Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una plataforma para el análisis de Twitter

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jose Luis Pitarch Miravete
Tutor: Ferran Pla Santamaría
Cotutor: José Ángel González Barba

Curso 2020 - 2021

Resumen

Vivimos en una época donde el flujo de datos ha aumentado exponencialmente, sobre todo con la llegada de las redes sociales las cuales han permitido que prácticamente toda la población mundial pueda compartir información u opiniones.

Esta inmensa cantidad de datos dificulta contrastar la información más relevante haciendo paradójicamente, que estos pierdan visibilidad. Así pues, surge la necesidad de aplicar un filtro para que cada persona pueda ver la información que realmente le interese sin necesidad de indagar más de lo necesario.

En el caso que nos ocupa, *Twitter*, no es una excepción. Esta red social dispone de millones de usuarios activos y como es obvio, se publica muchísima información por segundo. En este proyecto, se aborda la posibilidad de que el usuario visualice de una manera gráfica y rápida qué datos se están publicando sobre algún tema en específico.

En concreto, el objetivo principal de este trabajo es la de contrastar la información recogida de *Twitter* y mostrarla de tal forma que el usuario emplee el tiempo justo en cubrir sus necesidades sobre algún tema de su interés y que esta experiencia le resulte más gratificante posible.

Resum

Vivim en una època on el flux de dades ha augmentat exponencialment, sobretot amb l'arribada de les xarxes socials les quals han permès que pràcticament tota la població mundial puga compartir informació o opinions.

Aquesta immensa quantitat de dades dificulta contrastar la informació més rellevant fent paradoxalment, que aquestes perden visibilitat. Així doncs, sorgeix la necessitat d'aplicar un filtre perquè cada persona puga veure la informació que realment li interessa sense necessitat de indagar més del necessari.

En el cas que ens ocupa, Twitter, no és una excepció. Aquesta xarxa social disposa de milions d'usuaris actius i com és obvi, es publica moltíssima informació per segon. En aquest projecte, s'aborda la possibilitat que l'usuari visualitze d'una manera gràfica i ràpida quines dades s'estan publicant sobre algun tema específic.

En concret, l'objectiu principal d'aquest treball és la de contrastar la informació recollida de Twitter i mostrar-la de tal manera que l'usuari faça servir el temps just en cobrir les seves necessitats sobre algun tema del seu interès i que aquesta experiencia li resulte el més gratificant possible.

Abstract

We live in a time where the flow of data has increased exponentially, especially with the arrival of social networks, which have allowed practically the entire world population to share information or opinions.

This immense amount of data makes it difficult to contrast the most relevant information, paradoxically causing them to lose visibility. Therefore, the need to apply a filter appears in order to let people see information that interests them without having to search more than necessary.

In the present case, Twitter is not an exception. This social network has millions of active users and a lot of information is published per second. In this project, the possibility is addressed for the user to visualize in a graphic as a fast way what data is being published on a specific topic, concretely, the main objective of this task is to contrast the information collected from Twitter and display it in a way that the user employs only the necessary time to meet his needs on a topic of interest and make this experience as rewarding as possible.

Tabla de contenidos

Contenido

Índice de figuras

1. Introducción

Actualmente, se estima que alrededor de un cuarenta por ciento de la población hace uso de las redes sociales. Así pues, la cantidad de información que circula por este nicho de internet es sumamente elevada.

La red social que vamos a estudiar, *Twitter*, es una red más destinada a que los usuarios compartan sus pensamientos y hechos de la actualidad, a diferencia de, por ejemplo, *Instagram*, que está más destinada a compartir contenido más cotidiano.

Twitter tiene más de 300 millones de usuarios activos, lo cual nos indica que esta red social abarca muchos datos de todo tipo. Aproximadamente se estima que se escriben alrededor de 350 mil *tweets* al día.

De este modo, realizar una aplicación web que recoja información de *Twitter* sobre cierto tema de actualidad y la simplifique puede resultar muy útil para los usuarios. Por ejemplo, recoger datos sobre algún acontecimiento importante que ha ocurrido y mostrar aquello con mayor relevancia y de una manera distinta a la propia red social, como puede ser el uso de gráficos.

Este proyecto abarca tanto el contraste de la información a mostrar como el cuidado de una interfaz gráfica elegante y amigable para el consumidor final con el fin de que su experiencia sea lo más gratificante posible a la par que eficaz.

1.1. Motivación

Desde un punto de vista personal, el tratamiento de grandes cantidades de información me apasiona ya que se debe seguir una serie de estrategias para que el coste temporal sea el menor posible a la hora de recuperar información. Esto es conocido como *Big Data*. Además, el campo del desarrollo web también resulta muy interesante puesto que es ampliamente demandado en el ámbito laboral. Por último, la presencia de un diseño en la aplicación también es de gran interés ya que también es una parte importante en el desarrollo web.

Desde un punto de vista profesional, como he mencionado anteriormente, el desarrollo web está en pleno crecimiento puesto que cada vez hay más tecnología a nuestra disposición la cual permite que sea una tarea más fácil haciendo que las empresas opten por desarrollar aplicaciones web frente a las de escritorio.

1.2. Objetivos

El objetivo principal de este proyecto consiste en recuperar la información depositada en una base de datos de la manera más eficaz posible y mostrarla a través de un diseño compacto y sencillo. De una manera esquemática los puntos principales de este trabajo serían los siguientes:

- Depositar la información en la base de datos
  - Crea las colecciones a partir del *JSON*
- Crear un controlador
  - Escalable
  - Procesar la información
- Diseñar una interfaz
  - Muestra la información recuperada de la base de datos
  - Compacta y amigable
  - Implementar la lógica para interactuar con el usuario

1.3. Impacto esperado

El impacto que se espera sobre el usuario final es la de mostrarle la información más relevante sobre un tema de su interés de la manera más sencilla e intuitiva posible para mejorar su experiencia. Se busca ahorrarle el mayor tiempo posible en su busca sobre algún tema en concreto.

1.4. Estructura

Este trabajo está constituido por siete capítulos. A continuación, se detallan los temas tratados en cada capítulo:

En el **capítulo 1. Introducción**, se realiza una pequeña presentación del tema que aborda este proyecto, así como como la motivación personal y profesional donde se explica porque se ha escogido este trabajo.

En el **capítulo 2. Contexto tecnológico**, se expone qué herramientas existen en el mercado actualmente relacionados en este ámbito y qué sitio puede ocupar este proyecto en él mencionando en qué se diferencia del resto. Finalmente, se explica de una manera sencilla qué es internet y cuáles son sus aspectos más básicos.

En el **capítulo 3. Tecnologías utilizadas**, se da paso a citar y explicar todas los lenguajes y tecnologías que se han utilizado para poder llevar a cabo el proyecto que nos ocupa, así como el porqué de la elección de estas frente al resto.

En el **capítulo 4. Desarrollo de la aplicación**, se explica detalladamente cómo se ha implementado los tres componentes de la aplicación: el modelo, el controlador y la vista.

En el **capítulo 5. Experimentación**, se adjunta una tabla con los tiempos obtenidos que reflejan la eficiencia de la aplicación y aquellos puntos a mejorar en un futuro.

En el **capítulo 6. Relación con los estudios cursados,** se realiza desde una perspectiva introspectiva, un análisis de los conceptos adquiridos a lo largo del grado que se han empleado en la realización del trabajo.

En el **capítulo 7. Conclusiones,** se hace un breve repaso de lo que ha supuesto este trabajo, cuál es su fin y posibles mejoras que se pueden añadir.

En el **capítulo 8. Trabajos futuros**, se exponen posibles mejores a implementar en este trabajo para seguir mejorando su funcionalidad.

2. Contexto tecnológico

Una vez presentado el problema a tratar y la posible solución a llevar a cabo ante éste, vamos a contemplar en qué contexto se encuentra el proyecto.

Como era de esperar, existen varias herramientas ya disponibles en el mercado que también tratan de condensar la información, incluso la misma red social ofrece este servicio. No obstante, estas aplicaciones están más destinadas a la de analizar el perfil del usuario. Es decir, el usuario introduce su perfil y la aplicación hace un rastreo y analiza su cuenta de *Twitter* devolviendo información del tipo: visitas al perfil, alcance de la cuenta, repercusión de los *tweets* y una larga lista de datos. También existe otro tipo de aplicaciones destinadas para las empresas que desean tener una mayor presencia en las redes sociales con el fin de que sus productos o servicios lleguen a más clientes.

A continuación, se nombran algunas de estas herramientas para reforzar lo expuesto en el párrafo anterior:

1. *Twitter Analytics*: permite recoger información de los últimos treinta días indicando las menciones a tu cuenta, impresiones de los *tweets*, *tweets* destacados, nuevos seguidores y muestra los *tweets* más relevantes, entre otros. Además, ofrece un servicio para tener un mayor alcance a cambio de una cuota.

2. *Audiense:* está más destinada para el sector empresarial donde puedes conocer detalles de tu competencia, explorar nuevas tendencias y en definitiva analizar una campaña de marketing digital.

3. *Tweepi*: se centra más en la conexión entre las personas, por ejemplo, personas que un usuario sigue y estas a él no, listado de gente que sigue alguien de interés o un tema determinado para el usuario, gestionar los seguidores, entre otros.

Estas son sólo tres de las muchas aplicaciones que se encuentran en este contexto. Cabe mencionar que la gran mayoría de estas webs presentan dos versiones: una gratuita y otra de pago. La primera ofrece unos servicios más básicos mientras que la versión de pago expande el abanico a más prestaciones que a su vez son más complejas y, por tanto, los datos que aportan son más significativos y proyectables para el usuario en concreto.

2.1 Propuesta

La principal diferencia entre este trabajo y los ya existentes en el mercado digital viene dada por el servicio ofrecido, es decir, qué información va a mostrar la aplicación. Este proyecto, se va a centrar en mostrar contenido de actualidad, temas de crispación social y, en definitiva, temas que, en un momento determinado, pueden ser de gran interés para la población en general.

En cambio, las aplicaciones que ya existen en el mercado profundizan más en mostrar información entorno al propio usuario. En otras palabras, realizan un seguimiento sobre la cuenta con datos de interés para el internauta, como puede ser seguidores ganados en los últimos treinta días, alcance de los *tweets*, etc.

En un inicio, este servicio será completamente gratuito lo cual añade otra diferencia importante ante las ya existentes puesto que la gran mayoría ofrecen un servicio de calidad en su versión de pago. No obstante, tampoco se descarta la opción de seguir añadiendo funcionalidades más complejas a la aplicación que ofrezcan unos servicios que el usuario esté dispuesto a pagar.

2.2 Cliente – Servidor

Internet o la web pueden definirse como un conjunto de dispositivos o computadoras que están conectadas entre sí y son capaces de transmitirse información. Dos computadores pueden conectarse entre sí a través de distintas formas. En este caso, se van a exponer las dos más relevantes y utilizadas en la actualidad.

2.2.1. Comunicación vía cable

Fue la primera y única manera de comunicar dos computadoras antiguamente. Básicamente un cable une ambas máquinas y a través de esté la información es capaz de viajar de una computadora a otra. Mediante unos protocolos de comunicación y un *hardware* preparado para interpretar los datos transmitidos, las máquinas son capaces de comunicarse.

2.2.2 Comunicación vía *Wi-Fi*

Apareció más tarde y revolucionó el campo de la web puesto que trajo consigo la capacidad de comunicar dispositivos sin necesidad de conectar un cable a ambos. Esta vía permite que los datos viajen a través del aire en formato de ondas que las máquinas son capaces de captar e interpretar.

2.2.3. Funcionamiento y características de un cliente y servidor

Actualmente, cualquier dispositivo puede conectarse a otro independientemente de su ubicación ya que la red engloba a todo el planeta. Una vez puesto en contexto qué es internet de una manera sencilla se puede hablar sobre qué es un sitio web. Cuando se habla sobre cualquier sitio web se distingue principalmente dos pilares fundamentales que forman el comportamiento más primario de internet. Es cierto que depende de las tecnologías que se empleen surgen diferencias, pero, en definitiva, esta es la base de toda página web.

Por un lado, existe lo que se conoce como cliente, que no deja de ser una computadora capaz enviar información o recibir datos a través de la red. Este cliente es una persona qué a través de un navegador, como puede ser *Google Chrome* o *Mozilla Firefox*, tiene la capacidad de acceder a un sitio web.

Por otro lado, existe lo que se conoce como servidor, que también es un dispositivo conectado a internet con la capacidad de enviar y recibir información. Éste se encarga de recibir la petición que realiza el cliente, procesarla y enviar la respuesta correspondiente.

Aparentemente, tanto el cliente como el servidor tienen la misma definición y en cierto modo, así es. Ambos pilares no dejan de ser máquinas con las mismas funcionalidades y posibilidades. La diferencia se establece a la hora de definir quién es el que envía y quién el que recibe. A continuación, se muestra un esquema gráfico para entender del todo la estructura de internet.

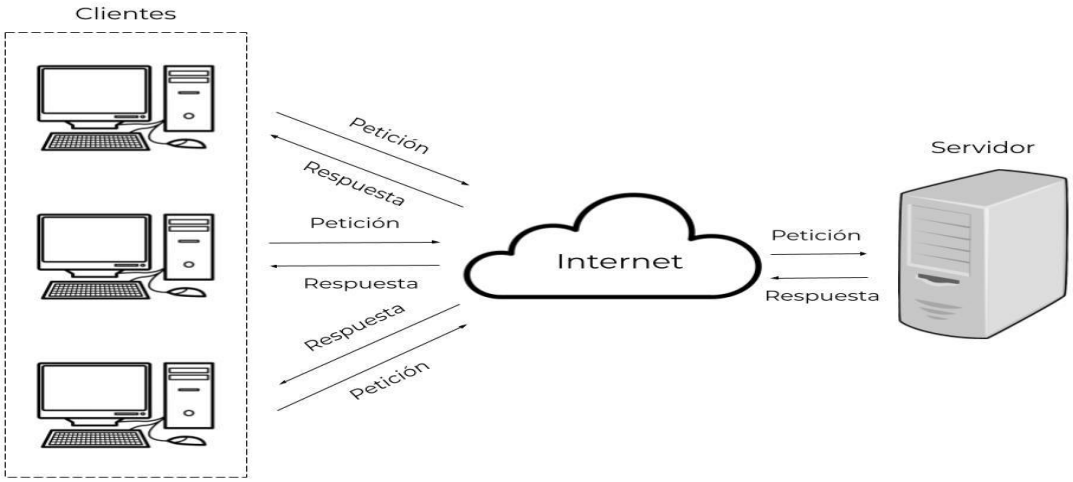

Figura 1. Ejemplo del funcionamiento de una pequeña red formada por tres clientes y un servidor.

Un servidor es una máquina computacional destinada exclusivamente a recibir peticiones por parte del cliente. Así pues, su *hardware* es bastante más potente que el de un cliente para poder aguantar una carga elevada y evitar que deje de funcionar si recibe un número de peticiones muy elevado. Naturalmente, un servidor puede atender a varios clientes o, dicho de otra forma, varios clientes pueden conectarse a un mismo servidor. Para este proyecto, el cliente cuando quiere acceder a la aplicación web el servidor le envía el código fuente de la página para que su navegador sea capaz de mostrarla en la pantalla del dispositivo.

Hoy en día, el 92 % de los internautas, según el periódico *La Vanguardia* consultado el 15 de junio de 2021, acceden a Internet a través de su dispositivo móvil. En los últimos años, los teléfonos móviles han tenido un avance excelente ofreciendo unas prestaciones más que suficientes. No obstante, por su tamaño tan reducido no pueden competir contra los ordenadores de estas generaciones. Además, las baterías de los móviles todavía siguen teniendo una duración media. Así pues, ante este escenario y aprovechando que los servidores son bastante potentes, la estrategia que se emplea o, al menos, la que se debería emplear, es la de realizar toda la carga computacional en el lado del servidor y enviarle los datos ya totalmente procesados al cliente. Con esta metodología se disminuye la cantidad de operaciones a realizar a la máquina que realiza la petición y, en el caso de que resulte ser un dispositivo móvil, conseguir que utilice la menor energía posible.

A su vez, el servidor además recibir peticiones por parte del cliente también puede realizar consultas a otros servidores. Por ejemplo, puede darse el caso que necesite información de otra aplicación web o incluso de varias. Como se ha mencionado anteriormente, no dejan de ser computadoras ambas partes con la capacidad de conectarse entre sí. Para este proyecto, el servidor no necesita realizar una petición a un servidor externo puesto que toda la información necesaria la tiene guardada en su propio almacenamiento. No obstante, por una serie de motivos como, por ejemplo, motivos económicos o geológicos, esta información podría almacenarse en otros servidores.

Por último, una base de datos es una colección organizada de información de forma estructurada donde se almacenan datos. Es aquí donde se almacenará toda la información extraída de *Twitter* y, por tanto, de donde se recuperará a la hora de mostrarla en la aplicación web. A su vez, para lograr comunicar el servidor con la base de datos, ésta tiene integrado un gestor el cual se puede definir como una colección de programas cuyo objetivo es servir de interfaz o, dicho de otro modo, intermediario entre estos dos componentes. Este gestor es invisible para el desarrollador, éste solo ha de hacer las operaciones necesarias y de una manera totalmente ajena a su persona se administran a través del gestor.

Una vez se ha entendido qué es cliente y servidor, se puede introducir dos tecnicismos de la programación web, el *front-end* y el *back-end*.

2.2.3.1. *Front-end*

Cuando se habla de este concepto se hace referencia a aquella parte del proyecto que se ejecuta en el lado del cliente. En otras palabras, este tecnicismo se refiere a aquellos lenguajes que el servidor va a enviar para que éste sea capaz de visualizar la página web en el navegador. Así pues, todos los lenguajes que el cliente necesita para poder cargar el sitio web y también la lógica necesaria para poder realizar acciones sobre ella se dice que son lenguajes *front-end*.

2.2.3.2 Back-end

Cuando se habla de este concepto se hace referencia a aquella parte del proyecto que se encarga de recibir las peticiones, procesarlas y enviarla la respuesta correspondiente. En otras palabras, este tecnicismo a aquellos lenguajes que son capaces de ejecutarse en el lado del servidor cuyo objetivo es escuchar peticiones por parte de los clientes, realizar las operaciones que requiera dicha petición y enviar al cliente una respuesta que, en la mayoría de los casos, contienen lenguajes *front-end*. Normalmente, se escoge un lenguaje para implementar la parte del servidor en su totalidad. Los lenguajes más utilizados hoy en día son: *Python*, *C#*, *PHP*, *Java* y *NodeJS*.

Una vez explicado la definición de lo que es internet que, básicamente son muchas computadoras capaces de enviar y recibir información entre sí, y su concepto más básico como es el de distinguir entre que es un cliente y un servidor y qué papel emplea en la red, se puede dar paso a explicar qué tecnologías son las que se usan actualmente para el desarrollo web y cuáles se han escogido entre un abanico de posibilidades en base a las necesidades y el objetivo del proyecto. En otras palabras, esta aplicación web se podría haber desarrollado otros lenguajes y tecnologías, pero posiblemente habría resultado una tarea más compleja y, en consecuencia, obtener un código que perjudique la escalabilidad y posibilidades de este trabajo.

3. Tecnologías utilizadas

Para llevar a cabo este proyecto se han usado varias tecnologías elegidas en base a las necesidades de la aplicación para conseguir un desarrollo fácil y escalable ya que en un futuro podría seguir añadiéndose funcionalidades.

Antiguamente, se disponía de mucha menos tecnología en comparación a estos tiempos. Simplemente estaban presente aquellos lenguajes necesarios para poder implementar tanto el *front-end* como el *back-end*. Para este último, solamente existía un lenguaje capaz de correr en el lado del servidor, que era *PHP*. En la actualidad, esto se ha ampliado significativamente puesto que otros lenguajes como, *Python* o *Java*, también son capaces de desplegarse en el *back-end*.

No obstante, en el *front-end* siguen existiendo las mismas tecnologías que hace años. En concreto son tres lenguajes los que corren en el lado del cliente: *HTML*, *CSS* y *JavaScript*. Lo primero que se puede llegar a preguntar cualquier persona es por qué el *back-end* ha evolucionado y el *front-end*, en cambio, sigue ofertando lo mismo. Esto no es del todo cierto debido a que los tres lenguajes se han ido ampliando y mejorando haciendo que el lado del cliente también sufra una mejora a la hora del desarrollo. No han aparecido más lenguajes para esta parte porque, en realidad, estos tres lenguajes cubren con todas las necesidades que surgen a la hora de desarrollar el *front-end* y, por así decirlo, no hay un hueco para la aparición de otro lenguaje que sea capaz de aportar algo nuevo a lo ya existente y que consiga que los proyectos web migren a dicha tecnología.

Además, hoy en día existe una comunidad inmensa que crea nuevas tecnologías a partir de las ya existentes con el fin de seguir simplificando la dificultad del desarrollo. Esto es lo que ha ocurrido en el front-*end*, se han creado nuevas tecnologías a partir de los tres lenguajes pilares existentes del cliente. En el lado del *back-end* también ha ocurrido algo parecido, a través de estas nuevas tecnologías se ha permitido que otros lenguajes sean capaces de ejecutarse en un servidor. Con lo cual, ambas partes han sufrido un avance más significativo de lo que parecía hace un momento.

Estas nuevas tecnologías que han permitido que el campo del desarrollo web haya sufrido una mejora se conocen como *frameworks* y librerías. A continuación, se explica que representan ambos conceptos, pero por otorgarles una breve introducción, ambos comparten la característica de nacer a partir de otro lenguaje ya existente y su principal objetivo es sobre dicho lenguaje reducir la complejidad de desarrollo. Así pues, ambos conceptos cada vez se usan con mayor frecuencia hasta tal punto que hoy en día prácticamente todos los proyectos hacen uso de los *frameworks* y librerías.

3.1 Framework

Se puede definir como un marco de trabajo formado por un conjunto de herramientas basándose en un único lenguaje de programación. Ofrece una estructura para un proyecto completo y el conjunto de funcionalidades, desde el inicio hasta el final. Es decir, el programador se ha de adaptar al *framework* en cuestión. Esto puede parecer una desventaja, pero la realidad es que la estructura a la que se ha adaptar el desarrollador está pensada para que resulte una tarea intuitiva y que, en definitiva, se consiga que éste lleve a cabo el desarrollo de una manera más amigable.

3.2 Librería

Se puede definir como un conjunto de funcionalidades que resuelven problemas específicos. Por ejemplo, si un proyecto necesita validar un formulario se puede hacer uso de una librería que su único fin sea ese, evitando al desarrollador la tarea de controlar que todos los cambios sean válidos, controlar excepciones y un sinfín de casos a tener en cuenta. Al igual que para la validación de formularios, existen para animaciones, para manejar peticiones, fechas y una larga lista de tareas comunes que existen hoy en día. Básicamente es código que ya ha realizado otra persona y decide compartirlo de manera gratuita en internet.

Un error frecuente es el de intentar comparar ambas herramientas. Se suele caer en el error de pensar que una librería es menos potente que un *framework* puesto que este último te ofrece una estructura de trabajo completa. Pero lo cierto es que no tiene ningún sentido hacer una comparativa entre ambos. Cada uno tiene sus ventajas y sus desventajas, lo que no hay que olvidar es que cada uno se utiliza según qué proyecto o incluso es habitual emplear ambos, el uso de uno no resta al otro, se complementan.

Como es evidente, para este proyecto se ha hecho uso tanto de un *framework* como de librerías para reducir la complejidad del desarrollo y construir un código más sencillo y legible.

Antes de pasar a explicar qué tecnologías se han empleado hay que poner en contexto otro tecnicismo que, en realidad, ya se ha hecho una pequeña referencia en este mismo punto. Este tecnicismo es el de código abierto o, en inglés, *open source*. Este concepto se refiere al código fuente del software permitiendo a toda persona interesada acceder a él con el fin de ser cambiado para mejorar o cambiar alguna funcionalidad para un uso personal o distribuirlo para que cualquier otra persona pueda hacer uso de él.

3.3 HTML

Las siglas hacen referencia a *HyperText Markup Language*, es decir, Lenguaje de Marcas de Hipertexto. Este lenguaje básicamente indica qué elementos tiene una página, como puede ser un párrafo de texto o un botón y en qué orden o disposición se han de mostrar. Por ejemplo, para el encabezado del sitio web a través del *HTML* se ha definido que éste tenga una imagen de fondo, un título, un subtítulo, un buscador y un menú desplegable para cambiar de idioma.

El navegador se encarga de leer este lenguaje interpretando los elementos especificados y es capaz de construir la interfaz añadiendo dichos elementos. Así pues, este lenguaje se carga en el lado del cliente o *front-end*.

3.4 CSS

Las siglas hacen referencia a *Cascading Styles Sheets*, es decir, Hojas de Estilo en Cascada. Este lenguaje se encarga de darle estilos a los elementos que tiene una página, es decir, al *HTML*. En el apartado 3.3, se ha nombrado un párrafo de texto o botón, este lenguaje podría definir que letra y tamaño usar o que color de fondo puede tener estos elementos.

Se le denomina en cascada porque se aplican de arriba abajo siguiendo un patrón denominado herencia y en caso de ambigüedad, se siguen una serie de normas para resolverlas. *CSS* permite crear una interfaz visible y bonita que hace que la experiencia del usuario sea más gratificante. Además, también permite gestionar el diseño dependiendo del dispositivo como, por ejemplo, un móvil o un ordenador. La diferencia del tamaño de pantalla obliga a distinguir un estilo para dispositivo para que la navegabilidad no se vea afectada. Esto recibe el nombre de *responsive design*. Un sitio web es *responsive design* cuando éste se adapta a cualquier dispositivo. En la actualidad, esto es un requisito indispensable para cualquier sitio web. *CSS* se trata de otro lenguaje que transcurre en el lado del *front-end*.

HTML y *CSS* conforman la interfaz, el primero indica el contenido gráfico que tiene la página y el segundo el aspecto visual que han de tener cada uno de ellos.

3.5 JQuery

Es una librería de *JavaScript* de código abierto. Esta librería, que también se ejecuta en la parte del cliente, se encarga de ofrecer una lógica en un sitio web, es decir, permite al usuario permitir interactuar con el sitio web. Cuando se habla de lógica se hace referencia a, por ejemplo, cuando se acciona un botón que ha de suceder o cuando se rellena un formulario comprobar que los campos rellenados son correctos.

También se utiliza para manejar información dinámica, esta clase de información hace referencia a valores que varían dependiendo de unos criterios. Por ejemplo, en una cesta de la compra a medida que se añaden productos el valor de gasto va incrementando, ese campo se dice que es dinámico. Así pues, a través de este lenguaje se puede modificar la interfaz o lo que se conoce en el mundo del desarrollo web como el *DOM* (*Document Object Model*) en tiempo real. *JQuery* permite emplear todas estas funcionalidades y muchas más de una manera más sencilla que si se usara *JavaScript*.

3.6 Boostrap

Esta es una librería basada en *CSS* y *JavaScript*. Esta librería tiene como principal objetivo conseguir que el sitio web se adapte a cualquier dispositivo con la capacidad de conectarse a internet. Esta característica de una web recibe el nombre de *responsive design*. Esto se puede conseguir directamente con *CSS*, pero como se ha explicado anteriormente, a través de esta librería se consigue que la misión sea mucho más sencilla consiguiendo un código más minimalista y escalable.

Además, ofrece diseños por defecto como puede ser la tipografía y los márgenes haciendo que el desarrollador tenga más facilidad para construir la interfaz. Su uso de *JavaScript* es debido a que permite crear elementos que pueden ser interactivos o no con el usuario como pueden ser los menús desplegables, ventanas emergentes, etc. Como la grandísima mayoría de librerías y *frameworks*, es de código abierto, lo cual permite cambiar el código fuente como, por ejemplo, cambiar algún margen o color por defecto o simplemente añadirle más características.

Con estos tres lenguajes se ha construido el *front-end*. Con *HTML*, *CSS* y *Boostrap* se ha creado la UI o interfaz gráfica, esto se refiere al aspecto visual que va a tener la página web, es decir, la disposición, forma y colores de todos elementos que conforman la aplicación. Por último, con *JQuery*, se ha creado la UX que es la experiencia del usuario al navegar por el sitio web. La experiencia del usuario se puede medir en la facilidad o lo intuitiva que sea la aplicación. En resumen, se busca que el usuario no haya de pensar o hacer un esfuerzo para conseguir su objetivo final.

3.7 Django

En la parte del servidor o *back-end* se encuentra este *framework* de código abierto, escrito en *Python* basándose en el modelo-vista-controlador (MVC). *Python* en los últimos se ha convertido en los lenguajes más populares hasta tal punto que en el ranking PYPL (*Popularity of Programming Language Index*), consultado el 15 de junio de 2021, se ha situado en el primer puesto. Este ranking se crea a partir de datos extraídos de *Google Trends*, que analiza cuantas búsquedas se ha hecho en cuanto documentación, foros, tutoriales, etc. Para este proyecto se ha elegido este *framework* por basarse precisamente en *Python* puesto que este lenguaje encaja muy bien con las necesidades del proyecto.

En esta aplicación web se maneja una gran cantidad de información. Pues bien, *Python* es ideal para este tipo de proyectos por su sintaxis y oportunidades a la hora de manipular grandes cantidades de datos. En otras palabras, el desarrollo del proyecto usando este lenguaje es un reto más sencillo y cómodo en comparación a los otros lenguajes existentes. Además, este lenguaje posee miles de librerías, propias y creadas por la comunidad expandiendo así sus funcionalidades. Por ejemplo, para esta aplicación, se trabaja con *Twitter*, pues se dispone de varias librerías para comunicarse con esta red social y recoger datos de la misma.

Django fue diseñado para transportar estas cualidades de *Python* al campo de la web, es decir, lograr que este lenguaje pueda funcionar en un servidor. Además, la curva de aprendizaje de este *framework* es muy corta, es decir, se aprende con mucha facilidad. Para este trabajo, se ha hecho un uso muy básico de este *framework* ya que solo se solicitan peticiones, se procesan los datos y se devuelven. Pero *Django* ofrece bastantes funcionalidades por defecto que, además al ser código abierto, permite modificarlas en base a las necesidades del proyecto en cuestión.

Como se ha mencionado anteriormente, *Django* presenta la estructura modelo-vista-controlador. El principal objetivo de esta estructura es el de tener un código bien diferenciado por estas tres partes con el fin de conseguir un código escalable y bien diferenciado. La vista hace referencia a la interfaz del usuario desde donde éste podrá realizar peticiones. Estas peticiones llegan al controlador que, a su vez, las manda al modelo. El modelo, que es donde se almacena todos los datos, procesa los datos solicitados y los devuelve al controlador. Por último, éste los envía a la vista para que el usuario pueda visualizar estos datos. Más adelante se ampliarán estos conceptos cuando se explique la implementación que se ha llevado a cabo.

3.8 *MongoDB*

Se mencionado en numeras ocasiones de que esta aplicación maneja una gran cantidad de información, pero no se ha explicado dónde y cómo se guarda esta información. Pues bien, para este proyecto se ha elegido la base de datos *MongoDB* debido a sus características.

En la actualidad, se distinguen dos bloques en cuanto a base de datos se refiere: *SQL* y *NoSQL*. Cuando se habla de *SQL*, hace referencia a que los datos se almacenan en registros en forma de tablas y estas tablas se pueden relacionar unas con otras. Este tipo de base de datos también son denominadas relacionales. Por ejemplo, se podría tener una tabla de usuarios y otra de *tweets*. Es evidente que un usuario puede tener cero, uno o muchos *tweets*. Así pues, habría que enlazar ambas tablas de tal manera que se pueda saber cuántos *tweets* tiene un usuario o, dicho de otra manera, quién es el autor de cada *tweet*. Este es un caso muy sencillo, pero cuando se tienen muchos datos se requieren de un número de mayor de tablas y, por tanto, de relaciones. Esta tarea es bastante tediosa pues el diseño lógico de la base de datos repercute directamente con la eficiencia de la aplicación en cuestión.

NoSQL o base de datos no relaciones no se organiza en tablas sino en documentos. Este tipo de almacenamiento empezó a aparecer con la llegada de las redes sociales puesto que contienen datos masivos y las bases de datos relacionales ralentizaban las operaciones. *MongoDB* es la base de datos *NoSQL* más popular hoy en día. Los documentos se almacenan en *BSON*, una representación binaria de *JSON*.

Se ha elegido esta base de datos para este proyecto en contra de las bases de datos relaciones por varias razones: su estructura puede ser dinámica, permite redundancia de datos para agilizar las consultas y prima la velocidad sobre la integridad de los datos, que es una característica que no se necesita para este proyecto. Además, simplifica significativamente el diseño lógico de la base de datos a diferencia de las relacionales, que hay que elegir sabiamente cuantas tablas se ha de tener y cómo relacionarlas.

Para tener una idea más clara sobre el formato de *NoSQL*, este podría ser un pequeño ejemplo de la estructura que se usa para almacenar los datos:

{

id_tweet: 4265748,

texto: las bases de datos no relacionales ofrecen más velocidad, pero no integridad de los datos #nosql #bbdd

hashtags: [#nosql, #bbdd]

}

Aquello que va antes de los dos puntos se dice que es la clave o *key* y se usa para identificar y poder recuperar el valor o *value* que va continuación de los dos puntos. En el atributo *hashtags* se puede observar que tiene una estructura más característica, en concreto, se ha utilizado los signos de los corchetes. Esto en el campo de la programación hace referencia a una lista de elementos. En este caso, un *tweet* puede tener cero, uno o muchos *hashtags*, es decir, una lista. En una base de datos relacional esto se haría creando una tabla de datos exclusiva para almacenar los *hashtags* y posteriormente enlazar los *hashtags* con sus correspondientes *tweets*. Este es un ejemplo del gran potencial de *MongoDB* puesto que ha permitido llevar a cabo esta relación de manera inmediata.

4. Desarrollo de la aplicación

El desarrollo de esta aplicación tiene como misión principal recoger información publicada en *Twitter* y mostrarla en una interfaz amigable y compacta. La muestra que se recogió estaba formada por un total de cien mil *tweets* recogidos el 19 de marzo del año 2020 sobre la COVID-19. Cada *tweet* tiene el aspecto del ejemplo expuesto al final del apartado anterior con todos los atributos útiles para mostrar. Para explicar este apartado, se va a diferenciar entre las tres partes que componen este proyecto.

4.1 Modelo-vista-controlador

Antes de pasar a explicar cada parte del proyecto individualmente se va a explicar cómo funcionan conjuntamente y, en definitiva, cómo funciona el flujo de información de la aplicación para lograr su correcto funcionamiento. Como ya se ha mencionado en el apartado 3, para el desarrollo de este trabajo se ha utilizado el *framework Django*. Este *framework* utiliza una estructura bastante común actualmente que consiste en dividir la aplicación en cuestión en tres partes: modelo, vista y controlador (MVC). Este tipo de estructura es muy famoso y utilizado porque permite distinguir a la perfección cada parte del proyecto obteniendo ventajas como saber identificar de una manera rápida un posible error. A continuación, se adjunta un esquema que representa como funciona esta aplicación web.

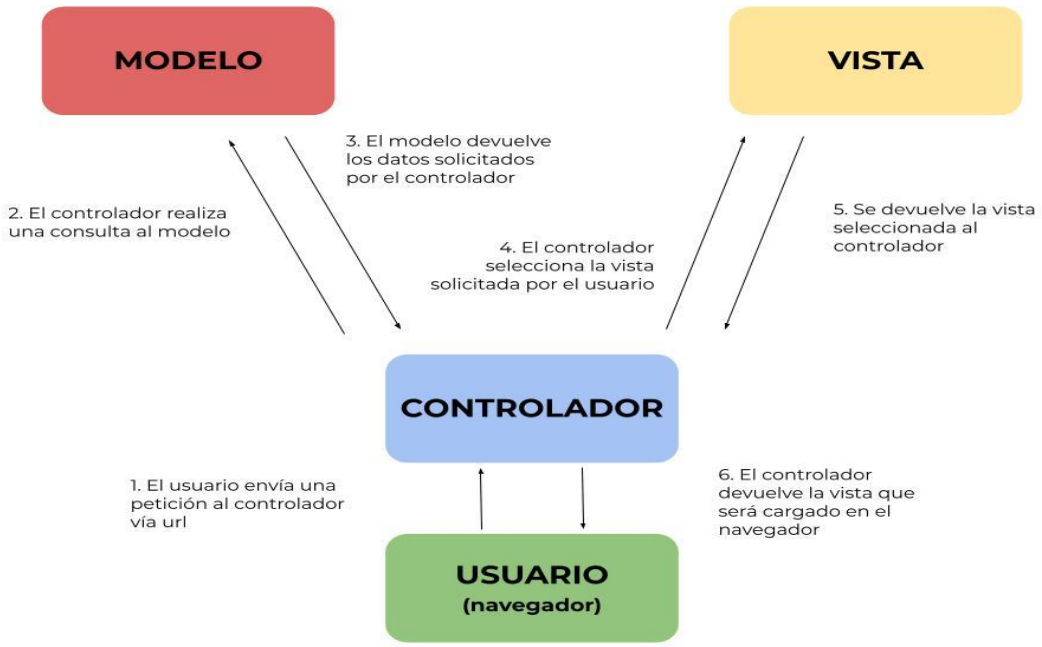

Figura 2. Esquema donde se reflejan los tres componentes y el flujo de datos de una manera estandarizada.

Como se puede observar en la Figura 2, hay un total de seis pasos enumerados en orden con una breve explicación de lo que representan cada uno de estos pasos. Básicamente, el usuario, a través de un navegador, cuando accede al sitio web desencadena el primer paso. Al controlador le llega la petición por parte del navegador. Para este proyecto, existe la presencia de un modelo donde se alojan los datos recogidos de *Twitter*. Así pues, se da paso al segundo paso, el controlador para devolver toda la información necesaria al navegador necesita hacer una consulta al modelo. Tras recoger todos los datos pertinentes, el modelo devuelve estos datos al controlador dando paso al tercer paso. A continuación, el controlador selecciona la vista correspondiente donde volcar los datos obtenidos en el anterior paso. Por último, la vista se devuelve al controlador para posteriormente enviarla al navegador del usuario donde se podrá visualizar la página web.

El párrafo anterior explica las bases de un sitio web que utilice la estructura de modelo-vista-controlador y, por tanto, de este trabajo. No obstante, para este proyecto se han utilizado otras estrategias y tecnologías que hacen que el esquema no refleje a la perfección el funcionamiento y flujo de datos de este trabajo.

Así pues, existen distintos matices que hay que poner en contexto. Pero antes de ello, hay que diferenciar dos tipos de información que están presentes en la aplicación:

1. Información estática: esta clase de datos hace referencia a aquellos elementos que son constantes, es decir, siempre tienen el mismo valor. Por ejemplo, el título de la página web, la descripción, una imagen, etc.

2. Información dinámica: esta clase de datos hace referencia a aquellos elementos que son variables, es decir, su valor puede cambiar en base a distintos factores como, por ejemplo, dependiendo del rol del usuario.

Realmente esta distinción es muy particular de cada página, una imagen puede ser dinámica o estática, es decir, que un elemento sea estático o dinámico no viene definido por su tipo, sino por su comportamiento. Por ejemplo, una imagen de perfil de usuario es dinámica porque dependiendo del usuario, la imagen será una u otra. En cambio, el logo de la página que, no deja de ser una imagen, es un dato estático porque siempre va a ser la misma. También, por ejemplo, el precio final de la cesta de una tienda *online*, a medida que se van añadiendo productos al carrito el valor de este campo va incrementando pues es la suma de todos los precios de los productos añadidos a la cesta.

Una vez aclarada esta distinción, esta aplicación web está desarrollada de tal forma que el usuario hace la petición al controlador y éste le devuelve la información estática y, posteriormente, la dinámica. En otras palabras, cuando el usuario hace la petición para cargar los datos en su navegador, el controlador carga la vista sin hacer la consulta al modelo. Como es evidente, la página se carga sin los datos recogidos de *Twitter*, únicamente con la información estática como puede ser los títulos, imágenes de fondo, etc. Es en este momento, a través de un fichero *JavaScript*, cuando se solicita al controlador que realice las consultas al modelo. El motivo de esta implementación es debido a que cargar la información estática es inmediato y la correspondiente al modelo requiere de una duración que depende del tamaño y el diseño lógico de la base de datos.

Así pues, si la implementación fuera como en el esquema donde solamente se realiza una petición para obtener toda la información necesaria para rellenar la vista, el usuario estaría viendo una página en blanco en su navegador hasta el momento que se devuelva la vista al navegador. Como se ha mencionado anteriormente, la población actual tiende a abandonar las páginas que tardan en cargar por lo que es un requisito necesario que el sitio web no tarde demasiado en cargar. Además, puede crear incluso frustración y desconcierto puesto que pueden llegar a pensar que tienen un problema de conexión a internet con su dispositivo, cuando la realidad es que el sitio web está procesando la petición.

Con esta estrategia, se consigue que, al menos, el usuario visualice de manera instantánea en su navegador el sitio web con la información estática consiguiendo un efecto de éxito, el sitio web el cual ha intentado acceder funciona y es consciente de que está cargando.

En el momento que la vista ha cargado los datos estáticos, se realizan las peticiones dinámicas para terminar de mostrar toda la información deseada. Estas peticiones (anteriormente se ha explicado porque se hacen varias peticiones más pequeñas) se van mostrando conforme el controlador haya recuperado la información del modelo. El usuario podrá saber que los datos dinámicos se están cargando porque en sus huecos aparece un cargador o, *loader* en inglés, que refleja precisamente este estado. Conforme se van cargados los datos dinámicos van desapareciendo estos *loaders* y en su lugar aparece la información correspondiente.

Estas peticiones se han hecho en hilos independientes lo que permite al usuario poder navegar por la aplicación mientras estos datos se cargan. Una vez el controlador ha recuperado y procesado los datos, los envía a la vista y mediante la librería de JQuery se incluyen en el *HTML5*. Esta tecnología que permite hacer peticiones en un segundo plano recibe el nombre de *AJAX*, *Asynchronous JavaScript And XML*.

En la figura 3 se muestra el flujo de datos de la aplicación. Se puede observar que es muy parecido al anterior, pero con los matices que se han explicado hace un momento.

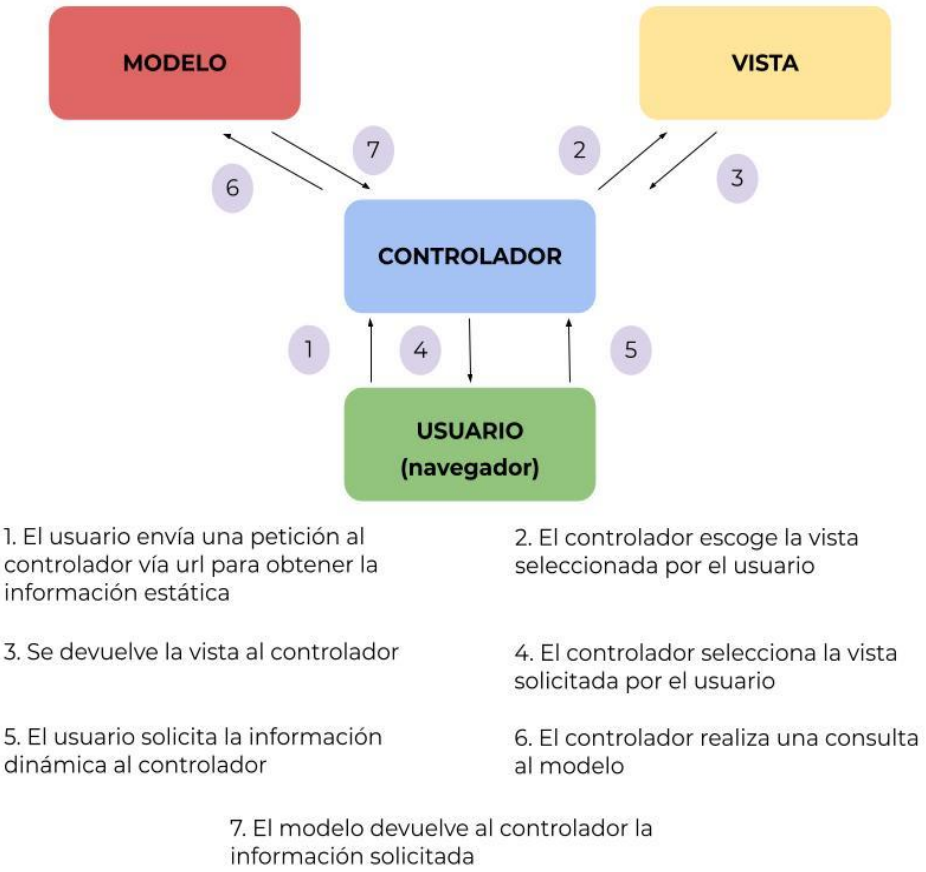

Figura 3. Esquema donde se reflejan los tres componentes y el flujo de datos de este proyecto.

4.2 Modelo

Para un primer desarrollo de la aplicación se ha pensado en mostrar la siguiente información:

1. Los *hashtags* más importantes
2. Los *tweets* más importantes
3. Los usuarios con mayor repercusión
4. Los usuarios con mayor actividad
5. Gráficas temporales para saber en qué momento ha habido mayor incidencia

Toda esta lógica es en base a la muestra proporcionada. Por ejemplo, los *hashtags* más importantes serán aquellos que más se repiten en la colección. Un *tweet* puede tener *retweets* y *likes* por parte de los usuarios. Cuando un usuario hace un *retweet* a un *tweet* de otro usuario está añadiendo al tablón de su perfil dicho *tweet,* como si lo hubiera escrito él mismo. Un *like* se da cuando a un usuario le gusta cierto *tweet* ya sea por humor, mismo pensamiento, etc. pero no es lo suficientemente bueno para el usuario como para darle *retweet* y viaje hasta su tablón.

Así pues, para determinar que *tweets* son los más importantes, se mira el número de *retweets* y *likes* que ha tenido y se sumarán siendo las cifras más elevadas de estas sumas los *tweets* más importantes. Cabe mencionar que se ha premiado en un 50% más los *retweets* ya que es más complicado que algún usuario realice esa acción frente a un *like*. Para ello se multiplica el número de *retweets* que ha obtenido un *tweet* por 1,5.

Los usuarios con mayor repercusión serán aquellos que tengan la puntuación más elevada de la suma de la puntuación de sus *tweets* en la colección. La ecuación que refleja esta puntuación es la siguiente:

$$Puntuación = \sum_{i=0}^{n} 1.5 * retweets_i + likes_i$$

donde *n* es la cantidad de *tweets* que tiene un usuario. Como es evidente, los usuarios más activos serán aquellos que tengan la mayor cantidad de *tweets*. Para ello, se hace una suma del total de *tweets* que aparecen en la colección.

Para armar las gráficas que reflejan el tiempo y la cantidad de *tweets* que se han publicado se hace de uso del atributo *date*, el cual presenta el siguiente formato:

- Thu Mar 19 12:44:36 2020

El primer elemento representa el día de la semana y el segundo representa el mes del año. Ambos abreviados con la nomenclatura inglesa. El tercer elemento representa la hora en la que se publicó el *tweet* siendo primero la hora, después los minutos y por último los segundos. En la última posición, se encuentra el año de la publicación del *tweet*.

En total el cliente ha de hacer cinco peticiones para recuperar toda la información que se necesita mostrar. Tal y como están los datos almacenados, hay que recorrer varios documentos para lograr obtener toda la información. Como se ha explicado durante esta memoria, recuperar los datos a través de una sola petición, es decir, consultar solo un documento, para este trabajo, es una tarea poco eficiente y la rapidez de la aplicación es un requisito indispensable.

Según fuentes como *Acens* consultado el 15 de junio de 2021, el 57% de los españoles abandonan una web si tarda más de tres segundos en cargar. Esto hace que el tiempo de carga sea una prioridad en el desarrollo de cualquier sitio web. En próximos apartados se aborda los tiempos que maneja esta aplicación a la hora de recuperar los datos.

Ante una sociedad tan impaciente surge la necesidad de reducir el coste temporal para obtener una respuesta más rápida. Para ello, se ha hecho una restructura de los datos para lograr esto mismo. En un inicio, se obtiene un fichero *JSON* a partir de *Twitter* donde cada elemento es un *tweet* con sus correspondientes atributos. Con la toda la información que se quiere mostrar y con esta estructura inicial el coste temporal sería demasiado elevado. Así pues, se ha creado un *script* escrito en *Python* que se encarga de leer este fichero obtenido de *Twitter*, crear la base de datos y depositar toda la información en ella.

Se ha diseñado este diseño lógico en la base de datos con el fin de reducir el coste temporal a la hora de recuperar los datos:

1. tweets_es
2. tweets_en
3. hashtags_es
4. hashtags_en
5. users_es
6. users_en
7. tweets_of_user_es
8. tweets_of_user_en
9. score_tweets_es
10. score_tweets_en

Como se puede observar, cada registro se repite dos veces cambiando la parte final. Este código hace referencia al lenguaje o procedencia del tweet. El fichero obtenido de la API de *Twitter* contiene *tweets* procedentes de todo el mundo.

No obstante, para esta primera versión, se ha decidido trabajar solo con dos idiomas: el español y el inglés, es decir, el resto de los lenguajes son despreciados y, por tanto, no se almacenan en la base de datos. Esto es debido a que un usuario principalmente quiere ver información acerca de su país o leer *tweets* en su mismo idioma para poder entenderlos. No tiene mucha cabida mostrar un *tweet*, por ejemplo, en ruso a una persona española.

Aunque inicialmente la aplicación solamente guarda estos dos idiomas, el fichero que se encarga de guardar el *tweet* si está en español o en inglés o, por el contrario, desecharlo, está preparado para añadir tantos lenguajes como se desee. Básicamente, se ha creado una lista de aquellos lenguajes que se quieren conservar. Así pues, si se quisiera añadir otro lenguaje al proyecto, por ejemplo, el francés, bastaría con añadir esa lista el código de lenguaje del francés que, en este caso, es *fr* y automáticamente se crearían todos los registros anteriores para este idioma.

Respecto sobre qué almacena cada par de registros, este es el contenido y su finalidad:

1. *tweets_es* o *tweets_en*: en estos registros se guardan los *tweets* tal y como se recuperan de *Twitter*.
2. *hashtags_es* o *hashtags_en*: en estos registros se guardan los *hashtags* cuyo identificador es el texto. Además, cada uno tiene una lista con los identificadores de aquellos *tweets* donde aparecen.
3. *users_es* o *users_en*: en estos registros se almacenan todos los usuarios encontrados en la colección, procedentes de países de habla hispana o inglesa. Además, cada uno de ellos tiene una lista con los identificadores de los *tweets* que han publicado.
4. *tweets_of_user_es* o *tweets_of_user_en*: en estos registros se guardan como elemento identificador el usuario con el fin de almacenar y recuperar todos aquellos *tweets* que han publicado. Estos registros se utilizan para recuperar los usuarios con mayor actividad de la colección.
5. *score_tweets_es* o *score_tweets_en*: en estos registros se encuentran los identificadores de los tweets y la puntuación de cada uno de ellos. Estos registros se hacen uso a la hora de querer recuperar los *tweets* más importantes de la colección.

Como se puede observar, ahora la base de datos presenta una estructura más organizada. Es cierto que se repiten ciertos datos, pero esta es una de las características que ofrecen las bases de datos no relacionales y, aunque el coste espacial sea algo mayor, los beneficios conseguidos respecto al coste temporal son mayores. En este tipo de aplicaciones no se busca la consistencia de los datos, sino la velocidad.

4.3 Controlador

Una vez presentado el modelo del proyecto, se puede dar paso a poner en contexto el controlador. Como ya se ha explicado anteriormente, el controlador hace las consultas a la base de datos para consultar o modificar información. Para este caso, solo hace falta realizar consultas.

El controlador presenta una estructura similar que el modelo con el fin de conseguir un código compacto y fácil de comprender. La base de datos tiene una serie de registros, cada una con una estructura propia en base a los datos guardados. Así pues, el controlador tiene tantas peticiones como colecciones en la base de datos sin tener en cuenta los idiomas, es decir, aunque existan dos colecciones para guardar los *hashtags* más relevantes, el controlador solo hace una petición a una de ellas.

Con esto se consigue una estructura separada por módulos o modular, que actualmente, es muy utilizado en el mundo del desarrollo y hace posible que el proyecto sea escalable y fácil de mantener.

Estas consultas se hacen de manera paralela. Esto quiere decir que no se ejecuta una detrás de otra o de manera secuencial, sino que se llaman al mismo tiempo y será el gestor de la base de datos quien se encargue automáticamente de gestionarlas. El tiempo que se tarde en recuperar los datos dependerá de las características del servidor. En cualquier caso, el coste temporal se ve mejorado y el cuello de botella es por parte del *hardware* y no del *software*, quedando fuera del alcance del desarrollador esta limitación.

En este momento del desarrollo todavía no existía una vista con la que poder realizar las peticiones y comprobar el correcto funcionamiento del controlador. No obstante, existen servicios de clientes que permiten realizar consultas al controlador sin la necesidad de tener una vista. Para este caso, se ha utilizado la aplicación *Postman*. Gracias a esta herramienta se ha podido realizar pruebas o *tests* y poder monitorizar estas consultas que se hacen al controlador o API.

4.4 Vista

Una vez terminado el desarrollo del modelo y controlador, se dio paso a la creación de la vista o interfaz. Se podría definir esta parte como la más importante del proyecto y, por tanto, donde más esfuerzo se ha dedicado. El usuario va a interactuar únicamente con esta parte del proyecto, por eso y como se ha mencionado anteriormente, hay que tener cuidado tanto con el diseño como con la usabilidad. De nada sirve que el *back-end* sea rápido si luego la información mostrada no está bien distribuida y, en definitiva, el resultado es engorroso.

Se ha querido hacer un diseño minimalista pero compacto para que el usuario no tenga que hacer ningún esfuerzo para entender cómo funciona el sitio web. La apariencia está un poco basada en la red social de *Twitter* en cuanto a los colores que usa ésta, en concreto, el color predominante siempre ha sido el azul y también la estructura que tiene para mostrar los *tweets*. Se ha usado esta estrategia con el fin de intentar causar un efecto en el usuario de que se encuentra en la propia red social.

A continuación, se van a adjuntar varias imágenes para poder ver de una manera más clara como ha quedado la interfaz y qué interacciones con el usuario presenta. Las imágenes están ordenadas de tal forma que la primera es lo primero que ve el usuario y las siguientes conforme se va deslizando hacia abajo en la página.

Figura 4. Primera parte de la vista de la aplicación web.

En la figura 4, se puede observar que lo primero que captan los ojos del usuario es un título y subtítulo que tratan de definir en que consiste este sitio web acompañado de una imagen de fondo que le da una apariencia moderna y llamativa que sigue con esta misma temática. Como se puede observar, en la esquina de la derecha superior aparece un botón que al hacer *click* sobre él libera un menú desplegable con todos los idiomas disponibles en la aplicación para que el usuario pueda cambiar el idioma si así lo desea. Por defecto, *Django* permite conseguir de manera inmediata en que lenguaje tiene el usuario su navegador y en base a esto recuperar en la base de datos la información en su idioma. Si este idioma que tiene el cliente no está en el modelo, entonces se extrae la información en inglés ya que es el lenguaje estándar a nivel mundial. No obstante, se ha querido dar la opción y libertad al usuario de tener la opción de poder cambiar el idioma a mostrar y con ello, los datos que se vuelcan en la vista. El controlador y el modelo están preparados para añadir de manera inmediata un nuevo lenguaje si se desea, como se ha explicado en el apartado 4.2.

Figura 5. Primera parte de la vista con el menú desplegable de los idiomas activo.

En este primer contacto del usuario con la web, también se ha incluido un elemento interactivo, como es un buscador. En concreto, este buscador permite al usuario buscar un *hashtag*. En caso de que éste se encuentre en la base de datos, se cargará una nueva vista con toda la información relacionada con dicha búsqueda. El campo ocupa casi todo el ancho de la pantalla con el fin de que el usuario lo visualice rápidamente y con un texto de fondo, que recibe el nombre de *placeholder* en el campo del desarrollo web, para indicarle al internauta que es un buscador exclusivamente de *hashtags.* Por último, a la derecha está al botón que lanza la acción de buscar *hashtag* introducido por el usuario que, en lugar, de añadir un texto se elegido poner un icono que representa una lupa para conseguir un mejor diseño. No obstante, también puede ocurrir que el *hashtag* introducido no esté almacenado. Ante esta circunstancia pueden ocurrir dos casos.

El primero ocurre cuando el *hashtag* no se ha encontrado, pero existen otros que son muy parecidos al introducido, es decir, se escriben prácticamente igual. Para rastrear estos *hashtags* parecidos se ha hecho uso de un algoritmo que calcula exactamente cómo de parecidas son dos palabras o qué distancia tienen. Así pues, el algoritmo calcula cuántas letras se han tenido que manipular en una de las dos palabras para que se escriba igual que la otra, es decir, que sean idénticas. Este número de manipulaciones empieza con un valor de cero y se va incrementando a medida que se va modificando dicha palabra. Así pues, esta distancia es la que define cómo de parecidas son dos palabras y es el valor que devuelve este algoritmo, que recibe el nombre de *Levenshtein*. Para entender un poco mejor el funcionamiento, veamos un pequeño ejemplo:

1. casa → casi: sustitución de ‘a’ por ‘i’, distancia de uno
2. cala → calla: inserción de una ‘l’, distancia de uno

Como se ha comentado en el párrafo anterior, por cada edición que se realice la distancia se sumará en una unidad. Para este caso, se ha elegido una distancia máxima de cuatro. El uso de este algoritmo tan eficaz es debido a la implementación de una lista de sugerencias de *hashtags* en caso de que el introducido por el usuario no se encuentre. Con esto se pretende dar una posible solución al internauta. Además, a esta lista de sugerencias se le ha añadido una mejora para mejorar la experiencia del usuario. En concreto, se ha implementado una lógica la cual permite al usuario clicar a cada elemento de la lista de sugerencias y automáticamente se escribe esta sugerencia en el buscador. Con esto se evita que el usuario tenga que escribir de nuevo un *hashtag* y simplemente bastará con que pulse en el icono de búsqueda para poder ver la información del *hashtag* que haya seleccionado.

Figura 6. Primera parte de la vista con la lista de hashtags sugeridos activa.

No obstante, si no existe ningún *hashtag* que tenga una distancia menor o igual a cuatro respecto al introducido, entonces se le notifica al usuario que el tema que le interesa no se encuentra disponible. Esto dependerá directamente de que tan grande sea el modelo.

Figura 7. Primera parte de la vista con el mensaje de error al intentar buscar un hashtag no almacenado en el modelo activo.

Si se desliza hacia abajo la página o el término en inglés, *scroll*, aparece al lado izquierdo un gráfico circular donde se muestran los *hashtags* más relevantes. Además, se ha añadido la funcionalidad de permitir al usuario clicar en uno de ellos para mostrar la información respecto éste. Esta acción es el equivalente al buscador del inicio. A su derecha, se muestran los *tweets* más importantes de la colección pudiendo ver el texto, el nombre del usuario que lo ha publicado, su imagen de perfil, la fecha de la publicación, contenido multimedia si lo tiene como puede ser un enlace, foto o video y desde que dispositivo se escribió, desde la web móvil con sistema operativo *Android* o *iPhone* teniendo, además, su propio *scroll* independiente para poder visualizar todos los *tweets* manteniendo el resto de la página web en el mismo estado.

Figura 8. Segunda parte de la vista donde se muestran los hashtags con mayor tendencia y los tweets con mayor repercusión.

Durante el desarrollo surgió el pequeño inconveniente de qué si el usuario había cambiado su imagen de perfil, ésta ya no estaba disponible en la ruta de almacenamiento de *Twitter* por lo que no se podía recuperar. Esto causaba que el diseño perdiera calidad pues donde debería ir la imagen quedaba un espacio en blanco. Así pues, se ha optado por realizar una comprobación de todas estas imágenes para ver si siguen existiendo o no y en el último caso mostrar una imagen por defecto de perfil.

La parte positiva es que el diseño sigue siendo elegante, pero la negativa es que aumenta el coste temporal puesto que al coste de recuperar los datos se suma la comprobación de las imágenes de perfil. Esta comprobación básicamente es hacer una petición a la ruta donde se aloja la imagen y observar que tipo de respuesta devuelve. Existen muchos códigos de respuesta por parte de un servidor, por simplificar solo se va a mencionar solo los dos que interesan para este caso. El primero y que significa que la imagen existe es el código 200 y el segundo que significa que la imagen no existe es el famoso error 404. A continuación, se adjunta la imagen que muestra esta parte de la página donde aparecen los *hashtags* con mayor tendencia de la colección y los *tweets* con mayor repercusión.

Debajo de la gráfica de los *hashtags*, se encuentra la lista de los usuarios con mayor repercusión en cuanto al impacto que han tenido el total de sus *tweets* en base a la ecuación citada en el apartado 6.1. De estos se muestran la imagen perfil, donde también se hace la comprobación a la ruta, el nombre de usuario, el *nickname*, la verificación de *Twitter* si la tiene y el número de seguidores que tiene redondeado hacia abajo, por ejemplo, si a un usuario le siguen 65 432 mil personas se transforma en *+65 000 seguidores* con el fin de facilitar la lectura al usuario.

Figura 9. Segunda parte de la vista donde se muestran las personas con mayor influencia.

Como se puede observar en la Figura 9, se presenta la sección con el título «Personas que quizás te interesen» con la finalidad de causar un efecto de sugerencia en el internauta. Se pretende que en el usuario surja la necesidad de entrar en algún perfil de los mostrados. En el mejor de los casos, puede encontrar algún usuario de su interés y finalmente le siga para poder ver en el tablón la actividad de esta persona.

A continuación, una gráfica aborda todo el ancho de la pantalla para que su lectura resulte sencilla al usuario. Esta gráfica de líneas refleja el tiempo en el eje horizontal y la cantidad de *tweets* publicados en su eje vertical. Así pues, la finalidad de esta gráfica es mostrar al usuario una línea temporal en la que puede observar en qué momentos ha habido una mayor actividad en *Twitter* sobre cierto tema. Además, arriba de esta gráfica hay dos que permiten una interacción del usuario con la gráfica. En concreto, el usuario tiene la opción de cambiar el eje horizontal, que representa el tiempo.

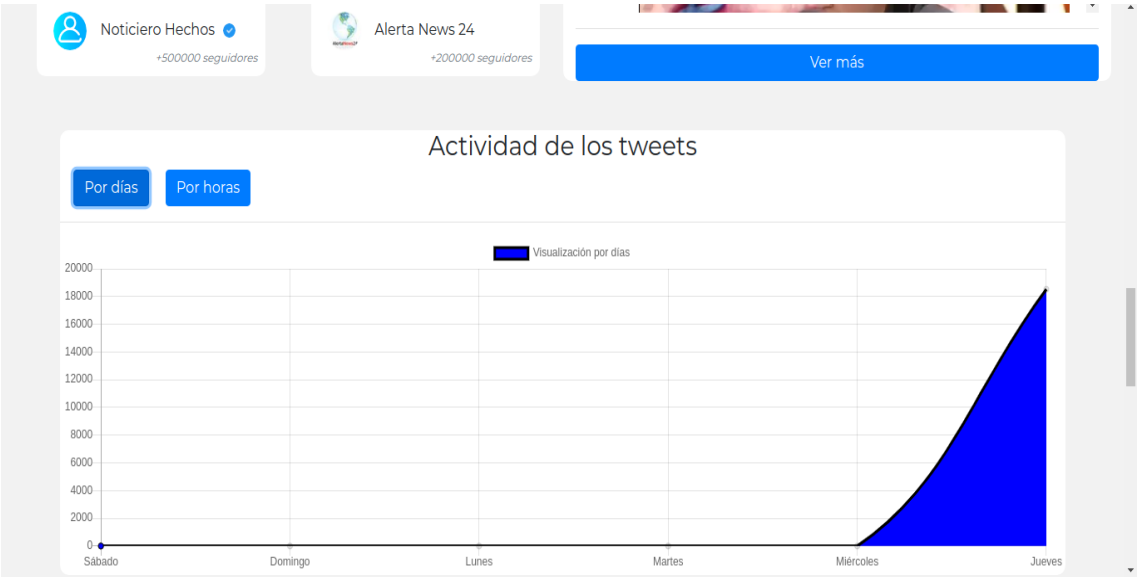

Figura 10. Tercera parte de la vista donde se muestra la gráfica que refleja la actividad de los tweets de los últimos cinco días.

Por defecto, la gráfica muestra la actividad de los datos recogidos de los últimos cinco días, por lo que el eje horizontal mostraría los cinco días previos de la semana al actual. No obstante, a través de estos botones situados en la parte superior de la gráfica, el internauta también puede ver la actividad del día actual, con lo cual el eje horizontal se modifica y muestra las últimas 24 horas. Con esto se le da la posibilidad al usuario de tener la libertad que espacio de tiempo quiere visualizar, añadiendo así al sitio web una mayor usabilidad y, por ende, una mejor experiencia. Además, se ha añadido un título a la gráfica que indica al internauta cómo está visualizando los datos, o los últimos cinco días o las últimas 24 horas.

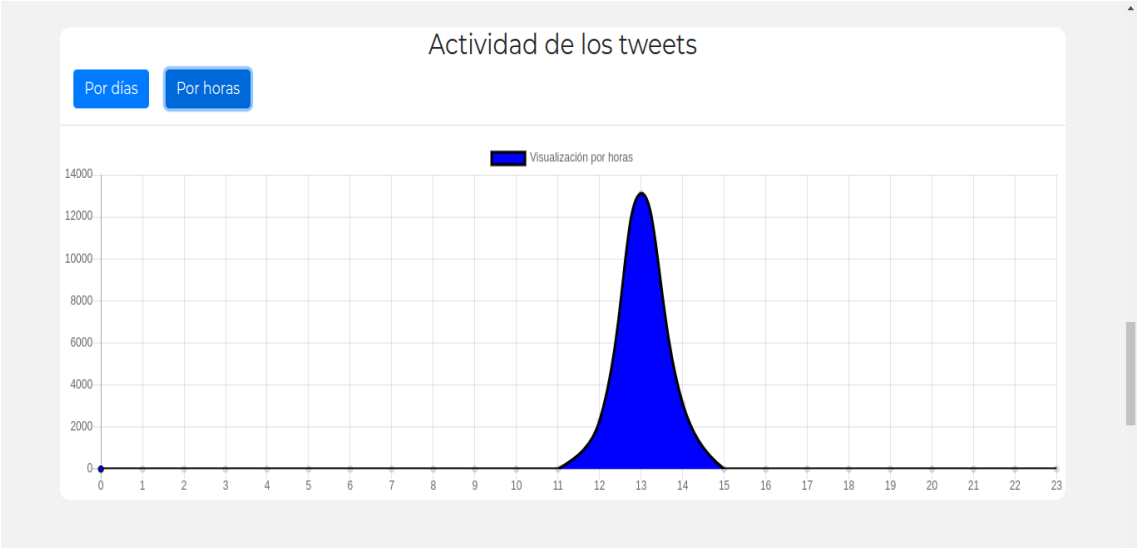

Figura 11. Tercera parte de la vista donde se muestra la gráfica que refleja la actividad de los tweets de las últimas 24 horas.

Por último, se muestra la lista de los usuarios más activos, es decir, aquellos que han publicado más *tweets*. La información que se muestra de los usuarios es igual al de los usuarios con mayor impacto en cuanto a los campos y estructura. Esta última parte se compone de dos bloques, el primero se compone de dos gráficas. La primera muestra la actividad del usuario en los últimos cinco días y la segunda muestra las últimas 24 horas. Por defecto, ambos gráficos muestran la actividad del primero usuario de la lista de los usuarios más activos que, además, siempre será el que mayor actividad presenta.

No obstante, el usuario puede interactuar con ambas gráficas haciendo *click* en cada uno de los usuarios de la lista de los más activos. Esto desencadena una acción que actualiza las dos gráficas mostrando la actividad del usuario *clickado* por parte del internauta. Ambas gráficas tienen un título que su valor depende de la actividad que se está mostrando de los usuarios más activos. Por ejemplo, si se está mostrando la actividad de un usuario llamado Josep, el título de ambas gráficas será dicho nombre. Con esto se consigue que el usuario tenga en todo momento claro que actividad está visualizando.

Esta implementación permite evitar una saturación al usuario ya que, de lo contrario, existirían dos gráficas por cada usuario de la lista de los usuarios más activos y quedaría una vista muy extensa y dificultosa de leer. En definitiva, se ha mejorado la experiencia y la usabilidad del internauta. A continuación, se adjunta dos imágenes que muestran esta parte de la vista.

Figura 12. Cuarta parte de la vista donde se ven los usuarios más activos y la gráfica que muestra su actividad en los últimos cinco días.

Figura 13. Cuarta parte de la vistan donde se ven los usuarios más activos y la gráfica que muestra su actividad en las últimas 24 horas.

Todas estas partes formarían la página principal del sitio web, lo que el cliente ve al cargar. Además de esta vista, existe una segunda que se ha creado para mostrar información relacionada con un *hashtag*. A esta ventana se accede mediante el buscador del inicio o también mediante la gráfica de los *hashtags* que permitía la opción de hacer *click* en alguno de ellos y automáticamente despliega esta página. Esta segunda ventana, presenta una estructura casi idéntica la principal. Se ha realizado así para conseguir un sitio compacto y con la finalidad de que el usuario no tenga que realizar apenas esfuerzo para entender cómo funciona y se visualizan los datos para que su navegación sea lo más agradable posible. La información que se muestra también es la misma, solo que esta vez sobre un *hashtag* en concreto.

5. Experimentación

Un punto fundamental de este proyecto es el tiempo de respuesta que tiene el servidor para responder a las peticiones del cliente. La sociedad actual está acostumbrada a qué todo vaya fluido y rápido, así que es una variable siempre a tener en cuenta y priorizar y más aún cuando se trabaja con una cantidad de datos grande y que, posiblemente, pueda ser mayor todavía en un futuro.

Monitorizar las peticiones y los tiempos de respuesta es una opción muy interesante para tener un control en todo momento sobre la velocidad. Además, no es suficiente con realizar las pruebas una vez puesto que la base de datos va a sufrir modificaciones constantemente y puede disparar el tiempo de respuesta. Así pues, se ha optado por crear un *script* que se encargue de realizar peticiones y calcular los tiempos de respuesta. Este fichero permitirá monitorizar esta variable tan importante de una manera rápida y sencilla.

Este *script* presenta una estructura muy repetitiva. Básicamente realiza diez consultas a cada petición que tiene el proyecto en un momento dado y almacena cada tiempo de respuesta que obtiene en esas diez peticiones. Finalmente, se calcula la media de estos diez resultados con el fin de obtener una aproximación más exacta ya que dependiendo del momento el servidor puede tener más recursos libres o menos y para una misma consulta se puede llegar a obtener tiempos de respuesta diferentes de forma significativa. El *script* genera un fichero de texto con las medias finales de cada petición disponible. A continuación, se adjunta una tabla con los resultados obtenidos.

Estas pruebas son orientativas y dependen bastante de la máquina donde se aloja el servidor. Para este caso, estos resultados se han obtenido de un ordenador particular Lenovo i7, 8GB de RAM, disco duro 256 GB SSD.

Nombre de la petición	**URL**	**Tiempo de respuesta (segundos)**
Buscar hashtag (existe)	find_hashtag	2.125
Buscar hashtag (no existe)	find_hashtag	2.30
Hashtags más relevantes (español)	load_important_hashtags	0.027
Hashtags más relevantes (inglés)	load_important_hashtags	0.043
Tweets con mayor repercusión (español)	load_important_tweets	0.951
Tweets con mayor repercusión (inglés)	load_important_tweets	0.974
Usuarios más influentes (español)	load_important_users	0.942
Usuarios más influentes (inglés)	load_important_users	0.965
Actividad de los *tweets* (español)	load_activity_tweets	1.002
Actividad de los *tweets* (inglés)	load_activity_tweets	2.80
Usuarios más activos (español)	load_activity_users	2.46
Usuarios más activos (inglés)	load_activity_users	2.71
Usuarios más activos de un *hashtag* (español)	load_activity_users_hashtag	1.7
Usuarios más activos de un *hashtag* (inglés)	load_activity_users_hashtag	6.132
Usuarios más influentes de un *hashtag* (español)	load_important_users_hashtag	0.023
Usuarios más influentes de un *hashtag* (inglés)	load_important_users_hashtag	0.038

Actividad de los *tweets* de un *hashtag* (español)	load_activity_tweets_hashtag	2.206
Actividad de los *tweets* de un *hashtag* (inglés)	load_activity_tweets_hashtag	10.558

Como se puede observar todos los tiempos bastante rápidos, algunos incluso ni llegan al segundo. El tiempo más repetido ronda casi los tres segundos. La petición para recuperar los usuarios más activos de habla inglesa es un poco más elevada que el resto, pero está dentro de un rango aceptable. La única petición que tiene un coste algo preocupante y a tener en cuenta su mejora en un futuro es la de recuperar la actividad de los *tweets* respecto un *hashtag* llegando a los diez segundos. Es una marca que es válida para esta primera versión del trabajo, pero una futura mejora podría ser la de mejorar la rapidez de esta consulta ya que conforme el modelo vaya teniendo más datos el coste temporal de ésta tenderá a subir de forma notable hasta tal punto que el tiempo de respuesta será tan alto que será un problema para la aplicación significante.

Hay que mencionar que se ha hecho la distinción entre los lenguajes que soporta la aplicación actualmente, español e inglés, debido a que no hay la misma cantidad de datos para cada uno de ellos. También, añadir que para calcular el coste temporal respecto un *hashtag*, se ha seleccionado el que mayor cantidad de información posee para obtener los peores resultados o tiempos mayores de la muestra utilizada. En otras palabras, en resto de *hashtags* almacenados en el modelo obtendrían mejores resultados que los que se presentan en la tabla anterior.

En conclusión, el balance de los resultados es bastante positivo. Además, cabe recordar que están pruebas se han llevado a cabo en un ordenador particular con unas prestaciones bajas. Estas mismas pruebas mejorarían ejecutadas en un servidor web debido a que son máquinas que están dedicadas exclusivamente a este fin y las piezas que lo conforman o, el término en inglés *hardware*, ofrece unas mejores prestaciones.

6. Relación con los estudios cursados

A lo largo del grado y desde un inicio se han adquirido una serie de conocimientos y conceptos teóricos que han sido fundamentales para llevar a cabo este trabajo y que se han empleado.

En primer lugar, las asignaturas de introducción a la programación (IP) y programación (PRG) que dan las bases y abren las puertas a este ámbito del desarrollo tecnológico. Ambas cursadas en primero de carrera.

En el segundo año, se empieza a ver otra materia ligada a este campo, como puede ser interfaz persona computador (IPC) donde se dan conceptos sobre las interfaces gráficas en cuanto a diseño y usabilidad. En esta asignatura se aprendió a darle a un programa un aspecto visual para que un usuario estándar pueda darle uso. Más tarde, en tercero, se refuerzan estas bases con la llegada de ingeniería del software (ISW) que añadió como estructurar de manera correcta un proyecto. Estructura de datos (EDA) introdujo la manipulación de datos organizándolos de una forma estratégica para conseguir recuperarlos de una manera más eficiente. Otra asignatura con menos peso pero que aun así merece una mención puesto que se estudió conceptos teóricos imprescindibles para el desarrollo web es redes de computadores (REDES).

En tercero de carrera, cabe mencionar a bases de datos y sistemas de la información (BDA) ya que se introduce y se adquieren unos conocimientos bastante sólidos sobre las bases de datos. Cierto es que las bases de datos que se vio no es la misma que la usada en el proyecto. No obstante, los conocimientos adquiridos han permitido poder aprender otro modelo de una manera sencilla.

Por último, en la rama de computación se vieron asignaturas más avanzadas y especializadas en este campo. En concreto, la materia sistemas de almacenamiento y recuperación de información (SAR) es bastante fundamental puesto que en este proyecto se abarca una gran cantidad de información y se hace un uso elevado sobre ellos. También cabe mencionar la asignatura de algorítmica (ALT) donde se estudia como minimizar los costes espaciales y temporales de los algoritmos.

7. Conclusiones

En este trabajo se ha desarrollado una aplicación web que surge de la necesidad de la cantidad de datos que trafica la red social de *Twitter*. El objetivo principal se ha cumplido y consistía en ofrecer al usuario una información sobre algún tema que pueda resultar de su interés de una manera gráfica e intuitiva.

Para lograr este objetivo, primero se ha construido una base de datos donde almacenar la información a mostrar con una estructura que permita recuperarla de una manera eficaz. Para ello, se ha creado un fichero que es capaz de leer la información extraída de *Twitter* y crear la base de datos o el modelo. Para conseguir recuperar los datos de una manera eficaz, se ha optado por empeorar el coste espacial duplicando algunos campos con el fin de mejorar el coste temporal ya que en este tipo de aplicaciones la integridad de los datos no es requisito imprescindible. Por el contrario, la velocidad de este tipo de proyectos sí que es relevante.

A continuación, se ha desarrollado la segunda parte de la aplicación que actúa como intermediario entre la base de datos y la vista, que es la parte donde se muestran los datos. Esta parte recibe el nombre de controlador y se encarga de recibir las peticiones de la vista y enviarle la respuesta pertinente. En algunas ocasiones, el controlador ha de hacer consultas a la base de datos y en otras no, esto dependerá del tipo de petición que realice la vista.

Por último, se ha desarrollado la parte que es la encargada de mostrar los datos almacenados en la base de datos. Esta última parte, la vista, se puede definir como la más importante de este trabajo porque el usuario va a interactuar con ella y donde se va a crear la experiencia final. Así pues, se ha optado por crear un diseño minimalista y fácil de navegar para conseguir que la experiencia sea lo más gratificante posible para el internauta. Además, para conseguir que la aplicación sea rápida se han seguido una serie de estrategias como enviar, en lugar de una petición al controlador para que éste nos devuelva toda la información necesaria para mostrarla en la vista, varias peticiones más pequeñas que requieren un menor coste temporal y que la vista se vaya llenando conforme van llegando las respuestas.

8. Trabajos futuros

Este trabajo presenta una aplicación web totalmente operativa para colgarlo en internet. No obstante, es una primera versión donde toda su implementación se puede mejorar y ampliar.

En el apartado de 5, se puede observar que hay algunos tiempos obtenidos que son posibles mejorar. Un posible trabajo podría ser mejorar estos tiempos. No obstante, también puede estudiarse almacenar la información de distinta manera con el fin de lograr unos costes temporales en todas las peticiones.

En cuanto al controlador, aumentar su escalabilidad, es decir, crear una estructura que consiga que añadir nuevas funcionalidades al controlador sea una tarea sencilla y rápida. Además, también se puede añadir un fichero de propiedades donde almacenar todas las variables globales del proyecto consiguiendo que si en futuro se requiere añadir, cambia o eliminar alguna de ellas se tengan perfectamente localizadas y cualquier modificación de este fichero tenga un efecto en todo el proyecto de manera instantánea.

La vista también es otro aspecto que constantemente puede mejorarse ya sea por añadir nuevas funcionalidades o simplemente ir perfeccionando su diseño y usabilidad para que la experiencia del usuario sea aún mayor.

En definitiva, todo el proyecto en sí está receptivo a cambios y mejoras con el que cada vez se iría consiguiendo una aplicación más sofisticada y, por tanto, capaz de ofrecer un servicio de mayor calidad de cara al público.

9. Bibliografía

1. BRACERO, FRANCESC. El 92% de los usuarios españoles se conecta a través del móvil. *La Vanguardia* [en línea]. 2018. [Consulta: 15 de junio de 2021]. Disponible en:

https://www.lavanguardia.com/tecnologia/20180307/441323800422/movil-internet-online-egm.html

2. El 57% de los usuarios abandona una web si tarda más de 3 segundos en cargar. *Acens* [en línea]. 2013. [Consulta: 15 de junio de 2021]. Disponible en:

https://blog.acens.com/infografias/57-usuarios-abandona-web/

3.PYPL PopularitY of Programming Language index:

https://pypl.github.io/PYPL.html

4. Django Software Foundation. (2019). *Django*. Retrieved from https://djangoproject.com

5. Patel, K. (2013). Incremental journey for World Wide Web: introduced with Web 1.0 to recent Web 5.0–a survey paper. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(10).

6. Duckett, J. (2011). *HTML & CSS: design and build websites* (Vol. 15). Wiley Indianapolis, IN.

7. Flanagan, D. (2006). *JavaScript: the definitive guide*. " O'Reilly Media, Inc."

8. James F. Kurose & Keith W. Ross. *REDES DE COMPUTADORAS: UN ENFOQUE DESCENDENTE*. ISBN: 978-84-7829-119-9

9. [Consulta: 15 de junio de 2021]. https://analytics.twitter.com/about

10. [Consulta: 15 de junio de 2021]. https://es.audiense.com/

11. [Consulta: 15 de junio de 2021]. https://tweepi.com/