UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

INDIVISA

Plataforma para el despliegue de topologías modulares y configurables de procesamiento masivo de datos web en tiempo real

Autor:
Marcos Fernández Pichel

Directores:
David Enrique Losada Carril
Juan Carlos Pichel Campos
Rodrigo Martínez Castaño

Grado en Ingeniería Informática

Julio 2019

Trabajo de Fin de Grado presentado en la Escola Técnica Superior de Enxeñaría de la Universidad de Santiago de Compostela para la obtención del Grado en Ingeniería Informática

D. David Enrique Losada Carril, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela, **D. Juan Carlos Pichel Campos** , Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela, y **D. Rodrigo Martínez Castaño**, investigador predoctoral en el CiTIUS,

INFORMAN:

Que la presente memoria, titulada *INDIVISA: Plataforma para el despliegue de topologías modulares y configurables de procesamiento masivo de datos web en tiempo real*, presentada por **D. Marcos Fernández Pichel** para superar los créditos correspondientes al Trabajo de Fin de Grado de la titulación del Grado en Ingeniería Informática, se realizó bajo nuestra dirección en el Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela.

Y para que así conste a los efectos oportunos, expiden el presente informe en Santiago de Compostela, a 27 de xuño de 2019:

El director,	El codirector,	El codirector,	El alumno,
LOSADA CARRIL DAVID ENRIQUE - 33297020N Firmado digitalmente por LOSADA CARRIL DAVID ENRIQUE - 33297020N Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-33297020N, givenName=DAVID ENRIQUE, sn=LOSADA CARRIL, cn=LOSADA CARRIL DAVID ENRIQUE - 33297020N Fecha: 2019.06.25 13:47:28 +02'00'	PICHEL CAMPOS JUAN CARLOS - 36115626Z Firmado digitalmente por PICHEL CAMPOS JUAN CARLOS - 36115626Z Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-36115626Z, givenName=JUAN CARLOS, sn=PICHEL CAMPOS, cn=PICHEL CAMPOS JUAN CARLOS - 36115626Z Fecha: 2019.06.25 15:37:08 +02'00'	MARTINEZ CASTAÑO RODRIGO - 35482168E Firmado digitalmente por MARTINEZ CASTAÑO RODRIGO - 35482168E Fecha: 2019.06.25 12:49:05 +02'00'	
D. David E. Losada Carril	D. Juan Carlos Pichel Campos	D. Rodrigo Martínez Castaño	D. Marcos Fernández Pichel

Agradecimientos

- A mi familia por ser mi apoyo incondicional todos y cada uno de los días.
- A mis tutores, David Losada y Juan Carlos Pichel, por la confianza depositada y por el trato, como uno más del equipo, desde el primer día.
- A Rodrigo Martínez, sin él la realización de este proyecto sería imposible, gracias por su colaboración y amistad.
- A Víctor Gallego, por resolver siempre las dudas de React a pesar de no ser su obligación.
- A los incondicionales, que siempre han estado ahí: Martín, Javier, Adrián, Rubén, Miguel, Lina, Natalia, Irene, Rocío, Pedro, Denise, Clara. Gracias por verme crecer como persona y llegar hasta este momento.
- A mi nueva familia de Santiago: Javi, Anxo, Jaime, María, Paula, Joaquín. Gracias por alegrarme todos los días y los que quedan por venir.
- A mis compañeros de clase, por ser la mejor promoción con la que he podido compartir estos 4 años.

Índice general

Índice de figuras

Índice de cuadros

Índice de fragmentos de código

Capítulo 1

Introducción

1.1. Big Data

Big Data es un término que describe el gran volumen de datos –estructurados y no estructurados– que somos capaces de recolectar y generar hoy en día. Existen múltiples aplicaciones que hacen uso de estos datos para distintos fines como, por ejemplo, publicidad dirigida o seguimiento de usuarios, entre otros.

Una de las principales fuentes de información de nuestra época son las redes sociales. Han supuesto una revolución en la manera que tenemos las personas de interactuar e implican que gran parte de nuestra vida sea pública. Por ello, el interés en poder monitorizar este tipo de plataformas es cada vez mayor por parte de las compañías y los cuerpos y fuerzas de seguridad del estado. Además, existe el valor añadido de poder hacerlo en tiempo real.

Se conoce como **Social Big Data** al análisis de la gran cantidad de información que se produce en las redes sociales y que puede ser utilizada por las empresas para la toma de decisiones. Algunos datos que respaldan la importancia de este fenómeno son, por ejemplo: Twitter genera más de 481000 tuits por minuto y Facebook más de 977000 publicaciones y 34000 "Me gusta" por minuto[1].

1.2. Motivación del proyecto

El objetivo de este TFG es el desarrollo de una plataforma capaz de procesar estos medios sociales web en tiempo real. Para ello, se utilizará como base la librería Python *Catenae*[2] desarrollada en el CiTIUS[1]. Ésta permite la construcción de topologías de procesamiento donde los nodos son módulos Python distribuidos, encapsulados en contenedores, y que se comunican mediante una

[1] https://www.juancmejia.com/redes-sociales/social-big-data-definicion-e-importancia-de-big-data-en-redes-sociales/

[2] https://github.com/catenae/catenae

cola de mensajes implementada en Apache Kafka[3].

En cuanto a la plataforma en sí, se proporcionarán una serie de módulos y fuentes preinstalados como, por ejemplo, un filtro de búsqueda a partir de una consulta proporcionada por el usuario, un analizador de tópicos o un generador de nubes de palabras y fuentes como Twitter o Reddit. Además, cualquier usuario podrá subir sus propios módulos y conectarlos a la topología de la forma que mejor le convenga. También se podrán añadir más fuentes que las preinstaladas de manera sencilla. El volumen de información generado en tiempo real es muy grande, por tanto, se necesitan tecnologías eficientes que permitan filtrar contenidos, analizarlos automáticamente y presentárselos a los usuarios finales de forma apropiada.

El objetivo es que se trate de un entorno de carácter generalista que proporcione el soporte para realizar un monitoreo simple y personalizado de los contenidos que se emiten en determinados medios web en tiempo real. Entre sus principales ventajas destacan la escalabilidad, la facilidad de despliegue mediante *scripts* de Python y la facilidad para definir una topología de consumo compleja de manera rápida y sencilla mediante un archivo .yml.

Como se ha comentado anteriormente, para este TFG se ha decidido utilizar el lenguaje Python, ya que es el tercero más utilizado según el último índice TIOBE[4], sólo por detrás de C y Java, y es uno de los dominadores en el ámbito del análisis de datos. Además, cabe destacar su facilidad de desarrollo.

Indivisa surge de la librería Catenae, la cual permite la construcción de topologías de procesamiento masivo de datos de manera muy simple. Estas topologías están compuestas por nodos llamados micromódulos. Cada uno de estos micromódulos consiste en un contenedor Docker[5] que ejecuta un programa Python en su interior. Además, las conexiones entre ellos representan sus comunicaciones y se implementan utilizando Apache Kafka. La gran ventaja que ofrece *Catenae* frente a otras soluciones es que es fácilmente escalable, puesto que Kafka nos ofrece la posibilidad de crear grupos de consumo[2]. Si dos micromódulos pertenecen al mismo *consumer group*, se repartirán el flujo de entrada a la mitad para cada uno, si fuesen tres, un tercio para cada uno y, así, sucesivamente. Además, la librería incluye la posibilidad de realizar llamadas **RPC** entre módulos, lo cual puede llegar a ser de gran utilidad.

1.3. Stream vs batch processing

Indivisa proporcionará dos modos de trabajo: en *streaming* o *batch* (procesamiento por lotes)[3]. El primero de ellos, se caracteriza por hacer operaciones simples en un tiempo muy reducido, de apenas unos segundos, y prácticamen-

[3]https://kafka.apache.org/documentation/
[4]https://www.tiobe.com/tiobe-index/
[5]https://docs.docker.com/

te indistinguible del tiempo real. Mientras que el segundo, se caracteriza por computar grandes cantidades de datos de manera compleja en un período de tiempo medido en minutos o más. Un ejemplo de *stream processing* podría ser acumular en tiempo real el número de menciones de una determinada persona en una red social, mientras que el modo *batch* consistiría en acumular un *corpus* incrementalmente sobre un determinado tema para, a posteriori, hacer una operación sobre él como, por ejemplo, un análisis de tópicos.

El caso de uso principal de los módulos de la plataforma es actuar como *stream processing systems*. Sin embargo, se proporcionará, a modo de ejemplo, un nodo que utilice el procesamiento por lotes y que sirva de demostrador de cómo *Indivisa* soporta ambos modos de funcionamiento. Se puede obtener una descripción más detallada de los módulos en el siguiente apartado.

1.4. Breve descripción técnica

El objetivo principal es la construcción de una plataforma basada en *Catenae* que tenga un valor por sí misma, siendo capaz de dar soporte al procesamiento masivo de datos web en tiempo real[4]. Para ello, se aportarán una serie de módulos preinstalados (que serán descritos a continuación) y será diseñada de manera que los usuarios puedan configurar sus propias topologías y modificarlas en tiempo real para poder llevar a cabo una monitorización más precisa y más ligada a sus intereses.

Las fuentes de datos se implementan como una serie de *crawlers* con capacidad de extracción masiva de datos. Estos se encuentran al principio de la topología y suponen los flujos principales para el resto de micromódulos. Serán configurables, es decir, se proporcionará soporte para poder añadir nuevas fuentes como, por ejemplo, Telegram y se podrán seleccionar los flujos de interés, descartando otros.

El primero de los módulos preinstalados será un filtro en tiempo real. Funcionará como un buscador por una consulta dada filtrando las publicaciones que contienen dichas palabras. Constará de dos modos: búsqueda exacta y búsqueda inexacta (*bag of words*). La primera de ellas, simplemente devolvería publicaciones que contengan la consulta dada en el mismo orden, mientras que la segunda realiza una limpieza de *stopwords* y admite variaciones en el orden, por ejemplo: si el usuario teclea *the White House*, el filtro eliminaría el artículo *the* y devolvería textos que contengan las palabras *white* y *house*, pero no estrictamente en ese orden o una a continuación de la otra.

El segundo módulo, será un generador de *tag clouds* dinámicos. Consumirá de un determinado flujo de textos, por ejemplo el flujo ya filtrado por el módulo anterior, y producirá una nube de palabras que sirva de resumen de lo más relevante. El valor añadido de esto es que se actualizará de forma dinámica con los textos que le vayan llegando en tiempo real. Esta nube de palabras es una forma primitiva de resumen automático que permite al usuario observar los términos

más recurrentes. Como posible trabajo futuro, podría extenderse a mecanismos más sofisticados de resumen autómatico de textos.

El tercer módulo preinstalado será un analizador de tópicos basado en Gensim[5] que tratará de proporcionar una idea de los asuntos o aspectos más relevantes relacionados con una determinada consulta de usuario, como si de *trending topics* se tratase. Constará de dos modos de funcionamiento: por número de textos o en una franja temporal. En el primer caso, el módulo definirá una estructura de datos circular que se irá actualizando en tiempo real con los textos más recientes e irá aplicando el topic analysis. En el segundo caso daría respuesta al caso de uso de monitorizar qué se habla sobre un tema durante un determinado tiempo, por ejemplo, quiero saber qué se ha dicho sobre Trump en la última hora. Para ello, la estructura utilizada deja de ser circular y se seleccionan sólo los mensajes que cumplan la condición temporal expresada y es con ellos con los que se realiza el análisis.

El cuarto módulo aporta una visión diferente a la plataforma, ya que se distancia de la idea del tiempo real para centrarse en el procesamiento por lotes. La idea consiste en recopilar grandes cantidades de textos o publicaciones para luego poder coger ese lote y aplicarle técnicas de minería como las que se han descrito anteriormente. Puede parecer que esto se aleja de la filosofía de Indivisa, pero en realidad surge de manera natural: Kafka conserva en sus colas todos los mensajes por un tiempo configurable, dando pie a la posibilidad de realizar trabajos más pesados con esos lotes almacenados.

Cabe aclarar que para conseguir la persistencia de los datos se utilizará MongoDB[6] y que se proporcionarán las principales directrices para conseguir un despliegue en un gestor de contenedores, como por ejemplo Kubernetes[7], de la plataforma.

Por último, se definirá una API RESTful que permita el acceso a los servicios de la plataforma. Se implementará un último módulo, el cual consistirá en una interfaz web que consuma estos servicios y muestre de manera sencilla cómo sería la interacción con los usuarios.

1.5. Objetivos

El objetivo principal del proyecto consiste en el desarrollo de una plataforma genérica que permita el monitoreo de medios web en tiempo real mediante la utilización de la librería Python *Catenae*. De modo más concreto, los objetivos del TFG son los que siguen:

1. Llevar a cabo un diseño lo más genérico posible y que permita realizar operaciones a los usuarios como: reconfigurar la topología en tiempo real,

[6] https://docs.mongodb.com/
[7] https://kubernetes.io/docs/home/

poder añadir sus propios módulos o añadir/quitar fuentes de datos.

2. Integración con *crawlers* para Twitter/Reddit.
3. Desarrollo de un módulo de filtrado por consultas proporcionadas por el usuario.
4. Desarrollo de un módulo que permita la generación de *Tag Clouds* dinámicos, es decir, nubes de palabras que se actualicen en tiempo real, dando una visión resumida sobre un determinado tema.
5. Desarrollo de un módulo de análisis de tópicos que muestre los términos más relevantes para una determinada consulta.
6. Desarrollo de un módulo para procesamiento por lotes genérico.
7. Desarrollo de un módulo de generación de estadísticas.
8. Desarrollo de un módulo que consista en una interfaz web que sirva como demostrador de los módulos preinstalados (los mencionados en los puntos anteriores) y que permita a los usuarios personalizar y crear sus propias topologías.

1.6. Estructura de la memoria

La presente memoria se estructura de la siguiente forma:

- **Capítulo 1. Introducción**: el presente capítulo. Pretende dar una visión general de la plataforma y sus objetivos.
- **Capítulo 2. Gestión del proyecto**: cómo se ha realizado el proyecto desde un punto de vista de metodología, planificación, riesgos y costes.
- **Capítulo 3. Especificación de requisitos**: descripción del sistema en requisitos de información, funcionales y no funcionales.
- **Capítulo 4. Análisis de tecnologías y herramientas**: descripción de las tecnologías empleadas en el desarrollo y en la documentación del proyecto.
- **Capítulo 5. Diseño e implementación**: descripción detallada de la arquitectura de la aplicación, así como, de su implementación.
- **Capítulo 6. Pruebas y validación**: pruebas llevadas a cabo para comprobar la correcta implementación de las funcionalidades descritas en el capítulo 3.
- **Capítulo 7. Conclusiones y trabajo futuro**: posibles ampliaciones de plataforma y trabajo futuro.

Capítulo 2

Gestión del proyecto

2.1. Alcance del proyecto

En esta sección de la memoria se intentará proporcionar una idea de los límites que conforman el producto a desarrollar para, así, aclarar las funcionalidades que se implementarán y las que no. Se trata de un proceso fundamental para el éxito del proyecto.

2.1.1. Descripción del alcance del producto

El producto a desarrollar consiste en una plataforma software distribuida para el procesado de datos web en tiempo real. Los diferentes nodos que la componen consisten en programas Python *dockerizados*. Todos ellos se comunican entre sí utilizando colas de Apache Kafka generando, así, una topología de procesamiento.

Al principio de la misma, se encontrarán una serie de *crawlers* encargados de la extracción masiva de datos web. Estos se basan en procedimientos de *web scraping*, los cuales procesan documentos web y extraen cierta información de los mismos. Se proporcionarán una serie de fuentes preinstaladas, como Twitter y Reddit, pero a mayores se ofrecerá la posibilidad de que los usuarios de *Indivisa* puedan subir una imagen Docker de una fuente o *crawler* programado por ellos y que puedan conectarlo a una topología existente de manera sencilla.

Asimismo, la plataforma contará con una serie de módulos ya instalados que reflejan operaciones de gran interés relacionadas con la minería de textos y que pueden servir a los usuarios sin necesidad de que tengan que desarrollarlos. Entre ellos destacan: un filtro en tiempo real por una consulta dada, un generador dinámico de nubes de palabras, un analizador de tópicos, un módulo de procesamiento por lotes y un módulo complementario de estadísticas acerca de la plataforma (textos y usuarios totales procesados y textos y usuarios por segundo).

Al igual que en el caso de las fuentes de datos, será posible programar diferentes módulos y subirlos contenidos en una imagen Docker a *Indivisa*. Con esto, lo que se intenta es conseguir un sistema lo más genérico posible que permita la construcción de topologías personalizables de procesamiento masivo de datos.

Por otra parte, también será posible el despliegue de topologías enteras descritas mediante un archivo en formato *yaml*, utilizando las funcionalidades de *docker-compose*. Estas composiciones permiten lanzar múltiples aplicaciones Docker y el principal objetivo es conseguir que *Indivisa* sea compatible con archivos utilizados tanto por *Kubernetes* como por *Catenae*.

Por último, se podrá acceder a los datos, añadir y quitar módulos o configurar la disposición de los mismos mediante una API de tipo RESTful. Para que sea más visible y cómodo, se programará una interfaz web que permita una fácil interacción.

2.1.2. Entregables del proyecto

El proyecto estará conformado por los siguientes entregables:

- Software funcional de Indivisa
- Código fuente de la plataforma
- La presente memoria junto con los manuales pertinentes

2.1.3. Criterios de aceptación

A continuación, se exponen los criterios mínimos para considerar el proyecto como aceptado:

- Se permite el filtrado de *tweets* y publicaciones de *reddit* mediante una consulta dada por un usuario en tiempo real.
- Se permite la generación de *tag clouds* dinámicos y el análisis de temas de dichas fuentes.
- Se aporta un ejemplo de uso de procesamiento *batch* para *Indivisa*.
- El usuario puede construir sus propias topologías de consumo creándolas módulo a módulo o mediante la aportación de un fichero descriptivo en formato .yaml.
- La API permite recuperar los valores emitidos por los diferentes módulos.
- La API es capaz de recibir peticiones para levantar un módulo concreto o desplegar una topología entera.

2.1.4. Exclusiones del proyecto

Se excluirá del proyecto todo aquello que impida su correcta finalización, en tiempo y en forma, de acuerdo con lo establecido en esta memoria. Cabe destacar, que no se llevará a cabo el despliegue de la plataforma en un clúster. Sin embargo, sí que será diseñada y se proporcionarán las principales directrices para que en un futuro sea viable y sencillo hacerlo.

Por otra parte, señalar que no se implementarán restricciones de carga para los módulos. Véase el caso de un filtro que ya está ejecutando muchas consultas, lo lógico sería que cuando se llegue a un número dado de las mismas, se levante otro módulo equivalente para repartirse la carga. Esto no se llevará a producción, pero el sistema sí que estará diseñado para soportarlo. Relacionado con esto, en el código se introducen las principales directrices para poder indicarle a cada módulo el porcentaje de cpu y memoria que podría utilizar, sin embargo esto tampoco será implementado por restricciones temporales.

2.1.5. Restricciones del proyecto

La plataforma podrá ser ejecutada en máquinas x86-64 con distribuciones GNU-Linux modernas. Se utilizarán librerías open source para su desarrollo y cabe señalar que el principal gasto del proyecto radicará en los RRHH que participen en él.

El proyecto debe suponer 412,5 horas de trabajo y no puede ser entregado más tarde del 30/06/2019.

2.1.6. Supuestos del proyecto

Indivisa funcionará siempre bajo los dos siguientes supuestos: el correcto funcionamiento de los métodos importados de la librería *Catenae* y de las colas de mensajes de Apache Kafka, las cuales implementarán las comunicaciones entre los módulos de la plataforma.

2.1.7. Diagrama del sistema

En el diagrama se puede contemplar un proceso central "Procesar datos con *Indivisa*" que se corresponde con la función que el software o el proyecto debe abordar, dicho de otro modo, limita el alcance del proyecto. Como entidades externas con las que se comunica el proceso se encuentran "Web" y "Usuario interfaz". A continuación, se abordará cada una de ellas por separado.

"Web" hace referencia a cualquier medio que pueda ser monitoreado a través de un procedimiento de web scraping, que es el utilizado por los *crawlers* de *Indivisa*. En la plataforma, las fuentes que se estudian por defecto son las redes

sociales Twitter y Reddit, sin embargo, sería fácilmente ampliable a servicios de mensajería como Telegram o a periódicos digitales.

"Usuario interfaz" hace referencia a la persona que interactúa con el sistema a través de la interfaz web: configurando sus propias topologías, observando el resultado de los módulos, etc.

Figura 2.1: Diagrama de sistema.

2.2. Metodología de desarrollo

La correcta elección de la metodología de desarrollo es una decisión fundamental para alcanzar los objetivos de un proyecto software. Se deben tener en cuenta las particularidades del proyecto a la hora de escoger una de ellas y de adaptarla al caso específico.

En el caso que nos incumbe, existen una serie de características que cabe mencionar:

- Las tecnologías principales sobre las que se sustenta el proyecto (*Docker* y *Apache Kafka*) son totalmente nuevas para el equipo de desarrollo (en este caso, el alumno del TFG), lo cual puede provocar cierta incertidumbre y retrasos. Así como, las técnicas *Big Data*, las cuales tampoco le son familiares antes del comienzo del mismo.

- Como se ha mencionado en el punto anterior, el equipo de desarrollo estará formado por un solo miembro: el alumno del TFG que contará con el apoyo de sus tutores.

- Los requisitos pueden experimentar procesos de refinamiento o incluso aparecer nuevos al estar ligados a un ámbito tan cambiante como el del *Big Data*, en el cual pueden aparecer novedades que se deban reflejar en el proyecto.

Por todo ello, se ha decidido que lo más adecuado para este proyecto es una metodología ágil, más concretamente se utilizará **Kanban**, el cual se explicará con mayor detalle en el apartado siguiente.

2.2.1. Kanban

Para el desarrollo de este proyecto se ha decidido utilizar la metodología Kanban[6]. Se clasifica dentro de las llamadas metodologías ágiles y está basada en el desarrollo incremental con un flujo de trabajo continuo, dividiendo el trabajo en distintas partes. Destaca principalmente por ser una técnica que permite la gestión de tareas de una manera muy visual mediante el llamado **tablero Kanban**. También permite cambiar rápidamente las prioridades de las tareas y tener una visión muy clara del flujo de trabajo.

El sistema Kanban se organiza como un gran tablón dividido en columnas. El número de columnas depende también de la complejidad del proyecto y de cómo se organice el equipo de desarrollo, en este caso se ha decidido optar por tres: *Open*, *In Progress* y *Done*. La principal diferencia con Scrum[7] es que en este caso el tablero es continuo. A medida que se avanza, las nuevas tareas se acumulan en la sección inicial y, en reuniones periódicas, con el cliente se priorizan y se colocan en la sección que se considere oportuna. Kanban no limita el tiempo para realizar una determinada cantidad de trabajo, como sí hace Scrum con los *sprints*. Es un progreso continuo en el desarrollo de las tareas. Otro concepto importante es el del *Limit Work In Progress* (WIP), es decir, el número de tareas que se puede realizar en cada fase debe ser conocido. Esto se suele basar en las capacidades de desarrollo del equipo y permite garantizar la calidad. Este concepto no existe en Scrum, ya que se hace de manera indirecta mediante el *sprint planning*. Aunque cabe señalar que, en ocasiones, Scrum y Kanban se utilizan juntos en el llamado Scrumban.

Kanban permite priorizar tareas pendientes, permitiendo responder ante ciertos imprevistos. Además, se centra en la calidad. Se ha elegido esta opción por delante de Scrum por su sencillez y mayor flexibilidad al tratarse de un equipo de desarrollo de una única persona. Por tanto, cumple con los requisitos expuestos en el apartado anterior. Por último, cabe mencionar que para realizar el seguimiento del proyecto se utilizará la herramienta YouTrack[1] de JetBrains.

[1] https://www.jetbrains.com/youtrack/

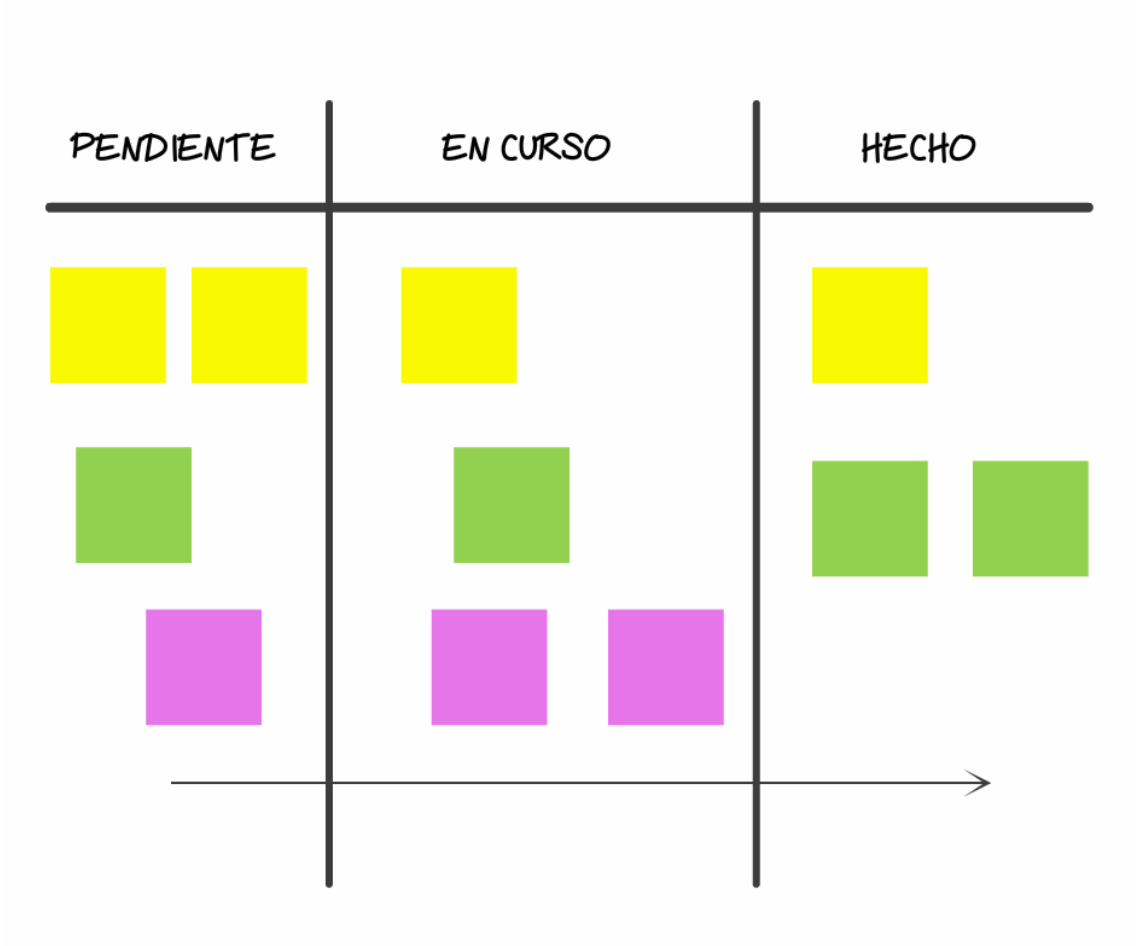

Figura 2.2: Tablero Kanban
Cortesía de `https://www.20milproductos.com/blog/gestion-de-proyectos-con-post-it/`

2.3. Planificación del proyecto

Llegado este punto, cabe recordar que la estimación temporal para un trabajo de fin de grado de la titulación de Ingeniería Informática supondrá 412,5 horas de trabajo. Además, la dedicación semanal prevista, en este caso, será de 25 horas/semana durante 16 semanas y media.

2.3.1. Estructura de descomposición del trabajo (EDT)

La EDT o WBS es, en gestión de proyectos, una descomposición jerárquica orientada al entregable del trabajo a ser ejecutado por el equipo de proyecto para cumplir con los objetivos de este[8]. Todo lo que no se encuentre recogido en la EDT, no se considerará parte del alcance del proyecto. A continuación, se presenta el diagrama de descomposición del trabajo de *Indivisa*:

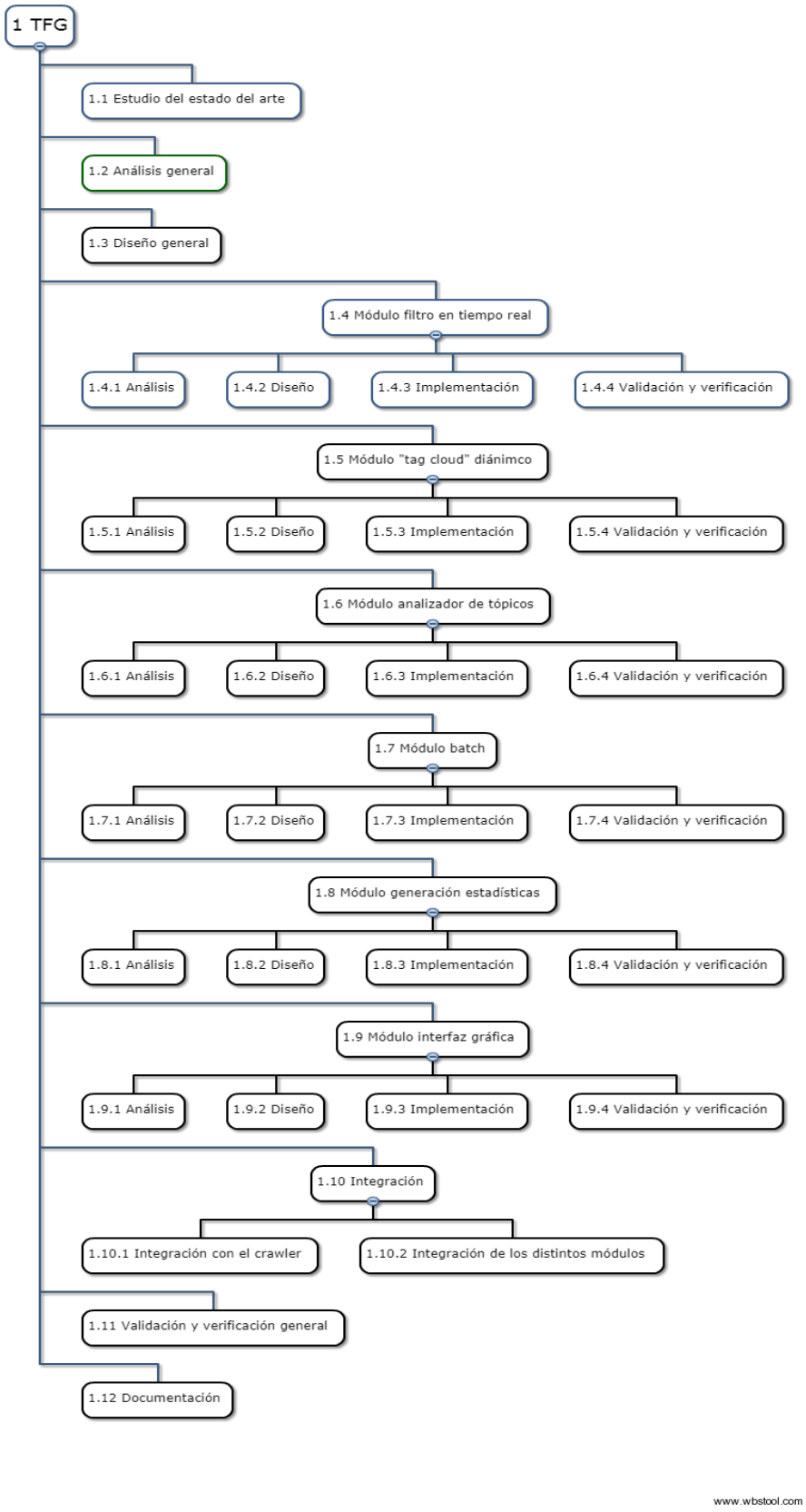

Figura 2.3: Diagrama de Descomposición de Trabajo del proyecto

2.3.2. Diccionario de la EDT

- **Estudio del estado del arte.** Estudio y selección de las diferentes alternativas existentes en la bibliografía de la minería de textos. **Estimación temporal:** 20 horas.

- **Análisis general.** Recopilar, examinar y formular los requisitos necesarios para la plataforma y para que cumpla con su propósito de ser lo más genérica posible. **Estimación temporal:** 20 horas.

- **Diseño general.** Definición precisa de la arquitectura general de la plataforma. Se debe tener en mente siempre una idea generalista de la misma que incluya los módulos preinstalados y que permita cargar otros o reconfigurar los existentes. **Estimación temporal:** 20 horas.

- **Módulo filtro en tiempo real.** Se corresponde con el tercer objetivo del proyecto.

  - **Análisis.** Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables. **Estimación temporal:** 8 horas.
  - **Diseño.** Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes. **Estimación temporal:** 4 horas.
  - **Implementación.** Codificación del módulo. **Estimación temporal:** 12 horas.
  - **Validación y verificación.** Tareas que permiten asegurar que este módulo esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 4 horas

- **Módulo tag cloud dinámico.** Se corresponde con el cuarto objetivo del proyecto.

  - **Análisis.** Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables. **Estimación temporal:** 8 horas.
  - **Diseño.** Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes. **Estimación temporal:** 4 horas.
  - **Implementación.** Codificación del módulo. **Estimación temporal:** 12 horas.
  - **Validación y verificación.** Tareas que permiten asegurar que este módulo esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 4 horas

- **Módulo analizador de tópicos.** Se corresponde con el quinto objetivo del proyecto.

  - **Análisis.** Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables. **Estimación temporal:** 8 horas.
  - **Diseño.** Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes. **Estimación temporal:** 8 horas.
  - **Implementación.** Codificación del módulo. **Estimación temporal:** 16 horas.
  - **Validación y verificación.** Tareas que permiten asegurar que este módulo esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 8 horas

- **Módulo batch.** Se corresponde con el sexto objetivo del proyecto.

  - **Análisis.** Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables. **Estimación temporal:** 8 horas.
  - **Diseño.** Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes. **Estimación temporal:** 4 horas.
  - **Implementación.** Codificación del módulo. **Estimación temporal:** 4 horas.
  - **Validación y verificación.** Tareas que permiten asegurar que este módulo esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 4 horas

- **Módulo de generación de estadísticas.** Se corresponde con el séptimo objetivo del proyecto.

  - **Análisis.** Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables. **Estimación temporal:** 8 horas.
  - **Diseño.** Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes. **Estimación temporal:** 4 horas.
  - **Implementación.** Codificación del módulo. **Estimación temporal:** 4 horas.
  - **Validación y verificación.** Tareas que permiten asegurar que este módulo esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 4 horas

- **Módulo interfaz gráfica.** Se corresponde con el octavo objetivo del proyecto.
  - **Análisis.** Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables. **Estimación temporal:** 12 horas.
  - **Diseño.** Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes. **Estimación temporal:** 8 horas.
  - **Implementación.** Codificación del módulo. **Estimación temporal:** 60 horas.
  - **Validación y verificación.** Tareas que permiten asegurar que este módulo esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 20 horas
- **Integración.** Tiene dos partes:
  - **Integración con el *crawler*.** Conexión de la plataforma con los crawlers que actuarán como fuentes de datos para la misma. **Estimación temporal:** 12 horas.
  - **Integración de los distintos módulos.** Conexión de los módulos preinstalados como topología de ejemplo. **Estimación temporal:** 12 horas.
- **Validación y verificación general.** Tareas que permiten asegurar que la plataforma entera esté acorde con su especificación y satisfaga sus objetivos iniciales. **Estimación temporal:** 14 horas.
- **Documentación.** Todas las tareas relacionadas con la documentación del proyecto, la documentación de los distintos módulos y la memoria global. Así como, la preparación de la defensa del trabajo y generación de la correspondiente presentación y vídeos demostrativos. **Estimación temporal:** 78,5 horas.

2.3.3. Cronograma del proyecto: Diagrama de Gantt

El diagrama de Gantt tiene por objetivo representar el tiempo previsto para las tareas descritas en la estructura de descomposición del trabajo (EDT). Además, establece las dependencias existentes entre cada una de ellas y les asigna una serie de recursos.

Cabe aclarar que la tarea de documentación se realizará a lo largo de todo el proyecto, por tanto, contará con una dedicación constante del 20 % por parte del alumno. El resto de actividades supondrán una asignación del 80 %.

En cuanto a las reuniones presenciales con los tutores, se llevarán a cabo al inicio de las tareas de cada módulo, en la fase de análisis. Por ello, se les asigna un 10 % a cada uno de ellos en dichas tareas y un 15 % a Rodrigo Martínez Castaño al tener un rol de responsable técnico por su desarrollo de la librería *Catenae* en la cual se basa *Indivisa*.

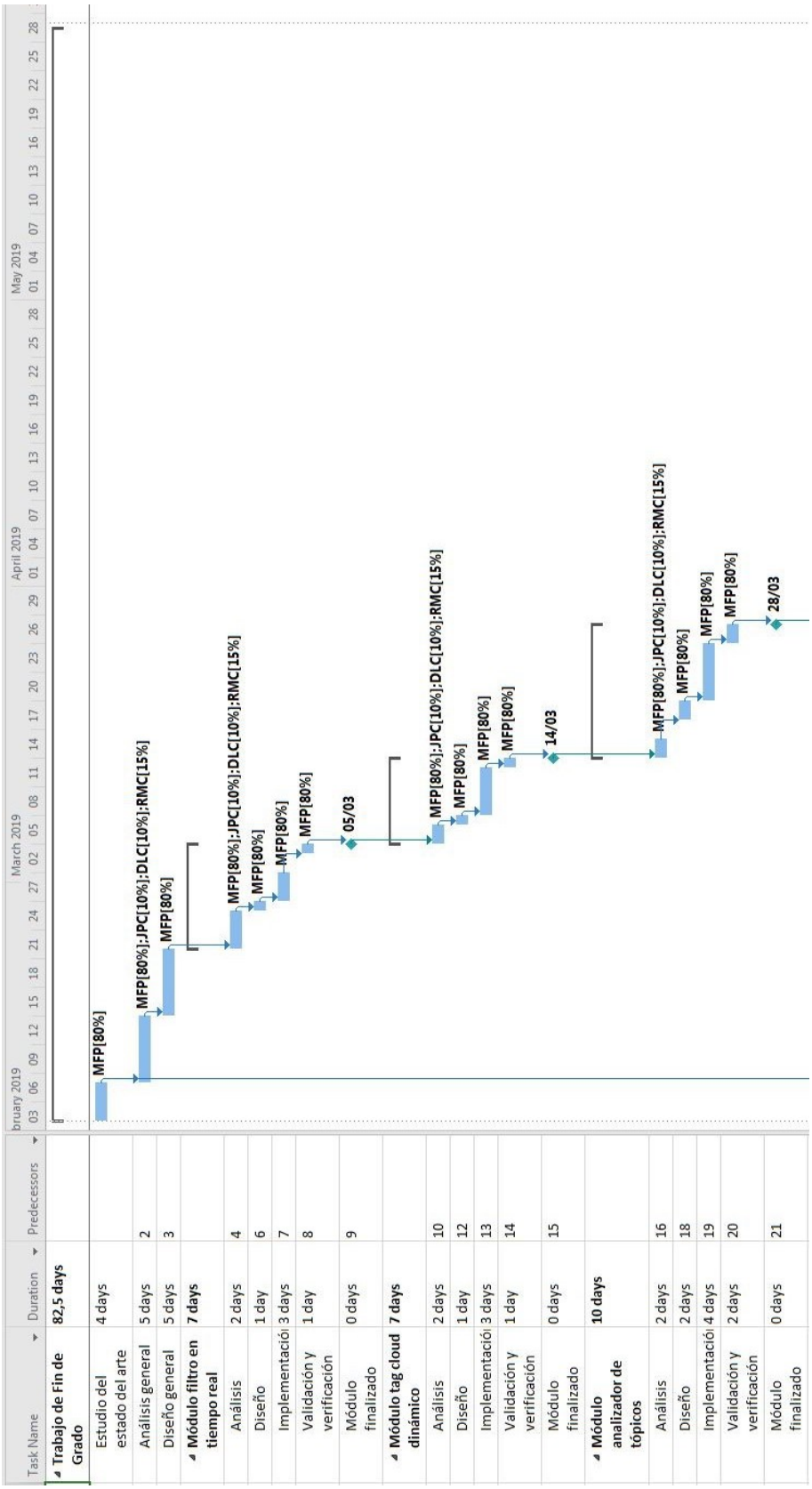

Figura 2.4: Diagrama de Gantt (1/3)

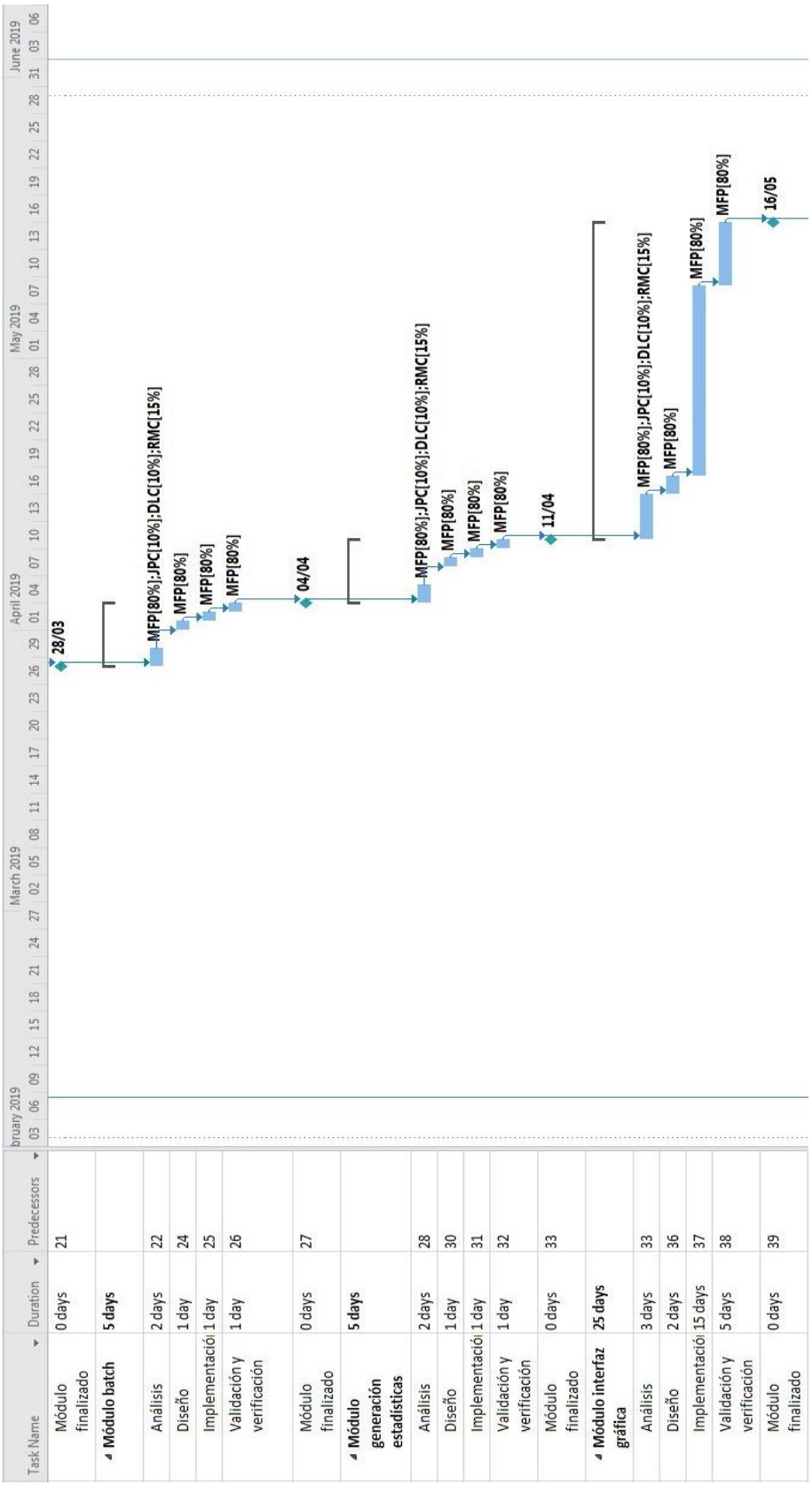

Figura 2.5: Diagrama de Gantt (2/3)

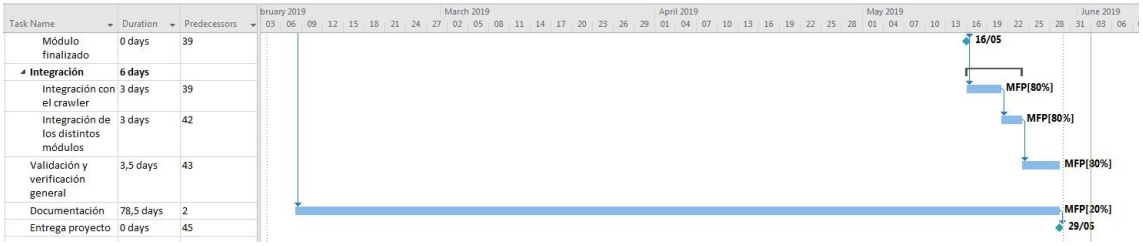

Figura 2.6: Diagrama de Gantt (3/3)

2.4. Gestión de la configuración

La gestión de la configuración trabaja con elementos de la configuración del software o ECS, los cuales se pueden definir como la información creada como parte del proceso de ingeniería de software y que pueden controlarse de manera separada. En el caso concreto de este proyecto, se puede distinguir entre el código de la plataforma y la documentación de la misma.

En cuanto al control de cambios, cabe aclarar que, en este caso, al tratarse de tareas desarrolladas por un sólo alumno que será el único que modificará los elementos de la configuración, la gestión de la configuración se limitará al control de versiones.

2.4.1. Gestión del código

Para la gestión del código, se ha decidido crear un repositorio en la plataforma Gitlab[2], la cual utiliza Git de manera remota. El alumno trabaja sobre la rama "master" y cada cambio se acompaña de un mensaje explicativo. La utilización de una copia remota previene el riesgo de pérdida del código fuente.

2.4.2. Gestión de la documentación

Para la gestión de la documentación del proyecto se utilizará la plataforma Google Drive[3], la cual nos permite tener una copia remota a la que poder acceder simultáneamente tanto el alumno como los tutores. La estructura de carpetas elegida es la siguiente:

[2]https://about.gitlab.com/
[3]https://drive.google.com

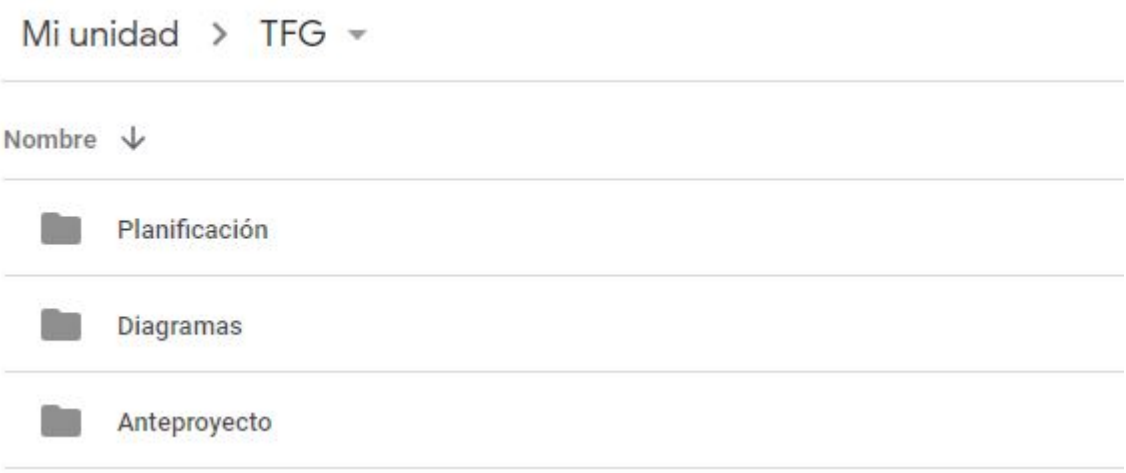

Figura 2.7: Estructura de carpetas de Google Drive

La utilización de esta copia remota previene contra la pérdida de la documentación del proyecto. En cuanto a la edición de la memoria, se utilizará la plataforma online *Overleaf v2*[4]. Dispone de control de versiones y almacenamiento en la nube, por lo que también previene contra posibles pérdidas. Por último, cabe señalar que para la realización de los diagramas se utilizó la herramienta Draw.io[5].

2.5. Gestión de riesgos

Un riesgo es un evento o condición incierta que, en caso de ocurrir, tiene un efecto positivo o negativo sobre los objetivos de un proyecto. Habitualmente se gestionan los riesgos con efecto negativo, es decir, aquellos que suponen una amenaza para el éxito del proyecto.

La función de la gestión de riesgos del software es identificar, estudiar y eliminar las fuentes de riesgo antes de que empiecen a amenazar la finalización satisfactoria de un proyecto o tener medidas para superarlo con ese mismo objetivo.

En esta sección, se identificarán una serie de riesgos concretos que pueden afectar a los activos de este proyecto. A continuación, se valorará su probabilidad e impacto y se seleccionarán aquellos más relevantes para llevar a cabo acciones sobre ellos que los prevengan o los minimicen.

[4]https://v2.overleaf.com/
[5] https://www.draw.io/

2.5.1. Identificación de riesgos

- **RSG-01:** Experimentar retrasos debido a la inexperiencia con las tecnologías.
- **RSG-02:** Fallos inesperados en *Catenae*
- **RSG-03:** Experimentar retrasos al ponerse enfermo el alumno.
- **RSG-04:** Aparición de nuevos requisitos.
- **RSG-05:** Falta de disponibilidad del responsable técnico.
- **RSG-06:** Requisitos mal definidos o inexactos.
- **RSG-07:** Estimación temporal demasiado optimista.
- **RSG-08:** Equipo de desarrollo estropeado.
- **RSG-09:** La plataforma Google Drive no se encuentra disponible.
- **RSG-10:** La plataforma Gitlab no se encuentra disponible.
- **RSG-11:** Problemas en la integración de los distintos módulos.
- **RSG-12:** Tareas de validación y verificación mal definidas.
- **RSG-13:** Errores inesperados en el código.
- **RSG-14:** Pérdida de parte/totalidad de la documentación/código.
- **RSG-15:** Disponibilidad de los datos insuficiente.

2.5.2. Análisis de riesgos

En este apartado, se realizará el análisis de riesgos en función de su probabilidad de ocurrencia y el impacto que provocarían sobre los objetivos del proyecto. En primer lugar, se va a proceder a definir las escalas que se utilizarán para cada una de estas magnitudes para, en segundo lugar, calificar cada uno de los riesgos individualmente.

Alta	Probabilidad estimada mayor que el 80 %
Media	Probabilidad estimada entre el 30 % y el 80 %
Baja	Probabilidad estimada menor que el 30 %

Cuadro 2.1: Escala de probabilidad de ocurrencia de riesgos.

Catastrófico	Podría provocar alteraciones graves en la planificación o el alcance del proyecto o, en casos extremos, la cancelación del proyecto
Serio	Podría provocar alteraciones moderadas en la planificación o el alcance del proyecto
Tolerable	Podría provocar alteraciones leves en la planificación o el alcance del proyecto

Cuadro 2.2: Escala de impacto de los riesgos.

RSG-01	Experimentar retrasos debido a la inexperiencia con las tecnologías
Descripción	Debido a la utilización de tecnologías *Big Data* novedosas para el alumno, como *Apache Kafka* y *Docker*, se podrían experimentar retrasos en la planificación del proyecto que impliquen un reordenamiento de las prioridades.
Probabilidad	Alta
Impacto	Serio

Cuadro 2.3: RSG-01.

RSG-02	Fallos inesperados en Catenae
Descripción	*Catenae* es una librería que todavía se encuentra en desarrollo, por tanto, se pueden producir cambios que alteren la manera de trabajar con *Indivisa* y, en el peor de los casos, que lleven a fallos.
Probabilidad	Alta
Impacto	Serio

Cuadro 2.4: RSG-02.

RSG-03	Experimentar retrasos al ponerse enfermo el alumno
Descripción	El alumno podría caer enfermo y, al ser el recurso principal asociado al proyecto, se podrían producir retrasos en la planificación que deben ser tenidos en cuenta.
Probabilidad	Media
Impacto	Serio

Cuadro 2.5: RSG-03.

RSG-04	Aparición de nuevos requisitos
Descripción	El *Big Data* es un ámbito muy cambiante en el que aparecen innovaciones constantemente. Esto puede provocar que sea interesante contemplar nuevos requisitos para la plataforma que un primer momento no se tuvieron en cuenta.
Probabilidad	Media
Impacto	Serio

Cuadro 2.6: RSG-04.

RSG-05	Falta de disponibilidad del responsable técnico
Descripción	Rodrigo Martínez, como creador de Catenae, es considerado una especie de responsable técnico del proyecto. Sin embargo, cabe la posibilidad de que no se encuentre disponible cuando el alumno tenga un problema o se produzca un fallo con la librería.
Probabilidad	Alta
Impacto	Tolerable

Cuadro 2.7: RSG-05.

RSG-06	Requisitos mal definidos o inexactos
Descripción	Si no se dedica el tiempo suficiente a la definición de lo que se desea realizar y a poner los objetivos en común con los directores de proyecto, cabe la posibilidad de que los requisitos se definan de manera incompleta o inexacta.
Probabilidad	Media
Impacto	Serio

Cuadro 2.8: RSG-06.

RSG-07	Estimación temporal demasiado optimista
Descripción	La estimación temporal planteada puede no ser realista y el alumno verse sobrepasado por las expectativas y tener que reordenar sus prioridades.
Probabilidad	Alta
Impacto	Serio

Cuadro 2.9: RSG-07.

RSG-08	Equipo de desarrollo estropeado
Descripción	El portátil del alumno se estropea, provocando retrasos inesperados.
Probabilidad	Media
Impacto	Serio

Cuadro 2.10: RSG-08.

RSG-09	La plataforma Google Drive no se encuentra disponible
Descripción	La plataforma seleccionada para la gestión de la configuración de la documentación está caída.
Probabilidad	Baja
Impacto	Serio

Cuadro 2.11: RSG-09.

RSG-10	La plataforma Gitlab no se encuentra disponible
Descripción	La plataforma seleccionada para la gestión de la configuración del código está caída
Probabilidad	Baja
Impacto	Serio

Cuadro 2.12: RSG-10.

RSG-11	Problemas en la integración de los distintos módulos
Descripción	Surgen incompatibilidades a la hora de conectar los distintos módulos preinstalados de *Indivisa*, los cuales han sido desarrollados de manera independiente
Probabilidad	Alta
Impacto	Serio

Cuadro 2.13: RSG-11.

RSG-12	Tareas de verificación y validación mal definidas
Descripción	Si las pruebas no se definen de manera precisa, se pueden producir incorrecciones a la hora de validar un requisito como correctamente implementado.
Probabilidad	Alta
Impacto	Serio

Cuadro 2.14: RSG-12.

RSG-13	Errores inesperados en el código
Descripción	Aparición de errores de programación inesperados en alguno de los módulos, los cuales pueden provocar retrasos en la planificación.
Probabilidad	Alta
Impacto	Serio

Cuadro 2.15: RSG-13.

RSG-14	Pérdida de parte/totalidad de la documentación/código
Descripción	Pérdida total o parcial de la documentación o el código del proyecto.
Probabilidad	Baja
Impacto	Catastrófico

Cuadro 2.16: RSG-14.

RSG-15	Disponibilidad de los datos insuficiente
Descripción	La caída temporal de alguna de las APIs de las fuentes (Twitter, Reddit, etc) provocaría la inoperancia de la plataforma por falta de datos. Si no se remedia, esto sería catastrófico para el desarrollador, por ejemplo, a la hora de realizar pruebas.
Probabilidad	Baja
Impacto	Catastrófico

Cuadro 2.17: RSG-15.

2.5.3. Respuesta a riesgos

Una vez realizado el análisis de los riesgos del proyecto, se deben seleccionar aquellos a los que se quiere dar respuesta, puesto que es imposible hacer seguimiento de todos ellos. Para ayudar en la toma de decisiones, se utilizará una matriz de exposición de riesgos en la que se confrontan su probabilidad y su

		Impacto		
	(RSG)	Tolerable	Serio	Catastrófico
Probabilidad	Baja		RSG-09, RSG-10	RSG-14, RSG-15
	Media		RSG-03, RSG-04, RSG-06, RSG-08	
	Alta	RSG-05	RSG-01, RSG-02, RSG-07, RSG-11, RSG-12, RSG-13	

Figura 2.8: Matriz de exposición a riesgos

impacto.

Tomando como referencia la matriz anterior, se gestionarán con acciones de prevención y contingencia los seis riesgos identificados con probabilidad de ocurrencia alta e impacto serio:

RSG-01	Experimentar retrasos debido a la inexperiencia con las tecnologías
Plan de prevención	Elaborar una planificación previsora poniendo un margen por si estos retrasos surgiesen.
Indicadores	Experimentar un retraso igual o superior a dos días laborables en una tarea de implementación de un módulo.
Plan de contingencia	Analizar las tareas restantes, realizando una replanificación, e incluso reajustando los plazos de entrega si fuese necesario.

Cuadro 2.18: Respuesta a RSG-01.

RSG-02	Fallos inesperados en Catenae
Plan de prevención	Elaborar una planificación previsora poniendo un margen por si estos errores surgiesen.
Indicadores	Aparecen errores de una ejecución a otra, sin haber modificado una sola línea de código correspondiente a la plataforma.
Plan de contingencia	Analizar las tareas restantes, realizando una replanificación, e incluso reajustando los plazos de entrega si fuese necesario.

Cuadro 2.19: Respuesta a RSG-02.

RSG-07	Estimación temporal demasiado optimista
Plan de prevención	Elaborar una planificación en la que se incluya un colchón temporal por si se producen retrasos en alguna de las tareas planificadas.
Indicadores	Experimentar un retraso igual o superior a dos días laborables en cualquier tarea del proyecto.
Plan de contingencia	Analizar las tareas restantes, realizando una replanificación, e incluso reajustando los plazos de entrega si el colchón no fuese lo suficientemente amplio como para paliar los retrasos.

Cuadro 2.20: Respuesta a RSG-07.

RSG-11	Problemas en la integración de los distintos módulos
Plan de prevención	Tratar de seguir un esquema común, definido en la fase de diseño general, para la implementación de los distintos módulos que componen la plataforma .
Indicadores	Experimentar un retraso igual o superior a un día laborable en las tareas de integración de los distintos módulos de la plataforma.
Plan de contingencia	Analizar las tareas restantes, realizando una replanificación, e incluso reajustando los plazos de entrega si fuese necesario. En el peor de los casos, habría que recodificar alguno de los módulos si las incompatibilidades son muy fuertes.

Cuadro 2.21: Respuesta a RSG-11.

RSG-12	Tareas de verificación y validación mal definidas
Plan de prevención	Reservar un tiempo para la elección de las pruebas adecuadas en la fase de diseño de cada uno de los módulos que componen la plataforma.
Indicadores	Al realizar las tareas de validación y verificación de un módulo concreto, se constata que no cumple con los objetivos para los que fue pensado.
Plan de contingencia	Analizar las tareas restantes, realizando una replanificación, e incluso reajustando los plazos de entrega si fuese necesario.

Cuadro 2.22: Respuesta a RSG-12.

RSG-13	Errores inesperados en el código
Plan de prevención	Elaborar una planificación previsora poniendo un margen por si estos errores surgiesen.
Indicadores	Aparecen errores de una ejecución a otra de la plataforma, habiendo modificado o no el código de la misma.
Plan de contingencia	Analizar las tareas restantes, realizando una replanificación, e incluso reajustando los plazos de entrega si fuese necesario.

Cuadro 2.23: Respuesta a RSG-13.

2.6. Gestión de costes

Este apartado pretende analizar el coste económico que conlleva el desarrollo del proyecto. Para ello, se desglosarán los costes en directos e indirectos.

2.6.1. Costes directos

- **Costes de material:** en este caso, se tendrá en cuenta el coste del ordenador portátil con el que se ha desarrollado el proyecto. Se supondrá que la vida útil de un equipo es de 4 años y que, en este caso, todavía se encuentra dentro de la misma. Sabiendo su coste, 1300 €, y los días de proyecto, 82,5, se puede calcular el coste proporcional de la amortización:

$$1300 * 82,5/(4 * 365) = 73,46€ \tag{2.1}$$

- **Software y tecnologías utilizadas:** todo el software y tecnologías empleadas son de uso gratuito, salvo el MS Project, para el cual se pagó una mensualidad cuyo coste asciende a 30,6 €.

- **Sistema operativo:** se utilizó Ubuntu, distribución GNU/Linux sin coste asociado.

- **Mano de obra:** Para la estimación de los costes de los recursos, se ha realizado una búsqueda en la herramienta Experteer[6] para obtener cuál es el salario estimado en Santiago de Compostela para un recién licenciado en el Grado de Ingenería Informática y el resultado obtenido es de unos 19000€ anuales brutos a jornada completa. Para el cálculo total del coste de este recurso se ha hecho uso de la calculadora de contratos de la Oficina de Investigación y Tecnología de la USC[7]:

  1. Número de pagas: 14.
  2. Bruto mensual: 1357,00 €.
  3. Categoría: Grado – Investigador en formación.
  4. Período: 04-02-2019 al 29-05-2019.
  5. Horas semanales: 25.

Por otra parte, se tiene que tener en cuenta el coste de los tutores, ya que también son recursos que tienen asignadas 11,25 horas de tutoría. Para realizar el cálculo del coste, se utilizará la misma calculadora que para el alumno:

[6] https://www.experteer.es/salary_calculator/
[7] http://imaisd.usc.es/ferramentas/calculadora/calculadoracontratos.asp

Año	Bruto	Pagas extra	Liquidación	Coste contrato	Seguridad social	Total
2019	5195,06€	857,45€	196,80€	6249,31€	1975,85€	8225,16€

Cuadro 2.24: Costes recién graduado en el proyecto.

Tutor	Cargo USC	Salario mínimo mensual	Coste proyecto
David Losada	Doctor	1819,39 €	479,74 €
Juan Carlos Pichel	Doctor	1819,39 €	479,74 €
Rodrigo Martínez	Investigador predoctoral	1522,16 €	401,37 €

Cuadro 2.25: Tabla costes personal USC.

A continuación, se muestra la columna de coste desglosada para un doctor y para un investigador predoctoral:

Año	Bruto	Pagas extra	Liquidación	Coste contrato	Seguridad social	Total
2019	303.23€	49,71€	11,47€	364,41€	115,33€	479,74€

Cuadro 2.26: Costes de un doctor en el proyecto.

Año	Bruto	Pagas extra	Liquidación	Coste contrato	Seguridad social	Total
2019	253,69€	41,59€	9,60€	304,88€	96,49€	401,37€

Cuadro 2.27: Costes de un investigador predoctoral en el proyecto.

2.6.2. Costes indirectos

Además de los ya mencionados, es necesario tener en cuenta costes como la luz o Internet. Para estimar este coste, se tendrá en cuenta la cifra que aporta la USC para proyectos TIC: un 21 % adicional sobre los costes directos del proyecto.

2.6.3. Coste total

Costes de material	104,1 €
Costes de recursos	9586,01 €
Costes indirectos	2028,49 €
Total	**11718,56 €**

Cuadro 2.28: Tabla de costes del proyecto

Capítulo 3

Especificación de requisitos

3.1. Catálogo de requisitos del sistema

3.1.1. Identificación de subsistemas

En este apartado se presentarán los principales subsistemas que han sido identificados, los cuales facilitan la gestión del sistema que se desea crear o modelar. De forma más precisa, se han identificado seis subsistemas:

- **Subsistema principal:** engloba acciones que afectan a la configuración de las topologías de procesamiento por parte de los usuarios.
- **Subsistema de filtrado en tiempo real:** se corresponde con el módulo preinstalado del mismo nombre y soporta sus acciones principales.
- **Subsistema de tag clouds dinámicos:** se corresponde con el módulo preinstalado del mismo nombre y soporta sus acciones principales.
- **Subsistema de análisis de tópicos:** se corresponde con el módulo preinstalado del mismo nombre y soporta sus acciones principales.
- **Subsistema de procesamiento por lotes:** se corresponde con el módulo preinstalado del mismo nombre y soporta sus acciones principales.
- **Subsistema de estadísticas:** se corresponde con el módulo preinstalado del mismo nombre y soporta sus acciones principales.

3.1.2. Requisitos de información

Esta subsección presenta los distintos requisitos de información identificados, que pueden ser requisitos de almacenamiento y/o restricciones de información.

ID	**IRQ-01**
Nombre	*Información asociada a las publicaciones recuperadas.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Dependencias	.
Descripción	El sistema deberá almacenar la información correspondiente a cada uno de los textos recuperados para poder ser mostrada al usuario en la interfaz.
Campos de información	• Cuerpo de la publicación • Fuente • Timestamp • URL
Entidades asociadas	Usuario Interfaz, Web.
Tiempo de vida	Medio: 1 — Máximo: 2
Ocurrencias simultáneas	Medio: - — Máximo: -
Importancia	Importante
Urgencia	Hay presión
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.1: IRQ-01.

ID	**IRQ-02**
Nombre	*Información asociada a la salida de los módulos.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Dependencias	.
Descripción	El sistema deberá almacenar la salida de los distintos módulos para poder ser mostrada en la interfaz o para poder ser tratada con posterioridad.
Campos de información	• Identificador unívoco de qué módulo procede y con cuál de sus posibles salidas se corresponde (recordar que un módulo puede producir más de una salida) • Resultados producidos por el módulo en un formato dado
Entidades asociadas	Usuario Interfaz, Web.
Tiempo de vida	Medio: 7 — Máximo: -
Ocurrencias simultáneas	Medio: - — Máximo: -
Importancia	Importante
Urgencia	Hay presión
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.2: IRQ-02.

ID	**IRQ-03**
Nombre	*Información sobre los contenedores vivos.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Dependencias	.
Descripción	El sistema deberá almacenar qué contenedores se encuentran activos, de qué tipo son, cuáles son sus características y a dónde están conectados.
Campos de información	• Tipo del contenedor • Inputs de ese módulo • Características de rendimiento (% cpu y % memoria)
Entidades asociadas	Usuario Interfaz, Web.
Tiempo de vida	Medio: 7 — Máximo: -
Ocurrencias simultáneas	Medio: - — Máximo: -
Importancia	Importante
Urgencia	Hay presión
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	El almacenamiento de esta información permitirá no levantar módulos duplicados. Por ejemplo, si ya hay un filtro desplegado con las características solicitadas, en vez de levantar otro lo que se hará será añadir una nueva consulta al ya desplegado mediante una llamada rpc.

Cuadro 3.3: IRQ-03.

3.1.3. Requisitos funcionales

En esta subsección se presenta una lista de los requisitos funcionales que han sido seleccionados junto con el valor establecido para la identificación de los mismos:

- **Subsistema principal:**
  - RF-01: Lanzar módulo o fuente.
  - RF-02: Lanzar topología.
  - RF-03: Realizar llamada remota a módulo.
  - RF-04: Añadir y eliminar flujos en tiempo real.
  - RF-05: Eliminar módulo o fuente.
- **Subsistema de filtrado en tiempo real:**
  - RF-06: Realizar una búsqueda exacta.
  - RF-07: Realizar una búsqueda inexacta.
  - RF-08: Consultar un texto en la fuente.
- **Subsistema de tag clouds dinámicos:**
  - RF-09: Consultar *tag cloud*.
- **Subsistema de análisis de tópicos:**
  - RF-10: Seleccionar modo de análisis.
  - RF-11: Seleccionar número de tópicos.
  - RF-12: Consultar valores de cada tópico.
- **Subsistema de procesamiento por lotes:**
  - RF-13: Seleccionar franja temporal.
  - RF-14: Consultar textos emitidos en esa franja.
- **Subsistema de estadísticas:**
  - RF-15: Consultar textos totales.
  - RF-16: Consultar usuarios totales.
  - RF-17: Consultar textos por segundo.
  - RF-18: Consultar usuarios por segundo.

3.1.3.1. Actores

A continuación, se presentan los participantes en los distintos casos de uso identificados. Estos se presentarán de forma detallada en el siguiente apartado.

Actor	**Usuario interfaz**
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Descripción	Este actor representa a la persona que interactúa con la plataforma a través del módulo de la interfaz web.
Comentarios	-

Cuadro 3.4: ACT-01

3.1.3.2. Casos de uso del sistema

Para describir las funcionalidades de un sistema se pueden utilizar casos de uso o requisitos funcionales. En este caso se ha optado por utilizar las tablas de casos de uso y sus diagramas por considerarlos más ilustrativos, pero en los ids se ha mantenido la notación RF por coherencia con el resto de requisitos del sistema.

- **Subsistema principal**

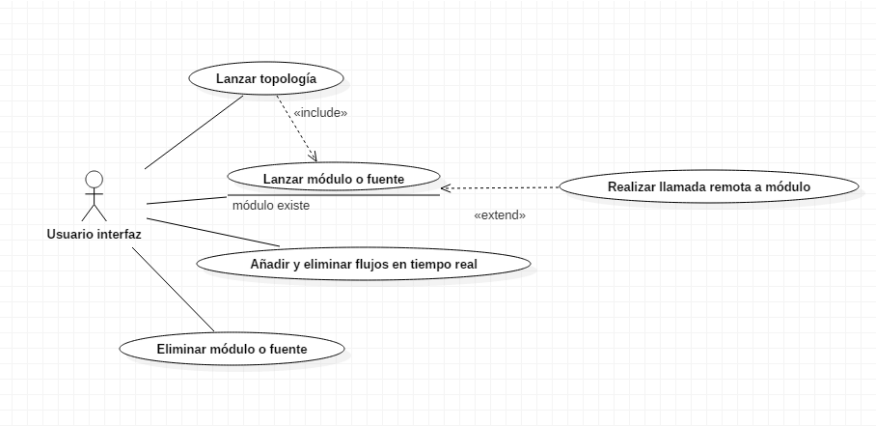

Figura 3.1: Diagrama casos de uso subsistema principal

- **Subsistema filtrado en tiempo real**

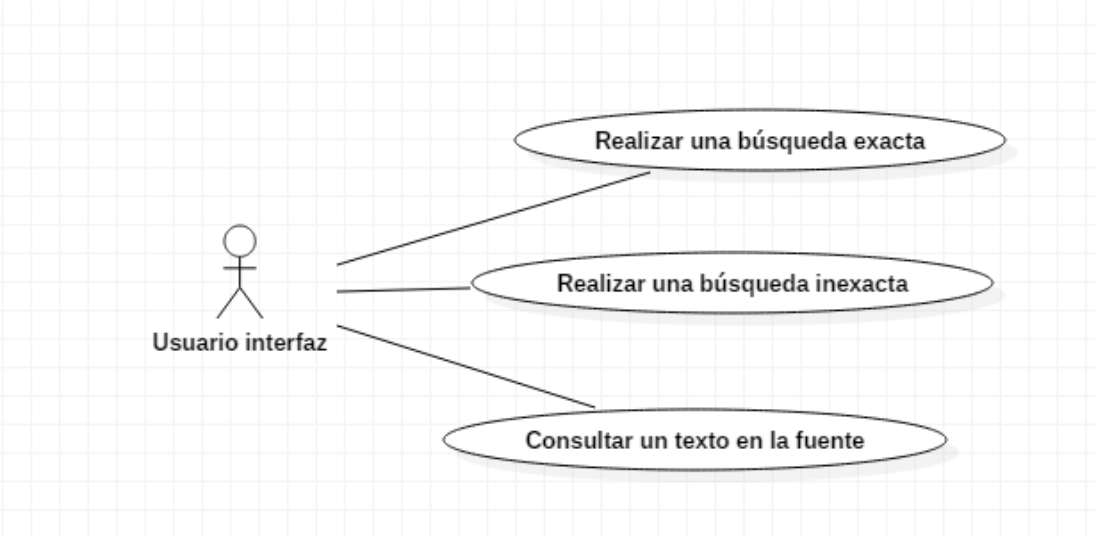

Figura 3.2: Diagrama casos de uso subsistema filtrado en tiempo real

- **Subsistema Tag Cloud dinámicos**

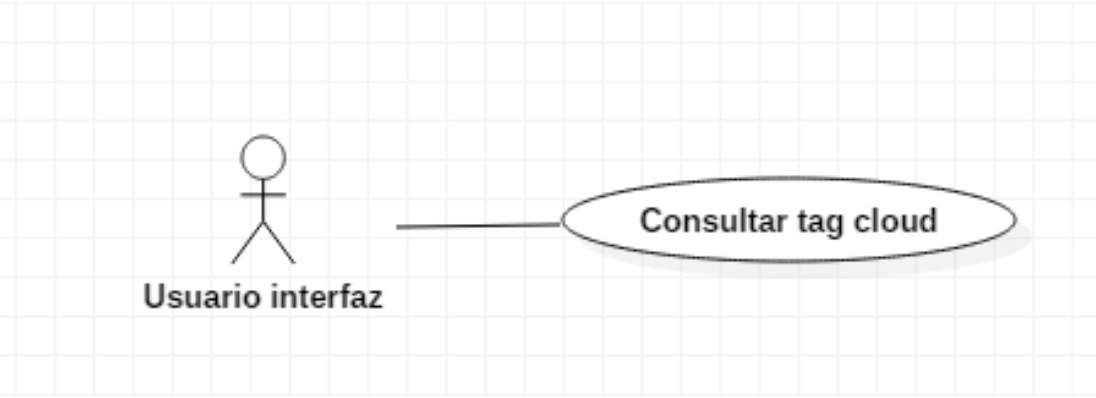

Figura 3.3: Diagrama casos de uso subsistema tag clouds dinámicos

- **Subsistema análisis de tópicos**

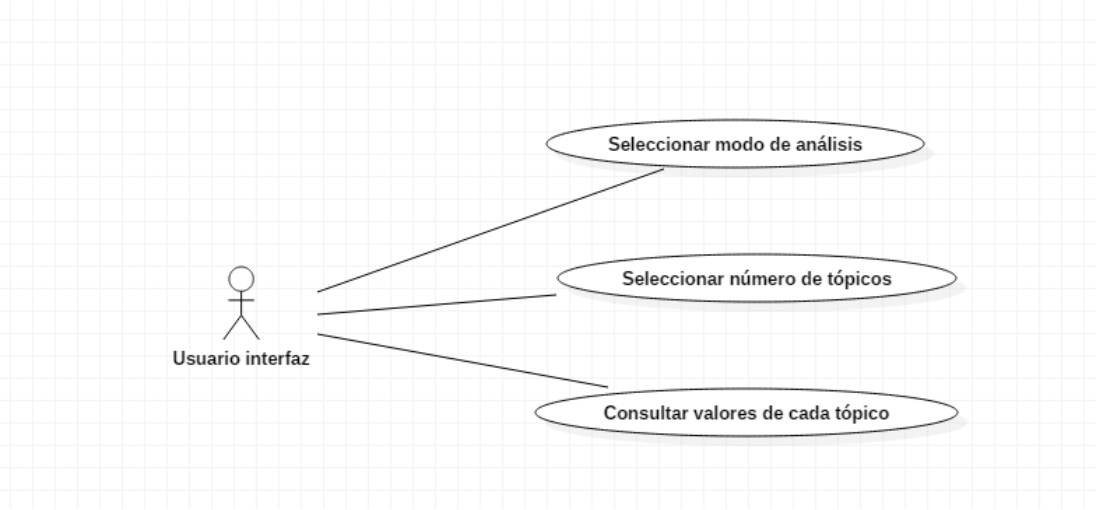

Figura 3.4: Diagrama casos de uso subsistema análisis de tópicos

- **Subsistema de procesamiento por lotes**

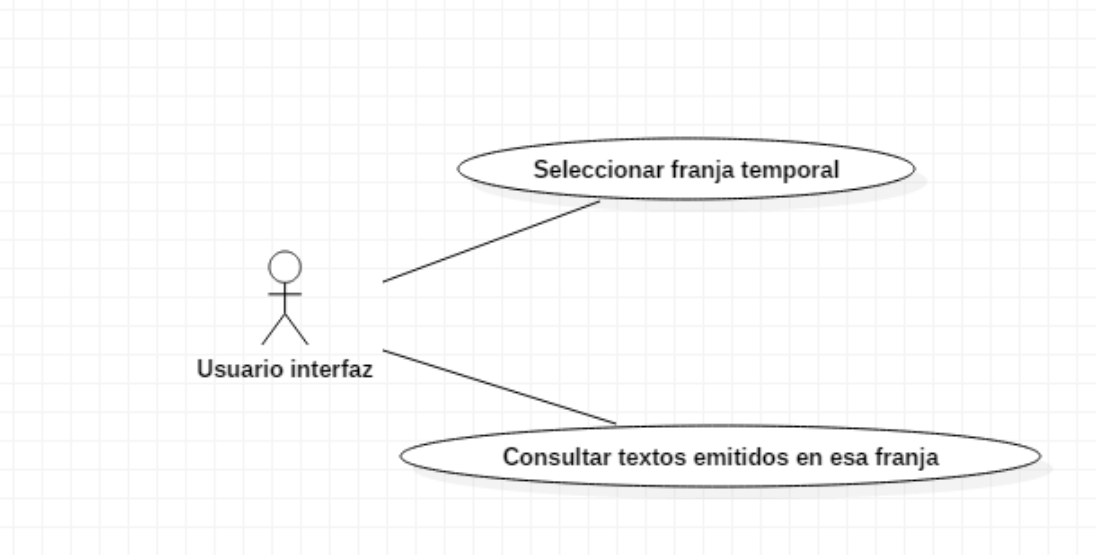

Figura 3.5: Diagrama casos de uso subsistema de procesamiento por lotes

- **Subsistema de estadísticas**

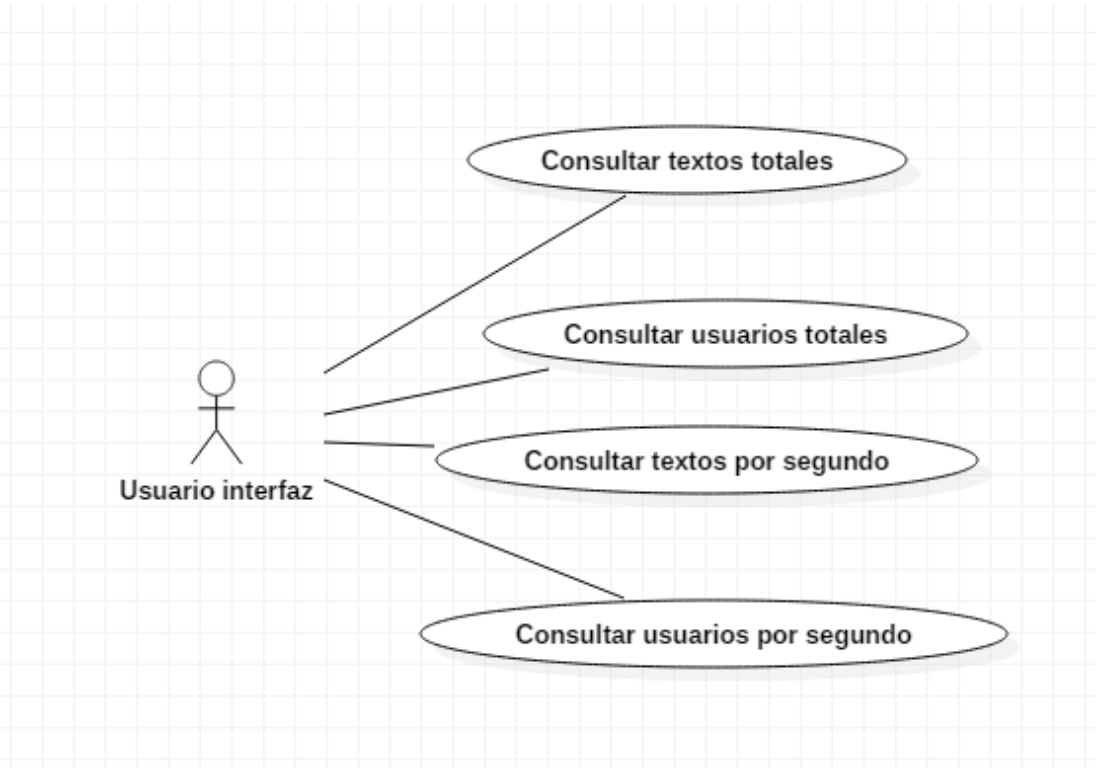

Figura 3.6: Diagrama casos de uso subsistema de estadísticas

ID	**RF-01**
Nombre	*Lanzar módulo o fuente.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-01 • OBJ-02 • OBJ-03
Descripción	Este caso de uso se centra en la posibilidad de realizar una petición a la plataforma a través de la API para que lance un contenedor que contenga un módulo o fuente compatible con nuestra plataforma. Este módulo puede ser uno de los preinstalados o una imagen Docker creada por uno de los usuarios de *Indivisa*.
Precondición	Tener la plataforma desplegada y la API escuchando
Secuencia normal	• El usuario indica la imagen que desea lanzar y sus características (a dónde quiere conectarla, opciones, etc.) • Se envía la petición a la API • La plataforma lanza el nuevo contenedor con el módulo dentro
Postcondición	El módulo ha sido lanzado en la plataforma y se integra según lo descrito en la petición
Excepciones	Ver RF-03
Criterio de aceptación	Lanzar una imagen de un módulo de los preinstalados y ver que se conecta según lo esperado. También probar con otra imagen que no sea de las "esperadas" por la plataforma, ya sea un *crawler* o un nuevo módulo y ver que se lanza correctamente.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.8: RF-01

ID	**RF-02**
Nombre	*Lanzar topología.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-01 • OBJ-02 • OBJ-08
Descripción	Este caso de uso se centra en la posibilidad de enviar un único fichero a la plataforma en el que se describa una topología completa y que ésta sea capaz de parsearlo y lanzarla.
Precondición	Tener la plataforma desplegada y la API escuchando
Secuencia normal	• El usuario indica la ruta al fichero que describe la topología en la petición • Se envía la petición a la API • La plataforma lanza los contenedores mediante llamadas individuales
Postcondición	La topología ha sido correctamente lanzada según las relaciones descritas en el fichero
Excepciones	-
Criterio de aceptación	Lanzar una topología y ser capaz de comprobar que los módulos desplegados en la plataforma cumplen con las relaciones descritas en el fichero pasado.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	El objetivo es que *Indivisa* sea totalmente compatible tanto con *Kubernetes* como con *Catenae*, esto quiere decir que sea capaz de procesar composiciones de ambos formatos.

Cuadro 3.9: RF-02

ID	**RF-03**
Nombre	*Realizar llamada remota a módulo.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-01 • OBJ-02 • OBJ-08
Descripción	Este caso de uso modela la posibilidad de que un determinado tipo de módulo ya exista en la plataforma y que, en vez de desplegar otro contenedor igual, se realice una llamada remota a dicho módulo para modificar su comportamiento.
Precondición	Tener la plataforma desplegada y la API escuchando. Además, debe existir el módulo solicitado (tipo de imagen y mismas conexiones).
Secuencia normal	• El usuario indica la imagen que desea lanzar y sus características (a dónde quiere conectarla, opciones, etc.) • Se envía la petición a la API • Se comprueba que ya hay un contenedor desplegado con esas características (punto de excepción) • Se realiza una llamada remota a dicho módulo para modificar su comportamiento, por ejemplo, añadir consulta si se trata de un filtro
Postcondición	No se ha levantado un nuevo módulo y el módulo existente ha modificado su comportamiento de acuerdo a lo recibido en la llamada remota
Excepciones	-
Criterio de aceptación	Realizar una llamada a un módulo preinstalado y ya desplegado y ver que no se levanta un nuevo contenedor y que su comportamiento se modifica.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	Llegado cierto punto de carga de los módulos habría que desplegar un segundo contenedor. No se contemplará en la implementación.

Cuadro 3.10: RF-03

ID	**RF-04**
Nombre	*Añadir y eliminar flujos en tiempo real.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-01 • OBJ-02 • OBJ-08
Descripción	Este caso de uso modela la posibilidad de añadir y eliminar flujos de entrada de un módulo en tiempo real.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario indica el módulo, el flujo de entrada y la acción (añadir o eliminar) • Se envía la petición a la API
Postcondición	Se añade o elimina el flujo deseado en el módulo indicado
Excepciones	-
Criterio de aceptación	Realizar una llamada a un módulo desplegado añadiendo y posteriormente eliminando un flujo existente en la topología.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.11: RF-04

ID	**RF-05**
Nombre	*Eliminar módulo o fuente.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-01 • OBJ-02 • OBJ-08
Descripción	Este caso de uso modela la posibilidad de eliminar un módulo desplegado.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario indica el módulo a eliminar • Se envía la petición a la API
Postcondición	Se elimina el módulo de la topología existente y desaparecen sus flujos de salida también
Excepciones	-
Criterio de aceptación	Probar a eliminar un módulo de los preinstalados y comprobar que desaparece, así como sus flujos de salida.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.12: RF-05

ID	**RF-06**
Nombre	*Realizar una búsqueda exacta.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-03
Descripción	Este caso de uso modela la posibilidad de realizar una búsqueda exacta de la consulta proporcionada por el usuario. Por ejemplo, si el usuario teclea "Theresa May", el filtro devolverá los textos que contengan esa cadena en ese orden sin ninguna modificación.
Precondición	Tener un módulo de filtrado de textos desplegado y conectado a alguna fuente de datos.
Secuencia normal	• El usuario teclea la consulta • El usuario la marca como exacta • El filtro devuelve los textos que la contengan
Postcondición	Se devuelven los textos que contengan la consulta expresada por el usuario.
Excepciones	-
Criterio de aceptación	Probar a realizar 5 consultas exactas en el módulo de filtrado preinstalado que ofrece *Indivisa*.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.13: RF-06

ID	**RF-07**
Nombre	*Realizar una búsqueda inexacta.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-03
Descripción	Este caso de uso modela la posibilidad de realizar una búsqueda inexacta de la consulta proporcionada por el usuario. Por ejemplo: si el usuario teclea *the White House*, el filtro eliminaría el artículo *the* (al tratarse de un *stopword*) y devolvería textos que contengan las palabras *white* y *house*, pero no estrictamente en ese orden o una a continuación de la otra.
Precondición	Tener un módulo de filtrado de textos desplegado y conectado a alguna fuente de datos.
Secuencia normal	• El usuario teclea la consulta • El usuario la marca como inexacta • El filtro devuelve los textos que la contengan
Postcondición	Se devuelven los textos que contengan la consulta expresada por el usuario.
Excepciones	-
Criterio de aceptación	Probar a realizar 5 consultas inexactas en el módulo de filtrado preinstalado que ofrece *Indivisa*.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.14: RF-07

ID	**RF-08**
Nombre	*Consultar un texto en la fuente.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-03
Descripción	Este caso de uso contempla la posibilidad de consultar un texto filtrado en la fuente original a través de su URL.
Precondición	Tener un módulo de filtrado de textos desplegado, conectado a alguna fuente de datos y filtrando por una consulta dada.
Secuencia normal	• El usuario selecciona uno de los textos filtrados y clica en su URL • Ésta lo redirecciona a la publicación original
Postcondición	El usuario accede a la publicación original.
Excepciones	-
Criterio de aceptación	Probar acceder a publicaciones de las dos fuentes preinstalas: *Twitter* y *Reddit*.
Frecuencia esperada	PD
Importancia	Quedaría bien
Urgencia	Puede esperar
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.15: RF-08

ID	**RF-09**
Nombre	*Consultar tag cloud.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-04
Descripción	Este caso de uso contempla la posibilidad de consultar una nube de palabras que sirve de resumen acerca de un determinado flujo de textos de entrada y que se va actualizando en tiempo real.
Precondición	Tener un módulo de generador de *tag clouds* desplegado y conectado a alguna fuente de datos .
Secuencia normal	• El usuario selecciona el módulo de *tag clouds* • Este observa la nube de palabras que se va actualizando en tiempo real y que sirve de resumen del flujo de entrada
Postcondición	-
Excepciones	-
Criterio de aceptación	Lanzar un generador de *tag clouds* conectado a un filtro y ver que se actualiza en tiempo real y que los porcentajes que genera tienen sentido.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.16: RF-09

ID	**RF-10**
Nombre	*Seleccionar modo análisis.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-05
Descripción	Este caso de uso contempla la posibilidad de seleccionar dos modos de análisis: por cantidad o en una franja temporal. En el primer caso, se especifica una cantidad máxima de textos para realizar el análisis. Se tratará de una estructura circular en la que los textos más antiguos vayan siendo sustituidos por los más nuevos. El segundo caso, modela preguntas del estilo: "me gustaría saber qué se dijo sobre Trump en la última hora". Por tanto, se seleccionan sólo los textos que cumplan con esa condición temporal.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona uno de los dos modos al desplegar el módulo de análisis • Se envía la petición a la API • Se despliega el módulo con el modo deseado
Postcondición	Se despliega el módulo con el modo deseado
Excepciones	-
Criterio de aceptación	Lanzar dos módulos de análisis, cada uno con su modo y comprobar que se realiza correctamente el despliegue.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.17: RF-10

ID	**RF-11**
Nombre	*Seleccionar número de tópicos.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-05
Descripción	Este caso de uso contempla la posibilidad de seleccionar el número de tópicos con los que se desea realizar el análisis. Cabe recordar que un módulo puede realizar más de un análisis a la vez, por ejemplo, para sus flujos de entrada puede emitir 5 y 20 tópicos o 6 y 30 tópicos.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona el número de tópicos • Se envía la petición a la API • El módulo recibe el dato y empieza a generar las palabras clave para ese número de tópicos
Postcondición	El número de tópicos emitido es el expresado por el usuario en su petición
Excepciones	-
Criterio de aceptación	Probar a desplegar un módulo de análisis de tópicos con un número de tópicos inicial y que se haga correctamente. A continuación, a ese mismo módulo, añadirle un nuevo número de tópicos mediante una llamada remota y que el resultado también sea el esperado: que conviva con el número de tópicos inicial de manera independiente sin mezclarse.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.18: RF-11

ID	**RF-12**
Nombre	*Consultar valores de cada tópico .*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-05
Descripción	Este caso de uso contempla la posibilidad de consultar la salida independiente de cada análisis de tópicos realizado. Cabe recordar que un módulo puede realizar más de un análisis a la vez, por ejemplo, para sus flujos de entrada puede emitir 5 y 20 tópicos o 6 y 30 tópicos.
Precondición	Tener un módulo de análisis de tópicos emitiendo resultados.
Secuencia normal	• El usuario puede seleccionar una de las salidas del módulo de análisis • Se envía la petición a la API • El usuario recibe los datos relativos al análisis (número de tópicos, palabras por tópico, porcentajes de cada palabra,etc)
Postcondición	El usuario recibe los datos del análisis que solicita
Excepciones	-
Criterio de aceptación	Probar a solicitar los datos de tres análisis diferentes de un mismo módulo.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.19: RF-12

ID	**RF-13**
Nombre	*Seleccionar franja temporal.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-06
Descripción	Este caso de uso contempla la posibilidad de seleccionar la franja temporal en la que se desea realizar el procesamiento por lotes.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona la franja temporal que es de su interés • Se envía la petición a la API • El módulo la recibe y empieza a procesar los textos que cumplan con esa condición temporal
Postcondición	El usuario recibe el resultado que le interesa para los textos que cumplen con la condición temporal que él expresó en un primer momento.
Excepciones	-
Criterio de aceptación	Probar a desplegar un módulo de procesamiento por lotes con una franja temporal ya especificada y que se haga correctamente. A continuación, a ese mismo módulo, añadirle una nueva ventana mediante una llamada remota y que el resultado también sea el esperado.
Frecuencia esperada	PD
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.20: RF-13

ID	**RF-14**
Nombre	*Consultar textos emitidos en esa franja.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-06
Descripción	Este caso de uso contempla la posibilidad de consultar el resultado del procesamiento por lotes en una franja temporal dada.
Precondición	Tener un módulo batch desplegado y procesando textos en una franja dada.
Secuencia normal	• El usuario selecciona la franja temporal que es de su interés • Se envía la petición a la API • El módulo retorna el número de textos asociados a esa franja temporal
Postcondición	El usuario recibe el número de textos de esa franja temporal.
Excepciones	-
Criterio de aceptación	Probar a repetir el procesamiento en la misma franja temporal un par de veces y comprobar que el resultado es el mismo.
Frecuencia esperada	PD
Importancia	Importante
Urgencia	Hay Presión
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.21: RF-14

ID	**RF-15**
Nombre	*Consultar textos totales.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-07
Descripción	Este caso de uso contempla la posibilidad de consultar los textos totales procesados por *Indivisa* desde que fue desplegada.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona la info que quiere consultar • Se envía la petición a la API • El módulo de estadísticas retorna el número de textos totales
Postcondición	El usuario recibe el número de textos totales desde que la plataforma fue arrancada.
Excepciones	-
Criterio de aceptación	Hacer dos solicitudes separadas en el tiempo, una a continuación de la otra, y ver que este número crece.
Frecuencia esperada	PD
Importancia	Quedaría bien
Urgencia	Puede esperar
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.22: RF-15

ID	**RF-16**
Nombre	*Consultar usuarios totales.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-07
Descripción	Este caso de uso contempla la posibilidad de consultar los usuarios totales procesados por *Indivisa* desde que fue desplegada.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona la info que quiere consultar • Se envía la petición a la API • El módulo de estadísticas retorna el número de usuarios totales
Postcondición	El usuario recibe el número de usuarios totales analizados desde que la plataforma fue arrancada.
Excepciones	-
Criterio de aceptación	Hacer dos solicitudes separadas en el tiempo, una a continuación de la otra, y ver que este número crece.
Frecuencia esperada	PD
Importancia	Quedaría bien
Urgencia	Puede esperar
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.23: RF-16

ID	**RF-17**
Nombre	*Consultar textos por segundo.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-07
Descripción	Este caso de uso contempla la posibilidad de consultar la velocidad a la que procesa *Indivisa* expresada en textos por segundo.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona la info que quiere consultar • Se envía la petición a la API • El módulo de estadísticas retorna los textos por segundo que la plataforma está procesando
Postcondición	El usuario recibe el número de textos por segundo que la plataforma se encuentra procesando.
Excepciones	-
Criterio de aceptación	Hacer dos solicitudes separadas en el tiempo, una a continuación de la otra, y ver que el número se mantiene más o menos constante.
Frecuencia esperada	PD
Importancia	Quedaría bien
Urgencia	Puede esperar
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.24: RF-17

ID	**RF-18**
Nombre	*Consultar usuarios por segundo.*
Versión	1.0 (23/02/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	Relacionado con los objetivos • OBJ-07
Descripción	Este caso de uso contempla la posibilidad de consultar la velocidad a la que procesa *Indivisa* expresada en usuarios distintos analizados por segundo.
Precondición	Tener la plataforma desplegada y la API escuchando.
Secuencia normal	• El usuario selecciona la info que quiere consultar • Se envía la petición a la API • El módulo de estadísticas retorna los usuarios por segundo que la plataforma está procesando
Postcondición	El usuario recibe el número de usuarios por segundo que la plataforma se encuentra procesando.
Excepciones	-
Criterio de aceptación	Hacer dos solicitudes separadas en el tiempo, una a continuación de la otra, y ver que el número se mantiene más o menos constante.
Frecuencia esperada	PD
Importancia	Quedaría bien
Urgencia	Puede esperar
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.25: RF-18

3.1.4. Requisitos no funcionales

ID	**RNF-01**
Nombre	*Utilización de la librería Catenae.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	La librería Python que se utilizará para el desarrollo de las topologías de procesamiento dentro de la plataforma será *Catenae*, desarrollada en el CiTIUS.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.26: RNF-01

ID	**RNF-02**
Nombre	*Utilización de MongoDB como sistema de base de datos.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	Se utilizará MongoDB para la persistencia de los datos relativos a los contenedores. Más concretamente, se almacenará una lista de los contenedores vivos, así como, las colas de salida de cada módulo y el valor que emiten, el cual se irá actualizando.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.27: RNF-02

ID	**RNF-03**
Nombre	*Utilización de Apache Kafka para las comunicaciones.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	Las comunicaciones entre los distintos módulos de las topologías se realizarán mediante colas de Apache Kafka. Esto tiene la ventaja de que nos permite escalar fácilmente gracias a la creación de grupos de consumo y, por otra parte, el procesamiento batch surge de manera natural al poder configurar el tiempo que los mensajes permanecen en las colas de Kafka.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.28: RNF-03

ID	**RNF-04**
Nombre	*Utilización de Gensim para el análisis de tópicos.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	Se utilizará la librería Python Gensim para el análisis de tópicos al tratarse de una herramienta especializada para el modelado y procesamiento del lenguaje natural en grandes colecciones de texto que es, precisamente, nuestro caso.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.29: RNF-04

ID	**RNF-05**
Nombre	*Compatibilidad con ficheros de composición de Kubernetes.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	A la hora de desplegar topologías, la plataforma leerá un fichero en formato .yaml y extraerá la información para desplegar los contenedores y sus conexiones. El objetivo es que sea capaz de leer ficheros en un formato para su despliegue en el gestor de contenedores *Kubernetes*, consiguiendo así la compatibilidad entre nuestra plataforma y este gestor.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.30: RNF-05

ID	**RNF-06**
Nombre	*Definición de una API de tipo RESTful.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	Definición de una serie de *endpoints* que permitan a un usuario externo interaccionar con la plataforma desde una aplicación web.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.31: RNF-06

ID	**RNF-07**
Nombre	*Desarrollo de una aplicación web en React.*
Versión	1.0 (30/03/2019)
Autores	Marcos Fernández Pichel
Fuentes	Equipo de investigación
Dependencias	-
Descripción	Se ha decidido optar por *React*, al tratarse de un framework ampliamente utilizado en el mercado y desarrollado por una gran compañía como Facebook.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Pendiente de verificación
Estabilidad	Alta
Comentarios	-

Cuadro 3.32: RNF-07

Capítulo 4

Análisis de tecnologías y herramientas

4.1. Tecnologías empleadas en el desarrollo

A continuación, se describen las tecnologías empleadas en el desarrollo del proyecto.

4.1.1. Lenguajes de programación: Python

Se ha decidido utilizar el lenguaje Python, ya que es el tercero más utilizado según el último índice TIOBE, sólo por detrás de C y Java, y es uno de los dominadores en el ámbito del análisis de datos.

Se encuentra por encima de sus competidores, gracias a su facilidad de aprendizaje, manejo sencillo y eficiencia a la hora de analizar los datos. Su sencillez ahorra muchas horas de capacitación y entrenamiento que pueden ser dedicadas al propio *Big Data* o *Business Intelligence*.

Por otra parte, alguna de las librerías clave del proyecto, como por ejemplo Gensim, se encuentran desarrolladas para Python fundamentalmente.

4.1.2. Docker

Los distintos módulos de la plataforma se encapsulan en contenedores Docker. Con ello, lo que se consigue es empaquetar una aplicación y sus dependencias en un entorno virtualizado que puede ser ejecutado en cualquier servidor Linux, favoreciendo así el aislamiento y la portabilidad. Además, carece de la penalización de E/S característica de otros sistemas de virtualización. Docker utiliza caracterísitcas de aislamiento del kernel de Linux, como *cgroups* y *namespaces* para permitir que contenedores independientes se ejecuten dentro de un mismo Linux anfitrión.

Es importante distinguir los conceptos **imagen** y **contenedor**. Los contenedores son imágenes desplegadas que se encuentran paradas o en ejecución. La principal manera de genera imágenes es a partir de un *Dockerfile*. Estos son scripts que modifican una imagen base, normalmente una distribución como Debian o Ubuntu. Además de modificar la imagen base a través de instalaciones, se pueden realizar otras acciones como, por ejemplo, exponer puertos o montar volúmenes.

4.1.3. React

Para la implementación de la interfaz web, se ha decidido utilizar el framework React js[1] desarrollado por Facebook. Los principales objetivos de React son la velocidad, la escalabilidad y la sencillez. Se correspondería con la vista de un modelo MVC y se puede combinar con otros frameworks como, por ejemplo, Angular JS. Permite a los desarrolladores crear grandes webs que pueden cambiar sus datos sin recargar la página y favorece la reutilización de componentes.

Actualmente, es el framework JavaScript más utilizado y mejor valorado entre los usuarios de Github[2].

4.1.4. Librerías

4.1.4.1. Catenae

Catenae[2] es una librería desarrollada en el CiTIUS. Está pensada para la construcción y ejecución de topologías Python de procesamiento de datos en tiempo real. Estas topologías están diseñadas para su despliegue en contenedores Docker, por lo que encajan de manera muy natural con lo expuesto en el apartado anterior. Además, se puede conectar con sistemas que no estén desarrollados en Python.

Una de las principales necesidades cuando se trabaja con ingentes cantidades de información es poder evitar los cuellos de botella y conseguir que la tecnología escale fácilmente. Aquí es donde *Catenae* tiene uno de sus puntos fuertes gracias a la utilización del framework Apache Kafka para la comunicación entre los nodos de la topología.

Este es un sistema distribuido en el que sus clientes pueden actuar como consumidores o productores de mensajes de un determinado topic escribiéndolos o leyéndolos de los brokers de Kafka. Internamente la plataforma se encarga de distribuir los mensajes por clave entre una serie de particiones, las cuales pueden estar en diferentes localizaciones físicas, y de replicarlos, para así protegerlos frente a posibles pérdidas.

[1] https://reactjs.org/

[2] https://www.c-sharpcorner.com/article/most-popular-front-end-javascript-framework-in-the-world/

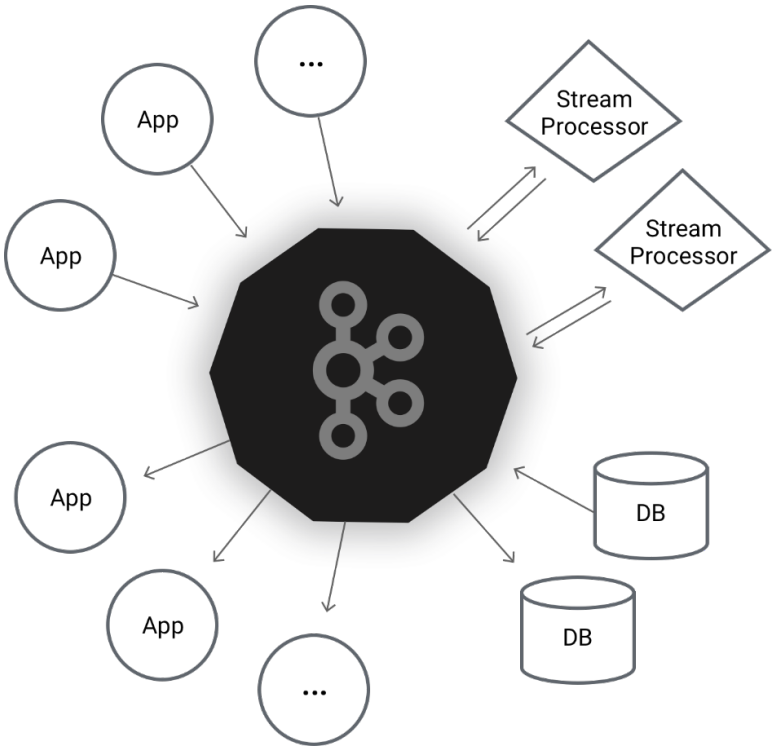

Figura 4.1: Diagrama de alto nivel de Apache Kafka
Cortesía de `https://kafka.apache.org/`

Sin embargo, la propiedad que más favorece a *Catenae* es la existencia de los denominados grupos de consumo: si a dos consumidores de Kafka se le asigna el mismo grupo de consumo se repartirán los mensajes del *topic* al que estén suscritos y, por tanto, se repartirán la **carga**. Esto hace que cada nodo pueda ser instanciado múltiples veces, de manera que si alguno de ellos se convierte en un cuello de botella, sólo es necesario replicarlo. Por último, cabe destacar que a diferencia de otros frameworks, *Catenae* asigna los recursos a cada nodo de manera individual.

4.1.4.2. Gensim

Gensim[5] es una librería para modelado de tópicos y el procesado en lenguaje natural, utilizando técnicas de *machine learning*. Está diseñado para manejar grandes cantidades de textos e incluye algoritmos como LDA, LSA o tf-idf.

En el caso concreto de este proyecto, se utilizará en el módulo de *topic analysis* para extraer los principales temas de los textos que ingeste este módulo desde Kafka. Cabe señalar que para la visualización de los tópicos, se utilizará la librería pyLDAvis[3].

[3]https://github.com/bmabey/pyLDAvis

4.1.4.3. Vis js

Se utilizará la librería Vis js[4] para la representación de la topología en la interfaz.

4.1.5. Mongo DB

Para guardar los datos relativos a la salida de los distintos módulos y a su tipo, se crearán dos colecciones en el sistema de base de datos NoSQL MongoDB. En lugar de guardar los datos en tablas, se guardan en estructuras de datos con un esquema dinámico (similares a un JSON), haciendo que la integración con cierta plataformas sea fácil y rápida.

4.2. Tecnologías empleadas en la documentación

4.2.1. Overleaf

Overleaf es un editor colaborativo de Latex en línea. Tiene control de versiones integrado y almacenamiento en la nube. Una vez terminado, permite exportar el archivo a PDF. Es utilizado para la redacción de la presente memoria.

4.2.2. WBS Tool

WBS Tool[5] es un editor en línea para la generación de Estructuras de Descomposición del Trabajo. Permite descargarlas en varios formatos y volver a subirlas para editarlas. El EDT del apartado 2.3.1 ha sido generado con dicha herramienta.

4.2.3. MS Project

MS Project[6] es una herramienta de Microsoft para la administración de proyectos. Con ella, se creó el Diagrama de Gantt del apartado 2.3.3. Se utilizó la versión 1905.

4.2.4. Draw.io

Draw.io es una herramienta para la generación de gráficos, con la que se crearón los diagramas de la presente memoria.

[4]https://github.com/visjs
[5]http://www.wbstool.com/
[6]https://products.office.com/es-es/project/project-and-portfolio-management-software

4.2.5. Star UML

Star UML[7] es una herramienta para la generación de diagramas UML. Se utilizó para la creación de los diagramas de casos de uso y de clases de la presente memoria. La versión empleada fue la 2.7.0.

[7]http://staruml.io/

Capítulo 5

Diseño e implementación

5.1. Arquitectura local del sistema

La Figura 5.1 ilustra la arquitectura de *Indivisa* para un despliegue en local, que es el que ahora mismo se encuentra implementado, a pesar de que el diseño planteado permitiría migrarlo a un clúster casi de manera trivial, tal y como se verá en apartados posteriores. Cabe aclarar que cada uno de los elementos presente en el diagrama estaría desplegado dentro de un contenedor *Docker*.

En esta figura, es importante distinguir entre lo que se encontraría desplegado desde un primer momento y lo que se iría añadiendo con la interacción del usuario: todo estaría levantado desde el principio salvo los módulos preinstalados (filtro, *tag cloud*, analizador de tópicos, estadísticas y *batch*), los cuales se podrían configurar para conformar la topología de consumo que se desee. En este caso, se ha decidido poner una de ejemplo en la que se encuentren todos los tipos de módulos para, así, poder explicarlos a continuación.

Otro aspecto a destacar son las conexiones entre los distintos módulos y los módulos y los crawlers, las cuales se realizan a través de *topics* o colas de mensajes de *Apache Kafka*.

Por último, se puede observar como la API es la que comunica la plataforma con la aplicación web en *React* y es el módulo Python encargado de mantener y actualizar las tablas de Mongo DB. Éstas guardan la salida de cada módulo y el tipo de módulos desplegados en un determinado momento en la plataforma.

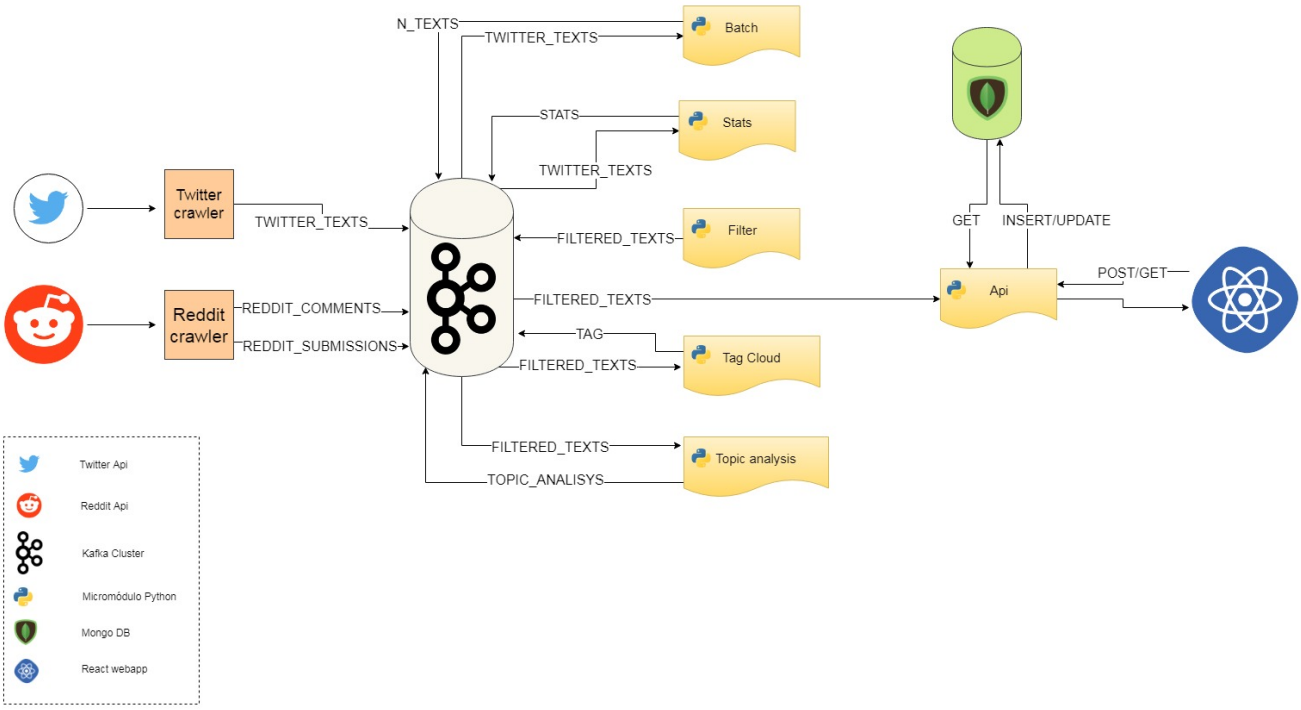

Figura 5.1: Diagrama Arquitectura del sistema local

5.1.1. Crawlers de consumo

Como se puede apreciar en la Figura 5.1, se ha decidido representar los *crawlers* como "cajas negras" que consumen de la API de Twitter y Reddit, respectivamente. Esto es debido a que son topologías enteras heredadas de *Catenae* que se encargan de realizar técnicas de *web scraping* para obtener los datos de la web en tiempo real. No pertenecen realmente a *Indivisa*, pero sí son fundamentales para que ésta disponga de los datos con los que poder trabajar.

Lo que realmente resulta interesante es la integración de la plataforma con estas cajas estancas. Como se puede apreciar, al trabajar tanto *Catenae* como *Indivisa* con colas de *Kafka*, la integración es muy simple, ya que los módulos que quieran consumir de algún *crawler* sólo tienen que suscribirse a su *topic* de salida y empezarán a recibir mensajes en tiempo real.

5.1.2. Módulo de filtrado en tiempo real

Este módulo es uno de los más importantes de la plataforma. Se suele colocar al principio de las topologías para realizar un primer barrido de la información obtenida de las fuentes. Su principal función es filtrar los textos por una consulta dada por el usuario.

Esta búsqueda puede ser exacta o inexacta. En el primero de los casos, si el usuario teclea *The White House*, el módulo le devolverá aquellos textos que contengan esa cadena exactamente igual y en el mismo orden. Mientras que, en el segundo caso, el módulo eliminaría las *stopwords*, en este caso, *the* y devolvería los textos anteriores y a mayores los textos que contengan *white* y *house*, pero ya en cualquier orden y pueden no estar consecutivos (tratamiento estilo *bag of words* con eliminación de palabras comunes o *stopwords*).

```
def transform(self, electron):
    key = electron.key
    value = electron.value
    for q,e,o,t in zip(self.fText,self.exact,self.original,self.
    topics):
        if e == "0": # exact_match
            title = ""
            if 'submission_title' in value:
                title = procChain(value['submission_title'])
            content = procChain(value['body'])
            if (value['type'] == 0 and q in title) or q in content:
                electron.topic = t
                self.send(electron)

        elif e == "1": # non-exact match
            title = set()
            if 'submission_title' in value :
                title = set(queryTokens(value['submission_title'],
    self.languages))
            content = set(queryTokens(value['body'],self.languages))
            if (value['type'] == 0 and title.issuperset(set(o))) or
    content.issuperset(set(o)):
                electron.topic = t
                self.send(electron)
```

Fragmento de código 5.1: Algoritmo de filtrado en tiempo real

En el fragmento de código superior, se puede observar la parte central del **algoritmo de filtrado**. En primer lugar, cabe realizar una serie de aclaraciones: el método *transform* es heredado de la clase *Link* de *Catenae* y se ejecuta cada vez que el módulo recibe un nuevo mensaje, mientras que la variable llamada "electron" hace referencia a ese mensaje que se recibe de *Kafka* en un determinado instante; por otra parte, como se puede apreciar, todo el código se encuentra encerrado dentro de un bucle for, ya que un mismo módulo puede estar filtrando por varias consultas o *queries* al mismo tiempo.

Una vez dentro del bucle, se comprueba si la consulta expresada por el usuario es exacta o no. En el primero de los casos, se procesa tanto el título como el contenido de la publicación recibida (o sólo el contenido si se trata de un comentario de una publicación) quitándole todos los caracteres especiales y pasándolo a minúsculas, proceso que previamente ya se ha realizado con el texto de la consulta. Una vez hecho esto se comprueba si alguno de ellos o ambos contienen **exactamente** el texto solicitado.

Mientras que, en el segundo de los casos, se realiza un preprocesado similar al anterior, se eliminan las *stopwords* de las publicaciones y se dividen por palabras para, posteriormente, construir un conjunto sin repeticiones. El texto de la

consulta habrá pasado por un proceso similar y la comprobación que se hace es si un **conjunto contiene al otro**.

Por último, se envía el "electron" al siguiente módulo de la topología a través del *topic* que los conecta.

5.1.3. Módulo de tag clouds dinámicos

Este módulo consiste en una primera aproximación a la generación de **resúmenes automáticos** en tiempo real. En este caso, se ha optado por la opción más sencilla de implementar, una nube de palabras, pero en futuras versiones se podría ampliar a mecanismos más sofisticados de resumen.

Como se puede apreciar en el fragmento 5.2, cada vez que llega un mensaje a este módulo se añade a una estructura de datos que lo almacena, en este caso se trata de una cola FIFO de tamaño limitado, para a continuación pasárselo a un método de generación de nubes de palabras que se puede ver en el fragmento 5.3.

En este último, se itera por cada uno de los mensajes almacenados y se le aplica un preprocesado (limpieza de *stopwords*, eliminación de caracteres especiales, etc). Una vez hecho esto se concatenan todos los mensajes en una única cadena para la generación del resumen. Por último, se retorna un diccionario con cada palabra y sus pesos, el cual se pasará a la interfaz a través de la API para poder ser renderizado.

```
1 def transform(self, electron):
2     self.msgs.add(electron.value.items())
3     electron.topic = self.output_topics[0]
4     electron.value = self.__generate_word_cloud(self.msgs)
5     return electron
```

Fragmento de código 5.2: Método transform del tag cloud dinámico

```
# Function that generates a word cloud of a given query
def __generate_word_cloud(self,msgs):
    clean_texts = []
    texts = list(msgs) # we need to put it as a list to avoid run
    time errors related with changing size of the set
    for text in texts:
        for element in text:
            if element[0] == b'body' or element[0] == b'
    submission_title':
                aux = preprocessForTag(str(element[1]),self.
    languages)
                if aux not in clean_texts:
                        clean_texts.append(aux)
    output = ' '.join(map(str,clean_texts))
    wordcloud = WordCloud(background_color="white").generate(output)
    result = []
    for t, v in wordcloud.words_.items():
        result.append({"text":t,"value":v*1000})
    return result
```

Fragmento de código 5.3: Agoritmo de generación del tag cloud dinámico

5.1.4. Módulo analizador de tópicos

Se trata de uno de los módulos más importantes de la plataforma. Su función es conseguir extraer las palabras más relevantes asociadas a un número de tópicos dado por el usuario a partir de un *corpus* textual. Para ello, se utilizará la librería Gensim, desarrollada en Python, y, más concretamente, el algoritmo **LDA**[9]. Este asume que un tópico es una distribución probabilística sobre palabras y, a su vez, los tópicos se asocian a documentos ponderando la probabilidad con la que un documento trata ese determinado tópico o tema.

LDA es una técnica no supervisada que a partir de un *corpus* textual extrae los principales tópicos y encuentra asociaciones entre las palabras en base a cómo concurren en los textos. Por ejemplo, si "raqueta" y "tenis" tienden a aparecer mucho en los documentos de un *corpus* son candidatas a ser asociadas al mismo tópico.

Imaginemos que contamos con un *corpus* compuesto por 1000 documentos y que tenemos un conjunto de las 1000 palabras más comunes en dichos documentos. Para determinar la categoría a la que pertenece cada uno de ellos, se podría unir cada documento con cada palabra basándose en las apariciones de éstas. Estas relaciones permitirían saber de qué habla cada uno de ellos. Sin embargo, esto presenta un claro problema de escalabilidad, véase Figura 5.2.

La solución más factible es añadir una capa de **tópicos** o **temas** intermedia. En este caso, se relacionarían los documentos con los temas que mejor encajen y

los temas con las palabras, véase Figura 5.3. En la realidad, cada uno de los tópicos de la capa intermedia se correspondería con un vector de palabras y sus correspondientes pesos, por ejemplo, *Tech* sería *(0.7*AI,0.3*flying,0.2*cars,0.1*driven)* [10].

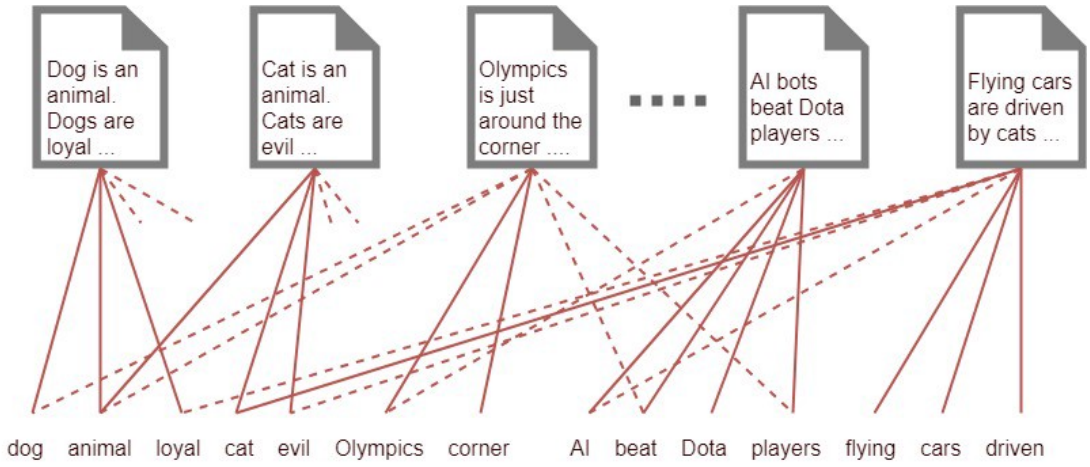

Figura 5.2: Modelado de documentos sólo con palabras
Cortesía de `https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158`

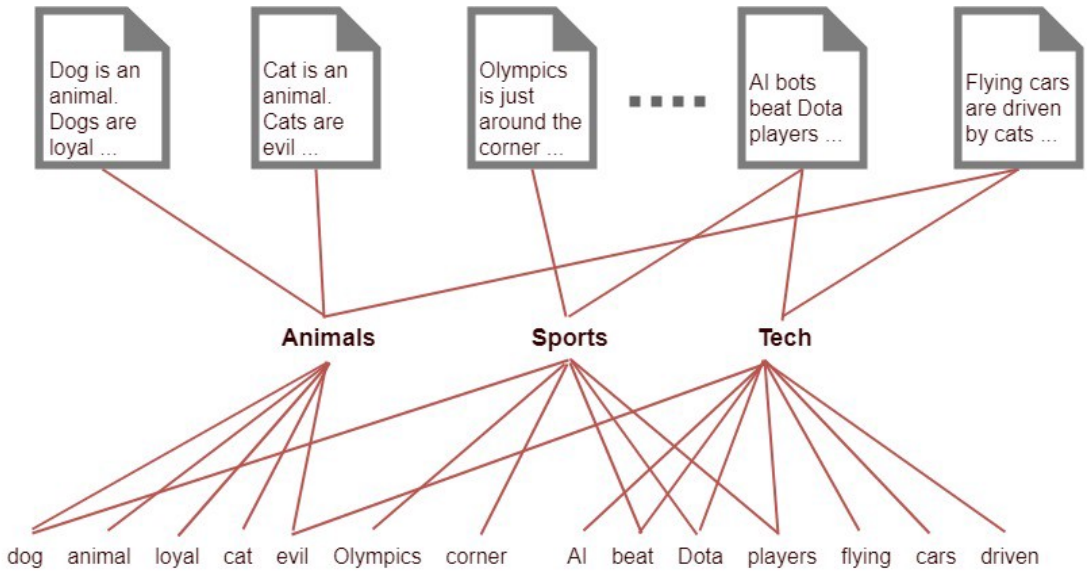

Figura 5.3: Modelado de documentos con un conjunto de tópicos
Cortesía de `https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158`

El principal objetivo es utilizar LDA para, a partir de un *corpus* que se va ingiriendo, descubrir los principales temas que se van tratando en tiempo real, es decir, usar LDA para descubrir conocimiento sobre el flujo dinámico de datos que se trata.

En primer lugar, cabe aclarar que dispone de **dos modos de funcionamiento**: uno similar al del módulo de generación de nubes de palabras dinámicas, con una cola FIFO limitada en tamaño, pero también cuenta con otro que permite recuperar sólo los textos que se encuentren dentro de una determinada franja temporal. Este último cubriría casos de uso del estilo: "Quiero sacar los principales temas de lo que se comentó sobre Trump en las dos últimas horas". Esta disyuntiva entre los dos modos se puede apreciar en el fragmento de código 5.4.

Antes de realizar el análisis propiamente dicho, se debe conseguir expresar en forma matricial los documentos textuales. Esto se hará en la llamada *preprocess_corpus* del fragmento 5.5. De este proceso se obtiene un diccionario asociando un id a cada palabra de un documento y un nuevo corpus en el que ahora cada documento se representa como una lista de tuplas en la que el primer elemento indica el id de la palabra y el segundo, el número de ocurrencias, véase fragmento 5.6.

Por último, se le pasan estos datos al modelo LDA para que devuelva los tópicos asociados, véase fragmento 5.7. Como se puede apreciar, se devuelve ya en un formato entendible por la interfaz, en este caso, HTML, pero esto se verá con más profundidad en el apartado 5.1.7.2.

```
1  def transform(self, electron):
2      if type(self.corpus) is list:
3          timestamp = int(electron.value['timestamp'])
4          if electron.value['src'] == 'twitter':
5              timestamp /= 1000 # because tweet timestamps are
       expressed in miliseconds
6          if int(self.window[0]) <= timestamp <= int(self.window[1]):
       # if the text belongs to the given range
7              self.corpus.append(electron.value.items())
8              self.send_electrons(electron)
9      else:
10         self.corpus.add(electron.value.items())
11         self.send_electrons(electron)
```

Fragmento de código 5.4: Método transform del analizador de tópicos

```
def send_electrons(self,electron):
    corpus,id2word = self.__preprocess_corpus(self.corpus)
    for n,o in zip(self.ntopics,self.output_topics):
        electron.topic = o
        electron.value = self.__make_topic_analysis(corpus,id2word,n
    )
        self.send(electron)
```

Fragmento de código 5.5: Método de envío de electrones del analizador de tópicos

```
print(corpus)
[(0, 1), (1, 1), (2, 1)]
[(0, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)]
[(2, 1), (5, 1), (7, 1), (8, 1)]
[(1, 1), (5, 2), (8, 1)]
[(3, 1), (6, 1), (7, 1)]
[(9, 1)]
```

Fragmento de código 5.6: Corpus textual en forma matricial

```
def __make_topic_analysis(self,corpus,id2word,ntopics):
    # Build LDA model
    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                id2word=id2word,
                                                num_topics=ntopics,
                                                random_state=100,
                                                update_every=1,
                                                chunksize=100,
                                                passes=10,
                                                alpha='auto',
                                                per_word_topics=True)
    return self.__render_model(lda_model,corpus,id2word,ntopics)
```

Fragmento de código 5.7: Algoritmo LDA de análisis de tópicos

5.1.5. Módulo de generación de estadísticas

Este módulo se encarga de emitir datos acerca de los textos y usuarios distintos analizados por *Indivisa*. Como se puede apreciar en el fragmento 5.8, devuelve un diccionario con el número total de usuarios y textos analizados desde que se levantó la plataforma, así como, el número de ellos que se encuentra procesando por segundo.

```
def transform(self, electron):
    self.texts_counter += 1
    # adds a user id to set if it doesn't already exist
    self.users.add(electron.value['user_id'])
    self.n_users = len(self.users)
    now = datetime.datetime.now()
    diff = (now - self.start_timestamp).seconds
    if diff >= 5:  # Refresh every 5 seconds
        self.start_timestamp = datetime.datetime.now()
        self.texts_second = (self.texts_counter-self.iter_texts) /
    diff
        self.users_second = (self.n_users-self.iter_users) / diff
        self.iter_texts = self.texts_counter
        self.iter_users = self.n_users
        electron.topic = self.output_topics[0]
        electron.value = {'texts_sec':self.texts_second,'users_sec':
    self.users_second,'total_users':len(self.users),'total_texts':
    self.texts_counter}
        return electron
```

Fragmento de código 5.8: Algoritmo de generación de estadísticas

5.1.6. Módulo de procesamiento batch

La intención de este módulo es demostrar que la plataforma puede trabajar tanto con *stream processing* como con tareas *batch*. En este caso la tarea programada es muy simple: se trata de contar el número de textos en una ventana temporal, véase fragmento 5.9. Además, ofrece la ventaja de que se pueden consultar los resultados parciales del procesamiento en tiempo real si así se desea.

```
def transform(self, electron):
    for window,topic in zip(self.windows,self.output_topics):
        timestamp = int(electron.value['timestamp'])
        if electron.value['src'] == 'twitter':
            timestamp /= 1000 # because tweet timestamps are
    expressed in miliseconds
        if  window[0] <= timestamp <= window[1]: # if the text
    belongs to the given range
            self.batch_msgs[window].append(electron.value.items())
            electron.topic = topic
            electron.value = len(self.batch_msgs[window])
            self.send(electron)
```

Fragmento de código 5.9: Algoritmo batch

5.1.7. Módulo de interfaz gráfica

Este módulo se compone de una API Python, encargada de recibir y procesar peticiones, y de una interfaz web desarrollada en *React* para la interacción con el usuario final.

5.1.7.1. API Flask

Tal y como se puede observar en la Figura 5.1, se trata de un módulo que hereda de *Catenae* y se conecta a través de las colas de mensajes de *Apache Kafka*, al igual que el resto. Además, es el encargado de recibir y procesar las peticiones que le llegan desde la interfaz web.

Por otra parte, mantiene actualizadas las colecciones de la base de datos correspondiente en Mongo DB. En este caso son dos: una para guardar la salida de cada módulo y otra que contiene el tipo de módulos que se encuentran desplegados en la plataforma en un determinado instante. Imaginemos que el usuario desea levantar un filtro en tiempo real que consuma de Twitter, esta petición llegaría a la API, se comprobaría en la tabla si existe algún módulo ya desplegado con estas características y, en caso afirmativo, no se levantaría un nuevo módulo sino que se enviaría una **llamada RPC** al ya desplegado para que empezase a filtrar también por la nueva consulta expresada por el usuario. Al llegar a cierto límite de carga de un mismo módulo, se debería levantar otro.

5.1.7.2. Interfaz web

Para la interacción con el usuario final, se ha decidido desarrollar una interfaz web en el framework *React*. A continuación, se presentan una serie de figuras en las que se muestran las principales vistas de dicha interfaz.

Requiere una especial atención la Figura 5.8, ya que para dicha vista ha sido empleada la librería de visualización de tópicos **pyLDAvis**[11]. Como se puede apreciar, consta de dos paneles. El de la izquierda representa los tópicos a modo de círculo en un plano bidimensional donde los centros se determinan computando las distancias entre tópicos y proyectándolas a varias dimensiones. Para conocer la relevancia general de cada tópico hay que fijarse en su área. Por otra parte, el panel de la derecha representa las palabras asociadas a cada tema. Lo hace además en forma de gráfico de barras horizontal, permitiendo así consultar la frecuencia de un término en el *corpus* (barra azul) y en el tópico (barra roja). Ambos paneles pueden interectuar entre sí, de manera que si se clica encima de uno de los círculos se mostrarán las palabras asociadas a él y si se clica en una de las palabras, se mostrarán a qué temas se vincula. Otro aspecto interesante radica en cómo esta librería selecciona los términos más relevantes para un determinado tópico y le deja al usuario modificarlos a través del parámtero λ. Para ello, utiliza la siguiente fórmula:

$$r(\omega, \kappa|\lambda) = \lambda * log(\phi_{\kappa\omega}) + (1 - \lambda) * log(\frac{\phi_{\kappa\omega}}{\rho_{\omega}}). \tag{5.1}$$

En ella, $\phi_{\kappa\omega}$ representa la probabilidad de que el término ω pertenezca al tema κ, mientras que ρ_{ω} representa la probabilidad del término en el *corpus*. λ puede tomar valores entre 0 y 1: si λ=1, se devuelven los términos en orden descendente para su probabilidad específica en el tópico, mientras que si λ=0, los términos se ordenan solamente por su probabilidad general.

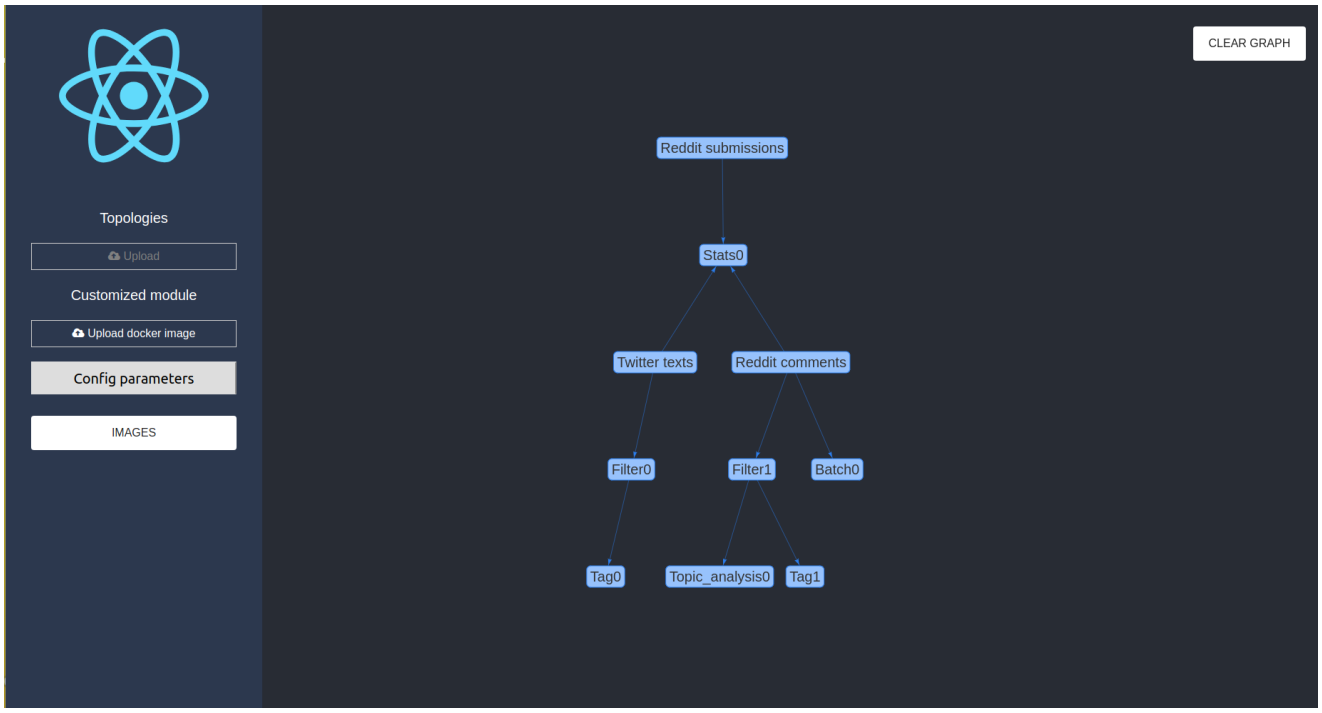

Figura 5.4: Vista principal

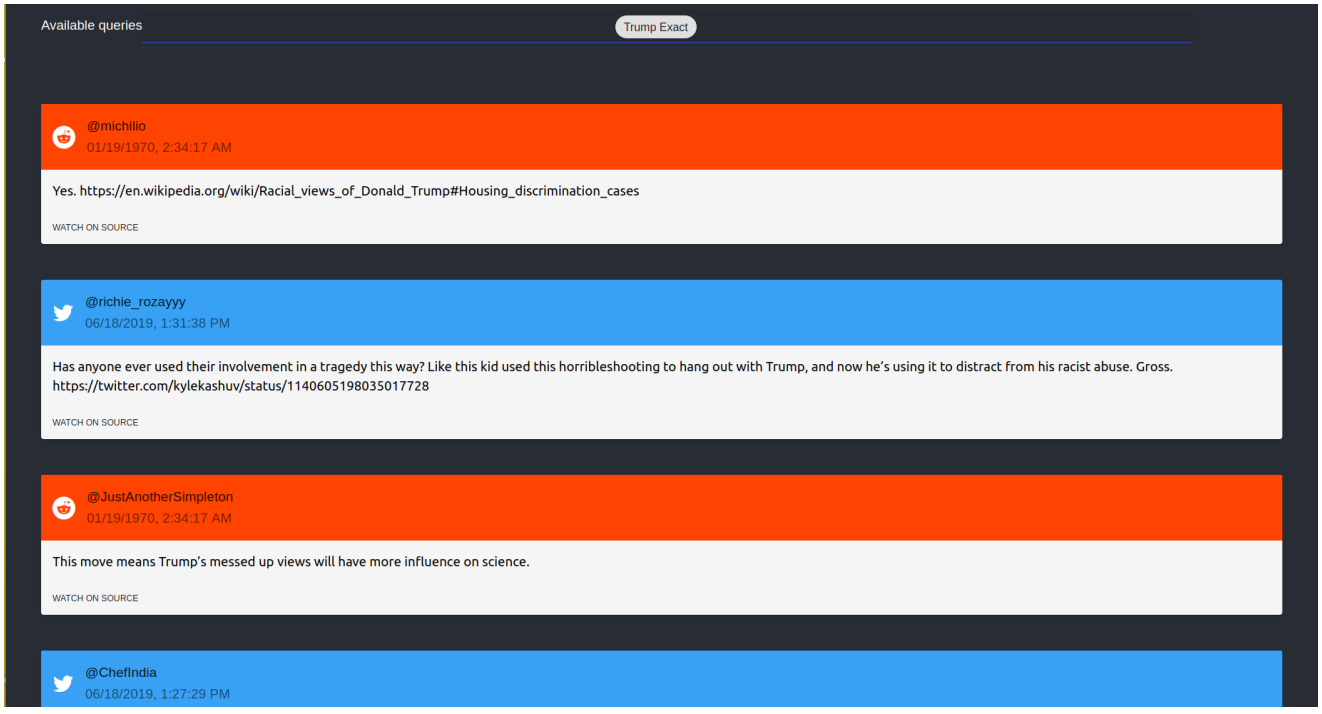

Figura 5.5: Vista filtro en tiempo real

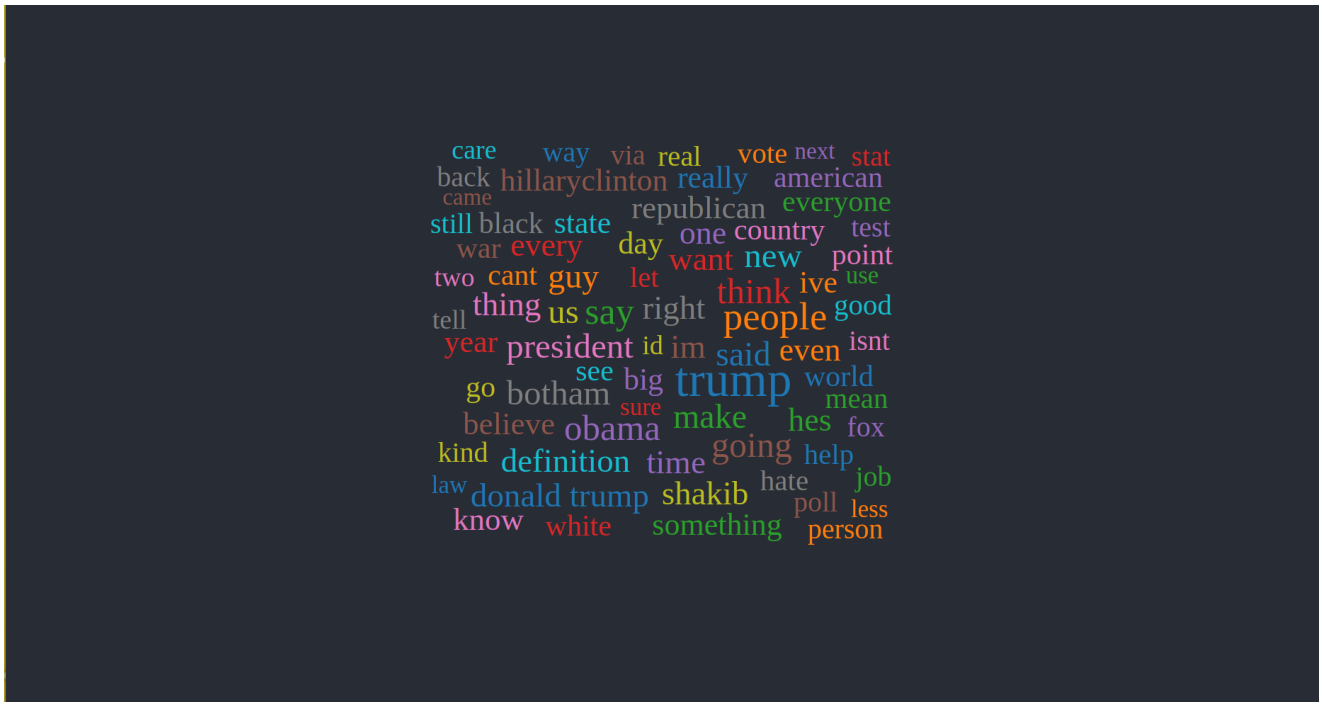

Figura 5.6: Vista tag cloud dinámico

Figura 5.7: Vista procesamiento por lotes

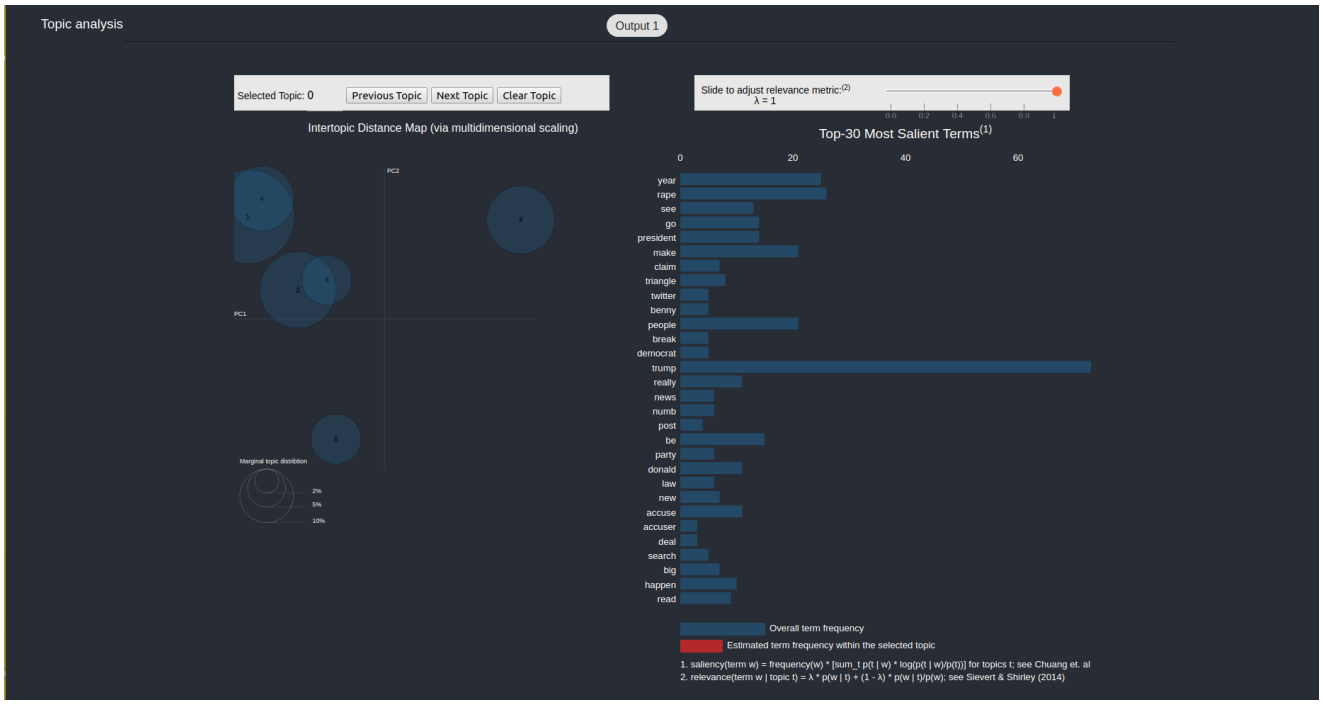

Figura 5.8: Vista analizador de tópicos

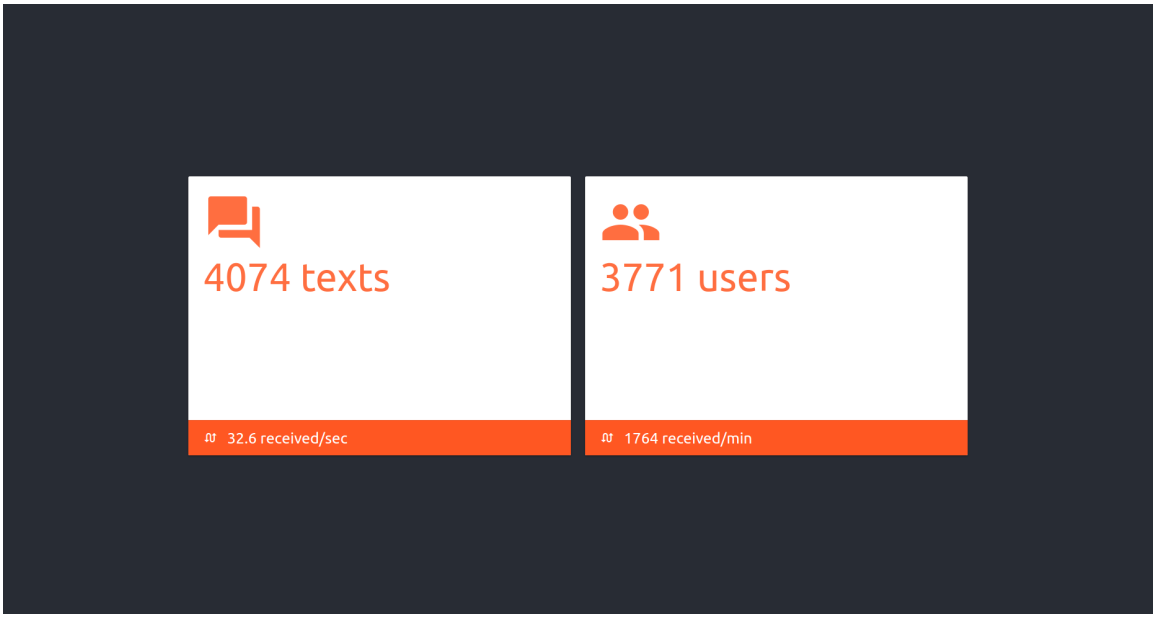

Figura 5.9: Vista generación de estadísticas

5.2. Arquitectura en clúster del sistema

Como se ha explicado en el apartado 5.1, se ha desarrollado una implementación de la plataforma en local, pero migrarla a una arquitectura distribuida en un clúster es trivial. Esto es debido a la facilidad que ofrece tener todos los componentes dentro de contenedores *Docker*.

En la Figura 5.10, se propone una posible implementación para esta arquitectura, pero no es la única. En ella, todos los elementos se encontrarían, por ejemplo, en una nube o *cloud*. La API sería la encargada de lanzar los módulos que el usuario le indique desde la interfaz dentro de un gestor de contenedores, por ejemplo, *Kubernetes*. La utilización de una herramienta de este tipo permite aprovechar las propiedades de escalabilidad y replicación que ofrecen los nodos.

En el apartado 7.2.1 se presentará en mayor detalle, como posible trabajo futuro, un despliegue en el gestor de contenedores *Kubernetes*.

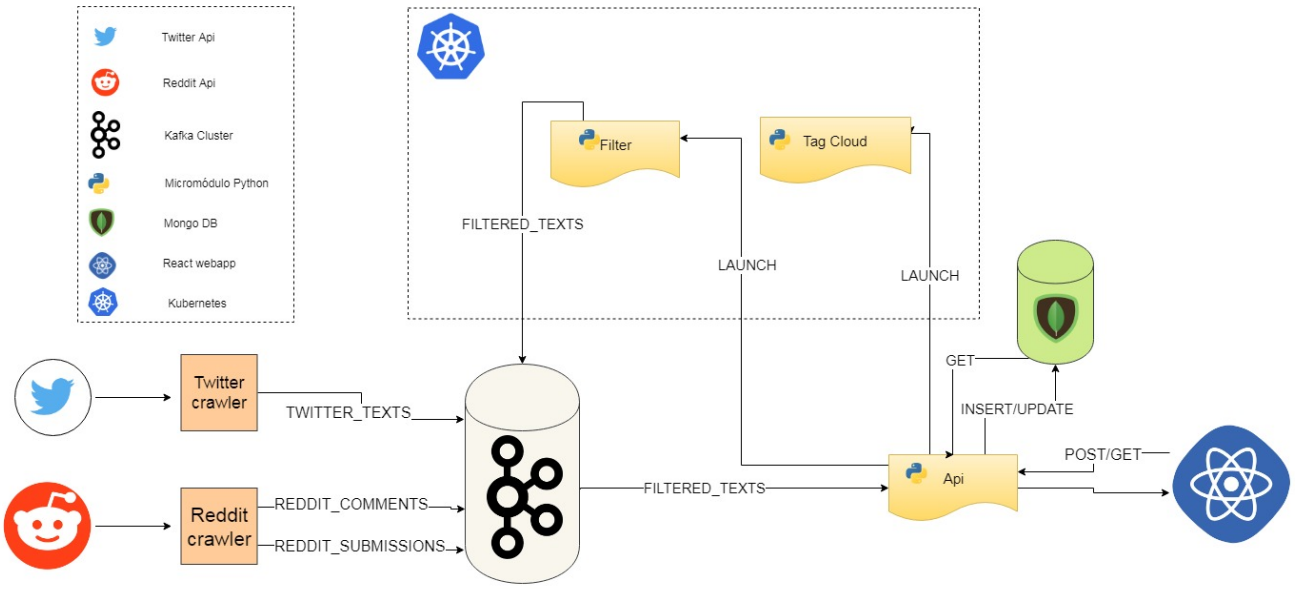

Figura 5.10: Diagrama Arquitectura del sistema en clúster

5.3. Diagramas de clases

5.3.1. Diagrama de clases de un módulo

Todos los módulos preinstalados de la plataforma comparten la misma estructura de clases que se puede ver en la Figura 5.11. De la clase *Link* de *Catenae* heredan dos métodos básicos: el *setup*, en el cual se realizan todas las inicializaciones y configuraciones pertinentes, y el *transform*, que como ya se ha visto

con anterioridad se ejecuta cada vez que se recibe un mensaje y actúa como manejador del mismo. Por otra parte, cabe señalar que todos los módulos de *Indivisa* proporcionan dos métodos para configurar sus suscripciones en tiempo real, *add_topic* y *remove_topic*, los cuales se invocan desde la API mediante llamadas RPC. Como se puede apreciar, reciben por parámetro el *topic* a añadir o eliminar, el contexto que identifica si son el destinatario del mensaje y todo lo necesario para actualizar el contenido de la colección en Mongo DB que indica el tipo de contenedor que se encuentra desplegado. Además, pueden implementar otros métodos a mayores.

Esta estructura tan simple permite que un usuario con conocimientos de programación pueda definir sus propios módulos y conectarlos a una topología de forma transparente. Se puede ver un ejemplo en el apartado B.4 del manual de usuario.

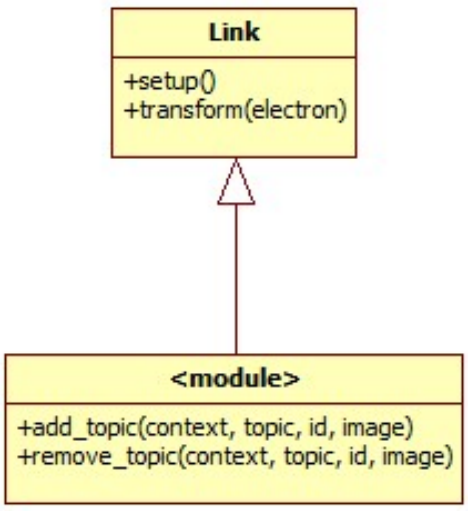

Figura 5.11: Diagrama de clases de un módulo de *Indivisa*

5.3.2. Diagrama de clases de la API

Como ya se había comentado con anterioridad, la API funciona como un módulo más de la plataforma y es la encargada de recibir y procesar las peticiones que envía el usuario desde la interfaz web. La parte izquierda de la Figura 5.12 hace referencia a esta primera funcionalidad. En este caso, no se implementan métodos para añadir o eliminar suscripciones en tiempo real, ya que no se consideró necesario para este módulo concreto.

Mientras que en la parte derecha, se puede ver la API propiamente dicha. Para su desarrollo, se ha decidido implementarla utilizando el framework Flask. Un concepto muy importante a tener en cuenta son las clases que heredan de

Resource. Éstas se añaden a la API Flask junto con una URL y permiten definir una serie de *endpoints* en su interior. En este caso concreto, se han creado dos clases: *GenericService*, la cual implementa el borrado de los contenedores desplegados, la recuperacón de los resultados de un módulo y el despliegue de un módulo preinstalado o de una topología y *CustomResource*, que implementa la carga en el servidor de una imagen *Docker* aportada por el usuario en formato .tar.

Sin embargo, un servidor web tradicional no es capaz de ejecutar aplicaciones Python. Por ello, surgió la interfaz estándar WSGI que los módulos podían implementar. De ahí la necesidad de la parte superior del diagrama: en este caso se seleccionó Gunicorn[1] como medio para lanzar la aplicación, ya que implementa el estándar de servidor PEP3333 WSGI[12] y es capaz de llamar a una aplicación web compatible con WSGI, ya sea Django, Flask u otro, véase la Figura 5.13.

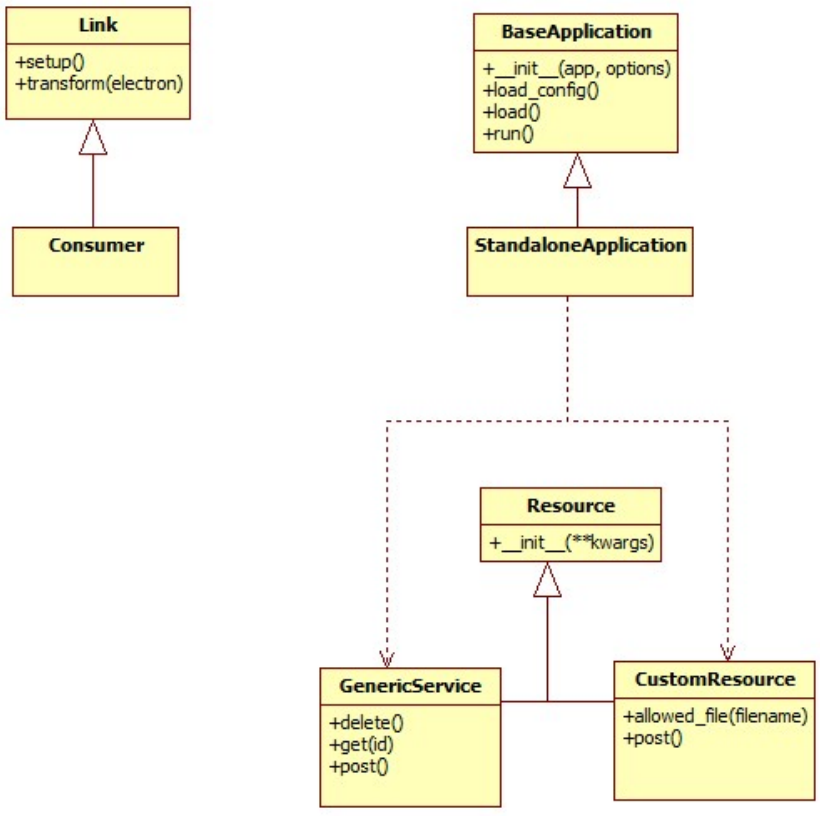

Figura 5.12: Diagrama de clases de la API

[1]https://gunicorn.org/

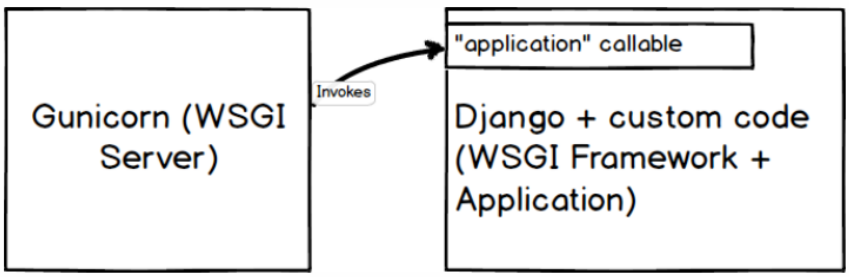

Figura 5.13: Gunicorn and Django implementation
Cortesía de `https://www.fullstackpython.com/green-unicorn-gunicorn.html`

5.4. Diagramas de interacción

A continuación, se presentan una serie de diagramas que tratan de representar la interacción del usuario con el sistema en un nivel de abstracción elevado, debido a la complejidad que supondría la elaboración de un diagrama de secuencia detallado para todas las opciones de la plataforma.

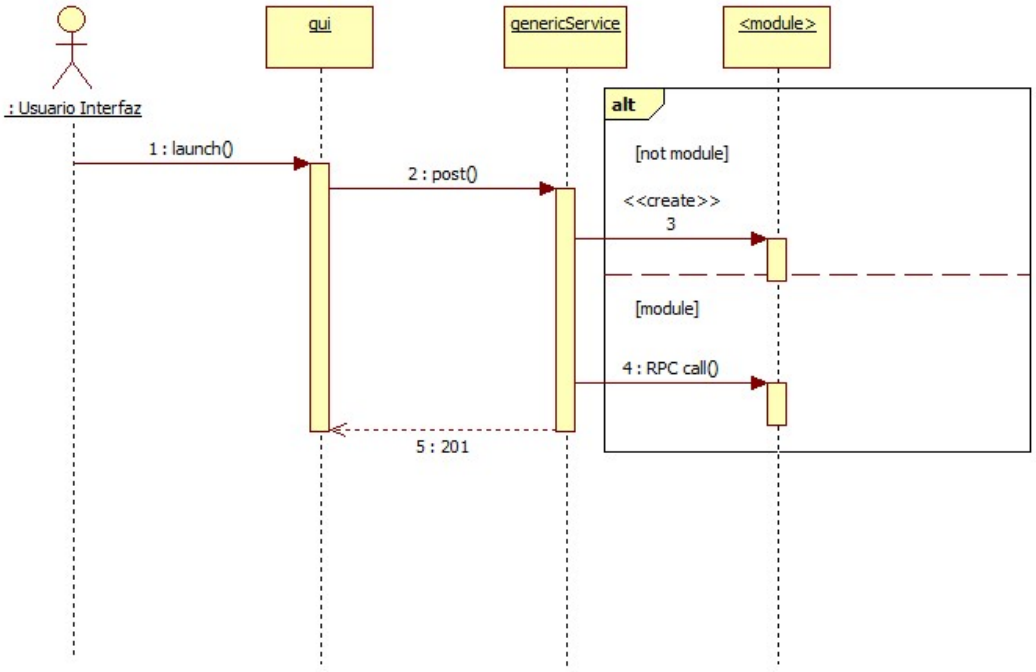

Figura 5.14: Diagrama de interacción de alto nivel para lanzar un módulo (preinstalado o subido por el usuario)

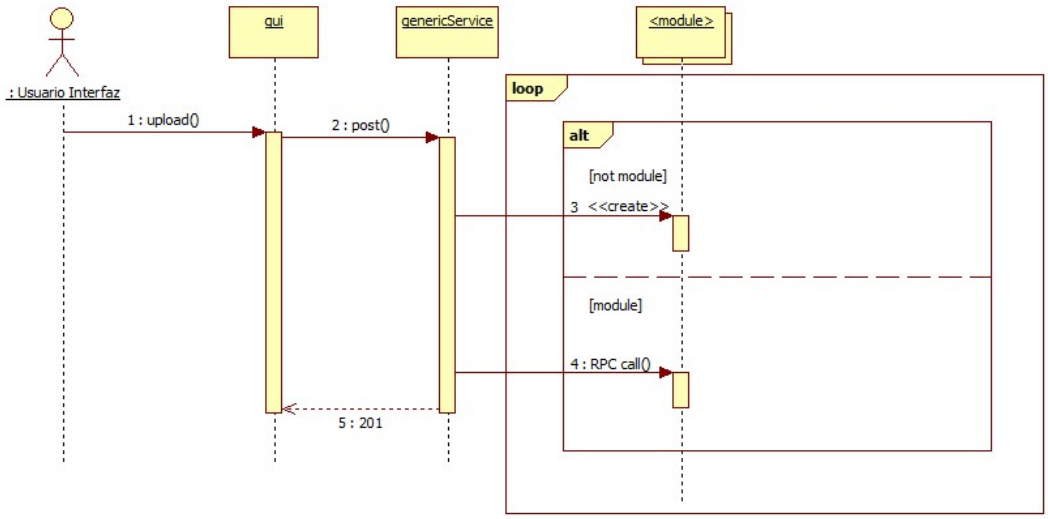

Figura 5.15: Diagrama de interacción de alto nivel para lanzar una topología de consumo

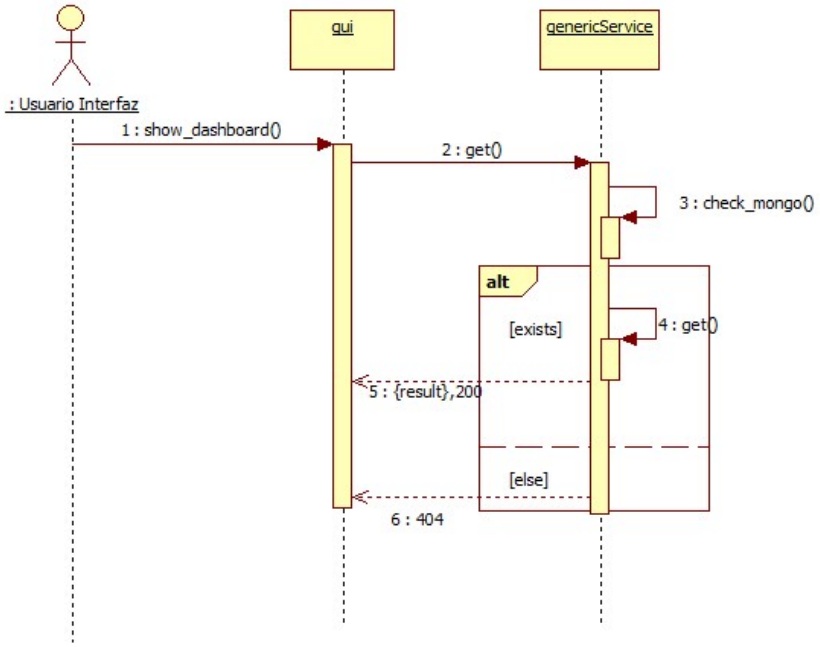

Figura 5.16: Diagrama de interacción de alto nivel para recuperar el resultado de un módulo

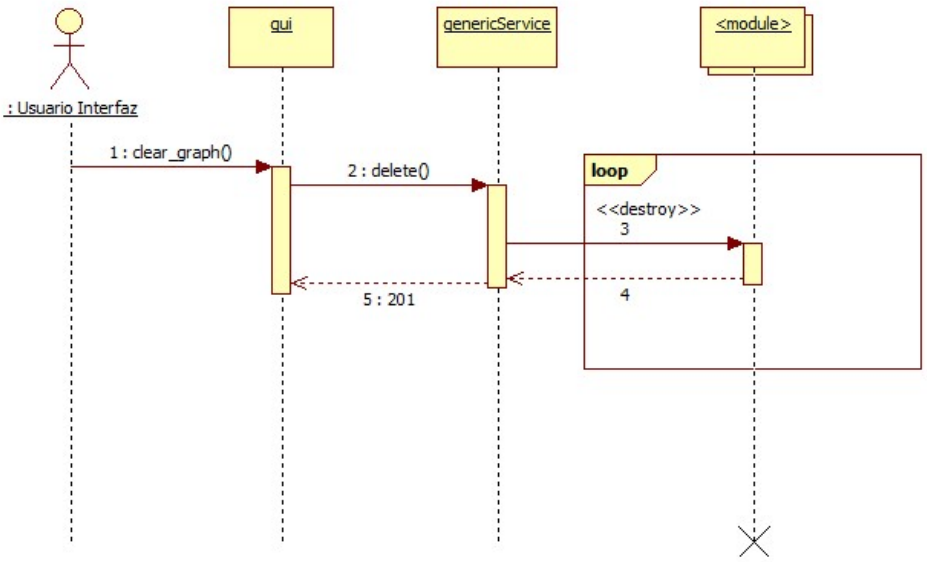

Figura 5.17: Diagrama de interacción de alto nivel para borrar los módulos desplegados

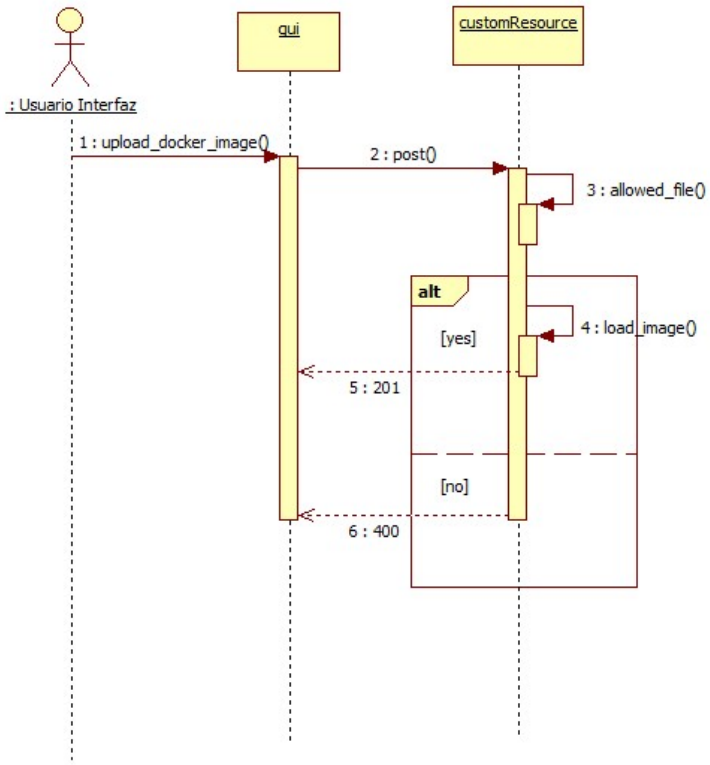

Figura 5.18: Diagrama de interacción de alto nivel para subir una imagen creada por un usuario

Capítulo 6

Pruebas y validación

A continuación, se presentan una serie de pruebas diseñadas para comprobar que los requisitos descritos en el capítulo 3 de la presente memoria han sido implementados de manera satisfactoria.

ID	**PR-01**
Requisitos involucrados	RF-01
Descripción	Lanzar una imagen de un módulo de los preinstalados. También probar con otra imagen que no sea de las "esperadas" por la plataforma, ya sea un *crawler* o un nuevo módulo subido por un usuario.
Resultado esperado	Ver que se lanzan y se conectan correctamente
Estado	Superada

Cuadro 6.1: PR-01

ID	**PR-02**
Requisitos involucrados	RF-02
Descripción	Lanzar una topología de consumo a partir de un fichero .yaml.
Resultado esperado	Ser capaz de comprobar que los módulos desplegados en la plataforma cumplen con las relaciones descritas en el fichero pasado.
Estado	Superada

Cuadro 6.2: PR-02

ID	**PR-03**
Requisitos involucrados	RF-03
Descripción	Realizar una llamada a un módulo preinstalado y ya desplegado.
Resultado esperado	Ver que no se levanta un nuevo contenedor y que su comportamiento se modifica.
Estado	Superada

Cuadro 6.3: PR-03

ID	**PR-04**
Requisitos involucrados	RF-04
Descripción	Realizar una llamada a un módulo desplegado añadiendo y posteriormente eliminando un flujo existente en la topología.
Resultado esperado	Comprobar que cuando se añade empieza a consumir de ese flujo y que cuando se elimina, para.
Estado	Superada

Cuadro 6.4: PR-04

ID	**PR-05**
Requisitos involucrados	RF-05
Descripción	Probar a eliminar un módulo de los preinstalados y comprobar que desaparece, así como sus flujos de salida.
Resultado esperado	Comprobar que desaparece de la topología y que sus módulos suscriptores dejan de recibir información.
Estado	Superada

Cuadro 6.5: PR-05

ID	**PR-06**
Requisitos involucrados	RF-06
Descripción	Probar a realizar 5 consultas exactas en el módulo de filtrado preinstalado que ofrece *Indivisa*.
Resultado esperado	Comprobar que los mensajes devueltos contienen exactamente la consulta expresada por el usuario.
Estado	Superada

Cuadro 6.6: PR-06

ID	**PR-07**
Requisitos involucrados	RF-07
Descripción	Probar a realizar 5 consultas inexactas en el módulo de filtrado preinstalado que ofrece *Indivisa*.
Resultado esperado	Comprobar que los mensajes devueltos contienen las palabras clave de la consulta en cualquier orden.
Estado	Superada

Cuadro 6.7: PR-07

ID	**PR-08**
Requisitos involucrados	RF-08
Descripción	Probar acceder a publicaciones de las dos fuentes preinstalas: *Twitter* y *Reddit*.
Resultado esperado	Se puede ver el mensaje en la fuente original.
Estado	Superada

Cuadro 6.8: PR-08

ID	**PR-09**
Requisitos involucrados	RF-09
Descripción	Lanzar un generador de *tag clouds* conectado a un filtro.
Resultado esperado	Ver que se actualiza en tiempo real y que los porcentajes que genera tienen sentido.
Estado	Superada

Cuadro 6.9: PR-09

ID	**PR-10**
Requisitos involucrados	RF-10
Descripción	Lanzar dos módulos de análisis de tópicos, cada uno con su modo.
Resultado esperado	Comprobar que se realiza correctamente el despliegue.
Estado	Superada

Cuadro 6.10: PR-10

ID	**PR-11**
Requisitos involucrados	RF-11
Descripción	Probar a desplegar un módulo de análisis de tópicos con un número de tópicos inicial. A continuación, a ese mismo módulo, añadirle un nuevo número de tópicos mediante una llamada remota.
Resultado esperado	Comprobar que en ambos casos el resultado sea el esperado y que ambos números de tópios convivan de manera independiente sin mezclarse.
Estado	Superada

Cuadro 6.11: PR-11

ID	**PR-12**
Requisitos involucrados	RF-12
Descripción	Probar a solicitar los datos de tres análisis diferentes de un mismo módulo.
Resultado esperado	Comprobar que son correctos e independientes entre sí.
Estado	Superada

Cuadro 6.12: PR-12

ID	**PR-13**
Requisitos involucrados	RF-13
Descripción	Probar a desplegar un módulo de procesamiento por lotes con una franja temporal ya especificada. A continuación, a ese mismo módulo, añadirle una nueva ventana mediante una llamada remota.
Resultado esperado	Comprobar que en ambos casos el resultado sea el esperado y que ambos cómputos se realicen de manera independiente.
Estado	Superada

Cuadro 6.13: PR-13

ID	**PR-14**
Requisitos involucrados	RF-14
Descripción	Probar a repetir el procesamiento por lotes en la misma franja temporal un par de veces.
Resultado esperado	Comprobar que el resultado es el mismo.
Estado	Superada

Cuadro 6.14: PR-14

ID	**PR-15**
Requisitos involucrados	RF-15, RF-16
Descripción	Hacer dos solicitudes separadas en el tiempo al módulo de generación de estadísticas, una a continuación de la otra.
Resultado esperado	Comprobar que el número de textos y usuarios totales crece.
Estado	Superada

Cuadro 6.15: PR-15

ID	**PR-16**
Requisitos involucrados	RF-17, RF-18
Descripción	Hacer dos solicitudes separadas en el tiempo al módulo de generación de estadísticas, una a continuación de la otra.
Resultado esperado	Comprobar que el número de usuarios y textos por segundo se mantiene más o menos constante.
Estado	Superada

Cuadro 6.16: PR-16

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

El objetivo de este proyecto fue el desarrollo de una plataforma para el procesamiento masivo de datos web en tiempo real a través del despliegue de topologías de consumo modulares y configurables. Se trata de un sistema que surge de la librería *Catenae*, desarrollada por investigadores del CiTIUS. Hoy en día, el *Big Data* es uno de los principales activos de las organizaciones y les permite dar respuesta a preguntas que antes ni siquiera se planteaban. Estas tecnologías trabajan con grandes volúmenes de datos y tratan de extraer información de utilidad de ellos.

Indivisa se enmarca dentro de este ámbito de las TI. Su principal objetivo es el de recuperar y procesar ingentes cantidades de datos web para presentárselos al usuario final de manera entendible y útil para su actividad. Para ello, se han utilizado tecnologías que favorecen tanto la portabilidad como la escalabilidad. Entre ellas, se pueden destacar *Docker* y *Apache Kafka*, las cuales están siendo muy utilizadas hoy en día. Los nodos de la plataforma se encapsulan en contenedores para, así, favorecer la portabilidad y un posible futuro despliegue en un clúster. Mientras que las comunicaciones se han implementado utilizando colas de mensajes de *Kafka*, un sistema *Big Data* para tiempo real que utiliza el modelo publica-suscribe.

Por otra parte, se ha querido que la aplicación estuviese dirigida tanto a usuarios con conocimientos informáticos como a usuarios estándar. Para los primeros, se ofrece la posibilidad de programar sus propios módulos, subirlos y conectarlos a la plataforma de manera totalmente transparente, haciendo, así, de *Indivisa* un sistema mucho menos rígido y con muchas más posibilidades de uso.

En cuanto a la metodología utilizada, considero que la elegida fue la correcta ya que, al ser un único participante, *Kanban* me dio la flexibilidad necesaria para poder reordenar las tareas en función de las prioridades y requisitos cambiantes del proyecto. Además, me permitió tener siempre una visión global de lo que tenía

hecho y de lo que faltaba por hacer, pudiendo así orientar mis pasos a cumplir con los criterios mínimos de aceptación del proyecto lo antes posible.

En líneas generales, considero que el proyecto se implementó de manera correcta, puesto que a pesar de que se materializó algún riesgo, su impacto no fue grave y se respondió adecuadamente. Además, opino que todavía queda trabajo por hacer y que se puede ampliar y mejorar más la plataforma, tal y como explicaré en el apartado siguiente.

7.2. Trabajo futuro

A continuación, se expondrán una serie de posibles mejoras y ampliaciones para la plataforma *Indivisa*.

7.2.1. Despliegue en Kubernetes

Kubernetes[1] es una plataforma para la automatización del despliegue, escalado y administración de contenedores en un entorno de producción. Sus principios son los que permiten a Google ejecutar miles de millones de contenedores.

Docker es una herramienta para la construcción, distribución y ejecución de contenedores. Mientras que *Kubernetes* es un sistema de orquestación de contenedores, *Docker* u otros, pensado para coordinar clústers de nodos en producción de manera eficiente[13].

Trabaja alrededor del concepto de **Pods**, los cuales son unidades de planificación *Kubernetes* que pueden contener uno o más contenedores y se distribuyen entre los nodos para conseguir una alta disponibilidad. Los contenedores de un mismo Pod están siempre en la misma localización y comparten contexto, por tanto, se pueden comunicar mediante llamadas a **localhost**. Mientras que en el caso de contenedores en distintos Pods, deben conocer la dirección IP del Pod para poder comunicarse, para ello *Kubernetes* proporciona un servicio de DNS, véase Figura 7.1. Por otra parte, en el fragmento de código 7.1, se puede ver un ejemplo de Pod en el que se describe un servicio mysql y se expone su puerto 3306. Otra utilidad importante a tener en cuenta es el comando **kubectl**[2], el cual permite realizar operaciones contra el clúster de *Kubernetes*.

Hoy en día, existen diferentes opciones en el mercado para poner una aplicación en producción en *Kubernetes* como, por ejemplo, Amazon EKS o IBM Cloud. Esta podría ser una de las mejoras a aplicar a *Indivisa*, ya que, como se ha explicado con anterioridad, su diseño favorece la escalabilidad y la implementación en un entorno distribuido. Una posible arquitectura es la propuesta en el apartado 5.2, pero no es la única solución posible.

[1] https://kubernetes.io/es/

[2] https://kubernetes.io/docs/reference/kubectl/overview/

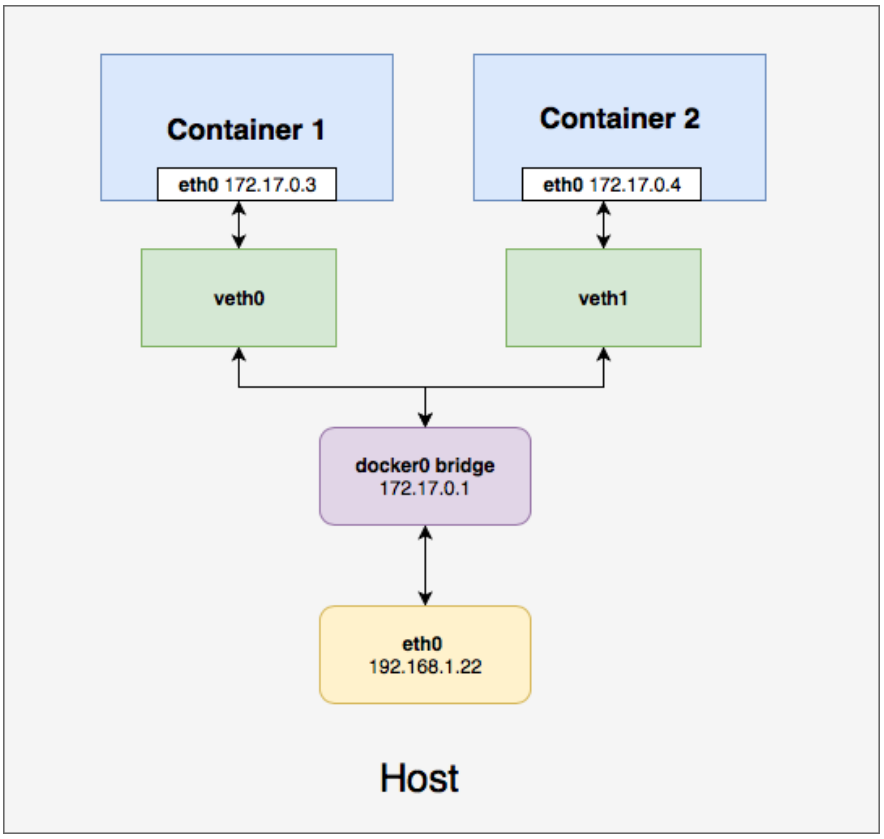

Figura 7.1: Kubernetes: Comunicación entre Pods
Cortesía de `https://supergiant.io/blog/kubernetes-networking-explained-introduction/`

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-svc
spec:
  ports:
  - name: db-port
    protocol: "TCP"
    port: 3306
    targetPort: 3306
  selector:
    app: mysql
```

Fragmento de código 7.1: Pod example
Cortesía de `https://stackoverflow.com/questions/53941391/communicate-between-pods-in-kubernetes`

7.2.2. Otras posibles mejoras

Aparte del mencionado despliegue en un gestor de contenedores, existen otras mejoras susceptibles de ser aplicadas en la plataforma. Entre ellas destacan:

- Ampliar la oferta de módulos preinstalados con la inclusión de otros como, por ejemplo, un analizador de sentimientos o un módulo encargado de hacer un **PageRank** de páginas web, igual que lo hace Google.
- Mejorar el módulo de generación de nubes de palabras dinámicas y sustituirlo por un método de resumen automático más sofisticado.
- Implementar que, llegado un límite de carga de un determinado módulo, por ejemplo un número máximo de consultas en un filtro, se despliegue uno nuevo.
- Hacer efectivo el poder limitar por parte del usuario los recursos, tanto de porcentaje de cpu como de memoria, que posee un determinado módulo.

Apéndice A

Manual técnico de despliegue

A.1. Requisitos previos

Para la correcta ejecución de la aplicación, se debe disponer de una máquina x86-64 con una distribución GNU-Linux y con conexión a la red. Antes de comenzar, es necesario instalar *Docker*[1].

A.2. Arranque de contenedores

La aplicación *Indivisa* incluye todos los archivos necesarios para su correcta utilización y ejecución. Para su instalación, solo es necesario descomprimir el archivo incluido en el CD que acompaña a esta memoria y ejecutar el script **launch.sh**. Este contiene la siguiente secuencia de instrucciones:

- Descarga de los repositorios de Docker Hub[2] la imagen correspondiente de *Catenae* y de los *crawlers*.
- Ejecuta un docker-compose en el que se levanta la estructura correspondiente a los *crawlers*, un contenedor que contiene Mongo DB y otro que contiene *Kafka*.
- Genera las imágenes correspondientes a los módulos preinstalados de la plataforma.
- Levanta la API y la conecta con Mongo DB y *Kafka*.
- Levanta la aplicación web y la pone a escuchar en el puerto 8080.

Una vez hecho esto, podríamos acceder al portal web a través de la siguiente dirección desde un navegador en el anfitrión: `http://localhost:8000`.

[1]https://www.docker.com/
[2]https://hub.docker.com/

Apéndice B

Manual de usuario

A través de la URL `http://localhost:8000`, si la aplicación se está ejecutando en local, se puede acceder a la interfaz web.

B.1. Inicio de la aplicación

Al arrancar la aplicación, se muestra la pantalla principal con sólo los nodos correspondientes a las fuentes de datos, véase Figura B.1. En ella, el usuario puede seleccionar varias opciones: cargar una topología de consumo desde fichero, subir una imagen en formato .tar o lanzar un módulo preinstalado individual.

Si se opta por la primera opción, se cargará un grafo con las relaciones descritas en el fichero, véase Figura B.2. Mientras que si se decide lanzar un módulo individual, se desplegará un *popup* en el que poder configurar las opciones del mismo, en la Figura B.3 se puede ver un ejemplo para el caso de un filtro en tiempo real.

Por otra parte, el botón "Config parameters" se encuentra deshabilitado por defecto, ya que da acceso a un *popup* similar al anterior, pero para el caso de un módulo creado por el usuario. Por tanto, es necesario que previamente se haya subido una imagen, en formato .tar, en la opción "Upload docker image".

Por último, la opción "Clear graph" permitiría borrar el grafo actual y volver a la situación inicial, reestableciendo las opciones por defecto de la plataforma.

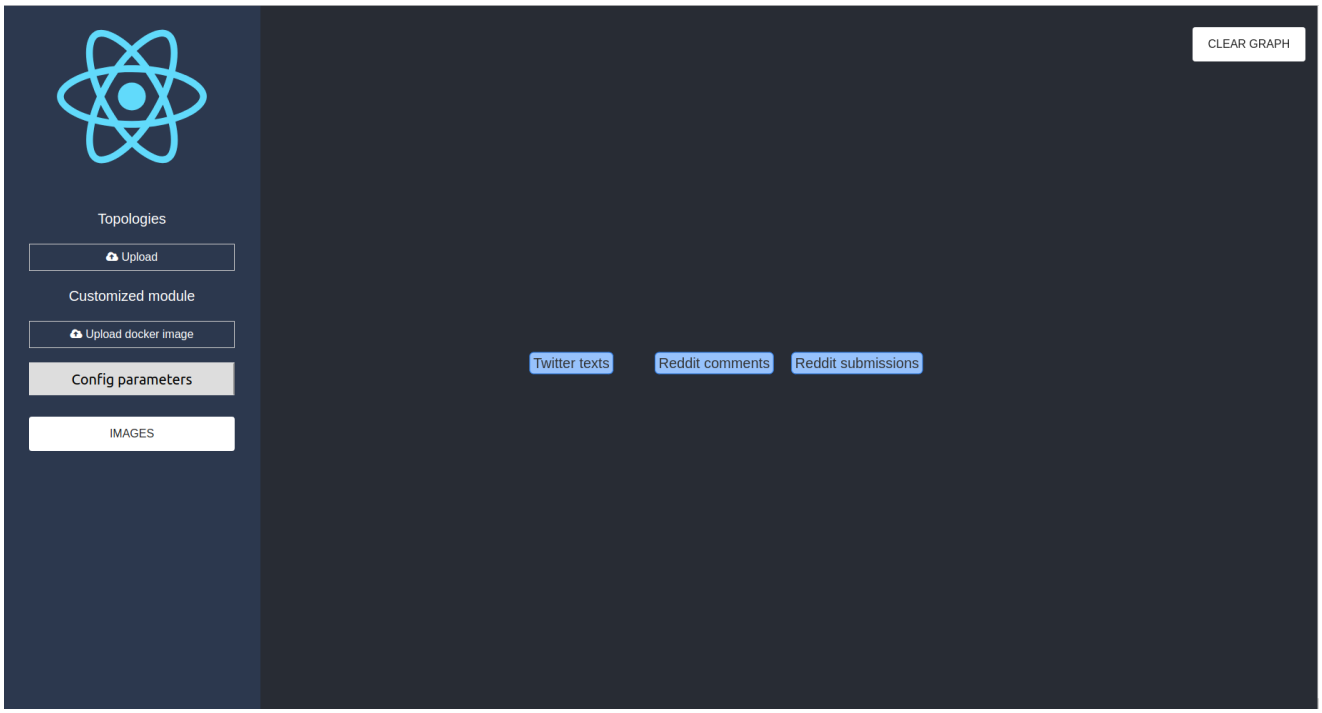

Figura B.1: Pantalla principal de la aplicación web

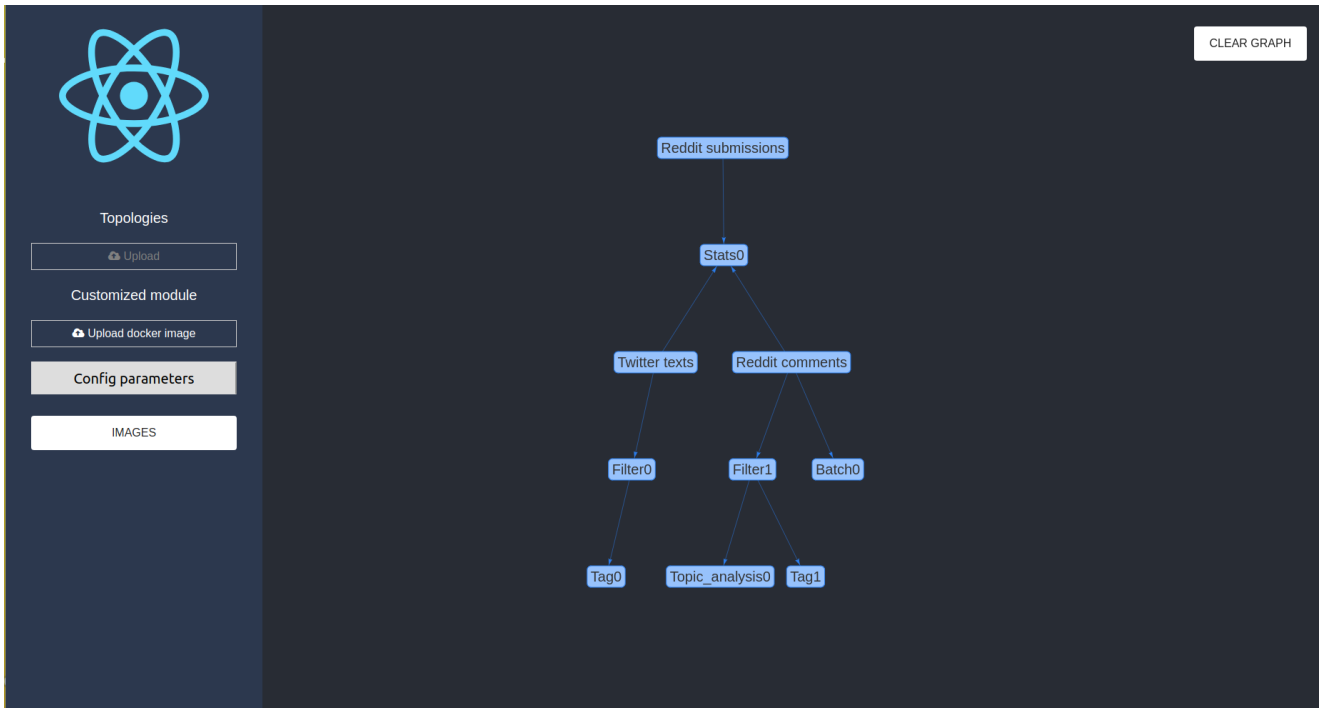

Figura B.2: Topología de consumo cargada desde fichero

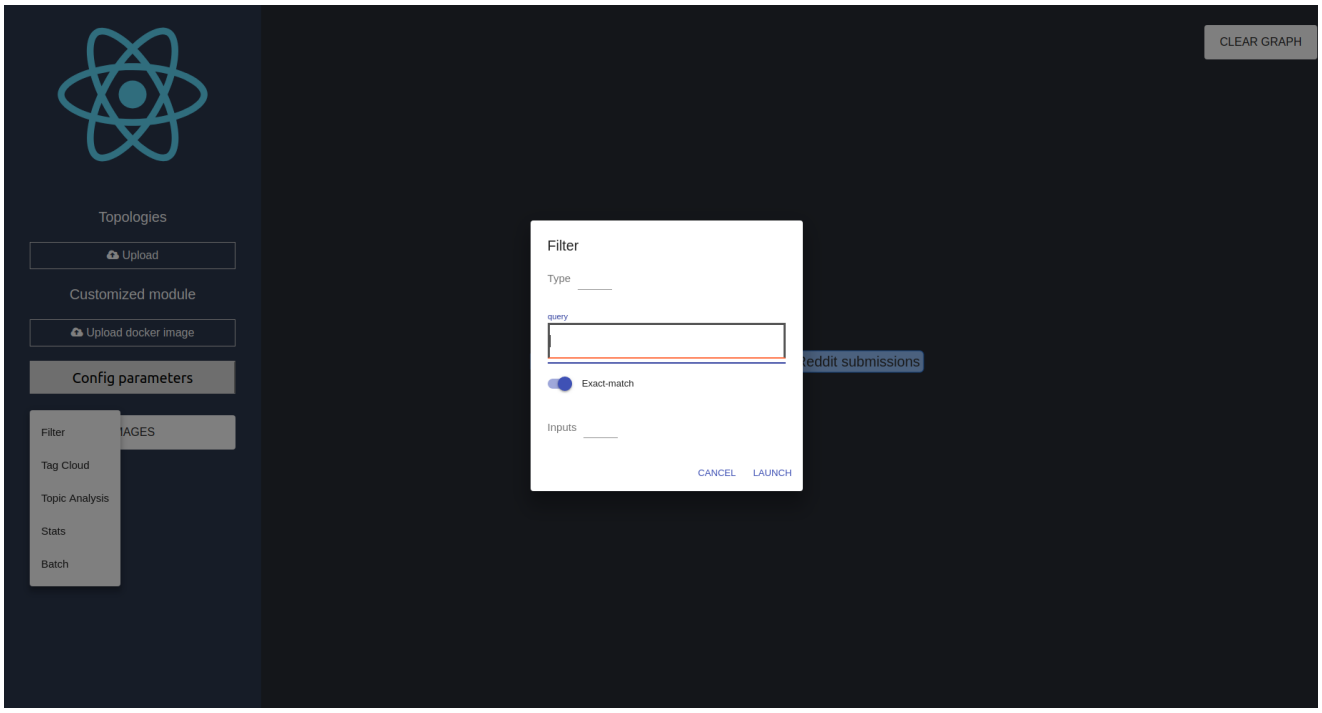

Figura B.3: Opciones de configuración de un filtro en tiempo real

B.2. Configuración de la topología

Una vez que se tiene una topología de consumo definida, ésta se puede redefinir en tiempo real. Si se hace click en uno de los nodos de la misma, se desplegará un menú como el de la Figura B.4, en el cual se podrán redefinir relaciones que se actualizarán al instante.

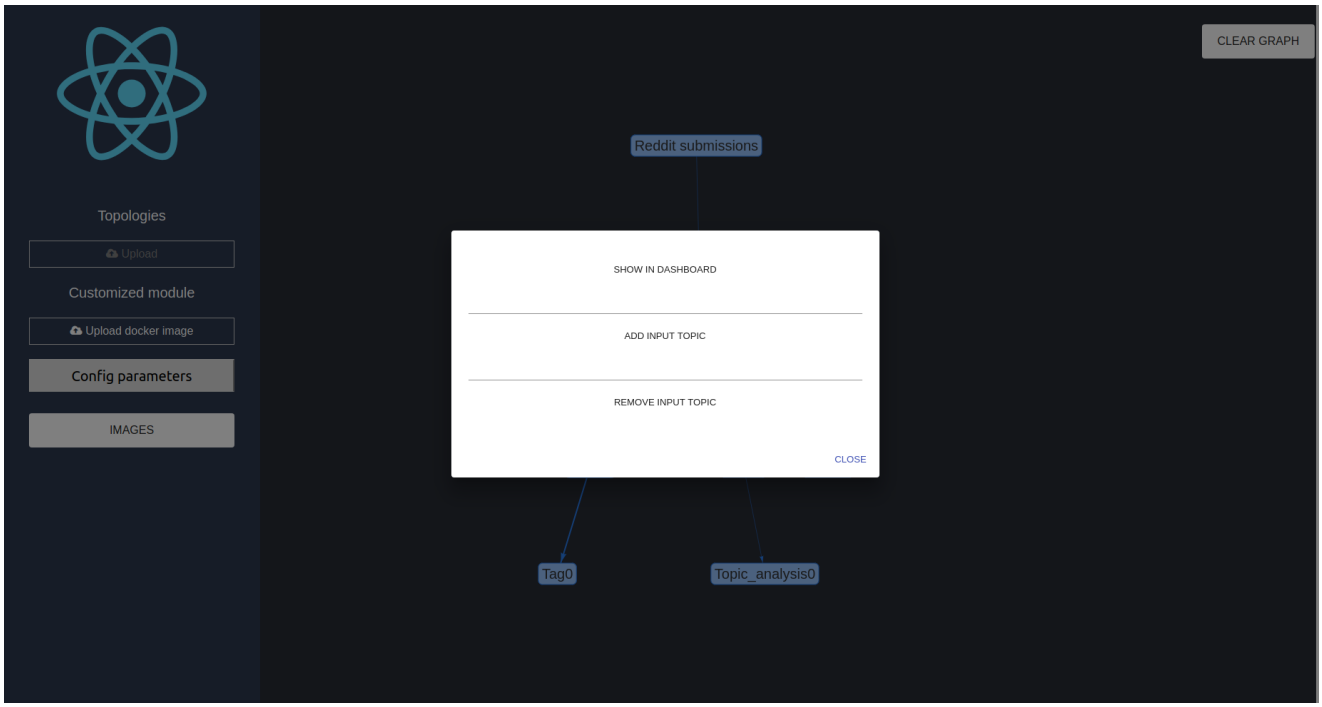

Figura B.4: Opciones de modificación de la topología en tiempo real

B.3. Dashboards de los módulos

En el panel previo, se puede seleccionar la opción de mostrar en un *dashboard* la salida de un módulo. En el caso de los módulos preinstalados de la plataforma, existen una serie de vistas que se expondrán y explicarán a continuación, mientras que para el caso de un módulo subido por un usuario se ha optado por mostrar la salida como un JSON formateado, ya que poder adaptar la vista también excedería en mucho la complejidad del proyecto.

En la Figura B.5, se puede observar un filtro en tiempo real recuperando textos que contengan la cadena **"Trump"** de manera exacta. Cabe destacar la posibilidad de elegir entre varias consultas, ya que un mismo módulo puede estar realizando más de una búsqueda al mismo tiempo.

A la salida de dicho filtro, se conectan tanto un generador de nubes de palabras dinámico, Figura B.7, como un analizador de tópicos, Figura B.6. Este último es el más complejo y requiere de una explicación más exhaustiva: consta de dos paneles, en el de la izquierda se representan los tópicos en un plano bidimensional, mientras que, en el de la derecha, se muestran las palabras asociadas a cada uno de ellos. Ambos pueden interactuar entre sí, de manera que si se clica en uno de los círculos se mostrarán los términos asociadas a ese tema y si se clica en una de las palabras, se mostrarán los tópicos principales con los que se relaciona. Por otra parte, las palabras se representan como un diagrama de barras horizontal, ya que

esto permite ver su frecuencia de aparición en el *corpus* (barra azul) y en el tema (barra roja). Por último, se debe prestar atención al *slide* que permite ajustar el valor del parámetro λ, este es un factor de peso definido por los creadores de la librería gráfica *pyLDAvis*, el cual permite ajustar el parámetro con el que se seleccionan los 30 términos más relevantes[11].

En cuanto al módulo de procesamiento por lotes, Figura B.8, permite consultar el número de textos recibidos en una franja temporal. Se puede esperar a que finalice el trabajo y sólo recuperar el valor final, o también se ofrece la posibilidad de consultar los resultados parciales en tiempo real.

Por último, en la Figura B.9 se puede observar el módulo de generación de estadísticas. En él se ofrecen datos del número de textos y usuarios distintos procesados por la plataforma desde que se levantó y de la velocidad a la que lo está haciendo.

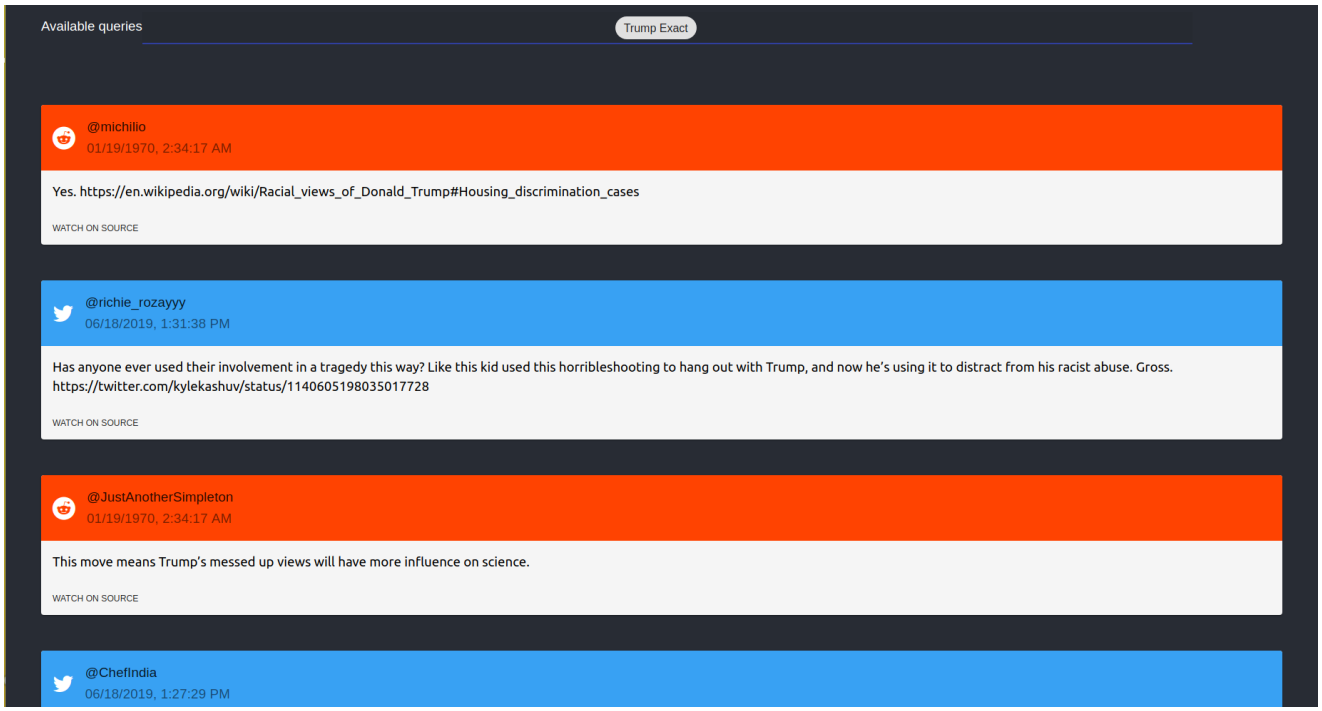

Figura B.5: Dashboard módulo filtro en tiempo real

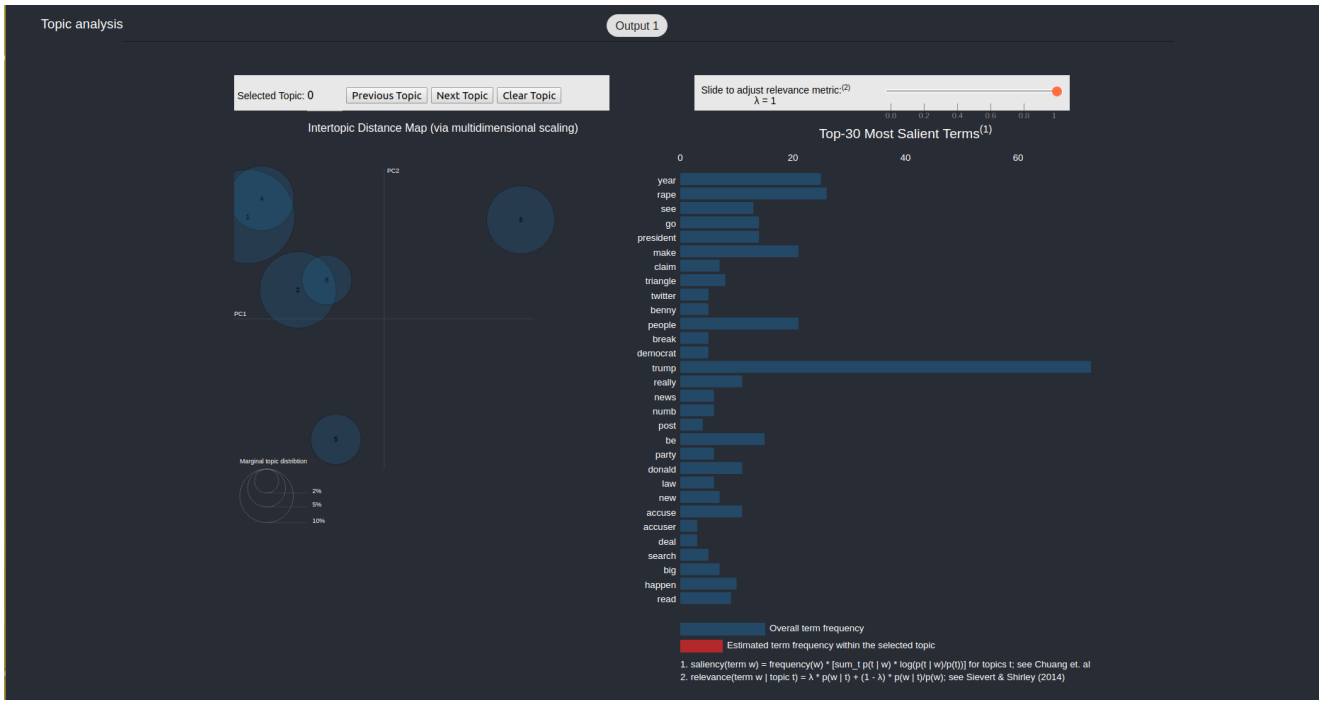

Figura B.6: Dashboard módulo analizador de tópicos

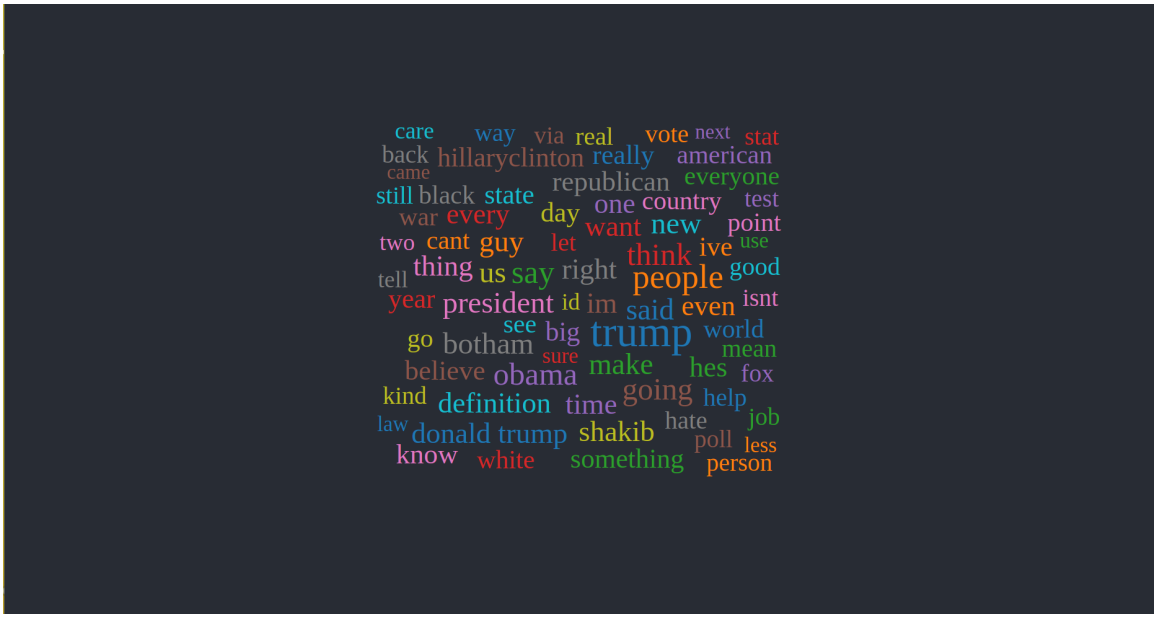

Figura B.7: Dashboard módulo generador nubes de palabras dinámicas

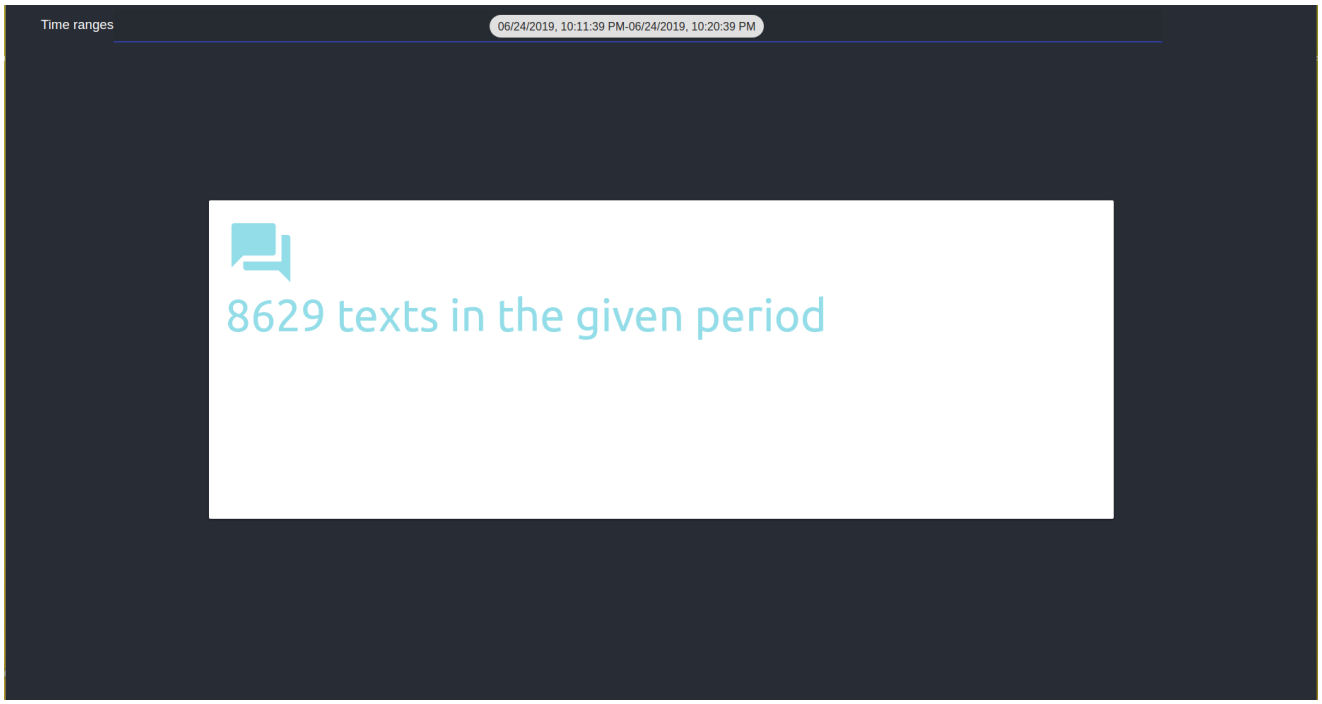

Figura B.8: Dashboard módulo de procesamiento por lotes

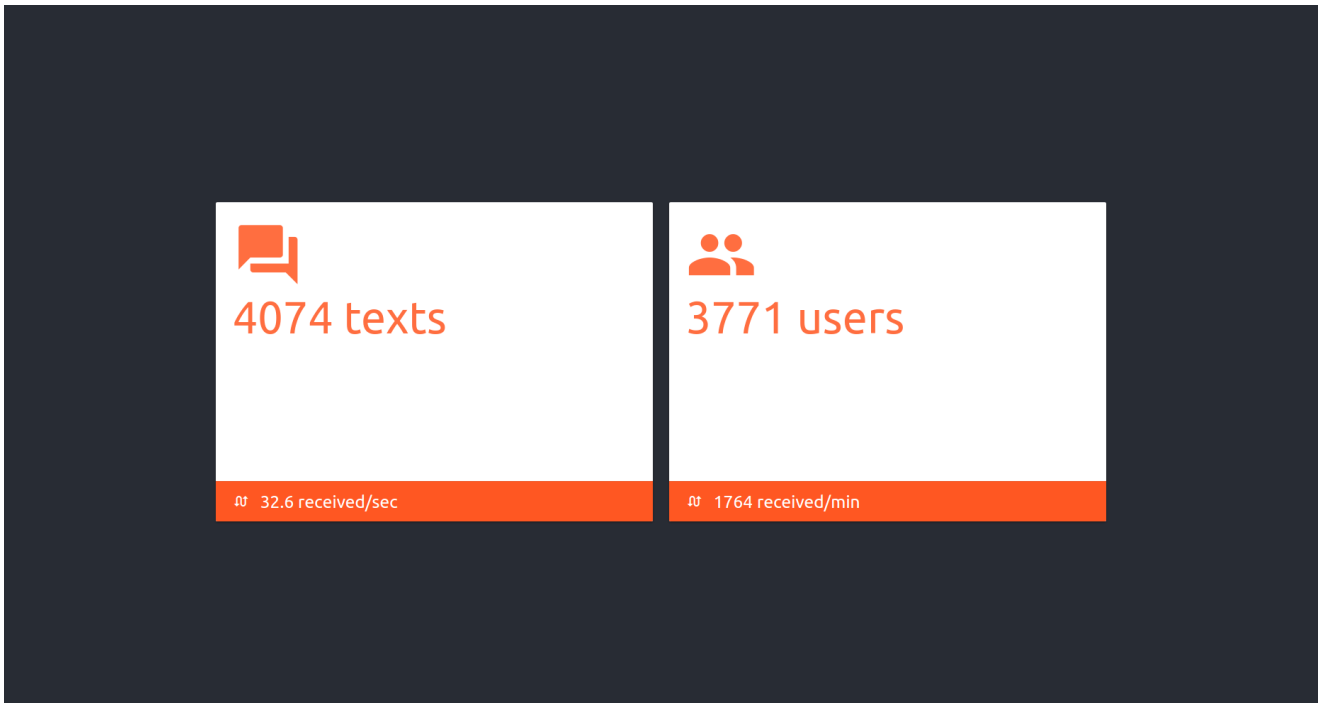

Figura B.9: Dashboard módulo de generación de estadísticas

B.4. Ejemplo de módulo creado por el usuario

Como se ha comentado anteriormente, existe la posibilidad de que un usuario programe y suba sus propios módulos a la plataforma. Para ello, debe generar una imagen *Docker* y guardarla en un fichero .tar.

En el fragmento de código B.1, se puede ver un *Dockerfile* de ejemplo. En él se puede apreciar cómo se parte de la versión de *Catenae* correspondiente y se copia en su interior un *script* de Python, el del fragmento B.2. En este caso, se trata de un módulo muy simple que tan sólo cuenta los textos que recibe, pero se podría complicar todo lo que se quisiese e implementar algún módulo que todavía no lo esté en la plataforma.

```
1  FROM catenae/link:1.0.0rc2
2
3  RUN \
4      apt-get update && apt-get install libsqlite3-0 && apt-get -y
       install gcc mono-mcs &&  apt-get install -y libxft-dev
       libfreetype6 libfreetype6-dev \
5      && pip install --upgrade pip \
6      && pip install \
7          catenae \
8      && apt-get clean \
9      && rm -rf \
10         /root/.cache/pip \
11         /var/cache/apk/* \
12     && find / -type d -name __pycache__ -exec rm -r {} \+
13
14 # Topology links
15 COPY prueba.py /opt/catenae/
16
17 ENTRYPOINT ["python", "prueba.py"]
```

Fragmento de código B.1: Dockerfile de ejemplo para módulo creado por el usuario

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from catenae import Link, Electron
import traceback

class Prueba(Link):

    def setup(self):
        self.contador = 0

    def transform(self, electron):
        try:
            self.contador += 1
            electron.value = {"texts":self.contador}
            self.send(electron)
        except Exception as e:
            print("Exception",e)

if __name__ == "__main__":
    p = Prueba()
    p.start()
```

Fragmento de código B.2: Módulo creado por el usuario de ejemplo

Apéndice C

Licencia

Todo el código fuente que acompaña a la presente memoria es software libre: puede ser redistribuido y/o modificado bajo los términos de la *GNU General Public License* como es publicada por la Free Software Foundation en su versión 3[1].

[1]https://www.gnu.org/licenses/gpl-3.0.html

Bibliografía

[1] Rodrigo Martínez-Castaño, Juan Carlos Pichel, David E. Losada y Fabio Crestani, "A Micromodule Approach for Building Real-Time Systems with Python-Based Models: Application to Early Risk Detection of Depression on Social Media", en *40th European Conference on Information Retrieval, Grenoble(France)*, Springer Verlag, pp. 801-805, 2018.

[2] Rodrigo Martínez-Castaño, Juan Carlos Pichel y David E. Losada, "Building Python-Based Topologies for Massive Processing of Social Media Data in Real Time", en *5th Spanish Conference in Information Retrieval, Zaragoza(España), ACM*, pp. 18:1-18:8, 2018.

[3] Streaming vs Batch: The Differences.(*https://softwareengineeringdaily.com/2015/08/03/batch-vs-streaming-the-differences/*). (2015). Consultado por última vez el 03/06/2019.

[4] Iñaki San Vicente, Xabier Saralegi y Rodrigo Agerri. "Talaia: a Real time Monitor of Social Media and Digital Press", 2018.

[5] Radim Rehurek, Petr Sojka, "Software framework for topic modelling with large corpora", en *Proceedings of the LREC 2010 Workshop on New Challenges for NLP frameworks*, pp 45—50, 2010.

[6] Kanban – A brief introduction (*https://es.atlassian.com/agile/kanban*). Consultado por última vez el 13/06/2019.

[7] Scrum: qué es, cómo funciona y por qué es excelente (*https://es.atlassian.com/agile/scrum*). Consultado por última vez el 13/06/2019.

[8] Estructura de descomposición del trabajo. Artículo de la Wikipedia (*https://es.wikipedia.org/wiki/Estructura_de_descomposici%C3%B3n_del_trabajo*). Consultado por última vez el 03/06/2019.

[9] David M. Blei, Andrew Y. Ng y Michael I Jordan, "Latent Dirichlet allocation", en *Lafferty, John. Journal of Machine Learning Research 3 (4–5)*, pp. 993–1022, 2003.

[10] Intuitive Guide to Latent Dirichlet Allocation (*https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158*). Consultado por última vez el 17/06/2019.

[11] Carson Sievert y Kenneth E. Shirley, "LDAvis: A method for visualizing and interpreting topics", en *2014 ACL Workshop Interactive Language Learning, Visualization, and Interfaces, Baltimore (EEUU)*,pp. 63-70, 2014.

[12] Python Web Server Gateway Interface v1.0.1 (*https://www.python.org/dev/peps/pep-3333/*). Consultado por última vez el 19/06/2019.

[13] Kubernetes vs Docker (*https://www.sumologic.com/blog/kubernetes-vs-docker/*). Consultado por última vez el 19/06/2019.