



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Estudio y configuración de un
sistema ELK**



Presentado por Hugo de la Cámara Saiz
en Universidad de Burgos — 8 de julio de 2024
Tutores: Jesús Manuel Maudes Raedo y
Antonio Canepa Oneto



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Jesús Manuel Maudes Raedo, profesor del departamento de nombre departamento, área de nombre área y D. Antonio Canepa Oneto profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Hugo de la Cámara Saiz, con DNI 71566986W, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado *Estudio y configuración de un sistema ELK*.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 8 de julio de 2024

Vº. Bº. del Tutor:

D. Jesús Manuel Maudes Raedo

Vº. Bº. del co-tutor:

D. Antonio Canepa Oneto

Resumen

La gestión de datos y monitorización de sistemas forma parte del esqueleto básico de toda empresa del siglo XXI a la hora de potenciar y maximizar el rendimiento de la misma.

Este trabajo de fin de grado se centrará en explorar lo que nos ofrecen los elementos de un sistema basado en las herramientas ELK (ElasticSearch, Logstash y Kibana) para monitorizar la información y gestionar los datos a la hora de tomar decisiones.

Nos vamos a enfocar en comprender cómo estas herramientas nos permiten recolectar, procesar, almacenar y visualizar la información para poder detectar anomalías, y resolverlas de manera precisa. También vamos a analizar cómo ELK ayuda a mejorar el rendimiento de procesamiento de información y análisis de las diferentes tendencias o comportamientos que puedan surgir en un entorno de la vida real.

Descriptores

ElasticSearch, tratamiento de datos, herramientas ETL, Kibana, Logstash, ...

Abstract

Data management and system monitoring are part of the basic skeleton of any 21st century company when it comes to enhancing and maximizing its performance.

This final degree project will focus on exploring what the elements of a system based on ELK tools (Elasticsearch, Logstash, Kibana) offer us to monitor and manage data to make the best decisions.

We are going to focus on understanding how these tools allow us to collect, process, store and visualize information in order to detect anomalies and resolve them accurately. We will also discuss how ELK helps improve the performance of information processing and analysis of different trends or behaviors that may arise in a real-life environment.

Keywords

ElasticSearch, tratamiento de datos, herramientas ETL, Kibana, Logstash, ...

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vii
1. Introducción	1
1.1. Materiales adjuntos	2
1.2. Estructura de la memoria	2
1.3. Estructura de los anexos	3
2. Objetivos del proyecto	5
2.1. Objetivos del estudio	5
2.2. Objetivos técnicos	6
3. Conceptos teóricos	7
3.1. Definiciones de conceptos	7
3.2. Monitorización de sistemas	9
3.3. Tratamiento de datos (ETL)	9
4. Técnicas y herramientas	11
4.1. ElasticSearch	11
4.2. Logstash	12
4.3. Kibana	14
4.4. Miscelánea	16
5. Aspectos relevantes del desarrollo del proyecto	19
5.1. Evolución temporal	19

5.2. Resultados del estudio	23
6. Trabajos relacionados	49
6.1. PowerBI	49
6.2. Spoon y Qlik	50
6.3. Tableau	52
6.4. Splunk	52
6.5. Comparativa entre alternativas	53
7. Conclusiones y Líneas de trabajo futuras	55
7.1. Conclusiones	55
7.2. Mejoras futuras	56
Bibliografía	57

Índice de figuras

3.1. Estructura de un proceso ETL	10
4.1. Menú principal del <i>Index Management</i> de Elastic.	12
4.2. Ejemplo de uso de filtro <i>mutate</i>	13
4.3. Ejemplo de uso de filtro <i>date</i>	13
4.4. Ejemplo de uso de filtro <i>csv</i>	14
4.5. Menú principal de los <i>Data Views</i> de Kibana.	15
4.6. Apartado <i>Discoverer</i>	15
4.7. Dashboard de Kibana	16
5.1. Dashboard final	24
5.2. Dashboard final del primer escenario	26
5.3. Gráfico de barras por edades y género	26
5.4. Métricas en función del filtrado	26
5.5. Gráfico de donut en función de la clase del billete	27
5.6. Sección filter del archivo de configuración.	29
5.7. Función que renombra los campos indicados	30
5.8. Función que convierte los tipos de los campos indicados	30
5.9. Función en la que se elimina el campo <i>baño</i>	30
5.10. Función en la que se discretiza en función del precio de la vivienda	31
5.11. Dashboard final del escenario 2	31
5.12. Gráfico de barras del número de viviendas por categoría de precio.	32
5.13. Métrica y tabla del número de viviendas.	32
5.14. Gráfico de donut que muestra el porcentaje de viviendas por ciudad y barrio.	33
5.15. Gráfico de árbol que muestra la media de metros cuadrado por ciudad.	33
5.16. Gráfico de barras que muestra el número de habitaciones y de viviendas por barrio.	34

5.17. Datos del índice del escenario 2	34
5.18. Dashboard final del tercer escenario	36
5.19. Dashboard final del cuarto escenario	38
5.20. Plugin <i>aggregate</i> para aplicar MapReduce	40
5.21. Dashboard final del quinto escenario	41
5.22. Gráfico de barras del número de pedidos con cada método de pago	41
5.23. Gráfico de donut del porcentaje de compra en cada categoría . .	42
5.24. Métrica del gasto promedio de cada cliente	42
5.25. Tabla mostrando los clientes más recurrentes	43
5.26. Funcionalidad de Machine Learning de Elastic.	44
5.27. Visualizaciones mostrando información del <i>dashboard</i> así como el rendimiento de la regresión y la precisión del clasificador.	46
5.28. Visualización de dos dimensiones del gráfico obtenido tras aplicar clustering.	47
5.29. Visualización del gráfico obtenido tras aplicar PCA.	48
6.1. Visualizaciones de PowerBI [22]	50
6.2. Estructura de Spoon [18]	51
6.3. Visualizaciones con Qlik [27]	51
6.4. Dashboard principal de Tableau [28]	52
6.5. Dashboard básico de Splunk [2]	53

Índice de tablas

6.1. Comparación entre herramientas ETL	54
---	----

1. Introducción

Con el paso del tiempo, la cantidad de datos presentes en nuestro sistemas y equipos es cada vez de mayor tamaño, lo que implica que el manejo de los mismos adquiera una complejidad notoria.

Las empresas están al tanto de esto y ya son muchas las que incorporan en su plantel un departamento dedicado al estudio de los datos. Dicho departamento recibe el nombre de Data Science, y las personas que trabajan en él, el nombre de Data Scientists.

En la actualidad existen diferentes maneras de abordar este problema de sobrecarga de datos para poder analizarlos, filtrarlos y mostrarlos de manera que sean transformados en información de utilidad real y concisa. Es por esa razón por la que en este trabajo vamos a abordar una de las posibles soluciones: la creación y configuración de un sistema basado en la arquitectura ELK, conformada por los programas ElasticSearch, Logstash y Kibana. Para todo ello este trabajo se apoya en lenguajes de programación como Python entre otros, además de manipulación de ficheros tipo JSON, logs,...

Analizaremos situaciones que se le pueden presentar a un particular o empresa en su día a día, sea del ámbito que sea, profundizando en las posibilidades que nos ofrecen este conjunto de herramientas a la hora de ingestar datos de diversas maneras.

Estos escenarios estarán clasificados en función de la forma en la que se ingestan los datos. Procesándolos de manera diferente y aplicando filtros.

1.1. Materiales adjuntos

- Memoria del proyecto
- Anexos del proyecto
- [Repositorio GitHub de este proyecto](#)
- [Videotutoriales](#)
- [Máquina virtual](#)

1.2. Estructura de la memoria

La documentación estará formada en primer lugar por la memoria, documento en el que se encuentra toda la información relacionada con qué se ha hecho, el resultado obtenido y el por qué de este resultado, dividiendo la estructura en los siguientes puntos:

1. Introducción

En esta sección abordamos el contenido clave del trabajo, además de la estructura de la documentación del mismo.

2. Objetivos del proyecto

En este apartado se explica cuáles son los objetivos a conseguir con la realización del proyecto.

3. Conceptos teóricos

En este apartado haremos énfasis en los contenidos de tipo teórico que sean necesarios conocer para poder comprender el proyecto.

4. Técnicas y herramientas

Se expondrán los diferentes programas y técnicas empleados para la realización del proyecto así como descripciones de los mismos, su utilidad, etc.

5. Aspectos relevantes del desarrollo del proyecto

En este apartado recopilaremos los aspectos más interesantes del desarrollo del proyecto, como puede ser la motivación del mismo, la evolución cronológica o las dificultades encontradas. También se incluyen los resultados del estudio incluyendo las configuraciones experimentadas.

6. Trabajos relacionados

En esta sección incluiremos un pequeño resumen comentado los trabajos y proyectos ya realizados en el campo del proyecto en curso, así como una comparación de los mismos-

7. Conclusiones y líneas de trabajo futuras

Se hará un síntesis del proyecto en su totalidad así como recomendaciones de posibles mejoras a realizar de cara a trabajos futuros.

1.3. Estructura de los anexos

La otra parte de la documentación la conforman los anexos, en los que se hará más hincapié en el proceso seguido así como los detalles técnicos para la explotación y mantenimiento del mismo. Al ser este proyecto un estudio de un sistema y no un TFG consistente de un desarrollo informático, la estructura difiere ligeramente de la habitual. Concretamente, no hay manual de usuario como tal, pero se ha sustituido a cambio de un anexo estudiando las posibilidades gráficas de Kibana.

A. Plan de proyecto

Este apartado consistirá en un análisis de como se estructurará el proyecto temporalmente así como la situación tanto económica como legal.

B. Requisitos

Trataremos las diferentes tareas que se han ido desarrollando a lo largo del estudio, así como un caso de uso por cada escenario estudiado.

C. Diseño

Se representarán los roles de los tres elementos software que componen el sistema ELK, más una fuente de datos, añadiendo diagramas UML para cada escenario analizado.

D. Manual del programador

En esta sección se explica todo lo que le incumbiría a un posible programador que se quiera adentrar en el proyecto, permitiéndole replicar los experimentos estudiados.

E. Manual de Kibana

Este apartado está pensado de cara a un posible cliente real para que pueda comprender minuciosamente el funcionamiento del programa de visualizaciones gráficas del ecosistema ELK.

F. Anexo de sostenibilización curricular

Se hará una breve reflexión personal sobre los aspectos de la sostenibilidad en los que contribuye este trabajo.

2. Objetivos del proyecto

El objetivo principal de este trabajo es estudiar de manera amplia el abanico de posibilidades que nos ofrece un sistema ELK que sea capaz de completar todos los requisitos que pueda tener un particular o empresa en un entorno de ingesta, procesamiento, análisis y visualización de datos. Al tratarse este trabajo de un estudio y no de un desarrollo se ha modificado la estructura de el apartado "*Objetivos del software* .^a "Objetivos del estudio"^z se ha seguido con la sección de objetivos de tipo técnico al ponerse en práctica lo estudiado.

2.1. Objetivos del estudio

- Plantear una estructura de la arquitectura que sea fácil de comprender y aplicar a la hora querer mejorar la comprensión del flujo de procesamiento de los datos.
- Crear distintos escenarios en los que, modificando la fuente de la ingesta de datos, podamos exponer todo el potencial de las funciones de Logstash, Machine Learning y los dashboards de Kibana.
- Procesar datos en tiempo real a través de protocolos de red como WebSockets e integrarlo en nuestro ecosistema.
- Poder procesar una ingesta masiva de datos a través de la aplicación de Map Reduce.
- Poder detectar fallos o anomalías que puedan surgir en los registros de manera rápida y efectiva.

- Documentar de manera adecuada todo lo necesario para gestionar un sistema ELK, de manera que cualquier usuario pueda comprender su funcionamiento y elegir la configuración que más le convenga.
- Encontrar una solución gratuita de integración de Machine Learning en el sistema ELK.
- Intentar adaptar los escenarios estudiados lo máximo posible a situaciones reales que se puedan plantear.

2.2. Objetivos técnicos

- Emplear las tecnologías que nos ofrece el stack de herramientas ELK basado en el ecosistema Elastic para hacer un despliegue del mismo.
- Diseñar scripts de apoyo en lenguajes de programación como Python.
- Emplear las funciones de Machine Learning que nos ofrece la librería scikit-learn e implementarlas en nuestro ecosistema.
- Mantener actualizaciones de los progresos y avances en el proyecto a través de las issues en el repositorio GitHub.
- Generar distintas *pipelines* de ingestión de datos aplicando filtros y transformaciones a las mismas.
- Crear diferentes índices en Elastic para cada escenario basado en situaciones de un entorno real que vamos a generar.
- Plasmar todo este tratamiento de los datos en un resultado final en Kibana a través de sus *dashboards*.

3. Conceptos teóricos

En esta sección voy a tratar algunos temas de carácter teórico que me parecen interesantes de conocer de cara a la comprensión básica de este proyecto.

3.1. Definiciones de conceptos

IoT

Este concepto hace referencia a todos los objetos físicos con algún tipo de sensor o software capaz de almacenar y procesar información que están interconectados a través de internet u otra forma de comunicación[9]. Estos sensores pueden ser manejados a través de ordenadores, móviles o tablets que posean la característica de ser *inteligentes*. Ejemplos del uso de este están en la domótica de las casas, sistemas sanitarios o monitoreo ambiental.

IoT es una fuente de datos potencial que tiende cada vez a ser más común presentando problemas de procesamiento de grandes volúmenes de datos. En este estudio, abordaremos ese problema planteando una posible solución basada en el uso de MapReduce.

Map Reduce

Este modelo de programación (*framework*) nos sirve como ayuda para la computación paralela de grandes volúmenes de datos[12]. Consiste en establecer una serie de normas y parámetros para que los datos seán reagrupados en función de los parámetros indicados, mejorando así la eficiencia de procesamiento y ahorrando espacio de almacenamiento en la fuente destino. También proporciona la obtención de agregados estadísticos, como pueden

ser la media, la moda o la varianza, los cuáles resultan interesantes a la hora de realizar visualizaciones gráficas.

Big Data

Este término hace una referencia general a los voluminosos conjuntos de datos que se procesan en las actuales aplicaciones informáticas[8]. La clave está en encontrar patrones que se repitan en estos datos para poder analizar su comportamiento de forma precisa, plasmarlos en documentos estadísticos, visualizarlos y poder aplicar *machine learning* [10].

WebSocket

Se trata de un protocolo de conexión que va a actuar como medio de comunicación bidireccional (*full-duplex*) entre un servidor Web o API y un lugar destino especificado en el código del mismo[21]. Tiene una estructura de aplicación cliente/servidor y nos va a ser útil de cara a procesar escenarios con *Data Streams*. La función que cumplirá en el estudio, será la de método de suscripción a un envío de datos en vivo, de manera que permita seleccionar a donde se mandan esos datos y con que forma.

3.2. Monitorización de sistemas

La monitorización de sistemas^{[1][14][30]} es la función que se encarga de gestionar el estado tanto de la infraestructura como del sistema. Aplicarlos en nuestro entorno tiene una serie de ventajas:

1. **Configuración de alarmas** Pudiendo ser más o menos severa un sistema monitorizado responderá notificando casos como puede ser el llenado de la memoria o la alta demanda de recursos en circulación
2. **Detección temprana de amenazas** Se tiene en cuenta 2 medidas a la hora de la detección: la precisión y la velocidad. Pudiendo variar ambas en función de la criticidad de la alarma.
3. **Detectar el origen de los incidentes** En caso de alarma, un sistema bien monitorizado nos puede ayudar a aislar el problema e identificar que es lo que está fallando y en qué momento, para su posterior eliminación y solución.
4. **Reducir los costes** La rentabilidad de un sistema aplicando la monitorización del mismo aumenta sustancialmente, ahorrando costos cuando se toman decisiones y mejorando la escalabilidad.

Las grandes corporaciones ya se han hecho eco de este gran avance, y son muchas las que han desarrollado sus propias herramientas para uso privado dentro de sus ecosistemas. Pero cabe destacar algunas de código abierto y amplio uso: Nagios, Zabbix, Grafana, entre otras son las punteras en el sector freeware de monitorización de datos.

3.3. Tratamiento de datos (ETL)

ETL^{[31][11]} es un proceso pensado para gestionar datos provenientes de diferentes fuentes, poder moldearlos a gusto de cada uno y cargarlos en una base de datos para su posterior análisis y exposición. El proceso está subdividido en 3 partes esenciales:

- (E) Extract
- (T) Transform
- (L) Load

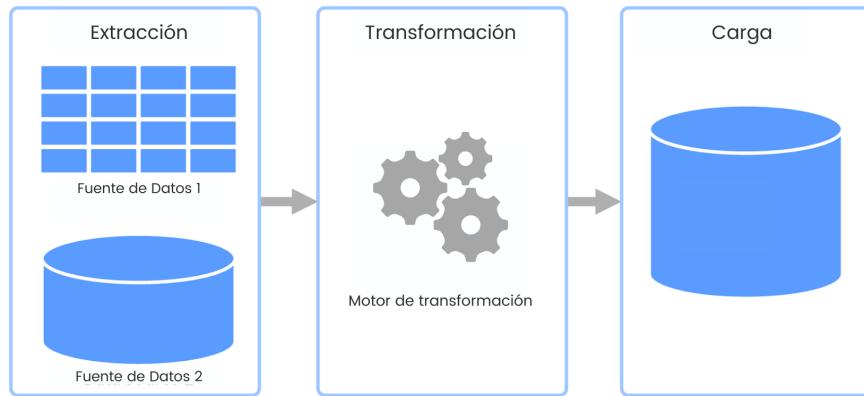


Figura 3.1: Estructura de un proceso ETL

[13]

Extracción de Datos: en este primer sector del proceso se va a encargar de recopilar toda la información de un lugar origen que se le indique. Esta parte hará el trabajo más costoso, potencialmente hablando, gastando los menos recursos posibles.

Transformación de Datos: en esta segunda fase los datos van a ser transformados y limpiados, ya que las herramientas que pertenecen a esta parte del proceso ETL permiten: cargar solo ciertas secciones o columnas, modificar tipos de variables, agrupar los datos en rangos preseleccionados o generar nuevos campos a partir de los existentes.

Carga de Datos: en la última parte del proceso consiste en que una vez tengamos los datos transformados, estos vayan siendo cargados en el lugar destino que le indiquemos (e.g., ElasticSearch) para que allí sean indexados y gestionados de una manera concreta.

4. Técnicas y herramientas

En este apartado se muestran y desarrollan todas las técnicas y herramientas empleadas para completar el estudio y su posterior documentación. Esta sección fundamentalmente se centra en explicar los tres elementos del *stack ELK* que se compone de ElasticSearch, Logstash y Kibana, tal y como indica su acrónimo. ElasticSearch almacena la información proveyendo un acceso eficiente, Logstash la transforma antes de ser almacenada y Kibana la presenta de forma atractiva mediante *dashboards*.

4.1. ElasticSearch

Este programa de código abierto va a ser la piedra angular y sobre la que van a pivotar el resto de elementos software que van a componer las distintas configuraciones a experimentar. Consiste en un motor de búsqueda analítica y distribuida de cara a almacenar todos los datos de un proyecto. [6]

En el presente trabajo se ha empleado el servicio local de Elastic, teniendo que descargar el programa íntegro y operar desde ahí. Se nos ofrece una versión Cloud que promete grandes mejoras en el rendimiento, pero no vimos oportuna su adquisición puesto que el estudio está pensado para herramientas *freeware* sin suponer ningún coste económico.

Funciona como una base de datos estructurada en índices que agrupan colecciones de datos sustituyendo a lo que serían, por ejemplo, tablas de las bases de datos relaciones. Elastic no está enfocado en trabajar con transacciones CRUD, sino en indexar grandes volúmenes de información reduciendo todo lo posible los tiempos de acceso a los datos. Cada índice estructura la información como documentos JSON de manera que cada fila de información quede ordenada y accesible. Desde el apartado *Index*

The screenshot shows the Elasticsearch Index Management interface. At the top, there are tabs for Indices, Data Streams, Index Templates, Component Templates, and Enrich Policies. The Indices tab is selected. Below the tabs, there is a search bar and filters for Lifecycle status and Lifecycle phase. Two checkboxes are present: 'Include rollup indices' and 'Include hidden indices'. A 'Reload indices' button and a 'Create index' button are also visible. The main area displays a table of indices with columns: Name, Health, Status, Primaries, Replicas, Docs count, Storage size, and Data stream. The indices listed are:

Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
casas	yellow	open	1	1	80	64.08kb	
iris	yellow	open	1	1	570	95.93kb	
kibana_sample_data_ecommerce	yellow	open	1	1	4675	4.16mb	
test-data-stream	yellow	open	1	1	31642	31.77mb	
titanic	yellow	open	1	1	891	141.17kb	
transactions_aggregate	yellow	open	1	1	3	15kb	

Figura 4.1: Menú principal del *Index Management* de Elastic.

Management, Elastic permite visualizar los diferentes índices presentes en el sistema (ver ilustración 4.1).

4.2. Logstash

Actuará como intermediador entre la fuente de datos y Elastic, es decir, cubrirá todo lo relacionado tanto con la parte de transformación de datos como con la de carga de los mismos. Cabe aclarar que se pueden cargar datos en Elastic sin Logstash, pero entonces se cargarían tal y como estén, por lo que todas las transformaciones necesarias tendrán que haberse hecho previamente.

Logstash permite definir *pipelines* para procesar datos del lado de la fuente de ingesta de datos que permite transformarlos para después enviarlos y cargarlos en el destino indicado (i.e., ElasticSearch) [4].

Logstash está enfocado tanto a la ingesta de datos, permitiendo cargar datos de múltiples orígenes con compatibilidad con protocolos, como al envío de datos a diversos destinos mediante un *pipeline* configurable.

En este *pipeline*, se pueden realizar modificaciones a los datos que entran. Algunos ejemplos de transformaciones con Logstash pueden ser estos:

- El filtro *mutate* permite cambiar nombres de campos, añadir nuevos o eliminar los que se consideren redundantes. Un ejemplo de su uso es la figura 4.2 en la cuál se realiza un cambio de nombre de un campo, se añade un campo nuevo y se borra un campo.

```
filter {
  mutate {
    rename => { "ClientName" => "Cliente" }
    add_field => { "Fecha recibida" => "%{@timestamp}" }
    remove_field => [ "@version", "host" ]
  }
}
```

Figura 4.2: Ejemplo de uso de filtro *mutate*

```
filter {
  date {
    match => [ "Fecha", "ISO8601" ]
    target => "@timestamp"
  }
}
```

Figura 4.3: Ejemplo de uso de filtro *date*

- El filtro *date* permite analizar campos que sean fechas indicándole con qué formato están escritas. Un ejemplo de su uso es la figura 4.3.
- El filtro *csv* permite analizar los campos presentes en un fichero de tipo CSV de manera que divida cada línea en campos indicados. Un ejemplo de su uso es la figura 4.4.

```

filter {
  csv {
    separator => ","
    columns => ["timestamp", "Cliente", "Identificador"]
  }
  date {
    match => ["timestamp", "YYYY-MM-dd HH:mm:ss"]
    target => "@timestamp"
  }
}

```

Figura 4.4: Ejemplo de uso de filtro *csv*

4.3. Kibana

Por último, la parte de final del stack esta protagonizada por este programa, que se encargará de ejecutar las analíticas de los datos mandados por Logstash a Elastic y de mostrarlos en diferentes *dashboards* con gráficas de datos que nos permitirá generar interacciones que resulten útiles e interesantes [5].

Kibana permite administrar los datos presentes en Elastic desde el apartado *Data Views*, facilitando el acceso a los diferentes índices de datos. Un *Data View* consiste en una representación de los datos de un índice para que pueda haber representaciones de los datos en un *dashboard* (ver ilustración 4.5).

Una vez se accede una *view*, el apartado *Discoverer* es la herramienta que facilita Kibana para poder ver de manera completa la información de los datos presentes en un índice. Se pueden aplicar filtros o clasificaciones para encontrar determinados datos (ver ilustración 4.6).

Por últimos, la función principal de Kibana es crear *dashboards* con visualizaciones y demás herramientas sobre los datos presentes en Elatic a través del apartado *Dashboards*, en el cuál se desarrollan tanto la creación como modificación de estos (ver ilustración 4.7).

Data Views

[Create data view](#)

Create and manage the data views that help you retrieve your data from Elasticsearch.

Search...		Spaces	Actions
<input type="checkbox"/>	Name ↑		
<input type="checkbox"/>	Kibana Sample Data eCommerce ⓘ Default	D	Edit
<input type="checkbox"/>	IRIS ⓘ	D	Edit
<input type="checkbox"/>	Logstash Stack Monitoring Metrics ⓘ	D	Edit
<input type="checkbox"/>	Viviendas ⓘ	D	Edit
<input type="checkbox"/>	casas ⓘ	D	Edit
<input type="checkbox"/>	test-data-stream ⓘ	D	Edit
<input type="checkbox"/>	titanic ⓘ	D	Edit
<input type="checkbox"/>	transactions ⓘ	D	Edit
<input type="checkbox"/>	websocket ⓘ	D	Edit

Figura 4.5: Menú principal de los *Data Views* de Kibana.

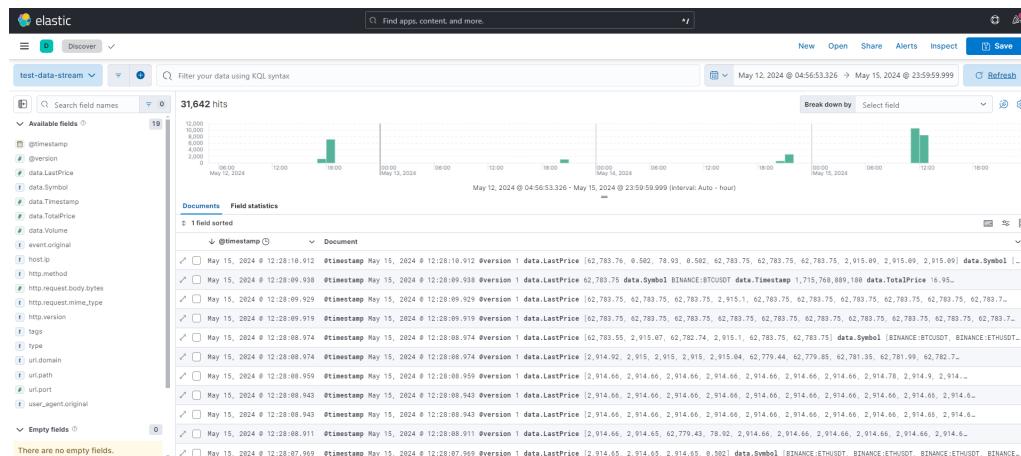


Figura 4.6: Apartado *Discoverer*

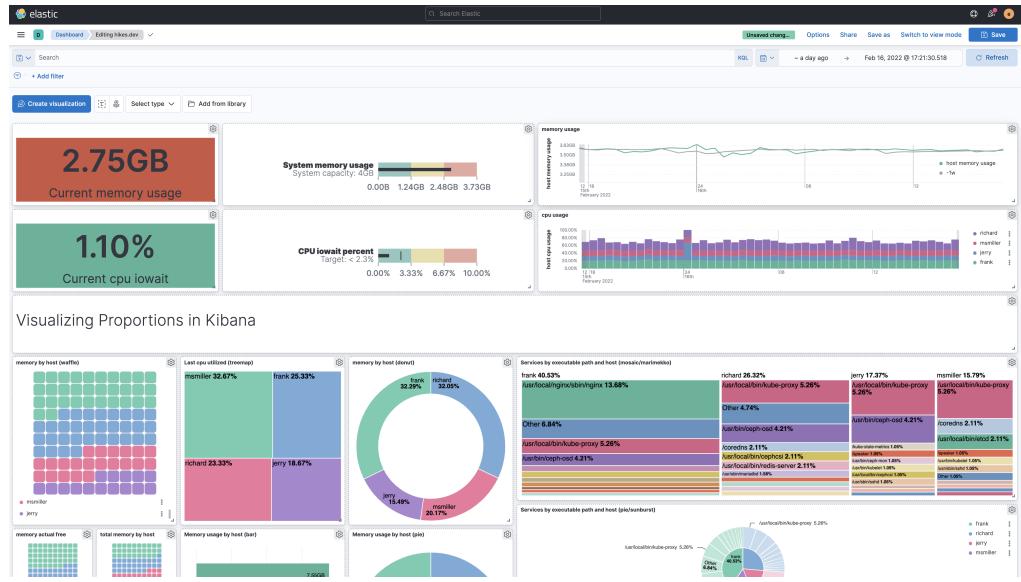


Figura 4.7: Dashboard de Kibana

4.4. Miscelánea

Scikit-Learn

Biblioteca de aprendizaje automático de código abierto para *Python* que permite utilizar algoritmos de clasificación, regresión, clustering y reducción de características a los datos que se le indiquen [7].

Jupyter Notebook

Aplicación web basada en *Python* que se usará para crear y compartir documentos en los que realizamos cálculos computacionales con el resto del entorno de trabajo de los escenarios planteados [25].

Python

Este lenguaje de programación permite el diseño e implementación de manera sencilla y eficiente de *scripts*, en los cuales se emplean diferentes bibliotecas que utilizaremos para procesar los datos de diferentes maneras. Una vez finalizados son integrados en nuestro ecosistema de trabajo para ofrecer ayuda en distintos escenarios [26].

Visual Studio Code

Editor de código fuente desarrollado por Microsoft que permite crear, depurar y procesar archivos de código con total soltura. En este proyecto será empleado como apoyo para los diferentes *scripts* de procesamiento de datos realizados [19].

Overleaf

Editor online de LATEX que permite la edición y exportación de documentos con este formato de manera fácil y rápida. En este proyecto tiene utilidad en la parte de la documentación tanto de la memoria como de los anexos [23].

GitHub

Este *software* sirve como alojamiento de proyectos, permitiendo modificaciones en el mismo, creaciones de archivos, modificaciones y demás. Será utilizado para mantener un orden en cuánto a la evolución del proyecto [16].

5. Aspectos relevantes del desarrollo del proyecto

En esta sección de la memoria se mostrarán las diferentes fases de maduración y evolución por las que ha pasado el proyecto, haciendo especial hincapié en las dificultades encontradas a la hora de afrontar distintas tareas. También se mostrarán los resultados obtenidos en el estudio.

5.1. Evolución temporal

Inicio y motivación

A principios de curso, tuve la oportunidad de realizar las prácticas curriculares en la consultoría informática local CSA, más concretamente en el departamento de Business Intelligence. Ahí estuve trabajando durante meses con distintas herramientas de la categoría ETL que me resultaron prácticas e interesantes. Por lo que una vez finalizadas las prácticas, le propuse a mi tutor de las mismas la posibilidad de realizar mi trabajo de fin de grado sobre algo relacionado con el tratamiento de datos. Y así llegamos a la conclusión de empezar a trabajar con las herramientas ElasticSearch, Kibana y Logstash, que cumplían el papel de las tres fases de un proceso ETL.

La intención en este proyecto siempre fue desde el principio un estudio de estas herramientas ELK en el que se representarán situaciones que pudieran ocurrir en un entorno real. Por lo que desde un inicio, se pensó en plantear distintos escenarios en función de la manera en la que se ingestaban los datos.

El primer interrogante que se planteó por parte del tutor, Jesus Manuel Maudes Raedo, fue el de investigar que posibilidades ofrecía Logstash a la hora de ingestar, filtrar y exportar datos. En un principio, la idea se antojaba sencilla pero debido a que estos programas son novedosos y no han tenido tanto impacto (aún) como otros como puede ser PowerBI, la documentación presente en Internet sobre conceptos relacionados con este programa era escasa y en su mayoría en lengua anglosajona. Por lo que se comenzó desde lo más básico, tutoriales y guías tanto de plataformas como YouTube, como de las páginas oficiales de Elastic.

La familiarización con el programa Logstash llevó un gran lapso de tiempo, por lo que se compaginó con investigar el funcionamiento de los plugins que ofrece Elastic desde su *marketplace*, comprender como se podía realizar integraciones con *Python* en el ecosistema Elastic, y estudiar la posibilidad de integrar *machine learning* en el análisis de los datos, y este fue el siguiente punto que se tocó en profundidad.

En este periodo, se completó el primero de los escenarios a analizar y más sencillo, el de importar directamente un fichero de tipo CSV a Elastic. El fichero elegido fue uno que incluía datos sobre los pasajeros del Titanic y que ofrecía la posibilidad de mostrar diferentes herramientas que nos ofrece Kibana a la hora de mostrar los datos. También se analizó un ejemplo de datos que nos ofrece Elastic que permite explotar al máximo las capacidades de análisis de Kibana. Se trata de los datos de una empresa electrónica que se dedica al comercio de artículos, y gracias a este escenario se pudo comprender con mayor facilidad el funcionamiento de Kibana de cara a desarrollar los siguientes escenarios que se avecinaban.

Cronología de los experimentos

Machine Learning

Habiendo pasado ya un mes desde el comienzo del estudio, la posibilidad de integrar análisis de datos con *machine learning* se antojaba fundamental, pues a priori era una función de pago que ofrece Elastic, por lo que esa vía de integración quedó descartada.

Se introdujo la idea de trabajar con la biblioteca de aprendizaje automático de código abierto *Scikit-Learn*. Para el segundo sprint del proyecto, se giró entorno al estudio, comprensión e investigación de esta biblioteca que se consiguió integrar a través la herramienta *Jupyter Notebook*, en la que se realizaron distintos scripts de pruebas para comprender el funcionamiento de los algoritmos de clasificación, regresión, clustering y reducción de características sobre el *dataset Iris*, un popular conjunto de datos empleado como ejemplo de la utilidad del *machine learning* en el análisis de los mismos. Este conjunto de datos ya lo tenemos almacenado en Elastic, y es desde donde lo cargamos al script para poder trabajar sobre esa información. Una vez los los algoritmos los han procesado, estos datos son ingestados en índices individuales en ElasticSearch. Una vez se hubo finalizado la carga de los resultados, estos fueron expuestos en un *dashboard* en la herramienta visual Kibana.

Al no haber cursado la asignatura optativa de Minería de Datos, este proceso de aprendizaje fue costoso y ocupó un gran período de tiempo hasta que se comprendieron de manera correcta los conceptos tratados.

También se mencionó la posibilidad de realizar funciones como Map Reduce en algún momento del ETL, y se dedicó un tiempo a investigar en qué momento y en qué herramienta era más óptimo implementar esta función.

Data Streams

Con lo mencionado anteriormente se finalizó el segundo sprint y dieron comienzo los dos siguientes, en los que la misión principal fue la de estudiar el funcionamiento de los *streams* de datos, cómo funcionan y de qué forma los podemos ingestar en Elastic.

El tutor de este proyecto proporcionó distintas fuentes para encontrar *streamings* de datos con los que poder experimentar y estudiar las posibilidades reales que ofrecen. Se continuó por un repositorio de GitHub de nombre [awesome public streaming datasets](#) en el que se incluían distintas interfaces via *WebSockets* de *data streams*. El problema venía dado cuando queríamos ingestar estos *streamings* con Elastic, ya que bastantes de las APIs que se ofrecían eran complejas, y las que no, ofrecían envíos de datos muy básicos. A todo esto se le suma que no habiendo cursado la asignatura de Sistemas Distribuidos, en la cuál se estudiaba el funcionamiento de los *WebSockets* detenidamente, me supuso un mayor tiempo para la compresión de estos conceptos.

Por lo que se optó por una opción que era gratuita y satisfacía en gran parte los requisitos que se pedían: la *RESTful API* y *WebSocket* de la empresa de operaciones de criptomonedas **Finnhub**.

Gracias a la guía que presentan su [página web](#), y los diferentes tutoriales y documentación presente en Internet, se consiguió desarrollar un script en Python que se suscribiese al *stream* de datos y mandará los datos directamente a Elastic, sin realizar ninguna operación intermedia, completando así otro de los escenarios tratados.

En vista de que existía la posibilidad de realizar alguna manipulación o transformación a los datos originales que Finnhub mandaba, se optó por generar un script que modificará el nombre de las variables y mandará a Logstash los datos del *stream* para poder realizar más filtros desde allí. El envío de datos del script a Logstash se realiza a través del protocolo de red HTTP, a través del puerto 8080 más concretamente.

Una vez los datos llegan a Logstash, se les realiza unas modificaciones, omitiendo el envío de campos vacíos, modificando el tipo de las variables para que sean legibles por Elastic, y generando nuevos campos en base a los originales realizando operaciones sobre ellos. Una vez que los filtros se realizan los datos procesados por Logstash son mostrados por pantalla y enviados a Elastic para que desde allí puedan ser ilustrados y expuestos a través de Kibana, dando fin así a otro de los posibles escenarios analizados.

Cabe remarcar que al encontrar dificultades en el envío de datos a través de las fuentes del repositorio mencionado, se estudió paralelamente la posibilidad de mandar los datos con la herramienta *Filebeat*, la cuál ofrece una integración sencilla con Elastic pero teniendo acceso a *streamings* de datos más complejos. Por lo que añadir esta opción hubiera sido aumentar la carga de trabajo, por lo que se deja como posible mejora de cara a trabajos futuros.

5.2. Resultados del estudio

Introducción

Una vez maduradas las posibilidades que ofrece el sistema ELK, se ha decidido estructurar este apartado en varios, en función de las conclusiones sacadas. Se hará hincapié en los conceptos teóricos mencionados previamente como pueden ser MapReduce o Machine Learning entre otros.

ELK con Elastic en la nube

Elastic ofrece la versión de trabajo *cloud*, en la que se dispone de un espacio de almacenamiento para los distintos índices de datos de manera que se puedan trabajar con ellos desde distintas ubicaciones. Esta funcionalidad es ofrecida pagando un precio efectivo, por lo que no lo contemplamos como posibilidad de trabajo. Pero de forma gratuita, Elastic pone a servicio del usuario diversas fuentes de datos en su nube para poder experimentar y aprender a sacar el máximo potencial tanto a ElasticSearch como a Kibana.

Tras evaluar la situación, se llegó a la conclusión de que los datos utilizados en los diferentes escenarios permitían a Kibana mostrar parte de su potencial, pero no todo. Por lo que para mostrar la herramienta al completo se utilizaron datos que nos ofrece Elastic a modo de ejemplo.

En este caso, se eligieron los datos sobre un *e-commerce* el cuál recibe pedidos de todo el mundo y de distintos tipos. Permitiendo así poder mostrar visualizaciones de mapas, histogramas y demás funciones avanzadas, como pueden ser los distintos gráficos de líneas para evaluar la evolución de los gastos por categorías o gráficos de barras en lo que se muestra la evolución de las ganancias a lo largo de un marco temporal.

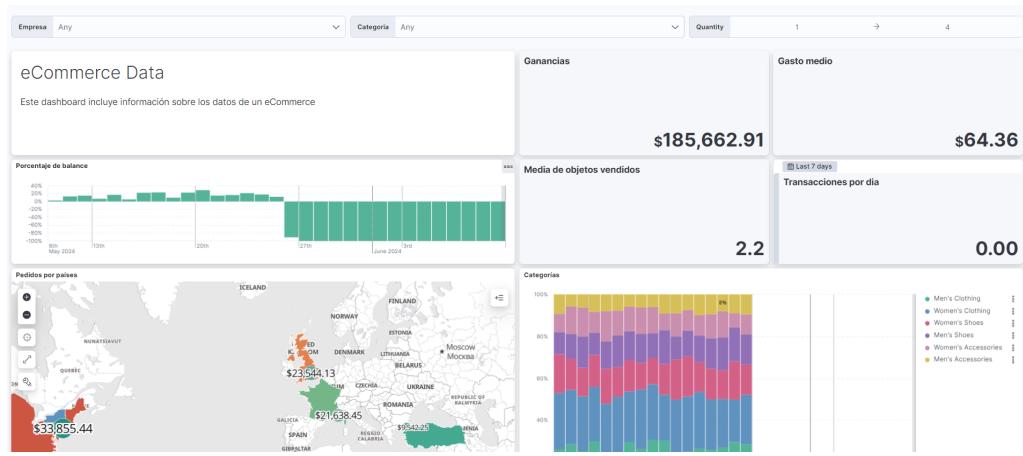


Figura 5.1: Dashboard final

Escenarios de ingesta de datos

En este apartado se hará un desglose de los diferentes escenarios de ingesta de datos de diversas fuentes sobre los que se ha trabajado en este estudio incluyendo su desarrollo y resultados finales.

Escenario 1: ingesta desde fichero en Elastic

Este primer escenario es el más simple de todos, consiste en la importación de un fichero local a Elastic a través del servicio de importación de datos básico que se ofrece.

El archivo que se va a analizar recibe el nombre de *titanic.csv*, es un archivo de valores separados por comas que muestra datos sobre los pasajeros que formaban parte de la tripulación del hundido barco *Titanic*. Es un archivo interesante como punto de partida para entender el funcionamiento de las herramientas software empleadas.

Teniendo este archivo en el sistema local, en el apartado que dice "import from file", y desde ahí se selecciona el mencionado CSV. Una vez que lo seleccionado, Elastic cargará los datos y se podrán visualizar desde la pestaña *Discover*, en la que aparecerán las filas del fichero. Desde aquí se pueden clasificar y filtrar de manera que se pueda comprobar que los datos han sido cargados correctamente.

Una vez tenemos los datos cargados, ya se puede crear el *Dashboard* pertinente accediendo a la pestaña *Visualize* y desde aquí a *Dashboard*. Una vez ahí, se seleccionará cuál es la fuente que proveerá de datos las visualizaciones, seleccionamos la del archivo que se acaba de importar, y ya se podrá empezar a generar visualizaciones.

La estructura básica estará formada por un texto descriptivo del *dashboard* en formato Markdown incluyendo datos del autor, así como descripciones de las distintas visualizaciones presentes. Este elemento es fundamental en todo *dashboard* cumpliendo el papel de introducción a la hora de la toma de contacto con el contexto de las visualizaciones.

En el *dashboard* resultante se han añadido gráficos de las edades (ver ilustración 5.3), los géneros, métricas (ver ilustración 5.4), y gráficos de donut (ver ilustración 5.5) en función de la clase del pasajero, la cuál se puede filtrar en función de lo que interese desde la barra superior dónde están presentes los diferentes filtrados que se pueden hacer por género del pasajero, edad de los mismos, por qué puerta de embarque accedieron, la clase de su billete o si sobrevivieron (ver ilustración 5.2).

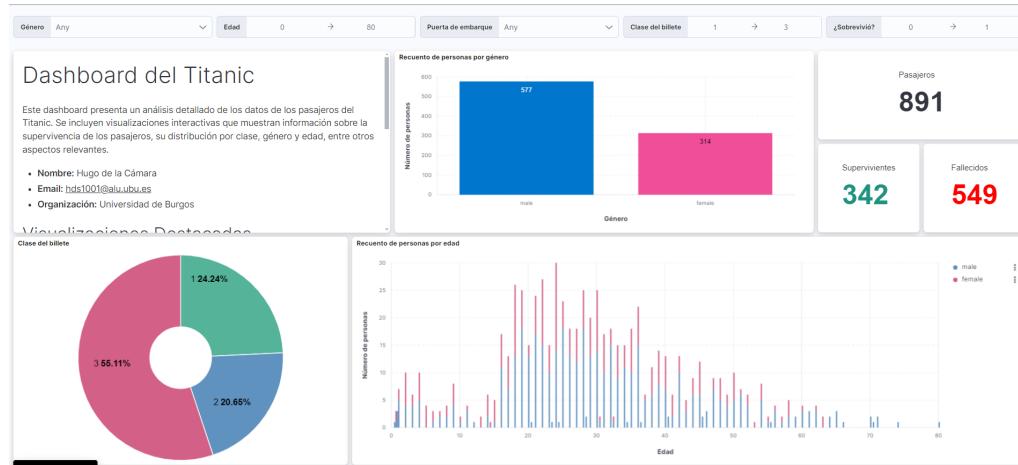


Figura 5.2: Dashboard final del primer escenario

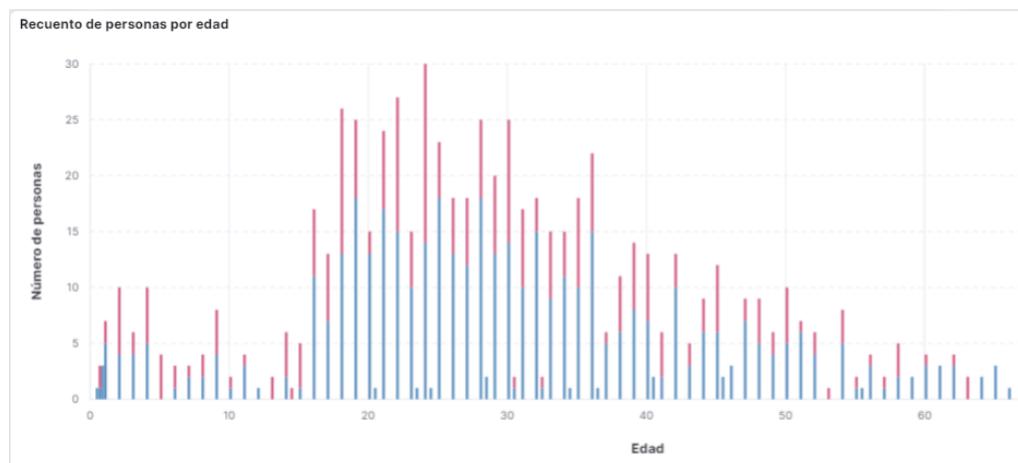


Figura 5.3: Gráfico de barras por edades y género

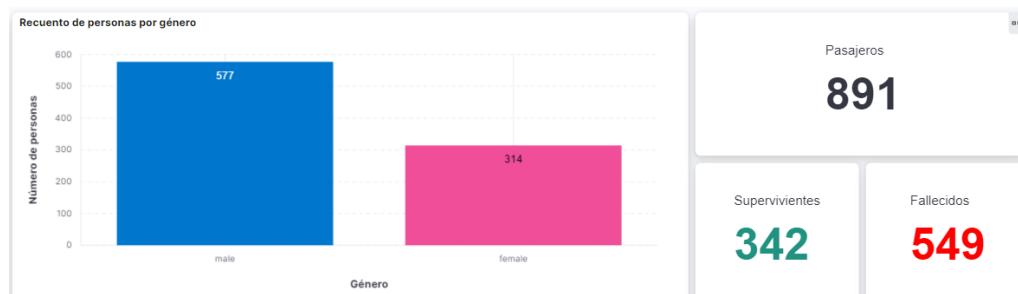


Figura 5.4: Métricas en función del filtrado

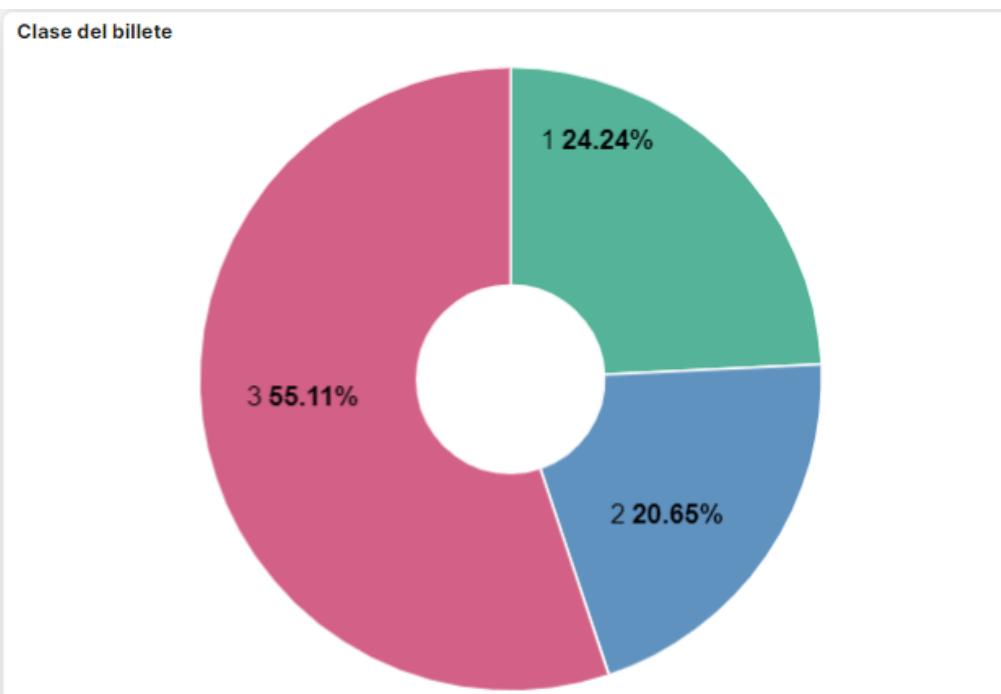


Figura 5.5: Gráfico de donut en función de la clase del billete

Escenario 2: ingestá desde fichero con Logstash

Este segundo caso, es una modificación del primero, lo que se pretende es importar un archivo presente en el sistema en Logstash, realizarle una serie de modificaciones y cargar esos datos tratados en Elastic para poder visualizarlos en Kibana.

El archivo sobre el que se va a trabajar es uno de tipo *log* que indica las diferentes viviendas ofertadas en una inmobiliaria, incluyendo campos como el número de la casa, la ciudad en la que se ubican, el barrio en el que se encuentra, el precio de la misma, el número de habitaciones que tiene, el número de baños disponibles y los metros cuadrados de la vivienda.

Para trabajar con logstash, se necesita un archivo de configuración para que éste entienda lo se le pide que haga. Este archivo es de tipo *.conf* e incluye 3 apartados:

- **Input:** ruta hacia el archivo que se quiere leer
- **Filter:** operaciones a realizarle a estos datos
- **Output:** salida por pantalla de los datos mandados a Elastic una vez son tratados.

En este escenario se le va a dar toda la relevancia al apartado *filter* del fichero de configuración (ver ilustración 5.6), que va a ser en el que se realizan las transformaciones de los datos.

La primera operación que se va a realizar es la de modificar el nombre de los campos *numero* por *num casa* y de *habitaciones* por *num habitaciones* con la función *rename* (ver ilustración 5.7), la cuál permite cumplir esta orden. Esto es de gran utilidad cuándo el nombre de algún campo no deja claro la información que éste ofrece.

La siguiente transformación que se ha aplicado es la de cambiar los tipos de los campos *precio* y *metros cuadrados* con la función *convert* (ver ilustración 5.8), la cuál es de gran ayuda cuando los campos que se van a mandar a Elastic no poseen un tipado que permita el correcto procesamiento de la información.

La tercera operación aplicada en este filtro va a consistir en la eliminación del campo *baños* con la función *remove field* (ver ilustración 5.9), la cuál permite eliminar del mensaje final mandado a Elastic contenido que no se considere oportuno o que carece de relevancia.

```
filter {  
  
  mutate {  
  
    rename => { "numero" => "num_casa" }  
    rename => { "habitaciones" => "num_habitaciones" }  
  
    convert => { "precio" => "float" }  
    convert => { "metros_cuadrados" => "float" }  
  
    remove_field => ["baños"]  
  
    if [precio] {  
      if [precio] < 100000 {  
        mutate {  
          add_field => { "categoria" => "Barata" }  
        }  
      } else if [precio] >= 100000 and [precio] < 300000 {  
        mutate {  
          add_field => { "categoria" => "Intermedia" }  
        }  
      } else {  
        mutate {  
          add_field => { "categoria_precio" => "Cara" }  
        }  
      }  
    }  
  }  
}
```

Figura 5.6: Sección filter del archivo de configuración.

```
rename => { "numero" => "num_casa" }
rename => { "habitaciones" => "num_habitaciones" }
```

Figura 5.7: Función que renombra los campos indicados

```
convert => { "precio" => "float" }
convert => { "metros_cuadrados" => "float" }
```

Figura 5.8: Función que convierte los tipos de los campos indicados

```
remove_field => ["baños"]
```

Figura 5.9: Función en la que se elimina el campo *baño*

La última transformación es una discretización en función del campo *precio* de cada vivienda (ver ilustración 5.10). En ella se indica la creación de un nuevo campo para cada registro llamado *categoría*, en el cuál si la vivienda tiene un precio menor de 100000 será considerada de categoría barata, si el precio oscila entre 100000 y 300000 será considerada de categoría intermedia y si excede la última cantidad será catalogada como cara.

Una vez ejecutado Logstash, a Elastic le llega un índice con toda la información que se ha procesado en el filtro. Y una vez se tienen los datos en el entorno, podemos generar el *dashboard* (ver ilustración 5.11) para mostrar toda la información.

En este tablero se han incluído visualizaciones como un gráfico de barras en el que se muestra información sobre el número de viviendas por categoría de precio clasificada (ver ilustración 5.12), también una métrica y una tabla que muestran el número total de viviendas (ver ilustración 5.24), un gráfico de donut mostando el porcentaje de las viviendas que pertenecen a cada ciudad y barrio (ver ilustración 5.23), un gráfico de árbol que muestra la media de metros cuadrados de la vivienda por ciudad (ver ilustración 5.15) y un gráfico de barras apiladas que muestra el número de viviendas y el número de habitaciones medio por barrio (ver ilustración 5.16).

```

if [precio] {
  if [precio] < 100000 {
    mutate {
      add_field => { "categoria" => "Barata" }
    }
  } else if [precio] >= 100000 and [precio] < 300000 {
    mutate {
      add_field => { "categoria" => "Intermedia" }
    }
  } else {
    mutate {
      add_field => { "categoria_precio" => "Cara" }
    }
  }
}

```

Figura 5.10: Función en la que se discretiza en función del precio de la vivienda

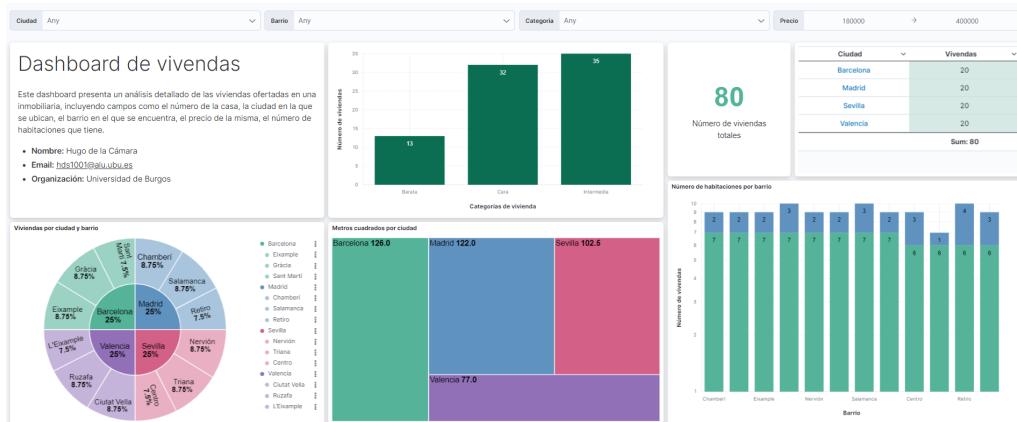


Figura 5.11: Dashboard final del escenario 2

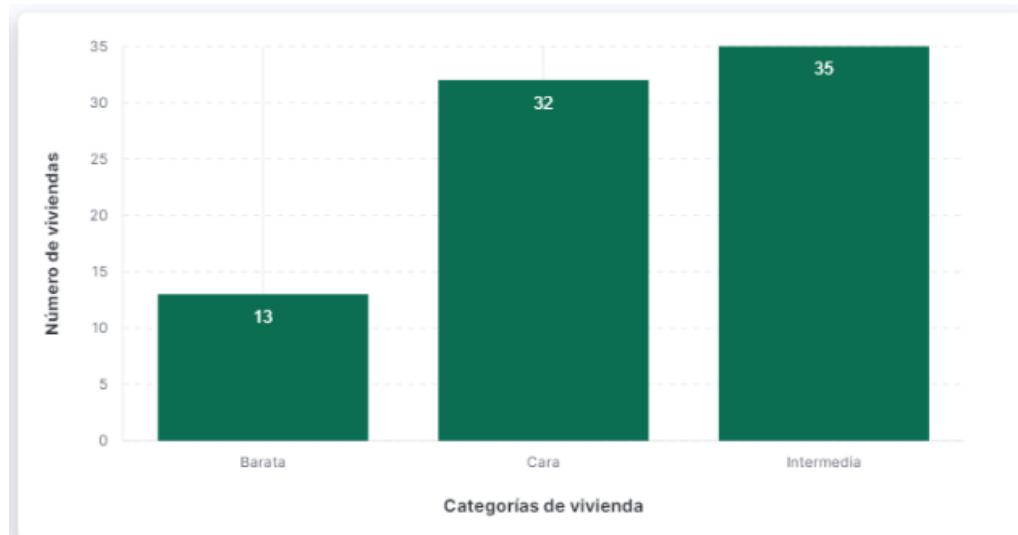


Figura 5.12: Gráfico de barras del número de viviendas por categoría de precio.



Figura 5.13: Métrica y tabla del número de viviendas.

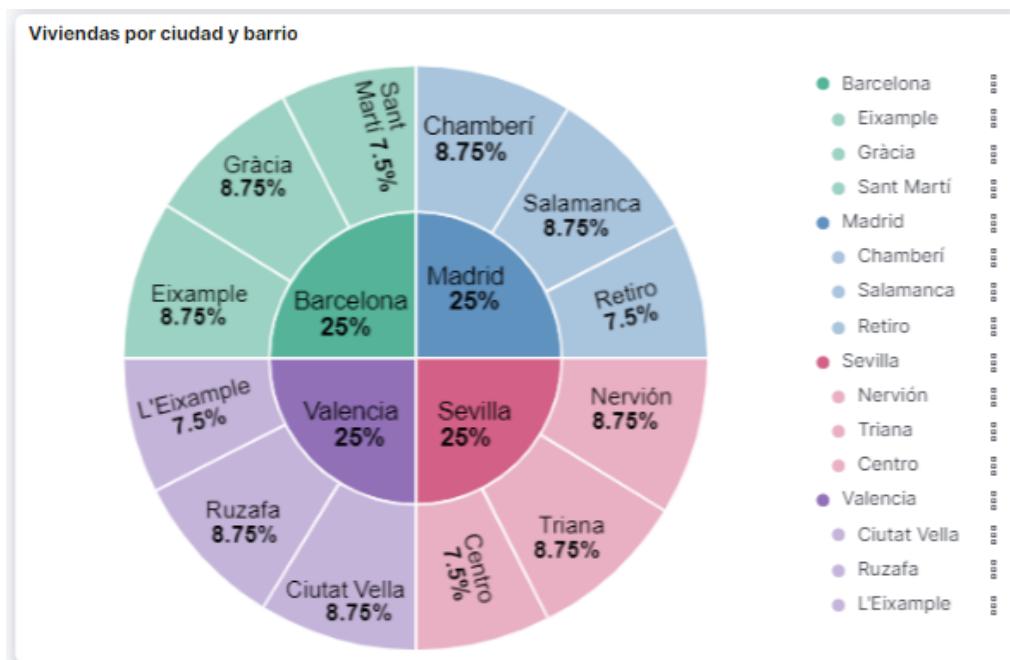


Figura 5.14: Gráfico de donut que muestra el porcentaje de viviendas por ciudad y barrio.

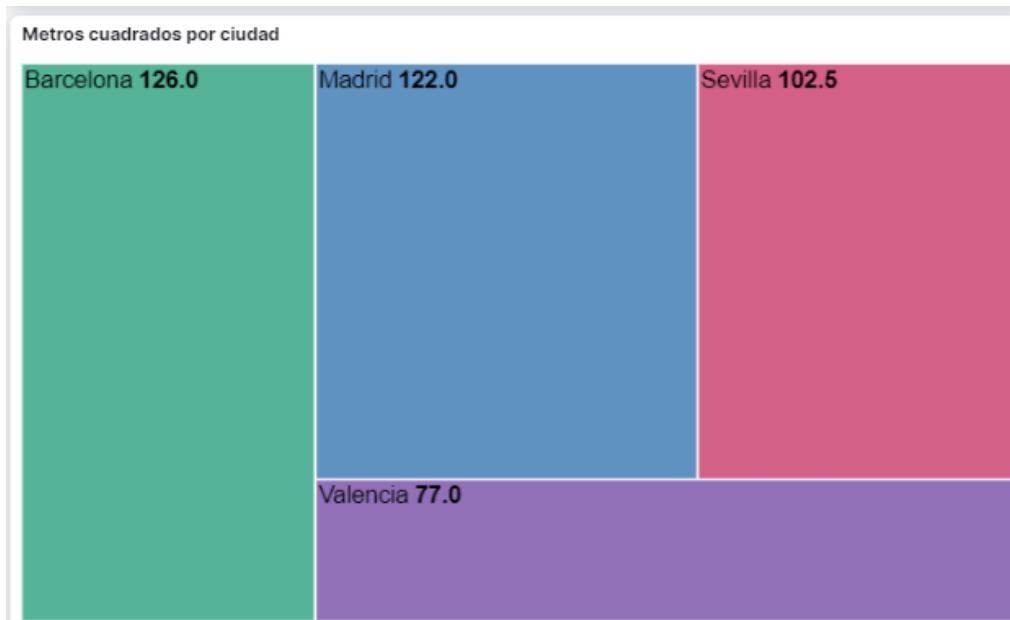


Figura 5.15: Gráfico de árbol que muestra la media de metros cuadrado por ciudad.

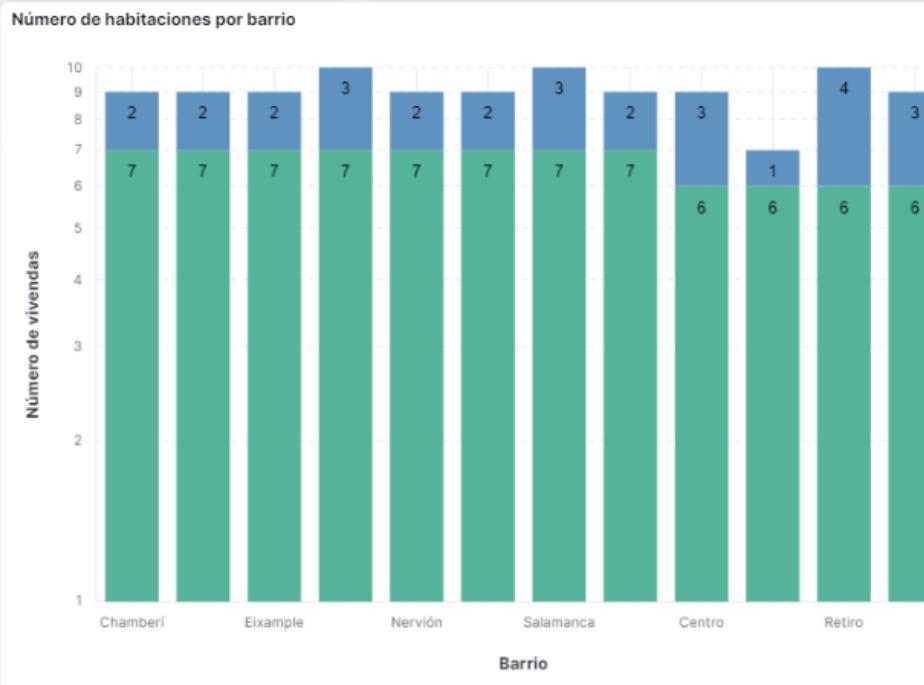


Figura 5.16: Gráfico de barras que muestra el número de habitaciones y de viviendas por barrio.

viviendas		80 documents
2: 1 field sorted		
✓	🕒 @Timestamp	* Document
<input type="checkbox"/>	Jun 29, 2024 0 17:39:36.581	#timestamp Jun 29, 2024 0 17:39:36.581 Version 1 barrio Ciutat Vella categoría Intermedia ciudad Valencia event.original {"numero": 88, "ciudad": "Valencia", "barrio": "Ciutat Vella", "precio": 220000, "habitaciones": 2, "baños": 1, "metros_cuadrados": 100}
<input checked="" type="checkbox"/>	Jun 29, 2024 0 17:39:36.581	#timestamp Jun 29, 2024 0 17:39:36.581 Version 1 barrio Nervión categoría Intermedia ciudad Sevilla event.original {"numero": 79, "ciudad": "Sevilla", "barrio": "Nervión", "precio": 240000, "habitaciones": 2, "baños": 1, "metros_cuadrados": 100}
<input type="checkbox"/>	Jun 29, 2024 0 17:39:36.588	#timestamp Jun 29, 2024 0 17:39:36.588 Version 1 barrio Eixample categoría Cara ciudad Barcelona event.original {"numero": 78, "ciudad": "Barcelona", "barrio": "Eixample", "precio": 350000, "habitaciones": 2, "baños": 1, "metros_cuadrados": 100}
<input checked="" type="checkbox"/>	Jun 29, 2024 0 17:39:36.499	#timestamp Jun 29, 2024 0 17:39:36.499 Version 1 barrio Chamberí categoría Intermedia ciudad Madrid event.original {"numero": 77, "ciudad": "Madrid", "barrio": "Chamberí", "precio": 280000, "habitaciones": 2, "baños": 1, "metros_cuadrados": 100}
<input type="checkbox"/>	Jun 29, 2024 0 17:39:36.498	#timestamp Jun 29, 2024 0 17:39:36.498 Version 1 barrio Ruzafa categoría Barata ciudad Valencia event.original {"numero": 76, "ciudad": "Valencia", "barrio": "Ruzafa", "precio": 180000, "habitaciones": 2, "baños": 1, "metros_cuadrados": 95}
<input type="checkbox"/>	Jun 29, 2024 0 17:39:36.497	#timestamp Jun 29, 2024 0 17:39:36.497 Version 1 barrio Triana categoría Intermedia ciudad Sevilla event.original {"numero": 75, "ciudad": "Sevilla", "barrio": "Triana", "precio": 260000, "habitaciones": 2, "baños": 1, "metros_cuadrados": 90}
<input checked="" type="checkbox"/>	Jun 29, 2024 0 17:39:36.496	#timestamp Jun 29, 2024 0 17:39:36.496 Version 1 barrio Gracia categoría Intermedia ciudad Barcelona event.original {"numero": 74, "ciudad": "Barcelona", "barrio": "Gracia", "precio": 290000, "habitaciones": 3, "baños": 2, "metros_cuadrados": 110}
<input checked="" type="checkbox"/>	Jun 29, 2024 0 17:39:36.495	#timestamp Jun 29, 2024 0 17:39:36.495 Version 1 barrio L'Eixample categoría Barata ciudad Valencia event.original {"numero": 73, "ciudad": "Valencia", "barrio": "L'Eixample", "precio": 180000, "habitaciones": 1, "baños": 1, "metros_cuadrados": 90}

Figura 5.17: Datos del índice del escenario 2

Escenario 3: ingestá a través de WebSocket a Elastic

El tercer caso se tratará de suscribirse a un data stream a través de la tecnología WebSocket que nos irá mandando datos en tiempo real a Elastic y se irán mostrando en Kibana.

En un principio el tema de los envíos de datos en vivo se planteó como una de las bases del trabajo, puesto que es la situación que más se puede asemejar a un entorno de trabajo con IoT. El problema estaba en que no se disponía de una suscripción a un servicio de ingestá de datos en vivo, por lo que se tuvo que investigar de qué manera se podía implementar esta funcionalidad en el entorno de trabajo. Llegando así al siguiente [repositorio GitHub](#), el cuál incluía distintos datasets públicos con actualizaciones en vivo.

Tras probar distintos conjuntos de este repositorio, se llegó a la conclusión de que la opción más interesante era la de [Finnhub](#), ya que proporciona una *Real-Time RESTful API* y *WebSocket* para interactuar con *stocks* y criptomonedas del mercado.

La [página oficial](#) es sencilla de entender, nos proporciona tanto documentación sobre el funcionamiento del *WebSocket*, como una *API key* gratuita para suscribirnos al servicio. Una vez que la tenemos, en la documentación se nos proporciona la estructura básica del script del *WebSocket* escrito en Python. Se debe introducir la *key* obtenida y correr el programa.

Al ejecutarlo, se mostrará por pantalla las actualizaciones de los distintos precios y características de los símbolos de mercado a los que se haya suscrito en el script.

En este primer escenario trabajando con *data streams*, la ingestá de datos se hace de la manera más sencilla y estando lo más cerca posible de la fuente de los datos. La información se manda directamente del *WebSocket* a Elastic a través del puerto 9200, llegando al índice "websocket-data", donde los datos en vivo son almacenados, de manera que se pueda usarlos en Kibana para exponerlos y analizarlos en un *dashboard*.

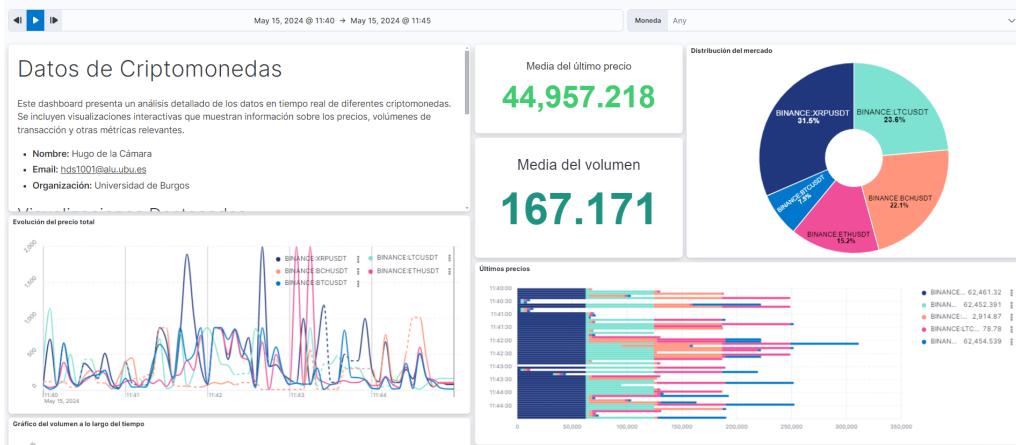


Figura 5.18: Dashboard final del tercer escenario

Escenario 4: ingestá a través de WebSocket con Logstash

En este cuarto escenario se pretendió rizar el rizo del anterior, y estudiar la posibilidad de modificar esos datos mandados por el WebSocket a través de Logstash antes de mandárselos a Elastic.

Teniendo Logstash presente en el sistema ELK, resulta interesante ver las posibilidades que ofrece a la hora de filtrar y modificar datos en vivo con el apoyo de la documentación presente sobre el tema tanto en la web oficial de Elastic como en foros como StackOverflow.

Para ello, se utilizará el *data stream* del script del tercer escenario, sobre el que se realizarán una serie de modificaciones. La primera va a ser en el mensaje que se manda a Logstash, puesto que las variables sobre cada *stock*, eran mandadas con una letra identificatoria, las cuales sin tener conocimientos sobre el tema, no son comprensibles. Así se modificó la forma en la que se mandaban, cambiando la letra por el nombre completo de la variable en cuestión.

Tras esto, la siguiente modificación era cambiar el destino de los datos, el cuál ahora iba a ser Logstash. Se utilizó HTTP para realizar este envío en lugar de al puerto 9200 (Elastic) al 8080, en el cuál está Logstash a la escucha de eventos. Con estas modificaciones, al ejecutar el código, los datos son mandados a Logstash para continuar el tratamiento allí.

En Logstash, se aplicaron cambios al mensaje final, eliminando campos vacíos para ahorrar volumen de información, modificando el tipo de las variables para poder realizar posibles operaciones con ellas de manera más sencilla con datos de tipo *float* e *integer*. También se añadieron nuevos campos calculados a partir de los originales calculando el precio total en función del volumen y del último precio de la divisa. Una vez todas las operaciones se han hecho, la información es mandada al puerto 9200 al índice "test-data-stream", para que los datos puedan ser mostrados en un *dashboard* de Kibana.



Figura 5.19: Dashboard final del cuarto escenario

MapReduce para BigData

En los tiempos que corren, el volumen de transacciones de datos ha aumentado notoriamente, y sigue aumentando a día de hoy. Por lo que los costos de almacenamiento de datos se ha encarecido notablemente. Para reducir estos costes tanto de procesamiento como de almacenamiento, es apropiado utilizar el paradigma de programación MapReduce, que ya se ha mencionado previamente en los conceptos teóricos.

Así que una vez se llegó al quinto sprint del proyecto, hubo un enfoque en continuar desarrollando la implementación de MapReduce en una fuente de ingesta de datos sencilla. Por lo que se optó el realizarselo a partir de un fichero de tipo *.log* que contenía distintas transacciones realizadas en una tienda.

Escenario 5: ingesta de data stream con MapReduce en el servidor mediante Logstash

En este caso lo que es generar un streaming de datos a partir de un script en Python, cargar esos datos en Logstash, realizarle una serie de modificaciones entre las que se encuentra aplicar el plugin *aggregate*, el cuál cumple el papel de aplicar MapReduce, y cargar esos datos tratados en Elastic para poder visualizarlos en Kibana.

En el data stream sobre el que se va a trabajar, se indican las diferentes transacciones que se han cometido en un comercio con datos sobre las mismas, como el instante de tiempo, el método de pago, la sección en la que se ha comprado, entre otros muchos otros.

Para trabajar con Logstash, se necesita un archivo de configuración para que éste entienda lo que le pedimos que haga. Este archivo es de tipo *.conf* e incluye 3 apartados:

- **Input:** ruta hacia el archivo que contendrá las transacciones
- **Filter:** operaciones a realizarle a estos datos, es decir, el MapReduce con la función *aggregate*
- **Output:** salida por pantalla de los datos mandados a Elastic una vez son tratados.

```

filter {
    aggregate {
        task_id => "%{payment_method}"
        code =>
        map['payment_method'] = event.get('payment_method')
        map['total_amount'] ||= 0
        map['total_amount'] += event.get('amount')
        "
        push_map_as_event_on_timeout => true
        timeout_task_id_field => "payment_method"
        timeout => 20
        inactivity_timeout => 40
    }
}

```

Figura 5.20: Plugin *aggregate* para aplicar MapReduce

Logstash nos ofrece la función *aggregate* (ver ilustración 5.20), que permite fusionar muchas líneas de información en una con un determinado criterio en el que se le especifica sobre que campo de los datos va a centrarse. Durante un periodo de tiempo, indicado en la variable *timeout*, la función irá agrupando los registros que concidan con el criterio para que una vez finalizado sean mandados al lugar destino. Se incluirá en el apartado *filter*, indicándole el campo sobre el que hará las agrupaciones en la variable *task id*, así como los datos que se quieren colecciónar una vez son agrupados en la variable *code*. También hay que indicarle un tiempo de inactividad para parar el servicio en la variable *inactivity timeout*.

En la figura se indica que las agrupaciones se van a realizar en función del método de pago utilizado en cada transacción, y que cada 20 segundos se actualizan estas agrupaciones, y si trás 40 segundos no ha ocurrido ningún cambio, que se detenga el servicio.

En el apartado *output* se indica que los datos procesados seán mandados a ElasticSearch a través del puerto 9200, y que allí se cree un índice con los datos que le lleguen. También se indica que los datos procesados sean mostrados por pantalla para comprobar que todo funciona.

Una vez ejecutado Logstash, a Elastic le llega un índice con tan solo tres filas, las cuáles son las equivalentes a los tres tipos de métodos de pago. Teniendo los datos en el entorno, podemos generar el *dashboard* para

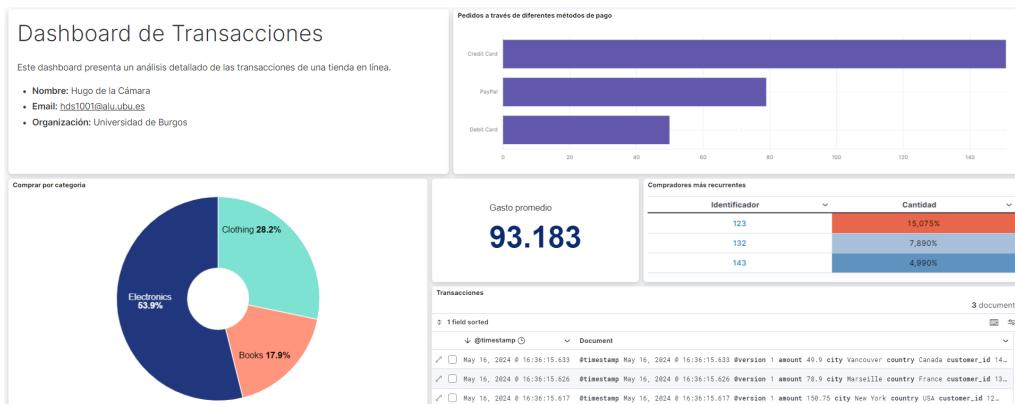


Figura 5.21: Dashboard final del quinto escenario

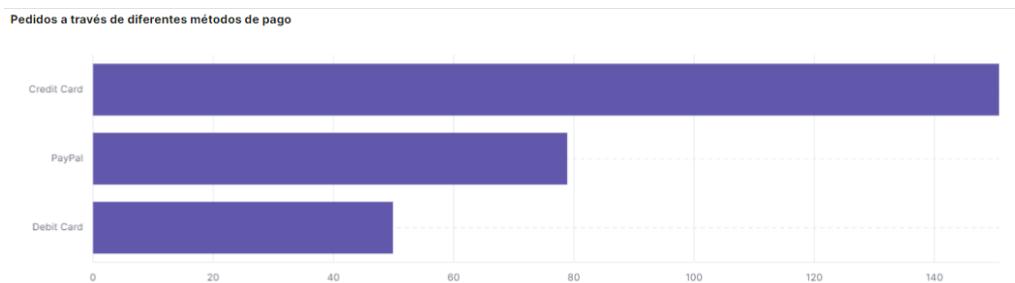


Figura 5.22: Gráfico de barras del número de pedidos con cada método de pago

mostrarlos y trabajar sobre estas agrupaciones, mostrando el número de compras con cada método de pago en un gráfico de barras horizontal (ver ilustración 5.22), las categorías en las que más se ha gastado (ver ilustración 5.23), los clientes que más han comprado (ver ilustración 5.25) o el gasto promedio (ver ilustración 5.24).

Una vez la carga de los datos en Logstash fue completada con éxito, y se pudo empezar a experimentar con la función que este nos ofrece equivalente a un MapReduce, la función *aggregate*, en la cuál se hizo un agrupado de los datos tomando como referencia el método de pago usado, resultando en que el envío a Elastic fueron 3 líneas con los datos de las cantidades gastadas con cada método y las categorías más compradas entre otros.

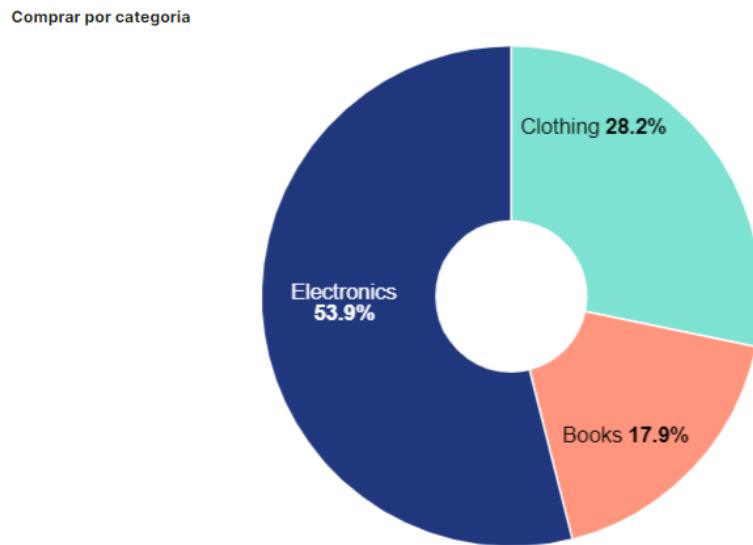


Figura 5.23: Gráfico de donut del porcentaje de compra en cada categoría



Figura 5.24: Métrica del gasto promedio de cada cliente

Compradores más recurrentes	
Identificador	Cantidad
123	15,075%
132	7,890%
143	4,990%

Figura 5.25: Tabla mostrando los clientes más recurrentes

En el caso de este estudio, se aplicó MapReduce a una fuente de datos modesta, puesto que las capacidades de los sistemas en los que se está ejecutando el proyecto son limitadas, pero eso no quita que esta función sea extrapolable a grandes volúmenes de datos con sistemas que sean capaces de procesarlos y reducirlos de cara a moderar el coste de almacenamiento de los mismos. Estos sistemas deben ser clústeres de ordenadores que soporten almacenamiento distribuido , y mediante programas como Hadoop [15], esta función se pueda aplicar más fácilmente a grandes masas de datos.

En sistemas de IoT que poseen un gran número de sensores que están mandado datos constantemente es donde resulta interesante aplicar estos conceptos, puesto que se tendrá una estructura ágil para poder analizar la información en tiempo real reduciendo costes.

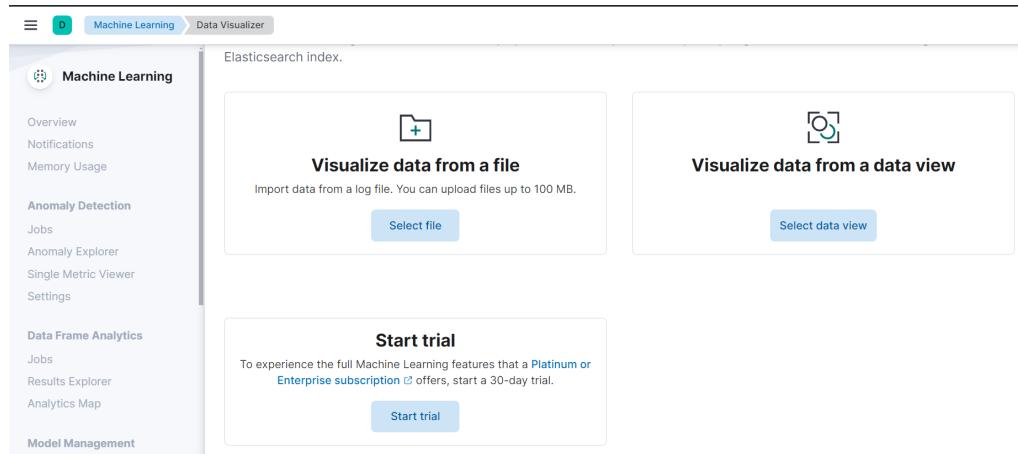


Figura 5.26: Funcionalidad de Machine Learning de Elastic.

Machine Learning en ELK

Desde el principio del proyecto se planteó la idea de trabajar con Machine Learning de manera automática con una fuente de ingestión de datos. Al investigar más profundamente, se llegó a la conclusión de que está función si que la ofrece el ecosistema Elastic (ver ilustración 5.26), pero pagándola, ya que como se puede observar en la figura las funciones están bloqueadas y solo permite visualizar datos, función que cubre el *Discover*. Hubo que reconducir este escenario de manera que se pudieran aplicar estas técnicas de análisis de manera gratuita.

Y así es como se comenzó a utilizar la biblioteca de análisis predictivo *Scikit-Learn*, la cuál está construida sobre las bibliotecas *Python* de *NumPy*, *SciPy* y *matplotlib*. Para integrar esta herramienta en nuestro ecosistema de manera que pudiera interactuar con Elastic, se hizo uso de la herramienta *Jupyter Notebook*.

Con *Jupyter*, se generó un script escrito en lenguaje *Python* que importaría las librerías necesarias para poder ejecutar los algoritmos de *Scikit-Learn* sobre un conjunto de datos que se le indicara. Como fuente de datos se utilizó un conjunto clásico de las ciencias estadísticas, como lo es el *dataset Iris*, el cuál contiene información sobre tres especies de flores iris, donde cada muestra tiene cuatro características (longitud y anchura del sépalo y del pétalo).

Lo primero que se hizo fue un script que cargara los datos de Iris en un índice de Elastic, para que un segundo script lea estos desde ese índice, y una vez se tienen los datos cargados en el script desde Elastic, se procede a ejecutar el algoritmo de clasificación *KNN* (*K vecinos más cercanos*)[29], el cuál es un método de clasificación supervisada con el que podemos calcular con qué probabilidad un elemento x desconocido puede pertenecer a una clase a partir de la información que le proporcionemos como prototipo. Una vez se ha obtenido el rendimiento del clasificador, se guardan los resultados localmente en un archivo CSV, y se mandan al puerto 9200 hacia el índice *iris clasificacion*, que será el que almacene los datos en Elastic para que puedan ser mostrados en un *dashboard* de Kibana.

Este proceso de envío de datos va a ser común en los siguientes tres algoritmos que ejecutaremos, siendo el siguiente el de regresión lineal[3], en el cuál se estudia la relación entre la longitud y la anchura de los pétalos de manera que se pueda predecir un comportamiento típico.

Tras este, vamos a trabajar con algoritmos más visuales, como es el de *clustering* o análisis de grupos con K-Means[17] , con el cuál agruparemos los ejemplares de la flor Iris que tengan características similares en diferentes grupos ilustrados con colores.

Y así concluimos el script con el último algoritmo estudiado, tratándose del de una reducción de dimensionalidad como es *PCA*[24], con el cuál describiremos el conjunto de datos Iris con nuevas variables no correlacionadas, resultando en un gráfico de dispersión bidimensional.

Una vez tenemos toda la información procesada y cargada en Elastic, la creación de un *dashboard* de Kibana viene de la mano, indicándole que los datos se sacarán del índice *Iris* el cuál contiene toda la información que le hemos ido enviando desde el script.

En este *dashboard* se muestran las métricas calculadas por los dos primeros algoritmos (ver ilustración 5.27), y las gráficas calculadas por los dos últimos (ver ilustraciones 5.28 y 5.29, además de información sobre el dataset Iris y los diferentes datos que han cargado los algoritmos como resultado del procesamiento.

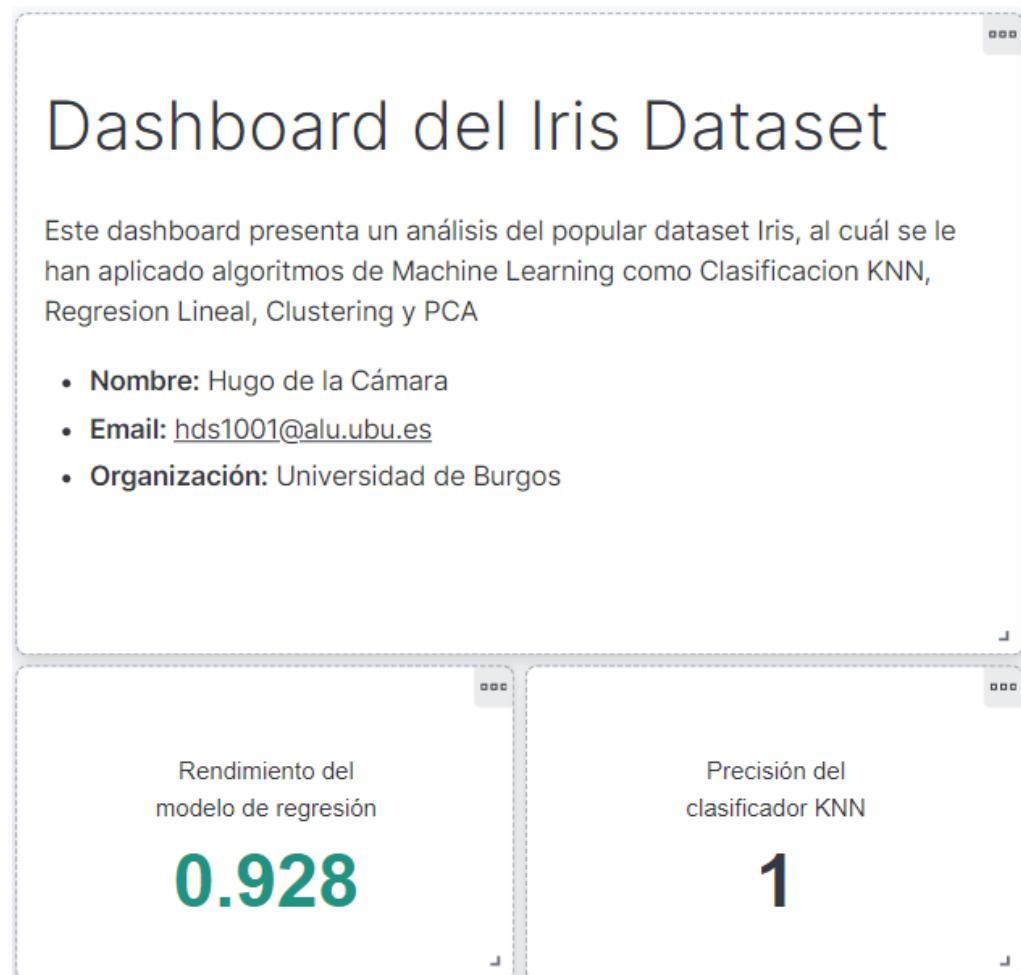


Figura 5.27: Visualizaciones mostrando información del *dashboard* así como el rendimiento de la regresión y la precisión del clasificador.

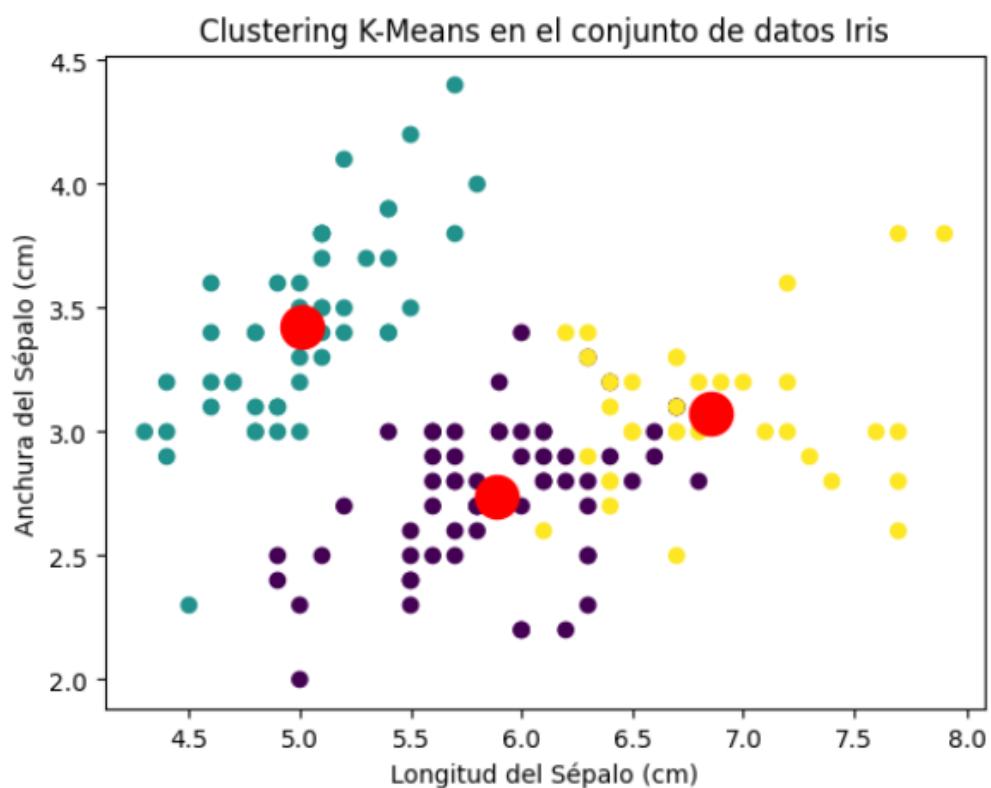


Figura 5.28: Visualización de dos dimensiones del gráfico obtenido tras aplicar clustering.

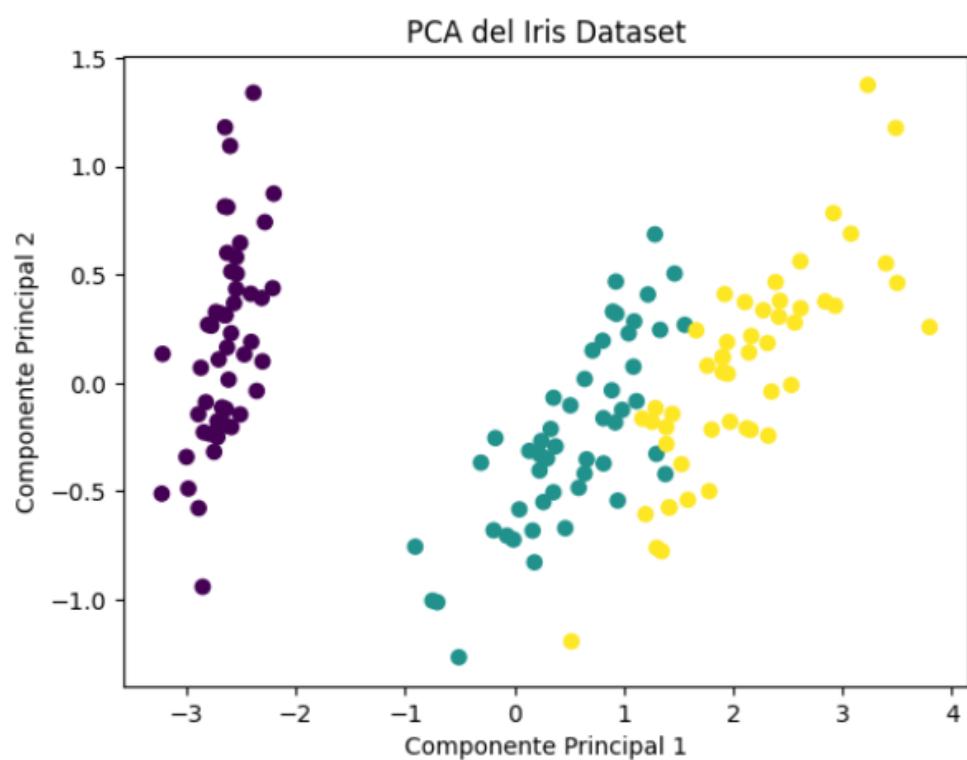


Figura 5.29: Visualización del gráfico obtenido tras aplicar PCA.

6. Trabajos relacionados

En este apartado se van a presentar diferentes alternativas de herramientas ETL que cubran de manera parcial o total los requisitos de las tareas completadas con el sistema ELK.

6.1. PowerBI

Esta herramienta de análisis de datos desarrollada por *Microsoft* está más enfocada al mundo de los negocios[20]. Y al igual que el *stack ELK* proporciona distintos complementos para realizar visualizaciones de datos y poder crear informes interactivos de los mismos.

A diferencia de el *stack ELK* usado en este proyecto, **PowerBI** (ver ilustración 6.1) está basado en la nube, alojando ahí el almacenamiento de los datos, la preparación de estos y las visualizaciones personalizadas.

Por su facilidad de uso, las distintas integraciones que ofrece con programas como *Excel* o *SQL Server*, y sus numerosas funciones avanzadas, **PowerBI** se destaca como el programa líder del sector y más utilizado a nivel global dentro de la ciencia de los datos.

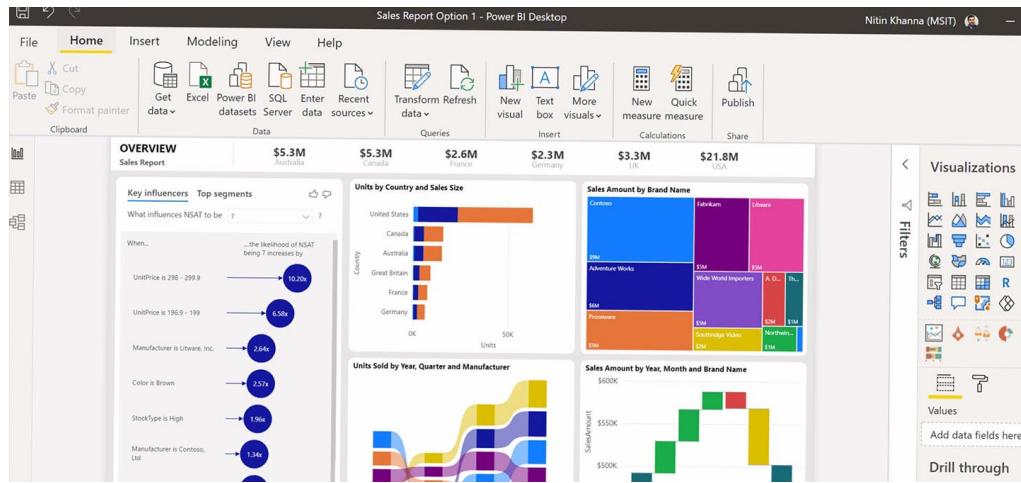


Figura 6.1: Visualizaciones de PowerBI [22]

6.2. Spoon y Qlik

Tuve la oportunidad de realizar las prácticas en la empresa informática CSA (Centro de Servicios Avanzados), concretamente en el departamento de Business Intelligence (BI) y Automatización y Robótica de Procesos (RPA).

Aquí estuve trabajando con su sistema de herramientas ETL el cuál tenía una estructura diferente pero la finalidad era la misma que la del *Stack ELK*, extraer, transformar y cargar información.

Lo primero era extraer la información de los diferentes servidores y bases de datos de la empresa, y el primer paso para administrar estas bases se hacia a través el programa **DBeaver**. Una vez teníamos claros los archivos a tratar con la herramienta **WinSCP** gestionábamos los datos en cuestión.

La herramienta equivalente a Logstash era **Spoon** (ver ilustración 6.2), una interfaz para diseñar soluciones de transformación de datos que nos permite crear procesos de carga y lectura de manera intuitiva y visual.

Y por último, la herramienta estrella, **Qlik** (ver ilustración 6.3), que nos permite pintar gráficos e interfaces con todos los datos procesados por las herramientas anteriores.

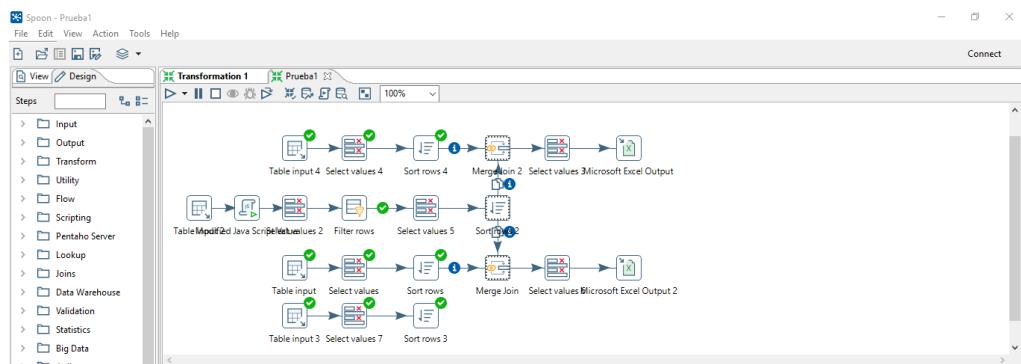


Figura 6.2: Estructura de Spoon [18]

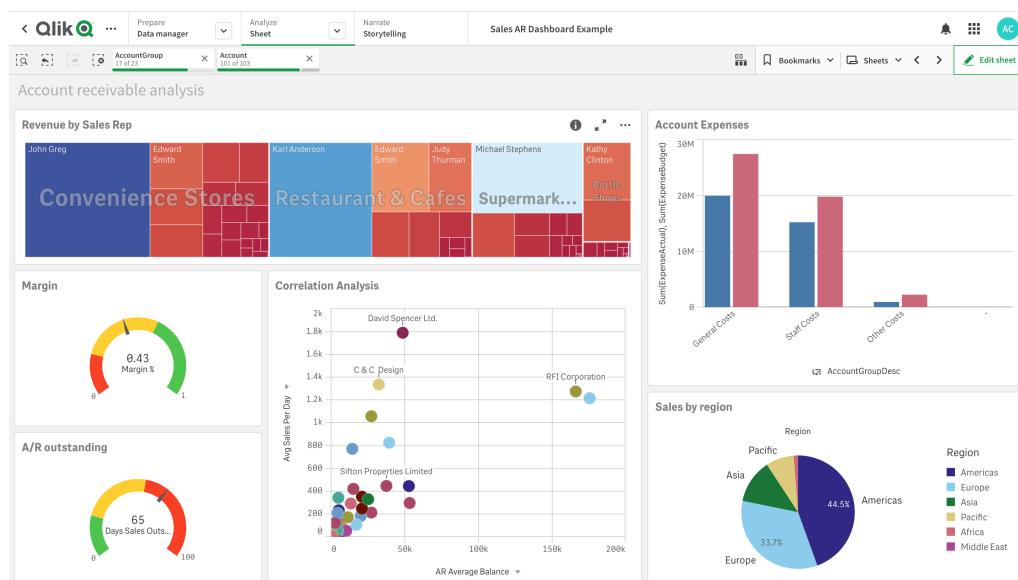


Figura 6.3: Visualizaciones con Qlik [27]

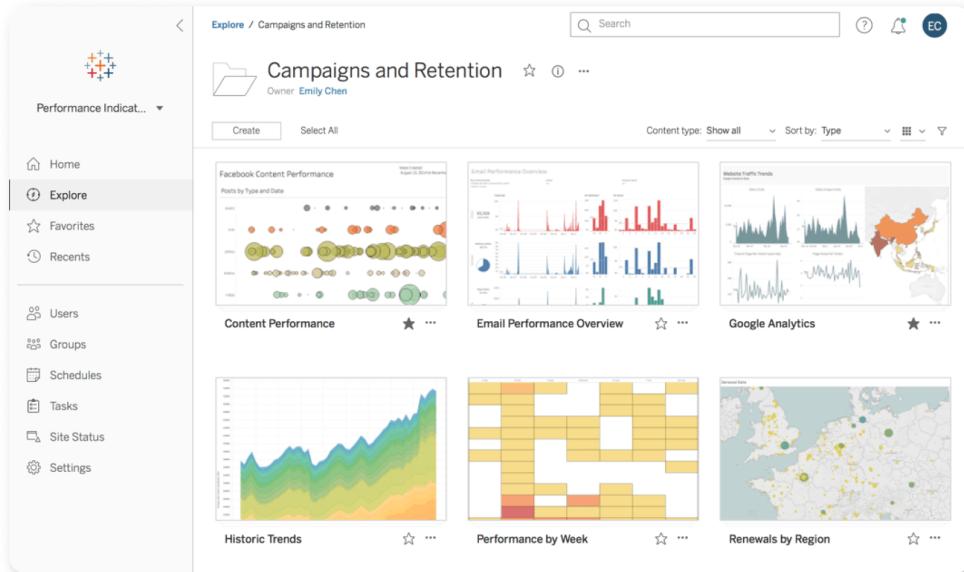


Figura 6.4: Dashboard principal de Tableau [28]

6.3. Tableau

Este software está pensado para el desarrollo de visualizaciones interactivas de datos en forma de gráficos en *dashboards* (ver ilustración 6.4). Fue diseñado para el análisis y comprensión de datos complejos ya que soporta la ingestión de datos tanto desde hojas de cálculo como de bases de datos. Posee una interfaz intuitiva basada en *drag and drop* de manera que cualquier usuario pueda comprender y tomar decisiones basadas en información de datos.

6.4. Splunk

Consiste en una plataforma diseñada para analizar y monitorear datos ingestados en tiempo real (ver ilustración 6.5). Este software recopila e indexa grandes conjuntos de datos de manera que se puedan filtrar y analizar de manera profunda y avanzada a la hora de identificar anomalías, optimizar procesos o identificar patrones.



Figura 6.5: Dashboard básico de Splunk [2]

6.5. Comparativa entre alternativas

A continuación se realizará una tabla comparativa entre las diferentes opciones expuestas en este apartado como alternativas al stack ELK, así como más opciones presentes en el mercado. Se incluyen algunas características o propiedades que resultan interesantes (ver tabla 6.1).

Cabe destacar que las funciones de inteligencia artificial en el caso del stack ELK se consideran limitadas, ya que ofrece detección de anomalías y un plugin para uso de machine learning. Ambas funciones solo están disponibles en el plan de suscripción Platinum, por lo que se consideran escasas en comparación con PowerBI, el cuál integra machine learning para realizar modelos predictivos o analizar series temporales. Qlik, Tableau y Splunk ofrecen características muy similares de este tipo sobre todo enfocadas al análisis y procesamiento de datos en tiempo real.

Como conclusión indicar que el stack ELK está pensado para monitorizar y analizar datos, principalmente de tipo log, en tiempo real, no como PowerBi que está más enfocado al ámbito empresarial a la hora de generar informes (ideal si el ecosistema en el que se trabaja es el de Microsoft). Qlik y Spoon son una buena combinación para negocios basados en ETL, al igual que Tableau y Splunk al ser todas herramientas para usos muy similares.

	Stack ELK	Power BI	Qlik + Spoon	Tableau	Splunk
Función principal	Análisis de logs y datos en tiempo real	Análisis de negocios y generación de informes	Análisis de negocios y ETL	Visualización de datos de negocios	Análisis de datos en tiempo real
Facilidad de uso	Requiere conocimientos técnicos avanzados	Interfaz intuitiva y fácil de usar	Qlik es más intuitivo que Spoon	Interfaz intuitiva y fácil de usar	Requiere conocimientos técnicos avanzados
Escalabilidad	Alta	Limitada	Alta	Alta	Alta
Coste	Gratis, opciones avanzadas de pago	Suscripción de pago	Qlik licencia de pago, Spoon es gratis	Suscripción de pago	Suscripción de pago
Almacenamiento en la Nube	proporciona el servicio de pago Elastic-search Service	Sí, integrado con Azure	Qlik sí que lo ofrece y Spoon puede integrarse	Sí, integrado con servicios en la nube	Sí, integrado con servicios en la nube
Análisis en Tiempo Real	Sí, nativo	Depende de la fuente de datos	Sí, Qlik permite análisis en tiempo real	Depende de la fuente de datos	Sí, nativo
Inteligencia Artificial	Funcionalidades limitadas	Ofrece capacidades de IA y análisis predictivo	Qlik tiene capacidades de IA, Spoon no	Ofrece IA y análisis predictivo	Ofrece IA y análisis predictivo
Sistemas Operativos	Windows, Linux y Mac	Windows y Mac	Windows, Linux y Mac	Windows, Linux y Mac	Windows, Linux y Mac

Tabla 6.1: Comparación entre herramientas ETL

7. Conclusiones y Líneas de trabajo futuras

El último apartado de este documento memoria va a consistir en la redacción de las conclusiones y valoraciones finales del proyecto, así como diferentes mejoras que se le podrían realizar de cara a un futuro.

7.1. Conclusiones

A lo largo de este estudio se han planteado objetivos que se han podido conseguir y que se van a remarcar en este apartado de que manera, y otros que no han sido posibles y que se dejan como posibles mejoras en el siguiente apartado de cara a trabajos futuros.

Se ha logrado crear cinco escenarios que muestran diferentes maneras ingestar datos con el stack ELK, ya sea a través de un fichero estático o un data stream, aplicándole modificaciones de cara al producto final mostrado.

No se ha podido realizar escenarios con otras fuentes de datos puesto que suponía aumentar la carga de trabajo. Se intentó crear un escenario en el que se mandarán datos desde *Filebeat*, pero no se pudo lograr por la dificultad de comprensión del mismo. Por lo que como sustituto se recurrió a los WebSockets para que cubrieran la función de ingesta de data streams en el tercer y cuarto escenario.

No obstante, desde el punto de vista funcional, los resultados se consideran extrapolables a conjuntos de datos de gran tamaño. Sin embargo, ha quedado por explorar cómo optimizar la configuración de un sistema hardware de gran potencia, típicamente un cluster, para optimizar el rendimiento del *stack ELK* ante situaciones de ingesta masiva. Este punto se ha quedado fuera del alcance del TFG por falta de recursos hardware.

En general la experiencia ha resultado enriquecedora e interesante por el hecho de poder aplicar conocimientos que se han adquirido estos últimos años y que han resultado útiles en el desarrollo del proyecto. La documentación presente sobre las herramientas era escasa en ciertos ámbitos por lo que este aspecto a dificultado parte del estudio.

7.2. Mejoras futuras

De cara a continuar con este trabajo se considera que se le podrían aplicar ciertos matices para que los escenarios puedan ser más completos y útiles.

- Realizar pruebas de tiempos de respuesta para calcular la eficiencia del *stack ELK* en comparación a otros programas como PowerBI.
- Incluir más escenarios utilizando como origen de los datos *Filebeat* o algún otro API diferente.
- Mejorar la calidad de los datos, incluyendo que su limpieza y normalización se realice mas eficientemente.
- Explorar nuevos escenarios que combinen datos de múltiples orígenes.
- Probar la arquitectura ELK sobre un sistema de Big Data con Hadoop.
- Intentar programar y publicar un plugin para Elasticsearch orientado a incorporar técnicas de Machine Learning.

Bibliografía

- [1] Iker Castaños. ¿En que consiste la monitorizacion de sistemas? <https://www.semantic-systems.com/semantic-noticias/articulos-tecnologicos/en-que-consiste-la-monitorizacion-de-sistemas/>. [Consultado el 23 de mayo de 2024].
- [2] Cisco. Splunk enterprise. https://www.splunk.com/en_us/products/splunk-enterprise.html. [Consultado el 4 de junio de 2024].
- [3] Curvefit. Introduction to linear regression. https://web.archive.org/web/20080222195200/http://www.curvefit.com/linear_regression.htm. [Consultado el 31 de mayo de 2024].
- [4] Elastic. Centraliza, transforma y almacena tus datos. <https://www.elastic.co/es/logstash>. [Consultado el 22 de mayo de 2024].
- [5] Elastic. Descubre, itera y resuelve con es|ql en kibana. <https://www.elastic.co/es/kibana>. [Consultado el 22 de mayo de 2024].
- [6] Elastic. El corazón del elastic stack, gratuito y abierto. <https://www.elastic.co/es/elasticsearch>. [Consultado el 22 de mayo de 2024].
- [7] Alexandre Gramfort Vincent Michel Bertrand Thirion Olivier Grisel Mathieu Blondel Peter Prettenhofer Ron Weiss Vincent Dubourg Jake Vanderplas Alexandre Passos David Cournapeau Matthieu Brucher Matthieu Perrot Édouard Duchesnay Fabian Pedregosa, Gaël Varoquaux. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

- [8] FundEU. Macrodatos e inteligencia de datos, alternativas a big data. <https://www.fundeu.es/recomendacion/macrodatosalternativa-abig-data-1582/>. [Consultado el 22 de mayo de 2024].
- [9] Alexander Gillis. What is the internet of things (iot)? <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>. [Consultado el 22 de mayo de 2024].
- [10] Marilín Gonzalo. Los datos masivos (o big data) son el nuevo oro. https://www.eldiario.es/tecnologia/diario-turing/big-data_1_5767121.html. [Consultado el 22 de mayo de 2024].
- [11] Jim Goodnight. Etl what it is and why it matters. https://www.sas.com/en_us/insights/data-management/what-is-etl.html, 2018. [Consultado el 22 de mayo de 2024].
- [12] Jerry Zhao Grzegorz Czajkowski, Marián Dvorský and Michael Conley. Sorting petabytes with mapreduce - the next episode. <https://research.google/blog/sorting-petabytes-with-mapreduce-the-next-episode/>. [Consultado el 22 de mayo de 2024].
- [13] Ejona Hoxa. La importancia de un proceso etl en el business intelligence. <https://algoritmia8.com/2022/11/23/la-importancia-de-un-proceso-etl-en-el-business-intelligence/>, Abril 2024. [Consultado el 22 de mayo de 2024].
- [14] Soluciones IM. La monitorización de sistemas informáticos en 8 pasos. <https://www.soluciones-im.com/es/la-monitorizacion-de-sistemas-informaticos-en-8-pasos>. [Consultado el 24 de mayo de 2024].
- [15] ASF Infrabot. Projectdescription. <https://cwiki.apache.org/confluence/display/HADOOP2/ProjectDescription>. [Consultado el 28 de mayo de 2024].
- [16] IONOS. ¿qué es github? - control de versiones de un vistazo. <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/github/>. [Consultado el 2 de junio de 2024].
- [17] Anil K. Jain. Data clustering: 50 years beyond k-means. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.151.4286&rep=rep1&type=pdf>. [Consultado el 31 de mayo de 2024].

- [18] Paula Martín. ¿qué es pentaho data integration (pdi) y para qué sirve? <https://itop.academy/blog/item/que-es-pentaho-data-integration-pdi-y-para-que-sirve.html>. [Consultado el 28 de mayo de 2024].
- [19] Microsoft. Getting started with visual studio code. <https://code.visualstudio.com/docs>. [Consultado el 2 de junio de 2024].
- [20] Microsoft. Power bi | interactive data visualization bi tools. <https://www.microsoft.com/en-us/power-platform/products/power-bi/>. [Consultado el 27 de mayo de 2024].
- [21] Microsoft. The websocket api. [https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/dev-guides/hh673567\(v=vs.85\)?redirectedfrom=MSDN](https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/dev-guides/hh673567(v=vs.85)?redirectedfrom=MSDN). [Consultado el 22 de mayo de 2024].
- [22] Profesional Online. ¿qué es power bi de microsoft? <https://www.profesionalonline.com/blog/big-data/que-es-power-bi-de-microsoft/>. [Consultado el 28 de mayo de 2024].
- [23] Overleaf. Write, collaborate, and create with overleaf. <https://www.overleaf.com/about/features-overview>. [Consultado el 2 de junio de 2024].
- [24] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.
- [25] Fernando Perez. Project jupyter. <https://speakerdeck.com/fperez/project-jupyter>. [Consultado el 2 de junio de 2024].
- [26] Python. Python. <https://es.wikipedia.org/wiki/Python#>. [Consultado el 2 de junio de 2024].
- [27] Qlik. Trabajar con apps. https://help.qlik.com/es-ES/cloud-services/Subsystems/Hub/Content/Sense_Hub/Visualizations/create-apps-visualizations.html. [Consultado el 28 de mayo de 2024].
- [28] Salesforce. Tableau server: análisis de autoservicio gobernado escalable. <https://www.tableau.com/es-es/products/server>. [Consultado el 4 de junio de 2024].

- [29] B. W. Silverman and M. C. Jones. Fix and hedges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hedges. *International Statistical Review / Revue Internationale de Statistique*, 57(3):233–238, 1989.
- [30] Carlos Sobrino. La monitorizaciÓn de sistemas y sus ventajas. <https://www.captia.es/blog/monitorizacion-sistemas.html>. [Consultado el 23 de mayo de 2024].
- [31] Wikipedia. Extract, transform and load. https://es.wikipedia.org/wiki/Extract,_transform_and_load, 2018. [Consultado el 22 de mayo de 2024].