Obtención, análisis y visualización de series temporales mixtas de datos estructurados y no estructurados obtenidos de Twitter

Xavier Gervilla Machado, autor
David Prat Robles, director
Lluís Padró Cirera, ponente
Joaquim Deulofeu Aymar, tutor de GEP
Datapta Systems & Services
25-01-2023
Grado en Ingeniería Informática
Especialidad de computación
Facultad de informática de Barcelona (FIB)
Universidad Politécnica de Cataluña (UPC) – BarcelonaTech

Índice

1. Abstract

El mercado de activos y criptomonedas ha crecido significativamente en los últimos años al igual que el número de jóvenes interesados en invertir y ganar dinero "fácil". Este aumento, sin embargo, no se ha acompañado de un incremento en formación, resultando en un gran número de errores evitables y la pérdida de grandes cantidades de dinero.

Una metodología comúnmente utilizada para minimizar dichos riesgos consiste en consultar las redes sociales en busca de expertos que orienten o de identificar tendencias. El uso de la inteligencia artificial puede ser muy beneficioso para automatizar los pasos seguidos en esta metodología, en combinación con métricas como la cantidad de menciones o el sentimiento asociado a los comentarios de las redes sociales.

En este proyecto se pretende desarrollar una aplicación capaz de obtener y procesar datos de redes sociales en tiempo real con la que fomentar la inversión consciente y no especulativa, valorando también el rendimiento de la base de datos utilizada y creando una página web donde los usuarios puedan consultar las distintas métricas generadas.

The asset market has grown significantly in the last few years as well as the number of young people interested in investing and making *"easy"* money. This increase in demand, however, has not been met with an increase in formation or education, resulting in lots of avoidable mistakes and the loss of considerable amounts of money.

A common approach on minimizing losses consists of using social media to look for the guidance of specialists or try to identify the latest trends. The use of artificial intelligence can be highly beneficial to automate the steps followed by this approach along with the use of metrics such as the number of mentions or the sentiment of the comments.

The purpose of this project is to develop an app capable of obtaining and processing data from social media in real time, automating the process of identifying trends and promoting conscious investment against speculation. Besides the AI engine behind the app, this project also focuses on the efficiency and performance of the chosen database and the implementation of a webpage where users can see the processed data and generated metrics.

2. Introducción

El aumento del interés en las criptomonedas ha provocado que muchos jóvenes hayan probado suerte con la esperanza de generar mucho dinero de forma rápida. Es posible que algunos de ellos hayan acertado y hayan sido capaces de obtener grandes beneficios, pero para la mayoría de ellos la inversión ha acabado en números rojos; y es que la falta de información y conocimiento provoca que las inversiones sean mayormente especulativas.

Partiendo de este problema podemos identificar que hay una falta de herramientas que permitan a todos aquellos interesados en invertir hacerlo con la formación o los recursos necesarios para poder decidir de forma crítica. Con este trabajo se pretende dar solución a dicho problema desarrollando una aplicación donde mostrar distintas métricas o valores de referencia que permitan decidir conscientemente y con la formación adecuada.

En la actualidad, uno de los principales recursos utilizados para la toma de decisiones sobre inversiones son los comentarios o publicaciones en redes sociales. Existen foros de discusión dedicados exclusivamente a hablar sobre criptomonedas o activos, y en muchos casos los propios usuarios que participan en ellos también invierten. Sin embargo, las plataformas ya existentes que utilizan estos datos como referencia tienen un uso muy limitado por lo que la mayoría de inversores las desconoce por completo.

Es así como surge la solución que se plantea en este trabajo, con un objetivo más generalista: proporcionar una visualización de bases de datos de series temporales, aplicadas al mercado de acciones y a tareas de predicción. En otras palabras, la aplicación planteada tiene un foco más amplio que las plataformas ya existentes para poder orientar mejor a los usuarios no sólo en cuanto a inversiones sino a nivel económico para fomentar de forma complementaria una concienciación sobre la economía general.

Para poder implementar dicha aplicación los pasos seguidos son: obtener datos relacionados con los mercados de acciones y criptomonedas y conceptos de economía, aplicar modelos de aprendizaje automático para identificar información relevante adicional, almacenar los datos y analizar el rendimiento de la base de datos para garantizar una interacción fluida en la visualización. Además, en este proyecto también se valorarán distintas alternativas, los costes de desarrollar la aplicación y la sostenibilidad.

3. Alcance y contexto del proyecto

El proyecto propuesto se realiza en colaboración con la empresa Datapta Systems & Services y forma parte del desarrollo de un nuevo producto, Finlooker. Finlooker es una herramienta que permite a sus usuarios ver y entender el mercado de activos y acciones poniendo el foco en las sensaciones que tienen aquellas personas que están interesadas en el mercado en vez de centrarse únicamente en el valor monetario de los activos. Además, Finlooker pone en relevancia el contexto económico para contextualizar las sensaciones del mercado dentro de la situación económica a nivel global.

3.1. Objetivos

Dentro del desarrollo de Finlooker, este proyecto abarca dos pasos clave: la obtención de los datos que se utilizan para detectar las sensaciones sobre el mercado y la situación económica, y el desarrollo y análisis de una base de datos donde almacenar dichos datos. Dentro de estos pasos clave, hay que tener en cuenta también que los datos obtenidos deben ser procesados mediante tareas de predicción antes de almacenarlos. Además de estos dos pasos como parte de Finlooker se plantea un tercero que corresponde a la visualización de los datos para facilitar a los usuarios su comprensión.

Podemos establecer por lo tanto el objetivo general del proyecto: proporcionar una visualización de bases de datos de series temporales, aplicadas al mercado de acciones y a tareas de predicción. Este objetivo general se puede descomponer en tres sub-objetivos que corresponden a los distintos pasos planteados: obtención y procesamiento de los datos, almacenamiento y visualización.

Adicionalmente, un alto rendimiento del sistema resultante es esencial y crítico para el buen funcionamiento de la aplicación. Dada la gran cantidad de datos a procesar, almacenar y mostrar, la eficiencia de los distintos componentes que forman el proyecto y un rendimiento elevado son especialmente importantes. Para reflejar esta relevancia se considera como objetivo adicional la optimización de la base de datos, para lo cual se utilizarán varias métricas que permitan comparar distintos diseños e implementaciones.

A lo largo del documento también se referirá a los diferentes sub-objetivos como fases dada la dependencia entre ellos de cara al desarrollo, ya que sin datos procesados el segundo y

tercer sub-objetivos carecen de significado. Sin embargo, en todo momento se tendrán en consideración el objetivo general del proyecto y las fases posteriores para adaptar el diseño y desarrollo a los requerimientos de los diferentes sub-objetivos y se considerará también la relevancia del proyecto dentro del contexto de Finlooker.

3.2. Actores implicados

El desarrollo de este proyecto cuenta con la implicación de diversos actores con un nivel de relevancia no equitativa. Dentro del contexto de Datapta, la empresa donde se desarrolla, podemos encontrar los tres primeros actores más relevantes para el desarrollo: el autor y desarrollador principal Xavier Gervilla, el director y guía David Prat y el producto en desarrollo Finlooker, principal beneficiario de este proyecto. Pero más allá de la empresa encontramos otros dos actores: Lluís Padró y Joaquim Deulofeu como personal académico y los usuarios finales.

A pesar de formar parte de un producto en desarrollo el proyecto se puede adaptar a otros entornos, lo cual implica que los usuarios finales puedan variar según se valore su puesta en el mercado. Por lo tanto debemos diferenciar según el propósito original de Finlooker y según un potencial más general del proyecto. En el primer caso, los usuarios finales son todas aquellas personas interesadas en el mercado de activos y criptomonedas.

Si por el contrario consideramos las funcionalidades que ofrece el motor de obtención de datos y procesado de forma independiente estamos delante de un producto con un gran potencial para ver cómo se siente la población sobre ciertos temas en tiempo real. En este caso el producto principal consistiría en adaptar el motor de datos y ofrecerlo como servicio a empresas o entidades para buscar los tweets que sean relevantes para los clientes. Por ejemplo, se podría adaptar para monitorear grandes eventos como la final de la Champions y ofrecerlo a los clubes de fútbol participantes para su propio análisis interno u otros fines según los datos obtenidos. Determinar con concreción quiénes son los usuarios finales en este segundo caso no es posible pero a nivel general se podrían identificar las propias empresas a las que está dirigido o de forma indirecta a los consumidores finales de los productos que las empresas desarrollaran a partir de los datos.

4. Estado del arte

Teniendo en cuenta el interés al alza sobre el mercado de activos, es razonable plantearse el uso de productos ya existentes con funcionalidades similares a las propuestas en el apartado anterior. La posibilidad de poder adaptar productos ya existentes o incluso de usarlos como referencia puede llegar a suponer un gran ahorro en tiempo, recursos y dinero.

4.1. Productos similares

Los dos principales productos encontrados que ofrecen funcionalidades similares al proyecto son *TopStonks* [1] y *SocialSentiment.io* [2]. Ambos proyectos ofrecen una visualización de la percepción social sobre activos del mercado de acciones, pero con diferencias más relevantes en los datos utilizados y cómo mostrarlos. La diferencia más significativa entre los dos productos y la aplicación planteada son los activos considerados, en concreto el uso exclusivo de activos relacionados con empresas -como Apple o Google- sin considerar criptomonedas ni datos de economía con los que contextualizarlos, lo cual sí se considera en este proyecto.

TopStonks

El primer producto valorado difiere principalmente del proyecto propuesto en la red social utilizada para la obtención de los datos. En concreto, se obtienen de *Reddit* -del subreddit *r/wallstreetbets* para ser más exactos-, *4chan* y *Twitch*. El foco de este producto está en la evolución de la cantidad de menciones mientras que el sentiment se muestra como un valor único que complementa la evolución. A pesar de considerar diferentes activos, como punto favorable destaca una pequeña contextualización de los datos utilizados para obtener el sentiment mostrando los comentarios de las redes sociales utilizados tal y como se aprecia en la figura 1.

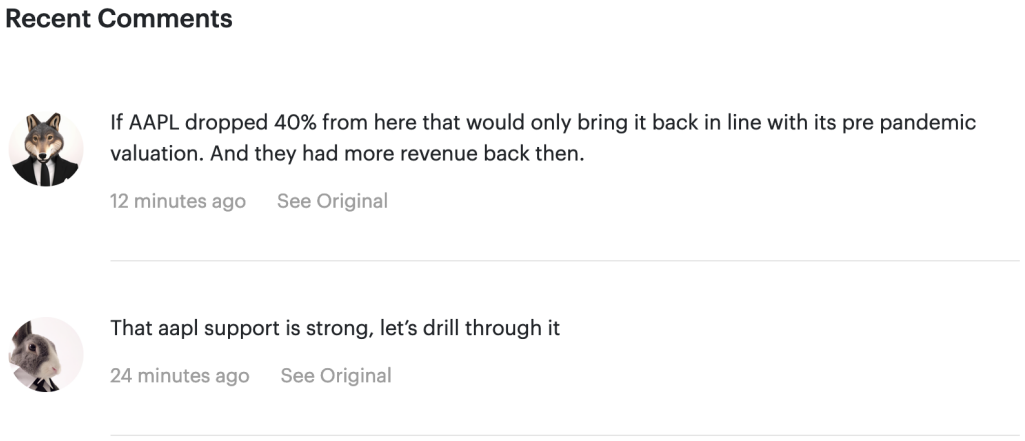

Figura 1: Comentarios en TopStonks sobre AAPL

SocialSentiment.io

El segundo producto valorado sí que utiliza Twitter como fuente de datos por lo que la principal diferencia son los activos considerados. Al utilizar datos de Twitter se puede plantear la opción de aprovecharlos y así limitar el uso del motor de búsqueda a la búsqueda de tweets sobre criptomonedas y del contexto económico, pero esta limitación del uso no se traduce en una limitación de la carga de trabajo.

Para aprovechar los datos de activos de empresas SocialSentiment.io dispone de una API mediante la cual se pueden obtener los datos que la empresa ha obtenido de Twitter. El principal inconveniente de la API son las limitaciones que tiene, es decir, no es posible obtener todos los datos en tiempo real o con un margen de tiempo aceptable.

Teniendo en cuenta las dos alternativas planteadas, al no incluir datos de criptomonedas ni de la situación económica a nivel general y no disponer de los datos en tiempo real, la necesidad de implementar un motor de obtención de datos se mantiene. Además, ninguno de los dos productos especifica detalles sobre cómo se obtienen los datos, por lo que no se pueden aprovechar las tecnologías que utilizan o la metodología de obtención y procesamiento que siguen. Sin embargo, ambos productos sí que pueden servir como referencia en el desarrollo de los gráficos para una visualización que sea fácil y entendible para los usuarios finales.

4.2. Frameworks y librerías

En lo que respecta a las herramientas utilizadas para la implementación y el desarrollo del proyecto, se establece la implementación con Python 3.7 y se prevé el uso de cuatro principales librerías: *Tweepy* [3], *PySpark* [4], *Awswrangler* [5] y *Plotly* [6].

Más allá de Tweepy para la obtención de los datos de Twitter y de Awswrangler para la interacción con AWS -donde se despliega la aplicación-, el uso de estas principales librerías no está determinado por las funcionalidades sino que es elección del desarrollador. Esto implica que se podrían utilizar librerías similares para procesar los datos, interactuar con el almacenamiento y visualizarlos.

De esta manera, un método alternativo podría consistir en recibir los datos de Twitter y controlar programáticamente la paralelización del proceso sin depender de frameworks

especializados como Spark, o se podrían generar los gráficos de la visualización con Matplotlib por ejemplo. Además de utilizar distintas librerías, utilizar un lenguaje de programación distinto como Go podría facilitar la implementación con alguna de las alternativas como la paralelización, por lo que existen distintos métodos con los que desarrollar el proyecto y llegar así al objetivo planteado.

En cuanto al sub-objetivo del almacenamiento, hay una cantidad más que considerable de bases de datos que se pueden utilizar. La complejidad para adaptar la estructura de los datos a almacenar a cada una de ellas es variable pero en ningún caso se garantiza que una mayor complejidad se pueda traducir en un mejor rendimiento. Para el desarrollo de este proyecto se considera el uso de InfluxDB y la posibilidad de explorar bases de datos distintas en caso de obtener un rendimiento peor al deseado.

4.3. Justificación del proyecto

Teniendo en cuenta los productos ya existentes y los frameworks y librerías planteados es necesario justificar la necesidad de desarrollo del proyecto.

Teniendo como referencia los dos productos valorados, es razonable pensar que existe una demanda de información que utilizar como referencia para valorar el mercado de activos y criptomonedas. Partiendo de estos dos productos, la aplicación planteada amplía las funcionalidades de los productos ya existentes tanto a nivel de activos mostrados como en el contexto económico general, por lo que en cuanto a funcionalidades respecta está justificada su implementación. Pero entrando en más detalle en este proyecto, la implementación de los tres sub-objetivos se debe justificar por partes. Esto se debe a que las diferencias entre sub-objetivos requieren distintas necesidades de implementación y esto implica que a nivel de sub-objetivo haya recursos ya existentes que se pueden reutilizar.

Si nos fijamos en el primer sub-objetivo, la obtención de los datos, podemos afirmar que no hay alternativas que permitan lo que se pretende conseguir. Existen productos que permiten obtener datos de Twitter pero no se pueden aprovechar para este proyecto por tres razones: no permiten la obtención en streaming o tiempo real, tienen un coste muy elevado y no ofrecen todos los tweets necesarios. Además, las especificaciones de los motores de búsqueda que utilizan estos productos externos no son de dominio público por lo que no es posible adaptar

tecnologías ya existentes. Ante la falta de tecnologías que adaptar o aprovechar está justificado el desarrollo de este sub-objetivo desde cero.

En cuanto al segundo sub-objetivo, el almacenamiento de los datos, no existen argumentos suficientes que justifiquen reutilizar bases de datos ya existentes. Utilizar una base de datos ya existente podría reducir los costes de desarrollo, pero al tratarse de una estructura de datos de nueva creación -a excepción de los dos productos valorados, que no especifican detalles sobre el método de almacenamiento-, para utilizar una base de datos existente se debería adaptar el formato de datos. Esta adaptación supone un incremento en la complejidad y el coste de desarrollo que puede ser incluso mayor que el coste de diseñar e implementar la base de datos desde cero, por lo que sería contraproducente. Diseñar la base de datos para el proyecto, por lo tanto, permite adaptarse a las necesidades reales del sistema y evitar un incremento de la complejidad innecesario.

Si en cambio vemos el tercer sub-objetivo, la visualización, hay un contraste respecto a las dos fases anteriores ya que los productos valorados sí pueden utilizarse como referencia. Inspirarse en TopStonks o SocialSentiment.io puede mejorar los gráficos y utilizar así un formato con el que los usuarios tengan más familiaridad o facilidad de comprensión. Además, se pueden utilizar otros proyectos ya existentes que utilicen la librería Plotly para reutilizar metodologías de generación o personalización de gráficos. Por lo tanto, en esta fase sí que puede ser positivo inspirarse en productos ya existentes o especializados en la visualización.

5. Arquitectura

Como paso previo al desarrollo más detallado de los distintos sub-objetivos vamos a exponer la arquitectura sobre la que se basará el proyecto. Hay dos distintos métodos para valorar la arquitectura: a nivel funcional y a nivel de despliegue o deployment.

5.1. Funcionalidades

A nivel funcional la arquitectura resultante corresponde a los distintos componentes que se ejecutan y al flujo de datos que se genera. Dichos componentes se especifican y justifican más adelante, por lo que en esta sección los componentes se identifican a nivel general.

Teniendo en cuenta la división funcional por secciones planteada a lo largo del proyecto podemos identificar tres grandes componentes: motor de obtención y procesamiento, almacenamiento y visualización. Sin embargo, los datos se utilizan en varios procesos dentro del motor de obtención y procesamiento y del almacenamiento, por lo que estos dos grandes componentes se dividen en dos distintos componentes según su funcionalidad. En la figura 2 a continuación se pueden apreciar las 5 componentes que forman las distintas funcionalidades de este proyecto: Twitter engine, Spark engine, base de datos, lago de datos y motor de visualización.

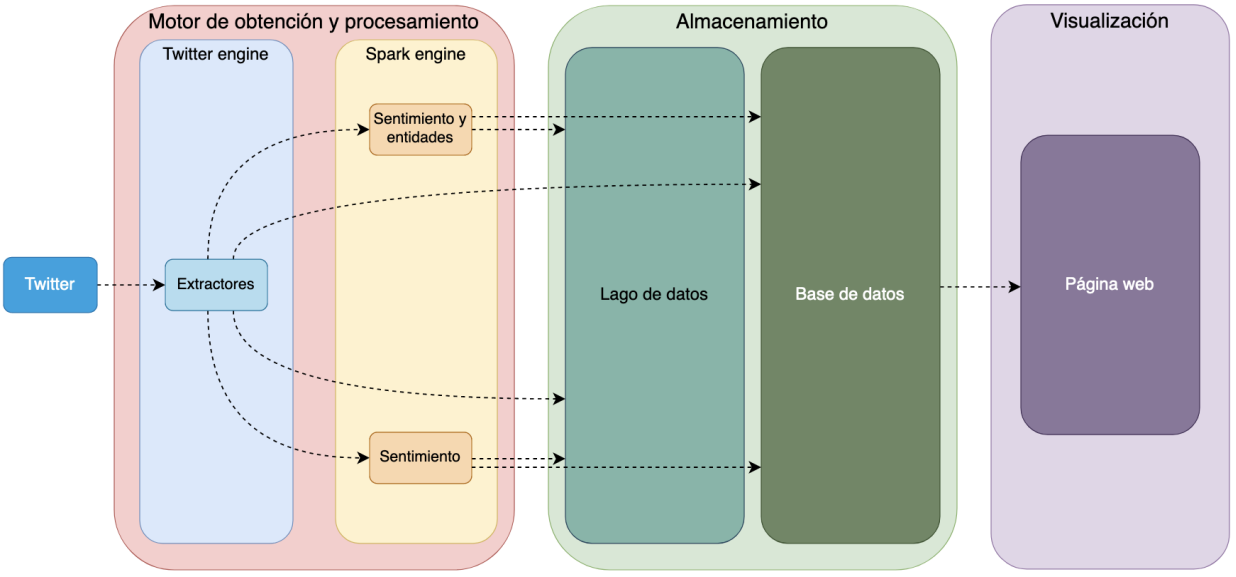

Figura 2: Arquitectura de las distintas funcionalidades

Dentro del Spark engine podemos identificar dos elementos con funcionalidades muy similares pero lo suficientemente distintas como para poder diferenciarlos. En concreto, la diferencia reside en los modelos de inteligencia artificial que se aplican sobre los datos, más

detallado en la implementación del Spark engine. El Twitter engine, por otro lado, tal y como se verá en su respectiva sección también se divide en dos elementos dependiendo del punto de acceso utilizado para recibir los datos de Twitter, pero a nivel funcional ambos elementos son idénticos.

Si además de las componentes nos fijamos en las flechas que representan el flujo de los datos, en la figura 2 se puede observar también el planteamiento hecho sobre el desarrollo proyecto, donde se justifica el desarrollo del proyecto de forma progresiva según los sub-objetivos por la dependencia entre implementaciones.

5.2. Deployment

A nivel de despliegue la arquitectura resultante corresponde a los distintos recursos que se utilizan para poder ejecutar todos los componentes y procesos necesarios. En este caso, distintas funcionalidades se agrupan para ejecutar en un mismo recurso, por lo que el esquema de la figura 2 evoluciona al esquema resultante en la figura 3.

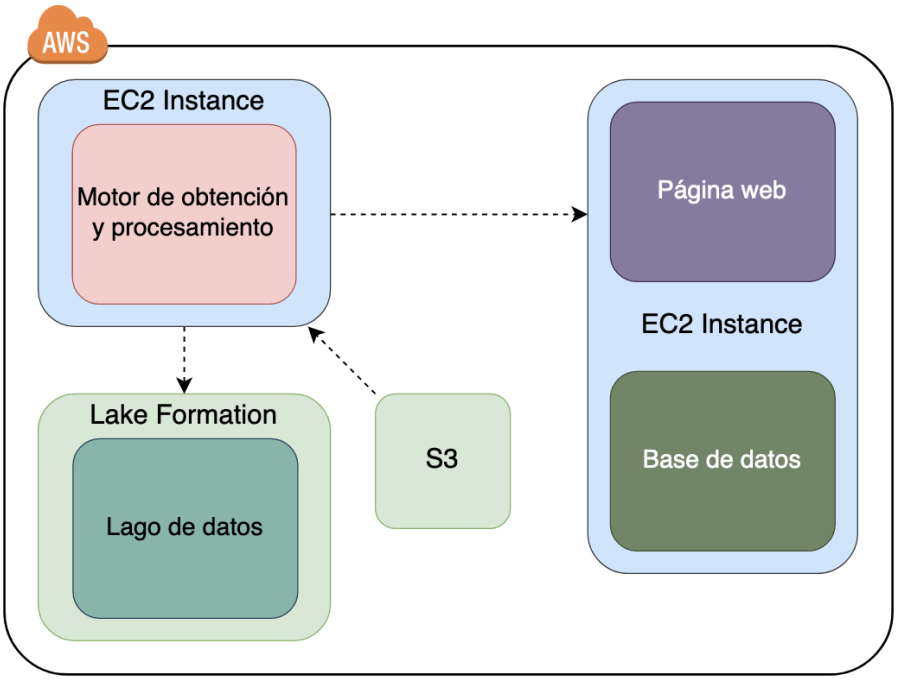

Figura 3: Arquitectura de los distintos recursos utilizados

Es importante destacar que este esquema incluye un elemento que no aparece en el esquema de funcionalidades, el sistema de almacenamiento S3. Esto se debe a que este almacenamiento no se corresponde con ninguna funcionalidad per se sino que el único

propósito que tiene es servir como sistema de referencia donde almacenar las distintas listas que se utilizan para personalizar la búsqueda del Twitter engine.

La arquitectura de implementación resultante, por lo tanto, tiene cuatro principales recursos de Amazon Web Services: una instancia EC2 para el motor de obtención y procesamiento, una segunda instancia EC2 para la base de datos escogida y la página web donde interactúa el usuario, la implementación del lago de datos mediante Lake Formation y S3 para el almacenamiento de las listas mencionadas. Sin embargo ante esta arquitectura se puede plantear una mejora directa para evitar uno de los cuatro recursos, trasladar las listas a la EC2 Instance que ejecuta el Twitter engine.

Trasladar las listas puede parecer un pequeño gesto con un gran impacto ya que aparentemente se puede descartar un recurso sin sacrificar rendimiento en la instancia -teniendo en cuenta que las listas ocupan unos pocos kilobytes-, pero hay un motivo por el cual no se ha optado por implementar dicha mejora. Al pensar en la escalabilidad del sistema junto con la arquitectura de funcionalidades se hace evidente que la implementación es claramente modular, ya que pese a las dependencias de los flujos de datos cada componente puede ejecutarse con sus propios recursos de forma independiente.

Por lo tanto es razonable pensar que cada componente se pueda optimizar para distintos recursos y así maximizar el rendimiento global, es decir, si el Spark engine se puede adaptar para ejecutar con AWS EMR en modo cluster se incrementa la velocidad de procesamiento a la vez que disminuyen los costes de los recursos utilizados por ejemplo. De forma equivalente, se podrían utilizar nuevas instancias de menor potencia pero más focalizadas en la ejecución de una sub-componente del Twitter engine, lo cual permitiría tener en ejecución únicamente los motores de obtención y procesamiento necesarios sin desperdiciar recursos.

El desarrollo de este proyecto no abarca la implementación de dichas optimizaciones pero sí que las tiene en consideración como futuras mejoras o pasos adicionales, y es por ello que se utiliza S3 como método de almacenamiento de las listas a modo de preparación para una mayor escalabilidad.

6. Obtención y procesamiento de los datos

Los datos utilizados para el desarrollo de este proyecto provienen de *Twitter* y una vez obtenidos se procesan mediante *Spark*, donde se les aplican los modelos de NLP apropiados. Dentro de esta sección se determina qué tipos de datos son necesarios, cómo obtenerlos y cómo procesarlos una vez obtenidos.

6.1. Tipos de datos utilizados

Para poder definir qué datos son necesarios hay que considerar el propósito del proyecto dentro de Finlooker. En este contexto, podemos identificar dos conjuntos de datos: percepción social sobre el mercado de activos y percepción social sobre la economía.

En ambos conjuntos se trata el concepto de "percepción social", que en el contexto de este proyecto corresponde a cómo ven los usuarios de las redes sociales ciertos temas. Utilizando valor numérico se clasifican los datos obtenidos según sean positivos, cercanos a 1, o negativos, cercanos a -1.

El primer conjunto de datos corresponde a los tweets que tratan sobre el mercado de activos, en concreto sobre un conjunto de criptomonedas y activos de grandes empresas que cotizan en bolsa. Para identificar los tweets que contienen estos datos se utilizan los *cashtags* (*$*), identificadores de Twitter reservados a los símbolos de los activos. De forma similar al uso de *hashtags* (#), los cashtags permiten identificar de forma directa de qué trata el tweet como en la figura 4, por lo que se pueden aprovechar para filtrar los tweets que interesan.

Figura 4: Tweet de muestra con el activo $SOL (Solana).

El segundo conjunto de datos corresponde a los tweets que tratan sobre la economía. En concreto, se considera una lista de conceptos económicos como inflación o deflación con el objetivo de proporcionar una visión general de la situación económica. La contextualización

de los datos de activos es un factor clave para que los datos de activos tengan la relevancia esperada, es decir, si se valoran los datos con intención de invertir a largo plazo es igual o más importante considerar la situación económica general antes que la evolución más reciente de los activos. Al tratarse de datos de carácter más general no hay identificadores tan concretos como los cashtags de los activos, por lo que la mejor aproximación para filtrar los datos es utilizar hashtags o, directamente, los propios conceptos como *inflation* en la figura 5.

Además de los conceptos de economía, este conjunto de datos se ve complementado por una serie de usuarios que denominamos cuentas de expertos. Estas cuentas pertenecen a empresas o profesionales del sector de la economía, por lo que considerar sus tweets es beneficioso en el sentido de contrastar la percepción social con la valoración de expertos en el sector. A los tweets del conjunto de economía se les debe aplicar, además de un modelo de ML para obtener la percepción social, un modelo NER para obtener entidades o temas relacionados.

Figura 5: Tweet de muestra sobre la inflación (*inflation*)

6.2. Obtención de los datos

Para obtener los datos de Twitter hay varias aproximaciones posibles, principalmente obtenerlos de proveedores externos o directamente de Twitter. Dentro de estas dos opciones hay distintas ventajas e inconvenientes, pero tal y como se verá a continuación obtener los datos directamente de Twitter es la única opción viable.

La primera opción, obtenerlos de proveedores externos, tiene una ventaja innegable: permite aprovechar recursos ya existentes. Ya sea porque los proveedores ofrecen los datos de Twitter como producto, como es el caso de *Tweet Binder* [7], o porque los obtienen para desarrollar sus productos y los ofrecen bajo ciertas condiciones, como es el caso de *SocialSentiment.io*,

obtener los datos de terceros supone un ahorro de tiempo además de un impacto positivo en el medio ambiente respecto a la alternativa.

Sin embargo, actualmente no hay ningún proveedor que permita obtener una cantidad de datos significativa a un bajo coste, por lo que el coste de obtención es superior al coste de desarrollar un motor de obtención para este proyecto. Hay que tener en cuenta también que obtener los datos mediante terceros tiene una principal desventaja, y es que no se pueden obtener los datos en *streaming*. Por ello, la única opción viable para obtener los datos teniendo en cuenta el propósito de este proyecto es obtenerlos directamente de Twitter.

Para obtener datos directamente de Twitter la propia compañía dispone de herramientas para desarrolladores que permiten crear proyectos con los que interactuar con otros usuarios de forma automática, integrar datos con otras aplicaciones o automatizar la publicación de tweets, por ejemplo. Para ello hay dos principales endpoints con los que interactuar con la API de Twitter: endpoint v1.1 y endpoint v2 [8].

A pesar de tener ciertas limitaciones que se verán a continuación, tanto el endpoint v1.1 como el endpoint v2 ofrecen distintos niveles de acceso que permiten reducir o incluso anular dichas limitaciones: *Standard*, *Premium* y *Enterprise*. El primer nivel permite una gran variedad de accesos a nivel general, el segundo ofrece accesos adicionales centrados en la búsqueda de tweets y el tercer nivel es totalmente personalizado según las necesidades de la empresa interesada o del producto en desarrollo. Los dos últimos niveles, pero, tienen un coste significativamente elevado.

Dentro del nivel *Standard* se encuentran tres distintos subniveles que permiten reducir las limitaciones: *Essential*, *Elevated* y *Academic Research*. De los tres, el único que ofrece una reducción de las limitaciones considerables es Academic Research, pero para poder acceder se deben cumplir estrictas condiciones de uso. Estas condiciones no permitirían integrarlo en el proyecto como parte de Finlooker, ya que se prohíbe entre otras cosas usar los datos obtenidos como parte de un producto con fines lucrativos, por lo que en los endpoints se considerará el nivel Standard con acceso Elevated ya que Essential no permite utilizar el endpoint v1.1.

Endpoint v1.1

El endpoint v1.1 es el endpoint inicial u original de Twitter creado en 2012. Al ser un endpoint con 10 años de antigüedad ha ido evolucionando y mejorando con el paso del tiempo, pero las distintas mejoras han sido insuficientes para evolucionar al mismo ritmo que las tecnologías y productos que lo utilizan. Es por ello que este endpoint está siendo lentamente desmantelado y algunas de sus funcionalidades dejarán de ser usables durante el desarrollo de este proyecto [9].

El endpoint v1.1 permite obtener tweets en tiempo real sin un límite de cantidad mensual. Además permite filtrar los tweets por cashtag, lo cual es un punto clave de las necesidades del motor de búsqueda. Obtener tweets en estas condiciones se hace mediante el punto de acceso *https://stream.twitter.com/1.1/statuses/filter.json* y una lista de activos que buscar con el formato "*%24activo1,...,%24activoN*". Este formato es equivalente a hacer una operación lógica OR entre los diferentes activos con su cashtag, es decir, es equivalente a la expresión "*$activo1 OR $activo2 ... OR $activoN*".

La principal limitación de este punto de acceso es la cantidad de activos que se pueden pasar en una misma llamada, ya que dependiendo de los activos pasados la cantidad de tweets a obtener será demasiado elevada y se formará un cuello de botella. Con la lista de activos propuesta este límite está en torno a los 250 activos. Esta limitación se puede solucionar en parte ejecutando un segundo código en paralelo y dividiendo la lista en dos, lo cual disminuye la cantidad de tweets a obtener en una sola ejecución considerablemente y así se evita el cuello de botella, dejando margen en caso de producirse un pico de tweets.

Endpoint v2

El endpoint v2 se puso en producción en noviembre de 2021 [10] para dar solución a las limitaciones tecnológicas del endpoint v1.1. Este endpoint permite las mismas funcionalidades que su antecesor junto con nuevas funcionalidades y da mayores facilidades a los desarrolladores para poner en marcha sus productos y/o poner a prueba sus aplicaciones. Por lo tanto, y teniendo en cuenta también que el endpoint v1.1 está en desmantelamiento, es el punto recomendado para el desarrollo de nuevas aplicaciones.

Su principal ventaja respecto al endpoint v1.1 es la facilidad de controlar qué datos se obtienen mediante reglas. Cada regla tiene un límite de 512 carácteres y se pueden crear hasta 25 reglas a la vez. Además, las reglas se pueden ver, crear o modificar fuera del punto de acceso [11, 12], lo cual favorece una implementación por módulos con la que reutilizar código. El formato de las reglas permite combinar operadores para obtener mejores resultados.

El principal inconveniente del endpoint v2 es una limitación mensual en la cantidad de tweets, de 2 millones de tweets con acceso Elevated pero que con un acceso superior se podría llegar a incrementar [13].

6.3. Motor de búsqueda: *Twitter engine*

Teniendo en cuenta ambos endpoints y el tipo de datos utilizados o que queremos obtener, podemos definir las características de nuestro motor de búsqueda. El motor de búsqueda tiene dos componentes, uno por cada endpoint.

El primer componente, que corresponde al endpoint v1.1, es el encargado de obtener los datos relacionados con activos ya que se identifican mediante cashtags y el volumen de tweets relacionados con activos supera con creces el límite mensual del endpoint v2.

El segundo componente, que corresponde al endpoint v2, es el encargado de obtener los datos relacionados con la economía -identificados mediante hashtags- y de las cuentas de expertos. Las mejoras de diseño de este endpoint lo convierten en una mejor opción para filtrar los tweets mediante nombres de usuarios -cuentas de expertos- y el volumen de tweets relacionados con la economía es considerablemente menor que el de los activos, por lo que no se espera superar el límite mensual.

Como medidas de preparación, en caso de superar el límite mensual se han creado unas credenciales complementarias para cambiar de credenciales de forma automática y así aumentar el límite a 4 millones de tweets totales. Esta medida de contención permitirá adaptar también el código del primer componente cuando el endpoint v1.1 deje de ser operativo, aunque entonces la cantidad de credenciales necesarias para no superar el límite será mayor.

6.4. Motor de procesamiento: *Spark engine*

Una vez definidos qué tipos de datos se utilizan y cómo obtenerlos es necesario determinar cómo se procesan hasta llegar a la fase de almacenamiento. En concreto, es necesario ver cómo tratar los datos en tiempo real y qué modelos de NLP aplicar para obtener el valor de percepción social y las entidades relacionadas.

Para tratar los datos en tiempo real se ha optado por utilizar *Spark*, definido como un framework de computación de datos a gran escala [14, 15, 16]. Este framework está ideado para procesar de manera paralela automáticamente por lo que para entender mejor la evolución de los datos especificamos primero los pasos que se deben seguir:

1. Recibir los datos del motor de búsqueda.
2. Hacer un procesado previo con el que prepararlos para los modelos NLP.
3. Aplicar los modelos NLP.
4. Agrupar los datos según sea conveniente.
5. Hacer un procesado posterior con el que prepararlos para las bases de datos.
6. Almacenarlos en las bases de datos.

Teniendo en cuenta que hay dos distintos conjuntos de datos es lógico pensar que dos códigos serán necesarios para el Spark engine. Sin embargo, como varios pasos son comunes hacer un buen diseño inicial permite aprovechar la estructura de implementación para añadir o modificar los distintos pasos de forma directa. De este modo, el código necesario para procesar los datos que sólo requieren de percepción social -datos de activos- se puede aprovechar como base para el código encargado de los datos que requieren tanto la percepción social como las entidades relacionadas -datos de economía y de expertos-.

Más allá de detalles de implementación concretos que pueden variar, hay tres pasos clave que se deben especificar para poder entender en profundidad el motor de procesamiento: cómo se reciben los datos, cómo se agrupan para el almacenamiento y qué modelos NLP se aplican.

Recepción y agrupación de los datos

Estos dos pasos son los más relevantes de cara a la metodología que sigue el motor de procesamiento como parte de la fase de obtención y procesamiento de los datos del proyecto.

El framework escogido para el procesamiento de los datos impone ciertas restricciones sobre cómo se pueden recibir los datos, lo que en Spark se conoce como *input sources*. Para este proyecto se ha escogido utilizar una conexión con el Twitter engine mediante sockets, ya que la integración con *Kafka* supondría un incremento considerable en la complejidad de la conexión y los recursos necesarios y utilizar archivos intermedios supondría un coste adicional que a la larga podría ser elevado.

En caso de fallida o como plan alternativo, la opción de utilizar archivos sería la opción escogida dado el sistema de agrupación que se especificará a continuación para el almacenamiento. Recibir los datos agrupados en intervalos de tiempo determinado generaría un esquema similar a las agrupaciones que se hacen previas al almacenamiento en las bases de datos, por lo que la complejidad de la implementación sería inferior que la integración con Kafka.

Las agrupaciones que se hacen previas al almacenamiento son dependientes de las fases posteriores -el almacenamiento y la visualización-, concretamente qué intervalo de tiempo se espera entre los datos. La visualización -especificada más adelante- debe ser interactiva y permitir aumentar o disminuir el eje temporal de los datos, por lo que se debe encontrar un equilibrio entre una ventana de agrupación pequeña -es decir, mucha precisión- pero sin suponer una cantidad de datos a almacenar desproporcionada.

Teniendo este equilibrio en mente, se ha valorado el producto TopStonks donde se muestra un nuevo punto de datos cada 30 minutos (figura 6), pero es un intervalo demasiado grande que sería más apropiado en caso de utilizar archivos como *input source*. A pesar de ser considerado demasiado grande, nos permite valorar que un intervalo demasiado pequeño puede ser contraproducente al tener que almacenar muchos datos que no son relevantes.

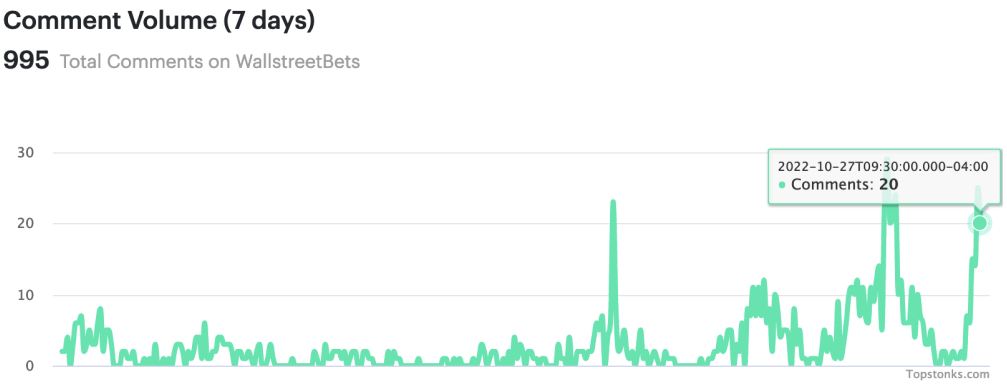

Figura 6: Gráfico de TopStonks sobre el volumen de comentarios de AAPL

Por lo tanto, se ha considerado un intervalo de 5 minutos entre puntos de datos, aportando una mayor precisión sin añadir una cantidad de puntos desproporcionada. En intervalos de 5 minutos -"ventanas" temporales de 5 minutos- los tweets recibidos se agrupan por activos o conceptos y entidades y se obtienen tanto la percepción media del intervalo como la cantidad de tweets considerados al hacer la media.

Elección de los modelos NLP

El tercer paso clave del motor de procesamiento es la elección de los modelos de *Natural Language Processing* que se aplican. Esta elección tiene dos impactos directos: los recursos necesarios para poder ejecutarlos y la calidad de la predicción, es decir, de los datos utilizados.

Al utilizar el framework Spark hay una serie de recursos adicionales que se pueden utilizar de forma complementaria para aplicar modelos de *Machine Learning* en Spark, como es el caso de John Snow Labs [17]. Esta compañía ofrece de forma gratuita el framework *Spark NLP*, especializado en aplicar modelos de NLP en Spark. Dentro de los diferentes modelos y *pipelines* que ofrecen [18] destacan las de uso médico, pero hay también de uso genérico que a priori pueden ofrecer mejores predicciones al aplicarlas sobre los datos del motor de búsqueda. En la tabla 1 se especifican las características de las pipelines analizadas.

Nombre	**Tamaño**	**Valoración**	
		Predicción	**Tamaño**
analyze_sentiment	25 MB	Razonable	Muy bueno
analyze_sentimentdl_use_twitter	937 MB	Buena	Desproporcionado
Combinación de **sent_bert_base_cased** y **sentimentdl_use_twitter**	438 MB	Buena	Demasiado grande
finpipe_org_per_role_date_en (NER)	828 MB	Buena	Desproporcionado
recognize_entities_dl (NER)	178 MB	Razonable	Mejorable

Tabla 1: análisis de las pipelines de Spark NLP escogidas

Teniendo en cuenta el tamaño de las pipelines, para la tarea de predicción NER *recognize_entities_dl* es la opción escogida ya que la diferencia de calidad en la predicción no se ve compensada por el incremento de tamaño.

En el caso de la tarea de predicción de sentimiento sucede algo similar ya que hay una pipeline con un tamaño entre 17 y 37 veces más pequeño que las alternativas. Sin embargo esto no debería ser motivo suficiente para escogerlo ya que la predicción del sentimiento determina la calidad de la información mostrada, por lo que es demasiado relevante como para no tenerla en consideración. En este caso, el salto cualitativo no es tan elevado como para compensar el incremento de tamaño, por lo que la pipeline escogida para la tarea de predicción de sentimiento es *analyze_sentiment*.

Hay que destacar que esta elección tiene en cuenta los recursos limitados que se disponen para el desarrollo de este proyecto. En entornos o proyectos con mayor cantidad de recursos se podrían llegar a valorar como opciones válidas pipelines que en este proyecto no se han escogido por su elevado tamaño.

6.5. Flujo de datos

Teniendo en cuenta los motores de búsqueda y procesamiento podemos establecer el flujo de datos de la figura 7, donde se destacan las dos principales características o ventajas de cada componente.

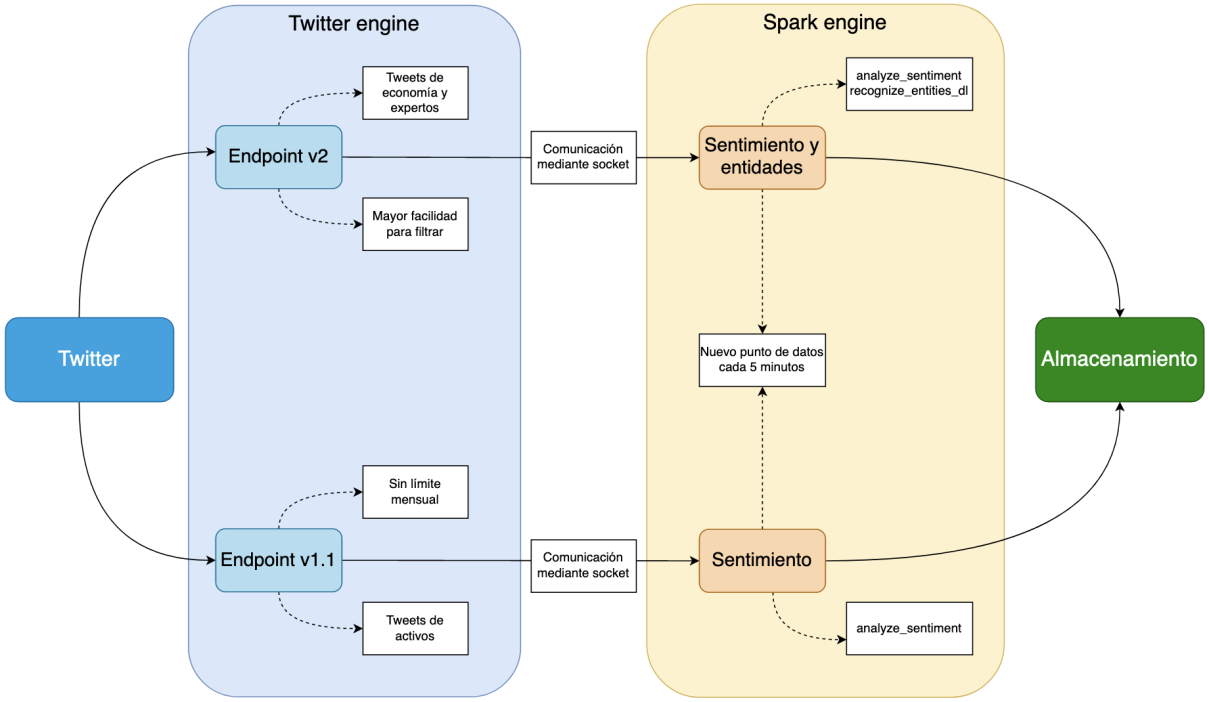

Figura 7: Flujo de datos entre el motor de búsqueda y el motor de procesamiento

7. Almacenamiento

Los datos obtenidos y procesados se deben almacenar en una base de datos para poder mostrarlos en la fase de visualización o utilizarlos con otros fines como ofrecerlos como producto a otras empresas. En esta segunda fase se plantea el diseño de la base de datos utilizada y se actualiza el flujo de datos. Las pruebas de rendimiento que se deben realizar en esta fase se encuentran en la siguiente sección.

Dada la componente temporal de los datos, es decir la importancia de cuándo se han creado, es razonable pensar que una base de datos especializada en series temporales puede tener mayor rendimiento que una base de datos convencional. Siguiendo el ránking de DB-Engines [19] se ha escogido InfluxDB como base de datos de series temporales (TSDB) por ser la más popular, por lo que no se consideran comparativas de rendimiento ante esta elección. A modo de alternativa en caso de que la base de datos principal falle durante su uso en un entorno de producción, se considera un lago de datos [20] -Data Lake en inglés- con un menor rendimiento a la hora de interactuar con los datos guardados pero con un coste considerablemente menor. El Data Lake escogido es *Lake Formation* de AWS.

La base de datos en InfluxDB se implementa sobre dos entornos o versiones diferentes: InfluxDB Cloud e InfluxDB Open Source [21]. La implementación en ambos entornos sigue el mismo diseño y estructuras de datos propuestos en los siguientes apartados pero difieren en el entorno de ejecución: InfluxDB Cloud se ofrece como servicio en la nube mientras que InfluxDB OSS se ejecuta en local con recursos propios.

En relación a las pruebas de rendimiento que se deben realizar sobre la implementación de las bases de datos, en caso de que las implementaciones en InfluxDB OSS e InfluxDB Cloud no tengan un rendimiento lo suficientemente bueno según los resultados de las distintas pruebas se contempla la posibilidad o alternativa de implementar bases de datos adicionales distintas a InfluxDB. En caso de requerir de estas implementaciones adicionales se mantendrá el diseño planteado a continuación para garantizar que las pruebas realizadas -y como consecuencia, la evaluación del rendimiento- sean equivalentes, lo cual permitirá comparar con rigor las diferencias entre alternativas.

7.1. Diseño de la base de datos

Antes de poder implementar la base de datos es necesario entender qué tipos de datos se deben guardar y qué estructura deben tener. Teniendo todo en consideración se podrá hacer el diseño de la base de datos.

Datos a almacenar

El primer punto para realizar el diseño es valorar qué datos se deben almacenar. En este caso, los datos vienen dados por los motores de búsqueda y procesamiento especificados previamente. El motor de búsqueda genera dos tipos de datos que almacenar: los propios tweets obtenidos y la cantidad de tweets leídos de cada activo o concepto; mientras que el motor de procesamiento genera datos únicamente sobre la percepción social.

El primer tipo de datos del motor de búsqueda no tiene un uso directo en la parte de visualización sino que su almacenamiento tiene valor como recuperación en caso de fallo del sistema e incluso como producto independiente que ofrecer a otras empresas. Además, la cantidad de tweets obtenida es considerablemente elevada por lo que almacenar los tweets en la BD supondría un coste elevado sin beneficios notables, por lo que estos datos son idóneos para almacenar en el Data Lake. El segundo tipo de datos del motor de búsqueda es más relevante de cara al almacenamiento en la BD ya que son datos importantes en la visualización. Estos datos reflejan el volumen de tweets obtenidos de cada topic por lo que ayudan a contextualizar los datos obtenidos del motor de procesamiento.

Los datos obtenidos del motor de procesamiento muestran la percepción social sobre los tweets y son muy similares entre los dos componentes que forman el Spark engine. Sin embargo, son lo suficientemente diferentes como para que se deban separar en dos subconjuntos según sean datos relacionados con activos o con la economía. Esta diferencia viene determinada por la aplicación de un modelo de ML para la obtención de entidades sobre los tweets de economía.

Dado el bajo coste del lago de datos, los datos almacenados en InfluxDB también se almacenarán por duplicado en el Data Lake como medida de prevención en caso de fallo de InfluxDB.

Estructura de almacenamiento

Teniendo en cuenta los datos a almacenar especificados podemos entrar en más detalle sobre qué atributos y variables contiene cada tipo de datos, con lo que generamos la estructura de almacenamiento.

Tweets obtenidos. Este primer conjunto de datos tiene tres principales componentes: el cuerpo del tweet obtenido, el grupo de expertos al que pertenece y una lista con los enlaces de las imágenes que contiene. Con estos tres atributos se puede mantener el anonimato de los usuarios sin entrar en conflicto con protección de datos a la vez que se puede identificar la clasificación de experto y, en caso de ser necesario, poder acceder a las imágenes adjuntas para nuevas funcionalidades como mostrar imágenes de cierto grupo de expertos a los usuarios. La estructura resultante se puede ver en la figura 8.

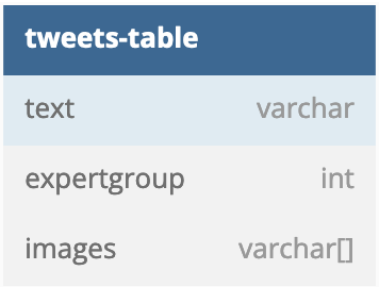

Figura 8: Estructura de almacenamiento de los tweets obtenidos

Volumen de tweets. Este segundo conjunto de datos tiene tres principales componentes junto con la marca temporal que indica el momento de su creación: el topic utilizado en la búsqueda, la cantidad de tweets obtenidos cada 5 minutos y un indicador según el topic pertenece al conjunto de activos o por el contrario es de economía. Este valor booleano tiene el objetivo de facilitar la separación de los datos en la visualización. La estructura de estos datos es la siguiente:

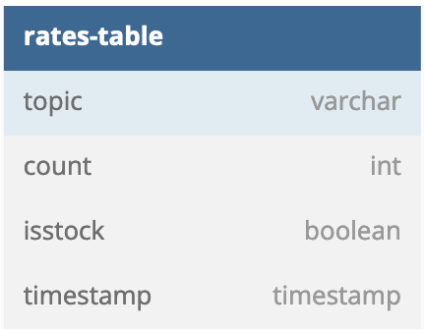

Figura 9: Estructura de almacenamiento del volumen de tweets

Sentiment de activos. Este tercer conjunto de datos tiene cuatro principales componentes junto con la marca temporal que indica el momento de su creación: el activo identificado en el tweet, la cantidad de tweets considerados en la agrupación, el sentiment medio de la agrupación y la clasificación del grupo de experto del tweet original. A diferencia del conjunto anterior, este tipo de datos sólo puede estar relacionado con los activos por lo que no es necesario un identificador para diferenciar entre activos y economía. La estructura de estos datos es la siguiente:

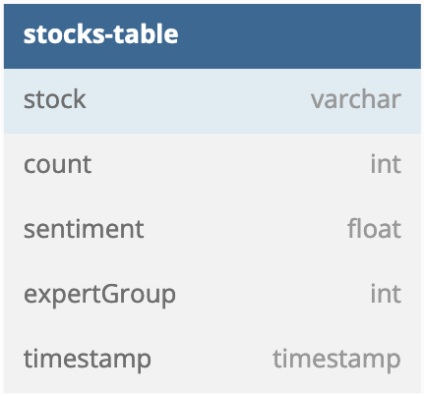

Figura 10: Estructura de almacenamiento del sentiment de activos

Sentiment de economía. Este cuarto y último conjunto de datos tiene cinco principales componentes junto con la marca temporal que indica el momento de su creación: el concepto económico identificado en el tweet, la cantidad de tweets considerados en la agrupación, el sentiment medio de la agrupación, la entidad obtenida identificada en el tweet y la clasificación del grupo de experto del tweet original. La estructura por lo tanto es muy similar a la del conjunto anterior, con la diferencia de que el identificador principal pertenece a conceptos de economía y se añade un atributo para identificar las entidades obtenidas al aplicar el modelo de ML mencionado anteriormente:

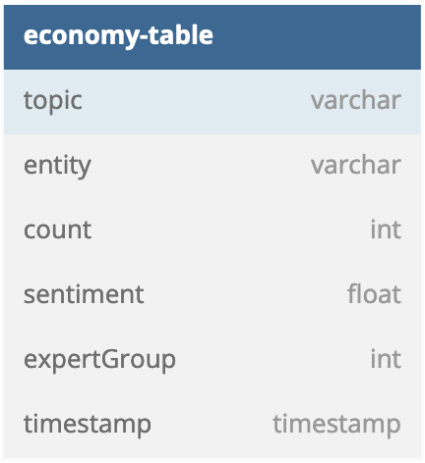

Figura 11: Estructura de almacenamiento del sentiment de economía

Adaptación a InfluxDB

Las estructuras propuestas son útiles para el Data Lake y para bases de datos relacionales pero se deben adaptar para su uso en InfluxDB. A diferencia de bases de datos convencionales, InfluxDB se organiza en *buckets* y *measurements* [22, 23], donde los buckets son similares al almacenamiento global de la base de datos mientras que los measurements tienen un papel más similar al de las tablas.

Teniendo en cuenta que la tabla tweets-table y su contenido se almacenan únicamente en el Data Lake, es necesario redefinir la estructura de los datos de tablas a measurements. La conversión es muy similar pese a tratarse de estructuras de almacenamiento distintas, lo cual genera los tres measurements de la figura 12.

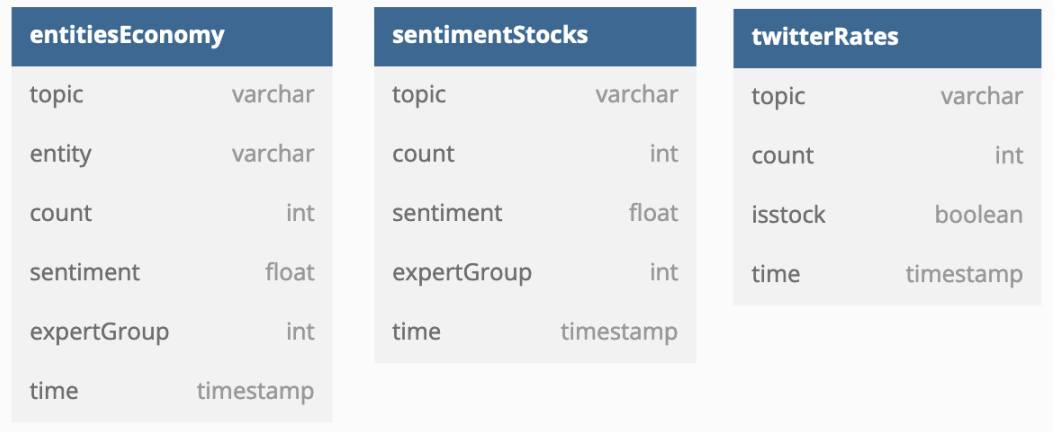

Figura 12: Conversión directa de las estructuras de datos

Sin embargo, debido a facilidades en el diseño de las queries con Flux se plantea un segundo diseño de los measurements en el que las dos primeras tablas -entitiesEconomy y sentimentStocks- se mantienen igual y la tabla de los rates de Twitter se divide en dos según el dato pertenezca al conjunto de activos o criptomonedas o por el contrario forme parte del conjunto de conceptos económicos. Los measurements alternativos se pueden observar en la figura 13.

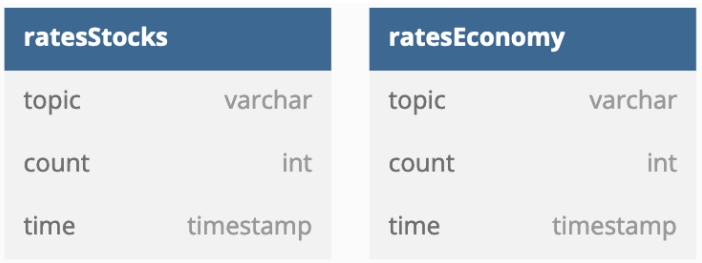

Figura 13: Conversión alternativa de la tabla rates-table

Ambas alternativas de diseño deben ser valoradas en las pruebas de rendimiento para determinar qué arquitectura ofrece un mejor rendimiento, lo cual puede ser crítico al escalar la aplicación.

7.2. Flujo de datos

Una vez definidos los datos a almacenar y sus distintas estructuras se puede actualizar el esquema de flujo de datos de la figura 7 para incluir esta segunda fase de almacenamiento.

Dentro de la capa de almacenamiento explicada a lo largo de esta segunda fase es necesario distinguir entre la base de datos con la que se interactúa para la visualización y el lago de datos. Hay que recordar que la mayoría de los datos se guardan duplicados como medida de prevención ante fallos de la base de datos, pero no todos los datos son necesarios para la visualización, por lo que las relaciones entre los componentes y ambos métodos de almacenamiento no son idénticas. En la figura siguiente se pueden observar los nuevos flujos resultantes con el diseño de la segunda opción planteada, es decir, la tabla rates-table dividida en dos measurements distintos.

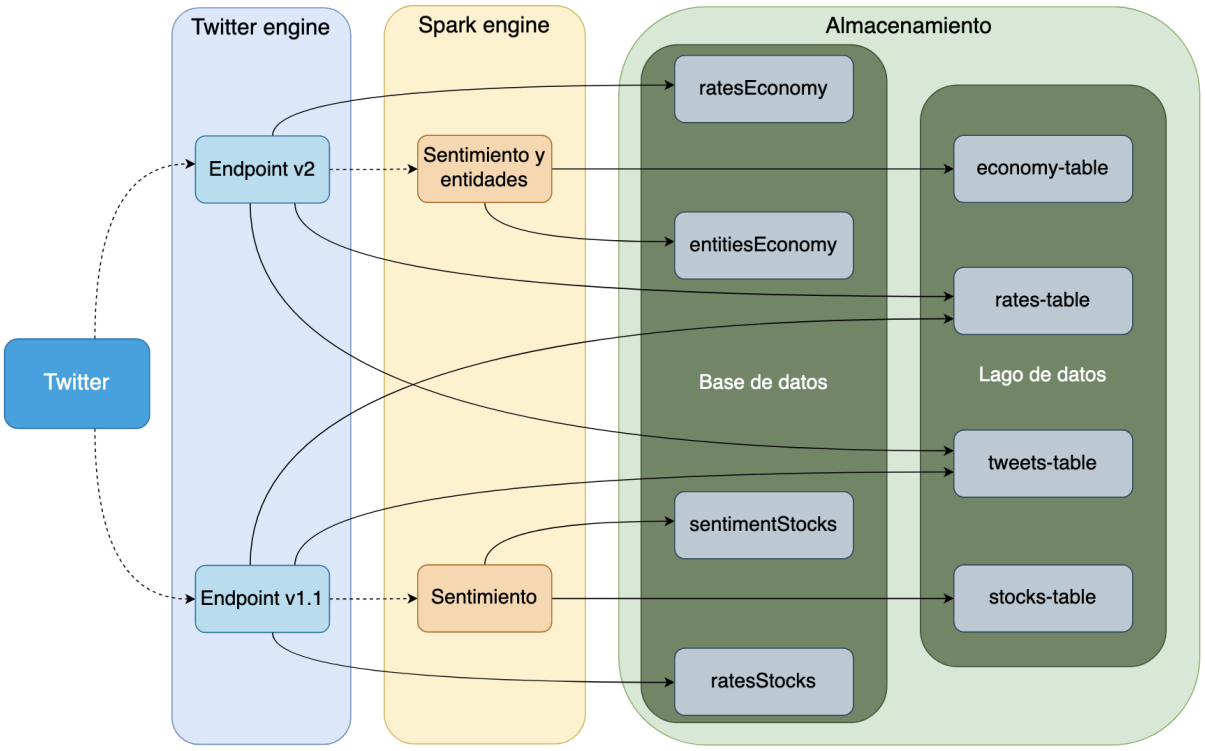

Figura 14: Flujo de datos entre los componentes de la primera fase y las bases de datos planteadas

8. Pruebas de rendimiento

Una vez hecho el diseño de los diferentes sistemas de almacenamiento necesarios se deben medir las características principales de las bases de datos para poder valorar su rendimiento, lo que en inglés se conoce como *benchmarking*.

Las siguientes pruebas de rendimiento no consideran el lago de datos ya que por definición el lago de datos es un sistema de almacenamiento que prioriza el respaldo de datos a un coste menor a cambio de un sacrificio considerable en las operaciones que interactúan con los datos almacenados. Dentro de la elección de InfluxDB como base de datos hay distintas opciones de implementación, donde destacan InfluxDB Cloud e InfluxDB OSS, y es la elección entre ambas implementaciones el principal objetivo de las pruebas especificadas a continuación.

Tal y como se ha especificado anteriormente, en caso de que las métricas obtenidas para valorar el rendimiento de las implementaciones no sea satisfactorio o suficiente como para utilizar InfluxDB para el sistema, se contempla realizar un subconjunto de las pruebas propuestas manteniendo el diseño de la implementación y de la base de datos con el objetivo de utilizar un método de almacenamiento con un rendimiento aceptable u óptimo.

8.1. Diseño de las pruebas

Para poder diseñar las pruebas que se realizarán es necesario definir bien cuál es el propósito de hacer el benchmarking y qué partes se van a testear y cómo [24, 25, 26]. Teniendo en cuenta estos dos puntos clave se puede definir qué clase de benchmark se realiza, lo cual en conjunto permite que otras entidades puedan utilizar los resultados como referencia en sus propias pruebas. Tal y como se define en el artículo de *HotStats* [27] sobre los tipos de benchmarking, el benchmark que se realizará es un benchmark interno ya que se compararán dos implementaciones de una misma base de datos en diferentes entornos pero no se valorará en relación a estándares del sector.

Objetivo

El objetivo de las pruebas que se especifican a continuación no es únicamente la elección de una base de datos para el desarrollo de este proyecto sino que también incluye comprobar la correctitud del diseño de la base de datos y medir el rendimiento del sistema para comprobar su escalabilidad.

La comprobación de la correctitud del diseño de la base de datos se centrará en la elección de separar la tabla twitter-rates (figura 12) en dos tablas diferentes: rates-economy y rates-stocks (figura 13). A diferencia de la implementación en una base de datos relacional -donde diferenciar entre unos datos u otros es tan fácil como añadir la condición "WHERE isstock" o su negación-, la implementación con InfluxDB tiene una complejidad mayor al basarse en Flux y no en SQL para realizar las queries. Es por ello que la diferencia entre las dos adaptaciones del diseño puede llegar a ser crítico.

El rendimiento del sistema, por otro lado, tiene como principal valor comprender las limitaciones que tiene la base de datos. El rendimiento del sistema no sólo puede ser utilizado como argumento para la elección de la base de datos entre las dos alternativas presentadas sino que se puede utilizar como referencia para comparar con implementaciones de otras bases de datos como *PostgreSQL* [28] o *MongoDB*. Por lo tanto en caso de querer comparar la base de datos escogida con otros referentes del sector o realizar un benchmark externo, son las métricas obtenidas con este objetivo las que se deberían usar.

Metodología

La metodología seguida para realizar las pruebas es un factor clave a la hora de poder replicar los tests o exportarlos a otros ámbitos. En concreto, es necesario profundizar en los pasos realizados para llegar a los distintos objetivos así como las métricas o valores que permitirán valorar la correctitud del sistema [29, 30, 31].

En lo que respecta a pruebas realizadas se pueden identificar dos principales tests: un primer test para comprobar la correctitud del diseño y un segundo test para obtener las métricas necesarias. Teniendo en cuenta que las pruebas tienen como objetivo principal escoger entre InfluxDB Cloud e InfluxDB OSS las métricas se deberán obtener en ambos entornos, pero, al tratarse de una misma base de datos con dos implementaciones distintas, es razonable pensar que la prueba de diseño se pueda limitar a uno de los dos entornos ya que no hay indicios que puedan justificar una diferencia notable.

El primer test -denominado *Architecture test*- es un test de poca complejidad con el objetivo de comprobar si las decisiones tomadas en la fase de diseño de los measurements/tablas son correctas. En concreto, el test se centra en la separación de la tabla twitter-rates en dos

distintos measurements. Para comparar ambas arquitecturas se utiliza un mismo conjunto de datos almacenados y una serie de queries equivalentes para ambos diseños; midiendo en cada ejecución el tiempo de respuesta y el tamaño de los datos obtenidos. Para obtener un valor más fiel a un entorno de producción y evitar a la vez ruido producido por la congestión de la red o factores externos no controlables cada query se repite un número determinado de veces y se utiliza como valor representativo la media de todas las ejecuciones.

El segundo test -denominado *Metrics test*- es un test de complejidad notoria con el objetivo de obtener una serie de métricas que puedan utilizarse como referencia para comparar distintas implementaciones de bases de datos. En lo que respecta a métricas específicas de bases de datos no hay un consenso establecido sobre cuáles utilizar como referencia, por lo que se pueden encontrar desde pruebas muy concretas de rendimiento como el uso de la CPU por operación o la cantidad de lecturas en memoria por operación hasta pruebas de carácter más general como la cantidad de operaciones por segundo que se pueden realizar. Una de las limitaciones de utilizar una implementación basada en la nube -InfluxDB Cloud- es que no se puede acceder a los recursos que dedica el sistema operativo, lo cual limita pruebas muy concretas sobre el funcionamiento interno de la base de datos. Por este motivo, las dos métricas que se obtendrán son de carácter general: la latencia y el throughput.

Para obtener ambas métricas se considera un mismo conjunto de datos almacenados en ambas implementaciones -InfluxDB Cloud e InfluxDB OSS- y una serie de operaciones tanto de escritura como de lectura con una complejidad variable. Para obtener cada métrica se coge de manera aleatoria un subconjunto de operaciones variando el porcentaje de tipo de operaciones y se ejecutan en orden aleatorio varias veces para obtener un valor medio de la métrica deseada. Variar el porcentaje de tipo de operaciones permite obtener una visión más genérica del comportamiento del sistema ante distintas situaciones, no se valora de forma exclusiva que el sistema reciba sólo operaciones de lectura o de escritura. La prueba de latencia se centra en la cantidad de segundos que tarda cada operación mientras que el throughput se centra en la cantidad de operaciones que se pueden realizar cada segundo. Como hipótesis previa a la realización de las pruebas se espera que la inversa del throughput sea considerablemente similar a la latencia.

8.2. InfluxDB Cloud

La primera parte de las pruebas de rendimiento se centra en la implementación de la base de datos en InfluxDB Cloud. En esta implementación se realizan ambos tests especificados previamente.

Architecture test

Los resultados obtenidos de realizar esta prueba muestran una ligera mejora en la arquitectura que separa la tabla de twitter-rates en dos measurements respecto a la arquitectura que utiliza un único measurement.

En la figura 15 se pueden apreciar los datos obtenidos para valorar ambas arquitecturas, el tamaño y el tiempo respectivamente. En ambos gráficos mostrados se muestra la diferencia numérica entre la implementación con dos measurements y un único measurement, por lo que un valor negativo indica que la arquitectura con dos measurements es mejor y un valor positivo indica lo contrario. Para valorar diferentes escenarios se utilizan 6 queries con características distintas: la obtención de todos los datos tanto de economía como de activos por separado como juntos y la obtención de datos que cumplan ciertas condiciones. El conjunto resulta en los 6 distintos valores que se muestran en ambos gráficos.

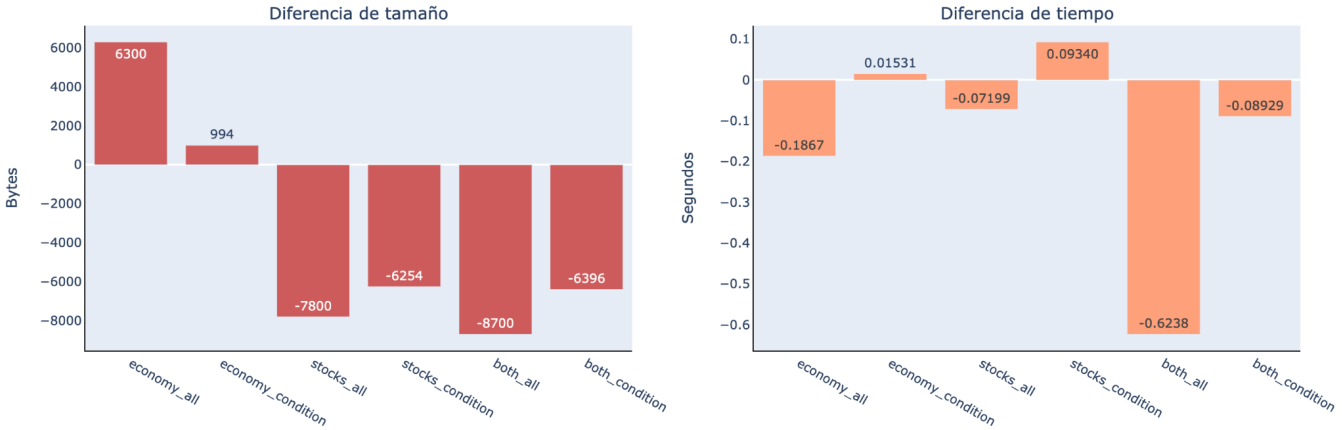

Figura 15: Diferencia de tamaño y tiempo entre ambas arquitecturas

A pesar de que los datos obtenidos en cada query son equivalentes en ambas arquitecturas, el tamaño utilizando un único measurement es generalmente superior a utilizar dos measurements distintos, lo cual se debe a la columna necesaria para indicar a qué tipo de datos pertenece el valor. Al considerar además el tiempo de ejecución, los resultados favorecen de nuevo a la arquitectura con dos measurements pero en este caso con una diferencia de menor relevancia.

Teniendo estas valoraciones en cuenta podemos concluir que la decisión de implementar la base de datos con dos measurements es favorable, pero los valores obtenidos no son lo suficientemente significativos como para que la diferencia entre arquitecturas suponga una mejora crítica sobre la implementación del diseño.

Metrics test

Los resultados obtenidos de realizar esta prueba son las métricas que se utilizarán para comparar y decidir entre implementaciones, por lo que sin valorar InfluxDB OSS no se pueden sacar conclusiones. Dentro de esta prueba se obtienen dos métricas distintas, el throughput y la latencia, por lo que para valorarlos se deben analizar por separado.

En la figura 16 se puede observar la evolución del throughput -es decir, la cantidad de operaciones por segundo- variando el porcentaje de número de lecturas respecto al número de escrituras. De los diferentes valores cabe destacar la evolución del número de operaciones según disminuye el porcentaje de lecturas, lo cual se traduce en que el tiempo de escritura es menor que el de lectura. El valor medio del throughput es de 2,845 operaciones por segundo.

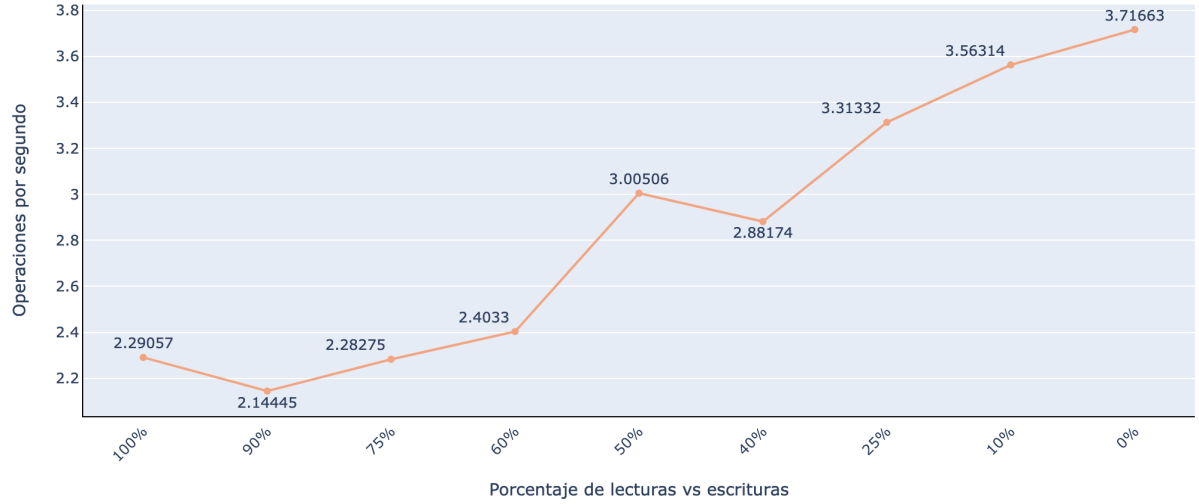

Figura 16: Valores del throughput variando el porcentaje de tipo de operación en Cloud

En la figura 17 se puede observar un gráfico similar a la figura 16 pero con los valores obtenidos de la latencia -es decir, la cantidad de segundos por operación-. Tal y como se podría esperar, la evolución es inversa al throughput, con un valor menor según disminuye el porcentaje de operaciones de lectura. Esto confirma la afirmación anterior de que las operaciones de escritura son más rápidas que las de lectura. El valor medio de la latencia es de 0,3563 segundos por operación.

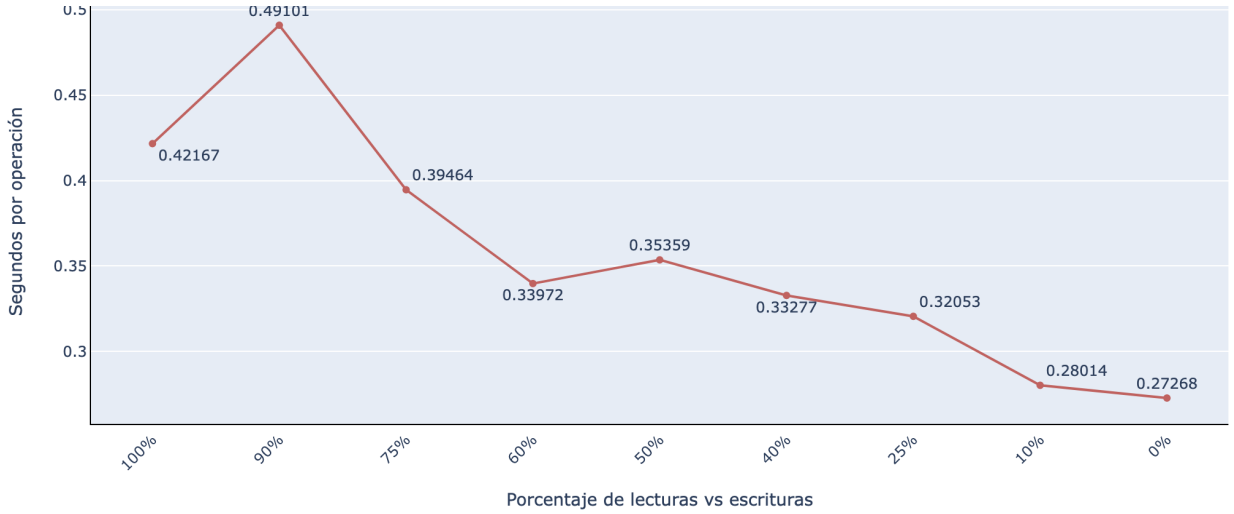

Figura 17: Valores de la latencia variando el porcentaje de tipo de operación en Cloud

Una vez valorados el throughput y la latencia por separados es necesario comprobar la hipótesis planteada anteriormente sobre la relación entre ambas métricas. Esta relación no se puede hacer de manera directa dada la diferencia de unidades -operaciones/segundo y segundos/operación respectivamente-, por lo que se utiliza la inversa del throughput para establecer una unidad común, segundos/operación. La transformación resultante de aplicar la inversa se puede observar en la figura 18.

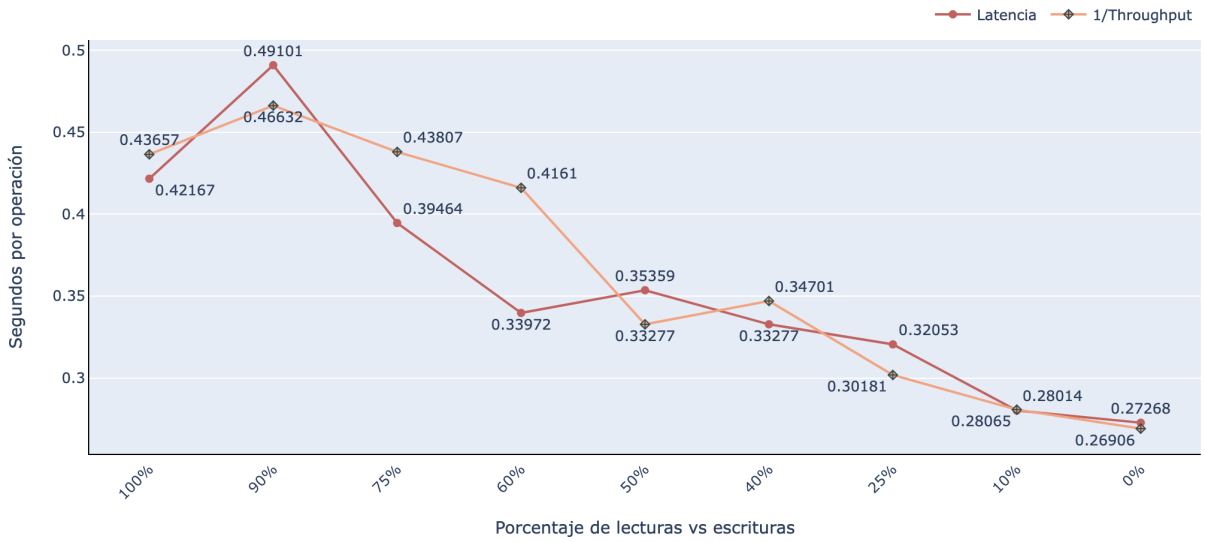

Figura 18: Relación entre latencia y throughput variando el porcentaje de tipo de operación en Cloud

A pesar de diferencias más notables en los porcentajes 75 y 60, se puede apreciar una similitud considerable entre los distintos pares de valores. Si tenemos en cuenta el valor medio, hay una estrecha relación de los valores -90 milésimas de segundo- que es despreciable a gran escala y se puede deber a pequeñas variaciones en el estado de la red.

8.3. InfluxDB OSS

La segunda parte de las pruebas de rendimiento se centra en la implementación de la base de datos en InfluxDB OSS. En esta implementación se realiza únicamente la prueba para la obtención de las métricas.

Metrics test

Al igual que en la implementación en InfluxDB Cloud, esta prueba se centra en obtener las métricas que se utilizarán para comparar implementaciones. El hecho de que esta implementación se realice en local da paso a la hipótesis de que las métricas serán mejores que en la implementación basada en la nube.

En la figura 19 se puede observar la evolución del throughput en la segunda implementación propuesta. La primera diferencia que se puede destacar es la diferencia en la escala de valores, con un valor máximo cercano a 60 operaciones por segundo. Incluso el valor más bajo -3,64 operaciones por segundo- es sólo inferior al valor máximo de la implementación previa por menos de una décima. Podemos avanzar, por lo tanto, que esta implementación es significativamente superior.

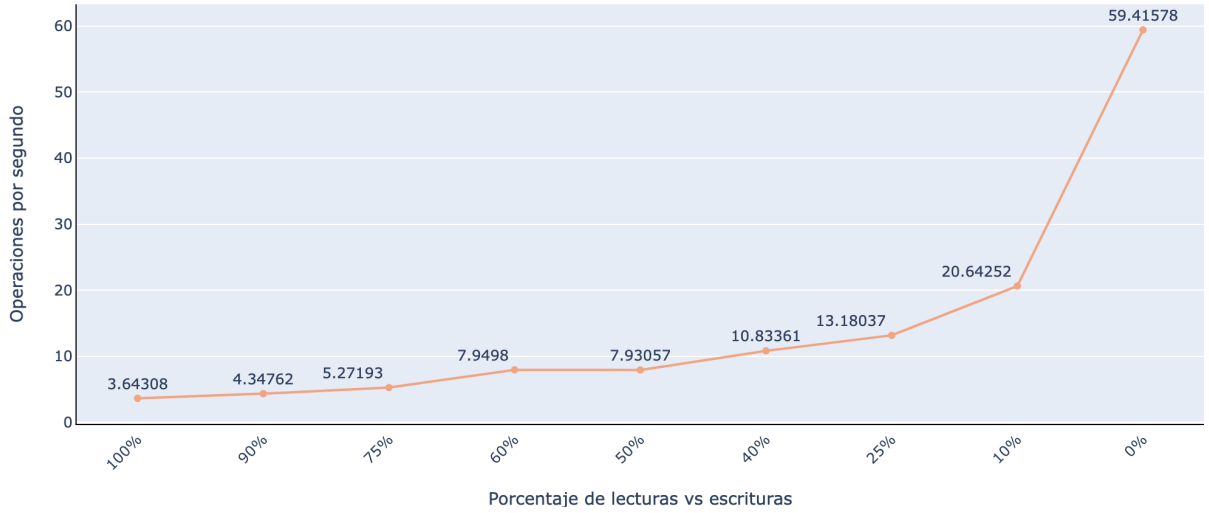

Figura 19: Valores del throughput variando el porcentaje de tipo de operación en OSS

Además de la diferencia en la escala de valores nos debemos centrar también en la evolución de los valores, donde se puede identificar una tendencia parecida a la implementación anterior: a menor porcentaje de lecturas, mayor número de operaciones por segundo se pueden hacer. Sin embargo, en este caso el incremento de las operaciones por segundo sigue una evolución exponencial, por lo que podemos intuir que las operaciones de lectura tienen

un coste temporal mucho mayor que las operaciones de escritura. Tal es esta diferencia que el caso en que todas las operaciones son escrituras es 16 veces superior al caso opuesto en que todas las operaciones son lecturas.

En la figura 20 se confirman las mejoras respecto a la implementación anterior, en este caso con la métrica de la latencia. De forma similar a lo que sucede con el throughput, todos los valores son menores que los de la figura 17, con una evolución lineal decreciente. Se puede apreciar de forma más clara la diferencia de tiempo entre operaciones; una operación de lectura tarda 22 centésimas de segundo mientras que una de escritura tarda una centésima y media de segundo.

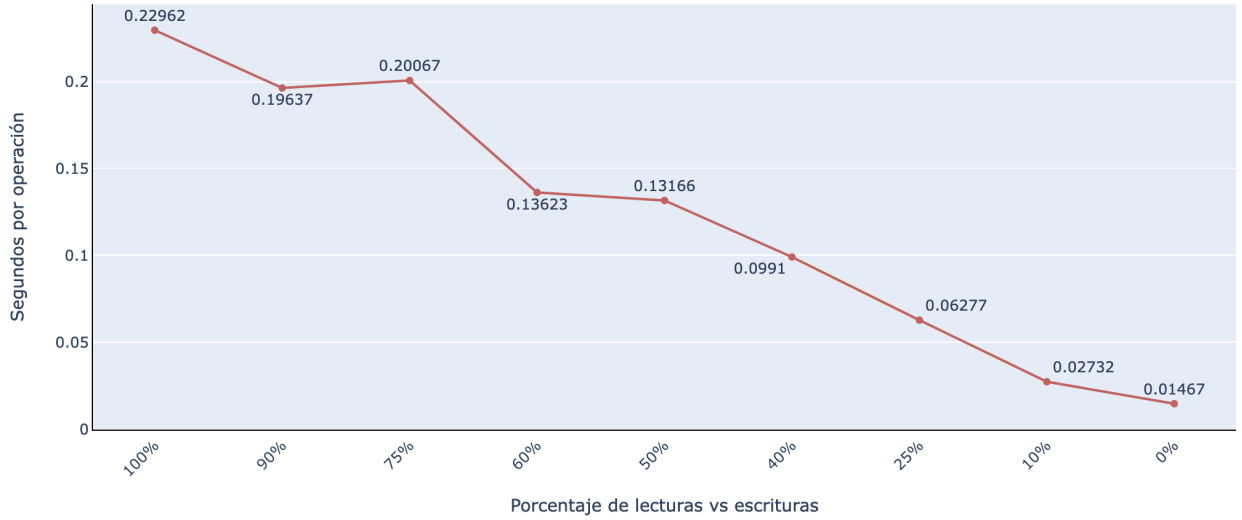

Figura 20: Valores de la latencia variando el porcentaje de tipo de operación en OSS

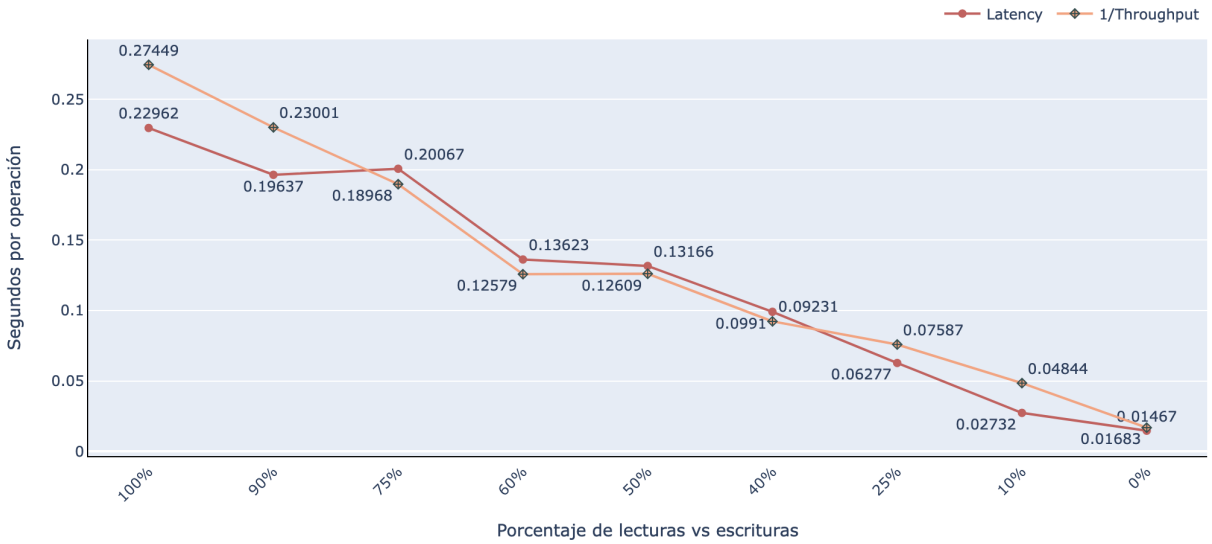

Figura 21: Relación entre latencia y throughput variando el porcentaje de tipo de operación en OSS

A pesar de que en la sección anterior se muestran indicios sobre la relación entre latencia y throughput -visibles en figura 18-, no son suficientes como para confirmar o refutar la

hipótesis planteada inicialmente. En la figura 21 se puede observar la relación entre los valores de la latencia y el throughput obtenidos en esta segunda implementación, en este caso, la relación entre los los distintos pares de valores por cada porcentaje es más estrecha que en la implementación basada en la nube. Por lo tanto podemos concluir que la hipótesis de que existe una estrecha relación entre la latencia y el throughput es cierta, en concreto una relación inversa.

8.4. Valoración de los resultados

Llegados a este punto, las diferencias entre ambas implementaciones son lo suficientemente significativas como para poder escoger una sobre la otra.

La implementación de InfluxDB OSS tiene un rendimiento mucho mayor en operaciones de escritura y considerablemente mejor en operaciones de lectura. La agrupación de estas métricas en la tabla 2 sirve para recordar los valores obtenidos y respaldar a la vez esta afirmación. En el caso de las operaciones por segundo, un mayor valor es favorable mientras que en los segundos por operación es favorable un valor más bajo.

Métrica		**InfluxDB Cloud**	**InfluxDB OSS**
Lecturas por segundo	QPS	2,291	3,643
Escrituras por segundo	IPS	3,717	59,42
Operaciones por segundo	OPS	2,845	14,80
Segundos por lectura	SPQ	0,4217	0,2296
Segundos por escritura	SPI	0,2727	0,01467
Segundos por operación	ART	0,3563	0,122

Tabla 2: recopilación de las métricas obtenidas en las pruebas de rendimiento

La diferencia entre implementaciones es más notable entre las operaciones de escritura -métricas IPS y SPI-, pero esta mejora no se traduce en una mejora similar en las lecturas -métricas QPS y SPQ- tal y como se podría esperar. Esta diferencia de rendimiento entre tipos de operaciones se debe a la base de datos escogida, InfluxDB, ya que tal y como indica la propia documentación [32] se debe hacer un balance entre la cantidad de datos y la precisión para tener un buen rendimiento.

Los datos utilizados en el benchmark corresponden a un extracto representativo de los datos de Finlooker recopilados aproximadamente durante una semana, por lo que son datos similares o idénticos a lo que se podría esperar en producción pero con un volumen mucho menor. Al ser una muestra representativa y haber aplicado las optimizaciones recomendadas, el rendimiento de las operaciones de lectura está lejos de una implementación deseable.

Hay que recordar que la elección de InfluxDB como implementación estaba basada en un ranking de popularidad y no de rendimiento, por lo que es coherente que los resultados obtenidos no sean los mejores, sobre todo en las lecturas. Ante esta situación se opta por realizar un conjunto de pruebas complementarias con las bases de datos *Timescale* [33] y *QuestDB* [34] tal y como se planteaba al inicio de la sección.

Exploración de alternativas: Timescale y QuestDB

Las pruebas que se realizan en estas dos implementaciones son variaciones de las pruebas especificadas en el diseño, con un foco en las operaciones que se consideran mejorables de InfluxDB: las lecturas.

Para garantizar que las nuevas métricas obtenidas son representativas y que la comparación con InfluxDB no está negativamente influenciada por el cambio de Flux a SQL -lenguajes para interactuar con InfluxDB y Timescale y QuestDB respectivamente-, se considera el mismo extracto de datos de Finlooker y un conjunto de operaciones de lectura equivalentes en ambos lenguajes. El diseño de las implementaciones es igual al especificado anteriormente con los resultados del architecture test -es decir, separar la tabla twitter-rates en dos tablas distintas- y la metodología seguida es idéntica a las pruebas de throughput con la diferencia de que sólo se obtiene la métrica QPS.

	Lecturas por segundo (QPS)
InfluxDB OSS	0,11
Timescale Cloud	0,54
QuestDB	6,75

Tabla 3: resultados de las pruebas complementarias en las distintas implementaciones

Los resultados de las pruebas muestran una mejora considerable con las nuevas implementaciones. Timescale Cloud tiene una métrica de lecturas por segundo 5 veces

superior a InfluxDB OSS, lo cual ya supone una implementación mejor que las anteriores, pero es en QuestDB donde la diferencia es más mayor. En las dos implementaciones valoradas previas a QuestDB las lecturas por segundo no llegan a la unidad, mientras que QuestDB la supera con creces.

Las alternativas planteadas suponen una gran diferencia en cuanto a las operaciones de lectura, pero escoger una base de datos sin tener en cuenta las escrituras sería una decisión errónea -similar a escoger InfluxDB OSS basándose únicamente en las métricas de escritura-. Sin embargo, el proceso de preparación de las pruebas requería almacenar los datos que obtener en las operaciones de lectura, y al hacer estas preparaciones se ha observado un rendimiento similar entre las tres opciones por lo que las operaciones de escritura tienen un rendimiento favorable.

Además, teniendo en cuenta la gran cantidad de datos que se deberán almacenar durante la vida útil de Finlooker, es relevante destacar que el peso de las operaciones de lecturas es superior al de las escrituras, por lo que dentro del rendimiento similar en las escrituras se prioriza aquella implementación con mejor rendimiento de lecturas.

Es por lo tanto la implementación con QuestDB la escogida para utilizar de cara a la visualización y al desarrollo como parte de Finlooker.

9. Visualización

Una vez realizado el segundo sub-objetivo y habiendo escogido e implementado una base de datos que utilizar para el almacenamiento podemos avanzar al tercer sub-objetivo: la visualización. A pesar de que esta fase pueda dar la sensación de ser menos compleja o de tener menor dificultad a la hora de desarrollarla o implementarla, es de hecho el sub-objetivo que da sentido a todo el proyecto.

Un motor de búsqueda y procesamiento puede ser muy potente o tener muchas funcionalidades, almacenar los datos generados mediante el motor permite poder accederlos en todo momento, pero es el uso que se le da a dichos datos lo que da más valor a la potencia o funcionalidades del motor de búsqueda y procesamiento. Esta relevancia se puede extrapolar a Finlooker como producto en desarrollo, donde los usuarios finales sólo interactúan con los datos generados mediante el motor de búsqueda y procesamiento y la base de datos desarrollada en este proyecto.

La visualización que se ofrece en Finlooker como producto es más compleja y por lo tanto tiene su propio equipo de desarrollo, pero la visualización que se presenta a continuación sirve como base para definir qué datos mostrar, su relevancia y qué método utilizar para que su comprensión sea óptima.

9.1. Datos a mostrar

Esta primera subsección de la visualización se centra en qué datos se muestran a los usuarios y cuál es su significado.

A modo de recordatorio del diseño de la base de datos y de su estructura de almacenamiento, hay tres tipos de datos repartidos en cuatro tablas:

- Resultados del motor de procesamiento de los datos de activos. Corresponde a la tabla stocks-table -figura 10-.
- Resultados del motor de procesamiento de los datos de economía. Corresponde a la tabla economy-table -figura 11-.
- Cantidad de tweets obtenidos mediante el motor de búsqueda. Corresponde a las tablas ratesStocks y ratesEconomy -figura 13-.

De los tres tipos de datos, la cantidad de tweets obtenidos es un indicador de la relevancia de cada activo o concepto, por lo que es la evolución de esta relevancia lo que debe ayudar al usuario a contextualizar la importancia de los datos generados mediante el motor de procesamiento. Respecto a los otros dos conjuntos de datos hay varias opciones que se deben valorar para entender los puntos fuertes de cada una pero para ello hay que tener en cuenta los distintos atributos de cada tabla.

En el caso de la tabla de activos, stocks-table, además del identificador del activo y de la timestamp del dato hay dos atributos relevantes para la visualización: la cantidad de tweets considerados y el valor medio de la percepción social al hacer la agrupación de 5 minutos. De estos dos valores, es el sentiment el que indica al usuario cómo de buena o de mala es la percepción que tienen los usuarios de Twitter respecto al activo del identificador, por lo que se puede considerar como el atributo de mayor relevancia.

Hay varias formas de mostrar la relación entre los diferentes activos y su sentiment, pero se pueden destacar 3 metodologías:

Valor actual. Esta primera metodología es muy sencilla y se basa en mostrar el valor más reciente del sentiment. Simplemente pone en relevancia el último valor obtenido sobre la percepción social, por lo que más allá de si la percepción es positiva o negativa no permite valorar cómo era previamente o detectar si se han producido cambios favorables.

Evolución del sentiment. Esta segunda metodología es similar a la planteada sobre la cantidad de tweets obtenidos pero ayudando a contextualizar el valor actual. Es con esta metodología con la que se puede valorar la evolución de los cambios en la percepción, lo cual puede ayudar al usuario a plantearse si es buen momento para invertir o vender, por ejemplo.

Distribución del sentiment. Esta tercera metodología es menos útil de cara a la contextualización del sentiment pero al mismo tiempo puede ayudar a entender mejor la evolución. Dado el pequeño rango de valores sobre los que trabaja el modelo de ML que se utiliza para obtener el sentiment es posible que en la evolución haya picos puntuales que destaquen respecto a la evolución que se podría esperar. Mostrar la distribución de los valores permite normalizar dichos picos y así poder entender mejor la evolución que se muestra mediante la segunda metodología.

En el caso de la tabla de los conceptos económicos, economy-table, aparece un nuevo atributo resultante de aplicar el modelo NER sobre los tweets. La naturaleza de estos datos tiene como objetivo contextualizar la situación económica a nivel general por lo que su función es informar al usuario de qué está pasando más allá de los activos mostrados. De este modo, aunque la percepción social sobre un activo sea muy positiva, si los tweets sobre inflación tienen una gran relevancia el usuario puede plantearse si realmente el precio del mercado corresponde a un precio justo o por el contrario es más elevado sin justificación aparente.

Ante la creciente normalización de jóvenes invirtiendo en activos o criptomonedas es probable que gran parte de los usuarios no tenga el conocimiento para juzgar por sí mismos la situación del mercado a nivel macroeconómico, ya sea por falta de experiencia o de educación económica. Es por esta razón que las entidades obtenidas mediante el modelo ML tienen una relevancia considerable, permiten valorar de forma rápida qué está causando una percepción más positiva o negativa sobre los conceptos económicos. Estas entidades no pretenden tener la misma calidad o rigor que notícias específicas sobre la economía pero sí permiten una aproximación inicial al usuario.

Teniendo en cuenta la relevancia de las entidades, además de las tres metodologías valoradas sobre los datos de activos -con el cambio de activos a conceptos económicos-, aparece una nueva metodología con el objetivo de mostrar la relevancia de las distintas entidades. Esta metodología es similar al valor actual del sentimiento pero mostrando por cada concepto económico las distintas entidades más relevantes identificadas.

9.2. Gráficos generados

Una vez valorados qué datos mostrar y diferentes metodologías para facilitar su comprensión llega el momento de definir qué gráficos se generan, es decir, la conversión de las distintas definiciones a la visualización propiamente dicha.

Antes de ver los gráficos generados es necesario destacar que el estilo y la visualización son dependientes de la implementación, es decir, utilizar una u otra librería de Python puede generar gráficos estilísticamente completamente distintos pese a mostrar los mismos datos. Para el desarrollo de este proyecto se han utilizado las librerías Plotly [6] y Wordcloud [35]

para generar los gráficos y las librerías Dash y Dash-bootstrap-components para agruparlos bajo una visualización en formato de página web. Los gráficos mostrados pertenecen a la versión final de dicha página web y, tal y como se adelantaba en la sección de justificación del proyecto, se han utilizado como referencia de diseño ejemplos disponibles en la galería de Dash Enterprise [36] y el feedback de potenciales usuarios -tarea T3.4, especificada más adelante-.

Cantidad de tweets

Los dos primeros gráficos corresponden a la evolución de la cantidad de tweets de los conceptos de economía y de los activos respectivamente -figuras 22 y 23-. En ambos casos se puede identificar la lista con todos los parámetros de búsqueda, bajo el eje horizontal en la figura 22 y en el margen derecho en la figura 23. Gracias a la implementación con Plotly se pueden seleccionar elementos de las listas para que se muestren de forma única, lo cual facilita su comprensión.

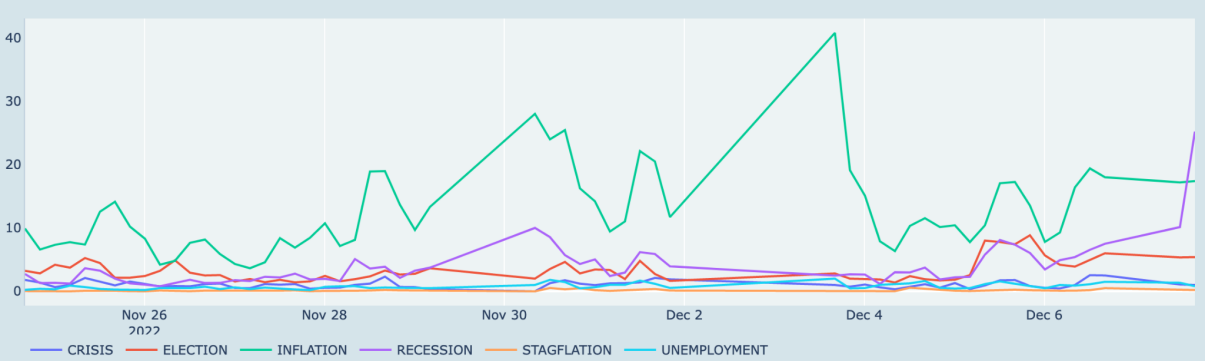

Figura 22: Evolución de la cantidad de tweets de economía

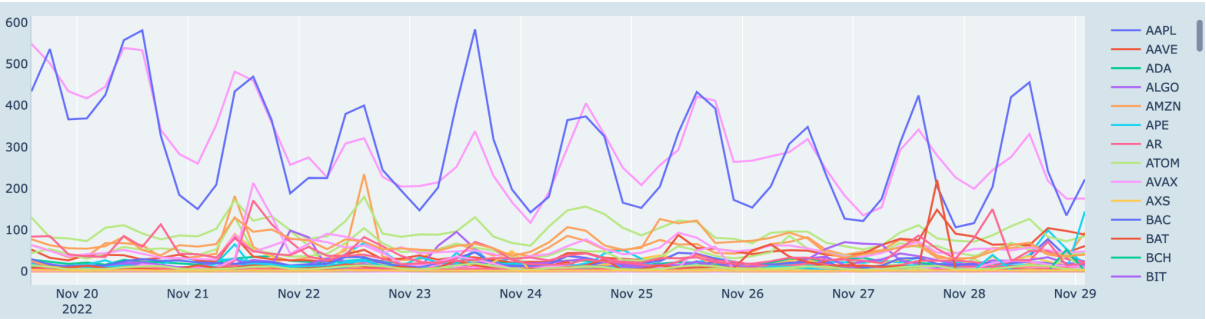

Figura 23: Evolución de la cantidad de tweets de activos y criptomonedas

Dentro de estos gráficos es destacable las variaciones según la hora del día, con valles alrededor de medianoche y máxima actividad al mediodía. Esta evolución está más presente en aquellos activos de mayor relevancia, como por ejemplo Solana y Ethereum en la figura 23 -Solana (SOL) en azul y Ethereum (ETH) en rosa-.

Datos de activos

Dentro de este tipo de datos ya se han mencionado previamente las distintas metodologías posibles para mostrar los datos. De estas metodologías, a la vez, hay variedades a la hora de representar los datos manteniendo el significado.

Para poder apreciar de forma más visual la relación entre cantidad de tweets utilizados y sentimiento se ha optado por utilizar un *treemap* con las siguientes características: el tamaño de cada celda u hoja corresponde a la cantidad de tweets y su color corresponde a la percepción social. Una percepción positiva tiene un color verde más intenso mientras que una percepción negativa tiene un color rojo más intenso y una percepción neutra corresponde a un color blanquecino. Al pasar el ratón por encima de cada celda se puede observar la cantidad de tweets y el valor concreto del sentiment tal y como se muestra en la figura 24 con TSLA.

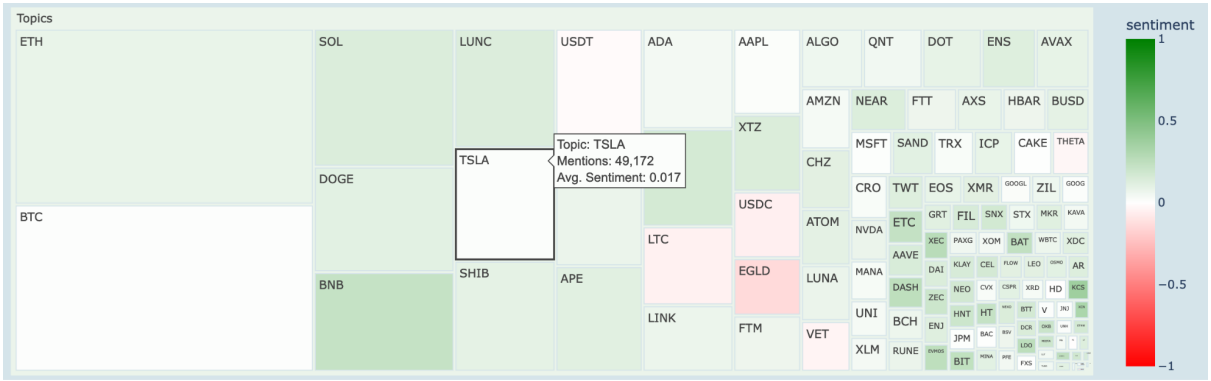

Figura 24: Treemap de la cantidad y el sentiment de los activos

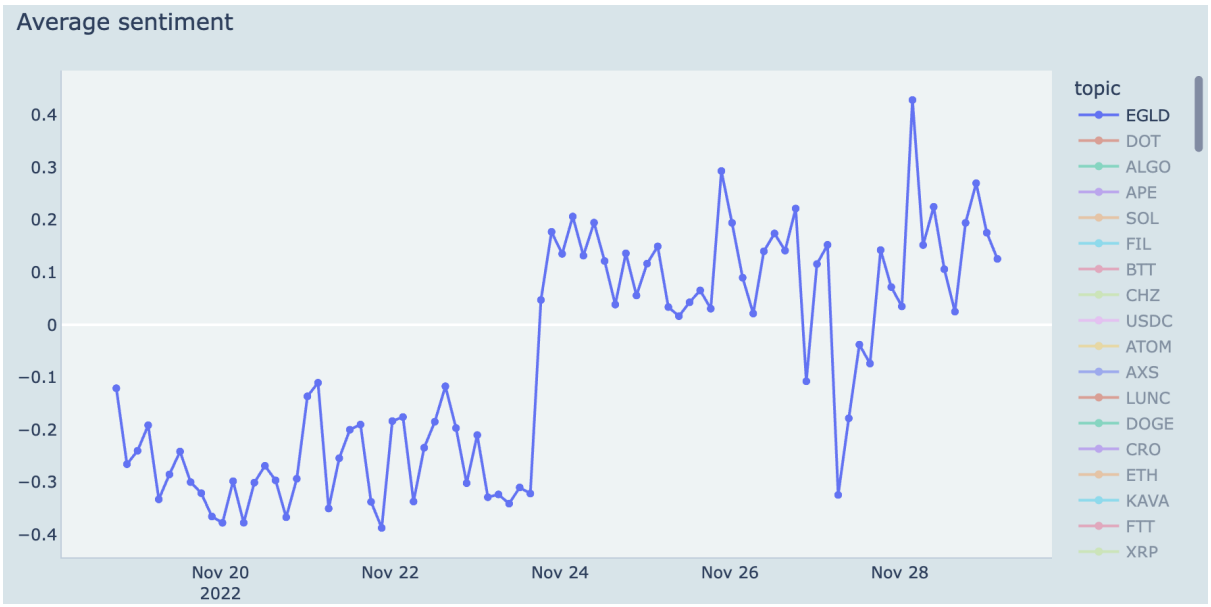

Figura 25: Evolución del sentiment de la criptomoneda Elrond eGold (EGLD)

En la figura 25 se puede observar la evolución del sentiment con el paso del tiempo. Al igual que en la evolución de la cantidad de tweets obtenidos sobre activos se puede observar también la lista de activos en la zona derecha, con la diferencia de que el gráfico de la figura 25 muestra un ejemplo de la selección de un único activo. El activo seleccionado -en este ejemplo Elrond eGold (EGLD)- destaca sobre el resto de activos y criptomonedas para poder identificar mejor qué se muestra. Como alternativa a la evolución del sentimiento, el usuario también puede observar la evolución de la cantidad de tweets considerados y la distribución del sentiment -figuras 26 y 27 respectivamente-.

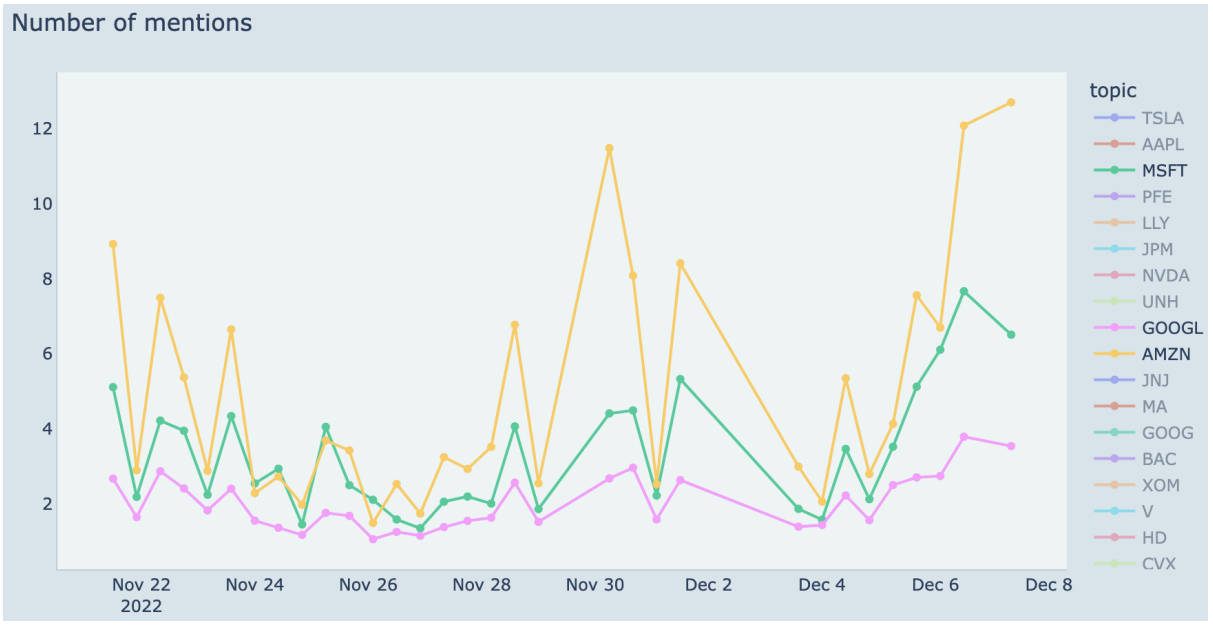

Figura 26: Evolución de la cantidad de los activos de Microsoft (MSFT), Google (GOOGL) y Amazon (AMZN)

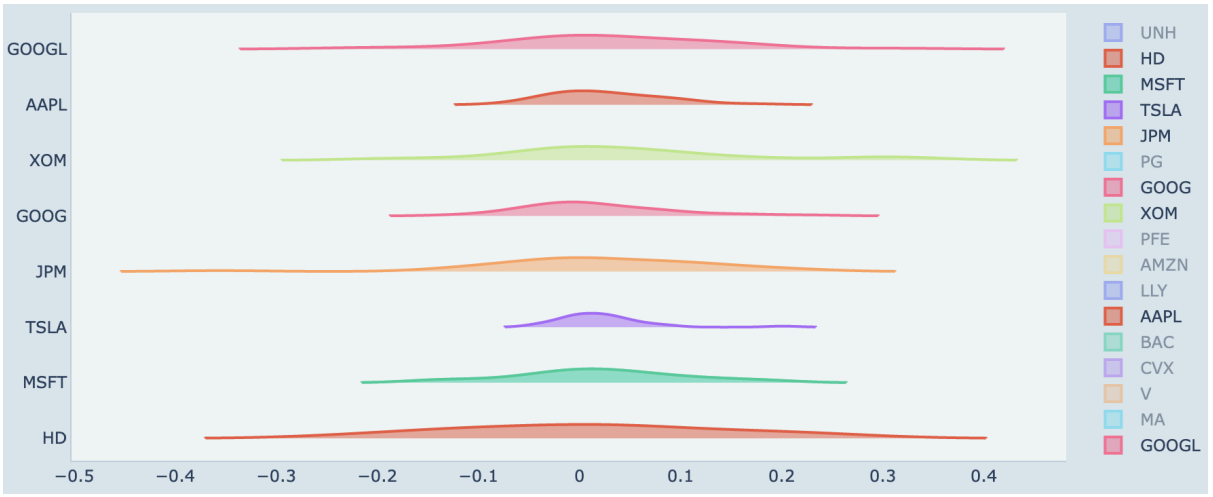

Figura 27: Distribución del sentiment de varios activos de empresas

Datos de economía

Tal y como se ha especificado, los datos de economía se pueden visualizar con las mismas metodologías que los datos de activos por lo que los usuarios tienen disponibles los gráficos de las figuras 24, 25, 26 y 27 de forma equivalente para los conceptos de economía. La figura 28 es equivalente a la 25 por ejemplo, destacando '*election*' en vez de EGLD.

Figura 28: Evolución del sentiment del economy concept 'Election'

Sin embargo, al treemap de la figura 24 se le añade un nuevo nivel con las entidades, es decir, al clicar sobre uno de los conceptos de economía que se muestra inicialmente de forma similar a la figura 24 se carga un segundo nivel con las distintas entidades y la percepción de los tweets en los que se han identificado. A menor relevancia de la entidad menor es su tamaño y su texto, pero al pasar el ratón por encima se pueden ver con claridad los valores exactos como se podía apreciar en la figura 24 con el activo TSLA.

Figura 29: Treemap de la cantidad y el sentiment de las entidades de 'RECESSION'
(segundo nivel del treemap)

Además de los gráficos equivalentes con los datos de activos, para mostrar las entidades se añade el segundo nivel del treemap ya especificado y se añade una nueva visualización en formato de *word cloud* o nube de palabras. Esta nueva visualización permite identificar las entidades según su relevancia dentro de cada concepto de economía tal y como muestra la figura 30. A diferencia del treemap, en este caso no se muestra la percepción social por lo que los colores únicamente sirven para identificar las distintas entidades, no tienen más significado. En el siguiente ejemplo se muestran los wordclouds de los conceptos de crisis y elección: dentro de 'CRISIS' se pueden apreciar varios países como Ucrania, China y Rusia mientras que en 'ELECTION' se pueden identificar Georgia, Arizona e India por ejemplo; lo cual puede ayudar al usuario a entender mejor en qué contexto se han utilizado los conceptos de economía.

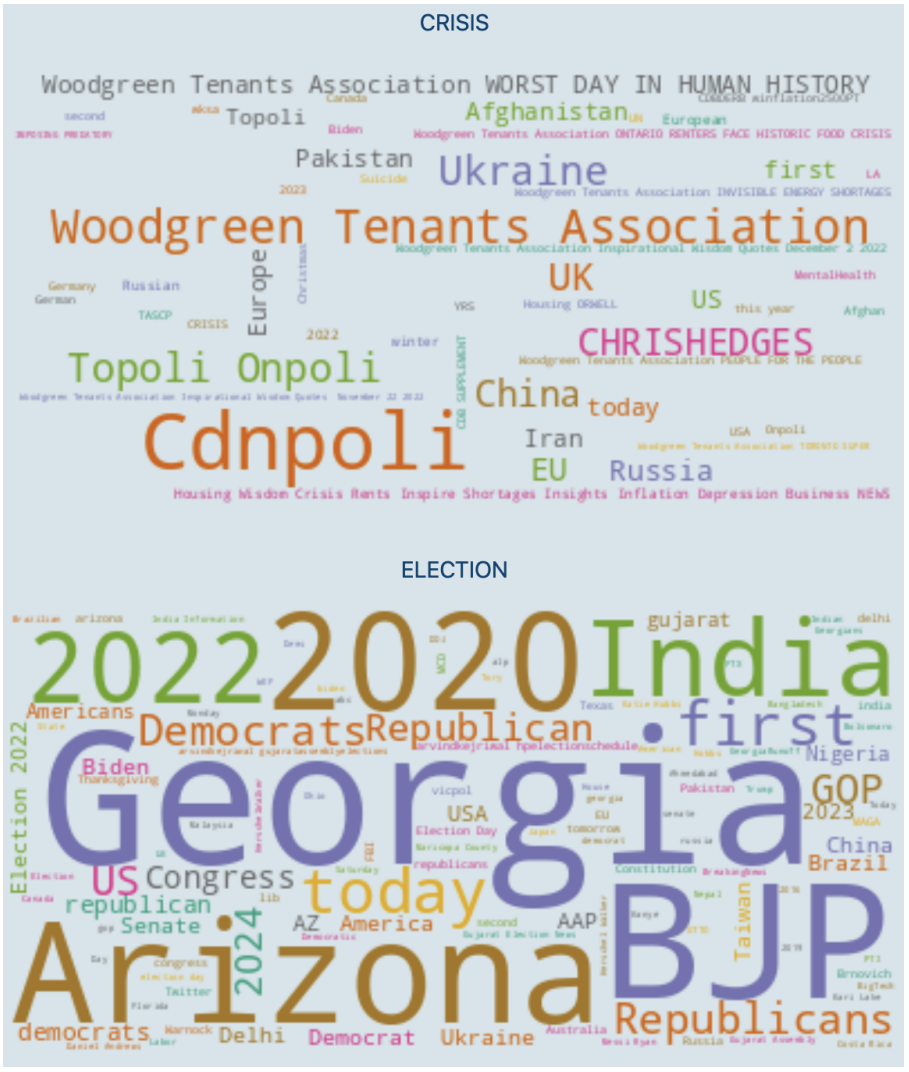

Figura 30: Nubes de palabras de los conceptos 'CRISIS' y 'ELECTION'

9.3. Página web

Considerando los gráficos generados para cada tipo de datos se ha desarrollado una página web en la que el usuario puede interactuar y personalizar ligeramente los gráficos que se muestran y su contenido. Para actualizar los datos que se muestran en un gráfico se ha añadido un botón en la cabecera de la página y así evitar tener que refrescar la página y perder los filtros ya añadidos tal y como se puede observar en la figura 31.

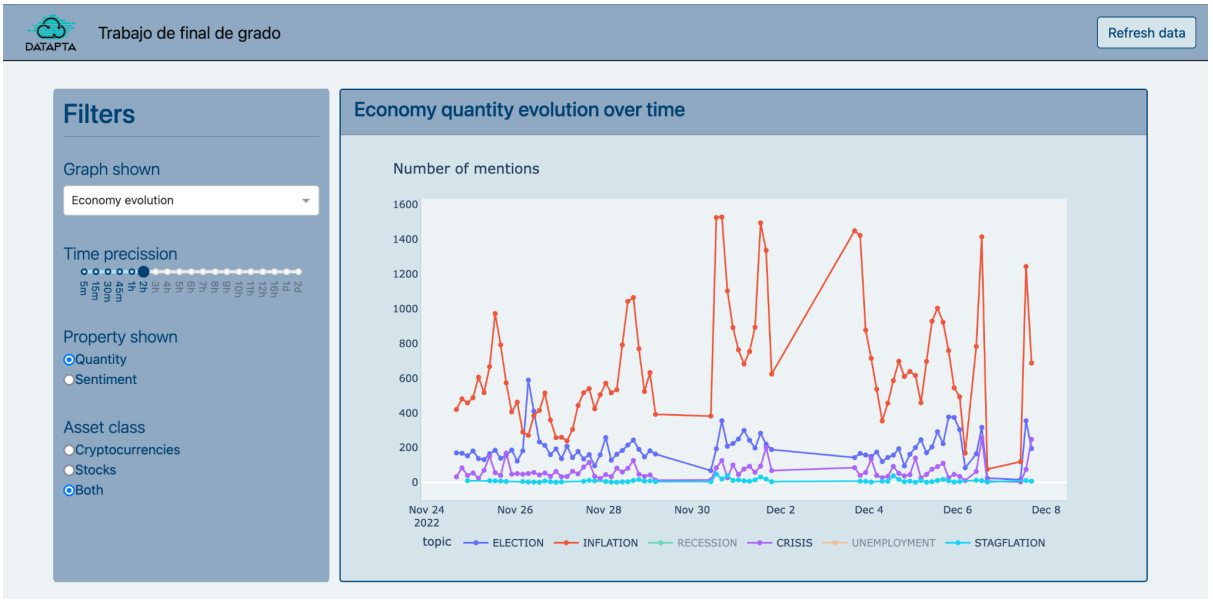

Figura 31: Captura de la página web implementada - sección 1

Dichos filtros se encuentran en la columna de la izquierda de la figura 31 y abarcan desde las funcionalidades que ya se han mencionado previamente, como mostrar el sentiment o la cantidad de menciones, hasta funcionalidades adicionales más centradas en la visualización como ajustar la precisión temporal de los datos mostrados. Esta columna con los filtros permite escoger el gráfico que se muestra, la cantidad de datos que se cargan -es decir, la precisión temporal-, qué atributo se muestra -la cantidad de tweets o el sentiment- y mostrar toda la lista de activos o empresas o criptomonedas de manera exclusiva en el caso de los datos de activos.

Además de la primera sección que se muestra en la figura 31, la página web cuenta con tres secciones adicionales que se pueden observar en todo momento: los treemaps, la evolución de la cantidad de tweets y las distintas nubes de palabras. En vez de un selector para escoger qué gráfico se muestra, estas tres secciones utilizan pestañas con las que navegar entre los gráficos de forma rápida y sencilla y se muestran filtros con funcionalidades similares a los identificados en la primera sección.

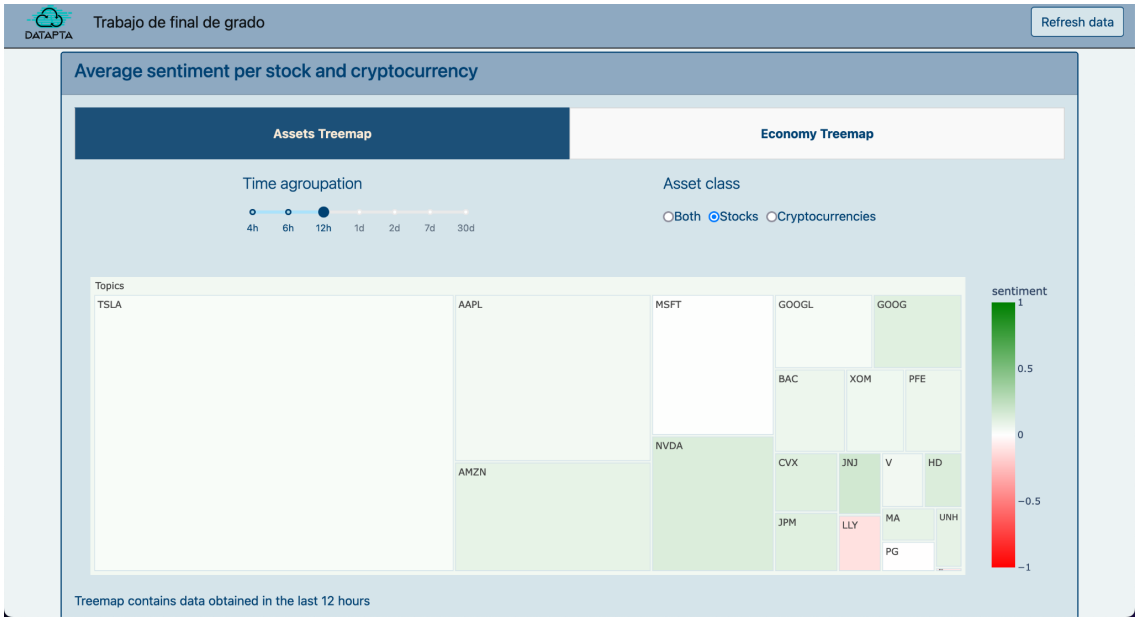

Figura 32: Captura de la página web implementada - sección 2

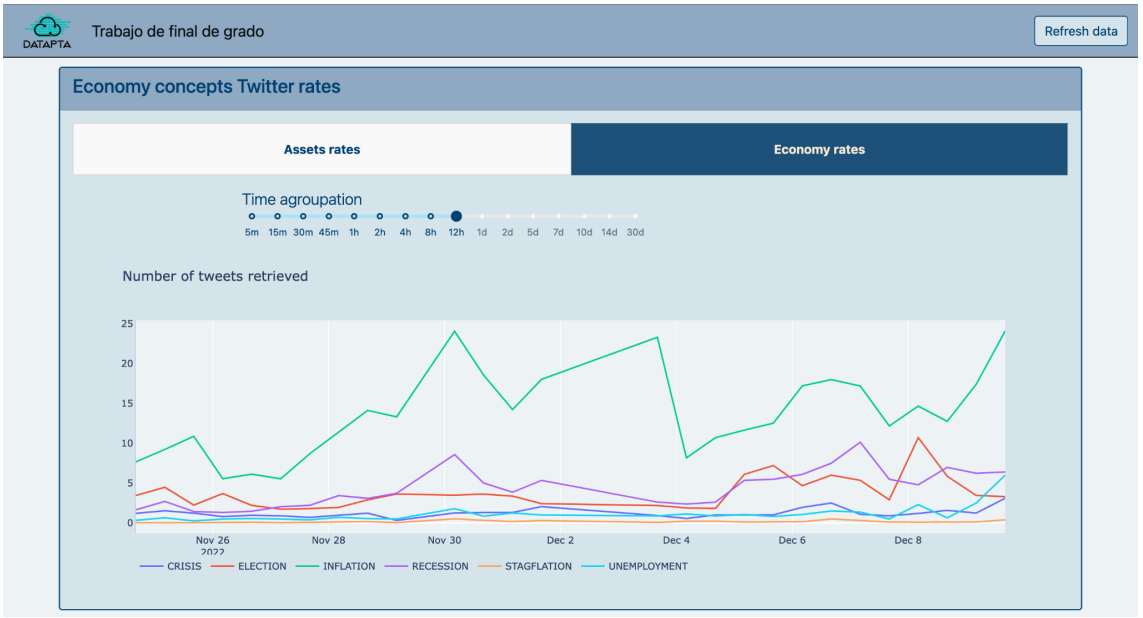

Figura 33: Captura de la página web implementada - sección 3

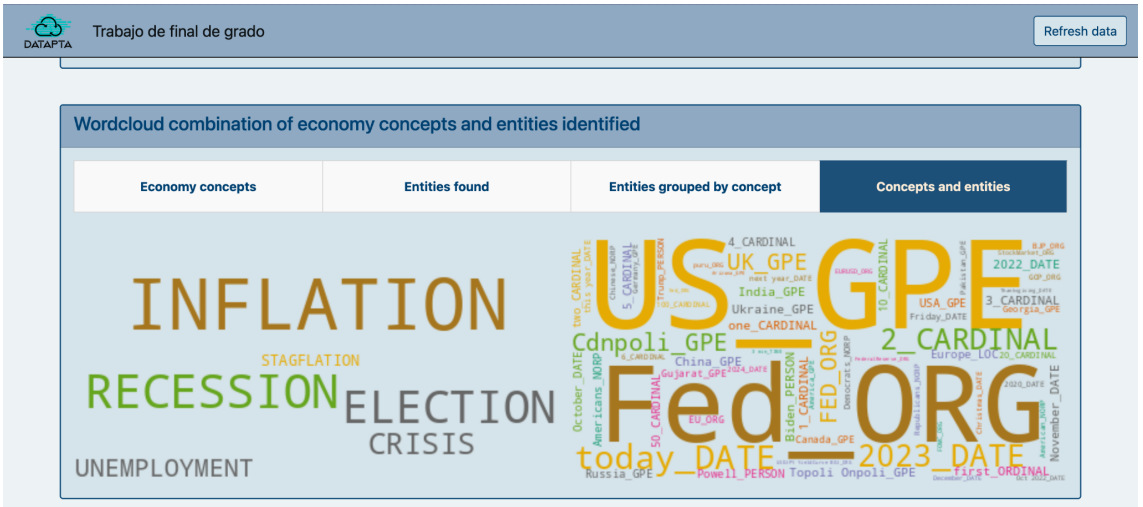

Figura 34: Captura de la página web implementada - sección 4

10. Contratiempos y obstáculos

Llegados a esta sección se han especificado los sub-objetivos y motivaciones del proyecto, las tecnologías utilizadas actualmente que ofrecen soluciones a nivel parcial del problema abordado, la arquitectura sobre la que se despliega y el desarrollo del proyecto por fases. Esta nueva sección tiene como objetivo tratar con más detalle los contratiempos y obstáculos que han surgido a lo largo de la implementación y el desarrollo del proyecto.

La segunda fase del proyecto -que corresponde al diseño e implementación del almacenamiento- contempla la posibilidad de tener que realizar implementaciones adicionales, por lo que en esta sección no se valorarán las implementaciones realizadas tras valorar los resultados obtenidos al hacer las pruebas de rendimiento. Sin tener en cuenta dichas implementaciones, los principales contratiempos y obstáculos surgidos durante el desarrollo del proyecto corresponden en su mayoría a la primera fase del proyecto, la obtención de los datos y su procesamiento mediante modelos de aprendizaje automático.

Dentro de la primera fase podemos encontrar varios contratiempos y obstáculos de importancia considerable, algunos de ellos ya mencionados previamente: las diferentes limitaciones entre cuentas de Twitter, el desmantelamiento del endpoint v1.1 de Twitter, el control de la memoria de los modelos de ML utilizados en Spark y las dificultades de aprendizaje de los distintos frameworks y entornos utilizados.

En primer lugar, debido a las diferentes limitaciones entre cuentas de Twitter el desarrollo de este proyecto se podría ver limitado e incluso no ser viable sin una cuenta con permisos de acceso adecuados. En este aspecto, el acceso que permite una cuenta de Academic Research tiene unos límites lo suficientemente elevados como para poder desarrollar todo el proyecto sin problemas, pero al mismo tiempo las condiciones de uso de dicha cuenta son más restrictivas por lo que no es posible utilizar los datos obtenidos con fines comerciales. Ya que el proyecto se desarrolla como parte de un producto comercial esta limitación impide utilizar este tipo de acceso, por lo que sólo se pueden utilizar cuentas con acceso Elevated.

Este primer obstáculo está íntimamente relacionado con el primer contratiempo encontrado, el desmantelamiento del endpoint v1.1. Las limitaciones que tiene este endpoint compensan las dificultades adicionales que tiene configurar los parámetros de búsqueda, por lo que su

uso permite una búsqueda con parámetros menos restrictivos en cuanto a volumen de tweets, es decir, al no considerarse en el límite mensual de tweets obtenidos se pueden hacer búsquedas más genéricas o de parámetros que obtengan una cantidad de tweets mayor. Debido al desmantelamiento progresivo del punto de acceso este proyecto se ha tenido que adaptar para utilizar una única cuenta con acceso al endpoint v1.1 en combinación con cuentas con acceso al endpoint v2 para poder llegar a una solución viable.

El segundo gran contratiempo ha sido el control de la memoria consumida por los modelos de ML utilizados en el Spark engine. El framework de Spark permite definir funciones con las que procesar de manera más personalizada las agrupaciones de datos o *batches*, por lo que una aproximación inicial a la implementación del procesamiento de datos consistía en utilizar modelos de librerías de Python conocidas por sus modelos de predicción como SpaCy [37] o NLTK [38]. Sin embargo, en esta versión inicial el consumo de memoria RAM aumentaba considerablemente tras procesar cada batch.

Como solución a este problema se ha optado por utilizar un framework compatible con Spark, Spark-NLP, que facilita el uso de modelos de inteligencia artificial en Spark. Este framework mejora de forma notable el uso de la RAM pero de no ser así el proyecto podría haberse visto seriamente afectado y se debería haber optado por una aproximación distinta al procesamiento en streaming para evitar una ejecución permanente con un consumo creciente.

Además de estos contratiempos y obstáculos es necesario destacar un obstáculo que usualmente se infravalora: la falta de conocimiento. El uso principal de este proyecto es como parte de un producto en desarrollo, pero al no haber productos ya existentes que se puedan aprovechar para reducir la carga de trabajo o facilitar funcionalidades, se debe desarrollar mayormente desde cero. Para ello se han utilizado frameworks, librerías y entornos no conocidos previamente por lo que ha sido necesario un aprendizaje a medida que se ha desarrollado el proyecto.

Es necesario destacar, eso sí, que los contratiempos y obstáculos mencionados se han podido abordar a tiempo sin causar retrasos o desviaciones en la planificación del proyecto.

11. Análisis de tiempo y costes

Antes de valorar la sostenibilidad del proyecto y exponer las conclusiones es necesario valorar el coste de la realización, tanto a nivel temporal como económico.

En esta sección se incorpora información detallada en el informe inicial del proyecto, es decir, el informe final correspondiente a la asignatura de gestión de proyectos -entrega 4 de GEP-. Gracias al planteamiento de dicho informe se puede valorar la correctitud de la planificación y de la metodología utilizada así como la diferencia de costes entre el presupuesto planteado y el coste real.

Antes de entrar en detalle en el tiempo dedicado y los costes del proyecto es necesario recordar la especificación de tareas hecha en el informe inicial. Las siguientes tareas se corresponden a la planificación inicial, por lo que ciertos puntos tratados en secciones anteriores -como las pruebas adicionales en caso de rendimiento insuficiente en la base de datos- no se especifican, ya que con resultados de rendimiento favorables no se hubieran realizado. Cada tarea tiene un identificador numérico con el formato *T'fase'.'tarea'* excepto las reuniones de seguimiento y control (RSC) y la documentación (DOC).

ID	Descripción	Requerimientos	Tiempo
T1.1	Configuración inicial del entorno de desarrollo e implementación del Twitter engine	Cuenta AWS para el entorno de desarrollo e información sobre los puntos de acceso de Twitter	17h
T1.2	Búsqueda de información sobre Spark y desarrollo del Spark engine	Entorno de desarrollo operativo en AWS	33h
T1.3	Conexión entre ambos engines y elección de los modelos NLP	Versiones iniciales de Spark engine y Twitter engine	5h
T1.4	Aplicación de los modelos NLP	Modelos NLP escogidos	40h
T1.5	Pruebas de ejecución de los modelos NLP	Ambos engines operativos	5h
T2.1	Versión de almacenamiento inicial en formato de archivos en AWS S3	Acceso a S3 en AWS	5h
T2.2	Aprendizaje sobre TSDB y diseño de la BD en InfluxDB Cloud	Conocimiento del formato de los datos a guardar	10h

ID	Descripción	Requerimientos	Tiempo
T2.3	Implementación en InfluxDB Cloud y conexión con los engines	Obtención de los datos funcional	14h
T2.4	Aprendizaje sobre *Flux* y planteamiento de las queries según la fase de visualización	TSDB en InfluxDB Cloud funcional	38h
T2.5	Diseño y realización de las pruebas de rendimiento	Aprendizaje para realizar pruebas de rendimiento	55h
T2.6	Migración de Cloud a OSS y realización del benchmark en OSS	Pruebas de desarrollo de T2.5 como referencia	60h
T2.7	Implementación del lago de datos	Acceso a Lake Formation en AWS	22h
T3.1	Diseño e implementación de una versión inicial de la visualización	Aprendizaje sobre visualización y creación de gráficos con Plotly	20h
T3.2	Conexión entre la visualización y la base de datos	Datos almacenados en la base de datos	15h
T3.3	Mejoras para de la versión inicial de la visualización	Diferentes alternativas para la visualización	20h
T3.4	Muestra ante posibles usuarios y mejoras según las valoraciones proporcionadas	Posibles usuarios a los que mostrar la visualización	15h
RSC	Reunión de seguimiento y control	--	26h
DOC	Redacción de documentación	--	75h
		Tiempo total estimado	475h

Tabla 4: especificación de las tareas a desarrollar

Teniendo en consideración estas tareas ya es posible realizar el análisis temporal y económico en función del desarrollo real.

11.1. Análisis temporal

Para entender mejor cómo difiere la planificación inicial con el desarrollo real se utilizan diagramas de Gantt, valorando así si se han producido desviaciones respecto a la planificación inicial. Para facilitar la comparación de las tareas se dividen los diagramas de Gantt en tres períodos de tiempo.

El primer período valorado corresponde del 15 de septiembre al 19 de octubre e incluye toda la fase de obtención y procesamiento de los datos y las cuatro primeras tareas del almacenamiento. En este primer intervalo de tiempo que se puede observar en la figura 35 no hay cambios significativos entre la planificación inicial y el desarrollo real. El cambio más relevante es la ampliación de las tareas T1.1 y T1.2 un día más, debido a una planteamiento erróneo de cuántos días corresponde cada tarea, pero las horas estimadas sí son correctas.

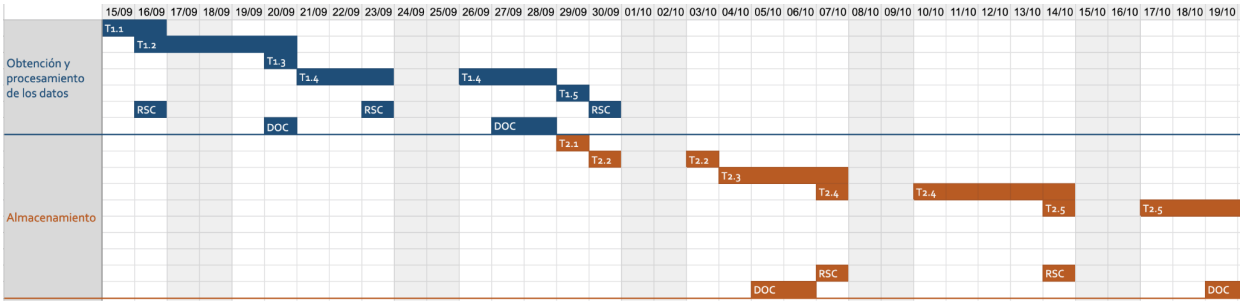

Figura 35: diagrama de Gantt, duración real del período del 15/09-19/10

En este primer período, por lo tanto, los obstáculos y contratiempos especificados previamente se han podido desarrollar dentro de las horas estimadas sin suponer un retraso en el desarrollo del proyecto.

El segundo período valorado corresponde del 19 de octubre al 22 de noviembre e incluye las tareas restantes de la fase del almacenamiento y las primeras tareas de la fase de la visualización. En este segundo período, a diferencia del anterior, hay cambios significativos entre la planificación y la dedicación real. Estos cambios se deben a dos principales motivos: la rapidez en la implementación y desarrollo del benchmark de las bases de datos y la tarea adicional resultante de las valoraciones de las pruebas: la implementación de bases de datos alternativas.

En el primer caso, se puede observar en la figura 36 -planificación inicial- cómo las tareas T2.5, T2.6 y T2.7 tienen una duración mayor que en la figura 37 -duración real-. Esto se debe principalmente al tiempo extra añadido en la estimación de las tareas para la resolución de pequeños problemas o dificultades del desarrollo. Al no encontrarse grandes dificultades el tiempo se ve considerablemente reducido, por lo que de haber obtenido resultados favorables en las pruebas de rendimiento el proyecto se hubiese beneficiado de un adelanto en su finalización.

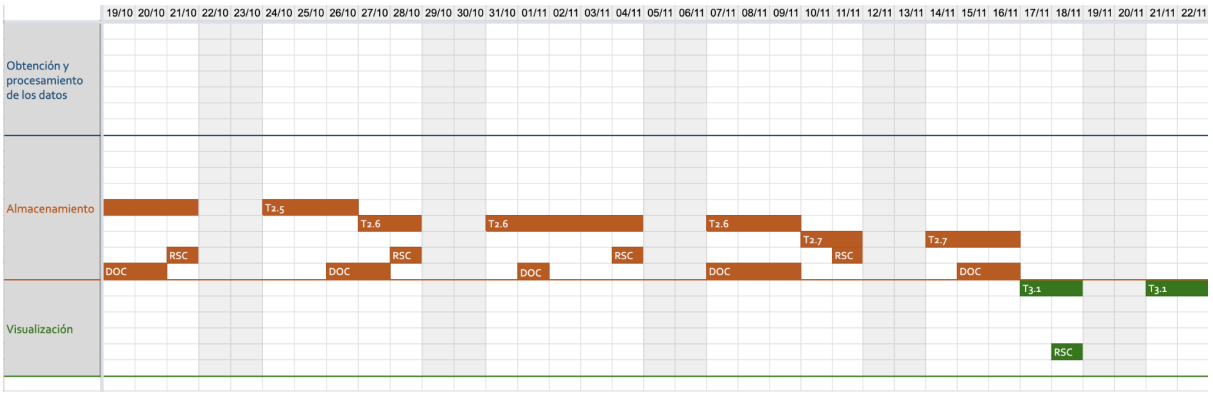

Figura 36: diagrama de Gantt, planificación del período del 19/10-22/11

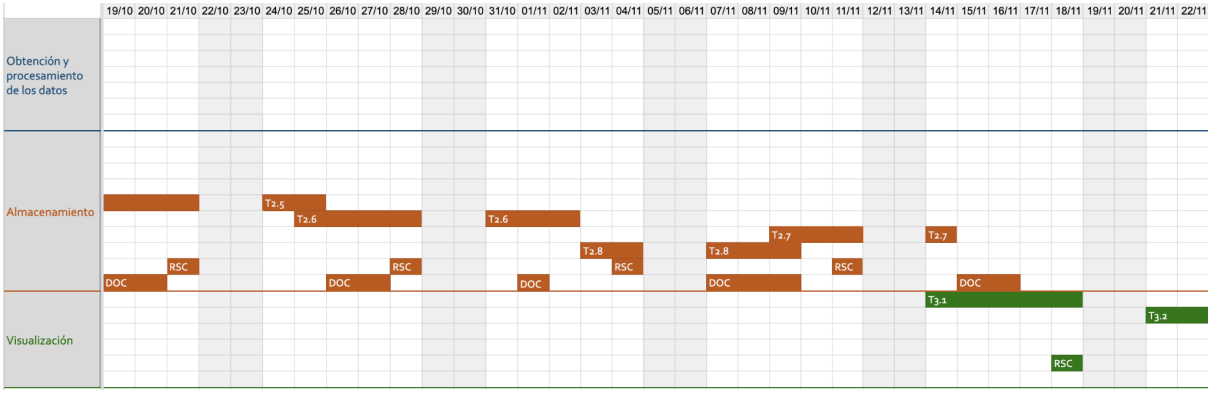

Figura 37: diagrama de Gantt, duración real del período del 19/10-22/11

Sin embargo, al valorar negativamente los resultados obtenidos de las pruebas de rendimiento se añade una nueva tarea en la figura 37: la tarea T2.8. Esta tarea adicional corresponde a la implementación de nuevas bases de datos -QuestDB y Timescale- junto a la realización de un subconjunto de pruebas de rendimiento de menor complejidad. Gracias a la migración realizada en la tarea T2.6, el desarrollo de la tarea T2.8 se puede realizar rápidamente sin causar retrasos en el desarrollo del proyecto, pudiendo empezar así la tarea T3.1 tres días antes de lo previsto.

El tercer período corresponde del 12 de noviembre al 16 de diciembre e incluye principalmente el desarrollo de la fase de la visualización. La posibilidad ya mencionada de adelantar el inicio del desarrollo de la visualización ha tenido un impacto positivo inesperado ya que el desarrollo de las tareas T3.2 y T3.4 requieren más tiempo del estimado en la planificación. Por otro lado, la tarea T3.3 requiere menos tiempo del estimado, lo cual permite que el tiempo adicional de dichas fases no suponga un retraso del proyecto.

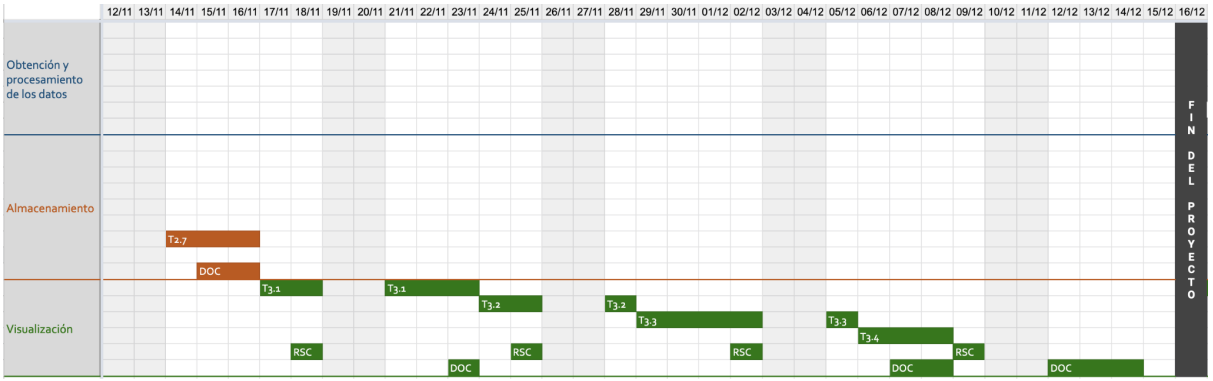

Figura 38: diagrama de Gantt, planificación del período del 12/11-16/12

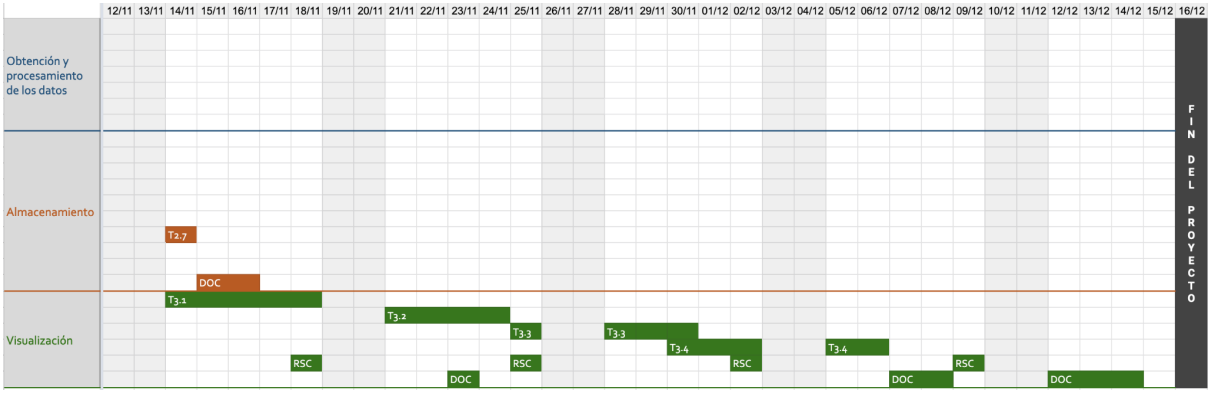

Figura 39: diagrama de Gantt, duración real del período del 12/11-16/12

En las figuras 38 y 39 se puede observar la diferencia de días dedicados a cada tarea, pero es necesario destacar que la fecha de finalización del proyecto se mantiene. El tiempo adicional requerido se debe a los cambios planteados al mostrar la visualización a potenciales usuarios, con una complejidad mayor a la esperada.

Como pequeña conclusión se debe destacar que las estimaciones al alza del desarrollo en el planteamiento inicial han permitido abordar los distintos contratiempos y obstáculos sin tener un impacto negativo en el coste temporal del desarrollo del proyecto. Además, en aquellas tareas donde no ha habido obstáculos ni contratiempos considerables el tiempo ha sido menor al estimado, por lo que se ha podido desarrollar en menos de 475 horas. Hay que tener en cuenta, eso sí, que la incorporación de la tarea T2.8 y las ampliaciones de las tareas T3.2 y T3.4 hacen que esta reducción no sea óptima sobre la planificación, con 460 horas en total.

A fecha de 6 de diciembre se considera el desarrollo del proyecto finalizado y a fecha de 14 de diciembre se considera finalizada una versión inicial de la documentación del proyecto, es decir, la memoria del trabajo final de grado. Se contempla, eso sí, la posibilidad de realizar

pequeñas mejoras o modificaciones según se valore junto con el director y el ponente del trabajo.

11.2. Análisis económico

Para poder realizar el análisis económico hay que recordar primero la previsión inicial del presupuesto y los distintos componentes que componen los gastos de realización del proyecto.

Coste previsto

Mediante el uso de la herramienta *AWS Pricing Calculator* el coste esperado de la infraestructura utilizada en Amazon Web Services es de 58,62 USD mensuales [39], lo cual corresponde a 175,86 USD al considerar los tres meses de desarrollo del proyecto. Esto es considerando la arquitectura especificada en la figura 3. Sin embargo, considerando el desglose de las tareas por días según la planificación el coste es el siguiente:

Recurso	**Días**	**Coste al día (USD)**	**Coste (USD)**
EC2 engines	65	1,61	104,65
EC2 base de datos	30	0,33	9,9
Almacenamiento S3	52	0,0	0,0
Lake Formation	20	0,01	0,2
		Coste total (USD)	114,75

Tabla 5: costes estimados de AWS

Además de los costes de la arquitectura hay que tener en cuenta los costes del personal de desarrollo, que corresponden a Xavier Gervilla como autor y desarrollador y a David Prat como director y encargado del control y seguimiento. Para la estimación inicial se consideran las horas de la tabla 4: el total para autor y las horas de RSC para el director.

Personal	**Participación (h)**	**Salario (€/h)**	**Coste (€)**
Xavier Gervilla	475	9	4.275
David Prat	26	16	416
		Coste total (€)	4.691

Tabla 6: estimación del coste de personal

Por último, hay que tener en cuenta costes adicionales relacionados con los recursos necesarios para el desarrollo del proyecto, es decir, el lugar en el que se desarrolla, el ordenador utilizado y la conexión a internet que es esencial para poder utilizar los servicios en la nube. En este caso los costes estimados corresponden a 688€ mensuales [40, 41], lo que supone un total de 2.084€. El coste mensual se desglosa de la siguiente manera: 38€/mes para la tarifa de internet, 650€ para el alquiler del lugar de trabajo y se considera que el coste del ordenador es nulo por ser de propiedad.

Teniendo en cuenta los tres tipos de costes la estimación inicial es de 6.940€: 165€ de AWS, 4.691€ de personal y 2.084€ de costes adicionales. Sin embargo, para la infraestructura en Amazon Web Services se ha considerado un margen mensual del 25% sobre el precio obtenido mediante la calculadora, es decir, un presupuesto mensual de 75€. Este margen incrementa la estimación inicial del presupuesto hasta un total de 7.000€.

Coste real

Para obtener el coste real se consideran las horas totales dedicadas resultantes de la implementación y desarrollo del proyecto y los costes de infraestructura reales de AWS. Al igual que en la previsión, se tienen en cuenta los recursos especificados en la figura 3. Antes de entrar en detalle en el desglose de los costes, hay una primera diferencia con el coste previsto y es que los costes adicionales no están disponibles. Ni el autor ni el director se encargan de proporcionar de manera directa la conexión a internet ni la localización de desarrollo por lo que estos costes, pese a estar presentes, no han sido posibles de calcular.

Los costes que sí se han podido obtener corresponden por lo tanto a los costes de personal y de arquitectura, 4.556€ y 138,50€ respectivamente. A nivel de personal de desarrollo se mantienen los salarios por hora de la tabla 6 pero la cantidad de horas del autor del proyecto, Xavier Gervilla, disminuye de 475 a 460 horas, por lo que el coste de personal disminuye de los 4.691€ hasta los 4.556€.

En cuanto a la arquitectura, al valorar que no se llega a sobrepasar la previsión mensual del coste en AWS se ha optado por invertir el margen para errores en el servicio de soporte que ofrece Amazon Web Services. Este servicio ha supuesto por lo tanto un coste adicional no especificado inicialmente pero se ha pagado con los recursos ya previstos, por lo que no ha

resultado en un incremento en el coste. Sin embargo, ha permitido solucionar de forma más rápida las dudas o problemas surgidos relacionados con los recursos de AWS, lo cual ha sido muy útil por el desconocimiento y la complejidad que ha supuesto configurar el entorno desde cero. Además del servicio de soporte, también hay una cantidad menor del presupuesto que se ha dedicado al servicio de InfluxDB Cloud. Estos dos costes adicionales no tienen un coste por horas sino que se especifica el total según su uso.

Tanto InfluxDB OSS como las bases de datos analizadas al realizar las pruebas de benchmarking -incluyendo QuestDB, la BD escogida para la implementación final-, en cambio, no han tenido costes adicionales. En concreto, los 138,50€ se desglosan en la siguiente tabla.

Recurso	Tiempo de uso (h)	Coste ($/h)	Coste ($)
EC2 Instance t2.micro	1272	0,0116	14,76
EC2 Instance t3a.large	1512	0,0752	113,70
InfluxDB Cloud	--	--	3,00
AWS Support	--	--	16,00
		Coste total	147,46 $

Tabla 7: costes reales de AWS

Los costes reales del proyecto, por lo tanto, alcanzan un total de 4.694,50 euros. Hay que tener en cuenta que a esta cifra se le deberían añadir los costes adicionales mencionados, pero al no estar disponibles se comparará con el presupuesto sin estos costes, el cual disminuye a 4.916 euros. Considerando únicamente los costes de arquitectura y personal se puede apreciar que ha habido un ahorro respecto al presupuesto inicial de 221,50€.

Hay que tener en cuenta que el coste real tiene dos conceptos no valorados en el presupuesto inicial -AWS Support y el coste de InfluxDB Cloud-, por lo que si se hubiesen contemplado desde un inicio estaríamos hablando de un ahorro aún mayor. Tanto en el presupuesto como en el coste real la mayor parte corresponde a costes de personal pero hay que recordar que, tal y como se ha mencionado en secciones anteriores, de haber obtenido resultados favorables en las primeras pruebas de rendimiento el proyecto se hubiese podido terminar con una antelación mayor, reduciendo así los costes de personal.

12. Sostenibilidad

Aunque ya se ha valorado la justificación del desarrollo a nivel del estado del arte, también es necesario valorar la sostenibilidad del proyecto así como el impacto ambiental, económico y social que tiene. Al tratarse de un proyecto que forma parte de un producto en desarrollo, medir el impacto que puede tener durante su vida útil tiene una complejidad más elevada, por lo que esta sección se centra principalmente en la fase de proyecto puesto en producción, conocida también como PPP. Previamente a la valoración del impacto ambiental, económico y social se evalúa el conocimiento de sostenibilidad del autor del trabajo.

12.1. Autoevaluación de la competencia de sostenibilidad

La reflexión principal tras realizar la autoevaluación es que a nivel general se dispone de conocimientos sobre buenas prácticas y hay una capacidad de pensamiento crítico para valorar distintas opciones según su sostenibilidad.

A nivel ambiental destaca el compromiso por reducir y optimizar recursos siempre que sea posible. Priorizar la búsqueda de información para tener un buen planteamiento es una buena práctica que ayuda tanto en este proyecto como en muchos otros ámbitos a desarrollar trabajo de mejor calidad en menor tiempo, ya que un buen planteamiento inicial permite evitar errores comunes y así optimizar el tiempo y los recursos. Sin embargo, hay margen de mejora en lo que respecta al conocimiento y al uso de métricas e indicadores de sostenibilidad.

A nivel económico destaca la capacidad de gestión. Hay un buen uso de pensamiento lógico que permite analizar mejor el presupuesto con tal de sacar el mayor rendimiento posible dados unos recursos determinados y el desarrollo de código se suele enfocar con la idea de que sea usable y replicable en otros sistemas, lo cual ahorra el dinero que corresponde a implementar de nuevo las aplicaciones. Al igual que a nivel ambiental, hay margen de mejora en lo que respecta al conocimiento y al uso de métricas que permitan valorar el impacto.

A nivel social destaca el compromiso con el usuario final. En todo momento se tiene en cuenta al usuario final para plantear las soluciones de forma entendible y usable. En el aspecto social es donde hay una mayor autocrítica ya que se puede mejorar mucho en cuanto a planteamiento de alternativas o soluciones accesibles para personas vulnerables ya que las alternativas planteadas priorizan por lo general el impacto ambiental o económico.

12.2. Impacto ambiental

Una vez hecha la autoevaluación podemos entrar en detalle en el impacto que tiene el proyecto en diferentes niveles, empezando por el impacto ambiental.

Durante la fase PPP el impacto ambiental corresponde en su mayor totalidad a los recursos utilizados en AWS. Sin embargo, calcular la huella de carbono resultante de utilizar AWS es un proceso muy complejo. Los servicios de desarrollo en la nube y Amazon Web Services en concreto no ofrecen una transparencia que permita determinar la totalidad de los recursos físicos que se utilizan, por lo que determinar la electricidad que consumen y su impacto ambiental no es posible. Lo que sí que podemos hacer es establecer una serie de buenas prácticas que, independientemente de los recursos físicos asignados, permitirán reducir su consumo y por lo tanto su impacto. De estas buenas prácticas destacamos dos:

- Aturar las EC2 Instances cuando no se esté desarrollando ni ejecutando nada, cuando se busca información o se redacta documentación por ejemplo. Esto reducirá el consumo de energía correspondiente a los procesadores prácticamente a 0 cuando no se esté utilizando la instancia.
- Optimizar el diseño de la base de datos y del data lake y disminuir la cantidad de datos que almacenar. El data lake en AWS no se puede parar mientras no se está ejecutando nada, por lo que esta buena práctica tiene el objetivo de reducir los recursos asignados y aprovecharlos al máximo.

Además de las buenas prácticas, es razonable pensar en reutilizar recursos para ahorrar tiempo y dinero. Sin embargo y tal y como se justifica en la documentación correspondiente al estado del arte, no hay recursos de dominio público que permitan aprovechar código ya implementado.

En la fase PPP hay recursos secundarios que corresponden a la búsqueda de información y al propio desarrollo pero estos se ven reducidos únicamente al mantenimiento una vez el proyecto esté en la fase de vida útil, por lo que se puede considerar que durante la vida útil del proyecto el impacto ambiental corresponde de forma única a los recursos de AWS.

Hay que destacar también que al desarrollarse en remoto el impacto ambiental derivado de los desplazamientos a oficinas es cero, el cual correspondería al combustible consumido en dos viajes en transporte público de 30 kilómetros cada día laborable.

Una mejora considerable para minimizar el impacto ambiental es utilizar recursos propios de manera controlada como sustituto de los recursos que ofrece Amazon Web Services. Para desarrollar un prototipo es razonable aprovechar recursos en la nube y así poder valorar la viabilidad antes de hacer una inversión en los recursos necesarios para poder ejecutar en un entorno de propiedad. Es por ello que en la fase de la vida útil la migración de los recursos a un servidor de propiedad permitiría controlar mejor los recursos utilizados y hacer las optimizaciones necesarias para reducir el impacto ambiental.

12.3. Impacto económico

En lo que respecta al impacto económico hay que valorar tanto los costes de desarrollo como el impacto que tiene dentro de Datapta.

El coste de desarrollar el proyecto tiene un valor total de 4.694,50€ tal y como se ha especificado en la sección anterior. Este coste corresponde íntegramente a la fase PPP, por lo que hasta que no se integra el proyecto con Finlooker como producto el impacto económico es negativo. Una vez implementado, pero, hay dos opciones para generar un impacto positivo partiendo del proyecto.

Como primera opción se considera la integración con Finlooker, con un impacto variable según el modelo de negocio planteado desde Datapta. Siguiendo un modelo similar a los productos especificados en la sección del estado del arte, podemos ver que un modelo de suscripción mensual es la opción escogida para obtener beneficios. La cantidad exacta de la suscripción es difícil de determinar ya que el primer producto, TopStocks, tiene suscripciones a partir de 20€/mes mientras que el segundo producto, SocialSentiment.io, ofrece planes desde 5€/mes. Como Finlooker ofrece más datos y de mayor calidad o relevancia, es razonable pensar que el precio de la suscripción pueda ser similar para captar más clientes.

Como segunda opción se considera la venta del motor de obtención y procesamiento, tanto como producto que pueden utilizar las propias empresas según sus necesidades como servicio

que se ofrece a las empresas, en ambos casos con un coste que permita generar beneficios. Esta segunda opción no sólo supone un impacto positivo para Datapta sino que ofrece a las otras empresas la posibilidad de utilizar una tecnología ya desarrollada con la que ahorrar recursos, disminuyendo así el impacto económico negativo. Ambas opciones, por lo tanto, permiten un ahorro de tiempo y dinero a sus usuarios.

12.4. Impacto social

En lo que respecta al impacto social del proyecto se deben valorar dos puntos de vista: el impacto a nivel personal y el impacto sobre los usuarios finales.

A nivel personal, el proyecto supone una gran oportunidad para valorar con criterio cómo es el proceso de desarrollar un producto a prácticamente todos los niveles: desde la comprensión de las necesidades o problemas que se deben resolver hasta el desarrollo de un prototipo o versión inicial. Permite además trabajar con infraestructura en la nube y aprender sobre tecnologías ampliamente utilizadas como Spark. La oportunidad de hacerlo en colaboración con la empresa Datapta proporciona una experiencia adicional muy valiosa tanto para el crecimiento personal como para la formación sobre nuevo conocimiento.

A nivel de los usuarios finales, debemos valorar el proyecto como parte del producto Finlooker. En este aspecto, se puede identificar la responsabilidad de qué uso se le da a los datos generados gracias al proyecto. Estos datos corresponden a la contextualización del mercado de acciones y activos para aquellas personas interesadas en invertir o vender, un colectivo que ha crecido durante los últimos años gracias a noticias favorables sobre las subidas de las bolsas a nivel global [42].

Este aumento de personas interesadas tiene un doble impacto sobre Finlooker y el proyecto, hay un mayor número de potenciales usuarios finales pero a la vez hay un mayor riesgo de que estos usuarios no estén bien formados, por lo que un mal uso de la información podría suponer pérdidas económicas importantes. Por ello, el proyecto incluye información a nivel económico que productos similares no tienen para ayudar a contextualizar los datos y como parte de Finlooker se espera complementar los datos del proyecto con información que permita a sus usuarios obtener una formación con la que mejorar su criterio y disminuir la toma de malas decisiones, fomentando la inversión consciente y no especulativa.

12.5. Impacto de los riesgos

Por último es necesario valorar el impacto que tienen los riesgos planteados, en concreto hay dos principales factores que podrían hacer que el proyecto o Finlooker no sea viable durante su vida útil: la obtención de los datos de Twitter y la implicación de los usuarios finales.

Tal y como se ha mencionado anteriormente, la API de Twitter tiene ciertas limitaciones que influyen en la cantidad de tweets que se pueden obtener. Durante la fase PPP no se han superado las limitaciones, pero es un riesgo al escalar la aplicación durante la vida útil, por lo que en caso de hacerlo habría un claro impacto a nivel económico y/o en la calidad de la información. Las alternativas planteadas en caso de superar las limitaciones son credenciales de empresa, obtener los datos de proveedores como Tweet Binder o descartar datos y coger una muestra representativa. En los dos primeros casos, el coste aumentaría considerablemente y podría suponer la no viabilidad económica de Finlooker, mientras que el tercer caso supondría una disminución en la calidad de los datos proporcionados a los usuarios finales.

Por otro lado, si la cantidad de usuarios finales no es suficientemente elevada el proyecto podría ser viable a nivel de implementación y desarrollo pero dentro del contexto de Finlooker y Datapta podría no ser viable económicamente. Para disminuir el impacto de este riesgo, que tiene más relevancia como parte de Finlooker que como proyecto independiente, hay dos posibles alternativas que lo podrían hacer viable y así no generar un impacto negativo. La primera opción es vender la tecnología utilizada a empresas del mismo sector y la segunda opción es cambiar el modelo de negocio modificando el motor de obtención de datos tal y como se especifica en la sección de actores implicados. Estas alternativas permitirían amortizar el coste de desarrollo e incluso llegar a generar beneficios.

13. Conclusiones

En este trabajo se ha presentado el desarrollo de una aplicación con la que fomentar la inversión consciente. Se han valorado distintas alternativas ya existentes pero las distintas carencias funcionales que tienen han justificado el desarrollo de una nueva aplicación, con todo lo que ello conlleva. Al requerir una nueva aplicación para dar solución al problema planteado, en este proyecto se han desarrollado tanto el diseño como la implementación de las distintas funcionalidades y componentes que la forman.

El primer componente, el motor de obtención y procesamiento de datos, ha supuesto el desarrollo de tecnología no existente con la que obtener datos de Twitter y procesarlos de forma masiva mediante el framework Spark. Al no tener aplicaciones similares como referencia, el desarrollo ha sido considerablemente complejo pero el planteamiento realizado permite que este componente se pueda adaptar fácilmente a otros ámbitos e incluso ofrecerlo como producto independiente.

El segundo componente, la base de datos, ha tenido una complejidad menor pero no por ello ha supuesto menos trabajo del esperado. Al plantear este componente se ha puesto en relevancia la importancia de hacer un buen diseño de las estructuras de datos utilizadas en las bases de datos, así como la importancia del rendimiento según el uso. Pese a mantener un diseño muy similar desde un inicio, se ha podido comprobar que las distintas bases de datos disponibles en la actualidad tienen un rendimiento variable. No todas las bases de datos están diseñadas para abordar el mismo problema sino que se deben valorar de forma individualizada según su uso y aplicación. El desarrollo de las pruebas de rendimiento ha permitido valorar y aprender sobre distintas métricas que se pueden utilizar como referencia para comparar usos e implementaciones.

El tercer componente, la visualización, se ha implementado mediante una página web desarrollada en Python. Esta página web tiene una complejidad menor que las páginas web convencionales ya que se ha construido sobre módulos básicos, pero aún así sirve su propósito para mostrar los datos almacenados y generados mediante las componentes anteriores. Con la implementación de esta tercera componente la aplicación es funcional pero requiere de pasos adicionales para poder utilizarse como producto de cara a los usuarios finales, es decir, el motor de obtención y procesamiento de datos y la base de datos son

completamente funcionales pero para poder ofrecer la aplicación como producto se requiere de una página web más especializada y de mayor complejidad.

El desarrollo de la página web de mayor complejidad no se ha contemplado en este trabajo ya que se considera como paso adicional en la integración del proyecto dentro de Finlooker, es decir, es un paso cuyo desarrollo está valorado y fuera del alcance de este trabajo.

Finalmente se han valorado las condiciones de desarrollo del proyecto, es decir, el coste a nivel temporal y económico y la sostenibilidad del trabajo. El desarrollo del trabajo aplicando medidas de ahorro ha permitido reducir los costes económicos y el diseño inicial ha permitido poder desarrollar la componente del almacenamiento más rápido de lo previsto inicialmente. Sin embargo, hay ciertos puntos que se podrían modificar para mejorar la sostenibilidad y los costes de la aplicación durante su vida útil, como es por ejemplo la migración de los recursos en Amazon Web Services a un servidor de propiedad. Esta migración se contempla como futura continuación del proyecto al igual que la mejora de la página web o su desarrollo junto a un equipo especializado en desarrollo front-end.

Estos puntos de mejora pueden reducir el impacto negativo económico y ambiental de la aplicación, proporcionando un mayor control sobre los recursos utilizados. Por mucho que la aplicación pueda tener beneficios sociales o económicos notables, como desarrolladores no podemos olvidarnos del medio ambiente; sería ilógico desarrollar una aplicación con la que ayudar a concienciar sobre la inversión olvidando por completo el impacto medioambiental que puede tener durante su vida útil. En este aspecto, la autoevaluación realizada en la asignatura GEP ha permitido ser más críticos y aplicar mejores prácticas durante el desarrollo del trabajo.

Por último acabaremos con una reflexión. A día de hoy la inteligencia artificial sigue teniendo detractores por su polémico uso en grandes empresas para rastrear a personas o por su uso con fines poco éticos. Pero es responsabilidad de los profesionales aplicar sus conocimientos para construir una mejor sociedad y un mejor futuro, ya que su uso para fomentar el conocimiento o ayudar a las personas puede tener un impacto positivo incluso en campos controvertidos o polémicos como es el mercado de las criptomonedas.

14. Referencias

[1] *"TopStonks - The best advice from the worst investors on the internet"*, Topstonks.com. [En línea] [Consultado: 14 oct. 2022] <https://topstonks.com>

[2] *"SocialSentiment.Io - social media sentiment analysis"*. Socialsentiment.io. [En línea] [Consultado: 14 oct. 2022] <https://socialsentiment.io/>

[Figura 1] *Comentarios en TopStonks sobre AAPL.* Captura de TopStonks. [En línea] [Consultado: 14 oct. 2022] <https://topstonks.com/stocks/AAPL?st_aapl>

[3] *"Tweepy Documentation — tweepy 4.10.1 documentation"*. Tweepy. [En línea] [Consultado: 23 sept. 2022] <https://docs.tweepy.org/en/stable>

[4] *"PySpark Documentation — PySpark 3.3.9 documentation"*. Apache Spark. [En línea] [Consultado: 23 sept. 2022] <https://spark.apache.org/docs/latest/api/python>

[5] *"Quick Start — AWS SDK for Pandas 2.18.0 Documentation"*. Readthedocs [En línea] [Consultado: 23 sept. 2022] <https://aws-sdk-pandas.readthedocs.io/en/stable/>

[6] *"Plotly Python Graphing Library"*. Plotly. [En línea] [Consultado: 23 sept. 2022] <https://plotly.com>

[Figura 2] *Arquitectura de las distintas funcionalidades.* Imagen de generación propia mediante *diagrams.net*. [Creación: 14 oct. 2022] <https://app.diagrams.net/>

[Figura 3] *Arquitectura de los distintos recursos utilizados.* Imagen de generación propia mediante *diagrams.net*. [Creación: 14 oct. 2022] <https://app.diagrams.net/>

[Figura 4] *Tweet de muestra con el activo $SOL.* Captura de Twitter. [En línea] [Consultado: 14 oct. 2022] <https://twitter.com/VibezXBT/status/1580928349332905984>

[Figura 5] Tweet de muestra sobre la inflación (inflation). Captura de Twitter [En línea] [Consultado: 14 oct. 2022] <https://twitter.com/markminervini/status/1585262482813325313>

[7] *"Comprar datos de Twitter"*. Tweet Binder. [En línea] [Consultado: 23 sept. 2022] <https://www.tweetbinder.com/es/planes-y-precios>

[8] "API reference index | Docs". Twitter Developer [En línea] [Consultado: 15 sept. 2022] <https://developer.twitter.com/en/docs/api-reference-index>

[9] "*Deprecation announcement: Removing compliance messages from statuses/filter and retiring statuses/sample from the Twitter API v1.1"*. Twitter Community [En línea] [Consultado: 15 sept. 2022]<https://twittercommunity.com/t/deprecation-announcement-removing-compliance-messages-from-statuses-filter-and-retiring-statuses-sample-from-the-twitter-api-v1-1/170500>

[10] "*Twitter API v2 support*". Twitter Developer [En línea] [Consultado: 15 sept. 2022] <https://developer.twitter.com/en/support/twitter-api/v2>

[11] *"Filtered stream introduction | Docs"*. Twitter Developer [En línea] [Consultado: 15 sept. 2022] <https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/introduction>

[12] *"Filtered stream - How to build a rule | Docs"*. Twitter Developer [En línea] [Consultado: 15 sept. 2022] <https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/integrate/build-a-rule>

[13] *"Rate limits | Docs"*. Twitter Developer. [En línea] [Consultado: 16 sept. 2022] <https://developer.twitter.com/en/docs/twitter-api/rate-limits>

[14] *"Apache Spark"*. Wikipedia contributors. [En línea] [Consultado: 16 sept. 2022] <https://en.wikipedia.org/w/index.php?title=Apache_Spark&oldid=1116471977>

[15] *"Overview - Spark 3.3.1 Documentation"*. Apache.org. [En línea] [Consultado: 16 sept. 2022] <https://spark.apache.org/docs/latest/>

[16] *"Structured streaming programming guide - Spark 3.3.1 Documentation"*. Apache.org. [En línea] [Consultado: 16 sept. 2022] <https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>

[Figura 6] *Gráfico de TopStonks sobre el volumen de comentarios de AAPL.* Captura de TopStonks. [En línea] [Consultado: 27 oct. 2022] <https://topstonks.com/stocks/AAPL?st_aapl>

[17] *"John Snow Labs"*. John Snow Labs [En línea] [Consultado: 20 sept. 2022] <https://www.johnsnowlabs.com/>

[18] *"John Snow Labs NLP - The Models Hub"*. John Snow Labs [En línea] [Consultado: 20 sept. 2022] <https://nlp.johnsnowlabs.com/models>

[Figura 7] *Flujo de datos entre el motor de búsqueda y el motor de procesamiento.* Imagen de generación propia mediante *diagrams.net*. [Creación: 12 nov. 2022] <https://app.diagrams.net/>

[19] *"DB-Engines Ranking of Time Series DBMS"*. DB-Engines [En línea] [Consultado: 23 sept. 2022] <https://db-engines.com/en/ranking/time+series+dbms>

[20] *"¿Qué es un lago de datos?"*. Amazon Web Services [En línea] [Consultado: 23 sept. 2022] <https://aws.amazon.com/es/big-data/datalakes-and-analytics/what-is-a-data-lake/>

[21] *"Time Series Data Products | Influxdata"*. InfluxData [En línea] [Consultado: 23 oct. 2022] <https://www.influxdata.com/products/>

[Figura 8] *Estructura de almacenamiento de los tweets obtenidos.* Imagen de generación propia mediante *dbdiagram.io*. [Creación: 10 nov. 2022] <https://dbdiagram.io/d>

[Figura 9] *Estructura de almacenamiento del volumen de tweets.* Imagen de generación propia mediante *dbdiagram.io*. [Creación: 10 nov. 2022] <https://dbdiagram.io/d>

[Figura 10] *Estructura de almacenamiento del sentiment de activos.* Imagen de generación propia mediante *dbdiagram.io*. [Creación: 10 nov. 2022] <https://dbdiagram.io/d>

[Figura 11] *Estructura de almacenamiento del sentiment de economía.* Imagen de generación propia mediante *dbdiagram.io*. [Creación: 10 nov. 2022] <https://dbdiagram.io/d>

[22] *"What are series and bucket in InfluxDb"*. Stack Overflow [En línea] [Consultado: 29 sept. 2022] <https://stackoverflow.com/questions/58190272/what-are-series-and-bucket-in-influxdb>

[23] *"Data layout and schema design best practices for InfluxDB"*. A. Dotis-Georgiou, InfluxData [En línea] [Consultado: 29 sept. 2022] <https://www.influxdata.com/blog/data-layout-and-schema-design-best-practices-for-influxdb>

[Figura 12] *Conversión directa de las estructuras de datos.* Imagen de generación propia mediante *dbdiagram.io*. [Creación: 10 nov. 2022] <https://dbdiagram.io/d>

[Figura 13] *Conversión alternativa de la tabla rates-table.* Imagen de generación propia mediante *dbdiagram.io*. [Creación: 10 nov. 2022] <https://dbdiagram.io/d>

[Figura 14] *Flujo de datos entre los componentes de la primera fase y las bases de datos planteadas.* Imagen de generación propia mediante *diagrams.net*. [Creación: 12 nov. 2022] <https://app.diagrams.net/>

[24] *"What is Database Benchmarking?"*. Benchant [En línea] [Consultado: 14 oct. 2022] <https://benchant.com/blog/database-benchmarking>

[25] *"Benchmark Definition & Meaning"*. Merriam-Webster [En línea] [Consultado: 14 oct. 2022] <https://www.merriam-webster.com/dictionary/benchmark>

[26] *"Benchmarking Databases 101 - part 1"*. K. Ksiazek, Severalnines [En línea] [Consultado: 14 oct. 2022] <https://severalnines.com/blog/benchmarking-databases-101-part-1>

[27] *"Types of benchmarking"*. HotStats [En línea] [Consultado: 14 oct. 2022] <https://www.hotstats.com/hotel-industry-resources/types-of-benchmarking>

[28] *"Sysbench: PostgreSQL 12, 13 and 14 on a small server"*. M. Callaghan, Small Datum. [En línea] [Consultado: 14 oct. 2022] <http://smalldatum.blogspot.com/2021/12/sysbench-postgresql-12-13-and-14-on.html>

[29] *"How to Benchmark a Database"*. John Page, Medium.com [En línea] [Consultado: 14 oct. 2022] <https://medium.com/@johnlpage/how-to-benchmark-a-database-7088040f1c27>

[30] *"Database and transaction processing performance handbook"*. J. Gray, Azurewebsites [En línea] [Consultado: 14 oct. 2022] <http://jimgray.azurewebsites.net/benchmarkhandbook/chapter1.pdf>

[31] *"8 crucial database performance metrics"*. Scout Apm, 28-ene-2021. [En línea] [Consultado: 14 oct. 2022] <https://scoutapm.com/blog/database-performance-metrics>

[Figura 15] *Diferencia de tamaño y tiempo entre ambas arquitecturas*. Gráfico de generación propia mediante la librería *Plotly*. [Creación: 21 oct. 2022]

[Figura 16] *Valores del throughput variando el porcentaje de tipo de operación en Cloud.* Gráfico de generación propia mediante la librería *Plotly*. [Creación: 24 oct. 2022]

[Figura 17] *Valores de la latencia variando el porcentaje de tipo de operación en Cloud.* Gráfico de generación propia mediante la librería *Plotly*. [Creación: 24 oct. 2022]

[Figura 18] *Relación entre latencia y throughput variando el porcentaje de tipo de operación en Cloud.* Gráfico de generación propia mediante la librería *Plotly*. [Creación: 24 oct. 2022]

[Figura 33] *Captura de la página web implementada - sección 3.* Web creada mediante las librerías *Plotly* y *Dash*. [Creación: 02 dic. 2022]

[Figura 34] *Captura de la página web implementada - sección 4.* Web creada mediante las librerías *Plotly* y *Dash*. [Creación: 02 dic. 2022]

[37] *"SpaCy · industrial-strength Natural Language Processing in Python"*. SpaCy [En línea] [Consultado: 20 sept. 2022] <https://spacy.io/>

[38] *"NLTK :: Natural Language Toolkit"*. NLTK [En línea] [Consultado: 20 sept. 2022] <https://www.nltk.org/>

[Figura 35] *Diagrama de Gantt, duración real del período del 15/09-19/10*. Imagen de generación propia utilizando la herramienta Google Sheets. [Creación: 12 dic. 2022]

[Figura 38] *Diagrama de Gantt, planificación del período del 19/10-22/11*. Imagen de generación propia utilizando la herramienta Google Sheets. [Creación: 29 sept. 2022]

[Figura 37] *Diagrama de Gantt, duración real del período del 19/10-22/11*. Imagen de generación propia utilizando la herramienta Google Sheets. [Creación: 12 dic. 2022]

[Figura 38] *Diagrama de Gantt, planificación del período del 12/11-16/12*. Imagen de generación propia utilizando la herramienta Google Sheets. [Creación: 29 sept. 2022]

[Figura 39] *Diagrama de Gantt, duración real del período del 12/11-16/12*. Imagen de generación propia utilizando la herramienta Google Sheets. [Creación: 12 dic. 2022]

[39] "*AWS Pricing Calculator*", Amazon Web Services. [En línea] [Consultado: 5 oct. 2022] <https://calculator.aws/#/estimate?id=4789ea8df0eaca31d2eee1107958cfcb049dad33>

[40] *"Palau-solità i Plegamans (08184): Precio de la Vivienda y precio por m2 en 2022 · EUR / m²"*. Realadvisor. [En línea] [Consultado: 5 oct. 2022] <https://realadvisor.es/es/precios-viviendas/08184-palau-solita-i-plegamans>

[41] "*Oferta de Fibra óptica o ADSL 300Mb - Movistar*". Movistar. [En línea] [Consultado: 5 oct. 2022] <https://www.movistar.es/particulares/internet/adsl-fibra-optica>

[42] "*Coronavirus: por qué las bolsas del mundo llegaron a niveles récord en 2020 mientras la economía global se hundía"*. BBC, BBC News Mundo [En línea] [Consultado: 6 oct. 2022] **<**https://www.bbc.com/mundo/noticias-55536375>