

# UNICAR

Integrantes: Javier Falcon Real, Moisés J. Pérez  
García, Miguel Adam Gavira Aabed



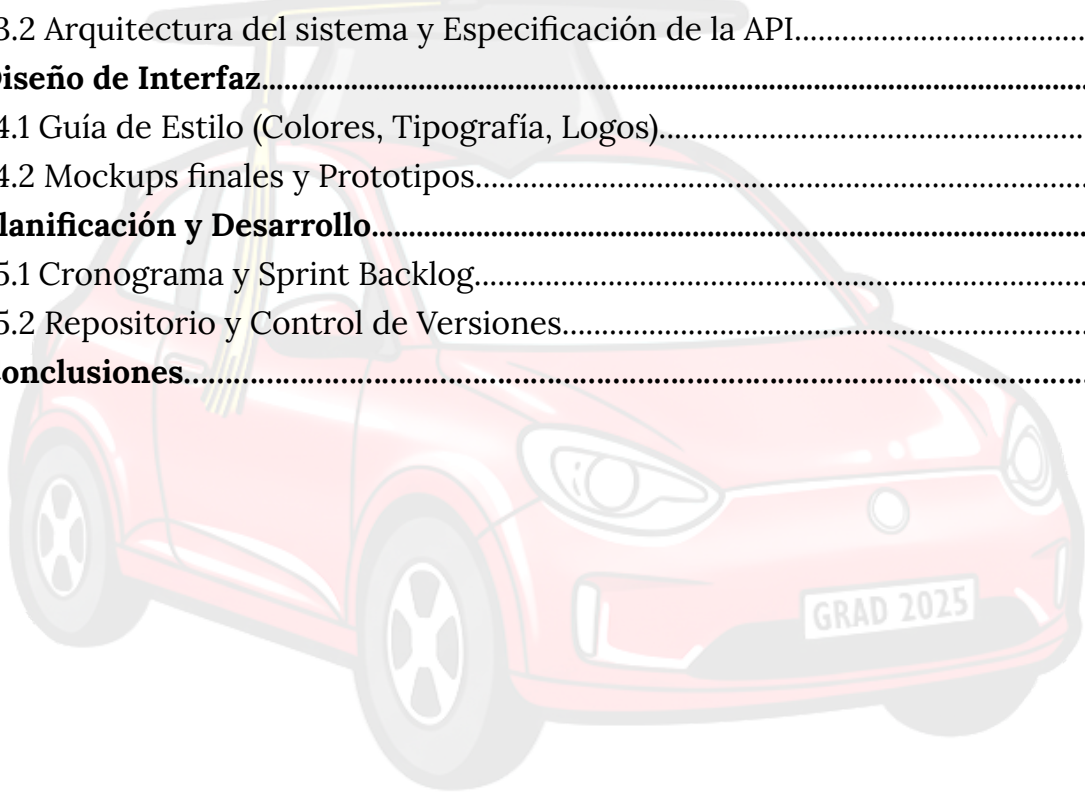
**Curso:** 2ºDesarrollo de Aplicaciones Web.

**Módulo:** Proyecto integrado, grupo nº3.

**Fecha:** 28 de Noviembre de 2025

# ÍNDICE

<b>1. Introducción y Objetivos.....</b>	<b>4</b>
1.1 Descripción del programa y la solución.....	4
1.2 Público objetivo y Tecnologías elegidas.....	4
<b>2. Diseño Funcional.....</b>	<b>5</b>
2.1 Requisitos del sistema.....	5
2.2 Mapa del sitio.....	6
<b>3. Diseño Técnico.....</b>	<b>7</b>
3.1 Diagrama Entidad-Relación (E-R) y Tablas de Base de Datos.....	7
3.2 Arquitectura del sistema y Especificación de la API.....	8
<b>4. Diseño de Interfaz.....</b>	<b>11</b>
4.1 Guía de Estilo (Colores, Tipografía, Logos).....	11
4.2 Mockups finales y Prototipos.....	12
<b>5. Planificación y Desarrollo.....</b>	<b>13</b>
5.1 Cronograma y Sprint Backlog.....	13
5.2 Repositorio y Control de Versiones.....	13
<b>6. Conclusiones.....</b>	<b>14</b>



## Resumen Ejecutivo

**UniCar** es una aplicación web de movilidad compartida diseñada específicamente para la comunidad universitaria de la provincia de Sevilla. Su objetivo principal es ofrecer una solución eficiente y económica al desafío diario del transporte entre los campus universitarios y los pueblos circundantes.

El proyecto aborda una necesidad crítica identificada en nuestro **Público Objetivo** (estudiantes de 18 a 30 años): la reducción de gastos y tiempo invertido en desplazamientos. Al facilitar la conexión directa entre conductores con plazas disponibles y estudiantes pasajeros con destinos comunes, UniCar cumple su **Propuesta Única de Valor (PUV)**: convertirse en el medio de transporte preferente y más accesible, permitiendo a los usuarios "ahorrarse horas infinitas en la parada del autobús". El foco en la provincia de Sevilla y el entorno universitario garantiza una alta relevancia, fomentando la colaboración y creando un ecosistema de confianza.

Técnicamente, UniCar se construye sobre una arquitectura robusta y escalable bajo un patrón Cliente-Servidor Desacoplado. El **Frontend** se desarrolla mediante el *framework* **Angular**, creando una Single Page Application (SPA) modular y moderna. El **Backend** está gestionado por **Spring Boot (Java)**, que proporciona la lógica de negocio, la seguridad y la gestión de una **Base de Datos MySQL** para asegurar la integridad de datos clave (usuarios, viajes, vehículos). Las funcionalidades primarias incluyen el registro de usuarios y vehículos, la publicación y búsqueda de viajes, un sistema de valoraciones, y un crucial **Chat en Tiempo Real** para la comunicación. Esta arquitectura, respaldada por una API REST bien definida, garantiza la escalabilidad y el enfoque claro en resolver el problema del transporte universitario. UniCar es la solución tecnológica que optimiza los recursos de la comunidad estudiantil.

# 1. Introducción y Objetivos

## 1.1 Descripción del programa y la solución

El proyecto consiste en el desarrollo de una aplicación web denominada "UniCar", diseñada para facilitar la movilidad compartida. La arquitectura propuesta permitirá a los usuarios publicar y buscar viajes, gestionando la interacción entre conductores y pasajeros.

La solución aborda la necesidad de transporte mediante una plataforma que centraliza la oferta y demanda de plazas en vehículos privados. El sistema está diseñado para permitir a los usuarios registrarse, iniciar sesión, gestionar un perfil, registrar vehículos y administrar viajes (creación, edición y búsqueda). Además, el modelo de datos contempla funcionalidades de seguridad y confianza como un sistema de valoraciones (estrellas) y reportes de usuarios y viajes.

## 1.2 Público objetivo y Tecnologías elegidas

**Público Objetivo** El sistema está orientado a usuarios que requieren desplazarse entre campus universitarios y pueblos, tal como sugieren los campos de búsqueda "Campus" y "Pueblo" definidos en el diseño de la interfaz, fomentando el uso compartido de vehículos.

**Tecnologías Elegidas** Para el desarrollo del sistema se ha seleccionado una arquitectura moderna y escalable, ajustada durante la fase de configuración del entorno final:

- **Frontend (Cliente):** Se utiliza el framework **Angular**. Esta elección facilita la creación de una *Single Page Application* (SPA) robusta y modular. El desarrollo se apoya en **TypeScript/JavaScript** y herramientas como Angular CLI para la gestión de componentes y el despliegue en el puerto 4200.
- **Backend (Servidor):** Se utiliza **Spring Boot** (Java) como núcleo del servidor.
  - Esta tecnología proporciona lógica de negocio, seguridad y conexión a base de datos.
  - El servidor se despliega sobre un contenedor Apache Tomcat embebido, ejecutándose en el puerto 8000, tal como evidencia el log de arranque del sistema.
- **Base de Datos:** Se emplea **MySQL** como sistema de gestión de base de datos relacional. Esta elección garantiza la consistencia e integridad de los datos, dando soporte a las relaciones complejas definidas entre usuarios, vehículos y viajes.
- **Infraestructura:** El despliegue se realiza sobre un servidor **Ubuntu Server**, elegido por su estabilidad y seguridad en entornos de producción.

# 2. Diseño Funcional

## 2.1 Requisitos del sistema

El diseño funcional de UniCar se basa en un análisis de mercado (**Benchmarking**) y en la identificación de las necesidades específicas de nuestro **Público Objetivo** universitario. La aplicación aspira a ser la solución más eficiente para el transporte diario entre campus y pueblos en la provincia de Sevilla, tal como se define en su **Propuesta Única de Valor (PUV)**: una plataforma de transporte compartido enfocada a universitarios para optimizar gastos y tiempo.

### Requisitos Funcionales (In-Scope)

Los requisitos funcionales, que definen las acciones que el sistema debe realizar, incluyen:

- **Autenticación y Perfil:**
  - **Registro de Usuario:** Permitir a nuevos usuarios crear una cuenta.
  - **Login de Usuario:** Autenticación segura para el acceso al sistema.
  - **Registro de Coche:** Capacidad para que los conductores registren la información de sus vehículos.
- **Gestión de Viajes:**
  - **Registro de Viaje:** Funcionalidad para que los conductores publiquen detalles de un nuevo trayecto (origen, destino, fecha, hora, precio).
  - **Publicaciones de Viajes:** Permite la visualización y búsqueda de viajes disponibles por parte de los pasajeros.
  - **Valoraciones del Viaje:** Implementación de un sistema para que los usuarios valoren la experiencia de viaje.
- **Comunicación:**
  - **Chat en Tiempo Real:** Establecer un sistema de mensajería para la comunicación entre el conductor y los pasajeros inscritos en un viaje.

### Requisitos Excluidos (Out-of-Scope)

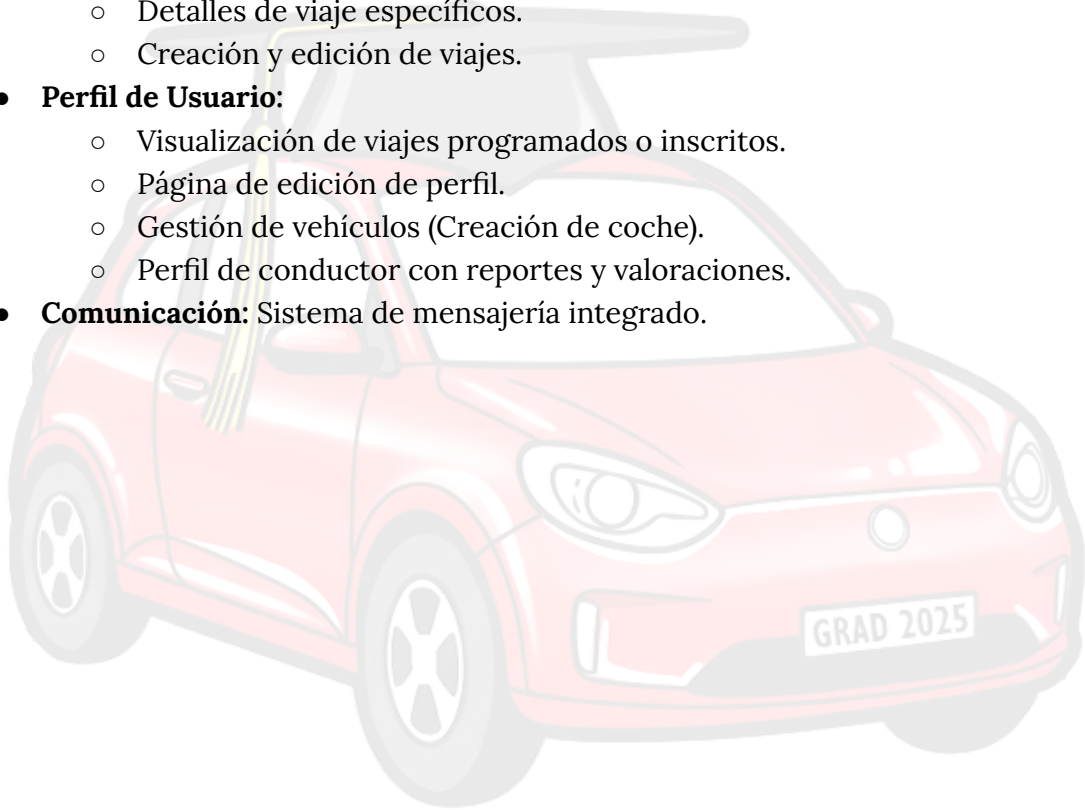
Las siguientes funcionalidades han sido deliberadamente excluidas en la fase inicial del proyecto para centrar los esfuerzos en el **Core** de la aplicación, pero son consideradas para futuras iteraciones:

- Sistema de pagos online (se opta por una gestión de pagos fuera de la plataforma para evitar comisiones y complejidad inicial).
- Sistema de visualización de rutas específicas (GPS) para el viaje.
- Sistema de foro de recomendación de aparcamientos.
- Sistema de estabilización de precios.

## 2.2 Mapa del sitio

La estructura de navegación de la aplicación se ha diseñado para ofrecer un acceso rápido a las funcionalidades principales. El organigrama de la aplicación se compone de las siguientes secciones jerárquicas:

- **Inicio (Landing/Login):** Punto de entrada para autenticación y registro de usuarios.
- **Página Principal:** Panel central que incluye accesos rápidos, sugerencias de viajes y campos de búsqueda.
- **Gestión de Viajes:**
  - Búsqueda de viajes (filtrado por origen/destino).
  - Detalles de viaje específicos.
  - Creación y edición de viajes.
- **Perfil de Usuario:**
  - Visualización de viajes programados o inscritos.
  - Página de edición de perfil.
  - Gestión de vehículos (Creación de coche).
  - Perfil de conductor con reportes y valoraciones.
- **Comunicación:** Sistema de mensajería integrado.



# 3. Diseño Técnico

## 3.1 Diagrama Entidad-Relación (E-R) y Tablas de Base de Datos

El modelo de datos relacional se ha diseñado para soportar la integridad de usuarios, viajes y vehículos.

### Entidades Principales y Atributos:

- **Usuario:** Almacena la información personal y credenciales.
  - Atributos: id, nombre completo, DNI, nombre de usuario, contraseña, fecha de nacimiento, sexo, teléfono, email, strikes, estado de baneo y antigüedad del carnet.
  - Claves Foráneas: id\_destinoUsual, id\_residencia.
- **Vehículo:** Registra los automóviles asociados a los usuarios.
  - Atributos: id, matrícula, modelo.
  - Relación: Vinculado al usuario propietario (id\_usuario).
- **Viaje:** Representa el trayecto ofertado.
  - Atributos: id, es\_ida (booleano), hora\_salida, fecha, ubicación\_salida, precio.
  - Relaciones: Vinculado a un vehículo, un destino y un pueblo.
- **Ubicaciones (Destino y Pueblo):**
  - Destino: id, nombre, dirección.
  - Pueblo: id, nombre.
- **Gestión y Seguridad (Reporte y Valoración):**
  - Reporte: id, motivo, descripción, gravedad. Vincula al usuario que reporta y al viaje reportado.
  - Valoración: Relación N:M entre usuarios para asignar estrellas.

### Relaciones Clave del Sistema:

- **Usuario - Vehículo:** Relación 1:N (Un usuario puede tener varios vehículos).
- **Viaje - Pasajero:** Relación N:M (Un usuario puede apuntarse a varios viajes).
- **Usuario - Destino/Pueblo:** Relación 1:N para definir residencias y destinos habituales.

## 3.2 Arquitectura del sistema y Especificación de la API

El sistema UniCar se implementa bajo un patrón de **Arquitectura Cliente-Servidor Desacoplada**, utilizando la tecnología REST (Representational State Transfer) para la comunicación. El *Frontend* (Angular) interactúa con el *Backend* (Spring Boot) mediante peticiones HTTP que gestionan la lógica de negocio y el acceso a la Base de Datos (MySQL).

### Especificación de la API REST

Funciones públicas				
Función	Tipo de operación	URL	Información enviada	Información devuelta
Funciones: Autenticación				
Login	POST	/api/auth/login	Datos del usuario existente	Confirmación de inicio de sesión
Registro	POST	/api/auth/signup	Datos del usuario a crear	Confirmación de registro
Cerrar sesión	GET	/api/auth/logout	Nada	Confirmación de cierre de sesión
Recuperar pswd	GET	/api/auth/recovery-password	Correo de la cuenta	Confirmación de que el correo existe
Funciones: Viaje				
Obtener viajes	GET	/api/travels	Filtros del viaje	Viajes coincidentes con los filtros
Crear viaje	POST	/api/travel	Datos del viaje	Confirmación de creación de viaje
Editar viaje	PUT	/api/travel/{id}	Datos del viaje	Confirmación de actualización del viaje
Cancelar viaje	DELETE	/api/travel/{id}	Id del viaje	Confirmación de la eliminación del viaje

Suscribirse a viaje	POST	/api/travel/{id}/join	Id del viaje	Confirmación de solicitud de inscripción a un viaje
Cancelar suscripción al viaje	DELETE	/api/travel/{id}/leave	Id del viaje	Confirmación de cancelación de inscripción a un viaje
Confirmar solicitud de viaje	POST	/api/travel/{idViaje}/accept/{idUser}	Id del viaje y del usuario a aceptar	Confirmación de aceptación
Denegar solicitud de viaje	POST	/api/travel/{idViaje}/deny/{idUser}	Id del viaje y del usuario a denegar	Confirmación de anulación
Reportar viaje	POST	/api/travel/{id}/report	Detalles del reporte	Nada
Buscar todos los viajes	GET	/api/travel/findAll	Nada	Listado con todos los viajes
<b>Funciones: Usuario</b>				
Obtener datos de X perfil	GET	/api/profile/{id}	ID/usuario buscado	Usuario con esos datos
Editar info. de X perfil	PUT	/api/profile/{id}	Nueva información del usuario	Nada
Obtener vehiculos de X perfil	GET	/api/profile/{id}/vehicles	ID/Usuario propietario del coche	Coches asignados a ese usuario
Valorar perfil conductor	POST	/api/profile/{id}/rate	Estrellas a otorgar	Nada
Eliminar perfil	DELETE	/api/profile	Nada	Confirmación de eliminación del perfil
<b>Funciones: Vehiculo</b>				

Obtener vehiculos	GET	/api/vehicles	Filtros de vehículo	Lista de vehículos con filtros aplicados
Crear vehiculo	POST	/api/vehicle	Datos del vehículo	Vehículo creado
Editar vehículo	PUT	/api/vehicle/{id}	Nuevos datos del vehículo	Nada
Eliminar vehículo	DELETE	/api/vehicle/{id}	Vehículo seleccionado	Confirmación de supresión

Funciones Administrativas				
Función	Tipo de operación	URL	Información enviada	Información devuelta
Funciones Admin: Ubicación				
Crear ubicacion	POST	/api/manager/location	Datos ubicación	Nada
Editar ubicación	PUT	/api/manager/location/{id}	Nuevos datos	Nada
Dar de baja ubicación	PATCH	/api/manager/location/{id}/disable	Nuevo estado (cerrar)	Nada
Dar de alta ubicación	PATCH	/api/manager/location/{id}/enable	Nuevo estado (abrir)	Nada
Funciones Admin: Usuario				
Vetar Usuario	PATCH	/api/manager/profile/{id}/ban	Nuevo estado (baneado)	Nada
Quitar veto usuario	PATCH	/api/manager/profile/{id}/unban	Nuevo estado (desbaneado)	Nada

# 4. Diseño de Interfaz

## 4.1 Guía de Estilo (Colores, Tipografía, Logos)

La identidad visual de la aplicación busca transmitir profesionalidad y coherencia mediante una guía de estilo definida.

**Paleta de Colores** Se utiliza un esquema de colores jerárquico para guiar la atención del usuario:

- **Primario (#4A90E2):** Usado en elementos principales como botones y enlaces destacados.
- **Secundario (#A78BFA):** Para fondos suaves o pestañas informativas.
- **Acento (#5DADEC):** Para acciones importantes e iconos.
- **Feedback:**
  - Éxito: Verde (#22C55E).
  - Error: Rojo anaranjado (#F87171).
  - Advertencia: Amarillo (#FACC15).
- **Neutros:** Fondo principal claro (#F9FAFB) y texto principal oscuro (#1F2937).

**Tipografía** Se ha seleccionado una combinación de fuentes modernas para la interfaz final:

- **Títulos (H1, H2, H3):** Se utiliza la fuente **Poppins** (pesos 700 y 600) para aportar carácter y modernidad a los encabezados.
- **Cuerpo y Etiquetas:** Se utiliza la fuente **Inter** (peso 400) para párrafos y etiquetas, garantizando una lectura clara.
- **Nota:** Se establece **Arial** como tipografía de respaldo (fallback) para asegurar la legibilidad en cualquier dispositivo.

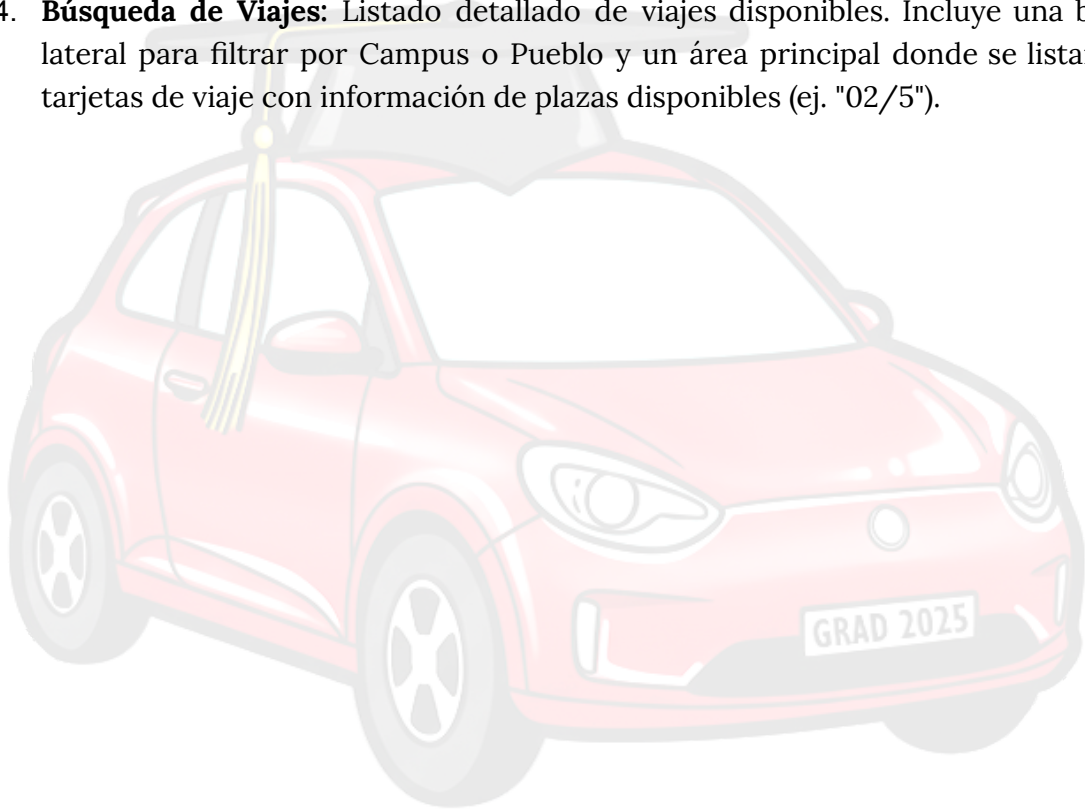
### Componentes de Interfaz

- **Botones:** Diseño con bordes redondeados. Estado normal en azul (#4A90E2), cambiando a azul oscuro (#1E3A8A) al pasar el cursor (hover).
- **Formularios:** Campos con fondo blanco y borde gris claro, resaltando en azul al recibir el foco.

## 4.2 Mockups finales y Prototipos

El diseño visual de las pantallas clave se ha estructurado de la siguiente manera:

1. **Página de Inicio (Home):** Presenta una barra de búsqueda destacada con filtros para "Campus", "Pueblo" y "Fecha y hora", facilitando el acceso inmediato a la funcionalidad principal .
2. **Login:** Formulario centralizado limpio con campos para usuario y contraseña, y opción de registro rápido.
3. **Perfil de Usuario:** Muestra la información del usuario en la cabecera, seguida de una lista de tarjetas que representan los viajes o interacciones (ej. "Uni 3 -> Pueblo 3").
4. **Búsqueda de Viajes:** Listado detallado de viajes disponibles. Incluye una barra lateral para filtrar por Campus o Pueblo y un área principal donde se listan las tarjetas de viaje con información de plazas disponibles (ej. "02/5").



# 5. Planificación y Desarrollo

## 5.1 Cronograma y Sprint Backlog

Debido a la naturaleza flexible del equipo y la ausencia de horarios definidos, se ha optado por una metodología basada en **Kanban**. Esto permite un flujo de trabajo continuo adaptado a la disponibilidad de los integrantes.

**Sprint Backlog (Tablero Kanban):** El plan de trabajo para la fase de desarrollo incluye las siguientes tareas, actualmente identificadas en el backlog:

- Buscar/Obtener Viajes.
- Cerrar Sesión.
- Crear/Registrar un Vehículo.
- Crear un Nuevo Viaje.
- Editar Perfil y Viaje.
- Gestión de Solicitudes (Aceptar/Denegar).
- Inicio de Sesión y Registro.
- Suscribirse a un Viaje.

## 5.2 Repositorio y Control de Versiones

El código fuente del proyecto se encuentra alojado en GitHub, facilitando el trabajo colaborativo y el control de versiones.

- **URL del Repositorio:** <https://github.com/Farolita23/Demo-UniCar.git>.
- **Estado del Entorno:** Se ha configurado exitosamente el entorno de desarrollo inicial. El backend (Spring Boot) se ha iniciado correctamente en el puerto 8000 , y el frontend (Angular) se ha compilado y desplegado en el puerto 4200, validando la conexión básica de la arquitectura.

## 6. Diseño Funcional

La transición a la fase de desarrollo del proyecto UniCar representa una etapa fundamental para aplicar los conocimientos teóricos adquiridos. El equipo espera obtener un profundo aprendizaje práctico en la integración de tecnologías *full-stack*: dominando la conexión de la API REST desarrollada en **Spring Boot (Java)** con el *framework* **Angular** y su gestión de estados en el *Frontend*. Además, esta fase será crucial para afianzar el uso de metodologías ágiles como **Kanban**, optimizar el flujo de trabajo colaborativo mediante **Git/GitHub**, y enfrentarse a desafíos de rendimiento, seguridad (implementación de la autenticación) y despliegue del sistema en un entorno **Ubuntu Server**. En última instancia, el desarrollo se concibe como el campo de pruebas definitivo para validar la solidez del diseño técnico y funcional propuesto.

