Informe Rent-a-Car

Ignacio Gómez Simonsohn, Gonzalo Vazquez Acevedo, Juan Unanue, Joaquin Abbona Lerena.

# Misión y Alcance del Proyecto

El presente proyecto tiene como objetivo el desarrollo de un sistema de gestión integral para la compra y alquiler de vehículos, orientado a una empresa ficticia radicada en la ciudad de Mar del Plata. A pesar de su carácter ficticio, el sistema ha sido diseñado con la suficiente flexibilidad y escalabilidad como para ser implementado en una empresa real del sector, adaptándose mediante personalizaciones específicas según los requerimientos del cliente.

La misión del proyecto es optimizar y simplificar los procesos operativos relacionados con la venta y el alquiler de vehículos, tanto desde la perspectiva del usuario final como de los administradores del sistema. De esta manera, se busca mejorar la experiencia del cliente y, al mismo tiempo, aumentar la eficiencia de la gestión interna de la empresa.

## Alcance Funcional del Sistema

El sistema contempla funcionalidades para dos tipos principales de usuarios: clientes y administradores/empleados.

## Cliente:

- Acceso a una interfaz intuitiva donde puede visualizar el catálogo de vehículos disponibles para alquiler o compra.  
- Consulta de su historial de reservas y operaciones.  
- Posibilidad de cancelar una reserva activa.  
- Acceso a diversas opciones de pago integradas en la plataforma.

## Administrador/Empleado:

- Gestión del stock de vehículos (alta, baja y modificación).  
- Administración de los datos de clientes registrados.  
- Control y mantenimiento de la información relacionada con las distintas sucursales de la empresa.

## Proyección y Adaptabilidad

El sistema ha sido concebido con una arquitectura modular y adaptable, lo que permite su futura expansión e integración con otros servicios o plataformas (por ejemplo, pasarelas de pago externas, sistemas de geolocalización de vehículos, etc.). Esto también habilita su personalización para ajustarse a distintos modelos de negocio dentro del rubro automotor.

En conclusión, el proyecto no sólo responde a una necesidad académica, sino que también representa una solución tecnológica viable para el mercado real, orientada a mejorar los procesos administrativos y la experiencia del cliente en empresas dedicadas a la comercialización y alquiler de vehículos.

# Explicación de la Arquitectura

El proyecto ha sido desarrollado siguiendo una arquitectura basada en Angular para el front-end. Angular fue seleccionado por su estructura modular, su capacidad para construir aplicaciones web reactivas, y su compatibilidad con servicios RESTful, lo cual lo convierte en una herramienta ideal para construir interfaces dinámicas y robustas.

El sistema se apoya en tres componentes principales:

## 1. Front-end (Angular):

Toda la interfaz de usuario ha sido implementada utilizando Angular. A través de sus componentes, servicios e inyección de dependencias, se construyó una experiencia fluida tanto para clientes como para administradores. La comunicación con los servicios de back-end se realiza mediante HTTPClient y suscripciones a observables, permitiendo manejar eficientemente las respuestas asincrónicas.

## 2. Back-end simulado con JSON Server:

Para simular la funcionalidad de una base de datos real, se utilizó JSON Server, una herramienta ligera y eficiente para prototipado rápido de APIs REST. Esto permitió almacenar, consultar y modificar datos relacionados con vehículos, usuarios, reservas y sucursales. JSON Server se ejecuta de manera local y expone una API REST que es consumida directamente por el front-end en Angular.

## 3. API externa integrada:

Además del back-end simulado, se integró una API externa ([**https://distancematrix.ai/es**](https://distancematrix.ai/es)) como parte del proyecto. Esta se utilizó para complementar una funcionalidad específica (para calcular la distancia del acarreo del vehículo). La integración se realizó mediante servicios HTTP dentro de Angular, utilizando claves de autenticación si fuera necesario.

## Arquitectura general:

La arquitectura general del sistema puede clasificarse como una SPA (Single Page Application) apoyada en servicios distribuidos. El cliente (navegador) ejecuta la aplicación Angular, que se comunica con:  
- Un servidor local JSON Server (simulando una base de datos persistente)  
- Una API externa real, consumida mediante HTTP  
  
Esto permite modularidad, separación de responsabilidades y una clara escalabilidad hacia un entorno de producción más complejo si fuera necesario (por ejemplo, reemplazando JSON Server por una API real en Node.js, Java o Python).

Esta estructura no solo cumple con los requisitos académicos, sino que además sienta las bases para una evolución del sistema hacia un entorno real de despliegue con servidores reales y bases de datos persistentes.

# Especificación de Requisitos

# Requisitos Funcionales

## RF-1 - Gestión de Vehículos

Descripción del requisito:

Contexto: Este requisito forma parte del módulo de administración del sistema. Se relaciona con los requisitos de gestión de sucursales y clientes, ya que los vehículos pueden estar asignados a diferentes ubicaciones y alquilados por diversos usuarios.  
  
Objetivo: Permitir que los administradores puedan dar de alta, baja o editar información de vehículos disponibles para alquiler o venta.  
  
Entradas y salidas:  
- Entrada: datos del vehículo (marca, modelo, año, tipo, disponibilidad, precio, URL imagen).  
- Salida: confirmación de alta/modificación/baja y actualización del listado de vehículos.  
  
Proceso:  
- Solo accesible por administradores autenticados.  
- El sistema valida los datos ingresados.  
- Se actualiza la base de datos local (JSON Server).  
- No se permite eliminar vehículos con operaciones activas.

Prioridad: Alta

## RF-2 - Compra y alquiler de vehículos

Descripción del requisito:

Contexto: Requisito principal del cliente. Se conecta con el historial de operaciones, el sistema de pagos y la disponibilidad de vehículos.  
  
Objetivo: Permitir a los usuarios seleccionar un vehículo disponible y realizar una compra o alquiler.  
  
Entradas y salidas:  
- Entrada: selección de vehículo, tipo de operación (compra o alquiler), fecha, método de pago.  
- Salida: confirmación de operación, actualización del estado del vehículo, generación de registro en el historial.  
  
Proceso:  
- Solo clientes autenticados.  
- El sistema valida disponibilidad y fechas.  
- Se permite pagar en efectivo o tarjeta.  
- En caso de alquiler, se registra la duración y se bloquea la disponibilidad.

Prioridad: Alta

## RF-3 - Visualización y cancelación de historial

Descripción del requisito:

Contexto: Se conecta con la base de operaciones del usuario y las funcionalidades de cancelación y control de disponibilidad.  
  
Objetivo: Que el cliente pueda ver sus operaciones pasadas y cancelar reservas activas si están dentro del tiempo permitido.  
  
Entradas y salidas:  
- Entrada: autenticación del usuario.  
- Salida: listado de operaciones y botones de cancelación si aplica.  
  
Proceso:  
- Solo clientes autenticados.  
- Se listan operaciones activas y finalizadas.  
- Si la reserva aún no comenzó, puede cancelarse.

Prioridad: Media

## RF-4 - Gestión de sucursales y usuarios

Descripción del requisito:

Contexto: Requisito exclusivo del panel de administración. Se conecta con vehículos (ubicación), empleados y clientes.  
  
Objetivo: Permitir la gestión de los datos relacionados con sucursales, empleados (administradores) y clientes.  
  
Entradas y salidas:  
- Entrada: formularios con datos de usuario o sucursal.  
- Salida: confirmación de creación/modificación/borrado.  
  
Proceso:  
- Solo accesible por administradores autenticados.  
- CRUD completo de usuarios y sucursales.

Prioridad: Alta

## RF-5 - Cierre de sesión

Descripción del requisito:

Contexto: Común a ambos tipos de usuario.  
  
Objetivo: Permitir a los usuarios cerrar su sesión de forma segura.  
  
Entradas y salidas:  
- Entrada: acción de logout.  
- Salida: redirección al login.  
  
Proceso:  
- Disponible para cualquier sesión activa.  
- Elimina token de autenticación o sesión local.

Prioridad: Alta

## ****RF-6 - Cálculo de distancia para acarreo****

Descripción del requisito:

Contexto: Este requisito se relaciona directamente con el proceso de alquiler de vehículos. Se activa cuando un cliente elige alquilar un auto y requiere la entrega del mismo en una ubicación determinada. Para ello, se utiliza una API externa que permite calcular la distancia entre la sucursal de retiro y la sucursal de devolución.  
  
Objetivo: Determinar automáticamente la distancia en kilómetros entre la sucursal de retiro y la ubicación de devolución para calcular el costo adicional por acarreo/logística. Este valor se suma al total del alquiler antes de proceder al pago.

Entradas y salidas:  
- Entrada dirección de origen (o coordenadas) y dirección de la sucursal destino.  
- Salida: distancia en kilómetros y costo adicional por acarreo.  
  
Proceso:  
- El usuario selecciona la sucursal de entrega del vehículo

- El sistema consulta la API externa con ambas direcciones.

-Se calcula la distancia en tiempo real y se aplica una tarifa.

-El costo total se actualiza y se muestra al cliente antes de confirmar el pago.

Prioridad: Media

Notas adicionales: Este requisito depende del correcto funcionamiento de la API externa

## RF-7 – Registro de nuevos clientes

Descripción del requisito:

Contexto: Este requisito forma parte del flujo de autenticación del sistema. Está vinculado con los procesos de inicio de sesión y con la gestión de clientes por parte del administrador, ya que los datos ingresados durante el registro alimentan la base de usuarios del sistema.  
  
Objetivo: Permitir que una persona que no tiene cuenta en el sistema pueda registrarse como cliente mediante un formulario. Esto habilita su acceso a funcionalidades como alquilar o comprar vehículos, ver historial, editar perfil, etc.

Entradas y salidas:

-Entradas: nombre, apellido, email, contraseña (y confirmación de contraseña), teléfono, dirección, fecha de nacimiento.

-Salida: mensaje de éxito o error; creación del cliente en la base de datos.

Proceso:

- El usuario accede a la pantalla de registro desde la página de inicio.

- Completa el formulario con los datos solicitados.

- El sistema valida la información (formato del email, contraseñas coincidentes, email único).

- Si todo es válido, se guarda el nuevo cliente en la base de datos (JSON Server).

Prioridad: Alta

# Requisitos No Funcionales

## RNF-1 - Usabilidad

Descripción del requisito:

Contexto: Aplica a toda la aplicación.  
  
Objetivo: Asegurar que el sistema sea intuitivo, accesible y fácil de usar tanto para administradores como para clientes.  
  
Detalles:  
- Diseño responsivo.  
- Navegación clara y consistente.  
- Mensajes de error amigables y útiles.

Prioridad: Alta

## RNF-2 - Rendimiento

Descripción del requisito:

Contexto: General a toda la aplicación Angular.  
  
Objetivo: Garantizar tiempos de respuesta rápidos.  
  
Detalles:  
- Consultas optimizadas.  
- Uso eficiente de observables y servicios.  
- Carga perezosa (lazy loading) de componentes pesados.

Prioridad: Media

## RNF-3 - Escalabilidad

Descripción del requisito:

Contexto: Estructura del proyecto.  
  
Objetivo: Permitir que el sistema pueda evolucionar con facilidad.  
  
Detalles:  
- Arquitectura modular.  
- Servicios desacoplados.  
- Separación clara entre capas de datos, lógica y presentación.

Prioridad: Media

# Diagrama de entidades

## Entidades

**Cliente**

* id (PK)
* nombre
* apellido
* email
* contraseña
* teléfono
* fecha de nacimiento

**Administrador**

* id (PK)
* nombre completo
* username
* email
* contraseña

**Vehículo**

* id (PK)
* idSucursal (FK)
* marca
* modelo
* tipo
* año
* descripción
* precio
* disponible venta (booleano)
* disponible alquiler (booleano)
* imageURL

**Reserva**

* id (PK)
* idCliente (FK)
* idVehiculo (FK)
* fechaInicio
* fechaFin
* origen sucursal
* destino sucursal

**Compra**

* id (PK)
* idCliente (FK)
* idVehiculo (FK)
* precio

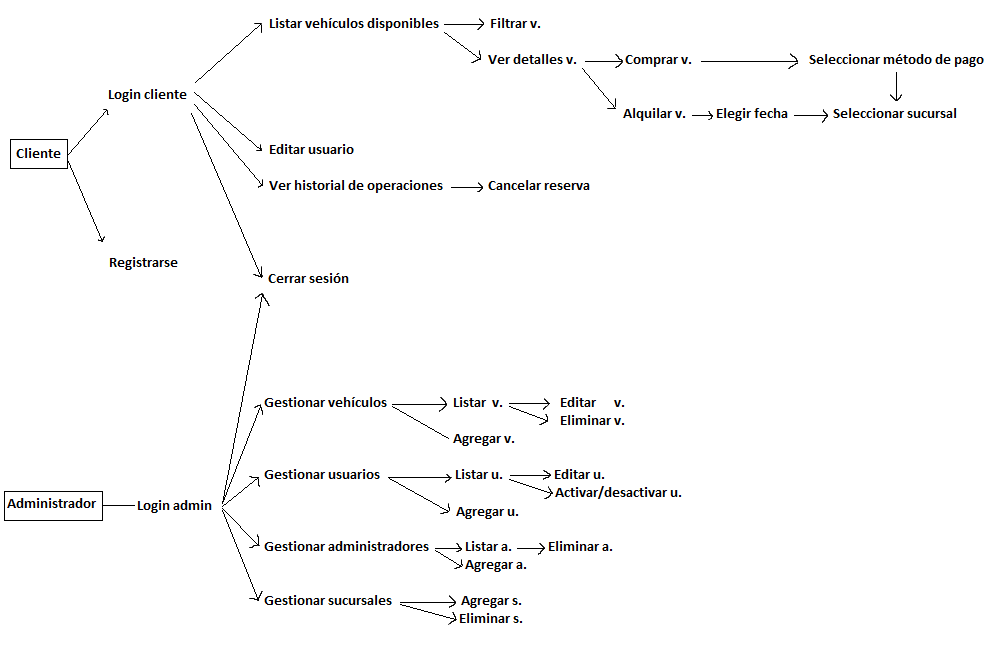
**Sucursal**

* id (PK)
* nombre

## Relaciones

* Un **Cliente** puede tener muchas **Reservas** (1:N)
* Un **Cliente** puede tener muchas **Compras** (1:N)
* Un **Vehículo** puede tener **una sola Reserva o una sola Compra** (relación 1:1 con ambas entidades, exclusivas entre sí)
* Una **Sucursal** puede tener muchos **Vehículos** (1:N)

# Diagrama de casos de uso



# URL del tablero Jira:

<https://juanunanue99.atlassian.net/jira/software/projects/TPFINAL/boards/3>

# URL del repositorio:

<https://github.com/JuanUnanue/Rent-a-Car>