UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

F1nalLap

Presentado por Alberto García Alcolado
en Universidad de Burgos — 11 de junio de 2025
Tutor: César Represa Pérez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. César Represa Pérez, profesor del departamento de nombre Ingeniería Electromecánica, área de nombre Tecnología Electrónica.

Expone:

Que el alumno D. Alberto García Alcolado, con DNI 06288487B, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado F1nalLap.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 11 de junio de 2025

Vº. Bº. del Tutor:

D. César Represa Pérez

Resumen

Este proyecto consiste en el desarrollo de una aplicación web dedicada a la Fórmula 1, la cual proporciona información detallada sobre las temporadas actuales y pasadas (resultados, clasificaciones, estadísticas de pilotos y equipos) utilizando APIs externas. Además, incluye un simulador de estrategias de carrera que permite a los usuarios tener una visión estimada de las mejores paradas y cambio de neumáticos posibles para cada Gran Premio.

La aplicación busca ofrecer una fuente accesible y gratuita de información sobre la F1, con un enfoque único en la simulación y el análisis de datos.

Está desarrollada con Angular, utilizando TypeScript, HTML y SCSS, y se implementa el alojamiento mediante Netlify. El proyecto también integra la gestión de datos mediante APIs externas, mejorando la experiencia del usuario en tiempo real.

Descriptores

Fórmula 1, aplicación web, Angular, simulador de estrategias, estadísticas, API de datos, Netlify, diseño responsive

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad, la tecnología desempeña un papel crucial en el mundo del deporte, transformando la manera en que los aficionados interactúan con sus disciplinas favoritas y proporcionando herramientas poderosas para el análisis de datos en tiempo real. La Fórmula 1 [1], uno de los deportes más emocionantes y seguidos globalmente, genera una gran cantidad de datos que pueden aprovecharse para mejorar la comprensión de los resultados y las estrategias durante cada temporada.

La F1 es considerada la máxima categoría del automovilismo, caracterizada por ser una de las competiciones más avanzadas tecnológicamente y por su elevado nivel de competencia. Cada temporada está compuesta por una serie de carreras conocidas como Grandes Premios (GP), que se celebran en circuitos alrededor del mundo. Cada GP es una competencia independiente, pero todas se suman para determinar a los campeones del mundo en dos categorías: Campeón del Mundo de Pilotos y Campeón del Mundo de Constructores.

Cada año se presentan características únicas que influyen en la estrategia de equipos y pilotos, así como en la planificación a largo plazo. El calendario de carreras se extiende aproximadamente de marzo a diciembre, con más de 20 eventos en diferentes países. La estructura de cada Gran Premio incluye sesiones de prácticas libres, clasificatorias y la carrera final. Además, las reglas de la Fórmula 1 evolucionan continuamente, destacando aspectos como:

- **Uso obligatorio de múltiples compuestos:** Cada piloto debe usar al menos dos tipos de neumáticos para fomentar la gestión estratégica durante la carrera.

- **Peso mínimo del monoplaza:** En 2025, el peso mínimo combinado de coche y piloto es 800 kg, con un mínimo para el piloto de 82 kg, ajustable mediante lastre para igualdad competitiva.

- **Penalizaciones por infracciones:** Se aplican sanciones por incumplimientos como adelantar bajo bandera amarilla o cambios ilegales, que pueden incluir tiempo añadido, pérdida de posiciones o exclusión.

Este trabajo se enfoca en el desarrollo de una aplicación web dedicada a la Fórmula 1, diseñada para ofrecer a los usuarios una fuente completa de información sobre las temporadas actuales y pasadas [2]. La aplicación no solo muestra resultados y clasificaciones, sino que también incorpora un simulador de estrategias de carrera, permitiendo a los usuarios experimentar y optimizar diferentes enfoques tácticos basados en las condiciones específicas de cada Gran Premio. El objetivo es proporcionar una plataforma única que haga accesible esta información de manera gratuita y detallada, cubriendo aspectos que no siempre están disponibles en otras páginas web de la misma temática.

Tradicionalmente, el análisis de la Fórmula 1 se ha basado en estadísticas generales, muchas veces difíciles de interpretar sin la herramienta adecuada. Con el avance de las tecnologías web y la disponibilidad de datos en tiempo real, se ha abierto la posibilidad de mejorar esta experiencia para los aficionados. En este contexto, el proyecto busca ofrecer una solución accesible y dinámica para todos aquellos interesados en profundizar en el mundo de la Fórmula 1.

El desarrollo de la aplicación se realiza utilizando Angular como marco tecnológico, lo que facilita la creación de una interfaz de usuario interactiva y eficiente. Además, la aplicación integra diversas APIs externas que permiten acceder a los datos más actuales sobre los resultados y estadísticas de las carreras, clasificaciones de pilotos y equipos, entre otros. Este enfoque asegura que la plataforma se mantenga actualizada y ofrezca una experiencia en tiempo real.

Uno de los elementos distintivos de este proyecto es el simulador de estrategias de carrera. Esta herramienta permite a los usuarios experimentar con diferentes variables y observar como la elección de neumáticos, los tiempos de pit stop y la duración de los stints puede influir en una simulación realista de las tácticas empleadas en las carreras de Fórmula 1. Además, la aplicación cuenta con un diseño adaptable a distintos dispositivos, garantizando una experiencia fluida y accesible en cualquier plataforma.

En resumen, este proyecto representa la convergencia entre la pasión por la Fórmula 1 y las capacidades tecnológicas actuales. Al integrar datos en tiempo real y simulaciones de estrategias, la aplicación busca enriquecer la experiencia de los aficionados y proporcionarles una herramienta útil tanto para el entretenimiento como para el análisis de las complejas decisiones estratégicas que definen las carreras.

1.1. Estructura de la memoria

- **Introducción:** Descripción general del propósito y contexto del proyecto, así como la estructura de la memoria y de los anexos.
- **Objetivos del proyecto:** Descripción de los objetivos generales y técnicos que se pretenden alcanzar con el proyecto, junto con los objetivos personales del autor.
- **Conceptos teóricos:** Explicación de los conceptos y fundamentos necesarios para comprender los aspectos clave del proyecto, como la estructura de la Fórmula 1 y el sistema de puntuación.
- **Técnicas y herramientas:** Detalle de las herramientas, tecnologías y técnicas utilizadas en el desarrollo de la aplicación, incluyendo Angular, APIs externas y la integración con Netlify.
- **Aspectos relevantes del desarrollo del proyecto:** Resumen de los principales logros y desafíos superados durante el desarrollo de la aplicación.
- **Trabajos relacionados:** Análisis de otros proyectos o aplicaciones similares, destacando sus fortalezas y debilidades en comparación con el proyecto actual.
- **Conclusiones y líneas de trabajo futuras:** Reflexión final sobre el impacto del proyecto, sus conclusiones y propuestas de mejora o ampliación para futuras versiones.

1.2. Materiales entregados

Enlaces a los materiales entregados del proyecto:

- Video demostración: https://youtu.be/Ur3CPtEAbiU

- Video descripción: https://youtu.be/2xJzHYHgfgc

- Repositorio del proyecto: https://github.com/Ixo22/F1nalLap

- Web del proyecto: http://f1nallap.netlify.app/

2. Objetivos del proyecto

En este apartado se detallan los objetivos que se buscan alcanzar con la realización de este proyecto, distinguiendo entre objetivos generales relacionados con los requisitos funcionales del software, objetivos técnicos necesarios para su desarrollo, y objetivos personales enfocados en las metas individuales que se esperan lograr durante el proceso de creación de la aplicación.

2.1. Objetivos generales

- **Desarrollar una aplicación web con Angular:** Crear una aplicación web interactiva utilizando el framework Angular, aplicando sus características de componentes y servicios para desarrollar una plataforma modular, escalable y de alto rendimiento.
- **Proporcionar información detallada sobre la Fórmula 1:** Desarrollar una interfaz que ofrezca acceso a datos actualizados sobre las temporadas actuales y pasadas de la Fórmula 1, incluyendo resultados de carreras, clasificaciones, estadísticas de pilotos y equipos.
- **Desarrollar un simulador de estrategias de carrera:** Implementar un simulador interactivo que permita a los usuarios experimentar con distintas configuraciones de neumáticos, tiempos de pit stop y otras variables para optimizar sus estrategias de carrera según las condiciones específicas de cada Gran Premio.
- **Gestionar y visualizar datos en tiempo real:** Obtener y mostrar datos de la Fórmula 1 a través de APIs externas, garantizando que los resultados, estadísticas y clasificaciones se mantengan actualizados en tiempo real.

- **Optimizar la experiencia del usuario:** Diseñar una interfaz visualmente atractiva e intuitiva que ofrezca una experiencia fluida en diferentes dispositivos, garantizando la accesibilidad a través de un diseño responsive.
- **Documentar y mantener el proyecto:** Desarrollar una documentación clara y detallada del proceso de desarrollo, incluyendo las decisiones tecnológicas y los métodos utilizados, para facilitar futuras actualizaciones y mantenimiento del sistema.

2.2. Objetivos técnicos

- **Integrar con APIs externas:** Implementar la integración con diversas APIs que proporcionen datos actualizados sobre la Fórmula 1, como resultados de carreras, clasificaciones, estadísticas de pilotos y equipos.
- **Desarrollar una app responsive con Angular:** Utilizar Angular y tecnologías web estándar como HTML [21], SCSS [35] y TypeScript [38] para garantizar que la aplicación sea totalmente funcional en dispositivos móviles, tabletas y escritorios.
- **Gestionar datos en tiempo real:** Establecer un sistema eficiente para la gestión y actualización de datos en tiempo real, utilizando APIs externas y optimizando la carga de datos de acuerdo con las necesidades de la aplicación.
- **Desarrollar e implementar una simulación de estrategias de carrera:** Crear el motor detrás del simulador de estrategias de carrera, implementando cálculos precisos para la simulación de paradas en boxes, degradación de neumáticos y tiempos de vuelta.
- **Implementar Netlify para el despliegue:** Configurar y desplegar la aplicación en Netlify, utilizando sus características de hosting continuo para mantener la aplicación siempre actualizada y disponible.

2.3. Objetivos personales

- **Aprender sobre desarrollo frontend con Angular:** Adquirir experiencia práctica en el desarrollo frontend [8] utilizando Angular, mejorando mis habilidades en la creación de aplicaciones web dinámicas y reactivas.

- **Profundizar en la integración y gestión de APIs:** Mejorar mis habilidades en la integración y manejo de APIs externas, aprendiendo a trabajar con datos en tiempo real de manera eficaz y segura.
- **Desarrollar habilidades en simulación y cálculo de estrategias de carrera:** Aprender a implementar simuladores y algoritmos complejos, como la simulación de estrategias de carrera en la Fórmula 1, con un enfoque en la optimización de variables dinámicas.
- **Mejorar mis capacidades en diseño de interfaces de usuario:** Incrementar mis habilidades en la creación de interfaces atractivas y funcionales, garantizando una experiencia de usuario positiva y accesible.
- **Adquirir experiencia con el despliegue y hosting de aplicaciones web:** Familiarizarme con el proceso de despliegue de aplicaciones web utilizando plataformas como Netlify, gestionando el ciclo de vida de la aplicación desde su desarrollo hasta su publicación en línea.
- **Documentar todo el proceso de desarrollo:** Mejorar mis habilidades en la documentación técnica del proyecto, asegurando que todo el proceso de desarrollo esté bien documentado para futuras referencias y mantenimiento.

3. Conceptos teóricos

3.1. Fórmula 1

Estrategia en la Fórmula 1

La estrategia de carrera en Fórmula 1, que incluye la elección de neumáticos [30], la gestión de paradas en boxes y el ajuste del rendimiento del coche, es fundamental para el éxito de los equipos y puede ser tan decisiva como la habilidad de los pilotos.

Neumáticos

Los neumáticos son clave en la F1, pues influyen directamente en el agarre y rendimiento del coche. Su tipo varía según las condiciones del circuito y la pista:

- **Neumáticos de seco (slicks):** Diseñados para condiciones secas, maximizan la adherencia y están disponibles en tres durezas principales:
  - **Blandos:** Máximo agarre y velocidad, pero con vida útil corta; ideales para fases iniciales o carreras cortas.

Figura 3.1: Neumático Blando.

- **Medios:** Equilibrio entre rendimiento y durabilidad, usados en la mayoría de estrategias para mantener ritmo competitivo.

Figura 3.2: Neumático Medio.

- **Duros:** Mayor durabilidad y menor agarre, adecuados para condiciones exigentes o finales de carrera.

Figura 3.3: Neumático Duro.

- **Neumáticos de lluvia (full wet):** Diseñados para lluvia intensa, con ranuras profundas que evitan el aquaplaning y compuesto blando para buen agarre en agua, aunque se desgastan rápido en seco. Son ideales para pistas completamente mojadas.

Figura 3.4: Neumático de lluvia.

- **Neumáticos intermedios:** Pensados para pistas húmedas o con lluvia ligera, con ranuras menos profundas que los de lluvia, ofrecen equilibrio entre agarre y durabilidad, ideales para condiciones cambiantes entre seco y mojado.

Figura 3.5: Neumático intermedio.

Estrategia de paradas

Las paradas en boxes son vitales para el resultado de la carrera, ya que su momento y número influyen en el rendimiento final. Las tácticas comunes incluyen:

- **Undercut:** Parar antes que el rival para aprovechar neumáticos nuevos y reducir tiempos, buscando adelantar al rival en boxes.
- **Overcut:** Permanecer más tiempo en pista para aprovechar el desgaste de los neumáticos del rival, manteniendo un ritmo constante.

Estas estrategias dependen de factores como la degradación de neumáticos, condiciones meteorológicas y cambios en la pista, requiriendo decisiones flexibles y en tiempo real.

Tipos de Circuitos

Los circuitos de F1 varían entre urbanos, con curvas cerradas y pocas oportunidades de adelantamiento, y tradicionales, que permiten mayor velocidad y maniobras, influyendo significativamente en las estrategias de carrera.

3.2. API

Una API [41] (Interfaz de Programación de Aplicaciones, por sus siglas en inglés) es un conjunto de definiciones y protocolos que permiten que un software o sistema interactúe con otros. Las APIs facilitan la comunicación entre diferentes aplicaciones, permitiendo el intercambio de datos y la ejecución de funciones específicas sin necesidad de conocer la implementación interna de las mismas.

Funcionamiento de una API

Una API se basa en un modelo de comunicación cliente-servidor, donde el cliente (generalmente una aplicación que utiliza la API) envía peticiones al servidor, y este último responde con los datos o resultados solicitados. Las peticiones a una API se realizan mediante protocolos HTTP [14], y las respuestas se envían en formatos estructurados como JSON [39] o XML [40].

El funcionamiento básico de una API es el siguiente:

- El cliente realiza una solicitud a la API utilizando una URL específica (conocida como endpoint).
- La solicitud puede incluir parámetros (datos adicionales) que especifican la información requerida.
- El servidor procesa la solicitud y devuelve una respuesta.
- La respuesta generalmente está en formato JSON o XML, y contiene los datos solicitados o el resultado de una operación.

Métodos HTTP Comunes en APIs

Las APIs utilizan diversos métodos HTTP [16] para definir la acción a realizar sobre los recursos del servidor. Los métodos más comunes son:

- **GET:** Se usa para obtener datos del servidor. Por ejemplo, para consultar los resultados de una carrera o los detalles de un piloto.
- **POST:** Permite enviar datos al servidor para crear nuevos recursos o enviar formularios.
- **PUT:** Se utiliza para actualizar datos existentes en el servidor, como modificar información de un piloto o constructor.
- **DELETE:** Sirve para eliminar datos del servidor, como borrar registros de pilotos o equipos.

¿Por qué usar una API?

Las APIs son esenciales porque permiten acceder a datos y servicios externos sin necesidad de desarrollar toda la funcionalidad internamente, agilizando el desarrollo y mejorando la eficiencia del software. En el caso de

la Fórmula 1, las APIs posibilitan el acceso a información actualizada sobre carreras, pilotos y resultados sin requerir la gestión manual de esos datos.

Además, las APIs facilitan la integración con otros servicios, como bases de datos externas o plataformas en la nube, permitiendo ofrecer funcionalidades avanzadas de forma rápida y con menor esfuerzo.

3.3. Integración de APIs en el Proyecto

En este proyecto se utiliza principalmente la API de Fórmula 1 de Jolpi [23] para obtener datos actualizados sobre la temporada, incluyendo clasificaciones de pilotos y constructores, calendario de carreras y resultados. Esta API ofrece información histórica y en tiempo real desde 1950 hasta la actualidad, facilitando el acceso a datos estructurados en formato JSON.

Además, se integran otras APIs como la API-Sports [7] para ampliar la cobertura de datos y páginas como F1Technical [17] para aspectos técnicos detallados, Autosport [11] para noticias y análisis, Motorsport.com [26] para la cobertura general de F1, y Reddit [12] para debates comunitarios sobre temas variados y a Pitpass [31] para obtener información adicional.

API del Campeonato de Pilotos

La API permite consultar la clasificación actual de pilotos para cualquier temporada mediante un endpoint que devuelve un JSON con datos como posición, puntos, número de victorias, y detalles personales del piloto (nombre, nacionalidad). Por ejemplo, para la temporada 2021, la URL utilizada es:

`https://api.jolpi.ca/ergast/f1/2021/driverstandings/`

Los campos más relevantes en la respuesta incluyen:

- **position:** Posición en la clasificación.
- **points:** Puntos acumulados.
- **wins:** Número de victorias.
- **Driver:** Información personal del piloto.
- **Constructors:** Datos del equipo asociado.

Esta información se procesa para mostrar en la interfaz los datos de los pilotos junto con sus imágenes oficiales.

API del Campeonato de Constructores

Similarmente, la API ofrece la clasificación actual de constructores para cualquier temporada a través del endpoint:

`https://api.jolpi.ca/ergast/f1/2021/constructorstandings/`

El JSON resultante incluye datos como posición, puntos, victorias y detalles del equipo. Se utiliza además una lista interna con logotipos para mejorar la visualización en la interfaz.

Lista Interna de Equipos

Para asociar los equipos con sus logotipos, se utiliza una lista interna (teamsData) obtenida de una API externa, que proporciona imágenes y nombres actuales de los equipos. Esta lista es esencial para mostrar gráficamente la información de constructores y pilotos con su respectivo equipo.

API de Carreras y Resultados

La API también proporciona el calendario de carreras, con detalles de fechas, circuitos y ubicaciones, mediante:

`https://api.jolpi.ca/ergast/f1/2025/`

Además, permite obtener resultados detallados de cada Gran Premio, incluyendo posiciones finales, puntos, tiempos y vueltas rápidas, a través de endpoints como:

`https://api.jolpi.ca/ergast/f1/2025/1/results/`

Los datos principales que se extraen incluyen:

- **position:** Posición final en la carrera.
- **points:** Puntos obtenidos.
- **Driver:** Nombre del piloto.
- **Constructor:** Nombre del equipo.
- **Time:** Tiempo total de carrera.
- **FastestLap:** Tiempo de la vuelta más rápida.

Estos datos se integran dinámicamente para mostrar calendarios actualizados y resultados detallados en la aplicación, incluyendo cuadros de diálogo interactivos para consultar información completa por carrera.

Importancia de las APIs en el Proyecto

La integración de estas APIs es fundamental para mantener la información actualizada, precisa y detallada sin necesidad de almacenar grandes volúmenes de datos localmente. El uso de servicios externos permite que el proyecto sea escalable, eficiente y se mantenga sincronizado con las temporadas oficiales de Fórmula 1.

Desarrollo y Consumo de APIs

Para integrar datos externos mediante APIs, las aplicaciones realizan solicitudes HTTP a endpoints específicos y procesan las respuestas recibidas, generalmente en formato JSON (JavaScript Object Notation). JSON es un formato ligero, fácil de leer y escribir, que representa datos estructurados mediante pares clave-valor.

Ejemplo de respuesta JSON

A continuación, se muestra un ejemplo simplificado de una respuesta JSON para la clasificación de pilotos:


```
"StandingsLists": [{
    "DriverStandings": [{
        "position": "1",
        "points": "131",
        "wins": "4",
        "Driver": {
            "givenName": "Oscar",
            "familyName": "Piastri",
            "nationality": "Australian"
        },
        "Constructors": [{
            "name": "McLaren",
            "nationality": "British"
        }]
    }]
}]
```

En este ejemplo:

- `position` indica la posición del piloto en la clasificación.
- `points` y `wins` reflejan los puntos acumulados y victorias respectivamente.
- `Driver` contiene datos personales del piloto.
- `Constructors` es una lista con el equipo al que pertenece el piloto.

Proceso básico para consumir una API

El flujo típico para consumir una API es:

1. La aplicación realiza una petición HTTP al endpoint deseado.
2. El servidor responde con un objeto JSON que contiene los datos solicitados.
3. La aplicación procesa el JSON para extraer la información relevante.
4. Los datos se utilizan para actualizar la interfaz de usuario o para otras operaciones internas.

Este mecanismo permite que la aplicación mantenga datos actualizados y dinámicos sin necesidad de gestionar grandes bases de datos propias, aprovechando servicios externos fiables y actualizados.

3.4. Simulador de Estrategias de Carrera

El simulador de estrategias de carrera implementado en este proyecto permite evaluar distintas combinaciones y duraciones de compuestos de neumáticos para una carrera de Fórmula 1. Su objetivo principal es facilitar la experimentación con estrategias para entender cómo afectan la gestión de neumáticos y las paradas en boxes al tiempo total de carrera y, por ende, al rendimiento global.

Modelado de la Degradación y Vida Útil de los Neumáticos

El simulador considera tres tipos principales de neumáticos: blandos, medios y duros, cada uno con una vida útil estimada (22, 33 y 50 vueltas respectivamente) [13] que determina el máximo de vueltas que se pueden recorrer sin perder excesivamente rendimiento.

El tiempo total de un stint se calcula como la suma del tiempo base por vuelta y la degradación acumulada a lo largo de todas las vueltas del stint:

$$\text{Tiempo total del stint} = \sum_{vuelta=1}^{\text{vueltas}} (\text{Tiempo Base} + \text{Degradación})$$

Donde el tiempo base depende del compuesto y el circuito, y la degradación se ajusta a medida que avanzan las vueltas.

La degradación del neumático se modela de forma no lineal y depende de la fase del stint (periodo de uso de un mismo compuesto) en que se encuentre el neumático. La fórmula utilizada para calcular la degradación de cada vuelta es la siguiente:

$$\text{Degradación} = \text{Random}(0,1) \times (max - min) + min$$

Ejemplo: Supongamos que el compuesto es blandos, la vuelta actual es 8 (fase media).

- 1. [min, max]: [0.15, 0.2]
- 2. Cálculo de la degradación (si Random(0,1) = 0.6):
  $\text{Degradación} = 0{,}6 \times (0{,}2 - 0{,}15) + 0{,}15 = 0{,}03 + 0{,}15 = 0{,}18$

Donde los valores de Max y Min dependen de la fase del compuesto (inicial, media, final).

- **Fase inicial:** Los neumáticos experimentan una degradación media-alta debido al esfuerzo inicial y las altas fuerzas sobre el compuesto.
- **Fase media:** La degradación disminuye, permitiendo mantener un ritmo más estable y eficiente durante un periodo prolongado.

- **Fase final:** La degradación aumenta rápidamente debido al desgaste acumulado, elevadas temperaturas y pérdida de goma, afectando negativamente el rendimiento.

Este enfoque dinámico permite simular con mayor realismo la evolución del rendimiento de los neumáticos durante la carrera.

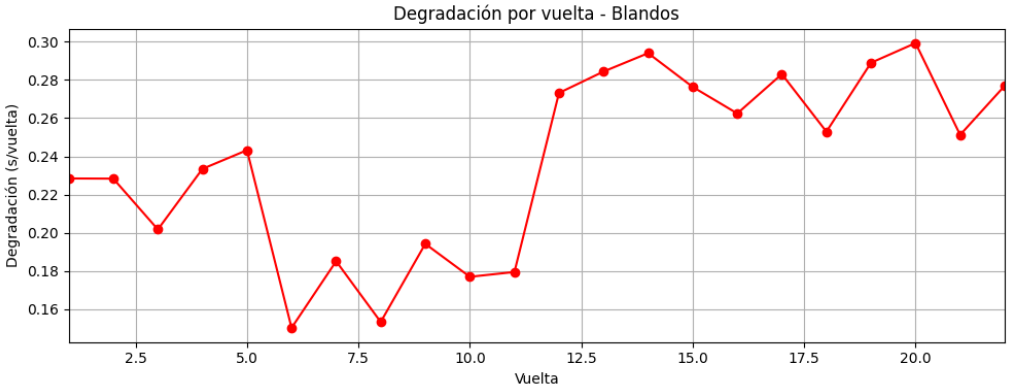

Figura 3.6: Degradación del compuesto Blando por fase.

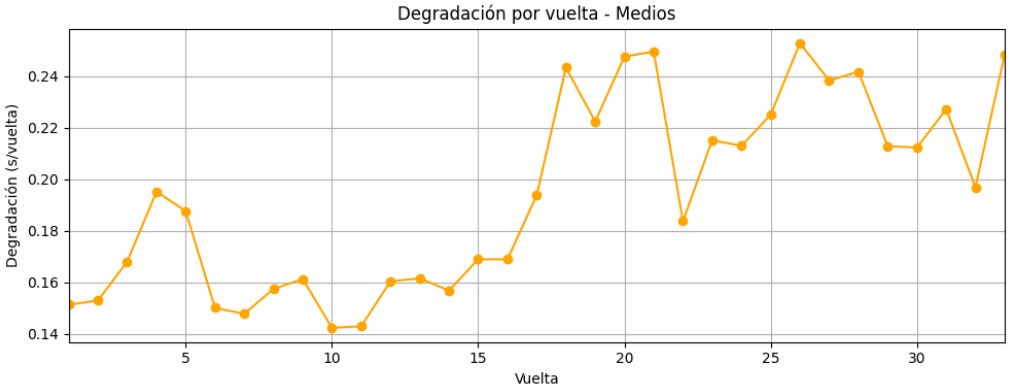

Figura 3.7: Degradación del compuesto Medio por fase.

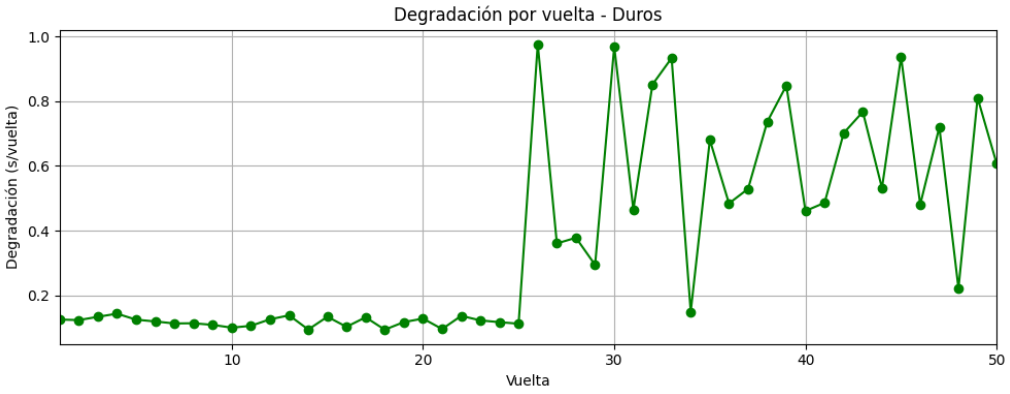

Figura 3.8: Degradación del compuesto Duro por fase.

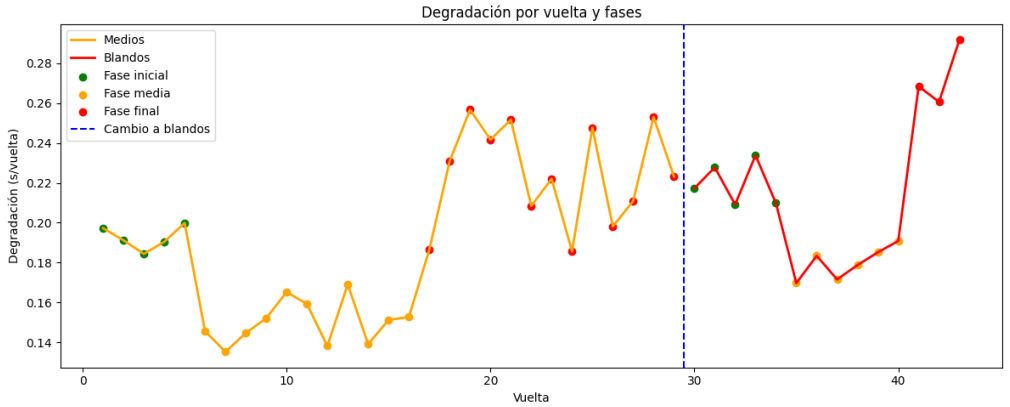

Figura 3.9: Degradación por vuelta y fase para una estrategia con dos compuestos.

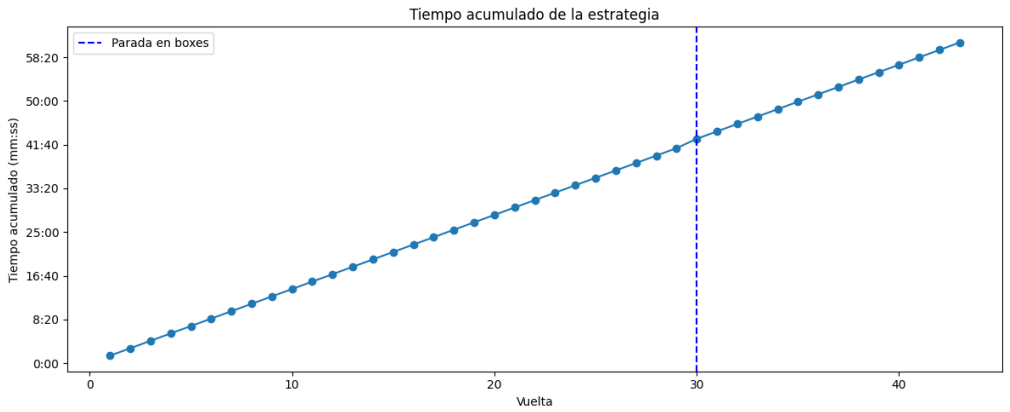

Figura 3.10: Tiempo total generado de la estrategia añadiendo degradación.

Restricciones y Ajustes de Compuestos

Para asegurar que las estrategias generadas sean realistas y viables según la normativa, el simulador incluye restricciones basadas en mínimos y máximos de vueltas por compuesto. Además, adapta automáticamente el compuesto a usar en función de la duración del stint, por ejemplo, cambiando a un compuesto más blando si el stint es demasiado corto para un compuesto más duro.

Generación y Evaluación de Estrategias

El sistema genera combinaciones de estrategias para una o dos paradas, incluyendo opciones con dos o tres compuestos distintos (aunque el uso de tres compuestos suele ser menos frecuente en la práctica). Cada estrategia propuesta se simula calculando el tiempo total de carrera, sumando los tiempos estimados por vuelta considerando la degradación y añadiendo el tiempo asociado a las paradas en boxes.

Para filtrar las estrategias, el simulador aplica criterios que identifican diferencias significativas entre ellas en cuanto a compuestos y duración de stints, descartando opciones demasiado similares para mostrar solo las alternativas relevantes.

Sistema de Penalizaciones y Bonificaciones

El simulador incorpora un sistema de penalizaciones para desalentar estrategias poco realistas o no óptimas, como el uso repetido del mismo compuesto en stints consecutivos (penalización significativa de tiempo). Por otro lado, premia las estrategias de una sola parada con una bonificación de tiempo, reflejando la ventaja competitiva de reducir paradas en boxes.

Resultado y Aplicaciones del Simulador

El resultado del simulador es una lista de estrategias optimizadas, ordenadas según el tiempo total estimado de carrera, que incluye la cantidad de paradas, los compuestos utilizados en cada stint y los tiempos calculados. Esta herramienta facilita a los usuarios entender el impacto de diferentes decisiones estratégicas y sirve como plataforma educativa para el análisis de la gestión de neumáticos en Fórmula 1.

Simulación de Estrategia Libre

Además de la generación automática de estrategias optimizadas, el simulador también permite la simulación de estrategias libres, donde el usuario decide manualmente la cantidad de paradas y los compuestos a utilizar en cada stint.

En este modo, el usuario puede definir la secuencia de compuestos y la duración de cada uno según sus preferencias o hipótesis, lo que facilita el análisis personalizado y la experimentación con diferentes escenarios estratégicos.

La simulación calcula el tiempo total de la carrera sumando los tiempos por vuelta ajustados por la degradación dinámica de cada compuesto, considerando además el tiempo asociado a las paradas en boxes entre stints. Este cálculo tiene en cuenta el aumento progresivo de la degradación en fases inicial, media y final del uso del neumático, como en la simulación automatizada.

La degradación del neumático, cuando excede su vida útil, sigue un patrón exponencial. La fórmula utilizada es la siguiente:

$$\text{Degradación} = 0{,}5 + (\text{vueltas fuera de vida útil})^{1{,}5} \times 1{,}5$$

Este modelo de degradación muestra un incremento acelerado del desgaste conforme el neumático supera su vida útil.

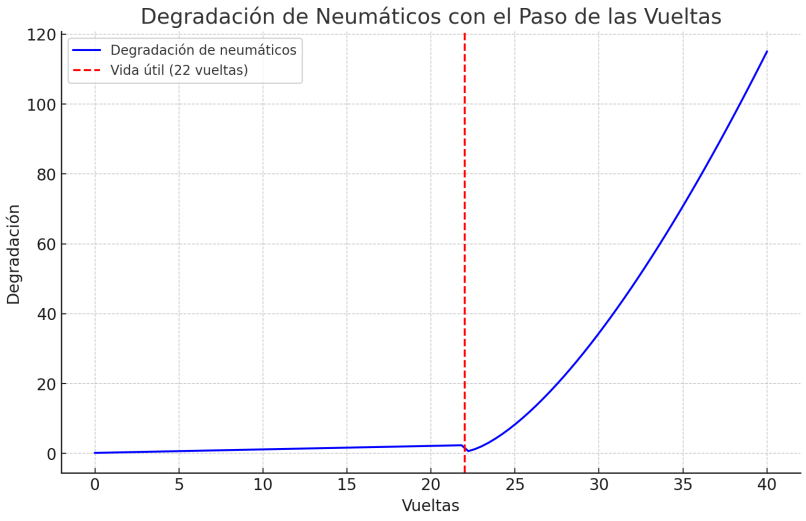

Figura 3.11: Degradación de los neumáticos una vez superada su vida útil.

Este enfoque flexible es ideal para usuarios avanzados que desean probar hipótesis concretas o comparar estrategias particulares sin las restricciones automáticas de generación de opciones, fomentando un aprendizaje más activo y una comprensión profunda del impacto de la gestión de neumáticos y paradas en boxes en el resultado final de la carrera.

4. Técnicas y herramientas

4.1. Angular

Angular [3] es un framework avanzado de JavaScript orientado al desarrollo de interfaces web modernas, especialmente útil para aplicaciones de una sola página (SPA)[19] y progresivas (PWA)[18]. Su arquitectura modular y su enfoque en las buenas prácticas permiten crear soluciones escalables, mantenibles y de alto rendimiento.

El trabajo con Angular se apoya en TypeScript, un lenguaje que amplía las capacidades de JavaScript añadiendo tipado estático y otras funcionalidades útiles para proyectos grandes. Aunque Angular puede utilizarse con JavaScript, el uso de TypeScript es el estándar recomendado por la comunidad y el propio equipo de desarrollo.

La elección de Angular para este proyecto se debe a su robustez, la amplia comunidad que lo respalda y la facilidad para estructurar aplicaciones complejas. Para iniciar y gestionar el proyecto, ha sido imprescindible el uso de Angular CLI [4].

Angular CLI

Angular CLI es la herramienta oficial de línea de comandos para Angular, diseñada para simplificar y agilizar el ciclo de vida de las aplicaciones desarrolladas con este framework.

Gracias a Angular CLI, es posible crear la estructura inicial de un proyecto en cuestión de segundos, así como generar componentes, servicios y módulos de manera automatizada. Además, facilita tareas como la ejecución

de pruebas, la compilación y el despliegue, y prepara los archivos necesarios para producción de forma eficiente.

Para su instalación, es necesario contar previamente con NodeJS [28]. La CLI se instala mediante el gestor de paquetes npm, lo que permite su integración sencilla en cualquier entorno de desarrollo.

4.2. Visual Studio Code (VSCode)

Visual Studio Code es un editor de código multiplataforma desarrollado por Microsoft, conocido por su rapidez y versatilidad. Su sistema de extensiones permite adaptar el entorno a las necesidades de cada desarrollador, ofreciendo soporte para numerosos lenguajes y herramientas.

Durante el desarrollo del proyecto, VSCode ha sido el editor principal, aprovechando extensiones específicas para Angular, TypeScript y Git, así como utilidades para depuración, control de versiones y terminal integrado, lo que ha incrementado la productividad y la calidad del código.

4.3. Netlify

Netlify [27] es una solución moderna para el despliegue y alojamiento de aplicaciones web y sitios estáticos. Ofrece integración continua, despliegue automático desde plataformas como GitHub y funcionalidades avanzadas como HTTPS, dominios personalizados y vistas previas de cambios antes de publicar.

En este trabajo, Netlify ha permitido automatizar el proceso de publicación de la aplicación, asegurando que cada actualización en el repositorio se refleje de inmediato en el entorno de producción y facilitando el acceso y revisión desde cualquier dispositivo.

4.4. TypeScript

TypeScript [38], desarrollado por Microsoft, es un lenguaje que extiende JavaScript [15] incorporando tipado estático y otras características orientadas a grandes proyectos. Su uso ayuda a detectar errores en etapas tempranas, mejora la legibilidad y facilita el mantenimiento del código.

Angular está pensado para funcionar principalmente con TypeScript, por lo que su integración en el proyecto ha sido esencial para aprovechar

todas las ventajas del framework y garantizar un desarrollo más seguro y eficiente.

4.5. HTML y CSS

HTML y CSS constituyen la base de cualquier desarrollo web. HTML se encarga de definir la estructura y el contenido de las páginas, mientras que CSS permite aplicar estilos y adaptar la presentación a distintos dispositivos y resoluciones.

En este proyecto, ambos lenguajes se han utilizado junto a Angular para construir la estructura de los componentes y personalizar la apariencia de la aplicación, logrando un diseño atractivo y funcional.

4.6. Angular Material

Angular Material [5] es una colección de componentes de interfaz de usuario basada en las directrices de Material Design, pensada para integrarse perfectamente con Angular. Incluye elementos como botones, menús, formularios y tablas, todos ellos accesibles y adaptables a dispositivos móviles.

El uso de Angular Material ha permitido acelerar el desarrollo de la interfaz, garantizando una experiencia visual coherente y profesional en toda la aplicación.

Ngx-Charts

Para la generación de gráficos interactivos y declarativos en Angular se ha utilizado ngx-charts [37], un framework de charting que facilita la visualización de datos complejos en componentes reutilizables

4.7. Git

Git es una herramienta de control de versiones distribuido que facilita la gestión de los cambios en el código fuente, permitiendo trabajar de forma colaborativa y segura. Permite mantener un historial completo de las modificaciones, gestionar ramas de desarrollo y fusionar cambios de manera eficiente.

GitHub

GitHub [20] es una plataforma online que complementa a Git, proporcionando alojamiento para repositorios, herramientas de colaboración, gestión de incidencias, integración continua y revisión de código. En este proyecto, GitHub ha sido clave para centralizar el código, coordinar el trabajo y automatizar el despliegue mediante Netlify.

4.8. Trello

Trello es una aplicación para la gestión visual de proyectos y tareas, pensada para facilitar la colaboración y el seguimiento del trabajo en equipo. Su interfaz, basada en tableros, listas y tarjetas, permite organizar y priorizar tareas de forma flexible y clara.

Entre sus principales ventajas destacan:

- Visualización clara del estado y avance de cada tarea o fase del proyecto.
- Gestión ágil de tareas, desde las más simples hasta las más complejas, mediante tarjetas que se pueden mover entre listas según su progreso.
- Herramientas colaborativas como asignación de responsables, comentarios y archivos adjuntos.
- Seguimiento detallado de los objetivos y cumplimiento de plazos.
- Acceso compartido y en tiempo real para todos los miembros del equipo.

La elección de Trello se debe a su facilidad de uso y su capacidad para adaptarse a las necesidades cambiantes del proyecto, permitiendo una gestión eficiente y visual de todas las fases del desarrollo.

4.9. LaTeX y Overleaf

LaTeX es un sistema avanzado de composición de documentos, especialmente valorado en el ámbito académico y científico por la calidad tipográfica que ofrece y su capacidad para gestionar documentos extensos y complejos.

Overleaf, por su parte, es una plataforma en línea que permite editar, compilar y colaborar en documentos LaTeX de manera sencilla y en tiempo

real. Para la elaboración de la memoria del proyecto, Overleaf ha facilitado la redacción colaborativa y el control de versiones del documento final.

5. Aspectos relevantes del desarrollo del proyecto

5.1. Ciclo de Vida del Proyecto

El proyecto comenzó con la creación del repositorio en GitHub en noviembre, momento en el que definí la idea general y los objetivos de la aplicación. Durante los meses siguientes, hasta principios de abril, no empecé el desarrollo activo ni la implementación del código. Este periodo fue fundamental para planificar el flujo de trabajo y para que pudiera aprender progresivamente Angular, que es el framework utilizado para construir la aplicación.

Durante este tiempo también realicé una búsqueda exhaustiva para encontrar las APIs que mejor se adaptaran a las necesidades del proyecto y que ofrecieran la información más completa y actualizada sobre la Fórmula 1. Inicialmente utilicé una API gratuita durante 2024, pero al cambiar su modelo a pago en 2025, tuve que modificar varios métodos e implementar nuevas soluciones para mantener la calidad y continuidad de los datos. Este cambio me llevó a adoptar la API que estoy usando actualmente, que se ajusta mejor a los requisitos del proyecto y ofrece un acceso más estable y completo a la información necesaria.

El desarrollo se ha llevado a cabo siguiendo una metodología Scrum [9], con sprints planificados que han permitido avanzar de manera ordenada y adaptativa. Además, se han mantenido reuniones periódicas con el tutor para evaluar el rumbo de la aplicación, revisar avances y tomar decisiones sobre los siguientes pasos, garantizando que el proyecto mantuviera la dirección correcta y cumpliera con los objetivos establecidos.

5.2. Análisis

En la fase de análisis definí de manera precisa los requerimientos funcionales y no funcionales de la aplicación, basándome en la idea inicial de crear una plataforma web centrada en la Fórmula 1 que integrara datos reales y herramientas de simulación estratégica. Realicé un estudio exhaustivo de las APIs públicas disponibles para la Fórmula 1, evaluando su cobertura, actualizaciones y limitaciones para seleccionar las más adecuadas.

Durante esta etapa, identifiqué las funcionalidades clave que debía incluir la aplicación, como la visualización en tiempo real del campeonato de pilotos y constructores, consulta de calendarios y resultados de carreras, simulación dinámica de estrategias de neumáticos, y una interfaz responsiva e intuitiva orientada a mejorar la experiencia del usuario.

También anticipé y analicé los retos técnicos, principalmente relacionados con la gestión eficiente de múltiples llamadas API, el manejo de datos asincrónicos, y la implementación de un simulador con parámetros realistas y ajustables según circuito y neumáticos.

Paralelamente, se consideraron aspectos de usabilidad para asegurar una interfaz clara y accesible, pensada tanto para aficionados como para usuarios con conocimientos técnicos.

Cabe destacar que esta fase contó con revisiones periódicas junto con el tutor, para validar el enfoque y garantizar que el proyecto avanzaba en línea con los objetivos establecidos, siguiendo la metodología Scrum con sprints iterativos.

5.3. Diseño y Experiencia de Usuario

El diseño visual y la experiencia de usuario han sido aspectos fundamentales en el desarrollo de la aplicación para garantizar una navegación intuitiva y una presentación clara de la información, alineada con la identidad visual de la Fórmula 1.

Paleta de Colores

Se ha optado por una paleta de colores inspirada en los elementos icónicos de la Fórmula 1, predominando el negro, rojo y blanco, que reflejan dinamismo y modernidad. Estos colores se emplean para destacar información relevante y crear contrastes que faciliten la lectura y el enfoque visual.

Figura 5.1: Paleta de colores de la web.

Diseño Responsive

La aplicación está diseñada con un enfoque responsive, asegurando que la interfaz se adapte correctamente a diferentes tamaños de pantalla y dispositivos, desde móviles y tablets hasta ordenadores de escritorio. Esto permite ofrecer una experiencia coherente y accesible para todos los usuarios, independientemente del dispositivo que utilicen.

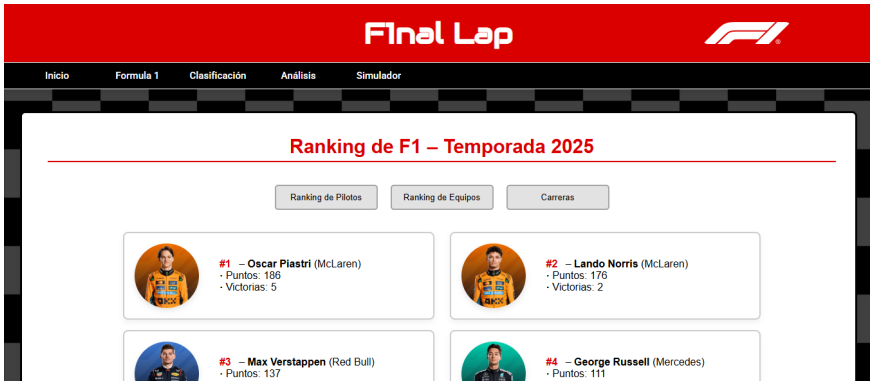

Figura 5.2: Diseño responsive para una pantalla grande (1.500px).

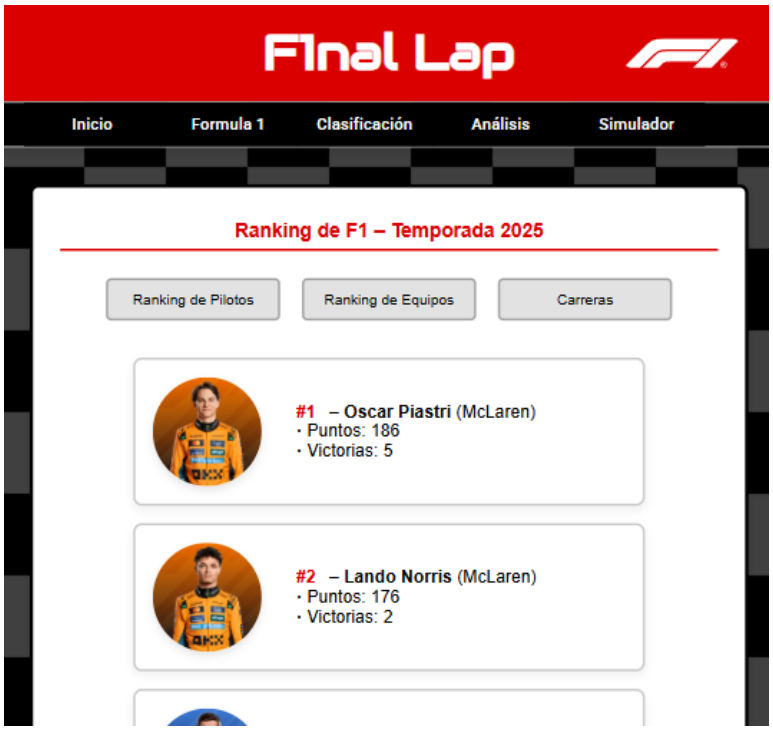

Figura 5.3: Diseño responsive para una pantalla mediana (768px).

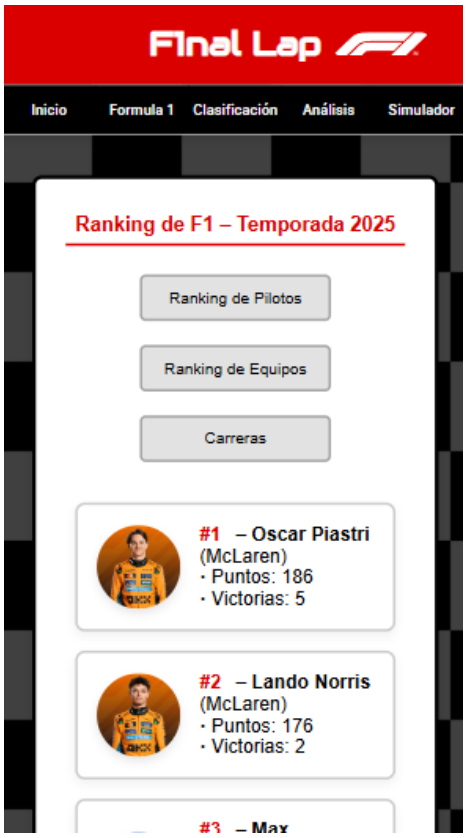

Figura 5.4: Diseño responsive para una pantalla pequeña (390px).

Organización Visual y Jerarquía

La interfaz se estructura con una jerarquía clara, utilizando tipografías legibles y tamaños adecuados para diferenciar títulos, subtítulos y contenido. Los componentes visuales, como tarjetas y listas, se diseñaron para mostrar la información de forma ordenada y destacada, facilitando la comprensión rápida de los datos.

Navegación y Accesibilidad

Se implementó una navegación sencilla y accesible, con menús claros y rutas intuitivas para facilitar el acceso a las secciones principales: temporadas actual y pasadas, clasificación de pilotos y equipos, y el simulador de estrategias. Además, se consideraron aspectos de accesibilidad como buen contraste y textos alternativos para imágenes.

5.4. Formación

Dado que muchas de las tecnologías utilizadas en este proyecto no formaron parte de mi formación universitaria, dediqué tiempo a investigar y formarme mediante cursos y tutoriales que me permitieron comprender y dominar estas herramientas.

Angular

Aunque conocía Angular, no había trabajado en proyectos personales con este framework, por lo que decidí profundizar en su aprendizaje durante el desarrollo del proyecto. Para ello, realicé varios cursos y tutoriales en YouTube que fortalecieron mis conocimientos en aspectos clave como la arquitectura basada en componentes, servicios, enrutamiento, y la integración con APIs.

Destacan los siguientes recursos formativos:

- Tour of Heroes (tutorial oficial de Angular)[6]: Introducción práctica a la creación de componentes, servicios y enrutamiento básico.
- Fundamentos de Angular (tutoriales de YouTube)[24]: Refuerzo de conceptos esenciales para un uso avanzado del framework.

- Integración de APIs externas en Angular (tutoriales de YouTube): Aprendizaje del uso de HttpClient para gestionar peticiones, respuestas y errores.
- Diseño con Angular Material (tutoriales de YouTube): Uso de componentes modernos para interfaces atractivas y responsivas.
- Personalización de temas en Angular Material (tutoriales de YouTube): Creación y gestión de temas personalizados y paletas de colores.

Netlify

Para el despliegue y alojamiento de la aplicación, utilicé Netlify, lo que implicó aprender a configurar y publicar la web en esta plataforma. Para ello, realicé tutoriales en YouTube que me guiaron en:

- La integración con repositorios de GitHub para despliegue automático.
- La configuración de dominios personalizados y HTTPS.
- El manejo de entornos y variables de configuración.
- La optimización del proceso de build y despliegue.
- Esta formación fue esencial para que la aplicación estuviera disponible en la web con un despliegue ágil y seguro.

5.5. Implementación y Desarrollo

La fase de implementación y desarrollo ha sido el núcleo del proyecto, donde se han materializado las ideas y diseños planteados en etapas previas. En este apartado, se describen los aspectos más relevantes y las decisiones técnicas que han marcado el desarrollo.

Arquitectura y Estructura del Proyecto

He optado por utilizar Angular como framework principal para el desarrollo frontend, aprovechando su arquitectura basada en componentes para organizar la aplicación de forma modular y escalable. Esto ha facilitado la reutilización de componentes y la gestión eficiente del estado a través de señales (signals) y servicios reactivos.

El proyecto se estructura en carpetas que separan claramente las funcionalidades, componentes, servicios y modelos de datos, facilitando la mantenibilidad y la escalabilidad.

Integración de APIs

La integración con la API externa de Fórmula 1 ha sido uno de los retos principales. Para garantizar un consumo eficiente y limpio de la API, se ha implementado un servicio dedicado que centraliza las llamadas HTTP y maneja la transformación de los datos JSON en modelos utilizables por la aplicación.

Se aplicaron buenas prácticas como el manejo de errores con RxJS, uso de operadores pipe para transformación y filtrado, y la suscripción controlada para evitar fugas de memoria.

```
// Servicio para cargar la clasificación de pilotos
loadDrivers() {
  this.isDrivers.set(true);
  this.carreras.set([]);
  this.http.get<DriverApiResponse>(DRIVER_URL).pipe(
    map(res =>
      // Extraemos la lista de pilotos del JSON anidado
      res.MRData.StandingsTable. +
      StandingsLists?.[0]?.DriverStandings ?? []
    ),
    // Mapeo a modelo interno
    map(list => list.map(ds => this.mapDriver(ds))),
    // Manejo de errores para evitar fallos en la app
    catchError(() => of([]))
  ).subscribe(data => this.items.set(data));
}
```

```
// Método para pasar los datos de API a la estructura usada
private mapDriver(ds: DriverStanding): DisplayItem {
  const family = ds.Driver.familyName;
  return {
    position: ds.position,
    points: ds.points,
    wins: ds.wins,
    driverName: `${ds.Driver.givenName} ${family}`,
    teamName: ds.Constructors[0]?.name ?? '-',
```

```
    imageUrl:
    `https://media.formula1.com/image/upload/f_auto, +
    c_limit,q_auto,w_1320/content/dam/fom-website/ +
    drivers/2025Drivers/${family}.jpg`  };
}
```

Este fragmento ejemplifica un patrón de consumo de API REST eficiente y claro, con manejo de errores, transformación limpia y separación de responsabilidades. Además, facilita futuras extensiones, como incorporar nuevos endpoints o campos.

Gestión del Estado y Reactividad

Se ha utilizado un enfoque reactivo para la gestión del estado en la aplicación, implementando señales y stores para actualizar la UI de forma eficiente cuando los datos cambian. Esto optimiza el rendimiento al evitar renderizados innecesarios y facilita la sincronización de datos entre componentes.

Diseño UI y UX

Para la interfaz de usuario se ha utilizado Angular Material, lo que permitió construir una experiencia visual coherente y responsiva con componentes accesibles y estilizados. Se ha personalizado la paleta de colores para adecuarla a la temática del proyecto, asegurando buena legibilidad y contraste.

Simulador de Estrategias de Carrera

El simulador es una pieza clave de la aplicación. Su implementación ha implicado el diseño de algoritmos que modelan la degradación de neumáticos, el cálculo del tiempo de carrera y la generación de estrategias de paradas óptimas.

```
// Función principal que simula una estrategia completa de carrera
private simularEstrategia(
  circuito: any,
  vueltasTotales: number,
  estrategia: [keyof typeof this.vidaUtil, number][]
): [number, [keyof typeof this.vidaUtil, number][]] {
```

```
  let tiempoTotal = 0;
  const estrategiaAjustada:
        [keyof typeof this.vidaUtil, number][] = [];
  const compuestosUsados = new Set<keyof typeof this.vidaUtil>();
  const tiemposBase = this.calcularTiemposBase(circuito);

  for (const [compuesto, vueltas] of estrategia) {
    const compuestoAjustado =
        this.ajustarCompuesto(compuesto, vueltas);

    if (vueltas > this.vidaUtil[compuestoAjustado]) {
      throw new Error(`El compuesto ${compuestoAjustado}
      no puede durar más de
      ${this.vidaUtil[compuestoAjustado]} vueltas.`);
    }

    const tiempoBase = tiemposBase[compuestoAjustado];
    const tiempoStint =
        this.calcularTiempoStint(
            compuestoAjustado, vueltas, tiempoBase
        );

    tiempoTotal += tiempoStint + this.tiempoParada;
    estrategiaAjustada.push([compuestoAjustado, vueltas]);
    compuestosUsados.add(compuestoAjustado);
  }

  if (compuestosUsados.size < 2) {
    throw new Error("La estrategia debe usar al menos dos
    compuestos diferentes según la normativa.");
  }

  return [tiempoTotal - this.tiempoParada, estrategiaAjustada];
}
```

Se ha seguido un enfoque modular, separando la lógica del simulador en un servicio dedicado que puede ser fácilmente probado y mantenido. Además, se incorporaron penalizaciones y bonificaciones para reflejar condiciones reales de carrera, aportando realismo a la simulación.

Despliegue y Automatización

El despliegue se ha automatizado mediante Netlify, integrando el repositorio de GitHub para que cada cambio en el código se refleje automáticamente en la versión publicada. Esto ha permitido un ciclo de desarrollo ágil y seguro, con versiones estables siempre disponibles para pruebas y demostraciones.

6. Trabajos relacionados

El desarrollo de la aplicación web F1nalLap se inserta en un contexto en el que ya existen diversos proyectos relacionados con el análisis de la Fórmula 1, el simulador de estrategias y el tratamiento de datos en tiempo real. A continuación, se describen algunos de los trabajos y proyectos más relevantes en este campo, con el fin de contextualizar el presente proyecto y destacar sus características diferenciadoras.

6.1. Plataformas de análisis de Fórmula 1

Existen varias plataformas dedicadas al análisis de la Fórmula 1, muchas de las cuales proporcionan estadísticas detalladas sobre las temporadas, pilotos y equipos. Un ejemplo destacado es el sitio oficial de la Fórmula 1 [1], que ofrece información sobre resultados de carreras, clasificaciones y noticias, pero no incluye herramientas de simulación de estrategias de carrera.

Otra plataforma relevante es RaceFans [32], que se especializa en proporcionar análisis en profundidad de las carreras, pero carece de una herramienta interactiva para simular y optimizar estrategias de carrera en tiempo real.

6.2. Simuladores de estrategias de carrera

En el campo específico de los simuladores de estrategias de carrera, existen trabajos como los desarrollados por *Formula 1's own strategy tools*, que permiten a los equipos profesionales de la F1 simular las mejores estrategias de paradas en boxes y uso de neumáticos. Sin embargo, estos simuladores están limitados a los equipos de competición y no están disponibles públicamente para los aficionados. A diferencia de estos, el simulador desarrollado

en F1nalLap está diseñado para ser accesible a los usuarios generales y permite una personalización más profunda, considerando variables como la degradación dinámica de los neumáticos y el ajuste de los tiempos de parada.

6.3. Integración de APIs en plataformas de deportes

El uso de APIs externas para obtener datos en tiempo real es una práctica común en las plataformas deportivas. Proyectos como *SportsRadar*[36] o *API-Football*[33] han sido pioneros en la integración de datos de partidos en vivo, clasificaciones y estadísticas. Sin embargo, en el caso de la Fórmula 1, la API de Jolpi utilizada en este proyecto es una de las más completas y accesibles para obtener datos detallados sobre las temporadas, pilotos y equipos en tiempo real.

6.4. Diferenciación del proyecto

A diferencia de los trabajos y plataformas mencionados, F1nalLap se diferencia principalmente en su capacidad para integrar y simular estrategias de carrera de manera interactiva, teniendo en cuenta factores como la degradación de los neumáticos, la duración de las paradas en boxes y las penalizaciones asociadas a cada estrategia. Además, la aplicación está diseñada para ser completamente responsive, lo que permite una experiencia óptima en dispositivos móviles y de escritorio.

En resumen, aunque existen diversas plataformas y simuladores relacionados con la Fórmula 1, F1nalLap aporta un valor añadido al ofrecer una herramienta interactiva y detallada para la simulación de estrategias de carrera, integrando datos en tiempo real de manera accesible y atractiva para los usuarios.

6.5. Comparativa de características entre plataformas relacionadas

En esta sección se presenta una tabla comparativa de las principales características ofrecidas por algunas plataformas destacadas en el ámbito de la Fórmula 1. La tabla incluye aspectos clave como simuladores de

estrategias, noticias, estadísticas y otros servicios que ofrecen cada una de las plataformas.

Características	**F1nalLap**	**Formula1.com**	**RaceFans**
Simulador de estrategias	X		
Noticias		X	X
Resultados en tiempo real	X	X	X
Estadísticas de pilotos y equipos	X	X	X
Calendario de carreras	X	X	X
Análisis de estrategias profesionales			
Acceso público	X	X	X
Interactividad	Alta	Baja	Baja
Gráficos y visualizaciones de datos	X		
Estadísticas históricas	X	X	X
App para Android o iOS		X	
Rediseño visual	X	X	X
Gratuito	X		X

Tabla 6.1: Tabla comparativa de características entre plataformas relacionadas

La tabla anterior muestra las principales características de las plataformas relevantes en el ámbito de la Fórmula 1, comparando servicios como simuladores de estrategias, noticias, resultados en tiempo real y otros aspectos clave. Como se observa, F1nalLap se destaca por su simulador de estrategias, mientras que otras plataformas, como *Formula1.com* y *RaceFans*, se centran más en proporcionar noticias y resultados en tiempo real.

7. Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

El proyecto F1nalLap ha sido un esfuerzo exitoso en la creación de una aplicación web interactiva dedicada a la Fórmula 1, que no solo proporciona datos detallados y en tiempo real sobre las temporadas de este deporte, sino que también permite a los usuarios simular y analizar estrategias de carrera. A lo largo del desarrollo de la aplicación, se han alcanzado los siguientes logros:

- **Desarrollar una plataforma dinámica y accesible:** La aplicación permite a los usuarios consultar resultados, clasificaciones y estadísticas tanto de temporadas actuales como pasadas de manera eficiente y accesible desde cualquier dispositivo.

- **Simular estrategias de carrera:** El simulador ha demostrado ser una herramienta útil para la optimización de las decisiones estratégicas en la Fórmula 1. Los usuarios pueden experimentar con diversas configuraciones de neumáticos y tiempos de pit stop, con una simulación realista que toma en cuenta factores como la degradación de neumáticos y las penalizaciones.

- **Integrar efectivamente APIs:** La integración de APIs externas ha sido crucial para garantizar la actualización en tiempo real de los datos, mejorando la experiencia del usuario al acceder a información precisa sobre las carreras y clasificaciones.

- **Desarrollar utilizando tecnologías modernas:** El uso de Angular y TypeScript ha permitido crear una aplicación escalable y modular, garantizando un alto rendimiento y facilidad de mantenimiento. Además, la implementación de una arquitectura responsive asegura que la experiencia de usuario sea coherente en dispositivos móviles, tabletas y escritorios.

- **Aprender y formarse continuamente:** Durante el desarrollo, se ha aprendido y profundizado en diversas herramientas y tecnologías como Angular, APIs REST, y despliegue en Netlify, contribuyendo a la mejora de las habilidades técnicas y el conocimiento en desarrollo frontend.

7.2. Líneas de trabajo futuras

Aunque el proyecto ha alcanzado sus objetivos iniciales, hay varias áreas en las que se puede continuar trabajando para mejorar la aplicación y expandir sus funcionalidades:

- **Ampliar las opciones del simulador:** Se podría implementar la simulación de estrategias más complejas, como la opción de tres paradas o la personalización de condiciones meteorológicas, lo cual añadiría mayor realismo y flexibilidad al simulador. La mejora del simulador podría beneficiarse de las herramientas de visualización avanzada de D3.js [29], que permitiría realizar gráficos aún más detallados y dinámicos.

- **Mejorar la interfaz de usuario (UI):** A pesar de que la aplicación es responsiva, se podría optimizar aún más el diseño para mejorar la accesibilidad para usuarios con necesidades especiales, añadiendo opciones como el modo oscuro, mejor contraste y soporte para lectores de pantalla.

- **Optimizar el rendimiento en tiempo real:** A medida que aumente el volumen de datos accesibles a través de las APIs, sería beneficioso implementar un sistema de caché para reducir la latencia y la carga en el servidor, mejorando la velocidad de respuesta de la aplicación.

- **Ampliar las fuentes de datos:** Incluir más fuentes de datos de APIs o bases de datos propias que ofrezcan información más detallada, como estadísticas sobre las condiciones del circuito, el estado del clima en

tiempo real y las decisiones estratégicas previas a cada carrera, para enriquecer el análisis de estrategias. Plataformas como Medium [10] o Towards Data Science [34] ofrecen valiosa información sobre el análisis y visualización de datos relacionados con la Fórmula 1

- **Integrar características sociales:** Una funcionalidad interesante sería permitir que los usuarios compartan sus simulaciones de estrategias o compitan por obtener la mejor estrategia, promoviendo la interacción social y el aspecto competitivo entre los aficionados.
- **Adaptar a otras categorías de automovilismo:** En el futuro, podría ampliarse la aplicación para incluir otras competiciones de automovilismo, como la MotoGP [25] o la IndyCar [22], con sus respectivas características y simuladores de estrategias.

Con estos avances y mejoras, el proyecto tiene un gran potencial para seguir evolucionando y ofrecer a los usuarios una herramienta aún más poderosa para el análisis de la Fórmula 1 y otros deportes de motor.

Bibliografía

[1] Formula 1. Formula 1 - the official f1 website. https://www.formula1.com, 2024.

[2] Alberto García Alcolado. F1nallap. https://f1nallap.netlify.app/, 2025.

[3] Angular. Angular. https://angular.io, 2025.

[4] Angular. The angular cli. https://angular.dev/tools/cli, 2025.

[5] Angular. Angular material. https://material.angular.io, 2025.

[6] Angular. Tour of heroes application and tutorial. https://v17.angular.io/tutorial/tour-of-heroes, 2025.

[7] API-Sports. Api-sports real-time sports data. https://api-sports.io/, 2024.

[8] Arsys. Frontend: ¿qué es y para qué se utiliza en desarrollo web? https://www.arsys.es/blog/frontend-que-es-y-para-que-se-utiliza-en-desarrollo-web, 2025.

[9] Atlassian. Qué es scrum y cómo empezar. https://www.atlassian.com/es/agile/scrum, 2025.

[10] Medium Authors. Analyzing f1 data with python and r. https://medium.com/data-driven-formula1, 2025.

[11] Autosport. Autosport – f1 news, analysis and commentary. https://www.autosport.com/f1/, 2025.

[12] Reddit Community. r/formula1. https://www.reddit.com/r/formula1/, 2025.

[13] Instituto de Automovilismo Deportivo. Análisis técnico de los neumáticos de fórmula 1. https://iad.la/blogs/analisis-tecnico-de-los-neumaticos-en-la-formula-1/#:~:text=Duraci%C3%B3n%3A%20Aproximadamente%2025%2D40%20vueltas%20dependiendo%20de%20las%20condiciones., 2025.

[14] MDN Web Docs. Http. https://developer.mozilla.org/es/docs/Web/HTTP, 2025.

[15] MDN Web Docs. Javascript. https://developer.mozilla.org/es/docs/Web/JavaScript, 2025.

[16] MDN Web Docs. Métodos de petición http. https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Methods, 2025.

[17] F1Technical.net. F1technical – the technical side of formula 1. https://www.f1technical.net, 2025.

[18] Geeksforgeeks. What is a progressive web app (pwa)? https://www.geeksforgeeks.org/what-is-a-progressive-web-app-pwa/, 2025.

[19] Geeksforgeeks. What is spa (single page application) in angularjs ? https://www.geeksforgeeks.org/angular-js/what-is-spa-single-page-application-in-angularjs/, 2025.

[20] GitHub. Github: Where the world builds software. https://github.com, 2025.

[21] HTML. Html: Lenguaje de etiquetas de hipertexto. https://developer.mozilla.org/es/docs/Web/HTML, 2025.

[22] IndyCar. Indycar - official site. https://www.indycar.com/, 2024.

[23] Jolpi. Api de fórmula 1. https://api.jolpi.ca/ergast/f1, 2025.

[24] midulive. Aprende angular 17 desde cero para principiantes gratis. https://www.youtube.com/watch?v=f7unUpshmpA&t=2081s&ab_channel=midulive, 2025.

[25] MotoGP. Motogp - campeonato mundial de motogp. https://www.motogp.com/es, 2024.

[26] Motorsport. Motorsport.com – formula 1 coverage. https://www.motorsport.com/f1/, 2025.

[27] Netlify. Netlify: Cloud platform for web hosting and serverless functions. https://www.netlify.com, 2025.

[28] NodeJS. Run javascript everywhere. https://nodejs.org/en, 2025.

[29] Observable. Introduction to d3.js. https://observablehq.com/@d3/learn-d3, 2025.

[30] Pirelli. Pirelli motorsport – neumáticos de f1. https://www.pirelli.com/tyres/es-es/motorsport/f1/neumaticos, 2025.

[31] Pitpass. Pitpass – inside formula 1. https://www.pitpass.com, 2025.

[32] RaceFans. Racefans - formula 1 news, analysis and opinion. https://www.racefans.net, 2025.

[33] RapidAPI. Api-football. https://rapidapi.com/api-sports/api/api-football, 2025.

[34] Towards Data Science. Visualizing motorsport data with python. https://towardsdatascience.com/visualizing-motorsport-data-123456, 2025.

[35] SCSS. Scss: Css with superpowers. https://sass-lang.com/, 2025.

[36] Sportradar. The sports technology company. https://sportradar.com/, 2025.

[37] Swimlane. ngx-charts – getting started guide. https://swimlane.gitbook.io/ngx-charts/, 2025.

[38] TypeScript. Typescript. https://www.typescriptlang.org, 2025.

[39] W3Schools. What is json. https://www.w3schools.com/whatis/whatis_json.asp, 2025.

[40] W3Schools. What is xml. https://www.w3schools.com/whatis/whatis_xml.asp, 2025.

[41] Wikipedia. Api. https://es.wikipedia.org/wiki/API, 2025.