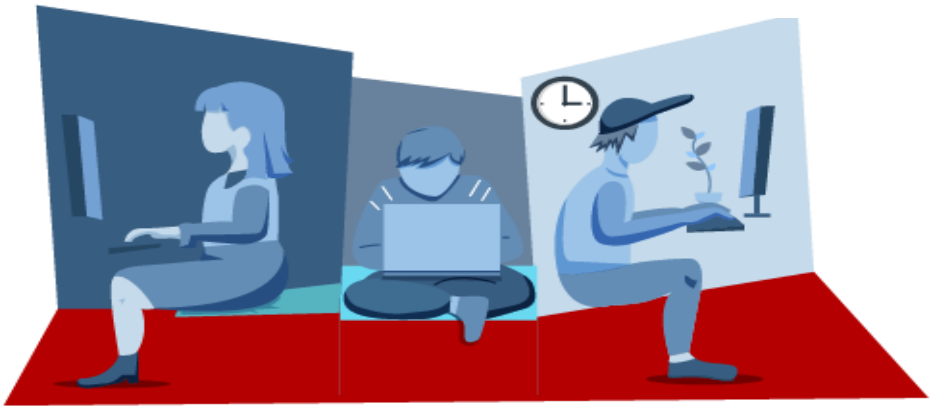
PARKGO

Sistema Web Centralizado para la Gestión de arriendos de Estacionamientos en Concepción.



MÓDULO: **Proyecto de especialidad**

Docente: Víctor Valderrama M.

Estudiante: Julieth A. Timaure Chirinos

Carrera: Técnico en Programación y Análisis de Sistemas

Fecha: 17-12-2025

INDICE

Tabla de contenido

INTRODUCCION

El parque automotor de Chile, ha presentado un incremento que se ha sostenido, logrando convertirse en un factor importante para la planificación urbana y la calidad de vida. Según datos del Instituto Nacional de Estadísticas (INE), el total de permisos de circulación en el país superó los 6.5 millones en 2023, marcando un alza de 4.6% respecto al año anterior.

Esta tendencia nacional se refleja con especial intensidad en la Región del Biobío, la cual se posiciona como la tercera con mayor cantidad de vehículos del país, superando los 550,000 permisos y concentrando más del 8% del parque automotriz nacional. Bajo este contexto surge una nueva necesidad entre los usuarios con automóviles, la búsqueda de un estacionamiento sea cerca de su hogar, universidad o trabajo. Actualmente los métodos tradicionales carecen de eficiencia, gracias a este análisis surge como propuesta de proyecto “ParkGo”, centralizando tecnológicamente esta demanda de estacionamientos.

Este informe detalla el ciclo completo de ingeniería de software aplicado para la creación de una plataforma web transaccional, priorizando la escalabilidad y la seguridad de los datos. A lo largo del documento, se evidenciará cómo se ha pasado de la definición de requerimientos y modelado de base de datos relacional, hasta la implementación de una API RESTful capaz de gestionar autenticación segura, subida de archivos multimedia en la nube y transacciones atómicas. Asimismo, se expone la metodología de trabajo ágil utilizada (Kanban y filosofía Lean), diseñada para maximizar la entrega de valor y asegurar la calidad del producto final mediante pruebas exhaustivas de integridad referencial y lógica de negocio.

1. Problemática del proyecto

En la región del Biobío, específicamente en la transcurrida ciudad de Concepción se presenta el caso de una **demanda de estacionamientos que supera la oferta fácilmente accesible**, se ha presentado debido a un mayor parque automotor. Esta situación actual afecta a un numeroso grupo de personas que van desde profesionales y estudiantes, hasta residentes que no poseen estacionamiento y buscan alguno cerca de su trabajo, universidad u hogar, y en el otro extremo se encuentran las personas que poseen espacios grandes o empresas con espacio sin uso que necesitan un canal de difusión de la oferta.

En la actualidad el proceso que se debe llevar a cabo para encontrar un estacionamiento está basado en métodos ineficientes. La persona interesada en el espacio debe recurrir a buscar carteles físicos o publicaciones en redes sociales, teniendo una gran pérdida de tiempo ya que en muchas ocasiones la publicación no contiene la información concreta del espacio.

Este problema presenta una consecuencia directa tanto en conductores como en arrendatarios, los cuales **pierden mucho tiempo y son poco fiables los métodos de búsqueda y verificación**. Por lo tanto, se hace necesaria una **solución tecnológica que centralice la oferta y demanda en un solo lugar**, que logre la optimización de la búsqueda y entregue datos certeros mediante filtros de búsqueda y que a su vez logre brindar la confianza entre ambas partes para facilitar el arriendo y pagos de forma segura.

2. Objetivo general

Optimizar el proceso de arriendos de estacionamientos fijos y a largo plazo en la comuna de Concepción. Buscando realizar una mejora en la eficiencia de los procesos de búsqueda para conductores y lograr un mayor alcance de oportunidad económica para los propietarios de un espacio, garantizando a su vez un espacio seguro para ambas partes de la transacción con la implementación de un sistema web centralizado que permita la optimización del proceso.

3. Objetivos específicos

Objetivos SMART: Desarrollar una plataforma web transaccional ("Park Go") que digitalice el proceso de arriendo de estacionamientos privados en la región del Biobío, logrando gestionar el ciclo completo de publicación, búsqueda, reserva y calificación durante el segundo semestre académico de 2025.

1. (S/M - Específico y Medible): Implementar una arquitectura de backend RESTful que soporte la gestión de usuarios, vehículos, publicaciones y arriendos, asegurando tiempos de respuesta eficientes y la integridad de las transacciones en el 100% de las operaciones simuladas.

2. (A - Alcanzable): Desarrollar un módulo de validación de identidad y control vehicular que verifique automáticamente el formato nacional de identificación (RUT) y garantice la unicidad de las patentes registradas en el sistema.

3. (R - Relevante): Integrar servicios de geolocalización interactiva que permitan visualizar la oferta de estacionamientos en un mapa digital, facilitando la ubicación exacta de los espacios disponibles para los usuarios.

4. (T - Temporal): Desplegar el prototipo funcional en un entorno de nube, incluyendo la base de datos y el almacenamiento de archivos multimedia, asegurando su disponibilidad operativa antes de la fecha de entrega final del proyecto.

4. Innovación

Desarrollar un sistema web que tendrá como principal factor una innovación funcional, que contenga filtros estructurados en conjunto de campos de información obligatorios, logrando que el 100% de las publicaciones realizadas cuenten con todo lo necesario para la toma de decisiones, reduciendo el tiempo de preguntas y respuestas realizadas entre conductor y arrendatario para lograr tomar una decisión. Contando con una ventaja sobresaliente de implementar el contacto directo entre arrendatario y conductor, sin la necesidad de intermediarios que demora el proceso.

Si bien existen distintas aplicaciones y redes sociales “ParkGo” introduce la verticalidad en el mercado, estandarizando los datos, obligando a declarar datos esenciales para la toma de decisiones. Presentando un componente de innovación social al permitir que pequeños

propietarios moneticen activos en desuso, con herramientas tecnológicas que antes solo estaban a disposición de grandes empresas.

Como valor agregado al proyecto tiene la inclusión de un contrato digital automático (visible en el módulo de checkout) que formaliza la relación entre partes, y un sistema de "Wizard" (paso a paso) que simplifica la publicación para usuarios no técnicos.

Este proyecto será realizado bajo la Licencia MIT, teniendo como propósito principal la posibilidad del aprendizaje a partir del código libre, la cual solo tendrá el requerimiento de dar aviso de auditoria sin necesidad de imponer ninguna restricción a futuros usuarios y colaboradores.

La catedral y el Bazar

Mediante el análisis realizado se determina que este proyecto nace reconociéndose como un Bazar, ya que se prioriza la entrega rápida de las funcionalidades mínimas y esenciales para que funcione el sistema, y así lograr solucionar el problema real que presenta el usuario. Su posible éxito y continua evolución depende de la retroalimentación constante por parte de la comunidad de conductores y arrendatarios.

5. Metodología de desarrollo

Para lograr implementar la estructura del proyecto estilo “Bazar”, la metodología utilizada para el desarrollo de este proyecto se basa en la mezcla de dos tecnologías clave, del Lean Development que es la filosofía escogida para entregar un valor masificado al cliente, que elimine cualquier desperdicio, mientras que el método para llevarla a cabo es Kanban, poniendo la estrategia de no solo mover las tarjetas sino saber las que aportaran mayor valor al cliente. **Otorgando flexibilidad, eficiencia y agilidad al proyecto para entregar valor al MVP (Producto Mínimo Viable),** implementando una estrategia de versionamiento (Git y GitHub) para asegurar la trazabilidad del desarrollo y evitar la pérdida de avances.

Para el presente proyecto existen múltiples desperdicios que deben ser eliminados para entregar con mayor rapidez un MVP (producto mínimo viable) con la ayuda de Kanban, como la aplicación de filtros sencillos. A su vez, aporta un **rápido aprendizaje de las posibles dificultades que se presentan y como ciertas tareas pueden demorar más, y el tablero de Kanban ayudara a priorizar la tarea de más rápido desarrollo.**

6. Tecnologías y arquitectura del proyecto

El sistema utiliza una arquitectura MVC (Modelo-Vista-Controlador) con las siguientes tecnologías:

- Frontend: HTML5 semántico, CSS3 modular y JavaScript Vanilla (ES6+) para la lógica del cliente.
- Backend: API RESTful construida con Express.js.
- Base de Datos: Relacional PostgreSQL.

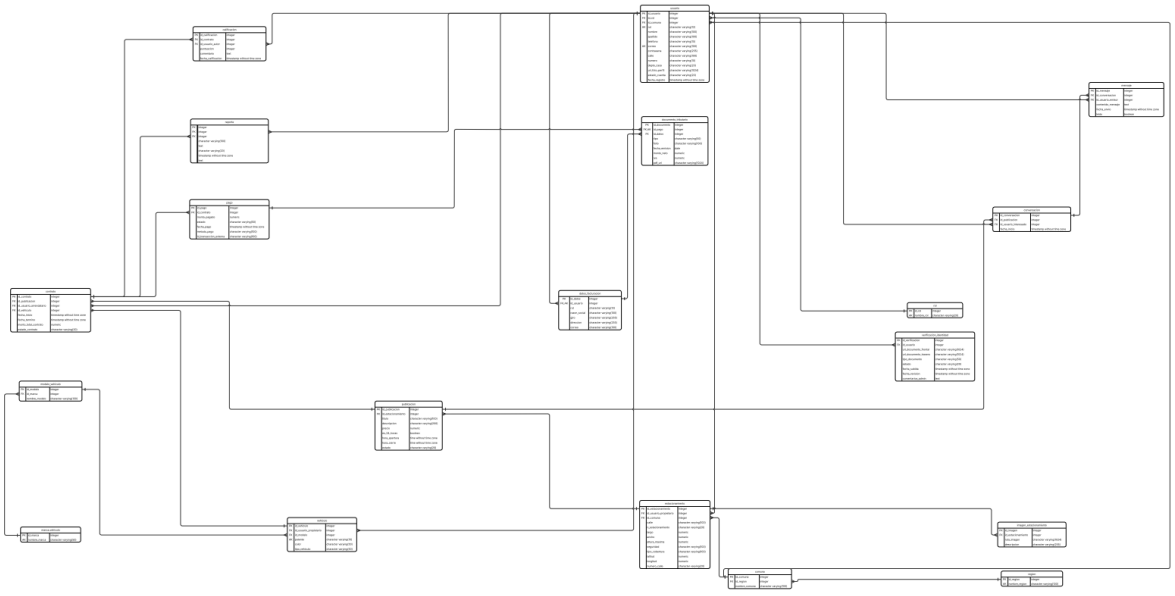
6.1. Modelo de Base de Datos (Normalización):

El esquema de base de datos (Parkgo/sql) cumple con la Tercera Forma Normal (3FN) para evitar redundancia:

- Normalización Geográfica: Tablas separadas para región y comuna, vinculadas por IDs.

- Normalización vehicular: Tablas maestras para marca_vehiculo y modelo_vehiculo.

- Integridad referencial: Uso estricto de Claves Foráneas (FK) entre publicación, estacionamiento, usuario y contrato.



6.2. Estructura del Código:

El proyecto sigue una estructura de directorios estándar profesional:

```
Parkgo/
├── Backend/
│   ├── controllers/   # Lógica de negocio (auth, parking, vehicle...)
│   ├── middlewares/   # Seguridad (authMiddleware) y Carga (uploadMiddleware)
│   ├── routes/        # Definición de endpoints
│   ├── db.js          # Pool de conexión PostgreSQL
│   └── cloudinary.js  # Configuración de servicio de medios
└── Frontend/
    ├── css/           # Hojas de estilo modulares
    ├── js/            # Lógica de cliente separada por módulos
    └── *.html         # Vistas principales
```

7. Historias de usuarios

HU-01: Registro de Nuevo Usuario (Asociada a: RF-01, RF-02, RF-03)

Usuario: Como Conductor o Propietario quiero registrarme en la plataforma ingresando mis datos personales y credenciales, para poder acceder a las funcionalidades de arriendo y publicación.

Criterios de Aceptación:

1. El formulario debe solicitar RUT, Nombre, Apellido, Correo, Teléfono y Contraseña.
2. El RUT debe validarse en tiempo real (Módulo 11) y formatearse automáticamente.
3. El sistema debe bloquear el registro si el RUT o correo ya existen (Error 409).
4. La contraseña debe almacenarse encriptada.

HU-02: Inicio de Sesión Seguro (Asociada a: RF-04)

Usuario: Como Usuario Registrado, quiero iniciar sesión con mi correo y contraseña, para acceder a mi panel de control privado de forma segura.

Criterios de Aceptación:

1. Autenticación de credenciales contra la base de datos PostgreSQL.

2. Generación de Token JWT con vigencia de 24 horas tras login exitoso.

3. Bloqueo de acceso y alerta visual si el usuario tiene estado "SUSPENDIDO".

HU-03: Gestión de Perfil y Foto (Asociada a: RF-05, RF-06)

Usuario: Como Usuario Autenticado, quiero actualizar mis datos de contacto y subir una foto de perfil, para mantener mi información al día y generar confianza.

Criterios de Aceptación:

1. Edición habilitada para nombre, apellido, teléfono y dirección.

2. Carga de imagen procesada vía Cloudinary (redimensión a 500x500px).

3. Actualización inmediata del avatar en la interfaz tras la subida.

HU-04: Registro de Vehículo (Asociada a: RF-07, RF-08)

Usuario: Como Conductor, quiero agregar los datos de mi vehículo (patente, marca, modelo, color) a mi cuenta, para seleccionarlo rápidamente al arrendar.

Criterios de Aceptación:

1. La patente debe guardarse normalizada (mayúsculas, sin guiones).

2. Carga dinámica de modelos basada en la marca seleccionada.

3. Validación de patente única en el sistema para evitar duplicados.

HU-05: Visualización y Gestión de Flota (Asociada a: RF-09)

Usuario: Como Conductor, quiero ver mis vehículos registrados y eliminarlos si es necesario, para mantener mi lista de transporte organizada.

Criterios de Aceptación:

1. Visualización de vehículos en tarjetas con diseño de patente chilena.

2. Eliminación permitida solo si el vehículo pertenece al usuario (validación de ID propietario).

HU-06: Publicación de Estacionamiento (Asociada a: RF-10, RF-11, RF-12)

Usuario: Como Propietario, quiero publicar mi estacionamiento mediante un asistente paso a paso, para cargar la información de horarios y características sin errores.

Criterios de Aceptación:

1. Flujo dividido en 3 pasos: Detalles, Galería y Ubicación.

2. Selección de disponibilidad (24/7 o rango horario).

3. Selección múltiple de atributos de seguridad (cámaras, portón, etc.).

HU-07: Geolocalización del Estacionamiento (Asociada a: RF-14)

Usuario: Como Propietario, quiero ubicar mi estacionamiento en un mapa interactivo, para que los conductores conozcan la ubicación exacta.

Criterios de Aceptación:

1. Integración de mapa (Leaflet/Mapbox) con búsqueda por dirección.

2. Captura automática de coordenadas (latitud/longitud) al arrastrar el marcador para guardarlas en la base de datos.

HU-08: Gestión de Galería de Fotos (Asociada a: RF-13)

Usuario: Como Propietario, quiero subir fotos de mi estacionamiento, para mostrar el estado real del lugar a los interesados.

Criterios de Aceptación: 1. Carga simultánea de hasta 5 imágenes.

2. Procesamiento de imágenes con Multer y almacenamiento en Cloudinary antes de guardar referencias en la base de datos.

HU-09: Búsqueda Inteligente (Asociada a: RF-15, RF-16)

Usuario: Como Conductor, quiero buscar estacionamientos filtrando por ubicación, precio y tipo, para encontrar una opción que se ajuste a mis necesidades.

Criterios de Aceptación:

1. Filtros activos por texto, rango de precios y tipo de cobertura.

2. Actualización dinámica de resultados. 3. Listado exclusivo de publicaciones con estado "Disponible".

HU-10: Arriendo y Contrato (Asociada a: RF-18, RF-19, RF-20, RF-21)

Usuario: Como Conductor, quiero reservar un espacio seleccionando duración y vehículo, para asegurar el arriendo formalmente.

Criterios de Aceptación:

1. Cálculo automático de costo con descuentos por tramo (1, 3, 6, 12 meses).

2. Visualización y aceptación obligatoria de Contrato Digital antes del pago.

3. Transacción atómica en BD: creación de contrato, registro de pago y cambio de estado de publicación a "No Disponible".

HU-11: Administración de Usuarios (Asociada a: RF-23)

Usuario: Como Administrador, quiero gestionar el estado de los usuarios (Activar/Banear), para moderar la plataforma y bloquear cuentas indebidas.

Criterios de Aceptación:

1. Acceso restringido a Rol 3 (Admin).

2. Funcionalidad de cambio de estado (ACTIVO/SUSPENDIDO) en listado general.

3. Bloqueo de auto-baneo para el propio administrador.

HU-12: Calificación del Servicio (Asociada a: RF-24)

Usuario: Como Arrendatario, quiero calificar con estrellas y comentarios mi experiencia, para informar a futuros usuarios sobre la calidad del servicio.

Criterios de Aceptación:

1. Habilitación de calificación solo tras finalizar contrato.

2. Validación de que el usuario sea parte del contrato.

3. Cálculo y visualización del promedio de estrellas en el perfil del estacionamiento.

8. Requerimientos Funcionales (RF)

8.1. Módulo 1: Gestión de Usuarios y Autenticación

- RF-01 Registro de Usuarios: El sistema debe permitir el registro mediante un formulario que solicite: RUT, Nombre, Apellido, Correo, Teléfono, Contraseña y Dirección (Región, Comuna, Calle, Número).

- RF-02 Validación de RUT: El sistema debe validar en tiempo real que el RUT ingresado cumpla con el algoritmo del Módulo 11 y formatearlo automáticamente con puntos y guion.

- RF-03 Unicidad de Credenciales: El sistema debe impedir el registro de dos cuentas con el mismo RUT o Correo Electrónico, informando el error al usuario.

- RF-04 Inicio de Sesión Seguro: El sistema debe autenticar usuarios mediante correo y contraseña, utilizando encriptación (bcrypt) para comparar credenciales y generando un token de sesión (JWT) con duración de 24 horas.

- RF-05 Gestión de Perfil: El usuario debe poder modificar sus datos personales (excepto RUT y Correo) y actualizar su contraseña.

- RF-06 Foto de Perfil: El sistema debe permitir subir, recortar y almacenar una foto de perfil en la nube (Cloudinary), actualizando la referencia en la base de datos.

8.2. Módulo 2: Gestión de Vehículos (Flota)

- RF-07 Registro de Vehículos: El usuario debe poder registrar múltiples vehículos ingresando: Patente, Marca (seleccionable), Modelo (dependiente de la marca), Tipo (Sedán, SUV, Moto, etc.) y Color.

- RF-08 Normalización de Patente: El sistema debe almacenar las patentes en mayúsculas y sin caracteres especiales, validando que no exista la misma patente registrada previamente en el sistema.

- RF-09 Listado y Eliminación: El usuario debe visualizar sus vehículos en tarjetas con el formato visual de patente chilena y poder eliminarlos si no están asociados a un contrato activo.

8.3. Módulo 3: Publicación de Estacionamientos (Wizard)

- RF-10 Flujo de Publicación por Pasos: El sistema debe guiar al usuario mediante un asistente (Wizard) de 3 pasos: (1) Información Básica y Características, (2) Galería de Fotos, (3) Ubicación Geoespacial.

- RF-11 Gestión de Horarios: El usuario debe poder definir si el estacionamiento es "24/7" o especificar un rango horario (Apertura/Cierre).

- RF-12 Características del Espacio: El sistema debe permitir seleccionar atributos como: Techado/Aire Libre, Seguridad (Cámaras, Portón, Conserje) y Dimensiones (Largo, Ancho, Altura).

- RF-13 Galería de Imágenes: El usuario debe poder subir hasta 5 fotografías, con pre visualización y capacidad de eliminar antes de guardar. El sistema debe validar que se suba al menos una foto.

- RF-14 Geolocalización Exacta: El sistema debe integrar un mapa interactivo donde el usuario pueda buscar su dirección (geo codificación) y arrastrar un marcador para definir la latitud y longitud precisa del estacionamiento.

8.4. Módulo 4: Búsqueda y Catálogo

- RF-15 Filtros Avanzados: El sistema debe permitir buscar estacionamientos filtrando por: Texto (Comuna/Calle), Rango de Precios (Mín./Máx.), Tipo de Cobertura y Ordenamiento (Menor/Mayor Precio).

- RF-16 Visualización de Resultados: Los resultados deben mostrarse en tarjetas con: Foto principal, Título, Precio formateado (CLP), Ubicación y Características destacadas.

- RF-17 Detalle de Publicación: Al seleccionar un estacionamiento, el sistema debe mostrar: Carrusel de imágenes, Mapa de ubicación (radio aproximado), Descripción completa, Información del dueño y Opiniones de otros usuarios.

8.5. Módulo 5: Proceso de Arriendo y Contrato

- RF-18 Simulación de Costos: El sistema debe calcular el total a pagar según la duración seleccionada (1, 3, 6, 12 meses), aplicando descuentos automáticos definidos por reglas de negocio (5%, 10%, 15%).

- RF-19 Selección de Vehículo: El usuario debe seleccionar obligatoriamente uno de sus vehículos registrados para asociarlo al contrato.

- RF-20 Contrato Digital: El sistema debe generar y mostrar un contrato de arrendamiento digital con los datos de las partes y fecha actual, exigiendo su aceptación explícita antes del pago.

- RF-21 Transacción de Arriendo: Al confirmar el pago (simulado), el sistema debe realizar una transacción atómica en base de datos que: (1) Cree el contrato, (2) Registre el pago, (3) Cambie el estado de la publicación a "No Disponible" para evitar doble arriendo.

8.6. Módulo 6: Administración y Feedback

- RF-22 Dashboard de Admin: El administrador debe visualizar estadísticas clave: Total Usuarios, Total Estacionamientos y Arriendos Activos.

- RF-23 Moderación de Usuarios: El administrador debe poder listar a todos los usuarios y cambiar su estado (Activar/Banear), impidiendo el acceso a usuarios suspendidos.

- RF-24 Sistema de Calificaciones: Al finalizar un contrato, el arrendatario debe poder calificar su experiencia (1 a 5 estrellas) y dejar un comentario. El sistema debe validar que el usuario efectivamente arrendó ese espacio antes de permitir la calificación.

9. Requerimientos No Funcionales (RNF)

- RNF-01 Seguridad de Datos: Las contraseñas no deben guardarse en texto plano bajo ninguna circunstancia.

- RNF-02 Escalabilidad de Imágenes: Las imágenes deben ser redimensionadas y optimizadas automáticamente al subirse para no exceder 1000px de ancho, ahorrando ancho de banda.

- RNF-03 Persistencia Geográfica: El sistema debe utilizar tipos de datos decimales de alta precisión para almacenar coordenadas (latitud/longitud) y garantizar la exactitud en el mapa.

- RNF-04 Disponibilidad: La base de datos debe estar alojada en un servicio de alta disponibilidad (Neon.tech) con conexiones seguras SSL.

Reglas de Negocio:

- RN-01: Un usuario no puede arrendar su propio estacionamiento.

- RN-02: Una publicación arrendada no debe aparecer en los resultados de búsqueda.

- RN-03: No se pueden registrar vehículos con patentes duplicadas en todo el sistema.

- RN-04: Un usuario baneado no puede iniciar sesión ni realizar transacciones.

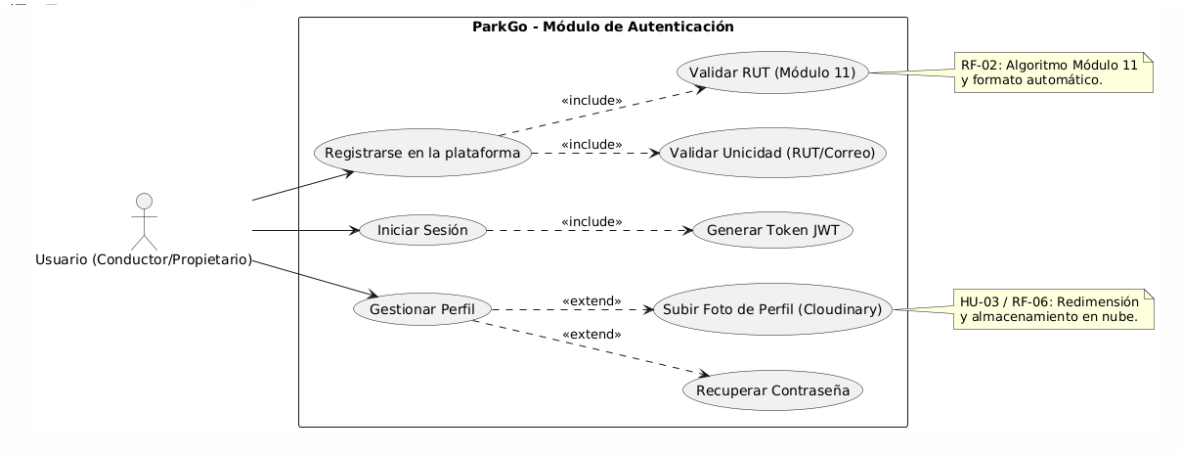
10. Matriz de trazabilidad

ID RF	Descripción del Requerimiento	Módulo del Sistema	Caso de Uso Asociado (CU)
RF-01	Registro de usuarios con validación de RUT y datos personales.	Gestión de Usuarios (Auth)	CU-01: Registrar Nuevo Usuario
RF-03	Control de unicidad de cuentas (RUT y Correo no duplicados).	Gestión de Usuarios (Auth)	CU-01: Registrar Nuevo Usuario
RF-04	Inicio de sesión seguro con encriptación y Token JWT.	Gestión de Usuarios (Auth)	CU-02: Iniciar Sesión
RF-05	Edición de perfil de usuario y actualización de contraseña.	Gestión de Usuarios (Perfil)	CU-03: Gestionar Perfil
RF-07	Registro de vehículos con validación de patente única.	Gestión de Vehículos	CU-04: Registrar Vehículo
RF-09	Visualización y eliminación de vehículos propios.	Gestión de Vehículos	CU-05: Gestionar Flota
RF-10	Publicación de estacionamiento mediante asistente (Wizard).	Gestión de Parkings	CU-06: Publicar Estacionamiento
RF-13	Carga y gestión de galería de imágenes (Cloudinary).	Gestión de Parkings	CU-08: Subir Fotos
RF-14	Geolocalización exacta mediante mapa interactivo.	Gestión de Parkings	CU-07: Geolocalizar Espacio
RF-15	Búsqueda de estacionamientos con filtros (precio, ubicación).	Búsqueda y Arriendo	CU-09: Buscar Estacionamiento
RF-18	Simulación de costos y aplicación de descuentos por tiempo.	Búsqueda y Arriendo	CU-10: Arrendar Espacio
RF-20	Generación y aceptación de contrato digital.	Búsqueda y Arriendo	CU-10: Arrendar Espacio

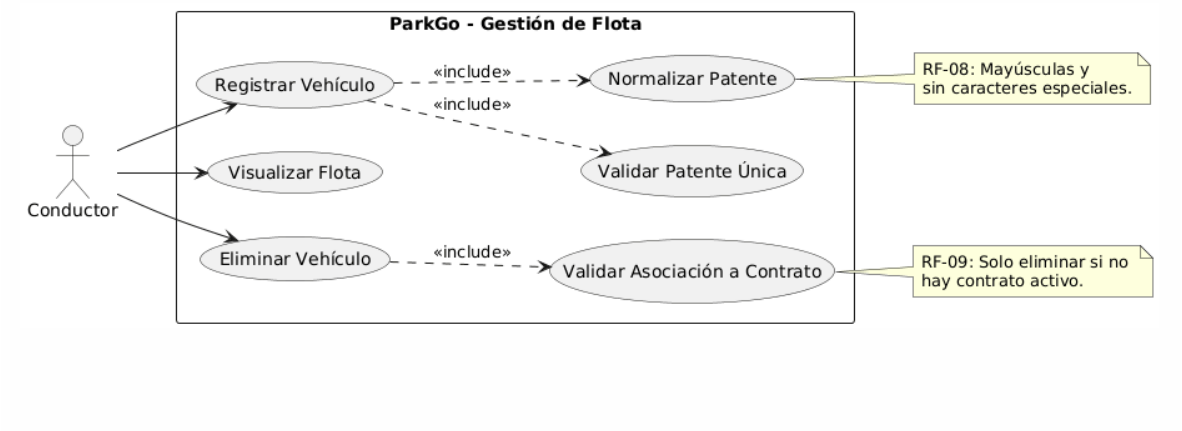
RF-21	Transacción de arriendo y bloqueo de disponibilidad.	Búsqueda y Arriendo	CU-10: Arrendar Espacio
RF-23	Moderación de usuarios (Banear/Activar) por Administrador.	Administración	CU-11: Administrar Usuarios
RF-24	Calificación del servicio (Estrellas y comentarios).	Feedback	CU-12: Calificar Servicio

11. Casos de uso del sistema

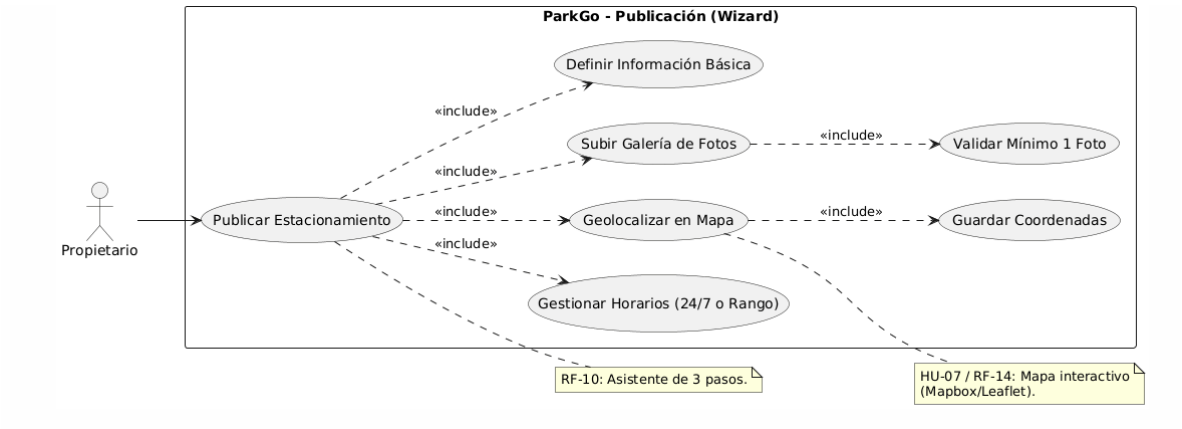
1. Diagrama de casos de uso “Gestión de Usuarios y Seguridad”: Este diagrama cubre el registro, inicio de sesión y gestión de perfil, incluyendo las validaciones de RUT e identidad descritas en los RF-01 al RF-06 y las HU-01 a HU-03.



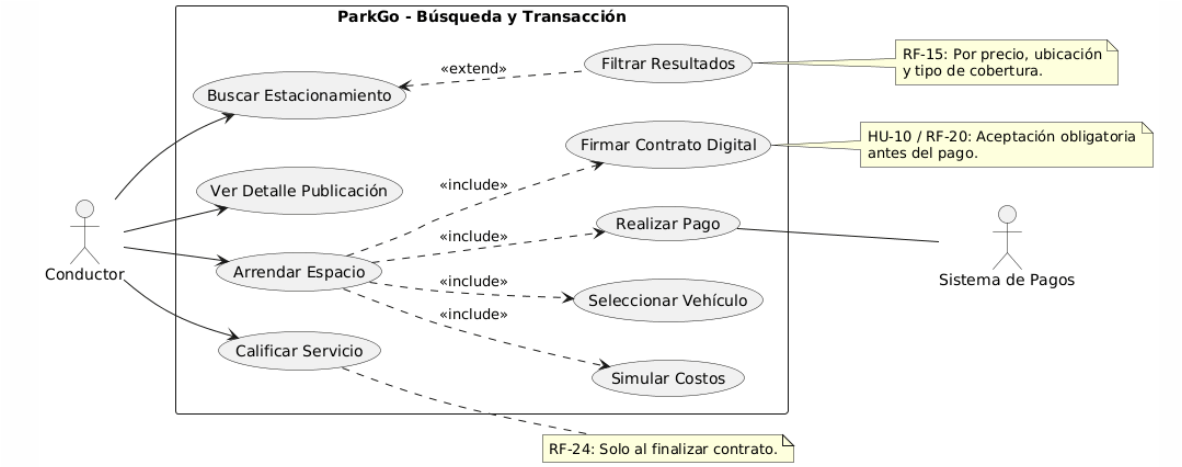
2. Diagrama de casos de uso “Gestión de Vehículos (Conductor)”: Basado en el Módulo 2, enfocado en las HU-04 y HU-05 y los requerimientos RF-07 al RF-09. Se destaca la normalización de la patente.



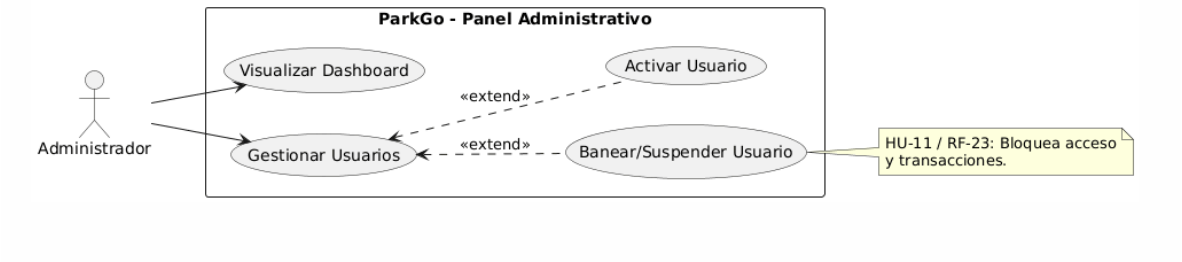
3. Diagrama de casos de uso “Publicación de Estacionamientos (Propietario)”: Este flujo representa el "Wizard" de 3 pasos descrito en la HU-06 y los RF-10 al RF-14. Incluye la geolocalización y la galería de imágenes.



4. Diagrama de casos de uso “Búsqueda y Arriendo (Core)”: Este es el proceso central del negocio. Abarca la búsqueda (HU-09), el arriendo con contrato digital (HU-10) y el feedback (HU-12). Referencia los RF-15 al RF-21 y RF-24.

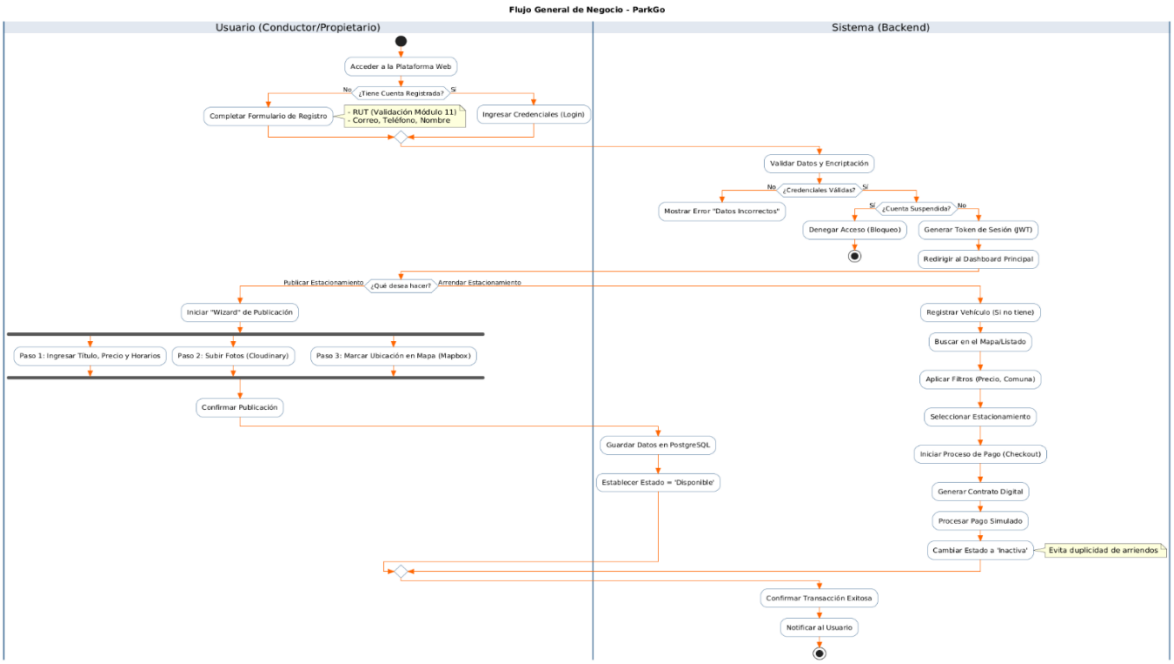


5. Diagrama de casos de uso “Administración”: Cubre el rol del Administrador para la gestión de usuarios según la HU-11 y RF-23.

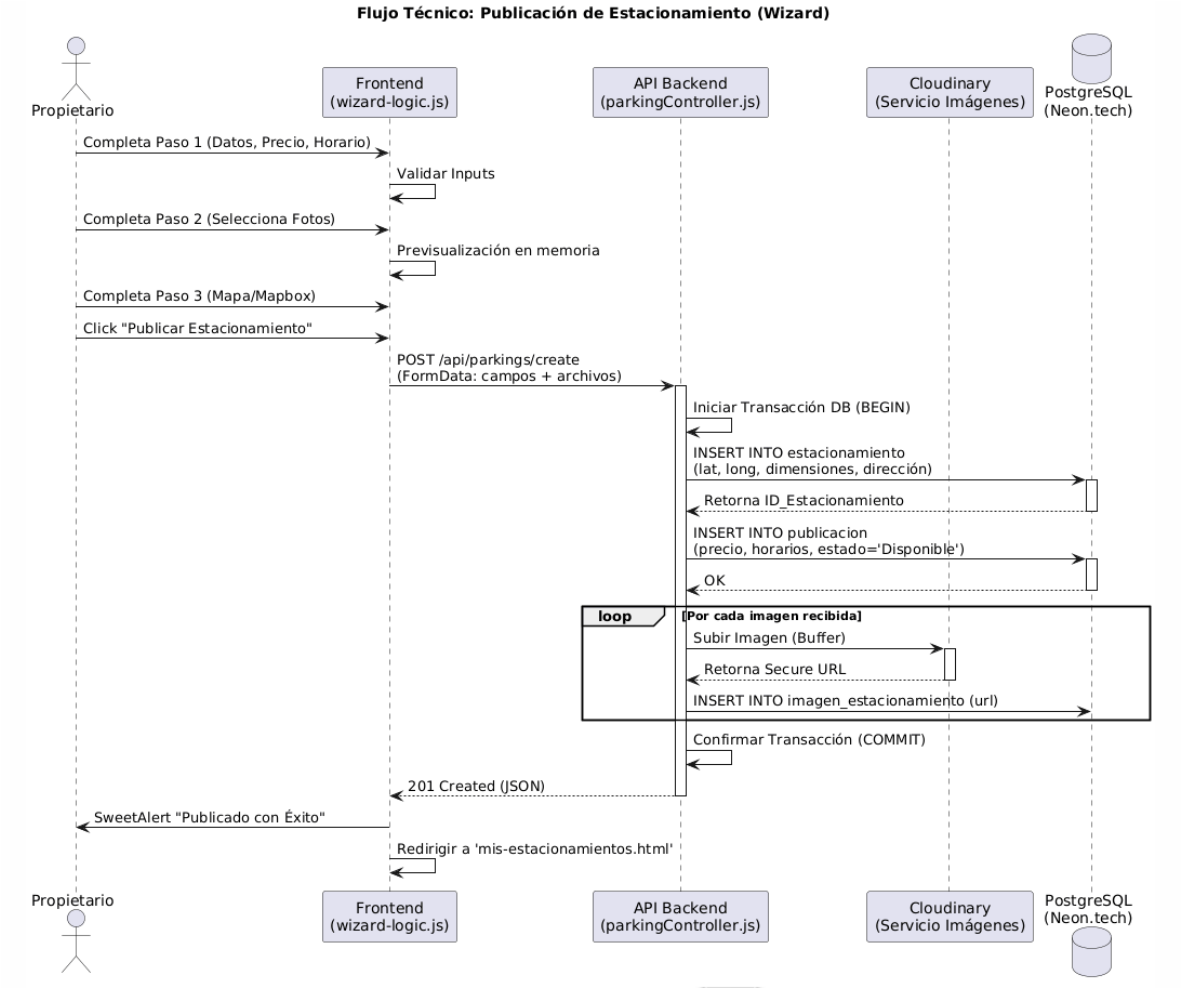


12.Diagramas de flujo general de la aplicación “ParkGo”

1. Diagrama de actividad “Flujo general”: Comienza con el inicio con dos caminos posibles, el “Inicio de sesión” y el “Registro” del usuario, se aplican las validaciones de unicidad de correo y rut, y si el usuario no está bloqueado. Se encuentra una bifurcación de roles, la del usuario que publica y el usuario que arrienda con sus respectivas validaciones y funciones.

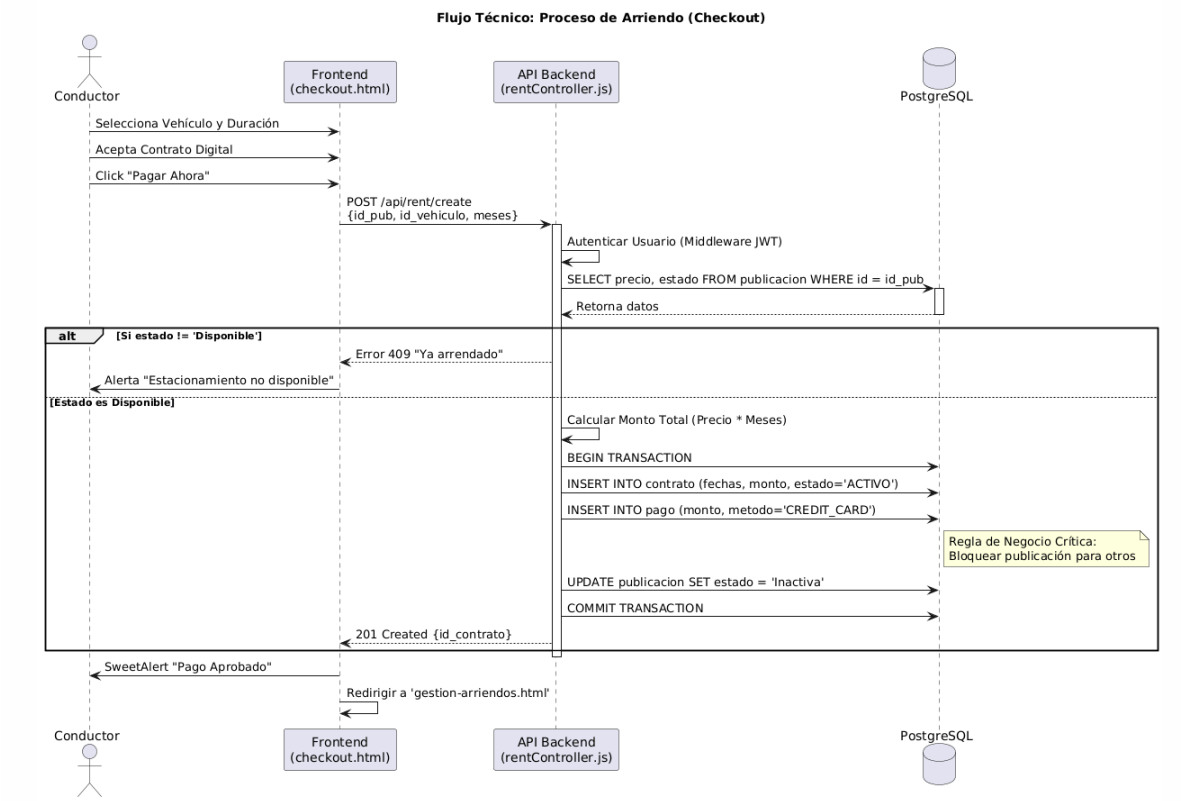


2. Diagrama de secuencia “Publicación de estacionamiento (Wizard”): Detalla técnicamente el proceso de crear un estacionamiento, con base en los archivos **wizard-logic.js** y **parkingController.js.** Muestra la interacción del frontend, el envío de datos a la API y su posterior transacción en la base de datos con el guardado de dirección y dimensiones, paralelo la API envía las imágenes a Cloudinary y esta envía un link seguro. La API guarda esas url en la base de datos.

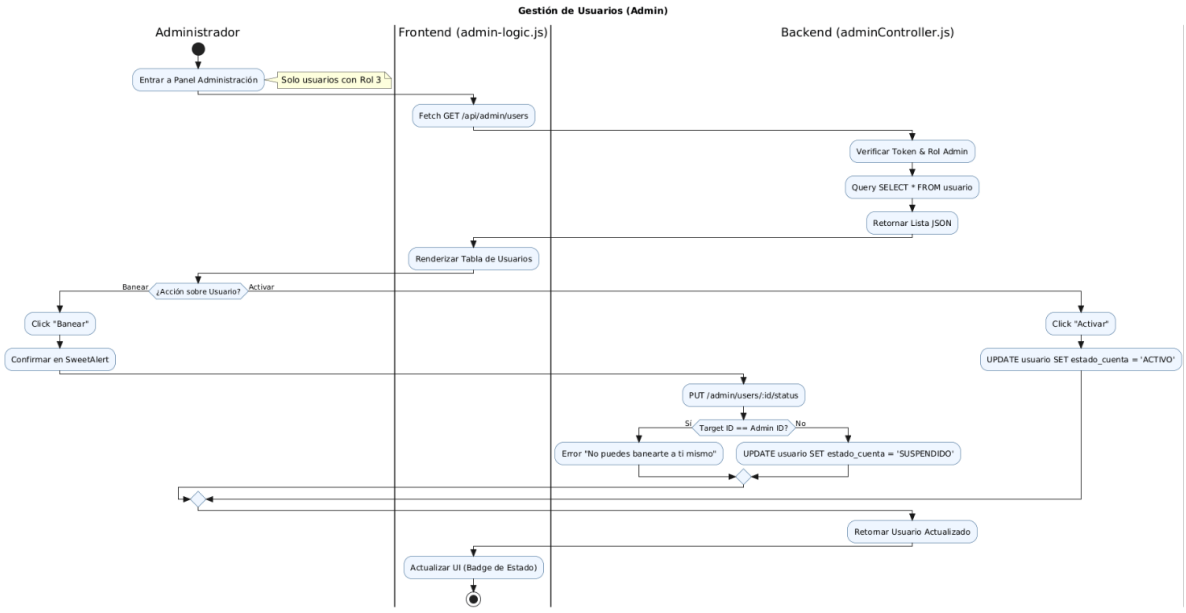


3. Diagrama de secuencia “Proceso de arriendo” (Checkout): Explica el intercambio de dinero y derechos de uso basado en **rentController.js.** El conductor elige que auto usara y el tiempo de arriendo del estacionamiento. Luego ocurre la validación critica, la API revisa que el estacionamiento aun esta disponible para evitar duplicidad de arriendo. Si esta libre la API realiza tres procesos simultaneos:

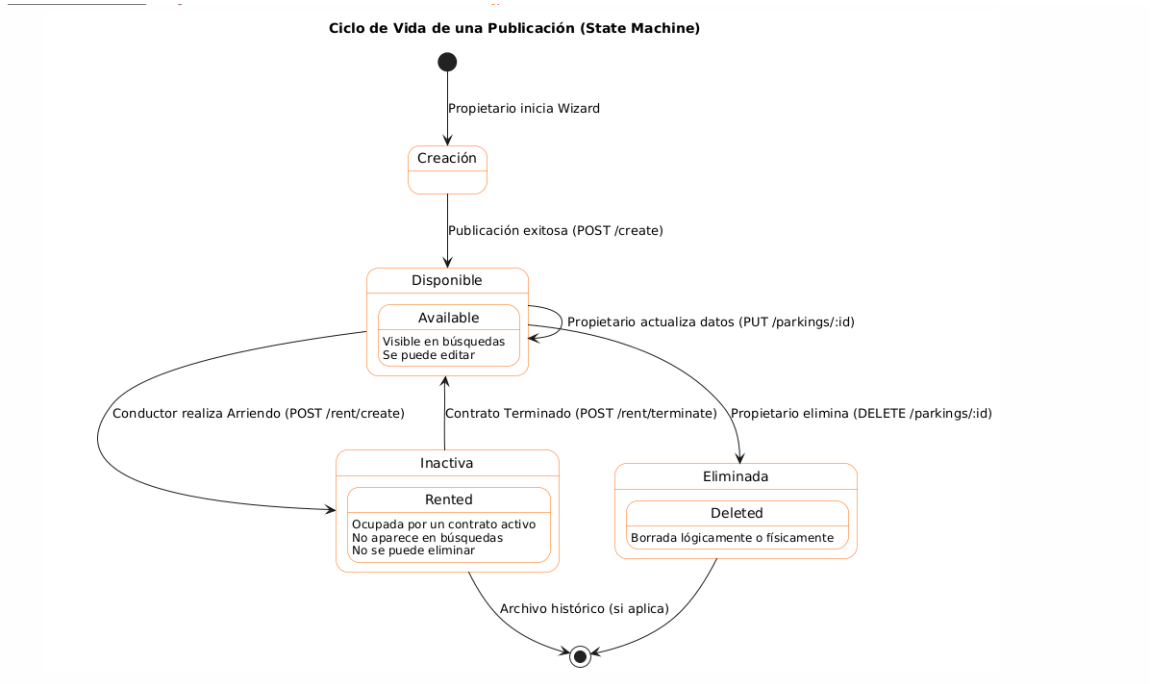
- Crea el contrato legal.
- Registra el pago (simulado).
- Cambia el estado de la publicación a “inactiva.



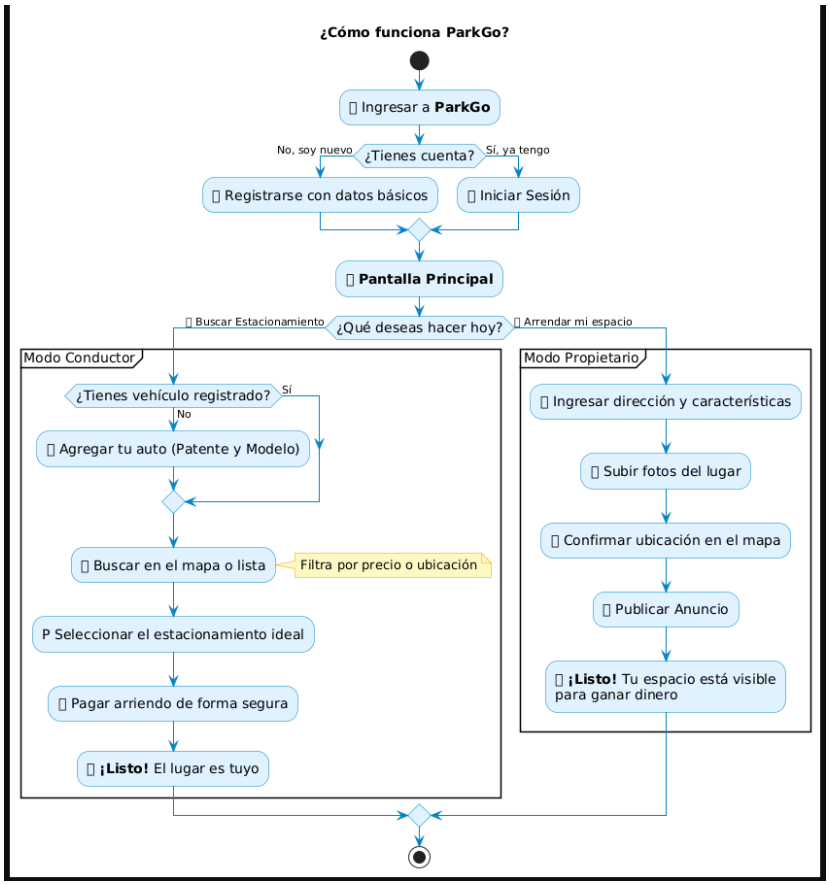
4. Diagrama de actividad “Gestión de usuarios (Admin)”: Muestra los permisos del administrador para manejar la plataforma basado en **adminController.js.** El sistema verifica que el usuario tenga el id_rol = 3 para poder cargar la lista. La visualización en el panel muestra los roles de los usuarios entre conductores y arrendatarios. Con opción en el panel para “Banear” a un usuario, en el backend en adminController.js se maneja la lógica de que el admin no puede banearse a sí mismo.



5. Diagrama de estado “Ciclo de vida de una publicación”: Muestra el estado con el tiempo de una publicación, el estado inicial no comienza hasta que el dueño termina el wizard, luego aparece como disponible, luego a transición o inactiva, ocurre cuando un usuario realiza el pago de un arriendo y desaparece del buscador. Cuando existe un término de contrato la publicación vuelve a estar disponible. En el estado de eliminada cuando el usuario dueño decide borrarla permanentemente.

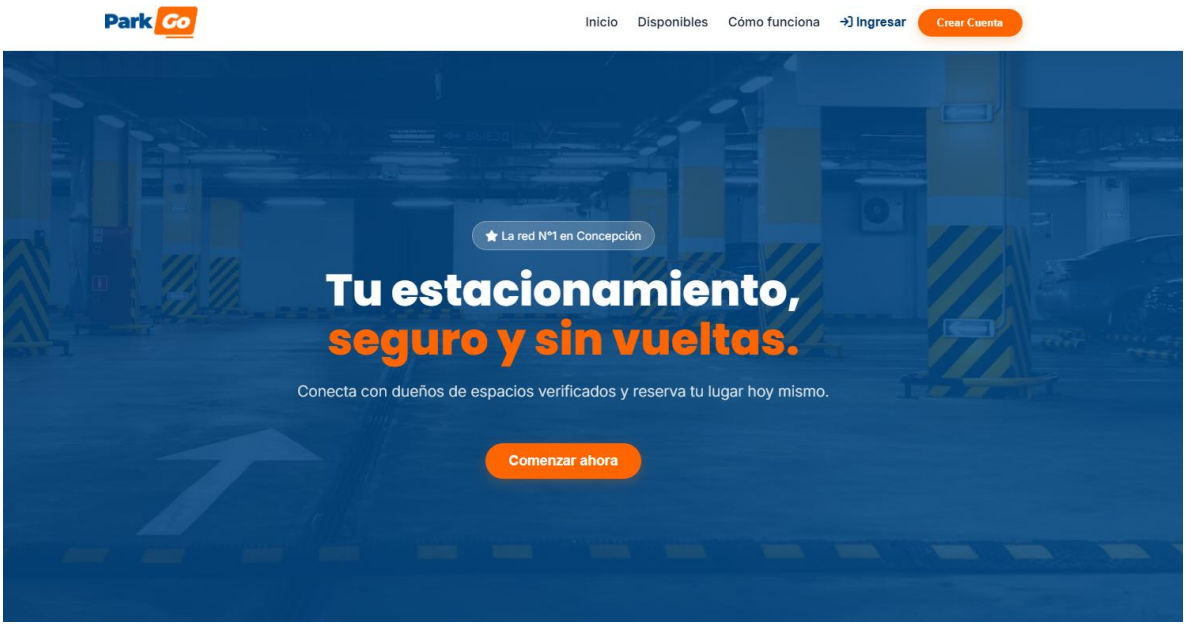


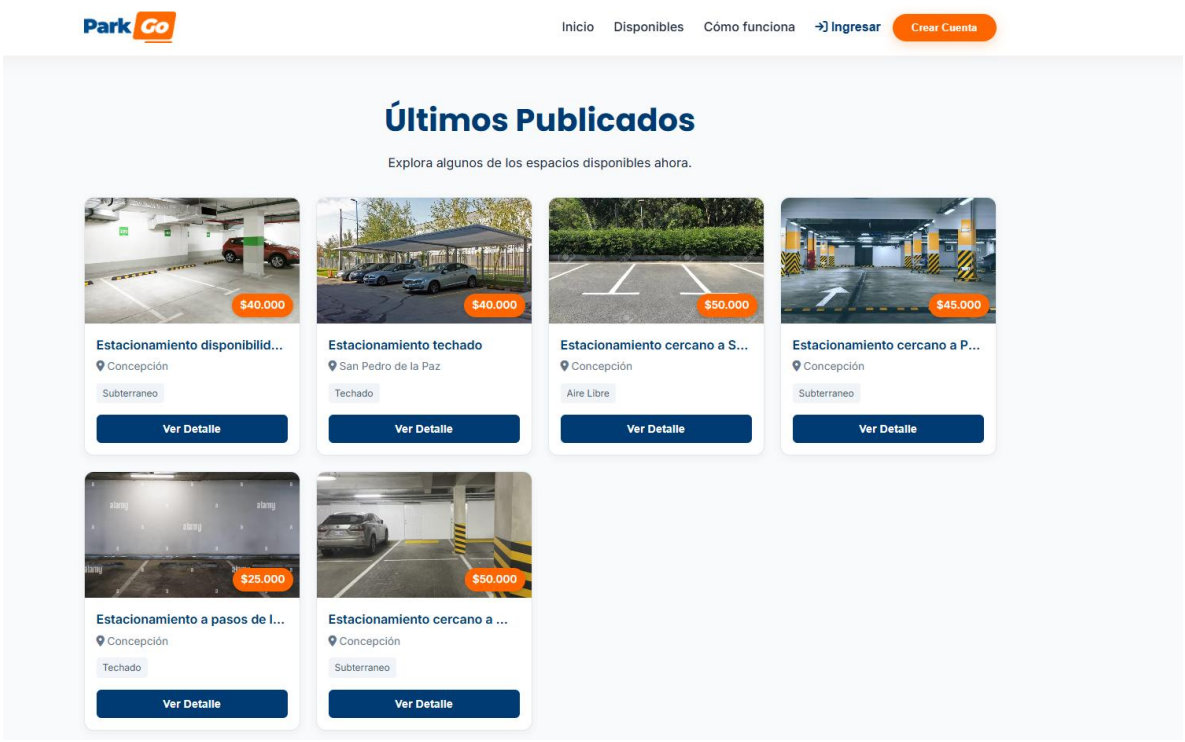
6. Diagrama simplificado con flujo del sistema.



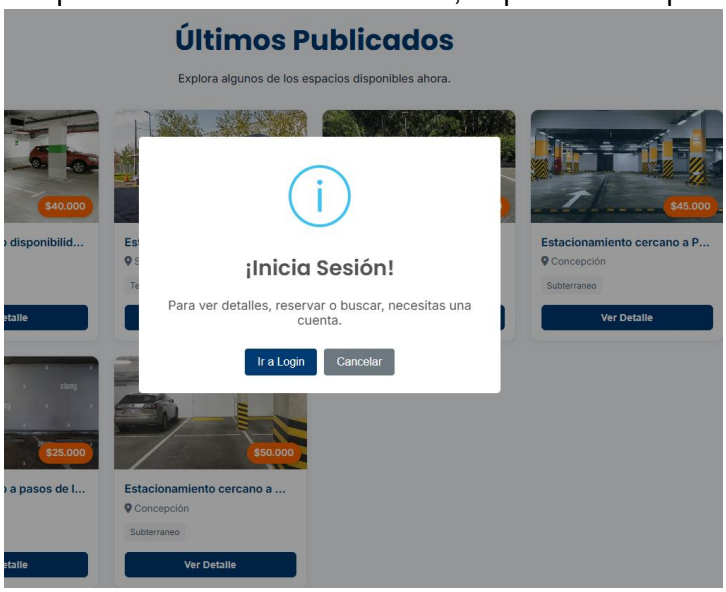
13. Flujo de pantallas del sistema

1. Pantalla de inicio con botones necesarios y vista previa de estacionamientos publicados.

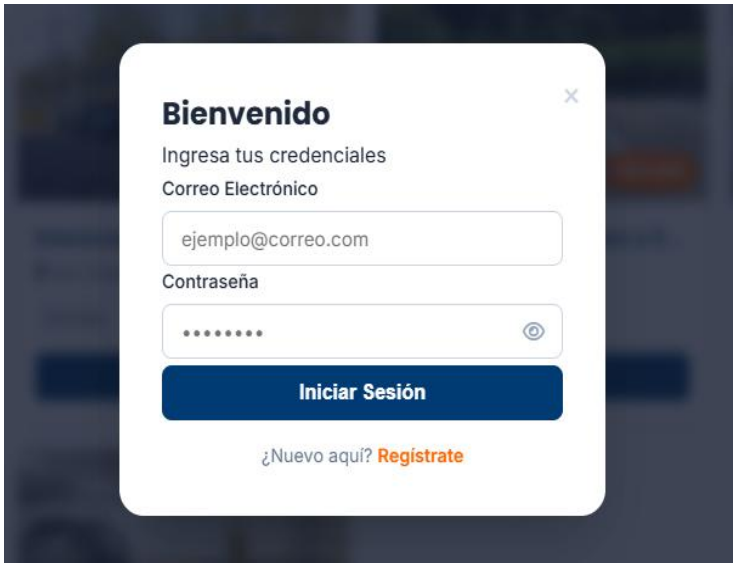




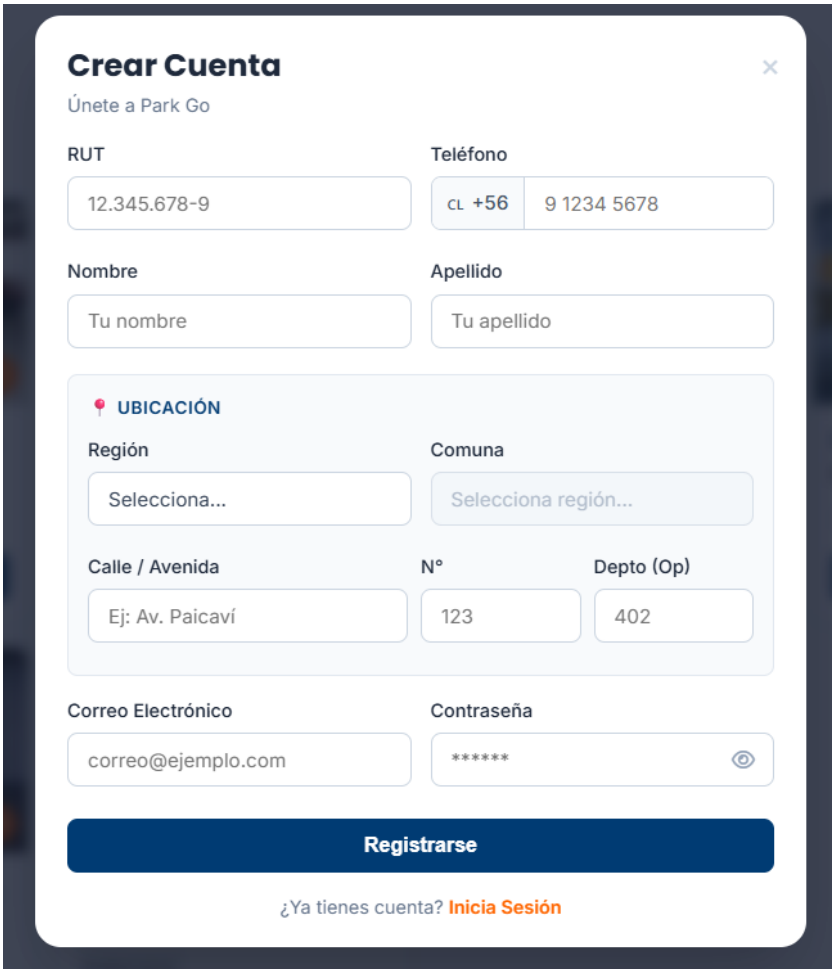
2. Al presionar en “Ver detalles”, aparece la pantalla de login o registro.



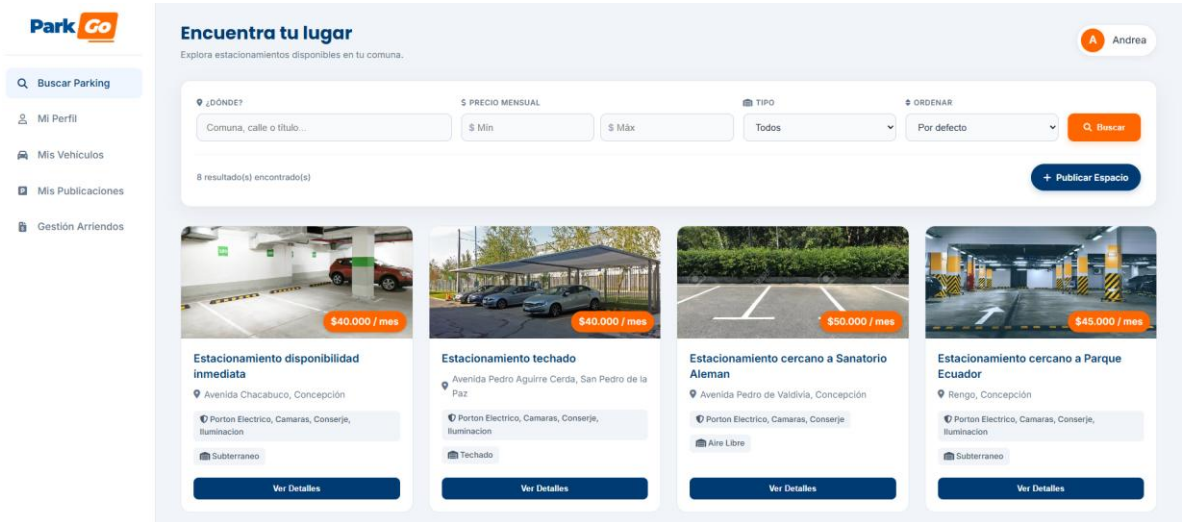
3. Pantalla de inicio de sesión.



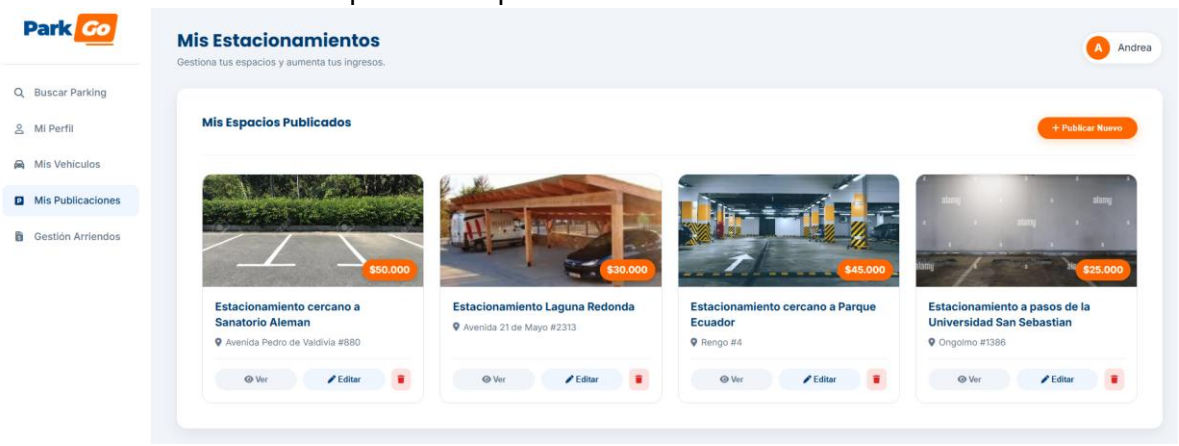
4. Pantalla de registro:



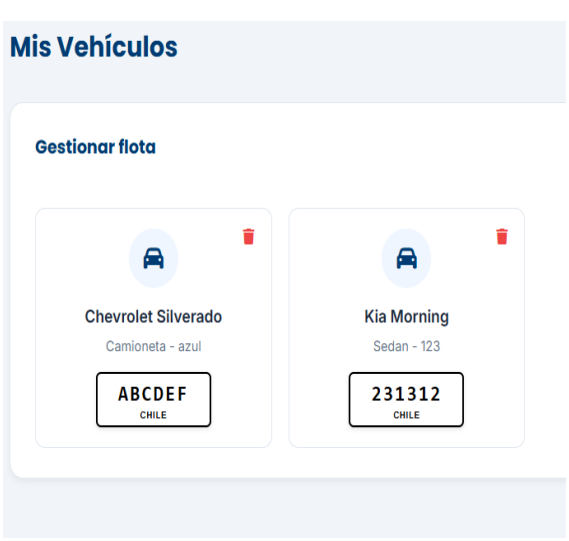
5. Login de usuario común registrado:

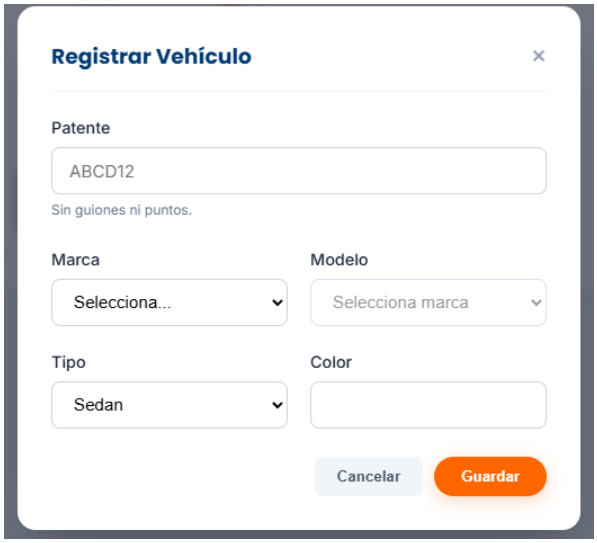


6. Vista de estacionamientos publicados por el usuario:

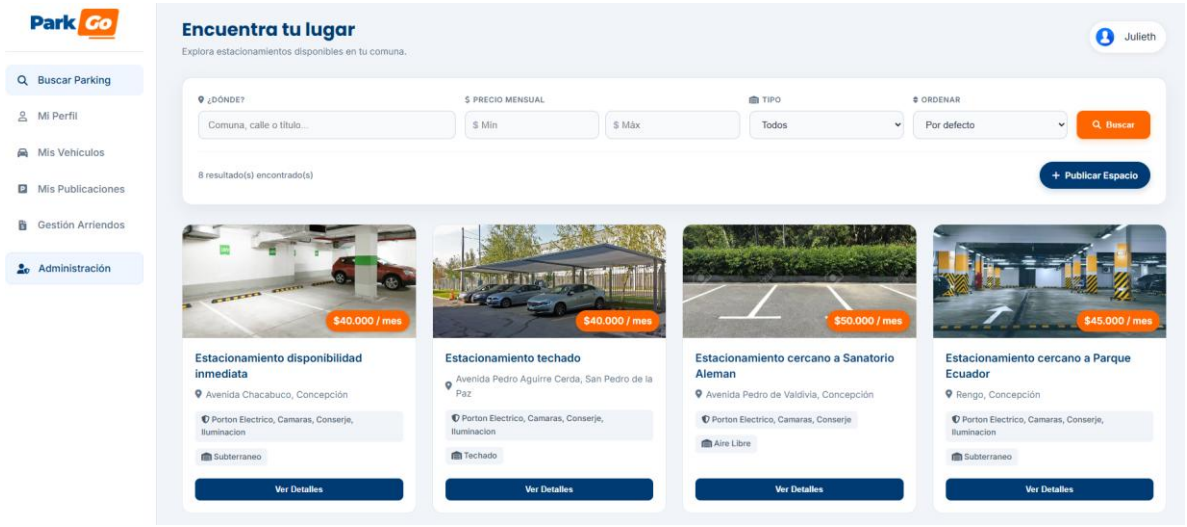


7. Vista de vehículos y formulario de registro:

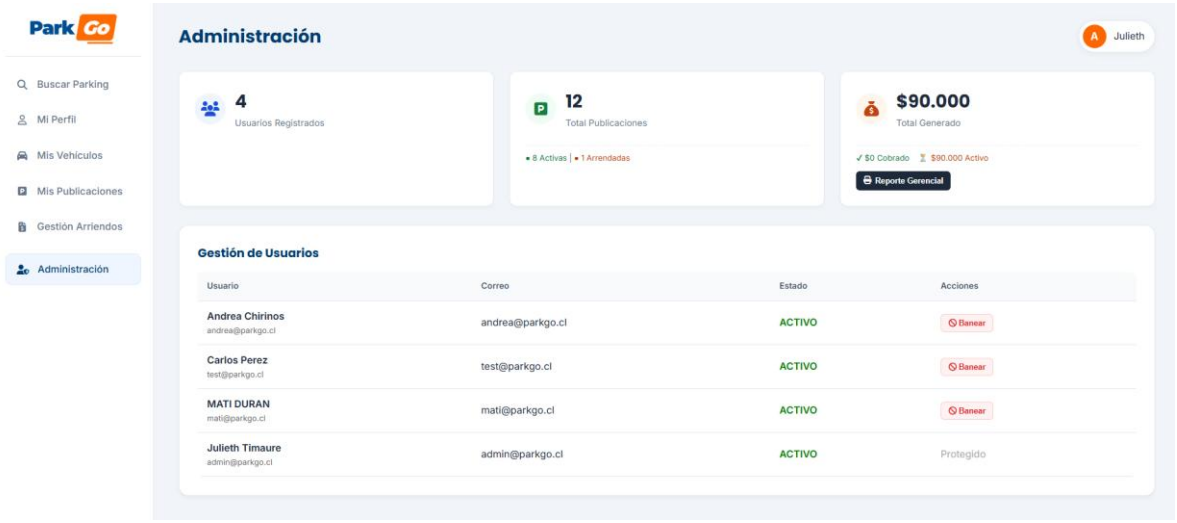




8. Login de usuario administrador:



9. Funciones especiales de administrador:



14. Plan de pruebas ParkGo

14.1. Introducción y Objetivos:

Este plan tiene como objetivo validar la robustez de la entrada de datos críticos y la integridad referencial de la base de datos.

14.2. Objetivo Principal: Asegurar que no se registren datos inválidos (RUT erróneo) o duplicados, y verificar el comportamiento del sistema al intentar eliminar registros vinculados (Publicaciones).

14.3. Alcance:
Se probará:

- Registro de usuarios (authController.js): Validación del módulo 11.
- Registro de usuarios (authController.js): Unicidad y formato del correo.
- Gestión de Estacionamientos (parkingController.js): Eliminación de registros.

14.4. Recursos:
Código Fuente: Archivos authController.js, parkingController.js.

Base de Datos: PostgreSQL con datos pre-cargados (Usuarios y Publicaciones activas).

14.5. Cronograma:

Inicio: Inmediato.

Ejecución: 1 hora (Pruebas manuales guiadas).

Cierre: Entrega de reporte de bugs al finalizar.

14.6. Entorno de Pruebas:
Servidor Local (localhost:3000).

Base de datos con al menos:

1 Usuario registrado.

1 Publicación activa asociada a un contrato o reserva.

14.7. Criterios de Entrada y Salida:
Entrada: Base de datos operativa y API respondiendo.

Salida: Documentación del error de eliminación y confirmación de validaciones de registro.

14.8. Diseño de Casos de Prueba (Guiados)

A continuación, los casos simplificados solicitados:

14.9. CP-001 Validación de Algoritmo RUT (Módulo 11)

Objetivo: Verificar si el backend valida que el dígito verificador del RUT coincida matemáticamente con el cuerpo del número.

Precondiciones: Servicio de registro activo.

Pasos:

- Enviar petición POST /api/auth/register.
- Datos: rut: "11.111.111-K" (Este RUT es matemáticamente invalido).
- Resultado Esperado (Ideal): El sistema debe retornar 400 Bad Request indicando "RUT Inválido".
- Resultado Real (Bajo Prueba):

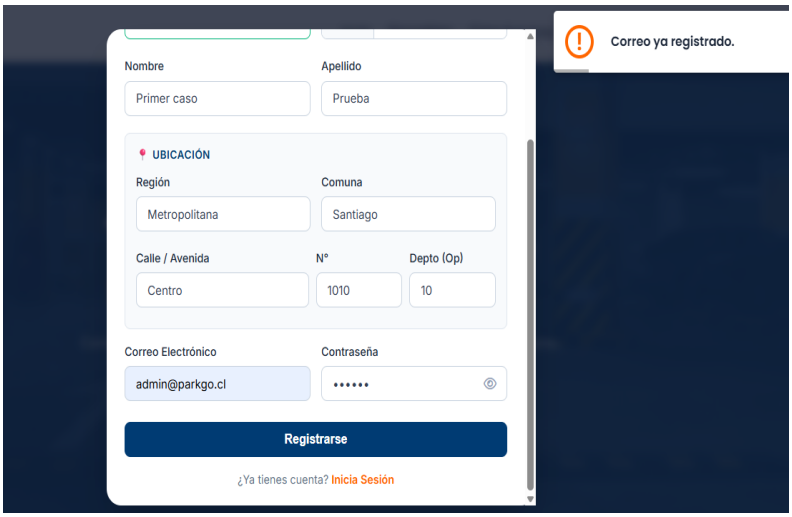
14.10. CP-002 Unicidad de Usuario (RUT y Correo)

Objetivo: Asegurar que no se puedan crear dos cuentas con el mismo RUT o correo.

Precondiciones: Ya existe el usuario con correo admin@parkgo.cl.

Pasos:

- Enviar petición POST /api/auth/register.
- Intentar registrar nuevamente admin@parkgo.cl.
- Resultado Esperado: Código 409 Conflict con mensaje "El usuario o correo ya existe".



14.11. CP-003 Eliminación de Publicación con Dependencias (Integridad Referencial)

Objetivo: Verificar el comportamiento al eliminar un estacionamiento que ya tiene publicaciones o arriendos asociados.

Precondiciones: Existe un estacionamiento que tiene una publicación y esta a su vez tiene arriendos (tabla contrato).

Pasos:

Identificar el estacionamiento ocupado.

Enviar petición DELETE /api/parkings/delete/1.

Resultado Esperado: El sistema debería validar previamente si tiene hijos y bloquear la acción amigablemente o realizar un "Soft Delete" (marcar como inactivo).

Resultado real CP-03: Eliminación de publicación – Fallido con error crítico.

Detalle de errores encontrados en CP-03:

Error de integridad referencial al eliminar estacionamiento. El código intenta ejecutar directamente:

JavaScript

```
const result = await pool.query('DELETE FROM estacionamiento WHERE id_estacionamiento = $1', [id]);
```

Dado que la base de datos (según archivo sql) tiene relaciones de llave foránea (FOREIGN KEY) desde la tabla publicación hacia estacionamiento, y desde contrato hacia publicación, la base de datos bloquea la eliminación lanzando una excepción de "Violación de restricción de integridad".

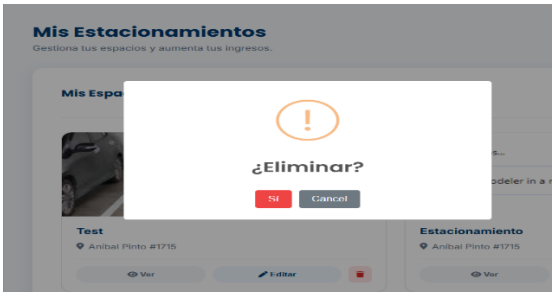
Comportamiento Observado:

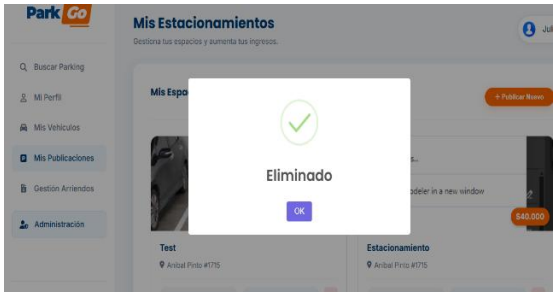
El servidor captura el error en el bloque catch y devuelve un error interno sin informar al usuario, en la pantalla aparece un “Eliminado”.

Plan de Mitigación:

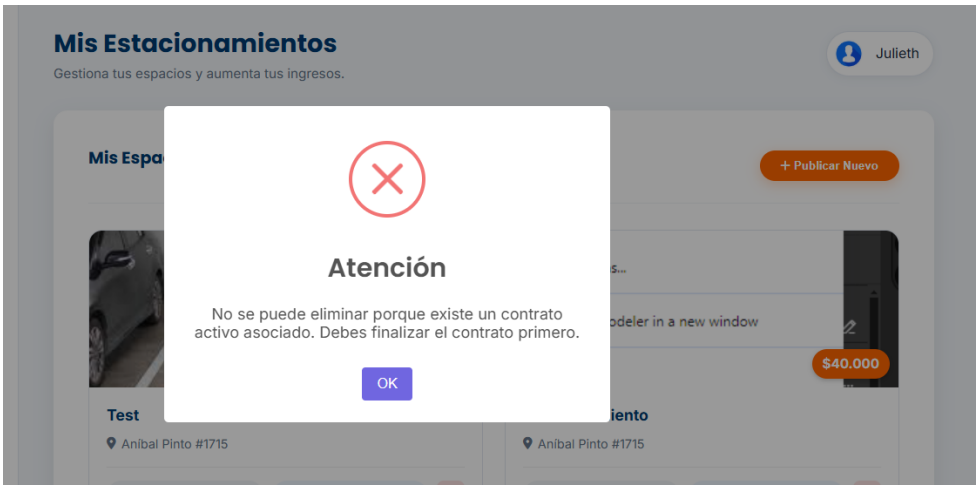
Cambiar la lógica a Eliminación Lógica y modificar los archivos: **parkingController.js, mis-estacionamientos-logic.js** y **wizard-logic.js.**

Evidencias:

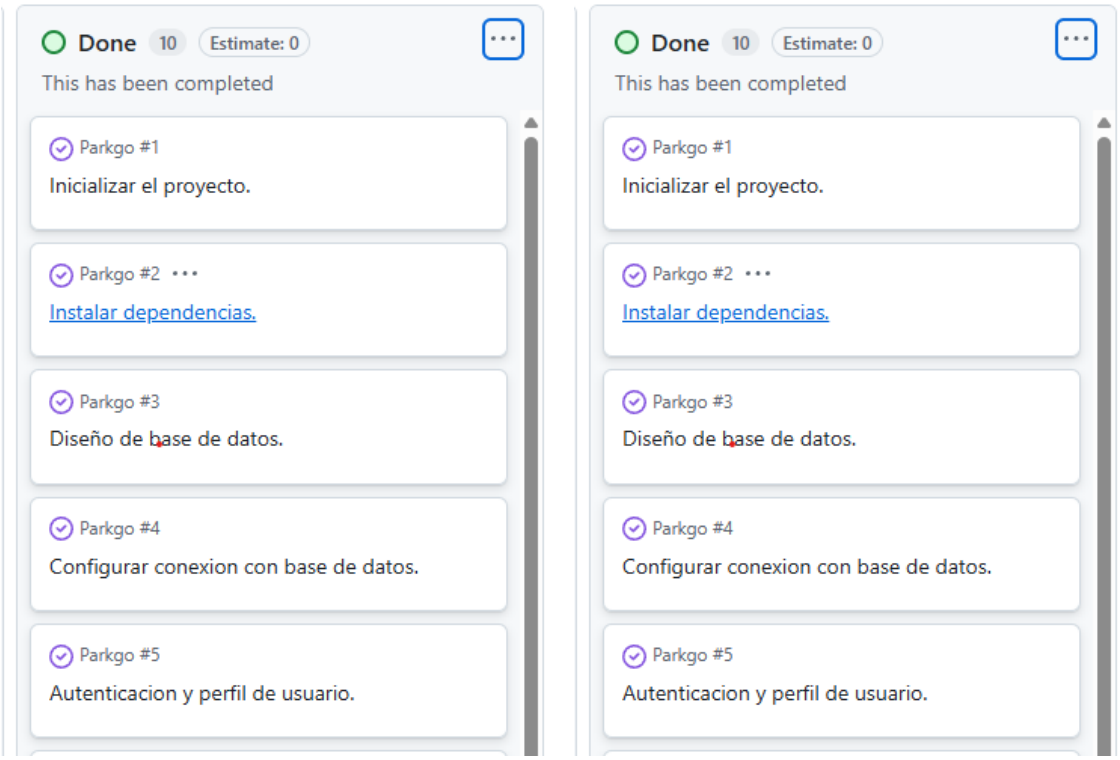




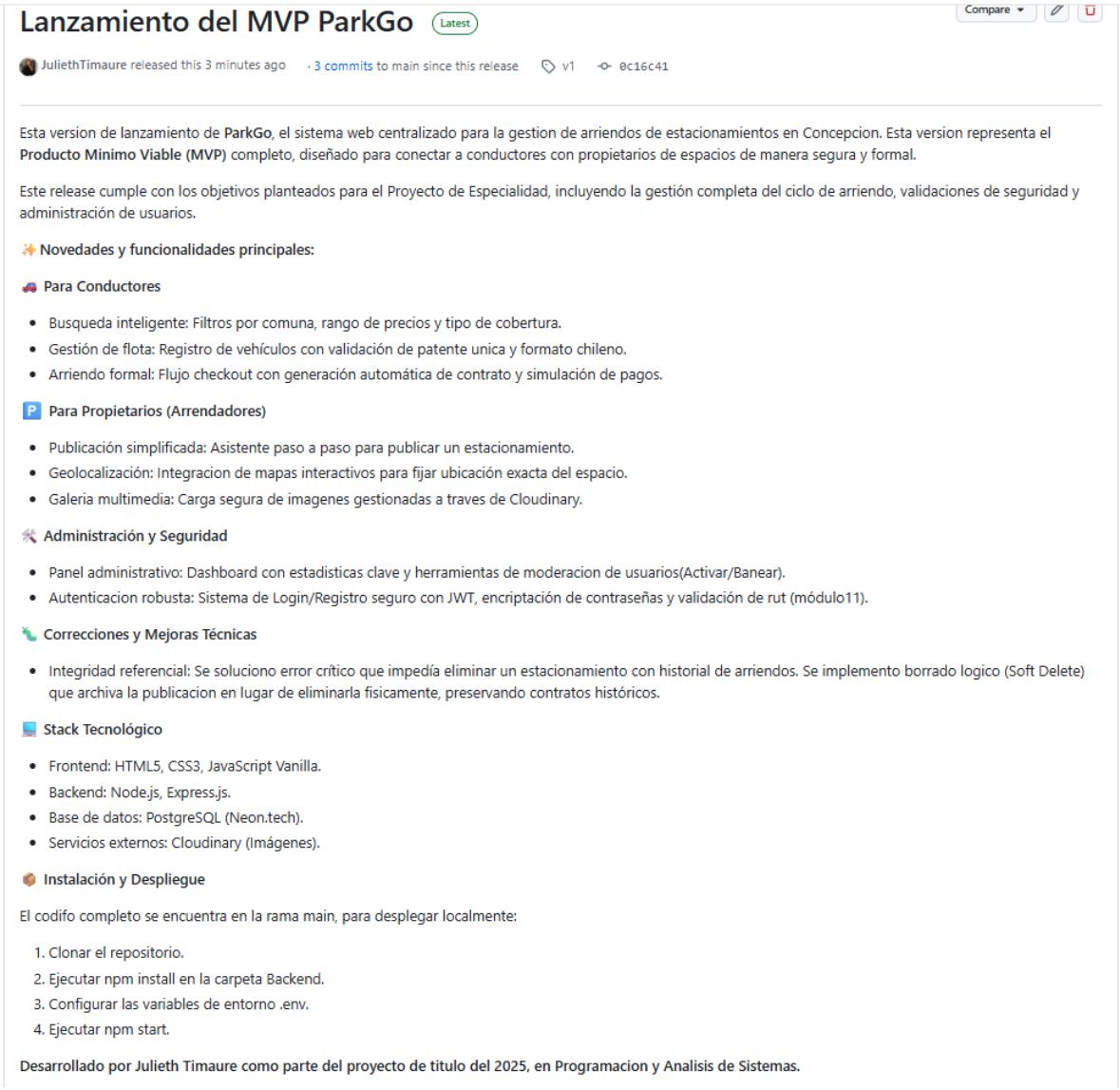
Correcciones aplicadas:



15. Tablero kanban GitHub



16. Release con versión final del proyecto



Lanzamiento del MVP ParkGo

Latest

Compare

JuliethTimaure released this 3 minutes ago · 3 commits to main since this release v1 0c16c41

Esta version de lanzamiento de **ParkGo**, el sistema web centralizado para la gestion de arriendos de estacionamientos en Concepcion. Esta version representa el **Producto Minimo Viable (MVP)** completo, diseñado para conectar a conductores con propietarios de espacios de manera segura y formal.

Este release cumple con los objetivos planteados para el Proyecto de Especialidad, incluyendo la gestión completa del ciclo de arriendo, validaciones de seguridad y administración de usuarios.

✨ **Novedades y funcionalidades principales:**

🚗 **Para Conductores**

- Busqueda inteligente: Filtros por comuna, rango de precios y tipo de cobertura.
- Gestión de flota: Registro de vehículos con validación de patente unica y formato chileno.
- Arriendo formal: Flujo checkout con generación automática de contrato y simulación de pagos.

🅿️ **Para Propietarios (Arrendadores)**

- Publicación simplificada: Asistente paso a paso para publicar un estacionamiento.
- Geolocalización: Integracion de mapas interactivos para fijar ubicación exacta del espacio.
- Galeria multimedia: Carga segura de imagenes gestionadas a traves de Cloudinary.

🛠️ **Administración y Seguridad**

- Panel administrativo: Dashboard con estadisticas clave y herramientas de moderacion de usuarios(Activar/Banear).
- Autenticacion robusta: Sistema de Login/Registro seguro con JWT, encriptación de contraseñas y validación de rut (módulo11).

🦎 **Correcciones y Mejoras Técnicas**

- Integridad referencial: Se soluciono error crítico que impedía eliminar un estacionamiento con historial de arriendos. Se implemento borrado logico (Soft Delete) que archiva la publicacion en lugar de eliminarla fisicamente, preservando contratos históricos.

💻 **Stack Tecnológico**

- Frontend: HTML5, CSS3, JavaScript Vanilla.
- Backend: Node.js, Express.js.
- Base de datos: PostgreSQL (Neon.tech).
- Servicios externos: Cloudinary (Imágenes).

📦 **Instalación y Despliegue**

El codifo completo se encuentra en la rama main, para desplegar localmente:

1. Clonar el repositorio.
2. Ejecutar npm install en la carpeta Backend.
3. Configurar las variables de entorno .env.
4. Ejecutar npm start.

Desarrollado por Julieth Timaure como parte del proyecto de titulo del 2025, en Programacion y Analisis de Sistemas.

Repositorio GitGub link: https://github.com/JuliethTimaure/Parkgo.git

17. Trabajo futuro y proyección tecnológica

Si bien la versión actual de ParkGo cumple con el entregable de un producto mínimo viable (MVP), se lograron identificar áreas estratégicas para la evolución del software, orientadas a escalar la plataforma hacia un entorno productivo masivo:

- Integración de una pasarela de pago real: Actualmente el proyecto cuenta con una simulación de transacciones financieras. El siguiente paso es integrar Webpay plus (Transbank), permitiendo el flujo real de dinero entre conductores y arrendatarios.

- Desarrollo de aplicación móvil: Dada la naturaleza con geolocalización del servicio, la migración o extensión hacia una aplicación móvil, podría mejorar la experiencia de usuario, integrando notificaciones de próximo pago o vencimiento de contrato.

- Verificación de identidad avanzada: Para lograr elevar los estándares de seguridad, se proyecta la implementación de un sistema de verificación de identidad automatizado para escanear cedula de identidad y validar la veracidad de los usuarios y la propiedad de los vehículos.

- Chat en tiempo real: Implementar la comunicación instantánea entre arrendatarios y conductores, para que logren coordinar detalles menores sin salir de la plataforma, mejorando la privacidad al no exponer datos personales.

18. Valor más allá el CRUD

Según lo exigido en el perfil de egreso, ParkGo cumple con las operaciones básicas de almacenamiento (Crear, Leer, Actualizar, Borrar) e incorpora un pequeño modulo con información general y relevante sobre la aplicación y su uso:

- Tablero de control para la toma de decisiones (Dashboard Administrativo): El sistema procesa los datos transaccionales en tiempo real para ofrecer al administrador métricas clave como el "Total generado" y la cantidad de "Arriendos activos".

- Valor aportado: Esta información permite a la administración detectar zonas de baja demanda o periodos de inactividad económica, facilitando la decisión de lanzar campañas de marketing dirigidas o promociones estacionales en comunas específicas. No es solo un listado de datos, es una herramienta de diagnóstico del negocio.

- Automatización Legal (Generación de Contratos): Más allá de guardar un registro en la base de datos, el sistema actúa como un gestor legal automatizado. Al momento de confirmar un arriendo, el backend orquesta la creación instantánea de un Contrato Digital vinculante.

- Gestión Inteligente de Disponibilidad (Estado de Máquina): El sistema implementa una lógica de negocio que gestiona los estados de las publicaciones (Disponible -> Inactiva -> Eliminada) de forma automática tras cada transacción.

- Valor aportado: Esto previene el error humano de "doble venta" o sobreventa de espacios, garantizando la integridad operativa del servicio sin que el dueño tenga que dar de baja su aviso manualmente cada vez que alquila su espacio.

19. Reflexión final del proyecto

La realización del presente proyecto de título ha representado un desafío integral que ha puesto a prueba tanto mis capacidades técnicas como de gestión. Más allá de la escritura de código, este proceso ha significado el enfrentamiento y superación de obstáculos complejos que simulan con precisión el entorno laboral real. A continuación, detallo las principales dificultades encontradas y cómo fueron resueltas.

19.1. Dificultades Técnicas:

Integridad referencial y el problema del "Borrado Físico": Uno de los desafíos técnicos más significativos ocurrió durante la implementación de la eliminación de estacionamientos en el controlador **parkingController.js**. Inicialmente, el sistema fallaba al intentar eliminar un estacionamiento que ya tenía contratos históricos asociados, debido a las restricciones de llave foránea (Foreign Key) en la base de datos PostgreSQL. Comprender que no se podía simplemente "borrar" un dato que forma parte de una historia transaccional fue clave.

Solución: Se tuvo que refactorizar la lógica del backend para implementar un "Soft Delete" (borrado lógico). En lugar de eliminar el registro, el sistema ahora captura el error de violación de integridad y actualiza el estado de la publicación a "Bloqueada" o "Eliminada", preservando la consistencia de los datos históricos.

Manejo de Asincronía en Cargas Multimedia (Cloudinary): La integración del servicio de Cloudinary para la gestión de imágenes presentó una complejidad inesperada. El reto consistía en sincronizar la subida de múltiples archivos al servicio en la nube con la inserción de sus respectivas URLs en la base de datos local, todo dentro de una misma transacción. Si la base de datos fallaba, las imágenes quedaban "huérfanas" en la nube, o viceversa.

Solución: Implementé un manejo robusto de promesas y bucles asíncronos en el controlador createParking, asegurando que las referencias se guarden correctamente en la tabla imagen_estacionamiento solo si la carga externa fue exitosa, optimizando así el rendimiento y el almacenamiento.

Configuración de Base de Datos en la Nube (SSL y Conectividad): La transición del entorno de desarrollo local a una base de datos en la nube (Neon.tech) generó problemas de conexión iniciales debido a los protocolos de seguridad. El backend rechazaba las conexiones por falta de certificados SSL válidos en el entorno de desarrollo.

Solución: Fue necesario investigar y configurar el pool de conexiones en el archivo db.js. Esto me enseñó la importancia crítica de la configuración de infraestructura.

19.2. Dificultades Personales:

Gestión del alcance: Personalmente, la mayor dificultad fue limitar el alcance del proyecto, y entender a profundidad la dinámica “Más allá del CRUD”, y como el desarrollo del software no solo se basa en construir un sistema, sino que debe brindar verdaderamente una solución a un problema real, que a su vez permita entregar información clave para la organización y su toma de decisiones, entender esta dinámica y aplicarla al proyecto represento una dificultad mayor.

Curva de aprendizaje y autonomía: Enfrentarme a la integración completa del stack (Frontend, Backend y Base de Datos) en solitario requirió un nivel de autoaprendizaje intenso. Hubo momentos de frustración al depurar errores que no eran evidentes en la documentación oficial. Sin embargo, desarrollar la resiliencia para investigar, leer logs de error y buscar soluciones en la comunidad técnica ha sido el mayor crecimiento personal de este proceso, dándome la confianza necesaria para mi futura inserción laboral.

CONCLUSION

El desarrollo de ParkGo ha culminado exitosamente con la entrega de un Producto Mínimo Viable (MVP) funcional que cumple con los objetivos de optimizar y asegurar el arriendo de estacionamientos en la región del Biobío. La plataforma logra digitalizar un proceso que anteriormente era manual, aportando valor tangible tanto a arrendatarios, quienes ahora cuentan con una herramienta de gestión de activos, como a conductores, que ganan inmediatez y confianza en su búsqueda.

Desde una perspectiva técnica, el sistema demuestra una solidez arquitectónica notable. La implementación de una base de datos normalizada en PostgreSQL, alojada en servicios de alta disponibilidad como Neon.tech, junto con un backend en Express.js, ha permitido manejar flujos complejos de datos garantizando la integridad referencial. Se ha logrado solucionar problemas críticos como la duplicidad de arriendos mediante transacciones atómicas y el manejo eficiente de recursos multimedia a través de servicios externos como Cloudinary.

En definitiva, ParkGo no solo resuelve la problemática operativa inicial, sino que establece una base tecnológica escalable. El proyecto valida la importancia de la coherencia entre el diseño de software y su implementación, demostrando que la aplicación de estándares de codificación y una metodología de trabajo ordenada son indispensables para el desarrollo de soluciones profesionales de calidad.

Bibliografía:

https://www.ine.gob.cl/sala-de-prensa/prensa/general/noticia/2024/09/13/m%C3%A1s-veh%C3%ADculos-en-chile-permisos-de-circulaci%C3%B3n-crecieron-4-6-en-el-2023