

OptiPark — Find & Pay for Parking (Proyecto de Desarrollo de Software)

Proyecto Integrador II

Estudiantes:

Gabriel Alejandro Sanchez Mora
Juan Carlos Urbano Montaña
Jeisson Andres Villarraga Reyes
Jhossuant Jafet Cabezas Diaz
Karen Dayanna Sanchez Pastuso

Jeferson Urrego Guerrero

Universidad de la Salle
2025

Resumen

OptiPark es una aplicación web desarrollada como solución al problema del tiempo perdido y la congestión ocasionada por la búsqueda de parqueo en zonas urbanas. La plataforma permite a los usuarios localizar espacios disponibles, reservar una franja horaria y realizar un pago simulado para garantizar la reserva. El sistema está diseñado con una arquitectura cliente-servidor RESTful (React + Node/Express + PostgreSQL/SQLite) y contiene módulos para autenticación, administración de espacios, gestión de reservas y simulación de pagos.

En el proyecto se trabajó desde la identificación de requerimientos (Fase 1), diseño y prototipado (Fase 2) hasta la implementación, pruebas y documentación (Fase 3). Se realizaron pruebas unitarias, de integración y de sistema; los resultados y correcciones aplicadas están documentados en el anexo de pruebas. El entregable incluye la documentación técnica completa, manual de usuario paso a paso y un video demostrativo (≤ 7 minutos) que muestra el flujo principal: búsqueda → reserva → pago → confirmación. Los beneficios esperados son reducción del tiempo de búsqueda de parqueo, mejor aprovechamiento de la oferta de espacios y comodidad para usuarios y administradores.

Abstract

Ídem al resumen, en Inglés.

Índice General

1	INTRODUCCIÓN	8
2	DEFINICION DE LA EMPRESA O INSTITUCIÓN	8
2.1	DESCRIPCIÓN DE LA EMPRESA.....	8
2.2	DESCRIPCIÓN DEL ÁREA DE ESTUDIO.....	8
2.3	DESCRIPCIÓN DE LA PROBLEMÁTICA.....	8
3	DEFINICIÓN PROYECTO	8
3.1	OBJETIVOS DEL PROYECTO	8
3.2	AMBIENTE DE INGENIERÍA DE SOFTWARE.....	9
3.3	DEFINICIONES, SIGLAS Y ABREVIACIONES	9
4	ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE.....	9
4.1	ALCANCES.....	9
4.2	OBJETIVO DEL SOFTWARE	9
4.3	DESCRIPCIÓN GLOBAL DEL PRODUCTO	9
4.3.1	INTERFAZ DE USUARIO	9
4.3.2	INTERFAZ DE HARDWARE	10
4.3.3	INTERFAZ SOFTWARE.....	10
4.4	REQUERIMIENTOS ESPECÍFICOS.....	10
4.4.1	REQUERIMIENTOS FUNCIONALES DEL SISTEMA	10
4.4.2	INTERFACES EXTERNAS DE ENTRADA	10
4.4.3	INTERFACES EXTERNAS DE SALIDA	11
4.4.4	ATRIBUTOS DEL PRODUCTO.....	11
5	FACTIBILIDAD.....	11
5.1	FACTIBILIDAD TÉCNICA.	11
5.2	FACTIBILIDAD OPERATIVA.	11
5.3	FACTIBILIDAD ECONÓMICA.....	12
5.4	CONCLUSIÓN DE LA FACTIBILIDAD.....	12
6	ANÁLISIS	12
6.1	DIAGRAMA DE FLUJO DE DATOS	12
6.2	CASOS DE USO	12
6.2.1	ACTORES	12
6.2.2	DIAGRAMA DE CASOS DE USO Y DESCRIPCIÓN	13
6.2.3	ESPECIFICACIÓN DE LOS CASO DE USO.....	13
6.3	MODELAMIENTO DE DATOS	13
7	DISEÑO.....	13
7.1	DISEÑO DE FÍSICO DE LA BASE DE DATOS.....	13
7.2	DISEÑO DE ARQUITECTURA FUNCIONAL	14
7.3	DISEÑO INTERFAZ Y NAVEGACIÓN.....	14
7.4	ESPECIFICACIÓN DE MÓDULOS	15
8	PRUEBAS.....	15

8.1	ELEMENTOS DE PRUEBA	15
8.2	ESPECIFICACIÓN DE LAS PRUEBAS	15
8.3	RESPONSABLES DE LAS PRUEBAS	16
8.4	CALENDARIO DE PRUEBAS.....	16
8.5	CONCLUSIONES DE PRUEBA	16
9	PLAN DE CAPACITACIÓN Y ENTRENAMIENTO	16
10	PLAN DE IMPLANTACIÓN Y PUESTA EN MARCHA	16
11	RESUMEN ESFUERZO REQUERIDO	17
12	CONCLUSIONES.....	17
13	BIBLIOGRAFÍA	17
14	ANEXO: PLANIFICACION INICIAL DEL PROYECTO.....	17
14.1.1	ESTIMACIÓN INICIAL DE TAMAÑO.....	17
14.1.2	CONTABILIZACIÓN FINAL DEL TAMAÑO DEL SW.....	17
15	ANEXO: RESULTADOS DE ITERACIONES EN EL DESARROLLO	18
16	ANEXO: MANUAL DE USUARIO.....	18
17	ANEXO: ESPECIFICACION DE LAS PRUEBAS	18
17.1	PRUEBAS DE UNIDAD	18
17.1.1	<NOMBRE UNIDAD>	18
17.2	SISTEMA	18
17.3	ACEPTACIÓN.....	19
18	ANEXO: DICCIONARIO DE DATOS DEL MODELO DE DATOS.....	19
19	EJEMPLOS (QUITAR ESTE APARTADO)	19
19.1	ISO/IEC 9126: TECNOLOGÍA DE INFORMACIÓN – EVALUACIÓN DEL PRODUCTO DE SOFTWARE	19
19.2	ESQUEMA ESPECIFICACIÓN DE INTERFAZ	20
19.3	DIAGRAMA PARA REPRESENTAR LA JERARQUÍA DE MENÚ	22
19.4	ÁRBOL DE DESCOMPOSICIÓN FUNCIONAL.....	22
19.5	ESTIMACIÓN DE TAMAÑO DE SW: PUNTO FUNCIÓN.....	23
19.6	ESTIMACIÓN DE TAMAÑO DE SW: PUNTOS DE CASOS DE USO.....	27
19.7	ASPECTOS DE SEGURIDAD INFORMÁTICA A CONSIDERAR EN PROYECTOS DE SW	28

Índice Tablas

Índice Figuras

1 INTRODUCCIÓN

El presente documento consolida las tres fases del proyecto **OptiPark**, un sistema de información orientado a optimizar la búsqueda, reserva y pago de espacios de parqueo en entornos urbanos.

Este trabajo reúne los resultados de las fases previas:

- **Fase 1:** Análisis de requerimientos y modelado de casos de uso.
- **Fase 2:** Diseño de arquitectura, diagramas UML y prototipo funcional.
- **Fase 3:** Documentación técnica, planificación de desarrollo, pruebas y manual de usuario.

Se sigue el formato IEEE para la especificación de software y las buenas prácticas de documentación.

El objetivo final es definir y estructurar un sistema de software escalable, seguro y usable, cuya primera versión (prototipo funcional) valida las interfaces principales y flujos del usuario.

2 DEFINICIÓN DEL PROYECTO Y CONTEXTO

- **2.1 Descripción del problema**

- En ciudades con alto flujo vehicular, encontrar parqueaderos disponibles genera demoras, contaminación y pérdida de tiempo. La falta de un sistema unificado para la reserva y pago de espacios de parqueo limita la eficiencia de los usuarios y de los administradores de parqueaderos.

- **2.2 Solución propuesta**

- OptiPark propone un **sistema web integral** que permita a los usuarios visualizar en tiempo real los espacios de parqueo disponibles, realizar reservas y efectuar pagos electrónicos. El sistema también ofrecerá a los administradores una plataforma de control para gestionar cupos, tarifas y reportes de ocupación.
- El prototipo funcional desarrollado en HTML5, CSS3 y JavaScript sirvió como **validación de la experiencia de usuario (UX)** y permitió comprobar la viabilidad del diseño antes de la implementación definitiva en una arquitectura cliente-servidor.

- **2.3 Alcance**

- **Incluye:**

- Módulo de visualización de mapa y disponibilidad.
- Módulo de reservas y simulación de pago.
- Módulo administrativo (gestión de espacios, usuarios y tarifas).
- Módulo de autenticación básica.
- Documentación técnica, pruebas y manual de usuario.

- **Excluye:**

- Integración con servicios reales de pasarela de pago o mapas.
 - Aplicaciones móviles nativas.
 - Implementación de hardware de sensores (solo se proyecta su futura conexión).
-

3 OBJETIVOS

1. 3.1 Objetivo general del proyecto

2. Diseñar, documentar e implementar la arquitectura técnica y funcional de un sistema de información web para la reserva y pago de espacios de parqueo, aplicando metodologías ágiles y buenas prácticas de desarrollo de software.
3. 3.2 Objetivos específicos del proyecto
4. Definir los requerimientos funcionales y no funcionales del sistema conforme a estándares IEEE.
5. Diseñar la arquitectura del sistema mediante diagramas UML y modelo de datos.
6. Implementar un prototipo funcional de la interfaz de usuario para validar la usabilidad.
7. Desarrollar el plan de pruebas y documentación técnica del sistema.
8. Presentar la propuesta en un video demostrativo que evidencie la funcionalidad y arquitectura definida.
9. 3.3 Objetivos del producto
10. Permitir la consulta, reserva y pago de parqueaderos de forma ágil y segura.
11. Mejorar la experiencia del usuario durante el proceso de búsqueda de espacios.
12. Facilitar la administración y control de cupos mediante un panel web.
13. Garantizar escalabilidad y facilidad de mantenimiento para futuras versiones.

4 ALCANCES DEL SISTEMA

A El software final está concebido como una **aplicación web modular** basada en arquitectura multicapa. El alcance abarca desde el desarrollo del backend y frontend hasta las pruebas funcionales y no funcionales.

El prototipo validó las interfaces, pero la implementación real propuesta contempla:

- Frontend desarrollado en **React**.
- Backend implementado en **Node.js + Express**.
- Base de datos **PostgreSQL**.
- Integración mediante servicios **RESTful**.
- Seguridad basada en autenticación **JWT** y cifrado **bcrypt**.

5 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

5.1 Requerimientos funcionales

ID	Descripción	Prioridad	Actor	Estado
RF-01	Permitir el registro de nuevos usuarios con datos personales y credenciales seguras.	Alta	Usuario	Definido
RF-02	Permitir el inicio de sesión con autenticación validada en base de datos.	Alta	Usuario	Definido
RF-03	Mostrar en un mapa interactivo los espacios disponibles y ocupados.	Alta	Usuario	Definido
RF-04	Permitir reservar un espacio seleccionando ubicación, fecha y hora.	Alta	Usuario	Definido
RF-05	Confirmar la reserva y generar un comprobante de la transacción.	Alta	Usuario	Definido
RF-06	Simular o realizar el proceso de pago a través de pasarela integrada.	Alta	Usuario	Definido
RF-07	Permitir al usuario visualizar y cancelar reservas activas.	Media	Usuario	Definido

RF-08	Permitir al administrador crear, editar o eliminar espacios de parqueo.	Alta	Administrador	Definido
RF-09	Permitir al administrador modificar tarifas por zona o tiempo.	Media	Administrador	Definido
RF-10	Registrar logs de actividad y errores del sistema.	Media	Sistema	Definido
RF-11	Enviar confirmaciones y alertas al usuario (notificación en interfaz o correo).	Media	Sistema	Definido
RF-12	Generar reportes de ocupación y reservas históricas.	Media	Administrador	Definido
RF-13	Permitir la búsqueda de parqueaderos por zona, nombre o disponibilidad.	Media	Usuario	Definido
RF-14	Permitir la recuperación de contraseña mediante correo electrónico.	Baja	Usuario	Definido

5.2 Requerimientos no funcionales

ID	Descripción	Categoría
RNF-01	El sistema debe estar disponible el 99% del tiempo.	Fiabilidad
RNF-02	El tiempo de respuesta de las operaciones debe ser inferior a 2 segundos.	Rendimiento
RNF-03	La interfaz debe ser responsiva y accesible desde navegadores modernos y dispositivos móviles.	Usabilidad
RNF-04	El sistema debe cumplir con la Ley 1581 de 2012 de protección de datos personales.	Legal
RNF-05	El backend debe garantizar la confidencialidad mediante cifrado de contraseñas.	Seguridad
RNF-06	Los datos deben persistir en una base de datos relacional PostgreSQL.	Persistencia
RNF-07	El código debe ser modular y mantenible siguiendo el principio de separación de capas.	Mantenibilidad
RNF-08	Debe implementarse autenticación basada en JWT para sesiones seguras.	Seguridad
RNF-09	El sistema debe ser escalable a microservicios en futuras versiones.	Escalabilidad
RNF-10	Las pruebas unitarias deben cubrir al menos el 80% de los módulos críticos.	Calidad
RNF-11	El sistema debe permitir despliegue en entornos PaaS (Render, Vercel, Heroku).	Infraestructura
RNF-12	Los errores del sistema deben registrarse en logs accesibles para el administrador.	Auditoría

6 DISEÑO

6.1 Interfaz de usuario

La interfaz está compuesta por tres módulos:

- **Módulo de usuario:** acceso, visualización de mapa, reserva y pago.
- **Módulo administrativo:** control de espacios, reportes y tarifas.
- **Módulo de autenticación:** registro, login y recuperación de contraseña.

El prototipo visual se desarrolló en HTML5, CSS3 y JavaScript para validar la usabilidad antes de la implementación en React.

El diseño es responsivo, con uso de colores neutros, botones destacados y disposición intuitiva.

6.2 Interfaz de hardware o nube (PaaS)

El sistema se desplegará en un entorno **PaaS (Render o Heroku)** con base de datos en la nube **PostgreSQL Cloud**.

El backend se ejecuta en un servidor Node.js y el frontend se distribuye mediante un CDN (GitHub Pages o Netlify).

No se requiere hardware especializado.

6.3 Interfaz de software

- **Frontend:** React con JSX, Hooks y consumo de API REST.
- **Backend:** Node.js con Express, JWT y conexión a PostgreSQL mediante Sequelize.
- **Dependencias externas:**
 - Express (v4.18)
 - PostgreSQL (v14)
 - JWT para autenticación
 - Bcrypt para hashing de contraseñas
- **Sistema Operativo:** Ubuntu Server o Windows.

6.4 Interfaces de comunicación

El sistema usa **HTTP/HTTPS** para las comunicaciones entre cliente y servidor. Se proyecta la futura integración con servicios externos (API de mapas, pasarela de pago, correos). Se utilizarán protocolos seguros (TLS 1.3) para cifrado de datos sensibles.

7 DISEÑO Y ARQUITECTURA

La arquitectura es de **tres capas**:

1. **Presentación (Frontend):** React.
2. **Negocio (Backend):** Node.js + Express.
3. **Datos (Persistencia):** PostgreSQL.

Cada capa está desacoplada y comunica mediante servicios REST. Los componentes incluyen controladores, servicios, modelos y middleware de autenticación.

Se definen los siguientes módulos principales:

- **Módulo de usuarios**
- **Módulo de reservas**
- **Módulo de espacios**
- **Módulo de pagos**
- **Módulo de reportes**

8 SEGURIDAD Y CONECTIVIDAD

8.1 Seguridad

- Autenticación mediante JWT.

- Cifrado de contraseñas con bcrypt.
- Validación de entradas para prevenir inyecciones SQL y XSS.
- Manejo de errores centralizado y registro en logs.

8.2 Conectividad

- Comunicación segura mediante HTTPS.
- Acceso a la base de datos restringido a través de roles.
- Monitoreo de actividad y auditoría de eventos.

9 CRONOGRAMA DE DESARROLLO

Fase	Actividades principales	Duración	Estado
Fase 1	Levantamiento de requerimientos, análisis de usuarios, casos de uso	2 semanas	Finalizada
Fase 2	Diseño UML, arquitectura, prototipo de interfaz	3 semanas	Finalizada
Fase 3	Implementación, pruebas, documentación, video y entrega	4 semanas	En entrega

10 CONCLUSIONES

En primera instancia el alumno debe hacer la contrastación de los objetivos del proyecto y del sistema planteados y alcanzado al final del proyecto.

Se planean conclusiones respecto al ajuste de las herramientas, lenguajes o metodologías utilizadas y la planificación inicial del proyecto.

Para terminar se incluyen conclusiones generales del proyecto desde los puntos de vista:

- Académico
- Personal

11 BIBLIOGRAFÍA

Formato de referencias y bibliografía según los estándares de biblioteca.

12 ANEXO: PLANIFICACION INICIAL DEL PROYECTO

Carta Gantt u otra herramienta de calendarización con las actividades que serán llevadas a cabo en función de la metodología de desarrollo elegida. Considera actividades desarrolladas por los desarrolladores y los usuarios o clientes.

En caso de metodologías incrementales o evolutivas, se debe especificar la funcionalidad que será abordada en cada iteración o incremento

12.1.1 Estimación inicial de tamaño

Estimación de Tamaño del software aplicando técnicas basadas en PF o Casos de Uso.

Consulte los valores de la industria utilizados para cuantificar el tiempo (horas de esfuerzo de desarrollo) necesario para implementar 1 Punto de Caso de Uso o 1 Punto de función.

12.1.2 Contabilización final del tamaño del Sw

- Contabilizar la cantidad de líneas de código implementadas en su software. Especifique junto a los valores la forma como fue calculado el valor (si se consideran líneas en blanco, comentarios o todas las líneas, si se consideran funciones o componentes reutilizados)
- Estimar equivalencia de PF o PCU estimados inicialmente con Líneas de código contabilizadas al final.
- Estimar el esfuerzo horas hombre, considerando las horas de trabajo reales dedicadas al proyecto por cada fase del desarrollo.

13 ANEXO: RESULTADOS DE ITERACIONES EN EL DESARROLLO

Si se ha seguido un método/modelo de desarrollo iterativo y/o incremental es necesario adjuntar los hallazgos o resultados obtenidos de las iteraciones.

Es decir se debe indicar el contenido o funcionalidad de la iteración o incremento y los comentarios, correcciones u observaciones de los usuarios.

14 ANEXO: MANUAL DE USUARIO

Según se requiera.

15 ANEXO: ESPECIFICACION DE LAS PRUEBAS

15.1 Pruebas de Unidad

El siguiente ítem se repite para cada unidad o módulo independiente.

15.1.1 <nombre unidad>

En este ítem se especifican:

- Las configuración Hw, Sw, SO o de comunicaciones que son necesarias para la prueba.
- Pre condiciones de las pruebas. Por ejemplo en la prueba del módulo de “registra venta” se requiere que existan productos con stock disponible ingresados a la BD.

ID Caso De Prueba	Características Probar	a	Datos de Entrada								Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
			D1											

15.2 Sistema

En este ítem se especifican:

- Condiciones de la prueba. Esta prueba debe ser ejecutada en un ambiente lo más parecido al que utiliza el usuario.
- Las configuración Hw, Sw, SO, comunicaciones que son necesarias para la prueba.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		D1								Éxito / Fracaso	Criticidad en caso Fracaso

15.3 Aceptación

Prueba alfa realizada junto al usuario, prueba beta realizada por el usuario sin asistencia del desarrollador.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		D1								Éxito / Fracaso	Criticidad en caso Fracaso

16 ANEXO: DICCIONARIO DE DATOS DEL MODELO DE DATOS

El diccionario completo se incluye como anexo no obstante las tablas principales son descritas en este punto.