1.- Introducción:

Desde hace un tiempo hasta ahora nuestra sociedad se ha visto sacudida por varios escándalos en cuanto a la interpretación de los sucesos que ocurren en todo el mundo. No hablamos de las tensiones políticas que puedan existir entre países, o problemas gravísimos como las situaciones que se están viviendo en algunos países de Oriente Medio. A lo que nos referimos es a la manipulación de la información, a cómo ciertos medios pueden llegar a tergiversar cierta información para su beneficio o simplemente por el hecho de tener a la población “engañada”.

Este fenómeno ha existido toda la vida, en mayor o menor medida, pero quizás es hoy en día cuando tiene un impacto mucho mayor en nuestro día a día, por el efecto de Internet y las redes sociales, que magnifican todo mucho más. Tenemos ejemplos de esta manipulación y como ha llevado incluso a sugestionar a ciudadanos para distintos intereses, algunos ejemplos de ello son [el impacto de las "fake news" en las elecciones de EEUU](https://www.lavanguardia.com/tecnologia/20190408/461469794060/fake-news-propaganda-desinformacion-trump.html), en las [elecciones brasileñas](https://www.elespanol.com/mundo/america/20181016/fake-news-inundan-campana-electoral-brasil/345716563_0.html), y no sólo en este campo afectan este tipo de bulos, si no que también a otros aspectos de la vida, tan importantes como la [salud pública](https://saludsinbulos.com/vacunas/noticias-falsas-vacunas-eficacia-campanas/).

Cambiando un poco de tema, esta sobreexposición a información en algunas ocasiones es contraproducente, ya que cuando quieres saber algo al instante pueden ocurrir dos cosas principalmente, que no sepas donde buscarlo o que no te fíes de todos los sitios donde se refleje eso que buscas.

Conociendo esta serie de problemas he decidido enfocar el proyecto en esta dirección, en ofrecer información de la manera más rigurosa y completa posible, eso sí, centrándome en resultados deportivos, y más concretamente en resultados de Fórmula 1, algo que quizás no es tan polémico como los ejemplos anteriores pero que a fin de cuentas es bastante información que debe ser organizada y mostrada coherentemente a un público, que quizás en España un poco menos, pero que va constantemente en auge.

Aparte del propósito anterior, la finalidad del proyecto es exponer todos los datos que haya disponibles relacionados con la Fórmula 1, de forma clara, concisa y directa y lo mejor de todo, de forma automatizada y al momento. Para ello nos basaremos en distintas fuentes de código abierto y gratuitas, creando una herramienta fácil de usar y sin ningún coste.

Antes de iniciar el proyecto, y para la realización de este se han tenido que investigar una serie de tecnologías. El resultado de este estudio se ha plasmado en los siguientes apartados.

**1.1 Estudio del estándar API REST**

Buscando una definición sencilla, REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.

REST cambió por completo la ingeniería de software a partir del 2000. Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por [Roy Fielding](https://roy.gbiv.com/), el padre de la especificación HTTP y uno los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su estudio [‘Architectural Styles and the Design of Network-based Software Architectures’](https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm). En el campo de las APIs, REST (Representational State Transfer- Transferencia de Estado Representacional) es, hoy en día, el alfa y omega del desarrollo de servicios de aplicaciones.

Características de REST

* **Protocolo cliente/servidor sin estado:** cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché.
* Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
* Los objetos en REST siempre se manipulan a partir de la URI. Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado.
* Interfaz uniforme: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.
* **Sistema de capas:** arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.
* **Uso de “hipermedios”:** “hipermedia” es un término acuñado por [Ted Nelson](http://hyperland.com/) en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Ejemplo de respuesta de una petición a una API REST según el principio HATEOAS:

Ventajas que ofrece REST para el desarrollo

* **Separación entre el cliente y el servidor:** el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas (MUY importante en nuestro proyecto), aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.
* **Visibilidad, fiabilidad y escalabilidad:** La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.
* **La API REST siempre es independiente del tipo de plataformas o lenguajes:** la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.

**1.2 Estudio de los paradigmas Frontend y Backend**

Según los últimos estudios cada uno de nosotros pasa más de 4 horas al día frente a un ordenador, durante las cuales interactuamos con un numero “X” de páginas web. No podemos negar que la web forma parte de nuestro día a día, aunque va más allá de los diseños agradables y animaciones, pues por detrás encontramos una serie de tecnologías que hacen que la magia sea posible. Estas tecnologías, se encargan de que todo funcione correctamente para que el usuario no tenga ningún problema al mandar una petición, consulta u obtener datos. Estamos hablando de los lados frontend y backend que se unen para cumplir un mismo objetivo: buscar la satisfacción del usuario.

El **Frontend** se enfoca en el usuario, en todo lo que podemos interactuar y lo que vemos mientras navegamos. Así, nuestra web busca causar una buena impresión y agradar al usuario, para lo cual utiliza HTML, CSS y JAVASCRIPT. Buena experiencia de usuario, inmersión y usabilidad, son algunos de los objetivos que busca un buen frontend y hoy en día existen una gran variedad de frameworks, preprocesadores y librerías que nos ayudarán en esta tarea. Para un frontend la creatividad es el recurso más valioso, ya que tendrá que tomar fuentes, colores, imágenes y todos lo recursos de los cuales disponga para crear sitios agradables que se vean bien en todos los dispositivos y resoluciones.

Por otro lado, tenemos el **Backend** enfocado en hacer que todo lo que está detrás de un sitio web funcione correctamente. Toma los datos, los procesa y los envía al usuario, además de encargarse de las consultas o peticiones a la Base de Datos, la conexión con el servidor, entre otras tareas que debe realizar en su día a día. Cuenta con una serie de lenguajes y herramientas que le ayudan a cumplir con su trabajo como PHP, Ruby, Python, JavaScript, SQL, MongoDb, MySQL, etc, estos son usados para crear sitios dinámicos. Como en muchos sitios la información se encuentra en constante cambio o actualización, una buena capacidad de respuesta y una velocidad óptima del sitio son responsabilidades que un backend debe de afrontar.

Ya que entendimos las funciones de cada uno, podemos afirmar que ninguno de los dos lados es mejor o menos importante que el otro, pues se complementan con el fin de cumplir el mismo objetivo…

**1.3 Estudio del framework Angular**

Es la herramienta que utilizaremos principalmente en nuestro proyecto, así que nos centraremos de lleno en ella en este apartado.

**Angular** es un **framework** de desarrollo para JavaScript creado por Google. La finalidad de Angular es facilitarnos el desarrollo de aplicaciones web SPA y además darnos herramientas para trabajar con los elementos de una web de una manera más sencilla y optima.

Una aplicación web **SPA** creada con Angular es una web de una sola página, en la cual la navegación entre secciones y páginas de la aplicación, así como la carga de datos, se realiza de manera dinámica, casi instantánea, asincronamente haciendo llamadas al servidor (backend con un API REST) y sobre todo sin refrescar la página en ningún momento.

Antes de la aparición de este framework este tipo de aplicaciones web de una sola página se hacían usando solo JavaScript y jQuery, lo que era muy complejo, tedioso y difícil de mantener. Aparte de que necesitaba gran cantidad de código para hacer cualquier cosa. Este es el problema que viene a resolver Angular, convirtiendo el desarrollo de una web SPA en un proceso muy sencillo, muy limpio, más orientado a objetos y con una arquitectura basada en componentes y buenas prácticas.

Otra ventaja que tiene este framework es que está respaldado por Google y tiene una comunidad brutalmente grande detrás.

Ventajas de usar angular

1. **Facilidad de uso**

Dado que Angular es un framework, ofrece muchas funcionalidades de serie que una simple biblioteca. Con otro software similar, lo más común es tener que usar varias bibliotecas de terceros a la hora de desarrollar una aplicación. Lo más probable es que se necesite algunas adicionales para hacer el routing (enrutado de la aplicación web), para la gestión de dependencias, para realizar llamadas a APIs REST, para hacer testing, etc…

Con Angular ya sabes desde el primer momento cómo organizar el código, como se realizan las diferentes tareas que necesitas, la arquitectura de la aplicación…

1. **TypeScript**

Aunque se puede programar con ECMAScript (estándar de JavaScript) “puro”, el equipo de Angular decidió que haría todo el desarrollo con el lenguaje TypeScript, y casi toda la documentación utilizan este lenguaje.

Una de las ventajas de usar TypeScript en vez de JavaScript es la consistencia en la documentación. Por ejemplo, ES6 (o sea, ECMAScript 2015) ofrece varias formas diferentes de declarar un objecto, lo cual puede llegar a confundir. Con TypeScript esto no pasa, y toda la sintaxis y la manera de hacer las cosas en el código es la misma, lo que añade coherencia a la información y a la forma de leer el código.

Aunque Angular no te obliga a usar TypeScript, el equipo del core de Angular sí que lo ha adoptado y en la documentación sugiere usar TypeScript por defecto. Esta consistencia debería ayudar a evitar la confusión y la sobrecarga en la toma de decisiones derivadas de empezar con Angular.

Todo esto conlleva a largo plazo, en un mejor y más fácil mantenimiento de las aplicaciones.

1. **Componentes web**

Un componente en Angular es una porción de código que es posible reutilizar en otros proyectos de Angular sin apenas esfuerzo, lo que permite un desarrollo de aplicaciones mucho más ágil, pasando de un costoso MVC a una especie de puzle con los componentes.

El diseño de Angular adopta el estándar de los [componentes web](https://w3c.github.io/webcomponents/spec/custom/). Se trata de un conjunto de APIs que te permiten crear nuevas etiquetas HTML personalizadas, reutilizables y auto-contenidas, que luego puedes utilizar en otras páginas y aplicaciones web. Estos componentes personalizados en navegadores modernos y con cualquier biblioteca o framework de JavaScript que funcione con HTML.

El [soporte actual](https://www.webcomponents.org/) de este estándar por parte de los navegadores es muy reducido (básicamente Chrome), aunque existen [polyfills](https://www.imaginanet.com/blog/polyfills-para-css3-y-html5.html) para suplir esta carencia.

**2. Definición del problema**

En este apartado se va a describir las características del problema a resolver con el desarrollo e implementación del proyecto en cuestión. Para poder diferenciar bien los problemas y analizarlos de forma más exhaustiva vamos a distinguir entre Problema Real, visión que tiene el usuario del problema (que no debe saber cómo está hecho nuestra aplicación) y Problema Técnico, visión específica que se dará para los desarrolladores interesados en el proyecto.

**2.1 Definición del problema real**

El problema estudiado en el proyecto es la prevención del desconocimiento por parte de la población en general y de forma más específica a los aficionados al mundo de la Fórmula 1.

El problema que se le plantea al usuario es no saber dónde consultar los resultados deportivos de sus atletas favoritos, debido principalmente a dos motivos básicos: falta de credibilidad de algunas aplicaciones y ausencia de sitios o aplicaciones dedicadas exclusivamente a los resultados que los usuarios desean.

**2.2 Definición del problema técnico**

En este apartado se pretende dar una visión del problema desde el punto de vista técnico. Indicando todas las características más relevantes en cuanto a desarrollo y componentes utilizados.

**2.2.1 Funcionamiento**

El software desarrollado para este proyecto debe conseguir un objetivo fundamental, proporcionar a los usuarios una aplicación web para poder consultar los resultados históricos y actualizados del campeonato de Fórmula 1, reuniendo datos de pilotos, constructores, y circuitos.

Para conseguir tal resultado se usan datos de una API de dominio público la cual provee datos “planos” de todas las áreas citadas anteriormente. Nuestra aplicación se encargará de mapear esos datos y mostrárselos al usuario de una manera amigable y clara.

El software está dirigido al público en general, esto quiere decir que no se necesitan ningún tipo de conocimientos técnicos, aunque el usuario objetivo será el aficionado más acérrimo al campeonato de Fórmula 1. La aplicación va a estar desplegada en un entorno web, lo que la hace accesible desde todos los dispositivos que puedan acceder a un navegador web.

**2.2.2 Entorno**

Nuestro aplicativo se encuentra dividido en tres entornos que abarcan los aspectos principales a tener en cuenta: entorno hardware, entorno software y el entorno de usuario.

* **Entorno hardware:** Al ser un proyecto principalmente centrado en el desarrollo del software el entorno hardware es menos importante. Aun así se debe especificar que al ser una aplicación que se mueve directamente en el navegador, el único hardware necesario para que funcione sería cualquier dispositivo que cuente con un navegador web. De todas formas, para que esta aplicación sea accesible desde varios ordenadores al mismo tiempo y por varios usuarios a la vez será desplegada en un servidor web.
* **Entorno software:** Nuestro software ha sido desarrollado con el lenguaje de programación TypeScript (Derivado del tan famoso JavaScript), usando como framework de desarrollo el tan aclamado Angular. Para ello hemos tenido que instalar Node.js (entorno de ejecución de JavaScript, por hacer un símil, parecido al SDK de Java) que trae un potente gestor de paquetes integrado (npm, Node Package Manager).
* **Entorno de usuario:** El usuario final al que va destinado este software no debe tener ningún conocimiento previo, ya que nuestra herramienta es bastante intuitiva y de fácil uso.

**2.2.3 Vida esperada**

El desarrollo del software, desde sus inicios es conocido por tener un ámbito de aplicación perecedero, el software que se ha desarrollado mitiga éstos efectos al mínimo ya que si la API de la que consume no tiene cambios a gran escala, nuestra aplicación seguirá funcionando con normalidad (hay que tener en cuenta que la API de la que consume es experimental, por lo que en las primeras etapas de la aplicación puede ser que sea más propensa a errores). La vida de nuestro software, como la de cualquier otra aplicación, va a venir determinada por los cambios y el mantenimiento realizado, pero sobre todo, como hemos indicado antes por los cambios en la API.

**2.2.4 Ciclo de mantenimiento**

El sistema desarrollado no necesita de mantenimiento alguno, salvo el que implicaría su actualización respecto a nuevos conceptos de aplicación. Además, nuestra aplicación consta de una interfaz flexible y reutilizable para próximas versiones de este software. Lo único es que se deberá comprobar el correcto funcionamiento de la aplicación cada cierto tiempo, y sobre todo, al final y al inicio de la temporada del campeonato, ya que ahí la aplicación puede tener ciertos problemas para adaptarse a la nueva temporada.

El software aquí desarrollado puede ser utilizado como punto de partida para futuros trabajos y mejoras del mismo que impliquen nuevas funcionalidades. En trabajos futuros, capítulo 5 de este documento Conclusiones, se determinará diferentes usos que puede llegar a tener este software y las modificaciones que puede sufrir para llevarlos a cabo.

**2.2.5 Estandarización**

El sistema se ha desarrollado en el lenguaje de programación TypeScript, que al ser un lenguaje derivado de JavaScript (conocido por su flexibilidad y uso extendido) y diseñado bajo la tutela de Microsoft cumple todos los estándares que cualquier lenguaje de programación debe tener hoy día; programación orientada a objetos, funcional, asíncrona, etc…

La interfaz ha sido diseñada siguiendo los estándares que hay en el mercado, de manera que el usuario se familiarice con ella lo más rápido posible.

El código fuente está totalmente documentado, cualquier usuario con conocimientos de programación en estos lenguajes puede analizarlo sin dificultad. Por supuesto esta documentación del código ayuda a realizar las futuras mejoras y actualizaciones.

**2.2.6 Calidad y fiabilidad**

El término calidad se ha referido a la concordancia existente entre los requisitos funcionales y de rendimiento debidamente establecidos, la adaptación a los estándares de desarrollo seleccionados explícitamente documentados y a las características implícitas esperadas de todo software desarrollado profesionalmente. Nuestro sistema se ajusta a la definición de calidad, por lo que podemos afirmar que la posee. Se han conseguido los requisitos establecidos, así como un gran nivel de estandarización. El contenido se ha desarrollado a partir de multitud de fuentes. Tras sucesivas pruebas de control, podemos afirmar que el código fuente del sistema es correcto, lo que nos garantiza en un alto porcentaje la ausencia de fallos de ejecución o la correcta actuación del programa si estos ocurren.

**3. Antecedentes**

A continuación, se muestra el resultado de la investigación que se ha realizado para determinar que herramientas hay actualmente en el mercado que realicen una función parecida a la nuestra, explicaremos como funcionan (si hay documentación al respecto) y en qué nuestra aplicación las mejora.

Viendo que el campo donde nuestro aplicativo se centra es un campo bastante cubierto por los medios, solo basta buscar "Resultados Fórmula 1" para que aparezcan delante de ti cientos de resultados, esto es en parte porque casi todos los medios de comunicación cubren los resultados del campeonato o de las carreras, pero no llegan al mismo nivel de análisis que nuestro software.

Para evitar tener que analizar todos esos sitios nos vamos a centrar en el sitio oficial de la Fórmula 1 y en las distintas aplicaciones que hay relacionadas directamente con los datos, y que están desarrolladas basándose en la misma API de la que consumimos nosotros.

**3.1.1 Página oficial de la Fórmula 1 (**[**www.formula1.com**](http://www.formula1.com)**)**

Como era de esperar es la alternativa más completa a la hora de encontrar resultados sobre el campeonato. Analizando la sección que tiene por objetivo hacer lo mismo que nuestra aplicación, podemos hacer el siguiente resumen:

* Apartado "Races" (Carreras): Nos muestra una lista con las carreras del campeonato del año en curso, junto con el ganador de cada una de ellas (si lo hubiera), además de venir indicadas con un pequeño sombreado las que ya se han disputado. Cuando se pulsa encima de cualquiera de las carreras nos lleva a una página con el detalle de esta con los resultados y la información del circuito.
* Apartado "Standings" (Resultados): Vemos una serie de listas con información acerca de los resultados de pilotos, equipos, y (de nuevo) carreras. La información viene dispuesta de una forma que puede llegar a confundir al usuario ya que se muestra en una combinación de listas que el usuario debe combinar para ver los resultados que desee.

Dentro de este apartado posee una sección dedicada exclusivamente a un "archivo", donde están los resultados de todas las temporadas anteriores a la que se encuentre en curso. También incluye un apartado de "premios" donde se pueden consultar los "Driver of the Day" y el premio "DHL Fastest Pit Stop Award".

* Apartado "Drivers" (Pilotos): Accedemos a una lista de los pilotos de esta temporada, viendo su nombre, número y fotografía. Además nos la lista con las posiciones del campeonato de pilotos del año en curso, por lo que de un vistazo podemos ver una gran cantidad de información.
* Apartado "Teams" (Equipos): Nos lleva a una vista con un diseño bastante elaborado en la cual se listan los equipos oficiales del campeonato para esa temporada, en cada entrada de esta lista tenemos el nombre del equipo, los pilotos actuales, el número de podiums y el número de títulos que tiene el equipo en cuestión. Al pulsar encima de cada equipo nos llevará a una página dedicada exclusivamente al mismo, donde vemos estadísticas más detalladas del equipo.

Vistos estos apartados se ha realizado una valoración general del sitio, por lo que bajo nuestro punto de vista es quizás la alternativa más vistosa, ya que el diseño es espectacular, además que tenga el respaldo de ser un sitio oficial otorga de mucha confianza a los usuarios.

Por otro lado, encontramos un poco difícil encontrar algunos detalles en la información (quizás por el diseño tan recargado de la página), y puede que para los usuarios que no estén acostumbrados al diseño de la página les cueste un poco moverse por ella.

Otro aspecto negativo que hemos encontrado es el de tener tanto artículo de noticias y opinión, ya que los usuarios que quieran saber solo los resultados quizás les llegue a molestar un poco.

En comparación a este sitio, nuestra aplicación está más centrada en los resultados propiamente dichos, sin ofrecer más distracciones al usuario. Aun así pensamos que nuestra aplicación puede estar más enfocada, o que sea de más utilidad, a usuarios familiarizados con la estadística.

**3.1.2 Formula Graph (**[**www.formulagraph.com**](http://www.formulagraph.com)**)**

Nada más visitar este sitio lo primero que nos llama la atención es su simpleza, ya que todos los datos mostrados en la misma son a través de una gráfica por lo que tiene sus ventajas y sus inconvenientes.

Se puede alternar entre ver los resultados de una temporada en concreto o ver los resultados históricos de un determinado constructor o piloto. Aunque la disposición de la información es buena, la valoración se ve un poco lastrada por la falta de datos.

Por ejemplo, se echa en falta datos más específicos de los pilotos, tales como lugar de nacimiento, numero de carreras, etc. Lo mismo se puede llegar a decir de los constructores de los cuales solo se puede saber los resultados a lo largo de su historia, en el campeonato de constructores, información a todas luces, escasa.

**3.1.3 F1 Fansite (**[**www.f1-fansite.com**](http://www.f1-fansite.com)**)**

Antes de empezar el análisis de este sitio, debemos referirnos a su descripción:

"On F1-Fansite.com you can find all F1 results of each and every held Formula 1 grand prix. You can also find all championships since the beginning of Formula 1 in 1950 until the current season This page displays every F1 season. Behind each year you will find the F1 results of each F1 GP and the drivers and constructors championship tables for each F1 season.

En F1-Fansite.com puede encontrar todos los resultados de F1 de todos y cada uno de los grandes premios de Fórmula 1 celebrados. También puede encontrar todos los campeonatos desde el comienzo de la Fórmula 1 en 1950 hasta la temporada actual. Esta página muestra cada temporada de F1. Detrás de cada año, encontrarás los resultados de F1 de cada GP de F1 y las mesas de campeonato de pilotos y constructores para cada temporada de F1."

Leído esto tenemos que decir que es fiel a lo que la página ofrece, ya que en materia de datos es bastante competente, teniendo una cantidad abundante, quizás su talón de Aquiles sea el diseño, que recuerda a los de las primeras páginas de Internet y su adaptación a distintos dispositivos es bastante mediocre.

**3.1.4 “4Mula1” (**[**http://www.4mula1.ro/**](http://www.4mula1.ro/)**)**

En esta ocasión esta aplicación tiene mejor diseño que la analizada anteriormente, aún así sigue sin convencer debido a los altos estándares de diseño que se han impuesto hoy día. En lo relativo a los datos vemos como tiene unos datos bastante precisos, aunque se echa de menos algo más de contenido multimedia, tales como vídeos, imágenes de equipos y pilotos, etc.

No se ha encontrado nada relacionado con los circuitos del campeonato, ni del actual ni de las temporadas anteriores, no sabemos si es que realmente no dispone de ellos o que simplemente el diseño no permite acceder a los mismos debido a su poca usabilidad.

**4. Posibles trabajos y solución elegida**

Visto lo que hay actualmente en el mercado se llega a la conclusión de que ningún sitio o aplicación que esté en producción es perfecta, por lo cual en nuestro desarrollo deberá tener la unión de una fuente de datos confiable, una programación fuerte y estructurada con unos estándares de calidad. Para ello se ha llegado a la siguiente conclusión:

Nada más empezar a pensar en la realización del proyecto lo primero que se nos ocurrió es desarrollar un sistema completo, es decir configurar un servidor propio con alguno de los lenguajes de Backend más conocidos (Java, Python, PHP) y ocuparnos del modelado de entidades, relación de estas y lo más importante de todo, conseguir una consistencia en los datos. Esto además tenía otro inconveniente incluso peor, el inmenso volumen de datos. Y es que para tener alojados en un servidor tal cantidad de datos (recordemos que nuestra herramienta debe ofrecer registros de todos los campeonatos de Fórmula 1 desde el principio, año 1950) en el desarrollo debe primar la alta disponibilidad y la confiabilidad cosa que nosotros, debido a la falta de tiempo para desarrollar el proyecto (apenas mes y medio quitando el tiempo que nos llevaría aprender todas las tecnologías necesarias para acometer el proyecto) y al poco conocimiento que tenemos en la tecnología desestimamos desde el primer momento.

En cuanto al Frontend, es algo que había que hacer si o sí, ya que era lo que queríamos al inicio de todo, una interfaz amigable para el usuario, sencilla y con una orientación multiplataforma. Para ello se pensó desde un principio en Angular, debido a su alta popularidad y su conocida robustez en cuanto a la estructura de código. Dicho esto, en los siguientes apartados se detallará en profundidad como se ha llevado a cabo el desarrollo de este.

**5. Restricciones**

En este apartado se expondrán todas aquellas restricciones existentes en el ámbito del diseño que condicionan la elección de una u otra alternativa.

**5.1 Factores dato**

En este punto se enumeran todas aquellas restricciones que son impuestas por el propio proyecto, son las siguientes:

* **Tiempo:** El tiempo de desarrollo del proyecto está estrechamente relacionado a la fecha de entrega de este. Es decir, aproximadamente la primera quincena de junio, dentro del curso académico 2018/2019. Dicho tiempo se ve aún mas limitado teniendo en cuenta las nuevas tecnologías usadas en el proyecto.
* **Recursos humanos:** Para este proyecto se cuenta con un solo desarrollador, que en este caso es el alumno.
* **Recursos hardware y software:** En cuanto al hardware un ordenador personal y en cuanto a software ha sido bienvenido todo el contenido relacionado que se ha encontrado por Internet.
* **Limitaciones de tiempos de ejecución:** Estas limitaciones son casi inexistentes, ya que la aplicación desarrollada es bastante liviana.

**5.2 Factores estratégicos**

A diferencia de las anteriores, estas restricciones son impuestas por el propio desarrollador del proyecto.

* Se utilizará una metodología basada en etapas para realizar este proyecto, basada en el desarrollo de las metodologías ágiles. Entre las ventajas destaca la simplicidad, la temporalización y la potencia de este tipo de metodologías.
* El lenguaje de programación seleccionada para la programación de la interfaz será [TypeScript](https://www.typescriptlang.org/). Se ha seleccionado por ser un lenguaje orientado a objetos de alto nivel, multiplataforma, libre, seguro y recomendado por la comunidad de programadores. Además proporciona todo lo que queremos conseguir con el proyecto, que es simpleza y estabilidad.
* El framework de desarrollo elegido para hacer el desarrollo ha sido Angular en su versión 7.2.15 y será programado a través del entorno de desarrollo de Visual Studio Code todo ello instalado en una máquina con Windows 10.
* El software desarrollado será desplegado en un [Dyno](https://www.heroku.com/dynos) de [Heroku](https://www.heroku.com/) (ligeros contenedor de aplicaciones), lo que nos aporta robustez y la confianza que da una plataforma como Heroku.
* La herramienta para realizar la documentación será el editor Microsoft Office Word, también instalado en una máquina con Windows 10.
* El paradigma de desarrollo del proyecto escogido ha sido el de desarrollo de prototipos, lo que permite ir desarrollando funcionalidades básicas de la aplicación, a esto se le llama MVP (Minimum Viable Product) para después en fases intermedias del proyecto poner en común todas estas características e integrarlas para construir la aplicación.

**6. Recursos**

En este apartado se presentan los recursos con los que se cuenta para la realización del proyecto. Se han dividido en recursos humanos, recursos software y recursos hardware.

* Recursos humanos: El equipo de personas con los que cuenta el proyecto está formado por: D: Miguel

D. Juan Antonio Sánchez Luque, alumno de segundo curso del Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma