PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

Página Web de Gestión de Alquiler de Vehículos

Proyecto de titulación previo a la obtención del título de:

Tecnóloga en Desarrollo de Software

Autor: Antonella Liceth Játiva Álvarez

Tutor: Garzón Navarrete Billy Andrés

Quito, Ecuador

2025

Dedicatoria

Dedico esta tesis con todo mi amor y gratitud a mi mamá y a mis abuelitos, quienes han sido mi mayor fuente de fortaleza, inspiración y apoyo incondicional a lo largo de este camino. Gracias por acompañarme en cada etapa de esta travesía, por creer en mí incluso cuando yo misma dudaba, y por brindarme siempre su cariño, comprensión y sabios consejos en los momentos más difíciles.

Su presencia constante, sus palabras de aliento y su amor infinito me dieron la fuerza necesaria para seguir adelante y superar cada obstáculo. Este logro no habría sido posible sin su entrega, sacrificio y confianza en mí. Por todo eso y mucho más, esta meta alcanzada es también suya.

Tabla de contenidos

Lista de tablas

Lista de figuras

DECLARACIÓN y AUTORIZACIÓN

Yo, ANTONELLA LICETH JATIVA ALVAREZ con C.I. 1750637900 autor(a) del trabajo de Unidad de Integración Curricular intitulado: **"Pagina de Gestión de alquiler de vehículos"**, previa a la obtención del título de **Tecnología superior en desarrollo de software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 28/08/2025

ANTONELLA LICETH JATIVA ALVAREZ

C.I. 1750637900

Agradecimientos

En primer lugar, quiero expresar mi más sincero y profundo agradecimiento a la Magíster Diana Molina, quien fue un pilar fundamental en la realización de esta tesis y en el desarrollo del proyecto. Su apoyo constante, su compromiso incondicional y su guía experta marcaron una diferencia significativa en cada etapa del proceso. No solo me brindó orientación académica de alto nivel, sino que también me ofreció su tiempo, confianza y motivación cuando más lo necesité. Su acompañamiento cercano fue clave para superar los desafíos y mantenerme enfocada en mis objetivos. Gracias a sus valiosos consejos, sugerencias precisas y al impulso que me dio en momentos críticos, hoy puedo culminar esta etapa con satisfacción y gratitud. Sin duda, su dedicación y entrega fueron determinantes para el éxito de este trabajo, y me siento profundamente agradecida por haber contado con su apoyo tan cercano y comprometido.

De igual manera, extiendo mi agradecimiento al Ing. Billy Garzón, tutor de este trabajo, por su disposición y acompañamiento durante el proceso de desarrollo del proyecto. Sus observaciones constructivas en momentos clave contribuyeron a fortalecer algunos aspectos importantes del trabajo. Su experiencia profesional y sus aportes técnicos fueron valiosos para afinar ciertos detalles y mantener la orientación general del proyecto. Como también su apertura para atender inquietudes específicas.

1. Introducción

El presente trabajo de titulación tiene como objetivo el desarrollo de un sistema web para la automatización del proceso de alquiler de vehículos en la empresa Classic Rent a Car, ubicada en la ciudad de Quito. Esta iniciativa surge como respuesta a la necesidad de optimizar los procesos manuales que actualmente se realizan mediante formularios físicos, los cuales resultan propensos a errores, pérdida de información y dificultades en el seguimiento de mantenimientos.

Desde una perspectiva académica, este proyecto muestra todos los conocimientos adquiridos durante la carrera, integrando conceptos de análisis, diseño, programación y gestión de bases de datos. Asimismo, se adoptaron metodologías agiles, que facilitaron la organización y planificación por etapas, permitiendo una entrega gradual y funcional del sistema.

La solución tecnológica propuesta fue implementada utilizando herramientas modernas y de alto rendimiento: React para el desarrollo del frontend, proporcionando una interfaz de usuario amigable, intuitiva y responsiva; Node.js con Express para la construcción del backend, garantizando una gestión eficiente de la lógica del servidor y las rutas API; y PostgreSQL como sistema de gestión de base de datos, ofreciendo robustez, integridad y eficiencia en el almacenamiento de la información.

2. Objetivos

2.1 Objetivo General

2.1.1 Desarrollar una página web de gestión de alquiler de vehículos para la empresa Classic Rent a Car, que permita automatizar y optimizar los procesos de registro de clientes, administración de vehículos, control de alquileres, gestión de mantenimientos y almacenamiento de historiales, utilizando tecnologías modernas que garanticen eficiencia, seguridad y facilidad de uso.

2.2 Objetivos Específicos

2.2.1 Implementar al proyecto tecnologías como React, Node.js, Express y PostgreSQL, asegurando una comunicación fluida entre el frontend, backend y la base de datos.

2.2.2 Desarrollar funcionalidades clave como el registro de usuarios, gestión de clientes y vehículos, asignación de alquileres, alertas por mantenimiento, y visualización de historial.

2.2.3 Aplicar validaciones automáticas en formularios y controles visuales que mejoren la experiencia del usuario y prevengan errores comunes de ingreso de datos.

3. Levantamiento de Requisitos y Diseño del Sistema

3.1 Requisitos funcionales

3.1.1 Registro y autenticación de usuarios

El formulario de registro y autenticación de usuarios se los realizo utilizando React. Durante el proceso de registro, el usuario ingresa su nombre de usuario, contraseña, etc en un formulario con sus respectivas validaciones. Una vez completado, se envía una solicitud al servidor mediante una petición POST, la cual es procesada por el backend construido con Node.js y Express.

Ya registrado el usuario, se procede con la autenticación en su respectivo formulario. En el backend, el controlador correspondiente realiza la verificación de credenciales. Primero, se comprueba si el usuario ingresado existe en la base de datos PostgreSQL. Si existe, se procede a comparar la contraseña ingresada con la almacenada, previamente encriptada con bcrypt. En caso de coincidencia, se permite el acceso al sistema y se redirige al menú principal.

En caso de que el usuario no exista, o los campos estén vacíos y las credenciales sean incorrectas el sistema despliega automáticamente una alerta con el mensaje correspondiente (Usuario no encontrado, usuario y contraseña son requeridos y credenciales invalidas).

Figura 1:
Sección del Login

Nota. Captura de pantalla del sistema desarrollado (2025)

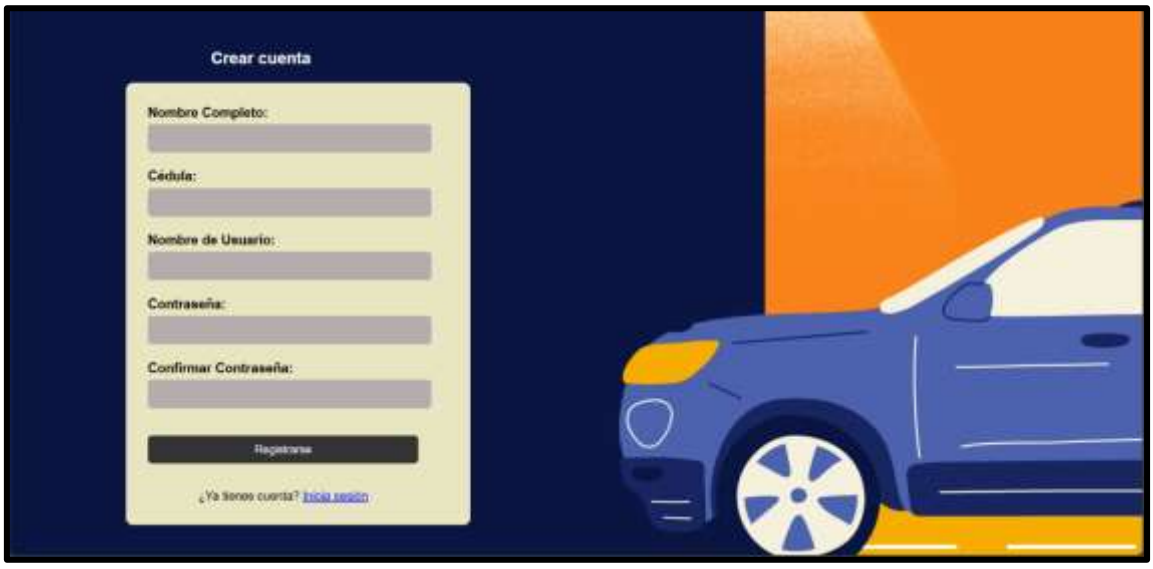

Figura 2:
Sección de Registro

Nota. Captura de pantalla del sistema desarrollado (2025)

3.1.2 Gestión de clientes

La interfaz de clientes fue construida con React y permite al usuario registrar, editar, buscar y eliminar información de clientes. La funcionalidad de búsqueda se realiza en tiempo real por nombre o número de cédula. Cada cliente puede ser vinculado a un alquiler mediante un botón de acción. Todas las operaciones están conectadas al backend mediante una API REST, la cual utiliza controladores desarrollados en Node.js. La persistencia de datos se realiza en PostgreSQL, utilizando Sequelize como ORM para la comunicación con la tabla correspondiente.

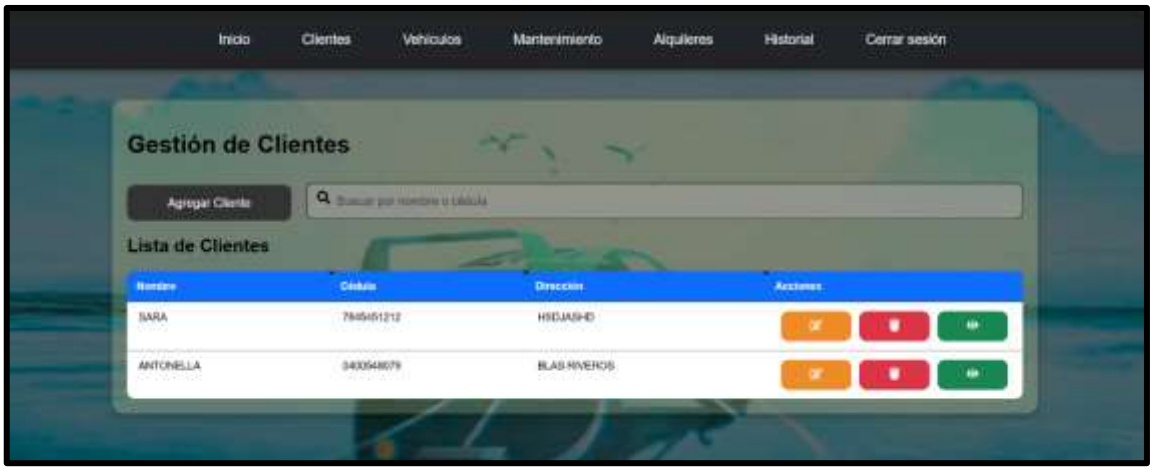

Figura 3:
Página principal de clientes

Nota. Captura de pantalla del sistema desarrollado (2025)

3.1.3 Gestión de vehículos

Se desarrolló una interfaz en React similar a la del módulo de clientes, desde la cual se pueden registrar, editar o eliminar vehículos. Los campos incluyen marca, modelo, año, color, placa. El botón "Ver detalles" dentro de vehículos permite acceder a una vista extendida donde se puede asignar un mantenimiento específico. El backend se encarga de validar los datos y registrar los cambios en la base de datos PostgreSQL, utilizando Sequelize.

3.1.4 Gestión de mantenimientos

Cuando en el alquiler el kilometraje o la fecha próxima en el mantenimiento ha alcanzado el límite, automáticamente redirige al formulario de mantenimiento. Este formulario, desarrollado con React, contiene validaciones que impiden guardar la información si no se actualiza el campo correspondiente al mantenimiento requerido. El backend, mediante controladores en Express, actualiza los valores en la tabla mantenimientos de PostgreSQL.

Figura 4:
Formulario de registro de mantenimientos

Nota. Captura de pantalla del sistema desarrollado (2025)

3.1.5 Gestión de alquileres

Se diseñó un formulario completo en React que permite registrar nuevos alquileres de vehículos. El formulario incluye un selector para elegir el vehículo, y automáticamente carga la información del cliente asociado. La fecha de inicio y finalización es validada para impedir fechas anteriores a la actual. Una vez ingresado el kilometraje de llegada, el sistema verifica si coincide con alguno de los valores de mantenimiento programado (por ejemplo, cambio de aceite). En ese caso, se dispara una

alerta automática y el usuario es redirigido al formulario de mantenimiento para registrar la actualización correspondiente.

3.1.6 Gestión del historial

El módulo de historial fue diseñado para registrar de forma automática todos los alquileres completos. Al concluir un alquiler, el formulario desarrollado en React genera una solicitud al backend para transferir los datos correspondientes desde la tabla de alquileres hacia la tabla historial en la base de datos PostgreSQL. Este procedimiento garantiza que se conserven todos los registros.

Además, se incorporó una funcionalidad visual que mejora la interpretación rápida de los registros: el color de fondo de cada fila del historial varía según la calificación del cliente**.** Esta calificación se determina por criterios definidos en el sistema (como puntualidad, estado del vehículo al devolverlo, cumplimiento de normas, entre otros).

Figura 5:
Página principal del historial

Nota. Captura de pantalla del sistema desarrollado (2025)

3.2. Requisitos no funcionales

3.2.1 Usabilidad

El programa presenta una interfaz amigable e intuitiva para el usuario desarrollada con React. Todos los formularios están organizados de forma clara con validaciones inmediatas para prevenir errores de ingreso. Además, se emplean mensajes emergentes (alertas) que guían al usuario en caso de errores u omisiones.

3.2.2 Interfaz responsiva y atractiva

La interfaz se adapta a diferentes resoluciones de pantalla, permitiendo su uso tanto en monitores de escritorio como en laptops. Se utilizaron estilos personalizados en CSS y componentes reutilizables que mejoran la presentación visual del sistema.

3.2.3 Validación y prevención de errores

Los formularios desarrollados en React incluyen validaciones en tiempo real que impiden el envío de datos vacíos, incorrectos o inconsistentes (como fechas inválidas o cédulas incompletas). Estas validaciones reducen errores en la base de datos.

3.2.4 Seguridad en la autenticación

El sistema implementa medidas básicas de seguridad para proteger el acceso. Las contraseñas de los usuarios son encriptadas con bcrypt antes de almacenarse en la base de datos. Durante el inicio de sesión, se valida la existencia del usuario y la coincidencia de la contraseña.

3.2.5 Modularidad y escalabilidad

El código tanto del frontend como del backend fue estructurado de forma modular. Cada entidad (usuarios, vehículos, alquileres, mantenimientos) cuenta con sus propios componentes, controladores y rutas, facilitando futuras ampliaciones o mantenimiento del sistema.

3.2.6 Rendimiento y estabilidad local

El programa fue ejecutado en entorno local con fluidez, sin caídas ni errores críticos. Las consultas a la base de datos PostgreSQL y las respuestas del backend son rápidas, gracias al uso eficiente de Express y Sequelize.

3.3 Estado del arte

En la actualidad, la tecnología ha avanzado mucho tanto que ha impulsado la creación de sistemas web que automatizan procesos empresariales, permitiendo mayor eficiencia y seguridad. Dentro del sector automotriz, específicamente en el área de alquiler de vehículos, diversas empresas han adoptado plataformas digitales que integran módulos como gestión de clientes, control de mantenimiento preventivo y manejo de historial de servicios. Este tipo de soluciones no solo optimiza recursos, sino que también mejora significativamente la experiencia del usuario.

3.3.1 Sistemas similares existentes

- Rent Centric y Navotar: plataformas comerciales que ofrecen soluciones completas para la administración de flotas, reservas online, contratos digitales y mantenimiento de vehículos. Estas herramientas suelen estar orientadas a grandes empresas y tienen costos elevados de implementación y suscripción.
- Fleet Complete: un sistema integral de gestión de flotas, más enfocado al monitoreo y mantenimiento vehicular, que incluye funcionalidades avanzadas como GPS en tiempo real y programación automática de mantenimientos.

3.3.2 Tecnologías utilizadas en sistemas actuales.

El uso de tecnologías modernas como React.js**,** Node.js**,** Express y PostgreSQL ha ganado gran popularidad en el desarrollo de soluciones web debido a su alto rendimiento, modularidad y escalabilidad. Estas herramientas permiten construir

aplicaciones SPA (Single Page Applications), que brindan una experiencia fluida y dinámica al usuario, sin necesidad de recargar la página constantemente. Además, su arquitectura basada en componentes facilita la reutilización de código y mejora el mantenimiento del sistema.

3.3.3 Novedad y aporte del presente proyecto

El sistema desarrollado para la empresa Classic Rent a Car se alinea con estas tendencias tecnológicas, pero además presenta un valor agregado importante: la automatización de alertas de mantenimiento. A través del uso de condiciones que verifican el kilometraje y fechas programadas, el sistema redirige automáticamente al módulo de mantenimiento, bloqueando cualquier intento de guardar registros sin actualizar los valores críticos. Esta lógica proactiva, implementada tanto en el frontend como en el backend, garantiza la conservación y buen estado de los vehículos, mejorando la confiabilidad del servicio.

Asimismo, el proyecto implementa una interfaz intuitiva, con validaciones automáticas, sistema de calificaciones visuales para los clientes y una estructura modular que permite futuras expansiones, como la incorporación de pagos en línea o reportes generados automáticamente.

3.3.4 Conclusión del estado del arte

Aunque existen múltiples sistemas de gestión para alquiler de vehículos en el mercado, el presente proyecto se distingue por su enfoque específico a una empresa local, su arquitectura moderna cliente-servidor, y su mecanismo automatizado de control de mantenimiento preventivo. Esta propuesta representa una solución accesible, funcional y adaptable a las necesidades reales de pequeñas empresas que buscan

digitalizar sus procesos sin depender de plataformas comerciales costosas o poco flexibles.

4. Construcción del Sistema

La construcción del sistema web de gestión de alquiler de vehículos para la empresa Classic Rent a Car fue dividido en módulos, y cada uno fue implementado de forma progresiva, realizando pruebas locales para asegurar su correcto funcionamiento.

4.1 Tecnologías utilizadas

4.1.1 Frontend (Interfaz de usuario):

Se utilizó React.js, una biblioteca de JavaScript. Gracias a su arquitectura basada en componentes, fue posible crear formularios reutilizables y vistas con navegación fluida.

4.1.2 Backend (Lógica del servidor):

El backend fue construido con Node.js junto al framework Express, lo que facilitó la creación de rutas API REST para gestionar las peticiones del sistema. Se implementaron controladores para cada entidad: usuarios, clientes, vehículos, alquileres, historial y mantenimiento.

4.1.3 Base de datos

Se utilizó PostgreSQL como sistema de gestión de base de datos por su fiabilidad, escalabilidad y compatibilidad con datos relacionales. La comunicación con la base fue gestionada mediante Sequelize, un ORM que permite realizar operaciones CRUD (crear, leer, actualizar y eliminar) con mayor facilidad y control.

4.1.4 Estilos y diseño:

Se empleó CSS personalizado para dar estilo a los formularios, botones y tablas. También se aplicaron colores condicionales para reflejar el estado de ciertos registros, como el historial.

4.2 Base de datos

El backend está conectado a una base de datos **PostgreSQL**, mediante el ORM **Sequelize**, lo que permite manejar de forma estructurada los datos relacionales entre las distintas entidades. El servidor actúa como intermediario entre la base de datos y la interfaz, garantizando seguridad, validaciones y consistencia de los datos.

4.3 Diseño de la base de datos

Cada tabla contiene claves primarias, foráneas y campos relevantes como fechas, kilometrajes, observaciones, calificación, etc.

4.3.1 Tablas utilizadas en postgreSQL

Tabla 1:

Tabla de Clientes

Campo	Tipo de Dato	Restricciones
id	SERIAL	PRIMARY KEY
nombre	VARCHAR(100)	NOT NULL
cedula	VARCHAR(20)	UNIQUE, NOT NULL
ciudad	VARCHAR(50)	
sector	VARCHAR(50)	
referencia_del_domicilio	TEXT	
direccion	TEXT	
celular	VARCHAR(20)	
tipo_de_licencia	VARCHAR(20)	
nombre_de_la_empresa	VARCHAR(100)	
direccion_de_la_empresa	TEXT	
telefono_de_la_empresa	VARCHAR(20)	
nombreref	VARCHAR(100)	

celularref	VARCHAR(20)
imagen_cedula_frontal	TEXT
imagen_cedula_trasera	TEXT

Nota. Especificaciones tipo de Datos (2025)

Tabla 2:

Tabla de Registro

Campo	Tipo de Dato	Restricciones
id	SERIAL	PRIMARY KEY
nombre	VARCHAR(100)	NOT NULL
cedula	VARCHAR(20)	UNIQUE, NOT NULL
usuario	VARCHAR(50)	UNIQUE, NOT NULL
contrasena	VARCHAR(100)	NOT NULL
confirmacion_contra	VARCHAR(100)	NOT NULL
fecha_registro	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Nota. Especificaciones tipo de Datos (2025)

Tabla 3:

Tabla de Vehículos

Campo	Tipo de Dato	Restricciones
id	SERIAL	PRIMARY KEY
marca	VARCHAR(50)	NOT NULL
modelo	VARCHAR(50)	
placa	VARCHAR(20)	UNIQUE, NOT NULL
color	VARCHAR(30)	
anio	INTEGER	

Nota. Especificaciones tipo de Datos (2025)

Tabla 4:

Tabla de Mantenimientos

Campo	Tipo de Dato	Restricciones
id	SERIAL	PRIMARY KEY
vehiculo_id	INTEGER	FOREIGN KEY (vehiculos.id) ON DELETE CASCADE
aceite_ultimo	BIGINT	NOT NULL
aceite_proximo	BIGINT	NOT NULL
llantas_ultimo	DATE	
llantas_proximo	DATE	

pastillas_ultimo	DATE
pastillas_proximo	DATE
plumas_ultimo	DATE
plumas_proximo	DATE
frenos_ultimo	DATE
frenos_proximo	DATE

Nota. Especificaciones tipo de Datos (2025)

Tabla 5:

Tabla de Alquileres

Campo	Tipo de Dato	Restricciones
id	SERIAL	PRIMARY KEY
cliente_id	INTEGER	FOREIGN KEY (clientes.id) ON DELETE CASCADE
placa	VARCHAR(255)	NOT NULL
marca	VARCHAR(255)	NOT NULL
lugar	VARCHAR(100)	
km_salida	BIGINT	NOT NULL
km_llegada	BIGINT	
fecha_salida	DATE	NOT NULL
hora_salida	TIME	NOT NULL
fecha_llegada	DATE	
hora_llegada	TIME	
calidad_cliente	INTEGER	DEFAULT 0
descripcion	TEXT	
precio_diario	NUMERIC(10,2)	NOT NULL
precio_pagado	NUMERIC(10,2)	DEFAULT 0
precio_total	NUMERIC(10,2)	DEFAULT 0
saldo_restante	NUMERIC(10,2)	DEFAULT 0
estado	VARCHAR(10)	DEFAULT 'activo', CHECK ('activo', 'completo')
created_at	TIMESTAMP WITH TIME ZONE	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP WITH TIME ZONE	DEFAULT CURRENT_TIMESTAMP

Nota. Especificaciones tipo de Datos (2025)

Tabla 6:

Tabla del Historial

Campo	Tipo de Dato	Restricciones
id	SERIAL	PRIMARY KEY
alquiler_id	INTEGER	FOREIGN KEY (alquileres.id) ON DELETE SET NULL

accion	VARCHAR(255)	NOT NULL
detalles	JSONB	NOT NULL
fecha	TIMESTAMP WITH TIME ZONE	DEFAULT CURRENT_TIMESTAMP
placa	VARCHAR(20)	
modelo	VARCHAR(50)	
cliente_nombre	VARCHAR(100)	
cliente_cedula	VARCHAR(20)	

Nota. Especificaciones tipo de Datos (2025)

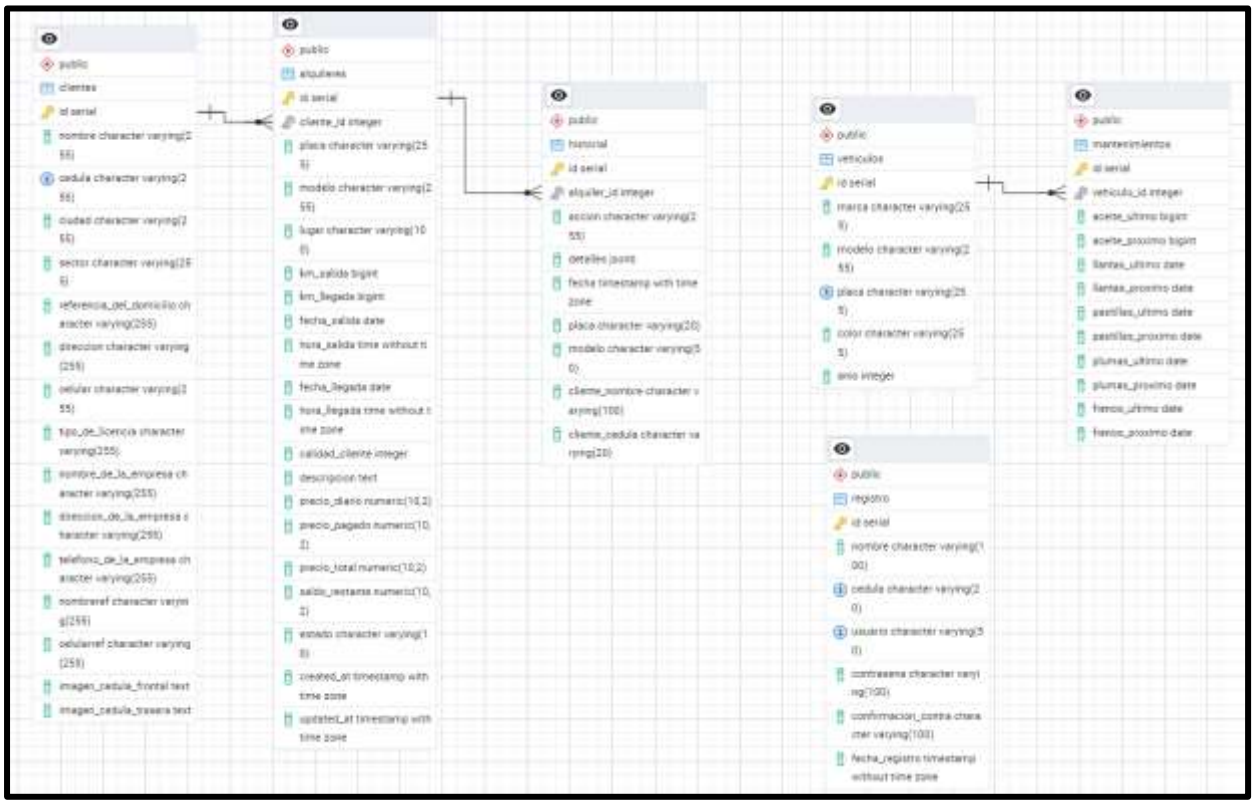

Figura 6:
Modelo Entidad-Relación obtenido de PostgreSQL

Nota. Captura de pantalla de la aplicación postgreSQL (2025)

4.4. Desarrollo por módulos

4.4.1 Gestión de usuarios

Se construyó un formulario de registro y otro de inicio de sesión. En el backend, se valida la existencia del usuario y se encripta la contraseña con bcrypt. Si las

credenciales coinciden, se permite el acceso al sistema. En caso contrario, se despliegan mensajes de error como “Usuario no encontrado”, “Credenciales inválidas”, etc.

4.4.2 Gestión de clientes

El formulario permite registrar, editar, buscar y eliminar clientes. Se incorporó un buscador en tiempo real por nombre o cédula. Además, cada cliente puede ser asociado a un alquiler.

4.4.3 Gestión de vehículos

Similar al módulo de clientes, permite registrar los datos del vehículo y asignar mantenimientos desde la vista de detalles. Se validan campos como marca, modelo, color, año y placa.

4.4.4 Gestión de alquileres

Permite registrar un nuevo alquiler. Se valida que la fecha no sea anterior a la actual y que el kilometraje de llegada no coincida con mantenimientos pendientes. Si lo hace, se genera una alerta y redirige al formulario de mantenimiento.

4.4.5 Gestión de mantenimientos

Desde este módulo se actualizan los valores de mantenimiento como cambio de aceite, llantas, pastillas, etc. El sistema bloquea el guardado si no se actualiza el campo crítico detectado como vencido.

4.4.6 Historial

Al concluir un alquiler, los datos se trasladan automáticamente al historial. En la interfaz, se visualizan todos los registros y se aplica un sistema de colores por calificación del cliente.

4.5 Arquitectura

El sistema fue construido bajo el modelo clásico de arquitectura cliente-servidor, donde los roles están claramente definidos entre el frontend (cliente) y el backend (servidor). Esta arquitectura permite una mejor organización del código, separación de responsabilidades y facilita futuras actualizaciones o despliegues del sistema.

4.5.1 Cliente (Frontend)

El cliente es la aplicación que interactúa directamente con el usuario final, y fue desarrollado utilizando React. Esta parte del sistema se encarga de renderizar las interfaces gráficas, validar los formularios, mostrar mensajes de alerta, y enviar las solicitudes correspondientes al servidor. Todas las vistas fueron desarrolladas con un enfoque modular, usando componentes reutilizables y adaptables.

4.5.2 Servidor (Backend)

El servidor fue desarrollado utilizando Node.js con el framework Express, y tiene como responsabilidad recibir, procesar y responder las peticiones del cliente. El backend expone una serie de rutas API REST que permiten al cliente realizar operaciones como crear un cliente, editar un vehículo, registrar un alquiler, consultar el historial, entre otras. Además, se implementaron controladores para encapsular la lógica de negocio.

4.5.3 Comunicación entre cliente y servidor

La comunicación se realiza a través de solicitudes HTTP (GET, POST, PUT, DELETE) que el frontend envía hacia las rutas definidas en el backend. Estas rutas procesan la información recibida y responden con datos en formato **JSON**, los cuales React interpreta para actualizar la interfaz en tiempo real.

5. Pruebas y Estabilización

Se realizaron pruebas de tipo GET, POST, PUT Y DELETE, para verificar que las rutas y controladores del servidor funcionen correctamente, se utilizó la herramienta Postman, que permite simular solicitudes HTTP y analizar las respuestas del servidor sin necesidad de una interfaz gráfica.

Cada prueba verificó que los datos se procesaran correctamente en el servidor, que las validaciones se cumplieran, y que se obtuviera una respuesta adecuada en formato JSON

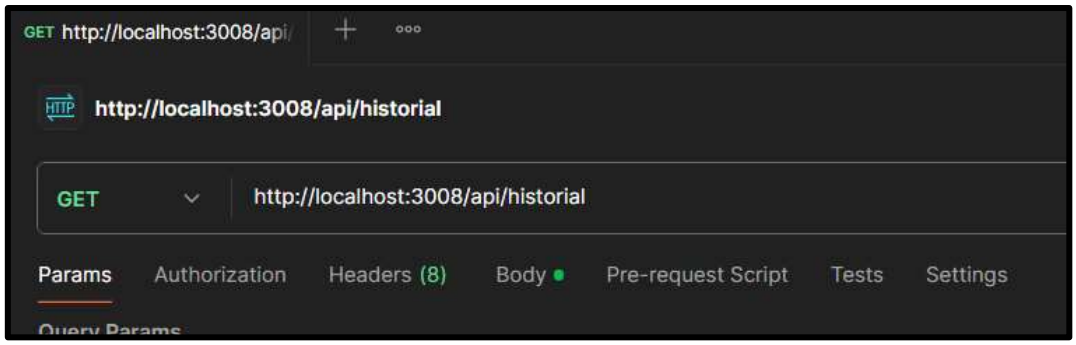

Figura 7:
Consulta en Postman

Nota. Captura de pantalla de la aplicación postman (2025)

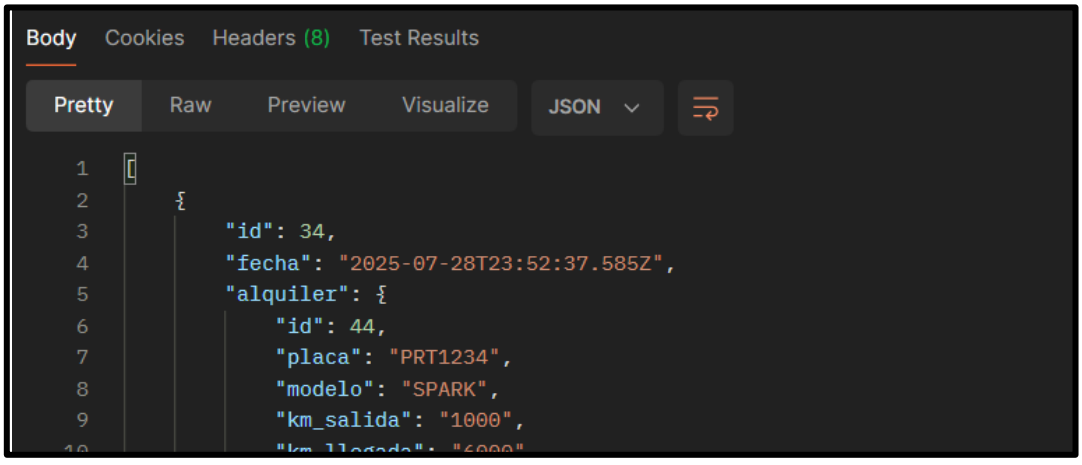

Figura 8:
Resultado de la consulta

Nota. Captura de pantalla de la respuesta de la consulta en postman (2025)

5.1 Pruebas en el frontend (React)

En la interfaz de usuario se realizaron pruebas funcionales para asegurar que cada vista y formulario cumpla con los requisitos establecidos. Se verificó que:

5.1.1 Todos los formularios tengan validaciones activas

5.1.2 Las alertas aparezcan correctamente cuando hay errores, inconsistencias

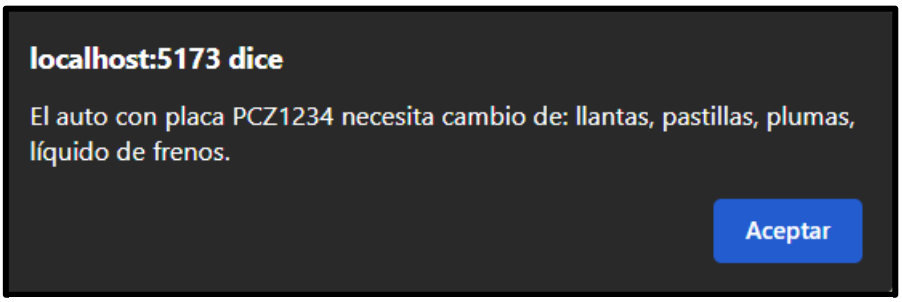

Figura 9:
Ejemplo de una notificación

Nota. Captura de pantalla del sistema desarrollado (2025)

5.1.3 El flujo de navegación entre componentes sea coherente y sin recargas innecesarias

5.1.4 Se carguen y muestren correctamente los datos provenientes del backend

5.1.5 Se ejecuten las redirecciones automáticas en casos como mantenimientos pendientes

5.2 Se probaron distintos flujos completos:

5.2.1 Registrar un cliente y alquilar un vehículo.

5.2.2 Registrar un alquiler con kilometraje que activa alerta de mantenimiento.

5.2.3 Editar un mantenimiento obligatorio y guardar los cambios.

5.2.4 Finalizar un alquiler y verificar que pase al historial.

5.2.5 Visualizar la calificación del cliente mediante el color correspondiente.

5.2 Estabilización

La estabilización del sistema consistió en una etapa posterior al desarrollo funcional, donde se buscó garantizar que todas las funcionalidades trabajen de forma consistente, sin errores inesperados, y que el sistema cumpla los estándares de calidad definidos.

Durante las pruebas funcionales realizadas tanto en backend como frontend, se detectaron errores menores (Campos obligatorios que permitían el envío en blanco, Botones que no cumplían con su funcionalidad, Kilometraje ingresado sin validación numérica, etc). Todos estos errores fueron corregidos, y se validó su solución en nuevas iteraciones de prueba.

6. Conclusiones

6.1 El sistema desarrollado permitió que la automatización de los procesos redujera el tiempo que antes se invertía en tareas manuales y disminuyó los errores humanos, lo que mejoró la eficiencia del trabajo diario y la calidad del servicio ofrecido a los clientes.

6.2 El uso de tecnologías como React, Node.js, Express y PostgreSQL facilitó bastante el desarrollo del sistema, ya que permitieron conectar bien todas las partes del proyecto. Gracias a estas herramientas, se logró que la página funcione correctamente.

6.3 Los formularios y validaciones automáticas que se añadieron ayudan bastante al usuario. La página muestra alertas cuando hay errores y no permite guardar información incorrecta, lo que mejora la experiencia de uso y asegura que los datos sean correctos.

6.4 Gracias a las pruebas realizadas, se logró identificar y corregir varios errores antes de entregar el sistema. Esto demuestra que probar el sistema constantemente es importante para garantizar que todo funcione bien y que el usuario tenga una buena experiencia.

7. Recomendaciones

7.1 Se recomienda realizar una capacitación breve a los empleados de la empresa para que puedan manejar el sistema con mayor facilidad y aprovechar todas sus funciones.

7.2 Se recomienda subir el sistema a internet usando un hosting con dominio propio, para que se pueda usar desde cualquier lugar y tenga mayor seguridad.

7.3 Se recomienda mantener el código actualizado y bien documentado, para facilitar futuras modificaciones o si otro desarrollador necesita trabajar en el sistema.

7.4 Se recomienda mantener un monitoreo constante del sistema en su ambiente de producción, para detectar errores a tiempo y brindar soporte oportuno.

7.5 Se recomienda realizar pruebas de rendimiento cuando el sistema se use en línea con varios usuarios al mismo tiempo, para garantizar su estabilidad.

8. Referencias bibliográficas

Developer Mozilla. (2023, enero 15). React: Introducción. MDN Web_Docs. https://developer.mozilla.org/es/docs/Learn_web_development/Core/Frameworks_libraries/React_getting_started

Node.js Foundation. (2024, febrero 10). Node: Introducción. Node. https://nodejs.org/en/about

PostgreSQL Global Development Group. (2024, abril 20). PostgresSQL: Introducción. PostgreSQL. https://www.postgresql.org/

Sequelize Contributors. (2024, mayo 9). Sequelize: Introducción. Sequelize. https://sequelize.org/docs/v7/

Anexos

Anexo A. Manual de Usuario

- En caso de no tener una cuenta creada, la puede crear haciendo clic en "¿no tienes cuenta? Regístrate"

Figura 1:
Pantalla de Login, para registrarse

Nota. Captura de pantalla de la aplicación (2025)

- Después de hacer clic, se desplegará la pantalla de registro en donde se tendrá que completar datos obligatorios, como el nombre, la cedula, etc.

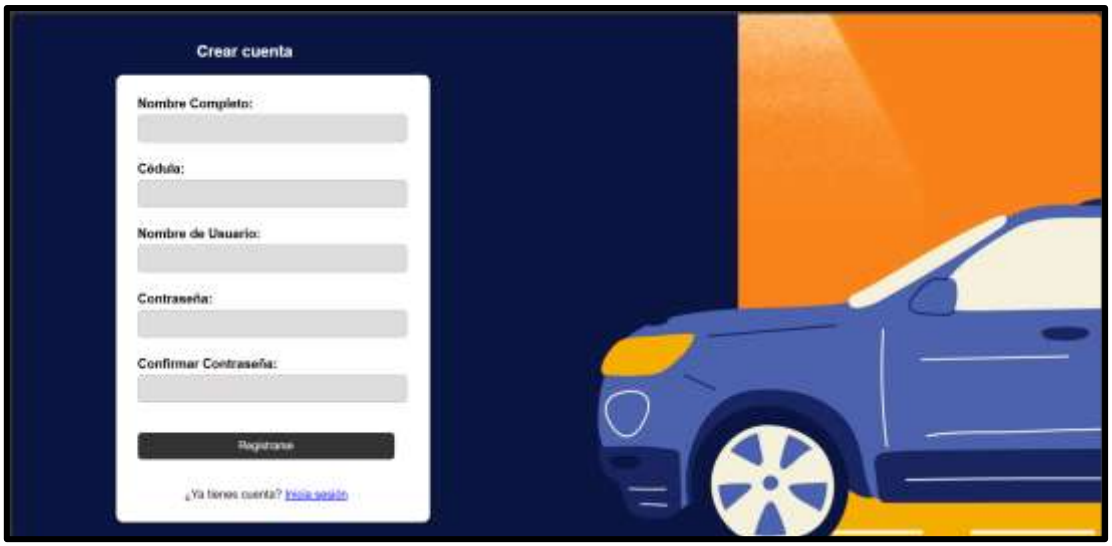

Figura 2:
Pantalla de registro

Nota. Captura de pantalla de la aplicación (2025)

- Una vez registrado, saldrá una alerta indicando que el usuario ha sido registrado correctamente, y después va a dirigir directamente a la pantalla principal para poder iniciar sesión con las credenciales registradas anteriormente. Damos clic en iniciar sesión y podremos ingresar al menú principal.

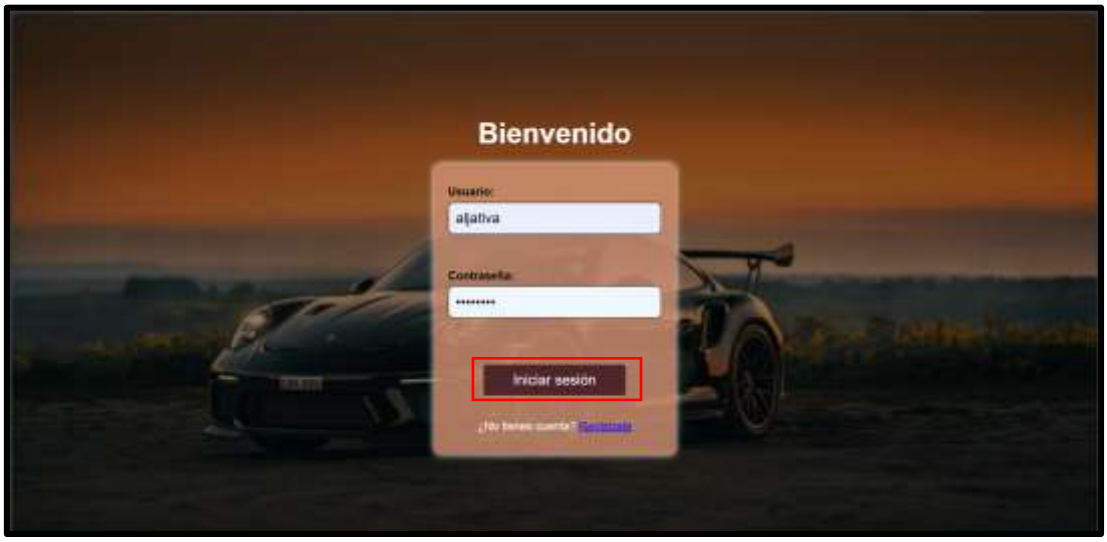

Figura 3:
Pantalla de Login, para iniciar sesión

Nota. Captura de pantalla de la aplicación (2025)

- Se mostrará el menú principal, para que el usuario tenga conocimiento de todas las funcionalidades que posee la página web

Figura 4:
Pantalla de inicio

Nota. Captura de pantalla de la aplicación (2025)

- Se dará clic en cualquiera de ellas, y se desplegará la respectiva sección

En la sección de clientes se podrá crear un nuevo cliente dando clic en el botón de "Agregar cliente"

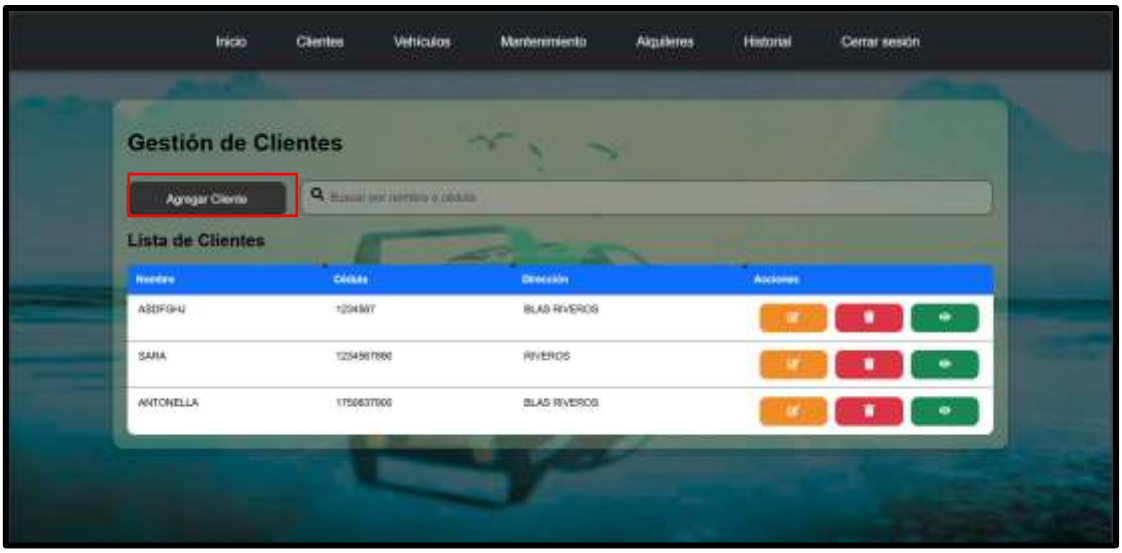

Figura 5:
Pantalla principal y botón para agregar clientes

Nota. Captura de pantalla de la aplicación (2025)

- Una vez dado clic en el botón, se mostrará un formulario para poder completar los datos del cliente, como nombre, cedula, teléfono, etc.

Figura 6:
Pantalla del formulario para registro de clientes

Nota. Captura de pantalla de la aplicación (2025)

- Cuando ya hayamos llenado los datos del cliente damos clic en guardar cliente, en donde el botón se encuentra al final del formulario.

Figura 7:
Botón de Guardar cliente

Nota. Captura de pantalla de la aplicación (2025)

- Cuando el cliente ya se encuentre registrado se mostrará en la tabla principal en donde ya están guardados todos los clientes.

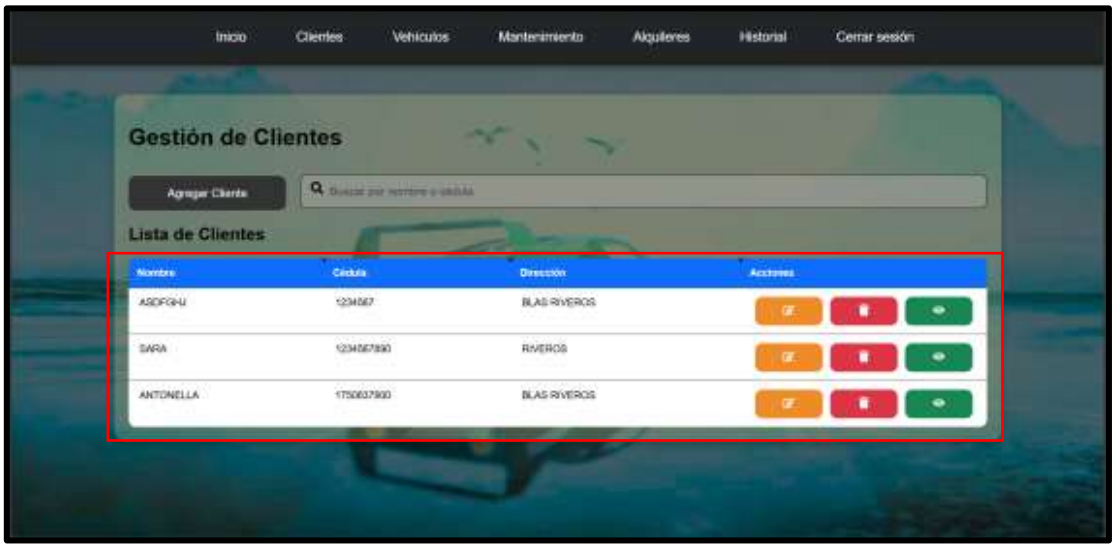

Figura 8:
Pantalla de la tabla en donde se guardó el cliente

Nota. Captura de pantalla de la aplicación (2025)

- En donde también está la disponibilidad de poder editar (botón amarillo), eliminar (botón rojo) y ver los detalles (botón verde) de cada cliente.

Figura 9:
Pantalla de los botones de editar, eliminar y ver detalles

Nota. Captura de pantalla de la aplicación (2025)

- En la sección de vehículos, registramos un nuevo vehículo dando clic en el botón de “Agregar Vehículo”.

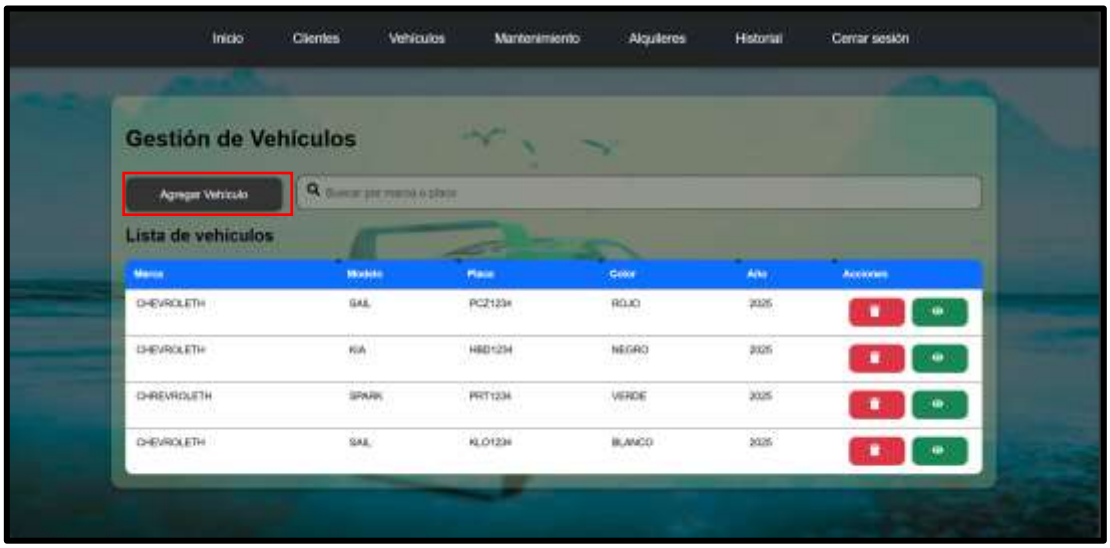

Figura 10:
Pantalla de la pantalla principal y del botón para agregar vehículos

Nota. Captura de pantalla de la aplicación (2025)

- Cuando damos clic en ese botón, se nos despliega un formulario para llenar los datos del respectivo vehículo, y cuando ya se complete el formulario lo guardamos dando clic en “Guardar Vehículo”.

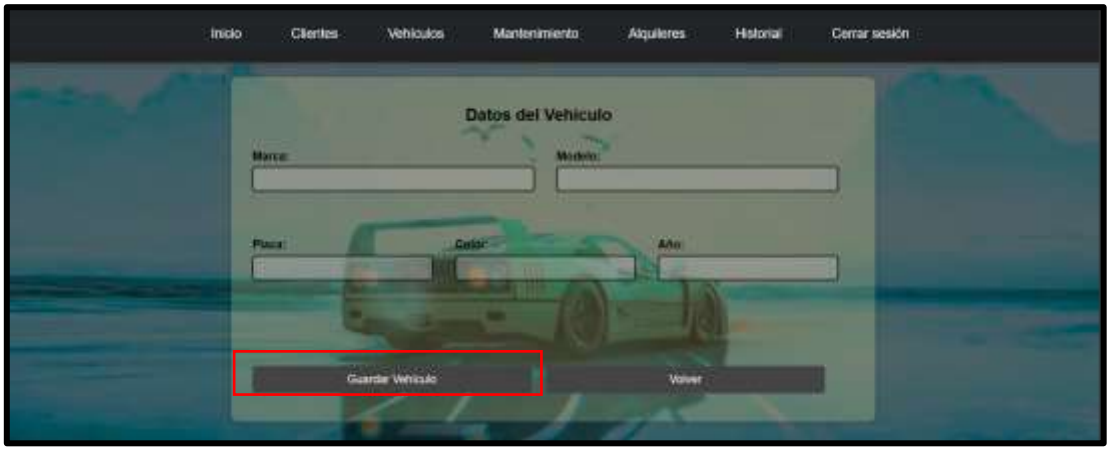

Figura 11:
Pantalla del formulario y del botón para guardar el vehiculo

Nota. Captura de pantalla de la aplicación (2025)

- Una vez guardado se podrá visualizar en la tabla principal de todos los autos guardados

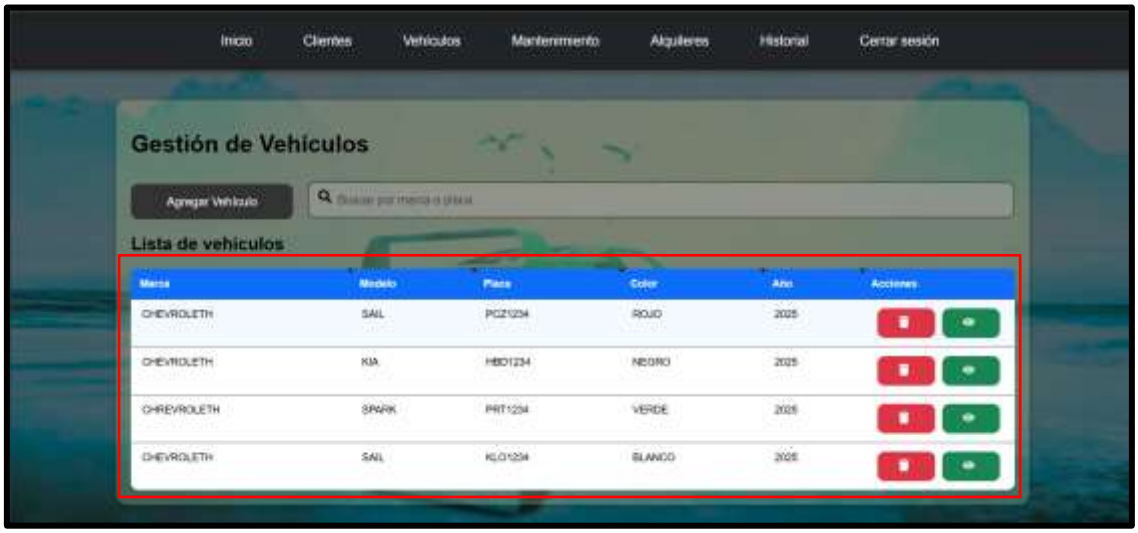

Figura 12:
Pantalla de la tabla en donde se guardó el vehículo

Nota. Captura de pantalla de la aplicación (2025)

- En donde también tenemos disponibilidad de poder eliminar (botón rojo) y poder ver los detalles (botón verde) de los autos

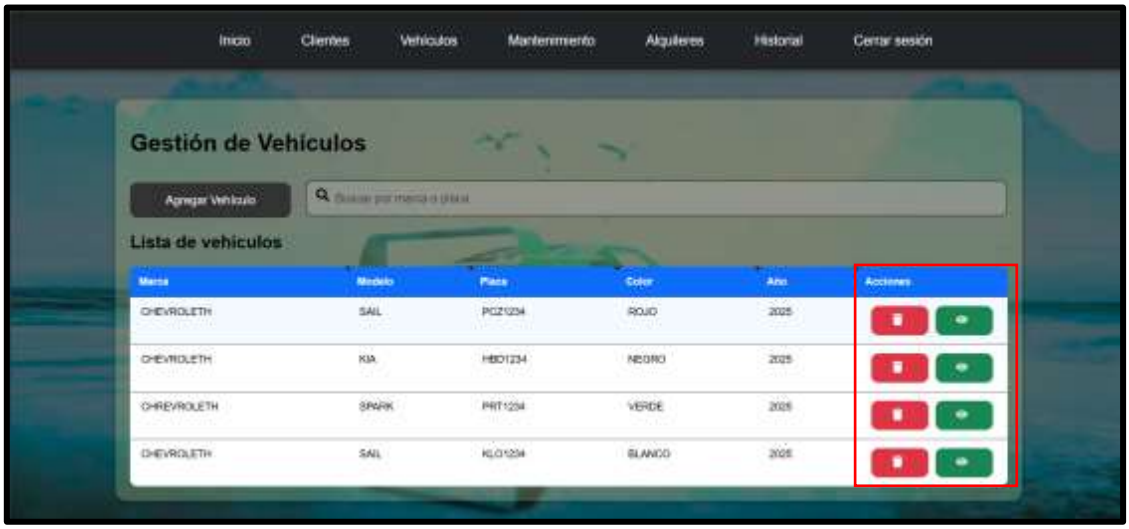

Figura 13:
Pantalla de la disponibilidad de los botones

Nota. Captura de pantalla de la aplicación (2025)

- En mantenimiento creamos un mantenimiento de un auto existente, los creamos dando clic en “ver detalles” de la sección de vehículos, escogemos el auto y damos clic en el botón ver detalles.

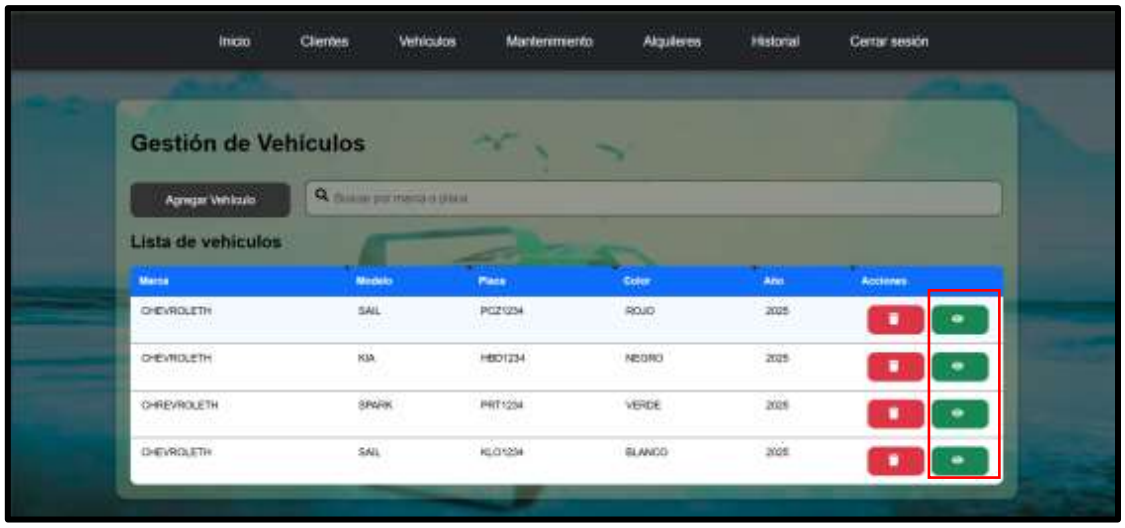

Figura 14:
Pantalla de la funcionalidad del botón ver detalles

Nota. Captura de pantalla de la aplicación (2025)

- Cuando damos clic en el botón verde, se nos despliega el mismo formulario, pero con los datos ya preestablecidos, en donde para ya asignar un alquiler damos clic en el botón de “asignar mantenimiento”.

Figura 15:
Pantalla para asignar un mantenimiento a ese vehículo

Nota. Captura de pantalla de la aplicación (2025)

- Cuando demos clic en el botón se redigira al formulario de mantenimiento para poder registrar el nuevo mantenimiento de ese auto.

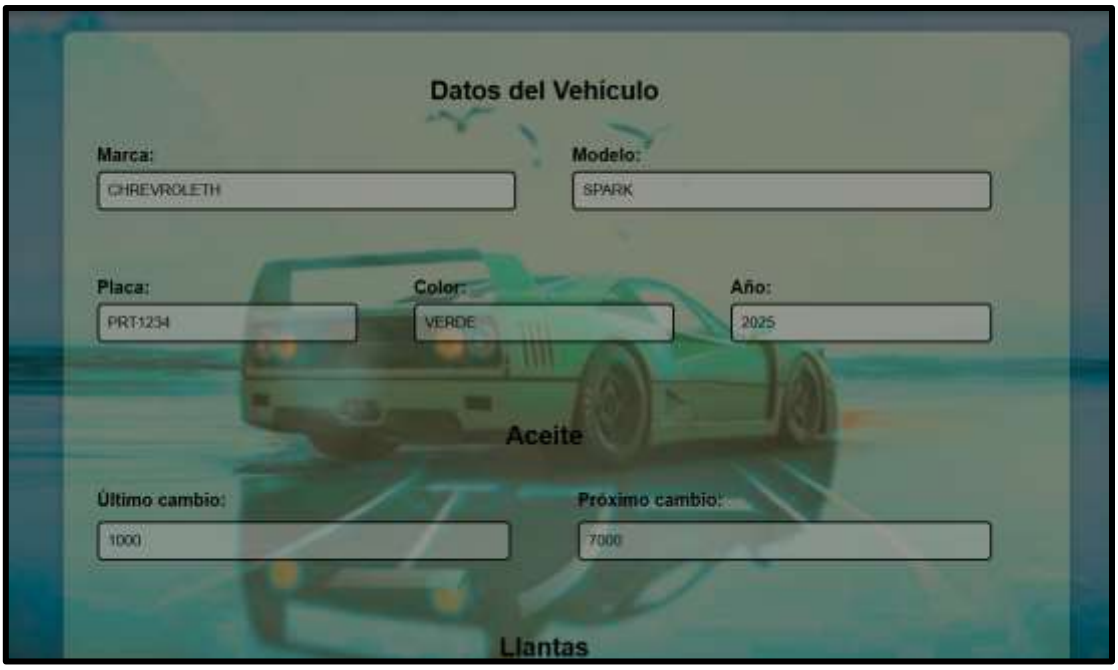

Figura 16:
Pantalla del formulario del mantenimiento

Nota. Captura de pantalla de la aplicación (2025)

- Una vez completado todos los datos del mantenimiento, se lo podrá guardar dando clic en el botón de “Guardar mantenimiento”, ubicado al final del formulario.

Figura 17:
Pantalla del botón para guardar el mantenimiento

Nota. Captura de pantalla de la aplicación (2025)

- Cuando ya se haya guardado el manteniendo lo podremos observar en la tabla principal

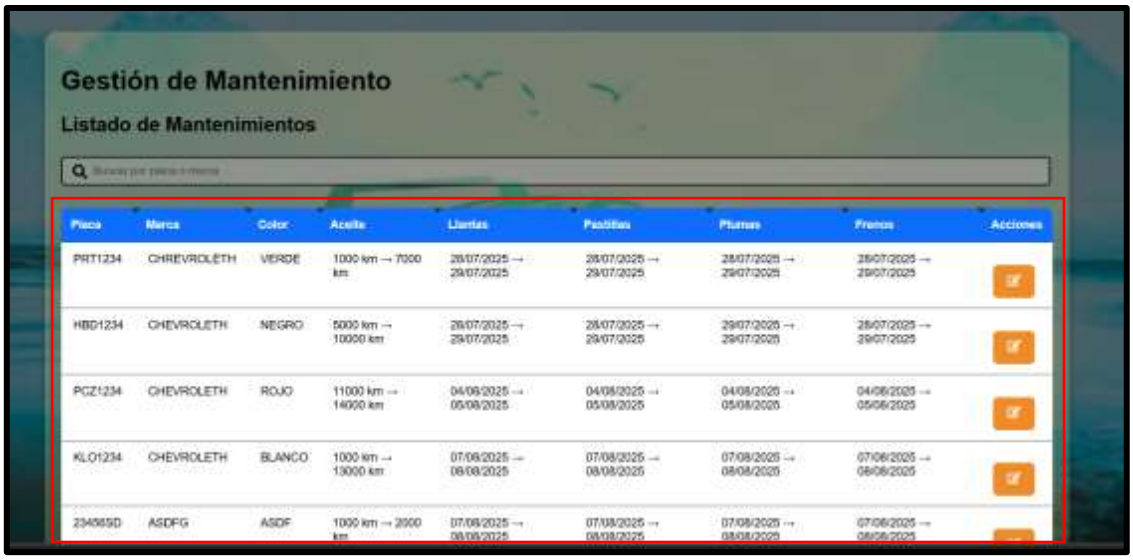

Figura 18:
Pantalla de la tabla y página principal para ver los mantenimientos registrados

Nota. Captura de pantalla de la aplicación (2025)

- Donde se podrá editar (botón amarillo) el mantenimiento cuando ya sea hora de cambiar algún estado critico

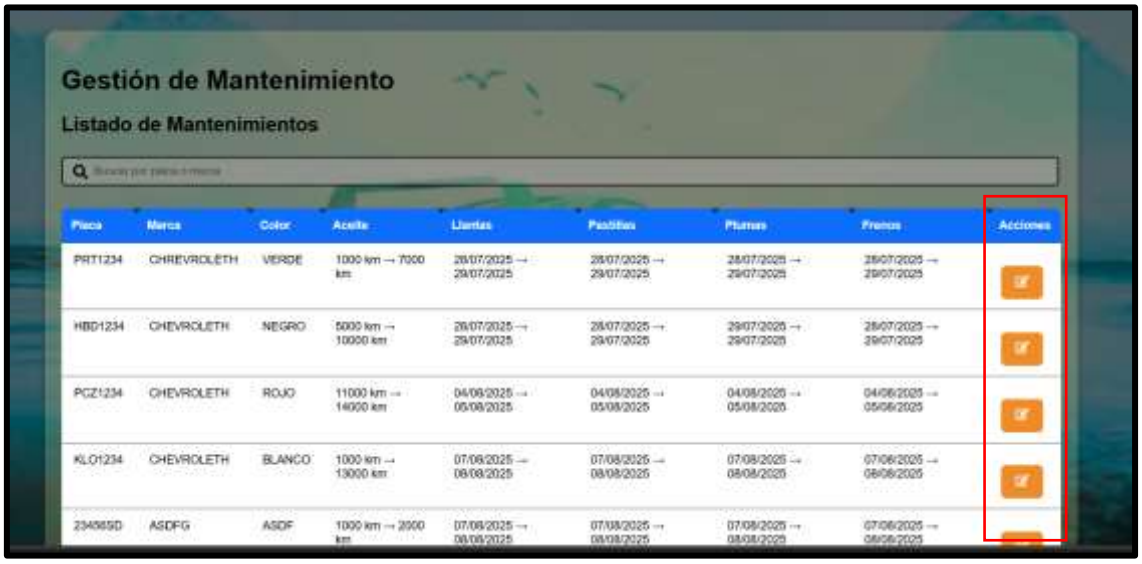

Figura 19:
Pantalla de la disponibilidad del botón de editar

Nota. Captura de pantalla de la aplicación (2025)

- Para asignar un alquiler, nos vamos a la sección de clientes dando clic en el botón de ver detalles (botón verde).

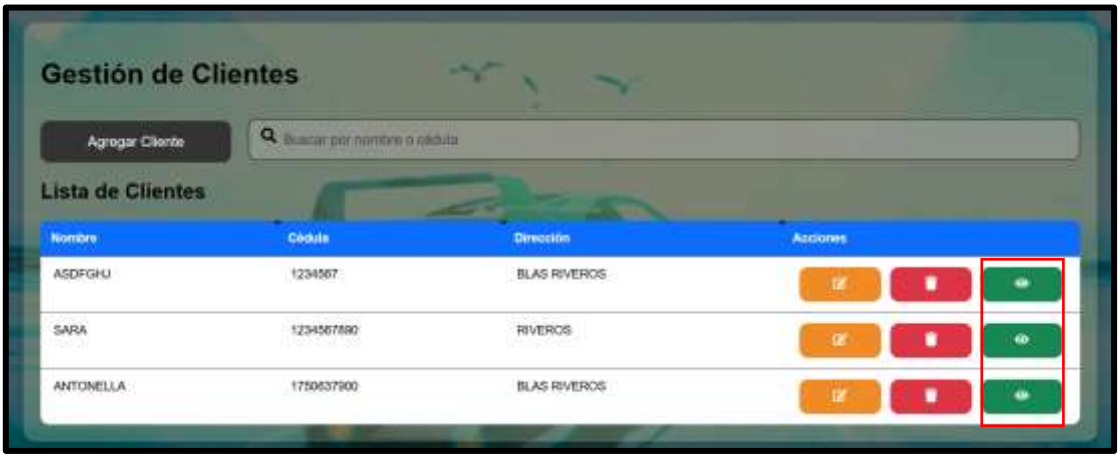

Figura 20:
Pantalla de la funcionalidad del botón de ver detalles

Nota. Captura de pantalla de la aplicación (2025)

- Y se nos desplegara un formulario con los datos preestablecidos de los clientes, en donde para asignar el alquiler le damos clic en el botón “Asignar alquiler”, ubicado al final del formulario.

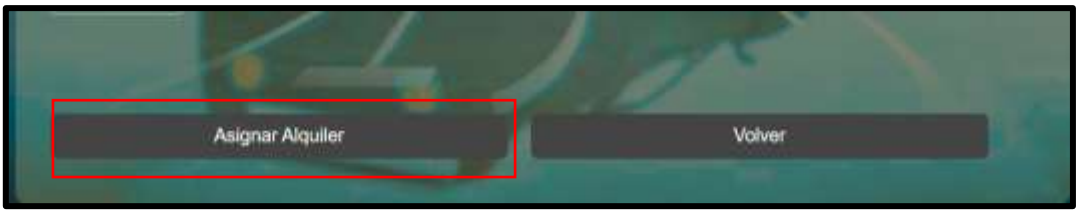

Figura 21:
Pantalla del botón para asignar un alquiler

Nota. Captura de pantalla de la aplicación (2025)

- Se dirigirá directo al formulario para llenar los datos del alquiler

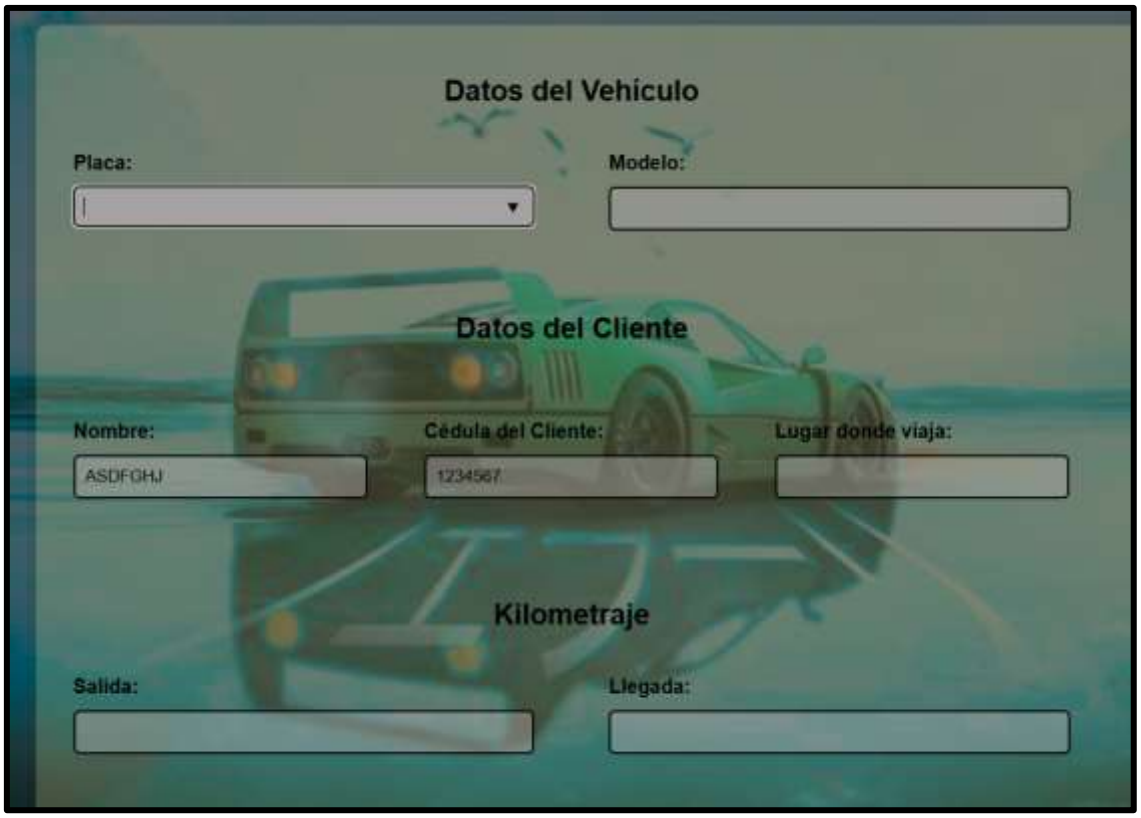

Figura 22:
Pantalla del formulario de alquiler

Nota. Captura de pantalla de la aplicación (2025)

- Y guardamos el alquiler, dando clic en el botón de “Guardar Alquiler”, ubicado al final del formulario

Figura 23:
Pantalla del botón para guardar el alquiler

Nota. Captura de pantalla de la aplicación (2025)

- Y se podrá visualizar el alquiler en la tabla principal de la sección de alquileres

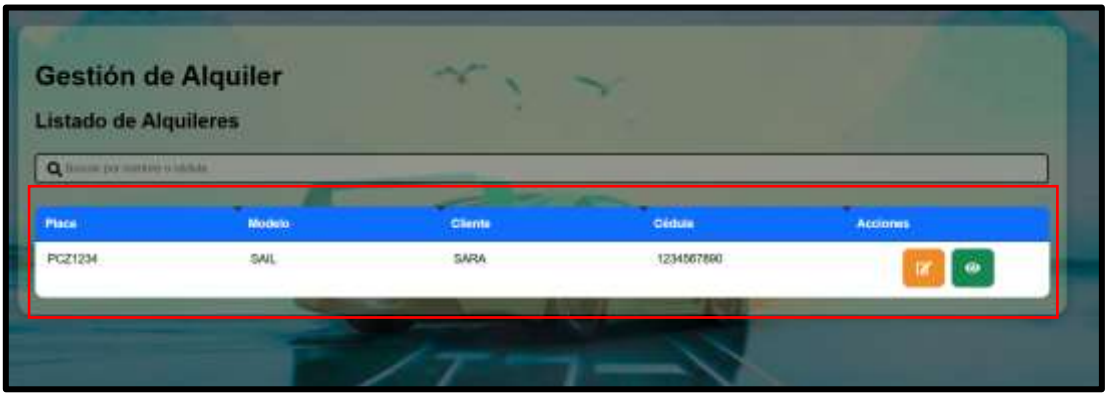

Figura 24:
Pantalla de la tabla y página principal de alquiler

Nota. Captura de pantalla de la aplicación (2025)

- En donde se dispone de los botones de editar y ver detalles de los alquileres

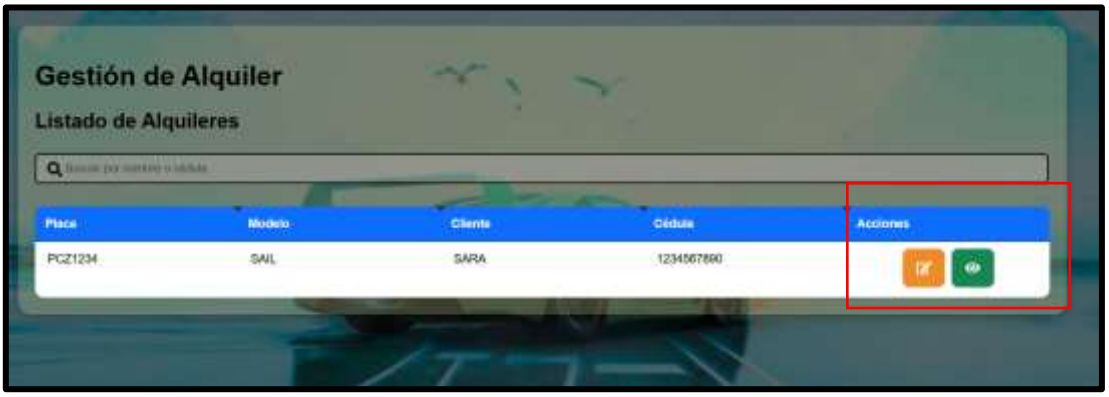

Figura 25:
Pantalla de la disponibilidad de los botones de alquiler

Nota. Captura de pantalla de la aplicación (2025)

- Cuando damos clic en el botón de editar el alquiler, nos lleva al formulario, pero ahora completamos el campo de kilometraje de llegada del auto, calificación y descripción del cliente.

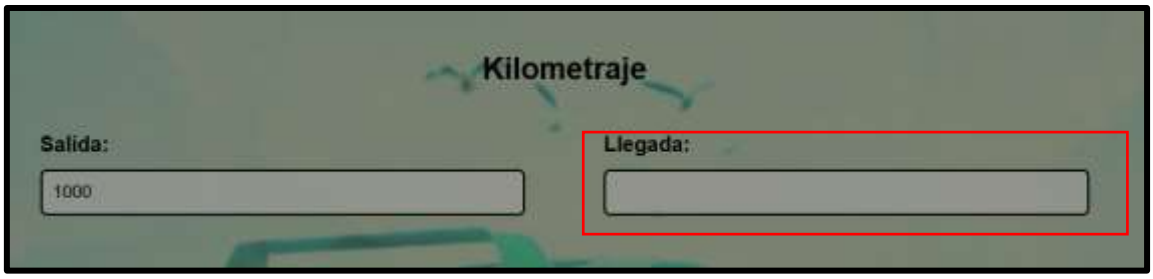

Figura 26:
Pantalla del campo de kilometraje

Nota. Captura de pantalla de la aplicación (2025)

Figura 26:
Pantalla de la calificación del cliente

Nota. Captura de pantalla de la aplicación (2025)

- Una vez que completamos esos datos, guardamos la información dando clic en el botón de “actualizar alquiler” al final del formulario.

Figura 27:
Pantalla del botón de actualizar el alquiler

Nota. Captura de pantalla de la aplicación (2025)

- Y cuando ya se complete el alquiler ya no aparecerá en la sección de alquileres, sino en la sección del historial, en donde se podrá visualizar la temática de los colores en base al número de estrellas del cliente.

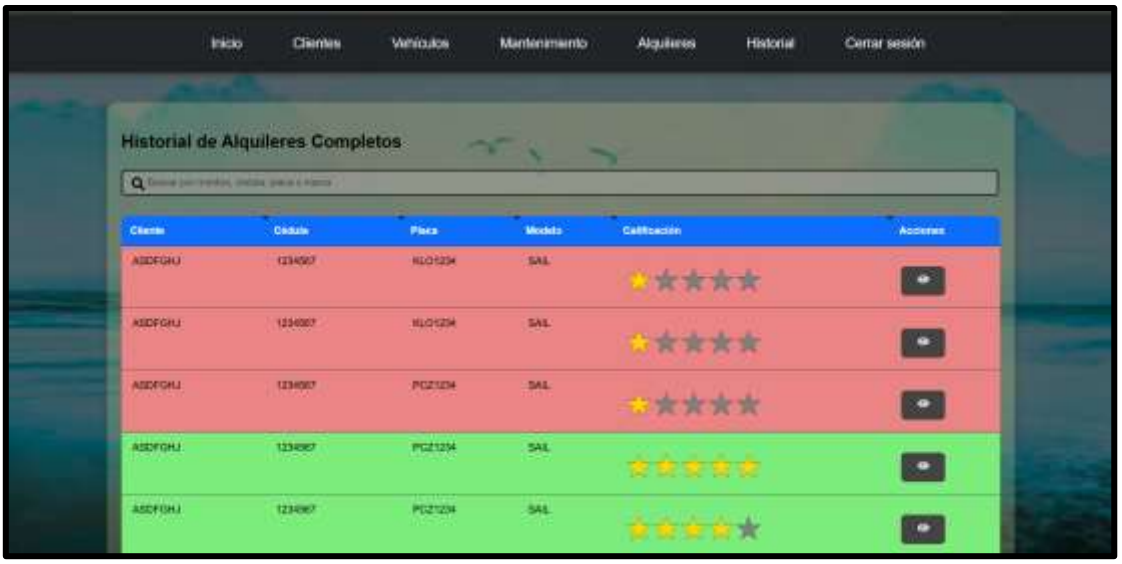

Figura 28:
Pantalla del historial de alquileres completos

Nota. Captura de pantalla de la aplicación (2025)

Anexo B. Validaciones

- Validación de completar los campos

Figura 29:
Pantalla de la validación de campo obligatorio

Nota. Captura de pantalla de la aplicación (2025)

- Cedula, teléfono y celular incompletos

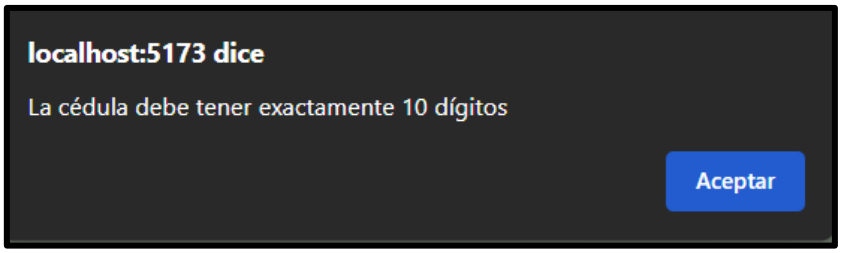

Figura 30:
Pantalla de la validación de cedula, teléfono y celular incompletos

Nota. Captura de pantalla de la aplicación (2025)

- Si un cliente ya tiene registrado un alquiler

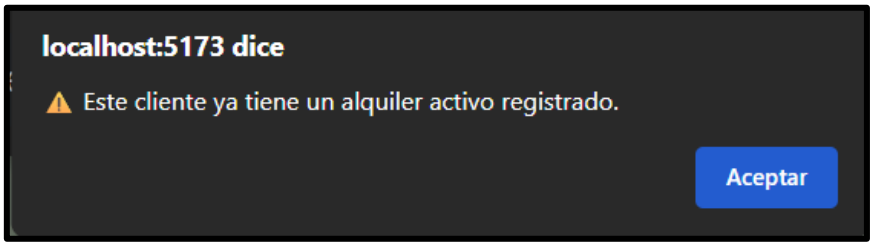

Figura 31:
Pantalla de la validación de volver a asignarle un alquiler al cliente

Nota. Captura de pantalla de la aplicación (2025)

- Si un vehículo ya tiene registrado un mantenimiento

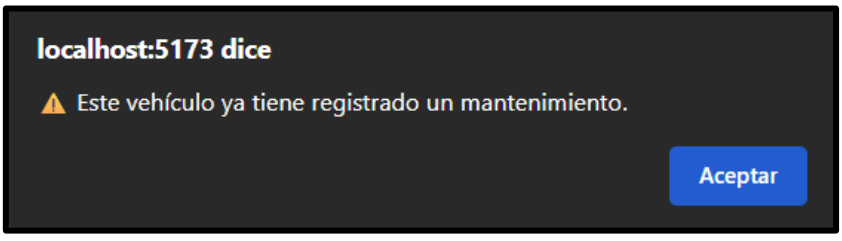

Figura 31:
Pantalla de la validación de volver a asignarle un mantenimiento a un vehículo

Nota. Captura de pantalla de la aplicación (2025)

- Un campo no actualizado del mantenimiento

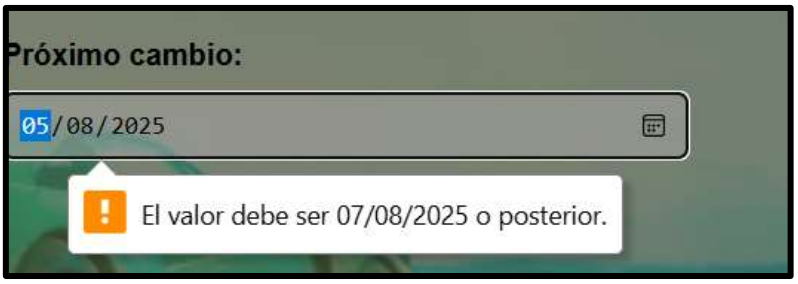

Figura 32:
Pantalla de la validación de un campo no actualizado

Nota. Captura de pantalla de la aplicación (2025)

- Si un vehículo necesita cambios, pastillas, líquido de frenos, etc.

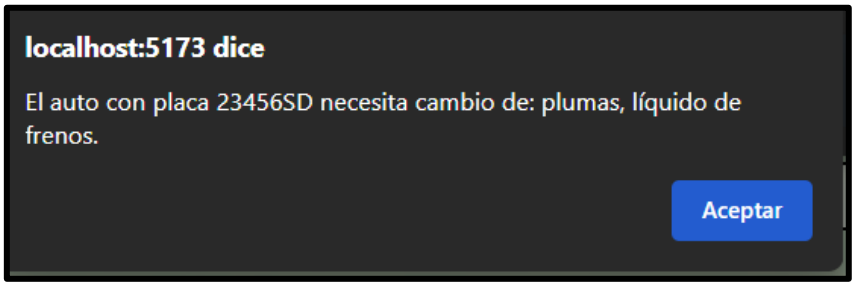

Figura 33:
Pantalla de la validación de realizar mantenimiento

Nota. Captura de pantalla de la aplicación (2025)

- Al editar el kilometraje de llegada del alquiler y si coincide con el cambio de aceite que está registrado en el mantenimiento

Figura 34:
Pantalla de la validación de registrar el valor del kilometraje de llegada

Nota. Captura de pantalla de la aplicación (2025)

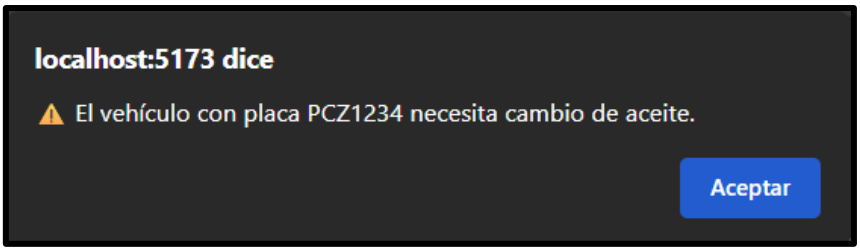

Figura 35:
Pantalla de la validación de coincidencia del valor de cambio de aceite y del kilometraje de llegada

Nota. Captura de pantalla de la aplicación (2025)

- Redirigirá al formulario del mantenimiento para cambiarle, y no dejara guardar ni volver si aún no se le ha cambiado el valor del cambio de aceite, pastillas, etc. del vehículo.

- Además de lo antes mencionado también se dispone de un menú en la parte superior paraqué pueda navegar fácilmente, que contiene los nombres de las secciones y adicional un botón para cerrar la sesión

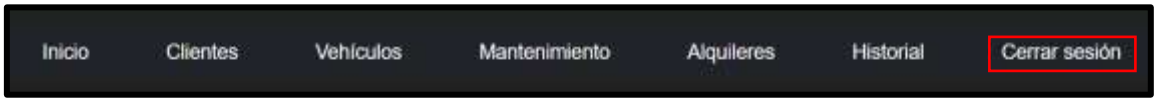

Figura 36:
Pantalla del menú principal en la parte superior

Nota. Captura de pantalla de la aplicación (2025)