Memoria del Proyecto de Fin de Ciclo

Plataforma Web de Comparativa Exhaustiva de Vehículos

Autor: Pedro Sánchez Serrano

Ciclo Formativo: Administración de Sistemas Informáticos en Red (ASIR)

Centro Educativo: I.E.S. Infanta Elena, Galapagar

Tutor del Proyecto: Román Ontiyuelo Martín

Curso Académico: 2024-2025

Fecha de Entrega Estimada (Final): Junio 2025

(El resto de apartados como Desarrollo de la Práctica, Pruebas, Conclusiones, Agradecimientos, Bibliografía y Anexos con diagramas detallados se desarrollarán en las siguientes entregas hasta completar el 100% de la memoria).

Glosario (Inicial)

- **API (Application Programming Interface):** Interfaz que he desarrollado para que el frontend y el backend de mi aplicación puedan comunicarse de forma estructurada.
- **Backend:** Es el "cerebro" de mi aplicación, la parte que se ejecuta en el servidor. Aquí he programado toda la lógica de negocio con Node.js y Express.js.
- **Frontend:** Es la cara visible de mi proyecto, la interfaz con la que interactuarán los usuarios. La estoy construyendo con React para que sea dinámica y moderna.
- **Full-stack:** Término que describe el tipo de desarrollo que estoy realizando, abarcando tanto el frontend como el backend.
- **MySQL:** El sistema gestor de base de datos relacional que he elegido para almacenar toda la información de los vehículos y usuarios.
- **Node.js:** El entorno de ejecución de JavaScript que me permite usar este lenguaje también en el lado del servidor.
- **Express.js:** Un framework de Node.js que me ha facilitado mucho la creación de la API RESTful.
- **React:** La biblioteca de JavaScript que estoy utilizando para construir la interfaz de usuario. Me permite crear componentes reutilizables y eficientes.
- **API RESTful:** El estilo de arquitectura que sigo para diseñar mi API, basándome en los principios de REST para la comunicación entre sistemas.
- **CRUD:** Acrónimo de Create, Read, Update, Delete (Crear, Leer, Actualizar, Borrar). Son las operaciones básicas que mi API permitirá realizar sobre los datos.
- **DDNS (Dynamic Domain Name System):** Un servicio que configuraré para poder alojar el proyecto en mi PC personal y que sea accesible desde el instituto como si estuviera en un hosting profesional, asignando un nombre de dominio a mi IP dinámica.
- **JWT (JSON Web Token):** Aunque inicialmente exploré express-session (visible en el código), estoy considerando JWT como estándar para crear tokens de acceso seguros para la autenticación de usuarios.
- **Argon2:** El algoritmo de hashing que he implementado para proteger las contraseñas de los usuarios en la base de datos.

1. Introducción

En un mundo donde la información es clave, y más aún cuando se trata de decisiones importantes como la adquisición de un vehículo, me di cuenta de la necesidad de una herramienta que fuera más allá de lo convencional. Así nace la idea de este proyecto: una "Plataforma Web de Comparativa Exhaustiva de Vehículos". Mi objetivo es crear una aplicación full-stack que no solo presente datos, sino que ofrezca una experiencia de usuario rica, permitiendo explorar, comparar de forma avanzada, guardar preferencias y organizar la información de manera personalizada.

La plataforma que estoy desarrollando permitirá a los usuarios sumergirse en un catálogo detallado de automóviles, contrastar sus especificaciones técnicas mediante comparativas visuales e interactivas, y gestionar sus propias listas y vehículos favoritos. Además, quiero ir un paso más allá de las típicas webs de coches, incorporando funcionalidades como la comparativa de datos de rendimiento en circuito o estimaciones de precios históricos ajustados por la inflación, aspectos que considero de gran valor para el usuario entusiasta o el comprador analítico.

Para llevar a cabo este proyecto, he decidido utilizar un stack tecnológico moderno y con el que me siento cómodo y motivado para aprender: Node.js con Express.js para construir un backend robusto y una API RESTful eficiente, y React para desarrollar un frontend dinámico y atractivo. La base de datos, corazón del sistema, será MySQL, por su fiabilidad y estructura relacional. A lo largo de este documento, detallaré cómo he abordado cada fase del proyecto, desde la concepción hasta los primeros desarrollos.

1.1. Resumen y Abstract

(Este apartado lo completaré en una fase más avanzada del proyecto, cuando tenga una visión global del sistema implementado y los resultados obtenidos. Incluiré un resumen conciso en español y su correspondiente traducción al inglés).

1.2. Objetivos del Proyecto

Al embarcarme en este proyecto, me he planteado una serie de metas que guiarán todo el proceso de desarrollo:

Mi Objetivo General:

Desarrollar una aplicación web full-stack completa y funcional para la comparativa de vehículos. Quiero que sea una herramienta intuitiva pero potente, que realmente aporte valor a los usuarios a la hora de investigar y tomar decisiones sobre automóviles, ofreciendo funcionalidades que la distingan de otras plataformas existentes.

Mis Objetivos Específicos (lo que me propongo conseguir):

- Construir una API RESTful sólida: Utilizaré Node.js y Express.js para crear el backend que manejará toda la lógica de negocio y el acceso a los datos de forma segura y eficiente.
- Diseñar e implementar una base de datos MySQL optimizada: Quiero que la base de datos sea capaz de almacenar y gestionar una gran cantidad de información técnica de vehículos, así como los datos de los usuarios y sus interacciones (favoritos, listas).
- Desarrollar una interfaz de usuario (frontend) atractiva y funcional con React: La experiencia de usuario es clave, por lo que buscaré una navegación intuitiva, un diseño responsivo y una presentación clara de la información.
- Implementar un sistema de búsqueda y filtrado avanzado: Los usuarios podrán encontrar vehículos por múltiples criterios (marca, modelo, año, tipo de combustible, potencia, precio, etc.).
- Crear un comparador de vehículos visual e interactivo: Permitiré comparar varios vehículos cara a cara, no solo con tablas de datos, sino también con gráficos que faciliten la comprensión de las diferencias.
- Desarrollar un sistema completo de autenticación y gestión de perfiles de usuario: Esto incluye registro, inicio de sesión, y la posibilidad de que los usuarios gestionen su información personal.
- Permitir a los usuarios guardar favoritos y crear listas personalizadas: Funcionalidades esenciales para que cada usuario pueda organizar la información según sus intereses.
- Construir un panel de administración: Desde aquí, podré gestionar el catálogo de vehículos, asegurando la calidad y actualización de los datos.
- Integrar funcionalidades diferenciadoras: Como las estimaciones de precios

históricos ajustados por inflación, comparativas de rendimiento en circuito y, si el tiempo lo permite, un modo "eco-friendly".
- Explorar la viabilidad de una pequeña red social interna: Me gustaría que los usuarios pudieran compartir sus listas y comentarios, añadiendo un componente social a la plataforma.
- Priorizar la seguridad: Implementaré medidas como el hashing de contraseñas con Argon2 y la validación exhaustiva de todas las entradas de datos.
- Configurar el alojamiento en mi propio PC con acceso externo mediante DDNS: Esto no solo es un requisito del proyecto, sino una gran oportunidad para poner en práctica mis conocimientos de administración de sistemas y redes.

1.3. Motivación y Justificación Personal

La idea de desarrollar una plataforma de comparativa de vehículos surgió de mi propio interés por el mundo del motor y de la observación de que, aunque hay muchas webs con información de coches, a menudo me resultaba complicado encontrar una que reuniera todo lo que buscaba de forma centralizada y con herramientas de análisis potentes. Quería poder comparar no solo las cifras básicas, sino profundizar en detalles de rendimiento, ver la evolución de los precios o simplemente organizar mis "coches soñados" de una manera flexible.

Este proyecto representa para mí un reto apasionante y una oportunidad inmejorable para aplicar y consolidar todos los conocimientos que he adquirido durante mi formación en ASIR. Me motiva especialmente el desarrollo full-stack, ya que me permite tener una visión completa del ciclo de vida de una aplicación web, desde la concepción de la base de datos y la lógica del servidor hasta la creación de una interfaz de usuario interactiva.

La elección de tecnologías como Node.js, Express y React no es casual. JavaScript es un lenguaje versátil que me permite trabajar en ambos lados de la aplicación, y estos frameworks son ampliamente utilizados en la industria, lo que supone un aprendizaje muy valioso de cara a mi futuro profesional. Además, la gestión de una base de datos MySQL y la configuración del entorno de servidor en mi propio equipo, incluyendo el acceso mediante DDNS, son tareas que se alinean perfectamente con el perfil de un administrador de sistemas.

En definitiva, este proyecto no es solo un requisito académico, sino una iniciativa personal para crear algo útil y bien hecho, explorando áreas que me interesan y superando los desafíos técnicos que ello conlleva.

1.4. Alcance del Proyecto

He definido el alcance de mi proyecto "Plataforma Web de Comparativa Exhaustiva de Vehículos" para que sea ambicioso pero realizable en el tiempo disponible. A continuación, detallo las funcionalidades y características que me propongo implementar:

Módulo de Usuario (lo que el usuario podrá hacer):

- **Navegación y Búsqueda:**
  - Acceder a un catálogo completo de vehículos, con paginación para una carga eficiente.
  - Realizar búsquedas por texto libre (marca, modelo, etc.).
  - Utilizar un sistema de filtros avanzados con múltiples criterios (marca, modelo, generación, motorización, año, tipo de carrocería, combustible, etiqueta DGT, potencia, precio, peso, y más).
  - Ordenar los resultados según diferentes criterios (precio, potencia, etc.).
- **Visualización de Información:**
  - Consultar una página de detalle para cada vehículo, donde se mostrarán todas sus especificaciones técnicas, una galería de imágenes y los tiempos en circuito.
- **Comparador de Vehículos:**
  - Seleccionar de 2 a 4 vehículos para una comparativa exhaustiva.
  - Ver las especificaciones de los vehículos seleccionados en un formato de tabla clara.
  - Visualizar gráficos interactivos para comparar aspectos clave del rendimiento (ej. curvas de potencia/par, aceleración).
- **Funcionalidades para Usuarios Registrados:**
  - Registrarse e iniciar sesión de forma segura.
  - Gestionar su perfil.
  - Marcar vehículos como "Favoritos" para un acceso rápido.
  - Crear y gestionar "Listas Personalizadas" de vehículos, pudiendo añadir notas y organizar los coches a su gusto.
  - (Potencial) Compartir sus listas si implemento la funcionalidad social.
- **Otras Funcionalidades (según mi borrador inicial y si el desarrollo avanza bien):**
  - Consultar estimaciones de precios históricos de vehículos, ajustados por inflación.
  - Ver rankings de vehículos basados en diferentes criterios.
  - Utilizar un "modo eco-friendly" para analizar la eficiencia y emisiones.

Módulo de Administración (lo que podré hacer yo para gestionar la web):

- Acceder a un panel de administración protegido.
- Gestionar (altas, bajas, modificaciones y consultas - CRUD) todo el catálogo: marcas, modelos, generaciones, motorizaciones y los vehículos específicos con todas sus características.
- Administrar las imágenes de cada vehículo.
- Gestionar los tiempos en circuito.

Aspectos Técnicos y No Funcionales (cómo funcionará "por debajo" y qué cualidades tendrá):

- **Backend:** Desarrollaré una API RESTful con Node.js y Express.js.
- **Frontend:** Crearé una Single Page Application (SPA) con React.
- **Base de Datos:** Utilizaré MySQL, con un diseño de base de datos relacional bien estructurado.
- **Seguridad:** Implementaré hashing de contraseñas (Argon2), validación de todas las entradas de datos y otras medidas para proteger la aplicación.
- **Usabilidad:** Buscaré una interfaz intuitiva y un diseño responsivo que se adapte a móviles, tabletas y ordenadores de escritorio.
- **Rendimiento:** Optimizaré las consultas a la base de datos y la carga de datos en el frontend para que la web sea rápida.
- **Alojamiento:** Montaré el proyecto en mi PC personal, haciéndolo accesible desde el IES Infanta Elena a través de una configuración DDNS.

Lo que no voy a incluir por ahora (para mantener el proyecto enfocado):

- Una aplicación móvil nativa (me centraré en la web responsiva).
- Integraciones con sistemas de pago o concesionarios.
- Sistemas de predicción de precios complejos basados en Machine Learning (más allá de las estimaciones basadas en inflación que he planeado).

1.5. Antecedentes y Estado del Arte

Antes de empezar a diseñar mi propia plataforma, investigué qué herramientas existen actualmente en el mercado. Hay muchas webs y aplicaciones que ofrecen información sobre coches, desde portales generalistas hasta las propias páginas de los fabricantes. Algunas de las más conocidas son:

- Coches.net: Un referente en España, con muchos anuncios, noticias y un comparador.
- Km77.com: Valorado por la calidad de su información técnica y sus pruebas.
- Autofacil.es, Caranddriver.com/es: Revistas del motor con mucha presencia online.

- Webs de fabricantes (BMW, Audi, etc.): Ofrecen información muy detallada de sus modelos, pero lógicamente no permiten compararlos con la competencia.
- Portales internacionales como Edmunds.com, Ultimatespecs o KBB.com: Son muy completos, especialmente en el mercado estadounidense.

Si bien estas plataformas son muy útiles, he identificado áreas donde mi proyecto puede aportar un valor diferencial. Mi idea es ofrecer una **mayor personalización y herramientas de análisis más profundas**. Por ejemplo, la creación de listas personalizadas con notas detalladas, la comparativa gráfica interactiva de múltiples especificaciones de rendimiento (no solo las típicas), la estimación de precios históricos ajustados por inflación o los tiempos en circuito, son funcionalidades que no siempre se encuentran juntas o con el nivel de detalle que quiero ofrecer.

Además, con este proyecto busco alinearme con la tendencia de facilitar al consumidor herramientas que le permitan analizar datos complejos de forma sencilla, ayudándole a tomar decisiones más informadas. La posibilidad, aunque sea a explorar, de añadir un componente social para compartir estas comparativas y listas, también podría ser un punto distintivo.

2. Metodología y Planificación

Para llevar a buen puerto un proyecto de esta envergadura, es fundamental contar con una metodología de trabajo clara y una planificación realista.

2.1. Metodología de Desarrollo Aplicada

He decidido abordar el desarrollo de esta plataforma web utilizando una **metodología ágil incremental**. ¿Por qué esta elección? Dada la complejidad y el número de funcionalidades que quiero implementar, un enfoque ágil me permite ser flexible y adaptarme a los cambios o nuevas ideas que puedan surgir. En lugar de intentar definirlo todo al milímetro desde el principio, iré construyendo la aplicación por partes (incrementos), desarrollando y probando funcionalidades en ciclos cortos.

Esto significa que empezaré por las características más esenciales, como la estructura básica del backend (API, conexión a la base de datos) y del frontend (autenticación, visualización de vehículos), para luego ir añadiendo capas de funcionalidad más específicas y avanzadas (el comparador detallado, las listas personalizadas, el panel de administración, los gráficos de rendimiento, etc.). Mi planificación inicial, detallada en el documento "Anteproyecto_TFC_Pedro Sánchez Serrano", contempla una serie de fases o sprints que me servirán de guía, aunque estaré abierto a reajustes según las necesidades del proyecto.

2.2. Herramientas Utilizadas para el Desarrollo y Gestión

Para el desarrollo y la gestión de este proyecto, estoy utilizando un conjunto de herramientas que me facilitan el trabajo:

- **Control de Versiones: Git** es fundamental. Todo el código fuente del proyecto lo gestiono con Git y lo alojo en un repositorio privado en **GitHub** (comparativa_proyecto). Esto me permite llevar un control exhaustivo de los cambios, volver a versiones anteriores si es necesario y tener una copia de seguridad de mi trabajo.
- **Entorno de Desarrollo Integrado (IDE):** Principalmente utilizo **Visual Studio Code**, por su gran cantidad de extensiones útiles para el desarrollo web con JavaScript, Node.js y React, su terminal integrada y su integración con Git. Migré a Cursor AI pro y aceleré el desarrollo.
- **Gestión de Tareas y Planificación:** Aunque no uso una herramienta formal de gestión de proyectos compleja, me apoyo en mi cronograma (presente en el "Anteproyecto_TFC") y en listas de tareas personales para organizar el trabajo semanal y los objetivos de cada fase. La comunicación con mi tutor, Román Ontiyuelo, a través de reuniones y correo electrónico, también es clave para el seguimiento.
- **Documentación:** Para la elaboración de esta memoria, estoy utilizando **Google Docs**, lo que me permite tenerla accesible desde cualquier lugar.
- **Pruebas de API:** Herramientas como **Postman** o la extensión Thunder Client de VSCode me resultan muy útiles para probar los endpoints de mi API backend a medida que los voy desarrollando.
- **Base de Datos:** Para diseñar y administrar la base de datos MySQL, utilizo herramientas como **MySQL Workbench**.

3. Análisis y Diseño del Sistema

En esta sección, voy a detallar cómo he analizado los requisitos de la plataforma y cómo he planteado el diseño tanto de la arquitectura general como de la base de datos.

3.1. Identificación de Requisitos

Para que la plataforma sea útil y cumpla con mis objetivos, he definido una serie de “requisitos funcionales” (lo que el sistema debe hacer) y “no funcionales” (cómo debe hacerlo).

3.1.1. Requisitos Funcionales (RF)

Estos son los servicios y funciones que la plataforma ofrecerá a los usuarios:

Gestión de Usuarios (RF-AUTH):

- **RF-AUTH-001:** Quiero que cualquier visitante pueda registrarse en la web, para lo cual necesitará proporcionar un nombre, un email válido y una contraseña.
- **RF-AUTH-002:** Los usuarios ya registrados podrán iniciar sesión utilizando su email y contraseña.
- **RF-AUTH-003:** El sistema mantendrá la sesión del usuario activa, dándole acceso a las funcionalidades que requieren estar logueado.
- **RF-AUTH-004:** Los usuarios podrán cerrar su sesión cuando lo deseen.
- **RF-AUTH-005:** Si un usuario olvida su contraseña, podrá solicitar un enlace de restablecimiento que se enviará a su email.
- **RF-AUTH-006:** Con el token recibido por email, el usuario podrá establecer una nueva contraseña.
- **RF-AUTH-007:** Implementaré roles de usuario (al menos 'user' y 'admin') para gestionar los permisos de acceso a diferentes partes de la aplicación.

Catálogo y Búsqueda de Vehículos (RF-VEH):

- **RF-VEH-001:** La web mostrará un catálogo de vehículos, con un sistema de paginación para no cargar todos los datos de golpe.
- **RF-VEH-002:** Habrá una barra de búsqueda para que los usuarios puedan buscar vehículos por texto (ej. "Ford Focus", "Golf GTI").
- **RF-VEH-003:** Incluiré un panel de filtros avanzados para que se puedan refinar las búsquedas por marca, modelo, generación, motorización, año, tipo de carrocería, combustible, etiqueta DGT, rango de potencia, precio, peso, etc.
- **RF-VEH-004:** Los resultados de las búsquedas y filtros se podrán ordenar por diferentes criterios (ej. precio más bajo primero, más potente, etc.).
- **RF-VEH-005:** Cada vehículo tendrá su propia página de detalle, donde se mostrará toda su información técnica, una galería de imágenes y los tiempos que haya podido registrar en circuitos.

Comparador de Vehículos (RF-COMP):

- **RF-COMP-001:** El usuario podrá seleccionar entre 2 y 4 vehículos del catálogo para realizar una comparativa.
- **RF-COMP-002:** Se mostrará una tabla con las especificaciones de los vehículos seleccionados, una al lado de la otra, para facilitar la comparación directa.
- **RF-COMP-003:** Integraré gráficos interactivos (usando Chart.js) para comparar visualmente aspectos clave del rendimiento, como potencia, par, aceleración, etc.

Funcionalidades de Usuario Registrado (RF-USER):

- **RF-USER-001:** Los usuarios logueados podrán marcar vehículos como "Favoritos".
- **RF-USER-002:** Podrán acceder a una sección donde verán todos sus vehículos favoritos.
- **RF-USER-003:** Podrán crear "Listas Personalizadas" de vehículos (ej. "Candidatos para mi próximo coche", "Deportivos de los 90").
- **RF-USER-004:** Podrán añadir y quitar vehículos de estas listas, y añadir notas o comentarios a cada vehículo dentro de una lista.
- **RF-USER-005:** Podrán gestionar sus listas: cambiarles el nombre, la descripción o eliminarlas.
- **RF-USER-006:** (Opcional, si progreso bien) Consideraré la opción de que los usuarios puedan hacer públicas algunas de sus listas para compartirlas.

Panel de Administración (RF-ADMIN):

- **RF-ADMIN-001:** Habrá un panel de administración con acceso restringido solo para mí (o futuros administradores).
- **RF-ADMIN-002:** Desde este panel, podré gestionar todas las entidades principales de la base de datos: Marcas, Modelos, Generaciones, Motorizaciones y Vehículos, incluyendo todas sus especificaciones (altas, bajas, modificaciones).
- **RF-ADMIN-003:** También podré gestionar las imágenes asociadas a cada vehículo.
- **RF-ADMIN-004:** Y administrar los tiempos en circuito.

3.1.2. Requisitos No Funcionales (RNF)

Estos requisitos definen cómo debe ser el sistema en términos de calidad y restricciones:

- **RNF-USA-001 (Usabilidad):** La interfaz de usuario debe ser intuitiva y fácil de usar. Quiero que la navegación sea clara y que los usuarios no se pierdan.
- **RNF-USA-002 (Responsividad):** La aplicación web se tiene que ver y funcionar bien en diferentes dispositivos: ordenadores de escritorio, tabletas y móviles.
- **RNF-REND-001 (Rendimiento):** Los tiempos de carga de las páginas y las respuestas a las acciones del usuario deben ser rápidos. Optimizaré las consultas a la base de datos para ello.
- **RNF-SEG-001 (Seguridad de Contraseñas):** Las contraseñas de los usuarios se guardarán en la base de datos hasheadas con Argon2, un algoritmo robusto.
- **RNF-SEG-002 (Validación de Entradas):** Todos los datos que introduzca el usuario (en formularios, por la URL, etc.) se validarán tanto en el frontend (React) como en el backend (Express) para evitar errores y posibles ataques.
- **RNF-SEG-003 (Protección CSRF/XSS):** Implementaré medidas básicas, como el

uso de la librería Helmet en el backend, para mitigar riesgos de ataques comunes como CSRF y XSS.

- **RNF-SEG-004 (Prevención de Fuerza Bruta):** Para los intentos de inicio de sesión, aplicaré alguna técnica de *throttling* (limitación de intentos en un periodo de tiempo) para dificultar los ataques de fuerza bruta.
- **RNF-HOST-001 (Alojamiento):** Como ya he mencionado, el sistema se alojará en mi PC personal y será accesible desde el IES Infanta Elena gracias a una configuración DDNS.
- **RNF-MANT-001 (Mantenibilidad):** Escribiré un código bien estructurado, con comentarios claros y siguiendo buenas prácticas. Esto es importante para que, en el futuro, yo mismo u otra persona pueda entenderlo y modificarlo fácilmente.
- **RNF-DAT-001 (Calidad de Datos):** Aunque la carga masiva inicial de datos es una tarea enorme y fuera del alcance del desarrollo puro, intentaré que los datos de los vehículos sean lo más precisos y completos posible para los ejemplos que utilice.

3.2. Diseño de la Arquitectura del Sistema

He optado por una arquitectura cliente-servidor, que es estándar para aplicaciones web, y la he dividido en tres capas principales.

3.2.1. Visión General de la Arquitectura

Mi proyecto se estructura de la siguiente manera:

1. **Capa de Presentación (Frontend):**
    - Es lo que el usuario ve y con lo que interactúa en su navegador. La estoy desarrollando con **React**.
    - Se encarga de mostrar la interfaz gráfica, capturar las acciones del usuario (clics, formularios) y visualizar los datos de los vehículos.
    - Se comunica con el backend mediante peticiones HTTP (usando Axios) para pedir o enviar información.
    - Gestiona el estado de la aplicación en el lado del cliente (quién está logueado, qué filtros están aplicados, etc.).
    - Algunos de los componentes React que ya tengo o planeo son: HomePage (la página de inicio), VehiclesPage (que contendrá el FilterSidebar y las VehicleCard), VehicleDetailPage (para ver un coche en detalle), ComparisonPage (el comparador), LoginPage, RegisterPage, y las páginas del panel de administración (AdminDashboardPage, AdminVehicleListPage, AdminVehicleFormPage), así como las de usuario (FavoritesPage, MyListsPage, ListDetailPage, ProfilePage).

2. **Capa de Lógica de Negocio (Backend):**
    - Es el motor de la aplicación, invisible para el usuario pero esencial. Lo estoy construyendo con **Node.js** y el framework **Express.js**.
    - Su función principal es exponer una **API RESTful** que el frontend pueda consumir.
    - Aquí reside toda la lógica de negocio: cómo se autentican los usuarios, cómo se gestionan sus sesiones, la validación de los datos que llegan del frontend, cómo se procesan las búsquedas y los filtros, cómo se guardan los favoritos y las listas, y todas las operaciones de alta, baja y modificación de vehículos que haré desde el panel de administración.
    - Los controladores principales que estoy desarrollando son: authController.js (para todo lo relacionado con usuarios), vehicleController.js (para vehículos), favoriteController.js (favoritos), listController.js (listas), imageController.js (imágenes) y timeController.js (tiempos en circuito).

2. **Capa de Datos (Persistencia):**
    - Aquí es donde se guardan todos los datos de forma permanente. He elegido **MySQL** como sistema gestor de base de datos relacional.
    - Almacenará la información de los usuarios, todos los detalles de los vehículos (organizados en tablas como marcas, modelos, generaciones, motorizaciones), las rutas de las imágenes, los tiempos en circuito, las listas de los usuarios y sus favoritos.
    - El backend se comunica con esta capa utilizando el driver mysql2 de Node.js, que me permite ejecutar consultas SQL.

¿Cómo se comunican estas capas?

El Frontend (React en el navegador del usuario) y el Backend (Node.js/Express en mi servidor) se comunican a través del protocolo HTTP. El frontend envía peticiones (GET para pedir datos, POST para enviar nuevos, PUT para actualizar, DELETE para borrar) y el backend responde, generalmente con datos en formato JSON.

El Backend, a su vez, se comunica con la base de datos MySQL para leer o escribir información, ejecutando sentencias SQL.

Sobre el Alojamiento:

Tanto el backend (la aplicación Node.js) como la base de datos MySQL se ejecutarán en mi PC personal.
Para que sea accesible desde el instituto, configuraré un servicio de DDNS. Esto asociará un nombre de dominio a la dirección IP pública de mi casa (que puede ser dinámica).

El frontend, una vez construido para producción (el build de React), será servido probablemente por el mismo servidor Node.js/Express, o podría usar un servidor web ligero como Nginx si fuera necesario, también en mi PC.

3.2.2. Elección de Tecnologías y Fundamentos

La elección de las tecnologías para este proyecto no ha sido aleatoria. He buscado un conjunto de herramientas que me permitan desarrollar una aplicación moderna, eficiente y escalable, y que al mismo tiempo me sirvan para profundizar en tecnologías muy demandadas en el mundo laboral.

- **Backend:**
  - **Node.js:** Elegí Node.js porque me permite usar JavaScript, un lenguaje que ya conozco del frontend, también en el servidor. Su modelo asíncrono y orientado a eventos es ideal para aplicaciones web que necesitan manejar múltiples peticiones concurrentes de forma eficiente, como será mi plataforma.
  - **Express.js:** Es un framework para Node.js muy popular, minimalista y flexible. Me facilita enormemente la creación de rutas para la API, la gestión de *middleware* (como los que uso para logging, errores o autenticación) y, en general, la organización del código del backend.
  - **MySQL2:** Es el driver de Node.js que estoy usando para conectarme a la base de datos MySQL. Es una versión mejorada del driver original, con soporte para promesas, lo que hace el código de acceso a datos más limpio y moderno.
  - **Argon2:** Para la seguridad de las contraseñas, he optado por Argon2. Es un algoritmo de hashing moderno y muy robusto, ganador de la Password Hashing Competition, lo que me da confianza en su capacidad para proteger las credenciales de los usuarios.
  - **JSON Web Tokens (JWT):** Aunque en el código actual del repositorio se puede ver el uso de express-session para la gestión de sesiones, estoy valorando seriamente migrar o complementar con JWTs para la autenticación basada en tokens, especialmente si decido separar más claramente el servidor de API del servicio de sesiones, o para facilitar una futura expansión a otros clientes (como una app móvil teórica). Los JWT son un estándar muy utilizado para transmitir información de forma segura entre partes como un objeto JSON.
  - **Express-validator:** Esta librería es muy útil para validar los datos que llegan en las peticiones a la API (ej. que un email tenga formato de email, que un campo requerido no esté vacío, etc.), ayudando a mantener la integridad de los datos y la seguridad.
  - **Dotenv:** Para no tener datos sensibles (como las credenciales de la base de

datos o claves secretas) directamente en el código, uso dotenv para cargarlos desde un archivo .env que no se sube al repositorio Git.
    - **Nodemailer:** La necesitaré para funcionalidades como el envío de correos para el restablecimiento de contraseñas.
    - **Helmet:** Es un middleware de Express que ayuda a securizar la aplicación estableciendo varias cabeceras HTTP importantes (protección contra XSS, clickjacking, etc.).
    - **Cors:** Para permitir que mi frontend (que se sirve en un dominio/puerto, aunque sea localhost durante el desarrollo) pueda hacer peticiones a mi backend (que se ejecuta en otro puerto).
    - **Morgan:** Un middleware para registrar las peticiones HTTP que llegan al servidor, muy útil para depuración y monitorización.
- **Frontend:**
    - **React:** He elegido React por su popularidad, su enfoque basado en componentes (que facilita la reutilización de código y la organización de la interfaz) y su eficiencia gracias al DOM virtual. Me permite construir interfaces de usuario complejas y dinámicas de forma declarativa.
    - **React Router DOM:** Esencial para una Single Page Application (SPA) como la mía, ya que gestiona la navegación entre las diferentes "páginas" o vistas de la aplicación sin necesidad de recargar la página completa.
    - **Axios:** Es mi cliente HTTP preferido para hacer las llamadas a la API del backend desde React. Es muy fácil de usar y está basado en promesas.
    - **Date-fns:** Para cualquier manipulación o formateo de fechas que necesite en el frontend (ej. mostrar fechas de forma amigable), date-fns es una librería ligera y potente.
    - **React-toastify:** Para mostrar notificaciones (toasts) al usuario de forma no intrusiva (ej. "Vehículo añadido a favoritos", "Error al iniciar sesión").
    - **React-icons:** Me proporciona una gran variedad de iconos SVG listos para usar en la interfaz, lo que mejora mucho el aspecto visual.
    - **Chart.js con react-chartjs-2:** Para la visualización de los gráficos comparativos de vehículos. Chart.js es una librería de gráficos muy versátil y react-chartjs-2 facilita su integración en componentes de React.
- **Base de Datos:**
    - **MySQL:** He optado por MySQL por ser un sistema de gestión de bases de datos relacional robusto, muy utilizado y con el que tengo cierta familiaridad. Su naturaleza relacional es adecuada para la estructura de datos de mi proyecto (vehículos con marcas, modelos, motorizaciones, etc., y las relaciones entre ellos y los usuarios).

- **Control de Versiones:**
  - **Git y GitHub:** Como mencioné, son imprescindibles para cualquier proyecto de desarrollo de software hoy en día.
- **Alojamiento y Despliegue (Simulado en mi PC):**
  - La decisión de alojar el proyecto en mi PC personal y configurarlo con un servicio de **DDNS** responde a un doble propósito: cumplir con una de las ideas del proyecto y, lo que es más importante para mí, adquirir experiencia práctica en la configuración de un servidor web accesible desde internet, la gestión de DNS dinámicos y la apertura/redirección de puertos en un router. Son conocimientos muy valiosos para un futuro administrador de sistemas.

Escalabilidad y Potencial:
Aunque el proyecto se desarrolla en un entorno limitado, he intentado tomar decisiones que favorezcan una posible escalabilidad futura. La arquitectura de tres capas y el uso de una API RESTful permiten que el frontend y el backend puedan evolucionar de forma independiente. Si el día de mañana quisiera, por ejemplo, crear una aplicación móvil, esta podría consumir la misma API. La base de datos MySQL es capaz de manejar grandes volúmenes de datos, y Node.js es conocido por su buen rendimiento bajo carga. Las funcionalidades que he planeado, como los análisis de precios o rendimiento, también sientan una base que podría expandirse con técnicas más avanzadas si el proyecto creciera.

3.3. Diseño de la Base de Datos

La base de datos es una pieza fundamental. He dedicado tiempo a pensar en su estructura para que sea eficiente, evite la redundancia de datos y mantenga la integridad de la información. El script database_setup.sql en mi repositorio contiene la definición completa de las tablas y sus relaciones.

3.3.1. Modelo Entidad/Relación Conceptual

(En esta primera entrega, describiré las entidades y sus relaciones. En una entrega posterior, incluiré un diagrama Entidad/Relación (E/R) visual, probablemente generado con alguna herramienta a partir de esta descripción o directamente desde MySQL Workbench, para que se vea claramente la estructura).

He diseñado la base de datos siguiendo un modelo relacional, identificando las principales entidades y cómo se conectan entre sí.

3.3.2. Descripción de las Entidades Principales

Las tablas más importantes que he creado en mi base de datos MySQL son:

- **Marca**: Aquí guardo los nombres de las marcas de los coches (ej. Volkswagen, BMW, Honda, SEAT).
    - Campos principales: id_marca (clave primaria, autoincremental), nombre (el nombre de la marca, único).
- **Modelo**: Almacena los diferentes modelos asociados a cada marca (ej. Golf, Serie 3, Civic, Ibiza).
    - Campos principales: id_modelo (PK), id_marca (clave foránea que referencia a Marca), nombre.
- **Generacion**: Para cada modelo, puede haber varias generaciones (ej. Golf MK1, Golf MK2, Golf MK8). Esta tabla las registra con sus años de producción.
    - Campos principales: id_generacion (PK), id_modelo (FK a Modelo), nombre (ej. "MK7", "Quinta Generación"), anio_inicio, anio_fin.
- **Motorizacion**: Contiene los detalles específicos de cada motor disponible para una generación de un modelo.
    - Campos principales: id_motorizacion (PK), id_generacion (FK a Generacion), codigo_motor (si lo tiene), nombre (ej. "1.9 TDI", "2.0 TSI"), combustible (Gasolina, Diesel, Híbrido, Eléctrico), potencia_cv, par_motor_nm, cilindrada_cc, etc.
- **Vehiculo**: Esta es la tabla central y más detallada. Representa una instancia específica de un coche, vinculada a una motorización concreta, y con todas sus especificaciones: tipo de carrocería, versión, pegatina ambiental, datos de rendimiento (velocidad máxima, 0-100 km/h), consumo, emisiones, dimensiones, peso, precio original y el precio actual estimado que quiero calcular, e incluso un campo detalle (JSON) para guardar especificaciones adicionales que no tengan su propia columna.
    - Campos principales: id_vehiculo (PK), id_motorizacion (FK a Motorizacion), anio, tipo_carroceria, version_especifica, pegatina_ambiental, velocidad_max_kmh, aceleracion_0_100_s, consumo_mixto_l_100km, emisiones_co2_g_km, peso_kg, precio_original_eur, precio_actual_estimado_eur, detalle (JSON).
- **Usuarios**: Aquí guardo la información de las personas que se registren en la web.
    - Campos principales: id_usuario (PK), nombre, email (que debe ser único), contrasena (donde guardaré el hash generado por Argon2), rol (para distinguir entre 'user' y 'admin').
- **Imagenes**: Para cada vehículo, podré tener varias imágenes. Esta tabla guarda las rutas (locales en mi servidor, por ahora) de esas imágenes.
    - Campos principales: id_imagen (PK), id_vehiculo (FK a Vehiculo),

ruta_local_imagen, descripcion_alt (texto alternativo), orden (para saber cuál es la imagen principal).

- **Listas**: Permite a los usuarios registrados crear sus propias colecciones o listas de vehículos.
    - Campos principales: id_lista (PK), id_usuario (FK a Usuarios), nombre_lista, descripcion, es_publica (booleano, para la futura función social).
- **Vehiculos_Listas**: Como una lista puede tener muchos vehículos y un vehículo puede estar en muchas listas (de diferentes usuarios), necesito una tabla intermedia para esta relación muchos-a-muchos.
    - Campos principales: id_vehiculo_lista (PK), id_lista (FK a Listas), id_vehiculo (FK a Vehiculo), fecha_agregado, notas_usuario (un pequeño texto que el usuario puede añadir para ese coche en esa lista).
- **Favoritos**: Similar a las listas, pero más simple. Es una relación muchos-a-muchos entre Usuarios y Vehiculo para que un usuario pueda marcar sus coches favoritos.
    - Campos principales: id_favorito (PK), id_usuario (FK a Usuarios), id_vehiculo (FK a Vehiculo), fecha_agregado. He puesto una restricción UNIQUE en (id_usuario, id_vehiculo) para que un usuario no pueda marcar el mismo coche como favorito varias veces.
- **Tiempos_Circuito**: Aquí registraré los tiempos de vuelta que los vehículos hayan conseguido en diferentes circuitos.
    - Campos principales: id_tiempo (PK), id_vehiculo (FK a Vehiculo), nombre_circuito, tiempo_vuelta (probablemente en formato TIME o como segundos), condiciones_pista, fecha_record.

Relaciones Clave y Consideraciones:
He definido claves foráneas entre las tablas para asegurar la integridad referencial. Por ejemplo, no puede existir un Modelo que no pertenezca a una Marca. En muchas de estas relaciones, como cuando se borra un vehículo, he configurado ON DELETE CASCADE. Esto significa que si borro un vehículo de la tabla Vehiculo, automáticamente se borrarán también sus imágenes asociadas en la tabla Imagenes, sus tiempos en Tiempos_Circuito, y sus apariciones en las tablas Vehiculos_Listas y Favoritos. Esto me ayuda a mantener la base de datos limpia y consistente.

3.4. Diagramas Conceptuales del Sistema

Para entender mejor cómo funciona todo el sistema y cómo interactúan sus partes, he pensado en varios diagramas. Aunque los dibujos los incluiré en las siguientes entregas, aquí describo lo que representarán.

3.4.1. Esquema de Interconexión de Componentes

(Este esquema lo dibujaré más adelante, pero básicamente mostrará los siguientes componentes y cómo se conectan).

El sistema que estoy construyendo tiene varios componentes principales que trabajan juntos:

1. **Cliente (Navegador Web del Usuario):**
    - Es el punto de entrada. Aquí se ejecuta mi aplicación frontend (React).
    - Desde aquí, el usuario navega, busca, compara, etc.
    - Cuando el usuario realiza una acción que requiere datos o guardar información, el frontend hace una petición HTTP (GET, POST, PUT, DELETE) a mi API backend.
2. **Servidor Web / Aplicación Backend (Mi PC con Node.js y Express.js):**
    - Este es el corazón de la aplicación, corriendo en mi ordenador personal.
    - Gracias al DDNS, es accesible desde fuera (por ejemplo, desde el instituto) a través de una dirección de internet.
    - Aquí es donde Express.js recibe las peticiones del cliente.
    - Node.js ejecuta la lógica de los controladores (authController, vehicleController, etc.), que procesan la petición: validan datos, aplican filtros, interactúan con la base de datos, etc.
    - Finalmente, el backend envía una respuesta al cliente, usualmente en formato JSON.
    - También se encarga de servir los archivos estáticos de la aplicación React cuando un usuario accede a la web por primera vez.
3. **Base de Datos (MySQL):**
    - También reside en mi PC.
    - Es donde el backend (Node.js) lee y escribe toda la información persistente (usuarios, coches, listas, etc.).
4. **Servicio DDNS (Externo):**
    - Es un servicio en internet que se encarga de que un nombre de dominio (ej. mi-comparador-pedro.dyndns.org, por inventarme uno) siempre apunte a la dirección IP pública de mi casa, aunque esta cambie.

Un ejemplo de flujo de datos (cuando un usuario busca coches):

1. El usuario, en el frontend (React), selecciona varios filtros (marca "BMW", año desde "2022").
2. El código JavaScript de React construye una petición, por ejemplo: GET /api/vehicles?marca=BMW&anioMin=2022.
3. Esta petición viaja por internet hasta mi PC, gracias al DDNS y la configuración de

mi router.
4. Mi servidor Express.js recibe la petición en la ruta /api/vehicles.
5. El vehicleController.js toma los parámetros (marca, anioMin).
6. Construye una consulta SQL (ej. SELECT * FROM Vehiculo WHERE ...).
7. Envía esta consulta a la base de datos MySQL.
8. MySQL ejecuta la consulta y devuelve los coches que cumplen los criterios.
9. El vehicleController recibe estos datos, los formatea como un array de objetos JSON.
10. Express.js envía esta respuesta JSON de vuelta al navegador del usuario.
11. El frontend (React) recibe el JSON y actualiza la interfaz para mostrar la lista de BMWs desde 2022.

3.4.2. Diagrama de Casos de Uso Principales

(También dibujaré este diagrama más adelante, pero aquí describo los actores y las acciones principales que podrán realizar).

Para visualizar quién hace qué en mi plataforma, un diagrama de casos de uso es muy útil.

Actores Principales (quiénes usan el sistema):

- **Usuario (Visitante / Registrado):** Cualquier persona que accede a la web. Si no está registrado, es un "Visitante". Si tiene cuenta y ha iniciado sesión, es un "Usuario Registrado".
- **Administrador:** Soy yo, o cualquier persona a la que le dé permisos especiales para gestionar el contenido de la web.

Casos de Uso más importantes para el Usuario:

- **UC-001: Registrarse en la plataforma**: Un visitante proporciona sus datos (nombre, email, contraseña) para crear una cuenta nueva.
- **UC-002: Iniciar Sesión**: Un usuario ya registrado introduce su email y contraseña para acceder a las funciones personalizadas.
- **UC-003: Cerrar Sesión**: Un usuario finaliza su sesión activa.
- **UC-004: Buscar Vehículos (con filtros y ordenación)**: El usuario utiliza los criterios de búsqueda y filtrado para encontrar vehículos de su interés.
- **UC-005: Ver Catálogo General de Vehículos**: El usuario navega por la lista completa de vehículos disponibles en la plataforma.
- **UC-006: Consultar Detalles de un Vehículo**: El usuario accede a la ficha técnica completa, imágenes y otros datos de un coche específico.
- **UC-007: Comparar Vehículos Seleccionados**: El usuario elige varios coches y visualiza una comparativa detallada de sus características y rendimiento.

- **UC-008: Gestionar Vehículos Favoritos** (Solo para Usuarios Registrados):
  - Añadir un vehículo a su lista de favoritos.
  - Ver su lista de vehículos favoritos.
  - Quitar un vehículo de sus favoritos.
- **UC-009: Gestionar Listas Personalizadas de Vehículos** (Solo para Usuarios Registrados):
  - Crear una nueva lista (ej. "Posibles SUV para la familia").
  - Ver todas sus listas creadas.
  - Acceder al detalle de una lista para ver los vehículos que contiene y las notas asociadas.
  - Añadir vehículos a una de sus listas.
  - Quitar vehículos de una lista.
  - Editar el nombre o la descripción de una lista.
  - Eliminar una lista completa.
- **UC-010: Solicitar Restablecimiento de Contraseña**: Si un usuario olvida su contraseña, puede pedir un email con instrucciones para crear una nueva.
- **UC-011: Restablecer la Contraseña**: Siguiendo el enlace del email, el usuario puede definir una contraseña nueva.
- **UC-012: Ver y Editar Perfil de Usuario**: El usuario puede consultar y modificar algunos datos de su perfil.

Casos de Uso para el Administrador (yo):

- **UC-ADM-001: Gestionar Catálogo Completo de Vehículos (CRUD)**: Como administrador, tendré acceso a un panel donde podré añadir nuevas marcas, modelos, generaciones, motorizaciones y vehículos con todas sus especificaciones. También podré verlos, editarlos y eliminarlos.
- **UC-ADM-002: Gestionar Imágenes de los Vehículos**: Podré subir, ver y eliminar las fotos asociadas a cada coche.
- **UC-ADM-003: Gestionar Tiempos en Circuito**: Podré añadir, modificar o eliminar los tiempos que los vehículos hayan hecho en diferentes circuitos.

Estos son los principales, pero podrían surgir algunos más detallados a medida que avance. La idea es tener una visión clara de todas las interacciones posibles.

Con esto cubro aproximadamente el primer tercio de la estructura de la memoria, enfocándome en la introducción, la planificación y el análisis inicial del sistema desde mi perspectiva como desarrollador del proyecto.