



TRABAJO FIN DE GRADO  
INGENIERÍA EN INFORMÁTICA

# Gestor de Trending Topics y Noticias

---

Estudio y Visualización de Trends de Twitter

**Autor**

Nikita Stetskiy Stetskiy (alumno)

**Directores**

Juan Francisco Huete Guadix (tutor)  
Carlos Alberto Cruz Corona (cotutor)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, 29 de octubre de 2022



**NIKITA STETSKIY STETSKIY (ALUMNO):**

*Gestor de Trending Topics y Noticias*

*Estudio y Visualización de Trends de Twitter.*

Supervisado por:

**JUAN FRANCISCO HUETE GUADIX (TUTOR)**

**CARLOS ALBERTO CRUZ CORONA (COTUTOR)**

Granada, 29 de octubre de 2022



## DECLARACIÓN

---

D. **Juan Francisco Huete Guadix (tutor)**, Profesor del Área de XXXX del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

D. **Carlos Alberto Cruz Corona (cotutor)**, Profesor del Área de XXXX del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado *Gestor de Trending Topics y Noticias, Estudio y Visualización de Trends de Twitter*, ha sido realizado bajo su supervisión por **Nikita Stetskiy Stetskiy (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 29 de octubre de 2022.

**Los directores:**

**Juan Francisco Huete Guadix (tutor)**  
**Carlos Alberto Cruz Corona (cotutor)**



## RESUMEN

---

**PALABRAS CLAVE:** Aplicación Web, Arquitectura de capas, Modelado API REST, Trending Topics, Noticias, Diseño Responsive.

...

El principal enfoque de este Trabajo de Fin de Grado se basa en la aportación de información útil, resumida y de fácil alcance para el usuario final, sin que esta sea abrumante y difícil de leer.

A día de hoy, las aplicaciones web sociales más dominantes del mercado intentan captar y mantener al usuario el mayor tiempo posible. Lo consiguen aportando cantidad ingente o casi infinita de información. Esta información se actualiza en cada momento, dando la posibilidad de mantenernos partícipes del mundo social en cada instante.

Dada la atosigante abundancia de información he decidido hacer una aplicación web que proporcione información de una manera mucho más simple sobre tendencias actuales. De esta manera, los usuarios no tendrán que estar tan pendientes en mantenerse actualizados, sino que al final del día tendrán simplificada o resumida la información más destacable y otros datos útiles en base a la tendencia.

El proyecto involucra todos los conocimientos, tanto en la ingeniería informática como fuera de ella, que he podido adquirir a lo largo de mi ciclo de aprendizaje del ámbito informático.

Entrando más en detalle en los conocimientos aplicados, el proyecto está compuesto por una arquitectura de tres capas: la interfaz de usuario, una capa de gestión y otra de datos. Las tecnologías usadas se pueden resumir en JavaScript, Python y NoSQL. JavaScript, en concreto Vue.js para el apartado Front-end. Para el Back-end, Python como el gestor o el procesador y MongoDB como el sistema de base de datos.

Al combinar estas tecnologías, el modelo que se termina usando es la transferencia de estado representacional o API REST, en el cual se usarán distintos enrutadores y peticiones web, pero que, de cara al usuario final, seguirá siendo una única aplicación.



## ABSTRACT

---

**KEYWORDS:** Web Application, Layered Architecture, REST API Modelling, Trending Topics, News, Responsive Design.

...

The main focus of this Final Degree Project is based on providing useful information, summarized and easily accessible to the end user, without it being overwhelming and difficult to read.

To this day, the most dominant social web applications on the market try to engage and keep the user as long as possible. They achieve this by providing a huge or almost infinite amount of information. This information is updated everytime, giving the possibility of staying involved in the social world almost forever.

Given the overwhelming abundance of information, I have decided to make a web application that provides you with information in a much simpler way about current trends. In this sense, users will not have to be so focused on keeping up to date, but at the end of the day they will have summarized the most valuable information and useful data based on the trend.

The project involves all the knowledge, both in computer engineering and outside of it, that I was able to acquire throughout my learning cycle in the computer field.

Going into more detail in the applied knowledge, the project is made up of a three-layer architecture: the user interface, a management layer and a data layer. The technologies used can be summarized in JavaScript, Python and NoSQL. JavaScript, specifically Vue.js for the Front-end section. For the Back-end, Python as the manager or processor and MongoDB as the database system.

By combining these technologies, the model that ends up being used is representational state transfer or REST API, in which different routes and web requests will be used, but which will continue to be a single application for the final user.



## AGRADECIMIENTOS

---

Agradecimientos...

Agradecimientos



# ÍNDICE GENERAL

---

## I INTRODUCCIÓN

1	CONTEXTO	3
1.1	Motivación o justificación del proyecto . . . . .	3
1.2	Objetivos . . . . .	4
1.2.1	Objetivo general . . . . .	4
1.2.2	Objetivos específicos . . . . .	4
1.3	Sección en proceso !!! . . . . .	5
2	PLANIFICACIÓN	7
2.1	Planificación inicial . . . . .	7
2.2	Planificación detallada . . . . .	8
2.3	Presupuesto . . . . .	10
3	ESTADO DEL ARTE	11
3.1	Dominio del problema presentado . . . . .	11
3.1.1	Causas del problema presentado . . . . .	12
3.1.2	Proceso gradual del problema . . . . .	16
3.1.3	Consecuencias producidas en base al problema	19
3.2	Conclusión elaborada en base al problema . . . . .	20

## II PROPUESTA

4	ANÁLISIS	25
4.1	Metodología de desarrollo . . . . .	25
4.1.1	Metodología de trabajo escogida (SCRUM) . . .	26
4.1.2	Adaptación de la metodología de trabajo escogida	28
4.2	Análisis del entorno . . . . .	29
4.2.1	Estudio competitivo . . . . .	31
4.2.2	Estudio competitivo técnico . . . . .	32
4.2.3	Conclusiones de los estudios realizados . . . .	34
4.3	Requisitos del sistema . . . . .	36
4.3.1	Requisitos Funcionales . . . . .	36
4.3.2	Requisitos No Funcionales . . . . .	39
4.4	Product Backlog . . . . .	40
4.5	Sprint Backlog . . . . .	42
4.5.1	Sprint 1 . . . . .	42
4.5.2	Sprint 2 . . . . .	44
4.5.3	Sprint 3 . . . . .	45
4.5.4	Sprint 4 . . . . .	47
4.5.5	Sprint 5 . . . . .	48
4.5.6	Sprint 6 . . . . .	49
4.5.7	Sprint 7 . . . . .	51
4.5.8	Sprint 8 . . . . .	54
4.5.9	Sprint 9 . . . . .	56

<b>5 DISEÑO</b>	<b>57</b>
5.1 Arquitectura planteada . . . . .	57
5.2 Capa de Base de Datos . . . . .	58
5.2.1 Estudio técnico . . . . .	59
5.2.2 Modelado de datos . . . . .	60
5.3 Capa de Dominio o Negocio . . . . .	63
5.3.1 Estudio técnico . . . . .	63
5.3.2 Diagramas de interacción de distintas rutas . .	66
5.4 Capa de Presentación . . . . .	69
5.4.1 Estudio técnico . . . . .	69
5.4.2 Transcurso de la Interfaz de Usuario . . . . .	71
5.4.3 Vista Principal . . . . .	73
5.4.4 Subvista Popularidad . . . . .	74
5.4.5 Subvista Palabras más Comunes y Keywords .	75
5.4.6 Subvista Sentimiento General . . . . .	76
5.4.7 Subvista Noticas Relacionadas . . . . .	76
<b>6 TERMINOLOGÍA DEVOPS</b>	<b>79</b>
6.0.1 Implementación Back-end . . . . .	80
6.0.2 Implementación Front-end . . . . .	80
<b>BIBLIOGRAFÍA</b>	<b>81</b>

## LISTADO DE FIGURAS

---

Figura 2.1	Planificación inicial del proyecto . . . . .	7
Figura 2.2	Planificación detallada del proyecto . . . . .	9
Figura 5.1	Diagrama conceptual de la arquitectura . . . . .	58
Figura 5.2	Diagrama conceptual con más detalle (1) . . . . .	59
Figura 5.3	Diagrama de relación de la base de datos . . . . .	60
Figura 5.4	Diagrama de relación de Tendencia . . . . .	61
Figura 5.5	Diagrama de relación de Popularidad . . . . .	61
Figura 5.6	Diagrama de relación de Palabras más Comunes y Keywords . . . . .	62
Figura 5.7	Diagrama de relación de Sentimiento General	62
Figura 5.8	Diagrama de relación de Noticas Relacionadas	63
Figura 5.9	Documentación generada por FastAPI . . . . .	65
Figura 5.10	Diagrama conceptual con más detalle (2) . . . . .	66
Figura 5.11	Diagrama interacción de rutas (1) . . . . .	66
Figura 5.12	Diagrama interacción de rutas (2) . . . . .	66
Figura 5.13	Diagrama interacción de rutas (3) . . . . .	67
Figura 5.14	Diagrama interacción de rutas (4) . . . . .	67
Figura 5.15	Diagrama interacción de rutas (5) . . . . .	67
Figura 5.16	Diagrama interacción de rutas (6) . . . . .	68
Figura 5.17	Diagrama interacción de rutas (7) . . . . .	68
Figura 5.18	Diagrama interacción de rutas (8) . . . . .	68
Figura 5.19	Diagrama conceptual con más detalle (3) . . . . .	69
Figura 5.20	Diagrama de actividad . . . . .	70
Figura 5.21	Formularios de selección y búsqueda . . . . .	71
Figura 5.22	Primer boceto de la aplicación . . . . .	71
Figura 5.23	Segundo boceto más detallado . . . . .	72
Figura 5.24	Concepto final de la aplicación . . . . .	72
Figura 5.25	Concepto del error en la visualización de parámetros . . . . .	72
Figura 5.26	Diagrama de clase de las Vistas . . . . .	73
Figura 5.27	Estructura y Diseño de la primera subvista . . . . .	74
Figura 5.28	Estructura y Diseño de la segunda subvista . . . . .	75
Figura 5.29	Diseño descartado de la segunda subvista . . . . .	75
Figura 5.30	Estructura y Diseño de la tercera subvista . . . . .	76
Figura 5.31	Estructura y Diseño de la última subvista . . . . .	77
Figura 6.1	Imagen explicativa del concepto DevOps . . . . .	79

## LISTADO DE TABLAS

---

Cuadro 2.1	Presupuesto de la mano de obra . . . . .	10
Cuadro 2.2	Presupuesto de servicios adicionales . . . . .	10
Cuadro 4.1	Comparación de metodologías . . . . .	25
Cuadro 4.2	Primer caso de estudio de entorno . . . . .	29
Cuadro 4.3	Segundo caso de estudio de entorno . . . . .	30
Cuadro 4.4	Títulos de Historias de Usuario . . . . .	40
Cuadro 4.5	Subniveles de Títulos de Historias de Usuario	41
Cuadro 4.6	Títulos de Sprint 1 . . . . .	42
Cuadro 4.7	Descomposición de la Historia de Usuario 1 .	42
Cuadro 4.8	Descomposición de la Historia de Usuario 6 .	43
Cuadro 4.9	Títulos de Sprint 2 . . . . .	44
Cuadro 4.10	Descomposición de la Historia de Usuario 2 .	44
Cuadro 4.11	Títulos de Sprint 3 . . . . .	45
Cuadro 4.12	Descomposición de la Historia de Usuario 3 .	45
Cuadro 4.13	Descomposición de la Historia de Usuario 4 .	46
Cuadro 4.14	Títulos de Sprint 4 . . . . .	47
Cuadro 4.15	Descomposición de la Historia de Usuario 5 .	47
Cuadro 4.16	Títulos de Sprint 5 . . . . .	48
Cuadro 4.17	Descomposición de la Historia de Usuario 7 .	48
Cuadro 4.18	Títulos de Sprint 6 . . . . .	49
Cuadro 4.19	Descomposición de la Historia de Usuario 8 .	49
Cuadro 4.20	Descomposición de la Historia de Usuario 8.1	50
Cuadro 4.21	Títulos de Sprint 7 . . . . .	51
Cuadro 4.22	Descomposición de la Historia de Usuario 9 .	51
Cuadro 4.23	Descomposición de la Historia de Usuario 9.1	52
Cuadro 4.24	Descomposición de la Historia de Usuario 10 .	53
Cuadro 4.25	Títulos de Sprint 8 . . . . .	54
Cuadro 4.26	Descomposición de la Historia de Usuario 11 .	54
Cuadro 4.27	Descomposición de la Historia de Usuario 11.1	55
Cuadro 4.28	Títulos de Sprint 9 . . . . .	56
Cuadro 4.29	Descomposición de la Historia de Usuario 12 .	56

## LISTADO DE ALGORITMOS

---

## ACRÓNIMOS

---

RRSS redes sociales

FOMO *fear of missing out*, «temor a dejar pasar» o «temor a perderse algo»

TT *trending topic*, «tendencia», «tema de tendencia» o «tema del momento»

PaaS *plataforma as a service*, «plataforma como servicio»

YAKE *Yet Another Keyword Extractor*

VADER *Valence Aware Dictionary and sEntiment Reasoner*

DOM *Document Object Model*

API *application programming interface*, «interfaz de programación de aplicaciones»

REST *representational state transfer*, «transferencia de estado representacional»

HTTP *Hypertext Transfer Protocol*, «Protocolo de Transferencia de Hipertexto»

JSON *JavaScript Object Notation*, «notación de objeto de JavaScript»

CRUD *Create, Read, Update and Delete*, «Crear, Leer, Actualizar y Borrar»

ASGI *Asynchronous Server Gateway Interface*

WSGI *Web Server Gateway Interface*

Parte I  
INTRODUCCIÓN



## CONTEXTO

---

### 1.1 MOTIVACIÓN O JUSTIFICACIÓN DEL PROYECTO

El año 1989 fue clave en distintos aspectos, conocido por la disolución del Telón de Acero en Europa y la caída del Muro de Berlín, aunque sobre todo destacó por la apertura del Internet al público. Una herramienta, en un principio, al alcance de pocos que inicio su meteórico ascenso a ser uno de los mayores inventos. Hoy en día, capaz de conectar al mundo como muy pocos y que situaciones como la reciente pandemia COVID-19 no han hecho más que impulsar el intercambio digital de información entre personas.

Para podernos situarnos con más claridad, el estudio anual de IAB Spain nos indica que en el año 2022 el 93 % de las personas que residen en España son usuarios del Internet. Además de ese porcentaje, el 85 % (28,3 millones de españoles) utilizan las redes sociales. [1]

Estos datos nos dejan en evidencia el hecho de que el uso de estas herramientas, tales como el Internet y las redes sociales, van en aumento. Aunque no nos debería de asombrar, ya que la generalización y el uso del Internet y las redes sociales viene de mucho antes. En el año 2012 se realizó un estudio en el que se mostraba que las redes sociales pueden llegar a crear una dependencia más grande aun que el tabaco. [2]

Incluso muchos programadores de redes sociales han admitido que el diseño de tales aplicaciones es adictivo de manera deliberada. Un ejemplo de esto puede ser la creación del scroll infinito, donde se brinda gran cantidad de información de una manera continua. [3]

La solución más sencilla, pero drástica, puede llegar a ser abandonar las redes sociales, aunque nos surge el problema de no estar tan *actualizados* como otras personas a nuestro alrededor. Otra solución es usar en una menor medida las redes sociales, aunque esto nos lleva de nuevo al problema de cuándo parar si la aplicación que usamos tiene scroll o contenido infinito.

Ante tal necesidad de querer estar actualizados y a la vez querer saber cuando parar de consumir contenido, la solución que propongo es un consumo finito de contenido diario y lo más objetivo posible. Una aplicación web que aporte la información necesaria de las tendencias más destacables del día.

## 1.2 OBJETIVOS

### 1.2.1 *Objetivo general*

La meta final de este proyecto es construir una aplicación web autosuficiente y bien estructurada, cuyo contenido se pueda actualizar diariamente de manera automática. Con el objetivo de que el usuario final pueda consumirlo de una manera eficaz y eficiente.

La información que se le brinda al usuario se basará en las tendencias del país seleccionado por él. A partir de las tendencias se deducirán datos, estadísticas, noticias y enlaces de interés.

Toda esta información deberá albergar un carácter sencillo y deberá ser fácil de consumir, alejándose de la posibilidad de ser abrumante o atosigadora.

### 1.2.2 *Objetivos específicos*

- El contenido que se le brinda al usuario debe de ser corto y finito, pero a la vez suficiente con el fin de informarse de manera adecuada.
- La tendencia que se aporta al usuario proviene de la API de Twitter y tiene que ser la más destacable de la red social.
- La aplicación web debe de ser intuitiva y fácil de usar, el diseño debe ajustarse para el formato móvil.
- La estructura del proyecto deberá seguir el modelo API REST, siguiendo una arquitectura web por capas.
- Cada contenido proporcionado para el usuario se brindará en un diseño de carta y ocupará el mayor espacio de pantalla posible. De esta manera se aprovecharán al máximo la información disponible, siendo esta una noticia, un dato de una estadística o un enlace de interés.

### 1.3 SECCIÓN EN PROCESO !!!

En esta sección se planificarán las tareas y los objetivos. Se crearán dos diagramas de Gantt, uno dónde se planifiquen los objetivos y uno final real. Los objetivos planteados son:

- Planificación y presupuesto. DONE
- Estado del arte: DONE
  - Dominio del problema a resolver DONE
  - Metodologías y tecnologías de base que podrían usarse ->(Conclusión) DONE
- Propuesta:
  - Metodología de desarrollo - SCRUM (profundidad) DONE
  - Análisis, requisitos e historias de usuario (Backlogs)
  - DevOps ? como metodología
- Diseño:
  - Explicación de la arquitectura usada, distintas capas y protocolos (diagramas y esquemas)
  - Diagrama de paquetes, clases, bocetos ...
  - Justificación API de Twitter (comparación con Pytrends) DONE antes
  - Integración continua
    - Gestor de paquetes: poetry
    - Linter y Prettier
    - Mypy y flake8
    - GitHub Actions
    - Test y virtualización: GH Actions, tox
    - Uso de docker para test (usar comparativas - alpine) (Opcional)
    - Despliegue en Vercel (usar comparativas - netlify)
  - Justificación de pydantic de python
  - Justificación del micro-framework - FastAPI
  - Justificación de front end framework - Vue3.js
  - Justificación de TailwindCSS
  - Justificación de Apexcharts
  - Justificación de la base de datos MongoDB
  - Implementación de sistema de Logs (Opcional)
  - Prueba de prestaciones con unicorn (Opcional)

■ Implementación:

- Explicación del algoritmo que recoge trends de la API de Twitter
- Explicación del algoritmo que recoge noticias a partir de los trends - Comparar con la api y scrapping
- Explicación del algoritmo que recoge estadísticas del volumen de seguimiento (rango de tendencia)
- Explicación del algoritmo que recoge estadísticas de las palabras más usadas en la tendencia y YAKE
- Explicación del algoritmo que recoge estadísticas del sentimiento general de la tendencia (Vader) comparar con ML y texblob

■ Conclusión

■ Trabajos futuros y Bibliografía

Preguntas a tutores:

- ¿El resumen debería llevar sangría?
- ¿Debería haber presupuesto en la parte del plan inicial?

Recordatorios:

- Cambiar los HOLA

# 2

## PLANIFICACIÓN

### 2.1 PLANIFICACIÓN INICIAL

La planificación mostrada a continuación fue realizada al inicio del proyecto, siendo ésta de muy alto nivel. De momento no vamos a indagar en detalles específicos sobre las metodologías usadas o la planificación real, esto en cambio, se detallará en la sección posterior.

Un trabajo de fin de grado está compuesto por 12 créditos en el Título de Grado en Ingeniería Informática de la Universidad de Granada. Además, por normativa de la universidad ea cada crédito ECTS se le establece un número de 25 horas de trabajo. En total, 300 horas de trabajo. Esto se traduce a casi dos meses de trabajo si contamos con una jornada laboral completa y descanso los fines de semana.

Aunque, al fin y al cabo, sigue siendo una estimación muy ligera a lo que realmente terminará siendo. Por lo que la estimación que usaré a continuación será en base a las fechas del inicio y el fin de la elaboración del proyecto, en mi caso será desde inicios de Julio hasta finales de Octubre:



Figura 2.1: Visión general sobre la estructura de la elaboración del proyecto.

- Inicio (*01-07-22 ~ 01-08-22*): etapa de planificación de la memoria y el estado del arte. Además se intentará tener un primer prototipo de la aplicación.
- Diseño (*01-08-22 ~ 01-09-22*): inicios de la codificación y la elaboración de la parte correspondiente de la memoria.
- Lanzamiento (*01-09-22 ~ 01-10-22*): etapa donde se realizarán pruebas de usuario y finalizará la codificación.
- Entrega final (*01-10-22 ~ 31-10-22*): elaboración de conclusiones respecto al cumplimiento del software y sus posibles mejoras.

## 2.2 PLANIFICACIÓN DETALLADA

Al planificar el proyecto de la manera que está representado en la figura 2.1, he optado por una metodología ágil para confrontar el proyecto.

Esto es debido a que necesitamos un marco de trabajo que sea adaptativo y flexible a la hora de realizar distintas pruebas y su correspondiente arreglo a la hora de encontrarnos errores o cambios exigidos por el cliente.

Todas estas ventajas son las que me encaminado a elegir esta metodología, la cual parece tener las mejores prácticas para este proyecto en concreto.

Hoy en día, el marco de trabajo más ampliamente conocido y a la vez el más ventajoso sea posiblemente SCRUM. Se ha elegido este marco de trabajo para tener el mejor resultado posible a la hora de abarcar el proyecto. Sus principales ventajas son la flexibilidad y la productividad, pero abarcaremos más en detalle sus propiedades en la sección 4.1.

En la siguiente página se podrá ver la planificación en mucho mayor detalle. En la tabla (figura 2.2) se detalla toda la perspectiva del trabajo, desde las partes más fundamentales de la documentación hasta el diseño de la propia aplicación.

	Sprint 0	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9	Sprint 10
	01-07-22   07-07-22	13-07-22   19-07-22	25-07-22   31-07-22	06-08-22   12-08-22	18-08-22   24-08-22	30-08-22   05-09-22	11-09-22   17-09-22	23-09-22   29-09-22	05-10-22   11-10-22	17-10-22   23-10-22	29-10-22   04-11-22
<b>Introducción</b>											
Resumen e abstracto											
Planeación inicial											
Planeación detallada											
Presupuesto											
<b>Estado del arte</b>											
Descripción e investigación											
Estudio de la competencia											
<b>Formación</b>											
Formación en JavaScript											
Formación en Vue 3											
Formación en Python											
Formación en Tailwind CSS											
Formación en Micro-Frameworks											
<b>Entorno de trabajo (SCRUM)</b>											
Análisis											
Explicación de los requisitos											
Profundizar en la metodología											
Selección de requisitos											
Planeación de la iteración											
Revisión (demonstración)											
Retrorespectiva											
<b>Diseño</b>											
Justificación del diseño tomado											
Codificación del diseño											
Integración continua											
Revisiones y pruebas											

Figura 2.2: Visión más detallada sobre la estructura de la elaboración del proyecto.

### 2.3 PRESUPUESTO

Las horas de trabajo las tenemos especificadas desde un principio, es decir, el trabajo de fin de grado por normativa tiene 300 horas. Gracias a las páginas web como Talent podemos saber que el salario medio de un desarrollador web está en torno de los 11,79 €. [4] En cuanto a las horas de trabajo y sus diferentes secciones quedaría de la siguiente manera.

	Horas	Porcentaje	Coste
<b>Introducción</b>	16,9014	5,6338 %	199,27 €
<b>Estado del arte</b>	8,4507	2,8169 %	99,64 €
<b>Formación</b>	21,1267	7,0422 %	249,08 €
<b>Entorno de trabajo</b>	147,8873	49,2957 %	1743,59 €
<b>Diseño</b>	105,6338	35,2112 %	1245,42 €
<b>Total</b>	≈ 300	≈ 100 %	3536,99 €

Cuadro 2.1: Presupuesto de la mano de obra, junto con las horas trabajadas.

En base a ello, podemos calcular que unas 300 horas de trabajo equivaldrían a 3537 € y la elaboración de la documentación o el manual equivaldrían a 300 €. En total 3837 € solamente por la mano de obra.

Buscaré además el uso de herramientas gratuitas de diferentes servicios para el mantenimiento del servicio web. En los servicios como Heroku varían los precios entre 100 € y 200 € al mes, pero también poseen herramientas gratuitas y muy limitadas, lo que nos permitirá un gran ahorro al inicio del proyecto. Igualmente para el proyecto necesitaré un área de trabajo y un ordenador capaz de manejar dicho proyecto.

	Cantidad	Coste
<b>Espacio de trabajo</b>	4 meses	≈ 400 €
<b>Ordenador</b>	1	≈ 900 €
<b>Total</b>		≈ 1300 €

Cuadro 2.2: Presupuesto de servicios adicionales, junto con las diferentes cantidades.

De modo, que el coste total del proyecto sería la suma de ambas cantidades. En este caso, quedaría en 5137 €.

# 3

## ESTADO DEL ARTE

---

### 3.1 DOMINIO DEL PROBLEMA PRESENTADO

Como se había mencionado en las secciones anteriores, la motivación detrás de este proyecto es la creación de una herramienta que permita al usuario un uso más moderado y responsable de la información dada por las redes sociales ([RRSS](#)). Para entender mejor el contexto, la definición que nos proporciona el autor Orihuela de las [RRSS](#) es:

*Son servicios basados en la web que permiten a sus usuarios relacionarse, compartir información, coordinar acciones y en general, mantenerse en contacto.* [5]

Dando a entender que las [RRSS](#) son percibidas como un gran entorno social, en el cual se comparte información de una manera constante.

El hecho de que estos entornos sociales tengan un fácil acceso, fomentando la posibilidad de adicción y esto siendo de una manera deliberada termina siendo el problema real del consumo de información en las [RRSS](#). Se han mencionado artículos y estudios anteriormente donde se explicaba que la adicción de las [RRSS](#) viene directamente de la parte de la programación [3] y que las [RRSS](#) pueden llegar a crear una dependencia más grande aún que el tabaco. [2]

Todo esto acaba reflejado directamente en los usuarios, los cuales son realmente el lucro de todas estas grandes compañías. El objetivo real de todas estas empresas es conseguir el mayor número de usuarios y el mayor uso de su aplicación posible. De este modo a ellos, a términos generales, les conviene conseguir algoritmos o técnicas que favorezcan la consumición de dicho producto.

La idea general, la cual nos adentraremos en más detalle a continuación, viene a ser la abrumación de la información donde el usuario ni si quiera está en control del consumo de dicha información y tampoco del tiempo empleado dentro de la aplicación.

Todo ello tendrá consecuencias reflejadas directamente y exclusivamente en los usuarios, lo cual se verá en el siguiente apartado [3.1.3](#).

Toda esta información puede llegar a ser obvia si se expone superficialmente. Por lo que es necesario entrar en mucho mayor detalle a la hora de exponer el problema.

### 3.1.1 Causas del problema presentado

El *smartphone* o teléfono móvil es solamente una mera herramienta que nos permite conectarnos a Internet y acceder a diferentes servicios virtuales o aplicaciones web. El sentimiento de la adicción no se origina en la propia herramienta, sino en los distintos servicios ofrecidos por ella.

El modelo de negocio de la mayoría de las aplicaciones web consiste en el intercambio de datos personales en cambio al uso del propio producto. De esta manera pueden posteriormente tener mejores resultados de ventas en base a anuncios personales. Se estima que alrededor de un 91 % de los compradores son más receptivos a la compra de un producto cuando una marca personaliza su comunicación con ellos, es decir, anuncios personalizados en base a sus perfiles. [6]

Por lo que no es de extrañar que los propietarios de las aplicaciones web tengan la intención de prologar el uso de la misma, ya que esto les proporcionará mayor estudio de los perfiles de sus usuarios, mayor posibilidad de compra en sus anuncios y obviamente mayor rendimiento o beneficio.

De esta manera, es bastante sencillo encontrar diseños, patrones o incluso algoritmos que tengan la intención de la prolongación de uso de los usuarios en la aplicación web. Existen distintos elementos utilizados para prolongar el tiempo de uso de las aplicaciones de redes sociales, algunos de ellos los detallaremos a continuación. [7]

#### 1. *Scroll* ó *Streaming* infinito:

Gran cantidad de aplicaciones tienen un diseño inmersivo, en consecuencia, llegan a un producir lo que en psicología se denomina *flujo* ó *la zona*<sup>1</sup>. Dicho flujo mental, es una cualidad que normalmente es positiva, ya que puede llegar a incrementar en una gran medida la productividad. En otros casos puede llegar a ser altamente peligrosa, como es el caso de los dispositivos móviles; la actividad puede llegar a ser altamente gratificante y tener una retroalimentación directa con el usuario, dando lugar a la perdida de la noción del tiempo o su propia distorsión.

Un claro ejemplo de esto es la prohibición del uso del teléfono móvil al circular con vehículo por las vías públicas, esto es debido del alto peligro que puede llegar a haber al prestar atención o variar el flujo mental por la mayor inmersión que ejerce el teléfono móvil y dichas aplicaciones instaladas en él.

---

<sup>1</sup> El *flujo* ó *la zona* es un estado mental que se consigue cuando una persona está completamente inmersa en la actividad que esté ejecutando, es decir, es un estado de concentración de energía o de motivación.

Este estado mental es provocado directamente de las aplicaciones instaladas en los teléfonos y esto de manera deliberada por muchos programadores. El *Scroll* ó *Streaming* infinito es solamente un diseño que fue propuesto por Raskin en 2006. Dicho diseño es sumamente adictivo, ya que no da tiempo a que lleguen todos los impulsos necesarios del cerebro mientras que el usuario sigue deslizando contenido tras contenido. El usuario además encuentra algo gratificante con cada contenido nuevo aportado, como puede ser un vídeo o foto graciosa. Este efecto llega a ser muy parecido al efecto que producen las máquinas tragaperras, la condición de inmersión al usuario es prácticamente igual. Otro diseño parecido puede ser el *Auto-play*<sup>2</sup>, donde la lógica es la misma.

El creador de dicha herramienta, Raskin, expuso en una entrevista su arrepentimiento de haber creado algo tan adictivo<sup>3</sup>. Su intención era más bien intentar innovar las interfaces que volver a las personas tan adictivas. [3]

## 2. Efecto dotación ó efecto de mera exposición:

El efecto dotación consiste en que las personas atribuyen más valor a un objeto por el simple hecho de poseer dicho objeto. Esto es, que si una persona llega a poseerlo, ya sea tangible o intangible, luego le es más difícil desprendérse de ello aunque le ofrezcan el mismo o incluso mayor valor que posee dicho objeto.

El efecto de mera exposición también es un efecto psicológico, este se basa en que la reiteración de exposiciones incide o estimula en nuestro agrado o desagrado. En otras palabras, las personas llegan a familiarizarse con distintos estímulos por el simple hecho de repetirlos en diferentes ocasiones.

Estos dos efectos influyen en nuestro contexto, de tal manera que los usuarios de distintas aplicaciones al gastar más tiempo, valor o dinero en ellas les resulte más complicado posteriormente abandonar o siquiera desinstalar la propia aplicación. Tanto por el simple hecho de llegar a tener algún tipo de valor en la propia aplicación, como por ejemplo seguidores o *likes*, ó directamente por estar familiarizados con la aplicación por haberla usado día a día y verla siempre en la pantalla de inicio.

## 3. Presión social:

<sup>2</sup> *Auto-play* es una herramienta donde se proporciona contenido infinito si no se llega a parar por el propio usuario. Un ejemplo de ello, aplicaciones como YouTube o Spotify, donde se requiere darle al botón *pause* para frenar dicho contenido.

<sup>3</sup> Raskin además de admitir que nunca pudo predecir lo adictivo que su invento podría ser, también comentó que él mismo está usando un filtro monocromático en su dispositivo móvil para minimizar lo adictivo que puede llegar a ser tener tantos estímulos en la pantalla.

WhatsApp actualmente es la aplicación de mensajería más usada en todo el mundo, ahora mismo, posee más de 2.000 millones de usuarios. [8] Al igual que otras redes sociales dominantes en el mercado, como Instagram o Twitter, posee diseños construidos de tal manera que llegan a inferir desde muchas personas a un individuo en concreto, ejerciendo algo parecido a la presión social.

Un claro ejemplo de esto puede ser la función el doble *tick* azul. Consiste, en el caso de la mensajería instantánea, notificar al usuario de que el mensaje ha llegado hacia la otra persona y además fue leído por ella. Esta función compromete al usuario de manera que tenga responder rápidamente o incluso por obligación si el mensaje fue leído o entregado. Dicha función aparece activada por defecto, la cual muchos usuarios ni son conscientes de que pueden llegar a cambiar. Según un reporte, únicamente un 5% de los usuarios de Microsoft Word llegó a cambiar algún ajuste que venía por defecto. [9]

De manera análoga, se puede establecer un vínculo de las características mencionadas anteriormente con la presión social, como es el caso del doble *tick* azul, y el síndrome o fenómeno *fear of missing out*, «temor a dejar pasar» o «temor a perderse algo» (**FOMO**). Como su nombre expresa, es el miedo a perderse algo o ser excluido, dando lugar a la necesidad de estar permanentemente conectados y *actualizados*. Las consecuencias pueden llegar a ser muy graves, ya que cuanto mayor es el uso del dispositivo móvil mayor es el grado de **FOMO**; el temor de perderse algo y de esta manera no satisfacer nuestras necesidades psicológicas retroalimenta el uso problemático y abusivo de las aplicaciones. [10]

#### 4. Contenido personalizado con la intención de causar agrado:

Al entrar a cualquier aplicación tipo Facebook, Instagram o Twitter lo primero que se encuentra el usuario es con una abrumación de contenido para que este no tenga momento de aburrirse. Este contenido suele recibir el nombre de *Feed*, *NewsFeed* o *TimeLine*. Al presentar el *Feed* a cualquier usuario, este viene personalizado de antes con los gustos del usuario, para intentar mantenerlo el mayor tiempo posible.

Los algoritmos que hay por detrás del *Feed* de cada usuario son enormemente complejos, ya que no solo se encargan de calcular el contenido que le agrada al usuario por medio de *likes* o *shares* del contenido, sino también de el contenido que el usuario gasta en mirar cada publicación.

Anteriormente el *Feed* de las aplicaciones funcionaba de manera cronológica, pero actualmente se basan en la retención de la

atención del usuario. Intentado mostrar al usuario contenido que sea afín a sus intereses. [11]

Aunque no siempre tienen la intención de causar agrado, en algunos reportes se declaró justo lo contrario. Gracias a Frances Haugen, en 2021 se filtró información sensible sobre FaceBook donde entre mucha información se exponía que esta red social también llega a mostrar contenido que el usuario puede odiar, solamente para tener más retención por parte del usuario. [12] Esto además se traduce a que el algoritmo muestre contenido cada vez más violento para que el usuario tienda siempre a interaccionar con él.

#### 5. Comparación y recompensa social:

El *Like* fue desarrollado por Justin Rosenstein, esta función probablemente es el mecanismo más emblemático de recompensa social que hay hasta el momento. Es recompensa social, ya que hay retroalimentación social positiva al interaccionar con los *likes* de una publicación. Esta retroalimentación social está demostrada científicamente, en un experimento neurocientífico se presentaron imágenes con muchos *likes* ó *me gustas*, estas imágenes provocan una actividad más fuerte en el cuerpo estriado del cerebro (parte del cerebro relacionada con los mecanismos de recompensa). Además también se detectó que los volúmenes bajos de materia gris se asociaron con el uso prolongado de este tipo de aplicaciones, lo que daba lugar a mayores tendencias adictivas. [13, 14]

Leah Pearlman, co-inventora del botón *Me gusta* de Facebook, dijo que se sintió adicta a Facebook porque había empezado a basar su autoestima en la cantidad de *likes* que tenía. [3] Aunque este normalizado, sigue siendo un problema el que una persona sea capaz de compararse numéricamente por el hecho de como es percibida por su red social. Aunque hablamos de las consecuencias más adelante (3.1.3), hay que mencionar el hecho de compararse de esta manera puede dar lugar a una baja autoestima y otros muchos problemas.

#### 6. Sistema de notificaciones:

La funcionalidad de un simple sistema de recordatorios, nos presenta notificaciones en determinados momentos a lo largo del tiempo, ya sea a través del sonido o recuadros visuales. El objetivo de estas notificaciones puede ser una utilidad de la aplicación, como un mensaje o solo para tener presente el hecho de que podemos volver a usar la aplicación.

De esta manera nos tienen pendientes mediante un sencillo sistema, el cual puede llegar a ser muy útil en determinados momentos, como un mensaje del banco. Sin embargo, el diseño de

dicho sistema puede llegar a ser perjudicial para el usuario. Un claro ejemplo de esto es Instagram, la cual agrupa notificaciones para poder otorgar al usuario mayor cantidad de recompensa o agrado, es decir una cantidad determinada de *likes*, esto se traduciría al control de diferentes neurotransmisores como la dopamina en el usuario, aunque indagaremos más a fondo en la sección 3.1.2.

Todos estos numerosos problemas están estrechamente relacionados con las aplicaciones que existen a día de hoy, ya sea solamente uno o numerosos de ellos en la misma aplicación. Estos además pueden llegar a aumentar su efecto de distintas maneras si se agrupan entre sí.

El tipo de aplicaciones que se sugiere en el párrafo anterior son las aplicaciones que anteponen la posibilidad de monetización a la éticidad respecto al usuario. Este tipo de aplicaciones no priorizan agilizar nuestro modo de vida, como es el caso de una aplicación bancaria, sino que intentan generar estímulos presentes entre emociones negativas como la soledad o el aburrimiento.

Sandy Parakilas, antiguo director de operaciones de FaceBook llegó a comentar lo complicado que le era abandonar la plataforma, comparándola con cigarrillos o máquinas de casino. Aunque también comentó cuestiones de mayor peso, explicando que el uso de estas aplicaciones construían un hábito en nuestras vidas: [3]

*There was definitely an awareness of the fact that the product was habit-forming and addictive. [...] You have a business model designed to engage you and get you to basically suck as much time out of your life as possible and then selling that attention to advertisers.*

Lo cual viene a ser un buen resumen de lo que se ha tratado de explicar en esta sección. En castellano tendría este significado:

*Definitivamente hubo un conocimiento del hecho de que el producto era capaz de crear hábitos y era adictivo. [...] Tienes un modelo de negocio diseñado para engancharte y hacer que básicamente absorba la mayor cantidad de tiempo posible de tu vida y luego venda esa atención a los anunciantes.*

### 3.1.2 Proceso gradual del problema

#### 3.1.2.1 Inflexión en el individuo

El uso de las redes sociales tiene numerosos efectos sobre el cerebro, estos pueden llegar a influir tanto de manera positiva como negativa.

Al navegar por las [RSS](#) el cerebro se adapta llegando a crear nuevas redes neuronales. [15] Además, también pueden llegar a producir diferentes cambios en la manera que funcionan distintos neurotransmisores como la oxitocina, la adrenalina, la dopamina, la serotonina, la testosterona y el cortisol.

Todos estos neurotransmisores tienen un determinado papel a la hora de estar usando las distintas aplicaciones en los teléfonos móviles. La oxitocina está relacionada con la influencia de la familia y la pareja, mientras que la adrenalina se vincula con la agresividad. El aumento de serotonina se llega a traducir a comportamientos sociales en ámbitos más introvertidos. Por otra parte, altos niveles de testosterona y el cortisol tienen que ver con una alta fidelidad aunque menor número de amistades virtuales.

Aunque la que podría llegar a ser más problemática para el usuario es la dopamina <sup>4</sup>, ya que está mucho más relacionada con las redes sociales. Anteriormente mencionamos que las [RSS](#) proporcionan al usuario diferentes tipos de interacciones y recompensas sociales, de esta manera generando al usuario emotividad y afectividad.

Existen cuatro vías que transmiten dopamina en nuestro cerebro llamadas vías dopaminérgicas, ellas son la mesocortical, la nigroestriada, la mesolímbica y la tuberoinfundibular. Las tres primeras de ellas están relacionadas con las recompensas que se mencionaba en el texto anterior. Al tener un estímulo satisfactorio o agradable por estas vías, ciertas acciones se disparan y nuestro cerebro refuerza dicho comportamiento. [17]

Aunque cabe mencionar que el abuso de estas vías por medio de sustancias o hábitos da lugar a la creación de la adicción. Además la falta de dicho estímulo puede provocar un rechazo por parte del cerebro creándonos malestar o ansiedad, esto es debido a que estas vías se hayan podido *deformar* por un mal hábito o aprendizaje, como por ejemplo la transmisión de la posible tranquilidad que nos puede aportar estar continuamente consultando el teléfono.

Los seres humanos somos seres sociales y es normal que dependamos de interacciones continuamente, dando lugar a una posible adicción de conducta. Esto es, debido a que no solo se puede ser adicto a una sustancia, sino a diferentes conductas o hábitos, un ejemplo de ello puede ser una máquina de un casino. El efecto que genera una máquina de casino y una red social en el cerebro puede llegar a ser parecida, además la recompensa que nos sirven este tipo de adic-

---

<sup>4</sup> La dopamina regula la emotividad y la afectividad, así como en la comunicación neuroendocrina. Algunas alteraciones en la transmisión dopaminérgica han sido relacionadas con la adicción a drogas (anfetaminas y cocaína por ejemplo). [16]

ciones son mucho más satisfactorias y requieren menos esfuerzo que cualquier actividad más sana como el ejercicio o la lectura. [18]

Según un estudio del Universidad de Philipps, las recompensas sociales que disparan los distintos mecanismos de las vías dopaminérgicas son sencillas interacciones que tenemos con los seres humanos, ya sea una caricia, sonrisa o palabras afectivas. Todas estas interacciones son suficientes para influir como una recompensa social y crear dicho estímulo. Los investigadores evolutivos explican este fenómeno como un mecanismo social del ser humano necesario y positivo. [19]

Entrando más en detalle en las adicciones de conducta y las notificaciones de las aplicaciones. Podemos recurrir al clásico caso de la teoría de Pavlov del siglo XX, donde se presentaba un estímulo que no tenía nada que ver y terminaba siendo reflejo condicional (Ley del reflejo condicional o condicionamiento clásico). En la teoría de Pavlov se tocaba una campana al servir comida a unos perros, al cabo de un tiempo los perros al escuchar el estímulo de la campana llegaban a salivar sin que se les sirviera la comida. [20]

En cierto sentido hay parecidos entre el condicionamiento clásico de Pavlov y las notificaciones de los dispositivos móviles hoy en día. El estímulo de la notificación nos crea la conducta de estar pendientes de los teléfonos y estar mirándolos continuamente, llegando a coger el móvil decenas o centenares de veces al día. Además este estímulo puede estar asociado a un sentimiento positivo, como puede ser el mensaje de un amigo. Un sencillo ejemplo de ello es acudir o prestar atención a un móvil ajeno por tener un sonido de notificaciones parecido al propio.

Según la investigadora de diseño de la red social Twitter, Ximena Vengoechea, afirma que el *enganche* de la notificación compromete a dos mecanismos, uno interno y otro externo. [17] Se refiere al interno como la emoción, mientras que el externo es la acción que pretende que hagamos. La sincronización de ambos es la combinación que despierta la motivación del usuario, creando la necesidad de la recompensa. Además, como ya sabíamos, las RRSS aprovechan la implicación de estímulos visuales o distintos diseños derivando en un mecanismo de muchos niveles. Esta mezcla nos crean distintas costumbres o hábitos que son, sin darnos cuenta, posiblemente adictivos.

Actualmente este tipo de temas son bastante comentados, aunque tristemente este uso problemático de las redes sociales fueron ignorados debido a la gravedad de la situación surgida por el COVID-19. La mayoría de las actividades que realizábamos día a día se volvieron virtuales, dando lugar a un incremento de dependencia y a una auténtica necesidad de comunicación por medios virtuales.

### 3.1.2.2 *Inflexión en la sociedad*

Otra consecuencia gradual en base al problema es la cantidad de información que es bombardeada al usuario día tras día, dando lugar a una gran cantidad de contenido que realmente es difícil de manejar. En cualquier red social, se masifica y se comparte la información o una noticia mediante opiniones. Creando así tendencias, pero quitando de esta manera la objetividad de la información o directamente dando lugar a su mal interpretación.

La noticia o tendencia por lo general suele tergiversarse debido a la cantidad ingente de opiniones e interacciones a las que se ve el sujeto. Dando lugar, a que el usuario no sea capaz de sacar en claro el estado de la noticia o tendencia, tanto si al final es positiva o negativa, en base a unas pocas compartidas. Además es de suma importancia el recibo que tienen estas opiniones mediante sus interacciones o la cantidad de opiniones parecidas que se van publicando a lo largo del tiempo de vida de la tendencia. Por lo que la medida más óptima de informarse de dicho contenido es a través de una recopilación objetiva, con la meta de alejarse de las opiniones y abarcar una información final respecto al tema que se intenta compartir.

### 3.1.3 *Consecuencias producidas en base al problema*

Los problemas existentes acaban afectando a todos los usuarios, independientemente de su edad. Por normativa, legalmente las redes sociales exigen que sus usuarios tengan al menos 13 años a la hora de crearse una cuenta. Aunque es cierto que esto no se cumpla del todo y haya usuarios con menos de 13 años exponiéndose a diferentes peligros de las RRSS.

Estudios realizados en 2021 demuestran que los niños a partir de 10 años tienen tendencias de uso de redes sociales. Estas exposiciones a tan temprana edad pueden terminar siendo un peligro y llegar a ser una adicción. [21]

Estos problemas también pueden surgir en personas de más avanzada edad. Ya hemos visto que el deseo de acceder a redes sociales como Twitter o Instagram se encuentra como el deseo más complicado de resistir y más sencillo de complacer. Sustancias adictivas como el tabaco o el alcohol generan un deseo mucho más débil, esto además es muy fácil de demostrar sabiendo que un usuario promedio de estado unidos mira el teléfono unas 344 veces al día, lo que se traduce como usarlo cada 4 minutos al día. [22]

Otro estudio, igual de reciente, mostraba que el 40 % de la población española llegaba a tener problemas relacionados con la adicción al Internet. Para hacernos una idea, en 2008 dedicábamos una media

de 18 minutos al móvil, en cambio, ya a partir de 2015 gastábamos alrededor de 3 horas diarias. Esto llegaba a afectar a los adolescentes de tal manera, que el 80 % de ellos tenían la necesidad de mirar el móvil, al menos, una vez cada hora a lo largo del día. [23]

Las consecuencias de la adicción a las redes sociales pueden llegar a ser la ansiedad, dependencia emocional, baja autoestima o problemas de sociabilización. Estudios realizados en 2021, demostraron la correlación existente entre la dependencia de las RRSS, la autoestima, ansiedad y la obsesión. Para ello participaron 100 alumnos, estudiantes universitarios. Este estudio corroboró información que existía previamente, resultados que señalaban que las personas que tenían adicción al uso del Internet presentaban una menor autoestima y relaciones personales más inestables. [24]

Esto también fue expuesto en las redes sociales, la autoestima influía en el uso de RRSS de manera reciproca, es decir, tener una alta autoestima también significaba un menor uso de RRSS. Además los estudiantes que presentaban una mayor autoestima exhibían niveles más bajos de obsesión, mayor control personal y empleo de dichas RRSS. En cambio, cuando el uso de las RRSS era mayor, los niveles de autoestima eran más bajos, incrementando de esta manera los niveles de ansiedad y depresión.

Sin embargo, el estudio no especificó la existencia de una correlación entre la adicción y la autoestima. Aún teniendo datos como el 66.7 % de los adolescentes fueron expuestos como adictos al Internet (no a las RRSS<sup>5</sup>) y 62.7 % manifestaron problemas de autoestima. Aunque si se demostró la existencia de correlación entre la adicción y la ansiedad. [24]

### 3.2 CONCLUSIÓN ELABORADA EN BASE AL PROBLEMA

En el texto anterior se quedó demostrado el hecho de que las redes sociales pueden llegar a ser adictivas, como son capaces de influir en un individuo y sus consecuencias. El problema real de esto recae en el propio diseño de las aplicaciones, si la información fuera expuesta de una manera clara y sin patrones con la intención de retener al usuario el mayor tiempo posible; el usuario podría usar dichas aplicaciones sin tan grave peligro.

La solución que se pretende proponer es una aplicación sin los diseños que se han comentado. En primer lugar, siendo una aplicación con un contenido finito y este siendo el más importante en el que se centre el usuario. La aplicación web se centraría en las tendencias

---

<sup>5</sup> En dicho estudio solamente se midió, mediante cuestionarios, la adicción del Internet en general y no específicamente las redes sociales.

más importantes de cada país del usuario, de esta manera mostrando los eventos más importantes que hayan ocurrido al final del día.

Siendo una aplicación web, se elimina la necesidad de tenerla instalada y sus posibles notificaciones. Además se centraría en ser objetiva, quitando el efecto **FOMO** y la personalización de contenido. Al quitar la parte subjetiva y distintas opiniones, es decir, eliminando la parte social de las **RRSS** y solo quedándonos con la información; eliminamos la posible comparación entre usuarios existente.

De esta manera, hay una posibilidad de seguir *actualizados* pudiendo ver el contenido importante que se ha proporcionado durante el día, pero quitando el peligro de volvemos adictos.



Parte II  
PROPUESTA



# 4

## ANÁLISIS

---

### 4.1 METODOLOGÍA DE DESARROLLO

El proceso de desarrollo de un software puede llegar a ser una tarea muy ardua, durante mucho tiempo esta labor se ha llevado a cabo sin ninguna metodología definida. Para situar el contexto, una metodología se define como una colección de procedimientos, técnicas, herramientas y documentos que llegan a ayudar a los desarrolladores a la hora de implementar software. [25]

Durante las últimas décadas, se han establecido dos grandes corrientes diferenciales que representan estas metodologías. Por un lado, las metodologías tradicionales se centran en el proceso y controlan estrictamente las actividades que lo acompañan. Las metodologías ágiles, por otro lado, se enfocan en el elemento humano, centrándose en la colaboración y participación del cliente en el proceso a través de iteraciones muy cortas. [25]

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (menos de 10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos y pocos roles	Más artefactos y más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Cuadro 4.1: Comparación de metodologías. [26]

La parte más importante de las fases iniciales es poder determinar qué metodología es mejor para nuestro proyecto con el fin de lograr los mejores resultados de manera oportuna.

El objetivo general de implementar una metodología es llegar a construir un producto de alta calidad. Esta elección implica un conjunto de principios básicos que deben ser seguidos y respetados. Estos incluyen actividades claras para comprender el problema y comunicarse con los clientes, un método definido para representar el diseño, las mejores prácticas para implementar la solución y estrategias sólidas. [27]

En base a las características mencionadas y lo explicado en la sección de planificación, se ha considerado que la solución óptima sería una metodología de desarrollo ágil. Considerando que el problema puede cambiar a lo largo del proyecto, por lo que necesitamos una metodología adaptativa y flexible, además la forma de trabajar concuerda mucho más (la cantidad del grupo o su propio manejo).

#### 4.1.1 *Metodología de trabajo escogida (SCRUM)*

El nacimiento de la idea de la metodología Scrum fue en el año 1986, con el propósito de aumentar la velocidad de desarrollo de una aplicación y a la vez conservando flexibilidad. Los japoneses Hirotaka Takeuchi e Ikujiro Nonaka se inspiraron en el concepto de un equipo de rugby de 15 jugadores unido y con el mismo objetivo. Durante los años posteriores este concepto fue remodelado y reforzado hasta convertirse en lo que es a día de hoy, el método ágil más usado en el mundo. [28]

Scrum puede llegar a ser ventajoso debido a que el cliente se siente comprometido con el proyecto, implicándose en distintas necesidades de tipo funcional y permitiendo realizar revisiones tempranas de los desarrollos fomentando de esta manera las propias entregas. Otras ventajas a destacar puede llegar a ser su adaptación o trabajo en equipo. [28]

Los componentes de la metodología Scrum son: [29]

1. **Los Roles de Scrum:** son los papeles o funciones de cada integrante del equipo, que puede ser:

- *El Dueño del Producto (Product Owner):* representará la función del cliente, con el objetivo de definir historias de usuario y validar la calidad del producto.
- *Scrum Master:* su función es velar por la actividad o ejecución del Scrum, para que se realice correctamente por el equipo. También será el responsable de dirigir y concretar los diferentes eventos.

- *El Equipo de Desarrollo*: son los que tienen la función de realizar y concretar las historias de usuario.
  - *Stakeholders*: no tienen función en la ejecución o realización del trabajo, pero están interesados en el producto y representan un papel de suma importancia.
2. Los **Eventos Scrum**: son las reuniones que se llevan a cabo en el equipo de trabajo. Están diseñados para la comunicación y colaboración continua entre equipos, con el objetivo de minimizar la necesidad de mantener reuniones indefinidas que pueden afectar de alguna manera a la planificación del proyecto. Se subdividen en estos tipos:
- *El Sprint*: trata de un tramo de tiempo fijo, que puede llegar a ser entre una y cuatro semanas, durante el cual el producto que se está desarrollando añade valor gradualmente, de manera que al final de cada sprint el cliente pueda tener una clara visión del producto.
  - *La planificación del Sprint (Sprint Planning)*: es una reunión que se lleva a cabo al comienzo o inicio de cada sprint para planificar el trabajo que se realizará a lo largo del sprint.
  - *Scrum Diario (Dailys)*: reuniones diarias de corta duración, en estas reuniones los miembros del equipo discuten y dialogan el trabajo realizado en el día anterior y el trabajo que se realizará en el día de la reunión.
  - *Revisión del Sprint (Sprint Review)*: una reunión donde cada miembro del equipo presenta lo que ha realizado a lo largo del sprint y lo que debería cambiar o mejorar en el producto para el próximo sprint.
  - *Retrospectiva del Sprint (Sprint Retrospective)*: reuniones con la intención de analizar el trabajo en equipo y discutir áreas para mejorar o colaborar como equipo para el próximo sprint.
3. Los **Artefactos de Scrum**: utilizadas para aumentar la transparencia de la información para que todos los miembros del equipo tengan constancia sobre lo que está ocurriendo. Se subdividen en:
- *El Backlog del Producto (Product Backlog)*: contienen todas las historias de usuario, pero no evalúan el proyecto, y su gestión o responsabilidad recae en el Dueño del Producto.
  - *El Backlog del Sprint (Sprint Backlog)*: historias de usuario que contienen estimaciones y planes a realizar en el sprint.
  - *Incremento*: es el producto que se produce o ocurre al final del sprint.

#### 4.1.2 *Adaptación de la metodología de trabajo escogida*

Habiendo definido los componentes en la sección anterior, es visible que Scrum está dirigido para trabajar en equipo, por lo que se tendrá que adaptar esta metodología al entorno académico del proyecto, que en el caso de este proyecto se reduciría a mi persona, Nikita Stetskiy. Debido a esto, me veré obligado a representar todos los roles del equipo y mis tutores darán función a los stakeholders y a Scrum Master.

Además, se puede dividir el proyecto en dos productos. La memoria, donde el cliente será el tribunal del trabajo de fin de grado y los tutores serán los Dueños del Producto. Mientras que para la propia aplicación, tendremos un cliente ficticio, por ejemplo Mauricio Mares, y el Dueño del Producto seré yo mismo, puesto que seré el que mejor conoceré los intereses del cliente.

En cuanto a los eventos, el Scrum diario no llega a tener sentido en este entorno, al estar formado por solo una persona y los demás eventos se pueden juntar en un mismo encuentro e iteración.

Finalmente, respecto a la velocidad del proyecto, se llegó a detallar en la sección [2.2](#) de la Planificación. Cada Sprint tiene acordada una duración de dos semanas.

## 4.2 ANÁLISIS DEL ENTORNO

Es necesario estudiar el entorno y a los posibles usuarios que necesiten de nuestro servicio antes de empezar a desarrollar el *Backlog* del Producto. Para ello propondré dos diferentes casos ficticios que estudiaremos a continuación.

1 <sup>a</sup> Persona	
Nombre	Mauricio Mares
Edad	23
Sexo	Masculino
Datos relevantes	
Terminó la carrera de bellas artes, aunque ahora mismo no se siente lo suficientemente apasionado para trabajar de ello. Uno de los principales motivos es el sobre uso de las redes sociales en su día a día.	
Contexto de uso	
El uso que pretende dar este usuario corresponde a un ámbito casual, ya que pretende usar menos las redes sociales, pero a la par quiere mantenerse actualizado de lo que pasa en el mundo. Por lo que la función que requiere de la aplicación es de informarse de eventos importantes que transcurran diariamente o incluso semanalmente.	
Necesidad diferencial	
La funcionalidad a destacar es que la aplicación deberá guardar todas las tendencias del día y poder dar datos que resuman información de dichas tendencias. Además, Mauricio no tiene que estar constantemente atento a la aparición de una nueva tendencia, ya que se van guardando por días. Esto le permitirá a Mauricio reducir las horas de consumo diario de aplicaciones que requerían su atención constante a cambio de información relevante o actual.	

Cuadro 4.2: Primer caso de estudio de entorno.

En este primer caso necesitamos que la aplicación sea capaz de resumir información de tendencias diarias y que a la vez no sea atosigante o que agobie con demasiada cantidad de información, por ende tiene que ser información simple y finita. La posible solución más óptima es organizar tendencias diarias en un *top* o una clasificación de diez tendencias más populares ocurridas a lo largo del día. Además esto se puede guardar por semanas para que el usuario no tenga la obligación de entrar a la web diariamente.

El carácter de la información, simplificada para el usuario que tendrá las tendencias, será puramente informativo y además intentará ser objetiva. La principal diferencia es que al usar las [RSS](#), la información relevante se consigue por medio de la atención. Cuantas más publicaciones u opiniones de personas se lean, más informado estará el usuario. Por lo que el factor diferencial, es que la página web en desarrollo mostrará datos simplificados o resumidos de manera objetiva. Todos estos datos estarán comprimidos de tal manera, que el usuario pueda tener una visión clara sobre la tendencia sin la necesidad de leer un alto número de publicaciones u opiniones.

La información que se aporta al usuario en la aplicación web será modular. De tal manera que cada módulo, que tendrá el diseño de una carta, informará de un dato de interés. Los datos más importantes que puede tener una tendencia y que sea útil para el usuario son datos como la relevancia que está teniendo la tendencia, las palabras o *keywords* más usadas, las opiniones de las personas (vistas de manera objetiva a través de un análisis de sentimientos) y diferentes noticias actuales relacionadas con la propia tendencia.

Des esta manera, al aportar información de manera simplificada, el usuario Mauricio podrá informarse diariamente de manera adecuada y sin la necesidad de perder el tiempo en las [RSS](#).

<b>2ª Persona</b>	
<b>Nombre</b>	Nuria Nevadas
<b>Edad</b>	37
<b>Sexo</b>	Femenino
<b>Datos relevantes</b>	
Terminó hace mucho el grado de Empresariales, en los últimos meses empezó a invertir en Bolsa, por su propia cuenta de manera profesional.	
<b>Contexto de uso</b>	
El uso que pretende dar este usuario corresponde a un ámbito profesional. Pretende tener una visión pequeña y objetiva de búsquedas concretas, en este caso del mercado financiero.	
<b>Necesidad diferencial</b>	
La funcionalidad a destacar en esta aplicación es que se podrá buscar tópicos y obtener la misma información actual que se obtienen también con las tendencias, es decir, datos de interés objetivos.	

Cuadro 4.3: Segundo caso de estudio de entorno.

Al estudiar el segundo caso, la solución que se le plantea al usuario llega a ser bastante parecida al primer caso. El principal factor

diferencial es que la información que se aporta en el primer caso es sobre las tendencias y en este caso tendrá que ser sobre búsquedas específicas.

No se podrá dar información sobre la popularidad del tópico, ya que no es una tendencia, pero si información sobre las *keywords* o palabras más usadas, el sentimiento general y las noticias más destacables. Esta información tendrá que ser recopilada de distintas publicaciones que tengan relevancia con el tópico.

#### 4.2.1 *Estudio competitivo*

En la actualidad las principales aplicaciones que existen que recopilan tendencias a diario son Google Trends y parcialmente la plataforma de Twitter. Ambas poseen fallos, el principal problema de Twitter se incluye en los riesgos de un uso continuo de una [RSS](#) que se explicó anteriormente en la sección 3. Por otro lado, Google Trends posee multiples problemas a la hora de afrontar nuestras necesidades. Primero, no funciona en todos los países, incluyendo España. Además no contienen información relevante que pueda resumir una tendencia o el por qué de su relevancia. Aunque Google Trends posea un exhaustivo historial de las búsquedas que ha tenido dicha tendencia, posibles noticias de interés y tópicos similares, no posee el sentimiento general que está teniendo dicha tendencia y tampoco diferentes *keywords* en relación (esto debido a que carece de publicaciones como Twitter). En resumen ambas aplicaciones principales, tienden a ser incompletas a nuestras necesidades.

También existen otras páginas que usan información de Twitter o Google. Al buscar páginas o aplicaciones web sobre la información de tendencias actuales de Twitter salen resultados como Trendinalia, GetDayTrends o Trends24. Este tipo de páginas web llega a mostrar un contenido puramente estadístico sobre las tendencias, con el uso exclusivo de diagramas o distribuciones globales para explicar el comportamiento de una tendencia.

Realmente este tipo de páginas no indagan en la explicación del contenido de una tendencia para que el usuario pueda estar informado. El objetivo de estas páginas es informar al usuario de datos numéricos que se calculan mientras la tendencia exista, como puede ser la posición de la tendencia respecto a otras durante las últimas 24 horas.

Esto puede llegar a ser información útil si solo quieres saber datos sobre el comportamiento de la tendencia a lo largo de 24 horas. En el caso de nuestra aplicación, aparte de dar datos sobre el comportamiento de la tendencia, tendrá el objetivo de explicar el propósito de la tendencia.

#### 4.2.2 Estudio competitivo técnico

Actualmente las librerías más populares sobre recopilación de tendencias actuales son Tweepy y Pytrends, ambas programadas en el lenguaje Python. Dichas librerías distan de ser perfectas, pero cada una tiene características de carácter diferencial.

##### ■ Pytrends [30]

Esta librería es una API no oficial que usa información de Google Trends. Por lo que es una librería que aporta información muy valiosa respecto a diferentes tendencias. Esta librería posee numerosas ventajas, a la par que desventajas. Las siguientes características son los métodos que tiene la librería disponibles:

1. *Interés a lo largo del tiempo*: devuelve datos históricos e indexados de las búsquedas sobre una palabra clave. Esta información se muestra en la sección Interés a lo largo del tiempo de Google Trends.
2. *Interés histórico por hora*: devuelve datos históricos, indexados y por hora de las búsquedas sobre una palabra clave. Envía múltiples solicitudes a Google, cada una de las cuales recupera una semana de datos por hora.
3. *Interés por región*: devuelve datos sobre el lugar en el que se busca la palabra clave. Información mostrada en la sección Interés por región de Google Trends.
4. *Temas relacionados*: devuelve datos de las palabras clave relacionadas con una palabra clave proporcionada. Se muestra en la sección Temas relacionados de Google Trends.
5. *Consultas relacionadas*: devuelve datos de las palabras clave relacionadas con una palabra clave proporcionada. Se muestra en la sección Consultas relacionadas de Google Trends.
6. *Búsquedas de tendencias*: devuelve datos de las búsquedas de tendencias más recientes. Se muestran en la sección Búsquedas de tendencias de Google Trends.
7. *Gráficos principales*: devuelve los datos de un tema determinado. Se muestra en la sección Gráficos principales de Google Trends.
8. *Sugerencia*: devuelve una lista de palabras clave sugeridas adicionales que se pueden usar para refinar una búsqueda de tendencia.

Estas características pueden llegar a ser extremadamente útiles a la hora de gestionar nuestra aplicación. Aunque las principales desventajas son demasiado grandes como para no tenerlas en cuenta:

1. *Disponibilidad geográfica*: debido a ciertas leyes y normativas, Google Trends no funciona en todos los países del mundo. Por desgracia, España está excluida de momento, aunque se tiene pensado cambiar la normativa o adaptarse a ella.
2. *Longevidad*: esta API al no ser oficial, no tiene garantía de estar funcionando siempre. Ya que llegará un momento donde se cambie el *Back-End* de Google e incidirá en el comportamiento de esta API.
3. *Limitaciones de llamadas*: El límite de peticiones no se conoce públicamente. Aunque los usuarios hicieron medidas aproximadas, donde se demostraba que 1400 solicitudes secuenciales en un período de tiempo de 4 horas daba lugar al límite (replicado en 2 redes), aunque se podía volver a hacer peticiones después de 60 segundos de tiempo de espera.

- **Tweepy [31]**

Es un paquete de código abierto que brinda una forma muy conveniente de acceder a la API de Twitter con Python. Tweepy incluye un conjunto de clases y métodos que representan los modelos de Twitter y los extremos de la API. Las principales características para nuestro uso son las siguientes:

1. *Obtención de tendencias cerca de una ubicación*: devuelve las 50 tendencias principales para un WOEID (código de país) específico, si la información de tendencias está disponible. La respuesta es una lista de objetos "trend" que codifican el nombre del tema de tendencia, el parámetro de consulta que se puede usar para buscar el tema en la búsqueda de Twitter y la URL de búsqueda de Twitter. Esta información se almacena en caché durante 5 minutos.
2. *Búsqueda de publicaciones*: devuelve una colección de publicaciones o *tweets* relevantes que coinciden con una consulta específica. Aunque la API de búsqueda no pretende ser una fuente exhaustiva de Tweets. No todos los Tweets se indexarán o estarán disponibles a través de la interfaz de búsqueda.

Estas características pueden llegar a cumplir las necesidades explicadas anteriormente, aunque también tenemos que tener en cuenta las desventajas que presenta esta API.

1. *Disponibilidad de versiones*: existen actualmente varias versiones de la API publicadas, cada una se accede de una manera diferente y tiene métodos diferentes. La documentación oficial de Twitter no está bien cuidada y no funciona de la misma manera que la aplicación web. Un sencillo

ejemplo de ello es que en la web se puede buscar tweets de un país en concreto, mientras que esta función no existe en la API.

2. *Historial pobre*: mientras que Google guarda todos los datos de interés sobre una tendencia, Twitter no, por lo que búsquedas antiguas no contendrán mucho valor al usuario en cuanto a la relevancia o interés que presenta un tópico.

#### 4.2.3 Conclusiones de los estudios realizados

En los apartados anteriores se definieron claramente las ventajas y desventajas del uso de ambas aplicaciones o APIs. Se puede concluir que, aunque Google Trends posea más funciones versátiles y con mayor peso de utilidad en cuanto a nuestras necesidades, sigue teniendo mayores desventajas a la hora de usarlo. La mayor de todas es que no esté disponible en todos los países. La plataforma de twitter en cambio se usa en casi todo el mundo, además se pueden llegar a implementar funciones parecidas con métodos propios y no tener que depender en un mayor nivel de una API externa.

En cuanto a los aspectos que se deberán tratar a la hora de definir las características o funcionalidades de la aplicación se definirán en la siguiente lista. Aunque se entrará más en detalle en la sección posterior.

- Las personas requieren información relevante, concisa y simplificada.
- La información debe ser de contenido finito, a la misma hora aportando al usuario su pleno entendimiento.
- Los usuarios deberán ser capaces de usar la aplicación en todos los dispositivos que tengan acceso web. Además deberá tener un diseño *responsive*, para que se adapte a la interfaz del usuario.
- Los usuarios tienen que ser capaces de navegar a través de distintas tendencias. Poder elegir tanto el país de búsqueda como la fecha para poder informarse debidamente.
- Los usuarios tienen que ser capaces de realizar búsquedas de distintos tópicos de interés y poder informarse con datos relacionados y actuales.
- El diseño será presentado mediante módulos o cartas. Cada carta tendrá información útil y de necesidad al usuario.
- Un módulo contendrá datos sobre la popularidad de la tendencia y su correspondiente gráfico.

- Otro módulo contendrá datos sobre palabras y *keywords* más usadas sobre la tendencia y su correspondiente gráfico.
- Otro módulo contendrá un sentimiento general y un gráfico visual sobre dicho comportamiento.
- También habrá tres módulos de noticias de interés general sobre la tendencia.
- Cada módulo deberá respetar tanto el idioma como la localización elegida por el usuario.
- Se deberá tener en cuenta que el diseño sea simple y agradable al usuario.
- Otro factor a tener en cuenta es el plan de peticiones que ofrece Twitter, el cual es muy limitado.

### 4.3 REQUISITOS DEL SISTEMA

Gracias a los estudios y conclusiones realizadas en el capítulo anterior, se puede detallar de una manera más sencilla los siguientes requerimientos o requisitos. Los requerimientos funcionales funcionan como servicios que puede prestar el sistema. Mientras que los requerimientos no funcionales especifican más bien la fiabilidad o el comportamiento de este.

#### 4.3.1 Requisitos Funcionales

R.F. 1 *Gestión de Trends*. El sistema deberá ser capaz de realizar llamadas al servicio de la API de Twitter para recopilar tendencias. En la API o su plataforma denominadas Trends o *trending topic*, «tendencia», «tema de tendencia» o «tema del momento» (TT).

R.F. 1.1 *Gestión de Países*. El sistema deberá ser capaz de manejar la petición concorde a diferentes países y almacenarlas adecuadamente.

R.F. 1.1.1 *Búsqueda por WOEID*. El sistema deberá ser capaz de buscar los diferentes países por el identificador que provee Yahoo y que utiliza la plataforma de Twitter. En este caso se denominan WOEIDs.

R.F. 1.2 *Gestión de la Fecha*. El sistema deberá ser capaz de manejar la respuesta generada por Twitter y poder diferenciarla por fecha, bajo el formato de yy-mm-dd.

R.F. 1.3 *Limpieza de Trends*. El sistema deberá poder limpiar los Trends repetidos. Ya que la mayoría de las veces Twitter no reconoce el mismo TT, casos como el uso de tilde, mayúsculas o si son una sucesión de varias palabras. Por ejemplo, «Balón», «balón», «balon» y «Balón de Oro» se referirán a la misma tendencia, pero Twitter los trata como Trends diferentes con datos respectivos diferentes.

R.F. 1.3.1 *Preferencias de Limpieza*. El sistema preferirá un nombre de tendencias más largo y por defecto si son iguales, preferirá el que tenga más popularidad.

R.F. 1.4 *Ordenación de los Trends*. El sistema deberá ser capaz de ordenar los TT bajo el atributo del volumen de popularidad proporcionado por Twitter.

R.F. 1.5 *Historial Inteligente*. Además de guardar los Trends ordenados, al usuario le será importante que se guarden y se vayan comparando los Trends viejos y los nuevos (mediante el atributo de la popularidad).

R.F. 2 *Gestión de Popularidad.* El sistema deberá ser capaz de guardar el atributo del volumen de popularidad en una lista, que fue anteriormente proporcionado por Twitter.

R.F. 2.1 *Cálculo de Media.* El sistema deberá poder sacar la media o la popularidad promedio a partir de la lista guardada anteriormente.

R.F. 2.2 *Cálculo de Moda.* El sistema deberá poder sacar la moda o la popularidad más grande a partir de la lista guardada anteriormente.

R.F. 2.3 *Gestión de Hora.* El sistema tiene que guardar en una lista la hora a la que se ha calculado el valor de la popularidad.

R.F. 2.4 *Gestión de Gráfico.* Se deberá proporcionar un gráfico a partir de los valores calculados anteriormente. Proporcionado en formato de área.

R.F. 2.4.1 *Eje X.* Se deberá proporcionar la lista de la hora a la que se ha calculado la popularidad como eje X del gráfico.

R.F. 2.4.2 *Eje Y.* Se deberá proporcionar la lista del vector de popularidad como el eje Y del gráfico.

R.F. 2.4.3 *Análisis del Gráfico.* Además del gráfico visual, se deberán proporcionar datos de valor. En este caso, el primer dato calculado y el último, además de mostrar la hora de su cálculo.

R.F. 3 *Gestión de Tweets.* El sistema deberá ser capaz de guardar como máximo alrededor de cien publicaciones o tweets de usuarios diferentes.

R.F. 3.1 *Filtrado de Tweets.* Se debe filtrar los tweets mediante la API de Twitter. Primero que las publicaciones tengan un nº mínimo de interacciones y que a la vez sean recientes, luego por el lenguaje del país y que sean publicaciones originales no repetidas.

R.F. 3.2 *Limpieza de Tweets.* Se debe limpiar los tweets como los nombres de usuarios, enlaces u otros caracteres que complicarían el análisis del texto.

R.F. 3.2.1 *Gestión de Stop Words.* El sistema deberá diferenciar las palabras significativas de las que no, frecuentemente denominadas como *stop words* también llamadas palabras vacías o palabras comunes, son palabras que no suelen aportar significado a la hora de analizar textos. Cada idioma tiene su propia lista de *stop words*.

R.F. 4 *Clasificación de Palabras y Gestión de Keywords.* Al tener un conjunto de palabras significativas, el sistema deberá disponerlas

al usuario como un top o clasificación ordenadas por el número de repeticiones en el texto. Por otro lado, el sistema deberá extraer *keywords* relevantes del conjunto de tweets mediante un algoritmo de extracción.

R.F. 4.1 *Filtrado de Keywords*. Se deberán eliminar *keywords* parecidas o repetidas.

R.F. 4.2 *Gestión de Gráfico*. Se deberá proporcionar un gráfico a partir de los valores calculados anteriormente. Mostrado como un gráfico de tipo barras radial.

R.F. 4.2.1 *Conjunto de Datos*. El conjunto de datos será proporcionado por la clasificación de seis palabras más repetidas de los tweets recogidos. Además deberán representarse como un porcentaje.

R.F. 5 *Gestión de Sentimiento General*. Al tener un conjunto de palabras significativas, el sistema deberá disponerlas al usuario como un sentimiento general, clasificando el conjunto de palabras como positivo, negativo o neutral.

R.F. 5.1 *Gestión de Gráfico*. Se deberá proporcionar un gráfico a partir de los valores calculados anteriormente. Mostrado como un gráfico de tipo burbujas con eje X y eje Y.

R.F. 5.1.1 *Eje X*. Este eje representará el sentimiento general de las publicaciones o tweets.

R.F. 5.1.2 *Eje Y*. Este eje representará la relación con la clasificación de palabras, calculadas anteriormente, de las publicaciones o tweets.

R.F. 5.1.3 *Tamaño de las Burbujas*. Se representará un tamaño mayor o menor, en base a la popularidad de los tweets o las publicaciones.

R.F. 6 *Gestión de Noticias*. Se deberán proporcionar tres noticias adecuadas a la tendencia. Se buscarán noticias parecidas gracias al nombre y las *keywords* relacionadas a la tendencia.

R.F. 7 *Gestión de Rutas*. El sistema deberá disponer de rutas de navegación por cada país y fecha correspondiente a las tendencias.

R.F. 8 *Sistema de Búsqueda*. El sistema deberá disponer de un sistema de búsqueda que proporcione tópicos y datos.

R.F. 9 *Gestión de Guardado*. El sistema deberá poder guardar las tendencias y que el usuario pueda acceder a diferentes países o fechas.

#### 4.3.2 Requisitos No Funcionales

- R.N. 1 *Diseño.* El diseño debe ser sencillo, minimalista, pero a la vez poder dar toda la información que el usuario requiera. Además debe ser *responsive*, enfocada en el uso móvil.
- R.N. 2 *Disponibilidad.* El sistema debe intentará estar disponible siempre y actualizar el contenido relevante cada hora. Confirando en *plataforma as a service*, «plataforma como servicio» ([PaaS](#)) a la hora de construir, correr y mantener la aplicación.
- R.N. 3 *Arquitectura de capas.* El sistema deberá tener una base de datos, una capa de negociación o gestión y por último una capa de presentación o interfaz.
- R.N. 4 *Sistema de Logs.* El sistema deberá tener un sistema de Logs, el cual recopile información fundamental o registros de la actividad del servidor.
- R.N. 5 *Tests.* El sistema deberá estar *testeado* para poder comprobar su correcto funcionamiento. Además tendrá reglas de escritura de código específicas, considerado como buenas prácticas para dicho lenguaje de programación.

## 4.4 PRODUCT BACKLOG

Después de listar los distintos requisitos, podemos visualizar más fácilmente la lista de Historias de Usuario que se trabajará y desarrollará posteriormente. La siguiente tabla contiene las diferentes Historias de Usuario, donde la estimación del esfuerzo (E) está expresada mediante los Puntos de Historia (1 a 10) y la prioridad (P) representada del 1 a 3, siendo el 1 el más prioritario para el Dueño del Producto, es decir, los tutores.

ID	Titulo de la Historia	E	P
H.U. 1	Cualquier usuario puede usar la aplicación sin necesidad de registrarse.	1	1
H.U. 2	El usuario debe poder visualizar las diez tendencias más populares por defecto.	5	1
H.U. 3	El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar el país como parámetro.	5	1
H.U. 4	El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar la fecha como parámetro.	5	1
H.U. 5	El usuario puede buscar sus propios tópicos y visualizarlos del mismo modo que las tendencias.	4	2
H.U. 6	El usuario debe saber, por medio de alertas, si el servicio o los parámetros no han funcionado.	3	3
H.U. 7	El usuario tiene que poder navegar intuitivamente, es decir por medio de gestos de <i>scroll</i> , por la página.	7	1
H.U. 8	El usuario debe poder analizar datos de interés sobre la popularidad.	8	1
H.U. 9	El usuario debe poder visualizar las palabras más repetidas o comunes mediante un porcentaje, a partir de los tweets recogidos referentes a la tendencia.	8	1
H.U. 10	El usuario debe poder visualizar las <i>keywords</i> , a partir de los tweets recogidos referentes a la tendencia.	8	1
H.U. 11	El usuario puede analizar los sentimientos generales mediante un porcentaje mostrado.	9	2
H.U. 12	El usuario debe poder ver tres noticias más actuales relacionadas con la tendencia y poder acceder a ellas.	10	1

Cuadro 4.4: Títulos de Historias de Usuario.

Además algunas Historias de Usuario pueden tener diferentes sub-niveles, como se detalla en la siguiente tabla.

ID	Titulo de la Historia	E	P
H.U. 8.1	El usuario debe poder visualizar la popularidad que está teniendo la tendencia por medio de una gráfica de área.	8	1
H.U. 9.1	El usuario verá los datos representados mediante un gráfico de radial de barras.	8	1
H.U. 11.1	El usuario debe poder ver los sentimientos generales, a partir de los tweets recogidos referentes a la tendencia, mediante una gráfica de burbujas.	9	2

Cuadro 4.5: Subniveles de Títulos de Historias de Usuario.

#### 4.5 SPRINT BACKLOG

Quedando los *Product Backlog* concluidos, ahora podemos dividirlos en los distintos *Sprint* acordados previamente.

##### 4.5.1 Sprint 1

ID	Título de la Historia	E	P
H.U. 1	Cualquier usuario puede usar la aplicación sin necesidad de registrarse.	1	1
H.U. 6	El usuario debe saber, por medio de alertas, si el servicio o los parámetros no han funcionado.	3	3

Cuadro 4.6: Títulos del Sprint 1.

H.U. 1	Cualquier usuario puede usar la aplicación sin necesidad de registrarse.
	<b>Descripción:</b> El usuario debe poder entrar a la página web y poder visualizar su contenido sin ningún tipo de restricción. Esto debido a que el contenido aportado por la aplicación web debe centrarse al público general.
<b>Estimación:</b> 1	<b>Prioridad:</b> 1
	<b>Pruebas de aceptación:</b> <ul style="list-style-type: none"> <li>■ Entrar a cualquier URL dinámico de la pagina y poder obtener una respuesta.</li> <li>■ Ante un error de carga de contenido o una URL mal escrita, el usuario deberá saber el error específico.</li> </ul>
	<b>Tareas:</b> <ul style="list-style-type: none"> <li>■ Instalar Vue.js, sus dependencias y el enrutador.</li> <li>■ Instalar un lint adecuado para Vue.</li> <li>■ Crear rutas y composiciones que devuelvan una página con contenido.</li> </ul>
	<b>Observaciones:</b> <ul style="list-style-type: none"> <li>■ Tener en cuenta rutas y errores por defecto.</li> </ul>

Cuadro 4.7: Descomposición de la Historia de Usuario 1.

H.U. 6	El usuario debe saber, por medio de alertas, si el servicio o los parámetros no han funcionado.
<b>Descripción:</b> El usuario debe poder entrar a la página web y poder si está funcionando o no. En caso de fallo, deberá poder ver un mensaje de error propiamente explicado el fallo en concreto.	
<b>Estimación:</b> 3	<b>Prioridad:</b> 3
<b>Pruebas de aceptación:</b>	
<ul style="list-style-type: none"> <li>■ Poder entrar a diferentes rutas por erróneas o cuando el servicio esté caído y diferenciar diferentes errores.</li> <li>■ Diferenciar errores como la carga de contenido, el propio funcionamiento del servidor o contenido erróneo.</li> </ul>	
<b>Tareas:</b>	
<ul style="list-style-type: none"> <li>■ Crear rutas por defecto, con alertas definidas para diferentes tipos de errores.</li> </ul>	
<b>Observaciones:</b>	
<ul style="list-style-type: none"> <li>■ Tener en cuenta todos los parámetros a la hora de realizar peticiones al servidor, como el parámetro de <i>status</i> de una petición.</li> </ul>	

Cuadro 4.8: Descomposición de la Historia de Usuario 6.

## 4.5.2 Sprint 2

ID	Titulo de la Historia	E	P
H.U. 2	El usuario debe poder visualizar las diez tendencias más populares por defecto.	5	1

Cuadro 4.9: Títulos del Sprint 2.

H.U. 2	El usuario debe poder visualizar las diez tendencias más populares por defecto.
	<b>Descripción:</b> El usuario debe poder visualizar el contenido de tendencias de un país en el día actual en la página de defecto, sin tener que poner ningún parámetro.
Estimación: 5	Prioridad: 1
	<b>Pruebas de aceptación:</b> <ul style="list-style-type: none"> <li>■ Entrar a la página web por defecto que haya una petición correcta al servidor.</li> <li>■ Que el servidor devuelva las tendencias de un país y fecha del día actual.</li> </ul>
	<b>Tareas:</b> <ul style="list-style-type: none"> <li>■ Instalar Tweepy, sus dependencias y aprender las diferentes consultas a la API de Twitter.</li> <li>■ Filtrar las tendencias parecidas o iguales. Diferenciando tildes, minúsculas, mayúsculas o simplemente frases más largas.</li> <li>■ Manejar las diferentes entidades a la hora de construir el contenido a devolver para el usuario.</li> <li>■ Manejar una Base de Datos Mongo y definir distintos módulos para su correcto funcionamiento.</li> </ul>
	<b>Observaciones:</b> <ul style="list-style-type: none"> <li>■ La Base de Datos se tendrá que actualizar cada hora.</li> <li>■ Al momento de actualizar la Base de Datos se debe realizar una Unión y Diferencia Simétrica de los datos viejo y nuevos. Con la Unión se actualizan los datos viejos y con la Diferencia Simétrica, agregamos los datos nuevos junto a los viejos.</li> </ul>

Cuadro 4.10: Descomposición de la Historia de Usuario 2.

## 4.5.3 Sprint 3

ID	Titulo de la Historia	E	P
H.U. 3	El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar el país como parámetro.	5	1
H.U. 4	El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar la fecha como parámetro.	5	1

Cuadro 4.11: Títulos del Sprint 3.

H.U. 3	El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar el país como parámetro.
	<b>Descripción:</b> El usuario debe poder entrar a la página web y poder visualizar el contenido, aportando el parámetro del país donde quiera buscar la tendencia.
Estimación: 5	<b>Prioridad:</b> 1
<b>Pruebas de aceptación:</b>	
<ul style="list-style-type: none"> <li>■ Entrar al URL dinámico de la página como país como parámetro y que el servidor devuelva el objeto correspondiente.</li> <li>■ Ante un error de carga de contenido o una URL mal escrita, el usuario deberá saber el error específico.</li> </ul>	
<b>Tareas:</b>	
<ul style="list-style-type: none"> <li>■ Configuración de respectivos enrutadores. Tanto en la parte Front-end y Back-end.</li> <li>■ Se deben poder buscar en la Base de Datos por países, con su fecha correspondiente. De modo que ya deben haber objetos creados con país como valor identificativo.</li> </ul>	
<b>Observaciones:</b>	
<ul style="list-style-type: none"> <li>■ Ninguna.</li> </ul>	

Cuadro 4.12: Descomposición de la Historia de Usuario 3.

H.U. 4	El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar el país como parámetro.			
<b>Descripción:</b> El usuario debe poder visualizar las diez tendencias más populares, pudiendo seleccionar la fecha como parámetro.				
<b>Estimación:</b> 5	<b>Prioridad:</b> 1			
<b>Pruebas de aceptación:</b>				
<ul style="list-style-type: none"> <li>■ Entrar al URL dinámico de la página como fecha como parámetro y que el servidor devuelva el objeto correspondiente.</li> <li>■ Ante un error de carga de contenido o una URL mal escrita, el usuario deberá saber el error específico.</li> </ul>				
<b>Tareas:</b>				
<ul style="list-style-type: none"> <li>■ Configuración de respectivos enrutadores. Tanto en la parte Front-end y Back-end.</li> <li>■ Se deben poder buscar en la Base de Datos por fecha, con su país correspondiente. De modo que ya deben haber objetos creados con fecha como valor identificativo.</li> </ul>				
<b>Observaciones:</b>				
<ul style="list-style-type: none"> <li>■ Ninguna.</li> </ul>				

Cuadro 4.13: Descomposición de la Historia de Usuario 4.

## 4.5.4 Sprint 4

ID	Titulo de la Historia	E	P
H.U. 5	El usuario puede buscar sus propios tópicos y visualizarlos del mismo modo que las tendencias.	4	2

Cuadro 4.14: Títulos del Sprint 4.

H.U. 5	El usuario puede buscar sus propios tópicos y visualizarlos del mismo modo que las tendencias.
	<b>Descripción:</b> El usuario puede buscar sus propios tópicos, mediante un buscador general, y visualizarlos del mismo modo que las tendencias. Con las mismas características que las tendencias, menos la popularidad, que solo se puede calcular si es una tendencia, un dato proveniente de Twitter.
Estimación: 4	<b>Prioridad:</b> 2
	<b>Pruebas de aceptación:</b> <ul style="list-style-type: none"> <li>■ Que el usuario identifique el buscador y pueda escribir sus propios tópicos que desee buscar.</li> <li>■ Que el usuario pueda cambiar de país como parámetro al buscador para los tópicos.</li> </ul>
	<b>Tareas:</b> <ul style="list-style-type: none"> <li>■ Adaptar el diseño para el buscador en Front-end.</li> <li>■ Adaptar la ruta correspondiente en forma de <i>query</i> y que el servidor sea capaz de procesarla.</li> </ul>
	<b>Observaciones:</b> <ul style="list-style-type: none"> <li>■ El tópico no se guarda en la Base de Datos, sino que se devuelve cada vez que se inicia una búsqueda.</li> </ul>

Cuadro 4.15: Descomposición de la Historia de Usuario 5.

## 4.5.5 Sprint 5

ID	Titulo de la Historia	E	P
H.U. 7	El usuario tiene que poder navegar intuitivamente, es decir por medio de gestos de <i>scroll</i> , por la página.	7	1

Cuadro 4.16: Títulos del Sprint 5.

H.U. 7	<p>El usuario tiene que poder navegar intuitivamente, es decir por medio de gestos de <i>scroll</i>, por la página.</p> <p><b>Descripción:</b> El usuario tiene que poder navegar intuitivamente. Tanto haciendo <i>scroll</i> verticalmente o horizontalmente por la página web. Además el contenido debe desplazarse automáticamente al centro de la pantalla, ocupando el mayor tamaño de espacio disponible. Todo esto por medio de gestos, tanto en una pantalla grande o en un móvil.</p>
<b>Estimación:</b> 7	<b>Prioridad:</b> 1
<p><b>Pruebas de aceptación:</b></p> <ul style="list-style-type: none"> <li>▪ Que el usuario pueda hacer <i>scroll</i> verticalmente y el contenido se mueva verticalmente dependiendo del gesto del usuario.</li> <li>▪ Que el usuario pueda hacer <i>scroll</i> horizontalmente y el contenido se mueva horizontalmente dependiendo del gesto del usuario.</li> </ul>	
<p><b>Tareas:</b></p> <ul style="list-style-type: none"> <li>▪ Adaptar el diseño en Front-end. Esto se hará gracias al <i>framework</i> Tailwind CSS. Consiguiendo que el gesto del usuario sea un <i>scroll</i> automático y el contenido se centre en medio de la pantalla siempre después de procesar cualquier gesto.</li> <li>▪ Adaptar Vue para que la página web pueda proporcionar contenido dinámico, es decir, un vector de tendencias que puede tener diferente tamaño cada vez que se haga una petición al servidor.</li> </ul>	
<p><b>Observaciones:</b></p> <ul style="list-style-type: none"> <li>▪ El diseño, al ser de cartas o módulos es fácilmente adaptable, pero hay que tener en cuenta las diferentes versiones de escritorio o móvil.</li> </ul>	

Cuadro 4.17: Descomposición de la Historia de Usuario 7.

## 4.5.6 Sprint 6

ID	Titulo de la Historia	E	P
H.U. 8	El usuario debe poder analizar datos de interés sobre la popularidad.	8	1
H.U. 8.1	El usuario debe poder visualizar la popularidad que está teniendo la tendencia por medio de una gráfica de área.	8	1

Cuadro 4.18: Títulos del Sprint 6.

H.U. 8	El usuario debe poder analizar datos de interés sobre la popularidad.
	<p><b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario serán datos de interés sobre la popularidad de la tendencia. Datos como la moda, la media y el primer o último dato calculado.</p>
Estimación: 8	Prioridad: 1
<p><b>Pruebas de aceptación:</b></p> <ul style="list-style-type: none"> <li>■ Que el usuario pueda informarse debidamente de la media o la popularidad promedio de la tendencia.</li> <li>■ Que el usuario pueda informarse debidamente de la moda o a popularidad más grande alcanzada de la tendencia.</li> <li>■ Que el usuario pueda informarse debidamente del primer y último volumen de popularidad calculados de la tendencia.</li> </ul>	
<p><b>Tareas:</b></p> <ul style="list-style-type: none"> <li>■ Adaptar el diseño en Front-end al diseño de la carta y desarrollar funciones para que el número tenga un separador de millares para una lectura más fácil.</li> <li>■ Crear el modelo correspondiente en la Base de Datos.</li> </ul>	
<p><b>Observaciones:</b></p> <ul style="list-style-type: none"> <li>■ Ninguna.</li> </ul>	

Cuadro 4.19: Descomposición de la Historia de Usuario 8.

H.U. 8.1	El usuario debe poder visualizar la popularidad que está teniendo la tendencia por medio de una gráfica de área.
<b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario será la gráfica de área que representará la propia popularidad de la tendencia. El eje X será la hora a la que se ha calculado la popularidad y el eje Y será la popularidad de la tendencia en cuestión.	
<b>Estimación:</b> 8	<b>Prioridad:</b> 1
<b>Pruebas de aceptación:</b>	
<ul style="list-style-type: none"> <li>▪ Que el usuario pueda informarse debidamente con el contenido gráfico sobre la popularidad de una tendencia en un momento dado.</li> </ul>	
<b>Tareas:</b>	
<ul style="list-style-type: none"> <li>▪ Adaptar el diseño en Front-end. Gracias a la librería ApexCharts.js, podremos mostrar al usuario un gráfico solamente necesitando dos listas, el eje X y el eje Y que habrán sido previamente calculados.</li> <li>▪ Se adaptará el gráfico al diseño de la carta, además se apagará toda la interactividad con él porque su función solo es gráfica.</li> </ul>	
<b>Observaciones:</b>	
<ul style="list-style-type: none"> <li>▪ El diseño tiene que ser <i>responsive</i>, afectando también al gráfico.</li> </ul>	

Cuadro 4.20: Descomposición de la Historia de Usuario 8.1.

## 4.5.7 Sprint 7

ID	Titulo de la Historia	E	P
H.U. 9	El usuario debe poder visualizar las palabras más repetidas o comunes mediante un porcentaje, a partir de los tweets recogidos referentes a la tendencia.	8	1
H.U. 9.1	El usuario verá los datos representados mediante un gráfico de radial de barras.	8	1
H.U. 10	El usuario debe poder visualizar las <i>keywords</i> , a partir de los tweets recogidos referentes a la tendencia.	8	1

Cuadro 4.21: Títulos del Sprint 7.

H.U. 9	El usuario debe poder visualizar las palabras más repetidas o comunes, a partir de los tweets recogidos referentes a la tendencia.
	<p><b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario será visualizar las palabras más repetidas o comunes de una tendencia. Mostrado en porcentaje de aparición respecto a todas las publicaciones o tweets.</p>
Estimación: 8	Prioridad: 1
	<p><b>Pruebas de aceptación:</b></p> <ul style="list-style-type: none"> <li>■ El usuario debe informarse debidamente de las palabras más repetidas a través de cálculos de porcentaje sobre las apariciones de las palabras en los tweets.</li> </ul>
	<p><b>Tareas:</b></p> <ul style="list-style-type: none"> <li>■ Adaptar el diseño en Front-end y que conjunte con los demás elementos.</li> <li>■ Filtrar las palabras significativas, eliminando las que pertenezcan al grupo de las <i>stop words</i> o palabras vacías, es decir las que no suelen aportar significado a la hora de analizar textos. Posteriormente sólo se enseñaran las seis palabras más comunes.</li> <li>■ Crear el modelo correspondiente en la Base de Datos.</li> </ul>
	<p><b>Observaciones:</b></p> <ul style="list-style-type: none"> <li>■ El diseño tiene que ser <i>responsive</i>.</li> </ul>

Cuadro 4.22: Descomposición de la Historia de Usuario 9.

H.U. 9.1	El usuario verá los datos representados mediante un gráfico de radial de barras.
<b>Descripción:</b> Otro contenido único, que aportará más significado a la hora de visualizar las palabras más repetidas o comunes de una tendencia será un gráfico de radial de barras, donde cada barra representará la palabra y el porcentaje de sus repeticiones.	
<b>Estimación:</b> 8	<b>Prioridad:</b> 1
<p><b>Pruebas de aceptación:</b></p> <ul style="list-style-type: none"> <li>■ El usuario debe informarse debidamente de las palabras más repetidas a través de un gráfico de radial de barras.</li> <li>■ El usuario debe poder comparar todas las palabras repetidas. También saber cual es la que tiene más valor y la que menos.</li> </ul>	
<p><b>Tareas:</b></p> <ul style="list-style-type: none"> <li>■ Adaptar el diseño en Front-end, gracias a la librería ApexCharts.js.</li> <li>■ Se debe configurar el gráfico de tal manera, que cada palabra y su valor tengan una representación única.</li> </ul>	
<p><b>Observaciones:</b></p> <ul style="list-style-type: none"> <li>■ El diseño tiene que ser <i>responsive</i>, afectando también al gráfico.</li> </ul>	

Cuadro 4.23: Descomposición de la Historia de Usuario 9.1.

H.U. 10	El usuario debe poder visualizar las <i>key-words</i> , a partir de los tweets recogidos referentes a la tendencia.
<b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario será visualizar las <i>keywords</i> más relevantes sobre los textos recogidos de las publicaciones o tweets.	
<b>Estimación:</b> 8	<b>Prioridad:</b> 1
<p><b>Pruebas de aceptación:</b></p> <ul style="list-style-type: none"> <li>■ El usuario debe informarse de cuales son las <i>key-words</i> más relevantes de una tendencia.</li> <li>■ Las <i>keywords</i> deben expresar al usuario una perspectiva de lo que es relevante o de importancia en el conjunto de las publicaciones o tweets de la tendencia.</li> </ul>	
<p><b>Tareas:</b></p> <ul style="list-style-type: none"> <li>■ Adaptar el diseño en Front-end.</li> <li>■ Configurar el algoritmo <i>Yet Another Keyword Extractor</i> (<a href="#">YAKE</a>), adaptándolo a extraer un conjunto de textos que serán parecidos, ya que son publicaciones sobre un mismo tópico.</li> <li>■ Filtrar el texto antes de extraer las <i>keywords</i> con el algoritmo <a href="#">YAKE</a>.</li> <li>■ Filtrar las <i>keywords</i> parecidas o iguales, luego enseñar solo las tres más relevantes.</li> </ul>	
<p><b>Observaciones:</b></p> <ul style="list-style-type: none"> <li>■ El diseño tiene que ser <i>responsive</i>.</li> </ul>	

Cuadro 4.24: Descomposición de la Historia de Usuario 10.

## 4.5.8 Sprint 8

ID	Titulo de la Historia	E	P
H.U. 11	El usuario puede analizar los sentimientos generales mediante un porcentaje mostrado.	9	2
H.U. 11.1	El usuario debe poder ver los sentimientos generales, a partir de los tweets recogidos referentes a la tendencia, mediante una gráfica de burbujas.	9	2

Cuadro 4.25: Títulos del Sprint 8.

H.U. 11	El usuario puede analizar los sentimientos generales mediante un porcentaje mostrado.	
<b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario será visualizar el sentimiento general que tienen los textos recogidos de las publicaciones o tweets. En este caso, será mostrado mediante un porcentaje.		
<b>Estimación:</b> 9	<b>Prioridad:</b> 2	
<b>Pruebas de aceptación:</b>		
<ul style="list-style-type: none"> <li>■ El usuario debe informarse de cual es el sentimiento general de una tendencia y saber cual es el porcentaje correspondiente de los sentimientos positivos, negativos y neutrales.</li> </ul>		
<b>Tareas:</b>		
<ul style="list-style-type: none"> <li>■ Mostrar al usuario tres tipos de porcentaje (positivos, negativos y neutrales). El porcentaje será calculado mediante el tamaño de la burbuja, ya que el tamaño o radio representa su relevancia.</li> <li>■ Crear el modelo correspondiente en la Base de Datos.</li> </ul>		
<b>Observaciones:</b>		
<ul style="list-style-type: none"> <li>■ El diseño tiene que ser <i>responsive</i> y adaptarse al contenido gráfico con sus colores correspondientes.</li> </ul>		

Cuadro 4.26: Descomposición de la Historia de Usuario 11.

H.U. 11.1	El usuario debe poder ver los sentimientos generales, a partir de los tweets recogidos referentes a la tendencia, mediante una gráfica de burbujas.
<b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario será visualizar el sentimiento general que tienen los textos recogidos de las publicaciones o tweets. El análisis de sentimientos será basados en reglas de léxicos y se mostrará mediante un gráfico.	
<b>Estimación:</b> 9	<b>Prioridad:</b> 2
<b>Pruebas de aceptación:</b>	
<ul style="list-style-type: none"> <li>■ El usuario debe informarse de cual es el sentimiento general de una tendencia de una manera visual o gráfica. Se le mostrará al usuario un gráfico de burbujas.</li> </ul>	
<b>Tareas:</b>	
<ul style="list-style-type: none"> <li>■ Configurar la herramienta <i>Valence Aware Dictionary and sEntiment Reasoner</i> (<a href="#">VADER</a>) y adaptar el léxico de cada país.</li> <li>■ Mostrar al usuario tres tipos de burbuja (Positivo, Negativo y Neutral). Cada burbuja tendrá un tamaño diferente dependiendo de su relevancia, mediante la popularidad de la publicación o si hay muchos sentimientos parecidos.</li> <li>■ Mostrar un posicionamiento de la burbuja relevante. El eje X será el sentimiento general de una publicación. Mientras el eje Y representará las relación con la clasificación de palabras relevantes, que fueron calculadas anteriormente.</li> </ul>	
<b>Observaciones:</b>	
<ul style="list-style-type: none"> <li>■ El diseño tiene que ser <i>responsive</i>.</li> </ul>	

Cuadro 4.27: Descomposición de la Historia de Usuario 11.1.

## 4.5.9 Sprint 9

ID	Titulo de la Historia	E	P
H.U. 12	El usuario debe poder ver tres noticias más actuales relacionadas con la tendencia y poder acceder a ellas.	10	1

Cuadro 4.28: Títulos del Sprint 9.

H.U. 12	<p>El usuario debe poder ver tres noticias más actuales relacionadas con la tendencia y poder acceder a ellas.</p> <p><b>Descripción:</b> Uno de los contenidos principales sobre una tendencia para el usuario será visualizar las distintas noticias relevantes que pueden tener alguna relación las tendencias.</p> <p><b>Estimación:</b> 10      <b>Prioridad:</b> 1</p> <p><b>Pruebas de aceptación:</b></p> <ul style="list-style-type: none"> <li>■ El usuario debe informarse de tres noticias con editoriales diferentes, pero que tengan afinidad respecto a la tendencia.</li> <li>■ El usuario podrá ver una foto, un título, la editorial, la fecha de publicación y algún texto sobre el artículo o noticia.</li> </ul> <p><b>Tareas:</b></p> <ul style="list-style-type: none"> <li>■ Desarrollar <i>web scraping</i> o raspado web, una técnica utilizada para extraer información de sitios web. En nuestro caso se utilizará para extraer noticias relevantes desde Google News y posteriormente desde cada página del artículo.</li> <li>■ Se buscarán artículos relacionados mediante el nombre de la tendencia y algunas palabras relevantes que se extrajeron anteriormente, con el objetivo de precisar la búsqueda.</li> <li>■ Crear el modelo correspondiente en la Base de Datos.</li> </ul> <p><b>Observaciones:</b></p> <ul style="list-style-type: none"> <li>■ El diseño tiene que ser <i>responsive</i> y adaptarse al contenido, tanto con la foto o el texto.</li> </ul>
---------	---

Cuadro 4.29: Descomposición de la Historia de Usuario 12.

# 5

## DISEÑO

---

En este capítulo se pretende abordar la principal tarea del diseño o la planificación de la estructura idónea sobre las necesidades que se han ido explicando en capítulos anteriores. Para ello se intentará plantear las decisiones tomadas desde un nivel alto, eligiendo la estructura o arquitectura correcta, bajando gradualmente hasta encajar todos los componentes, con el objetivo de obrar un sistema escalable y ágil que proporcione la solución a los problemas supuestos del cliente.

### 5.1 ARQUITECTURA PLANTEADA

La necesidades que se han ido explicando en capítulos anteriores se pueden dividir en diferentes aplicaciones, funcionalidades o divisiones de capas. Cada aplicación debe funcionar cumpliendo un único objetivo, pero finalmente deben coaccionar como una sola aplicación para el usuario final. Las distintas funcionalidades que surgen ante las necesidades que se explicaron anteriormente se pueden dividir en tres tipos (operaciones de datos, su gestión y su visualización). Al tratar de construir una aplicación que requiere muchas capas horizontales independientes y a la vez que puedan interactuar entre ellas, el patrón de software que surge ante esta necesidad es el patrón arquitectónico en capas. Además de ser un patrón muy común en el ámbito web, es bastante sencillo de entender e implementar. Cada capa tiene un papel determinado y una responsabilidad específica dentro de la aplicación, dichas capas pueden cooperar e interactuar entre si, pero no dependen una de otra. Se debe saber que una capa de nivel superior puede interactuar con la capa de nivel inferior, pero no al revés. [32]

Por norma general, las capas de una aplicación tienen la posibilidad de residir en una sola maquina física (misma capa) o pueden estar compartidas sobre diferentes máquinas (n-capas). En nuestro caso, la solución más óptima sería n-capas, debido a la posibilidad de mejorar la escalabilidad y optimización. Además, al dividir el proyecto por capas se mejora la flexibilidad y se puede usar en diferentes plataformas como servicio, esto último debido a que es fácilmente portable. Gracias a ello se puede dar uso de PaaS como Vercel, Heroku o MongoDB Atlas, consiguiendo numerosas ventajas ya que se facilita la configuración, el despliegue, las tareas de automatización y el escalado.

La arquitectura de n-capas se suele dividir en tres. Las necesidades diferenciales que se han planteado son operar datos, gestionarlos y visualizarlos. Estas necesidades se traducen a distintas capas en la arquitectura elegida. Desde el nivel bajo hasta el más alto, la capa que se encargaría de operar con los datos se domina Capa de Base de Datos, luego la capa encargada de gestionarlos se domina Capa de Dominio o Negocio y por último la capa encargada de visualizarlos es dominada como Capa de Presentación. Cada una de ellas tiene funcionalidades y características propias, además pueden llegar a subdividirse en distintos niveles para la correcta estructuración de la aplicación.

Cada capa consigue independencia lógica ya que funciona mediante una aplicación independiente. La indecencia física la podemos conseguir usando las diferentes plataformas como servicio PaaS planteadas anteriormente. La comunicación de distintas peticiones será bidireccional, aunque siempre siguiendo la norma de que una capa inferior no puede llegar a usar los servicios de una capa superior.

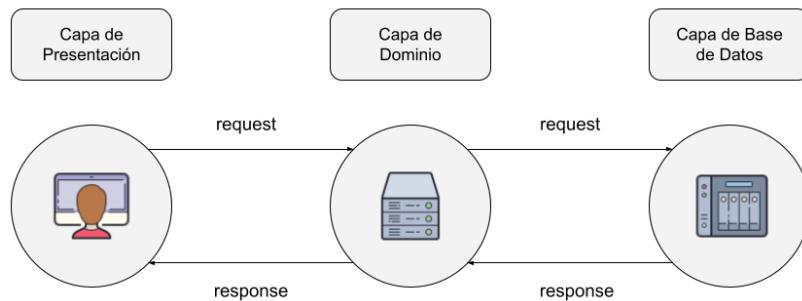


Figura 5.1: Diagrama conceptual de la arquitectura.

## 5.2 CAPA DE BASE DE DATOS

La capa de base de datos se encarga de las operaciones de los propios datos de la aplicación. Esta capa es invisible para el usuario, ya que su funcionamiento no le incumbe. En esta capa residen los datos y además se acceden a los mismos, mediante solicitudes de almacenamiento o recuperación de información desde la capa de dominio o negocio. Comúnmente esta capa suele pertenecer a la parte Back-end del sistema.

Al tener planteado el *Sprint Backlog* es importante estudiar y analizar las herramientas que compondrán la estructuración de esta capa. Incluso aunque se haya dado ligeras pinceladas de un estudio técnico (4.2.2), sigue siendo crucial reiterar y volver a incidir en este apartado antes de adentrarse en el diseño de dicha capa.

### 5.2.1 Estudio técnico

Los dos grandes pilares o modelos que existen actualmente en las bases de datos son denominados SQL (base de datos relacional) y NoSQL (base de datos no relacional). La principal diferencia es que NoSQL no requiere una estructura sólida y definida para trabajar con los datos, lo cual para nuestra aplicación es conveniente ya que requiere de flexibilidad a la hora de operar datos, debido a que estos cambian y se actualizan en un determinado intervalo de tiempo.

Se ha elegido MongoDB como el sistema de base de datos NoSQL, el cual es orientado a documentos y de código abierto. Es de los sistemas más fiables que existen actualmente, además de su conveniencia por ser orientado a objetos, simple, dinámico y escalable. [33]

Se ha elegido el lenguaje Python principalmente por el estudio que se realizó anteriormente en la sección 4.2.2, debido al peso que suponían ambas librerías. Por lo que se ha decidido continuar usando dicho lenguaje y adaptar la base de datos. Pese a que este lenguaje no ha demostrado ser el más eficiente, sigue siendo muy versátil a la hora de trabajar. Con esto, se han seguido las buenas prácticas y se ha usado el *driver* oficial de MongoDB, llamado PyMongo, consiguiendo de manera nativa poder almacenar datos en documentos tipo JSON. Además, Python tiene extensas bibliotecas que procesan directamente los formatos de datos de tipo JSON y BSON. [33]

En cuanto a la plataforma como servicio que se ha elegido en este caso será MongoDB Atlas. Esto debido a que posee un plan gratuito, aunque con muchas limitaciones. La base de datos en la nube funcionaría de la misma manera que en la versión local, pero teniendo en cuenta las limitaciones técnicas.

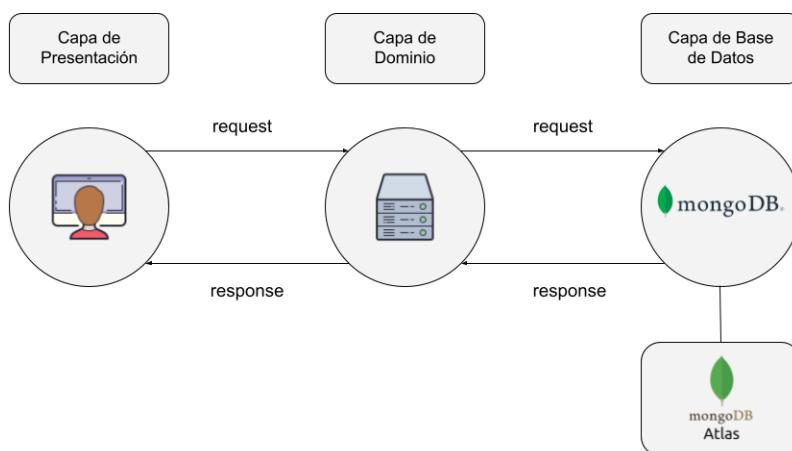


Figura 5.2: Diagrama conceptual de la arquitectura, con la Capa de Base de Datos detallada.

### 5.2.2 Modelado de datos

El modelo de datos principal será compuesto por dos clases. Las dos clases harán referencia al mismo modelo pero tendrán dos conceptos diferentes, uno para objetos nuevos y otro para su actualización. De esta manera tenemos dos clases, las cuales se llaman *Country* y *CountryUpdate*. La diferencia es que la primera clase creará un id nuevo para cada objeto y la segunda no.

Ambas poseen los mismo atributos y entidades. Están formadas por país y fecha, como los parámetros principales para poder clasificar el objeto y buscarlo en la base de datos. Luego está la lista de diferentes tendencias, la cual se va actualizando a lo largo del día.

Muchos de los atributos pueden llegar a ser opcionales, como por ejemplo la imagen del artículo, la cual si no existe se cargará otra por defecto desde la interfaz. En el siguiente diagrama se puede visualizar el conjunto del modelado, aunque las diferentes entidades serán explicadas en la capa de dominio o negocio.

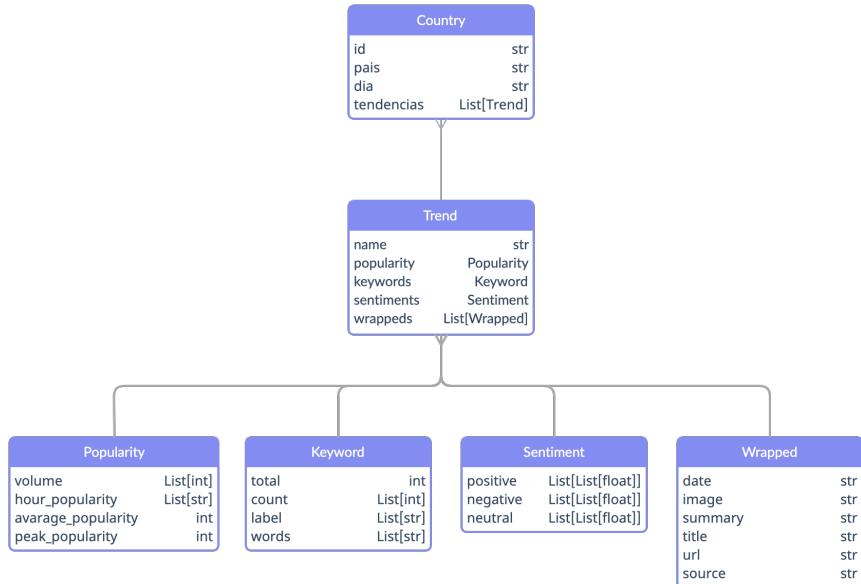


Figura 5.3: Diagrama de relación de la base de datos.

### 5.2.2.1 Entidad Tendencia

Esta entidad representa el modelado de datos de las historias de usuario cubiertas en el Sprint 2 (4.5.2). En ellas el usuario debe poder visualizar datos de diez tendencias como máximo.

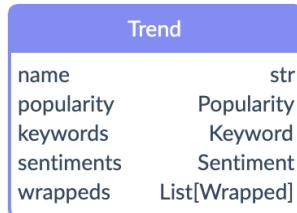


Figura 5.4: Diagrama de relación de Tendencia.

Para poder representar debidamente la historia de usuario se debe ir calculando progresivamente los siguientes valores:

*name*: el nombre de la tendencia.

*popularity*: el objeto popularidad de la tendencia.

*keywords*: el objeto *keywords* de la tendencia.

*sentiments*: el objeto sentimientos general de la tendencia.

*wrappeds*: lista de objetos noticia de la tendencia.

### 5.2.2.2 Entidad Popularidad

Esta entidad representa el modelado de datos de las historias de usuario cubiertas en el Sprint 6 (4.5.6). En ellas el usuario debe poder visualizar datos de interés sobre la popularidad y también una gráfica de área.

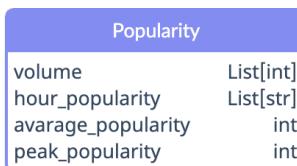


Figura 5.5: Diagrama de relación de Popularidad.

Para poder representar debidamente las H.U. se debe ir calculando progresivamente los siguientes valores:

*volume*: el volumen de popularidad que está teniendo la tendencia.

*hour\_popularity*: la hora en la que fue registrada el volumen.

*avarage\_popularity*: popularidad media de la tendencia.

*peak\_popularity*: popularidad más alta de la tendencia.

### 5.2.2.3 Entidad Palabras más Comunes y Keywords

Esta entidad representa el modelado de datos las historias de usuario cubiertas en el Sprint 7 ([4.5.7](#)). En ellas el usuario debe poder visualizar datos de interés sobre las palabras más comunes, las *keywords* y una gráfica de radial de barras correspondiente.

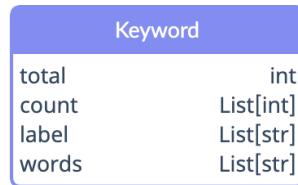


Figura 5.6: Diagrama de relación de Palabras más Comunes y Keywords.

Para poder representar debidamente las H.U. se debe ir calculando progresivamente los siguientes valores:

- total*: el volumen de publicaciones recogidas de la tendencia.
- count*: las repeticiones de palabras comunes.
- label*: las palabras comunes en cuestión.
- words*: las *keywords* extraídas.

### 5.2.2.4 Entidad Sentimiento General

Esta entidad representa el modelado de datos de las historias de usuario cubiertas en el Sprint 8 ([4.5.8](#)). En ellas el usuario debe poder visualizar datos de interés sobre el sentimiento general de los tweets o publicaciones y una gráfica de burbujas correspondiente.

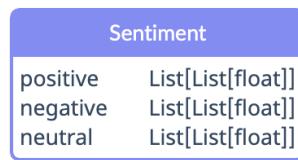


Figura 5.7: Diagrama de relación de Sentimiento General.

Para poder representar debidamente las H.U. se debe ir calculando progresivamente los siguientes valores:

- positive*: datos sobre el sentimiento positivo.
- negative*: datos sobre el sentimiento negativo.
- neutral*: datos sobre el sentimiento neutro.

### 5.2.2.5 Entidad Noticas Relacionadas

Esta entidad representa el modelado de datos las historias de usuario cubiertas en el Sprint 9 ([4.5.9](#)). En ellas el usuario debe poder visualizar datos de interés sobre las noticias o artículos relacionados sobre la tendencia.

Wrapped	
date	str
image	str
summary	str
title	str
url	str
source	str

Figura 5.8: Diagrama de relación de Noticas Relacionadas.

Para poder representar debidamente la H.U. se debe ir calculando progresivamente los siguientes valores:

- date*: fecha de publicación de la noticia.
- image*: imagen relacionada de la noticia.
- summary*: texto relacionado de la noticia.
- title*: título de la noticia.
- url*: enlace de la noticia.
- source*: editorial de la noticia.

## 5.3 CAPA DE DOMINIO O NEGOCIO

La capa de dominio o negocio permite la interacción entre la capa de base de datos y la capa de presentación. Es la capa encargada de gestionar tanto las distintas peticiones recibidas desde la capa de presentación, como los datos almacenados en la capa de base de datos. Comúnmente esta capa suele pertenecer a la parte Back-end del sistema.

### 5.3.1 Estudio técnico

Al plantear la aplicación en la capa de dominio o negocio, surgen ciertas pautas o normativas que conviene seguir. Surge la denominación *application programming interface*, «interfaz de programación de aplicaciones» ([API](#)) para las interacciones que surgen entre la capa de presentación y la capa actual. Otra denominación que conviene saber es *representational state transfer*, «transferencia de estado representacional» ([REST](#)), el cual es un conjunto de límites a la hora de implementar una [API](#). [\[34\]](#)

Cuando un usuario envía una petición a través de una API REST o RESTful, esta transfiere una representación del estado del recurso solicitado al solicitante o al extremo. Generalmente la información se envía a través de *Hypertext Transfer Protocol*, «Protocolo de Transferencia de Hipertexto» ([HTTP](#)) en formato *JavaScript Object Notation*, «notación de objeto de JavaScript» ([JSON](#)), un lenguaje popular y fácilmente entendible. [34]

La mayoría de las interacciones se pueden resumir en el nemónico de las cuatro funciones del almacenamiento persistente llamado *Create, Read, Update and Delete*, «Crear, Leer, Actualizar y Borrar» ([CRUD](#)). Las funciones [CRUD](#) implican el uso de las operaciones [HTTP](#), las cuales son PUT, GET, POST, DELETE o PATCH.

Habiendo definido propiamente las pautas de una [API REST](#), podemos proceder a elegir el *microframework* que cumpla y revele las operaciones salientes mediante la interfaz [REST](#) y [websockets](#). Para ello se ha elegido de nuevo el lenguaje Python, debido a que existe compatibilidad entre la capa de base de datos y la de dominio o negocio y además por su versatilidad a la hora de trabajar con diferentes librerías que serán extremadamente útiles para la realización de diversas historias de usuario.

Actualmente, en Python existen *frameworks* que funcionan con el estándar *Web Server Gateway Interface* ([WSGI](#)) o *Asynchronous Server Gateway Interface* ([ASGI](#)). El último mencionado es el más actual y utiliza corrutinas introducidas en las versiones más recientes de Python, lo que mejora el uso de la CPU en servidores web intensivos de E/S.

Las principales implementaciones de servidor [ASGI](#) existentes actuales son Uvicorn, Hypercorn y Daphne. Los tres son fácilmente implementables, pero el más estable es Uvicorn, además usa una implementación escrita directamente en el lenguaje C, lo cual lo vuelve más rápido y eficiente. [35]

Tras haber elegido el servidor, podemos continuar con el *microframework*. Actualmente existe una cantidad variada de ellos, como FastAPI, Starlette o Django, pero finalmente me he decantado por la primera opción. FastAPI es mucho más ligero que Django y además funciona de una manera parecida a Starlette, pero le añade funcionalidades como validación o documentación. [36]

La implementación de las rutas fue realizada con el objeto *router*, el cual tiene la función de declarar todas las rutas con decoradores y añadir el prefijo que llevaran dichas rutas. Posteriormente se crea la documentación automática.

The screenshot shows the FastAPI documentation for the 'countries' endpoint. At the top, there's a header with the FastAPI logo (0.1.0) and an OAS3 badge. Below the header, there's a link to /openapi.json. The main content area has a title 'countries' with a collapse/expand arrow (^). Below the title is a list of API endpoints:

- PUT** /country/ Update Country
- POST** /country/ Create Country
- GET** /country/countries List Countries
- GET** /country/{name}/{date} Find Country
- DELETE** /country/{name}/{date} Delete Country
- GET** /country/search/ Read Item
- PUT** /country/{id} Update Country Id
- DELETE** /country/{id} Delete Country Id

At the bottom, there's a section titled 'Schemas' with a collapse/expand arrow (^).

Figura 5.9: Documentación generada por FastAPI.

En cuanto a la plataforma como servicio ([PaaS](#)) he elegido Heroku, la cual tiene opciones gratuitas y está basada en contenedores. La gran ventaja de Heroku, aparte de tener un plan gratuito, es que tiene múltiple soporte de lenguajes y entre ellos Python. Las rutas declaradas y la de por defecto funcionarían de la misma manera que localmente. [37]

Además otra ventaja crucial, la cual es requerida para el correcto funcionamiento de la aplicación, es que un proceso se ejecute cada determinado intervalo de tiempo. Esto es debido a que necesitamos que se actualice la información de las tendencias, por lo que localmente tendríamos que configurar un *cron* ejecutando un *script* de Python y en Heroku tenemos un *Scheduler*, el cual tiene el mismo comportamiento. Dicho proceso ha sido configurado para que se ejecute cada hora y el *script* en cuestión recopilará toda la información necesaria.

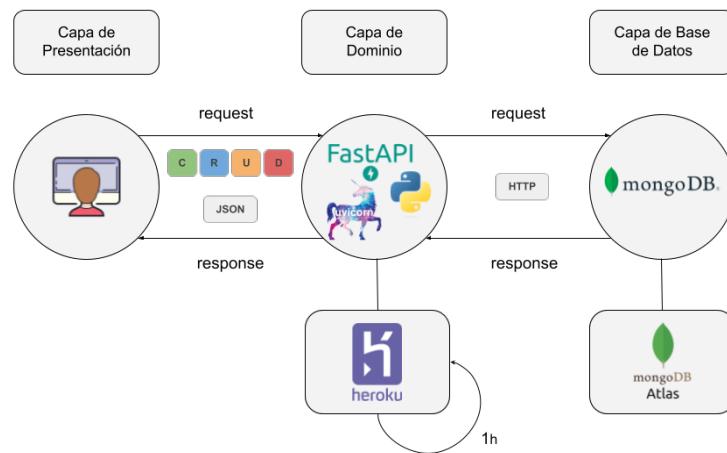


Figura 5.10: Diagrama conceptual de la arquitectura, con la Capa de Dominio o Negocio y Base de Datos detallada.

### 5.3.2 Diagramas de interacción de distintas rutas

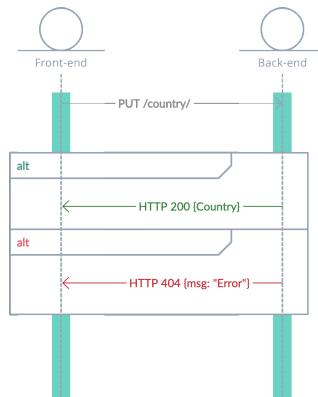


Figura 5.11: Diagrama interacción de rutas: Actualización de Country.

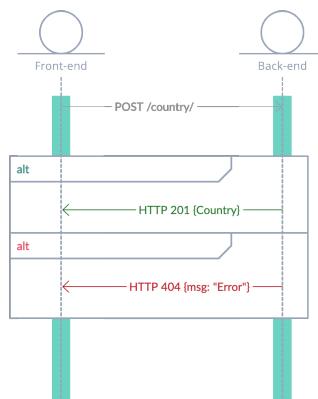


Figura 5.12: Diagrama interacción de rutas: Creación de Country.

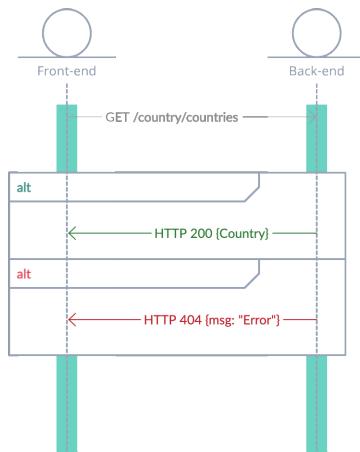


Figura 5.13: Diagrama interacción de rutas: *Lectura de lista de Country*.

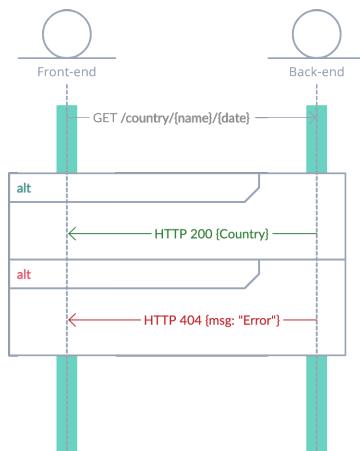


Figura 5.14: Diagrama interacción de rutas: *Lectura de Country por parámetros*.

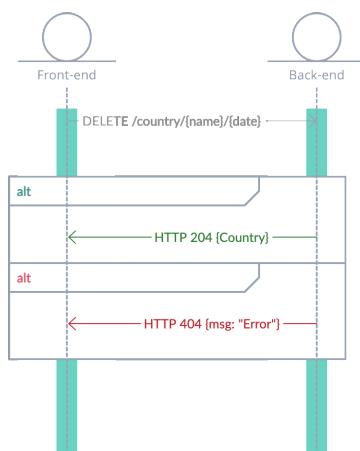


Figura 5.15: Diagrama interacción de rutas: *Eliminación de Country por parámetros*.

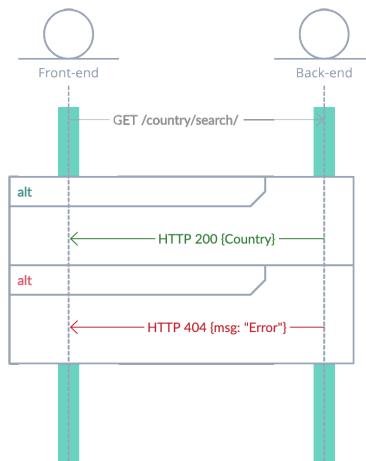


Figura 5.16: Diagrama interacción de rutas: *Lectura de la búsqueda de Country*.

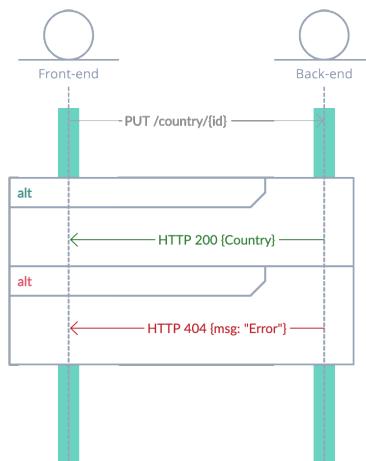


Figura 5.17: Diagrama interacción de rutas: *Actualización de Country por id*.

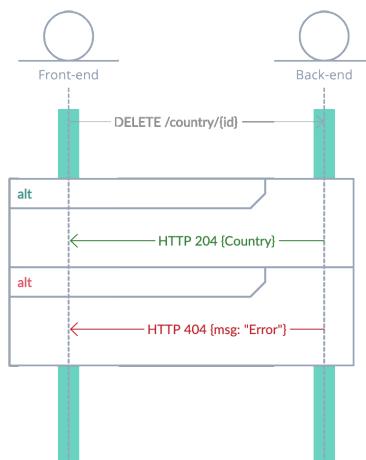


Figura 5.18: Diagrama interacción de rutas: *Eliminación de Country por id*.

## 5.4 CAPA DE PRESENTACIÓN

La capa de presentación se encarga de visualizar el contenido de la aplicación, donde el usuario puede interactuar directamente. De este modo esta capa será la única visible para el usuario. Comúnmente esta capa se suele denominar como Front-end.

### 5.4.1 Estudio técnico

Al buscar el *framework* idóneo para la visualización de la aplicación surgen muchas posibilidades. Una entre ellas fue Vue, esta opción destaca por su rendimiento, escalabilidad y buena documentación. A la vez es mucho más ligera que sus competidores y además ofreciendo las mismas características, como por ejemplo el uso del *Document Object Model (DOM)* virtual, el cual tiene sus ventajas a la hora de renderizar los mismos componentes de una página. En nuestro caso los componentes como las cartas se podrían beneficiar de ello. [38]

Uno de los principales objetivos de la aplicación, que ya se había planteado antes, es que el contenido se pueda visualizar por medio de módulos o cartas. Por lo que al diseñar la propia aplicación, tendremos una vista principal y cada una de las cartas estará dividida en una subvista. Cada subvista tendrá sus propiedades, sus propias funciones y clases CSS. La mayoría de las clases estarán implementadas directamente a mano, menos las más básicas o convenientes ya que usaré el *framework* TailwindCSS para agilizar el trabajo.

En cuanto a la plataforma como servicio (*PaaS*) para la capa de presentación me he decantado por Vercel, realmente tenía mucho parecido a otras plataformas como Netlify, aunque tras usar ambas plataformas he podido concluir que Vercel funciona más eficientemente y de una manera más simple. Además posee integración con Vue, lo cual me ha permitido exportar fácilmente el proyecto.

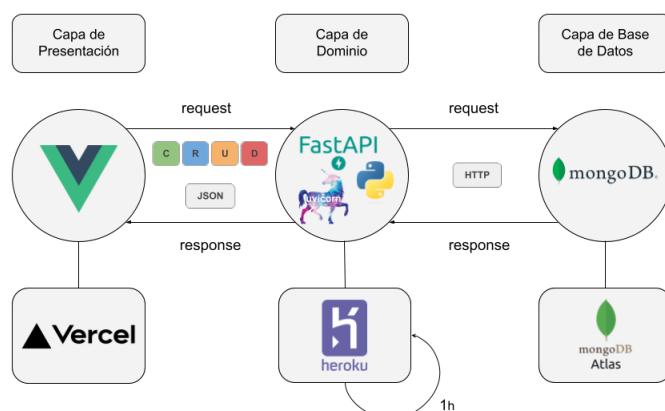


Figura 5.19: Diagrama conceptual de la arquitectura, con la Capa de Dominio o Negocio, Base de Datos y Presentación detallada.

El contenido de la vista principal se adapta mediante parámetros y rutas. Aunque existan diferentes rutas en la capa de presentación, todas ellas devuelven la misma vista principal con los mismos componentes, a no ser que haya un error. En el siguiente diagrama se explica el funcionamiento de las distintas rutas, tanto la ruta por defecto, la ruta de la búsqueda de un tópico o la ruta por parámetros.

- *Ejemplo de ruta por defecto:*  
`https://(...)/`
- *Ejemplo de ruta de búsqueda:*  
`https://(...)/search?name=Spain&query=baloncesto`
- *Ejemplo de ruta por parámetros:*  
`https://(...)/Spain/22-10-24`

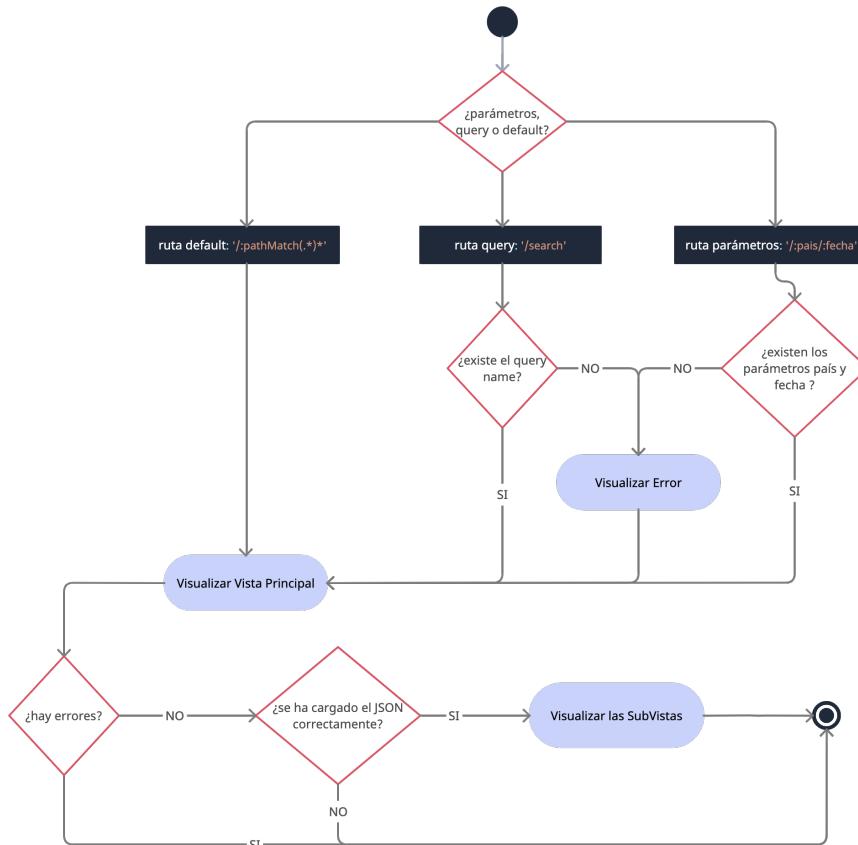


Figura 5.20: Diagrama de actividad.

La ruta por defecto cargará como parámetro el país España y la fecha del día actual. Las demás rutas son ejecutadas al interactuar con los componentes de distintos formularios de la página. En este caso son dos, un formulario de selección (fecha y país) y otro formulario de búsqueda (los tópicos).



Figura 5.21: Formularios de selección y búsqueda.

Cada uno funcionan de una manera independiente, mediante eventos por click. Aunque si precisan de la información que hay en cada formulario, es decir, si el usuario cambia el parámetro país, entonces se carga una ruta con la fecha y el país seleccionado. Esto ocurre de la misma manera seleccionando la fecha o buscando un tópico, ambos necesitan la información del país seleccionado para procesar la ruta.

#### 5.4.2 Transcurso de la Interfaz de Usuario

La vista principal es la única vista que se le llega mostrar al usuario, por lo que es importante mostrar el transcurso que ha tenido su propio diseño. Aunque como hemos explicado la vista principal se subdivide y cada subvista tiene sus propias clases de CSS, en español «Hojas de estilo en cascada». Cabe mencionar, que también se ha empleado una biblioteca que fue crucial llamada TailwindCSS.

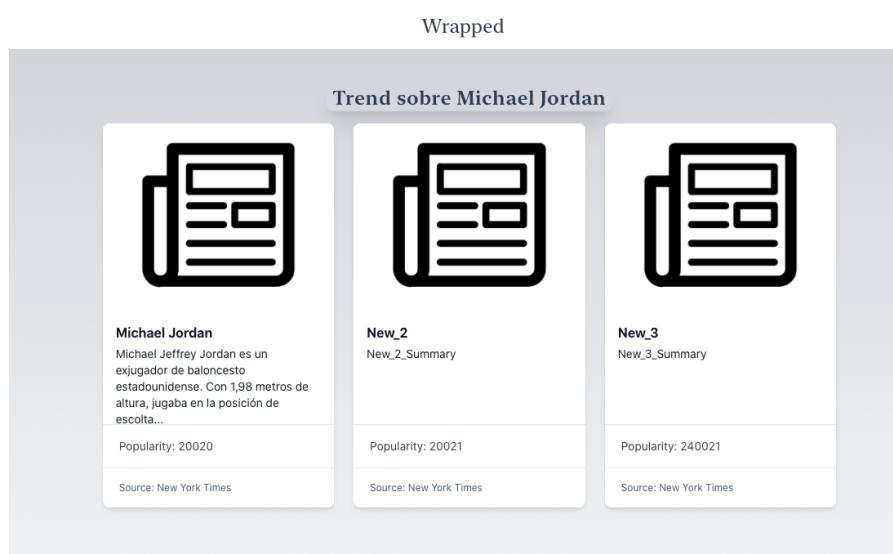


Figura 5.22: Primer boceto de la aplicación.

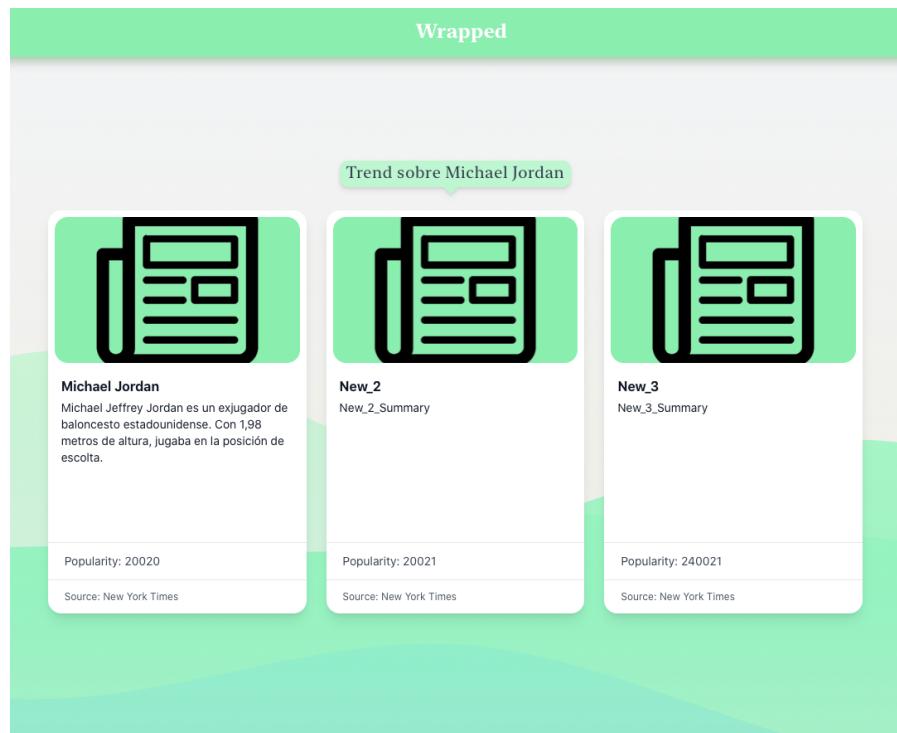


Figura 5.23: Segundo boceto más detallado.

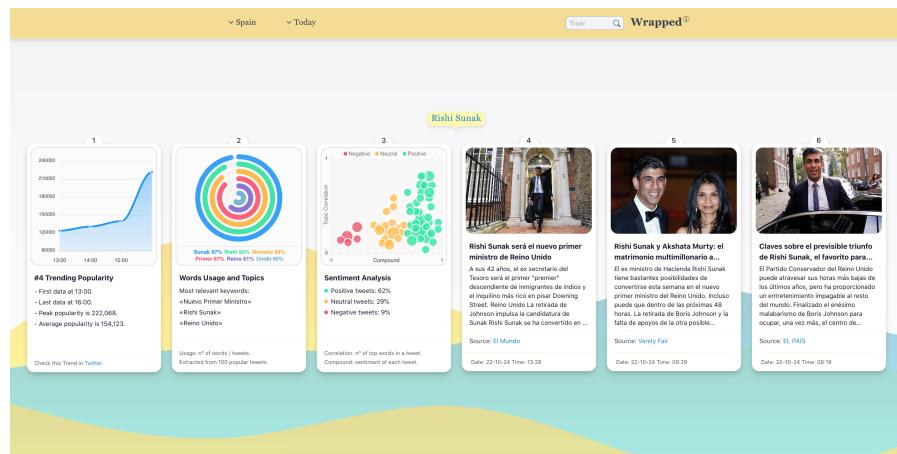


Figura 5.24: Concepto final de la aplicación.

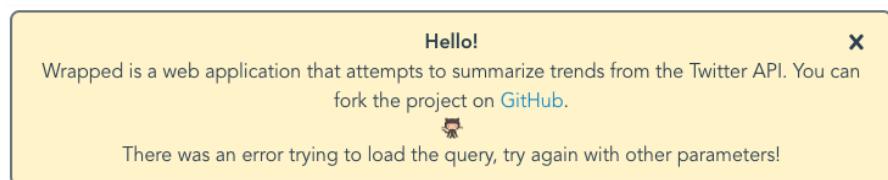


Figura 5.25: Concepto del error en la visualización de parámetros.

### 5.4.3 Vista Principal

La vista principal contiene todas las subvistas y componentes necesarios para el funcionamiento de la interfaz. Cada subvista funciona como un componente esencial dentro de la vista principal. Al cargar correctamente los datos desde la capa de dominio o negocio, estos se representan en la vista principal y a la vez cada entidad se representa con una subvista diferente.

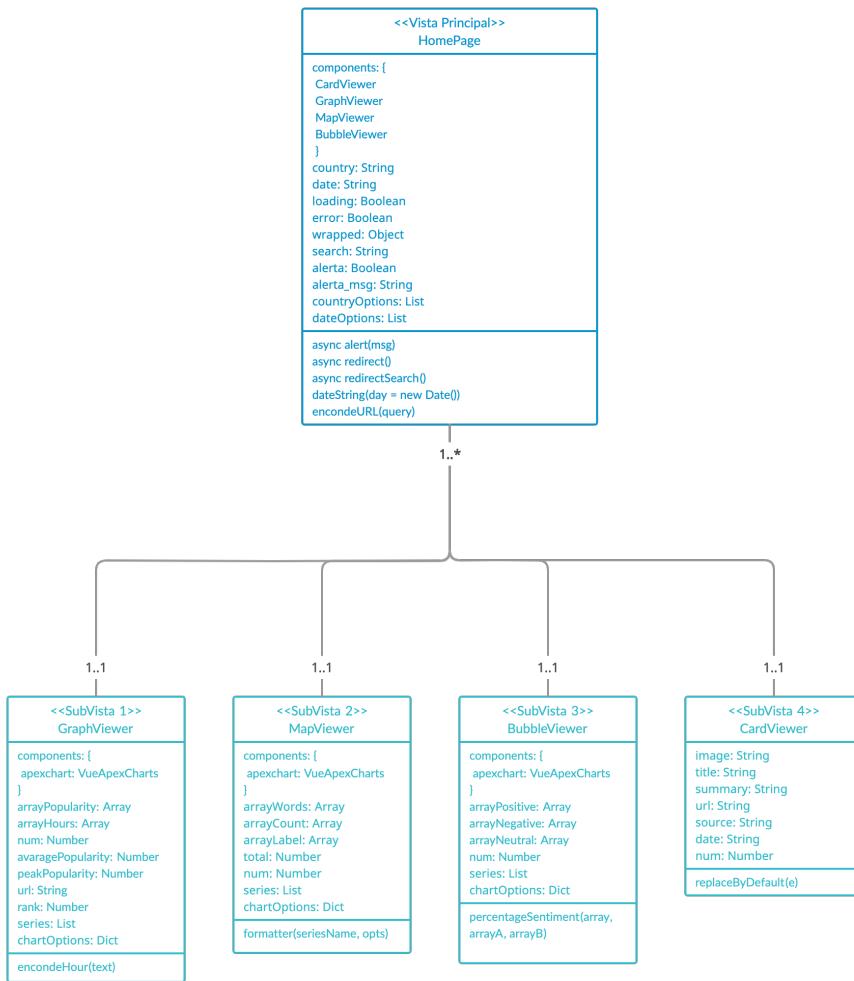


Figura 5.26: Diagrama de clase de las Vistas.

Cada subvista es llamada desde la vista principal, cada una con propiedades y funciones diferentes, ya que la representación de cada vista debe ser única y esencial.

La Vista principal representa la visualización de la historia de usuario que fue contemplada en el Sprint 2 (4.5.2) y esta relaciona con la entidad explicada en la sección 5.2.2.1. Esta vista es llamada *HomePage* y está formada por *components*, los cuales son las diferentes subvistas que se ha mencionado anteriormente. Los parámetros de las

distintas rutas están referenciados por *country*, *date*, *countryOptions* y *dateOptions*, además las funciones de esta vista tienen el objetivo de construir los distintos formularios y los datos de la tendencia se cargarán en *wrapped*.

También existen propiedades más complementarias como *num* y *rank*, para enumerar las tarjetas o cartas y el *ranking* de la tendencia, respectivamente. Las funciones de alerta o los distintos atributos como *loading*, existen para enriquecer las animaciones de la página y para mostrar el cuadro de dialogo que se ha mencionado anteriormente (5.25). Finalmente, la función de *encodeURL* codifica el nombre de la tendencia para redirigirlo a la plataforma Twitter mediante un enlace.

#### 5.4.4 Subvista Popularidad

Esta subvista representa la visualización de las historias de usuario que fueron contempladas en el Sprint 6 (4.5.6). En ellas el usuario debe poder visualizar datos de interés sobre la popularidad y también un gráfica de área. La implementación de la gráfica se consigue mediante la librería ApexCharts.

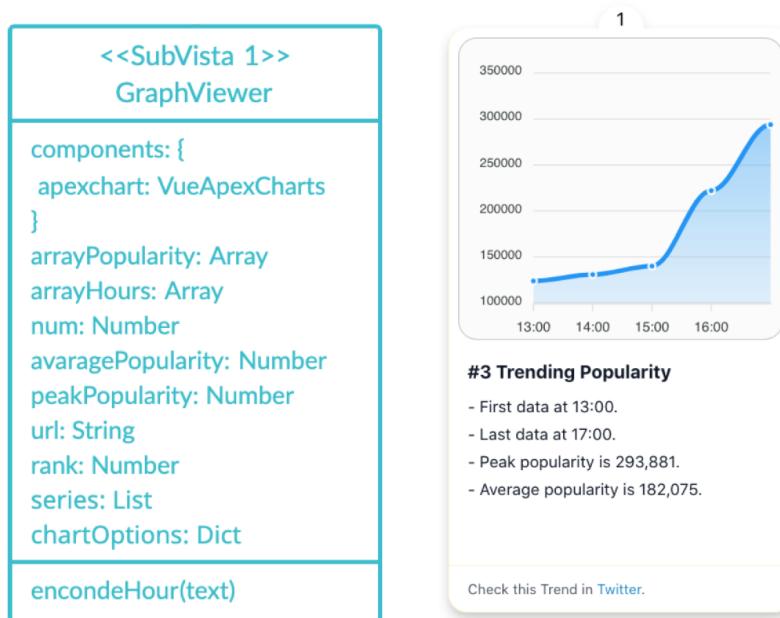


Figura 5.27: Estructura y Diseño de la primera subvista.

Los diferentes valores que posee esta subvista están relacionados con la entidad Popularidad (5.2.2.2), a excepción de propiedades de la gráfica y atributos complementarios como *url*, para redirigir a la plataforma de Twitter. La función debe transformar el texto en formato *Date* de JavaScript.

#### 5.4.5 Subvista Palabras más Comunes y Keywords

Esta subvista representa la visualización de las historias de usuario que fueron contempladas en el Sprint 7 (4.5.7). En ellas el usuario debe poder visualizar datos de interés sobre las palabras más comunes, las *keywords* y un gráfico de radial de barras correspondiente. La implementación de la gráfica se consigue mediante la librería Apex-Charts.

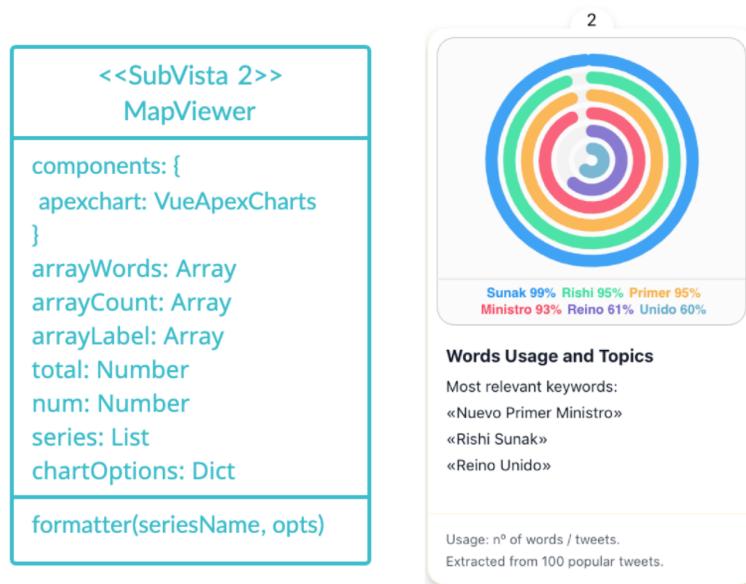


Figura 5.28: Estructura y Diseño de la primera subvista.

Los diferentes valores que posee esta subvista están relacionados con la entidad explicada en la sección 5.2.2.3. La función adapta el formato a los porcentajes.

Esta subvista tuvo un diseño inicial que fue descartado por problemas de diseño y también por posible confusión a la hora de interpretar el gráfico. El diseño tuvo problemas a la hora de encajar el gráfico en su encuadre y no quedaba muy claro el porcentaje de uso de las distintas palabras.

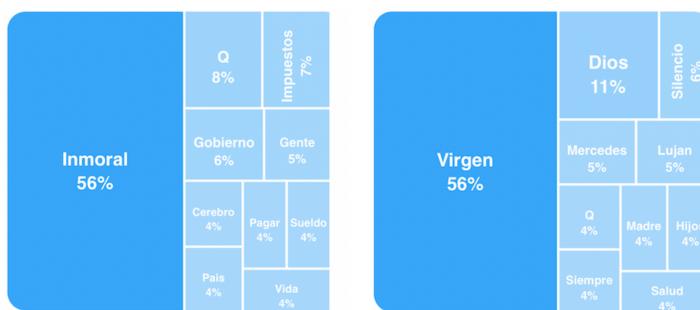


Figura 5.29: Diseño descartado de la segunda subvista.

#### 5.4.6 Subvista Sentimiento General

Esta subvista representa la visualización las historias de usuario que fueron contempladas en el Sprint 8 ([4.5.8](#)). En ellas el usuario debe poder visualizar datos de interés sobre el sentimiento general de los tweets o publicaciones y un gráfico de burbujas correspondiente. La implementación de la gráfica se consigue mediante la librería ApexCharts.

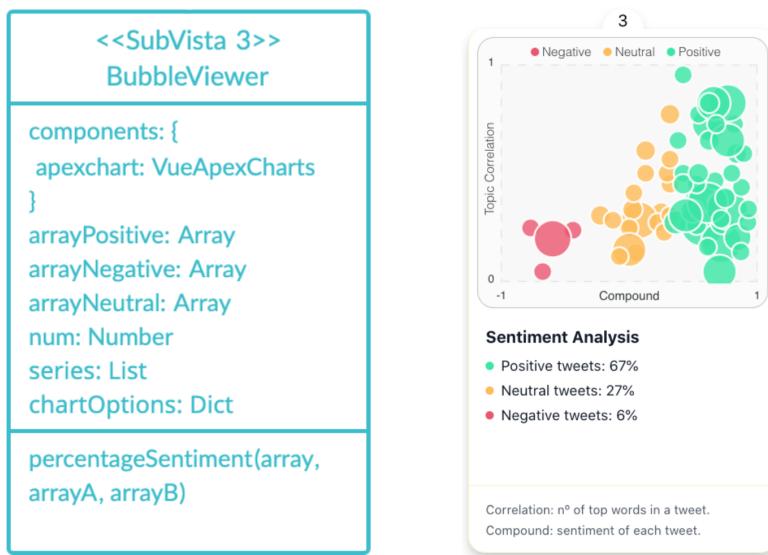


Figura 5.30: Estructura y Diseño de la tercera subvista.

Los diferentes valores que posee esta subvista están relacionados con la entidad explicada en la sección [5.2.4](#). La función calcula el porcentaje dependiendo del radio de las burbujas.

#### 5.4.7 Subvista Noticas Relacionadas

Esta subvista representa la visualización las historias de usuario que fueron contempladas en el Sprint 9 ([4.5.9](#)). En ellas el usuario debe poder visualizar datos de interés sobre las noticias o artículos relacionados sobre la tendencia.

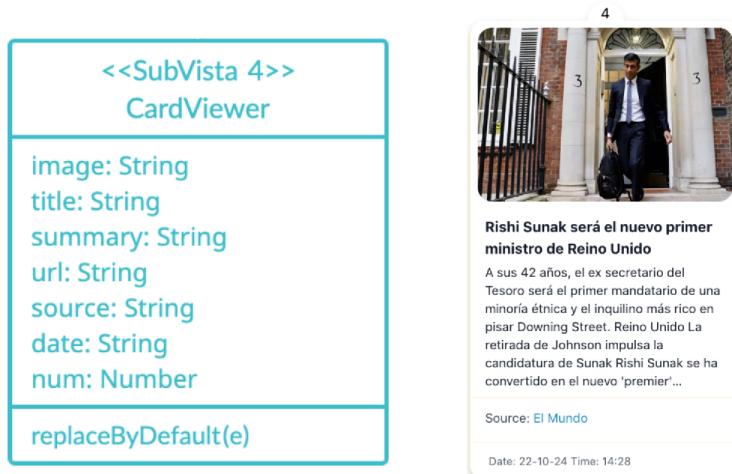


Figura 5.31: Estructura y Diseño de la última subvista.

Los diferentes valores que posee esta subvista están relacionados con la entidad explicada en la sección 5.2.2.5. La función devuelve una imagen por defecto si la carga de la imagen de la noticia falla.



# 6

## TERMINOLOGÍA DEVOPS

Al haber consolidado la metodología de desarrollo SCRUM en los capítulos anteriores (4.1), sabemos que dicha metodología pertenece a un ámbito de desarrollo iterativo e incremental, donde el proyecto se planifica en diversos bloques temporales. Debido a esto es muy frecuente que el equipo de desarrollo tenga numerosas entregas, ya que debe proporcionar una versión incrementada en cada iteración, esto también se suele denominar como despliegue.

Generalmente para proporcionar un despliegue el equipo de desarrollo se divide en dos ámbitos o departamentos. El primero tiene la función de desarrollo o la propia codificación de la aplicación, reconocido como «development» o «Dev» del inglés. El segundo se encarga de la parte operacional o el gestionado de versiones y la parte física del despliegue, reconocido también como «operations» o «Ops». [39]

El problema más usual es que ambos departamentos tienden a estar separados, por lo que el flujo de comunicación tiende a fallar y a tener retrasos a la hora de proporcionar el despliegue. Bajo esta premisa, surge una solución en 2009 denominada como *DevOps*, la cual tiene como propósito unificar estos dos ámbitos [39]

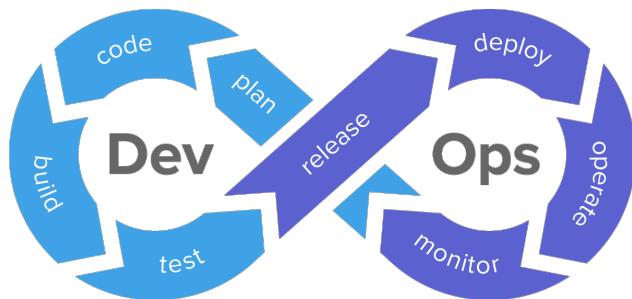


Figura 6.1: Imagen explicativa del concepto DevOps [40].

En resumen, *DevOps* trata de romper estos silos organizativos establecidos y intenta centrarse en la mejora de calidad de la entrega al cliente. Para ello, *DevOps* se apoya en tres principios o también llamadas «the three ways», las cuales se deben implementar de forma consecutiva. [41]

1. *Mejora del flujo de entrega:* consiste en general valor al producto, mediante entregas del software de una forma temprana y periódica. Tratando de dividir los procesos en tareas más pequeñas o automatizando todas las tareas que sean posibles, con esto se

consigue un aumento del flujo de trabajo global, ya que en este caso un proceso no ralentiza a otro.

Un ejemplo de este principio puede ser la ejecución de tests u otras operaciones cada vez que haya cambios en el repositorio del código fuente.

2. *Buen uso de los ciclos de feedback:* consiste en el uso de la retroalimentación con el objetivo de optimizar los procesos, es decir, cada vez que se lleve a cabo una entrega conviene tener un *feedback* rápido y constante. Con esto se consigue que los ciclos de retroalimentación obtengan la información necesaria para que las correcciones se puedan aplicar de manera continua. Un ejemplo de ello puede ser la monitorización del sistema y la comunicación ante la detección de errores.
3. *Asentar el aprendizaje y la mejora continua:* se centra en aplicar todo lo anteriormente realizado y aprendido para implementar un sistema de mejora continua. De esta manera se consigue innovar el producto pero sin criminalizar los errores mediante el aprendizaje y las pruebas continuas.  
Un ejemplo puede llegar a ser usar programas extremos de pruebas o prestaciones.

#### 6.0.1 *Implementación Back-end*

Explicar Poetry, mypy, flake8 y pytest + esquema CI

#### 6.0.2 *Implementación Front-end*

## BIBLIOGRAFÍA

---

- [1] Belén Villa. «Estudio de Redes Sociales 2022 de IAB Spain». En: *Studio Aloha* (2022).
- [2] Brainstation. «ADDICTION: Study Finds Social Media to Be Like Smoking Cigarettes». En: *Digital* (2012).
- [3] Hilary Andersson. «Social media apps are 'deliberately' addictive to users». En: *BBC Panorama* (2018).
- [4] Talent. *Salario medio para Programador Web en España, 2022*. Inf. téc. Talent, 2022.
- [5] José Luis Orihuela-Colliva. «Internet: la hora de las redes sociales». En: (2008).
- [6] Blake Morgan. «50 Stats Showing The Power Of Personalization». En: *Forbes* (2020).
- [7] Montag C., Lachmann B., Herrlich M. y Zweig K. «Addictive Features of Social Media/Messenger Platforms and Freemium Games against the Background of Psychological and Economic Theories». En: *National Library of Medicine* (2019). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6679162/#B67-ijsph-16-02612>.
- [8] Manuel Moreno. «10 estadísticas sobre WhatsApp que quizás no sabías - TreceBits». En: *TreceBits* (2022).
- [9] Jared Spool. «Do users change their settings?». En: *Archive UIE* (2011).
- [10] Orbium. «El fenómeno FOMO: fear of missing out». En: *Orbium* (2020).
- [11] Laura Montells. «Qué es Edgerank o cómo funciona el algoritmo de Facebook». En: *Metricool* (2021).
- [12] Chema Carvajal. «El algoritmo de Facebook te muestra contenido que odias para que interacciones más». En: *ComputerHoy* (2021).
- [13] Christian Montag y col. «Facebook usage on smartphones and gray matter volume of the nucleus accumbens». En: *Behavioral Brain Research* 329 (2017), págs. 221-228. ISSN: 0166-4328. DOI: <https://doi.org/10.1016/j.bbr.2017.04.035>. URL: <https://www.sciencedirect.com/science/article/pii/S0166432817305090>.
- [14] Lauren E. Sherman, Patricia M Greenfield, Leanna M Hernandez y Mirella Dapretto. «Peer influence via instagram: Effects on brain and behavior in adolescence and young adulthood». En: *Child development* 89.1 (2018), págs. 37-47.

- [15] Javier Robledo Vico. «Así afectan las redes sociales a nuestro cerebro». En: *AS* (2017).
- [16] Bahena-Trujillo, Ricardo, Flores, Gonzalo, Arias-Montaño y José A. «Dopamina: síntesis, liberación y receptores en el Sistema Nervioso Central». En: *Revista Biomédica* 11.1 (2000), págs. 39-60.
- [17] Santiago Campillo. «Lo que pasa en nuestro cerebro cuando recibimos una notificación, según la ciencia». En: *Xataka* (2019).
- [18] Quintero-Cacique, Juan Camilo, Lesmes-Silva y Anggy Karina. «¿ Por qué el ser humano es adicto a las redes sociales?» En: *Revista CONVICCIONES* 8.15 (2021), págs. 6-9.
- [19] Krach S., Paulus F.M., Bodden M. y Kircher T. «The rewarding nature of social interactions». En: *Frontiers in behavioral neuroscience* (2010).
- [20] Jairo Rozo Castillo. «Pavlov y los reflejos condicionados: la verdadera historia sobre un descubrimiento». En: *Psicología Científica* (2015).
- [21] Raquel Rodríguez Cortés. «Adicción a las redes sociales: síntomas y consecuencias». En: *Psiquion* (2021).
- [22] Trevor Wheelwright. «2022 Cell Phone Usage Statistics: How Obsessed Are We?». En: *Reviews* (2022).
- [23] Sergio Parra. «La razón de que sea tan fácil volvernos adictos a contenidos digitales». En: *MuyInteresante* (2020).
- [24] Portillo-Reyes, J. A. V. Ávila Amaya y J. W. Capps. «Relación del Uso de Redes Sociales con la Autoestima y la Ansiedad en Estudiantes Universitarios». En: *Enseñanza e Investigación en Psicología* (2021).
- [25] Oscar Tinoco Gómez, Pedro Pablo Rosales López y Julio Salas Bacalla. «Criterios de selección de metodologías de desarrollo de software». En: *Industrial data* 13.2 (2010), págs. 70-74.
- [26] José H Canós, Patricio Letelier y M<sup>a</sup> Carmen Penadés. «Metodologías ágiles en el desarrollo de software». En: *Universidad Politécnica de Valencia, Valencia* (2003), págs. 1-8.
- [27] Esteban Gabriel Maida y Julián Pacienza. «Metodologías de desarrollo de software». En: (2015).
- [28] César Rodríguez y Rubén Dorado. «¿ Por qué implementar Scrum?». En: *Revista Ontare* 3.1 (2015), págs. 125-144.
- [29] Nader K Rad y Frank Turley. *Los Fundamentos de Agile Scrum*. Van Haren, 2019.
- [30] John Hogue y Burton DeWilde. *Librería Pytrends*. URL: <https://pypi.org/project/pytrends/>.
- [31] Joshua Roesslein. *Librería Tweepy*. URL: <https://docs.tweepy.org/en/latest/index.html>.

- [32] J Pelaez. «Arquitectura basada en Componentes. Extractado de La Guía de Arquitectura Versión 2.0a del grupo de Patterns and Practices de Microsoft.» En: ().
- [33] MongoDB. *Why Use MongoDB and When to Use It?* URL: <https://www.mongodb.com/why-use-mongodb>.
- [34] Red Hat. *What is a REST API?* URL: <https://www.redhat.com/es/topics/api/what-is-a-rest-api>.
- [35] Uvicorn. *Uvicorn Documentation.* URL: <https://www.uvicorn.org>.
- [36] FastAPI. *FastAPI Documentation.* URL: <https://fastapi.tiangolo.com>.
- [37] Heroku. *Heroku Documentation.* URL: <https://devcenter.heroku.com/articles/getting-started-with-python>.
- [38] Maja Nowak. «Vue vs React in 2022 - Comparison of Two Most Popular JS Frameworks.» En: (2022).
- [39] Red Hat. *El concepto de DevOps.* URL: <https://www.redhat.com/es/topics/devops>.
- [40] Angel Pérez. «Azure DevOps + Herramientas de Comunicación». En: (2020).
- [41] iLimit. «DevOps». En: (2020).



## COLOFÓN

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both L<sup>A</sup>T<sub>E</sub>X and LyX:

<https://bitbucket.org/amiede/classicthesis/>

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>

Thank you very much for your feedback and contribution.

*Final Version as of 29 de octubre de 2022 (classicthesis Version 0.1).*