Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Computación

Ingesta de datos automatizada para un proyecto de Big Data

Alumno: Diego de Castro Fernández-Paniagua

Tutor UVa: Jesús María Vegas Hernández

Tutor Empresa: Javier Alonso Preciados

Aunque todo se haya acabado, lo pasé muy bien aprendiendo.

Agradecimientos

Gracias a mi hermano y a mis padres por estar siempre ahí.

Gracias a todos los *Lame9*, que son lo mejor que me podía haber dado esta carrera, no me voy a olvidar nunca de ninguno de vosotros.

Gracias en especial a Juan, que me ha acompañado durante una gran parte de mi vida y se ha convertido en un pilar fundamental de ella.

Gracias a mi tutor Jesús María Vegas por gestionar este proyecto, y a todo el profesorado que me haya enseñado a lo largo de la carrera.

Gracias a mi tutor de la empresa Javi y a Álvaro por echarme una mano con todo lo que he necesitado.

Y por último, gracias a mí.

Resumen

El objetivo de este proyecto es diseñar e implantar un proceso de ingesta de datos que se ejecute periódicamente de forma automática para maximizar la utilidad y facilidad de uso de los mismos.

Hoy en día se crean cantidades de datos tan grandes que resultaría inviable que los tratase una persona. Para arreglar esto, se puede programar una malla por la que se pasarán de forma automática esos datos cada vez que se reciban.

Esta malla estará compuesta por una cadena de *jobs* que se ejecutarán en orden y se encargarán de procesar los datos y pasarlos a un formato más limpio, ordenado y normalizado. Además, asegurará que los datos cumplan con un estándar de calidad fijado previamente.

Este proyecto será desarrollado para una entidad bancaria cliente, que enviará los datos mensualmente. Por este motivo todo el desarrollo se llevará a cabo en el entorno del banco, con las condiciones y requisitos que ellos impongan.

Por lo tanto, este proceso consistirá en una malla que hará la explotación y el uso de los datos más fácil y menos engorroso para el personal del banco, además de agilizar sus procesos y abaratar costes.

Palabras clave: Datos, ingesta, banco, *Big Data*, calidad, automatización, *cloud*, preprocesado, modelado, tablas.

Abstract

The goal of this project is to design and implement a data ingestion process that runs automatically on a monthly basis to maximize the ease of use and usefulness of the processed data.

Nowadays, such large amounts of data are created that it would be unfeasible for a person to process them. To fix this, a mesh can be programmed to automatically pass the data through each time these data are received.

This mesh will be composed by a chain of *jobs* that will run in order and will be responsible for processing the data and transforming it into a cleaner, more orderly and normalized format. In addition, it will ensure that the data meets a previously set quality standard.

This project will be developed for a client bank, which will send data on a monthly basis. For this reason, all the development will be carried out within the bank's environment, with the conditions and requirements that they impose.

Therefore, this process will consist of a mesh that will make the exploitation and use of the data easier and less cumbersome for the bank's staff, in addition to streamline their processes and reduce costs.

Keywords: Data, ingestion, bank, *Big Data*, quality, automation, *cloud*, preprocessing, modeling, tables.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Hoy en día se generan unos volúmenes de datos tan masivos que resultan muy difíciles de procesar y tratar mediante tecnologías convencionales. Esto se conoce como *Big Data*, y juega un papel crucial en la toma de decisiones empresariales, la investigación científica y la optimización de procesos.

Gracias a los avances en tecnologías de almacenamiento y análisis, se pueden manejar y analizar estas grandes cantidades de datos de forma automatizada y en tiempo real. Esto permite obtener una comprensión más profunda y valiosa de los datos, además de reducir los errores humanos y liberar recursos.

El objetivo de este proyecto es, pues, automatizar el procesamiento de una gran cantidad de datos de forma que se maximice la usabilidad y calidad de los mismos.

1.1. Contexto

Este trabajo de fin de grado ha sido realizado en convenio con la empresa *NTT Data España*. A la hora de llevar a cabo el proyecto el alumno ha contado con la ayuda de un tutor dentro de la empresa, Javier Alonso Preciados, quien le ha guiado con el desarrollo del proyecto.

El proyecto se trata de desarrollar una ingesta de datos para una entidad bancaria de España de la cual no es posible dar detalles por motivos de confidencialidad. Para comunicarse con ellos, la empresa de *NTT Data España* ha actuado como intermediaria entre el banco y el alumno.

1.2. Objetivos

El objetivo de este proyecto es el de desarrollar el proceso de una ingesta de datos para una entidad bancaria que se ejecute de forma automática y permita transformar los datos que reciba a un formato regularizado, asegurando que cumplen ciertos estándares de calidad.

Estos datos se recibirán una vez al mes, pero no se conocería en qué momento se recibirán ni qué tamaño tienen. Se puede suponer que estos datos sigan un formato que proporcione previamente la entidad bancaria.

La ingesta se podrá realizar a través de una malla ordenada de *jobs*, que se ejecutarán para ir adecuando los datos al formato deseado y asegurando que cumplen los estándares de calidad impuestos por la entidad bancaria.

Una vez haya ocurrido la ingesta, esta malla podría mostrar el resultado de su ejecución, ya sea positivo o negativo. Si el resultado es negativo, debería poderse detectar qué *job* ha fallado y, si es posible, el motivo del error.

Por último, sería lógico que solo miembros del personal del entorno de desarrollo de la entidad bancaria autorizados pudieran acceder a cualquier tipo de información relacionada con la ingesta.

1.3. Motivación

En cuanto a la motivación y objetivos personales por parte del alumno, son los siguientes:

- Conocer cómo se trabaja con tecnologías relacionadas con *Big Data*.
- Aprender cómo funciona la automatización en el procesamiento de datos.
- Participar en un proyecto que se vaya a implantar de verdad.
- Trabajar con una metodología y un entorno de trabajo reales.
- Aportar utilidad a la empresa y facilitar una posible incorporación a la misma.

1.4. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 2 Marco conceptual: Describe los conceptos teóricos que abarca el proyecto que sean necesarios para la comprensión del mismo.

Capítulo 3 Planificación: Describe la metodología seguida para el desarrollo del proyecto y las fases en que se compone, así como analizar los costes y posibles riesgos.

Capítulo 4 Análisis y diseño: Describe los requisitos, el modelo de dominio, los casos de uso y las decisiones de diseño tomadas a lo largo del proyecto.

Capítulo 5 Tecnologías y herramientas utilizadas: Describe las tecnologías y herramientas utilizadas en el desarrollo del proyecto.

Capítulo 6 Implementación y pruebas: Describe el proceso de desarrollo del proyecto y las pruebas que se han realizado hasta la solución implementada.

Capítulo 7 Seguimiento: Describe el histórico de cómo ha ido el desarrollo real del proyecto respecto a la planificación previa.

Capítulo 8 Conclusiones: Describe cómo ha ido la resolución del proyecto.

Bibliografía: Cita las fuentes que se han utilizado para consultar información en el desarrollo del proyecto.

Capítulo 2

Marco Conceptual

En este capítulo se tratarán los conceptos teóricos que engloban al proyecto para dar contexto al desarrollo y facilitar la comprensión de esta memoria. También se describen las tecnologías actuales más utilizadas para los conceptos descritos.

2.1. *Big Data*

El *Big Data* se define como aquella cantidad de datos tan masiva que no puede ser analizada o procesada por los métodos tradicionales.

A día de hoy se calcula que, solamente las empresas, ya generan una cantidad equivalente a dos trillones de *bytes* al día; y para 2025 se estima que se creen, en general, más de 465 trillones de *bytes* al día. Esta cifras toman una importancia tan grande que el 94 % de las empresas asegura que el entender y ser capaces de utilizar estos datos es esencial para asegurar un crecimiento competente de la empresa[15].

Lo que hace que el *Big Data* tome tanta importancia para las empresas es el hecho de que proporciona respuestas a muchas preguntas que las empresas ni siquiera sabían que tenían[26]. El análisis de estos datos permite a las empresas aprovechar mejor sus recursos y utilizarlos para encontrar nuevas oportunidades. Esto se traduce en una toma de decisiones más inteligente, más eficiente y con mayores ganancias.

El Big Data se distingue por tres características principales[31]:

- **Volumen:** Vistas las estadísticas anteriores, el *Big Data* se caracteriza por la cantidad tan masiva de información que lo compone, y esta información está creciendo de forma exponencial.
- **Variedad:** En la actualidad la naturaleza de los datos puede llegar a ser muy variada:

textos, imágenes, audios, vídeos, correos electrónicos, archivos, etc. Todo lo que se pueda medir y almacenar es información.

- **Velocidad:** La velocidad a la que se crean los datos a día de hoy es desmesurada. Páginas web, redes sociales, correos electrónicos, etc. Todo lo que hacemos en el día a día genera información.

2.1.1. Tipos de *Big Data*

Según la estructura que tengan los datos, pueden ser clasificados como uno de estos tres tipos[11]:

- **Datos estructurados:** Estos datos tienen un formato y tamaño fijo y definido. Este tipo de datos es el que se conoce tradicionalmente. Se suelen almacenar en bases de datos relacionales en forma de tabla. Algunos ejemplos pueden ser fechas, *strings* o *arrays*.
- **Datos semiestructurados:** Estos datos no se limitan a seguir un formato fijo, pero cuentan con marcadores que permiten separar sus distintos campos. Se caracterizan por poseer sus propios metadatos, que sirven para describir sus elementos y las relaciones entre ellos. Algunos ejemplos pueden ser las hojas de cálculo o los archivos de tipo CSV, HTML, JSON o XML.
- **Datos no estructurados:** Estos datos se presentan en el formato exacto en el que fueron tomados, puesto que no tienen un formato específico. Resulta imposible guardar este tipo de datos en una tabla, puesto que su información no se puede desgranar a tipos básicos de datos de una forma sencilla. Algunos ejemplos pueden ser los archivos PDF, documentos multimedia o los textos escritos a mano.

2.1.2. *Big Data* en la nube

La nube, también conocida como *cloud*, permite utilizar servicios a través de internet de manera que solo se pague por los recursos que usa y en el momento en que los necesita. Dicho de otro modo, es la entrega bajo demanda de potencia de cómputo, bases de datos, almacenamiento, aplicaciones y otros recursos informáticos, a través de Internet[2].

Esta forma de tratar la infraestructura proporciona varias ventajas[2]:

- Reducción de los costes. Al no tener que montar el sistema no hay que pagar por el hardware ni el software. Además, al tener unas tarifas concretas, los costes son más predecibles y hay menos posibilidades de sufrir improvistos en este ámbito.
- Mayor escalabilidad. Si se necesita ampliar el sistema basta con contratarlo y estará hecho casi instantáneamente.

- Mayor disponibilidad y alcance. El sistema estará disponible 24 horas al día y podrá ser accesible desde cualquier parte del mundo.
- Tolerancia a errores. Este tipo de sistemas suele tener incorporadas muchas medidas de seguridad como puedan ser copias de seguridad o servidores adicionales ante caídas del servicio.

Pero no todo son ventajas, algunos de los inconvenientes de optar por este tipo de infraestructuras pueden ser la necesidad de una conexión a internet rápida y si cortes, posible pérdida de seguridad al estar el sistema más expuesto a ataques, funcionalidad limitada únicamente a la que decidan implementar las empresas o simplemente la dependencia hacia otras compañías.

Actualmente existen varios proveedores que ofrecen plataformas de servicios en la nube. A continuación se nombran los más utilizados a día de hoy[2]:

- **Amazon Web Services:** Propiedad de Amazon. Fue el primer proveedor *cloud*, lo que le posicionó como una elección casi obligatoria para muchas empresas. AWS proporciona una plataforma en la nube confiable que utilizan miles de empresas de todo el mundo.
- **Azure:** Propiedad de Microsoft. Se ha invertido una gran cantidad de recursos en esta plataforma en los últimos años y por eso es la que más ha crecido. Ofrece varios tipos de servicios adaptados a las necesidades de las empresas.
- **Google Cloud:** Propiedad de Google. Google también es un proveedor de sistemas en la nube mediante su plataforma Google Cloud Platform. Debido a las opciones citadas anteriormente, le ha resultado complicado hacerse un hueco en este negocio, pero en los últimos años ha crecido en gran medida y a día de hoy es utilizado por grandes compañías.

En cuanto al proyecto, la entidad bancaria cliente hace uso de la plataforma de Amazon Web Services por las prestaciones que ofrece, por lo que ha sido la plataforma que se ha usado en el desarrollo.

2.1.3. Tecnologías *Big Data*

Para manejar cantidades de datos tan grandes son necesarios tanto el *hardware* como el *software* adecuados. Respecto al *hardware*, existen tecnologías diseñadas específicamente para el procesamiento paralelo de cantidades masivas de datos[17], que agilizan el procesamiento de los mismos. En cuanto al *software* para tratar datos no estructurados o semiestructurados, es necesario utilizar tecnologías distintas a las tradicionales[11]. Para ello existen muchas herramientas que han ido surgiendo a lo largo de los últimos años pero, sin duda alguna, la opción más extendida por un largo margen es Hadoop.

Apache Hadoop es un *framework open source* que permite el procesamiento distribuido de grandes conjuntos de datos en grandes clústeres de ordenadores utilizando modelos de programación sencillos. Implementa un paradigma llamado *MapReduce*, donde las ejecuciones se

dividen en pequeños fragmentos de trabajo de forma que cada uno de ellos se pueda ejecutar en cualquier nodo del clúster. Además, Hadoop incorpora un sistema de archivos distribuido llamado *Hadoop File Distributed System* (HDFS), que almacena los datos de forma distribuida entre los nodos de cómputo pero simulando que pertenezcan a un único archivo[12]. De esta forma, Hadoop está diseñado para ser escalable desde servidores individuales a miles de máquinas, cada una de las cuales ofrecería potencia de cómputo y almacenamiento local. En lugar de depender del *hardware* para ofrecer una alta disponibilidad, está diseñado para detectar y gestionar los fallos en la capa de aplicación, prestando así un servicio de alta disponibilidad que funciona sobre un gran clúster de ordenadores cada uno de los cuales podría ser propenso a fallos de forma individual[6].

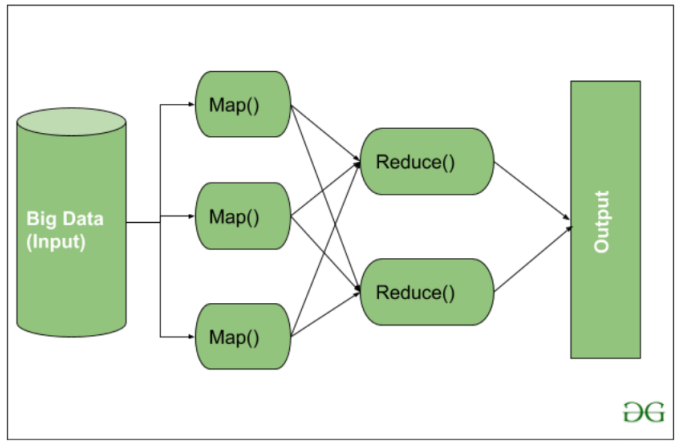

Figura 2.1: Diagrama explicativo de *MapReduce*[16]

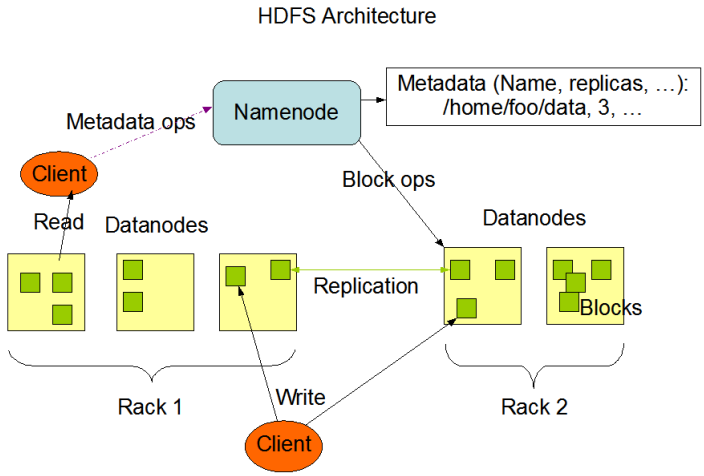

Figura 2.2: Diagrama explicativo de HDFS[7]

Hadoop cuenta con un gran ecosistema de tecnologías que hacen uso de él, algunas de las más importantes son las siguientes[3]:

- **Hive:** Permite acceder a HDFS como si fuera una base de datos relacional, mediante consultas muy similares a SQL. Esto hace más fácil el desarrollo y la gestión de un servicio mediante Hadoop, puesto que el lenguaje SQL es conocido de forma más general en el ámbito de la informática.
- **Sqoop:** Es una herramienta diseñada para transferir de forma eficiente grandes volúmenes de datos entre un almacenamiento estructurado como una base de datos relacional y un clúster de datos en Hadoop, y viceversa.
- **Flume:** Sirve para recolectar grandes cantidades de datos y cargarlos en Hadoop mediante *streaming*, por ejemplo, tomar datos de la red social Twitter a tiempo real.
- **Spark:** Es una herramienta de procesamiento de información a gran escala. Al contrario que *MapReduce*, realiza el procesamiento en tiempo real almacenando los datos en memoria en vez de en disco, lo cual le proporciona una velocidad de cómputo muy superior.

2.1.4. Análisis del *Big Data*

Por último, para que el *Big Data* resulte de utilidad, no basta con captarlo y almacenarlo, hay que hacer un análisis de la información. Este análisis supone un proceso de inspección, limpieza, transformación y modelado los datos con el objetivo de descubrir información útil, obtener conclusiones sobre los datos y ayudar en la toma de decisiones.

Para ello se han desarrollado muchas herramientas, entre ellas destacan las bibliotecas de Python Numpy y Pandas, cuyo uso está a la orden del día de los *data scientists*. Sin embargo, la herramienta más potente a la hora de analizar cantidades tan grandes de datos es Apache Spark[1].

Spark es un *framework* de computación distribuido similar al *MapReduce* de Hadoop. La principal diferencia entre ellos es que en vez de almacenar los datos en un sistema de ficheros distribuido como HDFS, Spark lo hace en memoria, lo que le hace ser mucho más ágil y eficiente que *MapReduce*. Sin embargo, si se requiere almacenar estos datos en un sistema distribuido, Spark es compatible con Hadoop, lo que le permite hacer uso de HDFS. Por lo tanto, en vez de verlos como dos tecnologías enfrentadas sería más correcto verlas como complementarias[1].

El diseño de Spark se basa en cuatro características principales[1]:

- **Velocidad:** Haciendo uso de tecnologías *multithread* y de procesamiento en paralelo y guardando los resultados intermedios de las ejecuciones en memoria, Spark consigue una velocidad de cómputo que ninguna otra herramienta alcanza.

- **Facilidad de uso:** Spark utiliza varias capas de abstracción sobre los datos como pueden los *Dataframes*, lo cual puede facilitar el desarrollo de aplicaciones *Big Data* incluso para desarrolladores que no tengan tanta experiencia con ello.
- **Modularidad:** Spark permite el uso de varios lenguajes de programación como Scala, Java, Python, SQL o R, así como una gran cantidad de bibliotecas. De esta forma es posible hacer varias funciones en una única aplicación desarrollada en Spark.
- **Extensibilidad:** Al centrarse Spark más en el procesamiento, la gestión de los datos se puede hacer con Hadoop, Cassandra, HBase, MongoDB, Hive o cualquier base de datos relacional, pero haciendo todos los cálculos en memoria.

2.2. Ciclo de vida del dato

El ciclo de vida de los datos es una serie de etapas por las que pasan los datos a lo largo de su vida útil durante el tiempo en el que son de utilidad. Existen varias visiones sobre cuáles son las fases del dato, así que en este caso se utilizará la descrita por Jeannette Wing en el documento *The Data Life Cycle*[32] publicado en el *Harvard Data Science Review*. Se utilizará esta clasificación porque toma un enfoque más centrado en la comprensión de la información que pueda aportar un dato que en el dato como unidad de almacenamiento de la misma. Esta clasificación consiste en las siguientes etapas:

Figura 2.3: Diagrama explicativo del ciclo de vida del dato

- **Generación:** El ciclo comienza con la generación de los datos. Todo lo que hacemos en el día a día, todo lo que se pueda medir, todo es información. Cada día se generan millones de datos que se captan de diferentes formas.

- **Colecta:** No todos los datos que se generan se colectan, ya sea porque no se necesitan, porque no son relevantes o porque se generan demasiado rápido.

- **Procesado:** En esta etapa se aplican diferentes procesos desde la limpieza de los datos y el ajuste de su formato hasta la compresión o encriptación de los mismos.

- **Almacenamiento:** Es aquí donde se guardan los datos procesados en el soporte de memoria deseado. Actualmente el almacenamiento más extendido son las memorias de estado sólido, pero en unos años se prevé que se empiecen a usar tecnologías más eficientes como el ADN.

- **Gestión:** Una vez almacenados, hay que gestionar este almacenamiento para que el acceso a los datos sea óptimo. Para los datos estructurados ya existen muchas estructuras de bases de datos que se encargan de esto, pero para los datos no estructurados o semiestructurados hay que desarrollar métodos nuevos, por ejemplo, mediante metadatos.

- **Análisis:** Esta etapa supone el grueso de la ciencia de datos. Aquí se incluyen todas las técnicas computacionales y estadísticas que puedan proporcionar información nueva en base a los datos que se utilizan: algoritmos, inteligencia artificial, *data mining*, *machine learning*, inferencia estadística, clasificadores, etc.

- **Visualización:** Tras el análisis se realiza una visualización de los resultados de forma que estos se puedan entender de manera más sencilla y visual. Como se suele decir, una imagen vale más que mil palabras, y es que en esta fase hay que tener muy en cuenta el factor humano y cómo percibimos la información.

- **Interpretación:** Por último, la información ha de interpretarse de forma correcta. Una vez esto ocurre y se han comprendido los resultados, el ciclo de vida de los datos se da por concluido.

2.3. Ingesta de datos

Una ingesta de datos es el proceso por el cual se introducen unos datos desde una fuente a otro sistema de almacenamiento. Es el primer paso dentro de una arquitectura de *Big Data*, por lo cual es de vital importancia tener claro la finalidad de la misma y cómo va a ser su diseño. Este diseño suele consistir en un *pipeline*, un proceso que toma los datos, los limpia y los deposita en el destino deseado; que suele ser un *data lake*, un lugar único al que llega información de varias fuentes diferentes[4].

Los *pipelines* de datos se dividen en varios procesos para separar las fases de procesamiento del dato y así mejorar el rendimiento, por ejemplo, no requiere la misma potencia computacional la limpieza de los datos que su análisis. Existe un modelo concreto de *pipeline* denominado ETL del que se entrará más en detalle más adelante en esta sección, puesto que ha sido el modelo utilizado para la ingesta de este proyecto[4].

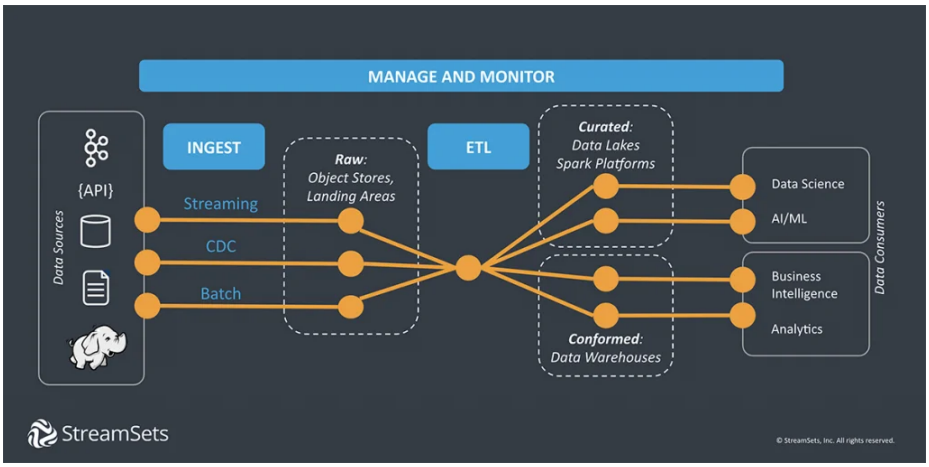

Figura 2.4: Diagrama explicativo del *pipeline* una ingesta de datos[28]

Existen dos formas de realizar el proceso de la ingesta[4]:

- ***Batch*:** Estos procesos se ejecutan con una periodicidad fija y unos datos estáticos. Se suele utilizar en situaciones en las que no se requiere de velocidad de ejecución y las cantidades de datos vienen de forma periódica y en cantidades muy grandes.

  La ingesta desarrollada en este proyecto es del tipo *batch*, ejecutándose con una periodicidad mensual.

- ***Streaming*:** Este tipo de procesos se ejecutan a tiempo real nada más se reciben los datos. En el momento en el que llega la información la ingesta la cargará de la forma más rápida posible en el destino deseado.

Algunas herramientas de las más utilizadas para diseñar ingestas son[4]:

- **Apache Sqoop:** Permite transferir datos entre el ecosistema de Hadoop y una base de datos SQL.

- **Apache Flume:** Sirve para ingestar datos semiestructurados o no estructurados a HDFS mediante *streaming*.

- **Apache Nifi:** Esta herramienta cuanta con una interfaz que permite diseñar gráficamente los procesos por los que pasan los datos, tanto en *batch* como *streaming*.

- **AWS Glue:** Ayuda a diseñar ETLs desde la consola de AWS y facilita su integración con el ecosistema de la plataforma de Amazon.

2.3.1. ETL

Las siglas ETL vienen de las palabras *Extract*, *Transform* y *Load*, por lo que una ETL es un proceso que extrae datos de una fuente, les aplica una serie de transformaciones y los carga en un destino[4].

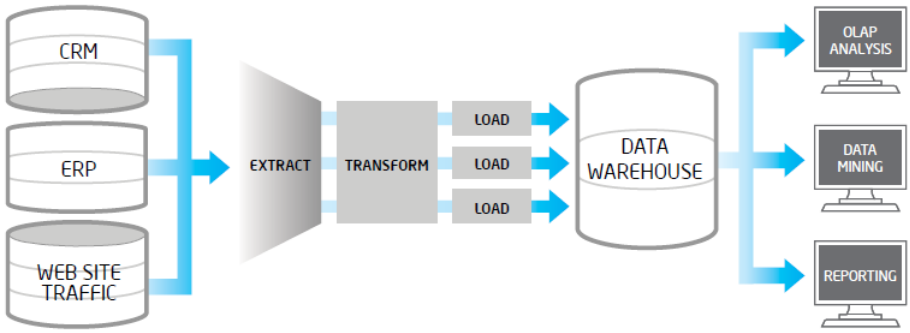

Figura 2.5: Diagrama explicativo de una ETL[13]

- **Extracción:** Es el proceso de captura de los datos y transporte al lugar donde se almacenarán. Las fuentes de datos pueden ser muy variadas y una única ETL puede beber de varias de ellas.

  Una parte importante de este proceso es el de analizar la calidad y el formato de los datos extraídos, asegurando así el correcto funcionamiento de la ingesta.

  La extracción debe de ser un proceso que no dependa del resto, con un coste computacional muy pequeño, y no debe modificar la fuente de la que se obtienen los datos.

- **Transformación:** En esta fase se efectúan los cambios necesarios en los datos para que se depositen en el destino en el formato deseado.

  Algunas transformaciones pueden ser: Eliminar datos duplicados, cruzar diferentes fuentes, agregar información en función de algún campo, generar identificadores, cambiar formatos de fechas o concatenar datos.

  Se puede necesitar transformar los datos para mejorar su calidad, integrarlos en sistemas ya existentes, normalizarlos, eliminar errores, etc.

- **Carga:** Por último, se guardan los datos en el destino deseado, ya sea un *data warehouse* o una base de datos.

  Esto significa que en esta fase la ETL interactúa directamente con el sistema de destino y deberá adaptarse a él buscando minimizar el tiempo de la operación.

  Existen diferentes tipos de sistemas en cuanto a la forma de cargar los datos se refiere: mediante SQL como Oracle, mediante ficheros como PostgreSQL o mediante operaciones propias como HDFS.

  Por último, tras informarse bien sobre el sistema de destino, hay realizar la distribución de índices, claves y particiones para que los datos se almacenen de la

forma correcta.

Capítulo 3

Planificación

Este capítulo describirá la planificación seguida para el desarrollo del proyecto. Se detallará la metodología seguida y las fases en las que he dividido el proyecto, así como un análisis de los costes y posibles riesgos que se puedan presentar.

3.1. Metodología

Para el desarrollo de este proyecto se ha utilizado la metodología *Scrum*. Esta metodología se basa en dividir el proyecto en diferentes fases progresivas, que se realizarán en periodos de una duración fija llamados *sprints*.

Se ha optado por esta metodología para familiarizar al alumno con ella de cara a una posible futura incorporación a la empresa, pues es la que utilizan normalmente sus equipos de trabajo.

En este caso se ha tomado la decisión de utilizar *sprints* con una duración estándar de 2 semanas. Entre estos *sprints* se han programado reuniones entre el alumno y el tutor asignado en la empresa para evaluar el desarrollo del proyecto y, si fuera necesario, realizar cambios en el alcance o la planificación del mismo.

En esta metodología se diferencian cuatro roles principales[27]:

- Cliente: Es el propietario del proyecto y quien aporta los fondos para su desarrollo. Debe conocer el alcance y los objetivos de forma clara para poder comunicarlos.
- *Product Owner*: Es quien conoce la visión del cliente y las necesidades del proyecto. Se encarga de la comunicación entre el cliente y el equipo del *Scrum*.
- *Scrum Master*: Se centra en asegurarse de que se sigan las prácticas *Scrum*. Ayuda a organizar y coordinar el proyecto entre el *Product Owner* y el equipo de desarrollo.

- Equipo de desarrollo: Grupo que se encarga de realizar las tareas definidas en el *sprint*.

En este proyecto, quien desempeña cada uno de los roles son los siguientes:

- Cliente: La entidad bancaria.
- *Product Owner*: Javier Alonso Preciados.
- *Scrum Master* y equipo de desarrollo: Diego de Castro Fernández-Paniagua.

3.2. Fases

Este proyecto se compone por 300 horas de trabajo, comenzando el día 12 de diciembre de 2022 y finalizando el día 20 de febrero de 2023. A un ritmo de trabajo de 6 horas por día, cinco días a la semana, esto equivale a cinco *sprints* de dos semanas cada uno.

El proyecto se ha dividido en cinco fases, una por *sprint*, en las que se realiza un trabajo diferenciado del resto pero siempre de forma incremental. Estas fases son las siguientes:

1. Análisis y modelado.
2. Desarrollo y pruebas en local.
3. Creación de los *jobs* y pruebas en *Work*.
4. Creación de la malla en *Live*, documentación e inicio de la memoria.
5. Memoria.

Por lo tanto las fases del desarrollo previstas quedarán de la siguiente forma:

Tarea	Sprint
Análisis de la tarea	1
Planificación	1
Modelado de las tablas	1
Registro de los *namings*	1
Repositorio	2
Códigos de ingestas	2
Códigos de calidad	2
Archivo CSV de prueba	2
Pruebas en local	2 y 3
Subida de los códigos	3
Jobs en Work	3
Pruebas en Work	3
Jobs en Live	4
Malla automatizada	4
Documentación	4
Memoria	4 y 5

Tabla 3.1: Fases del desarrollo

3.3. Costes

En este apartado se tratarán los costes que ha conllevado el desarrollo del proyecto en su totalidad.

3.3.1. Costes de personal

Este proyecto ha sido desarrollado por una única persona, al contar con menos de tres años de experiencia tomaré para su salario el salario medio de un ingeniero informático *junior*[29].

Este salario es de 12,02 € por hora, multiplicado por las 300 horas totales de trabajo, nos queda un total de 3.606,00 €.

Puesto	Coste
Ingeniero informático *junior*	3.606,00 €
Total	**3.606,00 €**

Tabla 3.2: Costes de personal

3.3.2. Costes de *hardware*

Los costes de *hardware* corresponden a la suma de los costes de cada pieza de equipo material que se ha utilizado para el desarrollo del proyecto.

Ahora bien, como el tiempo de desarrollo del proyecto ha sido de 300 horas no tendría sentido contar el coste total de los equipos, puesto que pueden ser utilizados para otros proyectos después. Para solucionar esto, he tomado como vida útil de los equipos un periodo de tres años.

Haciendo el cálculo, las 300 horas corresponden a un intervalo de 70 días desde el inicio del proyecto hasta su finalización, lo que supone aproximadamente un 6,39 % del total de su vida útil.

Equipo	**Coste**	**6,39 % del coste**
Ordenador portátil Dell Latitude 5420[14]	1.112,11 €	71,06 €
Ratón óptico Kensington Pro Fit[5]	37,48 €	2,39 €
Monitor Viewsonic VX Series VX2458-MHD[24]	361,00 €	23,07 €
Total	1.510,59 €	**96,52 €**

Tabla 3.3: Costes de *hardware*

3.3.3. Costes de *software*

Los costes de *software* han sido un total de 0 €, ya que todo el *software* utilizado en el desarrollo del proyecto ha sido gratuito o proporcionado por la entidad bancaria.

3.3.4. Otros costes

Estos son los costes relacionados con el entorno de trabajo. Se tomará en cuenta que la duración del proyecto ha sido de unas 300 horas totales en época de invierno.

Servicio	**Coste**
Internet	35,00 €
Electricidad	30,00 €
Agua	20,00 €
Calefacción	60,00 €
Total	**145,00 €**

Tabla 3.4: Otros costes

3.3.5. Total de costes

La suma de todos los costes corresponde a la siguiente:

Tipo	**Coste**
Costes de personal	3.606,00 €
Costes de *hardware*	96,52 €
Costes de *software*	0 €
Otros costes	145,00 €
Total	**3.847,52 €**

Tabla 3.5: Total de costes

3.4. Análisis de riesgos

Un riesgo es, según el PM-BOK[21]; *un evento o condición incierta que, si ocurre, tendrá un efecto positivo o negativo sobre al menos un objetivo del proyecto, llámese tiempo, costo, alcance o calidad.*

Este análisis trata de identificar, analizar y trazar un plan de actuación antes de que ocurran, para así prevenir o limitar el impacto que puedan tener a la hora de desarrollar el proyecto.

Para cuantificar el impacto que puedan tener los diferentes riesgos se usarán dos magnitudes diferentes: la probabilidad y el impacto.

- **Probabilidad:** Es la probabilidad de que el riesgo se materialice. Por supuesto, este valor no se puede medir de forma exacta, por lo que será una aproximación.

Probabilidad	**Rango**
Alta	Mayor de 50 % de probabilidad de ocurrencia
Significante	30-50 % de probabilidad de ocurrencia
Moderada	10-29 % de probabilidad de ocurrencia
Baja	Menos de 10 % de probabilidad de ocurrencia

Tabla 3.6: Intervalos de las probabilidades de los riesgos

- **Impacto:** Son las consecuencias que tendría el riesgo sobre el proyecto si llega a materializarse.

Impacto	Rango
Alto	Más de 30 % por encima del presupuesto
Significante	20-29 % por encima del presupuesto
Moderado	10-19 % por encima del presupuesto
Bajo	Menos de 10 % por encima del presupuesto

Tabla 3.7: Intervalos del impacto de los riesgos

3.4.1. Riesgos

A continuación se listan los principales riesgos detectados para el desarrollo del proyecto:

R001	Fallo en la estimación
Descripción	Debido a que la estimación es una aproximación, es posible que no se cumplan las fechas señaladas, si no se sigue la estimación afectará a la duración final del proyecto.
Probabilidad	Moderada.
Impacto	Significante.
Acciones de mitigación	Hacer una estimación realista y con algunos días de margen.

Tabla 3.8: R001 - Fallo en la estimación

R002	Desconocimiento de las tecnologías
Descripción	Debido a que el proyecto se desarrollará con tecnologías nuevas para el alumno, es posible que afecte a la duración final del proyecto.
Probabilidad	Significante.
Impacto	Bajo.
Acciones de mitigación	Solicitud y lectura de documentación y material formativo sobre las tecnologías que se van a utilizar.

Tabla 3.9: R002 - Desconocimiento de las tecnologías

R003	Incompatibilidad de versiones
Descripción	Las versiones de las herramientas y tecnologías utilizadas podrían no ser compatibles entre sí o con el equipo en el que se desarrolla el proyecto.
Probabilidad	Moderada.
Impacto	Significante.
Acciones de mitigación	Comprobar las dependencias de versiones y la compatibilidad de las herramientas con el equipo antes de empezar el desarrollo.

Tabla 3.10: R003 - Compatibilidad de versiones

R004	Indisponibilidad de las herramientas del banco
Descripción	Gran parte del desarrollo depende de las herramientas proporcionadas por la entidad bancaria, si estas no se encuentran disponibles repercutiría en la duración final del proyecto.
Probabilidad	Baja.
Impacto	Alto.
Acciones de mitigación	Procurar realizar todo el desarrollo necesario dentro de una misma herramienta en el mínimo número de periodos posibles.

Tabla 3.11: R004 - Indisponibilidad de las herramientas del banco

R005	Requisitos mal entendidos
Descripción	Los requisitos del proyecto los dicta la entidad bancaria y son transmitidos a través de un documento y diferentes correos electrónicos. Es posible que algunos de estos requisitos se hayan entendido mal y por tanto se comprometa la duración final del proyecto.
Probabilidad	Moderada.
Impacto	Significante.
Acciones de mitigación	Leer bien el documento y especificar de forma clara las dudas que se tengan para preguntárselas al banco.

Tabla 3.12: R005 - Requisitos mal entendidos

R006	Cambio en los requisitos del banco
Descripción	Como los requisitos los dicta la entidad bancaria cliente, pueden pedir un cambio en los mismos a mitad del desarrollo. Esto se vería reflejado en un aumento en la duración final del proyecto.
Probabilidad	Baja.
Impacto	Significante.
Acciones de mitigación	Planificar reuniones semanales con el cliente para mostrar el desarrollo y confirmar si el proyecto está avanzando en la dirección deseada.

Tabla 3.13: R006 - Cambio en los requisitos del banco

R007	Disponibilidad del alumno
Descripción	El alumno puede ver afectada su disponibilidad por motivos personales o de salud, lo que afectaría a la duración final del proyecto.
Probabilidad	Baja
Impacto	Moderado.
Acciones de mitigación	Realizar una buena planificación y dejar algunos días de margen en la estimación de los tiempos.

Tabla 3.14: R007 - Disponibilidad del desarrollador

R008	Indisponibilidad de la conexión a internet
Descripción	Si la conexión a internet del entorno de trabajo falla, la duración de muchas fases del desarrollo se verá comprometida.
Probabilidad	Baja
Impacto	Significante.
Acciones de mitigación	Tener una conexión alternativa como puedan ser los datos móviles.

Tabla 3.15: R008 - Indisponibilidad de la conexión a internet

R009	Pérdida del desarrollo
Descripción	Pérdida de fragmentos de código o documentación en desarrollo. Esto implicaría tener que repetir el desarrollo de lo que se ha perdido y afectaría a la duración del proyecto.
Probabilidad	Baja
Impacto	Alto.
Acciones de mitigación	Tener un repositorio y una copia de seguridad en la nube de todo lo que se vaya desarrollando.

Tabla 3.16: R009 - Pérdida del desarrollo

R010	Retrasos por parte del banco
Descripción	Si la entidad bancaria se demora mucho en responder o en enviar los datos desde el día en el que estaba estipulado, el proyecto podría quedar paralizado hasta el momento en el que se arreglase.
Probabilidad	Baja
Impacto	Alto.
Acciones de mitigación	Asegurarse de comunicar al banco de forma clara la importancia de que no existan retrasos por su parte.

Tabla 3.17: R010 - Retrasos por parte del banco

Capítulo 4

Análisis y diseño

Este capítulo describirá la información necesaria previa al desarrollo del proyecto. En la parte de análisis se encuentra la descripción del modelo de dominio, los requisitos del proyecto y sus casos de uso. Por otra parte, en cuanto al diseño, se hará una descripción de la solución que se ha implementado para el proceso de la ingesta de datos.

4.1. Modelo de dominio

En el modelo de dominio se definen las entidades, atributos y relaciones que dan forma al proyecto. En este caso, el modelo es el siguiente:

- **Datos:** Son la información en forma de tabla que se va a ingestar. Los datos son originalmente provenientes de la entidad bancaria y lo único que se sabe de ellos es la estructura que tienen inicialmente y la que se desea que tengan tras acabar el proceso. Inicialmente estarán en formato CSV, pero pasarán por los formatos Avro y Parquet a lo largo de la ingesta.
- **Esquemas:** Son documentos que detallan la estructura exacta que deberán seguir los datos. Existirá un esquema inicial para los datos originales y otro para los datos tras la ingesta. Tienen un formato JSON.
- ***Namings*:** Son los nombres logísticos que les da el banco a los diferentes campos de los datos, por separado de los nombres funcionales. Se utilizan para diferenciar de forma única a cada campo de cara a sus bases de datos y reutilizarlos si se repiten en más de una tabla. Deben ser únicos y hay que consultar al banco para registrarlos.
- **Ingesta:** Es el proceso por el cual se toman datos de una fuente y se llevan a otra, ya sea una base de datos, un modelo de procesamiento u otros destinos. En el caso de este proyecto el término ïngesta"se usa para referirse a una ETL.

- **ETL:** Siglas de *Extract*, *Transform*, *Load*. Consiste en un proceso que toma información, le aplica las transformaciones que se requieran y la carga en otro sitio. En este proyecto la ETL la conforma una malla compuesta por cinco *jobs*.
- **Malla:** Es una cadena de *jobs* que se ejecutarán en un orden estricto uno tras otro esperado a que acabe la ejecución del anterior. En esta ingesta la malla está formada por cinco *jobs* en el siguiente orden: Calidad, Ingesta, Calidad, Ingesta, Calidad.
- ***Job*:** Es un programa preparado para ejecutarse de forma automática en el sistema en la nube del banco. Cada *job* tiene asignado un archivo de configuración según el trabajo que tenga que desarrollar: los *jobs* de ingesta de datos y sus transformaciones hacen uso de la biblioteca de Yoshi y los que aseguran la calidad de los mismos hacen uso de la biblioteca de Sabium.
- **Yoshi:** Biblioteca propia de la entidad bancaria. Se utiliza para programar los archivos de configuración encargados de las ingestas de datos y las transformaciones que se les apliquen.
- **Sabium:** Biblioteca propia de la entidad bancaria. Se utiliza para programar los archivos de configuración encargados de asegurar que los datos cumplen los estándares de calidad impuestos por el banco.
- **Fases del dato:** En este proyecto los datos pasan por tres fases: *Staging*, que son los datos tal y como se reciben; *Raw*, que son los datos que están en proceso de ser transformados; y *Master*, que son los datos finales ya totalmente tratados.
- **Entornos del desarrollo:** En este proyecto, por convenio del banco, se trabaja en tres entornos distintos: primero se trabaja en el entorno local, donde se desarrollan los archivos de configuración y algunas pruebas; después se pasa al entorno de *Work*, donde se crean jobs temporales para hacer las pruebas de ejecución de los mismos; finalmente, en el entorno de *Live*, se crean la malla y los jobs definitivos una vez se ha comprobado que todo funciona correctamente.

4.2. Requisitos

Los requisitos son la descripción del comportamiento que deberá tener el total del proyecto al final del desarrollo. Estos requisitos se pueden agrupar en tres categorías:

- Requisitos funcionales: Describen las funciones que debe realizar el proyecto.
- Requisitos no funcionales: Describen las características implícitas que se esperan del proyecto.
- Requisitos de información: Describen los datos e información que debe almacenar el proyecto.

A continuación se enumeran los requisitos que se han detectado para el desarrollo del proyecto.

4.2.1. Requisitos funcionales

Código	Requisito	Descripción
RF-01	Ingesta	El sistema devolverá la tabla resultante de aplicar el proceso de ingesta a los datos de entrada.
RF-02	Asegurar calidad	El sistema mostrará si los datos han cumplido las reglas de calidad establecidas.
RF-03	Resultado de los *jobs*	El sistema mostrará el resultado de ejecución de cada job, tanto si es positivo como negativo.
RF-04	Códigos de error	En caso de que un *job* falle, el sistema mostrará un código de error que permita al banco conocer el motivo del fallo.

Tabla 4.1: Requisitos funcionales

4.2.2. Requisitos no funcionales

Código	Requisito	Descripción
RNF-01	Mensualidad	La malla de la ingesta ejecutará cada vez que lleguen los datos del banco, una vez al mes.
RNF-02	Reglas de calidad	Los datos deberán cumplir ciertas reglas de calidad de la biblioteca Sabium especificadas por el banco.
RNF-03	Transformaciones	Los datos deberán utilizar transformaciones de la biblioteca Yoshi para modelar los datos al formato deseado.
RNF-04	Origen de los datos	Los datos provendrán siempre del mismo origen, en el mismo formato y con la misma estructura.
RNF-05	Fases del dato	La ingesta deberá estar separada según las fases de *Staging*, *Raw* y *Master* por estándares de organización del banco.
RNF-06	Formato de los datos	Los datos siempre llegarán en formato CSV. Deberán pasar por formato Avro en la fase de *Raw* y acabar en formato Parquet en la fase de *Master*.
RNF-07	Particiones	Las tablas finales de la ingesta deberán acabar particionadas por la fecha de ejecución y el país de origen de los datos.
RNF-08	Esquema inicial	La estructura que tienen los datos de origen viene descrita en un documento proporcionado por el banco.
RNF-09	Esquema final	La estructura que se desea que tengan los datos finales está reflejada en esquemas JSON proporcionados por el banco.

RNF-10	Tecnologías	La ingesta se creará utilizando las herramientas del entorno de trabajo del banco. Las tablas se crearán en Power Designer, los *jobs* en Dataproc, la malla en Control-M y los archivos de configuración se crearán utilizando las bibliotecas de Yoshi y Sabium.

Tabla 4.2: Requisitos no funcionales

4.2.3. Requisitos de información

Código	Requisito	Descripción
RI-01	Datos origen	Archivo CSV que contiene los datos de origen proporcionados por el banco.
RI-02	Tablas	Se almacenarán tanto la tabla final de *Master* en formato Parquet como la intermedia de *Raw* en formato Avro.
RI-03	Esquemas	Archivos JSON que describen la estructura de los datos de origen y los datos finales.
RI-04	Archivos de configuración	Se adjuntan a los *jobs* para determinar su funcionamiento. Se crean mediante las librerías de Yoshi y Sabium.
RI-05	*Logs* de ejecución	Se almacenan los *logs* de las ejecuciones de los *jobs* para poder mostrar el resultado de las mismas.
RI-06	Tablas de calidad	Tablas que determinan internamente qué reglas de calidad se han pasado y con cuánto margen. Son el resultado de los jobs de calidad que ejecutan los archivos de configuración creados con Sabium.

Tabla 4.3: Requisitos de información

4.3. Casos de uso

En esta sección se analizarán los casos de uso del sistema desarrollado.

4.3.1. Diagrama de casos de uso

Este diagrama representa de forma esquemática las interacciones entre los actores y el sistema desarrollado para hacer uso de las diferentes funcionalidades implementadas.

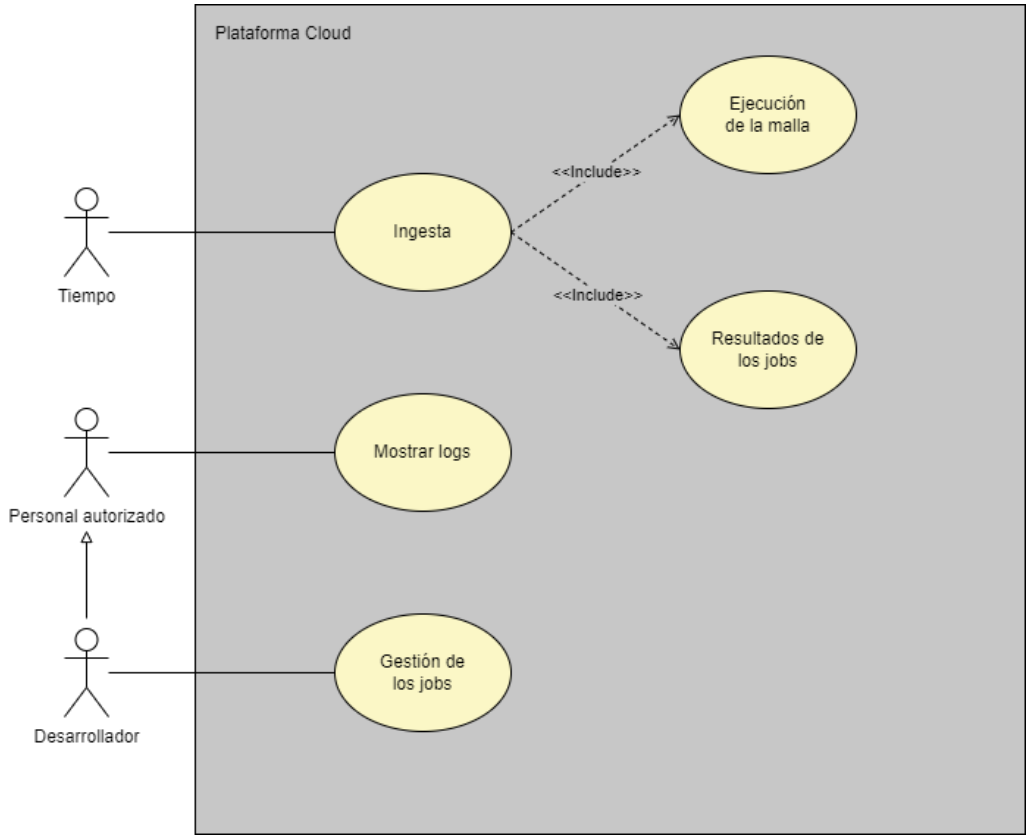

Figura 4.1: Diagrama de casos de uso

4.3.2. Descripción de los casos de uso

A continuación se detalla la información de cada caso de uso mostrado en la sección anterior más en profundidad.

Identificador	CU-01 Ingesta
Actor principal	Tiempo.
Descripción	Se ejecuta la ingesta de forma automática.
Secuencia normal	1. El sistema se activa el día 15 del mes si es un día hábil. 2. El sistema extrae el fichero de datos de la ruta de origen. 3. Se ejecuta el caso de uso 4. El sistema deposita la tabla con los datos resultantes en la ruta de destino. 5. Se ejecuta el caso de uso

Secuencia alternativa	1.1. Si el día 15 es festivo o fin de semana, el sistema planificará la ejecución para el próximo día hábil más cercano y el caso de uso termina. 2.2. Si la entidad bancaria no ha depositado el fichero de datos en la ruta de origen, la ingesta no se ejecutará y el caso de uso termina.
Postcondiciones	1. Se ha almacenado la tabla con los datos resultantes en la ruta de destino. 2. Se ha mostrado el resultado de las ejecuciones de todos los *jobs* de la malla. 3. Se han guardado los *logs* con información sobre las reglas de calidad de *Staging*, *Raw* y *Master*.

Tabla 4.4: CU-01. Ingesta.

Identificador	CU-02 Ejecución de la malla
Actor principal	Automático.
Descripción	Se ejecuta la ingesta de forma automática.
Secuencia normal	1. El sistema ejecuta el *job* de calidad de *Staging*. 2. El sistema guarda un *log* con información sobre las reglas de calidad de *Staging*. 3. El sistema ejecuta el *job* de ingesta de *Raw*. 4. El sistema ejecuta el *job* de calidad de *Raw*. 5. El sistema guarda un *log* con información sobre las reglas de calidad de *Raw*. 6. El sistema ejecuta el *job* de ingesta de *Master*. 7. El sistema ejecuta el *job* de calidad de *Master*. 8. El sistema guarda un *log* con información sobre las reglas de calidad de *Master*.
Secuencia alternativa	3.1 Si el *job* de ingesta de *Raw* falla, la ingesta no continúa y el caso de uso termina. 6.2 Si el *job* de ingesta de *Master* falla, la ingesta no continúa y el caso de uso termina.
Postcondiciones	1. Se han ejecutado todos los *jobs* de la malla. 2. Se han guardado los *logs* con información sobre las reglas de calidad de *Staging*, *Raw* y *Master*.

Tabla 4.5: CU-02. Ejecución de la malla.

Identificador	CU-03 Resultados de los *jobs*
Actor principal	Automático.
Descripción	Se muestran los resultados de las ejecuciones de los *jobs*.
Precondiciones	1. Se han ejecutado los *jobs* de la malla.

Secuencia normal	1. El sistema muestra el resultado de la ejecución del *job* de calidad de *Staging*. 3. El sistema muestra el resultado de la ejecución del *job* de ingesta de *Raw*. 4. El sistema muestra el resultado de la ejecución del *job* de calidad de *Raw*. 6. El sistema muestra el resultado de la ejecución del *job* de ingesta de *Master*. 7. El sistema muestra el resultado de la ejecución del *job* de calidad de *Master*.
Secuencia alternativa	
Postcondiciones	1. Se ha mostrado el resultado de las ejecuciones de todos los *jobs* de la malla.

Tabla 4.6: CU-03. Resultados de los *jobs*.

Identificador	CU-04 Mostrar *logs*
Actor principal	Personal autorizado.
Descripción	Se muestran los *logs* de reglas de los *jobs* de calidad.
Precondiciones	1. Se han ejecutado los *jobs* de la malla. 2. Se han guardado los *logs* con información sobre las reglas de calidad de *Staging*, *Raw* y *Master*.
Secuencia normal	1. El personal autorizado solicita ver los *logs* de reglas de los *jobs* de calidad. 2. El sistema muestra el *log* con información sobre las reglas de calidad de *Staging*. 3. El sistema muestra el *log* con información sobre las reglas de calidad de *Raw*. 4. El sistema muestra el *log* con información sobre las reglas de calidad de *Master*.
Secuencia alternativa	
Postcondiciones	1. Se han mostrado los *logs* de reglas de los *jobs* de calidad de la malla.

Tabla 4.7: CU-04. Mostrar *logs*.

Identificador	CU-05 Gestión de los *jobs*
Actor principal	Desarrollador.
Descripción	Se modifica la información de alguno de los *jobs*.

Secuencia normal	1. El desarrollador selecciona un *job* de la malla. 2. El sistema muestra los parámetros del job. 3. El desarrollador modifica los parámetros que desee: el nombre, la ruta al archivo de configuración, el día de planificación o el borrado del propio *job*. 4. El sistema modifica el *job* en la forma que se ha indicado y el caso de uso termina.
Secuencia alternativa	
Postcondiciones	1. Se han modificado los parámetros del *job*.

Tabla 4.8: CU-05. Gestión de los *jobs*.

4.4. Diseño del proceso de ingesta

El proceso de la ingesta se estructura de la misma forma que las fases del dato.

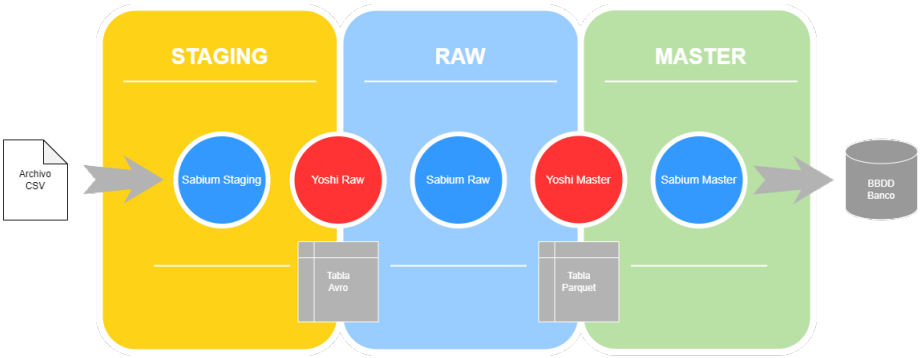

Figura 4.2: Diagrama explicativo del proceso de la ingesta

La ingesta comienza con un archivo en formato CSV que envía la entidad bancaria. Este archivo seguirá una estructura que el banco ha descrito previamente en un documento y que ha sido traducida a un esquema JSON por el alumno.

4.4.1. Fase de *Staging*

Esta fase engloba el proceso desde que se reciben los datos hasta que se transforman a un formato de tabla. Consta de dos *jobs*:

- **Sabium de *Staging* - Calidad:** Este *job* se encarga de asegurar que el archivo de

datos de origen cumple ciertas reglas de calidad impuestas por la entidad bancaria, para ello se hace uso de la biblioteca de Sabium. Las reglas en esta fase son las siguientes:

- El fichero no deberá estar vacío.
- Los campos que serán las claves primarias de las tablas no deberán ser nulos.
- Los campos que serán las claves primarias de las tablas deberán tener el formato especificado.
- Los campos que serán las claves primarias de las tablas no deberán estar duplicados o, en caso de haber más de una clave, no deberá existir ninguna combinación de las mismas repetida.

- **Yoshi de *Raw* - Ingesta:** Este *job* se encarga de transformar el archivo de datos de origen a formato tabla, para ello se hace uso de la biblioteca de Yoshi.

  Por motivos de confidencialidad, no es posible mostrar las transformaciones aplicadas, pero son diferentes tipos de transformaciones a nivel campo y a nivel registro. En esta fase las transformaciones se centran en adaptar el formato de los campos, por ejemplo, que las fechas se muestren en formato ISO[30].

  Este *job* supone el paso de la fase de *Staging* a *Raw*. El resultado será una tabla en formato Avro con los datos del fichero original adaptados al estándar deseado.

4.4.2. Fase de *Raw*

Esta fase engloba el proceso desde que se tienen los datos formateados en forma de tabla hasta que se aplican las transformaciones para dejarlos en el resultado final deseado. Consta de dos *jobs*:

- **Sabium de *Raw* - Calidad:** Este *job* se encarga de asegurar que la tabla con los datos de origen formateados cumple ciertas reglas de calidad impuestas por la entidad bancaria, para ello se hace uso de la biblioteca de Sabium. Las reglas en esta fase son las siguientes:

  - El número de registros de la tabla de datos formateados deberá coincidir con el número de registros del fichero original.

- **Yoshi de *Master* - Ingesta:** Este *job* se encarga de transformar la tabla de datos formateados al formato final que requiere la entidad bancaria, para ello se hace uso de la biblioteca de Yoshi.

  Por motivos de confidencialidad, no es posible mostrar las transformaciones aplicadas, pero son diferentes tipos de transformaciones a nivel campo y a nivel registro. En esta fase las transformaciones se centran en modificar los datos según información interna del banco, por ejemplo, crear identificadores o asignar diferentes tarifas según el país.

  Este *job* supone el paso de la fase de *Raw* a *Master*. El resultado será una tabla en formato Parquet con los datos finales deseados por la entidad bancaria.

4.4.3. Fase de *Master*

Esta fase engloba el proceso una vez los datos ya se encuentran en el formato final deseado por el banco. Consta de un solo *job*:

- **Sabium de *Master* - Calidad:** Este *job* se encarga de asegurar que la tabla final cumple ciertas reglas de calidad impuestas por la entidad bancaria, para ello se hace uso de la biblioteca de Sabium. Las reglas en esta fase son las siguientes:
  - El número de registros de la tabla final deberá coincidir con el número de registros de la tabla de datos formateados.
  - Los campos que sean las claves primarias de la tabla no deberán ser nulos.
  - Los campos que sean las claves primarias de la tabla no deberán estar duplicados o, en caso de haber más de una clave, no deberá existir ninguna combinación de las mismas repetida.

Una vez acabe la ejecución de este *job* se considera que los datos cumplen los estándares de calidad y la ingesta se da por finalizada. En este punto la tabla resultante puede ser traspasada a los sistemas de la entidad bancaria y ser utilizada para los fines que deseen.

4.5. Formato de los datos

Por motivos de confidencialidad no es posible revelar la estructura exacta de los datos, pero en esta sección se detallan de forma un poco más detallada los esquemas de origen y destino de los mismos.

Ambos esquemas comparten ciertas similitudes:

- Todos los campos son de tipo String, Date, Timestamp o Decimal.
- Ambos esquemas comparten los campos “País de origen” y “Fecha de auditoría”, que son los campos que usa el banco para crear las particiones de la ingesta y así diferenciarla de otras.
- Ambos esquemas cuentan con dos campos que actúan como clave primaria, estos campos coinciden en los dos esquemas.

La principal diferencia entre los esquemas es el número de campos. El esquema de los datos de origen consta de 28 campos, mientras que el esquema final cuenta con 33 campos. Estos cinco campos nuevos se generan en el job de Yoshi de *Master* y se informan en base a la información de otros campos según criterios internos del banco.

Capítulo 5

Tecnologías y herramientas utilizadas

En este capítulo se presenta un resumen de las tecnologías utilizadas para el desarrollo del proyecto.

Para el desarrollo de la ingesta he utilizado una gran variedad de tecnologías y herramientas diferentes. Muchas de ellas son propias de la entidad bancaria y debido a ello no puedo nombrarlas por motivos de confidencialidad, en estos casos usaré un seudónimo.

5.1. Modelado de las tablas

Figura 5.1: Logo de Power Designer

Power Designer[25] es una herramienta de modelización de datos que permite una fácil visualización, análisis y diseño de bases de datos. Además, cuenta con un enfoque basado en modelos, lo que permite alinear negocio con tecnología de la información.

He utilizado esta herramienta para crear el modelado de las tablas que tenía que ingestar, detallando cada campo y sus metadatos correspondientes.

5.2. Entorno de desarrollo

Figura 5.2: Logo de IntelliJ IDEA

IntelliJ IDEA[18] es un entorno de desarrollo integrado desarrollado por JetBrains.

Este es el IDE he utilizado para el desarrollo de los programas de la ingesta y sus respectivas pruebas en local, tanto de los archivos de configuración de ingesta como de los de calidad. Además me ha permitido sincronizarlo con el siguiente sistema de control de versiones.

5.3. Control de versiones

Figura 5.3: Logo de Bitbucket

Bitbucket[9] es un servicio web de alojamiento para el control de versiones basado en Mercurial y Git. Además, no solo sirve para gestionar repositorios, sino que al tener una muy buena integración con Jira permite visualizar, comparar y corregir el código de una manera muy visual y sencilla.

Para la ingesta he utilizado esta herramienta porque es la que usamos en el equipo de trabajo de la empresa. Esto me ha permitido gestionar el desarrollo de esta ingesta en un repositorio aparte de los que uso en el día a día para organizarme mejor el trabajo y la gestión de versiones.

Figura 5.4: Logo de JFrog Artifactory

Una vez se hayan probado los archivos de configuración y calidad de la ingesta se suben mediante una *pull request* a **JFrog Artifactory**[19]. Este es otro repositorio parecido al anterior pero conectado a ambos entornos, el de *Work* y el de *Live*, lo que facilita las pruebas finales y el posterior despliegue de la ingesta de forma definitiva.

5.4. Entorno local - Programación

Figura 5.5: Logo de Java

Para el desarrollo de los archivos de configuración de ingesta y de calidad he utilizado Java Development Kit (JDK), que es una colección de software con herramientas de desarrollo para programas en **Java**[22].

Figura 5.6: Logo de Apache Maven

Para organizar el proyecto he utilizado **Apache Maven**[8], una herramienta para la construcción y posterior gestión de proyectos Java. Con esta herramienta he creado un Project Object Model (POM) que me ha permitido gestionar el proyecto que estaba desarrollando y todas sus dependencias respecto a complementos externos, como los que se nombran a continuación.

Los siguientes módulos son propios de la entidad bancaria, así que se han modificado sus nombres por cuestiones de confidencialidad.

Para los archivos de configuración que se encargan de realizar las ingestas se usa **Yoshi**, un *framework* escrito en Scala y basado en Apache Spark para el procesamiento de *sets* de datos, desde su lectura e interpretación, hasta su escritura y disponibilización. Para ello, este *framework* permite realizar operaciones sobre el dato a nivel de registro o a nivel de campo. El objetivo es la masterización de los datos, es decir, disponibilizarlos para su posterior explotación. Para ello se llevan a cabo operaciones de limpieza, armonización, validación y enriquecimiento, para asegurar su calidad y maximizar su utilidad.

Una vez los datos se han ingestado se tiene que asegurar su calidad. Para ello se utiliza **Sabium**, una librería escrita en Scala utilizada para realizar validaciones de calidad sobre las tablas de datos ingestados. Esta validación se realiza a través de una serie de reglas que la librería te da opción a personalizar.

5.5. Entorno de Work - Pruebas

A la hora de realizar las pruebas en el entorno de *Work* se usa **Dataproc**, un software en la nube que permite la ejecución de los archivos de configuración subidos a Artifactory en forma de *jobs*. Basta con proveer al *job* con los parámetros necesarios para realizar la ejecución en la nube con un solo *click*.

Figura 5.7: Logo de Jupyter Notebook

Una vez se ha ejecutado el *job* en Dataproc, el resultado puede observarse en **Jupyter Notebook**[20], una aplicación que funciona como entorno de programación basado en web. He utilizado esta herramienta para la visualización de las tablas finales y hacer las pruebas respectivas.

5.6. Entorno de Live - Automatización

Figura 5.8: Logo de Control-M

A la hora de desplegar la ingesta en el entorno de *Live* se utiliza **Control-M**[10]. Esta herramienta permite automatizar el flujo de trabajo creando cadenas de *jobs*, con la finalidad de reducir las posibilidades de fallos y agilizar la ejecución de la ingesta.

Una vez la cadena está desplegada, la aplicación facilita el seguimiento de las ejecuciones de la misma, además de ofrecer la posibilidad de una ejecución manual si algo ha ido mal.

5.7. Memoria

Figura 5.9: Logo de Overleaf

Para la redacción de este memoria he utilizado la web de **Overleaf**[23], un editor en línea de LaTeX.

Capítulo 6

Implementación y pruebas

En esta sección de detallará la implementación de la solución final del proyecto. Las fases de la implementación se citan en el orden en el que se desarrollaron.

6.1. Análisis

Aunque realmente no forma parte de la implementación, lo primero que se hizo fue una fase de análisis para organizar y comprender bien el desarrollo.

Esta fase consistió en la lectura de un extenso documento que proporcionó la entidad bancaria. En este documento se describían tanto el objetivo del proyecto con los detalles necesarios para su implementación, como el formato que tendrán los datos de origen expresado en formato de texto.

6.2. Modelado

Una vez se ha hecho el análisis hay que modelar las tablas en las que se almacenarán los datos durante la ingesta, así como el esquema de origen, puesto que el banco solo proporcionó el formato de los datos de origen en forma de texto.

6.2.1. Tablas

Para modelar las tablas se hizo uso de la herramienta de Power Designer, ubicada en un escritorio virtual proporcionado por la entidad bancaria. En esta herramienta se crea un modelo de tabla en el formato deseado y se van añadiendo, uno a uno, los campos que tendrá la tabla.

Para cada campo hay que detallar varios parámetros, como si es clave primaria, el nombre funcional, el nombre logístico, el tipo de dato, el tamaño o una breve descripción del mismo, entre otros.

Hubo que modelar un total de dos tablas: la tabla intermedia donde se almacenarán los datos de origen tras la ingesta de *Raw*, y la tabla donde se almacenarán los datos finales tras la ingesta de *Master*.

La primera tabla, a la que llamaremos tabla de *Raw*, contaba con un total de 28 campos, dos de los cuales actuaban como clave primaria.

Para la segunda tabla, la tabla de *Master*, es una ampliación de la tabla de *Raw*. Cuenta con todos los campos de la primera pero con el añadido de cinco campos nuevos generados en el proceso de la ingesta.

6.2.2. *Namings*

En el apartado anterior se ha mencionado que a la hora de modelar un campo hay que detallar dos nombres diferentes: el funcional y el logístico. Esto se debe a que, como los datos se van a incorporar al sistema del banco una vez se realice la ingesta, estos deberán seguir el mismo convenio que el resto de sus bases de datos.

El nombre funcional es el nombre que tendrá el campo de cara a la programación de la ingesta, mientras que el nombre logístico es el nombre que el banco ha elegido de cara a integrar los datos en su sistema. Estos nombres logísticos venían listados en el documento que proporcionó la entidad bancaria, excepto para los cinco campos nuevos.

Para elegir estos nombres hubo que contactar con el banco y registrarlos en su sistema mediante *Namings*. Para registrar un *Naming* hay que asegurarse de que es único y que no dé lugar a confusión con otro ya existente. Tras registrarlos, la entidad bancaria comunicó que uno de ellos daba lugar a confusión porque se parecía demasiado a otro que ya existía, por lo que hubo que cambiarlo.

6.2.3. Esquema de origen

A la hora de ejecutar los futuros *jobs* de la ingesta es necesario un esquema en formato JSON que indique la estructura de los datos, tanto para los de origen como para los finales. Si bien el banco proporcionó el esquema para los datos finales, para los datos de origen solo estaba descrito en forma de texto en el documento inicial.

Para escribir el esquema JSON hubo que imitar el formato del esquema final y adaptarlo a los datos de origen. En estos esquemas se detallan para cada campo diferentes atributos como si son clave primaria, su nombre funcional, su tipo, su tamaño o su formato.

6.3. Entorno local

Una vez se tienen las tablas y los esquemas modelados, se pasa a trabajar en la programación de los archivos de configuración para los *jobs* y sus respectivas pruebas en el entorno local.

6.3.1. Archivos de configuración de ingesta - Yoshi

Para programar los archivos de configuración de los *jobs* de ingesta se hizo uso de la biblioteca de Yoshi, propia del banco. Por motivos de confidencialidad no es posible mostrar en este documento los códigos de forma explícita, pero a continuación se describe de una forma general su contenido.

- **Input:** En esta parte se detallan campos como los siguientes:
  - La ruta de los datos de entrada a ingestar.
  - El tipo de los datos de entrada: CSV y Avro, respectivamente.
  - El esquema JSON con la estructura de los datos de entrada.
- **Transformaciones:** En esta parte se detallan las transformaciones que se deberán llevar a cabo sobre los datos, tanto a nivel campo como a nivel registro. Algunos ejemplos pueden ser los siguientes:
  - Eliminar los espacios de los campos.
  - Formatear las fechas con formato ISO[30].
  - Informar un campo con la fecha y hora exactas de la realización de la ingesta.
  - Concatenar la información de uno o varios campos y guardar el resultado en otro.
  - Generar identificadores únicos en base a la información de los campos de un registro.
- **Output:** En esta parte se detallan campos como los siguientes:
  - La ruta donde se depositarán los datos de salida.
  - El tipo de fichero de salida: Avro y Parquet, respectivamente.
  - El esquema JSON con la estructura de los datos de salida.
  - Qué hacer con los registros corruptos o mal informados, si introducir nulos o eliminarlos completamente.
  - Si tienen que existir particiones, y si es así, por qué campos se han de particionar las tablas.
  - El modo de ingesta: este puede ser *append*, para agregar información a una tabla ya existente; o puede ser *reprocess*, que reemplaza completamente los datos que hubiera en la ruta de salida.

6.3.2. Archivos de configuración de calidad - Sabium

Para programar los archivos de configuración de los *jobs* de calidad se hizo uso de la biblioteca de Sabium, propia del banco. Por motivos de confidencialidad no es posible mostrar en este documento los códigos de forma explícita, pero a continuación se describe de una forma general su contenido.

El resultado de la ejecución de los archivos de configuración de ingesta será una tabla con los datos tras aplicar las transformaciones respectivas en el formato deseado.

- **Input:** En esta parte se detallan campos como los siguientes:
  - La ruta de los datos a comprobar.
  - El tipo de los datos a comprobar: CSV, Avro y Parquet, respectivamente.
  - El esquema JSON con la estructura de los datos a comprobar.
- **Reglas:** En esta parte se detallan las reglas de calidad que han de cumplir los datos. Estas reglas de calidad vienen impuestas por la entidad bancaria y son diferentes dependiendo de la fase en la que se encuentren los datos. Algunas de las reglas de calidad son las siguientes:
  - En *Staging*: El fichero no deberá estar vacío y, las claves primarias, deberán tener el formato especificado en la regla mediante una expresión regular y no deberán ser nulas ni deberán estar duplicadas.
  - En *Raw*: El número de registros de la tabla intermedia deberá coincidir con el número de registros del fichero original.
  - En *Master*: El número de registros de la tabla final deberá coincidir con el número de registros de la tabla intermedia y las claves primarias no deberán ser nulas ni deberán estar duplicadas.

El resultado de la ejecución de los archivos de configuración de calidad será un *log* en el que se mostrará qué reglas se cumplen y, si no lo hacen, por cuánto porcentaje están fallando.

6.3.3. Pruebas

Una vez se han programado los archivos de configuración hay que comprobar que funcionan de la forma esperada. Para ello es necesario simular en el IDE local la ejecución de lo que serán los futuros *jobs*. Esto es posible gracias a las librerías proporcionadas por el banco, puesto que ambas contienen una clase programada específicamente para realizar estas pruebas en un entorno local.

Pero para realizar estas pruebas primero son necesarios unos datos. Como no se tienen datos reales se optó por crear un archivo CSV *dummy*, es decir, un fichero con datos aleatorios que siga el esquema de los datos reales y que bastará para realizar estas pruebas.

Para probar los archivos de configuración de ingesta basta con ejecutarlos y comprobar que se crean las tablas de salida y que el resultado sea el esperado teniendo en cuenta el archivo de origen.

Por otra parte, para probar los de calidad no basta con comprobar que el archivo CSV y las dos tablas cumplan las reglas; también hay que probar casos en los que las reglas no se cumplan, para comprobar que el funcionamiento de las reglas sea correcto.

6.4. Entorno de Work

Ya terminada la programación y comprobado que todos los archivos de configuración funcionan de la forma en la que deberían, es hora de llevar el desarrollo al entorno de *Work*.

6.4.1. Creación y configuración de los *jobs*

Antes de crear los *jobs* primero hubo que subir los archivos de configuración desarrollados al repositorio del banco de JFrog Artifactory y pedir que aprobasen la *pull request*.

Después, se procedió a crear los cinco *jobs* necesarios en la plataforma de Dataproc de *Work* del banco. Estos jobs deberán tener un nombre que siguiera la nomenclatura propuesta por la entidad bancaria en el documento inicial.

Para configurar un *job* hay que indicar la ruta de JFrog Artifactory del archivo de configuración que corresponda y la versión de Yoshi o Sabium utilizada en el mismo. También hay que detallar las variables de entorno que se hayan utilizado en los archivos de configuración y los valores que debieran tomar a la hora de ejecutar el *job*.

6.4.2. Pruebas

Antes de empezar a probar nada, había que subir al repositorio el archivo CSV *dummy* que se creó previamente. Para hacer esto no hizo falta pedir otra *pull request*, esto se debe a que ese archivo no va a formar parte del entorno del banco, ya que solo se requería para hacer las pruebas.

Para realizar las pruebas en el entorno de *Work* bastaba con ejecutar cada uno de los *jobs* en orden y comprobar que la ejecución no hubiera fallado. Aún así, se realizaron comprobaciones adicionales para asegurar el correcto funcionamiento de los *jobs* en la plataforma antes de crearlos en *Live*.

Para comprobar los *jobs* de ingesta se utilizó la herramienta de Jupyter. Aquí se puede realizar una visualización de las tablas ingestadas y se pueden programar pequeños trozos de código para comprobar ciertas cosas como el número de registros o que no existan duplicados, y así no tener que hacer la comprobación de forma manual.

Para comprobar los *jobs* de calidad fue más sencillo, puesto que se pueden leer los *logs* con el reporte de reglas y así comprobar que se cumplan en un 100 %.

6.5. Entorno de Live

Cuando se ha comprobado que los *jobs* funcionan de manera correcta en la plataforma de Dataproc, ya se puede pasar al entorno de *Live*. Es aquí donde ejecutarán los *jobs* definitivos y donde la entidad bancaria hará sus ingestas mensuales.

En esta fase es muy importante comprobar que toda la implementación coincide exactamente con la que ya ha sido probada en el entorno de *Work*, puesto que si algo no funciona como debería podría ocasionarle problemas a terceras personas que dependan de los datos que se exportan en esta ingesta. Si se hubiera detectado un fallo o hubiese habido que realizar algún cambio, habría que haber vuelto a una fase anterior.

6.5.1. Creación y configuración de los *jobs*

De la misma forma que en el entorno de *Work*, se crearon y configuraron los mismos *jobs* con exactamente los mismos parámetros, pero esta vez en la plataforma de Dataproc en *Live*.

En el entorno de Live no se realizan pruebas, ya que no se pueden hacer ejecuciones puntuales de *jobs*. Esto se debe a dos motivos principales: que las ejecuciones puntuales gastarían recursos para probar unos *jobs* que ya se han probado anteriormente y que el planificador de ejecuciones del banco lleva un orden estricto con muchas otras mallas e ingestas, por lo que una ejecución no planificada alteraría ese orden de forma innecesaria.

6.5.2. Creación de la malla automatizada

Una vez los *jobs* estaban creados, había que relacionarlos entre sí y añadir la automatización. Esto se logró mediante la herramienta Control-M.

En esta herramienta se creó una malla, una cadena ordenada, formada por los 5 jobs de *Live* de Dataproc. A esta malla se le asignó un día de ejecución y una periodicidad, siendo en este caso el día 15 de cada mes si es un día hábil, o el día siguiente más próximo en caso de que no lo sea. Esto se hace para que, en caso de que la malla falle por el motivo que sea, haya personal del banco trabajando que pueda revisarla.

Con esto queda finalizada la implementación de la ingesta.

6.6. Documentación

Si bien en este punto la implementación está finalizada, hubo que escribir ciertos documentos informativos para enviar a la entidad bancaria. De esta forma si de cara al futuro necesitan conocer algún detalle sobre la implementación del proyecto puedan consultarlo.

6.6.1. Gestor documental

Consiste en un documento de Excel con información sobre cada uno de los *jobs* de *Live* como su nombre, qué día ejecutan, las transformaciones o reglas que aplican, o las rutas a los archivos de configuración en el repositorio, entre otros.

6.6.2. Traspaso

Es un documento escrito complementario al anterior en el que se describe de forma general el motivo del proyecto y la solución implementada. Cuenta también con una lista de requisitos mínimos que tiene que cumplir la ingesta para considerarse como terminada.

6.7. Posible *filewatcher*

Si bien el proyecto está completo y actualmente se encuentra funcionando correctamente, sí que existe una posible mejora para el mismo de cara al futuro.

Ahora mismo la ingesta ejecuta a mitad de mes con los archivos que el banco haya depositado en la ruta de origen. Esto puede causar ciertos problemas ya que, aunque la entrega de los archivos se supone que es una vez al mes, a la hora de la verdad pueden existir retrasos por el motivo que sea. Esto supone que si el día 15 no se han depositado los datos y se depositan más tarde, un mes no se ingeste nada y al mes siguiente se ingeste el doble de información de lo habitual.

Esto no debería ser un problema muy grande ya que las particiones se hacen con la fecha de los datos de origen y se separarían de forma correcta, pero tardarían más en estar disponibles y una carga de trabajo mayor de lo esperada puede causar retrasos inesperados en el planificador de ejecuciones del banco.

Una posible solución a este problema sería introducir un *filewatcher*. Un *filewatcher* es un *job* predecesor a la malla que se ejecutaría de forma más frecuente, por ejemplo diariamente, y que lo único que haría sería comprobar si existe un fichero nuevo en la ruta de origen, y si lo hay, que diera paso a la ejecución de la malla entera. Esto podría hacerse mediante un pequeño *script* con el comando "ls" dependiendo de cómo esté implementado el sistema, y solucionaría este problema a cambio de un coste muy bajo en el planificador de ejecuciones de la entidad bancaria.

Por último destacar que esta es una solución abordable únicamente de forma teórica, puesto que la respuesta del banco ha sido negativa debido a que se salía de los objetivos del proyecto y se necesitaría un estudio previo para determinar la viabilidad del plan.

Capítulo 7

Seguimiento del proyecto

7.1. *Sprint* 1: Análisis y modelado

- **Duración:** Del 12 de diciembre de 2022 al 26 de diciembre de 2022.
- **Objetivos:** Realizar el análisis del proyecto, definir una planificación para llevarlo a cabo, modelar las tablas y el esquema de origen y registrar los *namings* en el sistema del banco.
- **Actividades realizadas:**
  - Lectura comprensiva del documento de la entidad bancaria en el que se describía la tarea a realizar, así como el formato de los datos de origen.
  - Contacto con la persona responsable del banco para resolver las dudas que hubieran surgido.
  - Modelado a mano de las tablas intermedia y final en la herramienta Power Designer ubicada en un escritorio virtual proporcionado por el banco.
  - Modelado a mano del esquema JSON de origen de los datos según la información recaudada en el análisis del documento.
  - Registro de los *namings* nuevos necesarios en el sistema del banco.
- **Dificultades:**
  - El escritorio virtual no estuvo disponible durante algunos días, por lo que no era posible acceder a la herramienta de Power Designer.
  - A la hora de registrar los *namings* se cometió un error en la descripción de uno de ellos y hubo que contactar con la entidad bancaria para hacerlo de nuevo.

7.2. *Sprint* 2: Desarrollo y pruebas en local

- **Duración:** Del 26 de diciembre de 2022 al 9 de enero de 2023.
- **Objetivos:** Crear un repositorio para guardar el proyecto, programar los archivos de configuración para los *jobs* de ingesta y calidad, crear un archivo CSV *dummy* y empezar las pruebas en local.
- **Actividades realizadas:**
  - Instalación del IDE de Intellij.
  - Creación de un repositorio en Bitbucket y enlazarlo al IDE.
  - Instalación de las bibliotecas de Yoshi y Sabium.
  - Programación de los archivos de configuración de ingesta de *Raw* y *Master* mediante la biblioteca de Yoshi.
  - Programación de los archivos de configuración de calidad de *Staging*, *Raw* y *Master* mediante la biblioteca de Yoshi.
  - Creación de un archivo CSV *dummy* aleatorio pero que siga el esquema de origen para las posteriores pruebas en local.
  - Pruebas de los archivos de configuración de ingesta de *Raw* y *Master* y revisión de las tablas ingestadas.
- **Dificultades:**
  - Hubo muchos problemas a la hora de instalar las bibliotecas del banco por incompatibilidad de versiones, esto supuso un par de días enteros de retraso de las tareas.

7.3. *Sprint* 3: Creación de los *jobs* y pruebas en *Work*

- **Duración:** Del 9 de enero de 2023 al 23 de enero de 2023.
- **Objetivos:** Terminar las pruebas en local, subir los códigos a JFrog Artifactory, crear los jobs en *Work* y realizar las pruebas en *Work*.
- **Actividades realizadas:**
  - Pruebas de los archivos de configuración de calidad de *Staging*, *Raw* y *Master*; tanto positivas como negativas.
  - Subida de los códigos a la plataforma de la entidad bancaria de JFrog Artifactory.
  - Pedir a la entidad bancaria la aprobación de la *Pull Request* de la subida de los archivos.
  - Creación de los *jobs* y sus parámetros en el entorno de *Work* de la plataforma de Dataproc.

  - Vinculación de cada *job* con su archivo de configuración de JFrog Artifactory correspondiente.
  - Comienzo de las pruebas de funcionamiento de los *jobs* en la plataforma de Dataproc en el entorno de *Work*.

- **Dificultades:**

  - Debido al retraso sufrido en el *sprint* anterior no se pudieron completar las pruebas en *Work* en su totalidad.

7.4. *Sprint* 4: Creación de la malla en *Live*, documentación e inicio de la memoria

- **Duración:** Del 23 de enero de 2023 al 6 de febrero de 2023.

- **Objetivos:** Terminar las pruebas en *Work*, crear los *jobs* en *Live*, crear la malla automatizada en Control-M, escribir la documentación para el banco, contactar con el tutor y empezar con la memoria.

- **Actividades realizadas:**

  - Finalización de las pruebas de funcionamiento de los *jobs* en la plataforma de Dataproc en el entorno de *Work*.
  - Comprobación exhaustiva de las tablas ingestadas mediante la ayuda de Jupyter.
  - Creación de los *jobs* y sus parámetros en el entorno de *Live* de la plataforma de Dataproc.
  - Vinculación de cada *job* con su archivo de configuración de JFrog Artifactory correspondiente.
  - Creación de la malla automatizada y sus parámetros en la plataforma de Control-M.
  - Vinculación de cada *job* de la malla con su *job* de Dataproc en *Live* correspondiente.
  - Escritura y envío de la documentación de la ingesta que requerida por parte de la entidad bancaria.
  - Contacto con el tutor del TFG para concertar una reunión de revisión del desarrollo del proyecto.
  - Comienzo de la escritura de la memoria, organización de la estructura de la misma.

- **Dificultades:**

  - No hubo dificultades destacables en este *sprint*.

7.5. *Sprint* 5: Memoria

- **Duración:** Del 6 de febrero de 2023 al 20 de febrero de 2023.
- **Objetivos:** Tener una reunión de revisión del desarrollo del proyecto con el tutor del TFG y acabar la redacción de la memoria.
- **Actividades realizadas:**
  - Reunión con el tutor del TFG para revisar la calidad del desarrollo del proyecto y la estructura de la memoria previamente organizada.
  - Redacción de la memoria.
- **Dificultades:**
  - La redacción de la memoria llevó más tiempo del esperado, por lo que se necesitaría un tiempo extra de entre una y dos semanas para la finalización de la misma.

7.6. *Sprint* extra: Memoria

- **Duración:** Del 20 de febrero de 2023 al 27 de febrero de 2023..
- **Objetivos:** Finalizar la redacción de la memoria de cara a presentar el TFG.
- **Actividades realizadas:**
  - Redacción de los capítulos faltantes de la memoria.
  - Corrección de las secciones de la memoria que lo requieran.
- **Dificultades:**
  - No hubo dificultades destacables en este *sprint*.

7.7. Comparación entre estimación y realidad

En el proceso de planificación se hizo una estimación de lo que duraría cada una de las fases en las que se dividió el proyecto, sin embargo, debido a las complicaciones, esta estimación acabó variando de lo que ocurrió en realidad.

En la siguiente tabla se puede observar la diferencia entre los tiempos estimados y los tiempos que realmente llevó cada una de las fases del proyecto:

Fase	Inicio estimado	Inicio real	Finalización estimada	Finalización real
Análisis	12/12/2022	12/12/2022	26/12/2022	26/12/2022
Local	26/12/2022	26/12/2022	09/01/2023	12/01/2023
Work	09/01/2023	12/01/2023	23/01/202	27/01/2023
Live	23/01/2023	27/01/2023	06/02/2023	08/02/2023
Memoria	06/02/2023	08/02/2023	20/02/2023	27/02/2023

Tabla 7.1: Comparación entre los tiempos estimados y reales

Capítulo 8

Conclusiones

Una vez finalizado el proyecto es hora de mirar atrás y observar el proceso de desarrollo para comprobar si se han cumplido los objetivos propuestos.

En este proyecto he podido comprobar la gran importancia que tiene el *Big Data* hoy en día, así como la manera en la que se trata, creando una ingesta automatizada que facilite el trabajo a la entidad bancaria cliente. También he aprendido sobre las herramientas, técnicas y tecnologías que más se están utilizando a día de hoy para llevar a cabo estas tareas.

Al haber acabado el proyecto se puede asegurar que se han cumplido todos los objetivos planteados en la planificación inicial. El proyecto se ha implantado y se encuentra en correcto funcionamiento, se han cumplido las expectativas de la entidad bancaria cliente y ha sido una oportunidad para adquirir conocimientos y desarrollar una solución de *Big Data*. Por lo tanto, se podría considerar que ha sido un éxito tanto a nivel académico como personal.

Este ha sido un proyecto que me ha ilusionado desde un principio, puesto que me interesaba mucho el ámbito del *Big Data*. Aunque también es cierto que una vez empecé a darme cuenta de la cantidad de tecnologías y herramientas completamente nuevas de las que tenía que informarme, sí que me asusté un poco, pero nada que el trabajo e investigación durante el día a día no arreglen.

He aprendido cómo funciona el tratamiento de datos, qué tecnologías de *Big Data* existen y cómo funcionan, cómo funciona un entorno de trabajo de verdad y es el primer proyecto que realizo que tiene una utilidad real fuera de la universidad.

Añadir también que, aunque el proyecto haya finalizado, me quedo con la sensación de conocer solo una ínfima parte de todo lo que rodea a este campo, y que me hace pensar que solo ha sido el principio de una carrera profesional que apenas comienza.

Por último quiero destacar la ayuda que me han prestado mi tutor de la empresa y el tutor de la universidad. Ellos han conseguido que esto salga adelante resolviendo las dudas que me han ido surgiendo a lo largo del proyecto, y han conseguido orientarlo para que se convierta en este trabajo de fin de grado, que pone fin a mi paso por la universidad.

Bibliografía

[1] Aitor Medrano. Inteligencia artificial y big data. analítica de datos. spark. https://aitor-medrano.github.io/bigdata2122/apuntes/spark01rdd.html. (Febrero 2023).

[2] Aitor Medrano. Inteligencia artificial y big data. arquitecturas big data. cloud computing. https://aitor-medrano.github.io/bigdata2122/apuntes/nube01.html. (Febrero 2023).

[3] Aitor Medrano. Inteligencia artificial y big data. big data aplicado. hadoop. https://aitor-medrano.github.io/bigdata2122/apuntes/bdaplicado01hadoop.html#componentes-y-ecosistema. (Febrero 2023).

[4] Aitor Medrano. Inteligencia artificial y big data. ingesta de datos. etl. https://aitor-medrano.github.io/bigdata2122/apuntes/ingesta01.html. (Febrero 2023).

[5] Amazon. Kensington pro fit. https://www.amazon.es/Kensington-Pro-Fit-ptico-mediano/dp/B004S7R6MQ?th=1. (Febrero 2023).

[6] Apache Hadoop. Apache hadoop. https://hadoop.apache.org. (Febrero 2023).

[7] Apache Hadoop. Hdfs architecture guide. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. (Febrero 2023).

[8] Apache Maven. Welcome to apache maven. https://maven.apache.org. (Febrero 2023).

[9] Bitbucket. Code & ci/cd, built for teams using jira. https://bitbucket.org. (Febrero 2023).

[10] BMC. Control-m. https://www.bmcsoftware.es/it-solutions/control-m.html. (Febrero 2023).

[11] Camargo-Vega, Juan José, Camargo-Ortega, Jonathan Felipe, y Joyanes-Aguilar, Luis. Conociendo big data. revista facultad de ingeniería. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-11292015000100006&lng=en&tlng=es. (2015).

[12] Confluence. Hadoop project description. https://cwiki.apache.org/confluence/display/HADOOP2/ProjectDescription. (Febrero 2023).

[13] DATAPRIX. Consideraciones procesos etl en entornos big data: Caso hadoop. https://www.dataprix.com/es/blog-it/juanvidal/consideraciones-procesos-etl-entornos-big-data-caso-hadoop. (Febrero 2023).

[14] Dell. Latitude 5420. https://www.dell.com/es-es/shop/porttiles-dell/porttil-empresarial-latitude-5420/spd/latitude-5420-laptop/gxctol542014emea?redirectTo=SOC. (Febrero 2023).

[15] Findstack. La lista definitiva de estadísticas de big data para 2023. https://findstack.es/resources/big-data-statistics/. (Febrero 2023).

[16] Geeks for geeks. Hadoop – architecture. https://www.geeksforgeeks.org/hadoop-architecture/. (Febrero 2023).

[17] Giusti, A. E., Naiouf, M., Sanz, C. V., Tinetti, F. G., De Giusti, L. C., y Bertone, R. A. Procesamiento paralelo y distribuido en tratamiento masivo de datos. in iv workshop de investigadores en ciencias de la computación. (2002).

[18] JetBrains. Intellij idea – the leading java and kotlin ider. https://www.jetbrains.com/idea/. (Febrero 2023).

[19] JFrog. Jfrog artifactory. https://jfrog.com/artifactory/. (Febrero 2023).

[20] Jupyter. Jupyter. https://jupyter.org. (Febrero 2023).

[21] Mike Cotterell y Bob Hughes. Software project management. 5.a ed. tata mcgraw hill education private limited. (2011, ISBN: 978-0-07-107274-8.).

[22] Oracle. Java. https://www.oracle.com/es/java/. (Febrero 2023).

[23] Overleaf. Latex, evolucionado. https://es.overleaf.com. (Febrero 2023).

[24] PcComponentes. Viewsonic vx series vx2458-mhd. https://www.pccomponentes.com/viewsonic-vx-series-vx2458-mhd-236-led-fullhd-144hz-freesync. (Febrero 2023).

[25] Power Designer. Funcionalidades principales de powerdesigner. https://www.powerdesigner.biz/ES/powerdesigner/powerdesigner-features.html. (Febrero 2023).

[26] PowerData. Big data: ¿en qué consiste? su importancia, desafíos y gobernabilidad. https://www.powerdata.es/big-data. (Febrero 2023).

[27] Sergi Monroy. ¿cuáles son los roles de la metodología scrum? https://www.apd.es/roles-metodologia-scrum/. (Febrero 2023).

[28] StreamSets. Data ingestion: Tools, types, and key concepts. https://streamsets.com/learn/data-ingestion/. (Febrero 2023).

[29] Talent. Salario medio para ingeniero informatico en españa, 2023. https://es.talent.com/salary?job=ingeniero+informatico. (Febrero 2023).

[30] W3C. Use el formato de fecha internacional (iso). `https://www.w3.org/QA/Tips/iso-date.html.es`. (Febrero 2023).

[31] Wikipedia. Macrodatos. `https://es.wikipedia.org/wiki/Macrodatos#Caractersticas`. (Febrero 2023).

[32] Wing, J. M. The data life cycle. harvard data science review. (2019).