UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Agregador de notícias inteligente

Grado en Ingeniería de Software

Trabajo final de grado

Memoria

Autor: Cristian Ruiz Bonilla
Director del proyecto: Carles Farré Tost
Especialidad: Ingeniería del Software
Convocatoria: 2020/2021 Q2

Resum

Actualment les persones fem moltes coses al llarg del dia i tenim poc temps per informar-nos del que està passant al món. En aquest projecte es desenvolupa una API Rest amb Spring i una aplicació web amb Vue.

L'objectiu és oferir una aplicació que permet veure notícies d'actualitat de diferents fonts acord als gusts de cada usuari. Per això, cada usuari podrà valorar les notícies individualment i obtindrà recomanacions d'altres notícies basant-se en això.

Amb aquest projecte és pretén que la gent pugui informar-se de notícies actuals tenint en compte les seves preferències de manera ràpida i senzilla.

Resumen

Actualmente las personas hacemos muchas cosas a lo largo del día y tenemos poco tiempo para informarnos de qué está pasando en el mundo. En este proyecto se desarrolla una API Rest con Spring y una aplicación web con Vue.

El objetivo es ofrecer una aplicación que permite ver noticias de actualidad de diferentes medios acorde a los gustos de cada usuario. Para ello, cada usuario podrá valorar las noticias individualmente y obtendrá recomendaciones de otras noticias en base a ello.

Con este proyecto se pretende que la gente pueda informarse de las noticias actuales teniendo en cuenta sus preferencias de manera rápida y sencilla.

Abstract

Nowadays, people do a lot of things throughout the day and have short time to know what is happening in the world. An API Rest with Spring and an application web with Vue are developed in this project.

The objective is to offer an application to be able to see current news from different sources according to the preferences of each user. To do this, every user will be able to rate any news individually and will obtain some recommendations based on it.

The aim of this project is that people can find out about current news taking into account their preferences quickly and easily.

Índice

Lista de tablas

Lista de figuras

1. Contexto y alcance

1.1 Contexto

1.1.1 Introducción

Hoy en día existen multitud de páginas webs donde podemos buscar información sobre lo que ocurre en la actualidad, ya sea en nuestro entorno más cercano o en la otra punta del mundo. También existen multitud de agregadores de noticias que nos permiten poder acceder a diferentes noticias en la misma página web/aplicación e incluso podemos suscribirnos a ellos para estar al día.

El objetivo de este proyecto es construir un sistema similar al descrito en el párrafo anterior pero añadiendo la peculiaridad de que el software sería capaz de aprender del usuario y conocer cuales son las notícias que le son de mayor interés. Por tanto, el sistema sería capaz, conforme pase el tiempo, de proponer cada vez noticias más afines al usuario y por tanto obtener una experiencia más enriquecedora.

Este trabajo de final de grado de modalidad A pertenece al grado de Ingeniería Informática impartido por la Facultad de Informática de Barcelona. En concreto, bajo la especialidad de Ingeniería del Software. El proyecto surgió por iniciativa propia del estudiante con el objetivo de integrar diferentes tecnologías entre ellas cada vez más comunes en el ámbito laboral, así como conseguir un sistema innovador y de gran utilidad para cualquier persona.

1.1.2 Definición de conceptos

A continuación, se definen una serie de términos con el fin de especificar el significado que van a recibir a lo largo del documento para no confundir al lector.

Agregador de noticias

Un agregador de noticias es un software que permite recopilar contenidos (típicamente noticias) de diferentes fuentes de información con el objetivo de tenerlas todas juntas en un mismo acceso.

API Rest

Una API REST define un conjunto de funciones que los desarrolladores pueden realizar solicitudes y recibir respuestas a través del protocolo HTTP, como GET y POST.

Debido a que la API REST usa HTTP, pueden ser utilizados por prácticamente cualquier lenguaje de programación y son fáciles de probar (es un requisito de una API REST que el cliente y el servidor sean independientes entre sí, lo que permite programarlo en cualquier idioma y mejorar al soportar la longevidad y evolución).

Framework

Un entorno de trabajo, o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

1.1.3 Descripción del problema

Actualmente la gran mayoría de páginas webs/aplicaciones de noticias se limitan a mostrar noticias de actualidad sobre lo que ocurre en el mundo en ese mismo momento.

Alternativamente existen los conocidos agregadores de noticias que nos permiten tener múltiples fuentes de información en una misma página. Por tanto podemos tener toda la información en un vistazo sin necesidad de acceder a diferentes medios para encontrar la información que buscamos.

A pesar de que los agregadores de noticias son muy útiles, podrían serlo aún más si dicho agregador supiese que es lo que más nos interesa y nos lo mostrase directamente sin necesidad de estar filtrando por categorías o utilizando buscadores.

Por tanto, el problema a resolver es justamente ser capaz de crear un sistema inteligente que aprende del usuario a medida que va viendo noticias y por tanto con el tiempo saber sus gustos e inquietudes y poder mostrar información más relevante directamente.

1.1.4 Stakeholders

Los stakeholders son la parte interesada en el proyecto, que hace referencia a una persona, organización o empresa.

Desarrollador

Yo soy uno de los principales stakeholders del proyecto ya que seré quien se encargará de realizarlo de la mejor manera posible con el fin de acabar la carrera y, por supuesto, mejorar mis conocimientos.

Director del proyecto

Carles Farré Tost actúa como director del proyecto encargándose de dirigir y guiar durante todo el desarrollo del mismo.

Personas en general

Cualquier persona que busque saber que está ocurriendo en la actualidad es un stakeholder ya que serán los usuarios finales que utilizarán el sistema.

1.2 Justificación

A día de hoy hay multitud de fuentes de información como pueden ser webs de noticias, agregadores de noticias, comunidades de usuarios, etc. Podemos observar características muy diferenciadas entre ellos fácilmente.

En el primer caso, el modelo más tradicional como sería una web de noticias o un periodico como tal, podemos ver a simple vista las noticias más relevantes del día y en espacios más pequeños noticias menos importantes. El punto diferenciador de este tipo de fuente de información es mostrar de un vistazo las noticias del día actual.
El problema más común de este tipo de webs es que no cuentan con ningún sistema de búsqueda de noticias más antiguas ni de diferentes puntos de vista ya que el único punto de vista existente es el del periodico en cuestión.

En segundo caso, podríamos hablar de agregadores de noticias que nos permite suscribirnos a un conjunto de fuentes de información de nuestro interés con el objetivo de poder ver que noticias publican dichas fuentes. El punto fuerte está en poder tener diversas

fuentes de noticias y poder comparar entre ellas y hacernos una idea más general de cada noticia. Además, por su puesto proporcionan un fácil acceso a diferentes webs de noticias en un único punto.

Por último, tenemos las comunidades, donde son los propios usuarios los encargados de publicar contenido. Los puntos de vista varían mucho en función de que usuario publique información y es posible que no se encuentre toda la información actual en la comunidad si nadie la ha publicado. Como punto positivo tenemos que permiten seguir a gente de tu interés y establecer una red de contactos afines a tus gustos y poder debatir el contenido con ellos.

La propuesta que se pretende realizar engloba un poco de todas, se pretende tener acceso a multitud de fuentes de información de actualidad y como punto innovador y diferenciador que las noticias sean propuestas a cada usuario teniendo en cuenta sus gustos en función de las noticias previamente visitadas.

1.2.1 Soluciones existentes

Actualmente existen multitud de agregadores de noticias así como comunidades o redes sociales que pueden llegar a tener un efecto similar. Con el fin de crear un sistema capaz de mejorar las soluciones existentes se han analizado las aplicaciones principales.

Feedly

Feedly[1] es un lector de RSS que permite organizar y acceder rápidamente desde un navegador web o de sus aplicaciones para teléfonos inteligentes a todas las noticias y actualizaciones de blogs y demás páginas que el sistema soporta.

Entre otras características permite ordenar todos los contenidos de manera que facilita al usuario ahorrar tiempo por no tener que revisar una a una todas las fuentes de noticias.

Puntos fuertes:

- **Registro/Login:** Permite registrarse a través de diferentes medios como puede ser Google, Facebook, Twitter, Evernote y por último crear una cuenta directamente en su sitio web.

- **Aplicación y web:** Podemos acceder a Feedly tanto desde nuestro navegador web como a través de la aplicación para móviles.

- **Contenido on-site:** Se puede ver el contenido sin salir de la web de Feedly, además también si lo deseamos podemos ir directamente a la web donde originalmente se publicó la noticia.

Puntos débiles:

- **Interfaz:** Pese a que la interfaz es bastante limpia y bonita, a veces puede ser poco intuitiva afectando negativamente a la experiencia de usuario.

- **Funcionalidades de pago:** Las funcionalidades gratuitas son un poco básicas. La gran mayoría de desarrollos nuevos acaban en características de pago.

Reddit

Reddit[2] es un sitio web de marcadores sociales y agregador de noticias donde los usuarios pueden añadir texto, imágenes, vídeos o enlaces. Los usuarios pueden votar a favor o en contra del contenido, haciendo que aparezcan en las publicaciones destacadas.

Puntos fuertes:

- **Sencillez:** Es muy fácil de utilizar, basta con acceder a la página principal y empezar a mirar posts.

- **Subreddits:** El concepto es un poco diferente al de suscribirse mediante un agregador de noticias. La idea es que un subreddit es una pequeña comunidad de un tema específico donde los usuarios publicarán noticias relacionadas con este subreddit. Por ejemplo, tenemos el subreddit de programación:

Puntos débiles:

- **Contenido:** El contenido, tanto para lo bueno como para lo malo, lo publican los usuarios de la plataforma. Por tanto, es posible que haya contenido que no se publique debido a que nadie lo haya hecho aún. Aún así, gracias a la gran

comunidad de usuarios, normalmente prácticamente cualquier cosa suele estar publicada.

- **Filtrado complicado:** A veces puede llegar a ser complicado encontrar lo que queramos dado la gran cantidad de publicaciones que suceden al minuto, especialmente en los subreddits más poblados.

- **Anuncios:** Normalmente se publican anuncios de la misma manera que una publicación, lo cual puede dar lugar a confusión mientras nos desplazamos por la lista de publicaciones.

Twitter

Twitter[3] es un servicio de microblogueo. La red permite enviar mensajes de texto plano de corta longitud, con un máximo de 280 caracteres (originalmente 140), llamados tweets, que se muestran en la página principal del usuario. Los usuarios pueden suscribirse a los tweets de otros usuarios, "siguiendo" a dichos usuarios.

Puntos fuertes:

- **Seguimiento:** Tu eliges a quien seguir y por tanto qué tipo de contenido aparece en tu página principal. Además, es sencillo seguir a diferentes tipos de personas utilizando listas.

- **Tweets cortos:** Dado la limitación de caracteres, los mensajes deben ser cortos y por tanto resumir correctamente el contenido del mismo.

- **Cantidad de usuarios:** Twitter es utilizada por una gran cantidad de usuarios, por tanto es muy probable que podamos seguir a alguien de nuestro agrado que publique tweets que nos interesen.

Puntos débiles:

- **Desinformación:** Así como la cantidad de usuarios puede ser un punto a favor, también lo puede ser en contra, y es que existe una gran cantidad de noticias falsas publicadas por usuarios. Es responsabilidad del propio usuario seguir a quien considere una buena fuente de información.

- **Abrumación:** Cada segundo se publican una enorme cantidad de tweets. Si seguimos a mucha gente es posible que tengamos nuestra página principal inundada de nuevos tweets a cada segundo y por tanto perdamos información.

- **Puede ser adictiva:** Dado que el algoritmo de Twitter constantemente nos muestra noticias, trending topics y actividades para mantenernos en la web puede llegar a generar un poco de adicción.

1.2.2 Justificación de la elección

Todas las herramientas descritas en el apartado anterior tienen aspectos positivos por los que son una buena elección a dia de hoy por los usuarios. Sin embargo, ninguna de ellas basan sus contenidos en los gustos del usuario, de hecho, es el propio usuario quien debe generar sus propias listas, seguidores o subreddits, invirtiendo en ello un tiempo que podría dedicar a otras tareas.
Por ello, este proyecto pretende desarrollar un sistema que sea capaz de aprender de la persona que está utilizando el sistema y consiga saber cuales son sus gustos para mostrarle contenidos relevantes.

1.3 Alcance

El proyecto completo consiste en un sitio web donde cualquier usuario podrá inicialmente ver las noticias más relevantes del momento y si lo desea, podrá registrarse para tener una mejor experiencia. Una vez iniciada la sesión en la página, el usuario seguirá viendo noticias igual que antes pero a medida que vaya viendo más noticias, el sistema será capaz de aprender sus gustos y mostrarle contenidos relacionados con los vistos anteriormente.
Para ello, inicialmente se creará un sistema que se ejecutará cada día con el fin de obtener noticias de los medios más relevantes y poder almacenarlos en una base de datos para posteriormente poder disponer de un gran abanico de noticias y poder hacer algoritmos de recomendación para mejorar la experiencia de usuario.

En la figura 1 se muestra un esquema general del proyecto. Desde el backend se accedería a las APIs de noticias para almacenar estos contenidos en la BBDD para finalmente mostrar las noticias en la web.

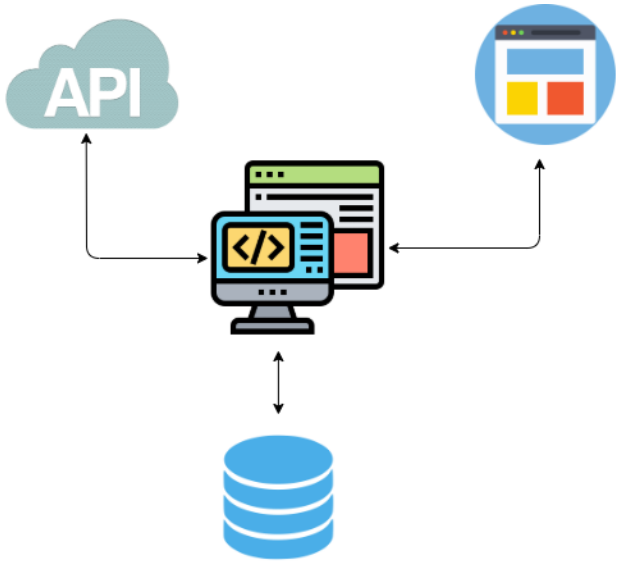


Figura 1: Situación global del proyecto.

1.3.1 Objetivos

El objetivo principal de este proyecto es crear un sitio web donde poder informarse de lo que ocurre en la actualidad, teniendo en cuenta los gustos del usuario sin necesidad de que el mismo los especifique. El algoritmo será capaz de aprender de él y mostrarle contenido adecuado a sus gustos. El fin es proporcionar al usuario una fuente de noticias fácil de utilizar y fiable. Para cumplir dicho objetivo, antes deben cumplirse diferentes puntos descritos a continuación:

- **Obtención de noticias a través de APIs**
  Mediante diferentes fuentes de APIs se obtendrán noticias de todo el mundo con la finalidad de almacenarlas en una base de datos y poder consultarlas posteriormente.

- **API REST de noticias**
  Se debe desarrollar una API con la finalidad de poder acceder al listado de noticias y poder consultarlas. Este servicio será el encargado de mostrar al usuario contenidos personalizados para él.

- **Aprendizaje en función de cada usuario**

Para el correcto funcionamiento del sistema, se debe obtener información del usuario y ser capaz de mostrar noticias relevantes para dicho usuario, en función de él mismo y de personas con gustos similares.

- **Frontend de visualización**
  Con el fin de que un usuario pueda acceder y obtener noticias, se creará un frontend donde se podrá iniciar sesión y obtener un feed de noticias de tipo infinito.

1.3.2 Requisitos no funcionales

- **Eficiencia:** El sistema debe funcionar de manera rápida, sin esperas considerables. Al menos, cualquier transacción debe responderse en menos de 5 segundos.

- **Usabilidad:** Debe ser fácil de utilizar e intuitiva. El proceso de aprendizaje debe ser corto.

- **Seguridad:** Los datos sobre gustos de noticias no se expondrán a vulnerabilidades de ningún tipo.

- **Escalabilidad**: Debe ser fácil mejorar el sistema, especialmente la parte backend.

- **Fiabilidad:** El sistema debe ser robusto y evitar tanto bugs como crasheos.

1.3.3 Obstáculos y riesgos

A continuación se detallan los obstáculos y riesgos que pueden surgir durante el desarrollo del proyecto:

- **Carencia APIs de noticias**: Riesgo potencial al empezar a descargar noticias mediantes las APIs seleccionadas de que no sean lo suficientemente completas o sean más complicadas de utilizar de lo previsto.

- **Desconocimiento de tecnologías utilizadas**: El uso de tecnologías y frameworks en los que se tiene poca experiencia supone un riesgo debido al tiempo de aprendizaje necesario.

- **Carga de trabajo elevada:** En caso de estimarse incorrectamente la carga de trabajo es posible que haya una carga superior a la prevista inicialmente.

1.4 Metodología y rigor

1.4.1 Metodología

Para llevar a cabo el proyecto, se utilizará una metodología Agile, en concreto **Scrum**[4]**.**
En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

En el comienzo de cada sprint se realiza el **sprint planning meeting**, que consiste en una reunión con todo el equipo del proyecto para acordar las tareas a realizar en el sprint. Al final de cada sprint, se realiza el **agile review meeting** y el **agile retrospective meeting**. En el primero se muestran todos los trabajos completados mientras que en el segundo cada miembro da su opinión e impresiones sobre las tareas realizadas, el trabajo en equipo y cosas a mejorar. El objetivo de dichas reuniones es conseguir un incremento en la productividad del equipo y por tanto mejorar el proceso.

Normalmente esta metodología divide el trabajo en diferentes roles que en este caso serán llevados a cabo por la misma persona adaptado a las circunstancias específicas del TFG: trabajo individual, requisitos temporales, de documentación y de calidad.

Gestión de proyectos: Jira

Jira[5] es una herramienta para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. Jira se basa en tableros que contienen listas. En dichas listas se definen las historias de usuario de manera que es fácilmente identificable el estado de cada tarea en todo momento. Para el proyecto se han definido las siguientes listas:

- To do: Tareas que están pendientes de hacer.
- In progress: Tareas que se están realizando en ese momento.
- In review: Tareas finalizadas pendientes de revisar.
- Done: Tareas finalizadas y revisadas.

Control de versiones: Git

Como software de control de versiones se utilizará **Git**[6], una herramienta de control de versiones que permite al desarrollador tener un control de todas las modificaciones realizadas en el código e incluso volver a puntos anteriores. Se trabajará utilizando ramas donde tendremos como rama principal **master** y como rama de desarrollo **develop** y a partir de esta, se crearán las ramas asociadas a cada tarea. Una vez realizada la tarea, se realizará un merge de la rama develop con la rama de la tarea en cuestión. La rama master se utilizará para despliegue de versiones.

Git nos permite subir el código a repositorios en la nube, de manera que podemos trabajar en el código desde cualquier lugar y además se puede pensar en ello como una especie de backup. Como repositorio de código donde alojar el código de manera remota se usará **GitHub**[7].

Project record track

Para llevar un registro de las horas dedicadas, se creará el documento **project record track** en Google Sheets. En dicho documento se rellenan las horas dedicadas diariamente a cada tarea.

1.4.2 Validación

Como validación se usará **TDD**[8] y se realizarán tanto tests unitarios como de integración sobre el código. Gracias a ello se podrá comprobar que el sistema funciona correctamente y además si se introduce algún cambio a código ya hecho, podemos volver a comprobar que todo sigue funcionando relanzando los tests de nuevo.

Para poder validar el número de horas requeridas para el proyecto, se utilizará el project record track definido en el apartado 4.1.3. Este documento permite saber cuantas horas se

han dedicado por dia y a que tarea, de manera que se puede saber con exactitud el total de horas dedicadas al proyecto.

Por último, cualquier decisión no planteada en la planificación inicial, deberá disponer de la aprobación del director del proyecto.

2. Planificación del proyecto

2.1 Planificación temporal

El proyecto comienza el día 23 de febrero y acaba a finales de junio (aproximadamente el día 20), ya que el turno de lectura es la última semana de junio o la primera de julio. Por tanto, entre la fecha de inicio y la fecha de finalización hay 117 días. Se espera una carga de trabajo media de unas 35 horas semanales, por tanto 5 horas diarias.

Se reservan las fechas comprendidas entre el 23 de febrero y el 22 de marzo para la gestión del proyecto. Dicha gestión incluye analizar el contexto, definir el alcance, planificar la duración temporal, análisis de riesgos y costes, y por último, informe de sostenibilidad.
Esta planificación durará 75 horas. Además, se necesitarán 5 horas más para la definición de historias de usuario y adición de las mismas a Jira.

A partir del 22 de marzo empezará el desarrollo, utilizando Scrum y acabará el 13 de junio, dejando una semana de margen para resolver posibles problemas e imprevistos. Entre estas dos fechas hay 12 semanas y por lo tanto, un total de 420 horas de trabajo.

Finalmente, del 14 de junio al 20 de junio, se espera una carga de trabajo de 35 horas dedicadas a documentación y últimos retoques.
En total, el proyecto ocupará unas 535 horas de trabajo.

2.2 Estimación del proyecto

El proyecto se divide en seis etapas. La etapa inicial hace referencia a la gestión del proyecto. Después de esta empiezan las etapas de desarrollo, en total 4 sprints con una duración de 3 semanas cada uno. Por último, un periodo de una semana dedicado a corregir errores, solventar imprevistos y finalizar la documentación.

Cada sprint tiene unas tareas asignadas a desarrollar y se complementa con la redacción de la documentación sobre dichas tareas.

Etapa	Fecha de inicio	Fecha de fin
Gestión del proyecto	23-02-2021	22-03-2021
Sprint 1	23-03-2021	11-04-2021
Sprint 2	12-04-2021	02-05-2021
Sprint 3	03-05-2021	23-05-2021
Sprint 4	24-05-2021	13-06-2021
Documentación final	14-06-2021	20-06-2021

Tabla 1: Planificación por etapas. (Fuente: elaboración propia)

2.3 Definición de tareas

A continuación se define una lista de tareas asociadas al proyecto. Por cada tarea, se muestra una pequeña descripción y una estimación en horas. A pesar de que la planificación es lo más precisa posible, hay que tener en cuenta que en el futuro pueden haber modificaciones en caso de imprevistos o necesidad de readaptación. Además, hay que tener en cuenta que para cada tarea de desarrollo, está incluido el tiempo de testeo.
La tabla 2 muestra un resumen de toda la información y la figura 1 muestra la planificación del proyecto.

2.3.1 T1: Gestión del proyecto

El objetivo de esta tarea es realizar la gestión inicial del proyecto que ayudará a tener una mejor organización en el futuro. Se calcula alrededor de 80 horas totales para esta tarea.

- **T1.1 Contexto y alcance:** Se define el contexto del proyecto y el alcance del sistema a desarrollar. La cantidad total de horas para esta tarea es de 20 horas.

- **T1.2 Planificación:** Se calcula una planificación del tiempo y de las horas a dedicar al proyecto desde el principio al final del mismo. Antes de poder hacer esta tarea, se debe haber acabado la tarea T1.1. Se dedicarán 20 horas a dicha tarea.

- **T1.3 Gestión económica y sostenibilidad:** Se realiza un presupuesto del proyecto y una parte inicial del informe de sostenibilidad. Es necesario haber acabado la tarea

T1.2 antes de empezar con esta. Se estiman 20 horas de dedicación para ello.

- **T1.4 Definición del proyecto:** Unificar los documentos realizados previamente y acabar de definir el proyecto. Las tareas T1.1, T1.2 y T1.3 deben estar finalizadas antes de poder empezar esta tarea. Se dedicarán 15 horas.

- **T1.5 Definir historias de usuario:** Se definen las historias de usuario y se añaden a Jira. Serán necesarias alrededor de 5 horas.

2.3.2 T2: Recopilación de noticias mediante APIs

Se requerirá hacer una investigación previa de las APIs disponibles y posteriormente implementar un sistema para obtener noticias y almacenarlas en una base de datos. En total, 65 horas.

- **T2.1 Investigación de APIs:** Se investigarán diferentes fuentes de noticias para obtener las mejores posibles. 25 horas de dedicación.

- **T2.2 Obtención de noticias:** Una vez investigado, se desarrollara un software para obtener las noticias. Antes de empezar esta tarea se habrá realizado la tarea T2.1 para tener un buen conocimiento de que APIs utilizar. Se calcula unas 25 horas para esta tarea.

- **T2.3 Almacenamiento de noticias:** Una vez se puedan conseguir noticias, se trabajará en poder almacenar estas noticias en la BBDD. La tarea T2.2 debe estar finalizada antes de poder continuar con esta ya que previamente el sistema debe ser capaz de descargar noticias para poder almacenarlas. 15 horas de dedicación para esta tarea.

2.3.3 T3: API Rest

Se implementará el backend que permitirá obtener las noticias previamente guardadas. Se trabajará con tokens para securización y desplegamiento en cloud. Se calculan alrededor de 105 horas totales.

- **T3.1 Endpoint obtener listado de noticias:** Se mostrarán listados de noticias vía JSON páginado. La tarea T2.3 debe haber finalizado antes de poder realizar esta tarea para poder mostrar noticias. Se estiman 35 horas para esta tarea.

- **T3.2 Endpoint información de una noticia:** Se mostrará una noticia en concreto vía JSON. Se debe haber acabado la tarea T2.3 para poder mostrar la información de una noticia. Alrededor de 25 horas de trabajo.

- **T3.3 Tokens:** Con el fin de securizar los datos, se implementará el uso de tokens. Se calculan 25 horas para dicha tarea.

- **T3.4 Desplegamiento en cloud:** Para poder usar la API desde cualquier sitio es necesario hacer un desplegamiento en la nube. Alrededor de 20 horas para esta tarea.

2.3.4 T4: Gestión de usuario

Se trabajará en registro, inicio de sesión y edición de perfil por la parte del backend. Se estiman unas 45 horas totales.

- **T4.1 Creación cuenta de usuario:** El usuario debe poder registrarse ya sea mediante un registro convencional o a través de Google Sign In. Se trabajará 15 horas en esta tarea.

- **T4.2 Iniciar sesión:** El usuario debe poder iniciar sesión ya sea mediante inicio de sesión típico o a través de su cuenta de Google. Es necesario haber finalizado la tarea T4.1 para poder iniciar sesión. Alrededor de 15 horas de trabajo.

- **T4.3 Editar perfil:** El usuario podrá modificar datos básicos de su perfil. Debe haber acabado la tarea T4.2 para poder editar el perfil de un usuario. Se estiman 15 horas de trabajo.

2.3.5 T5: Filtrado colaborativo

Se investigará y desarrollarán algoritmos de filtrado colaborativo para mostrar al usuario las noticias más interesantes para su perfil. Se trabajará 65 horas totales.

- **T5.1 Investigación de algoritmos de filtrado colaborativo:** Previamente se debe realizar una investigación para conocer cómo trabajan estos algoritmos. Se necesitarán 30 horas.

- **T5.2 Implementación algoritmo de filtrado colaborativo:** Una vez obtenido el conocimiento, se desarrollará la solución para recomendar noticias al usuario. Antes de empezar con esta tarea se debe haber realizado la tarea T.51 para tener un buen conocimiento sobre los algoritmos a implementar. Se calculan unas 35 horas de trabajo.

2.3.6 T6: Frontend

Con el fin de que el sistema sea usable, se diseñará e implementará un frontend para que el usuario pueda usar el sistema. Se estiman 75 horas totales.

- **T6.1 Diseño feed de noticias:** Se diseñará un feed de noticias de tipo infinito para poder ver el conjunto de noticias cómodamente. Se trabajará 30 horas.

- **T6.2 Diseño información de una noticia:** Se diseñará una página para poder ver una noticia en cuestión con toda su información. Se calculan 25 horas de trabajo.

- **T6.3 Diseño registro/inicio de sesión:** Se diseñará una página donde el usuario podrá registrarse o iniciar sesión. Alrededor de 20 horas para esta tarea.

2.3.7 T7: Documentación y seguimiento

Además del desarrollo del proyecto, es necesario proporcionar una documentación de calidad y hacer un seguimiento tanto vía mail como por meet con el director del proyecto. Para dicha tarea, se calculan unas 100 horas.

- **T7.1 Documentación:** Tiempo dedicado a la elaboración de toda la documentación necesaria del proyecto. Se calculan unas 80 horas.

- **T7.2 Meetings y seguimiento:** Reuniones con el director del proyecto para hacer un buen seguimiento del proyecto. Alrededor de 20 horas para esta tarea.

2.4 Recursos

A continuación se detallan los recursos necesarios para el proyecto. Se ha dividido en tres grupos: recursos humanos, recursos de hardware y recursos de software.

2.4.1 Recursos humanos

A continuación se define el personal que desarrollará el proyecto. Primero, el investigador **[I]**, responsable del desarrollo del proyecto. Será el encargado de investigar y desarrollar el proyecto. Por otra parte, tenemos al director del proyecto **[T]**, que se encargará de guiar al investigador con el fin de obtener un resultado exitoso del proyecto. Por último, tenemos al tutor de GEP **[TGEP]**, que se encargará de ayudar al investigador para hacer una buena gestión del proyecto durante el primer mes.

2.4.2 Recursos de hardware

El recurso más esencial es el ordenador. Para este proyecto se utilizará un ordenador con las siguientes características:

Ordenador de sobremesa: 16GB RAM, Intel® Core™ i5-4690 CPU @3.50GHz

Además, se debe tener en cuenta todos los recursos necesarios para una conexión a internet, como por ejemplo el router.

2.4.3 Recursos de software

Para poder llevar a cabo el desarrollo del proyecto serán necesarios varios software como:

- **Jira:** para la gestión de las tareas a desarrollar.
- **Git:** para almacenar el código desarrollado en un repositorio.
- **Google Docs**: Para generar la documentación.
- **Drive**: Para almacenar la documentación en la nube.
- **IntelliJ**: Para desarrollar la parte backend.
- **Visual Studio Code**: Para desarrollar la parte frontend.
- **Heroku**: Como herramienta donde desplegar tanto el backend como el frontend.

- **Ganttproject:** Para realizar el Gantt.
- **Gmail**: Para contactar con el director del proyecto y el tutor de GEP.
- **Meet**: En caso de realizar una reunión con el director del proyecto, se haría a través de Google Meet.

2.5 Gestión de riesgos

Previamente ya se han descrito los potenciales riesgos y obstáculos que pueden aparecer en el proyecto. En esta sección se comentará cómo solucionarlos, además de determinar el nivel de riesgo de cada uno de ellos.

Riesgo	Probabilidad	Horas
Carencia APIs de noticias	Bajo	10
Desconocimiento de tecnologías utilizadas	Medio	25
Carga de trabajo elevada	Medio	20

Tabla 2: Riesgos y estimaciones. (Fuente: elaboración propia)

Carencia APIs de noticias

Existe un riesgo no demasiado elevado de que las APIs de noticias disponibles no sean ideales. Probablemente muchas de ellas serán de pago o tendrán limitaciones importantes que obligarán a hacer un poco más de investigación con el fin de obtener fuentes de noticias de calidad.

En caso de no encontrar APIs de noticias con una calidad aceptable, se deberá añadir más tiempo de investigación en encontrar APIs (T2.1) mejores, adaptarse a las que tengan una calidad aceptable o investigar el formato RSS que es bastante popular en este ámbito. En caso de necesitar más tiempo, se necesitaría alrededor de 10 horas.

Desconocimiento de tecnologías utilizadas

Se pretende utilizar tecnologías en las que el desarrollador tiene poca experiencia o ninguna en la parte de frontend. Debido a esto, es posible que el desarrollo se ralentice o se necesite mejorar el aprendizaje en esta tecnología.

En caso de no ser capaz de desarrollar una solución con la calidad suficiente utilizando alguna tecnología en la que el desarrollador tenga poca experiencia, se añadirá una nueva

tarea con el objetivo de aprender la tecnología con un poco más de detalle. Esta tarea debería tener una duración de 25 horas aproximadamente.

Carga de trabajo elevada

Debido a la magnitud de un proyecto como el TFG, es posible que la carga de trabajo del proyecto sea elevada y sobretodo ocurriría por realizar una estimación errónea de las tareas al tener poca experiencia gestionando un proyecto.

En caso de ocurrir, será necesario hacer una planificación para recalcular el tiempo de las tareas. En caso de que no sea posible por estar muy cerca de la fecha de entrega, se puede solucionar añadiendo más tiempo de desarrollo por día. En este último caso, se calculan alrededor de 20 horas en caso de suceder.

ID	Nombre	Tiempo (h)	Dependencias	Recursos
T1	**Gestión del proyecto**	**80**		
T1.1	Contexto y alcance	20		PC, Drive, Google Docs, TGEP, I
T1.2	Planificación	20	T1.1	PC, Drive, Google Docs, TGEP, I
T1.3	Gestión económica y sostenibilidad	20	T1.2	PC, Drive, Google Docs, TGEP, I
T1.4	Definición del proyecto	15	T1.1, T1.2, T1.3	PC, Drive, Google Docs, TGEP, I
T1.5	Definir historias de usuario	5		PC, Jira, I
	Desarrollo	**355**		
T2	**Recopilación de noticias mediante APIs**	**65**		
T2.1	Investigación de APIs	25		PC, I
T2.2	Obtención de noticias	25	T2.1	PC, IntelliJ, Git, I
T2.3	Almacenamiento de noticias	15	T2.2	PC, IntelliJ, Git, I
T3	**API Rest**	**105**		
T3.1	Endpoint obtener listado de noticias	35	T2.3	PC, IntelliJ, Git, I
T3.2	Endpoint información de una noticia	25	T2.3	PC, IntelliJ, Git, I
T3.3	Tokens	25		PC, IntelliJ, Git, I
T3.4	Desplegamiento en cloud	20		PC, IntelliJ, Git, Heroku, I
T4	**Gestión de usuario**	**45**		
T4.1	Creación cuenta de usuario	15		PC, IntelliJ, Git, I
T4.2	Iniciar sesión	15	T4.1	PC, IntelliJ, Git, I
T4.3	Editar perfil	15	T4.2	PC, IntelliJ, Git, I
T5	**Aprendizaje del usuario (ML)**	**65**		
T5.1	Investigación de algoritmos inteligentes ML	30		PC, I
T5.2	Implementación algoritmo de recomendación con ML	35	T5.1	PC, IntelliJ, Git, I
T6	**Frontend**	**75**		
T6.1	Diseño feed de noticias	30		PC, VS Code, Git, I
T6.2	Diseño información de una noticia	25		PC, VS Code, Git, I
T6.3	Diseño registro/inicio de sesión	20		PC, VS Code, Git, I
T7	**Documentación y seguimiento**	**100**		
T7.1	Documentación	80		PC, Drive, Google Docs, Jira, I
T7.2	Meetings y seguimiento	20		PC, Meet, Gmail, T, I
Total		**535**		

Tabla 3: Resumen de tareas. (Fuente: elaboración propia)

Diagrama de Gantt

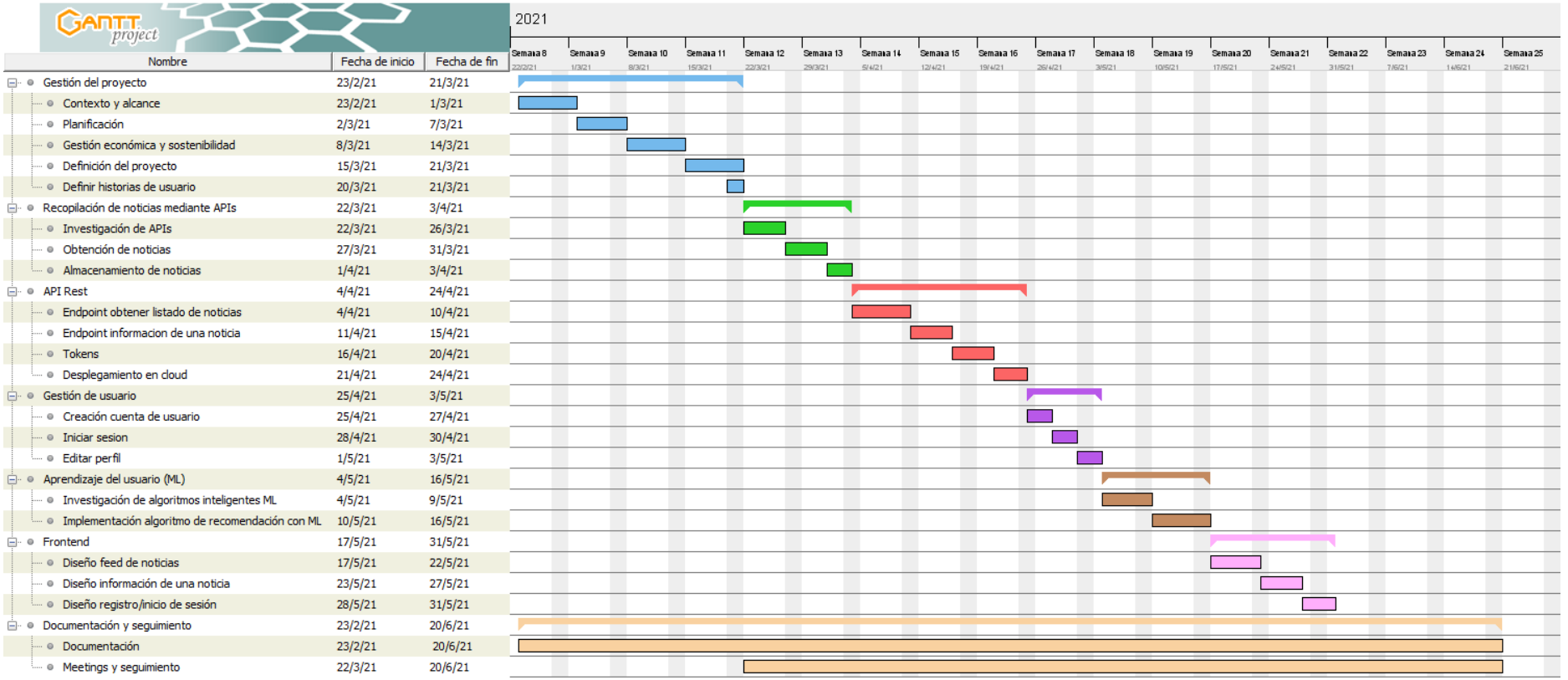


Figura 2: Diagrama de Gantt. (Fuente: elaboración propia)

2.6 Retrospectiva sobre la planificación inicial

2.6.1 Cambios de planificación

Desde el inicio del proyecto no ha habido ningún cambio de planificación considerable. Solo ha sido necesario utilizar 15 horas para realizar un aprendizaje básico del framework de Vue para poder empezar a crear la página web del sistema.

Esta nueva tarea ya se había planificado como un posible riesgo y se estimó alrededor de 25 horas, por tanto, habiendo utilizado solamente 15 horas, aún habría más margen en caso de necesitar aprender alguna peculiaridad específica del framework.

A pesar de esta nueva tarea, el desarrollo no se ha ralentizado, ya que ha sido posible avanzar más rápido en el resto de tareas y actualmente el proyecto se encuentra en una fase muy avanzada.

Sobre las tareas, se ha modificado la parte de gestión de usuarios. Inicialmente el objetivo era realizar la gestión internamente, realizando un registro de usuarios y almacenarlos en los servidores del sistema. Finalmente, consultando con el tutor, se decidió utilizar OAuth2 con Google para la gestión de usuarios. Esto ha permitido simplificar mucho el desarrollo y por tanto poder avanzar más rápido, además de aprender cómo funciona OAuth2 en detalle.

2.6.2 Conclusiones

El proyecto ha cumplido con la planificación estimada inicialmente y en parte ha sido gracias al uso de la metodología Agile. En algunas tareas se ha podido avanzar más rápido de lo esperado mientras que en otras se ha necesitado un poco más de tiempo, pero al final ha quedado compensado.

En general, creo que se ha estimado bastante bien todas las tareas y se pudo acabar un poco antes de lo previsto al realizar más trabajo por día de lo planificado inicialmente. Por tanto, de haber seguido completamente el número de horas diarias a trabajar, se hubiese acabado en una fecha muy cercana a la estimada.

3. Presupuesto y sostenibilidad

3.1 Presupuesto

3.1.1 Gastos de personal por actividad

En esta sección se calculará el coste total de cada tarea definida previamente. El coste de una tarea será calculado sumando el coste del personal. El coste de cada trabajador se calcula multiplicando su coste por hora por la cantidad de horas trabajadas en esa tarea.

En este proyecto hay cinco roles y cada uno tiene un coste diferente por hora. A continuación se muestra una tabla resumen por cada rol y el coste que representa.

Rol	Salario bruto anual (€)	Salario bruto con SS (€)	Precio por hora (€)
Gestor del proyecto	37.652	48947,60	28,46
Analista	28.114	36548,20	21,25
Arquitecto de Software	47.081	61205,30	35,58
Programador	27.890	36257,00	21,08
Tester	27.583	35857,90	20,85

Tabla 4: Salario anual por roles. Información obtenida de Glassdoor[9]. El precio por hora ha sido calculado en base al número de horas de trabajo de 2021, publicado en el BOE[10], que es de 1720. (Elaboración propia)

Teniendo en cuenta el precio por hora de cada rol, podemos calcular el coste del personal por actividad en función de las horas que cada rol dedica a cada tarea. A continuación se muestra dicha información en la tabla 5.

ID	Nombre	Tiempo (h)	Horas por rol					Coste (€)
			Gestor del proyecto	Analista	Arquitecto de software	Programador	Tester	
T1	**Gestión del proyecto**	**80**	**80**	**0**	**0**	**0**	**0**	**2.276,63**
T1.1	Contexto y alcance	20	20	0	0	0	0	569,16
T1.2	Planificación	20	20	0	0	0	0	569,16
T1.3	Gestión económica y sostenibilidad	20	20	0	0	0	0	569,16
T1.4	Definición del proyecto	15	15	0	0	0	0	426,87
T1.5	Definir historias de usuario	5	5	0	0	0	0	142,29
	Desarrollo	**355**	**0**	**35,5**	**35,5**	**248,5**	**35,5**	**7.995,97**
T2	**Recopilación de noticias mediante APIs**	**65**	0	6,5	6,5	45,5	6,5	**1.464,05**
T2.1	Investigación de APIs	25	0	2,5	2,5	17,5	2,5	563,10
T2.2	Obtención de noticias	25	0	2,5	2,5	17,5	2,5	563,10
T2.3	Almacenamiento de noticias	15	0	1,5	1,5	10,5	1,5	337,86
T3	**API Rest**	**105**	**0**	**10,5**	**10,5**	**73,5**	**10,5**	**2.365,01**
T3.1	Endpoint obtener listado de noticias	35	0	3,5	3,5	24,5	3,5	788,34
T3.2	Endpoint información de una noticia	25	0	2,5	2,5	17,5	2,5	563,10
T3.3	Tokens	25	0	2,5	2,5	17,5	2,5	563,10
T3.4	Desplegamiento en cloud	20	0	2	2	14	2	450,48
T4	**Gestión de usuario**	**45**	**0**	**4,5**	**4,5**	**31,5**	**4,5**	**1.013,57**
T4.1	Creación cuenta de usuario	15	0	1,5	1,5	10,5	1,5	337,86
T4.2	Iniciar sesión	15	0	1,5	1,5	10,5	1,5	337,86
T4.3	Editar perfil	15	0	1,5	1,5	10,5	1,5	337,86
T5	**Aprendizaje del usuario (ML)**	**65**	**0**	**6,5**	**6,5**	**45,5**	**6,5**	**1.464,05**
T5.1	Investigación de algoritmos inteligentes ML	30	0	3	3	21	3	675,72
T5.2	Implementación algoritmo de recomendación con ML	35	0	3,5	3,5	24,5	3,5	788,34
T6	**Frontend**	**75**	**0**	**7,5**	**7,5**	**52,5**	**7,5**	**1.689,29**
T6.1	Diseño feed de noticias	30	0	3	3	21	3	675,72
T6.2	Diseño información de una noticia	25	0	2,5	2,5	17,5	2,5	563,10
T6.3	Diseño registro/inicio de sesión	20	0	2	2	14	2	450,48
T7	**Documentación y seguimiento**	**100**	**100**	**0**	**0**	**0**	**0**	**2.845,79**
T7.1	Documentación	80	80	0	0	0	0	2.276,63
T7.2	Meetings y seguimiento	20	20	0	0	0	0	569,16
Total		**535**	**180**	**35,5**	**35,5**	**248,5**	**35,5**	**13.118,39**

Tabla 5: Coste de personal por tareas. (Fuente: elaboración propia)

3.1.2 Costes genéricos

Hardware

Hay que tener en cuenta los recursos materiales utilizados en el proyecto. En este caso, solo se utiliza un ordenador de sobremesa y se considera un uso medio de 5 horas diarias durante los 117 días que dura el proyecto.
Para calcular el coste de dicho recurso, se utiliza la siguiente fórmula:

Amortización (€) = Precio del recurso x $\frac{1}{4\ años}$ x $\frac{1}{117\ días}$ x $\frac{1}{5\ horas}$ x horas utilizado.

Teniendo en cuenta que el ordenador de sobremesa cuesta alrededor de 1500€ y el número de horas que se usará es de 535, la amortización de este recurso es de 342,94€

Recurso hardware	Precio (€)	Tiempo usado (h)	Amortización (€)
Ordenador sobremesa	1.500	535	342,95

Tabla 6: Recurso material y su amortización. (Fuente: elaboración propia)

Coworking

También es necesario un espacio de trabajo, así como electricidad, internet, agua, etc. Para facilitar este cálculo se ha decidido alquilar un espacio de coworking en Barcelona. La información de precios ha sido obtenida de MeetBCN[11].

Recurso	Precio (€)	Tiempo usado (meses)	Amortización (€)
Coworking full time	292	5	1.460,00

Tabla 7: Coste de coworking. (Fuente: elaboración propia)

3.1.3 Contingencias

Durante el proyecto pueden aparecer imprevistos no planificados, por ello se reserva un importe equivalente al 15% del coste del proyecto (gastos de personal por actividad + costes genéricos = 14921,34 €). Finalmente, el coste de contingencias es de 2238,20 €.

Tipo	Precio (€)	Contingencias (%)	Total (€)
Gastos de personal	13.118,39	15	1.967,76
Ordenador sobremesa	342,95	15	51,44
Coworking full time	1.460,00	15	219,00

Tabla 8: Costes de contingencias. (Fuente: elaboración propia)

3.1.4 Imprevistos

Previamente se ha comentado los posibles riesgos que podemos encontrarnos en el proyecto y además se ha decidido que acciones tomar en caso de que ocurran en el apartado de gestión de riesgos. En la siguiente tabla se muestra cada riesgo con su probabilidad de ocurrir y el tiempo que se estimó necesario para gestionarlo. El coste estimado ha sido calculado con el precio por hora del programador multiplicado por el tiempo estimado del imprevisto. Con toda esta información, obtenemos un coste detallado por cada riesgo.

Imprevisto	Riesgo (%)	Coste estimado (€)	Coste (€)
Carencia APIs de noticias (10h)	25	210,80	52,70
Desconocimiento de tecnologías utilizadas (25h)	50	526,99	263,50
Carga de trabajo elevada (20h)	50	421,59	210,80

Tabla 9: Coste de imprevistos. (Fuente: elaboración propia)

3.1.5 Coste total

A continuación se muestra la tabla 10 indicando el coste por actividad y el coste total del proyecto.

Actividad	Coste (€)
Gastos de personal	13.118,39
Gastos genéricos	1.802,95
Contingencias	2.238,20
Imprevistos	526,99
Total	**17.686,54**

Tabla 10: Coste total del proyecto. (Fuente: elaboración propia)

3.2 Control de gestión

Con el objetivo de llevar un control de las desviaciones sobre el presupuesto, cuando una tarea acabe, se actualizará el presupuesto en función de las horas trabajadas y las horas previstas. Para ello, se utilizarán un conjunto de fórmulas explicadas a continuación en las que se puede calcular las desviaciones.

- **Desviación de horas por tarea**

  (Horas estimadas - horas reales) * Coste real

- **Desviación costes de recursos humanos por tarea**

  (Coste estimado - coste real) * Horas reales

- **Desviación total costes genéricos**

  Coste genérico estimado - coste genérico real

- **Desviación total costes imprevistos**

  Costes imprevistos estimado - coste imprevistos real

- **Desviación total de horas**

  Horas totales estimadas - horas totales reales

- **Desviación total de costes**

  Coste total estimado - coste total real

Con esta serie de indicadores podremos saber cuándo ha ocurrido una desviación y de cuanto es el coste de dicha desviación.

3.3 Retrospectiva sobre el coste inicial

A continuación se muestran las tablas donde se puede comprobar la desviación tanto en horas como en coste siguiendo los indicadores mencionados en el apartado anterior:

ID	Horas estimadas (h)	Coste estimado (€)	Horas reales (h)	Coste real (€)	Desviación (h)	Desviación (€)
T1	**80**	**2.276,63**	**80**	2.276,63	0	0,00
T1.1	20	569,16	20	569,16	0	0,00
T1.2	20	569,16	20	569,16	0	0,00
T1.3	20	569,16	20	597,62	0	28,46
T1.4	15	426,87	15	398,41	0	-28,46
T1.5	5	142,29	5	142,29	0	0,00
	355	**7.995,97**	**361**	8.131,11	6	135,14
T2	**65**	**1.464,05**	**76**	1.711,81	11	247,76
T2.1	25	563,10	27	608,14	2	45,04
T2.2	25	563,10	29	653,19	4	90,09
T2.3	15	337,86	20	450,48	5	112,62
T3	**105**	**2.365,01**	**116**	2.612,77	11	247,76
T3.1	35	788,34	33	743,29	-2	-45,05
T3.2	25	563,10	26	585,62	1	22,52
T3.3	25	563,10	31	698,24	6	135,14
T3.4	20	450,48	26	585,62	6	135,14
T4	**45**	**1.013,57**	**30**	675,72	-15	-337,85
T4.1	15	337,86	0	0	-15	-337,86
T4.2	15	337,86	30	675,72	15	337,86
T4.3	15	337,86	0	0	-15	-337,86
T5	**65**	**1.464,05**	**61**	1.373,96	-4	-90,09
T5.1	30	675,72	25	563,1	-5	-112,62
T5.2	35	788,34	36	810,86	1	22,52
T6	**75**	**1.689,29**	**78**	1.756,86	3	67,57
T6.1	30	675,72	28	630,67	-2	-45,05
T6.2	25	563,10	24	540,57	-1	-22,53
T6.3	20	450,48	26	585,62	6	135,14
T7	**100**	**2.845,79**	**100**	2.845,79	0	0,00
T7.1	80	2.276,63	80	2.276,63	0	0,00
T7.2	20	569,16	20	569,16	0	0,00
Total	**535**	**13.118,39**	**541**	**13.253,54**	**6**	**135,15**

Tabla 11: Desviación en horas y coste por tareas. (Fuente: elaboración propia)

Total CG Estimado (€)	Total CG Real(€)	Desviación (€)
1.802,95	1.803	0

Tabla 12: Desviación en coste de los costes genéricos. (Fuente: elaboración propia)

Imprevisto	Total Imprevistos Estimados (€)	Total Imprevistos Real (€)	Desviación (€)
Carencia APIs de noticias	52,7	0	-52,7
Desconocimiento de tecnologías utilizadas	263,5	47,43	-216,07
Carga de trabajo elevada	210,8	0	-210,8

Tabla 13: Desviación en coste de los imprevistos. (Fuente: elaboración propia)

3.3.1 Conclusiones

En general se ha cumplido con la planificación de manera muy positiva. En la tabla 11 se puede comprobar que han sido necesarias 6 horas más y por tanto se ha incrementado el coste del proyecto en 135,15€. Por otra parte, en los imprevistos solo ha sido necesario utilizar unas pocas horas del imprevisto “Desconocimiento de tecnologías utilizadas”, en concreto 10 horas y por tanto se sumarían 47,43€ al coste total del proyecto pero también se descontaran los otros dos imprevistos ya que no han sido necesarios.

Finalmente, se muestra la tabla comparativa con el coste estimado del proyecto y el coste real. Se puede observar como el coste total del proyecto es ligeramente inferior al coste estimado, en concreto 344,42€.

Actividad	Coste estimado (€)	Coste (€)	Desviación (€)
Gastos de personal	13.118,39	13.253,54	135,15
Gastos genéricos	1.802,95	1.802,95	0,00
Contingencias	2.238,20	2.238,20	0,00
Imprevistos	526,99	47,43	-479,56
Total	**17.686,54**	**17.342,12**	**-344,42**

Tabla 14: Desviación total del proyecto. (Fuente: elaboración propia)

3.4 Sostenibilidad

Una vez realizada la encuesta de EDINSOST, he podido reflexionar sobre los diferentes aspectos relacionados con la sostenibilidad y las conclusiones que extraigo de ellas son que conozco las dimensiones sociales y económicas pero hasta ahora no comprendía del todo la dimensión medioambiental.

Comprendo que los programas que creamos los informáticos necesitan de cierta energía y recursos para ejecutarse pero no tengo del todo claro cómo analizar el impacto de esta dimensión en la sociedad.

Respecto a la dimensión económica sí que entiendo cómo analizarla ya sea mediante estudios de mercado para valorar la viabilidad de un producto o haciendo cálculos de presupuestos para tener una idea del coste general.

Sobre la dimensión social, puede ser un poco compleja de analizar, pero en general entiendo que el software debe ser ético, transparente y accesible, con el fin de ayudar a la gente con la tarea que necesiten.

3.4.1 Dimensión ambiental

¿Has estimado el impacto ambiental que tendrá la realización del proyecto?
En principio no hay un impacto ambiental demasiado grande ya que el proyecto a realizar no generará ningún producto físico, si no simplemente un software. El único impacto ambiental que puede haber es el de los recursos utilizados, como son el ordenador y la electricidad consumida.

¿Te has planteado minimizar el impacto, por ejemplo, reutilizando recursos?
Si, a pesar de que no lo he mencionado explícitamente, el ordenador ya tiene varios años de uso (alrededor de 6 años) y por tanto se ha evitado comprar uno nuevo para el proyecto.

¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)?
Actualmente se utilizan varias opciones: periódicos online, agregadores de noticias, redes sociales como Twitter, o comunidades/foros como Reddit. Además, también se pueden considerar periódicos tradicionales impresos en papel.

¿En qué mejorará ambientalmente tu solución a las existentes?

La solución propuesta permite reducir el uso de papel respecto a los periódicos tradicionales. El resto de soluciones ya existentes comentadas en la pregunta anterior también permiten la misma mejora ambiental.

Si hicieras de nuevo el proyecto, ¿podrías realizarlo con menos recursos?

Creo que los recursos utilizados han sido los justos y necesarios, realizar el proyecto con menos recursos, podría llegar a ser posible pero sinceramente lo veo complicado y quizás podría ralentizar el desarrollo del mismo.

3.4.2 Dimensión económica

¿Has estimado el coste de realización del proyecto (recursos humanos y materiales)?

Si, se ha realizado un presupuesto con toda la información necesaria como son costes de recursos humanos, costes genéricos, contingencias e imprevistos.

¿En que mejorará económicamente tu solución a las existentes?

Mejorará en un menor tiempo de utilización respecto a las soluciones existentes, tiempo que el usuario podrá utilizar en diferentes tareas y que antes lo usaría en las soluciones actuales.

¿Se ha ajustado el coste previsto al coste final?

Si, el coste final ha sido muy similar al coste previsto, de hecho ha sido un poco inferior debido principalmente a la reestructuración de una tarea que ha permitido implementarla de manera más sencilla a la par que moderna y por tanto se han podido reducir horas. Aún así, en algunas otras tareas, en general las horas han sido superiores a las previstas. Principalmente, la diferencia viene en que apenas se han producido imprevistos que pudiesen encarecer el proyecto.

3.4.3 Dimensión social

¿Qué crees que te va a aportar a nivel personal la realización de este proyecto?

Por un lado podré comprobar que soy capaz de llevar a cabo un proyecto formal desde el principio a fin con los imprevistos que pueda haber entremedio. Además, aprenderé sobre nuevas tecnologías muy usadas a día de hoy en el mercado y mejoraré mis conocimientos sobre tecnologías que ya conocía previamente.

¿Existe una necesidad real del proyecto?

Bajo mi punto de vista, si. Cualquier persona que utilice este sistema estará ahorrando tiempo respecto al resto de soluciones ya existentes. Hoy en día, poder mejorar nuestra gestión del tiempo, teniendo en cuenta la cantidad de cosas que una persona hace durante el día, es esencial.

¿En qué medida soluciona el proyecto el problema planteado inicialmente?

En general considero que es una buena solución y un buen punto de partida. El proyecto cumple las expectativas iniciales pero aún así es mejorable por ejemplo implementando algoritmos de recomendación híbridos que mejoran el sistema de recomendaciones.

4. Análisis de requisitos

4.1 Requisitos funcionales

A continuación se definen los requisitos funcionales en forma de historias de usuarios. De esta manera podemos definir correctamente una funcionalidad y saber cuando la misma está acabada.

#1 Ver feed de noticias
Historia de usuario: Como usuario del sistema quiero poder ver un listado de noticias para poder visualizar las noticias existentes.
Criterios de aceptación: ● Cada noticia debe tener título, descripción, enlace, imagen y fecha de publicación. ● A medida que el usuario haga scroll, se deben ir cargando nuevas noticias. ● Al pulsar en el título o en el botón de Leer más, se debe enviar al usuario a una página donde podrá ver la noticia al completo. ● Si la descripción de la noticia es muy larga, se debe cortar para no ocupar demasiada pantalla. ● Las noticias deben estar ordenadas por fecha de publicación descendente.

Tabla 15: Historia de usuario: ver feed de noticias

#2 Ver noticias puntuadas por un usuario
Historia de usuario: Como usuario registrado quiero poder ver las noticias que he puntuado para poder visualizarlas de nuevo.
Criterios de aceptación: ● Cada noticia debe tener título, descripción, enlace, imagen y fecha de publicación. ● A medida que el usuario haga scroll, se deben ir cargando nuevas noticias. ● Al pulsar en el título o en el botón de Leer más, se debe enviar al usuario a una página donde podrá ver la noticia al completo. ● Si la descripción de la noticia es muy larga, se debe cortar para no ocupar demasiada pantalla. ● Las noticias deben estar ordenadas por fecha de publicación descendente.

● Solo pueden aparecer noticias puntuadas previamente por el usuario. ● No pueden aparecer noticias puntuadas por otros usuarios que el usuario no haya puntuado.

Tabla 16: Historia de usuario: Ver noticias puntuadas por un usuario

#3 Ver feed de noticias recomendadas
Historia de usuario: Como usuario registrado quiero poder ver mis noticias recomendadas para poder visualizar noticias afín a mis gustos.
Criterios de aceptación: ● Cada noticia debe tener título, descripción, enlace, imagen y fecha de publicación. ● A medida que el usuario haga scroll, se deben ir cargando nuevas noticias. ● Al pulsar en el título o en el botón de Leer más, se debe enviar al usuario a una página donde podrá ver la noticia al completo. ● Si la descripción de la noticia es muy larga, se debe cortar para no ocupar demasiada pantalla. ● Las noticias deben estar ordenadas por fecha de publicación descendente. ● El usuario debe haber puntuado algunas noticias antes de poder ver noticias recomendadas para él. ● Solo pueden aparecer noticias que tengan un grado de recomendación superior a 2.5. El mínimo grado será 0.

Tabla 17: Historia de usuario: Ver feed de noticias recomendadas

#4 Ver noticia completa
Historia de usuario: Como usuario del sistema quiero poder ver una noticia completa para poder leer la noticia y votarla si así lo deseo.
Criterios de aceptación: ● La noticia debe tener título, fecha de publicación, autor, fuente, imagen (opcional, se asignará imagen por defecto en caso de no existir), descripción y valoración. En caso de que el usuario haya valorado la noticia previamente, debe aparecer su

valoración.

Tabla 18: Historia de usuario: Ver noticia completa

#5 Puntuar una noticia
Historia de usuario: Como usuario registrado quiero poder puntuar una noticia para posteriormente obtener recomendaciones
Criterios de aceptación: ● Se debe poder valorar una noticia del 1 al 5. ● Una vez votada, debe aparecer un modal como confirmación donde se indicará la puntuación realizada.

Tabla 19: Historia de usuario: Puntuar una noticia

#6 Iniciar sesión con Google
Historia de usuario: Como usuario del sistema quiero poder iniciar sesión con Google para poder ver mis noticias puntuadas y recomendadas y también para poder puntuar nuevas noticias.
Criterios de aceptación: ● Debe existir un botón con apariencia estilo Google para iniciar sesión. ● Al pulsar dicho botón, el usuario podrá elegir con que cuenta de Google desea iniciar sesión. ● Una vez iniciada la sesión, deben aparecer los botones de noticias puntuadas y recomendadas.

Tabla 20: Historia de usuario: Iniciar sesión con Google

#7 Cerrar sesión
Historia de usuario: Como usuario registrado quiero poder cerrar mi sesión para no dejar la sesión abierta.

Criterios de aceptación:
● Debe existir un botón con apariencia estilo Google para cerrar sesión. ● Al pulsar dicho botón, el usuario cerrará su sesión y será redirigido a la pantalla de últimas noticias. ● Una vez cerrada la sesión, deben desaparecer los botones de noticias puntuadas y recomendadas.

Tabla 21: Historia de usuario: Cerrar sesión

4.2 Requisitos no funcionales (Volere)

A continuación se describen los requisitos no funcionales del sistema que indican cómo ha de comportarse el sistema. Para la descripción de estos requisitos se utilizan las plantillas de Volere [12]

#10b. Requisitos de estilo
Descripción: El sitio web debe tener una apariencia agradable y sencilla.
Justificación: Las interfaces amigables facilitan el uso por parte del usuario y además incitan a ser usadas por el mismo.

Tabla 22: Requisito no funcional: #10b. Requisitos de estilo

#11a. Requisitos de facilidad de uso
Descripción: El sitio web debe ser fácil de utilizar, fácil de recordar y en caso de error, mostrar una buena descripción.
Justificación: Con el objetivo de que los usuarios quieran utilizar el producto, es necesario que el mismo sea fácil de utilizar.

Tabla 23: Requisito no funcional: #11a. Requisitos de facilidad de uso

#12a. Requisitos de velocidad y latencia
Descripción: El sitio web debe proporcionar noticias de manera rápida a medida que el usuario va cargando nuevas noticias.
Justificación: Una carga lenta implicaría una experiencia de usuario negativa y a largo plazo sería un motivo por el que un usuario decida no volver a utilizar el producto.

Tabla 24: Requisito no funcional: #12a. Requisitos de velocidad y latencia

#12e. Requisitos de robustez y tolerancia al fallo
Descripción: El sitio web no puede tener fallos críticos y debe ser tolerante al fallo del usuario mostrando descripciones fáciles de entender.
Justificación: Con el objetivo de que el usuario sepa que está pasando en todo momento, se debe mostrar descripciones detalladas.

Tabla 25: Requisito no funcional: #12e. Requisitos de robustez y tolerancia al fallo

#14c. Requisitos de adaptabilidad
Descripción: El sitio web debe poder verse correctamente en cualquier navegador y en cualquier dispositivo, es decir, debe ser responsive.
Justificación: Con el objetivo de atraer al mayor número de usuarios, es necesario que cualquier usuario pueda visualizarlo correctamente en cualquier dispositivo.

Tabla 26: Requisito no funcional: #14c. Requisitos de adaptabilidad

#15a. Requisitos de acceso
Descripción: Algunas funcionalidades del sitio web solo podrán ser accedidas una vez el usuario haya iniciado sesión.

Justificación: Con el objetivo de securizar el acceso a las secciones correspondientes y para poder obtener noticias recomendadas.

Tabla 27: Requisito no funcional: #15a. Requisitos de acceso

5. Especificación

5.1 Esquema conceptual de datos

5.1.1 Diagrama UML

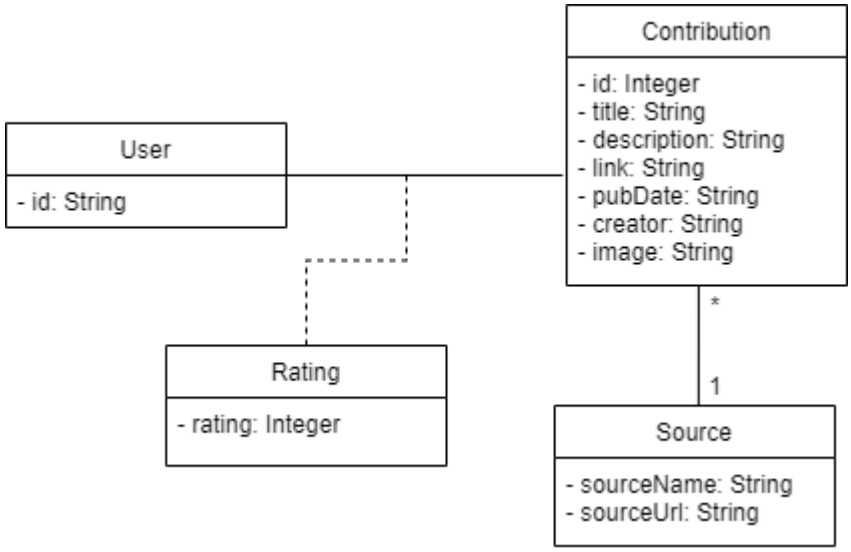


Figura 3:Diagrama UML conceptual.

5.1.2 Descripción de los modelos

- **User**. Representa un usuario registrado del sistema.
- **Contribution**: Representa una noticia del sistema.
- **Rating**: Representa el interés de un usuario por una noticia. Comprende valores entre 0 y 5, ambos incluidos.
- **Source:** Representa de donde proviene una noticia.

6. Diseño del sistema

6.1 Infraestructura

El sistema se basa en dos partes bien diferenciadas. Por un lado está la parte Backend que se encargará de gestionar las llamadas a la API Rest y por otro, la parte Frontend que se encargará de realizar llamadas a la API del Backend para después mostrar la información por pantalla.
Ambas partes se alojarán en Heroku [13] como dos aplicaciones diferentes para poder ser accesibles entre ellas y por supuesto, a cualquier usuario en Internet.

Hay muchos motivos por los que separar el Backend del Frontend. Principalmente es mucho más sencillo programar una funcionalidad específica que programar todo el código del producto en conjunto. Cuando se programa cada parte por separado se tiende a hacer que cada código se encargue de una sola cosa. En cambio, si se programa como una app monolítica muchas veces se acaba mezclando todo un poco, lo cual dificulta el código y sobretodo el mantenimiento del mismo.
Además, al tener el Backend separado, es muy sencillo reutilizarlo en otro proyecto. Como ejemplo se podría realizar un sitio web como el de este sistema y por otra parte, se podría desarrollar una aplicación móvil utilizando el mismo Backend. Sin embargo, si tuviésemos todo el código junto, esto directamente no sería posible.

El motivo de elegir Heroku como alojamiento es que ofrece planes gratuitos, sencillos de utilizar y con opción a migrar a opciones de pago a medida que nuestra aplicación necesita más recursos.

Como sistema de almacenamiento en el Backend se ha decidido utilizar PostgreSQL [14] por su fiabilidad y profesionalidad. Además, una gran ventaja es que Heroku ya viene configurado por defecto con PostgreSQL y por tanto se simplifica mucho la configuración.

Por tanto, un usuario accedería al Frontend para obtener información, el Frontend haría una petición al Backend a través de la API Rest y por último el Backend obtendría los datos necesarios de PostgreSQL, los transformaría a la salida deseada y los devolverá al Frontend que finalmente se los mostraría al usuario.

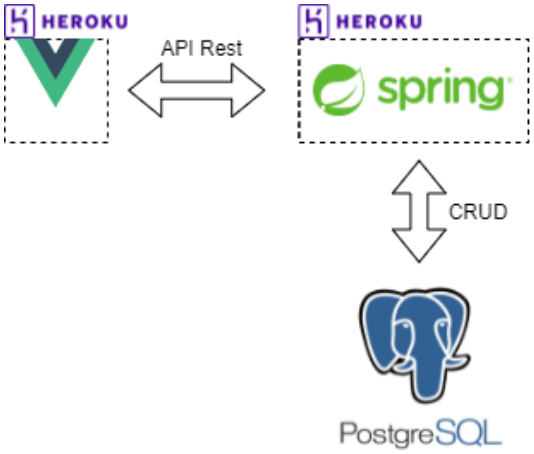


Figura 4: Diagrama de infraestructura

6.2 Backend

Como framework de desarrollo de aplicaciones web se ha elegido Spring Framework ya que es el más popular actualmente y permite un desarrollo J2EE muy sencillo que promueve el uso de buenas prácticas de programación y reutilización de código.

Típicamente en una aplicación construida con Spring, el código se distribuye en controladores, servicios y repositorios. Los controladores se encargan de recibir las peticiones y se comunican con los servicios a los que piden la información necesaria para satisfacer la petición. Por otro lado, los servicios se encargan de la lógica para generar los datos que el controlador a pedido y se comunican con los repositorios para obtener datos de la base de datos. Por último, los repositorios simplemente acceden a la base de datos, buscan la información pedida por el servicio y se lo envían.

Además de esta estructura, después tenemos toda la parte del modelo y clases que permitirán a los servicios realizar sus funciones. Normalmente, todos estos componentes son utilizados por los servicios.

Para una comunicación eficiente entre las peticiones del frontend y cómo se almacenan los datos en backend, se utiliza el patrón DTO y el patrón DAO. Los DTO son unos objetos que se utilizan para realizar la comunicación entre el frontend y el backend y los DAO son unos objetos que se utilizan para representar internamente el almacenamiento de un objeto en la

base de datos. Es decir, están muy ligados entre ellos pero pueden tener diferencias. En este proyecto por ejemplo, se ha utilizado para las noticias. A nivel de comunicación entre front y back se utiliza un objeto DTO llamado Contribution que entre otras propiedades tiene también la puntuación de la noticia del usuario actual. A nivel de almacenamiento, se utiliza un objeto DAO llamado ContributionDAO que tiene prácticamente las mismas propiedades que el DTO pero sin embargo, la puntuación no la necesita, ya que se guarda una noticia internamente sin saber la puntuación, esta se guarda en otra tabla.

6.3 Frontend

A día de hoy, existen tres frameworks muy populares para realizar un proyecto de frontend: Angular, React y Vue. Se han analizado los tres frameworks y para este proyecto, se ha preferido utilizar Vue (aunque cualquiera de ellos hubiese sido una buena opción), principalmente como motivación de aprenderlo ya que no había tenido la oportunidad de trabajar con Vue anteriormente.

Además, se ha hecho una comparativa entre los tres frameworks como se puede ver en la siguiente tabla:

	Angular	**React**	**Vue.js**
Estable	✓	✓	✓
Comunidad	Si, comunidad de Google	Si, comunidad de Facebook	No tan grande como los anteriores, pero respaldado por Laravel y Alibaba
Documentación	✓	✓	✓
Facilidad de aprendizaje	No (Typescript)	Media	Si
Integración con Bootstrap	✓	✓	✓
Tamaño	566K	139K	58,8K
Reutilización de código	✓	Solo CSS	Si, HTML y CSS
Velocidad de programación	Lenta	Normal	Rápida
Reactividad	A medias	✓	✓
Basado en componentes	✓	✓	✓

Tabla 28: Análisis frameworks frontend (Fuente: Belitsoft [15])

Las aplicaciones creadas con Vue normalmente distribuyen el código en componentes con extensión .vue y dentro de estos se dispone tanto de la parte visual que mostrarán como de una parte Javascript para realizar la pequeña lógica (en este proyecto llamadas a la API) que la vista necesitase.

Además de esto, también es común disponer de archivos con extensión .js en la carpeta src a modo de componentes globales que pueden utilizar todos los componentes. Por ejemplo, se puede utilizar esta configuración para tener todas las llamadas a la API de manera ordenada y poder reutilizarlas fácilmente entre componentes.

6.4 Diseño base de datos

En este proyecto, el diseño de la base de datos es muy sencillo. Es muy similar al modelo conceptual de datos, tenemos la tabla User que almacena simplemente la id de cada usuario ya que la información del mismo estaría en los servidores de Google al utilizar OAuth2 con Google Sign In. Después está la tabla Contribution que almacena cada noticia en el sistema para después poder representarla. Por último, se dispone de una relación entre User y Contribution para saber qué noticias ha puntuado cada usuario y con qué puntuación. Esta tabla, simplemente almacena por cada fila la id de la noticia, la id del usuario y la puntuación del usuario hacia esa noticia.

A continuación se muestra el diagrama para mayor claridad:

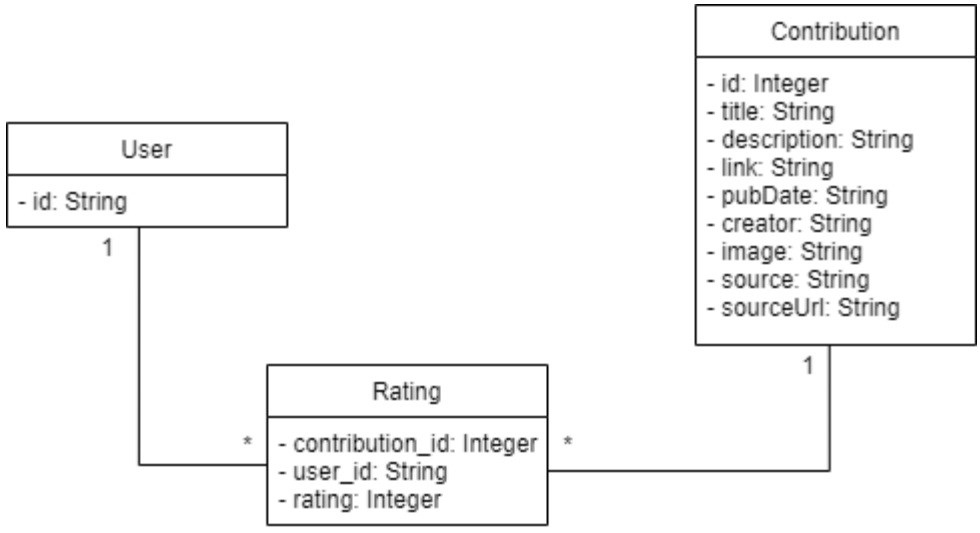


Figura 5: Diagrama UML del diseño de la base de datos.

6.5 Arquitectura

Como arquitectura del sistema se ha decidido utilizar Model Vista Controlador (MVC) ya que es un modelo muy utilizado en este tipo de aplicaciones y a lo largo de los años se ha demostrado que es efectivo.

La arquitectura se basa en tres partes diferenciadas:

- **Modelo:** Se trata de la parte de persistencia de datos en base de datos. En concreto se trata de todos los objetos DAO del backend que representan cómo se almacena un objeto en cuestión dentro de la base de datos.

- **Vista:** Es la capa de presentación que interactúa con el usuario. El usuario utiliza esta parte para obtener información que se consigue mediante llamadas a la API Rest del controlador. En este proyecto, se trata de todos los componentes de Vue que serán los encargados de mostrar al usuario la interfaz y que él mismo pueda interactuar con ella.

- **Controlador:** Se encarga de recibir peticiones de la vista y se comunica con el Modelo para obtener los datos y devolverlos a la vista. Se trata de todos los controladores backend que recibirán peticiones del frontend (componentes de Vue) y satisfacerá los datos necesarios.

Además de estos tres componentes también existe una capa adicional entre el controlador y el modelo, se trata de los **servicios**. Como ya se ha comentado previamente, utilizan el modelo y realizan operaciones juntamente con los **repositorios** (que acceden a base de datos) para devolver la información pedida a los controladores.

6.6 Patrones de diseño

Con el objetivo de utilizar buenas prácticas y diseños ya existentes y probados, en este proyecto se han utilizado diferentes patrones de diseño que ayudan a simplificar el código y tener un código de calidad.

6.6.1 Patrón DTO

El patrón DTO permite definir una serie de atributos que se utilizarán tanto para enviar datos como para recibirlos por parte del usuario. Es especialmente útil para separarlo de las entidades que acceden a la base de datos y por tanto tener objetos diferentes y por tanto no necesitamos mostrar todos los datos que nuestro sistema utiliza.

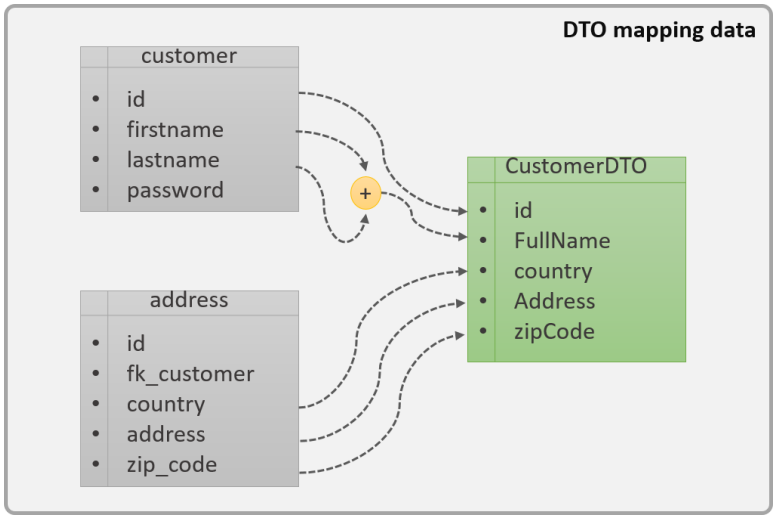


Figura 6: Patrón DTO. Fuente: Oscar Blancarte Blog [16]

6.6.2 Patrón DAO

El patrón DAO propone separar la lógica de negocio del acceso a los datos. Este patrón proporciona por tanto métodos para insertar, actualizar, borrar y consultar información.

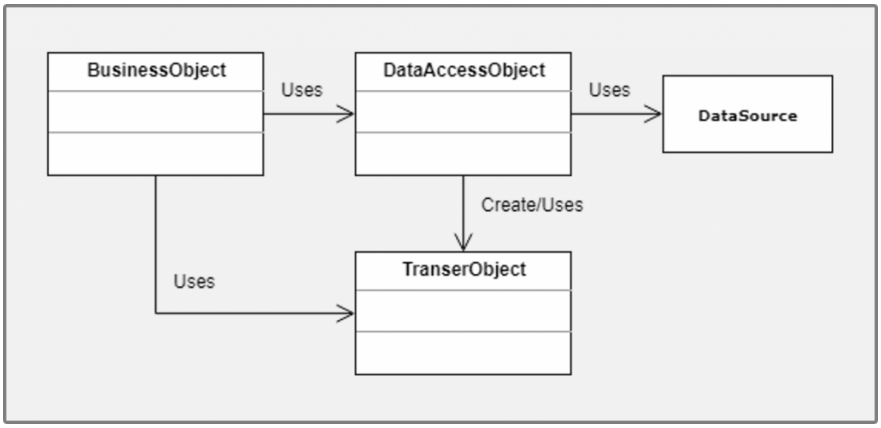


Figura 7: Patrón DAO. Fuente: Oscar Blancarte Blog [17]

6.6.3 Patrón Restful API

Se trata de un patrón que recibe una petición HTTP y devuelve datos mediante otra petición HTTP. Es de tipo stateless (no mantiene estados ni en cliente ni en servidor) y por tanto cada petición es necesario que tenga toda la información que el servidor necesite para proporcionar un resultado.

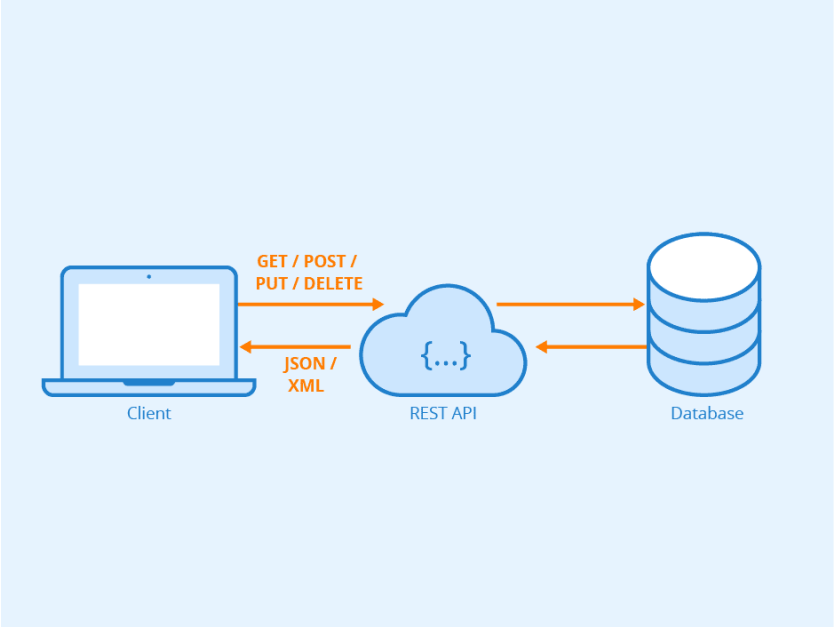


Figura 8: Patrón Restful API. Fuente: Datamounts [18]

6.7 Autenticación mediante OAuth2

Se han contemplado dos opciones de autenticación para el proyecto. La primera, más simple, pero también más laboriosa de programar, es la típica autenticación de usuarios con almacenamiento interno en el servidor de la propia aplicación.
La segunda, y además, la elegida, la autenticación mediante OAuth2, específicamente con el servicio de Google Sign-in. Se trata de una autenticación utilizando servicios de terceros, típicamente bien conocidos como puede ser Google, Facebook, Github, etc.

Gracias a utilizar este sistema de autenticación, se evita tener que programar toda la lógica para almacenar los usuarios, pero por otra parte se debe programar la lógica para interactuar con el servicio de terceros escogido.

A continuación se muestra un esquema para clarificar cómo funciona la autenticación mediante OAuth2:

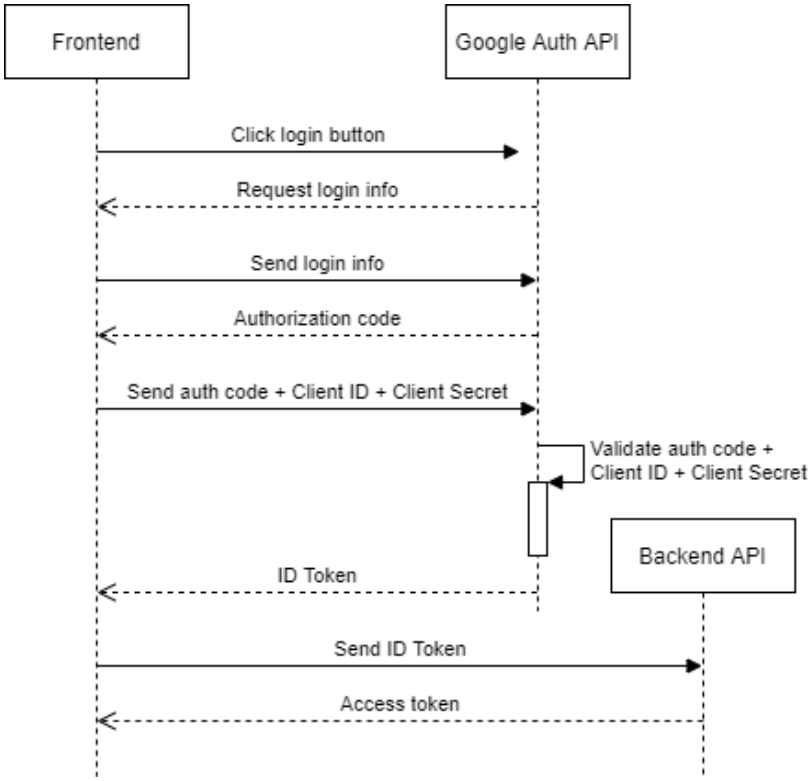


Figura 9: Diagrama autenticación OAuth2. Fuente: Elaboración propia

6.8 Comunicación frontend y backend

El sistema de comunicación entre ambos es el típico en aplicaciones actuales. El backend actúa como un servicio que no tiene ningún tipo de estado y por tanto no recuerda nada de peticiones anteriores y el frontend consume este servicio para obtener los datos que necesita en cada caso. A continuación se explican dos ejemplos, uno para obtener una noticia del backend y otra para puntuar una noticia.

6.8.1 Comunicación obtención de noticia

Para obtener una notícia completa y mostrar sus detalles el frontend tiene que hacer una petición al endpoint: /api/contributions y enviar una id por parámetro con nombre id. Por ejemplo podría ser algo asi: /api/contributions?id=42. Una vez al backend le llega la petición le pide al ContributionsService la información necesaria y este a su vez hace sus

operaciones y pide los datos necesarios al ContributionsRepository que internamente accede a la base de datos y finalmente devuelve la noticia. En la siguiente figura se puede ver el diagrama de comunicación para obtener una noticia:

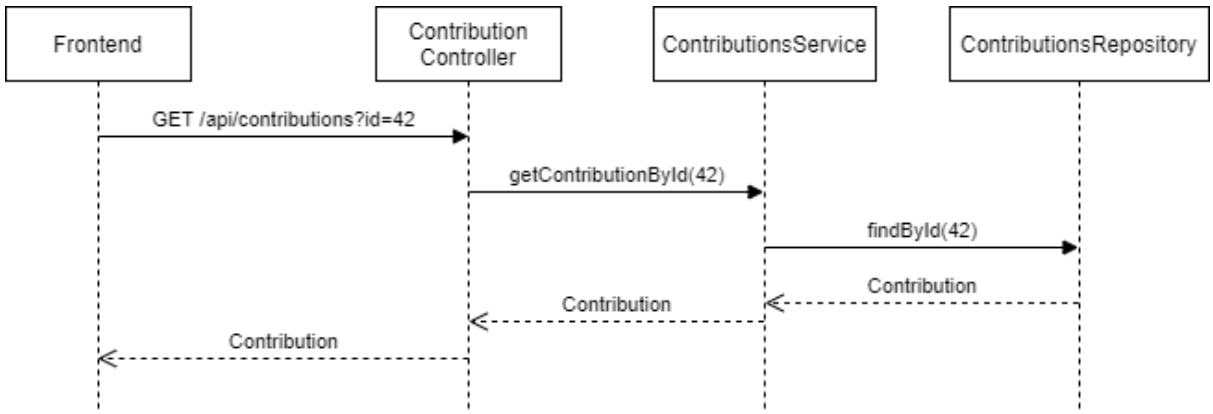


Figura 10: Diagrama de comunicación frontend y backend para obtener una noticia

6.8.2 Comunicación puntuar una noticia

Para puntuar una noticia se sigue el mismo patrón que en el caso anterior, el frontend envía un PUT con la puntuación de la noticia y la id de la noticia al backend. Este recibe esa información y pide al servicio UsersContributionService que actualice el voto de esa noticia para el usuario que ha realizado la votación y este servicio accede al UsersService para modificar la puntuación de esta noticia para ese usuario. En la siguiente figura se puede ver el diagrama de comunicación para puntuar una noticia:

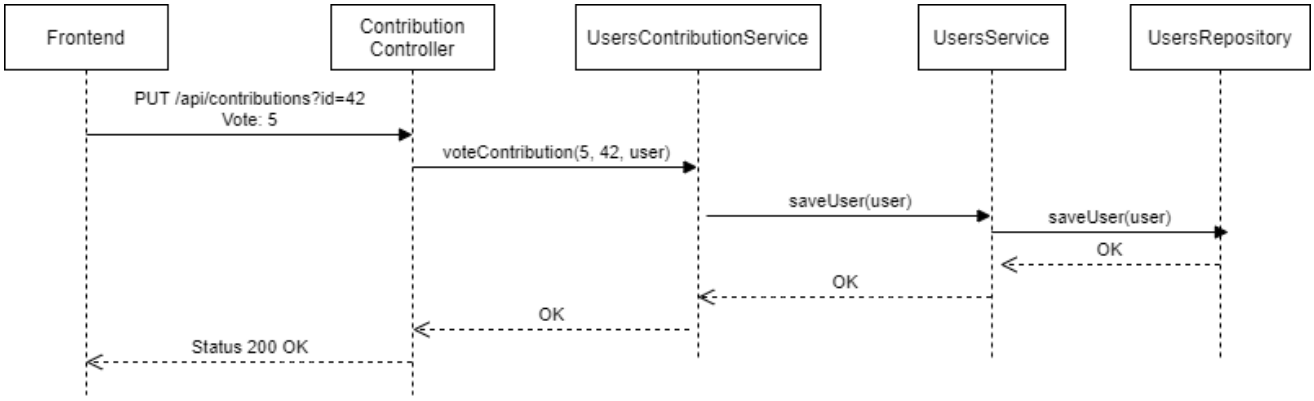


Figura 11: Diagrama de comunicación frontend y backend para puntuar una noticia

6.9 Vistas del sistema y navegabilidad

6.9.1 Vistas del sistema

Se puede pensar que el sistema utiliza 4 vistas: últimas noticias, noticias recomendadas, noticias puntuadas y detalles completos de una noticia pero en realidad se utilizan simplemente 2 vistas ya que las tres primeras pueden ser reutilizadas completamente debido a que muestran el mismo formato de información pero con información diferente.

A continuación se muestran las vistas del listado de noticias y detalles completos de una noticia.

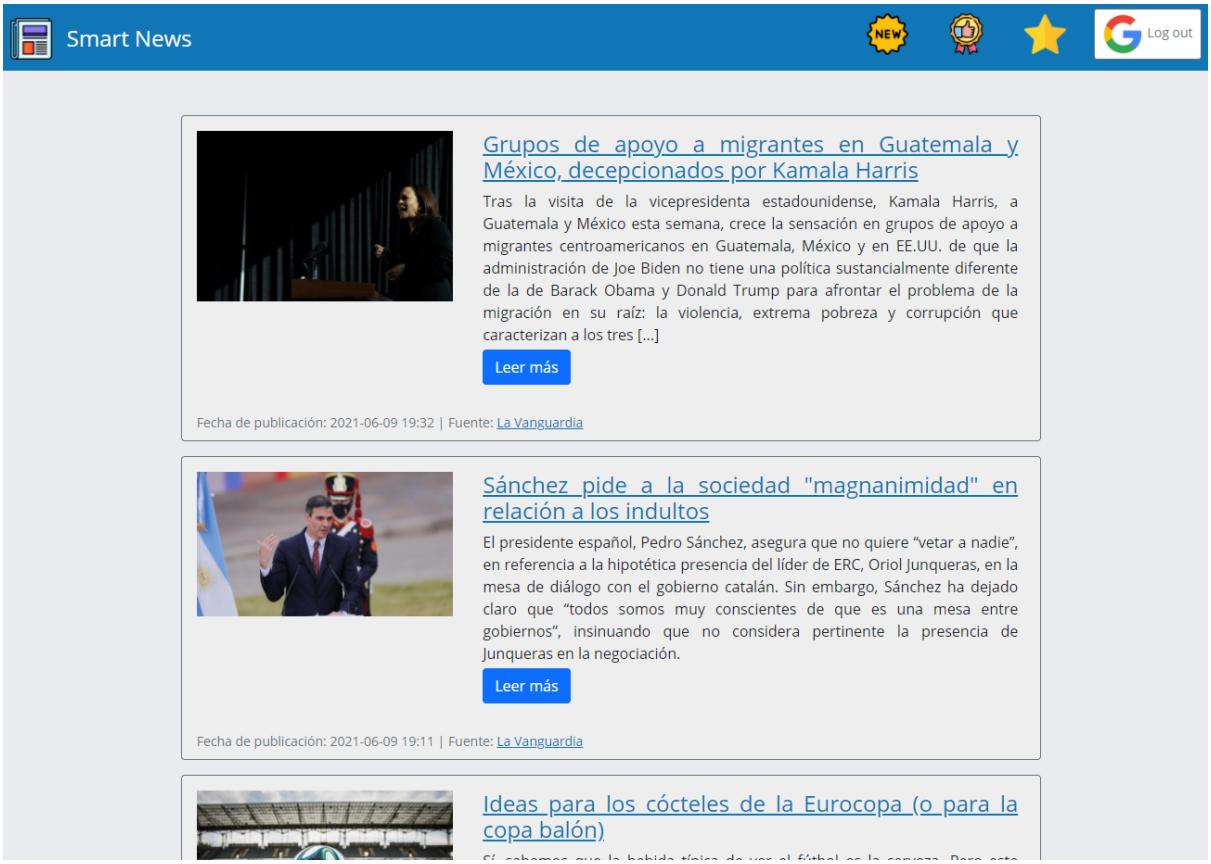


Figura 12: Vista de listado de noticias.

Figura 13: Vista de detalle de una noticia completa.

6.9.2 Navegabilidad

Para navegar entre pantallas se ha optado por un menú horizontal en la parte superior de la pantalla. Se trata de tres botones que permitirán acceder a las pantallas de últimas noticias, noticias recomendadas y noticias puntuadas por el usuario.

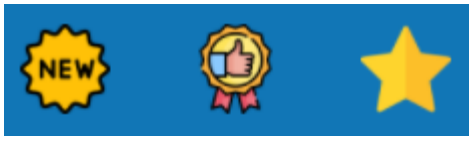


Figura 14: Botones de navegación de la UI

En caso de visualizar el sitio web a través de un móvil o tablet, el funcionamiento es el mismo. Se mostrarán estos tres botones pero adaptados a la pantalla en cuestión.

Para navegar a una noticia en cuestión y verla de manera completa, basta con clicar en el título de la noticia o en el botón “Ver más”

7. Sistemas de recomendación

Un sistema de recomendación es un software que provee sugerencias de “objetos” a un usuario. Los objetos pueden ser cualquier cosa, como por ejemplo, recomendaciones de películas, canciones, productos o en el caso de este proyecto, noticias. Estas recomendaciones normalmente varían en función de cada usuario y existen diferentes tipos de sistemas de recomendación en función de las necesidades del sistema.

7.1 Clasificación

7.1.1 Filtrado colaborativo

Es el sistema más popular y el más implementado, además, también ha sido el elegido para este proyecto. Este sistema de recomendación se basa en recomendar contenidos con valoraciones positivas de otros usuarios que tienen un perfil de gustos similar al usuario que se quiere recomendar, por eso se les llama colaborativos.

Existen dos tipos de filtrado colaborativo:

- Basado en usuarios: Se identifican usuarios similares y se recomiendan objetos a otro usuario similar que no haya puntuado ese objeto.

- Basado en objetos: Se identifican objetos similares y se recomiendan a usuarios que no los tenga puntuado y si tenga puntuados otros similares..

7.1.2 Filtrado basado en contenido

Este sistema se basa en ofrecer recomendaciones de objetos similares a otros objetos que el usuario ya valoró en el pasado. Principalmente utilizan la información del perfil del usuario para predecir qué le puede gustar al usuario.

7.1.3 Filtrado basado en conocimiento

Son sistemas que se basan en la información proporcionada previamente por el usuario para ofrecer recomendaciones. Son especialmente útiles en mercados que cambian muy rápido donde los gustos anteriores del usuario no son válidos actualmente.

7.1.4 Filtrado demográfico

Como el propio nombre indica, se basan en la demografía a la que pertenece un usuario para ofrecer recomendaciones. La idea es ofrecer recomendaciones diferentes para demográficas diferentes.

7.1.5 Híbridos

Se puede utilizar combinaciones de filtrados explicados en los puntos anteriores. Cuando esto sucede, podemos hablar de un sistema híbrido. Por ejemplo, es muy común ofrecer recomendaciones demográficas cuando un usuario lleva poco tiempo en el sistema y cuando ya ha valorado una cantidad suficiente de objetos, entonces se puede ofrecer recomendaciones de filtrado colaborativo.
De esta manera, el usuario desde el principio obtiene recomendaciones más o menos útiles, y a medida que va utilizando el sistema, obtendrá mejores recomendaciones al empezar a utilizar el filtrado colaborativo.

7.2 Algoritmo Slope One

Slope One es uno de los algoritmos que se pueden implementar para utilizar filtrado colaborativo. Tiene en cuenta la información de todos los usuarios que han valorado el mismo objeto y de otros objetos valorados por el usuario para calcular una matriz de similitudes.

Este ha sido el algoritmo elegido a implementar ya que es un algoritmo fácil de implementar y que ofrece buenos resultados, por tanto es ideal para un prototipo como el que se pretende realizar en este proyecto.
En el siguiente capítulo se darán detalles sobre la implementación de este algoritmo.

Primero, los usuarios valoran diferentes objetos del sistema. Después el algoritmo calcula una matriz de similitudes y finalmente, el sistema hace predicciones basadas en los usuarios para objetos que el usuario que recibe la predicción, aún no ha valorado.

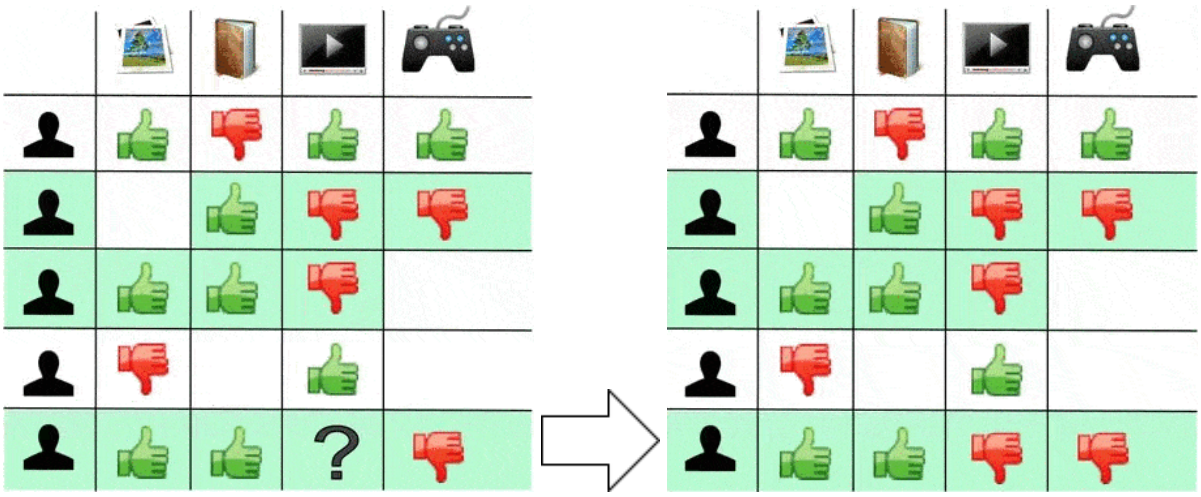


Figura 15: Ejemplo de filtrado colaborativo. Fuente: Baeldung [19]

La idea principal del algoritmo es crear una relación lineal entre las preferencias de los objetos como una relación F(x) = x + b.
Básicamente calcula la diferencia de puntuaciones de cada usuario a cada objeto. Después crea una diferencia media para cada par de objetos.

Supongamos la siguiente tabla para analizar un ejemplo:

	Objeto 1	Objeto 2	Objeto 3
Usuario A	3	3	?
Usuario B	2	5	3

Tabla 29: Puntuaciones de usuarios a objetos.

Primero se calcula la diferencia entre objetos:

- Diff (Objeto 3 x Objeto 1) = 3 - 2 = 1
- Diff (Objeto 3 x Objeto 2) = 3 - 5 = -2

Por último, calculamos la predicción:

- Pred (Objeto 3) = [(1 + 3) + (-2 + 3)] / 2 = 2,5

Por tanto, el resultado del Objeto 3 para el usuario A, sería de 2,5.

8. Implementación

En este apartado se explica como se ha implementado y desarrollado el proyecto, tanto las tecnologías y lenguajes utilizados, como las herramientas con las que se ha desarrollado.

8.1 Tecnologias utilizadas

Como ya se ha comentado anteriormente, el proyecto se divide en dos partes: Backend y Frontend. A continuación se comentan las tecnologías utilizadas en cada uno de ellos.

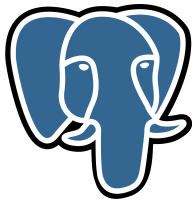
8.1.1 Tecnologias backend

El backend ha sido desarrollado con Spring Framework para crear la API Rest y como sistema de almacenamiento se ha utilizado PostgreSQL.

Spring Framework [20]

Se trata de un framework para el desarrollo de aplicaciones web, basado principalmente en el patrón MVC. Ofrece diferentes módulos en función de las necesidades de cada proyecto y facilita mucho tanto la arquitectura de una aplicación de este tipo como empezar a desarrollar código de manera rápida con muchas automatizaciones que ya trae por defecto.

PostgreSQL



PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto con más de 30 años de desarrollo activo.

8.1.2 Tecnologias frontend

El frontend ha sido desarrollado principalmente con el framework de Vue.js.

Vue.js [21]

Vue.js es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página (SPA). Permite utilizar componentes que se pueden reutilizar en diferentes partes de la página y por tanto ayuda a simplificar el código.

8.2 Servicios externos

Google Sign-In [22]

Google Sign-In es un servicio ofrecido por Google que permite autenticar a usuarios mediante la cuenta de Google. Permite al usuario no tener que registrarse con una nueva cuenta en una nueva página web y además le facilita el inicio de sesión de manera más rápida.

8.3 Herramientas utilizadas

Durante el desarrollo del proyecto se han utilizado varias herramientas para facilitar el desarrollo del mismo y para llevar un control.

Git

Git es un software de control de versiones que permite guardar dichas versiones por cada modificación que se ha ido haciendo. Se pueden crear diferentes ramas para poder trabajar en diferentes partes del código simultáneamente y finalmente juntarlas en una misma rama y tener todo el código desarrollado.

GitHub

GitHub permite alojar proyectos utilizando el sistema de control de versiones de Git. Se pueden compartir proyectos a diferentes personas para trabajar en conjunto. Sirve como un repositorio en la nube.

Jira

Jira es una herramienta para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. Permite llevar un control más claro de en qué estado se encuentra el proyecto, que tareas se han hecho y cuáles no.

8.4 Estructura del código

En este apartado se describe la estructura del código tanto en la parte backend como en la parte frontend. Los repositorios de GitHub se pueden consultar en los siguientes enlaces:

- Backend: https://github.com/cristianrb/smartnews
- Frontend:https://github.com/cristianrb/smartnews-front

8.4.1 Estructura del código backend

La estructura del código backend viene un poco predefinida por el uso de Spring. El código se basa en controladores que atienden las peticiones http, hacen llamadas a los servicios que se encargan de obtener los datos necesarios de la base de datos utilizando los repositorios y de realizar las operaciones necesarias para devolverlos al controlador que finalmente los envía a quien realizó la llamada, en este caso el frontend.
En la siguiente figura se puede observar dicha estructura:

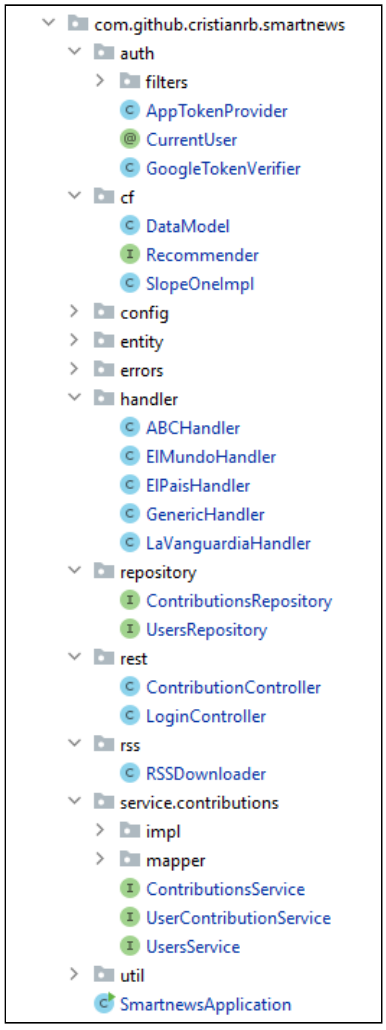


Figura 16: Estructura del código backend

En la siguiente figura se muestra una llamada al controlador y como él mismo utiliza un servicio para obtener los datos necesarios:

```
@ApiOperation(value = "Retrieves at most 10 contributions of a given page")
@GetMapping("/latest")
public Page<Contribution> getAllContributions(@RequestParam(name = "page", defaultValue = "0") Integer page) {
    return contributionsService.getAll(PageRequest.of(page, PAGE_SIZE))
            .map(ContributionsMapper::mapContributionDAOToContribution);
}
```

Figura 17: Petición en el controlador.

Finalmente tenemos la implementación en el servicio del método *getAll* que accede al repositorio:

```
@Override
public Page<ContributionDAO> getAll(Pageable paging) {
    return this.contributionsRepository.findAllByOrderByPubDateDescIdDesc(paging);
}
```

Figura 18: Implementación método *getAll* en el servicio.

De cómo acceder a los datos de la base de datos, se encarga el repositorio y lo hace internamente Spring. En una aplicación sin Spring se utilizaría JDBC y se debería implementar la query SQL que recogería los datos.

El uso del recomendador es exactamente igual que por ejemplo la obtención de las últimas noticias, es como si fuese un servicio aunque un poco diferente ya que no solo debe obtener los datos y enviarlos, sino que además debe tratarlos para poder enviar algo con sentido. Por tanto, la secuencia sería la misma que para obtener las últimas noticias:

1. El frontend envía una petición para obtener las recomendaciones del usuario actual.
2. El controlador del backend recibe la petición y pide a la interfaz Recommender las noticias recomendadas.
3. SlopeOneImpl que implementa la interfaz pide la información de las puntuaciones de cada noticia por usuario a DataModel.
4. SlopeOneImpl trata los datos recibidos de DataModel y los devuelve al controlador.
5. El controlador devuelve los datos recibidos de SlopeOneImpl al fronted.

A continuación se muestra un diagrama de secuencia para clarificar:

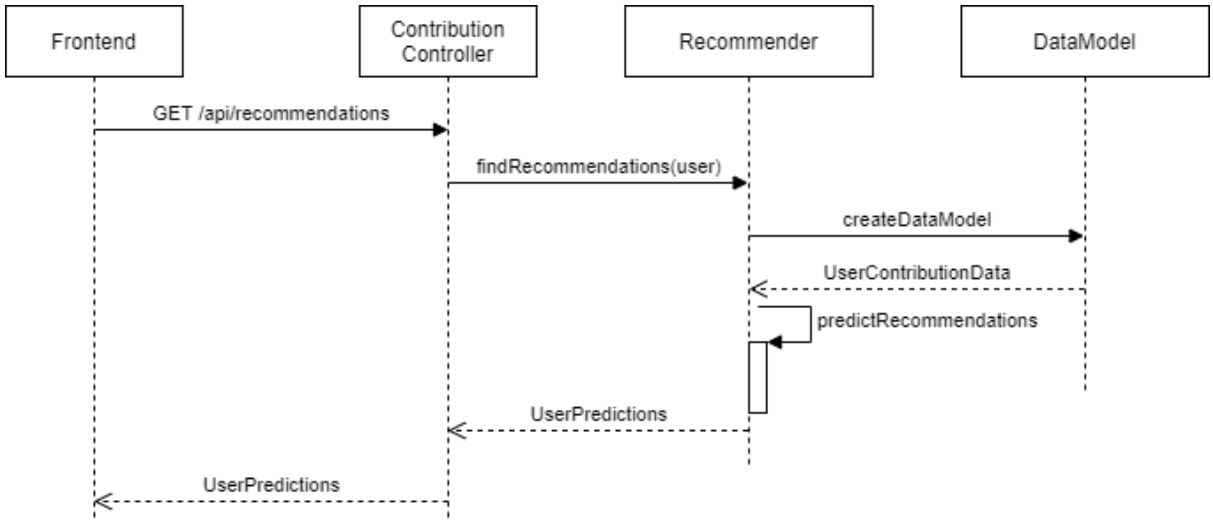


Figura 19: Diagrama de secuencia de uso del recomendador.

Finalmente, está la carpeta *test* donde se definen los tests utilizados para la comprobación del backend.

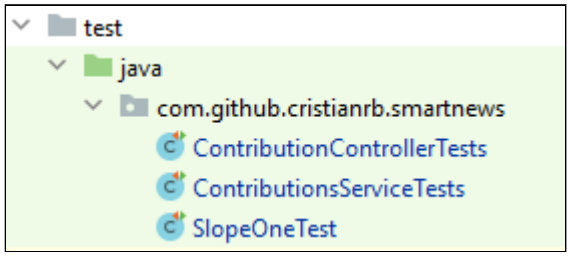


Figura 20: Estructura de los tests en backend.

En el siguiente capítulo se entrará en profundidad en los tests, por ello en este apartado simplemente se comenta la estructura de los mismos.

8.4.2 Estructura del código frontend

La estructura del código frontend se define en base a Vue.js. Este framework se basa en componentes para mostrar las vistas al usuario y son los mismos componentes los que tienen la lógica para acceder a la API del backend. Se puede mejorar un poco más creando un archivo que funcione como un servicio de peticiones al backend y de esta forma separar un poco la lógica de la petición de la vista. En este proyecto se ha creado este archivo con el nombre AxiosService.js.

Los tests del frontend han sido definidos en una carpeta llamada __tests__ siguiendo la convención de *jest* [23], una librería para hacer testing en frontend.

A continuación se muestra una figura de la estructura del frontend:

Figura 21: Estructura del frontend.

Siguiendo con el ejemplo del backend, a continuación se muestra un trozo de código de un componente (LatestNews) para obtener las últimas noticias:

```
getInitialNews() {
  this.isLoading = true
  AxiosService.getLatestNews(0, this.path)
    .then((res) => {
        this.contributions = this.formatAndReduceDescription(res.data.content);
    }).catch(() => {
        this.isLoading = false;
    });
},
```

Figura 22: Método *getInitialNews* del componente LatestNews.

Por último, se muestra como el servicio AxiosService realiza la petición al backend:

```
getLatestNews(pageNumber, path) {
    var apiPath = "/latest"
    if (path === "/rated" || path === "/recommendations") {
        apiPath = path;
    }
    return apiClient.get(baseURLApi + apiPath + '?page=' + pageNumber)
},
```

Figura 23: Método *getLatestNews* de *AxiosService*.

ApiClient es un objeto definido en el mismo AxiosService que se encarga de añadir el token del usuario a cada petición utilizando *interceptors* de JavaScript o de obtener el token mediante la autenticación de Google y posterior llamada al Backend en caso de que el usuario no tuviese la sesión iniciada. A continuación se muestra una figura mostrando todo el proceso:

```
const apiClient = axios.create({
    headers: {
        'Content-Type': 'application/json'
    }
})

apiClient.interceptors.request.use(function (config) {
    if (config.url.includes("/login")) return config;

    let authToken = localStorage.getItem("authToken")
    let googleIdToken = localStorage.getItem("googleIdToken")
    if (authToken) {
        config.headers.common = {
            "Authorization": authToken
        }
    } else {
        if (googleIdToken && !authToken) {
            let headers = {
                'Content-Type': 'application/json',
                'X-ID-TOKEN': googleIdToken
            };
            axios.post(baseURL + "/login", "", { 'headers': headers })
                .then((res) => {
                    if (res.status == 200) {
                        localStorage.setItem("authToken", res.headers.authorization)
                        localStorage.setItem("authorized", true)
                        config.headers.common = {
                            'Content-Type': 'application/json',
                            "Authorization": res.headers.authorization
                        }
                    }
                }).catch(error => {
                    console.log(error)
                });
        }
    }
    return config;
}, function (err) {
    return Promise.reject(err.response)
});
```

Figura 24: Objeto *apiClient* y uso de *interceptors* en *AxiosService*.

8.5 Implementación obtención de noticias

8.5.1 Clase RSSDownloader

RSSDownloader es una clase que tiene varias funcionalidades como establecer un scheduler para realizar descargas periódicamente, lanzar la función que descargara las noticias y por último almacenar las noticias en el sistema.

A continuación se muestra la función downloadFromAllSources qué es la que se encarga de que cada 15 minutos, se ejecute la descarga de noticias y almacenarlas:

```java
@Scheduled(fixedRate=900000) // Every hour = (60*60*1000=3600000), minute = 60000, 15 mins = 900000
public void downloadFromAllSources() {
    this.contributionsService = SpringContextConfig.getBean(ContributionsService.class);
    sources = new ArrayList<Pair<String, GenericHandler>>();
    sources.add(new Pair<>("https://www.abc.es/rss/feeds/abcPortada.xml", new ABCHandler()));
    sources.add(new Pair<>("https://feeds.elpais.com/mrss-s/pages/ep/site/elpais.com/portada", new ElPaisHandler()));
    sources.add(new Pair<>("https://e00-elmundo.uecdn.es/elmundo/rss/portada.xml", new ElMundoHandler()));
    sources.add(new Pair<>("https://www.lavanguardia.com/newsml/home.xml", new LaVanguardiaHandler()));
    List<Contribution> contributionsFromAllSources = new ArrayList<~>();
    for (Pair<String, GenericHandler> pair : sources) {
        List<Contribution> contributionList = downloadNews(pair.getKey(), pair.getValue());
        if (contributionList != null) {
            contributionsFromAllSources.addAll(contributionList);
        }
    }
    saveContributionsInDB(contributionsFromAllSources);
    logTime();
}
```

Figura 25: Método downloadFromAllSources para ejecutar el flujo de descarga de noticias.

Inicialmente se anota el método con la anotación @Scheduled a la que se le puede pasar un tiempo en milisegundos. En este caso, se ha decidido descargar noticias cada 15 minutos ya que es un periodo de tiempo aceptable en el que pueden haber cambiado las noticias publicadas por cada periodico y tampoco es demasiado abusivo.
Después se crea una lista de donde se van a obtener las noticias asociado a su handler, que se encarga de mapear la información de cada periodico a la entidad que se utiliza en la base de datos, la Contribution.

A continuación se utiliza el método downloadNews para descargar las noticias de cada periodico. Este método se encarga de crear una instancia de SAXParser que permite leer XML de manera sencilla, solo implementado tres métodos: **startElement**: permite saber cuando empieza un nuevo elemento**, endElement**: permite saber cuando acaba un elemento y **characters**: permite saber cuántos caracteres se han leído del elemento actual, pues un elemento puede ser muy largo y se deberá procesar en varias tandas**.**

```java
public List<Contribution> downloadNews(String xmlUrl, GenericHandler handler) {
    SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
    List<Contribution> contributionList = null;
    try {
        SAXParser saxParser = saxParserFactory.newSAXParser();
        saxParser.parse(new InputSource(new URL(xmlUrl).openStream()), handler);
        contributionList = handler.getContributionList();
    } catch (Exception e) {
        e.printStackTrace();
        // Continue reading news
    }
    return contributionList;
}
```

Figura 26: Método downloadNews

Por último el método llama a saveContributionsInDB que simplemente guarda todas las noticias en la base de datos.

8.5.2 Handlers

Tal y como se ha comentado en el apartado anterior para descargar noticias se utiliza SAXParser y a este objeto hay que pasarle un objeto de tipo DefaultHandler que es el que realmente implementa los métodos de startElement, endElement y characters, y se encarga de realizar las operaciones necesarias en función del XML que se este leyendo.

En el momento de el formato de cada XML de cada periodico, pude comprobar que eran muy parecidos entre ellos, aunque con ligeras diferencias. Debido a esto, inicialmente cree la clase GenericHandler (que implementa DefaultHandler) para tener un punto de partida común con todos los XML que se iban a leer.

Esta clase tiene una serie de atributos que serán comunes en todos los handlers y que definen una noticia:

```
protected String item = "item";
protected String title = "title";
protected String link = "link";
protected String description = "description";
protected String category = "category";
protected String pubDate = "pubDate";
protected String creator = "dc:creator";
protected String urlImage = "media:content";

private List<Contribution> contributionList = null;
private Contribution contribution;
private StringBuilder data;
```

Figura 27: Atributos de una notícia en GenericHandler

El contenido de cada atributo es el nombre de cada elemento que nos interesa en el XML. El motivo por el que tienen que haber varios handlers, es que por ejemplo estos elementos del XML pueden tener un nombre diferente para representar la misma información. Por ejemplo 3 de los 4 periódicos elegidos utilizan la etiqueta “title” para representar un titulo, pero sin embargo La Vanguardia utiliza la etiqueta “HeadLine” para representar esta misma información.

Una vez definidos los campos que queremos leer, basta con implementar los tres métodos mencionados anteriormente.

En primer caso, tenemos startElement, de lo que se trata en este método es que cada vez que empiece un nuevo elemento de tipo “item”, es decir, se trata de una noticia nueva, simplemente reiniciamos el atributo contribution y data.

```
@Override
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
    if (qName.equalsIgnoreCase(item)) {
        if (getContributionList() == null) {
            initializeContributionList();
        }
        initializeContribution();
    }
    initializeData();
}
```

Figura 28: Implementación startElement

Siguiendo con endElement, tal y como su nombre indica, una vez un elemento se cierra tenemos que almacenar esa información en el campo correspondiente de la contribution:

```
@Override
public void endElement(String uri, String localName, String qName) throws SAXException {
    if (getContribution() != null) {
        if (qName.equalsIgnoreCase(title)) {
            getContribution().setTitle(getData().toString());
        } else if (qName.equalsIgnoreCase(link)) {
            getContribution().setLink(getData().toString());
        } else if (qName.equalsIgnoreCase(description)) {
            getContribution().setDescription(getData().toString());
        } else if (qName.equalsIgnoreCase(category)) {
            getContribution().getCategories().add(getData().toString());
        } else if (qName.equalsIgnoreCase(pubDate)) {
            getContribution().setPubDate(getData().toString());
        } else if (qName.equalsIgnoreCase(creator)) {
            getContribution().setCreator(getData().toString());
        } else if (qName.equalsIgnoreCase(item)) {
            getContributionList().add(contribution);
        }
    }
}
```

Figura 29: Implementación endElement

Y por último, hay que implementar el método characters para ir almacenando los caracteres leídos y después poder guardar toda la información completa en el método endElement:

```
@Override
public void characters(char ch[], int start, int length) throws SAXException {
    data.append(new String(ch, start, length));
}
```

Figura 30: Implementación characters

El resto de handlers funcionan heredando de GenericHandler y en caso de necesitarlo deberán redefinir startElement y/o endElement. Además, en el endElement de cada handler se aprovecha para definir los atributos source y sourceUrl que no provienen como tal de ningún periodico pero si podemos saber a quién pertenecen sabiendo en qué handler nos encontramos.

A continuación se muestra todo el código de ElPaisHandler para observar cómo se resuelven las diferencias que pueda haber entre periódicos. Por supuesto, cada handler tendrá diferentes cambios.

```
public class ElPaisHandler extends GenericHandler {

    @Override
    public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
        super.startElement(uri, localName, qName, attributes);
        if (qName.equalsIgnoreCase(urlImage)) {
            String image = attributes.getValue( qName: "url");
            if (checkImageFormat(image)) getContribution().setUrlImage(image);
        }
    }

    @Override
    public void endElement(String uri, String localName, String qName) throws SAXException {
        super.endElement(uri, localName, qName);
        if (qName.equalsIgnoreCase(item)) {
            getContribution().setSource("El País");
            getContribution().setSourceUrl("www.elpais.com");
        } else if (qName.equalsIgnoreCase(pubDate)) {
            java.util.Date date = new Date(getData().toString());
            SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd hh:mm");
            String format = formatter.format(date);
            getContribution().setPubDate(format);
        }
    }
}
```

Figura 31: Código de ElPaisHandler

8.6 Implementación de Slope One

Para la implementación de este algoritmo se ha utilizado dos clases y una interfaz. La interfaz es muy sencilla, simplemente contiene un método que dado un usuario, devuelve una lista de noticias que le pueden interesar. El objetivo de utilizar una interfaz es para en el futuro poder utilizar diferentes algoritmos siempre aceptando el contrato de dicha interfaz y por tanto no tener necesidad de cambiar el código de los controladores.

```
public interface Recommender {

    public List<Contribution> findRecommendations(User user);
}
```

Figura 32: Interfaz Recommender

La siguiente clase a comentar es DataModel. Esta clase principalmente se encarga de obtener toda la información de todos los usuarios y sus puntuaciones a noticias. El objetivo de esta clase es obtener un objeto al que poder consultar la información de un usuario y nos devolverá cada noticia puntuada por ese usuario y qué puntuación tiene. Por tanto, nos da un punto de partida muy interesante para la siguiente clase que no tiene que preocuparse de cómo obtener estos datos. De nuevo, la idea de esta clase DataModel, es en el futuro poder jugar con otros algoritmos y tener que pensar solamente en la implementación.

```
public Map<User, Map<Contribution, Double>> createDataModel() {
    if (usersService == null) {
        usersService = SpringContextConfig.getBean(UsersService.class);
    }
    data = new HashMap<>();
    contributions = new HashSet<>();
    List<UserDAO> users = usersService.getAllUsers();
    for (UserDAO userDAO : users) {
        Set<UserContributionDAO> newsByUser = userDAO.getContributionsVisited();
        Map<Contribution, Double> newsVisited = new HashMap<>();
        for (UserContributionDAO userAndNew : newsByUser) {
            Contribution cont = ContributionsMapper.mapContributionDAOToContribution(userAndNew.getContribution());
            newsVisited.put(cont, (double) userAndNew.getVote());
                contributions.add(cont);
        }

        User user = new User(userDAO.getId());
        data.put(user, newsVisited);
    }

    return data;
}
```

Figura 33: Implementación de createDataModel en la clase DataModel

Por último, llegamos a la clase que realmente hace el trabajo, esta clase la he llamado SlopeOneImpl ya que es el algoritmo que he decidido implementar en el proyecto. Como se ha comentado previamente, esta clase implementa la interfaz Recommender y obtiene los datos de los usuarios y sus puntuaciones de DataModel.

Este algoritmo consta de dos partes, primero construye dos matrices de diferencias para calcular la diferencia entre las puntuaciones del usuario y de frecuencias para saber cuántas veces una misma noticia ha sido puntuada. Después se utilizan ambas matrices para crear las predicciones.

Finalmente se divide la diferencia de puntuaciones calculada entre el número de ocurrencias, es decir la frecuencia y se almacena en la matriz de diferencias.

```java
private void buildDiffFreqMatrix() {
    if (this.dm == null) this.dm = new DataModel();
    this.inputData = dm.createDataModel();
    for (Map<Contribution, Double> userRating : inputData.values()) {
        for (Map.Entry<Contribution, Double> entry : userRating.entrySet()) {
            if (!diff.containsKey(entry.getKey())) {
                diff.put(entry.getKey(), new HashMap<>());
                freq.put(entry.getKey(), new HashMap<>());
            }

            for (Map.Entry<Contribution, Double> entry2 : userRating.entrySet()) {
                int oldCount = 0;
                if (freq.get(entry.getKey()).containsKey(entry2.getKey())) {
                    oldCount = freq.get(entry.getKey()).get(entry2.getKey());
                }

                double oldDiff = 0.0;
                if (diff.get(entry.getKey()).containsKey(entry2.getKey())) {
                    oldDiff = diff.get(entry.getKey()).get(entry2.getKey());
                }

                double observedDiff = entry.getValue() - entry2.getValue();
                freq.get(entry.getKey()).put(entry2.getKey(), oldCount + 1);
                diff.get(entry.getKey()).put(entry2.getKey(), oldDiff + observedDiff);
            }
        }
    }

    for (Contribution cont : diff.keySet()) {
        for (Contribution c : diff.get(cont).keySet()) {
            double oldValue = diff.get(cont).get(c);
            int count = freq.get(cont).get(c);
            diff.get(cont).put(c, oldValue / count);
        }
    }
}
```

Figura 34: Creación de las matrices de diferencias y frecuencias en Slope One.

Una vez calculadas estas dos matrices, hay que predecir las puntuaciones no existentes. Para hacer esto, por cada usuario se itera sobre su lista de noticias puntuadas y aquí dentro por cada noticia puntuada, se itera sobre todas las noticias que tenemos en la matriz de diferencias. El objetivo es predecir un valor sumando la noticia de la matriz de diferencia y sumándole la puntuación del usuario actual. Después este valor será multiplicado por el número de ocurrencias de esa puntuación en la matriz de frecuencias.

```
for (Map.Entry<User, Map<Contribution, Double>> entry : inputData.entrySet()) {
    for (Contribution j : entry.getValue().keySet()) {
        for (Contribution k : diff.keySet()) {
            try {
                double predictedValue = diff.get(k).get(j) + entry.getValue().get(j);
                double finalValue = predictedValue * freq.get(k).get(j);
                uPred.put(k, uPred.get(k) + finalValue);
                uFreq.put(k, uFreq.get(k) + freq.get(k).get(j));
            } catch (NullPointerException ignored) {

            }
        }
    }
```

Figura 35: Predicción de puntuaciones no existentes.

Finalmente hay que limpiar un poco el resultado para obtener los datos relevantes, es decir solo noticias que tienen una frecuencia superior a 0 y que no las había puntuado previamente el usuario.

```
HashMap<Contribution, Double> clean = new HashMap<>();
for (Contribution c : uPred.keySet()) {
    if (uFreq.get(c) > 0) {
        clean.put(c, uPred.get(c) / uFreq.get(c));
    }
}

for (Contribution c : dm.getContributions()) {
    if (entry.getValue().containsKey(c)) {
        clean.put(c, entry.getValue().get(c));
    } else if (!clean.containsKey(c)) {
        clean.put(c, -1.0);
    }
}

outputData.put(entry.getKey(), clean);
```

Figura 36: Limpieza de datos a devolver.

Por último, se debe filtrar los datos para eliminar aquellas noticias que no tengan un valor de predicción superior a 2.5, ya que serán noticias que no le interesarán al usuario. Además, se ordenan las noticias por fecha de publicación utilizando un comparador interno de la clase Contribution que compara las fechas de publicación.

```
Map<Contribution, Double> recommendationMatrixOfUser = predictRecommendations().get(user);
Map<Contribution, Double> ratingsOfTheUser = dm.getData().get(user);
recommendationMatrixOfUser.values().removeIf(val -> val <= 2.5);

for (Map.Entry<Contribution, Double> entry : ratingsOfTheUser.entrySet()) {
    recommendationMatrixOfUser.remove(entry.getKey());
}

List<Contribution> recommendationList = new ArrayList<>(recommendationMatrixOfUser.keySet());
recommendationList.sort(Contribution.contributionComparator);
return recommendationList;
```

Figura 37: Filtrado de noticias y ordenación.

9. Testing

En este capítulo se explicarán los procesos seguidos para testear, tanto en backend como en frontend y que tipos de testeos se han realizado.

En backend se ha seguido la metodología TDD que es un desarrollo guiado por pruebas de software. Se basa en primero escribir los tests y después escribir el código. La idea es que a medida que vamos creando el test, se está pensando en cómo implementar la funcionalidad y por tanto se acaba produciendo un código de mayor calidad.

Para el frontend es un poco más complicado seguir esta metodología, especialmente si el programador tiene poca experiencia programando frontend, como es mi caso. Por ello, como excepción, en este caso, se creó primero el código y después se crearon los tests.

9.1 Testing en backend mediante JUnit

JUnit [24] es un conjunto de bibliotecas que se utilizan en programación para hacer pruebas unitarias de aplicaciones en Java y más recientemente, también en Kotlin. Para las pruebas en código ha sido la herramienta elegida ya que se pueden hacer todo tipo de pruebas, son fáciles de utilizar y es el estandard desde hace muchos años.

Considero el uso de testing muy recomendado, especialmente cuando se añaden funcionalidades nuevas que interfieren con código que se hizo anteriormente. En esta situación, basta con volver a ejecutar los tests ya creados para comprobar que todo sigue funcionando correctamente.

Utilizando JUnit he hecho testing de tipo unitario a nivel de service y test de controlador mockeando el service con Mockito [25].
El testing unitario es útil para comprobar que una determinada funcionalidad se comporta de la manera esperada.
El test de controlador sirve para comprobar que el comportamiento del controlador, sin tener en cuenta posibles dependencias, como por ejemplo el servicio de noticias en el caso de este proyecto, funciona como se espera.

En la siguiente figura se muestra un test de controlador donde se consiguen las noticias existentes en el servicio. Previamente se ha utilizado un método llamado *setUp* con la

anotación *@BeforeEach* que permite que este método se ejecute antes de cada test. Este método *setUp* se encarga de crear el objeto *contributionsPaged* que contiene dos noticias.

```
@Test
public void testRetrieveContributions() {
    when(contributionsService.getAll(paging)).thenReturn(contributionsPaged);
    Page<Contribution> contributionsResult = contributionController.getAllContributions(page);
    assertThat(contributionsResult.getTotalElements()).isEqualTo(2);
}
```

Figura 38: Test de controlador para obtener las noticias del sistema.

9.2 Testing manual en backend mediante Postman

Además del testing con JUnit, también es muy importante realizar pruebas con Postman [26] cuando se desarrollan APIs. Postman es una herramienta que permite realizar peticiones HTTP sin necesidad de desarrollar un cliente. Por lo tanto, podemos comprobar que los endpoints creados funcionan correctamente y devuelven la respuesta esperada. Es lo más cercano a un test de integración.

El funcionamiento es muy sencillo, en la aplicación se puede elegir el tipo de llamada a realizar (GET, POST, PUT…) y después se puede indicar el endpoint a probar escribiendo la url correspondiente. En caso de necesitar enviar parámetros a través del path podemos hacerlo sin problemas y también se puede añadir “body” (objetos JSON) en caso de que la petición así lo requiriese.

9.3 Testing de recomendaciones

El algoritmo de recomendaciones ha sido testeado igual que el servicio, utilizando JUnit, ofreciendo un input de noticias con una serie de puntuaciones por diferentes usuarios y comparando el resultado esperado con el devuelto por el algoritmo. En el siguiente test se definen dos usuarios que puntúan dos noticias el primero y tres el segundo. Finalmente se obtienen las noticias esperadas para el usuario con id = 100 que es principalmente la noticia con id = 3.

```java
@Test
public void testFindRecommendationOneExpected() {
    List<Contribution> constList = new ArrayList<>();
    constList.add(new Contribution( id: 1,  title: "Cont1",  pubDate: "1"));
    constList.add(new Contribution( id: 2,  title: "Cont2",  pubDate: "2"));
    constList.add(new Contribution( id: 3,  title: "Cont3",  pubDate: "3"));

    Map<User, Map<Contribution, Double>> inputData = new HashMap<>();
    Map<Contribution, Double> ratingsUser100 = new HashMap<>();
    ratingsUser100.put(constList.get(0), 3.0);
    ratingsUser100.put(constList.get(1), 5.0);
    inputData.put(new User( id: "100"), ratingsUser100);

    Map<Contribution, Double> ratingsUser200 = new HashMap<>();
    ratingsUser200.put(constList.get(0), 4.0);
    ratingsUser200.put(constList.get(1), 3.0);
    ratingsUser200.put(constList.get(2), 3.0);
    inputData.put(new User( id: "200"), ratingsUser200);

    when(dataModel.createDataModel()).thenReturn(inputData);
    when(dataModel.getData()).thenReturn(inputData);
    when(dataModel.getContributions()).thenReturn(new HashSet<>(constList));

    List<Contribution> conts = algorithm.findRecommendations(new User( id: "100"));
    List<Contribution> contsExpected = new ArrayList<>();
    contsExpected.add(constList.get(2));
    Assertions.assertEquals(conts, contsExpected);
}
```

Figura 39: Test de predicción de noticias

Además de este test, también se han realizado otros para comprobar que el algoritmo devuelve la puntuación predicha correctamente para cada noticia a cada usuario. En principio el servicio simplemente devuelve al frontend las noticias que le interesará a un usuario, en ningún momento devuelve el valor de predicción, aún así es importante testearlo para ver que realmente el algoritmo está funcionando como es debido.

9.4 Testing en frontend mediante Jest

Para realizar los tests en frontend se ha utilizado Jest, un framework creado en JavaScript para testear tanto visibilidad de elementos de una UI como comportamientos correctos de la misma.

Principalmente me he centrado en comprobar que los elementos son visibles cuando reciben datos, que tenían la información correcta y que los componentes se comportan correctamente en función de si el usuario está autenticado o si no lo esta.

En la siguiente figura se muestra un test donde se comprueba que una imagen es visible y otro test donde se comprueba que la imagen tiene el enlace correspondiente:

```
it('Images are visible', () => {
    const images = wrapper.findAll('[data-testid="news-image"]');
    images.map( (item) => {
        expect(item.isVisible()).toBe(true);
    });

});

it('Sources have the correct link', () => {
    const images = wrapper.findAll('[data-testid="news-image"]');
    contributionsMockData.map( (item, index) => {
        expect(images[index].attributes('src')).toEqual(item.urlImage);
    });
});
```

Figura 40: Testing con Jest en frontend

9.5 Testing de usabilidad

Previamente se ha mencionado como requisito no funcional la usabilidad. Considerando la importancia de la misma y que es muy subjetiva, he decidido realizar una pequeña prueba de usabilidad sencilla y compartirla con mi entorno más cercano para obtener su feedback y poder mejorar la aplicación.

Se ha realizado una prueba guiada con cada persona en la que inicialmente se ha cargado la página web y se les ha pedido que simplemente observen la interfaz y comenten de que creen que trata y para que sirve, además de cualquier opinión que se les pudiese ocurrir.

En general, la mayoría de personas han entendido rápidamente de que se trata al ver las noticias una detrás de otra y los diferentes enlaces. Inicialmente los iconos eran ligeramente diferentes a los actuales y algunas personas no lograban entender que indicaban. Gracias a estas indicaciones, se procedió a cambiarlos por unos iconos que destacan más, son más

grandes y son muy diferentes entre ellos para poder conseguir un mayor entendimiento de los mismos.

Después de esta primera prueba quería comprobar qué tan fácil es utilizar la página web y si los usuarios eran capaces de rápidamente saber como hacer ciertas tareas. Para ello, decidí asignarles tres tareas:

1. Intenta obtener más información de una noticia.
2. Intenta puntuar una noticia.
3. Intenta cargar más noticias.

En este caso, la respuesta fue muy positiva ya que todos entendieron que para obtener más información de una noticia bastaba con pulsar en el título de cualquiera de ellas y que para puntuarlas simplemente tenían que elegir una puntuación entre 1 y 5 pulsando las estrellas que simulan los números.
Sobre cargar más noticias, algunos usuarios menos acostumbrados al uso de este tipo de tecnología inicialmente se mostraron confusos al no ver enlaces para obtener las siguientes noticias o números al final de la página como por ejemplo hace Google. De todas formas, en poco tiempo comprobaron que a medida que hacían scroll se iban cargando más noticias y mostraron una reacción muy positiva al funcionamiento para cargar nuevas noticias.

9.6 Test de carga de recomendaciones

Actualmente cada vez que un usuario quiere ver sus recomendaciones, el sistema debe recalcular todas las recomendaciones en base a los usuarios y noticias puntuadas que existan en ese momento. Por tanto, a mayor número de usuarios y/o mayor número de noticias puntuadas, la velocidad en mostrar recomendaciones decrece de manera considerable.

Para comprobar cómo de eficiente es el sistema se ha elaborado la siguiente tabla que muestra el número de usuarios en el sistema y cuántas noticias ha puntuado cada uno (se ha supuesto que cada usuario puntúa el mismo número de noticias).

# Usuarios	# Noticias puntuadas	Tiempo (ms)
100	10	242
100	100	1250
1000	10	1725
1000	100	9229
10000	10	80107

Tabla 30: Tiempo en mostrar recomendaciones en función del número de usuarios

Se puede comprobar que a partir de unos 1000 usuarios con 100 noticias puntuadas cada uno, el sistema empieza a ser bastante lento, tardando hasta 9 segundos. Si incrementamos hasta 10000 usuarios con 10 noticias puntuadas cada uno, el sistema pasa a ser inutilizable ya que el tiempo de espera acaba siendo de 80 segundos.

Como prototipo considero que es aceptable, pero en el supuesto que se decidiese utilizar este sistema como un producto final, estos tiempos claramente deberían mejorar. Hay varias formas de conseguir esto, una muy evidente es tener previamente calculadas las noticias recomendadas para cada usuario en la base de datos y solo cuando un usuario puntúe una noticia, actualizar este listado. De esta manera conseguimos que cada vez que el usuario accede a sus recomendaciones, simplemente se haga una query a la base de datos y se devuelva su listado.
Esta solución tiene dos inconvenientes, el primero es que el espacio en base de datos incrementa considerablemente. El segundo es que en el momento que existan muchos usuarios y noticias puntuadas, al puntuar una nueva el recálculo de las noticias será igual de lento que en el caso actual.
Para solventar este segundo problema, se puede llegar a pensar el recalculado de noticias de manera que solo se recalculen las que afectan directamente al usuario que ha puntuado y las noticias que tenía puntuadas y por tanto se simplifica mucho el recálculo de recomendaciones.

10. Conclusiones

El objetivo principal de este proyecto era conseguir integrar un servicio de noticias con un sistema de recomendaciones y posteriormente mostrarlo a través de un sitio web. En algún momento me planteé incluso desarrollar una aplicación android además del sitio web para darle más valor al producto y realmente no hubiese sido complicado ya que el diseño del servicio se pensó de un inicio para que fuese totalmente independiente y las vistas no ofrecen ningún tipo de lógica.
Finalmente esta idea de la aplicación no la llegué a realizar ya que pese a que todo el trabajo de la lógica ya estaba hecho, hacer una aplicación lleva su tiempo y no estoy seguro de haber podido llegar a tiempo.

Estoy satisfecho tanto con el desarrollo del proyecto como con el resultado. Desde un inicio he intentado seguir la metodología Agile y el uso de TDD para el backend y he conseguido llevarlo a cabo con un resultado muy positivo.

10.1 Aprendizaje

Como ya se ha comentado en capítulos anteriores, he utilizado diversas herramientas y frameworks en el proyecto para desarrollarlo. Si bien es cierto que algunas ya las conocía, especialmente Spring framework, tampoco tenía un conocimiento muy profundo. El desarrollo de este proyecto me ha permitido mejorar mucho en este framework y es algo de lo que me alegro ya que era algo en lo que quería mejorar desde hacía tiempo.

Por otro lado, hay tecnologías con las que ni siquiera había programado nunca antes como son Vue y OAuth2 con Google. En el caso de Vue, decidí utilizar este framework para frontend ya que me parece bastante innovador y queria saber como funcionaba en un proyecto real. Estoy satisfecho con cómo funciona y con el resultado que he podido crear. Me parece un framework muy útil para el desarrollo web y que no tiene nada que envidiar a otros más conocidos como Angular o React.
Sobre OAuth2 si que alguna vez había tocado alguna cosa, pero nada similar a este proyecto donde he tenido que integrar tanto la parte del backend como la del frontend y que funcionen entre ellas.

En general, estoy contento con el stack tecnológico escogido, pero si hubiese tenido un poco más de tiempo, si que me hubiese gustado llegar a integrar esto en una aplicación además de en un sitio web.

10.1 Integración de conocimientos

En este proyecto se integran conocimientos de diversas asignaturas realizadas a lo largo de la carrera, especialmente, asignaturas de la especialidad de ingeniería del software.

Una de las más importantes es Aplicaciones y Servicios Web (ASW) ya que el proyecto se basa en la creación de una API Rest para posteriormente ser consumida por una página web utilizando el framework de Vue. En esta asignatura aprendí entre otras cosas, a utilizar APIs Rest en profundidad y a integrarla con los frameworks de Javascript más populares actualmente.

Otras dos asignaturas importantes han sido Proyecto de Ingeniería del Software (PES) y Gestión de Proyectos Software (GPS) que han sido importantes para saber planificar correctamente un proyecto y la gestión del mismo durante el desarrollo.

Arquitectura del Software (AS) también ha sido importante para poder diseñar sistemas complejos de la manera más óptima posible.

Ingeniería de Requisitos (ER) ha sido una asignatura útil de cara a saber analizar los requisitos de un proyecto.

Finalmente, Bases de Datos (BD) y Diseño de Bases de Datos (DBD) también han sido útiles para saber tanto utilizar correctamente una base de datos cómo para saber diseñarla en base a los requisitos del sistema.

10.2 Competencias técnicas

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos. [Un poco]

Para integrar el sistema con un recomendador de noticias se ha tenido que pensar previamente cómo diseñarlo acorde a las necesidades propias de un algoritmo de filtrado colaborativo.

CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles. [En profundidad]

Para desarrollar el sistema se han utilizado tecnologías web en el Backend y en el Frontend y se han conectado mediante el uso de una RESTful API que el Frontend consume del Backend. Además, se ha utilizado Heroku como plataforma de alojamiento para alojar ambos proyectos. Por último, como sistema de almacenamiento de datos se ha utilizado PostgreSQL y como autenticación se ha usado OAuth2 con Google.

CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que pudiesen presentarse. [Bastante]

Durante el desarrollo del proyecto han habido diferentes riesgos a los que se ha tenido que dar respuesta. Por ejemplo inicialmente se pensaba utilizar APIs de terceros para obtener noticias y rápidamente se vio que no era viable y finalmente se ha utilizado RSS de periódicos nacionales. Por otro lado, implementar OAuth2 también ha sido un poco complicado e incluso ha sido necesario replantear la implementación en una ocasión para poder implementarlo.

CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de red. [En profundidad]

Para poder comunicar el backend y el frontend a través de peticiones HTTP, teniendo alojados ambos proyectos en Heroku. Para la comunicación se ha utilizado el backend como una API Rest a la que el frontend enviaba peticiones mediante JSON y recibe el resultado con el mismo formato.

CES1.5: Especificar, diseñar, implementar y evaluar bases de datos. [Bastante]

Se ha tenido en cuenta para el diseño de la base de datos en PostgreSQL conocimientos de especificación y diseño para realizar una base de datos óptima con el uso del sistema. Para la implementación se ha utilizado Hibernate integrado en Spring que facilita el acceso a los datos.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. [Un poco]

Se ha utilizado TDD en el backend y se ha testeado usando JUnit utilizando tanto tests de controlador mockeando el servicio para testear el comportamiento del controlador y tests a nivel de servicio mockeando el repositorio para testear el comportamiento del servicio independientemente del repositorio.
Para el frontend se ha testeado usando Jest y se han realizado tests para controlar que cada componente renderizaba datos en el momento que debía y además que renderizaba los datos que debian ser visibles.

CES2.1: Definir y gestionar los requisitos de un sistema software. [Bastante]

Inicialmente se ha pensado en cómo diseñar el sistema y que funcionalidades debía tener para tener una idea general y saber como empezar el proyecto de manera correcta. A medida que se ha ido desarrollando, se ha priorizado entre tareas y se ha seguido la metodología Agile para ir generando valor a medida que el producto avanza.

10.3 Trabajo futuro

A pesar de que las funcionalidades definidas en el proyecto han sido implementadas, aún quedan diversas mejoras que me gustaría implementar.

Como ya se ha comentado, me gustaría haber podido desarrollar una aplicación Android y es algo que en el futuro acabaré haciendo ya que creo que aporta un gran valor y hoy en día, gran parte de los usuarios utilizan el móvil.

Sobre periódicos, actualmente solo hay 4 periódicos de los que se obtienen noticias. En el futuro me gustaría añadir más periódicos e incluso poder añadir periódicos internacionales y por tanto noticias en diferentes idiomas. Para realizar algo así, sería necesario el uso de filtros en el frontend para inicialmente poder filtrar por idioma o por periodico. En caso de que el usuario llegase a utilizar el sistema tal y como ha sido pensado, es decir, valorando noticias, llegaría un punto que no necesitaría los filtros, ya que probablemente se le recomendaría noticias de sus periódicos favoritos y su idioma de preferencia.

11. Leyes y regulaciones

11.1 Leyes

En este proyecto se identifican dos posibles problemas respecto a leyes y regulaciones, por un lado está la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales y por otro, el uso de noticias por parte de terceros.

Sobre el primer punto, en este proyecto se almacena la ID de una cuenta de Google y la relación que tendrán con cada noticia puntuada. En ningún caso se almacenará información sensible como la cuenta de correo, nombre y apellidos o cualquier otro tipo de información que pudiese identificar a la persona. Además, los datos son completamente privados y no se podrá acceder a los datos de otra persona (especialmente las noticias puntuadas) bajo ningún concepto.

Sobre el segundo aspecto, en este proyecto no se plantea obtener ningún beneficio económico ni la utilización de publicidad en el sitio, por lo tanto, tal y como informan los cuatro periódicos de los que se obtienen noticias (La Vanguardia, ABC, El País y El Mundo) no hay ningún tipo de problema legal.

En caso de querer obtener un beneficio económico (ya sea con publicidad o de cualquier otro tipo) al mostrar sus noticias, sería necesario contactar con ellos y llegar a un acuerdo. De nuevo, no es el caso de este proyecto (y además se trata de un prototipo sin intención de difusión por el momento) y por tanto no hay ningún inconveniente legal.

A modo de comprobación, se expone a continuación la normativa de RSS de cada periodico:

11.1.1 ABC

ABC en su sección RSS [27] expone que las explotaciones comerciales o ligadas a servicios de búsqueda están sujetas a autorización y retribución. En el caso de este proyecto, no es problema ya que no se está utilizando con fin comercial.

11.1.2 El Mundo

El Mundo en su sección RSS [28] da permiso explícito para integrar ya sea tanto en web como en un cliente RSS las noticias que proporcionan de forma completamente gratuita. De hecho, incluso incitan a utilizar su contenido con la siguiente frase: *“Es más, le proponemos el reto de desarrollar herramientas con nuestro contenido.”.*

Existen varios requisitos para utilizar los contenidos de El Mundo y pueden ser encontrados en su página de “ideas” [29]:

- La utilización de los contenidos debe ser para uso personal y no podrán emplearse con fines comerciales.
- No se podrán vender aplicaciones y/o servicios que usen o incorporen su contenido.

11.1.3 La Vanguardia y El País

En su sección de RSS no dan permiso explícito para enlazar sus contenidos mediante RSS. En el aviso legal de ambos sitios web se expone claramente que no permiten la reproducción y distribución de sus contenidos sin autorización previa.
Aún así, este TFG es un prototipo o prueba de concepto y por tanto por el momento no hay ningún problema. Si en el futuro se decidiese seguir adelante con el proyecto, sería necesario ponerse en contacto con dichos periódicos para obtener su autorización. En el peor de los casos, se debería eliminar todas las referencias a dichos periódicos.

12. Glosario

Api: La interfaz de programación de aplicaciones, conocida también por la sigla API, en inglés, application programming interface es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Aplicación: Tipo de programa informático diseñado como herramienta, para permitir a un usuario realizar diversos tipos de trabajos.

Backend: Es la parte o rama del desarrollo web encargada de que toda la lógica de una página funcione. Consiste en el conjunto de acciones que pasan dentro de una web, pero que no podemos ver. Un ejemplo de esto es la comunicación con el servidor.

Feed: Los feeds son flujos de contenido por los que los usuarios pueden desplazarse. El contenido aparece en bloques parecidos que se repiten uno después del otro.

Feedback: Feedback es una palabra del inglés que significa retroalimentación; podemos utilizarla como sinónimo de respuesta o reacción, o, desde un punto de vista más técnico, para referirnos a un método de control de sistemas.

Frontend: El desarrollo web Front-end consiste en la conversión de datos en una interfaz gráfica para que el usuario pueda ver e interactuar con la información de forma digital usando HTML, CSS y JavaScript

Oauth2: Open Authorization es un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas

Prototipo: Un prototipo es un ejemplo del primer molde fabricado o una figura u otra cosa. Un prototipo perfecto y modelo de una virtud, vicio o cualidad.

Scroll: Se denomina desplazar o deslizar, al movimiento en 2D de los contenidos que conforman el escenario de un videojuego o la ventana que se muestra en una aplicación informática.

13. Bibliografía

[1] “Goodbye information overload,” *Feedly*, fecha de consulta 9 febrero 2021, en https://feedly.com/.

[2] “*Reddit”*, fecha de consulta 9 febrero 2021, en https://www.reddit.com/.

[3] “*Twitter”*, Twitter, fecha de consulta 9 febrero 2021, en https://twitter.com/.

[4] “Qué es SCRUM,” *Proyectos Ágiles*, 2018, fecha de consulta 24 febrero 2021, en https://proyectosagiles.org/que-es-scrum/.

[5] Atlassian, “Jira: Software de seguimiento de proyectos e incidencias,” *Atlassian*, fecha de consulta 24 febrero 2021, en https://www.atlassian.com/es/software/jira.

[6] *Git*, fecha de consulta 24 febrero 2021, en https://git-scm.com/.

[7] “Where the world builds software,” *GitHub*, fecha de consulta 24 febrero 2021, en https://github.com/.

[8] *Introduction to Test Driven Development (TDD)*, fecha de consulta 24 febrero 2021, en http://agiledata.org/essays/tdd.html.

[9] “Búsqueda de empleo en Glassdoor | Encuentra el empleo ...,” , fecha de consulta 28 febrero 2021, en https://www.glassdoor.es/index.htm.

[10] “Agencia Estatal Boletín Oficial del Estado,” fecha de consulta 28 febrero 2021, en https://www.boe.es/diario_boe/txt.php?id=BOE-A-2020-1626.

[11] “Meet BCN: Tu Espacio de Coworking en el Eixample de Barcelona,” *Meet BCN - Coworking Barcelona*, 2020, fecha de consulta 28 febrero 2021, en https://meetbcn.com/.

[12] “Volere Requirements Specification Template,” *Volere Requirements*, fecha de consulta 10 mayo 2021, en https://www.volere.org/templates/volere-requirements-specification-template/.

[13] “Heroku,” *Cloud Application Platform*, fecha de consulta 16 mayo 2021, en https://www.heroku.com/home.

[14] Group, P. S. Q. L. G. D., *PostgreSQL*, 2021, fecha de consulta 16 mayo 2021, en https://www.postgresql.org/.

[15] Garbar, D., “React vs Angular vs Vue,” Belitsoft, 2020, fecha de consulta 21 abril 2021, en https://belitsoft.com/front-end-development-services/react-vs-angular.

[16] Oblancarte, “Data Transfer Object (DTO) – Patrón de diseño,” *Oscar Blancarte - Software Architecture*, 2020, fecha de consulta 16 mayo 2021, en https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/.

[17] Oblancarte, “Data Access Object (DAO) Pattern,” *Oscar Blancarte - Software Architecture*, 2018, fecha de consulta 16 mayo 2021, en https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/.

[18] Staff, E., “What the difference between REST API and RESTful API?,” *DataMounts*, 2020, fecha de consulta 16 mayo 2021, en https://www.datamounts.com/difference-rest-api-restful-api/.

[19] Baeldung, “A Collaborative Filtering Recommendation System in Java,” *Baeldung*, 2020, fecha de consulta 21 mayo 2021, en https://www.baeldung.com/java-collaborative-filtering-recommendations.

[20] “Spring makes Java simple.,” *Spring*, fecha de consulta 16 mayo 2021, en https://spring.io/.

[21] *Vue.js*, fecha de consulta 16 mayo 2021, en https://vuejs.org/.

[22] “Integrating Google Sign-In into your web app,” *Google*, Google, fecha de consulta 16 mayo 2021, en https://developers.google.com/identity/sign-in/web/sign-in.

[23] “Jest · Delightful JavaScript Testing,” *Jest Blog RSS*, fecha de consulta 17 mayo 2021, en https://jestjs.io/.

[24] “The 5th major version of the programmer-friendly testing framework for Java and the JVM,” *JUnit 5*, fecha de consulta 17 mayo 2021, en https://junit.org/junit5/.

[25] *Mockito framework site*, fecha de consulta 17 mayo 2021, en https://site.mockito.org/.

[26] “The Collaboration Platform for API Development,” *Postman*, fecha de consulta 17 mayo 2021, en https://www.postman.com/.

[27] “Noticias RSS de España y del mundo,” *ABC.es*, fecha de consulta 21 abril 2021, en https://www.abc.es/rss/.

[28] *RSS en elmundo.es*, fecha de consulta 21 abril 2021, en http://rss.elmundo.es/rss/.

[29] *Ideas para un nuevo milenio // elmundo.es*, fecha de consulta 21 abril 2021, en https://www.elmundo.es/imasd/explica/rss/ideas.html.

Anexo: Documentación de peticiones Rest API

A.1. Login

Permite que un usuario inicie sesión en el sistema a través de Google y pueda realizar posteriormente peticiones que requieran un usuario registrado.

Petición:

- Método: POST
- Endpoint: /login

Parámetros:

Tipo	Nombre	Tipo de dato
HEAD	X-ID-TOKEN	String

Respuesta:

200

Devuelve el parámetro **Authorization** en el Response Header que servirá para autenticarse en las posteriores llamadas.

401

```
{
   "status": “UNAUTHORIZED”,
    "timestamp": "04-06-2021 03:28:15",
    "message": "Invalid login."
}
```

A.2. Obtener últimas noticias

Permite obtener las últimas noticias añadidas al sistema.

Petición:

- Método: GET
- Endpoint: /api/latest

Parámetros:

Tipo	Nombre	Tipo de dato
URL PARAM	Page	Int

Respuesta:

200

```
{
   "content": [
      {
         "id": 14131,
         "title": "A news title",
         "description": "A news description",
         "link": "https://www.google.com",
         "categories": [
            "Sports"
         ],
         "pubDate": "2021-06-04 14:40",
         "creator": "Creator",
         "urlImage": "https://www.image.com",
         "source": "La Vanguardia",
         "sourceUrl": "www.lavanguardia.com",
         "vote": null
      },
      {
         "id": 14130,
         "title": "A news title",
         "description": "A news description",
         "link": "https://www.google.com",
         "categories": [
            "Sports"
         ],
         "pubDate": "2021-06-04 14:40",
         "creator": "Creator",
```

```
            "urlImage": "https://www.image.com",
            "source": "La Vanguardia",
            "sourceUrl": "www.lavanguardia.com",
            "vote": null
        }
    ],
    "pageable": {
        "sort": {
            "sorted": false,
            "unsorted": true,
            "empty": true
        },
        "pageNumber": 1,
        "pageSize": 10,
        "offset": 10,
        "unpaged": false,
        "paged": true
    },
    "last": false,
    "totalPages": 1404,
    "totalElements": 14034,
    "first": false,
    "sort": {
        "sorted": false,
        "unsorted": true,
        "empty": true
    },
    "numberOfElements": 10,
    "size": 10,
    "number": 1,
    "empty": false
}
```

400

```
{
   "status": “BAD_REQUEST”,
    "timestamp": "04-06-2021 03:40:15",
    "message": "Param. page is invalid."
}
```

A.3. Obtener noticias recomendadas

Permite obtener las noticias recomendadas de un usuario

Petición:

- Método: GET
- Endpoint: /api/recommendations

Parámetros:

Tipo	Nombre	Tipo de dato
HEAD	Authorization	String
URL PARAM	Page	Int

Respuesta:

200

```
{
   "content": [
      {
         "id": 234,
         "title": "A title of a news recommended",
         "description": "A description of a news recommended",
         "link": "https://www.google.com",
         "categories": [
            "Sports"
         ],
         "pubDate": "2021-03-01 17:30",
         "creator": "Creator",
         "urlImage": "https://www.image.com",
         "source": "La Vanguardia",
         "sourceUrl": "www.lavanguardia.com",
         "vote": null
      }
     ...
   ],
   "pageable": {
      "sort": {
         "sorted": false,
         "unsorted": true,
         "empty": true
      },
```

```
        "pageNumber": 1,
        "pageSize": 10,
        "offset": 10,
        "unpaged": false,
        "paged": true
    },
    "last": false,
    "totalPages": 9,
    "totalElements": 90,
    "first": false,
    "sort": {
        "sorted": false,
        "unsorted": true,
        "empty": true
    },
    "numberOfElements": 10,
    "size": 10,
    "number": 1,
    "empty": false
}
```

400

```
{
   "status": “BAD_REQUEST”,
    "timestamp": "04-06-2021 03:45:15",
    "message": "Param. page is invalid."
}
```

401

```
{
    "status": "UNAUTHORIZED",
    "timestamp": "04-06-2021 03:53:36",
    "message": "Log in to access to this resource"
}
```

A.4. Obtener noticias puntuadas

Permite obtener las noticias puntuadas del usuario.

Petición:

- Método: GET
- Endpoint: /api/rated

Parámetros:

Tipo	Nombre	Tipo de dato
HEAD	Authorization	String
URL PARAM	Page	Int

Respuesta:

200

```
{
    "content": [
        {
            "id": 983,
            "title": "A news title",
            "description": "A news description",
            "link": "https://www.google.com",
            "categories": [
                "Sports"
            ],
            "pubDate": "2021-03-10 19:29",
            "creator": "Creator",
            "urlImage": "https://www.image.com",
            "source": "La Vanguardia",
            "sourceUrl": "www.lavanguardia.com",
            "vote": 5
        },
        ...
    ],
    "pageable": {
        "sort": {
            "sorted": false,
            "unsorted": true,
            "empty": true
        },
```

```
        "pageNumber": 1,
        "pageSize": 10,
        "offset": 10,
        "unpaged": false,
        "paged": true
    },
    "last": false,
    "totalPages": 3,
    "totalElements": 30,
    "first": false,
    "sort": {
        "sorted": false,
        "unsorted": true,
        "empty": true
    },
    "numberOfElements": 10,
    "size": 10,
    "number": 1,
    "empty": false
}
```

400

```
{
   "status": “BAD_REQUEST”,
    "timestamp": "04-06-2021 03:48:19",
    "message": "Param. page is invalid."
}
```

401

```
{
    "status": "UNAUTHORIZED",
    "timestamp": "04-06-2021 03:53:36",
    "message": "Log in to access to this resource"
}
```

A.5. Obtener detalles de noticia

Permite obtener las noticias puntuadas del usuario.

Petición:

- Método: GET
- Endpoint: /api/contributions

Parámetros:

Tipo	Nombre	Tipo de dato
HEAD	Authorization	String
URL PARAM	Id	Int

Respuesta:

200

```
{
    "id": 18,
    "title": "Title",
    "description": "Description",
    "link": "https://www.google.com",
    "categories": [
        "Programming"
    ],
    "pubDate": "2021-04-15 11:12",
    "creator": "Creator",
    "urlImage": "https://www.image.com",
    "source": "ABC",
    "sourceUrl": "www.abc.es",
    "vote": null
}
```

400

```
{
   "status": “BAD_REQUEST”,
    "timestamp": "04-06-2021 03:48:19",
```

```
    "message": "Param. page is invalid."
}
```

404

```
{
    "status": "NOT_FOUND",
    "timestamp": "04-06-2021 03:51:24",
    "message": "No value present"
}
```

A.6. Puntuar una noticia

Permite puntuar una noticia.

Petición:

- Método: POST
- Endpoint: /api/contributions

Parámetros:

Tipo	Nombre	Tipo de dato
HEAD	Authorization	String
URL PARAM	Id	Int

Respuesta:

200

400

```
{
   "status": “BAD_REQUEST”,
    "timestamp": "04-06-2021 03:48:19",
    "message": "Param. id is invalid."
}
```

401

```
{
    "status": "UNAUTHORIZED",
    "timestamp": "04-06-2021 03:53:36",
    "message": "Log in to access to this resource"
}
```

404

```
{
    "status": "NOT_FOUND",
    "timestamp": "04-06-2021 03:51:24",
    "message": "No value present"
}
```

A.7. Actualizar puntuación de noticia

Permite actualizar la puntuación de una noticia.

Petición:

- Método: PUT
- Endpoint: /api/contributions

Parámetros:

Tipo	Nombre	Tipo de dato
HEAD	Authorization	String
URL PARAM	Id	Int

Respuesta:

200

400

```
{
   "status": “BAD_REQUEST”,
    "timestamp": "04-06-2021 03:48:19",
    "message": "Param. id is invalid."
}
```

401

```
{
    "status": "UNAUTHORIZED",
    "timestamp": "04-06-2021 03:53:36",
    "message": "Log in to access to this resource"
}
```

404

```
{
    "status": "NOT_FOUND",
    "timestamp": "04-06-2021 03:51:24",
    "message": "No value present"
}
```
