conllevan estas actividades se presenta la siguiente descomposición:

1. Configuración por etapas

La configuración por etapas corresponde a tres actividades dentro del proceso que permiten configurar distintos aspectos en la gestión de flotas:

- a. Primera etapa
 - i. Ubicación inicial (salida)
 - ii. Ubicación final (llegada)
 - iii. Fecha (operación)
 - iv. Hora de inicio (operación)
 - v. Hora de fin (operación)
 - vi. Flota asignada
 - vii. Vehículo asignado
 - viii. Conductor asignado
 - ix. Otras consideraciones (peajes, autopistas, transbordadores, consideraciones de tráfico)
- b. Segunda etapa
 - i. Servicios a domicilio con su dirección y duración.
- c. Tercera etapa (opcional)
 - i. Perfiles de costos.

ii. Límites de tiempo y distancia.

2. Generación de optimización

La generación de optimización es una actividad equivalente a la explicada en el proceso de planificación de viajes asistidos primera fase, solo que implementa Routes Optimization en lugar de Routes.

3. Presentación de resultados

La presentación de resultados es una actividad equivalente a la explicada en el proceso de planificación de viajes asistidos primera fase.

La segunda fase, representada en la Figura 30, muestra la obtención de los resultados definitivos y lo importante a destacar son las actividades de creación de evento programado de optimización y el almacenamiento de los resultados. Adicionalmente, en esta fase, también se implementa el subproceso reusable de manejo de reintentos de optimización (máximo 5 intentos) de igual manera que en el proceso de planificación de viajes asistidos segunda fase. Para entender el funcionamiento de cada actividad se presenta la siguiente descomposición:

1. Creación de evento programado de optimización

La creación de evento programado de optimización es la actividad que crea un evento que se ejecuta 1 hora antes de la hora de comienzo de la operación.

2. Almacenamiento de los resultados

El almacenamiento de los resultados es la actividad que guarda los resultados definitivos de la optimización para ser utilizados y presentados como guía operativa para el personal encargado de la ejecución.

En la sección anterior se explicó el funcionamiento del subproceso reusable de manejo de reintentos de optimización (máximo 5 intentos).

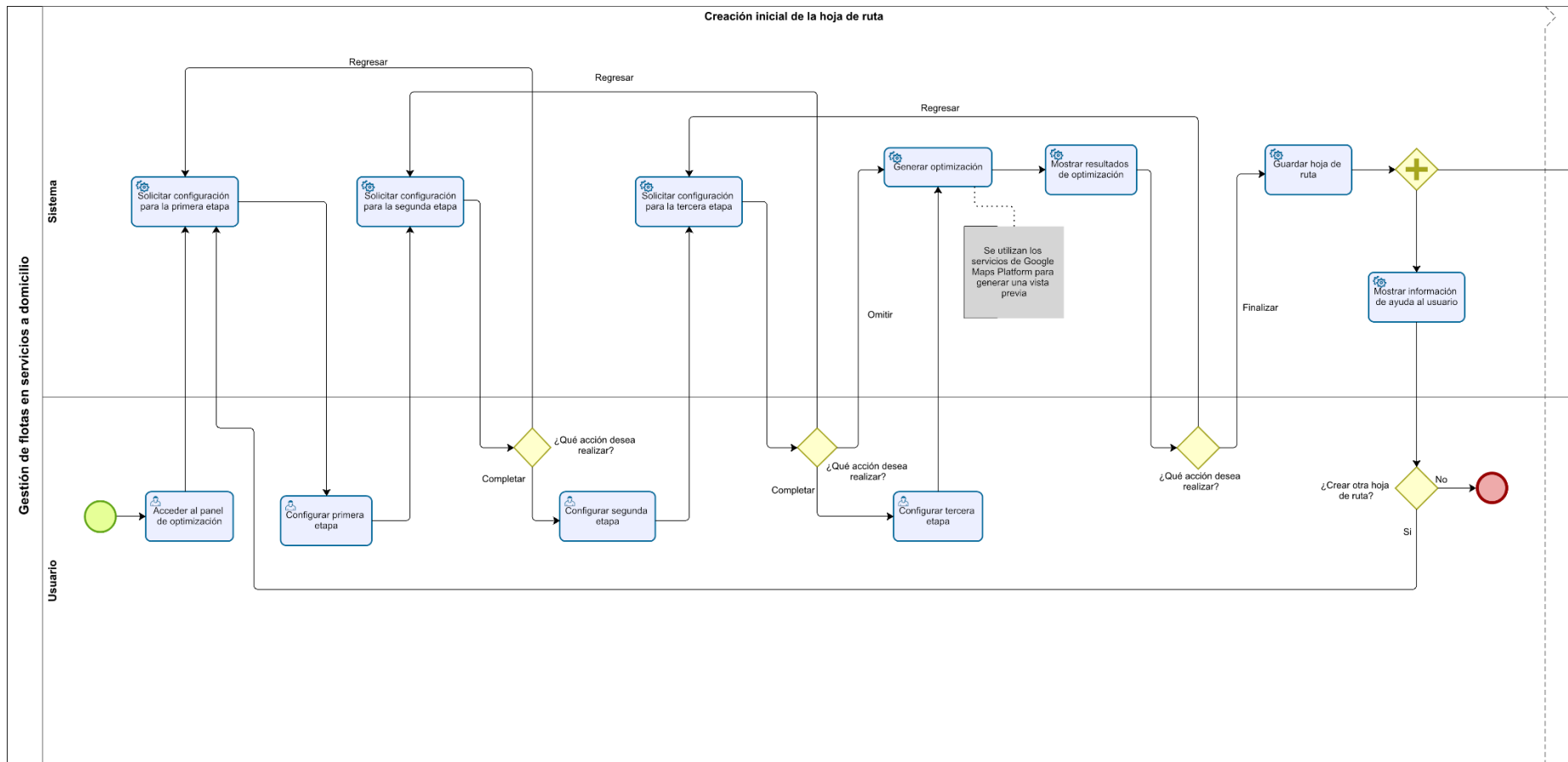


Figura 29. Diagrama BPMN del proceso de gestión de flotas en servicios a domicilio – Parte 1

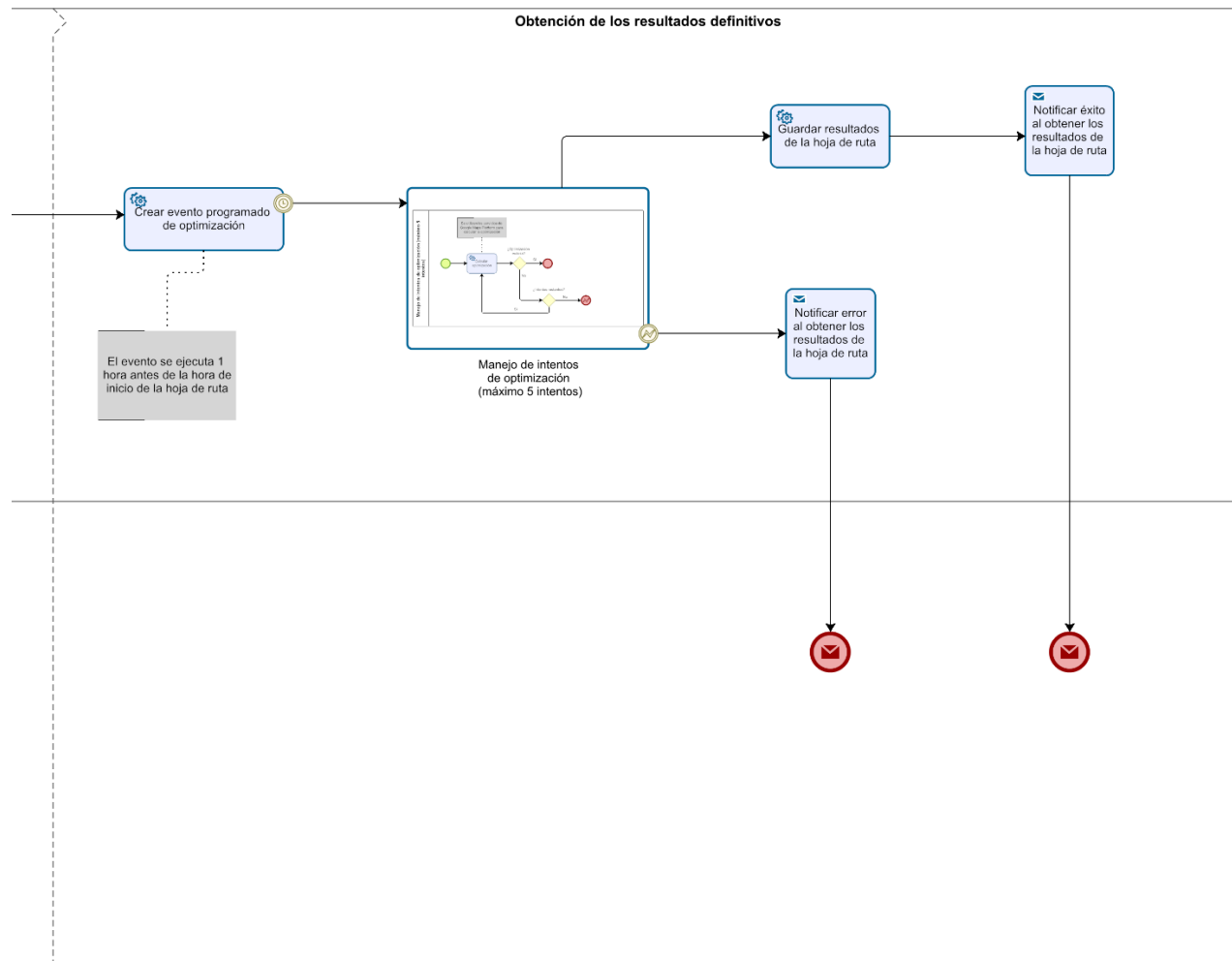


Figura 30. Diagrama BPMN del proceso de gestión de flotas en servicios a domicilio – Parte 2

4.6 Modelado de datos

El modelado de datos es un proceso que consiste en crear una representación visual de los datos y sus relaciones dentro de la organización. Brindan claridad sobre cómo se estructuran, almacenan y acceden a los datos para garantizar su coherencia y claridad en la gestión de datos

Entre los muchos beneficios de modelar los datos, destacamos que:

1. Permite a los equipos organizar la información de manera eficiente.
2. Mejora el rendimiento y la toma de decisiones.
3. Permite a diferentes aplicaciones compartir la misma información.
4. Sirve como documentación de la base de datos.
5. Ayuda a los analistas de datos, científicos e ingenieros a tener una vista unificada de los datos.

Para representar el modelo de datos, en el proyecto se realizó un diagrama de entidad-relación (DER) siguiendo las siguientes convenciones:

1. Se usan nombres representativos, sin abreviaturas y en inglés.
2. Para toda la nomenclatura se utiliza snake_case.
3. Las entidades se escriben en plural.
4. Los atributos de las entidades se escriben en singular.
5. Los atributos que son claves primarias se nombran simplemente como “id” y son del tipo uuid.
6. Los atributos que son claves foráneas llevan el nombre de la entidad con la que se relacionan más el sufijo “_id”. Por ejemplo “user_id” hace referencia a la entidad “user”. Si la entidad no existe se trata de una relación con datos externos, como por ejemplo “place_id” que guarda el identificador del lugar que brinda google.
7. Las claves primarias se indican con PK, las foráneas con FK y las únicas con UK. Generalmente las claves únicas se componen del atributo más el id de la entidad.
8. En las entidades que los requieran, se agregan los atributos de auditoría “created_at” y “updated_at”.

Con el objetivo de facilitar la comprensión del diagrama, este fue dividido en partes según los distintos casos de uso. Es importante aclarar que las entidades “accounts”, “notifications” y “activity_templates” están presentes en ambos casos, pero se han

incluido únicamente en la parte correspondiente a la planificación de viajes asistidos, a fin de simplificar la representación.

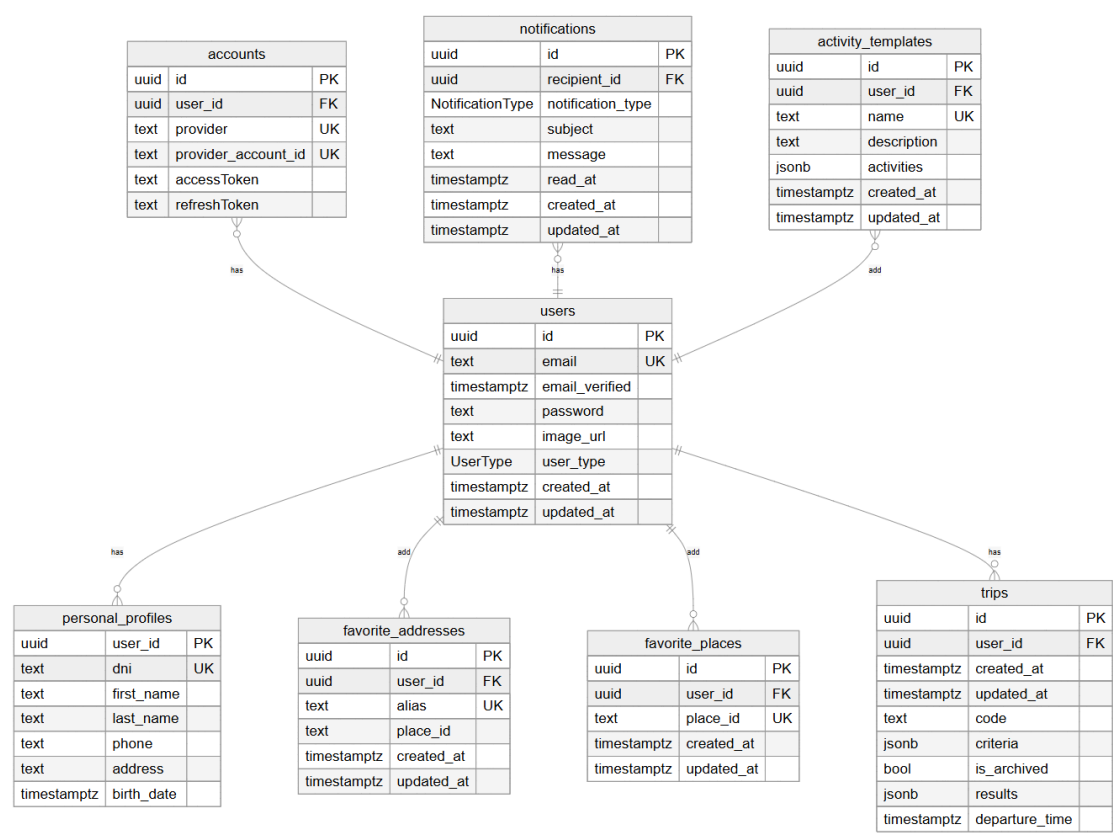


Figura 31. DER para la planificación de viajes asistidos

La Figura 31 presenta la primera parte del DER, donde se modela la estructura de datos correspondiente a la planificación de viajes asistidos.

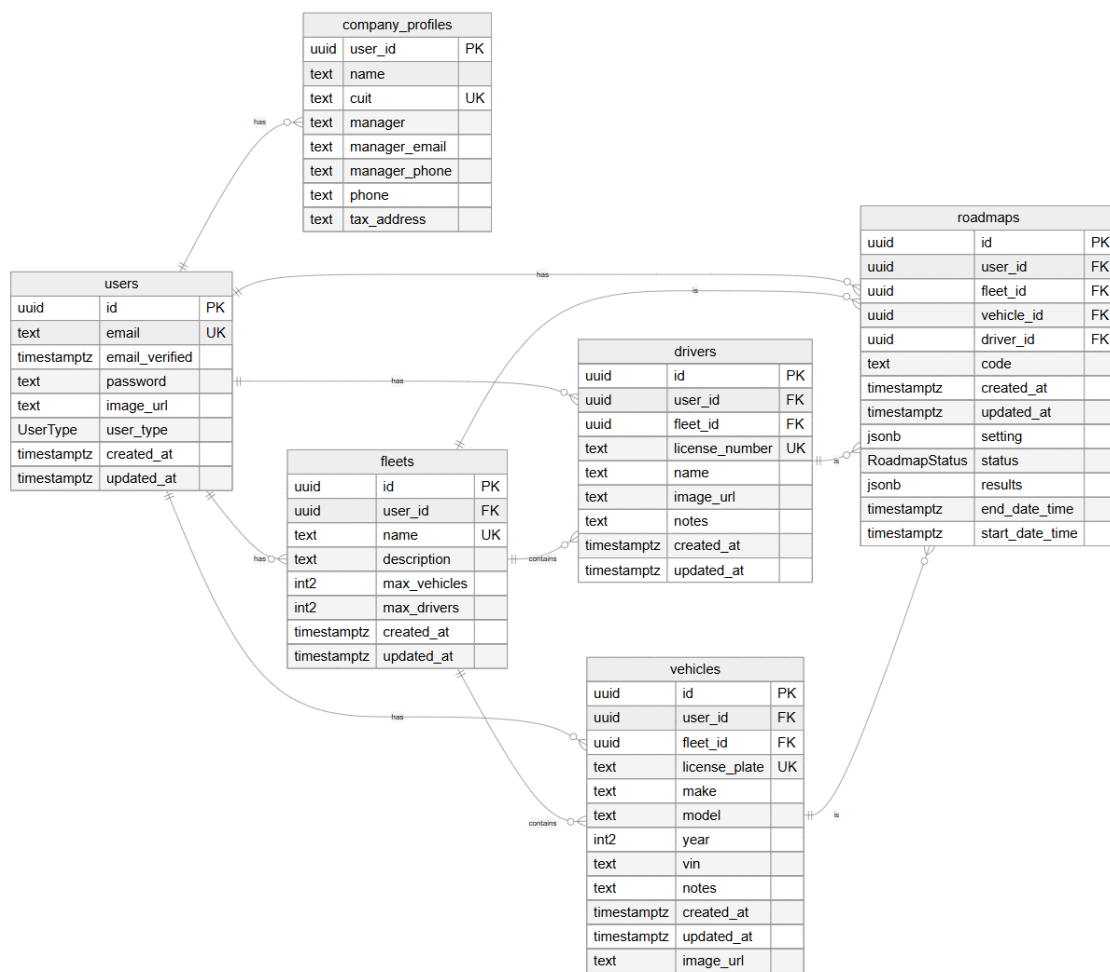


Figura 32. DER para la gestión de flotas en servicios a domicilio

La Figura 32 presenta la segunda parte del DER, donde se modela la estructura de datos correspondiente a la gestión de flotas en servicios a domicilio.

4.7 Arquitectura de la aplicación

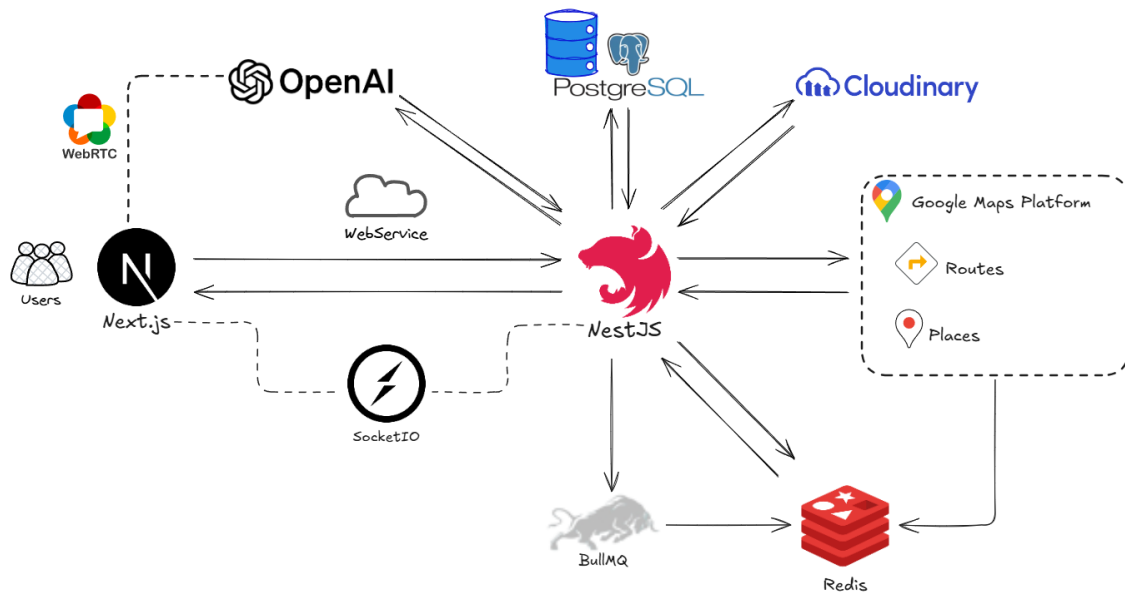


Figura 33. Arquitectura de la aplicación

La solución sigue un estilo de arquitectura donde el frontend (interfaz de usuario) está separado del backend (lógica del negocio). Con este enfoque conseguimos una flexibilidad técnica inigualable que nos permitió una mejor división de tareas, un desarrollo más rápido y brindar una experiencia de usuario fluida.

Desde el frontend llega una solicitud al backend mediante una API, el backend procesa esa petición y devuelve la información al frontend. El frontend utiliza esta información para presentarla al usuario.

Se utiliza Socket.IO para establecer la conexión por WebSocket y recibir notificaciones en tiempo real del procesamiento de un viaje o una hoja de ruta.

Para que el usuario pueda interactuar con el asistente inteligente, el backend recupera una key de OpenAI y el frontend la utiliza para crear una conexión en tiempo real WebRTC, para que funcione el usuario tiene que permitir el acceso al micrófono del dispositivo.

La lógica de negocio es manejada por el backend y toda la información propia del negocio con la que opera se persiste en una base de datos relacional Postgres para garantizar que esté disponible en tiempo y forma.

La información geoespacial y las optimizaciones provienen de Google Maps Platform, concretamente los lugares y las direcciones son del servicio Places y la optimización del servicio Routes.

Cuando se consulta por algún lugar o dirección se chequea primero si esa información existe en la base de datos clave-valor de Redis para brindarla de forma inmediata sin interactuar con Google. Al no existir, se hace la consulta correspondiente a través del SDK de Google Places y se almacena el resultado para la próxima consulta.

Con BullMQ, librería que trabaja con Redis, creamos una tarea programada cuando se registra un nuevo viaje o hoja de ruta para que minutos antes del horario de partida se le brinde al usuario la mejor ruta de acuerdo con sus preferencias configuradas y a las condiciones de tráfico actuales desde los servicios de Google Routes. También lo utilizamos para el cambio de estado automatizado de la hoja de ruta y el envío de notificaciones según el resultado de la ejecución de la tarea.

El manejo de multimedia se hace a través de Cloudinary mediante su SDK para aligerar la carga de nuestra base de datos principal y contar con un servicio especializado en el manejo de imágenes.

4.8 Cronograma

Según la Guía del PMBOK (6.^a edición), el cronograma del proyecto debe reflejar las actividades planificadas junto con sus respectivas fechas, duraciones, hitos y recursos asignados [40].

Si bien esta guía establece un conjunto completo de procesos para la gestión del cronograma, en este trabajo se presenta una versión simplificada que contempla los elementos mínimos necesarios para comprender la evolución temporal del proyecto.

El cronograma fue elaborado utilizando la herramienta Microsoft Project e incluye todas las etapas del ciclo de vida del proyecto, desde el inicio hasta su cierre. Además, se detallan la duración estimada de cada tarea, su secuencia lógica y los solapamientos entre actividades, proporcionando una visión clara de la estructura temporal adoptada. La Figura 34, Figura 35 y Figura 36 muestran el cronograma del proyecto incluyendo un diagrama de Gantt complementario.

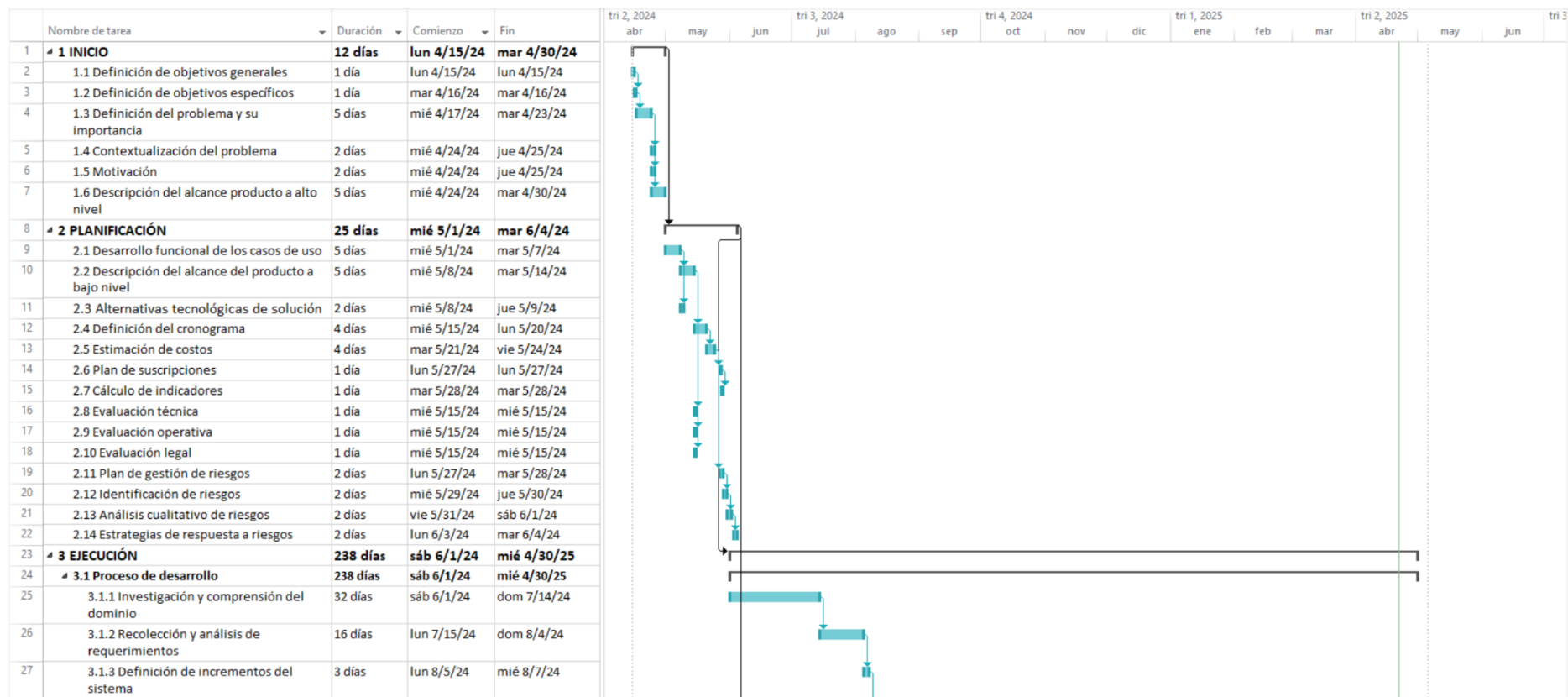


Figura 34. Cronograma del proyecto - Parte 1

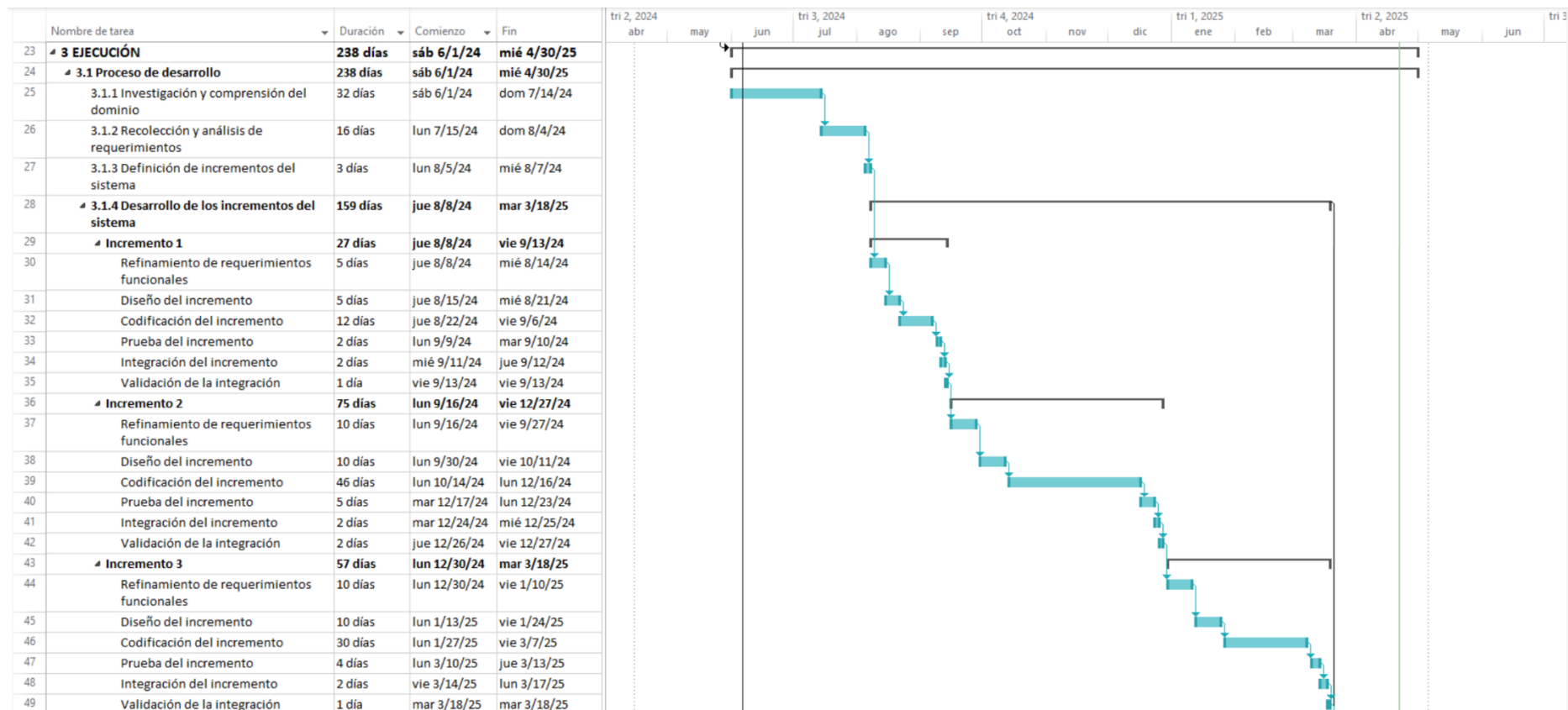


Figura 35. Cronograma del proyecto – Parte 2

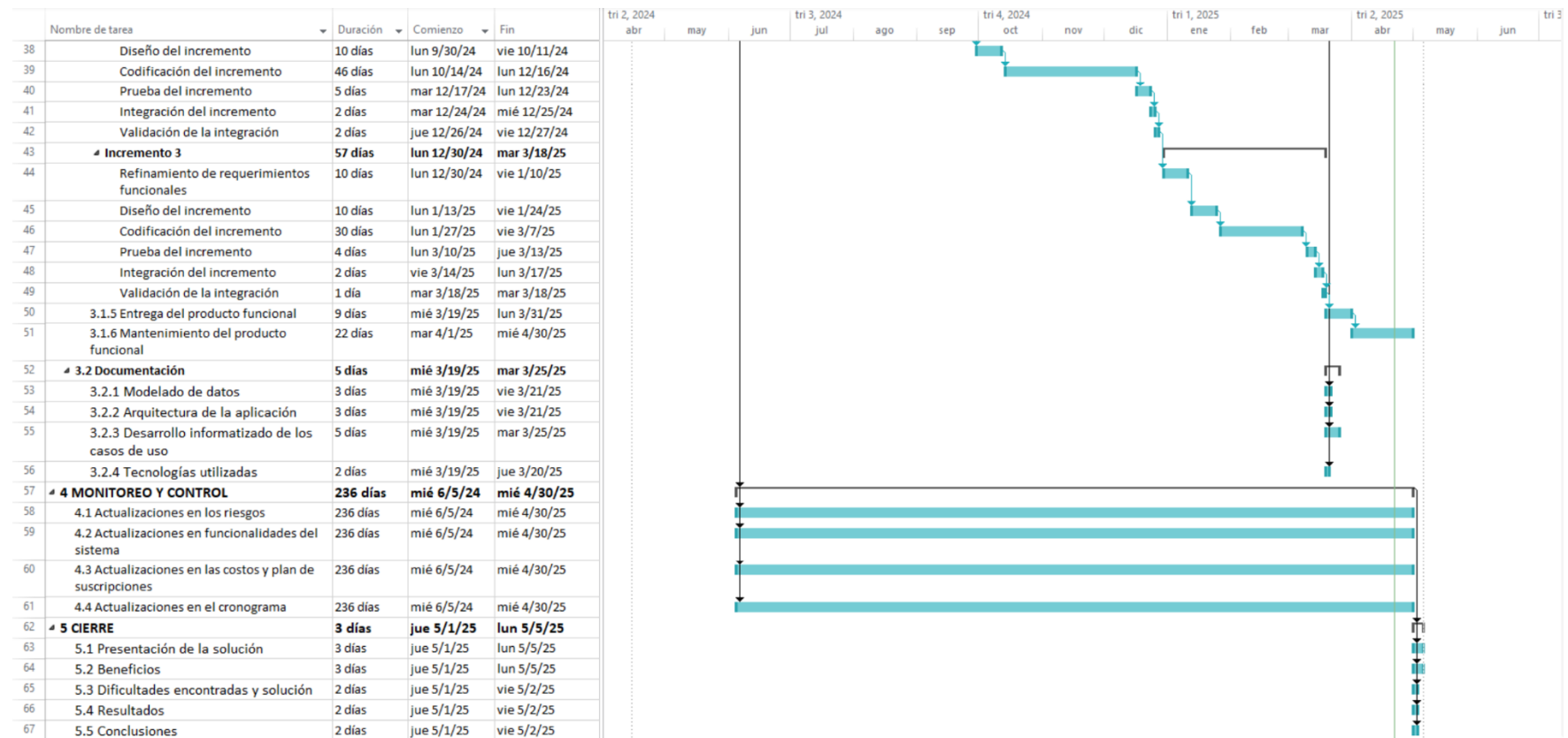


Figura 36. Cronograma del proyecto – Parte 3

4.9 Evaluación económica-financiera

4.9.1 Estimación de costos

Para estimar los costos del talento humano necesario para llevar a cabo el proyecto, se usó como referencia la escala de Honorarios de Informática del COPAIPA (Consejo Profesional de Agrimensores, Ingenieros y Profesionales Afines). Para la facilidad de cálculos posteriores, los valores se convirtieron al dólar oficial estadounidense tomando el valor del dólar oficial del banco nación.

Se considera una carga laboral de 160 horas mensuales con una duración total de desarrollo estimada en 6 meses.

Tabla 18. Costos en dólares de talento humano

Cargo	Cantidad	Horas en el proyecto	Costo por Hora	Subtotales
Analista Funcional	1	480	\$ 9.00	\$ 4,320.00
Analista Programador	2	960	\$ 7.50	\$ 14,400.00
Total				\$ 18,720.00

Los costos fijos de infraestructura y servicio necesarios para poder garantizar un excelente desempeño del sistema en un entorno productivo son los siguientes.

Tabla 19. Costos en dólares de infraestructura y servicio

Servicio	Tipo	Plan	Costo Mensual	Costo anual
Vercel (Frontend)	Infraestructura	Pro	\$ 20.00	\$ 240.00
Render (Backend)	Infraestructura	Profesional	\$ 19.00	\$ 228.00
PostgreSQL (Render DB)	Base de datos	Pro 4GB	\$ 70.00	\$ 840.00
Redis (Render Key Value)	Cache	Standard	\$ 32.00	\$ 384.00
Google Maps Platform	API externo	Varios	\$ 300.00	\$ 3,600.00
OpenAI Developer Platform	API externo	gpt-4o-realtime-preview	\$ 122.50	\$ 1,470.00
Cloudinary	Medios	Free	\$ 0.00	\$ 0.00
Dominio	Infraestructura	Anual	\$ 1.50	\$ 18.00
Certificado SSL	Seguridad	Let's Encrypt	\$ 0.00	\$ 0.00
Total			\$ 565.00	\$ 6,780.00

El detalle de costos por usar los servicios de Google Maps Platform se presenta en la siguiente tabla junto al SKU y la categoría de los productos contratados con los que funciona la aplicación.

Tabla 20. Detalle de costos en dólares de Google Maps Platform

Producto	SKU	Categoría	Peticiones por Mes	Costo por Mes
Places API	Autocomplete Requests	Essentials	20000	\$28.00
Places API	Place Details	Pro	10000	\$85.00
Places API	Text Search	Pro	6000	\$32.00
Places API	Nearby Search	Enterprise	2000	\$35.00
Route API	Route Optimization: Single Vehicle Routing	Pro	7000	\$20.00
Route API	Compute Routes	Enterprise	3000	\$30.00
Maps API	Dynamic	Essentials	20000	\$70.00
Total				\$300.00

4.9.2 Plan de suscripciones

Los ingresos de la aplicación se basan en un modelo de suscripciones ofrecido a personas y empresas que quieran acceder a todas las funcionalidades del sistema.

Para poder proyectar las ganancias obtenidas, se tomó como referencia la cantidad de 300 usuarios tradicionales y de 20 usuarios empresa, activos todos los meses. El valor de la suscripción se determinó teniendo en cuenta los precios standard de los diferentes servicios que ofrecen un modelo similar con el objetivo de maximizar ganancias y atraer a nuevos clientes.

Tabla 21. Planes de suscripción en dólares

Tipo Usuario	Plan Suscripción	Precio por Mes	Nº Aprox. De Suscriptores	Ingreso Mensual	Ingreso Anual
Tradicional	Pro	\$ 5.00	300	\$ 1,500.00	\$ 18,000.00
Empresa	Pyme	\$ 30.00	20	\$ 600.00	\$ 7,200.00
Total				\$ 2,100.00	\$ 25,200.00

4.9.3 Cálculo de indicadores

En el cálculo del ROI¹ se consideró como inversión inicial el total del costo de contratación del talento humano y los gastos previstos de infraestructura y servicios, ambos proyectado anualmente.

Tabla 22. ROI Simple

Ganancia	\$ 25,200.00
Inversión Inicial	\$ 25,500.00
ROI Simple	-1.18%

Como se puede apreciar en la tabla, el ROI simple resulta negativo, esto es entendible dado que el costo de la inversión es mayor que el de las ganancias por un pequeño margen. Con estos resultados inicialmente podemos concluir que el proyecto no es viable si se piensa en una inversión a corto plazo.

¹ ROI: retorno sobre la inversión

Al proyectar las ganancias y costos del proyecto con un horizonte de inversión de 5 años, considerando un crecimiento sostenido e ininterrumpido, podemos usar el VAN, TIR y ROI acumulado para tener una mejor visión del desempeño financiero de cara a los inversores interesados en desembolsar fondos para obtener ganancias con un horizonte de inversión a mediano plazo.

Tabla 23. Tasas por considerar para el cálculo de VAN y TIR

Tasa de descuento	10.00%
Tasa de aumento anual	10.00%
Tasa de gasto anual	6.00%

Tabla 24. Proyección de flujo de caja a 5 años – Parte 1

Año	Ingreso	Proyección Gastos	Flujo Efectivo	Flujo Neto Actualizado
0	\$ 0.00	-\$ 25,500.00	-\$ 25,500.00	-\$ 25,500.00
1	\$ 25,200.00	-\$ 6,780.00	\$ 18,420.00	\$ 16,745.45
2	\$ 27,720.00	-\$ 7,186.80	\$ 20,533.20	\$ 16,969.59
3	\$ 30,492.00	-\$ 7,618.01	\$ 22,873.99	\$ 17,185.57
4	\$ 33,541.20	-\$ 8,075.09	\$ 25,466.11	\$ 17,393.70
5	\$ 36,895.32	-\$ 8,559.59	\$ 28,335.73	\$ 17,594.26

Tabla 25. Proyección de flujo de caja a 5 años - Parte 2

Año	Flujo Neto Actualizado Acumulado	ROI Anual Acumulado	ROI Simple Anual
0	-\$ 25,500.00	0.00%	0.00%
1	-\$ 8,754.55	-134.33%	65.67%
2	\$ 8,215.04	-67.78%	66.55%
3	\$ 25,400.61	-0.39%	67.39%
4	\$ 42,794.31	67.82%	68.21%
5	\$ 60,388.56	136.82%	69.00%

Tabla 26. Cálculo de VAN y TIR

VAN	\$60,388.56
TIR	76%

Como se puede visualizar, el VAN y TIR son muy atractivos y nos indican que el proyecto si es viable financieramente obteniendo un amplio margen de ganancias en una inversión a mediano plazo. A su vez, el ROI acumulado refuerza la idea de que, afrontando las pérdidas de los primeros años, los resultados percibidos al final del ejercicio valen la pena por su alto margen de retorno de inversión.

4.10 Evaluación técnica

A nivel técnico, el proyecto es viable ya que:

1. Las tecnologías que se implementan forman parte del stack tecnológico de varias empresas reconocidas en la industria, se encuentran bien documentadas y son de acceso libre.
2. Todos los servicios de terceros con los que interactúa el sistema son fáciles de integrar gracias a las guías y SDK que facilitan el trabajo.
3. El modelo que utiliza el agente inteligente para asistir en los viajes personales cumple de forma satisfactoria los requisitos del proyecto.
4. El hardware necesario para el despliegue es accesible y se gestiona fácilmente a través de los proveedores de PaaS.
5. La capacidad de escalar la infraestructura en la nube garantiza la gestión eficiente de recursos y un modelo de negocio sostenible en el tiempo.

4.11 Evaluación operativa

El proyecto es operativamente factible, debido a que se dispone del capital humano y de la capacitación correspondiente para así poder llevar adelante el mismo de forma satisfactoria.

4.12 Evaluación legal

Desde el punto de vista legal, el desarrollo y despliegue del sistema es viable, dado que se garantiza el cumplimiento de todas las normativas, tanto en relación con el uso de software como con la integración de servicios de terceros y las regulaciones locales.

1. Uso de herramientas y bibliotecas de código abierto

Todas las herramientas, frameworks y bibliotecas empleadas en el desarrollo del sistema cuentan con licencias de código abierto, tales como MIT, Apache 2.0 o similares. Esto asegura que no existen restricciones legales vinculadas a derechos de propiedad intelectual que limiten su utilización, modificación o distribución en el contexto del presente proyecto.

2. Cumplimiento de los términos de uso de servicios de terceros

Se garantiza el cumplimiento de las condiciones impuestas por los proveedores externos integrados en el sistema:

- a. Google Maps Platform: se incorporan los mecanismos requeridos para la correcta atribución, incluyendo la visualización del logotipo oficial de Google en las interfaces que presentan datos provenientes de sus APIs. Además, se implementa un sistema de almacenamiento en caché basado en Redis, configurado para conservar los datos durante un máximo de 30 días calendario, en conformidad con las políticas de uso de Google.
- b. OpenAI: la integración con los modelos de lenguaje de OpenAI para el asistente de viaje se realiza de acuerdo con sus términos de servicio. No se almacena de forma persistente la información generada, y se asegura la transmisión de datos mediante canales seguros como WebRTC.

3. Regulaciones locales aplicables

El desarrollo del proyecto se enmarca en el territorio de la República Argentina y no contempla la recolección, procesamiento ni almacenamiento de datos sensibles como información financiera o de salud. Esto permite reducir significativamente la complejidad legal asociada a la protección de datos personales conforme a la Ley N.º 25.326 (Ley de Protección de los Datos Personales) y evita la necesidad de adecuación a normativas más restrictivas.

Se concluye que el sistema cumple con todas las condiciones necesarias para ser legalmente factible, tanto a nivel nacional como en lo relativo a los servicios externos integrados, sin incurrir en riesgos regulatorios significativos.

4.13 Gestión de riesgos

Todos los proyectos atraviesan riesgos con diferentes grados de complejidad. Estos riesgos pueden hacer que el proyecto se desvíe del plan y, por lo tanto, no se logre con los objetivos definidos.

Según la Guía PMBOK (6ª edición), el riesgo existe en dos niveles dentro de cada proyecto:

1. El primer nivel, o riesgos individuales del proyecto, está relacionado con eventos o condiciones inciertas que pueden o no pasar, pero si pasan inciden de forma negativa o positiva sobre los objetivos del proyecto.
2. El segundo nivel, o riesgos generales, está influenciado por el efecto de la incertidumbre sobre el proyecto en su conjunto y contempla todos los riesgos individuales más los factores externos que afectan de forma negativa o positiva el desarrollo del proyecto completo [40].

En consecuencia, es de suma importancia estudiar cómo se gestionará el riesgo estableciendo un plan de gestión de riesgos, para posteriormente desarrollar de forma práctica los diferentes procesos involucrados en el mismo, con el propósito de dar trazabilidad al proyecto en cuestión.

4.13.1 Plan gestión de riesgos

En el marco de esta tesis, resulta importante realizar un plan de gestión de riesgos que indique como se identificarán, evaluarán y priorizarán los riesgos del proyecto, así como también la estrategia y acción a tomar en cada uno de ellos. El mismo se estructura según lo propuesto en la Guía PMBOK (6ª edición) abarcando la identificación de riesgos, el análisis cualitativo de riesgos y las estrategias de respuesta a riesgos [40].

Identificación de riesgos

En la identificación de riesgos, se emplea una Estructura de Desglose de los Riesgos (RBS) adaptada al contexto del proyecto. Esta estructura permite identificar y categorizar los riesgos según su origen, facilitando la evaluación y priorización posterior. La Tabla 27 representa el extracto de la Estructura de Desglose de los Riesgos (RBS) que se utilizó en el proceso de identificación de riesgos.

Tabla 27. Extracto de la Estructura de Desglose de los Riesgos (RBS)

Nivel 0 de RBS	Nivel 1 de RBS	Nivel 2 de RBS
0. Todas las fuentes de riesgos del proyecto	1. Riesgos Técnicos	1.1 API de terceros
		1.2 Rendimiento
		1.3 Escalabilidad
		1.4 Herramientas
		1.5 Lógica de negocio
		1.6 Despliegues
		1.7 Dependencias
	2. Riesgo de Calidad	2.1 Experiencia de usuario
		2.2 Adaptación responsive
		2.3 Usabilidad
		2.4 Accesibilidad
		2.5 Contenido o comunicación visual
		2.6 SEO
	3. Riesgos de Gestión	3.1 Dirección del proyecto
		3.2 Recursos
		3.3 Estimaciones
		3.4 Organización
		3.5 Comunicación
	4. Riesgos Operativos	4.1 Errores humanos
		4.2 Capacitación
		4.3 Pruebas y automatizaciones
		4.4 Backups
		4.5 Flujos de trabajo
	5. Riesgos Externos	5.1 Competencia
		5.2 Legislación del país
		5.3 Tasas y precios
		5.4 Servicios de tercero
		5.5 Licencias y políticas de uso

Análisis cualitativo de riesgos

En el análisis cualitativo de riesgos, primero se definen los niveles de probabilidad e impacto, los cuales permiten evaluar la posibilidad de ocurrencia y el efecto potencial de cada riesgo sobre los objetivos del proyecto. Luego, se aplica una matriz de probabilidad e impacto que permite determinar la prioridad relativa de cada riesgo, facilitando así la toma de decisiones sobre su tratamiento. La Tabla 28 representa los niveles de probabilidad e impacto y la Figura 37 muestra la matriz de probabilidad e impacto aplicada para realizar el análisis cualitativo de riesgos.

Tabla 28. Definiciones para probabilidad e impactos

Escala	Probabilidad	+ / - Impacto sobre los objetivos del proyecto		
		Tiempo	Costo	Calidad
Muy alto	> 70%	> 40 días	> USD \$1500	Impacto muy significativo sobre la funcionalidad general
Alto	51 – 70%	21 – 40 días	USD \$700 – \$1500	Impacto significativo sobre la funcionalidad general
Mediano	31 – 50%	7 – 20 días	USD \$250 – \$700	Algún impacto sobre áreas funcionales clave
Bajo	11 – 30%	1 – 6 días	USD \$50 – \$250	Impacto menor sobre la funcionalidad general
Muy bajo	1 – 10 %	< 1 día	< USD \$50	Impacto menor sobre las funciones secundarias
Nulo	< 1%	Sin cambio	Sin cambio	Ningún cambio en la funcionalidad

Esta tabla muestra los niveles de impacto definidos sobre tres objetivos clave del proyecto: tiempo, costo y calidad. La estimación del impacto económico se basa en el presupuesto total del proyecto, compuesto por USD \$18.720 en costos de talento humano y USD \$6.780 en infraestructura y servicios, alcanzando un total de USD \$25.500. A partir de este monto, se establecieron rangos proporcionales que permiten cuantificar con precisión el impacto financiero de cada riesgo. En cuanto al impacto sobre el tiempo, se consideró la duración total del proyecto (390 días) para definir que un retraso superior a 40 días representa un impacto muy alto, ya que podría comprometer entregables críticos o el cierre del proyecto. Los niveles inferiores fueron determinados en función de esa referencia, abarcando desde demoras semanales hasta ajustes mínimos dentro de la

jornada. Finalmente, el impacto en la calidad se evaluó según la afectación sobre la funcionalidad del sistema, diferenciando entre impactos críticos en la funcionalidad general, afectaciones a áreas clave o secundarias, y situaciones sin afectación observable.

		Amenazas					Oportunidades						
Probabilidad	Muy alta 0,90	0,05	0,09	0,18	0,36	0,72	0,72	0,36	0,18	0,09	0,05	Muy alta 0,90	Probabilidad
	Alta 0,70	0,04	0,07	0,14	0,28	0,56	0,56	0,28	0,14	0,07	0,04	Alta 0,70	
	Mediana 0,50	0,03	0,05	0,10	0,20	0,40	0,40	0,20	0,10	0,05	0,03	Mediana 0,50	
	Baja 0,30	0,02	0,03	0,06	0,12	0,24	0,24	0,12	0,06	0,03	0,02	Baja 0,30	
	Muy baja 0,10	0,01	0,01	0,02	0,04	0,08	0,08	0,04	0,02	0,01	0,01	Muy baja 0,10	
		Muy bajo 0,05	Bajo 0,10	Moderado 0,20	Alto 0,40	Muy alto 0,80	Muy alto 0,80	Alto 0,40	Moderado 0,20	Bajo 0,10	Muy bajo 0,05		
Impacto negativo						Impacto positivo							

Figura 37. Matriz de probabilidad e impacto con esquema de puntuación [40]

Estrategias de respuestas a riesgos

En la estrategia de respuesta a los riesgos identificados, se consideran cinco alternativas para hacer frente a amenazas (riesgos negativos) y oportunidades (riesgos positivos), de acuerdo con los lineamientos del PMBOK (6ª Edición). A continuación, se detallan las estrategias posibles para cada caso:

1. Estrategias para amenazas

- Escalar: se traslada la gestión de la amenaza a niveles superiores cuando la amenaza esta fuera del alcance del proyecto o la respuesta necesaria supera la autoridad del director del proyecto.
- Evitar: se actúa para eliminar la amenaza o proteger el proyecto de su impacto. Esta estrategia resulta apropiada para amenazas de alta prioridad con alta probabilidad de ocurrencia y un gran impacto negativo.
- Transferir: se cambia la titularidad de una amenaza a un tercero para que maneje el riesgo y para que soporte el impacto si se produce.
- Mitigar: se toman medidas para reducir la probabilidad de ocurrencia y/o el impacto de una amenaza.
- Aceptar: se reconoce la existencia de la amenaza, pero no se toman medidas proactivas. Esta estrategia es apropiada para amenazas de baja prioridad y cuando no es posible o rentable hacer frente a la amenaza.

2. Estrategias para oportunidades

- a. Escalar: se eleva la oportunidad a niveles superiores cuando la oportunidad esta fuera del alcance del proyecto o la respuesta necesaria supera la autoridad del director del proyecto.
- b. Explotar: se busca capturar el beneficio asociado con una oportunidad especial garantizando de que suceda y aumentando la probabilidad de ocurrencia. Esta estrategia es apropiada para oportunidades con alta prioridad.
- c. Compartir: se transfiere la propiedad de una oportunidad a un tercero para que este comparta algunos de los beneficios si se produce la oportunidad.
- d. Mejorar: se aumenta la probabilidad y/o el impacto de una oportunidad.
- e. Aceptar: se reconoce la existencia de la oportunidad, pero no se toman medidas proactivas. Esta estrategia es apropiada para oportunidades de baja prioridad y cuando no es posible o rentable hacer frente a la oportunidad.

4.13.2 Identificación de riesgos

Según lo establecido en el plan de gestión de riesgos, la identificación de riesgos del proyecto se realizó utilizando como referencia el extracto de la Tabla 27, correspondiente a la estructura de desglose de riesgos (RBS). El proceso se centró exclusivamente en la identificación de riesgos individuales del proyecto, sin considerar riesgos generales. Para su categorización, se utilizó el nivel 1 de la RBS, agrupando los riesgos en categorías principales como técnicos, de calidad, de gestión, operativos y externos. La siguiente tabla represente el registro de riesgos identificados:

Tabla 29. Registro de riesgos – Parte 1

ID	Categoría	Tipo	Descripción	Causa probable	Efecto potencial
R1	Técnico	Amenaza	Error en las consultas realizadas a los servicios de Google Maps Platform.	Exceso de peticiones en un tiempo determinado, límites de uso superados o cambios en políticas.	El sistema no puede operar para el cálculo de rutas o la obtención de direcciones y lugares.
R2	Técnico	Amenaza	Falla en la integración de los servicios de Google Maps Platform.	Falta de autenticación a los servicios o servicios inhabilitados.	El sistema no puede operar en su totalidad.
R3	Técnico	Amenaza	Falla en la integración de los servicios de OpenAI.	Inestabilidad de red en la comunicación en tiempo real o errores de autenticación.	El sistema no permite utilizar el asistente inteligente.
R4	Técnico	Amenaza	Desempeño ineficiente por mala implementación de Redis.	Mala configuración	El sistema pierde rendimiento o deja de responder.
R5	Técnico	Amenaza	Falla en la integración de dependencias.	Implementación inadecuada o flujo mal realizado.	El sistema no permite autenticarse, enviar mails o gestionar imágenes.
R6	Calidad	Amenaza	Falta de diseño adaptable para dispositivos móviles y tablets.	Diseño no apto o falta de pruebas.	Mala experiencia de usuario.
R7	Calidad	Amenaza	Falta de cumplimiento WCAG 2.2 en un nivel AA.	No se consideran criterios de accesibilidad.	Usuarios con discapacidades o en contextos desfavorables no pueden usar el sistema.
R8	Calidad	Amenaza	Carga lenta de la interfaz de usuario.	Mala implementación de SSR o falta de optimización de recursos.	Mal rendimiento y experiencia de usuario.
R9	Gestión	Amenaza	Cambios en el alcance sin replanificación.	Requerimientos mal gestionados.	Sobrecarga de trabajo no prevista.
R10	Gestión	Amenaza	Falta de comunicación en el equipo de desarrollo.	Canales de comunicación no disponibles o perfiles mal definidos.	Retrabajos o sobrecarga no previstas.

Tabla 30. Registros de riesgos - Parte 2

ID	Categoría	Tipo	Descripción	Causa probable	Efecto potencial
R11	Gestión	Amenaza	Estimaciones de costos mal realizadas.	Indicadores mal seleccionados o plan de suscripciones muy ambicioso.	Desfasaje de evaluación económica-financiera.
R12	Operativo	Amenaza	Mal uso de los parámetros en la API de Routes y Routes Optimization.	Falta de comprensión de la documentación.	Optimización de rutas mal realizadas.
R13	Operativo	Amenaza	Mal uso de BullMQ en la programación de tareas.	Desconocimiento del equipo o pocas pruebas realizadas.	Tareas programadas que no se ejecutan correctamente.
R14	Externo	Amenaza	Cambios en la política de precios en los servicios de terceros utilizados.	Decisión unilateral del proveedor.	Incremento en el costo del proyecto.
R15	Externo	Amenaza	Falla en los servidores de Vercel o Render.	Dependencia externa de hosting.	Caída del sistema o inaccesibilidad.
R16	Técnico	Oportunidad	Automatización eficiente con BullMQ.	Correcta configuración y escalado de flujos.	Mayor productividad y procesamiento de eventos.
R17	Calidad	Oportunidad	Interfaz optimizada para accesibilidad.	Aplicación de WCAG 2.2 desde el diseño.	Mayor inclusión y reputación positiva.
R18	Gestión	Oportunidad	Documentación estructurada y compartida.	Uso de herramientas colaborativas efectivas.	Facilita mantenimiento y escalabilidad.
R19	Operativo	Oportunidad	Capacitación temprana del equipo de desarrollo.	Plan de capacitación implementado al inicio.	Evita retrabajos o sobrecarga en el equipo de desarrollo.
R20	Externo	Oportunidad	Acceso anticipado a mejoras en los servicios de tercero.	Registro en versiones beta del proveedor.	Ventaja competitiva e innovación.

4.13.3 Análisis cualitativo de riesgos

Según lo redactado en el plan de gestión de riesgos se presenta la siguiente tabla de análisis cualitativo de riesgos resultante de aplicar las definiciones de los niveles de probabilidad e impacto y la matriz de probabilidad e impacto:

Tabla 31. Análisis cualitativo de los riesgos identificados

ID	Categoría	Tipo	Probabilidad	Impacto	Puntuación
R1	Técnico	Amenaza	Alta	Muy alto	0.56
R2	Técnico	Amenaza	Mediana	Alto	0.2
R3	Técnico	Amenaza	Mediana	Moderado	0.1
R4	Técnico	Amenaza	Alta	Moderado	0.14
R5	Técnico	Amenaza	Baja	Alto	0.12
R6	Calidad	Amenaza	Mediana	Alto	0.2
R7	Calidad	Amenaza	Alta	Alto	0.28
R8	Calidad	Amenaza	Mediana	Moderado	0.1
R9	Gestión	Amenaza	Mediana	Moderado	0.1
R10	Gestión	Amenaza	Alta	Alto	0.28
R11	Gestión	Amenaza	Baja	Moderado	0.06
R12	Operativo	Amenaza	Alta	Moderado	0.14
R13	Operativo	Amenaza	Mediana	Moderado	0.1
R14	Externo	Amenaza	Alta	Alto	0.28
R15	Externo	Amenaza	Mediana	Muy alto	0.4
R16	Técnico	Oportunidad	Mediana	Alto	0.2
R17	Calidad	Oportunidad	Alta	Moderado	0.14
R18	Gestión	Oportunidad	Alta	Alto	0.28
R19	Operativo	Oportunidad	Mediana	Moderado	0.1
R20	Externo	Oportunidad	Mediana	Alto	0.2

4.13.4 Estrategia de respuesta a riesgos

De acuerdo con el plan de gestión de riesgos se presenta la siguiente tabla de estrategias de respuesta a los riesgos identificados con sus respectivas acciones requeridas:

Tabla 32. Estrategia de respuesta a los riesgos identificados – Parte 1

ID	Categoría	Puntuación	Estrategia	Justificación	Acción
R1	Técnico	0.56	Evitar	Riesgo negativo de prioridad alta que se debe eliminar.	Control de solicitud-respuesta en el backend, monitoreo en el uso de servicios y revisión periódica de las políticas.
R2	Técnico	0.2	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Configuración de autenticación en los entornos e inspección de servicios habilitados en Google Cloud.
R3	Técnico	0.1	Aceptar	Riesgo negativo con prioridad baja, no se requiere tomar medidas.	Documentar el riesgo y establecer puntos de monitoreo para reevaluación.
R4	Técnico	0.14	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Aplicar control de errores en las consultas a Redis y monitorear su rendimiento.
R5	Técnico	0.12	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Aplicar monitoreo y control en la implementación de dependencias.
R6	Calidad	0.2	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Mejorar documentación y capacitar al equipo en diseño adaptable.
R7	Calidad	0.28	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Mejorar documentación y capacitar al equipo en accesibilidad web.
R8	Calidad	0.1	Aceptar	Riesgo negativo con prioridad baja, no se requiere tomar medidas.	Documentar el riesgo y establecer puntos de monitoreo para reevaluación.
R9	Gestión	0.1	Aceptar	Riesgo negativo con prioridad baja, no se requiere tomar medidas.	Documentar el riesgo y establecer puntos de monitoreo para reevaluación.
R10	Gestión	0.28	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Facilitar canales de comunicación y gestionar los perfiles de acuerdo con las capacidades del equipo.

Tabla 33. Estrategias de respuesta a los riesgos identificados – Parte 2

ID	Categoría	Puntuación	Estrategia	Justificación	Acción
R11	Gestión	0.06	Aceptar	Riesgo negativo con prioridad baja, no se requiere tomar medidas.	Documentar el riesgo y establecer puntos de monitoreo para reevaluación.
R12	Operativo	0.14	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Capacitar al equipo de desarrollo en el uso de Routes y Routes Optimization.
R13	Operativo	0.1	Aceptar	Riesgo negativo con prioridad baja, no se requiere tomar medidas.	Documentar el riesgo y establecer puntos de monitoreo para reevaluación.
R14	Externo	0.28	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Inspeccionar de forma periódica los precios de los servicios de Google y evaluar alternativas.
R15	Externo	0.4	Mitigar	Riesgo negativo con prioridad media se recomienda aplicar medidas de control para reducir el impacto y/o probabilidad.	Inspeccionar las plataformas de forma periódica y evaluar las inestabilidades.
R16	Técnico	0.2	Mejorar	Riesgo positivo con prioridad media se recomienda aplicar medidas para aumentar la probabilidad y/o el impacto.	Capacitar al equipo de desarrollo y proporcionar documentación actualizada.
R17	Calidad	0.14	Mejorar	Riesgo positivo con prioridad media se recomienda aplicar medidas para aumentar la probabilidad y/o el impacto.	Facilitar documentación de las pautas WCAG 2.2 y capacitar en accesibilidad web.
R18	Gestión	0.28	Mejorar	Riesgo positivo con prioridad media se recomienda aplicar medidas para aumentar la probabilidad y/o el impacto.	Investigar sobre alternativas de herramientas colaborativas efectivas y facilitarlas al equipo.
R19	Operativo	0.1	Aceptar	Riesgo positivo con prioridad baja, no se requiere tomar medidas.	Documentar el riesgo y establecer puntos de monitoreo para reevaluación.
R20	Externo	0.2	Mejorar	Riesgo positivo con prioridad media se recomienda aplicar medidas para aumentar la probabilidad y/o el impacto.	Investigar de forma periódica sobre la existencia de nuevas actualizaciones de los servicios de tercero.

4.14 Dificultades encontradas y soluciones

4.14.1 Configuración para autenticar bibliotecas del cliente

Descripción:

Durante la integración de los servicios de Google Maps Platform se optó por utilizar las bibliotecas cliente oficiales, las cuales simplifican el acceso a las APIs mediante un conjunto de funciones compatibles con distintos lenguajes de programación. No obstante, estas bibliotecas requieren un mecanismo de autenticación adecuado tanto para entornos de desarrollo como de producción.

Google recomienda el uso de las credenciales predeterminadas de la aplicación (ADC, por sus siglas en inglés: Application Default Credentials), una estrategia que permite a las bibliotecas cliente detectar automáticamente las credenciales según el entorno de ejecución. Esta detección puede variar entre desarrollo local, ejecución en contenedores o despliegues en entornos gestionados por Google Cloud.

Establecer correctamente estas credenciales resultó fundamental para garantizar el acceso seguro a los servicios y evitar errores de autenticación durante la ejecución de las pruebas o en el despliegue del sistema.

Solución:

1. Entorno de desarrollo local: se configuró la autenticación siguiendo la guía oficial de Google para entornos locales, que implica la instalación del CLI gcloud y la ejecución del comando `gcloud auth application-default login`. Esta configuración permite que las bibliotecas detecten automáticamente las credenciales almacenadas localmente.
2. Entorno de producción: se generó una cuenta de servicio con los permisos necesarios, cuya clave fue utilizada como archivo de credenciales. Esta clave se proveyó al entorno de ejecución de forma segura, asegurando el acceso a las APIs en tiempo de producción.

4.14.2 Exceso de consumo de los servicios de Google Maps Platform

Descripción:

Durante la integración de funcionalidades que requerían datos geográficos —como direcciones, lugares y rutas— se detectó un uso intensivo de las APIs de Google Maps

Platform. Cada solicitud generaba un consumo adicional facturable, lo que representaba un riesgo de superar los límites gratuitos y elevar los costos del proyecto.

El problema se debía a que cada vez que se necesitaba información, se realizaba una nueva consulta a las APIs, incluso si el dato ya había sido solicitado previamente. Si bien una solución inmediata podría haber sido almacenar permanentemente esa información en la base de datos, esta alternativa fue descartada por no cumplir con las políticas de uso de Google Maps Platform, las cuales prohíben el almacenamiento persistente de ciertos tipos de datos obtenidos a través de sus servicios, como resultados del autocompletado de lugares o detalles de ubicaciones, salvo que se cumplan criterios específicos de permisos o licencias.

Solución:

Para resolver el problema respetando las políticas de uso de Google y optimizar al mismo tiempo el rendimiento y los costos del sistema, se implementó una estrategia de almacenamiento temporal utilizando Redis como caché. Las respuestas de las APIs se almacenan por un período máximo de 30 días, según lo recomendado por Google para mantener actualizada la información sin incurrir en almacenamiento permanente no permitido.

Esta solución permitió:

1. Reducir significativamente el número de llamadas redundantes a los servicios de Google.
2. Mejorar los tiempos de respuesta del sistema.
3. Minimizar el riesgo de incurrir en costos innecesarios.
4. Cumplir con las restricciones establecidas en las políticas de uso y almacenamiento de datos de Google Maps Platform.

4.14.3 Despliegue del sistema en producción

Descripción:

Al realizar el despliegue (deploy) de nuestra API en Render, se generaba un error de falta de memoria y el proceso de implementación (deploying) fallaba. Tras varias pruebas se detectó que el problema no era por falta de recursos de memoria, sino, por la forma en la que se estaban generando los archivos de distribución al compilar el proyecto lo que causaba que el script fallara desencadenando el error de falta de memoria.

Solución: Se verifico la ruta donde se encontraba el archivo “main.js” al compilar el proyecto y se corrigió el script con el que ejecuta los archivos compilados en producción.

4.14.4 Respuesta de error de las transacciones debido a la falta de recursos asociados al sistema y/o la configuración de estas en el ORM

Descripción:

Las solicitudes a la API que involucraban transacciones funcionaban correctamente en el entorno local, pero al desplegarse en producción comenzaban a fallar. Este comportamiento anómalo se debía a una combinación entre la limitación de recursos del servidor y una configuración inadecuada del ORM, lo que provocaba tiempos de espera insuficientes y cierres prematuros de conexión.

Solución:

Se incrementaron los recursos asignados al servidor en la plataforma Render y se ajustó la configuración del ORM, duplicando los valores de timeout y tiempos de cierre de conexión para mejorar la estabilidad de las transacciones en producción.

4.14.5 Organización de registros en cache

Descripción:

Era necesario utilizar una única instancia de Redis tanto para el almacenamiento en caché de los datos obtenidos desde los servicios de Google, como para la gestión de tareas programadas mediante la librería BullMQ. Esta decisión buscó optimizar el uso de recursos y reducir los costos asociados al mantenimiento de múltiples instancias de Redis.

Solución:

Se implementó una estrategia de organización de claves en Redis mediante sufijos específicos que permiten distinguir fácilmente el propósito de cada valor almacenado. Esto facilitó una gestión eficiente y escalable dentro de una única instancia compartida.

5. RESULTADOS

5.1 Resultados obtenidos

Como resultado, se presenta “*Journeys*”, un sistema informático desarrollado bajo un enfoque integral y orientado a la optimización de rutas en el proceso de transporte. El sistema entregado corresponde a una versión funcional construida a través de tres incrementos, que conforma un producto mínimo viable (MVP) validable, cumpliendo con los requerimientos especificados en la sección 4.4.4 del capítulo anterior y fundamentado en el marco teórico y metodológico desarrollado a lo largo de la presente tesis.



Figura 38. Logo de Journeys

El sistema se encuentra alojado en la nube bajo los servicios de Vercel y Render:

<https://journeys-web.vercel.app/>

<https://journeys-api.onrender.com/docs>

La disponibilidad de los enlaces depende de los servicios de despliegue utilizados al momento del desarrollo. Para mayor referencia, se presentan capturas de pantalla de las partes más importantes del sistema.

5.1.1 Generales

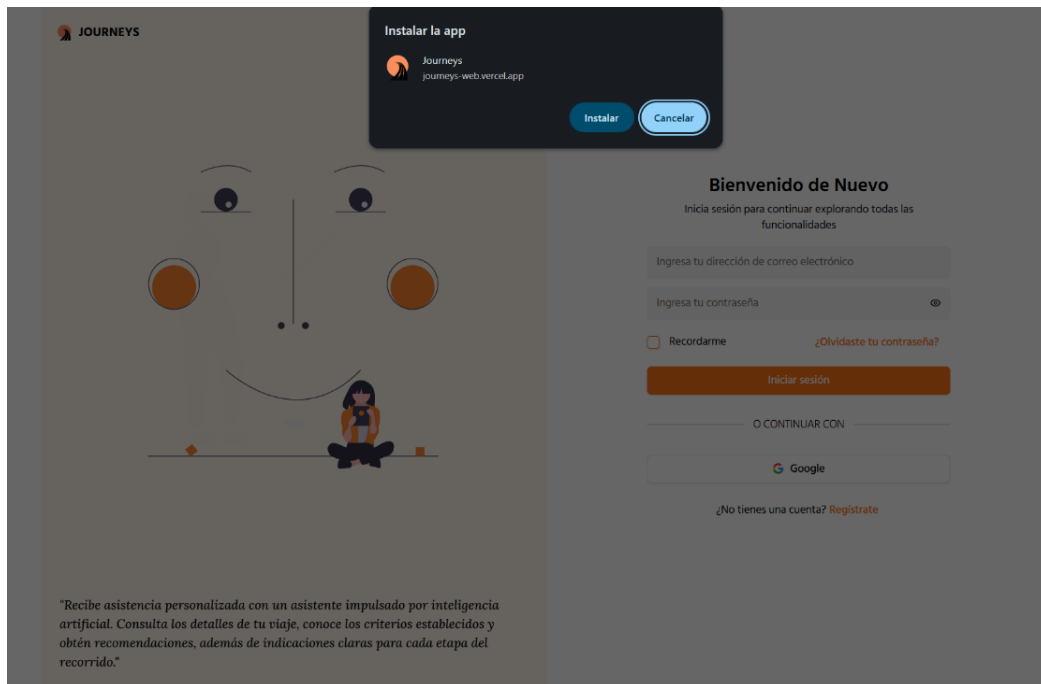


Figura 39. Pantalla principal del sistema

La Figura 39 muestra una captura de la pantalla principal del sistema con la opción de instalarla, característica particular de una PWA, donde el usuario puede iniciar sesión con sus credenciales o a través mediante los servicios de Google.

5.1.2 Planificación de viajes asistidos

A continuación, se presentan una serie de pantallas que corresponden al caso de uso planificación de viajes asistidos. A modo de referencia, se tuvieron en cuenta aquellas asociadas a las funcionalidades de optimización, estadísticas y viajes, por ser representativas del flujo principal.

Inicio

Viajes

Optimización

Lugares Favoritos

Direcciones Favoritas

Plantillas de Actividades

Optimización

Criterios Básicos
 En este paso, podrás definir los aspectos esenciales para planificar tu viaje de manera eficiente y personalizada. Aquí establecerás el punto de partida y el destino final, considerando factores clave que impactan tu experiencia en el camino.
[Ver menos](#)

Criterios Avanzados
 En este paso podrás personalizar aún más tu experiencia de viaje al incorporar detalles específicos y ajustes estratégicos
[Ver menos](#)

Resultados
 En este paso, podrás visualizar los resultados de tu planificación y ajustar los detalles necesarios para optimizar tu experiencia de viaje.
[Ver menos](#)

Criterios Básicos

Origen

Configurar

Casa - Radio San Salvador de Jujuy 3169, A4400 Salta

Destino

Configurar

Hermanucha - Av. Vélez Sarsfield 661, X5000 Córdoba

Fecha de salida

21 de abril de 2025

Hora de salida

08:00

Puntos de interés

Configurar

3 puntos de interés

Tipo de vehículo

☒ Automóvil
☐ Motocicleta

Consideraciones de tráfico

☐ Considerar el tráfico
☒ Considerar el tráfico y optimizar
☐ No considerar el tráfico

Otras consideraciones

☐ Evitar peajes
☐ Evitar autopistas
☒ Evitar transbordadores

Siguiente

Figura 40. Planificación de viajes asistidos – Pantalla de optimización – Criterios básicos

La Figura 40 muestra la pantalla correspondiente al proceso de optimización dentro de la planificación de viajes asistidos. En ella se visualizan los criterios básicos configurables que permiten personalizar el viaje y que son considerados al momento de generar la optimización.

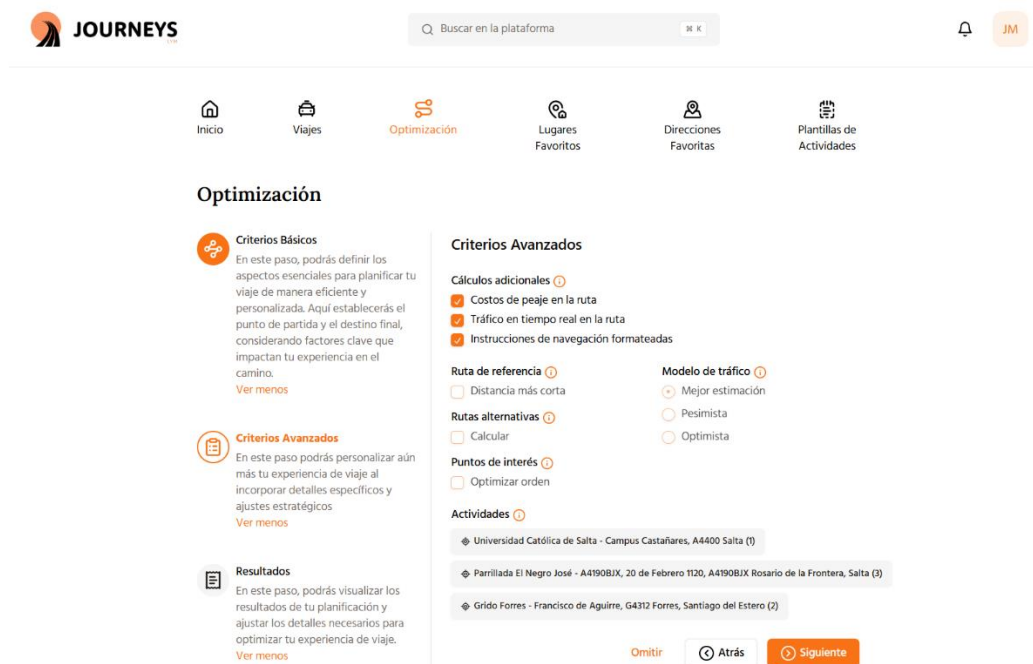


Figura 41. Planificación de viajes asistidos – Pantalla de optimización – Criterios avanzados

La Figura 41 muestra la pantalla correspondiente al proceso de optimización dentro de la planificación de viajes asistidos. En ella se visualizan los criterios avanzados configurables que permiten personalizar el viaje y que son considerados al momento de generar la optimización.

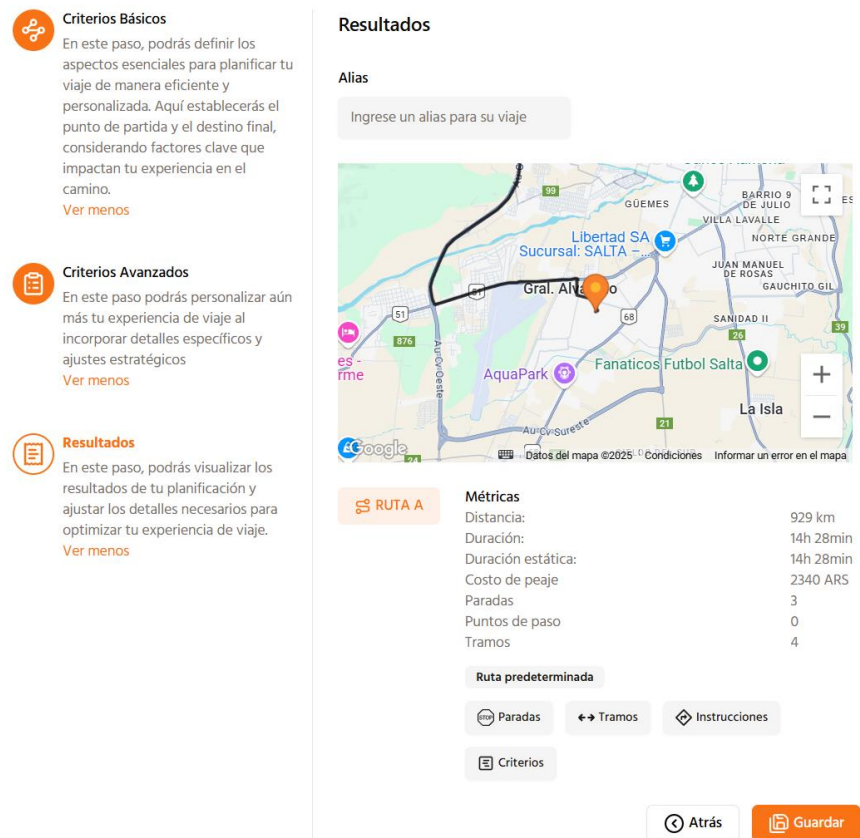


Figura 42. Planificación de viajes asistidos – Pantalla de optimización – Resultados

La Figura 42 muestra la pantalla correspondiente al proceso de optimización dentro de la planificación de viajes asistidos. En ella se visualizan los resultados de la optimización.

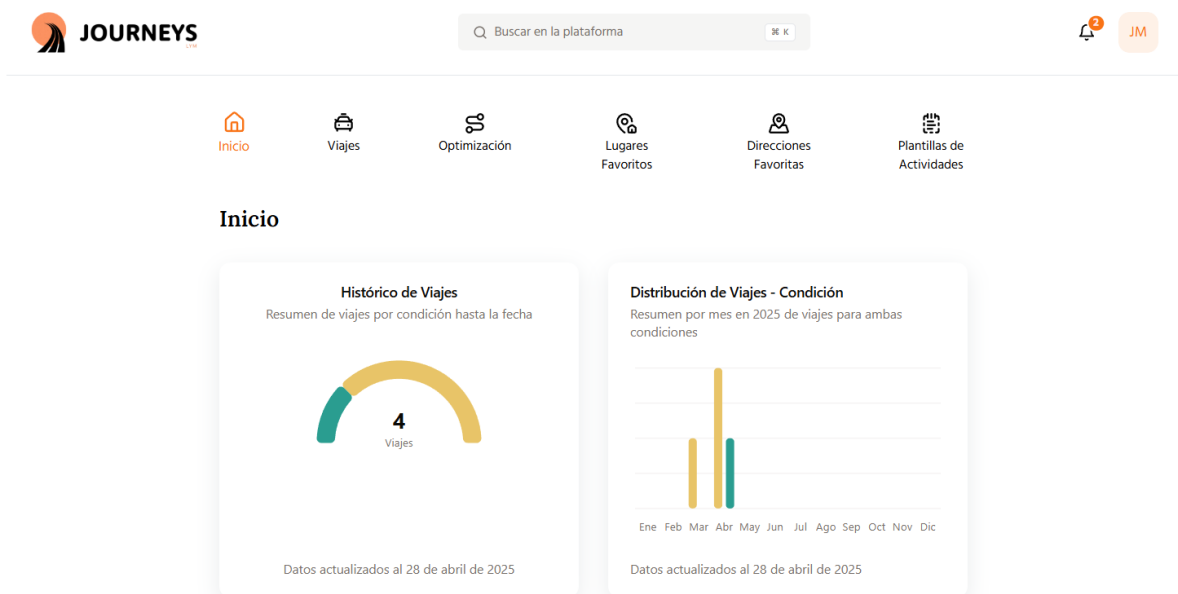


Figura 43. Planificación de viajes asistidos – Pantalla de inicio

La Figura 43 muestra la pantalla correspondiente a inicio dentro de la planificación de viajes asistidos. En ella se muestran las estadísticas básicas relacionadas al histórico de viajes y a la distribución de los viajes mensuales en el último año de acuerdo con su condición.

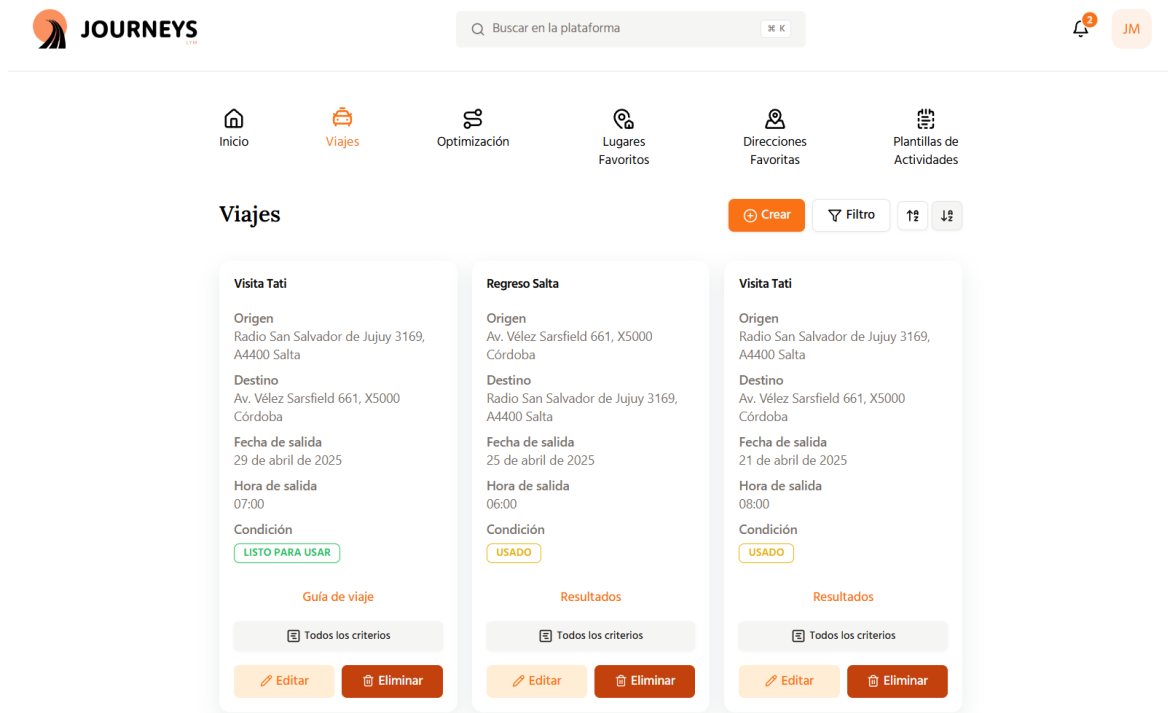


Figura 44. Planificación de viajes asistidos – Pantalla de viajes

La Figura 44 muestra la pantalla correspondiente a viajes dentro de la planificación de viajes asistidos. En ella se visualizan todos los viajes con sus determinadas acciones, resaltando el estado en el que se encuentran.

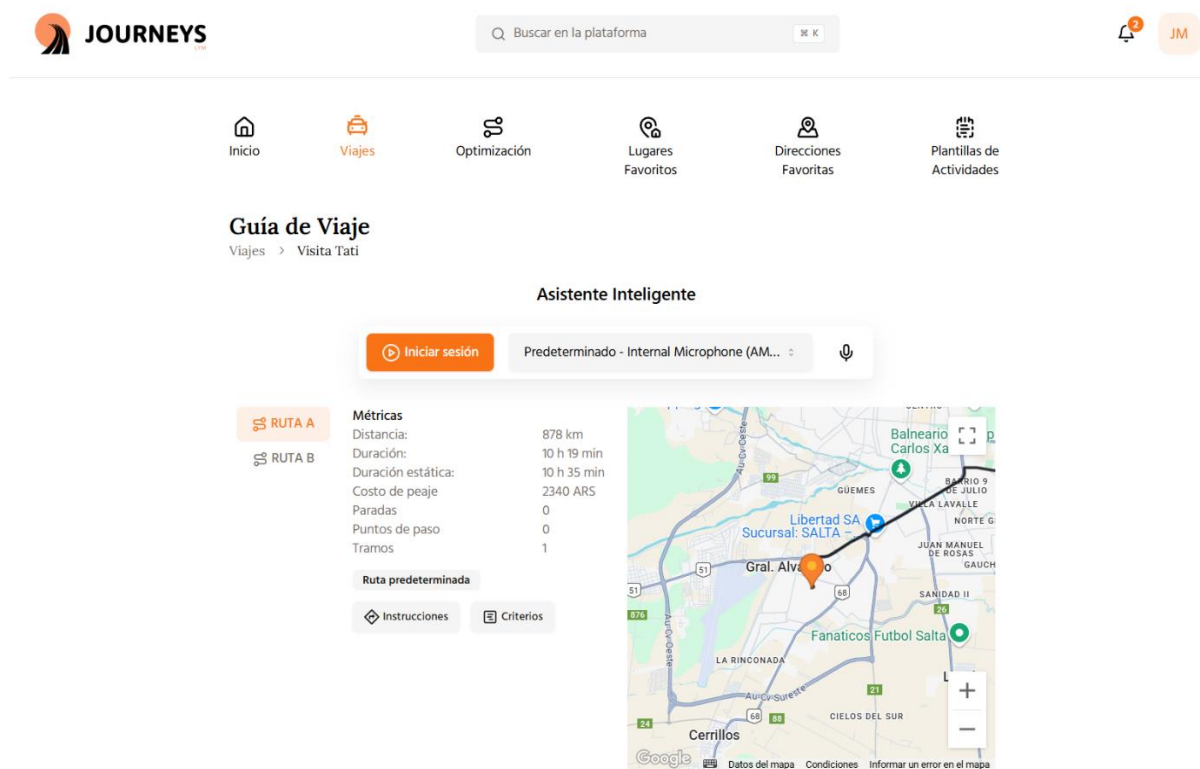


Figura 45. Planificación de viajes asistidos – Pantalla de guía de viaje

La Figura 45 muestra la pantalla correspondiente a la guía de viaje dentro de la planificación de viajes asistidos. En ella se destaca el asistente inteligente, las métricas del viaje, un mapa con información relevante, un selector de las diferentes rutas posibles, las instrucciones y los criterios. También cuando se dispone de puntos de interés se incluyen las paradas y los tramos.

5.1.3 Gestión de flotas en servicios a domicilio

A continuación, se presentan una serie de pantallas que corresponden al caso de uso gestión de flotas en servicios a domicilio. A modo de referencia, se tuvieron en cuenta aquellas asociadas a las funcionalidades de optimización, estadísticas y hojas de ruta, por ser representativas del flujo principal.

Q Buscar en la plataforma

Panel de Control

Hojas de Ruta

Optimización

Flotas

Conductores

Vehículos

Plantillas de Actividades

Optimización

1 **Primera Etapa**
Proporcione la información inicial para el cálculo de rutas, incluyendo ubicaciones, horarios y...
[Ver más](#)

2 **Segunda Etapa**
Configura los servicios a domicilio seleccionando actividades, estableciendo ubicaciones y...
[Ver más](#)

3 **Tercera Etapa**
Selecciona un perfil de costos para optimizar según tus necesidades. Si lo deseas, también puedes...
[Ver más](#)

4 **Resultados**
Visualiza los resultados de la optimización de acuerdo a las variables configuradas, incluyendo L...
[Ver más](#)

Primera Etapa

Ubicación inicial

Ubicación final

Los Incienso 149, A4408 Salt...

Los Incienso 149, A4408 Salt...

Fecha

22 de abril de 2025

Flota

Instalaciones

Conductor

Lionel Andrés Messi

Hora de inicio

Hora de fin

09:00

18:00

Vehículo

AA 002 AA

Otras consideraciones

☐ Evitar peajes

☐ Evitar autopistas

☒ Evitar transbordadores

☒ Considerar tráfico

Siguiente

Figura 46. Gestión de flotas en servicios a domicilio – Pantalla de optimización – Primera Etapa

La Figura 46 muestra la pantalla correspondiente al proceso de optimización dentro de la gestión de flotas en servicios a domicilio. En ella se visualizan los campos de configuración correspondientes a la primera etapa y que son considerados al momento de generar la optimización.

Optimización

- Primera Etapa**
Proporcione la información inicial para el cálculo de rutas, incluyendo ubicaciones, horarios y...
[Ver más](#)
- Segunda Etapa**
Configura los servicios a domicilio seleccionando actividades, estableciendo ubicaciones y...
[Ver más](#)
- Tercera Etapa**
Selecciona un perfil de costos para optimizar según tus necesidades. Si lo deseas, también puedes...
[Ver más](#)
- Resultados**
Visualiza los resultados de la optimización de acuerdo a las variables configuradas, incluyendo L...
[Ver más](#)

Segunda Etapa

Servicios a domicilio

[Agregar](#)

Servicio #1

Actividad: Alta Nuevo Cliente

Descripción: Se viaja al domicilio del cliente, se realiza el tendido de fibra necesario y se entregan los equipos para dar de alta un nuevo servicio de fibra y/o tv

Dirección

Radio San Salvador de Jujuy 3169, A4400 Salta, Arge...

Duración

02:00

Servicio #2

Actividad: Asistencia Técnica

Descripción: Se viaja al domicilio del cliente con problemas en su servicio para normalizarlo

Dirección

Cnel. Moldes 950, A4400 Salta, Argentina

Duración

01:20

Servicio #3

Actividad: Asistencia Técnica

Descripción: Se viaja al domicilio del cliente con problemas en su servicio para normalizarlo

Dirección

Los Cebiles 274, A4400 Salta, Argentina

Duración

01:20

Figura 47. Gestión de flotas en servicios a domicilio – Pantalla de optimización – Segunda Etapa

La Figura 47 muestra la pantalla correspondiente al proceso de optimización dentro de la gestión de flotas en servicios a domicilio. En ella se visualizan los campos de configuración correspondientes a la segunda etapa y que son considerados al momento de generar la optimización.

Optimización

- Primera Etapa**
Proporcione la información inicial para el cálculo de rutas, incluyendo ubicaciones, horarios y...
[Ver más](#)
- Segunda Etapa**
Configura los servicios a domicilio seleccionando actividades, estableciendo ubicaciones y...
[Ver más](#)
- Tercera Etapa**
Selecciona un perfil de costos para optimizar según tus necesidades. Si lo deseas, también puedes...
[Ver más](#)
- Resultados**
Visualiza los resultados de la optimización de acuerdo a las variables configuradas, incluyendo L...
[Ver más](#)

Tercera Etapa

Perfil de costos

Optimización equilibrada

Los valores de los costos representan unidades relativas.

Nombre: Optimización equilibrada
Descripción: Balancea la distancia y el tiempo total de cada servicio, garantizando un uso eficiente del tiempo sin afectar la planificación de visitas.

Costo fijo (CF): 20
Costo por kilómetro (CPK): 100
Costo por hora (CPH): 100
Costo por hora recorrida (CPHR): 50
Multiplicador de tiempo de tránsito: 1

Límites

Duración total [?](#) **Duración en tránsito** [?](#) **Distancia total** [?](#)

HH:MM HH:MM 0.00

[Omitir](#) [Atrás](#) [Siguiente](#)

Figura 48. Gestión de flotas en servicios a domicilio – Pantalla de optimización

La Figura 48 muestra la pantalla correspondiente al proceso de optimización dentro de la gestión de flotas en servicios a domicilio. En ella se visualizan los campos de configuración correspondientes a la tercera etapa y que son considerados al momento de generar la optimización.

1Primera Etapa

Proporcione la información inicial para el cálculo de rutas, incluyendo ubicaciones, horarios y...

Ver más

2Segunda Etapa

Configura los servicios a domicilio seleccionando actividades, estableciendo ubicaciones y...

Ver más

3Tercera Etapa

Selecciona un perfil de costos para optimizar según tus necesidades. Si lo deseas, también puedes...

Ver más

4Resultados

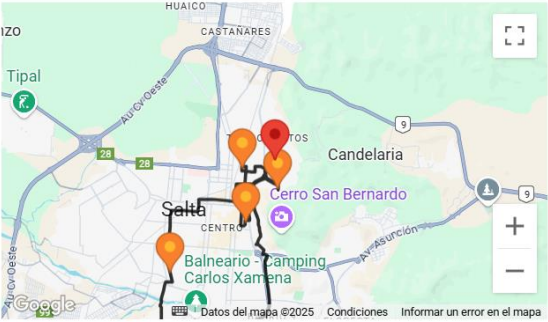
Visualiza los resultados de la optimización de acuerdo a las variables configuradas, incluyendo l...

Ver más

Resultados

Alias

Ingrese un alias para la hoja de ruta



Métricas

Distancia total: 33.20 km

Duración en visitas: 6h 30min

Duración en tránsito: 1h 15min

Servicios incluidos: 5

Servicios desestimados: 0

Costo fijo: -

Costo por hora: -

Costo por kilómetro: -

Costo por hora de recorrida: -

Costo total: 0.00

Operación

Fecha: 22 de abril de 2025

Hora de inicio: 09:00

Hora de fin: 16:45

Recursos

Flota: Instalaciones

Vehículo: AA 002 AA

Conductor: Lionel Andrés Messi

Visitas

Transiciones

Configuración

Atrás

Finalizar

Figura 49. Gestión de flotas en servicios a domicilio – Pantalla de optimización

La Figura 49 muestra la pantalla correspondiente al proceso de optimización dentro de la gestión de flotas en servicios a domicilio. En ella se visualizan los resultados de la optimización.

144

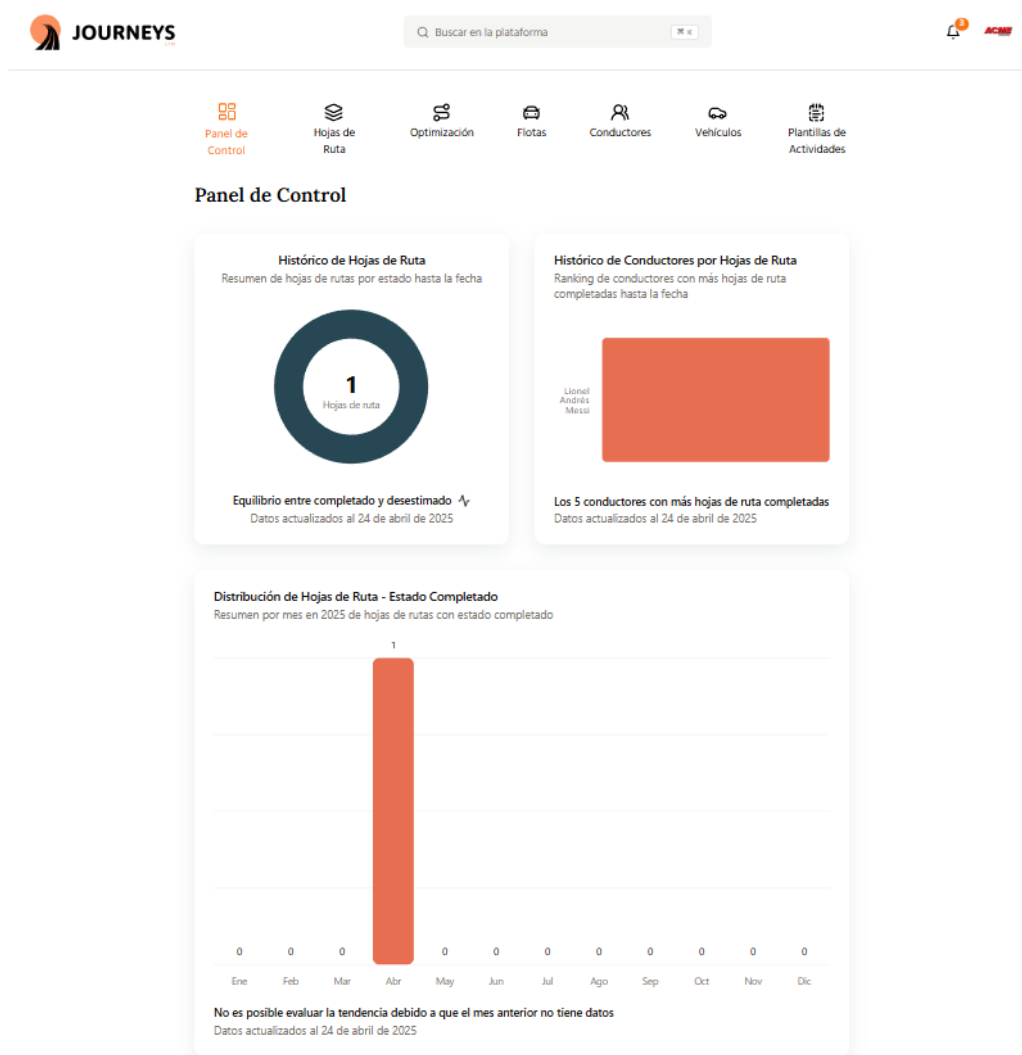


Figura 50. Gestión de flotas en servicios a domicilio – Pantalla de panel de control

La Figura 50 muestra la pantalla correspondiente al panel de control dentro de la gestión de flotas en servicios a domicilio. En ella se muestran las estadísticas básicas relacionadas al histórico de hojas de ruta por estado, el ranking de los 5 conductores con más hojas de ruta completadas y la distribución de hojas de ruta completadas por mes del año en curso.

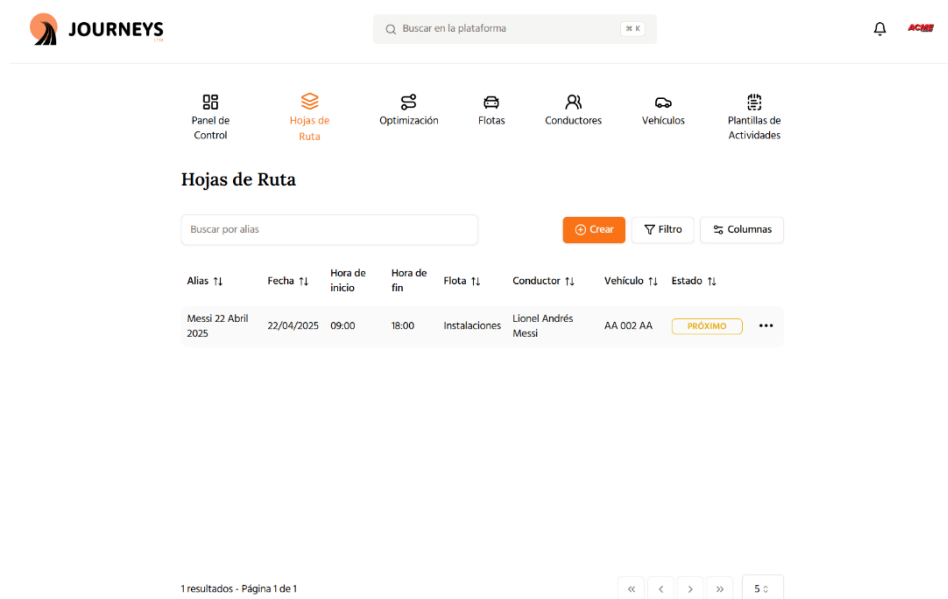


Figura 51. Gestión de flotas en servicios a domicilio – Pantalla de hojas de ruta

La Figura 51 muestra la pantalla correspondiente a hojas de ruta dentro de la gestión de flotas en servicios a domicilio. En ella se listan todas las hojas de rutas cargadas con la información más relevante para su gestión.

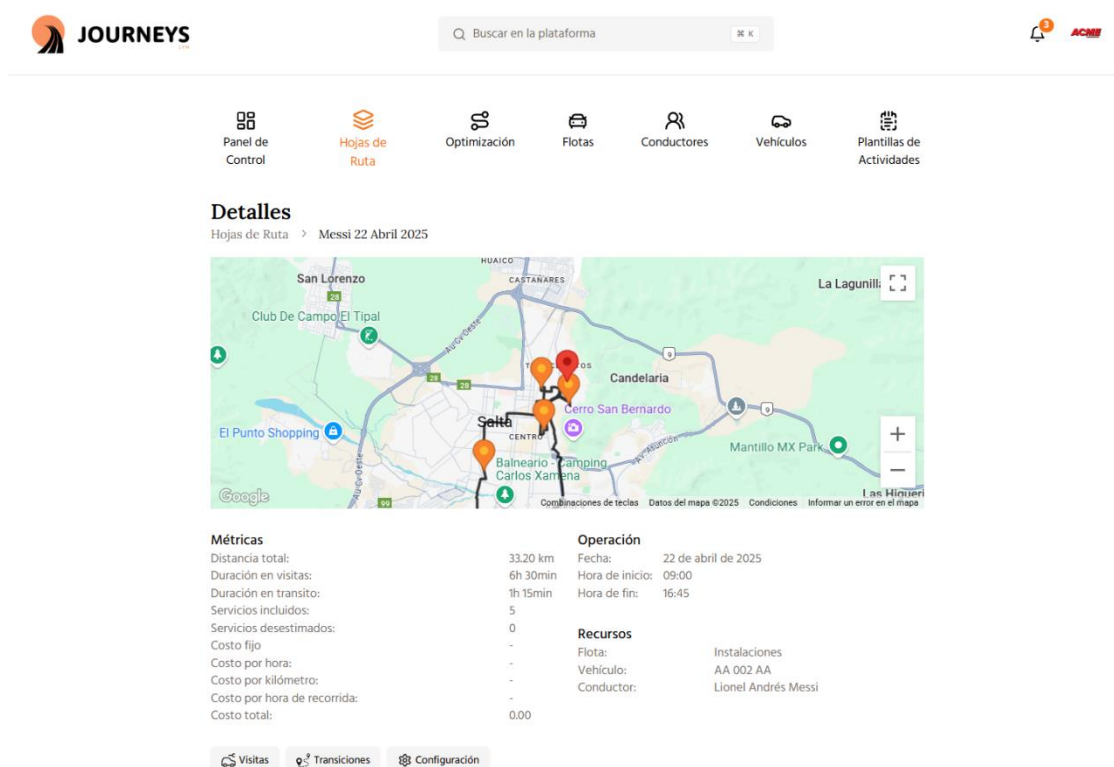


Figura 52. Gestión de flotas en servicios a domicilio – Pantalla detalles de hoja de ruta

La Figura 52 muestra la pantalla correspondiente a los detalles de la hoja de ruta dentro de la gestión de flotas en servicios a domicilio. En ella se visualizan un mapa con información relevante, los datos de operación, las métricas de la hoja de ruta y los recursos asignados, también desde esta pantalla se pudo acceder a las visitas, las transiciones en el recorrido y las configuraciones que se tuvieron en cuenta para realizar la optimización del recorrido.

5.2 Calidad de los resultados

Con el fin de evaluar la calidad del sistema informático desarrollado, se realizó una auditoría utilizando la herramienta automatizada Google Lighthouse, reconocida por su capacidad para medir diversos aspectos técnicos de aplicaciones web. Esta evaluación permitió obtener métricas objetivas en relación con el rendimiento, la accesibilidad, las buenas prácticas de desarrollo y el cumplimiento de criterios SEO.

La auditoría se enfocó en las páginas más relevantes del flujo principal del sistema estudiando los siguientes ejes:

1. Rendimiento (Performance)
2. Accesibilidad (Accessibility)
3. Buenas prácticas (Best Practices)
4. Optimización para motores de búsqueda (SEO)

La calidad de los resultados está determinada por las puntuaciones obtenidas por la herramienta y el puntaje mínimo esperado redactado en los criterios de aceptación en la sección 3.3.3 del capítulo 3.

A continuación, se presenta un informe completo de la página de optimización de la planificación de viajes asistidos y la puntuación del resto de páginas relevantes en el flujo principal.

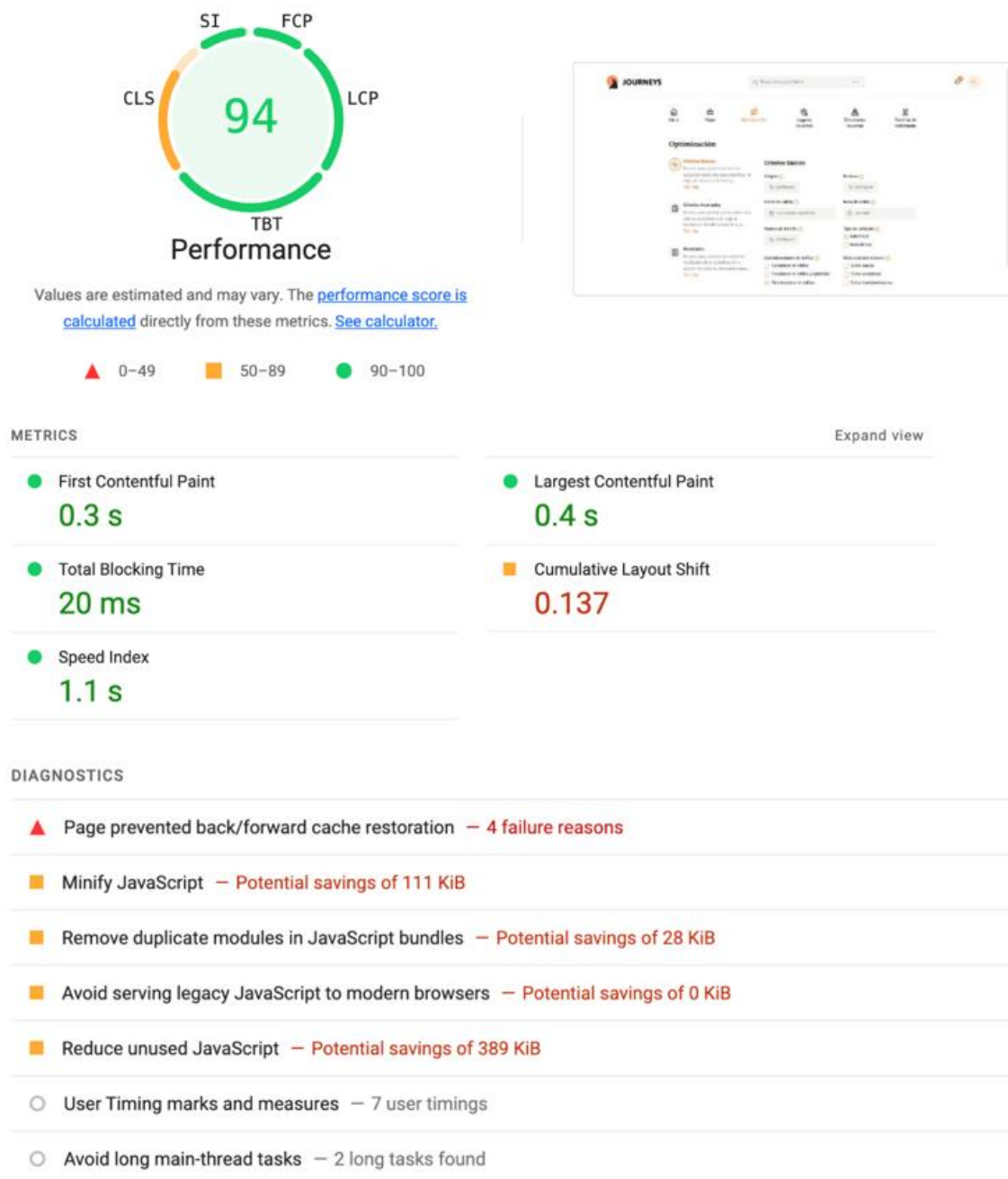


Figura 53. Informe de rendimiento de la página de optimización en la planificación de viajes asistidos – Parte 1

PASSED AUDITS (24)		Hide
☒	Layout shift culprits	▼
☒	Document request latency	▼
☒	Optimize DOM size	▼
☐	Font display	▼
☐	Forced reflow	▼
☐	INP by phase	▼
☐	LCP request discovery	▼
☒	LCP by phase	▼
☐	Long critical network tree	▼
☐	Render blocking requests	▼
☐	CSS Selector costs	▼
☒	Optimize viewport for mobile	▼
☒	Defer offscreen images	▼
☒	Minify CSS	▼
☒	Reduce unused CSS	▼
☒	Preconnect to required origins	▼
☒	Use HTTP/2	▼
☒	Avoids enormous network payloads — Total size was 1,096 KiB	▼
☒	Uses efficient cache policy on static assets — 0 resources found	▼
☒	JavaScript execution time — 0.2 s	▼
☒	Minimizes main-thread work — 0.9 s	▼
☐	Lazy load third-party resources with facades	▼
☒	Uses passive listeners to improve scrolling performance	▼
☒	Avoids <code>document.write()</code>	▼

Figura 54. Informe de rendimiento de la página de optimización en la planificación de viajes asistidos – Parte 2



Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

CONTRAST

- | | |
|---|---|
| ▲ Background and foreground colors do not have a sufficient contrast ratio. | ▼ |
|---|---|

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Hide

- | | |
|---|---|
| ☐ Interactive controls are keyboard focusable | ▼ |
| ☐ Interactive elements indicate their purpose and state | ▼ |
| ☐ The page has a logical tab order | ▼ |
| ☐ Visual order on the page follows DOM order | ▼ |
| ☐ User focus is not accidentally trapped in a region | ▼ |
| ☐ The user's focus is directed to new content added to the page | ▼ |
| ☐ HTML5 landmark elements are used to improve navigation | ▼ |
| ☐ Offscreen content is hidden from assistive technology | ▼ |
| ☐ Custom controls have associated labels | ▼ |
| ☐ Custom controls have ARIA roles | ▼ |

Figura 55. Informe de accesibilidad de la página de optimización en la planificación de viajes asistidos – Parte 1

PASSED AUDITS (25)	Hide
[aria-*) attributes match their roles	▼
[aria-hidden="true"] is not present on the document <body>	▼
[role]s have all required [aria-*) attributes	▼
[aria-*) attributes have valid values	▼
[aria-*) attributes are valid and not misspelled	▼
Buttons have an accessible name	▼
Image elements have [alt] attributes	▼
[user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.	▼
ARIA attributes are used as specified for the element's role	▼
[aria-hidden="true"] elements do not contain focusable descendents	▼
Elements use only permitted ARIA attributes	▼
[role] values are valid	▼
Document has a <title> element	▼
<html> element has a [lang] attribute	▼
<html> element has a valid value for its [lang] attribute	▼
Form elements have associated labels	▼
Links have a discernible name	▼
Lists contain only elements and script supporting elements (<script> and <template>).	▼
List items () are contained within , or <menu> parent elements	▼
No element has a [tabindex] value greater than 0	▼
Touch targets have sufficient size and spacing.	▼
Heading elements appear in a sequentially-descending order	▼
Uses ARIA roles only on compatible elements	▼
Deprecated ARIA roles were not used	▼
Image elements do not have [alt] attributes that are redundant text.	▼

Figura 56. Informe de accesibilidad de la página de optimización en la planificación de viajes asistidos – Parte 2



Best Practices

TRUST AND SAFETY

- | | |
|---|---|
| ☐ Ensure CSP is effective against XSS attacks | ▼ |
| ☐ Ensure proper origin isolation with COOP | ▼ |
| ☐ Mitigate clickjacking with XFO or CSP | ▼ |

PASSED AUDITS (14)

Hide

- | | |
|--|---|
| ☒ Uses HTTPS | ▼ |
| ☒ Avoids deprecated APIs | ▼ |
| ☒ Avoids third-party cookies | ▼ |
| ☒ Allows users to paste into input fields | ▼ |
| ☒ Avoids requesting the geolocation permission on page load | ▼ |
| ☒ Avoids requesting the notification permission on page load | ▼ |
| ☒ Displays images with correct aspect ratio | ▼ |
| ☒ Serves images with appropriate resolution | ▼ |
| ☒ Has a <code><meta name="viewport"></code> tag with <code>width</code> or <code>initial-scale</code> | ▼ |
| ☒ Page has the HTML doctype | ▼ |
| ☒ Properly defines charset | ▼ |
| ☒ No browser errors logged to the console | ▼ |
| ☒ No issues in the Issues panel in Chrome Devtools | ▼ |
| ☒ Page has valid source maps | ▼ |

Figura 57. Informe de buenas prácticas de la página de optimización en la planificación de viajes asistidos

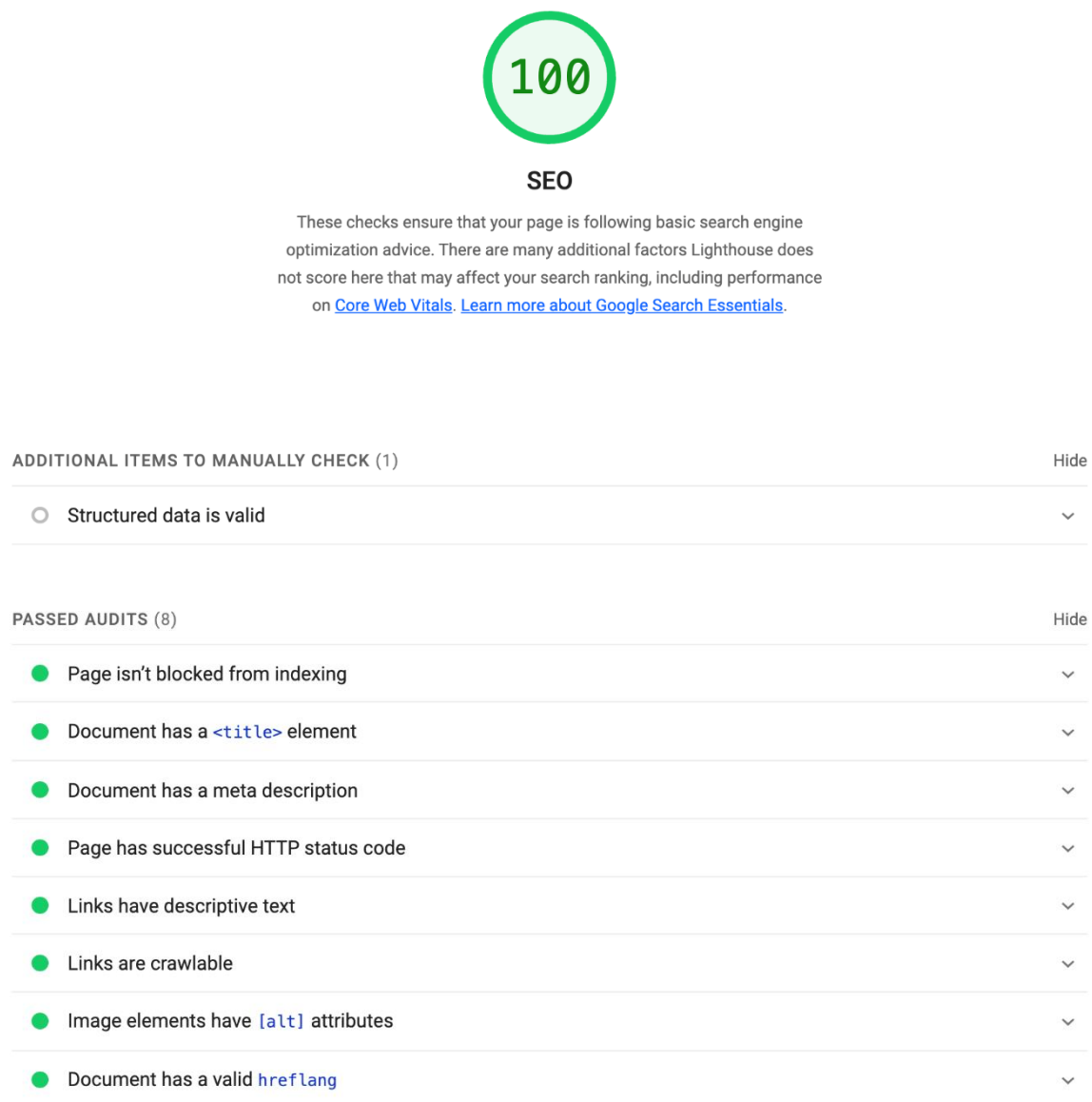


Figura 58. Informe de optimización para motores de búsqueda de la página de optimización en la planificación de viajes asistidos

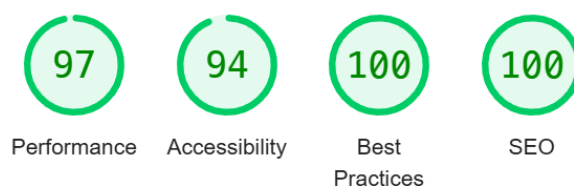


Figura 59. Puntuación obtenida en la página de inicio de planificación de viajes asistidos

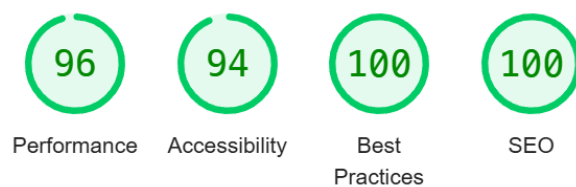


Figura 60. Puntuación obtenida en la página de viajes de planificación de viajes asistidos

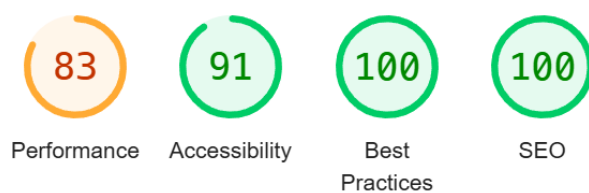


Figura 61. Puntuación obtenida en la página de optimización de gestión de flotas en servicios a domicilio

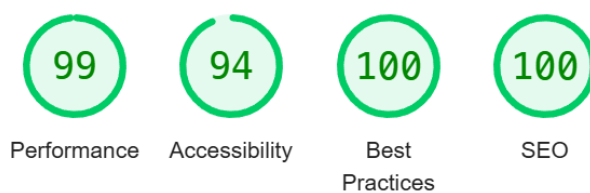


Figura 62. Puntuación obtenida en la página de panel de control de gestión de flotas en servicios a domicilio

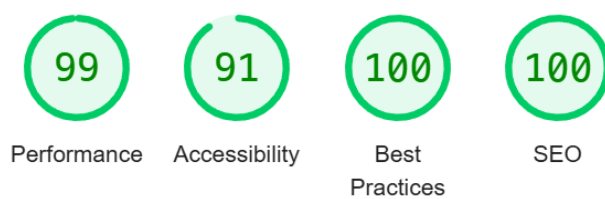


Figura 63. Puntuación obtenida en la página de hojas de ruta de gestión de flotas en servicios a domicilio

6. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

6.1 Conclusiones

El presente trabajo logró desarrollar exitosamente un sistema web moderno orientado a la optimización de rutas en el proceso de transporte, adaptable a distintos contextos y casos de uso. El resultado obtenido constituye un producto mínimo viable (PMV) funcional, validable y alineado con los requerimientos definidos, que permite demostrar su aplicabilidad en situaciones reales.

Para alcanzar los objetivos planteados, se abordaron dos contextos representativos: la planificación de viajes personales y la gestión de flotas en servicios a domicilio. En cada uno, se implementó un caso de uso específico con optimizaciones programadas, lo cual permitió validar el funcionamiento del sistema desde una perspectiva tanto técnica como operativa.

Los resultados obtenidos demostraron que es posible integrar múltiples contextos y necesidades en una única solución informática, proporcionando al usuario una herramienta flexible que se adapta a distintos escenarios. Asimismo, se evidenció que el uso de tecnologías modernas y buenas prácticas de desarrollo permitió construir una solución robusta, modular, escalable y mantenible, desplegable en la nube e integrada con servicios externos.

La experiencia de desarrollo fue altamente positiva, ya que permitió aplicar conocimientos adquiridos a lo largo de la carrera junto con aprendizajes provenientes de experiencias profesionales. Además, se incorporaron tecnologías emergentes que hoy lideran la industria del software, consolidando una base sólida para el desarrollo de soluciones futuras.

Finalmente, este proyecto no solo resuelve un problema puntual, sino que establece las bases para el diseño y construcción de sistemas informáticos orientados a la optimización de rutas en distintos dominios, permitiendo continuar su evolución hacia soluciones más inteligentes, personalizadas y sostenibles.

6.2 Futuras líneas de trabajo

Las posibles líneas a futuro que quedan abiertas para poder continuar con el proyecto son:

1. Incorporación de un sistema de navegación en tiempo real.
2. Control de métricas en tiempo real para la generación de estadísticas avanzadas.
3. Expansión a más países.
4. Función social para compartir rutas e interactuar con la comunidad de viajeros.
5. Interacción del asistente inteligente con el problema.
6. Optimizaciones de rutas dinámicas y tiempo real.
7. Uso de sensores y telemática para el procesamiento de datos en tiempo real de la ruta.
8. Inclusión de otras modalidades del proceso de transporte: objeto, ámbito, modo y naturaleza.
9. Integración de más contexto y casos de uso.
10. Ingreso como invitado, con acceso reducido, para descubrir todas las funcionalidades que se ofrecen.
11. Implementación del módulo de suscripciones para monetización.

BIBLIOGRAFÍA

- [1] J. C. Daza y J. A. Espinosa, “Hacia una arquitectura nacional para los sistemas inteligentes de transporte”, *Publicación Técnica N° 251*, núm. 251, 2004.
- [2] J. R. Requejo Falla, “Aplicación web utilizando geolocalización en tiempo real y aplicando la teoría de redes, para mejorar el proceso de trazabilidad de rutas y la distribución de insumos”, Tesis de grado, Universidad Católica Santo Toribio de Mogrovejo, Chiclayo, Perú, 2021. Consultado: el 12 de octubre de 2021. [En línea]. Disponible en: <http://tesis.usat.edu.pe/handle/20.500.12423/3296>
- [3] J. Abendibar, “Bidaiatzen. Aplicación web para viajar”, Trabajo final, Universitat Oberta de Catalunya, Barcelona, España, 2017. Consultado: el 23 de septiembre de 2023. [En línea]. Disponible en: <https://openaccess.uoc.edu/handle/10609/59535>
- [4] J. C. Vaquero, “Desarrollo de un sistema multimodal para la planificación de viajes”, Proyecto de grado, Universidad Carlos III de Madrid, Madrid, España, 2013. Consultado: el 23 de septiembre de 2023. [En línea]. Disponible en: <https://e-archivo.uc3m.es/handle/10016/18752>
- [5] F. Pérez, “Planificador colaborativo de viajes”, Trabajo de grado, Universitat Oberta de Catalunya, Barcelona, España, 2015. Consultado: el 23 de septiembre de 2023. [En línea]. Disponible en: <https://openaccess.uoc.edu/handle/10609/45846>
- [6] P. García Carbonell, “TouristApp: Planificador automático de viajes turísticos”, Trabajo de grado, Universidad de Alicante, San Vicente del Raspeig, España, 2023. Consultado: el 23 de septiembre de 2023. [En línea]. Disponible en: <http://rua.ua.es/dspace/handle/10045/135539>
- [7] A. Mesa Hernández, “Aplicación web para la creación de itinerarios turísticos personalizados”, Trabajo de grado, Universidad de La Laguna, San Cristóbal de La Laguna, España, 2022. Consultado: el 23 de septiembre de 2023. [En línea]. Disponible en: <https://riull.ull.es/xmlui/handle/915/29386>

- [8] C. Molina, “Optimización de rutas para la gestión de pedidos y entregas aplicado a las empresas distribuidoras de productos”, Proyecto de investigación, Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador, 2016. Consultado: el 12 de octubre de 2021. [En línea]. Disponible en: <http://repositorio.uteq.edu.ec/handle/43000/1875>
- [9] M. S. D’Cristóforo y A. H. Giannoni, “Asistencia inteligente en la planificación personalizada de la movilidad urbana”, Tesis de grado, Universidad Nacional del Centro de la Provincia de Buenos Aires, Provincia de Buenos Aires, Buenos Aires, 2016. Consultado: el 15 de octubre de 2021. [En línea]. Disponible en: <https://www.ridaa.unicen.edu.ar/xmlui/handle/123456789/1071>
- [10] J. L. de los Mozos Quiroga y S. Moreno López, “Optimización de flotas de vehículos, una herramienta para incrementar la eficiencia”, *Universia Business Review*, núm. 16, pp. 118–127, 2007.
- [11] C. G. De La Roja, *Transportes turísticos*, 1a ed. Madrid: Editorial Universitaria Ramon Areces, 2012.
- [12] Ecologistas, “¿Qué entendemos por movilidad?”, Ecologistas en Acción. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.ecologistasenaccion.org/9844/que-entendemos-por-movilidad/>
- [13] Z. Daoqing y J. Mingyan, “Parallel discrete lion swarm optimization algorithm for solving traveling salesman problem”, *Chinese Journal of Systems Engineering and Electronics*, vol. 31, núm. 4, pp. 751–760, ago. 2020, doi: 10.23919/JSEE.2020.000050.
- [14] G. B. Dantzig y J. H. Ramser, “The Truck Dispatching Problem”, *Manage Sci*, vol. 6, núm. 1, pp. 80–91, oct. 1959, doi: 10.1287/MNSC.6.1.80.
- [15] P. X. Zhao, W. H. Luo, y X. Han, “Time-dependent and bi-objective vehicle routing problem with time windows”, *Advances in Production Engineering & Management*, vol. 14, núm. 2, pp. 201–212, jun. 2019, doi: 10.14743/apem2019.2.322.

- [16] R. S. Pressman, *Ingeniería del Software: Un enfoque práctico*, 7a ed. México : McGraw-Hill Interamericana Editores, S.A, 2010.
- [17] I. Sommerville, *Ingeniería de software*. México: Pearson Educación de México, 2011.
- [18] J. A. Montilva, N. Arapé, y J. A. Colmenares, “Desarrollo de Software Basado en Componentes”, *IV Congreso de Automatización y Control*, nov. 2003.
- [19] D. M. Alaimo, *Proyectos ágiles con Scrum: Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos*, 1a ed. Ciudad Autónoma de Buenos Aires: Kleer, 2013. [En línea]. Disponible en: <http://www.kleer.la>
- [20] J. Martins, “¿Qué es la metodología Kanban y cómo funciona? [2022] • Asana”, Asana. Consultado: el 20 de febrero de 2024. [En línea]. Disponible en: <https://asana.com/es/resources/what-is-kanban>
- [21] B. Legierski, “Frontend vs Backend”, Evertop. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.evertop.pl/en/frontend-vs-backend/>
- [22] O. Blancarte, *Introducción a la arquitectura de software: Un enfoque práctico*, 1a ed. Ciudad de México : Independently published, 2020.
- [23] Mozilla Developer Network, “Introducción al protocolo HTTP – MDN Web Docs”, MDN Web Docs. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://developer.mozilla.org/es/docs/Web/HTTP>
- [24] Cloudflare, “¿Qué es el SSL? | Definición de SSL”, Cloudflare. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.cloudflare.com/es-es/learning/ssl/what-is-ssl/>
- [25] Cloudflare, “¿Qué es TLS (Transport Layer Security)?”, Cloudflare. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.cloudflare.com/es-es/learning/ssl/transport-layer-security-tls/>
- [26] D. Vargas, “¿Qué es TLS? definición, funcionamiento y diferencias con SSL y HTTPS”, Tutoriales Hostinger. Consultado: el 19 de febrero de 2024.

- [En línea]. Disponible en: <https://www.hostinger.com.ar/tutoriales/que-es-tls>
- [27] Google, “Descripción general del diseño adaptable - Ayuda de Google Web Designer”. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: https://support.google.com/webdesigner/answer/7002913?hl=es&ref_topic=12756364&sjid=15413345294349533153-SA
- [28] Y. Hassan Montero y F. J. Martín Fernández, “Qué es la Accesibilidad Web”, No Solo Usabilidad. Consultado: el 14 de abril de 2025. [En línea]. Disponible en: <https://www.nosolousabilidad.com/articulos/accesibilidad.htm?iframe=true&width=90>
- [29] World Wide Web Consortium (W3C), “Introduction to Web Accessibility”, W3C. Consultado: el 14 de abril de 2025. [En línea]. Disponible en: <https://www.w3.org/WAI/fundamentals/accessibility-intro/>
- [30] World Wide Web Consortium (W3C), “Web Content Accessibility Guidelines (WCAG) 2.2”, W3C. Consultado: el 13 de abril de 2025. [En línea]. Disponible en: <https://www.w3.org/TR/WCAG22/>
- [31] Amazon, “¿Qué es una API de RESTful? - Explicación de API de RESTful - AWS”, Amazon Web Services, Inc. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/restful-api/>
- [32] IBM, “What is a REST API? | IBM”, IBM. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.ibm.com/topics/rest-apis>
- [33] IBM, “What is an Application Programming Interface (API)? | IBM”, IBM. Consultado: el 24 de febrero de 2024. [En línea]. Disponible en: <https://www.ibm.com/topics/api>
- [34] S. Chacon y B. Straub, *Pro Git*. Berkeley, CA: Apress, 2014. doi: 10.1007/978-1-4842-0076-6.
- [35] M. B. Jones, J. Bradley, y N. Sakimura, “RFC 7519: JSON Web Token (JWT)”, IETF Datatracker. Consultado: el 17 de marzo de 2024. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc7519>

- [36] Google, “¿Qué es una PaaS? | Google Cloud”, Google Cloud. Consultado: el 15 de marzo de 2024. [En línea]. Disponible en: <https://cloud.google.com/learn/what-is-paas?hl=es>
- [37] OpenAI, “Real-Time Guide: Connect with WebRTC”, OpenAI Platform Documentation. Consultado: el 15 de abril de 2025. [En línea]. Disponible en: <https://platform.openai.com/docs/guides/realtime#connect-with-webrtc>
- [38] Mozilla Developer Network, “WebRTC API – MDN Web Docs”, MDN Web Docs. Consultado: el 15 de abril de 2025. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/WebRTC_API
- [39] Google, “WebRTC: Web Real-Time Communication”, WebRTC.org. Consultado: el 15 de abril de 2025. [En línea]. Disponible en: <https://webrtc.org/?hl=es-419>
- [40] Project Management Institute, *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)*, 6a ed. Newtown Square, Pennsylvania, EE.UU.: Project Management Institute, 2017. [En línea]. Disponible en: www.youtube.com/channel/UCJaX3Cg9ij60pYg_K8zyoFg
- [41] I. L. Spigariol, “Fundamentos teóricos de los Paradigmas de Programación Buenos Aires”, Buenos Aires, Argentina, abr. 2005.
- [42] J. Bonér, D. Farley, K. Roland, y M. Thompson, “The Reactive Manifesto”, Reactive Manifesto. Consultado: el 29 de octubre de 2023. [En línea]. Disponible en: <https://www.reactivemanifesto.org/>
- [43] A. Shvets, *Sumérgete en los patrones de diseño*. 2021.
- [44] E. Gamma, R. Helm, R. Johnson, y J. Vlissides, *Patrones de diseño. Elementos de software orientado a objetos reutilizable*, 1a ed. Madrid: Pearson Educación, S.A., 2003.
- [45] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*, 1a ed. United States of America: Pearson Education Limited, 2013.

- [46] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. United States of America: Pearson Education, Inc., 2017. [En línea]. Disponible en: www.wowebook.org
- [47] C. Losada, "Single Page App VS Multi Page App: ¿cuál elegir para tu web? - Making Science", Making Science. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.makingscience.es/blog/single-page-app-vs-multi-page-app-cual-elegir-para-tu-web/>
- [48] Mozilla Developer Network, "SPA (Single Page Application) – MDN Web Docs", MDN Web Docs. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
- [49] H. Farias, "SPA vs. MPA: ¿Cuál es la elección adecuada para tu proyecto?", LinkedIn. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://www.linkedin.com/pulse/spa-vs-mpa-cu%C3%A1l-es-la-elecci%C3%B3n-adecuada-para-tu-proyecto-henry-farias/>
- [50] K. C. Dodds, "Revolutionizing Web Architecture: PESPA - The Future of Single Page Apps", T. Schlottke, episodio 72, el 17 de marzo de 2023. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://open.spotify.com/show/08QvFNH32V1yRiwfNQhTYx>
- [51] F. Herrera, "01- ¿Qué son las aplicaciones web progresivas?", YouTube, el 15 de octubre de 2018. Consultado: el 23 de febrero de 2024. [En línea]. Disponible en: <https://www.youtube.com/watch?v=ttwwOecPS10>
- [52] F. Beaufort, P. LePage, T. Steiner, y A. Rodionov, "Add a web app manifest", web.dev. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://web.dev/articles/add-manifest>
- [53] Mozilla Developer Network, "IndexedDB API – MDN Web Docs", MDN Web Docs. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/IndexedDB_API
- [54] A. Osmani y J. Miller, "Rendering on the Web", web.dev. Consultado: el 19 de febrero de 2024. [En línea]. Disponible en: <https://web.dev/articles/rendering-on-the-web>

ANEXOS

Anexo 1: Conceptos relacionados

Este anexo detalla los conceptos relacionados con la solución que forman parte del marco teórico, pero no son relevantes para entender el problema desde un punto de vista funcional.

1.1 Paradigmas de programación

Un paradigma de programación es una forma o estilo para resolver los problemas en programación que nace de la necesidad de estrategias para estructurar y organizar las actividades en el software. La necesidad de estas estrategias surge debido a que existen muchas formas de encontrar una solución y de que la implementación de esta solución va a depender de las características del lenguaje de programación.

En el desarrollo de software existen diversas maneras de clasificar los paradigmas según qué criterio se priorice. Para nuestro trabajo no entraremos en detalle en la clasificación sino más bien analizaremos que paradigma existente nos será de utilidad. Por lo que a nuestra necesidad solo nos centraremos en la programación estructurada, programación orientada a objetos, programación funcional y programación reactiva.

La programación estructurada es la forma clásica en la que aprendemos a resolver problemas, la cual está basada en las estructuras básicas de control de un programa, es decir, en instrucciones secuenciales e implementación de ciclos y condicionales. La programación estructurada nace de una serie de técnicas para facilitar el desarrollo de sistemas en lenguajes que se basen en el paradigma imperativo —la ejecución repetitiva paso a paso de operaciones— pero adquiere ideas que fueron tenidas en cuenta en lenguajes de otros paradigmas. La base teórica argumenta que cualquier programa, por más complejo que sea, debe representarse de acuerdo con las estructuras de control: secuencia, selección e iteración. La idea es desarrollar el programa desde arriba hacia abajo proporcionando modularidad, es decir, dividiendo el programa en porciones que cumplan tareas específicas [41].

La programación orientada a objetos es el estilo más utilizado en el mundo que implica representar los problemas con entidades del mundo real, es decir, se concibe a todo el sistema como un conjunto de objetos que tienen un comportamiento y una clase determinada, y que a su vez se comunican entre sí para compartir datos. La programación

orientada a objetos es un paradigma que pretende desarrollar software lo más parecido a la realidad, facilitando la construcción de componentes y definiendo una estructura que pueda ser alterada mediante un bajo impacto [41]. Esta definición la complementa con desarrollar de forma modular basándose en objetos y teniendo en cuenta características peculiares como encapsulamiento, abstracción, polimorfismo y herencia.

La programación funcional es un estilo de programar que consiste en dividir el programa en funciones que definen tareas específicas. A su vez, cada función se comunica con otra, para que en conjunto se realice una tarea mayor. Este paradigma está basado en el modelo matemático de composición funcional, donde el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que se encuentre el resultado deseado. Por lo que un programa queda estructurado en una serie de funciones que cooperan para lograr un propósito [41].

La programación reactiva es un paradigma que está enfocado en escuchar los flujos de datos y responder automáticamente ante cambios de estos. En cierto sentido trabaja con flujos de datos finitos o infinitos de forma asíncrona. Los sistemas reactivos deben responder de manera oportuna si es posible y bajo diferentes cargas de trabajo, permanecer receptivo ante una falla, así como minimizar el acoplamiento entre componentes al establecer interacciones basadas en mensajes asíncronos. Estas cualidades previamente mencionadas hacen a un sistema responsivo, elástico, resiliente y basado en mensajes [42].

1.2 Patrones de diseño

Los patrones de diseño se conocen como buenas prácticas en el mundo del desarrollo de software que describen soluciones a problemas comunes en programación. En términos bibliográficos, los patrones de diseño son soluciones habituales a problemas comunes en el diseño de software [43].

Un patrón de diseño no se trata de una porción de código, sino a una forma conceptual para resolver algún problema en específico dentro de la programación, por lo que no es un algoritmo sino una descripción de alto nivel de una posible solución que nace de encontrarla reiteradas veces en varios proyectos.

Un patrón se compone de un nombre, el problema, la solución y las consecuencias. Por una parte, el nombre permite describir en pocas palabras el problema. Por otra parte, el

problema describe cuando aplicar el patrón. A su vez la solución describe los elementos necesarios para atacar el problema. Y por último las consecuencias están relacionadas con las ventajas y desventajas de aplicar el patrón [44].

Como los patrones de diseño comprenden un conjunto de soluciones habituales a problemas comunes en el desarrollo de software es de suma utilidad conocerlos, aun cuando no sean de nuestra necesidad, ya que nos enseñan un enfoque con buenas prácticas utilizando principios del diseño orientado a objetos. Además, conocerlos nos permite establecer un lenguaje común entre los programadores de manera que al encontrar un problema todos sepamos a que se refiere y como solventarlo.

También cabe destacar que el uso de patrones de diseño dependerá en cierto sentido del entorno de trabajo en el que nos encontremos y que existen entornos que tienen sus propios patrones exclusivos. En relación con esto, existen patrones que se implementan para tratar con el negocio o la aplicación permitiéndonos crear objetos y definir estructuras o comportamientos, pero también existen patrones que tienen que ver con la definición de la interfaz de usuario.

El uso de patrones de diseño es una forma de mejorar la flexibilidad de los componentes y facilitar su reutilización a lo largo del software u otro proyecto.

1.3 Principios SOLID

Los principios SOLID engloban un grupo de cinco principios básicos y fundamentales en el diseño de software.

Estos principios fueron ideados con el objetivo de crear software robusto, estable y escalable, así como escribir código limpio y flexible que permita la reutilización y el mantenimiento. En síntesis, la finalidad es realizar productos de calidad siguiendo un conjunto de reglas y mejores prácticas.

Cada letra del acrónimo SOLID se refiere a una de las cinco reglas o principios para diseñar software. A continuación, se menciona una breve descripción de cada principio:

1. Principio de responsabilidad única (SRP): las clases deben tener una sola razón para cambiar por lo que cada clase debe ser responsable de una única parte de la funcionalidad del software. A su vez esa responsabilidad debe quedar totalmente encapsulada de manera que permita reducir la complejidad.

2. Principio de abierto/cerrado (OCP): las entidades de software (clases, módulos, funciones, etc.) deben estar abiertas a su extensión, pero cerradas en la modificación de manera que se evite la descomposición del código existente cuando vayamos a implementar nuevas características.
3. Principio de sustitución de Liskov (LSP): cada subclase debe permanecer compatible con el comportamiento de la superclase de manera que las posibles instancias de objetos de una superclase pueden ser reemplazadas por instancias de objetos de sus subclases sin alterar la funcionalidad.
4. Principio de segregación de la interfaz (ISP): las interfaces deben ser específicas a los clientes de manera que los mismos no dependan de métodos que no utilicen. Es preferible tener muchas interfaces que definan pocas propiedades y métodos que forzar a una misma a tener muchos de estos ya que si, por necesidad, se realizan cambios en alguna propiedad o método se alterarían clientes que tal vez ni siquiera se utilizan.
5. Principio de inversión de la dependencia (DIP): por un lado, los módulos de alto nivel no deberían depender de los módulos de bajo nivel, en tal caso ambos deberían depender de abstracciones. Por otro las abstracciones no deberían depender de los detalles sino más bien los detalles de las abstracciones [45].

Conocer estos principios nos será de utilidad para introducir calidad en el desarrollo de software, siendo más específico en la etapa de programación.

1.4 Arquitectura Limpia

La arquitectura de un sistema software es la forma que le dan a ese sistema quienes lo construyen. La forma viene determinada por la división del sistema en componentes, la disposición de esos componentes y la manera en que esos componentes se comunican entre sí [46].

Una arquitectura limpia está relacionada con establecer una estructura que facilite el desarrollo, despliegue, operación y mantenimiento del sistema software haciéndolo robusto y escalable.

La arquitectura de un sistema tiene muy poca relación con el funcionamiento correcto del mismo sino más bien con su implementación, mantenimiento y desarrollo continuo. Esto último no significa que la arquitectura no desempeñe un papel en el comportamiento adecuado de un sistema, en cierto sentido lo hace y es fundamental pero no esencial.

El objetivo principal de la arquitectura es soportar el ciclo de vida del sistema haciendo al mismo fácil de entender, desarrollar, mantener e implementar. Además, el objetivo final es minimizar el costo de vida útil del sistema y maximizar la productividad del programador [46].

Tener una estructura que este particionada en módulos y componentes, y basada en abstracciones permitirán introducir simplicidad, claridad y legibilidad en el proceso de desarrollo, y agilizar la etapa de programación.

1.5 Aplicaciones web de acuerdo con su estructura y comportamiento

El desarrollo web ha experimentado una evolución significativa en términos de experiencia de usuarios y desarrolladores. Muchos sectores han digitalizado su negocio con el propósito de alcanzar mejores resultados. Por un lado, para un negocio la clave en obtener mejores resultados radica en la experiencia que se ofrece a los usuarios. Por otro, para las empresas desarrolladoras de software, además de crear software que garantice la experiencia de usuario, también se tiene en cuenta la experiencia con la que el desarrollador se enfrenta con los problemas de programación durante la creación [47].

Cuando se trata de desarrollar aplicaciones web tenemos múltiples alternativas que nos proveen diferentes beneficios. Su elección va a depender de las necesidades de negocio. De acuerdo con nuestros requerimientos, tenemos:

1. Single Page Application (SPA): es un tipo de aplicación web que carga solo un documento web, es decir, carga una sola página HTML y luego actualiza el contenido dinámicamente a medida que el usuario interactúa con la aplicación. La actualización dinámica se realiza a través de la API de JavaScript, cuando se debe mostrar un contenido diferente [48]. Al actualizar el contenido dinámicamente dentro de una sola página, ofrece una experiencia de usuario fluida y rápida, lo que hace que sea ideal para crear aplicaciones interactivas y altamente personalizables [49].

Por lo tanto, las SPAs fueron desarrolladas para manejar interacciones más complejas en el lado del cliente permitiendo crear aplicaciones ágiles y responsivas. Además, al cargar solo una vez la página inicial y actualizar el contenido de forma dinámica, minimizan el tiempo de carga y las solicitudes al servidor [49]. Sin embargo, debido a la falta de contenido indexable y otras particularidades que pueden aparecer de no seguir buenas prácticas de

programación o maquetación presentan problemas cuando se requiere de SEO. Adicionalmente, los desarrolladores tratan desafíos como la eliminación de código duplicado, procesos de compilación complejos y combinación de diferentes tecnologías, lo que deja en claro que, si bien las SPAs proveen una excelente experiencia de usuario, no lo es así con la experiencia de desarrollador [50].

2. Multiple Page Application (MPA): es un tipo de aplicación web que carga diferentes documentos web, es decir, múltiples páginas HTML con diferentes contenidos apoyándose en la navegación contra el servidor. Este tipo de aplicación sigue el modelo tradicional de navegación web, cargando páginas completas desde el servidor. En una aplicación MPA todo el código vive en el servidor y cada página se conecta mediante links con las demás.

Las MPAs fueron las primeras que se desarrollaron cuando se introdujo la web y son consideradas de las más adecuadas para proyectos con múltiples páginas y funcionalidades complejas. Además, ofrecen una ventaja en términos de SEO, debido a que cada página tiene su propia URL y por lo tanto puede ser indexada por los motores de búsqueda de manera más efectiva. Sin embargo, como cada página representa una petición específica al servidor puede tener una velocidad de carga ligeramente más lenta. Es claro entonces, que si por cada vez que el usuario navega a una nueva página se carga una página completamente nueva, la experiencia de usuario será menos fluida. Adicionalmente, al recargar toda la página cuando se produce la navegación proporcionan menor interactividad y responsividad. Sin embargo, si bien perdemos experiencia de usuario la experiencia de desarrollador es mucho más agradable al contar con una menor curva de aprendizaje, SEO más sencillo, depuración simple y menor complejidad en la gestión del estado de la aplicación [50].

3. Progressively Enhanced Single Page Application (PESPA): es un tipo de aplicación que cierra la brecha entre MPA y SPA, brindando una mejor experiencia para usuarios y desarrolladores. Un PESPA representa documentos, páginas HTML, tanto en el lado del cliente como en el servidor, lo que permite a los desarrolladores crear aplicaciones ricas en funciones que ofrecen una experiencia de usuario mejorada sin sacrificar la facilidad de desarrollo.

Al crear una PESPA los desarrolladores sienten como si estuvieran construyendo una MPA, con los beneficios adicionales de experiencia de usuario similares a las

de una SPA. Esto incluye funciones como animaciones entre diferentes acciones, gestión de enfoque y mensajes de error. Adicionalmente, una PESPA también admite mejoras progresivas, lo que permite a los desarrolladores agregar fácilmente funciones como estados pendientes y UI optimista.

El funcionamiento de una PESPA se logra bajo los conceptos de carga inicial de documento contra el servidor, enrutamiento a medida que los usuarios navegan entre páginas y revalidación de página en base a acciones. Esto es posible debido a que proporciona un marco para una comunicación fluida entre el cliente y el servidor gestionando tareas como carga de datos, renderizado del lado del servidor y transiciones del lado del cliente [50].

4. Progressive Web Application (PWA): es un tipo de aplicación web que progresivamente va implementando características nativas de dispositivos, funciona sin conexión a internet, se actualiza constantemente, es atractiva para los usuarios, es confiable, pesa muy poco y es rápida a la hora de cargar. Una PWA bien desarrollada es totalmente indiferenciable de una aplicación nativa. Sin embargo, las aplicaciones nativas pesan mucho más que una PWA y dependiendo el caso, los usuarios no siempre disponen de la última actualización [51].

La implementación de una PWA radica en el hecho que la mayor parte de nuestros visitantes web utilizan dispositivos móviles para acceder a la aplicación, pero su participación es más constante en las aplicaciones nativas del dispositivo, esto quiere decir que los usuarios que visitan nuestra web en su mayor medida no regresan. Por otra parte, es muy alto el porcentaje de usuarios que no descargan aplicaciones nativas en su dispositivo móviles debido a que por lo general utilizan las aplicaciones más usuales o las que vienen instaladas por defecto por lo que hay una gran cantidad de aplicaciones nativas que no se llegan a usar.

Para convertir una aplicación web en PWA solo basta con incorporar lo siguiente:

- a. Service Worker: es un archivo de JavaScript que actúa como proxy fuera de nuestra aplicación, es decir, es totalmente independiente y corre en otro hilo. Se encarga de administrar las solicitudes y las respuestas HTTPS [51].
- b. Manifest: es un archivo en formato JSON que informa al navegador sobre la PWA y cómo debe proceder cuando se instala en el escritorio o en un dispositivo móvil del usuario [52].

- c. IndexedDB: es una API de bajo nivel que ofrece almacenamiento en el cliente de cantidades significativas de datos estructurados, incluyendo archivos y blobs [53].
- d. Push Server: es el servidor encargado de mantener y manejar las suscripciones a las notificaciones push. Una notificación push es una manera que tiene un servidor para mandar una notificación directamente al dispositivo [51].

Además, hay que tener en cuenta que es necesario que contemos con un certificado SSL/TLS en nuestro servidor web, es decir, que nuestra aplicación cuente con seguridad y utilice el protocolo HTTPS, anteriormente explicado. Esto se debe a que los Service Worker tienen demasiado poder sobre la aplicación y cualquier persona podría interceptarlo para realizar actividades indebidas. La única excepción a esto es cuando se trabaja en modo de desarrollo.

En una PWA las peticiones HTTPS para solicitar recursos de la aplicación, como archivos de JavaScript, HTML, imágenes, entre otros, son interceptadas por el Service Worker. Este último, recibe las solicitudes y antes de ir al servidor web revisa si el recurso ya existe en cache. Por lo tanto:

- a. En caso de existir regresa la respuesta al navegador web, sin interactuar con el servidor web.
- b. En caso de no existir, reenvía la solicitud al servidor web, quien se encarga de devolverle una respuesta. Esta respuesta puede ser almacenada en cache, por si posteriormente el usuario solicita el mismo recurso. Para finalizar retorna la respuesta al navegador web.

La Figura 64 muestra un esquema que representa como se comunican el navegador y el servidor web en una PWA. Además, muestra como intervienen los conceptos de service worker, cache e indexedb en una aplicación de este tipo.

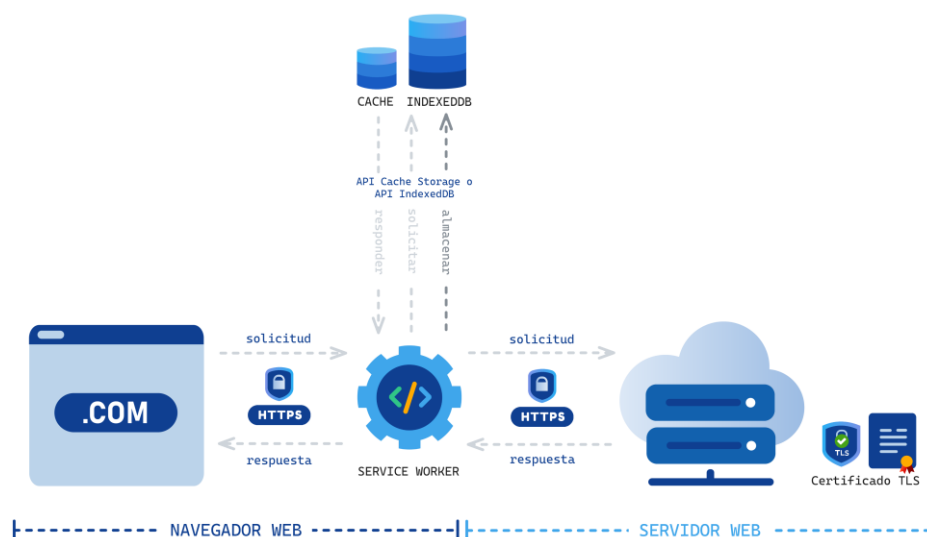


Figura 64. Funcionamiento de una Progressive Web Application

1.6 Estrategias de renderizado y generación de páginas web

El propósito del desarrollo web y del desarrollador web no solo es crear aplicaciones web funcionales, sino que dicha aplicación sea satisfactoria para los usuarios y siempre que sea necesario para el posicionamiento competitivo.

Al momento de crear una aplicación web lo que hay que tener en cuenta es a cuál negocio pertenece y cuál será el uso que se le dará.

Por un lado, si se tratase de una aplicación dirigida al público, existen ciertas particularidades que si tenemos en cuenta a la hora de desarrollarla nos darán cierta ventaja frente a la competencia. Primero, si la aplicación cuenta con una buena optimización para motores de búsqueda (SEO) será más accesible a los usuarios, facilitará destacarse frente a la competencia, dará mayor visibilidad en la industria y se tendrá una mejor experiencia de usuario. Segundo, en cuanto a lo que respecta de experiencia de usuario, la misma define la permanencia de los visitantes en la aplicación. Por si fuera poco, la experiencia de usuario depende del rendimiento de la aplicación que lograremos mejorar o mantener distribuyendo la carga de trabajo entre los diferentes componentes de la arquitectura, siguiendo buenas prácticas de desarrollo y explotando los recursos disponibles. Además, existen métricas claves que miden la posibilidad de que un usuario abandone la aplicación [54]. Una de ellas es el tiempo de renderizado de cada una de las páginas web y está totalmente relacionada con el rendimiento de la aplicación.

Por otro lado, si se tratase de una aplicación dirigida a uso interno de una organización solo se debería hacer énfasis en la experiencia de usuario ya que no disponemos de la necesidad de captar a un público.

En relación con esto tenemos diferentes estrategias de renderizado y generación de páginas web que podemos utilizar o combinar para desarrollar una aplicación web:

1. Client Side Rendering (CSR): es una estrategia que consiste en renderizar la página web en el lado del cliente de acuerdo con los recursos obtenidos del servidor. En esta estrategia cuando el usuario ingresa al dominio, el navegador web solicita los recursos al servidor web. El servidor web responde enviando una página HTML en blanco junto con un bundle JS (archivo de JavaScript). A partir de aquí todo sucede en el cliente. El bundle JS genera dinámicamente los nodos del DOM para terminar de renderizar la página. Posteriormente al navegar por la aplicación se manipula el DOM para renderizar los componentes de software y páginas necesarias. La Figura 65 muestra cómo y dónde se renderiza una página web que sigue esta estrategia.

En CSR disponemos de alta interactividad y carga reducida en el servidor, pero tenemos un SEO comprometido y el tiempo de carga inicial de la aplicación es más lento. Esto último se puede apaciguar incorporando una técnica conocida como Prerendering, la cual consiste en renderizar de antemano el contenido de las páginas que no cambian con frecuencia en un archivo HTML estático, por separado y en el servidor en tiempo de construcción. Por lo que, cuando se solicite una página prerenderizada, el servidor entregaría HTML estático al cliente quien posteriormente ejecutaría scripts con el propósito de hidratar la página, es decir, agregarle interactividad y estado. Esto último se conoce como CSR con Prerendering.

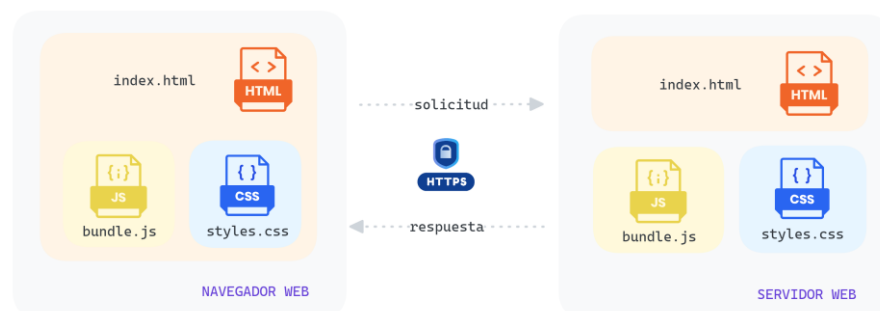


Figura 65. Client Side Rendering

2. **Server Side Rendering (SSR):** es una estrategia que consiste en renderizar la página web en el lado del servidor y posteriormente enviarla al cliente. Cuando un usuario solicita una página, el servidor obtiene los datos necesarios y genera el contenido HTML. Luego, la página es renderizada completamente y se envía al cliente. La Figura 66 muestra cómo, dónde y en qué momento se renderiza una página web que sigue esta estrategia.

En SSR el tiempo de carga inicial de la aplicación es más rápido y disponemos de un SEO mejorado, pero hay mayor carga en el servidor y una interactividad limitada lo que puede afectar la experiencia de usuario. Además, al trabajar la mayor parte de código en el servidor es más complejo en comparación con CSR. Para mejorar la interactividad podemos implementar una técnica conocida como Rehidratación, para introducirle interactividad y estado a las páginas desde el lado del cliente. Esta combinación se conoce como SSR con Rehidratación.

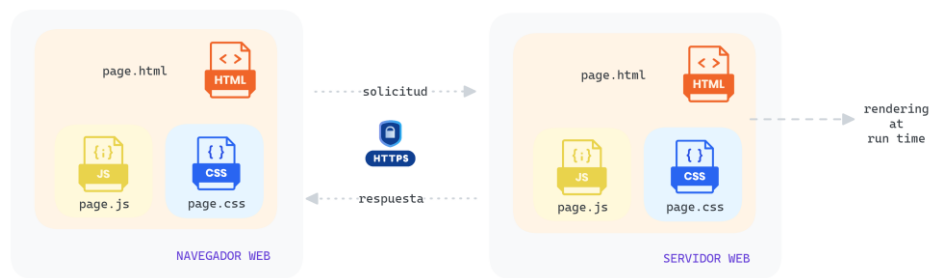


Figura 66. Server Side Rendering

3. **Static Site Generation (SSG):** es una estrategia similar a SSR solo que las páginas web se generan en el servidor en tiempo de construcción, es decir, cuando ejecutamos el build de la aplicación. El funcionamiento es simple, cuando un usuario solicita una página, el servidor responde de forma inmediata con la misma al ya tenerla generada. Queda claro que cada vez que el contenido o algún elemento de la página deba cambiar, nuestra aplicación debe pasar nuevamente por el proceso de construcción. Esta estrategia es ideal para sitios web o páginas web que incluyan contenido estático. La Figura 67 presenta el momento y el lugar donde se renderiza y genera una página web construida bajo esta estrategia.
- En esta estrategia disponemos de SEO, seguridad mejorada y rendimiento excelente, pero la interactividad y las actualizaciones en tiempo real se encuentran limitadas.

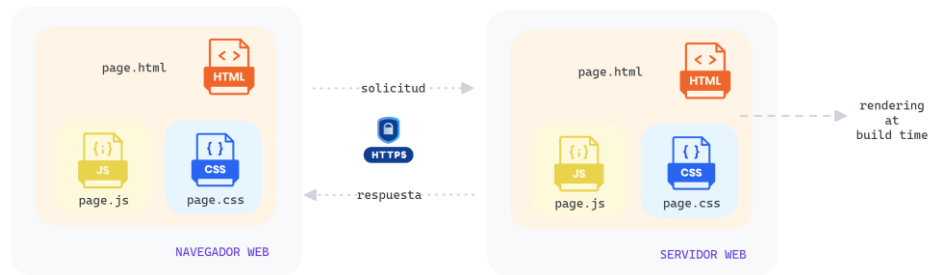


Figura 67. Static Site Generation

4. Incremental Static Regeneration (ISR): es una estrategia que permite a los desarrolladores actualizar contenido estático una vez completado el proceso de construcción, es decir, toma como base elementos de SSG y SSR, y da la posibilidad, mediante una configuración, de crear nuevas páginas o actualizar el contenido de estas después de la implementación. Entonces, ISR regenera páginas bajo demanda o actualiza el contenido mediante revalidación en tiempo de ejecución, lo que garantiza que la aplicación web este actualizada sin la necesidad de realizar una reconstrucción. La Figura 68 presenta los dos momentos en los que se puede renderizar una página web incluyendo las acciones posteriores al renderizado como la revalidación y el regenerado de la página.

Esta estrategia equilibra el contenido estático y las actualizaciones en tiempo real, pero a su vez requiere de una activación cuidadosa para datos dinámicos.

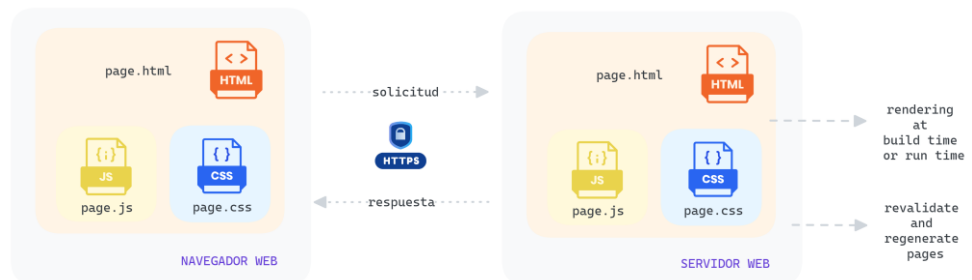


Figura 68. Incremental Site Regeneration

Anexo 2: Otras tecnologías implicadas en el desarrollo

Este anexo detalla el resto de las tecnologías implicadas en el desarrollo del sistema.

2.1 Lenguajes

1. TypeScript

Superset de JavaScript que incorpora tipado estático opcional y otras características propias de lenguajes fuertemente tipados. Se utilizó tanto en el

frontend como en el backend para mejorar la mantenibilidad del código, prevenir errores en tiempo de desarrollo y favorecer la escalabilidad.

2. HTML5

Lenguaje de marcado estándar utilizado para estructurar las interfaces de usuario en la web. Proporciona elementos semánticos que mejoran la accesibilidad, el SEO y la compatibilidad con dispositivos móviles.

3. CSS3

Lenguaje de hojas de estilo utilizado para definir la presentación visual del sitio. Aunque en este proyecto se utilizó principalmente mediante Tailwind CSS, CSS3 proporciona la base técnica para la definición de estilos responsivos, animaciones y diseño visual moderno.

2.2 Frontend

1. React

Librería de interfaz de usuario declarativa para la construcción de interfaces dinámicas y reutilizables. Fue la base de los componentes interactivos del sistema.

2. Tailwind CSS

Framework de utilidades CSS que permite un diseño ágil, mantenible y coherente, directamente desde el marcado HTML.

2.3 Backend

1. Passport

Middleware de autenticación que se integró con estrategias locales y JWT, utilizado para el inicio de sesión y validación de usuarios.

2. Nodemailer

Librería para el envío de correos electrónicos desde el servidor, utilizada en flujos como verificación de identidad y recuperación de contraseña.

3. Prisma ORM

Mapeador objeto-relacional que permitió la definición tipada de esquemas y relaciones en la base de datos PostgreSQL, favoreciendo la validación y seguridad.

2.4 Entorno de desarrollo

1. Docker

Plataforma de contenedores utilizada para levantar entornos locales replicables, incluyendo servicios como PostgreSQL y Redis, asegurando homogeneidad entre desarrollo y producción.

2. Prettier y ESLint

Herramientas utilizadas para formatear automáticamente el código (Prettier) y aplicar reglas de buenas prácticas y consistencia (ESLint), mejorando la calidad del software.

2.5 Herramientas de gestión

1. DBeaver

Cliente universal para la administración y consulta de bases de datos, utilizado para gestionar datos de PostgreSQL durante el desarrollo.

2. RedisInsight

Herramienta visual para la administración de instancias Redis. Facilitó la inspección de claves, métricas de rendimiento y depuración en tiempo real.

Anexo 3: Instalación local del sistema

Este anexo detalla el procedimiento requerido para ejecutar el sistema de forma local, tanto en el entorno del servidor (backend) como del cliente (frontend). Se describen los prerequisites, pasos de instalación y ejecución en entorno de desarrollo.

3.1 Instalación del backend

El código fuente correspondiente al backend de la aplicación se encuentra en el siguiente repositorio de GitHub:

<https://github.com/dlamas685/journeys-api>

Requisitos previos.

1. Node.js versión 18 o superior
2. Docker
3. Archivo `.env` con las variables de entorno definidas

Pasos para la ejecución local.

1. Clonar el repositorio del backend:
`git clone https://github.com/dlomas685/journeys-api.git`
2. Instalar dependencias:
`npm install`
3. Crear el archivo `‘.env’` utilizando como base el archivo `‘.env.template’`.
4. Ejecutar Docker Compose:
`docker compose up -d`
5. Aplicar migraciones a la base de datos y generar el cliente de Prisma:
`npx prisma migrate dev`
6. Ejecutar el servidor en modo desarrollo:
`npm run start:dev`
7. Acceder a la documentación desde el navegador en la URL:
<http://localhost:3001/docs>

3.2 Instalación del frontend

El código fuente correspondiente al frontend de la aplicación se encuentra en el siguiente repositorio de GitHub:

<https://github.com/dlomas685/journeys-web>

Requisitos previos.

1. Node.js versión 18 o superior
2. Archivo `‘.env’` con las variables de entorno necesarias

Pasos para la ejecución local.

1. Clonar el repositorio del frontend:
`git clone https://github.com/dlomas685/journeys-web.git`
2. Instalar dependencias:
`npm install`
3. Crear el archivo `‘.env’` utilizando como base el archivo `‘.env.template’`.
4. Ejecutar el servidor de desarrollo:
`npm run dev`
5. Acceder a la aplicación desde el navegador en la URL:
<http://localhost:3000>