ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

Desarrollo e implantación de un proceso ETL para la recomendación de vehículos

Autora: Elena Cabrera Casquet

Director: Domingo Guinea García-Alegre

Madrid

Enero 2025

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
MÁSTER EN BIG DATA. TECNOLOGÍA Y ANALÍTICA AVANZADA

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Desarrollo e implantación de un proceso ETL para la recomendación de vehículos en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Elena Cabrera Casquet Fecha: 07/01/2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Domingo Guinea García-Alegre Fecha: 07/01/2025

Vº Bº del Coordinador de Proyectos

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: ……/ ……/ ……

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor Dª. Elena Cabrera Casquet DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Desarrollo e implantación de un proceso ETL para la recomendación de vehículos, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

- El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 7 de enero de 2025

ACEPTA

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

Desarrollo e implantación de un proceso ETL para la recomendación de vehículos

Autora: Elena Cabrera Casquet

Director: Domingo Guinea García-Alegre

Madrid

Enero 2025

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
MÁSTER EN BIG DATA. TECNOLOGÍA Y ANALÍTICA AVANZADA

DESARROLLO E IMPLANTACIÓN DE UN PROCESO ETL PARA LA RECOMENDACIÓN DE VEHÍCULOS

Autor: Cabrera Casquet, Elena.
Director: Guinea García-Alegre, Domingo.
Entidad Colaboradora: Ozone Drive S.L.

RESUMEN DEL PROYECTO

Este proyecto se centra en el desarrollo y la implantación de un proceso ETL (Extract, Transform, Load) utilizando tecnologías como Apache Airflow, Docker y Python para la orquestación y lanzamiento de procesos de Big Data de forma automatizada. El proyecto abarca desde la toma de requerimientos y la definición del modelo de datos y sistema de almacenamiento, hasta la puesta en marcha del proceso completo y la validación de datos.

Palabras clave: ETL, Airflow, Big Data, Arquitectura

Índice de la memoria

Capítulo 1. Introducción 5

1.1 Motivación del proyecto 5

1.2 Descripción de las tecnologías 5

1.2.1 Procesos ETL 5

1.2.2 Apache Airflow 6

1.2.3 Docker 8

1.2.4 Amazon S3 9

Capítulo 2. Definición del Trabajo 12

2.1 Objetivos 12

2.1.1 Diseño y desarrollo de un pipeline automatizado de ejecución de procesos 12

2.1.2 Planificación y periodicidad de las ejecuciones 12

2.1.3 Supervisión y lanzamiento manual de la ETL en interfaz web 13

2.1.4 Relanzamiento automático de las tareas en caso de error 13

2.1.5 Mecanismos de validación 13

2.1.6 Generación de documentación y una guía de uso 14

2.2 Metodología 14

2.3 Planificación y Estimación Económica 15

2.3.1 Supuestos y datos de referencia 15

2.3.2 Etapas del proyecto, estimación de tiempos y costes 15

Capítulo 3. Desarrollo del proyecto 17

3.1 Análisis de los procesos actuales y futuros 17

3.1.1 Análisis "AS-IS" 17

3.1.2 Análisis gap 18

3.2 Contratación del servidor remoto 21

3.2.1 Selección del proveedor de servicios cloud 21

3.2.2 Especificaciones del servidor 22

3.3 Configuración del servidor 22

3.3.1 Instalación de Pyenv como gestor de entornos y versiones de Python 22

3.3.2 Repositorios 23

3.3.3 Instalación de Docker e imagen de Apache Airflow 23

3.3.4 Submódulos *25*
3.4 Script de validación 25
3.5 Diseño del modelo de datos en Amazon S3 26
3.6 Script de carga de ficheros en S3 29
3.7 Actualización de la Base de Datos 30
3.7.1 Configuración de Liquibase *30*
3.7.2 Proceso de carga *30*
3.8 DAG de Airflow 30
3.8.1 Configuración de parámetros globales *31*
3.8.2 Tareas *31*
3.8.3 Definición del pipeline *32*

Capítulo 4. Análisis de Resultados 34

Capítulo 5. Conclusiones y Trabajos Futuros 35
5.1 Conclusiones 35
5.1.1 Eficiencia y automatización de los procesos *35*
5.1.2 Interfaz web accesible y formaciones *35*
5.1.3 Almacenamiento en Amazon S3 ..... *36*
5.1.4 Calidad del dato *36*
5.1.5 Impacto en el negocio *36*
5.1.6 Lecciones aprendidas *36*

Capítulo 6. Bibliografía 38

Índice de figuras

Figura 1. Arquitectura de Airflow [5] 7
Figura 2. Diagrama ejemplo de un DAG sencillo en Apache Airflow..... 8
Figura 3. Diagrama del TO-BE del proceso ETL completo..... 20
Figura 4. Estructura de directorios del proyecto..... 24
Figura 5. Modelo de almacenamiento de ficheros en Amazon S3 28
Figura 6. Gráfico del DAG de Airflow..... 33
Figura 7. Interfaz web de Airflow 34

Índice de tablas

Tabla 1. Estimación de tiempos y costes de cada etapa del proyecto. 16

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

La empresa colaboradora, Ozone Drive, se encuentra en un proceso de crecimiento y, como parte de esta evolución, surge la necesidad de desarrollar un flujo de datos robusto dentro de la compañía, que pueda escalar junto con la propia organización y garantizar la disponibilidad del dato en todo momento.

Ozone Drive necesita crear un *pipeline* estable y automatizado mediante el que se recopilen las características de los últimos vehículos del mercado (así como otros *inputs* necesarios) y se procesen para acabar alimentando su base de datos, que es el centro de todos sus productos y de las recomendaciones de vehículos, su negocio principal.

Por un lado, la implementación de un proceso ETL, permite a la compañía dejar de depender de recursos individuales y locales (ordenadores de empleados individuales) y trasladar las ejecuciones de los procesos a un servidor remoto. Además, posibilita la calendarización de las actualizaciones de la base de datos y la implantación de controles de calidad y validación de los datos, clave para garantizar la calidad del dato, que es el núcleo del negocio. Finalmente, se establece la aspiración de tener un *pipeline* que pueda ser supervisado por cualquier miembro del equipo, de forma que, si el equipo de Ingeniería de Datos de la empresa no estuviese disponible, pueda tenerse visibilidad del estado de los procesos lanzados.

1.2 DESCRIPCIÓN DE LAS TECNOLOGÍAS

1.2.1 PROCESOS ETL

El Big Data se ha convertido en un recurso esencial para las empresas que buscan obtener ventajas competitivas a través de la información que contienen los datos [1]. El término Big Data se refiere a los datos que son demasiado voluminosos y complejos de capturar, procesar

y analizar para la infraestructura tecnológica tradicional [1]. En este contexto, es de esperar que se complejicen los procesos que llevan a cabo las compañías para poder obtener los datos, asegurar su calidad, transformarlos y almacenarlos acorde a las necesidades del negocio. Por lo tanto, resulta fundamental definir el proceso que abarca todo el ciclo de vida del dato previo a su uso, desde su obtención, hasta su carga, pasando por el procesamiento del dato. Este tipo de procesos se llaman procesos ETL, por sus siglas en inglés: *Extract, Transform, Load*.

La fase de extracción es aquella en la que se recopilan datos de distintas fuentes, que pueden incluir bases de datos, archivos en cualquier formato o páginas web. La siguiente fase es la de transformación, en la que los datos extraídos se limpian, transforman e integran para garantizar que coinciden con el esquema que espera el almacén de datos en el que finalmente se almacenarán [2]. La última fase es la de carga, en la que se insertan los datos en el DataWarehouse con el formato esperado y desde donde se alimentarán otros procesos de inteligencia de negocio o análisis predictivo [3].

1.2.2 APACHE AIRFLOW

Entre las herramientas existentes que permiten la orquestación de flujos de datos, Apache Airflow, es una de las más utilizadas y fáciles de implementar. Apache Airflow es una herramienta open-source que permite desarrollar y planificar el lanzamiento de procesos mediante código Python. Además, despliega una interfaz web a través de la que los usuarios autorizados pueden visualizar los procesos activos, así como el historial de ejecuciones, logs que han generado los procesos, tiempos que éstos han tomado, etcétera [4].

1.2.2.1 Arquitectura de Airflow

La arquitectura de Airflow consiste en varios componentes necesarios para la ejecución de la herramienta, que deben ser instalados para que ésta funcione correctamente [4].

- **Scheduler**: es el componente encargado de lanzar los procesos planificados y de enviar las tareas al executor para su ejecución. El executor es un sub-componente del

scheduler, que puede ser configurado manualmente o elegir uno de los que vienen ya configurados (LocalExecutor, CeleryExecutor y KubernetesExecutor).

- **Webserver**: Airflow viene con un servidor web que proporciona una interfaz de usuario, permitiendo a éstos monitorizar los flujos de trabajo y administrar las tareas (forzar su ejecución, cambiar la planificación, etcétera).
- **Carpeta de DAGs**: es necesario crear, si no es creada automáticamente durante la instalación de Airflow, un directorio que contenga los ficheros DAG, que son aquellos con las instrucciones de las tareas a ejecutar y la planificación de los procesos en código Python.
- **Metadata database**: se debe instalar una base de datos en la que los componentes de Airflow almacenan el estado de los procesos y las tareas, y que el servidor web utiliza para poder mostrarlo al usuario. No viene configurada por defecto, así que el administrador debe ser el encargado de ponerla en funcionamiento.

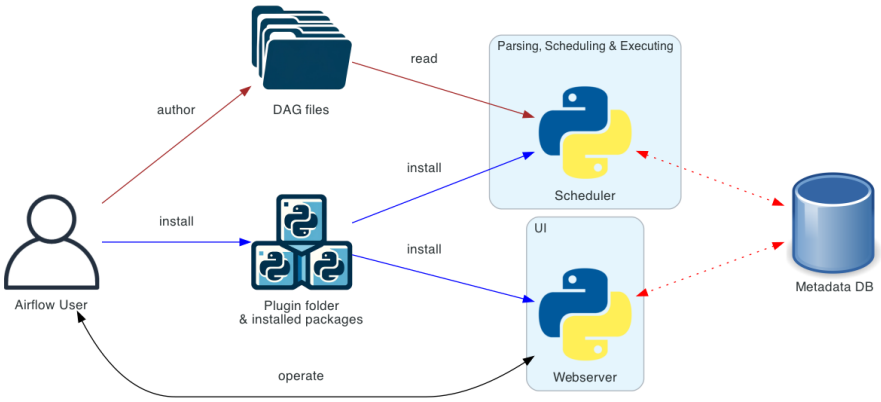

Figura 1. Arquitectura de Airflow [5]

1.2.2.2 Conceptos básicos de Airflow

El concepto principal de Airflow y a partir del cual se desarrollan todas las funcionalidades es el Grafo Acíclico Dirigido, DAG por sus siglas en inglés. Un DAG es el código Python que define un flujo de trabajo, mediante una serie de tareas (*tasks*) que se ejecutan siguiendo una secuencia específica y con dependencias entre ellas, es decir, se define el orden y las condiciones de ejecución de cada tarea. Además, cuando se define un

DAG se definen otras características como su nombre y la periodicidad con la que se va a ejecutar.

Por ejemplo, un DAG sencillo que consta de cuatro tareas y que se ejecuta una vez al día se programaría de la siguiente manera:

```
import datetime
from airflow import DAG
from airflow.operators.empty import EmptyOperator
my_dag = DAG(
    dag_id="my_dag_name",
    start_date=datetime.datetime(2021, 1, 1),
    schedule_interval="@daily",
)
# Definición de las tareas
first_task = EmptyOperator(task_id="first_task", dag=my_dag)
second_task = EmptyOperator(task_id="second_task", dag=my_dag)
third_task = EmptyOperator(task_id="third_task", dag=my_dag)
fourth_task = EmptyOperator(task_id="fourth_task", dag=my_dag)
# Definición de las dependencias
first_task >> [second_task, third_task]
fourth_task >> third_task
```

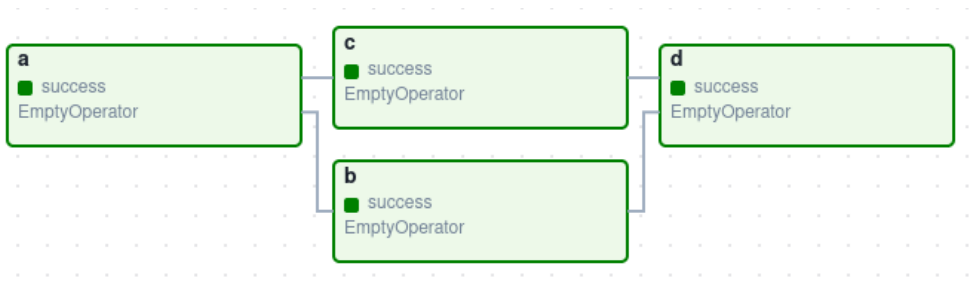

Figura 2. Diagrama ejemplo de un DAG sencillo en Apache Airflow[4]

1.2.3 DOCKER

Docker es una plataforma que permite separar las aplicaciones de la infraestructura [6], empaquetando aplicaciones y todas sus dependencias para que puedan correrse de forma aislada en cualquier máquina. Esto permite desplegar rápidamente una aplicación en un servidor remoto sin necesidad de máquinas virtuales ni afectar a otras aplicaciones.

1.2.3.1 Objetos básicos de Docker

- **Imagen** (*image*): es una plantilla de sólo lectura que se utiliza para crear contenedores, se construye a partir del archivo de configuración o *Dockerfile*.
- **Dockerfile**: contiene las instrucciones sobre cómo ensamblar una imagen, que se utilizará para construir un contenedor de Docker. Cada vez que se modifica este archivo y se vuelve a construir la imagen, se reconstruyen solamente aquellas capas que hayan cambiado o que se hayan añadido, lo que hace a Docker una tecnología de virtualización muy ligera en comparación con otras [6].
- **Contenedor** (container): es una instancia de la imagen. Es la unidad de software que ejecuta el código y sus dependencias de forma aislada del sistema y del resto de contenedores.
- **Docker Compose**: archivo YAML de configuración que define una aplicación Docker que contiene varios contenedores, en la que también se configuran las redes (puertos) y volúmenes montados.

1.2.3.2 Repositorios de Docker

Una de las características más útiles de Docker es poder encontrar, descargar y ejecutar imágenes que fueron creadas por otros desarrolladores [6]. Docker Hub es el repositorio público de imágenes de Docker más utilizado. La imagen de Apache Airflow está disponible en Docker Hub para descarga de los usuarios. La principal ventaja de desplegar Airflow como un container es que simplifica el proceso de despliegue de la aplicación, que a su vez necesita del despliegue de otros servicios (Redis, Postgres, etcétera). Además, permite mover una aplicación de Docker que funciona en una máquina a cualquier otra sin problemas de compatibilidad de versiones. Solamente es importante mantener una versión de Python consistente con la de Airflow.

1.2.4 AMAZON S3

Amazon S3 (Simple Storage Service) es un servicio de almacenamiento de objetos (cualquier tipo de archivo) de Amazon Web Services (AWS), diseñado para almacenar y recuperar cualquier cantidad de datos desde cualquier lugar del mundo a través de Internet.

1.2.4.1 Características de Amazon S3

Las principales características de Amazon S3, que son la razón de que su adopción haya crecido mucho en los últimos años, son las siguientes:

- Escalabilidad: la principal ventaja de utilizar un proveedor de servicios *cloud* como es AWS es la escalabilidad que permite a los usuarios, ya que, sin necesidad de contratar previamente cierta capacidad de almacenamiento, pueden escalar en necesidades y recursos que utilizan sin que suponga grandes desembolsos iniciales [7].
- Fiabilidad: el servicio Amazon S3 garantiza, con un 99,999999999% de seguridad [7], la durabilidad de los objetos. Se logra tal nivel de fiabilidad gracias a la replicación automática de los ficheros almacenados en múltiples instancias (distintas instalaciones) de la región contratada.
- Disponibilidad: Amazon S3 permite a las aplicaciones alcanzar tasas de solicitud de, aproximadamente, 3.500 PUT/COPY/POST/DELETE y 5.500 GET/HEAD por segundo y prefijo en un *bucket*. Si se realizan solicitudes paralelas, se puede llegar hasta a 55.000 solicitudes de lectura por segundo [8].
- Seguridad: se implementa cifrado de datos en tránsito, control de acceso según políticas definidas por los administradores y autenticación de usuario mediante AWS Identity and Access Management (IAM) [9].
- Usabilidad: la interfaz de AWS es muy sencilla de utilizar e intuitiva, existiendo además un servicio de línea de comandos, AWS CLI, y librerías para permitir la interacción con Amazon S3 a través de varios lenguajes de programación.

1.2.4.2 Conceptos principales de Amazon S3

Existen algunos conceptos básicos de Amazon S3 que son esenciales para entender su funcionamiento.

- *Buckets*: son los contenedores de objetos en Amazon S3. Todos los objetos que se almacenen deben pertenecer a un *bucket*. Los nombres de los *buckets* deben ser únicos a nivel global dentro de AWS [10].

- Objetos: entidad fundamental de almacenamiento, compuesto por datos, metadatos y una clave de identificación única "key". Pueden ser de cualquier tipo: imágenes, documentos, vídeos, etc.

Capítulo 2. DEFINICIÓN DEL TRABAJO

2.1 OBJETIVOS

El objetivo principal de este Trabajo de Fin de Máster es la construcción e implementación de un sistema ETL para gestionar una gran parte del ciclo de vida del dato de la empresa SwitchFleet. A continuación, se detallan, punto a punto, los objetivos específicos que se marcan para medir el éxito del proyecto.

2.1.1 DISEÑO Y DESARROLLO DE UN *PIPELINE* AUTOMATIZADO DE EJECUCIÓN DE PROCESOS

Diseño de una secuencia de procesos que incorpore todos los procesos actuales que forman parte de la generación de una nueva base de datos de vehículos actualizada e implementación técnica.

- Recopilar información sobre todos los procesos que se llevan a cabo actualmente como parte de la captación, transformación y carga de información en la base de datos de la compañía y las dependencias entre ellos.
- Optimizar los procesos, eliminando redundancias, inconsistencias y duplicados en los datos si los hubiera, así como creación de un repositorio común de datos.
- Implementación del *workflow* diseñado en Apache Airflow.
- Gestión de versiones del código y de submódulos de la ETL mediante GitHub.
- Ejecución extremo a extremo del *pipeline*.

2.1.2 PLANIFICACIÓN Y PERIODICIDAD DE LAS EJECUCIONES

Implementar un sistema que permita planificar las ejecuciones de los procesos de forma periódica y automática en un servidor remoto.

- Contratar y configurar un servidor remoto para la ejecución del proceso ETL.

- Definir la periodicidad con la que se ejecutarán los procesos y tiempos de espera para su compleción.

2.1.3 SUPERVISIÓN Y LANZAMIENTO MANUAL DE LA ETL EN INTERFAZ WEB

Poner a disposición de los usuarios una interfaz web intuitiva mediante la que poder lanzar el proceso de forma manual y supervisar su ejecución.

- Habilitar la interfaz web de Airflow UI a todos los usuarios con credenciales de acceso.
- Creación de *logs* informativos que se generen según se ejecutan los procesos y que aporten trazas y visibilidad de la evolución de cada uno, así como de los errores que puedan producirse.
- Capacitar al resto del equipo de datos de la compañía, y a quien se estimase necesario, en el uso de la interfaz, maximizando su disponibilidad.

2.1.4 RELANZAMIENTO AUTOMÁTICO DE LAS TAREAS EN CASO DE ERROR

Debido a la naturaleza de algunos procesos, que necesitan en su mayoría conexión a internet, es importante configurar mecanismos mediante los que volver a lanzar los procesos automáticamente en caso de que se produzcan errores.

- Implementar mecanismos que interrumpan los procesos y lancen avisos en caso de fallo.
- Definir tiempos de espera entre intentos de ejecución.
- Configurar los flujos de Airflow para que reintenten las tareas o el flujo entero en caso de error.

2.1.5 MECANISMOS DE VALIDACIÓN

Establecer métricas de validación de la calidad de los datos generados, garantizando su validez de cara a una actualización de los datos en producción.

- Identificar los indicadores de calidad del dato y establecer métricas para su medición.

- Implementar un procedimiento de validación de datos que se ejecute automáticamente tras cada ejecución de la ETL y que informe de las inconsistencias que pueda encontrar.

2.1.6 GENERACIÓN DE DOCUMENTACIÓN Y UNA GUÍA DE USO

Crear documentación detallada y una guía-manual de uso para el sistema ETL implementado, que permita que otros miembros del equipo trabajen en el futuro en la ETL y que cualquier usuario pueda utilizar la interfaz web.

- Redactar documentación técnica que describa el funcionamiento de la ETL y la forma de trabajar sobre ella.
- Crear un manual de uso sencillo sobre la interacción con la interfaz web.
- Llevar a cabo sesiones de formación para los usuarios y el equipo de datos.

2.2 METODOLOGÍA

Para llevar a cabo el proyecto con éxito, deben definirse una serie de pasos o etapas a llevar a cabo hasta su compleción.

- Contratación de un servidor remoto con sistema operativo Linux y los recursos necesarios para ejecutar el proceso ETL de principio a fin.
- Configuración del servidor con Docker y Apache Airflow. Instalación y configuración de Docker, descarga de la imagen de Airflow y personalización con Python y las librerías necesarias.
- Creación de un repositorio para el proyecto ETL en GitHub que integre, no solo el contenedor y la estructura de carpetas necesarias para lanzar Airflow en cualquier máquina, sino también los submódulos que conforman los procesos que deben ser lanzados en el *pipeline*.
- Desarrollo de un *script* de validación de los datos generados por el proceso ETL, que compruebe su calidad e informe de posibles inconsistencias y aborte la carga de datos en caso de ser necesario.
- Diseño del modelo de almacenamiento de datos en Amazon S3.

- Desarrollo de un *script* para subir los ficheros de datos generados durante el proceso ETL a un *bucket* de S3 y borrar los ficheros locales.
- Implementación de un DAG para la ejecución de las tareas (subprocesos de la ETL) y la gestión de las dependencias.

2.3 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

2.3.1 SUPUESTOS Y DATOS DE REFERENCIA

La estimación económica del coste a la empresa va a realizarse suponiendo un trabajador con las siguientes condiciones laborales:

- Horas de trabajo semanales: 20h.
- Salario anual bruto: 15.000 €.
- Horas de trabajo diarias disponibles para el proyecto: 2h (el resto del tiempo debe ser dedicado a otras tareas).
- Días laborales anuales: 260 días hábiles – 22 días de vacaciones = 238 días.
- Coste por hora del trabajador: 15.000 € / (238 días · 4h/día) = 15,76 €/hora.

2.3.2 ETAPAS DEL PROYECTO, ESTIMACIÓN DE TIEMPOS Y COSTES

Etapa	*Tiempo estimado (horas)*	*Coste estimado (euros)*
Identificación de las etapas y dependencias del proceso completo	14	220,64
Contratación de un servidor remoto	4	63,04
Configuración del servidor	20	315,20

Creación y estructuración del repositorio de código	10	157,60
Desarrollo del *script* de validación	15	236,40
Diseño del modelo de almacenamiento de datos en S3	10	157.60
Desarrollo del *script* para la carga en S3	16	252,16
Implementación del DAG	20	315,20
Resolución de errores y contingencias	25	394,00
Documentación	8	126,08
Sesiones de formación a compañeros	6	94,56
Almacenamiento en S3	NA	0,23 (mensual)
Servidor remoto	NA	27,71 (mensual)

Tabla 1. Estimación de tiempos y costes de cada etapa del proyecto.

Sobre el supuesto de dos horas de trabajo diarias empleadas al proyecto que fueron definidas en la sección anterior, se estima la duración total del desarrollo y puesta en marcha del proyecto en un total de 148 horas, es decir, 74 días laborables en la jornada estipulada, lo que supone un coste económico para la empresa de 2.332,48 €. A esta cifra, se le deben sumar los costes recurrentes del almacenamiento en Amazon S3 y del servidor remoto, que serían 27,94 €/mes.

Capítulo 3. DESARROLLO DEL PROYECTO

En este capítulo se detallan los pasos seguidos durante el desarrollo del proyecto de implantación de un proceso ETL. El desarrollo abarca todas las tareas descritas en apartados previos de este documento, desde la contratación y configuración de un servidor remoto, hasta la implementación de un DAG de Airflow y puesta en marcha del *pipeline* automatizado. El objetivo de este capítulo es describir en profundidad cada una de las tareas llevadas a cabo, junto con las configuraciones y herramientas empleadas.

3.1 ANÁLISIS DE LOS PROCESOS ACTUALES Y FUTUROS

3.1.1 ANÁLISIS "AS-IS"

En primer lugar, se analiza el proceso completo que conlleva la recopilación, transformación y carga de datos en S3 en la compañía previo a este proyecto, que actualmente se lanza proceso a proceso de forma manual. Se identifican las principales fuentes de datos de la compañía y los procesos que extraen información de cada una de ellas:

- *Web scraping* de características de vehículos.
- *Web scraping* de datos de mantenimiento de los vehículos.
- API de datos de seguridad de los vehículos.
- Obtención de precios de combustibles y factores de emisión mediante varias fuentes API y web.

Cada uno de los procesos de extracción de datos genera un archivo, en formato csv o json, que debe ser almacenado para tener la posibilidad de reconstruir la base de datos en un futuro.

Los archivos, pasan después por un proceso de transformación, denominado *merge*, en el que son limpiados y combinados en un solo archivo csv. Este mismo proceso se encarga de descargar imágenes de vehículos, eliminar el fondo de las mismas y subirlas a Google

Drive, almacenando la *url* de cada imagen y añadiéndola al archivo csv. La información de precios de combustibles y factores de emisión es introducida manualmente en la base de datos.

La conclusión de este primer análisis del as-is, o situación actual, es que la compañía utiliza procesos que son lanzados de forma manual sin una periodicidad establecida, lo que puede llevar a no tener los datos todo lo actualizados que deberían estar y a una dependencia del factor humano muy alta. Además, los datos se almacenan en máquinas locales y/o en una nube documental, lo que dificulta su uso por el resto de sistemas. Finalmente, se identifican algunos procesos completamente manuales como son la carga de archivos de datos a Google Drive y la actualización de los precios de combustibles y electricidad, así como de sus factores de emisión, manualmente en la base de datos. Esta situación supone una limitación en la capacidad de respuesta y de escalabilidad de la empresa.

3.1.2 ANÁLISIS GAP

Tras el análisis de la situación actual, se marcan ciertos cambios que deben producirse en los procesos actuales.

En primer lugar, se identifica un proceso demasiado pesado y con sub-procesos internos, que debe ser dividido. El proceso *merge*, además de unificar todos los datos recopilados previamente en un solo fichero, actualmente descarga, procesa y carga imágenes en Google Drive. A futuro, se separará la parte del proceso relativa a las imágenes, de forma que éstas sean procesadas y cargadas en el sistema de almacenamiento elegido de manera independiente a la limpieza y transformación de datos.

Adicionalmente, la compañía debe automatizar el lanzamiento de procesos, de manera que, cuando los datos de todas las fuentes hayan sido obtenidos, se proceda de forma automática a la preparación y carga de las imágenes de vehículos. A su vez, el final de este proceso debe desencadenar la creación del archivo de datos unificado (contenido actualizado de la base de datos en formato csv) y, después, la carga de todos los datos en el sistema de almacenamiento cloud elegido.

Actualmente, se emplea Google Drive como sistema de almacenamiento cloud, lo que conlleva muchos procesos manuales y dificulta la interacción con su API. Se migrará el almacenamiento de los datos del proceso ETL a AWS, concretamente, a S3. La estructura del modelo de almacenamiento de datos será discutida en otra sección de este capítulo.

También, se identifica la necesidad de validar de forma automática los archivos de datos generados. Tras recopilar los requisitos, se establecen las validaciones a realizar: número de registros, número de campos vacíos o sin información y columnas del archivo. Los valores de estos indicadores no deben variar de forma significativa en una ejecución exitosa del *pipeline* respecto a ejecuciones previas.

Finalmente, los datos que han sido validados y almacenados en S3, deberán actualizarse en una base de datos relacional Postgres, realizando un append de los datos nuevos y actualizando aquellas características que hayan cambiado.

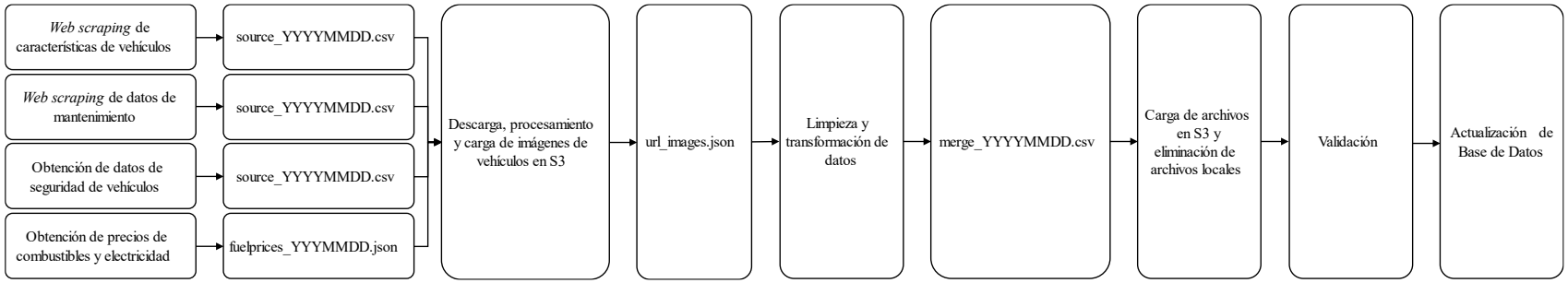

Figura 3. Diagrama del TO-BE del proceso ETL completo

3.2 CONTRATACIÓN DEL SERVIDOR REMOTO

La ETL va a ejecutarse en un servidor remoto, por lo que su contratación es uno de los primeros hitos de este proyecto. También, se debe seleccionar un proveedor de almacenamiento.

3.2.1 SELECCIÓN DEL PROVEEDOR DE SERVICIOS *CLOUD*

La contratación de un servidor remoto comienza por la selección de un proveedor de servicios en la nube que cumpla los requisitos, tanto económicos como técnicos, del proyecto. Los criterios de evaluación de proveedores fueron:

- Disponibilidad: el proceso de la ETL es migrable a cualquier otro sistema en unos tiempos medios (días) y, aunque no se trata de un proceso que deba ejecutarse diariamente, es importante que pueda ejecutarse manualmente en casi cualquier momento. Sin embargo, en el caso de los datos, al almacenar imágenes accesibles por clientes vía los productos de la compañía, deben estar disponibles siempre.
- Escalabilidad: se va a contratar un servidor con recursos superiores a los necesarios para este proyecto, como manera de prever un posible aumento en las etapas de la ETL. Sin embargo, no es necesario, al menos de momento, que el servidor contratado pueda escalar de forma flexible, ya que no se esperan picos de carga. De manera opuesta, los datos almacenados sí pueden escalar en cantidad, ya que el repositorio elegido será utilizado, posiblemente, por otras áreas de la compañía.
- Seguridad: el proveedor debe garantizar la seguridad de los datos, ya que el código de la compañía va a estar en sus servidores, así como datos clave del negocio.
- Costos: se busca un proveedor con costes lo más bajos posibles y, en el caso del almacenamiento de datos, con tarifas de transferencia de datos bajas.

Tras la obtención de información y posterior evaluación de distintos proveedores, se seleccionó Strato como proveedor del servidor remoto y Amazon S3 como sistema de almacenamiento *cloud*.

3.2.2 ESPECIFICACIONES DEL SERVIDOR

El proveedor del servidor, Strato, ofrece distintas opciones de especificaciones para el servidor a contratar. En esta etapa, en base a pruebas realizadas en el ordenador local de ejecución de los procesos, se establecen las especificaciones que debe tener el servidor contratado, dejando siempre un margen por encima de lo testado para posibles contingencias o escalabilidad del proceso ETL.

- Sistema Operativo: Ubuntu 22.
- CPU de 6 núcleos.
- Almacenamiento en disco: 500 Gb.
- Memoria RAM: 16 Gb.

Este servidor, tiene un coste mensual de 27,71 € (I.V.A. incluido).

3.3 CONFIGURACIÓN DEL SERVIDOR

Una vez contratado el servidor, éste debe ser configurado para poder instalar Apache Airflow *dockerizado* y ejecutar todos los procesos Python.

3.3.1 INSTALACIÓN DE PYENV COMO GESTOR DE ENTORNOS Y VERSIONES DE PYTHON

La instalación de Python en el servidor remoto, es esencial, no solo porque el propio Airflow funciona definiendo los flujos de trabajo en ese lenguaje, sino porque los *scripts* que van a lanzarse están desarrollados en Python. Existen muchas versiones de Python disponibles, por lo que se utiliza Pyenv, una herramienta muy popular entre usuarios de Python, para gestionar las versiones de Python y los entornos virtuales de la máquina.

Pyenv se instala clonando su repositorio de GitHub y definiendo las variables de entorno necesarias para poder llamarlo desde la terminal de Linux [11].

```
git clone https://github.com/pyenv/pyenv.git ~/.pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"' >>
~/.bashrc
```

```
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.profile
echo 'command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"' >>
~/.profile
echo 'eval "$(pyenv init -)"' >> ~/.profile
exec "$SHELL"
```

Una vez instalado Pyenv, se utilizan los siguientes comandos para instalar la versión necesaria de Python, que en este caso es la última compatible con Airflow, Python 3.11.0. Se requiere que ésta sea la versión de Python que se utilice sólo a nivel directorio del proyecto ETL, por si se utiliza el servidor para otros proyectos.

```
pyenv install 3.11.0
pyenv local 3.11.0
python –version
```

3.3.2 REPOSITORIOS

El siguiente paso en la configuración del servidor remoto, es la creación de un repositorio en la organización de GitHub de la empresa. Para ello, se siguen los pasos de creación dentro de la web de GitHub y se clona el repositorio en el servidor mediante el siguiente comando.

```
git clone <URL del Repositorio>
```

3.3.3 INSTALACIÓN DE DOCKER E IMAGEN DE APACHE AIRFLOW

En primer lugar, se debe instalar Docker acorde a la documentación que facilitan en su sitio web [6].

```
# Añadir certificado oficial de Docker
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Añadir el repositorio de Docker a los recursos de apt
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
```

```
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# Instalar Docker
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

# Verificar la instalación
sudo docker run hello-world
```

Después, se descarga la imagen de Apache Airflow y sus dependencias del repositorio oficial de Docker y se crea la estructura de carpetas que se montará en los contenedores de Docker.

```
curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.8.3/docker-
compose.yaml'
mkdir -p ./dags ./logs ./plugins ./config
export AIRFLOW_UID=50000
```

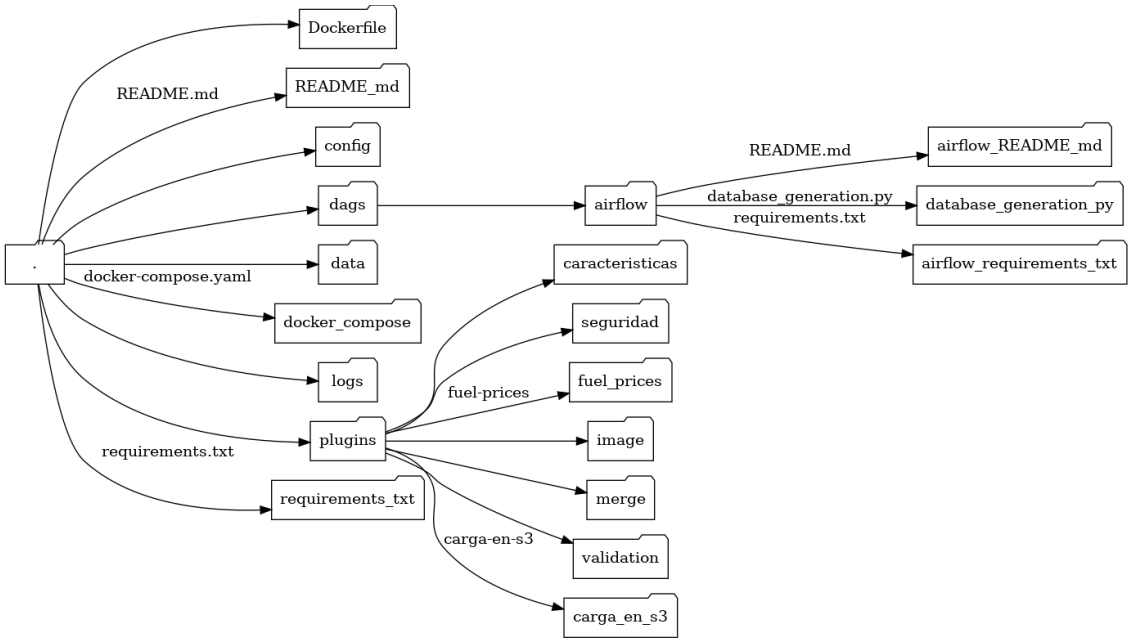

Figura 4. Estructura de directorios del proyecto

El siguiente paso es modificar el archivo docker-compose.yml, comentando la línea que contiene `#image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:2.8.3}` y des-comentando la

siguiente línea `build:` .. Después, se debe crear un archivo requirements.txt con todas las librerías de Python que vayan a utilizar los procesos que componen el proceso ETL.

Después, se debe crear un archivo Dockerfile con el siguiente contenido [12]:

```
FROM apache/airflow:2.8.3-python3.11
ADD requirements.txt .
RUN pip install apache-airflow==${AIRFLOW_VERSION} -r requirements.txt
```

Finalmente, se ejecuta el comando `docker compose up -d` para lanzar el contenedor de Docker en segundo plano. Se podrá acceder a la interfaz web de Apache Airflow a través del navegador, en la dirección http://localhost:8080 con usuario y contraseña "airflow" por defecto, que deben ser modificados por seguridad.

3.3.4 SUBMÓDULOS

Los submódulos en Git permiten agregar otros repositorios como subdirectorios a un repositorio principal, de manera que, si éstos sufren modificaciones, se vean reflejadas en el repositorio padre. Los procesos que componen la ETL serán añadidos como submódulos dentro del directorio *plugins* del repositorio de la ETL. Para ello, hay que navegar hasta el directorio *plugins* y ejecutar los siguientes comandos para cada subproceso de la ETL [13]:

```
git submodule add <URL_del_repositorio>
git commit -m 'Add submodule'
git push
```

3.4 SCRIPT DE VALIDACIÓN

Una parte esencial de cualquier proceso de transformación y carga de datos consiste en la comprobación de la calidad de éstos, de manera que pueda conocerse si el proceso se ha desarrollado de manera satisfactoria y abortar la carga de datos en caso contrario, para evitar tener datos de mala calidad o erróneos en el almacén de datos de la compañía. Es por ello que, como paso previo a la carga de datos en S3, se ejecutará un *script* de validación que debe ser desarrollado.

El *script* tomará como *input* el archivo "merge.csv" resultado de la extracción y transformación de datos, así como archivos generados anteriormente, para comparar y validar la consistencia de los datos. Tras el análisis de las fases de definición del As-Is y el Gap, se definen las principales métricas que deben comprobarse, arrojando un OK o un KO según si se considera válido o no. En caso de que el archivo se considere no válido, se abortará la ETL, previniendo la carga del archivo incorrecto y arrojando un error en el proceso total, con *logs* que aportan visibilidad sobre los fallos.

- Número de registros: se comprueba la variación en el número de registros respecto al archivo anterior. Si la variación es superior a un 0,1 %, el archivo se considera no válido. La ejecución de la ETL se realiza una vez en semana y una variación tan significativa supondría un aumento o disminución en el número de vehículos disponibles en el mercado muy significativo para ese periodo temporal.
- Columnas: comprobación de si las columnas (variables) del archivo nuevo y el anterior son las mismas. La validación se considera exitosa si no hay ningún cambio en las columnas.
- Valores en blanco: se compara la cantidad de valores en blanco de cada columna entre el archivo original y el nuevo. Se considera un archivo correcto si la variación es inferior al 10%, ya que al contener muchos procesos de *web scraping*, existe una mayor tendencia a obtener valores nulos en variables de relevancia menor.

El *script* se desarrolla en lenguaje Python, empleando las librerías pandas, logging y boto3 para la manipulación de datos, el *logging* de eventos y la interacción con Amazon S3.

3.5 DISEÑO DEL MODELO DE DATOS EN AMAZON S3

Se define como requisito el poder almacenar todos los archivos que se utilicen en el proceso de generar y actualizar la base de datos de la compañía, por lo que se diseña un modelo de datos acorde a esta necesidad, que además de organizar los datos según la fuente o destino de los ficheros, lo haga por fecha de generación.

En primer lugar, se identifica la necesidad de permitir el acceso público a las imágenes que se almacenen en S3, para que éstas puedan ser utilizadas por los distintos productos de la compañía. Por lo tanto, se dividen los datos en dos *buckets*, un *public-bucket* que contiene las imágenes de los vehículos con nomenclatura marca_modelo_YYYY.png y otro *private-bucket* que contiene todos los archivos de datos obtenidos por *web scraping* o peticiones API y los archivos de datos procesados.

El *private-bucket* está estructurado en dos carpetas o directorios, *scraped-data* (datos extraídos vía peticiones y *web scraping*) y *processed-data* (archivo de datos final).

- scraped-data: contiene los datos almacenados por fuente y fecha de extracción (private-bucket/scraped-data/fuente/YYYY/MM/DD/fuente.csv o .json).
- processed-data: contiene los datos extraídos combinados, limpiados y procesados, junto con la fecha de procesamiento (private-bucket/processed-data/merge/YYYY/MM/DD/merge.csv).

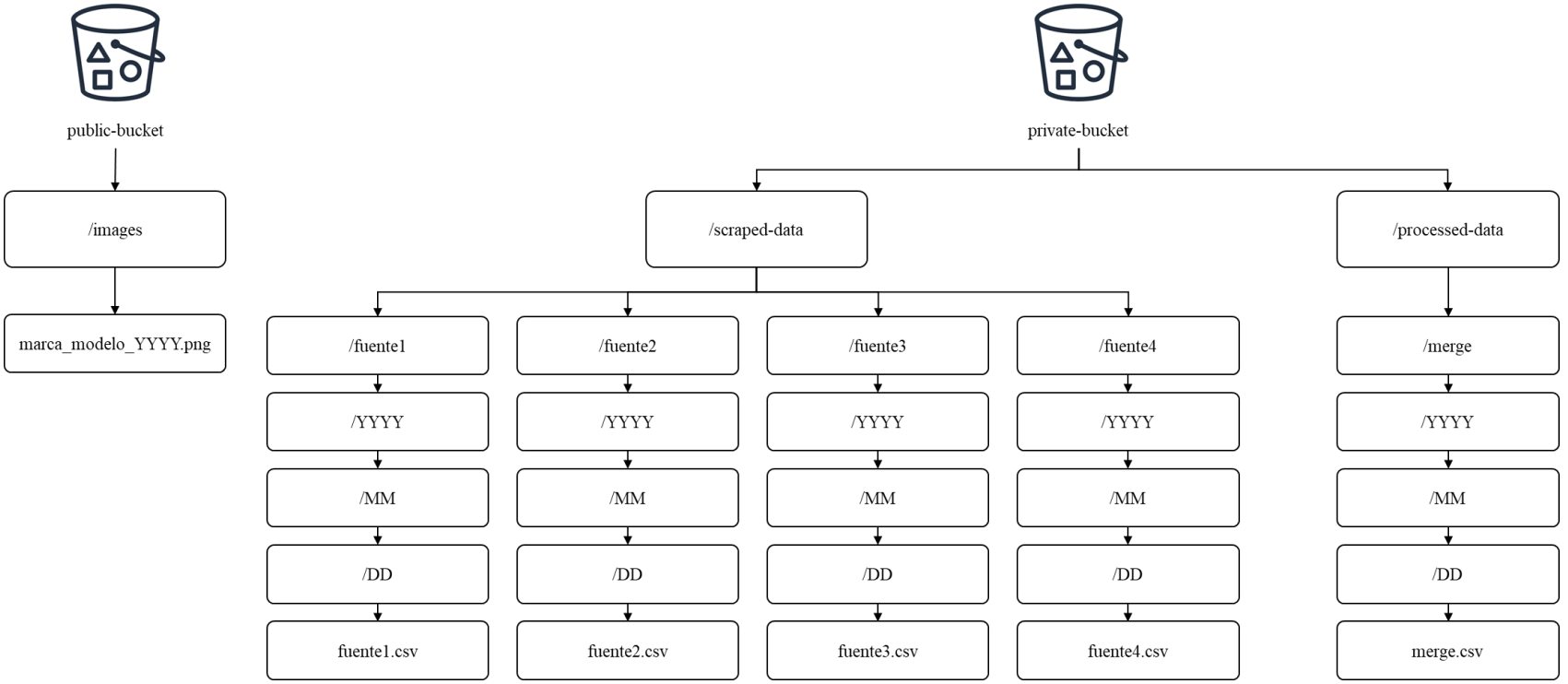

Figura 5. Modelo de almacenamiento de ficheros en Amazon S3

3.6 SCRIPT DE CARGA DE FICHEROS EN S3

La última etapa del proceso ETL (*Extract, Transform, Load*) es la de carga de los ficheros y datos generados a lo largo del proceso. En el contexto de este proyecto, se ha decidido utilizar Amazon S3 como plataforma de almacenamiento de ficheros. Se desarrolla un *script* de carga de todos los ficheros de datos extraídos mediante *web scraping* y peticiones a APIs, así como del fichero final merge (datos limpios y unificados), que recoge los ficheros locales y los sube a S3 con la nomenclatura y estructura adecuada, borrando del servidor los ficheros tras este proceso.

Se utilizan los módulos de Python *sys* y *os* para interactuar con el sistema operativo. El *script* recibe como argumentos parámetros sobre la ruta de los archivos a subir y el *bucket* de S3, comprueba si el directorio existe y tiene contenido, y recorre la lista de ficheros del directorio subiéndolos a S3.

Los ficheros que generan todos los procesos siguen la nomenclatura nombre_YYYYMMDD.csv o .json. Se desarrolla una función que separa el "nombre" del fichero de cada elemento de la fecha (año, mes, día) y de su extensión. Se desarrolla otra función, que es llamada a continuación, y sube a S3 el archivo local con la nomenclatura y estructura definida en el modelo de datos. Ésta última función recibe como argumentos el *bucket* al que se va a subir la información, la ruta al archivo local y la ruta de S3. Finalmente, si el archivo se ha subido correctamente, se elimina el archivo local del servidor.

El desarrollo de este *script* se ha realizado de forma modular, de manera que las funciones de carga a S3 se puedan reutilizar en otros proyectos. Parte del código, se incorpora también en el *script* de "Descarga, procesamiento y carga de imágenes en S3", que se desacopló del antiguo *merge.py*, como se vio en el análisis *gap*, sustituyendo la carga en Google Drive por una carga en S3.

3.7 ACTUALIZACIÓN DE LA BASE DE DATOS

Para garantizar una carga de datos robusta y gestionada en la base de datos PostgreSQL, se ha implementado Liquibase como herramienta de gestión de cambios. Liquibase permite versionar y controlar los cambios en la estructura de la base de datos, así como automatizar la carga de datos de manera segura y eficiente.

3.7.1 CONFIGURACIÓN DE LIQUIBASE

Se creó un directorio específico para los scripts de cambios (*changelogs*), donde se definen los cambios estructurales y la inserción de datos. Además, se configuró el archivo `liquibase.properties` con los parámetros de conexión a la base de datos PostgreSQL y la ruta a los *changelogs*.

3.7.2 PROCESO DE CARGA

- **Generación de changelogs**: Se diseñaron scripts XML para definir la estructura de las tablas y la inserción de datos procesados.
- **Integración con Airflow**: Se añadió una tarea en el DAG de Airflow para ejecutar los comandos de Liquibase (*update*) después de la validación de datos.
- **Validación de la carga**: Liquibase proporciona un control de versiones de los cambios aplicados, permitiendo auditar la ejecución de scripts y garantizando la consistencia de los datos.

3.8 DAG DE AIRFLOW

En Airflow, el DAG es el código Python que define el *pipeline* del proceso ETL de principio a fin y orquesta las tareas que se van a ejecutar, incluyendo la forma de ejecutar cada tarea individual, las dependencias entre ellas y la configuración del *workflow*. A partir de los procesos individuales, tanto aquellos ya desarrollados previos a este proyecto como los nuevos y los que han sido modificados, junto con el análisis de requerimientos y definición del flujo completo de la ETL, se desarrolla el DAG.

3.8.1 CONFIGURACIÓN DE PARÁMETROS GLOBALES

En coordinación con las necesidades de ejecución y planificación de la ETL y en base a los objetivos marcados, se define el DAG con los siguientes parámetros:

- Programación: `timedelta(weeks=1)` ejecución semanal del proceso ETL.
- Ejecuciones activas: `max_active_runs=1` para asegurar una única ejecución del DAG de forma simultánea.
- Ejecuciones retroactivas: `catchup=False` para evitar el lanzamiento de ejecuciones antiguas pendientes al reiniciar Airflow.
- Reintentos: `'retries': 1` y `'retry_delay': timedelta(minutes=5)` para reintentar las tareas en caso de error tras cinco minutos.

3.8.2 TAREAS

El DAG incluye como tareas cada uno de los sub-procesos de la ETL. Cada tarea es lanzada mediante un operador de Airflow que ejecuta el *script* correspondiente mediante comandos de terminal. Se incluyen dos tareas inicial y final *dummy* como nodos de inicio y fin del *pipeline* completo. El resto de las tareas pasan a los *scripts* de Python como argumento las rutas de los archivos a generar y/o subir a S3.

- **scraping_caracteristicas**: utiliza un BashOperator para ejecutar el spider de Scrapy y obtener datos de una fuente web.
- **scraping_mantenimiento**: se lanza un spider de Scrapy mediante BashOperator que obtiene datos de mantenimiento de vehículos.
- **get_seguridad**: ejecución por BashOperator de un *script* de Python que lanza peticiones y recopila datos de seguridad de vehículos.
- **get_fuel_prices**: obtiene precios de combustibles y factores de emisión de éstos mediante un *script* de Python que ejecuta distintas peticiones a APIs. Se lanza mediante un BashOperator.
- **images**: utiliza BashOperator para ejecutar un *script* de Python que descarga y procesa imágenes y las sube a S3.

- **merge**: ejecuta mediante BashOperator el archivo merge.py, que recibe la ruta con todos los archivos a unir en uno solo, los limpia, combina y guarda como archivo final.
- **validation**: se ejecuta con BashOperator el *script* de validación del fichero merge.csv, que registra las posibles inconsistencias con archivos pasados y aborta el proceso en caso de fichero no válido.
- **carga_a_s3**: se llama a través de un BashOperator al *script* de carga a S3, que carga todos los archivos generados en S3 según el modelo de almacenamiento de datos.
- **update_database**: utiliza un BashOperator para ejecutar el comando de Liquibase encargado de aplicar los cambios en la base de datos PostgreSQL. Esta tarea se encarga de ejecutar el comando liquibase update para insertar los datos validados en la base de datos, asegurando que la estructura y el contenido de las tablas estén actualizados conforme a los cambios definidos en los changelogs. En caso de error durante la ejecución, se generan logs detallados y se envían notificaciones al equipo técnico.

3.8.3 DEFINICIÓN DEL *PIPELINE*

Dentro del DAG, tras la definición de las tareas, se indica la secuencia en la que éstas deben ser ejecutadas y las dependencias entre ellas. En primer lugar, se ejecuta la tarea *dummy* de inicio, a continuación, se lanzan en paralelo todos los procesos de extracción de datos y, cuando éstos han acabado, se procede a la fase de transformación. En esta fase se procesan las imágenes y se genera el fichero final de datos con todas las fuentes de datos unificadas y formateadas. Finalmente, se valida este último fichero y, en caso de ser correcto, se procede a la carga de todos los archivos en S3.

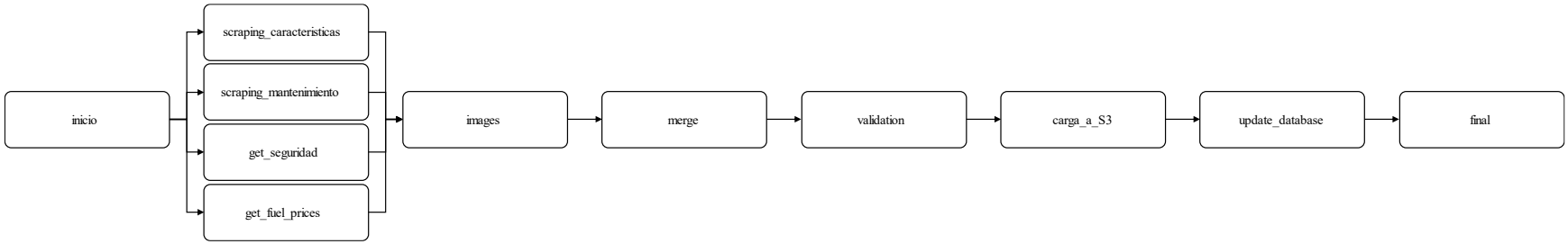

Figura 6. Gráfico del DAG de Airflow

Capítulo 4. ANÁLISIS DE RESULTADOS

El desarrollo del proceso ETL se realizó con éxito, cumpliendo los objetivos marcados. El *pipeline* ejecuta todos los procesos de extracción de datos y los procesa y une en un solo archivo, procesando y subiendo las imágenes de los vehículos por el camino. Finalmente, sube todos los archivos generados a S3, siguiendo la estructura de datos y nomenclatura de los ficheros definida, siendo previamente validados con el *script* de validación desarrollado.

Los archivos resultantes se contrastaron con aquellos provenientes de una ejecución similar con el proceso manual que se realizaba antes de este proyecto, obteniendo unos resultados idénticos.

Los tiempos de ejecución se mejoraron, obteniendo un archivo final en un tiempo de entre cinco y seis horas, lo que previamente podía suponer más de una jornada laboral de supervisión. Además, ahora, cualquier usuario con credenciales de acceso puede acceder a la interfaz web de Airflow, democratizando la ejecución y monitorización del proceso completo y evitando que, por ejemplo, las vacaciones o una baja de alguno de los responsables de los procesos, interrumpan la actualización de los datos de la compañía. Incluso sin supervisión por parte de ningún trabajador, el proceso se ejecutará, de forma periódica, una vez en semana, manteniendo actualizado el repositorio de datos de Ozone Drive.

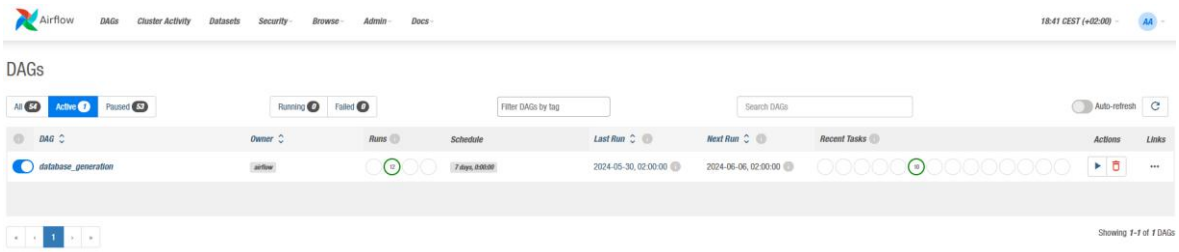

Figura 7. Interfaz web de Airflow

Capítulo 5. CONCLUSIONES Y TRABAJOS FUTUROS

El presente Trabajo de Fin de Máster se ha centrado en el desarrollo e implementación de un proceso ETL para la recomendación de vehículos en Ozone Drive. Se han utilizado herramientas y tecnologías actuales, como Apache Airflow como orquestador de procesos, Python como lenguaje de programación y Amazon S3 como servicio de almacenamiento *cloud*. En este capítulo se muestran las principales conclusiones alcanzadas tras la finalización del proyecto y los trabajos que pueden realizarse a futuro.

5.1 CONCLUSIONES

5.1.1 EFICIENCIA Y AUTOMATIZACIÓN DE LOS PROCESOS

La implementación de Apache Airflow como orquestador de procesos, ha permitido la ejecución automática y periódica de todos los procesos necesarios para actualizar el núcleo de datos de Ozone Drive. Esto ha supuesto un aumento en la eficiencia del equipo de datos y permite garantizar a todo el equipo que los datos van a estar actualizados siempre.

5.1.2 INTERFAZ WEB ACCESIBLE Y FORMACIONES

Uno de los objetivos del proyecto era permitir a todos los equipos que haga falta, no solo al de datos o aquellas personas con acceso al servidor remoto, tener visibilidad y poder monitorizar la ejecución de los procesos que conforman la ETL. Este objetivo se ha logrado de varias maneras:

- Poniendo a disposición de los usuarios que lo requieran credenciales de acceso a una interfaz web accesible desde cualquier ordenador y desde la que monitorizar el proceso y lanzarlo manualmente.
- Subiendo de forma automática todos los archivos de datos a Amazon S3 y dando acceso a toda la compañía. De esta manera se eliminan las trabas que suponía acceder a Google Drive y la gestión de permisos y propiedad de los archivos.

- Realizando sesiones divulgativas con el resto del equipo de datos, permitiendo que todo el mundo conozca el proceso ETL y cómo se ha desarrollado, así como la forma de utilizarlo.

5.1.3 ALMACENAMIENTO EN AMAZON S3

La elección de Amazon S3 como servicio de almacenamiento ha resultado ser una opción muy satisfactoria para todo el equipo. Ha permitido que todos los servicios de la compañía accedan a los datos más actualizados en tiempo real y de forma automatizada, mediante uso de la consola o librerías, que permiten a aquellos con credenciales acceder a los datos, sin la gestión manual que supone descargar un archivo de Google Drive.

5.1.4 CALIDAD DEL DATO

El desarrollo de un *script* de validación de los datos generados ha sido clave para ayudar a garantizar la calidad de los datos, que alimentan todos los productos de la compañía y son clave para su éxito. Definir junto con todo el equipo los indicadores de calidad de un fichero de datos ha sido uno de los puntos más importantes para poder evaluar los ficheros generados.

5.1.5 IMPACTO EN EL NEGOCIO

El impacto de este proyecto en el negocio tiene varias patas. Por un lado, permite la seguridad de tener datos actualizados con una frecuencia fija y sin depender de la disponibilidad de algunos empleados. También, es un punto clave en el proceso de definición de Ozone Drive de establecerse como una compañía de datos, permitiendo tener una base sólida de procesos interconectados a partir de la que escalar, añadiendo procesos, fuentes de datos, cambios en los formatos, etcétera.

5.1.6 LECCIONES APRENDIDAS

Entre las principales lecciones aprendidas durante el desarrollo de este Trabajo de Fin de Máster, destaco:

- Importancia de realizar una toma de requerimientos en profundidad y comunicarse con el equipo para entender las necesidades y expectativas del producto final.
- Unificación de librerías y versiones de Python. El lenguaje de programación Python está muy vivo, sufriendo actualizaciones continuas tanto de él como de sus librerías. Fue necesario establecer un entorno de desarrollo común para todos los procesos que conformasen la ETL y adaptar el código a una versión de Python soportada por Airflow, que va por detrás de las últimas actualizaciones.
- Documentar el código. El proceso ETL lanza muchos otros subprocesos que han tenido que ser adaptados para este proyecto. Muchos de los procesos, tenían escasa documentación disponible, complicando la tarea de entenderlos. Por tanto, se definió como necesidad mejorar la documentación del código y la claridad del mismo, de forma que cualquier persona pueda retomar los desarrollos sin un gran esfuerzo inicial.

Capítulo 6. BIBLIOGRAFÍA

[1] V. N. Gudivada, R. Baeza-Yates, y V. V. Raghavan, "Big Data: Promises and Problems," *Computer*, vol. 48, no. 3, pp. 20-23, Mar. 2015, doi: 10.1109/MC.2015.62.

[2] El-Sappagh, S. H., Hendawi, A. M. A., & El Bastawissy, A. H. (2011). A proposed model for data warehouse ETL processes. *Journal of King Saud University – Computer and Information Sciences*, 23(2), 91-104.

[3] Kimball, R., & Caserta, J. (2004). The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley.

[4] Apache Airflow Documentation, "Apache Airflow," 2023. [Online]. Disponible en: https://airflow.apache.org/docs. [Accedido: 18-jun-2024].

[5] Apache Airflow. "Basic Airflow Deployment", *Apache Airflow Documentation*, [Online]. Disponible en: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/overview.html#basic-airflow-deployment. [Accedido: 18-jun-2024].

[6] Docker. "Install Docker Engine on Ubuntu" en Docker Documentation, [Online]. Disponible en: https://docs.docker.com/engine/install/ubuntu/. [Accedido: 18-jun-2024].

[7] Amazon Web Services, "Amazon S3 - Overview", *Amazon Web Services Documentation*. [Online]. Disponible en: https://aws.amazon.com/s3/. [Accedido: 18-jun-2024].

[8] M. S. H. C. T. et al., "Amazon S3 Durability and Availability", *AWS Whitepapers*. [Online]. Disponible en: https://d1.awsstatic.com/whitepapers/Amazon_S3_Design_and_Usage_Guide.pdf. [Accedido: 18-jun-2024].

[9] Amazon Web Services, "Amazon S3 Security", *Amazon Web Services Documentation*. [Online]. Disponible en:

https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html. [Accedido: 18-jun-2024].

[10] Amazon Web Services, "Amazon S3 Concepts", *AWS Documentation*. [Online]. Disponible en: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html. [Accedido: 18-jun-2024].

[11] Pyenv. "pyenv/README.md" en GitHub, [Online]. Disponible en: https://github.com/pyenv/pyenv?tab=readme-ov-file#instalation. [Accedido: 18-jun-2024].

[12] Apache Airflow. "Docker Compose Quickstart" en Apache Airflow Documentation, [Online]. Disponible en: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html. [Accedido: 18-jun-2024].

[13] Git. "Git Tools - Submodules" en Pro Git Book, [Online]. Disponible en: https://www.git-scm.com/book/en/v2/Git-Tools-Submodules. [Accedido: 18-jun-2024].