Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Cámara de compensación del transporte público de Tenerife: Optimizando algoritmos a través de PySpark

Clearing house of public transport in Tenerife: Optimizing algorithms through PySpark

Laura Cañizares Herrera

La Laguna, 13 de Julio de 2023

D. **José Luis Roda García**, con N.I.F. 43.356.123-L, profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Ginés León Rodríguez**, con N.I.F. 78.408.712-X, responsable del Departamento de Big Data & Data Science de T.I.T.S.A, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

"Cámara de compensación del transporte público de Tenerife: Optimizando algoritmos a través de PySpark"

ha sido realizada bajo nuestra dirección por Dña. **Laura Cañizares Herrera**, con N.I.F. 43.385.014-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de julio de 2023.

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor José Luis Roda García, y cotutor, Ginés León Rodríguez, por concederme su tiempo, conocimiento y ayuda durante el desarrollo de este proyecto.

Gracias a la Universidad de La Laguna, por su esfuerzo, dedicación y trabajo a lo largo de estos años en los que se han tenido que adaptar a las circunstancias para formarnos en condiciones.

Por último, pero no por ello menos importante, a mi familia, en especial a mi padre y a mi madre, que han supuesto en todo momento un apoyo moral fundamental para superar cualquier obstáculo de esta etapa universitaria.

Licencia

Resumen

Una cámara de compensación es un sistema que liquida transacciones financieras entre múltiples partes. En este proyecto, se analizan las compensaciones de los importes de los abonos entre los distintos operadores del transporte público de Tenerife, de manera que, se reparte el valor de cada abono en función del uso que haya realizado el usuario en cada uno de ellos.

El objetivo principal es evaluar el algoritmo desarrollado por Transportes Interurbanos de Tenerife S.A. (TITSA) para el procesado de la cámara de compensación de los importes de los abonos entre los distintos operadores (T.I.T.S.A, Metropolitano y Transportes de la Esperanza), así como optimizar sus tiempos de ejecución y su consumo de recursos. Esto ayudará a liberar de recursos al servidor SQLServer, disminuir la ventana temporal en la que se tiene disposición de los datos y la experiencia del usuario en cuanto a la disposición de los mismos.

Para su implementación se ha utilizado el lenguaje de programación Python con la finalidad de comparar los tiempos de ejecución utilizando la librería de Pandas y PySpark. Además, la plataforma Power BI para convertir los datos tratados en información de calidad con el fin de mejorar la toma de decisiones, diseñar estrategias e implementar cambios en la organización.

Palabras clave: *cámara de compensación, tiempo de ejecución, Pandas, PySpark.*

Abstract

A Clearing House is a system that settles financial transactions between multiple operators. In this project, the compensation of the amounts of the subscriptions between the different public transport in Tenerife is analyzed, so that the value is distributed according to the use that the user has made in each one of them.

The main objetive is to evaluate the algorithm developed by Transportes Interurbanos de Tenerife S.A. (T.I.T.S.A) for the processing of the Clearing House of the amounts of the subscriptions between the differents operators (T.I.T.S.A, Metropolitano and Transportes de la Esperanza), as well as to optimize their execution times and their consumption of resources. This will help to free up SQLServer resources, reduce the window time in which the data is available and improve the user experience in terms of data availability.

For its implementation, the Python programming language has been used in order to compare execution times using the Pandas and PySpark libraries. In addition, the Power BI platform is used to convert the processed data into quality information in order to improve decision making, design strategies and implement changes in the organization.

Keywords: clearing house, execution time, Pandas, PySpark.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 Cámara de compensación

La cámara de compensación es un sistema que liquida transacciones financieras entre múltiples partes, actuando como intermediaria entre los diferentes operadores con el objetivo de garantizar seguridad y eficiencia, verificando la validez de las operaciones para que los fondos se transfieran correctamente.

En este proyecto se analizan las compensaciones de los importes de los abonos entre los distintos operadores del transporte público de Tenerife (T.I.T.S.A, Metropolitano y Transportes de la Esperanza), de manera que, se reparte el valor de cada abono en función del uso que haya realizado el usuario en cada uno de ellos.

1.2 Descripción general del proyecto y objetivos

Actualmente, T.I.T.S.A dispone de un sistema de cálculo para las compensaciones que computacionalmente consume muchos recursos y el tiempo para la obtención del resultado es considerablemente elevado. Por tanto, propone este trabajo de fin de grado, cuyo objetivo principal es agilizar el procesado de los datos que componen la cámara de compensación, con el fin de disminuir el tiempo de espera en el que se disponen de los mismos con un sistema mucho más óptimo y eficiente.

Se han elaborado diferentes algoritmos en distintas librerías de Python, comprobando en todo momento el tiempo de ejecución en cada una de ellas para asegurar que el tiempo de procesado haya mejorado significativamente. Este avance en el código ayudará a liberar de recursos al servidor SQLServer y mejorará la experiencia del usuario en cuanto a la disposición de los datos.

Paralelamente, se ha enlazado el resultado de cada una de las ejecuciones mediante una visualización en el Power BI, obteniendo información de calidad de los mismos.

1.3 Antecedentes

La cámara de compensación de importes tiene su origen en el desarrollo del comercio y las transacciones financieras a nivel internacional. En el pasado, las transacciones comerciales se realizaban de forma individual entre los compradores y los vendedores, lo que generaba un alto grado de complejidad y riesgo, debido a la necesidad de realizar pagos en diferentes monedas y en diferentes países.

Con el tiempo, se fueron desarrollando diversas soluciones y en la actualidad las cámaras de compensación se han convertido en una herramienta fundamental en el sistema financiero debido a la seguridad y eficiencia que aportan, agilizando procesos de pago entre las diferentes empresas que participan.

Son un componente esencial del comercio internacional, permitiendo a las empresas realizar transacciones de manera segura y sin complicaciones en el pago de la compensación, es por ello que T.I.T.S.A insiste en la importancia de la mejora del procesado del alto volumen de datos con los que trabaja, puesto que es un proceso recurrente en la empresa y es fundamental agilizarlo para ocasionar el mínimo coste de tiempo posible, optimizando el código con las herramientas adecuadas.

1.4 Estado del arte

Las cámaras de compensación de importes han evolucionado significativamente en las últimas décadas [12]. Actualmente, se han convertido en herramientas esenciales en el sistema financiero y han contribuido a mejorar la eficiencia y la seguridad de las transacciones.

Entre las principales innovaciones que se han ido produciendo con el tiempo en las cámaras de compensación de importes se encuentran:

1. Tecnología avanzada para mejorar la eficiencia y la rapidez en el procesamiento de las transacciones. Esto ha permitido una mayor automatización y reducción de errores en el procesamiento de pagos.

2. Mayor cobertura geográfica ampliando su alcance para abarcar un mayor número de países y regiones. Esto ha mejorado la capacidad de las cámaras de compensación para procesar transacciones internacionales y ha facilitado el comercio global.

3. Nuevos productos y servicios introducidos para satisfacer las necesidades de los participantes en las transacciones financieras (gestión de riesgos, la compensación de derivados y la gestión de garantías).

4. Mejoras en la seguridad de las transacciones financieras mediante la autenticación de usuarios y la criptografía de datos.

En la actualidad, el análisis, la visualización, el uso de la Inteligencia Artificial y las mejoras en el procesamiento de datos han demostrado ser unas de las tareas más importantes en las empresas que necesitan extraer conocimiento de la ingente cantidad de datos que acumulan y registran [4].

Es fundamental basarse en datos e información de calidad para la toma de decisiones, diseñar estrategias e implementar cambios en una empresa, con el objetivo de no llevarse simplemente por la intuición. Por tanto, trabajar con datos en una organización radica en que permite hacer análisis exhaustivos y certeros que garanticen tomar decisiones más acertadas, basadas en información veraz que permita conocer qué está pasando, prever lo que pueda ocurrir en el futuro y medir el rendimiento de las acciones llevadas a cabo.

Cada vez se presta mayor importancia a reducir el lapso de tiempo que transcurre entre el momento en que se genera el dato y el instante en el que se tiene disposición de él, es por ello que las mejoras del uso de Plataformas de Datos está cogiendo mayor relevancia y se quiere optimizar el tiempo del procesado de los datos de la cámara de compensación que utiliza con frecuencia Transportes Interurbanos de Tenerife, para que la experiencia del usuario durante la realización de la tarea sea mucho más llevadera.

Capítulo 2 Lenguajes de programación y tecnologías

En este capítulo se presentarán las diferentes tecnologías utilizadas para la realización del proyecto, analizando y comparando cada una de ellas para demostrar cuál se adapta mejor al tipo de procesamiento de la cámara de compensación.

2.1 Python

Python es un lenguaje de programación de alto nivel destacado por su sintaxis clara y legible, lo que hace que sea fácil de leer y escribir. Además, cuenta con una gran cantidad de librerías y módulos que facilitan el desarrollo de aplicaciones en distintas áreas, como la ciencia de datos, la inteligencia artificial, la web, la automatización de tareas, entre otras.

Es un lenguaje interpretado, lo que significa que no necesita ser compilado antes de ser ejecutado, sino que el intérprete de Python lee el código fuente y lo ejecuta directamente. Esto lo hace más flexible y rápido en el desarrollo, pero puede ser menos eficiente en términos de rendimiento en comparación con lenguajes compilados.

2.1.1 Pandas

Pandas es una poderosa librería de Python utilizada para el análisis y la manipulación de conjuntos de datos pequeños o medianos. Proporciona estructuras de datos flexibles y eficientes, así como herramientas para trabajar con conjuntos de datos estructurados: Series y Dataframes.

En este proyecto se ha utilizado el DataFrame, una estructura de datos bidimensional con columnas etiquetadas y filas indexadas. Se ha aprovechado la amplia gama de funciones que ofrece para manipular y transformar los datos. Además, promete la lectura y escritura en diferentes formatos, en este caso, en CSV.

Pandas permite el manejo de valores nulos y proporciona herramientas para trabajar con este tipo de datos. También, accede a los mismos utilizando etiquetas y ubicaciones numéricas, permitiendo indexar o filtrar para extraer subconjuntos de datos de manera eficiente.

Se basa en NumPy y aprovecha sus capacidades de cálculo vectorizado. Esto significa que se pueden aplicar operaciones a conjuntos de datos completos en lugar de iterar sobre ellos elemento por elemento, lo cual mejora la eficiencia y el rendimiento.

Debido a la limitación de escalabilidad, con dificultades para manejar grandes volúmenes de datos por su diseño basado en memoria, se ha decidido desarrollar el algoritmo de este proyecto en PySpark, observando notablemente la mejora del rendimiento. Además, el procesamiento en paralelo con Pandas está limitado ya que solamente puede ejecutarse en un solo hilo de CPU.

2.1.2 Apache Spark

Apache Spark [8] es un sistema de procesamiento distribuido de código abierto que se utiliza para el análisis y procesamiento de grandes volúmenes de datos. Se ha convertido en una de las tecnologías más populares para el procesamiento de Big Data, pues permite realizar operaciones avanzadas de análisis de datos, como consultas SQL, machine learning, etc.

Su capacidad para trabajar con datos distribuidos en clústeres, permite escalar fácilmente el

procesamiento para manejar grandes conjuntos de datos. Utiliza un modelo de programación llamado Resilient Distributed Datasets (RDD), que es una colección inmutable y distribuida de objetos.

2.1.3 PySpark

PySpark [13] es una librería de Python que proporciona una interfaz de programación para Apache Spark, un framework de procesamiento de datos distribuido y escalable. Esto permite procesar los datos en clústeres para aprovechar múltiples nodos y realizar operaciones en paralelo, mejorando el procesamiento de grandes volúmenes de datos y consiguiendo un alto rendimiento en el procesado.

Utiliza la optimización de consultas y ejecución en memoria para acelerar las operaciones. También admite el procesamiento en disco cuando los datos no caben en la memoria.

PySpark proporciona una API en Python que permite a los desarrolladores aprovechar la sintaxis y las capacidades del lenguaje, lo cual facilita el desarrollo y la implementación de aplicaciones de análisis y procesamiento de datos. También soporta múltiples fuentes de datos en la lectura y escritura de ficheros. Esto facilita la integración de diferentes tipos de datos en los flujos de trabajo de análisis.

Por otro lado, ofrece bibliotecas integradas para el procesamiento de datos, incluyendo SQL, machine learning (Spark MLlib), procesamiento de gráficos (GraphFrames) y procesamiento de streaming (Spark Streaming). Estas bibliotecas proporcionan funcionalidades avanzadas para realizar tareas como consultas SQL, construcción de modelos de machine learning y procesamiento de datos en tiempo real.

PySpark es altamente escalable y puede manejar grandes conjuntos de datos distribuidos en clústeres de cualquier tamaño. Puede ajustarse automáticamente para aprovechar los recursos disponibles y distribuir las tareas de procesamiento de manera eficiente en el clúster.

A continuación se puede observar una tabla que resume las ventajas y desventajas de PySpark y Pandas.

	PySpark	**Pandas**
Ventajas	Procesamiento distribuido y escalable en clústeres.	Fácil de aprender y usar para análisis de datos.
	Alto rendimiento en el procesamiento de datos.	Excelente para trabajar con datos en memoria.
	Soporte para grandes volúmenes de datos.	Amplia funcionalidad para manipulación de datos.
	Integración con lenguaje Python y otras bibliotecas.	Amplia comunidad y documentación disponible.
	Bibliotecas integradas para SQL, machine learning, etc.	Interfaz amigable y sintaxis intuitiva.
Desventajas	Mayor complejidad y curva de aprendizaje.	Limitado al tamaño de la memoria del sistema.
	Requiere configuración y	Rendimiento inferior al

	administración del clúster.	procesar grandes volúmenes de datos.
	Mayor uso de recursos y requerimientos de hardware.	No es adecuado para el procesamiento distribuido.
	Menor agilidad para análisis exploratorio y prototipado.	No tiene la capacidad de escalamiento horizontal.

Tabla 2.1: Ventajas y desventajas de PySpark y Pandas

Es importante tener en cuenta que la elección entre PySpark y Pandas depende del contexto y los requisitos del proyecto. PySpark es más adecuado para el procesamiento distribuido y el manejo de grandes volúmenes de datos, especialmente cuando se trabaja en entornos de Big Data. Pandas, por el contrario, es excelente para el análisis y la manipulación de datos en memoria en sistemas de tamaño moderado.

2.2 Power BI

Power BI es una plataforma de análisis de datos y visualización desarrollada por Microsoft. Permite a los usuarios conectarse a diversas fuentes de datos (base de datos relacionales, servicios en la nube, archivos locales, etc), transformarlos y modelarlos, para crear informes interactivos y paneles de control para visualizar y compartir los datos de manera efectiva.

Se utiliza en una amplia gama de organizaciones para el análisis de datos, el monitoreo del rendimiento empresarial, la toma de decisiones y la presentación de informes. Es una herramienta poderosa y versátil que permite a los usuarios transformar datos en información significativa y visualmente atractiva.

2.3 Visual Studio Code

Visual Studio Code (VS Code) [5] es un editor de código fuente desarrollado por Microsoft. Es una herramienta de desarrollo altamente personalizable que admite una amplia gama de lenguajes de programación y se utiliza en diversos entornos de desarrollo.

Es compatible en diversas plataformas, lo que permite a los desarrolladores utilizarlo en diferentes sistemas operativos. Además, se pueden instalar una amplia variedad de extensiones para agregar funcionalidad, como herramientas de depuración, control de versiones, etc. Proporciona una edición de código rápida y eficiente, con resultado de sintaxis para varios lenguajes de programación, lo que facilita la lectura y escritura de código.

Ofrece una terminal integrada en el editor, lo que permite a los desarrolladores ejecutar comandos y scripts directamente desde la interfaz de usuario. También permite personalizar el entorno de desarrollo según las preferencias (apariencia, comportamiento, configuración e instalación de temas y extensiones, etc).

Por último, Microsoft ofrece documentación detallada, soporte y actualizaciones regulares para mantener el editor actualizado y mejorar su funcionalidad.

2.4 Trello

Trello [6] es una herramienta en línea de gestión de proyectos y organización de tareas que utiliza tableros virtuales para ayudar al usuario a colaborar y realizar un seguimiento de sus

proyectos. Proporciona una interfaz visual y fácil de usar que permite organizar, priorizar y administrar tareas de manera efectiva.

En cada uno de los tableros/listas se pueden representar diferentes etapas o categorías de trabajo y en cada una de ellas, se pueden crear tarjetas para representar tareas específicas. Las tarjetas pueden tener descripciones, fecha límites, archivos adjuntos, etc.

Una de las ventajas es la colaboración en tiempo real, sobre todo, para proyectos grupales en los que se debe mantener una organización y observar el progreso de cada compañero. Además, permite la asignación de tareas, indicando quién debe completarla particularmente y agregar etiquetas para clasificarlas visualmente.

El uso de filtros para mostrar las tarjetas relevantes en función de ciertos criterios (miembro asignado, etiqueta, fecha, etc,) y el uso de notificaciones por correo electrónico o a través de la aplicación para mantener a los usuarios informados sobre cualquier cambio es bastante útil.

2.5 Java JDK

Java Development Kit [7] es un entorno de desarrollo de software utilizado para desarrollar aplicaciones. Proporciona las herramientas, compiladores y bibliotecas necesarias para escribir, compilar y ejecutar programas en Java.

En este proyecto es necesaria su instalación porque PySpark se ejecuta sobre la plataforma Apache Spark, que está escrita en Scala y se ejecuta en la Máquina Virtual de Java (JVM).

Aunque PySpark se escribe en Python, utiliza internamente la API de Spark, que está implementada en Scala. Cuando se ejecuta el código en PySpark, se produce una comunicación entre Python y el motor Spark, que se ejecuta en la JVM. Por tanto, es necesario tener el JDK instalado para que Spark pueda ejecutar y procesar los comandos y operaciones de PySpark.

En resumen, el JDK es necesario para ejecutar el entorno de ejecución de Java y permitir la comunicación entre Python y Apache Spark a través de PySpark.

2.6 Winutils Hadoop

WinUtils es una utilidad de Hadoop [9] diseñada específicamente para entornos Windows. Hadoop es un framework de código abierto utilizado para el procesamiento distribuido de grandes conjuntos de datos en clústeres de computadoras.

WinUtils proporciona una serie de comandos y herramientas que permiten a Hadoop funcionar correctamente en sistemas operativos Windows. Estas herramientas incluyen funciones esenciales como la creación de directorios, la gestión de permisos, la manipulación de archivos y otras operaciones relacionadas con el sistema de archivos distribuido de Hadoop (HDFS).

En este proyecto, se ha usado el comando chmod para permitir cambiar los permisos del directorio en HDFS, por tanto, se han realizado operaciones de administración y manipulación de datos de manera similar a como se haría en un entorno de Hadoop basado en Linux. Una vez situados en la carpeta de instalación de esta herramienta, ejecutamos en la terminal winutils chmod 777 C:\tmp\hive.

Es importante destacar que WinUtils debe ser configurado correctamente en el entorno de Hadoop en Windows para que las operaciones funcionen correctamente. Esto implica asegurarse de tener la versión correcta de WinUtils para la versión de Hadoop utilizada y configurar las variables de entorno apropiadas para que Hadoop pueda encontrar y utilizar las herramientas proporcionadas por WinUtils.

Capítulo 3 Metodología de trabajo

3.1 Metodología ágil Scrum

Durante la elaboración del proyecto se han mantenido reuniones semanales para valorar los avances del mismo. Por ello, la metodología ágil que más se ajusta a la organización que se ha llevado a cabo es la de Scrum [10].

Scrum es un enfoque de gestión de proyectos que se utiliza comúnmente en el desarrollo de software donde se requiere agilidad y colaboración. Se basa en principios de transparencia, inspección y adaptación para permitir la entrega de valor de manera incremental y continua.

Dado que el proyecto se ha realizado en colaboración con T.I.T.S.A, los roles que se han mantenido son los siguiente:

- Product Owner / Stakeholder: Responsable de definir los objetivos y gestionar el backlog del producto. Además, interesado en los resultados del desarrollo. En este caso, el cotutor del proyecto, Ginés Leon, se ha encargado de asumir este rol.

- Scrum Master y desarrollo: Asegura que se sigan las prácticas y los principios indicados del proyecto semanalmente. Además, realiza el trabajo necesario para entregar las funcionalidades solicitadas. Este cometido lo ha realizado Laura Cañizares.

Los eventos realizados son:

- Sprint: Una reunión semanal durante un mes para revisar los avances del proyecto.

- Planificación y revisión del Sprint: En las reuniones se han decidido las tareas que se comprometen a completar para el siguiente sprint. Además, se han comprobado los avances completados durante la semana de desarrollo para recibir retroalimentación del Product Owner y el tutor, Jose Luis, quien también ha ejercido el papel de Stakeholder. Se ha reflexionado sobre el pasado y se han buscado mejoras, listando las prioridades y los requisitos fundamentales del algoritmo.

La organización individual se ha realizado mediante la herramienta Trello. A continuación se puede observar la situación en la que se encontraba el proyecto durante el cambio de bibliotecas en el algoritmo.

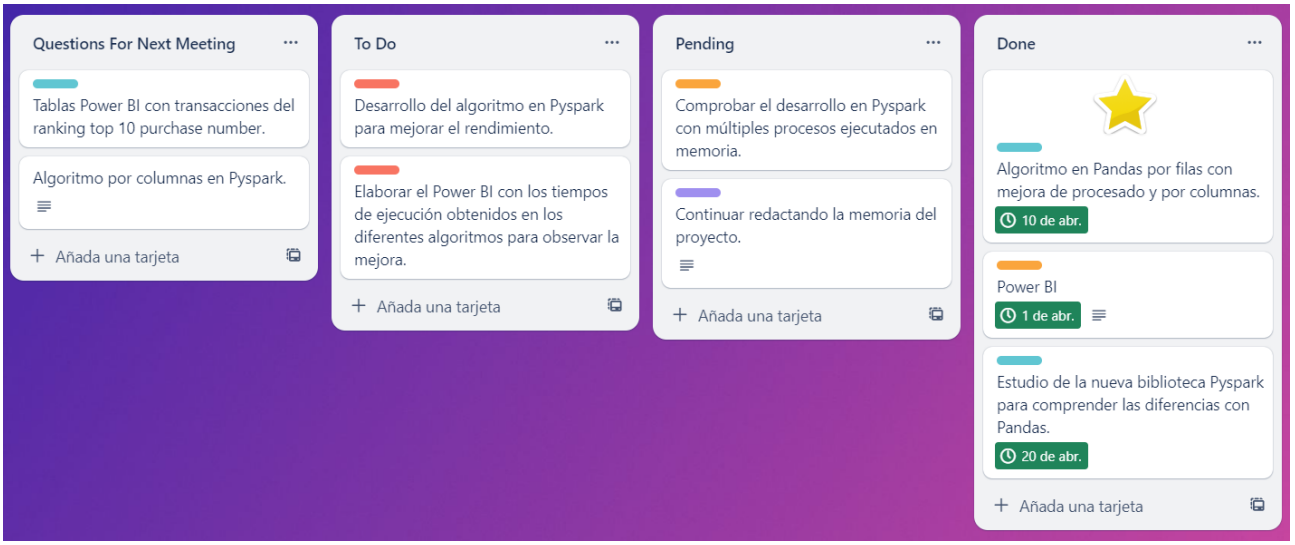

Figura 3.1: Organización del proyecto en Trello

Capítulo 4 Fases del desarrollo

4.1 Análisis del problema a resolver

Transporte Interurbanos de Tenerife, S.A, dispone de un sistema de cálculo para las compensaciones que computacionalmente consume muchos recursos y el tiempo para la obtención del resultado es considerablemente elevado. Esto ocurre porque la organización maneja grandes volúmenes de datos y utiliza Pandas, librería que manipula los datos en memoria en sistemas de tamaño moderado y realiza un excelente análisis, sin embargo, no es suficiente para optimizar los tiempos de la manera que desea la empresa colaboradora de este proyecto. Por ello, se pretende agilizar el procesado de los datos con el fin de disminuir el tiempo de espera en el que se disponen de los mismos, mejorando de esta manera la experiencia del usuario al ejecutar la tarea.

4.2 Análisis exploratorio de los datos

Tras solicitar los datos a la empresa colaboradora de este proyecto, Transporte Interurbano de Tenerife, S.A, se ha facilitado el Dataset en formato CSV con un muestreo de transacciones de dos meses consecutivos, además, el código del desarrollado hasta el momento por su parte. La primera fase del desarrollo por tanto, fue comprender el algoritmo y realizar varias trazas con ayuda del documento aportado por el cotutor, en el que se explicaba un ejemplo de cómo debía operar la cámara de compensación. A continuación se pueden observar los campos principales de la tabla con la que se trabaja en este proyecto.

Campo	**Nombre**
VENT_PURCHASE_NUMBER	ID del abono del usuario
VENT_FECHA_INS	Fecha de compra del abono
VENT_PRODUCT_CODE	Código del producto
VENT_IMPORTE	Importe del abono
TITULO_TIPO	Título del tipo del abono
VAL_TRANSACTION_ID	ID de la transacción de uso
VAL_FECHA_HORA	Fecha de validación de la transacción
VAL_TABLA_PROCEDENCIA	Tabla de la que proceden los campos
VAL_PASAJEROS	Número de pasajeros en la transacción
EMPRESA	Empresa en la que se realiza la transacción

Tabla 4.1 Campos de la tabla principal

Una vez entendido el código y cambiada la lógica para que se ejecutase con el formato CSV, ya que el algoritmo facilitado se alimentaba de la propia base de datos SQLServer de la organización, se ha conectado el resultado de la primera ejecución a una visualización en el Power BI, que tras

presentarlo en uno de los sprint semanales, se ha decidido continuar con el elaborado durante el resto de las ejecuciones del proyecto.

A continuación se detallarán las imágenes presentadas y una descripción de lo que suponen cada una de las gráficas. Todas ellas aportan información de calidad a la organización y logran una comprensión rápida de los datos explorados.

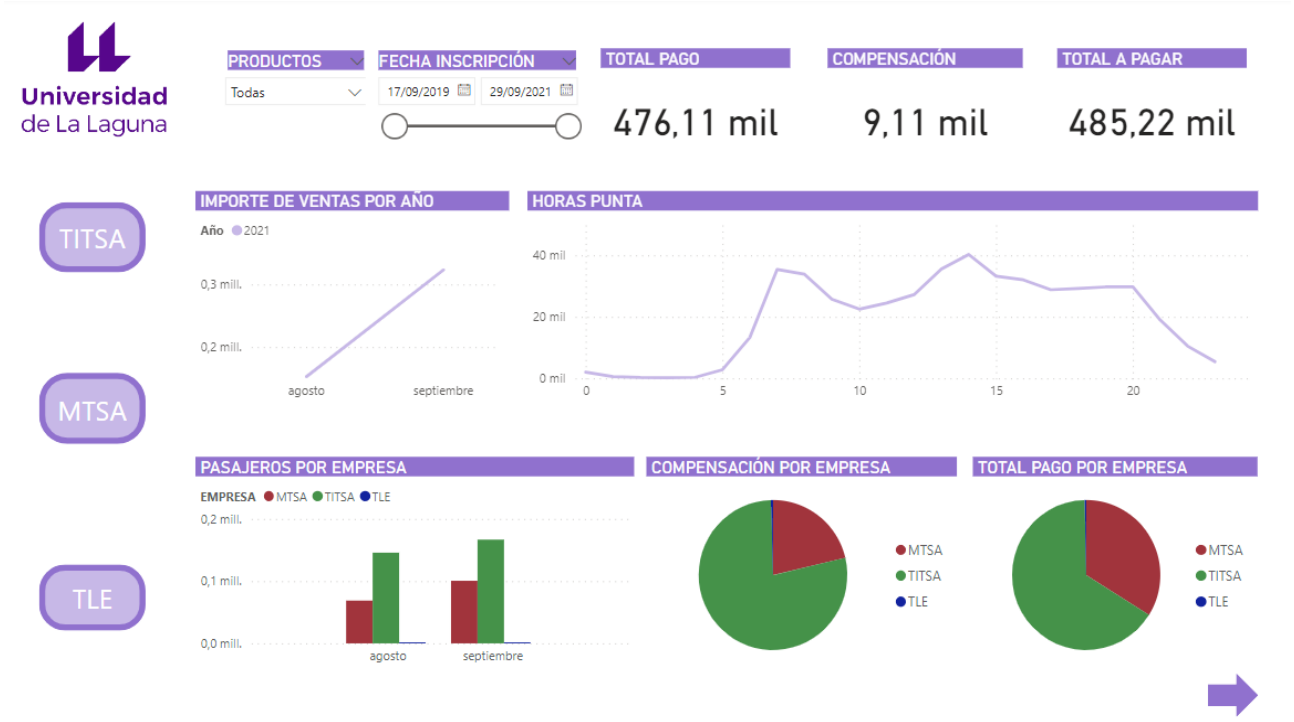

Figura 4.1: Panel de control inicial Power BI

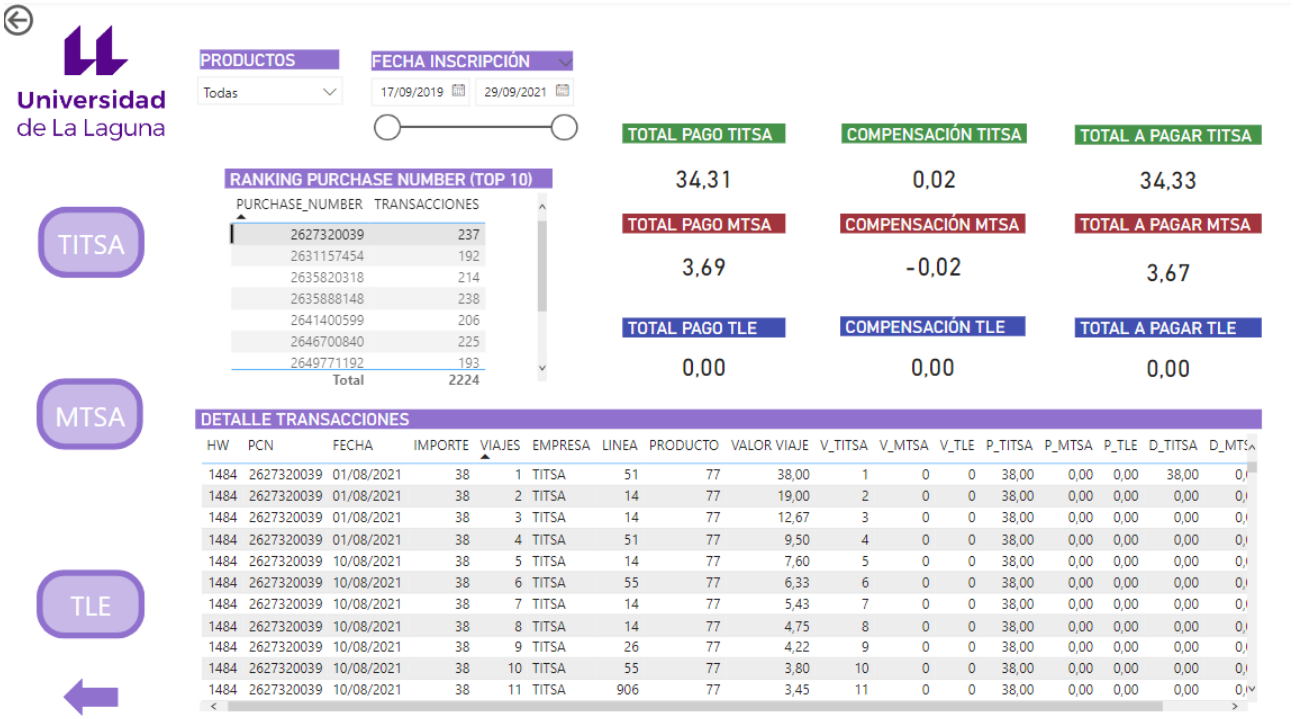

Figura 4.2: Panel de control inicial (2) Power BI

En el panel de control inicial del Power BI elaborado, se muestra un resumen general con los datos de todos los operadores. El dato más relevante es la compensación total a realizar, sin embargo,

también destacan las gráficas que informan de las horas puntas de los trayectos realizados, qué operador mueve más pasajeros, el mes con más ventas y el detalle de las transacciones de aquellos 10 bonos con más movimientos realizados durante un periodo de tiempo junto con su correspondiente pago y compensación por organización.

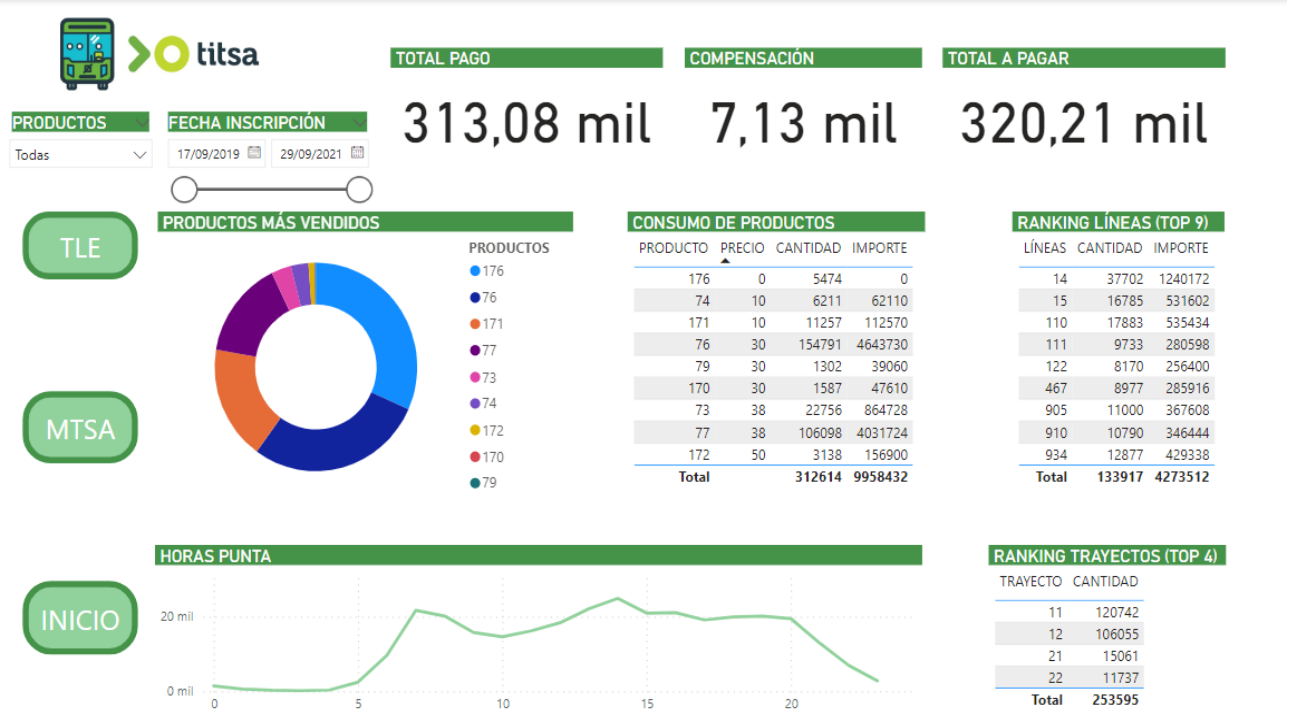

Figura 4.3: Panel de control del operador TITSA

Centrándose más en cada operador, se ha detallado su correspondiente análisis en cuanto a producto más vendido y su consumo, las líneas y los trayectos más usados junto con las horas puntas y las etiquetas que representan el análisis de cuánto se le debe compensar a cada uno de ellos. Destacar que también se puede acotar en un periodo determinado, pero los datos con los que se trabaja en este proyecto son dos meses consecutivos.

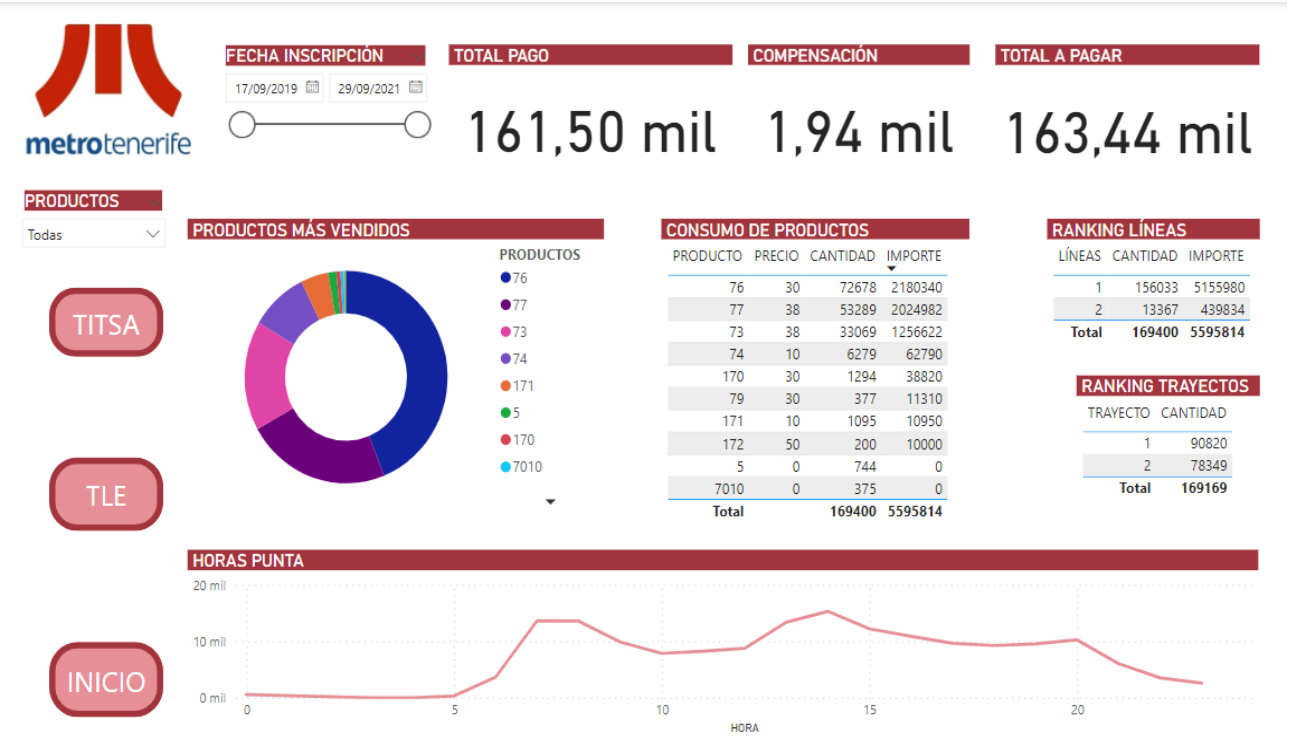

Figura 4.4: Panel de control del operador M.T.S.A

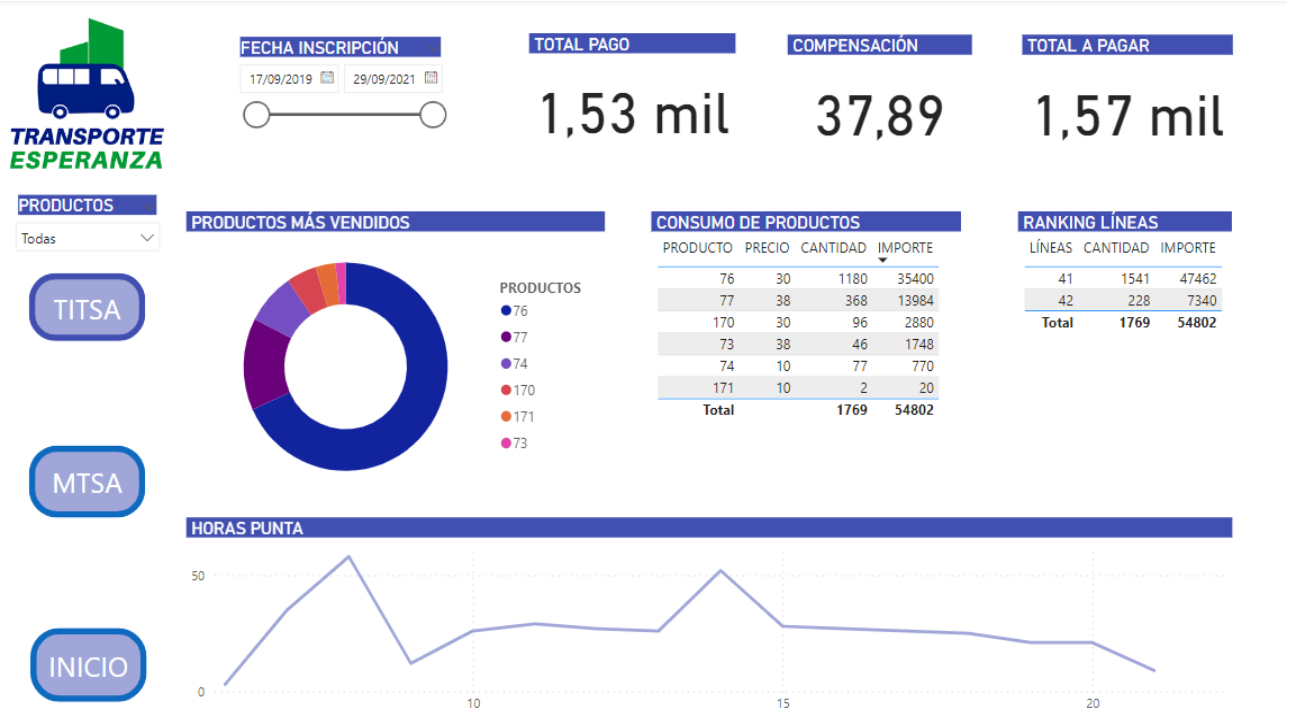

Figura 4.5: Panel de control del operador T.L.E

4.3 Análisis del algoritmo actual

Paralelamente, se ha continuado realizando mejoras de procesado al primer algoritmo aportado, insistiendo en que la idea más óptima era la de almacenar el último estado para evitar el procesado de lo antiguo en las nuevas transacciones, ya que este siempre contenía los datos necesarios para poder continuar con el cálculo de los próximos movimientos: precio del viaje, estado actual y la

compensación.

A continuación se detalla un ejemplo de un bono recargable semanal de 10 euros para que se entienda el algoritmo de la cámara de compensación con mayor claridad, y además, las fórmulas fundamentales utilizadas en la función principal de todos los algoritmos implementados durante este proyecto.

Es importante destacar que el usuario puede hacer uso ilimitado de todos los operadores en esa semana, y no se considera el precio de la tarifa que tiene fijado el operador para el servicio que vaya a tomar el usuario. El problema a solucionar busca dar respuesta a la pregunta: ¿De qué manera se reparten los 10 euros entre los tres operadores?

Si el usuario solo hace uso de T.I.T.S.A, los 10 euros irán para T.I.T.S.A. Si por el contrario, hace uso del abono en T.I.T.S.A y Metropolitano de Tenerife, los 10 euros irán equitativamente a ambos operadores, obteniendo 5 euros cada uno. En cualquier otro caso, se repartirá equitativamente el valor del título entre el número de veces que el usuario haga uso de cada uno de los operadores.

Para llevar un conteo del número de veces que un usuario hace uso de un operador, se rellena una matriz similar a la que se presenta en la Tabla 4.2, donde para cada transacción que compone el Dataset, teniendo en cuenta el operador que ha utilizado el usuario, se le da el valor de 1 y a los demás un 0.

FECHA - ID	TITSA	MTSA	TLE
DD/MM/AAAA - X	1	0	0
DD/MM/AAAA - X	0	1	0
DD/MM/AAAA - X	1	0	0

Tabla 4.2: Matriz que representa el viaje del usuario en el operador

Además, se tiene una matriz adyacente a la anterior que para cada una de las filas, se tiene un contador del número de viajes totales que se ha realizado por cada uno de los operadores y un contador acumulado total de viajes entre los operadores. Quedaría de la siguiente manera.

$$Total\ viajes\ TITSA = \sum Titsa$$

$$Total\ viajes\ MTSA = \sum Mtsa$$

$$Total\ viajes\ TLE = \sum Tle$$

$$Total\ viajes = \sum Titsa + \sum Mtsa + \sum Tle$$

Figura 4.6: Fórmula total viajes

FECHA - ID	TOTAL VIAJES	TOTAL VIAJES TITSA	TOTAL VIAJES MTSA	TOTAL VIAJES TLE
DD/MM/AAAA - X	1	1	0	0
DD/MM/AAAA - X	2	1	1	0
DD/MM/AAAA - X	3	2	1	0

Tabla 4.3: Matriz de viajes totales

En base a esta matriz, se calculan las columnas de compensación entre los operadores, puesto que se conoce el número de viajes para cada operador y el número de viajes totales que haya realizado el usuario, por lo que se irá repartiendo y compensando el valor del título entre las organizaciones. Hay que tener en cuenta que los operadores deberán de ir cediendo parte del valor del abono y posteriormente lo podrán o no recuperar en función del uso que realice el usuario. Para calcular el valor de las columnas valor viaje, estado actual y compensación, se emplean las siguientes fórmulas.

$$Valor\ viaje = \frac{Importe\ del\ abono}{Total\ viajes\ TITSA + Total\ viajes\ MTSA + \ Total\ viajes\ TLE}$$

$$Estado\ actual\ [TITSA] = Valor\ viaje * Total\ viajes\ TITSA$$

$$Compensación\ [TITSA] = Estado\ Actual\ [TITSA] - Estado\ Anterior[TITSA]$$

Figura 4.7: Fórmulas principales

La traza por tanto quedaría de la siguiente manera. En primer lugar, si el usuario realiza un viaje en T.I.T.S.A, el valor del bono recae íntegramente en T.I.T.S.A.

OPERACIONES					
OPERADOR	**VIAJES**	**VALOR VIAJE**	**ESTADO ACTUAL**	**ESTADO ANTERIOR**	**COMPENSACIÓN**
TITSA	1	10	10	0	10
MTSA	0	10	0	0	0
TLE	0	10	0	0	0

Tabla 4.4: Operaciones del primer viaje

RESUMEN									
VIAJES				COMPENSACIÓN			ESTADO		
TOTAL	TITSA	MTSA	TLE	TITSA	MTSA	TLE	TITSA	MTSA	TLE
1	1	0	0	+10€	0€	0€	+10€	0€	0€

Tabla 4.5: Resumen del primer viaje

En segundo lugar, el usuario realiza un viaje en M.T.S.A, por lo que T.I.T.S.A deberá de compensar a M.T.S.A con la parte proporcional que le toque, es decir, cada operador ha realizado un viaje, por lo que cada uno deberá de tener 5 euros. Como T.I.T.S.A tenía el importe completo del título, deberá de compensar a M.T.S.A con 5 euros.

OPERACIONES					
OPERADOR	VIAJES	VALOR VIAJE	ESTADO ACTUAL	ESTADO ANTERIOR	COMPENSACIÓN
TITSA	1	5	5	10	-5
MTSA	1	5	5	0	+5
TLE	0	5	0	0	0

Tabla 4.6: Operaciones del segundo viaje

RESUMEN									
VIAJES				COMPENSACIÓN			ESTADO		
TOTAL	TITSA	MTSA	TLE	TITSA	MTSA	TLE	TITSA	MTSA	TLE
2	1	1	0	-5€	+5€	0€	5€	5€	0€

Tabla 4.7: Resumen del segundo viaje

En tercer lugar, el usuario realiza un nuevo viaje en T.I.T.S.A, por lo que M.T.S.A deberá ceder parte del importe del título a T.I.T.S.A, debido a que ha realizado un viaje más.

OPERACIONES					
OPERADOR	VIAJES	VALOR VIAJE	ESTADO ACTUAL	ESTADO ANTERIOR	COMPENSACIÓN
TITSA	2	3,33	6,67	5	+1,67
MTSA	1	3,33	3,33	5	-1,67
TLE	0	3,33	0	0	0

Tabla 4.8: Operaciones del tercer viaje

RESUMEN									
VIAJES				COMPENSACIÓN			ESTADO		
TOTAL	TITSA	MTSA	TLE	TITSA	MTSA	TLE	TITSA	MTSA	TLE
3	2	1	0	+1,67€	-1,67€	0€	6,67€	3,33€	0€

Tabla 4.9: Resumen del tercer viaje

Por último, el usuario realiza un nuevo viaje en T.L.E, por lo que tanto M.T.S.A como T.I.T.S.A deberán ceder parte del importe del título a T.L.E.

OPERACIONES					
OPERADOR	VIAJES	VALOR VIAJE	ESTADO ACTUAL	ESTADO ANTERIOR	COMPENSACIÓN
TITSA	2	2,50	5	6,67	-1,67
MTSA	1	2,50	2,50	3,33	-0,83
TLE	1	2,50	2,50	0	+2,50

Tabla 4.10: Operaciones del cuarto viaje

RESUMEN									
VIAJES				COMPENSACIÓN			ESTADO		
TOTAL	TITSA	MTSA	TLE	TITSA	MTSA	TLE	TITSA	MTSA	TLE
4	2	1	1	-1,67€	-0,83€	+2,50€	5€	2,50€	2,50€

Tabla 4.11: Resumen del cuarto viaje

Se puede observar el orden de ejecución de la función principal del código inicial desarrollado en Pandas (línea a línea) con algunas de las sentencias principales.

En primer lugar, mediante la función apply se realiza la llamada a la función por fila con cada uno de los elementos que se encuentren en el dataset denominado *df_esqueleto*.

```
df_camara_compensacion = df_esqueleto.apply(__funcion_por_fila, axis=1)
```

Tabla 4.12: Función apply

Se aplica la función por fila, la cual reinicia los valores necesarios para el cálculo cuando se cambia de *Purchase Number*. De esta manera, se calcula la compensación correspondiente con la agrupación de transacciones correcta para cada empresa. Además, se ha modularizado el código con las llamadas a la *función por fila temporal* según el tipo de título que contiene la línea. Destacar que en este proyecto solamente se ha trabajado con las de tipo *temporal*, pero existen las de *monedero* y *viajes.*

```
funciones_por_fila_array = (None, None, __funcion_por_fila_temporal)
def __funcion_por_fila(row):
    if purchase_number_previo != purchase_number_actual:
        fecha_primera_validacion = row['VAL_FECHA_HORA']
        if previous_hw_number != row['VENT_HW_SERIAL_NUMBER']:
            balance_monedero = 0

        tipo_titulo_actual = funciones_por_fila_array[int(row['TITULO_TIPO_CODIGO'])]

        for empresa in pago_previo:
            pago_previo[empresa] = 0
            viajes_previo[empresa][0] = 0

        viajes_totales_previo += 1 if row['VAL_PASAJEROS'] >= 0 else -1
        row['Viajes Totales'] = viajes_totales_previo

    tipo_titulo_actual(row)
```

Tabla 4.13: Función por fila

```
def __funcion_por_fila_temporal(row):
    row['Precio Viaje'] = row['VENT_IMPORTE'] / row['Viajes Totales'] \
    if row['Viajes Totales'] != 0 else 0
    for empresa in pago_previo:
        pago_a_empresa = row['Precio Viaje'] * row[f'Viajes {empresa} Sin Transbordo']
        diferencia_empresa = pago_a_empresa - pago_previo[empresa]
```

Tabla 4.14: Función por fila temporal

En la función por fila temporal se aplican las tres fórmulas principales del algoritmo. Estas son: el cálculo del precio del viaje, resultado de la división del importe del abono entre la cantidad de viajes totales; el pago por empresa, el cual se obtiene del precio del viaje por la cantidad de viajes realizados en dicho operador. Y por último, la compensación de la transacción que se está realizando en ese momento, resta del pago actual menos el anterior.

4.4 Algoritmos propuestos

Tras el estudio previo de la idea más óptima y de cómo trabaja la cámara de compensación, se han instalado las librerías necesarias para poder realizar las mejoras oportunas en el procesado del código. Para tenerlas claras y ordenadas y por si en un futuro se quiere ejecutar en otra máquina, se ha elaborado un fichero de requerimientos el cual contiene todos los módulos necesarios para el proyecto. Mediante el comando *pip install -r requirements.txt* se instalarán todas las dependencias.

De este modo entra en juego la segunda fase del proyecto. Durante la misma, se desarrollan los dos algoritmos principales, uno de ellos menos óptimo, ideal para poder observar la mejora mediante visualizaciones con los cambios que sufre el código a lo largo del proceso.

El algoritmo inicial en Pandas, ha sufrido dos cambios considerables. El primero, guarda el último estado en un fichero externo el cual se lee en las siguientes ejecuciones para procesar los nuevos movimientos. Este algoritmo no mejoró el tiempo de procesado, el algoritmo era más complejo por las comprobaciones y la ordenación que requería el dataset, además, la lectura y escritura también ralentizaban las operaciones de ejecución.

Cuando se empezaron a realizar las operaciones por columnas, comenzó a mejorar el procesado en Pandas, fue entonces cuando se decidió comenzar a configurar el entorno para poder trabajar con PySpark con el procesamiento de múltiples procesos.

Se han instalado las siguientes tecnologías: Java JDK [1] para Windows x64, Apache Spark [2] y Winutils Hadoop [3]. Se ha tenido en cuenta la versión de descarga de esta última herramienta en la descarga del repositorio Git, ya que debe ser compatible con Apache para descomprimir la correcta, en este caso, la 2.7.1.

Una vez instaladas las herramientas, se ha procedido a configurar los valores de las variables de entorno del sistema operativo.

Variable	**Valor**
HADOOP_HOME	C:\winutils
JAVA_HOME	C:\java
SPARK_HOME	C:\spark-3.3.2-bin-hadoop2

Tabla 4.15: Variables de entorno

Además, se modifica la variable de entorno "Path" añadiendo las siguientes rutas de Spark y Java:

- %SPARK_HOME%\bin
- %JAVA_HOME%\bin

Destacar la importancia de la librería *findspark* del fichero de requerimientos mencionado anteriormente, ya que es la encargada de encontrar toda la instalación que se ha realizado de Apache Spark en el equipo local, por tanto, su función principal es conectar Python y Spark al encontrar la ruta de instalación del sistema y agregarla a la nueva variable de entorno *PYTHONPATH*. Esto es necesario porque la sentencia *findspark.init()* en el código, busca la ruta de instalación de Spark y la configura en la variable de entorno detallada. Posteriormente importa las clases y módulos de Spark necesarios para el desarrollo del código.

Tras superar las dificultades de configuración y desarrollo del código, se ha observado mejora del rendimiento al evitar el uso de bucles particionados por los subconjuntos de transacciones de los diferentes abonos. Esto hacía que los stage, unidades lógicas de trabajo en el plan de ejecución de Spark, empeoran el rendimiento tras el apilamiento de ejecuciones. Representan un conjunto de datos que se pueden ejecutar en paralelo y tienen las mismas dependencias de datos.

Se puede observar el código de la función principal desarrollada en PySpark, el cual realiza el cálculo por columnas.

```
def camara_compensacion(df, OPERADORES):
    #Definición de particiones por PN
    windowSpec = Window.partitionBy("VENT_PURCHASE_NUMBER") \
                                .orderBy("VIAJES") \
                                .rowsBetween(Window.unboundedPreceding, Window.currentRow)

    #Creación de columnas con las fórmulas principales
    df = df.withColumn("VIAJES", sum(f.col('VAL_PASAJEROS')) \
            .over(Window.partitionBy("VENT_PURCHASE_NUMBER") \
            .orderBy('VAL_TRANSACTION_ID')))

    df = df.withColumn("VALOR_VIAJES", (f.col('VENT_IMPORTE') / f.col('VIAJES')))
    for op in OPERADORES:
        df = df.withColumn(f"V_{op}", f  \
                .when(f.col('EMPRESA') == op, f.lit(1)).otherwise(f.lit(0)))
        df = df.withColumn(f"V_T_{op}", sum(f.col(f"V_{op}")).over(windowSpec))
        df = df.withColumn(f"E_{op}", (f.col('VALOR_VIAJES') * f.col(f"V_T_{op}")))
        df = df.withColumn(f"PREV_VALUE_{op}", coalesce(lag(f.col(f"E_{op}")) \
                .over(Window.partitionBy("VENT_PURCHASE_NUMBER") \
                .orderBy(f"VIAJES")), f.lit(0)))
        df = df.withColumn(f"C_{op}", (f.col(f"E_{op}") - f.col(f"PREV_VALUE_{op}")))
    return df
```

Tabla 4.16: Función cámara compensación PySpark por columnas

Ha sido fundamental el descubrimiento de las funciones de partición de windows y coalesce para abordar el objetivo principal del proyecto. La primera, particiona los datos y los ordena según la cláusula que se indique en la misma. Además, se han utilizado los límites del marco definidos del estándar, asignando el comienzo con valores ilimitados y el final con la fila actual. La segunda función, es útil cuando hay uno o más valores posibles que podrían asignarse a una variable o usarse en una situación determinada, y existe una preferencia conocida sobre qué valor entre las opciones debe seleccionarse para usar si está disponible.

Se han creado nuevas columnas para poder realizar el cálculo de la compensación en las mismas. En primer lugar, la columna viajes, en la que se suman los pasajeros de cada una de las transacciones, particionando las filas por cada purchase number y ordenando por transacción.

Por otro lado, el valor del viaje, división del importe del abono entre la columna anteriormente calculada, *VIAJES*.

Seguidamente, se crean tantas columnas como operadores existan. Se calcula la cantidad de viaje por empresa, formando una matriz con unos y ceros dependiendo si se ha usado ese operador o no, para posteriormente, calcular el total de viajes realizados en cada organización.

Luego, se calcula el pago actual a la empresa, resultado de la multiplicación del precio del viaje por la cantidad de viajes realizados en cada una de ellas. Finalmente para realizar la compensación, se ha utilizado la función coalesce, mencionada anteriormente, y lag para consultar en más de una fila de una tabla y poder obtener el valor de la fila anterior de la columna del pago a la empresa. De esta manera, se consigue restar el pago actual menos el de la transacción de la fila anterior para obtener la compensación del último movimiento realizado por el usuario.

4.5 Resultados

Cuando se ejecuta un programa en PySpark, el código pasa por un proceso llamado planificación de ejecución, en el que se construye un grafo dirigido acíclico (DAG) de transformaciones. Este grafo representa las dependencias entre las transformaciones y acciones que se deben ejecutar.

Este grafo se divide en etapas. Cada una consiste en un conjunto de transformaciones que no dependen de los datos de los otros, lo cual permite que las tareas se ejecuten en paralelo y mejoren el rendimiento.

Fue entonces cuando se ha decidido realizar el algoritmo sin el bucle que recorriera línea a línea y ejecutase la operación por columna en PySpark, aprovechando la mejor configuración del particionamiento posible. Además, una vez observada la mejora, se ha continuado avanzando en el código, mejorando la configuración [11] de la creación de la sesión de Spark. Se puede observar a continuación las sentencias del código de la configuración y lo que cada una de ellas significa.

```
#Creación de instancia en Apache Spark con su correspondiente configuración
  spark = SparkSession.builder.appName("Camara Compensacion")\
      .config('spark.master', 'local[8]')\
      .config('spark.executor.memory', '8g')\
      .config('spark.driver.memory', '8g')\
      .getOrCreate()

  spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")
```

Tabla 4.17: Mejora cálculo por columnas PySpark - multiprocesos

La propiedad *spark.master* indica una ejecución de paralelismo en 8 subprocesos. La siguiente, *spark.executor.memory* especifica la cantidad de memoria a asignar por ejecutor. Por otro lado, *spark.driver.memory.* El driver se iniciará ocupando un espacio de memoria. Como se puede configurar dicho espacio, mediante esta sentencia se especifica la cantidad de memoria RAM reservada para el driver que se ejecuta sobre el gateway.

De esta manera, se puede observar los resultados de las ejecuciones de los diferentes algoritmos en la siguiente figura, en la que se percibe la caída del tiempo con el uso de PySpark, mejorando el rendimiento que era el objetivo principal de este proyecto.

ID	Versión	Librería	Número de filas	Tiempo de lectura	Tiempo Python	Tiempo total
1	Inicial (línea a línea)	Pandas	483783	2,80	108,06	126,25
2	Mejora: Sin procesar lo antiguo (línea a línea)	Pandas	483783	2,46	146,00	148,83
3	Mejora: Cálculo por columna	Pyspark	483783	8,41	0,56	35,33
4	Mejora: Cálculo por columna - multiprocesos	Pyspark	483783	7,36	0,49	28,55

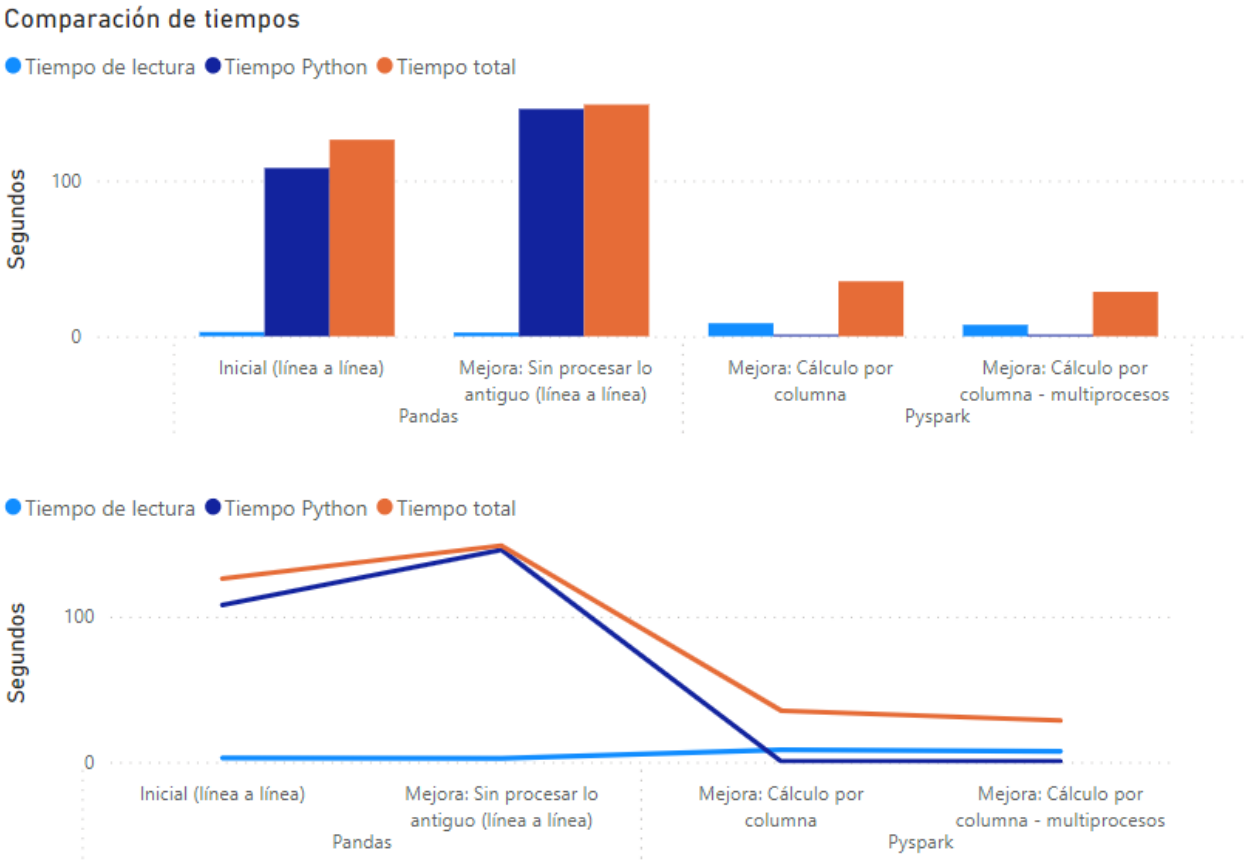

Figura 4.8: Gráficos de tiempos de ejecución

Capítulo 5 Conclusiones y líneas futuras

5.1 Conclusiones

El objetivo principal de este TFG se ha cumplido a través del estudio en detalle de los algoritmos iniciales propuestos, de las librerías y de las herramientas estudiadas culminando con una propuesta nueva de algoritmo de menor coste computacional para la liquidación de los importes entre operadores a través de la cámara de compensación.

Una de las dificultades encontradas durante la realización del trabajo de fin de grado fue el aprendizaje de PySpark, ya que muchos conceptos difieren con Pandas, y la elección de la idea más óptima en el algoritmo, pues al comienzo del proyecto se pensaba que lo mejor era evitar el reprocesamiento, guardando el último estado para que partiese de ahí en las nuevas transacciones. Esto implicaba tener que diferenciar en el código el último estado de cada bono y guardarlo en un nuevo fichero, para posteriormente leer este archivo y comenzar a partir de todos ellos con las nuevas transacciones. Sin embargo, con PySpark el cálculo realizado por columnas ha mejorado considerablemente el rendimiento, evitando tener que realizar las operaciones elemento por elemento.

La mejora en el procesado de la plataforma de datos ha supuesto una mayor eficiencia en el manejo y el análisis de los datos. Esto se traduce en mejores tiempos de respuesta y una mayor capacidad de procesamiento, lo que a su vez permite tomar decisiones más rápidas y precisas en la organización para elaborar estrategias mucho más confiables.

Por último, destacar la importancia de realizar el trabajo de fin de grado en colaboración con una empresa real como T.I.T.S.A, con la ayuda de expertos para analizar el problema y dar soluciones mucho más prácticas.

5.2 Líneas futuras

Se ha ejecutado el código utilizando el mejor recurso de memoria posible, pero si se tuviese que continuar con la mejora del procesamiento en un futuro, sería conveniente ejecutar el algoritmo en un clúster de máquinas distribuidas para que se realice el cálculo en paralelo. Esto debe de mejorar mucho más el coste en tiempo de espera.

Además, al mejorar el procesado de la plataforma, se pueden implementar técnicas y algoritmos más avanzados para limpiar y normalizar los datos. Esto resulta en una mayor calidad en los mismos, lo que a su vez mejora la confiabilidad de los análisis y resultados obtenidos. Se podría permitir la integración de nuevos tipos de datos y fuentes de información, lo cual brinda una mayor flexibilidad para ampliar el alcance de los análisis y aprovechar diferentes fuentes de datos para obtener una visión más completa de los problemas o situaciones que se están analizando.

Capítulo 6 Conclusions and future development

6.1 Conclusions

The main objective of this project has been fulfilled through the study of the initial algorithms proposed, the libraries and the tools studied, culminating with a new proposal algorithm with lower computational cost for the settlement of amounts between operators through the Clearing House.

One of the difficulties during the project was learning PySpark, because it has many different concepts with Pandas. Also the choice of the most optimal idea in the algorithm, because at the beginning it was expected that the best idea was to avoid reprocessing, saving the last state to start from there in new transactions. This implied having to differentiate in the code the last state of each bond and save it in a new file, to later read this file and start from all of them the new transactions. However, the calculation has improved with the algorithm by columns of PySpark, preventing having to perform the operations element by element.

The improvement in the processing of the data platform has supposed a greater efficiency in data management and analysis. This translates into better response times and greater processing capacity, which in turn allows faster and truthful decisions in the organization to develop better strategies.

Finally, I would like to highlight the importance of doing the final degree project in collaboration with a real company like T.I.T.S.A, with the help of experts to analyze the problem and give much more practical solutions.

6.2 Future development

The code has been executed using the best possible memory resource, but if I had to continue with improvements, it would be a good idea to run the algorithm on a cluster of distributed machines so that the computation is done in parallel. This should improve the cost in waiting time much more.

Also, improving the processing of the platform, we can implement more advanced techniques and algorithms to clean and normalize the data. This results in a higher quality in them and it improves the reliability of the analyzes and results obtained. The integration of new types of data and information sources could be allowed, giving greater flexibility to expand the scope of the analyzes and take advantage of different data sources to obtain a more complete vision of the problems or situations that are analyzed.

Capítulo 7 Presupuesto

La asignatura de Trabajo de Fin de Grado se basa en 12 créditos y cada uno equivale a 16 horas. Esto hace que la asignatura requiera en torno a 192 horas de trabajo. A continuación, se podrá observar en la tabla el desglose de los costes de desarrollo según las tareas llevadas a cabo durante el proyecto. Destacar que el precio de la hora de desarrollo es de 20 euros, partiendo de la experiencia laboral ofrecida por las empresas colaboradoras de proyectos innovadores. Además, se especifica el hardware utilizado para el desarrollo del trabajo con su correspondiente coste.

DESCRIPCIÓN	**HORAS**	**COSTE**
Análisis del algoritmo en Pandas y mejora de procesado	32	640
Enlace de resultados ejecutados con el Power BI	24	480
Estudio y configuración del entorno para utilizar PySpark	48	960
Algoritmo en PySpark	72	1440
Documentación	16	320
Lenovo Intel Core i7-1165G7, 16GB RAM		1384
Total	192	5224

Tabla 7.1: Coste de desarrollo

Capítulo 8 Códigos

Ambos códigos se pueden observar en el siguiente enlace que traslada al repositorio del perfil de la plataforma Github.

Bibliografía

[1] Oracle, “Java Downloads” https://www.oracle.com/es/java/technologies/downloads/#java20

[2] Downloads, “Apache Spark” https://spark.apache.org/downloads.html

[3] Github, “Winutils” https://github.com/steveloughran/winutils

[4] Linkedin, “Transformación digital en Mutua Tinerfeña” https://www.linkedin.com/pulse/transformaci%C3%B3n-digital-en-mutua-tinerfe%C3%B1a-/?originalSubdomain=es

[5] VS Code, “Qué es Visual Studio Code y qué ventajas ofrece” https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

[6] Trello, “Qué es Trello, para qué sirve y cómo funciona” https://blog.hubspot.es/marketing/que-es-trello

[7] JDK, “Java Development Kit” https://www.ibm.com/docs/es/i/7.3?topic=platform-java-development-kit

[8] Apache Spark, “Apache Spark: Introducción, qué es y cómo funciona” https://www.esic.edu/rethink/tecnologia/apache-spark-introduccion-que-es-y-como-funciona

[9] Apache Hadoop, “¿Qué es Apache Hadoop y para qué sirve?” https://www.profesionalonline.com/blog/big-data/que-es-apache-hadoop-y-para-que-sirve/

[10] IEBS, “Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla” https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

[11] Spark Conceptos Claves, “Spark Conceptos Claves” https://medium.com/@alonso.md/spark-conceptos-claves-33d0db88db8

[12] BOE, “Ley 41/1999” https://www.boe.es/eli/es/l/1999/11/12/41/con

[13] SparkBy{Examples}, “Pyspark Tutorial For Beginners (Spark with Python)” https://sparkbyexamples.com/pyspark-tutorial/