UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

Digitalización de los flujos de datos para predicción energética de edificios inteligentes a través de la implementación de un *data-lake* bajo técnicas *Big-Data*

Autor:

D. David Arévalo González

Tutores:

D. José Luis Hernández García
Dr. D. Ignacio de Miguel Jiménez

Valladolid, Julio 2024

TÍTULO:	**Digitalización de los flujos de datos para predicción energética de edificios inteligentes a través de la implementación de un *data-lake* bajo técnicas *Big-Data***
AUTOR:	**D. David Arévalo González**
TUTORES:	**D. José Luis Hernández García** **Dr. D. Ignacio de Miguel Jiménez**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

Tribunal

PRESIDENTE:	**Dr. D. Ramón J. Durán Barroso**
VOCAL:	**Dr. Juan Carlos Aguado Manzano**
SECRETARIO:	**Dr. Ramón de la Rosa Steinz**
SUPLENTE 1:	**Dra. D.ª Patricia Fernández del Reguero**
SUPLENTE 2:	**Dr. Juan Blas Prieto**
SUPLENTE 3:	**Dr. Alfonso Bahillo Martínez**
FECHA:	**Julio 2024**
CALIFICACIÓN:	

Agradecimientos

En primera instancia me gustaría expresar mi más sincero agradecimiento a todos los participantes y miembros del consorcio del proyecto DigiBUILD (financiado por la Unión Europea, *Grant Agreement* 101069658) [1] por los recursos proporcionados y la invaluable ayuda durante el desarrollo de este proyecto, así como por su dedicación y esfuerzo. Además, cabe destacar y agradecer los esfuerzos tanto sociales como económicos que ha supuesto la financiación proporcionada por la Comisión Europea, no sólo en el proyecto DigiBUILD sino en todos aquellos que favorecen la innovación y la investigación en todas sus formas.

En lo relativo al trabajo me gustaría agradecer el apoyo de Jose y Susana, así como de Nacho, por el apoyo y todos los consejos aportados para poder llevar a cabo el trabajo con la excelencia que requiere y facilitar el desarrollo del mismo.

Por otro lado, agradecer a mi familia y pareja, Teresa, Roberto, Álvaro y Carlos, el apoyo tan grande que han sido, no solo en la realización del trabajo, sino en todo mi periodo de formación superior. Gracias por estar en los mejores y en los peores momentos permitiendo terminar esta etapa siendo mi mejor versión.

Resumen

Actualmente, nos encontramos en un periodo de transición digital donde muchos procesos, tanto industriales como cotidianos, están sufriendo una migración hacia lo digital, específicamente en nuestro caso, en el sector de la construcción. Hasta ahora, el enfoque que se ha estado implementando se conoce como un enfoque de silo, donde cada uno de los actores interesados gestiona individualmente sus datos de forma aislada. Proponiendo una alternativa al enfoque actual, en este Trabajo Fin de Máster se han desarrollado metodologías de extracción, transformación y almacenaje de datos procedentes de un gran número de fuentes heterogéneas a lo largo de 10 edificios piloto dotándolos de homogeneidad y calidad. Además, se ha definido una base de datos que permite su consumo a través de una interfaz homogénea. Finalmente, se ha querido demostrar una forma de compartición y uso de los datos de los servicios de DigiBUILD a través de la definición de una API para la predicción de diferentes variables energéticas. Todo este desarrollo se ha podido llevar a cabo bajo los materiales y el marco de desarrollo del proyecto europeo DigiBUILD y, más en concreto en nombre de CARTIF, cuyo fin último es la solución que se propone.

Palabras Clave

Espacios de datos, ETLs, *big-data*, *data-lake*, interoperabilidad, calidad de datos, edificios inteligentes, eficiencia energética, *machine learning*.

Abstract

Currently, we are in a period of digital transition where many processes, both industrial and everyday-life, are undergoing a migration towards digitalization, specifically in our case, in the construction sector. Until now, the approach that has been implemented is known as a silo approach, where each stakeholder manages their data individually and in isolation. Proposing an alternative to the current approach, in this Master's Thesis, methodologies for extracting, transforming, and storing data from a large number of heterogeneous sources have been developed across 10 pilot buildings, endowing them with homogeneity and quality. Additionally, a database has been defined that allows its consumption through a homogeneous interface. Finally, an API has been developed to showcase how a sharing point to some energy-related forecasting servies work. All this development has been carried out under the materials and the development framework of the European project DigiBUILD and, more specifically as CARTIF, whose ultimate goal is the proposed solution.

Keywords

Data spaces, ETLs, big-data, data-lake, interoperability, data quality, smart building, energy efficiency, machine learning.

Índice general

1. Introducción **1**
- 1.1. Hipótesis . . . 1
- 1.2. Objetivos . . . 3
- 1.3. Estructura del documento . . . 3

2. Contexto del proyecto **5**
- 2.1. Big Data . . . 5
- 2.2. Diagrama conceptual . . . 6
- 2.3. Herramientas software empleadas . . . 9
  - 2.3.1. Pentaho . . . 9
  - 2.3.2. PostgreSQL . . . 10
  - 2.3.3. Librerías de Python . . . 10

3. Arquitectura del sistema **11**
- 3.1. Arquitecturas de referencia y de proyectos europeos para gestión *big data* . . . 13
  - 3.1.1. Arquitecturas de referencia estándares de la EU . . . 13
  - 3.1.2. Arquitecturas de referencia en proyectos europeos . . . 14
- 3.2. Arquitectura de referencia de DigiBUILD . . . 14

4. Extracción de Datos Dinámicos **17**
- 4.1. ETL del piloto 01 - UCL . . . 23
  - 4.1.1. Dimensiones del *big data* . . . 23
  - 4.1.2. Flujos de datos . . . 24
  - 4.1.3. Salidas . . . 26
- 4.2. ETL del piloto 02 - EDF . . . 28
  - 4.2.1. Dimensiones del *big data* . . . 28
  - 4.2.2. Flujos de datos . . . 29
  - 4.2.3. Salidas . . . 31
- 4.3. ETL del piloto 03 - IASI&SITTA . . . 36
  - 4.3.1. Flujo de datos . . . 37
  - 4.3.2. Salidas . . . 38
- 4.4. ETL del piloto 04 - VEOLIA . . . 40
  - 4.4.1. Dimensiones del *big data* . . . 40
  - 4.4.2. Flujos de datos . . . 41
  - 4.4.3. Salidas . . . 42
- 4.5. ETL del piloto 05a - EMOT . . . 44

4.5.1. Dimensiones del *big data* . . . 44
4.5.2. Flujos de datos . . . 45
4.5.3. Salidas . . . 47
4.6. ETL del piloto 05b - FOCCHI . . . 49
4.6.1. Dimensiones del *big data* . . . 49
4.6.2. Flujos de datos . . . 50
4.7. ETL del piloto 06 - HERON . . . 55
4.7.1. Dimensiones del *big data* . . . 55
4.7.2. Flujos de datos . . . 56
4.8. ETL del piloto 07 - FVH . . . 59
4.8.1. Dimensiones del *big data* . . . 59
4.8.2. Flujos de datos . . . 60
4.9. ETL del piloto 08 - IEECP . . . 62
4.9.1. Dimensiones del *big data* . . . 62
4.9.2. Flujos de datos . . . 63
4.10. ETL del piloto 09 - NTUA . . . 67
4.10.1. Dimensiones del *big data* . . . 67
4.10.2. Flujos de datos . . . 68
4.11. ETL de la API meteorológica . . . 71
4.11.1. Dimensiones del *big data* . . . 71
4.11.2. Flujos de datos . . . 72
4.11.3. Salidas del flujo . . . 72

5. Metodologías de Calidad de Datos **75**
5.1. Dimensiones bajo análisis . . . 75
5.2. Implementación y resultados . . . 77

6. Base de datos para de datos contextuales y dinámicos: Data Warehouse y Knowledge Graph **79**
6.1. DWH: Bases de datos relacionales y no relacionales . . . 80
6.2. DWH: Esquema en estrella . . . 81
6.3. Definición de la base de datos dinámica del Data Warehouse . . . 82
6.4. DWH: Propuestas para la mejora de la gestión de grandes volúmenes de datos . . . 84
6.5. Metadatos y su sincronización . . . 87
6.5.1. Elementos del data lake . . . 87
6.5.2. Mecanismos de sincronización . . . 88
6.6. Interfaz para compartición de datos . . . 88
6.6.1. *Endpoints* . . . 89

7. Conclusiones **96**

Glosario **99**

Bibliografía **102**

Índice de figuras

1. Logo de DigiBUILD 3

2. Diagrama conceptual del TFM 7
3. Pilotos del proyecto 8

4. Arquitectura de DigiBUILD 12

5. Flujos genéricos en una ETL 20
6. Job piloto 01 24
7. ETL piloto 01 24
8. Equivalencia entrada del DWH y mensaje de Kafka 26
9. Piloto 01 - Salida de metadatos 26
10. Piloto 01 - Salida de datos dinámicos 27
11. Job piloto 02 29
12. ETL piloto 02 - Línea Ethera 30
13. ETL piloto 02 - Línea Ellona 31
14. ETL piloto 02 - Línea Wattsense 32
15. Piloto 02: Línea Ethera - Salida de metadatos 32
16. Piloto 02: Línea Ethera - Salida de datos dinámicos 33
17. Piloto 02: Línea Ellona - Salida de metadatos 33
18. Piloto 02: Línea Ellona - Salida de datos dinámicos 34
19. Piloto 02: Línea Wattsense - Salida de metadatos 35
20. Piloto 02: Línea Wattsense - Salida de datos dinámicos 35
21. Job piloto 03 37
22. ETL piloto 03 - Línea Energía 38
23. ETL piloto 03 - Línea Calidad Aire 38
24. Piloto 03: Línea Calidad de Aire - Salida de metadatos 39
25. Piloto 02: Línea Calidad de Aire - Salida de datos dinámicos 39
26. Job piloto 04 41
27. ETL piloto 04a 42
28. ETL piloto 04b 42
29. Piloto 04: Líneas FASA y Río Vena - Salida de metadatos 43
30. Piloto 04: Líneas FASA y Río Vena - Salida de datos dinámicos 43
31. Job piloto 05a 45
32. ETL piloto 05a - Línea producción PV 46
33. ETL piloto 05a - Línea consumo edificio 47
34. ETL piloto 05a - Línea vehículo eléctrico 47

35. Piloto 05a: Línea CS - Salida de datos dinámicos 48
36. Job piloto 05b 50
37. ETL piloto 05b - Línea generación PV 51
38. ETL piloto 05b - Línea sala 1 52
39. ETL piloto 05b - Línea sala 2 52
40. ETL piloto 05b - Línea Google Drive 53
41. Piloto 05a: Línea generación PV - Salida de datos dinámicos 53
42. Piloto 05a: Línea Sala 2 - Salida de datos dinámicos 54
43. Job piloto 06 56
44. ETL piloto 06 57
45. Piloto 06 - Salida de metadatos 58
46. Piloto 06 - Salida de datos dinámicos 58
47. Job piloto 07 60
48. ETL piloto 07 60
49. Piloto 07 - Salida de metadatos 61
50. Piloto 07 - Salida de datos dinámicos 61
51. Job piloto 08 63
52. ETL piloto 08 - Línea MQTT 64
53. ETL piloto 08 - Línea Netatmo 65
54. Piloto 08: Línea MQTT - Salida de datos dinámicos 65
55. Piloto 08: Línea Netatmo - Salida de datos dinámicos 66
56. Job piloto 09 68
57. ETL piloto 09 69
58. Piloto 09: Tabla Aire Acondicionado e Iluminación - Salida de datos dinámicos 70
59. Piloto 09: Tabla Calidad Aire - Salida de datos dinámicos 70
60. Job API meteorológica 72
61. ETL API meteorológica 73
62. Salida de Kafka API meteorológica 73
63. Salida de DWH API meteorológica 74

64. Consistencia en una distribución genérica 77
65. Implementación de Data Quality en ELTs 78

66. Esquema en estrella 81
67. Diagrama de tablas del DWH 83
68. Endpoints definidos en la API 90
69. Endpoint description: get_models 91
70. Endpoint response: get_models 91
71. Endpoint description: get_info 92
72. Endpoint response: get_info 93
73. Endpoint description: get_prediction 94
74. Endpoint response: get_predicion 95

Índice de tablas

1. Interfaces para la extracción de datos dinámicos 18
2. *Topics* para la publicación de datos en tiempo real 21
3. *Topics* de predicción meteorológica 73
4. Clasificación dimensiones calidad de datos 76

Capítulo 1

Introducción

Este capítulo de la memoria pretende servir como contextualización del marco de trabajo y del propio desarrollo de este trabajo de fin de máster, de ahora en adelante, TFM. En este documento se expondrán las hipótesis iniciales, desarrollos y resultados obtenidos a lo largo de las tareas a realizar dentro del proyecto.

El presente TFM se desarrolla dentro del proyecto europeo llamado *High-Quality Data-Driven Services for a Digital Built Environment towards a Climate-Neutral Building Stock*, o su acrónimo DigiBUILD [2]. DigiBUILD es un proyecto en el que colaboran un consorcio de empresas europeas bajo la supervisión de la Agencia Ejecutiva del Clima, Infraestructura y Medioambiente Europea (CINEA) [3], supeditada a la Comisión Europea (EC) [4].

Este proyecto se ubica dentro del programa de innovación e investigación *Horizon Europe*, en el sector del uso de energía [5]. La convocatoria del proyecto es HORIZON-CL5-2021-D4-01 o *Efficient, sustainable and inclusive energy use* y su tema HORIZON-CL5-2021-D4-01-03 o *Advanced data-driven monitoring of building stock energy performance*. El proyecto tiene una duración estimada de 36 meses, habiendo comenzado el día 1 de junio del 2022 y se espera que finalice en junio del 2025, aunque las tareas que conciernen al TFM tiene una fecha de finalización anterior.

1.1. Hipótesis

Actualmente, los edificios de nueva generación están siendo diseñados atendiendo a las posibles medidas de digitalización de los mismos, teniendo la infraestructura ya instalada o teniendo en cuenta su potencial instalación. Por otra parte, algunos de los edificios que no son de nueva o reciente construcción también están sufriendo la transformación hacia la monitorización y el control de los procesos y consumos propios del edificio. Esta transformación se conoce como digitalización y consiste en la instalación de sensores o componentes electrónicos que permitan algún tipo de monitorización o gestión remota y automática del edificio, automatización

de procesos dentro del mismo o meramente la extracción de datos para un control y posible identificación de problemas. El fin último de esta digitalización, en gran parte de los casos, es la eficiencia energética, con una monitorización de los recursos de los edificios y mejorando u optimizando la gestión y el mantenimiento de los mismos.

DigiBUILD tiene como objetivo la transformación de edificios digitales e inteligentes que integren y gestionen de forma propia sus fuentes de datos de una forma proactiva. Esto es un paradigma de organización diferente al que se ha empleado tradicionalmente, conocido como "enfoque de silo". El enfoque de silo consiste en tener sistemas o agrupaciones de los mismos con bases de datos internas, y la interacción entre estos sistemas independientes es complicada debido a su estructura. Este enfoque es problemático en sistemas u organizaciones de gran tamaño, dado que los equipos independientes no pueden trabajar y realizar operaciones de forma rápida y eficaz sobre los datos dentro del dominio de otro equipo debido a la interacción reducida entre ellos [1].

El proyecto en el que trabajamos hace uso de datos de alta calidad y servicios de edificios digitales de nueva generación. Además, desarrolla un entorno inclusivo de intercambio de datos entre las diferentes partes interesadas para el diseño y desarrollo de servicios orientados al usuario final, alineado con las propuestas de la Nueva Iniciativa *Bauhaus* Europea. La Nueva Iniciativa *Bauhaus* Europea fue lanzada por la Comisión Europea en septiembre de 2020. Se inspira en la famosa Escuela *Bauhaus* que funcionó en Alemania entre 1919 y 1933, que era una escuela de arte y diseño que buscaba integrar las artes y la tecnología para hacer frente a los retos sociales y económicos de la época. La Nueva Iniciativa *Bauhaus* Europea pretende combinar diseño, sostenibilidad, accesibilidad, asequibilidad y atractivo estético en la creación de espacios y productos. La idea es fomentar la innovación y la creatividad para hacer frente a los retos contemporáneos, como el cambio climático y la transición a una economía más ecológica [6].

DigiBUILD proporciona diferentes herramientas abiertas accesibles desde la nube para la transformación de estos edificios "silo" en edificios digitales, interoperables e inteligentes. Estas herramientas mejoran la toma de decisiones en lo relativo a la supervisión y evaluación del rendimiento, la planificación de infraestructura de los edificios, la formulación de políticas y la reducción de riesgos en las inversiones [1]. Sobre este *framework* se crean servicios y análisis de datos que emplean inteligencia artificial (IA) y se desarrollan gemelos digitales que se alimenten de datos de alta calidad para proporcionar de forma transparente, una mejor toma de decisiones informadas y el intercambio de información dentro del entorno construido y el sector de la construcción.

Como conclusión de este pequeño apartado, se especifica que la hipótesis del trabajo es que, mediante la digitalización y la integración de sistemas de monitorización y gestión remota que se han implementado, se pueden optimizar los procesos y el consumo energético en edificios tanto nuevos como existentes. Se plantea que esta transformación hacia edificios inteligentes, que integran y gestionan sus datos de manera proactiva y colaborativa, permitirá mejorar significativamente la eficiencia energética y la sostenibilidad en el sector de la construcción.

La Figura 1 muestra el logo creado por el consorcio para el proyecto [7] en el que se trabaja en este TFM.

Figura 1: Logo de DigiBUILD

1.2. Objetivos

A continuación, se expondrán las diferentes tareas propuestas para cumplir en este TFM, siendo éstos soportados por el proyecto DigiBUILD, como se acaba de mencionar.

- Definir todas las conexiones con las diferentes interfaces y puntos de extracción de datos de los pilotos y otras fuentes de datos externas para la extracción de datos estáticos y datos en tiempo dinámicos, así como su tratamiento, formateo e ingestión.
- Aplicar metodologías de calidad de datos para evaluar los datos dinámicos adquiridos.
- Utilizar un *Data Warehouse* (DWH) de mejora donde se almacenarán los datos dinámicos extraídos de los pilotos mencionados anteriormente y generación de propuestas.
- Desarrollar una API como interfaz de compartición de datos en la capa de servicios de predicción basados en inteligencia artificial (IA) y en los datos almacenados en el DWH, sirviendo estos como una de las aplicaciones de los objetivos anteriores. El desarrollo de los propios servicios no está incluido en este trabajo.

Cada uno de estos objetivos no se considera como un hito aislado, sino que se pretende la construcción del flujo completo de ingestión de datos provenientes de una gran cantidad de fuentes muy heterogéneas y poder obtenerlos a través de una interfaz única y uniforme, siendo almacenados de forma persistente y manteniendo unos estándares de calidad altos.

1.3. Estructura del documento

En esta sección se describe la organización y el contenido de los capítulos y apartados que componen el documento:

- **Capítulo 2 (Contexto)**: En este capítulo de la memoria se exponen los diferentes campos de conocimiento necesarios para la comprensión y desarrollo del trabajo, así como la organización de los objetivos descritos anteriormente y su distribución dentro de la arquitectura de DigiBUILD. Se comienza mencionando la herramienta de creación de los procesos de tratamiento de datos empleada. Después, se explican las consideraciones generales respecto al *big data* para nuestro TFM. Por último, se muestra un diagrama conceptual de

las tareas en forma de diagrama de bloques, donde se explican las relaciones y los flujos de los bloques asociándolas con las tareas desarrolladas en el TFM.

- **Capítulo 3 (Arquitectura del sistema)**: En el siguiente capítulo, se desarrollan conceptos de arquitectura en sistemas *big data* y, en concreto, la arquitectura que se ha propuesto para el sistema de DigiBUILD, haciendo especial hincapié en las tareas que hemos desarrollado.
- **Capítulo 4 (Extracción de datos dinámicos)**: En el tercer capítulo del documento se encuentra el grueso del trabajo. Aquí se muestran los diferentes procesos de conexión a fuentes de datos, su extracción, la transformación y su almacenaje en el DWH.
- **Capítulo 5 (Metodologías de calidad de datos)**: En este capítulo se explican los conceptos más importantes sobre la calidad de los datos, las diferentes métricas seleccionadas para la evaluación de los datos en el proyecto y su implementación en el sistema.
- **Capítulo 6 (Base de datos para el almacenamiento de datos dinámicos contextuales: Data Warehouse)**: Se explican los conceptos más relevantes del DWH que se ha utilizado, así como el manejo *big data* en la definición de la base de datos y la optimización de *queries* para reducir la carga de procesamiento y el volumen de datos. Además, se proporciona alguna pincelada sobre la importancia de los metadatos en el proyecto y su sincronización con los datos dinámicos. Por último, se hace mención al desarrollo de una API que hace uso del DWH para proporcionar acceso a varios dervicios que generan predicciones para diferentes pilotos.
- **Capítulo 7 (Conclusiones)**: Finalmente se procede a mencionar las diferentes conclusiones a las que se ha llegado con el desarrollo de todas las tareas mencionadas.

Capítulo 2

Contexto del proyecto

En esta sección, se procede a la definición de los conceptos más relevantes relativos al desarrollo del proyecto que permitan entender su desarrollo, así como a la explicación de la organización de los diferentes objetivos definidos anteriormente en el contexto del TFM y de la arquitectura de DigiBUILD.

2.1. Big Data

Big data se refiere a colecciones de datos de gran tamaño que exceden las capacidades de los sistemas de procesamiento de datos tradicionales [8]. El interés en *big data* ha crecido exponencialmente en los últimos tiempos, estimulado por avances en comunicaciones y mejoras significativas en los sistemas para procesar y almacenar datos. En este periodo de expansión digital, se ha creado un volumen de datos sin parangón originado por distintas fuentes, incluyendo las redes sociales, transacciones electrónicas, dispositivos del Internet de las Cosas (IoT), entre otras. La habilidad para guardar, procesar utilizar este masivo volumen de datos ha marcado el inicio de una nueva era en decisiones informadas por datos, innovación y adaptación de servicios, resaltando su crucial importancia en una amplia gama de áreas como la ingeniería y administración de recursos.

Las cinco Vs o dimensiones del *big data* delinean las características esenciales que constituyen estos intrincados conjuntos de datos. Estas dimensiones son: Volumen, Velocidad, Variedad, Veracidad y Valor. Volumen alude a la cantidad de datos producidos a cada instante. Velocidad hace referencia a la rapidez con la que estos datos son generados, recolectados y analizados. Variedad destaca la amplia gama de tipos y fuentes de datos, abarcando desde formatos estructurados hasta no estructurados, como textos, imágenes, vídeos, etc. Veracidad señala la fiabilidad y exactitud de los datos, un elemento crucial para la elaboración de análisis confiables. Finalmente, Valor se centra en la habilidad de transformar estos datos en información útil y práctica, representando el fin último de cualquier proyecto de *big data*. Cada uno de estos aspectos introduce desafíos y posibilidades específicas, impulsando el avance de tecnologías y métodos particulares

para el manejo eficaz de los *big data* [9].

La importancia de las aplicaciones de análisis de *big data* radica en su capacidad para abordar y aprovechar estas dimensiones. Al gestionar el Volumen, permiten a las organizaciones procesar y analizar grandes conjuntos de datos que de otro modo serían inmanejables. En cuanto a la Velocidad, estas aplicaciones pueden capturar y analizar datos desde en tiempo real hasta frecuencias como de un día, siendo escenarios completamente diferente, lo que resulta crucial para tomar decisiones rápidas basadas en información actualizada. El hecho de tener frecuencias de generación de datos tan diferentes pone mucho más en valor la implementación de la dimensión de la velocidad con un enfoque *big data* en nuestro caso de aplicación. La Variedad se aborda permitiendo la integración y el análisis de datos procedentes de múltiples fuentes y formatos, desde estructurados a no estructurados. La Veracidad se garantiza mediante herramientas y algoritmos que limpian y verifican los datos, asegurando análisis precisos. Por último, el Valor surge de la transformación de grandes volúmenes de datos en información procesable y decisiones informadas, maximizando así el impacto y la eficiencia de la organización o de la aplicación [9].

2.2. Diagrama conceptual

En la figura 2 podemos ver un diagrama conceptual y general de los diferentes bloques en los que agrupamos las tareas que se desarrollan en este TFM, así como los bloques que se relacionan directamente con éstos. El objetivo será explicar desde un punto de vista más general la organización, flujos y relaciones de los diferentes objetivos de nuestro proyecto y su implementación.

En la parte inferior del diagrama, se distinguen las fuentes de datos, las cuales pueden clasificarse como internas o externas al proyecto. Las fuentes internas abarcan datos estáticos, como metadatos, y dinámicos, que comprenden información temporal de variables físicas y lógicas presentes en los pilotos o demostradores del proyecto. Por otro lado, las fuentes externas, que comprenden APIs y otros espacios de datos ajenos al proyecto, tales como pronósticos y registros meteorológicos históricos, además de otros espacios de datos preexistentes pertenecientes a organismos públicos europeos.

En el siguiente bloque del diagrama, encontramos los procesos ETL, que se abordarán de forma más extensa en el capítulo 4 de la memoria. Estos procesos conforman el grueso de nuestro trabajo y su función es la siguiente: se establecen conexiones con los diversos flujos de datos en cada uno de los pilotos existentes, a través de interfaces de bases de datos, APIs u otros protocolos de comunicación, considerando tanto su formato como su disponibilidad. Cabe destacar la amplia variedad de protocolos de acceso a estos datos, así como la gran cantidad de variables físicas y puntos medidos con los que se trabaja. Además, se aplicarán transformaciones a dichos flujos para dotar a los datos de un formato homogéneo y uniforme, así como de limpiar y filtrar los datos no relevantes para el sistema. Estos datos homogéneos y tratados servirán como entradas de los bloques *Data Lake* y Gemelos Digitales, cuyo desarrollo no se corresponde a las tareas de este trabajo.

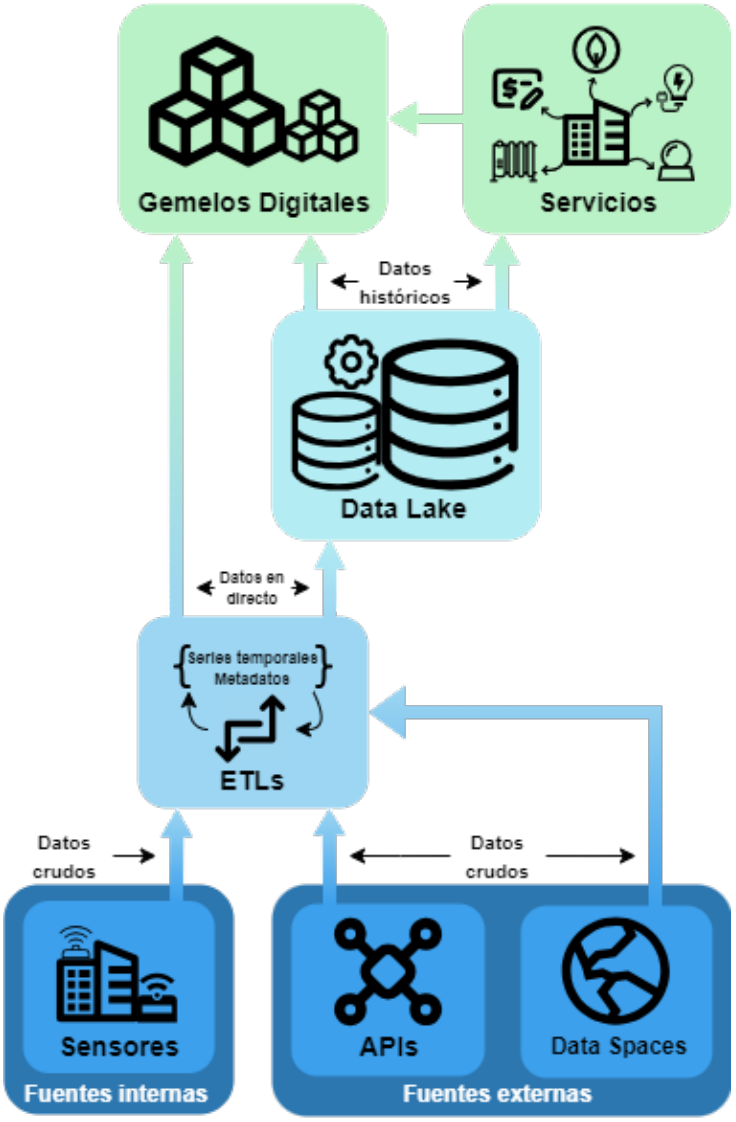

Figura 2: Diagrama conceptual del TFM

En estas ETLs se extraen datos dinámicos y, en los casos en los que sea posible, datos estáticos o metadatos. Los metadatos aportan información muy variada y diferente, como por ejemplo sobre la estructura física o lógica en la que se divide el piloto, unidades de medida de las variables de los datos dinámicos, localización, estado de los dispositivos de medida, etc. Los datos dinámicos son los valores de las magnitudes que se miden en los pilotos o fuentes externas y serán almacenados en el próximo bloque, el *Data Lake*, más concretamente en el Data Warehouse (DWH).

A continuación, encontramos el bloque *Data Lake*. Como ya hemos mencionado, este bloque se alimenta de los datos en dinámicos que producen las ETLs. Su función principal es almacenar datos, por lo que principalmente está compuesto de varias bases de datos. La más importante de ellas, y la que nosotros implementamos y poblamos es el *Data Warehouse*, explicado en el capítulo 6, y que consiste en el almacén de los datos dinámicos, es decir, de medidas obtenidas de los diferentes pilotos. El *Data Lake* tiene como salida los datos históricos que almacena para

los bloques de Servicios y de Gemelos Digitales.

Finalmente, se encuentra el bloque de servicios. Estos servicios consumen los metadatos y datos que se almacenan en el *Data Lake* provenientes tanto de fuentes internas como externas. Estos servicios emplean algoritmos de inteligencia artificial y se basan en la utilización de los datos extraídos en los pasos previos del flujo. Los servicios están diseñados *ad-hoc* para cada piloto, aunque cada servicio se comparta entre diferentes pilotos. El contenido relativo a los servicios que concierne a este trabajo de fin de Máster se centre únicamente en el desarrollo de una API para hacer uso de los mismos. Esta interfaz se describe en la sección 6.6 de la memoria. Esta API sirve como pequeño repositorio de modelos de IA para la predicción de variables energéticas de forma ajena a este TFM. Las salidas de los diferentes servicios se emplean tanto como entradas para los diferentes gemelos digitales desarrollados como resultados finales para los usuarios en una interfaz gráfica (GUI) cuyo desarrollo tampoco corresponde a los objetivos de este trabajo.

En esta sección de la memoria se ha mencionado el concepto de piloto, pero sin explicar qué es. Un piloto se puede corresponder con uno o varios edificios, sistemas energéticos o infraestructuras dedicadas a diferentes ámbitos, como podría ser un edificio de oficinas, un edificio educativo o una fábrica. En DigiBUILD se cuenta con 10 pilotos diferentes repartidos por toda Europa. Cada uno de ellos es gestionado por una de las empresa del consorcio. Estos pilotos se encuentran sensorizados y exportan y/o almacenan datos tanto de variables energéticas (como puede ser la energía que produce una caldera, la potencia que genera una instalación de placas fotovoltaicas, el consumo de un cargador de coches eléctricos, la temperatura de unas oficinas o la potencia que se consume en un enchufe) como variables lógicas (como puede ser la apertura de puertas y ventanas).

En la figura 3 podemos ver la distribución de los pilotos y su organización atendiendo a los objetivos de los servicios definidos para cada uno de ellos. De forma general, se organizan en tres *clusters* atendiendo a la orientación de los servicios que ofrecen, siendo el *cluster* 1: desempeño de los edificios, *cluster* 2: edificios vs. la gestión óptima de la infraestructura y *cluster* 3: políticas y finanzas.

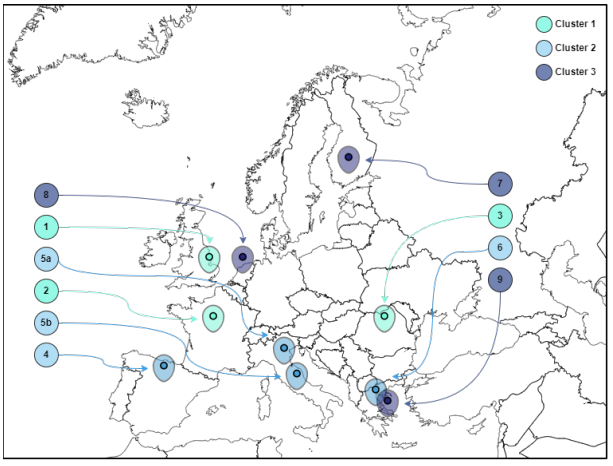

Figura 3: Pilotos del proyecto

2.3. Herramientas software empleadas

2.3.1. Pentaho

Pentaho [10] es una *suite* de programas libres enfocados en el *business intelligence* (BI). El BI consiste en la extracción y generación de información y conocimiento a partir de datos de una empresa para mejorar y facilitar la toma de decisiones dentro de la misma. Se emplean datos extraídos a lo largo de todo el proceso productivo para generar información sobre el funcionamiento y situación de la empresa, así como toma de decisiones anticipadas o respaldo a la hora de la toma de decisiones.

Se ha decidido utilizar *Pentaho* como la herramienta *big data* del proyecto debido varios motivos. El más importante de ellos es la multitud de tareas que un único *software* permite integrar: desde la extracción y tratamiento de los datos, hasta su almacenamiento, migración y sincronización. Otra razón es la facilidad de uso que tiene gracias a su interfaz gráfica y modelo de creación de ETLs basado en diagramas de bloques, haciendo que la curva de aprendizaje para un uso extensivo del programa sea muy rápida.

Además, actualmente una primera versión del *data warehouse* se encuentra desplegado *Pentaho Server*, que es una herramienta de la *suite* con funcionalidades que pueden aportar avances a las tareas que hemos desarrollado como parte de este proyecto. Entre estas tareas destacan: la integración de las ETLs, creación de informes personalizados con posibilidad de exportación a diferentes formatos, análisis OLAP de grandes volúmenes de datos de forma muy rápida e interactiva a través de cubos *Mondrian*, generación de *dashboards* y visualizadores especializados, entre otras.

Dentro de esta *suite* se encuentra *Pentaho Data Integration* (PDI) o *Kettle*. Esta es una aplicación de análisis de *big data* que facilita la gestión integral de datos, permitiendo a sus usuarios aprovechar el valor de sus datos para obtener información, tomar decisiones y obtener ventajas competitivas, es decir, para aplicar BI. Esta aplicación destaca en la gestión eficaz de estas dimensiones, cubriendo el tratamiento de todas ellas. *Kettle* facilita la extracción, transformación y carga (ETL) de grandes volúmenes de datos, abordando así el aspecto Volumen. Su capacidad para procesar datos dinámicos afecta directamente a la dimensión Velocidad. En cuanto a la Variedad, *kettle* admite una amplia gama de tipos y fuentes de datos, desde las bases de datos tradicionales hasta los formatos de datos modernos, como las redes sociales y los flujos de datos en directo. La veracidad se ve reforzada por sus potentes funciones de limpieza y validación de datos, que garantizan que sólo se utilicen para el análisis datos precisos y de alta calidad. Por último, al facilitar la transformación de datos en bruto en información significativa, *Kettle* permite a las organizaciones extraer el máximo valor de sus datos, impulsando decisiones estratégicas y operativas basadas en pruebas sólidas.

Dentro de DigiBUILD, nuestro trabajo se centra principalmente en asentar las bases para la aplicación del BI, esto es, la obtención, formateo y almacenaje de un gran volumen de datos provenientes de una gran cantidad de interfaces y fuentes.

2.3.2. PostgreSQL

PostgreSQL es una base de datos relacional de código abierto conocida por su robustez, escalabilidad y cumplimiento del estándar ACID [11] (Atomicidad, Consistencia, Aislamiento, Durabilidad). Ofrece características avanzadas como el soporte para consultas complejas, la extensibilidad, y la capacidad de gestionar grandes volúmenes de datos con eficiencia mediante técnicas como la indexación y el particionado de datos, lo cual es perfecto y complaciente con su aplicación en sistemas *big data*.

En este trabajo se ha empleado para la definición de las diferentes bases de datos de los pilotos, implementando herencias en la variable temporal de las telemetrías almacenadas. Esto permite gestionar de manera más eficiente los grandes volúmenes de datos generados. En secciones posteriores se explica más en profundidad su uso y los diferentes desarrollos.

2.3.3. Librerías de Python

Finalmente, mencionamos de forma anticipada que para la programación de los diferentes *scripts* usados para diferentes capítulos y secciones se ha empleado *Python* como lenguaje de programación. Además, para el tratamiento de los datos empleados, tanto de las fuentes externas e internas, se ha utilizado la librería Pandas, permitiendo una manipulación eficiente y estructurada de grandes volúmenes de datos de telemetrías históricas y meteorológicos. Por último, para los desarrollos relativos a la API de compartición de datos de los servicios se han usado las librerías de *FastAPI* para la propia construcción de la API, *Pydantic* para validar, deserializar datos y definir esquemas de datos con anotaciones de tipo, *Uvicorn* como servidor ASGI y *SQLAlchemy* para las interacciones de la API con una base de datos o definir estructuras de datos.

Capítulo 3

Arquitectura del sistema

En este apartado de la memoria se procede a la descripción de la arquitectura del sistema propuesto en DigiBUILD.

Para la definición de la estructura del sistema que se desarrolla en el proyecto existe una tarea específica en la que no se participa y ha finalizado previamente al comienzo de este TFM [12]. Es por esto que es expondrán las ideas más importantes que permitan una descripción clara de la arquitectura haciendo énfasis en la gestión *big data* de ésta y dónde se encuentran los desarrollos de nuestro proyecto.

En la figura 4 podemos ver la arquitectura completa del sistema [12]. Además, se resaltan con un encuadre rojo las partes que se corresponden con tareas desarrolladas en este trabajo.

En [12] se ha llevado a cabo una revisión de proyectos y trabajos existentes para la definición de la arquitectura de sistemas *big data*. A partir de ello, en la estructura de referencia de DigiBUILD se han definido las siguientes capas funcionales (Figura 4 a la derecha) de forma general:

- **Servicios de interoperabilidad y alta calidad de datos:** Esta capa se hace cargo de la recolección, tratamiento, interoperabilidad semántica y almacenaje de los datos obtenidos de los pilotos.
- **Servicios de IA:** En esta capa se engloban las funcionalidades de desarrollo e implementación de servicios energéticos basados en ML (*machine learning*) y DL (*deep learning*) para sistemas de apoyo de decisiones, procedimientos de simulación y extracción de valor de negocio de los datos de la capa anterior.
- **Gemelos digitales:** En esta última capa se recoge una *suite* de gemelos digitales, que permiten una mejor planificación de la infraestructura de los edificios para el diseño de futuros edificios y sus procesos de construcción, así como de la gestión y operación de edificios y distritos energéticamente eficientes.

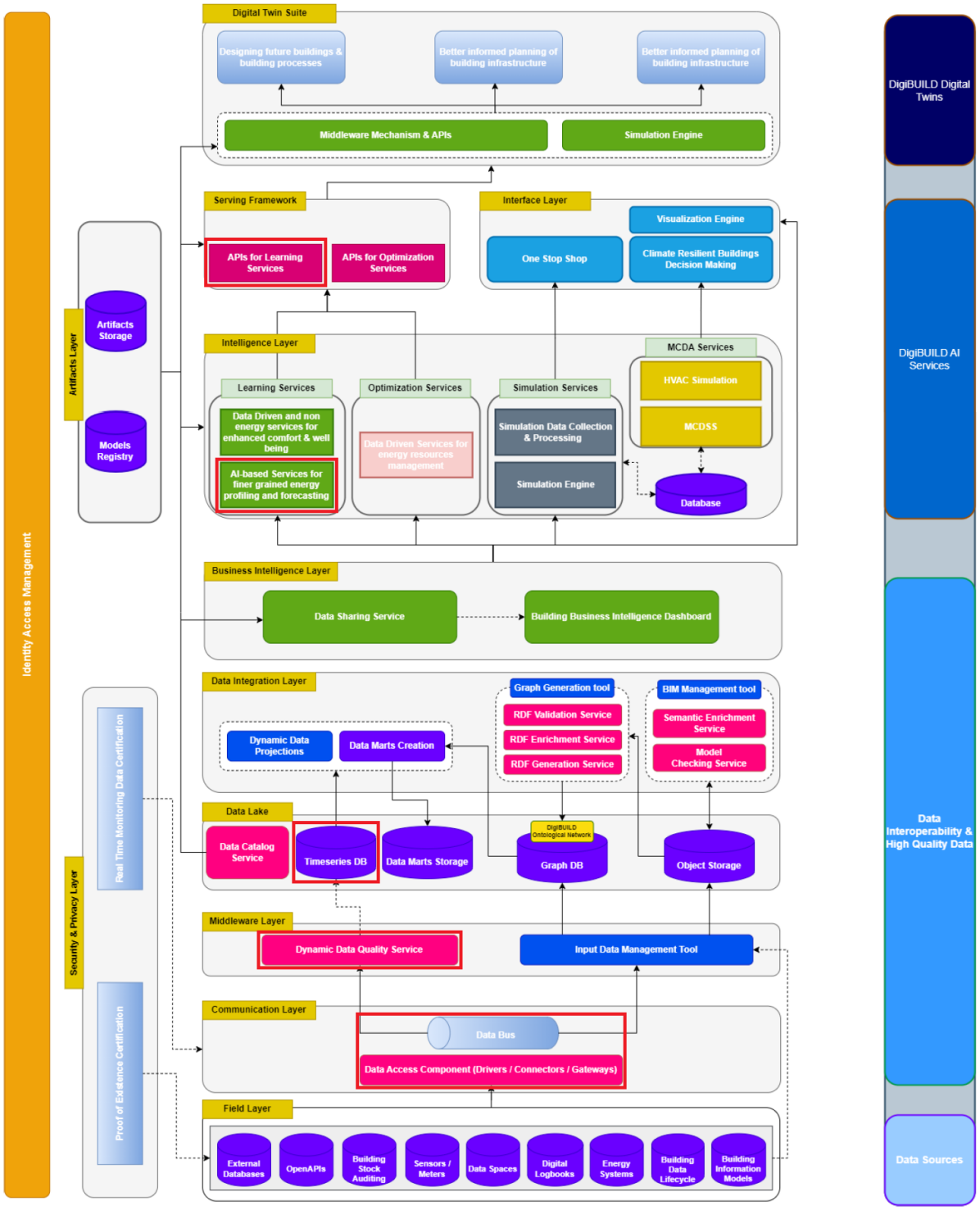

Figura 4: Arquitectura de DigiBUILD

3.1. Arquitecturas de referencia y de proyectos europeos para gestión *big data*

En esta sección de la memoria se van a exponer ejemplos concretos de iniciativas ya existentes para la gestión de espacios de datos en el entorno europeo en los que se ha basado el consorcio para el diseño de la arquitectura de referencia para DigiBUILD.

3.1.1. Arquitecturas de referencia estándares de la EU

La plataforma de DigiBUILD se construye siguiendo marcos de trabajo estándares de plataformas *cloud-data* e iniciativas de datos enfocados hacia espacios de datos de edificios energéticamente eficientes. Las iniciativa europeas revisadas y empleadas como base para *DigiBUILD* han sido las proporcionadas por la Asociación Internacional de Espacios de Datos (IDSA), GAIA-X y FIWARE.

El Modelo de Arquitectura de Referencia de la Asociación Internacional de Espacios de Datos (IDS-RAM) [13] busca facilitar un intercambio de datos seguro y promover la soberanía de datos en espacios virtuales. Este modelo, basado en el principio de soberanía de datos, permite a los propietarios controlar la compartición y gestión de sus datos. Se compone de cinco capas (de negocio, funcional, de proceso, de información y de sistema) que cubren desde el rol de los usuarios hasta el marco técnico necesario para el intercambio de datos. Dentro de este marco, se destacan cuatro roles clave: Propietario de Datos, Proveedor de Datos, Consumidor de Datos y Usuario de Datos, que marcan el flujo del intercambio de datos desde su origen hasta su destino final. Las operaciones de intercambio incluyen transferencia de datos, publicación de metadatos, registro de transacciones y uso de aplicaciones y vocabularios de datos, apoyadas por servicios como proveedores de servicios, *brokers*, y proveedores de vocabulario. Este enfoque asegura un intercambio de datos seguro y fomenta un ecosistema de datos abierto y eficiente, esencial para el avance de la economía digital.

La Arquitectura de Referencia *GAIA-X* [14] establece una infraestructura de datos abierta que se adhiere a normas universales y permite compartir datos y servicios de forma segura entre proveedores y consumidores. Fusiona un ecosistema de datos, que comprende los conjuntos de datos que deben compartirse, con un ecosistema de infraestructuras, que engloba todos los sistemas y procesos relacionados, en una plataforma de datos unificada. Esta integración se logra mediante servicios de federación, que son fundamentales para crear espacios de datos interoperables, conformes y fiables. Estos servicios de federación se clasifican en cuatro grupos principales: Identidad y confianza (gestión de identidades federadas e interacciones seguras), Catálogo federado (que permite el registro y descubrimiento de ofertas de datos), Intercambio soberano de datos (definición de normas de intercambio de datos para garantizar la soberanía de los mismos) y Cumplimiento (que abarca los procesos legales, de derechos, obligaciones y certificación). En el corazón de *GAIA-X*, la interacción entre proveedores y consumidores de datos se ve facilitada por un servicio de acuerdo de datos, que orquesta el proceso de intercambio de datos, garantizando que ambas partes estén informadas de forma coherente sobre las transacciones

y se adhieran a las políticas y restricciones de uso de datos acordadas.

Finalmente, *FIWARE* [15] ofrece una plataforma de código abierto, que utiliza estándares abiertos y componentes portátiles, destinada a facilitar la creación de espacios de datos empresariales inteligentes. El elemento central de esta plataforma es el *FIWARE Context Broker*, un componente único responsable de las actualizaciones del sistema, el acceso al estado contextual, el análisis, el procesamiento, la visualización y la publicación de los datos. El funcionamiento de esta plataforma se articula a través de la Arquitectura de Referencia Abierta *FIWARE*, que se estructura en tres capas. La capa fundacional engloba la integración de interfaces con el Internet de las Cosas (IoT), Robótica y sistemas de terceros para la recogida de datos y actuación de comandos. La capa intermedia se dedica a la gestión del núcleo a través del *Context Broker*, centrándose en el desarrollo y la gestión de aplicaciones contextuales. La capa superior se ocupa del análisis, procesamiento y visualización de datos. Esta configuración permite que los conjuntos de datos procesados se compartan con terceros a través de un portal mejorado *Comprehensive Knowledge Archive Network* o CKAN, que es una plataforma de código abierto diseñada para la gestión de datos abiertos y la creación de portales de datos, mostrando la capacidad de *FIWARE* para apoyar los ecosistemas inteligentes a través de la integración y utilización eficiente de los datos.

3.1.2. Arquitecturas de referencia en proyectos europeos

En esta sección se hará mención a proyectos europeos que se engloban dentro del mismo programa de investigación de la EC que DigiBUILD H2020 *Horizon Europe* [4]. La arquitectura de los sistemas desarrollada en estos proyectos se basa en revisiones de las arquitecturas estándares de la Unión Europea previamente explicadas y también han sido tomadas como referencia para la propia definición de la arquitectura del propio sistema de DigiBUILD. Esta lista de proyectos es:

- ***MATRYCS*** o *Modular Big Data Applications for Holistic Energy Services in Buildings* [16, 17].
- ***BD4NRG*** o *Big Data for Next Generation Energy* [18, 19].
- ***BIMSPEED*** o *Harmonised Building Information Speedway for Energy-Efficient Renovation* [20, 21].
- ***BIM2TWIN*** o *Optimal Construction Management & Production Control* [22, 23].
- ***BaaS*** o *BaaS – Building as a Service (Ecosystem)* [24, 25].
- ***mySMARTLife*** o *Transition of EU cities towards a new concept of Smart Life and Economy* [26, 27].

3.2. Arquitectura de referencia de DigiBUILD

Esta sección de la memoria se dedicará a la explicación de la arquitectura propuesta para el sistema de DigiBUILD, enfocada, como se ha explicado anteriormente, a la gestión *big data*,

pero además ofrecer un espacio de datos de edificios que provea a las diferentes partes interesadas de un intercambio de datos transparente. Su definición ha sido originada por las relaciones y dependencias de los diferentes paquetes de trabajo en los que se divide el proyecto y las tareas que compone a cada uno.

La gestión *big data* de la arquitectura permite la integración de modelos de análisis de datos (conocidos internamente como *data marts*) y servicios que permiten la interacción entre diferentes tipos de usuarios finales. Esta arquitectura está orientada a micro-servicios y se divide en capas, permitiendo que las tareas se repartan y se desplieguen de forma independiente. Las capas que mencionamos en la Figura 4 y sus funciones son las siguientes:

- **Capa de servicios de interoperabilidad y calidad de datos:** Esta parte de la arquitectura se encuentra sobre las fuentes de datos. Sus funciones principales son la recolección de los datos, llevar a cabo un pre-procesado de los mismo, así como su almacenamiento, enriquecimiento y generación de *data marts*.
    - **Capa de campo:** Esta capa está formada por todas las fuentes de datos de los diferentes pilotos (Figura 3) que proveen de datos a la plataforma. Estas fuentes de datos pueden ser de diferentes tipos como sistemas de energía, medidores y sensores, bases de datos externas, APIs o espacios de datos externos.
    - **Capa de comunicación:** Esta capa se encarga de la extracción de los datos que genera la capa de campo mediante la implementación de las conexiones apropiadas.
    - **Capa *middleware*:** En esta capa se llevan a cabo la aplicación de los servicios referidos a la calidad del dato.
    - ***Data lake*:** Es la capa donde se ubican todos los tipos de almacenamiento de datos del sistema. Estos almacenes son de diferentes tipos atendiendo a la naturaleza y características de los datos que albergan.
    - **Capa de integración de los datos:** Se encarga de la aplicación de algoritmos y tratamientos de post-procesado a los datos que han sido previamente almacenados.
    - **Capa de inteligencia de negocio:** Lleva a cabo la transmisión de la información a las capas superiores para permitir las consultas federadas sobre el *data lake*.
    - **Capa de seguridad y privacidad:** Añade las comprobaciones y medidas necesarias para ofrecer un marco de trabajo seguro a nivel físico.
- **Capa de servicios de IA:** En la siguiente etapa de la arquitectura podemos encontrar las capas que se dedican a la creación de una *suite* de servicios de IA basados en datos enfocados a los usuarios de los pilotos y los actores interesados. Su objetivo final es la promoción de la eficiencia energética, la mejora del confort y el bienestar de los ocupantes y promover buenas prácticas en edificios sostenibles.
    - **Capa de inteligencia:** Ésta es la capa donde se almacenan los diferentes algoritmos y modelos. Se encuentran clasificados en función del tipo de servicio que implementen.
    - **Capa de artefactos:** La capa de artefactos almacena los resultados generados en la capa de servicios.
    - ***Framework* de servicio:** Es el marco de trabajo encargado de exponer y hacer accesibles los modelos de predicción para hacer uso de los mismos.

- ***Suite* de gemelos digitales:** Esta capa es una capa similar a la capa de aplicación del modelo TCP/IP. Engloba todas las herramientas para el desarrollo de los diferentes gemelos digitales de los pilotos del proyecto. Estos gemelos se consideran claves para la transformación digital y son la contraparte digital del sistema físico.

La ubicación de las tareas que desarrollamos en base a las diferentes capas de la arquitectura y su definición y atribución de responsabilidades en nuestro trabajo se realiza de la siguiente forma. Las tareas que se desarrollan en el **capítulo 4** están relacionadas con la **capa de comunicación**. La aplicación de métricas de calidad de datos del **capítulo 5** son la responsabilidad asignada a la **capa de *middleware***. En el **capítulo 6** se exponen los desarrollos correspondientes con el componente *Data Warehouse* de la **capa del *Data Lake***, si bien se dedica la **sección 6.6** a explicar el desarrollo de una interfaz para la compartición de datos y así poder acceder a los servicios de predicción (**capa de *framework* de servicio**).

Capítulo 4

Extracción de Datos Dinámicos

En este capítulo de la memoria procederemos a describir las diferentes metodologías desarrolladas para la ingestión de datos dinámicos de los diferentes pilotos y fuentes externas. En relación con lo expuesto en el capítulo 2, las tareas desarrolladas en este capítulo se corresponden con las capas de campo y de interoperabilidad de datos que podemos observar en la figura 4. Sus funciones comprenden los siguientes objetivos: conexión con un gran número interfaces empleando diferentes protocolos, extracción de datos, preprocesado y formateo de los mismos mediante procesos de Extracción, Transformación y Filtrado (ETL).

Como primer paso para poder llevar a cabo el desarrollo de este objetivo, se construye un inventario de datos junto a desarrolladores de otra tarea relacionada dentro del proyecto [28]. Para poder llevar a cabo este inventariado de datos se siguen los diferentes pasos:

- Identificación de los servicios y gemelos digitales que se implementan teniendo en cuenta su definición en el caso de cada piloto.
- Identificación de los sistemas de energía que se necesitan en cada servicio o gemelo digital.
- Caracterización de esos sistemas, teniendo sus datos, descripciones técnicas y de los componentes de los forman.
- Selección de los *datasets* relevantes de cada piloto atendiendo a los requisitos de los servicios y gemelos digitales.
- Generación del propio inventario, incluyendo la información más importante, como la descripción de los *datasets*, disponibilidad de los datos, volumen anual que almacenar, velocidad, frecuencia de muestreo y de publicación, interfaces de extracción, veracidad y fiabilidad.

Interfaces de extracción de datos dinámicos

Una vez definido el inventario de datos, se seleccionan los *datasets* más relevantes. Para poder trabajar con estos datos, los responsables técnicos de los pilotos desarrollan las diferentes

interfaces de extracción de datos que sean necesarias en cada caso. La Tabla 1 [28] recoge un resumen de los diferentes *datasets* y los diferentes protocolos necesarios para su extracción.

Piloto	**Detalles de interfaz**			
	Dataset	***Protocolo***	***Formato***	***Frecuencia***
01	BMS[1]	MQTT[3] [29]	Texto	5 min.
	EMS[2]			5 min.
	Control de luz			Eventos
	Control de acceso			Eventos
	Ocupación			Eventos
02	Ethera	Nemocloud API[4]	JSON[5] [30]	5 min.
	Ellona	Ellonasoft API[4]		5 min.
	Wattsense	Wattsense API[4]		1 min.
03	3PhaseMeters	InfluxDB [31]	CSV[6] [32]	15 min. y 1 h.
04	BEMS[7]	SFTP[9] [33]	CSV[6] [32]	15 y 30 min.
	DEMS[8]			15 y 30 min.
05a	Estaciones de carga	API[4]	JSON[5] [30]	15 min.
	PV[10]			10 min.
	Datos edificio			5 y 20 min.
	EV[11]			10 min.
05b	PV[10]	SolarEdge API[4]	JSON[5] [30]	1 min.
	Confort	JotMotiqa API[4]		15 min.
	Consumo energético	MQTT[3] [29]	Texto	1 min.
	Ocupación	Google Calendar API[4] [34]	CSV[6] [32]	Eventos
	Consumo energético	Google Drive API[4] [35]		15 min.
06	Datos edificio	API[4]	JSON[5] [30]	30 seg.
07	BEMS[7]	SFTP[9] [33]	CSV[6] [32]	15 min.
08	Datos edificio	MQTT[3] [29]	Texto	1 min.
		Netatmo API[4]	JSON[5] [30]	5 min.
09	BMS[1]	PostgreSQL [36]	Texto	5 a 30 min.

Tabla 1: Interfaces para la extracción de datos dinámicos

A continuación, se dará una breve explicación de los protocolos abiertos que aparecen en la tabla anterior. MQTT o *Message Queuing Telemetry Transfer* es un protocolo de intercambio de mensajes a través de colas gestionadas por un *broker* que se emplea para permitir la comunicación

[1]BMS: Building Management System - Sistema de Gestión del Edificio
[2]EMS: Energy Management System - Sistema de Gestión de la Energía
[3]MQTT: Message Queuing Telemetry Transport - Transporte de Mensajes de Telemetría en Colas
[4]API: Application Programming Interface - Interfaz de Programación de Aplicación
[5]JSON: JavaScript Object Notation - Notación de Objetos de JavaScript
[6]CSV: Comma Separated Values - Valores Separados por Comas
[7]BEMS: Building Energy Management System - Sistema de Gestión de la Energía en Edificios
[8]DEMS: District Energy Management System - Sistema de Gestión de la Energía en Distritos
[9]SFTP: Secure File Transfer Protocol - Protocolo Seguro de Transferencia de Ficheros
[10]PV: Photovoltaic Energy - Energía Fotovoltaica
[11]EV: Electrical Vehicle - Vehículo Eléctrico

de mensajes de mediciones o similares entre instancias o aplicaciones. *InfluxDB* es una base de datos de series temporales diseñada y optimizada para manejar grandes volúmenes de datos que tienen una marca temporal. Es particularmente útil para aplicaciones que requieren el almacenamiento, consulta y análisis de datos que varían con el tiempo, como sistemas *big data*. El protocolo SFTP o SSH *File Transfer Protocol* se emplea para la transferencia de ficheros de forma segura sobre HTTP. Las APIs de *Google Calendar* y *Drive* permiten la extracción de datos sobre eventos y ficheros respectivamente a través de los diferentes *endpoints* que tienen disponibles. Por último, mediante el lenguaje SQL podemos hacer peticiones a la base de datos de *PostgreSQL* sobre HTTP.

Formatos de representación de los datos

Por otra parte, se emplean diferentes formatos de representación de los datos. Entre ellos podemos encontrar el formato de texto plano; el formato *JavaScript Object Notation* o JSON, que implementa estructuras clave-valor y listas con contenido legible; el formato *Comma-Separated Values* o CSV, que consiste e la enumeración de diferentes valores separados por un caracter concreto, como “,” o “;”.

Podemos observar cómo existe una gran variedad de interfaces para la extracción de los datos mencionados. Entre ellas podemos destacar el protocolo MQTT, las interfaces de bases de datos (*InfluxDB* y *PostgreSQL*), el protocolo de transferencia de ficheros (SFTP) o APIs genéricas que implementan los dispositivos. Cada una de estas interfaces se corresponderá con un flujo de datos dentro de las transformaciones para cada piloto.

Elementos relevantes de Pentaho

Previo a la inserción de los datos en el DWH (base de datos destinada a almacenar los datos de las series temporales), un módulo de sincronización se hará cargo de la integración de los propios datos, manteniendo la armonía entre las muestras de datos de los pilotos. Además, los datos estáticos pueden ser extraídos de las propias ETLs, por lo cual, también es necesario mantener la armonización de los metadatos de las ETLs y de los repositorios de metadatos estáticos. El desarrollo de este módulo es ajeno a los objetivos del TFM, pero se explicará más adelante en esta memoria.

La implementación de estos procesos ETL se ha empleado la herramienta *Pentaho Data Integration*, también conocida como *Kettle* dentro de la Suite de herramientas que se ofrecen dentro del *software*, como se ha mencionado en el capítulo 2. Este software basa la programación de las ETL mediante diagramas de bloques que desempeñan diferentes funciones de tratamiento de datos y dispone de una interfaz sencilla y un funcionamiento muy intuitivo. Además, permiten como entradas y salidas una gran variedad de fuentes de datos. También dispone de un *market-place* donde se publican diferentes *plug-ins*, lo cual aumenta todavía más la versatilidad que tenemos a nuestro alcance. Se ha decidido utilizar esta herramienta para poder mantener todos los flujos de datos alojados en una única aplicación de servidor y de forma centralizada donde se

encuentren corriendo simultáneamente.

Estos procesos ETL necesitan de otro elemento importante para su ejecución y orquestación, los *jobs*. En Pentaho, los *jobs* son componentes clave que permiten orquestar y gestionar flujos de trabajo complejos en procesos de integración de datos. Sus características principales comprenden: la orquestación de tareas controlando la ejecución de múltiples pasos y ETLs, el control de flujo mediante lógica condicional, la automatización mediante la ejecución repetitiva y programada, la integración de componentes variados como operaciones con el SO, ejecución de comandos o *scripts*, llamadas a servicios web o ejecución de ETLs u otros *jobs*.

Un modelo genérico de los flujos de datos en las ETLs implementadas se ilustra en la figura 5 [28]. El proceso se inicia con la fase de extracción de datos utilizando las interfaces descritas en la tabla 1, lo cual nos permite capturar los datos en su formato original. A partir de este conjunto inicial, se separan los datos en dos categorías principales: datos estáticos y datos dinámicos. Estos dos flujos de datos son dirigidas hacia el *data lake*, aunque se almacenan en bases de datos distintas dentro de este. En nuestro caso es de interés el DWH, almacén de los datos dinámicos. Además, los datos dinámicos se derivan hacia dos flujos adicionales: uno enfocado en el análisis de calidad de datos y otro destinado al consumo en tiempo real de datos, este último a través de *Kafka*.

Figura 5: Flujos genéricos en una ETL

El flujo de análisis de la calidad de los datos se implementa directamente embebida en las transformaciones de los pilotos, empleando los bloques nativos de *Kettle*. En el capítulo 5 se explicará en profundidad el contenido relativo a esta líneas e procesamiento de las ETLs. Por simplificar la estructura de éstas, se omitirá en las figuras de las transformaciones. Más adelante se describirá el flujo de calidad de datos, el cual es igual para todas las ETLs.

Kafka o *Apache Kafka* es una plataforma de *streaming* de eventos desarrollada especialmente para el manejo de grandes volúmenes de datos en tiempo real. Normalmente se emplea en escenarios donde la integridad de los datos y la tolerancia a fallos son puntos críticos, el cual es nuestro caso. En la tabla 2 se pueden ver los diferentes *topics* que se han definido en base a las necesidades de tanto los servicios como de los gemelos digitales de los pilotos del proyecto. En ellos se publican mensajes en función del piloto y la categoría de los datos. Observamos cómo para el piloto 09 o NTUA no se ha definido ninguno, debido a que este piloto no requiere de gemelo digital. También, se aprecia que el piloto 04 se ha desagregado en VEOLIA-FASA o 04a y VEOLIA-RVENA o 04b, dado que éste consta de dos localizaciones geográficas (Valladolid y Burgos respectivamente) y, por lo tanto, de dos gemelos digitales, los cuales necesitan el consumo de los mensajes en *topics* diferentes. También se producen datos sobre BEMS o DEMS

en todos los pilotos excepto en EDF (02) e IEECP (08); sobre calidad de aire en UCL (01), EDF, FOCCHI (05b) e IEECP ; sobre carga de vehículo eléctrico en 05a y 06 y sobre generación fotovoltaica en 05a. Finalmente, mencionar que para fuentes externas de datos se ha desarrollado una ETL para la ingestión de datos sobre predicciones meteorológicas. Esta transformación se emplea para los pilotos de UCL, EDF, IASI&SITTA (03), VEOLIA-FASA, VEOLIA-RVENA, EMOT (05a) y HERON(06).

La construcción de los *topics* se ha llevado a cambo mediante la concatenación de tres campos: el nombre del proyecto, el piloto al que está destinado el *topic* y el uso o la categoría de los datos en inglés separados por puntos (digibuild.*NN{x}*.*ORIGEN*). Esta decisión se ha tomado dado que *Kafka* no permite la creación de *subtopics* contenidos en *topics*, y esta forma es el estándar de facto pese a permitir otros caracteres de separación entre campos que conforman un *topic* [37, 38].

Piloto	**Origen**	***Topic***
01	Meteorología	digibuild.01.weather
	Datos edificio	digibuild.01.building
	Calidad aire	digibuild.01.airquality
02	Meteorología	digibuild.02.weather
	Calidad aire	digibuild.02.airquality
03	Meteorología	digibuild.03.weather
	Datos edificio	digibuild.03.building
04a	Meteorología	digibuild.04a.weather
	Datos distrito	digibuild.04a.district
04b	Meteorología	digibuild.04b.weather
	Datos distrito	digibuild.04b.district
05a	Meteorología	digibuild.05a.weather
	Datos edificio	digibuild.05a.building
	PV	digibuild.05a.pv
	EV	digibuild.05a.ev
05b	Datos edificio	digibuild.05b.building
	Calidad aire	digibuild.05b.airquality
06	Meteorología	digibuild.06.weather
	Datos edificio	digibuild.06.building
	EV	digibuild.06.ev
07	Datos edificio	digibuild.07.building
	Air quality	digibuild.07.airquality
08	Calidad Aire	digibuild.08.airquality

Tabla 2: *Topics* para la publicación de datos en tiempo real

Las transformaciones que se han implementado se definen dentro de lo que en *Pentaho* se conoce como *job* o Trabajo. En estos *jobs* se definen diferentes parámetros como la periodicidad a la que queramos que se ejecuten las transformaciones, las credenciales de autenticación e identificación o los propios parámetros de las mismas. Por otro lado es en estos *jobs* donde en algunos casos hemos tenido que definir algunas de las conexiones con las interfaces de datos

según la posibilidad de hacerlo dentro de la propia transformación o no, es decir, con la ejecución de *scripts* de *Shell* o *Python*. Además, se han implementado en todos estos *jobs* alarmas que notifican por correo a los administradores si sucede algún error en la ejecución tanto del *job* como de la transformación.

A continuación procedemos a la descripción detallada de las diferentes ETLs desarrolladas para cada piloto, incluyendo información sobre las fuentes de datos y la implementación de los *jobs*.

4.1. ETL del piloto 01 - UCL

El piloto de la *University College London* (UCL) se centra en la digitalización y monitoreo de datos en diversos edificios de la universidad. El objetivo es mejorar la eficiencia energética y el confort de los ocupantes mediante la recolección y análisis de datos detallados provenientes de diferentes sistemas de construcción.

Su objetivos engloban: Desarrollar auditorías energéticas efectivas, implementar un sistema de monitoreo dinámico para recolectar datos en tiempo real, mejorar la interoperabilidad de los sistemas de gestión de edificios, proporcionar análisis detallados para optimizar el uso de la energía o asegurar un alto nivel de confort para los ocupantes. Todos estos objetivos se alcanzan mediante el desarrollo de las ETLs y el cálculo de diferentes KPIs o *Key Performance Indicators*.

4.1.1. Dimensiones del *big data*

Tras analizar la información de este piloto, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: El piloto 01 está formado por dos edificios con localización geográfica cercana. El valor extraído de todos los datos de este piloto son los mencionados KPIs para poder alcanzar los objetivos propuestos, como la generación de servicios de referencia comparativa o *benchmarking*.
- **Variedad**: El piloto de UCL incluye para este proyecto la extracción de datos ambientales como la temperatura o el CO_2, variables de presencia como apertura de puertas, ventanas o sensores lumínicos y de consumos de agua fría y caliente y producción de energía fotovoltaica. Para este piloto nos encontramos con múltiples *datasets* los cuales contienen casi 700 variables relativas a BMS, EMS, controles de luz y acceso y ocupación, recolectando aproximadamente 3800 puntos.

  El piloto 01 consta de una única interfaz para la extracción de los datos. Esta interfaz se basa en el protocolo MQTT, que se emplea para la transmisión de mensajes entre diferentes aplicaciones. MQTT se basa en el paradigma publicación/suscripción [29], creando colas de mensajes para cada uno de los *topics* o temas en los que se publican dichos mensajes.
- **Volumen**: Entre los dos edificos que conforman el piloto se estima que se generan más de 3 GB de datos al año relevantes para el proyecto de DigiBUILD.
- **Velocidad**: En este piloto el periodo de muestreo también es muy variado, comprendiendo desde los 5 minutos hasta cada segundo o incluso basada en eventos. Por otra parte, la velocidad con la que se tienen disponibles los datos que se generan a las frecuencias mencionadas es en tiempo real gracias a la utilización del protocolo MQTT, dependiendo en la latencia de la propia red.
- **Veracidad**: El responsable del piloto mantiene la totalidad los datos generados bajo su propia gobernanza. Pudiendo ser accedidos desde el exterior mediante credenciales empleados en nuestra identificación en el *broker* MQTT.

4.1.2. Flujos de datos

Job

El *job* definido para este piloto (figura 6) está formado únicamente por la llamada a la ETL y el bloque de alarma por *mail* y no es necesario especificar ningún otro parámetro sobre frecuencia de ejecución o credenciales de autenticación o autorización.

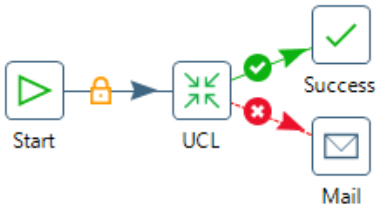

Figura 6: Job piloto 01

Transformación

La transformación correspondiente a este piloto puede ser observada en la figura 7. Puede verse como consta de un único flujo de procesamiento, como se ha hecho mención a lo largo de esta sección.

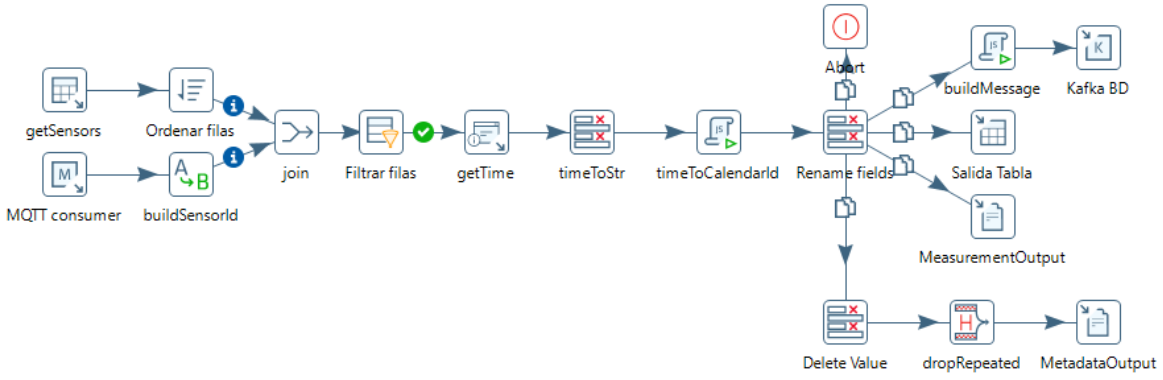

Figura 7: ETL piloto 01

La transformación comienza con la ingesta de datos a través de un *broker* MQTT. Primero, procedemos con la suscripción a los diferentes *topics* del *broker* o "servidor" MQTT. Los datos que obtenemos tienen un formato JSON y no cuentan con un parámetro temporal para ubicar las medidas generadas. En este bloque se especifican las credenciales de acceso para la propia

extracción de datos mediante usuario y contraseña, así como las características de la extracción, como especificar un *keep-alive* o temporizador de inactividad y ajustar el tamaño de los lotes que se reciben para no excedernos en cuanto a los recursos de ejecución.

A continuación, se filtran los *topics*, que se corresponden con el edificio y los sistemas cuyos datos van a ser integrados en el sistema. Este filtrado se hace extrayendo los puntos de medida definidos previamente en el DWH y que han sido proporcionados por el responsable técnico del piloto. Esta extracción se lleva a cabo mediante una petición SQL a la tabla del DWH donde se almacena el listado de los sensores relevantes para el proyecto. Este filtrado se lleva a cabo mediante la unión de los flujos y el filtrado de las filas que no encuentren una coincidencia con la lista de sensores del DWH.

Después, se conforma una variable temporal con un formato concreto para ser consistente con la estructura de tablas que se ha definido en el DWH llamado *calendar_id*. Este parámetro será calculado en todas las transformaciones y constará de la siguiente información: partiendo de una variable de fecha, con el formato **yyyy/mm/dd hh:MM:ss.SSS**, siendo *yyyy* el año, *mm* el mes, *dd* el día, *hh* la hora, *MM* los minutos, *ss* los segundos y *SSS* la parte decimal de los segundos. La variable temporal *calendar_id* que necesitamos asociar a cada medición se forma con los datos anteriores y debe tener el formato: **yyyymmddhhMM**. En el capítulo 6 se explicará la motivación de la utilización de este parámetro y su función dentro del DWH. La conformación del *calendar_id* se lleva a cabo mediante la extracción de la fecha y horas actuales, ya que suponemos que los retardos entre la medición y la consumición son despreciables.

Finalmente, se procede a la exportación de los datos transformados hacia diferentes ramas haciendo uso de un filtrado. Una de estas ramas es la de metadatos, los cuales no serán utilizados por nuestra parte y simplemente se eliminan todas las variables relativas a ellos del flujo (*topic*, equipo, punto, instancia, espacio y nivel). Las otras dos salidas del flujo las conforman los datos temporales, formados por una modificación parcial del *topic* para la generación del identificador del sensor asociado de forma unívoca al dispositivo y a la medición, el identificador temporal y el valor medición.

La parte final de esta transformación consiste en la inserción de los datos en el DWH. En la rama de los datos dinámicos se realiza el filtrado de datos separándolos por BMS y de calidad de aire y se publican en el *broker* de kafka en los *topics digibuild.01.building* y *digibuild.01.airquality* respectivamente.

Cada entrada de datos que se introduce en el DWH que se deba publicar en el *broker* de *Kafka* se tendrá se transformar de acuerdo a la estructura de la figura 8. La parte superior muestra cada una de las entradas del DWH. Se observan los campos *calendar_id*, que identifica el *timestamp* de la medida con el formato que se ha descrito anteriormente; *sensor_id*, que es el nombre único que identifica a cada sensor, y *f_value*, el cual es el valor de la medida. El formato del mensaje que se publica en el *broker* es una equivalencia directa en formato JSON.

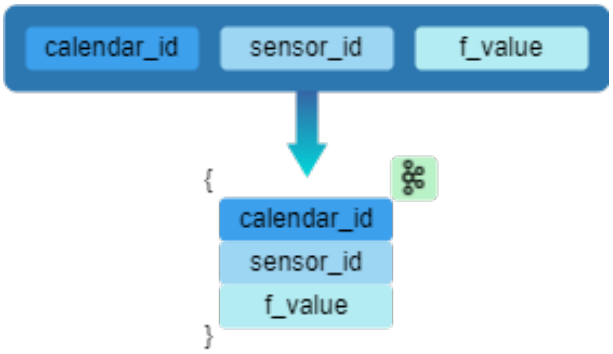

Figura 8: Equivalencia entrada del DWH y mensaje de Kafka

4.1.3. Salidas

Como resultado de esta transformación, contamos con dos flujos relevantes: los datos dinámicos y los metadatos. A continuación se describen ambos.

En la figura 9 podemos observar los metadatos que se generan en esta transformación. Su contenido contiene diferentes campos, siendo éstos: el tipo de equipamiento, el nombre del punto de medida, el identificador de la instancia del equipamiento, el identificador de la zona del edificio se se mide, el espacio que engloba esa zona y el piso en el que se hace la medida. Algunos de los campos se observan vacíos, debido a que no aplica o a que en el piloto no se han definido correctamente. Además, almacenamos el *topic*, que aporta algo más de información, como el identificador del edificio o el dataset del que se extrae la medida.

topic	equipment	point	instance	zone	space	level
UCL/OPSEBOAS/PSW TW20-XX-CE-001/Modbus TCP Gateway/L14-C2-HIU/StepperPos/Value	<null>	<null>	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TE16-XX-CE-001/Network Variables/Differential Pressure Sensor/Value	<null>	Differential Pressure Sensor	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW PD00-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/TW00-ATR_VAV01/SupVavCO2/Value	VAV	SupVavCO2	<null>	<null>	GC1	L00
UCL/OPSEBOAS/PSW TW20-XX-CE-001/Modbus TCP Gateway/L15-C2-HIU/StepperPos/Value	<null>	<null>	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/Application/Energy Monitoring/VAV/TW01_LBR_RPC_02/Air Flow VAV1/Value	VAV	Air Flow VAV1	1	<null>	101	L01
UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/AHU1/AHU01LTHWVlvCntrl/Value	AHU	AHU01LTHWVlvCntrl	1	EP-L2	<null>	L02
UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/VAV/TW02_WRK3_RPC_02/Air Flow VAV2/Value	VAV	Air Flow VAV2	2	<null>	206	L02
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_LBR_RPC_01/IO Resources/Sensor Bus/Wall Sensor/Occupancy/Val...	Wall Sensor	Occupancy	<null>	<null>	101	L01
UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/IP Network/TE01_SS2_RPC_01/IO Resources/Sensor Bus/Wall Sensor/Occupancy/Value	<null>	<null>	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_ATR_FCU_007/IO Resources/Sensor Bus/Wall Sensor/CO2/Value	Wall Sensor	CO2	7	<null>	1C2	L01
UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/VAV/TE01_NST1_RPC_01/Air Flow VAV1/Value	VAV	Air Flow VAV1	1	<null>	120	L01
UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/IP Network/TW02_Off_RPC_03/Application/Monitoring/VAV3 Flow/Value	VAV	VAV3 Flow	3	203-203B-203D	<null>	L02
UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/VAV/TE01_RSH_RPC_01/Air Flow VAV3/Value	VAV	Air Flow VAV3	3	<null>	121	L01
UCL/OPSEBOAS/PSW 0000-XX-CE-001/BACnet Interface/IP Network/TE00_GDI_RPC_01/Application/Monitoring/VAV1 Flow/Value	VAV	VAV1 Flow	1	<null>	G19	L00
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/Application/Energy Monitoring/VAV/TW00_MEGR1_RPC_01/Air Flow VAV1/Value	VAV	Air Flow VAV1	1	<null>	G04	L00
UCL/OPSEBOAS/PSW 0000-XX-CE-001/BACnet Interface/IP Network/TW00_CU_FCU_002/IO Resources/Sensor Bus/Wall Sensor/CO2/Value	Wall Sensor	CO2	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_LBR_RPC_01/Application/Monitoring/VAV2 Flow/Value	VAV	VAV2 Flow	2	<null>	101	L01
UCL/OPSEBOAS/PSW B000-XX-MCC-001/Modbus Master Network/B000-XX-HM-004-Secondary LTHW Student East HIUs/Modbus Signals/Table...	HM	262 Actual Power	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/IP Network/TE02_Off_RPC_03/Application/Monitoring/VAV2 Flow/Value	VAV	VAV2 Flow	2	219A-219B	<null>	L02
UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/IP Network/TE01_RSH_RPC_01/Application/Monitoring/VAV1 Flow/Value	VAV	VAV1 Flow	1	<null>	121	L01
UCL/OPSEBOAS/PSW TE01-XX-MCC-001/Modbus Master Network/0000-XX-HM-006/Modbus Signals/Table 1/258 Actual Flow/Value	HM	258 Actual Flow	<null>	<null>	G20	L00
UCL/OPSEBOAS/PSW PD00-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/TW00-ATR_VAV01/SupVavFlowSp/Value	VAV	SupVavFlowSp	<null>	<null>	GC1	L00
UCL/OPSEBOAS/PSW B000-XX-MCC-001/Modbus Master Network/B000-XX-HM-004-Secondary LTHW Student East HIUs/Modbus Signals/Table...	HM	266 Outlet Temperature T2	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/Application/Energy Monitoring/VAV/TW01_TS1_RPC_01/Air Flow VAV1/Value	VAV	Air Flow VAV1	1	<null>	106	L01
UCL/OPSEBOAS/PSW B000-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/Heat Meters/004-Secondary LTHW Student East HIUs...	HM	Power	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/IP Network/TW02_Off_RPC_03/Application/Monitoring/VAV1 Flow/Value	VAV	VAV1 Flow	1	203-203B-203D	<null>	L02
UCL/OPSEBOAS/PSW PD03-XX-CE-001/BACnet Interface/IP Network/TW03_CM1_FCU_002/IO Resources/Sensor Bus/Wall Sensor/CO2/Value	Wall Sensor	CO2	<null>	<null>	305	L03
UCL/OPSEBOAS/PSW B000-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/Heat Meters/004-Secondary LTHW Student East HIUs...	HM	Outlet Temp	<null>	<null>	<null>	<null>
UCL/OPSEBOAS/PSW PD03-XX-CE-001/BACnet Interface/Application/Energy Monitoring/MVHR/TW03-RES-MVHR-001/TW03 -RES-MVHR001_C...	MVHR	TW03 -RES-MVHR001_CO2 Sensor	<null>	<null>	<null>	L03
UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_LBR_RPC_02/Application/Monitoring/VAV1 Flow/Value	VAV	VAV1 Flow	1	<null>	101	L01
UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/Heat Meters/0000-XX-HM-006/Flow/Value	HM	Flow	<null>	<null>	G20	L00
UCL/OPSEBOAS/PSW B000-XX-MCC-001/Modbus Master Network/B000-XX-HM-003-Secondary LTHW Student West HIUs/Modbus Signals/Tabl...	HM	262 Actual Power	<null>	<null>	<null>	<null>

Figura 9: Piloto 01 - Salida de metadatos

Por otra parte, los datos dinámicos extraídos tienen la siguiente forma (Figura 10): El parámetro temporal, como se ha descrito anteriormente, el identificador del sensor, que se corresponde casi por completo con el *topic* y, finalmente el valor de las medidas.

calendar_id	value	topic
20231114091604	188	UCL/OPSEBOAS/PSW TW20-XX-CE-001/Modbus TCP Gateway/L14-C2-HIU/StepperPos/Value
20231114091604	1759	UCL/OPSEBOAS/PSW TE16-XX-CE-001/Network Variables/Differential Pressure Sensor/Value
20231114091604	463	UCL/OPSEBOAS/PSW PD00-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/TW00-ATR_VAV01/SupVavCO2/Value
20231114091604	17.7	UCL/OPSEBOAS/PSW TW20-XX-CE-001/Modbus TCP Gateway/L15-C2-HIU/StepperPos/Value
20231114091604	30.033	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/Application/Energy Monitoring/VAV/TW01_LBR_RPC_02/Air Flow VAV1/Value
20231114091604	12.79	UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/AHU1/AHU01LTHWVlvCntrl/Value
20231114091604	246.654	UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/VAV/TW02_WRK3_RPC_02/Air Flow VAV2/Value
20231114091604	0	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_LBR_RPC_01/IO Resources/Sensor Bus/Wall Sensor/Occupancy/Value
20231114091604	1	UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/IP Network/TE01_SS2_RPC_01/IO Resources/Sensor Bus/Wall Sensor/Occupancy/Value
20231114091604	439	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_ATR_FCU_007/IO Resources/Sensor Bus/Wall Sensor/CO2/Value
20231114091604	424.424	UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/VAV/TE01_NST1_RPC_01/Air Flow VAV1/Value
20231114091604	292.528	UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/IP Network/TW02_Off_RPC_03/Application/Monitoring/VAV3 Flow/Value
20231114091604	399.758	UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/VAV/TE01_RSH_RPC_01/Air Flow VAV3/Value
20231114091604	303.524994	UCL/OPSEBOAS/PSW 0000-XX-CE-001/BACnet Interface/IP Network/TE00_GDI_RPC_01/Application/Monitoring/VAV1 Flow/Value
20231114091604	131.576	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/Application/Energy Monitoring/VAV/TW00_MEGR1_RPC_01/Air Flow VAV1/Value
20231114091604	674	UCL/OPSEBOAS/PSW 0000-XX-CE-001/BACnet Interface/IP Network/TW00_CU_FCU_002/IO Resources/Sensor Bus/Wall Sensor/CO2/Value
20231114091605	183.01	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_LBR_RPC_01/Application/Monitoring/VAV2 Flow/Value
20231114091605	58	UCL/OPSEBOAS/PSW B000-XX-MCC-001/Modbus Master Network/B000-XX-HM-004-Secondary LTHW Student East HIUs/Modbus Signals/Table 1/262 Actual Power/Value
20231114091605	119.075	UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/IP Network/TE02_Off_RPC_03/Application/Monitoring/VAV2 Flow/Value
20231114091605	395.669	UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/IP Network/TE01_RSH_RPC_01/Application/Monitoring/VAV1 Flow/Value
20231114091605	1915	UCL/OPSEBOAS/PSW TE01-XX-MCC-001/Modbus Master Network/0000-XX-HM-006/Modbus Signals/Table 1/258 Actual Flow/Value
20231114091605	773.814	UCL/OPSEBOAS/PSW PD00-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/TW00-ATR_VAV01/SupVavFlowSp/Value
20231114091605	65.8099976	UCL/OPSEBOAS/PSW B000-XX-MCC-001/Modbus Master Network/B000-XX-HM-004-Secondary LTHW Student East HIUs/Modbus Signals/Table 1/266 Outlet Temperature T2/Value
20231114091605	276.304	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/Application/Energy Monitoring/VAV/TW01_TS1_RPC_01/Air Flow VAV1/Value
20231114091605	58	UCL/OPSEBOAS/PSW B000-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/Heat Meters/004-Secondary LTHW Student East HIUs/Power/Value
20231114091605	194.557	UCL/OPSEBOAS/PSW TE02-XX-MCC-001/BACnet Interface/IP Network/TW02_Off_RPC_03/Application/Monitoring/VAV1 Flow/Value
20231114091605	504	UCL/OPSEBOAS/PSW PD03-XX-CE-001/BACnet Interface/IP Network/TW03_CM1_FCU_002/IO Resources/Sensor Bus/Wall Sensor/CO2/Value
20231114091605	65.81	UCL/OPSEBOAS/PSW B000-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/Heat Meters/004-Secondary LTHW Student East HIUs/Outlet Temp/Value
20231114091605	504	UCL/OPSEBOAS/PSW PD03-XX-CE-001/BACnet Interface/Application/Energy Monitoring/MVHR/TW03-RES-MVHR-001/TW03 -RES-MVHR001_CO2 Sensor/Value
20231114091605	30.3525	UCL/OPSEBOAS/PSW TW01-XX-MCC-002/BACnet Interface/IP Network/TW01_LBR_RPC_02/Application/Monitoring/VAV1 Flow/Value
20231114091605	1.915e+06	UCL/OPSEBOAS/PSW TE01-XX-MCC-001/BACnet Interface/Application/Energy Monitoring/Heat Meters/0000-XX-HM-006/Flow/Value
20231114091605	105	UCL/OPSEBOAS/PSW B000-XX-MCC-001/Modbus Master Network/B000-XX-HM-003-Secondary LTHW Student West HIUs/Modbus Signals/Table 1/262 Actual Power/Value

Figura 10: Piloto 01 - Salida de datos dinámicos

4.2. ETL del piloto 02 - EDF

El piloto de EDF (*Electricité de France*) se enfoca en la digitalización y monitoreo de datos en diversas instalaciones de la empresa. El objetivo principal es optimizar la eficiencia energética y la gestión de recursos mediante la recolección y análisis de datos detallados de distintos sistemas y dispositivos en tiempo real.

Entre los objetivos de este piloto dentro del proyecto cabe destacar los siguientes: implementar un sistema de monitoreo continuo para capturar datos en tiempo real, mejorar la eficiencia energética de las instalaciones, facilitar la interoperabilidad de los sistemas de gestión de energía, realizar análisis detallados para identificar oportunidades de ahorro energético o assegurar un alto nivel de confort y seguridad en las instalaciones.

4.2.1. Dimensiones del *big data*

Tras analizar la información de este piloto, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: Este piloto cuenta con un único edificio donde actualmente no existe ningún sistema de gestión de la energía o del sistema de aire acondicionado. El valor obtenido es la generación de perfiles de consumo, utilización de las instalaciones por zonas y explotación de los datos en crudos para poder conseguir los objetivos propuestos.
- **Variedad**: El piloto 02 cuenta con diferentes *datasets*, englobando variables de calidad de aire tanto internas como externas y consumo eléctrico para calefacción, luz y enchufes. Además de estaciones de carga de vehículo eléctrico, producción de agua caliente e información meteorológica. Sólo se recogerán en el sistema datos sobre calidad de aire y consumo eléctrico de enchufes.

  Las interfaces para la extracción de datos en este piloto son tres, aunque todas ellas se basan en la compartición de datos mediante API. Una de las interfaces, llamada *Ethera*, la cual implementa una API de *Nemocloud*, contiene datos sobre calidad de aire tanto interior como exterior. La interfaz llamada Ellona, que implementa una API *Ellonasoft* permite el acceso a datos de calidad de aire interior. Por último, la API de *Wattsense* proporciona datos sobre los consumos eléctricos de enchufes. La principal diferencia entre las APIs yace en la forma que implementan la autenticación del usuario.
- **Volumen**: Genera una gran cantidad de datos diarios, acumulando varios terabytes de información anualmente.
- **Velocidad**: Las interfaces que conforman este piloto se actualizan entre 1 y 5 minutos. Para evitar un uso excesivo de los recursos por parte del *job* de este piloto, se ha establecido una frecuencia de ejecución de 10 minutos, pese a que los datos se muestrean con una frecuencia que varía, dependiendo del tipo de dato.
- **Veracidad**: Al igual que se ha mencionado en el caso del piloto 01, el acceso a los datos de las 3 interfaces del piloto es privada y no se puede acceder de forma libre. Por ello,

asumimos la incorruptibilidad de los datos, ya que se emplean llamadas a las APIs sobre la versión segura de HTTP.

4.2.2. Flujos de datos

Job

El *job* correspondiente al piloto 02 (figura 11) incluye únicamente la llamada a la transformación asociada al piloto y la alarma por *mail*. En este caso se especifican como parámetros las credenciales para acceder a las diferentes interfaces de extracción de datos junto con el parámetro que determina qué periodo de tiempo se extraen de las interfaces. Además, se ha configurado la repetición de la ejecución del *job* cada 10 minutos.

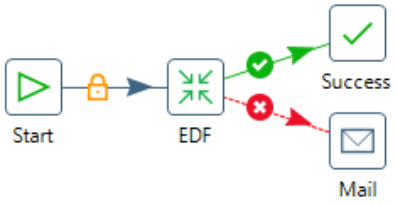

Figura 11: Job piloto 02

Transformación

La transformación desarrollada para el piloto 02 incluye 3 flujos de datos (Figuras 12, 13 y 14. Cada uno de ellos corresponde a una interfaz, dado que el formato en el que vienen los datos en origen es diferente y requieren un tratamiento diferente. A continuación procederemos a explicar cada uno de los flujos y las transformaciones que comprenden.

Para la interfaz *Nemocloud API* (Figura 12) comenzamos con la extracción de los datos a través de diferentes *endpoints* que permiten obtener el *token* de sesión mediante el proceso *Digest Access Authentication* [39]. Este proceso llamado *Digest Access Authentication* consiste en diferentes pasos. Primero se comienza con la petición del cliente a un recurso protegido. En respuesta a esta petición, el servidor envía un paquete con mensaje *Unauthorized* y campos en la cabecera para que el cliente resuelva un reto. Estos campos tienen nombres como *realm, opaque* y *nonce* entre otros. Después, en cliente lleva a cabo ciertas operaciones con dichos campos que, implican *hashing* o codificaciones diferentes y, en su nueva petición añade campos de respuesta para autenticarse ante el servidor. Finalmente, el servidor aprueba la autenticación devolviendo un *token* de sesión. A continuación, se procede a extraer el listado de dispositivos a través de un *endpoint* específico de la misma API. Con los datos del listado de dispositivos, se hace un nueva petición para obtener el listado de variables asociado a cada dispositivo empleando el *token*

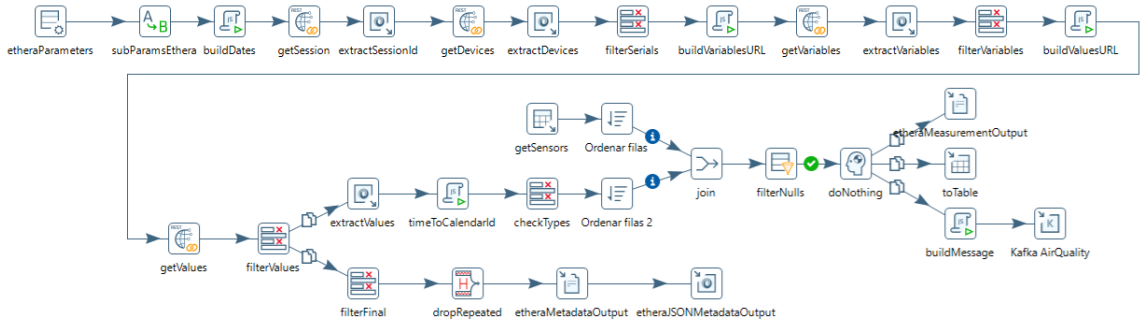

Figura 12: ETL piloto 02 - Línea Ethera

inicial y, por último y de la misma forma, el listado de los valores generados de cada variable de cada dispositivo. Posteriormente, se lleva a cabo una separación de los datos en metadatos y datos dinámicos. Los metadatos son limpiados y extraídos del flujo. A los datos dinámicos se les aplica el formato adecuado y se conforma el *calendar_id*. Finalmente, se aplica un filtrado para obtener únicamente los puntos que son relevante. Este filtrado se lleva a cabo mediante una petición SQL a la tabla del DWH donde se almacena el listado de los sensores del piloto de EDF relevantes para el proyecto. Este filtrado se lleva a cabo mediante la unión de los flujos y el filtrado de las filas que no encuentren una coincidencia entre la lista de puntos del DWH y las variables extraídas de la API. Para terminar, se almacenan en el mismo DWH y se publican en el *topic digibuild.02.airquality* de *kafka*.

En el segundo flujo de datos, que se corresponde con la interfaz de *Ellona* (Figura 13), comenzamos con nuestra autenticación mediante un *endpoint* específico en el que proporcionamos nuestro usuario y contraseña y se nos devuelve en la respuesta un *token* se sesión. A continuación, conformamos la fechas entre las cuales deseamos extraer los datos de la API (los últimos 10 minutos), importando esta información desde el *job* y se piden a través de otro *endpoint*. Una vez poseamos los datos relevantes, se filtran los metadatos del flujo y se formatean los datos dinámicos. Este formateo consiste en: adaptar la respuesta del *endpoint* a formato JSON, comprobar que esa respuesta no esté vacía (y en caso positivo, filtrarla del flujo), extraer los subcampos del JSON para identificar los puntos extraídos, construir la variable temporal *calendar_id* y hacer un pivotaje para tener los valores en una única columna. Por último se filtran las variables relevantes para el proyecto. Este filtrado se lleva a cabo mediante una petición SQL a la tabla del DWH donde se almacena el listado de variables de este piloto para el proyecto. Este filtrado se lleva a cabo mediante la unión de los flujos y el filtrado de las filas que no encuentren una coincidencia entre la lista de puntos del DWH y las variables extraídas de la API y se exportan tanto al DWH como al *topic digibuild.02.airquality* de *kafka*.

Para el tercer y último de los flujos del piloto 03 (Figura 14), comenzamos construyendo a partir de las fechas importadas desde el *job*, la URL del *endpoint* al que vamos a hacer la petición para la extracción de los datos. Además, al hacer la petición, incluimos en la cabecera del mensaje nuestras credenciales de usuario y contraseña para autenticarnos directamente. En este flujo, por lo tanto, sólo tenemos que hacer una única petición para extraer los datos, a diferencia de las anteriores. Una vez contamos con los datos de la API, se procede a la separación de los

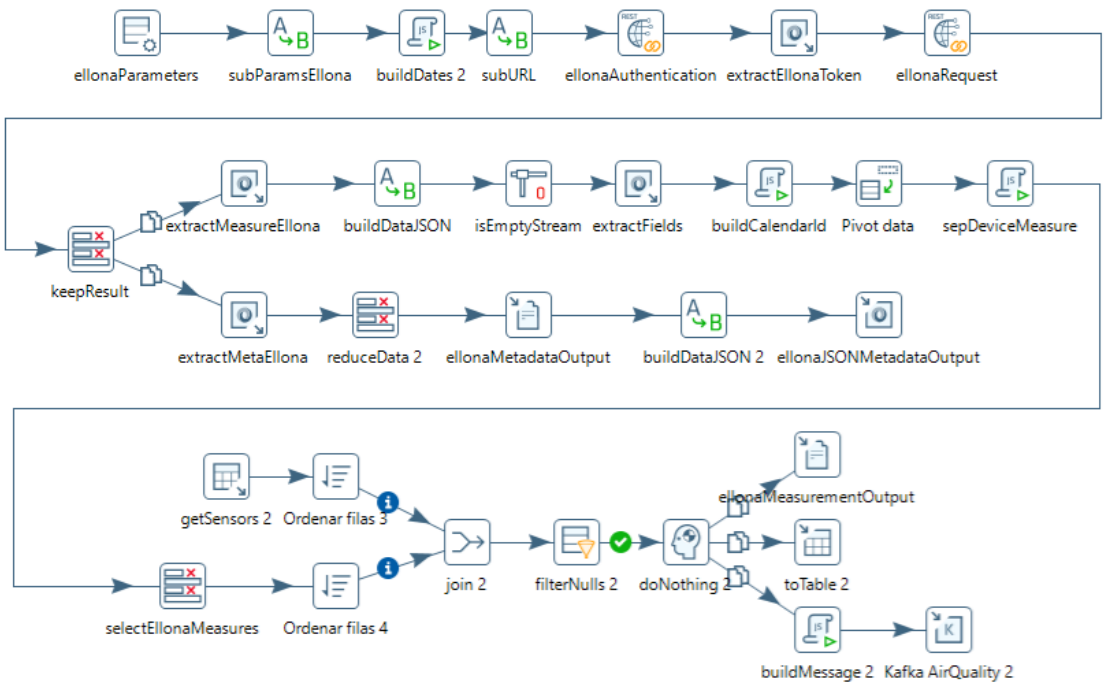

Figura 13: ETL piloto 02 - Línea Ellona

metadatos del flujo principal. Después, se formatea el campo temporal del *calendar_id* y se procede al filtrado de los puntos relevantes para el piloto a través de una petición SQL a la tabla del DWH donde se almacena el listado de variables de este piloto para el proyecto. Este filtrado se lleva a cabo mediante la unión de los flujos y el filtrado de las filas que no encuentren una coincidencia entre la lista de puntos del DWH y las variables extraídas de la API. Finalmente, se exportan los datos dinámicos tanto al DWH como al *topic digibuild.02.airquality* de *kafka*.

4.2.3. Salidas

Como resultado de los flujos de esta transformación, se obtienen dos salidas de datos por cada uno de los tres flujos que acabamos de describir, uno de metadatos y otro de datos dinámicos. A continuación procedemos a describir estas salidas.

Línea Ethera

Para la primera interfaz de extracción del piloto de EDF (*NemoCloud API*), obtenemos como salidas las instancias que se observan en las figuras 15 y 16.

En lo referente a los metadatos (Figura 15) vemos diferentes campos relativos a los puntos

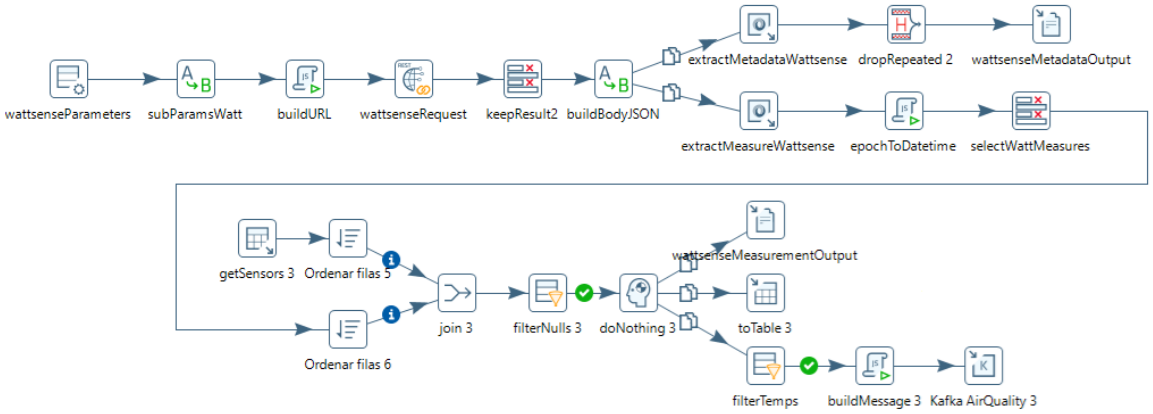

Figura 14: ETL piloto 02 - Línea Wattsense

meidos y los sensores. Entre ellos se encuentran, el identificador del sensor, el identificador de la variable, el nombre de la magnitud que mide o la unidad de medida empleada. Por otro lado,

structure	source	id	name	unit
32848	0	80	Pressure	mb
32852	0	84	Temperature	°C
32840	0	72	Humidity	HR%
32835	0	67	Carbon dioxide	ppm
32867	0	99	Light Volatile Organic Compounds	ppb
32834	0	66	Battery	V
33583	3	47	Particulate matter 1	µg/m3
33585	3	49	Particulate matter 2.5	µg/m3
33584	3	48	Particulate matter 4	µg/m3
33040	1	16	Nitrogen dioxide	ppb
33586	3	50	Particulate matter 10	µg/m3
33638	3	102	Total Volatile Organic Compounds	ppm
33327	2	47	Particulate matter 1	µg/m3
33329	2	49	Particulate matter 2.5	µg/m3
33328	2	48	Particulate matter 4	µg/m3
33330	2	50	Particulate matter 10	µg/m3
33296	2	16	Nitrogen dioxide	ppb
33071	1	47	Particulate matter 1	µg/m3
33073	1	49	Particulate matter 2.5	µg/m3
33074	1	50	Particulate matter 10	µg/m3
33126	1	102	Total Volatile Organic Compounds	ppm

Figura 15: Piloto 02: Línea Ethera - Salida de metadatos

para los datos dinámicos (Figura 16), podemos ver los campos que conforman el identificador de la variable medida (structure, source e id), el valor de la propia medida y la variable temporal *calendar_id*.

Línea Ellona

Para la segunda interfaz de la cual hemos extraído los datos, la API de Ellona, obtenemos las salidas expuestas en las figuras 17 y 18.

structure	source	id	value	calendar_id
32848	0	80	1013	20231011005435
32848	0	80	1013	20231011004435
32848	0	80	1013	20231011003436
32848	0	80	1013	20231011002435
32848	0	80	1013	20231011001435
32848	0	80	1013	20231011000435
32852	0	84	22.8	20231011005435
32852	0	84	22.8	20231011004435
32852	0	84	22.8	20231011003436
32852	0	84	22.8	20231011002435
32852	0	84	22.8	20231011001435
32852	0	84	22.8	20231011000435
32840	0	72	60	20231011005435
32840	0	72	60	20231011004435
32840	0	72	60	20231011003436
32840	0	72	60	20231011002435
32840	0	72	60	20231011001435
32840	0	72	60	20231011000435
32835	0	67	458	20231011005435
32835	0	67	462	20231011004435
32835	0	67	458	20231011003436
32835	0	67	453	20231011002435

Figura 16: Piloto 02: Línea Ethera - Salida de datos dinámicos

En cuanto a los metadatos de este flujo (Figura 17), se han podido obtener únicamente en formato JSON debido a su compleja estructura interna. Es por esta estructura que, a partir de la foto es difícil ver con qué metadatos cuenta. En definitiva tenemos información de los sensores empleados, su hardware, el estado del mismo y también información sobre las variables almacenadas que se han obtenido a través de la API.

Figura 17: Piloto 02: Línea Ellona - Salida de metadatos

Por otra parte, los datos dinámicos (Figura 18) sí se han podido extraer como en ocasiones anteriores. Podemos observar cómo más o menos se repite la misma estructura en todas las salidas de los datos dinámicos. Esto es debido a que las bases de datos del DWH en las que almacenamos

estos datos tienen todas la misma estructura de tablas. Esto permite la homogeneidad del acceso a los datos de la que se ha hablado en capítulos iniciales de esta memoria. Dicho esto, observamos la variable temporal *calendar_id*, el valor de las medidas y las dos columnas cuya concatenación produce el identificador de los puntos de medida.

calendar_id	sensor_id	measurement_id	value
20231023235500	POD2-00335	env_temp	19,1125004292
20231023235500	POD2-00335	sound_leqa	49,4708339373
20231023235500	POD2-00335	co2_co2	393,6666666667
20231023235500	POD2-00338	env_temp	19,7999992371
20231023235500	POD2-00338	sound_leqa	51,1166664759
20231023235500	POD2-00338	co2_co2	398,0416666667
20231023235500	POD2-00340	env_temp	19,0375001431
20231023235500	POD2-00340	sound_leqa	52,3583337466
20231023235500	POD2-00340	co2_co2	294,25
20231023235500	POD2-00341	env_temp	21,2000007629
20231023235500	POD2-00341	sound_leqa	52,7958332698
20231023235500	POD2-00341	co2_co2	508,875
20231023235000	POD2-00335	env_temp	19,1633339564
20231023235000	POD2-00335	sound_leqa	49,5266668955
20231023235000	POD2-00335	co2_co2	394,8666666667
20231023235000	POD2-00338	env_temp	19,7999992371
20231023235000	POD2-00338	sound_leqa	51,2866664886
20231023235000	POD2-00338	co2_co2	397,3666666667
20231023235000	POD2-00340	env_temp	18,9766665777
20231023235000	POD2-00340	sound_leqa	52,4766670227
20231023235000	POD2-00340	co2_co2	293,8333333333
20231023235000	POD2-00341	env_temp	21,2000007629

Figura 18: Piloto 02: Línea Ellona - Salida de datos dinámicos

Línea Wattsense

Por último, en el tercer flujo de la transformación de este piloto, obtenemos datos a través de la API de Wattsense. Las salidas de esta interfaz pueden verse en las figuras 19 y 20. A continuación se comenta su contenido.

Como parte de los metadatos de este piloto, en la figura 19 se observan los metadatos de Wattsense. En concreto, se extraen identificadores de los equipos, puntos de medida, dispositivos y propiedad de los mismos. Además, se recoge el nombre de la magnitud y su unidad de medida.

Por último, en cuanto a los datos dinámicos de este piloto, (Figura 20), vemos que se genera la variable temporal *calendar_id*, el identificador de medida, que se usa como identificador del punto en el DWH y el valor del mismo.

equipment_id	device_id	measure_id	property	name	unit
pMmfkclavpqP5wL2	K0XRLNIG	pmmfkclavpqp5wl2-total-active-power-wbl9bmguzr	ateOGlHedVWDRtgo	Total active power	kW
pMmfkclavpqP5wL2	K0XRLNIG	pmmfkclavpqp5wl2-total-active-energy-import-y4nvyq9464	MGOT0auv4bgkKR_z	Total active energy import	kWH
OiIcI04yBGT0X4sZ	K0XRLNIG	oiici04ybgt0x4sz-total-active-power-ee23c5qdvw	W0gvDpTqUI091n1A	Total active power	kW
OiIcI04yBGT0X4sZ	K0XRLNIG	oiici04ybgt0x4sz-total-active-energy-import-zu264dz1s1	5Vw2CYH7i_BRAdaF	Total active energy import	kWH
C7_enHP7ew9rgy2A	K0XRLNIG	c7-enhp7ew9rgy2a-total-active-power-tljoj4zvbm	yVVMnvd4uFsAKzId	Total active power	kW
C7_enHP7ew9rgy2A	K0XRLNIG	c7-enhp7ew9rgy2a-total-active-energy-import-jaf61ndl8x	IRHQiN3WdOobHIkK	Total active energy import	kWH
1lFhY8BDlHcTosDM	K0XRLNIG	1lfhy8bdlhctosdm-total-active-power-mcjc11ntjd	WvYqwA9GzVHU0UXB	Total active power	kW
1lFhY8BDlHcTosDM	K0XRLNIG	1lfhy8bdlhctosdm-total-active-energy-import-9fu7uegz8e	Tms1Fp111qWDIe9v	Total active energy import	kWH
WwQ5wxdQYRelgVUF	K0XRLNIG	wwq5wxdqyreigvuf-total-active-power-szv9hp12u3	LoJl7cyJjTs4Fb3_	Total active power	kW
WwQ5wxdQYRelgVUF	K0XRLNIG	wwq5wxdqyreigvuf-total-active-energy-import-ut9txb2h5r	HH4mYUUn5lhpmkwv	Total active energy import	kWH
r7WksALh3gmQn1bg	K0XRLNIG	r7wksalh3gmqn1bg-total-active-power-wd4394x35l	xK_WB9o6f4rRYCQ1	Total active power	kW
r7WksALh3gmQn1bg	K0XRLNIG	r7wksalh3gmqn1bg-total-active-energy-import-qo55e7ribf	wYnvtzkwqlY3y7k2	Total active energy import	kWH
tHPJSLxOHMmGOSla	K0XRLNIG	thpjslxohmmgosla-total-active-power-fvon1s8m4g	rYHXXuCHjb8SDRzO	Total active power	kW
tHPJSLxOHMmGOSla	K0XRLNIG	thpjslxohmmgosla-total-active-energy-import-3qyag9be46	r1Oy2O73Zy0w5Gs8	Total active energy import	kWH
AFwWMMQl8Jhvbb2z	K0XRLNIG	afwwmmqi8jhvbb2z-total-active-power-7qs8a4m6l6	rAAFENreRCCRHZCG	Total active power	kW
AFwWMMQl8Jhvbb2z	K0XRLNIG	afwwmmqi8jhvbb2z-total-active-energy-import-21o66jci2r	0PvqK8Thl2uoPXJG	Total active energy import	kWH

Figura 19: Piloto 02: Línea Wattsense - Salida de metadatos

calendar_id	measurement_id	value
20231023235634	pmmfkclavpqp5wl2-total-active-power-wbl9bmguzr	0.0012780420947819948
20231023235634	pmmfkclavpqp5wl2-total-active-energy-import-y4nvyq9464	46.540000915527344
20231023235634	oiici04ybgt0x4sz-total-active-power-ee23c5qdvw	0.00566672720015049
20231023235634	oiici04ybgt0x4sz-total-active-energy-import-zu264dz1s1	1262.7440185546875
20231023235634	c7-enhp7ew9rgy2a-total-active-power-tljoj4zvbm	0.4838246703147888
20231023235634	c7-enhp7ew9rgy2a-total-active-energy-import-jaf61ndl8x	5013.9990234375
20231023235634	1lfhy8bdlhctosdm-total-active-power-mcjc11ntjd	0.12356743961572647
20231023235634	1lfhy8bdlhctosdm-total-active-energy-import-9fu7uegz8e	13897.978515625
20231023235628	wwq5wxdqyreigvuf-total-active-power-szv9hp12u3	0.006831463426351547
20231023235628	wwq5wxdqyreigvuf-total-active-energy-import-ut9txb2h5r	115.13099670410156
20231023235628	r7wksalh3gmqn1bg-total-active-power-wd4394x35l	0.009655522182583809
20231023235628	r7wksalh3gmqn1bg-total-active-energy-import-qo55e7ribf	1449.4410400390625
20231023235627	thpjslxohmmgosla-total-active-power-fvon1s8m4g	0.4811059832572937
20231023235627	thpjslxohmmgosla-total-active-energy-import-3qyag9be46	2525.680908203125
20231023235627	afwwmmqi8jhvbb2z-total-active-power-7qs8a4m6l6	0.3152080178260803
20231023235627	afwwmmqi8jhvbb2z-total-active-energy-import-21o66jci2r	24070.6640625
20231023234634	pmmfkclavpqp5wl2-total-active-power-wbl9bmguzr	8.601997396908 7E-4
20231023234634	pmmfkclavpqp5wl2-total-active-energy-import-y4nvyq9464	46.540000915527344
20231023234634	oiici04ybgt0x4sz-total-active-power-ee23c5qdvw	-0.00017511844635009766
20231023234634	oiici04ybgt0x4sz-total-active-energy-import-zu264dz1s1	1262.740966796875
20231023234634	c7-enhp7ew9rgy2a-total-active-power-tljoj4zvbm	0.4892124831676483

Figura 20: Piloto 02: Línea Wattsense - Salida de datos dinámicos

4.3. ETL del piloto 03 - IASI&SITTA

El piloto IASI&SITTA incluye dos edificios administrativos propiedad y operados por el Municipio de *Iasi*, la tercera ciudad más grande de Rumanía. Los edificios son el Palacio Rasnoveanu, una construcción histórica de 1880 y una ocupación de 180 personas, y la Pirámide *Dubet*, un edificio más reciente y una ocupación de 170 personas. Ambos edificios operan los días laborales en horario de mañana y tarde.

El objetivo principal del piloto es desarrollar Certificados de Desempeño Energético (EPC) más efectivos y precisos, integrados con el Libro de Registro Digital del Edificio (DBL). La digitalización y monitoreo de las instalaciones son claves para asegurar la interoperabilidad y la recopilación de datos.

Tras analizar la información de este piloto, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: Como en los pilotos anteriores, en IASI&SITTA se obtendrán perfiles de consumos junto con información del entorno para poder alcanzar de forma satisfactoria la mejora de los EPCs mencionados junto con una mejora de la digitalización de los edificios y, por consiguiente, la explotación de los datos extraídos.
- **Variedad**: Este piloto cuenta con *datasets* con información sobre calidad de aire exterior e interior (como la temperatura, la humedad, la presión y la concentración de CO_2), consumos de electricidad, aire caliente y aire frío, almacenando un total de 10 variables diferentes.

  El piloto 03 cuenta únicamente con una interfaz. Esta interfaz se corresponde con la API HTTP que ofrece la base de datos *InfluxDB* en su versión 3.0 para atender peticiones SQL. Esta base de datos es de código abierto y se ha desarrollado especialmente para facilitar el almacenamiento de series temporales [40].

  En *Kettle* no existe actualmente un *plugin* que permita integrar como fuente de datos una base de datos InfluxDBv3, por lo que se ha tenido que integrar un código en Python que lleva a cabo las siguientes funciones:

  - Instalación de las librerías necesarias para la ejecución.
  - Definición del cliente de BBDD de InfluxDBv3 con los credenciales proporcionados por el piloto.
  - Generación de certificados TLS para la petición a la API.
  - Definición de las peticiones SQL para consumo de energía eléctrica y calidad de aire especificando el periodo temporal deseado.
  - Guardado de los datos obtenidos en dos ficheros CSV.

- **Volumen**: Para el piloto 3, contamos con una producción limitada de datos alcanzando apenas los 3,5 MB al año. Pese a tener un volumen más reducido que el resto de pilotos, mantiene la exuberancia de las otras dimensiones del *big data* sin dejar de ser un piloto interesante en este aspecto.

- **Velocidad**: En este piloto, las variables que se almacenan se generan con un periodo de 15 minutos para las magnitudes de calidad del aire y 1 hora para los consumos de energía.
- **Veracidad**: El único punto de acceso a los datos almacenado en el piloto 03 es a través de la API que proporciona la base de datos *InfluxDB*. Para poder hacer uso del *endpoint* se deben tener en cuenta diferentes medidas de seguridad para el acceso a los datos, como la identificación del usuario con sus credenciales o la creación de unos certificados TLS. Por esto, consideramos que las extracciones proporcionan datos no mutados y fiables.

4.3.1. Flujo de datos

Job

En el *job* definido para este piloto (figura 21) se encuentra el bloque de ejecución del *script* de Python que se ha explicado anteriormente. Este *script* genera como salidas dos ficheros CSV. Posteriormente se ejecuta la transformación, la cual no requiere la parametrización de ninguna variable. Además, definimos la línea de alarma por *mail* en caso de algún error en la ejecución. Este *job* esta configurado con un periodo de repetición de una hora.

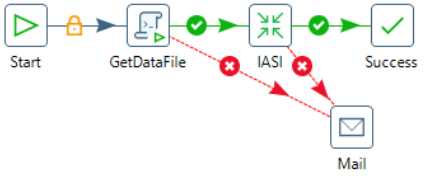

Figura 21: Job piloto 03

Transformación

La transformación asociada al piloto 03 consta de dos flujos de datos. Cada uno de ellos se corresponde con un fichero CSV extraído en el paso previo en el *job*. La primera de ellas (Figura 22), relativa al consumo eléctrico, comienza directamente con la separación de los datos estáticos y los dinámicos. Los metadatos no sufren ninguna transformación más, mientras que los datos temporales calculan el *calendar_id* antes del guardado de los datos en el DWH y en el *broker* en el *topic digibuild.03.building* (previa composición de los mensajes en formato JSON).

La otra línea se corresponde con los datos sobre calidad de aire (figura 23 inferior). Se comienza con un pivotaje de los datos, previo a la separación de los metadatos y datos temporales. De nuevo, los metadatos no sufren ninguna transformación adicional, mientras que los datos temporales concatenan dos de sus campos para la formación del identificador único de los sensores y calculan el *calendar_id*. En la línea de calidad de aire se exportan los datos al DWH.

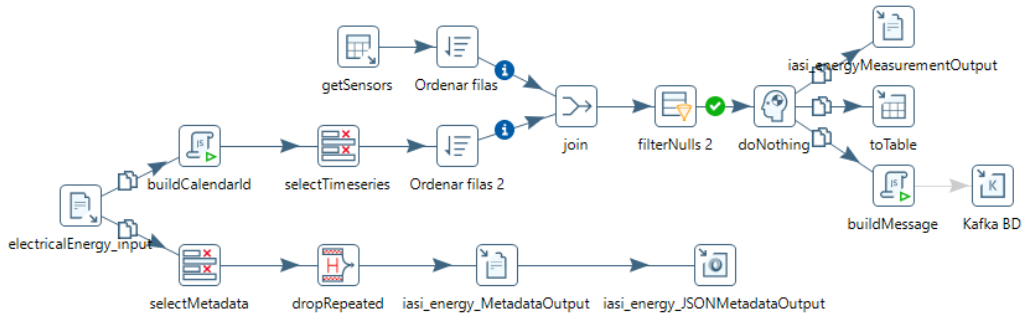

Figura 22: ETL piloto 03 - Línea Energía

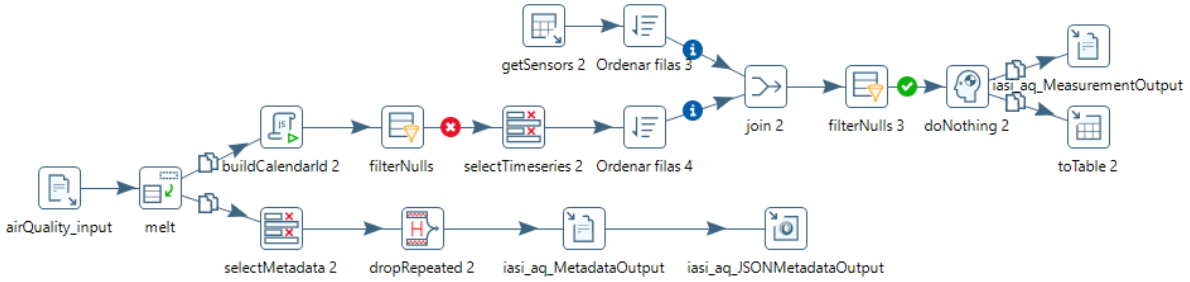

Figura 23: ETL piloto 03 - Línea Calidad Aire

4.3.2. Salidas

En para este piloto, la conexión con la base de datos en el flujo de datos de energía se produjo posterior al desarrollo de la transformación, por lo cual no se obtuvo una muestra de la salida producida. A pesar de ello, se ha expuesto que el formato de origen y las transformaciones son los mismos en ambos casos, por lo que podemos asegurar que el formato de la salida será igual en ambos flujos.

Como resultado de la línea de calidad de aire, se obtienen dos salidas de datos, uno de metadatos y otro de datos dinámicos. A continuación, procedemos a describir estas salidas.

En lo referente a los metadatos (Figura 24) vemos diferentes campos relativos a los dispositivos de medida. Entre ellos se encuentran, el identificador del sensor (dirección MAC), fecha de última actualización, tipo de dispositivo, nombre del módulo (nombre de la habitación), disponibilidad del dispositivo, listado de magnitudes medibles, localización, nombre de la estación, identificador del módulo (dirección MAC), fecha de actualización del módulo.

Por otro lado, para los datos dinámicos (Figura 25), podemos ver la MAC del dispositivo, que se usará como identificador del punto concatenado junto con el nombre de la magnitud, el valor de la propia medida y la variable temporal *calendar_id*.

device_id	device_last_setup	device_type	device_module_name	device_reach...	device_data_type
70:ee:50:7b:2e:ea	1696493034	NAMain	C5	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2e:ea	1696493034	NAMain	C5	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2f:52	1696764975	NAMain	C18	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2f:0e	1696770261	NAMain	C31	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2f:0e	1696770261	NAMain	C31	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2e:72	1696580580	NAMain	C11	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2e:72	1696580580	NAMain	C11	true	["Temperature","CO2","Humidity","Noise","Press...
70:ee:50:7b:2f:50	1696783863	NAMain	C14	true	["Temperature","CO2","Humidity","Noise","Press...

device_place	station_name	home_id	home_name	module_id	module_last_setup
{"altitude":102,"city":"Buftea","country":"RO","tim...	Ras (C5)	651e6dea2315c6b608075661	Ras	02:00:00:36:9d:88	1696493103
{"altitude":102,"city":"Buftea","country":"RO","tim...	Ras (C5)	651e6dea2315c6b608075661	Ras	03:00:00:0d:20:8e	1696494145
{"altitude":102,"city":"Buftea","country":"RO","tim...	Ras (C18)	651e6dea2315c6b608075661	Ras	02:00:00:37:07:64	1696765017
{"altitude":102,"city":"Buftea","country":"RO","tim...	Ras (C31)	651e6dea2315c6b608075661	Ras	02:00:00:37:10:6c	1696770276
{"altitude":102,"city":"Buftea","country":"RO","tim...	Ras (C31)	651e6dea2315c6b608075661	Ras	03:00:00:0c:df:98	1696770370
{"altitude":102,"city":"Buftea","country":"RO","tim...	Dub (C11)	651fc3e450e1e18cf10830d8	Dub	02:00:00:63:00:86	1696580596
{"altitude":102,"city":"Buftea","country":"RO","tim...	Dub (C11)	651fc3e450e1e18cf10830d8	Dub	03:00:00:0d:48:6e	1696581183
{"altitude":102,"city":"Buftea","country":"RO","tim...	Dub (C14)	651fc3e450e1e18cf10830d8	Dub	03:00:00:0d:3e:38	1696784001

Figura 24: Piloto 03: Línea Calidad de Aire - Salida de metadatos

calendar_id	device_id	temp	hum	co2	noise	pres
20231121001500	70:ee:50:7b:2e:ea	20.3	54	745	56	1017.2
20231121001500	70:ee:50:7b:2e:ea	20.2	54	728	55	1017.4
20231121001500	70:ee:50:7b:2e:ea	20.1	54	720	57	1017.4
20231121001500	70:ee:50:7b:2e:ea	20.1	54	724	56	1017.3
20231121001500	70:ee:50:7b:2e:ea	20	54	711	37	1017.2
20231121001500	70:ee:50:7b:2e:ea	19.9	54	699	37	1017.2
20231121001500	70:ee:50:7b:2e:ea	19.8	54	698	37	1017
20231121001500	70:ee:50:7b:2e:ea	19.7	55	690	37	1017.2
20231121001500	70:ee:50:7b:2e:ea	19.8	55	684	37	1017.2
20231121001500	70:ee:50:7b:2e:ea	20.4	54	652	37	1017.3
20231121001500	70:ee:50:7b:2e:ea	20.5	54	667	37	1017.4
20231121001500	70:ee:50:7b:2e:ea	20.3	54	627	37	1017.5
20231121001500	70:ee:50:7b:2e:ea	20.3	54	623	37	1017.7
20231121001500	70:ee:50:7b:2e:ea	20.6	53	615	37	1017.8
20231121001500	70:ee:50:7b:2e:ea	20.5	53	606	37	1017.9
20231121001500	70:ee:50:7b:2e:ea	20.3	53	626	37	1018.1

Figura 25: Piloto 02: Línea Calidad de Aire - Salida de datos dinámicos

4.4. ETL del piloto 04 - VEOLIA

El piloto 04 cuenta con dos distritos de redes de calefacción separados geográficamente. Estos distritos registran datos a nivel de BEMS y DEMS. Los objetivos de este piloto son, principalmente, la mejora de la calidad de los datos que actualmente se monitorizan en ambos distritos y, como producto de este objetivo, se propone como meta alcanzar el punto óptimo de operación para los distritos energéticos a través de los gemelos digitales, en definitiva, sus recursos energéticos.

En ambos distritos energéticos, tenemos diferentes objetivos. Comenzando por la recopilación de datos, integrando datos dinámicos y estáticos de múltiples fuentes en un sistema unificado. Además, optimización energética, mejorando la eficiencia energética mediante el análisis avanzado de datos, así como implementar sistemas de gestión que optimicen el uso de recursos energéticos. Finalmente, contribuir a la sostenibilidad ambiental reduciendo el consumo energético y las emisiones de carbono. Todos estos objetivos se consiguen mediante la adopción de tecnologías avanzadas para la transformación digital en la gestión de infraestructuras.

4.4.1. Dimensiones del *big data*

Tras analizar la información sobre VEOLIA, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: El valor o la información que obtendremos de los datos de ambos distritos serán patrones de necesidad de calefacción en función del tamaño de los propios distritos, del precio del gas y de parámetros ambientales externos, principalmente la temperatura y la radiación solar. De esta forma, se podrán alcanzar los objetivos propuestos para el piloto.
- **Variedad**: El distrito de FASA monitoriza mas de 750 variables, mientras que el de Río Vena, sólo 15. Estas variables comprenden *datasets* de las diferentes partes de las redes de calor, como la sala de calderas, datos meteorológicos, de la subestación y del sistema de generación fotovoltaica.
- **Volumen**: A diferencia del caso anterior y dado el elevado número de variables relevantes del piloto, el volumen de datos que se genera asciende a aproximadamente 290 MB mensuales o 3,5 GB al año.
- **Velocidad**: Las diferentes variables para le piloto 4 se muestrean cada 15 minutos. En este piloto, se cuenta con una única interfaz de datos, como en el caso del anterior piloto. Esta vez el protocolo que implementa esta interfaz es SFTP, que internamente emplea peticiones SQL para acceder a las tablas donde almacenan los datos. Para cada distrito se genera un fichero CSV diariamente conteniendo los registros de todas sus variables. Estos ficheros son publicados en el servidor SFTP de los edificios del piloto con una frecuencia, independientemente de la frecuencia de muestreo que tenga configurada cada punto de recogida de datos del sistema.

- **Veracidad**: Se ha mencionado el uso del protocolo SFTP, el cual se emplea en la transferencia de ficheros de forma segura, además de solicitar credenciales proporcionados por el responsable técnico del piloto.

4.4.2. Flujos de datos

Job

El *job* asociado al piloto 04 (figura 26) consta de dos flujos. En cada uno de los flujos de lleva a cabo la conexión y extracción de los ficheros CSV mencionados en el párrafo anterior. Tras la descarga de los datos energéticos, se procede a la llamada de los dos ficheros ETL generados. Para ambas líneas se ha definido la alarma vía *e-mail* en caso de error en la ejecución del *job*. Limitados por la frecuencia de publicación de dichos ficheros en el servidor FTP del piloto, se ha definido un periodo de ejecución para el *job* de un día.

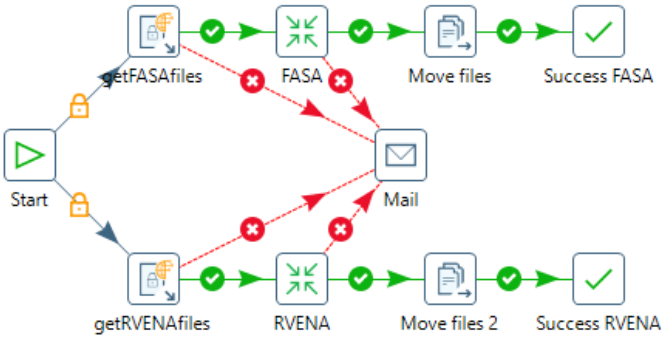

Figura 26: Job piloto 04

Transformación

Pese a ser dos transformaciones diferentes (Figuras 27 y 28), el formato de origen de los datos y, por tanto, el procesamiento es similar, por lo que se explicará simultáneamente para ambos casos.

Tras la importación de los datos almacenados en los ficheros CSV que se han extraído en el *job* del servidor *SFTP*, comenzamos cambiando el caracter empleado como punto decimal, de coma a punto (“,” por “.”). Después, se separan algunos de los campos en diferentes subcampos y es a partir de aquí cuando podemos diferenciar entre la parte de metadatos y la parte de datos dinámicos del flujo. Tras la separación, en la parte de los metadatos se vuelven a separar diferentes campos y se eliminan las columnas originales para dar lugar a la versión final de los mismos y se procede a eliminar las filas que se repiten. En la parte de los datos dinámicos, se filtran las variables sólo relevantes para su almacenamiento y uso. Este filtrado se lleva a cabo mediante una petición SQL a la tabla del DWH donde se almacena el listado de variables de VEOLIA

FASA y VEOLIA Río Vena y la unión de los flujos y el filtrado de las filas que no encuentren una coincidencia entre la lista de puntos del DWH y las variables extraídas de la API. Finalmente, los datos temporales se vuelcan en el DWH así como en los *topics digibuild.04a.district* y *digibuild.04b.district* tras la formación de los mensajes en formato JSON.

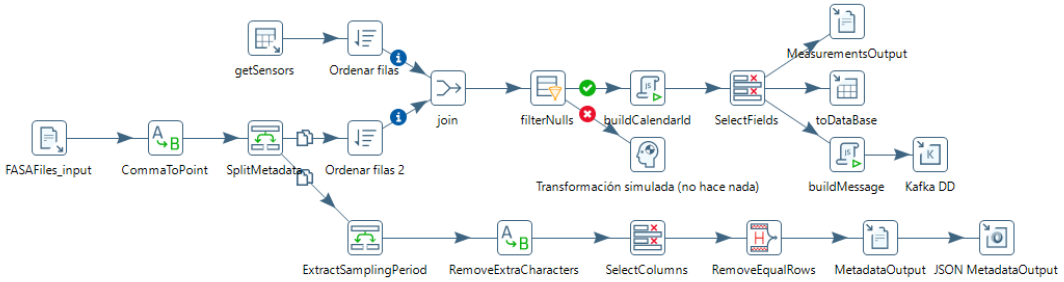

Figura 27: ETL piloto 04a

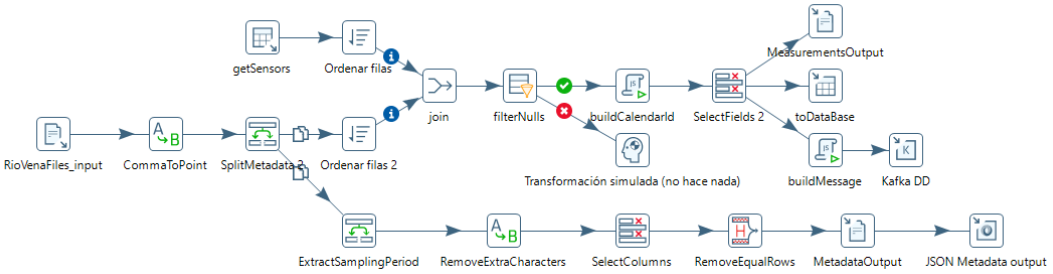

Figura 28: ETL piloto 04b

4.4.3. Salidas

Para los dos distritos del piloto 04 obtenemos salidas similares, tanto de datos dinámicos como de metadatos. Estas salidas pueden verse recogidas en las figuras 29 y 30.

Comenzando con los metadatos (Figura 29), observamos diferentes campos que nos dan información sobre las variables recogidas y el sistema que las gestiona. Estas variables son el identificador interno de los puntos de medida, el nombre de las variables, identificadores del distrito, nombre del distrito y los periodos de muestreo de las diferentes variables medidas. Este periodo de muestreo es de 15 minutos en FASA, mientras que 30 minutos en Río Vena.

Por otra parte, para los datos dinámicos (Figura 30) podemos apreciar una estructura que se repite en los piloto anteriores. Esta estructura la conforman los campos de identificador de variable (que es directamente obtenido de los flujos de datos), los valores de los puntos de medida y la variable temporal *calendar_id*.

VariableId	Variable	Metadata1	Metadata2	Demosite	SamplingPeriod
47212	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47221	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47310	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47319	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47401	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47410	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47498	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47507	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47594	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47603	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47688	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47697	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47781	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47790	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47874	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
47883	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
49150	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto
49159	POTENCIA CONTAD 1º ACS	EDEN1D	M209026	POBLADO DE FASA	15 minuto
48252	ENERGIA CONTADOR 1º CALEFACC	EDEN1D	M209026	POBLADO DE FASA	15 minuto

VariableId	Variable	Metadata1	Metadata2	Demosite	SamplingPeriod
3776	CONSUMO GAS	EDEN1C	M025824	C.P. RIO VENA	30 minutos
3767	ENERGIA ACUMULADA	EDEN1C	M025824	C.P. RIO VENA	30 minutos

Figura 29: Piloto 04: Líneas FASA y Río Vena - Salida de metadatos

VariableId	Value	Demosite	calendar_id
47212	252.40	POBLADO DE FASA	20210501000000
47212	252.40	POBLADO DE FASA	20210501001500
47212	252.40	POBLADO DE FASA	20210501003000
47212	252.40	POBLADO DE FASA	20210501004500
47212	252.40	POBLADO DE FASA	20210501010000
47212	252.40	POBLADO DE FASA	20210501011500
47212	252.40	POBLADO DE FASA	20210501013000
47212	252.40	POBLADO DE FASA	20210501014500
47212	252.40	POBLADO DE FASA	20210501020000
47212	252.40	POBLADO DE FASA	20210501021500
47212	252.40	POBLADO DE FASA	20210501023000
47212	252.40	POBLADO DE FASA	20210501024500
47212	252.40	POBLADO DE FASA	20210501030000
47212	252.40	POBLADO DE FASA	20210501031500
47212	252.40	POBLADO DE FASA	20210501033000
47212	252.40	POBLADO DE FASA	20210501034500
47212	252.40	POBLADO DE FASA	20210501040000
47212	252.40	POBLADO DE FASA	20210501041500
47212	252.40	POBLADO DE FASA	20210501043000

VariableId	Value	Demosite	calendar_id
3776	3406782.00	C.P. RIO VENA	20210101000000
3776	3406782.00	C.P. RIO VENA	20210101003000
3776	3406782.00	C.P. RIO VENA	20210101010000
3776	3406782.00	C.P. RIO VENA	20210101013000
3776	3406782.00	C.P. RIO VENA	20210101020000
3776	3406782.00	C.P. RIO VENA	20210101023000
3776	3406782.00	C.P. RIO VENA	20210101030000
3776	3406782.00	C.P. RIO VENA	20210101033000
3776	3406782.00	C.P. RIO VENA	20210101040000
3776	3406782.00	C.P. RIO VENA	20210101043000
3776	3406782.00	C.P. RIO VENA	20210101050000
3776	3406814.00	C.P. RIO VENA	20210101053000
3776	3406891.00	C.P. RIO VENA	20210101060000
3776	3406905.00	C.P. RIO VENA	20210101063000
3776	3406905.00	C.P. RIO VENA	20210101070000
3776	3406905.00	C.P. RIO VENA	20210101073000
3776	3406905.00	C.P. RIO VENA	20210101080000
3776	3406905.00	C.P. RIO VENA	20210101083000
3776	3406905.00	C.P. RIO VENA	20210101090000

Figura 30: Piloto 04: Líneas FASA y Río Vena - Salida de datos dinámicos

4.5. ETL del piloto 05a - EMOT

El piloto 05a o EMOT está ubicado en la región de Umbría, Italia, gestionado por una empresa del sector de reparación de automóviles. Una de sus *spin-offs*, actúa como operador de puntos de carga y proveedor de servicios de movilidad eléctrica. El objetivo principal del piloto es la gestión de la recarga de energía y la energía del vehículo eléctrico (EV). El piloto se desarrolla en un edificio terciario con una ocupación de 35 personas y un consumo anual de 90 MWh. Además, cuenta con una planta fotovoltaica (PV) de 50 kW y dos estaciones de carga inteligentes equipadas con dos conectores de 22 kW.

Sobre los objetivos de este piloto como parte del proyecto de DigiBUILD, cabe destacar la gestión eficiente de la recarga de energía para vehículos eléctricos, así como la integración de la generación de energía solar con el consumo y la recarga de vehículos eléctricos. Además, la mejora del monitoreo y la optimización del consumo energético del edificio junto con el desarrollo de modelos de energía del edificio (BEM) para mejorar la eficiencia energética y la mejora de la calidad del aire interior (IAQ) del mismo.

4.5.1. Dimensiones del *big data*

Tras analizar la información sobre EMOT, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: Al igual que en pilotos analizados con anterioridad, el valor que extraemos de los datos debe se útil en la consecución de los objetivos para este piloto. Es por esto que la forma más directa de generar información a partir de los datos sería la extracción de patrones de consumo por días, horario o estación, tanto para los cargadores de EV como de optimización de confort y consumos del edificio bajo estudio.
- **Variedad**: Los datos recopilados incluyen el consumo de energía del edificio, la calidad del aire interior, la generación fotovoltaica, las estaciones de carga y los vehículos eléctricos, así como información de BIM y BEM.

  En este piloto se cuenta con diferentes posibilidades atendiendo a los protocolos implementados para la extracción de los datos disponibles, siendo estas opciones mediante transferencia de ficheros CSV, a través de API o a través de MQTT. Se detectaron problemas en la integración de los flujos de datos mediante MQTT. Por otro lado, la integración mediante ficheros CSV implicaría un flujo de datos bastante alejado del tiempo real. Teniendo en cuenta estas dos razones, se ha procedido a la integración de todas las fuentes de datos mediante la API pública que ofrece el piloto.
- **Volumen**: El piloto genera más de 1,2 GB de datos al año debido a la alta frecuencia de medición.
- **Velocidad**: Se ha mencionado las diferentes fuentes de datos que componen este caso de uso. Debido a ello, las frecuencias de generación son muy variadas, comprendiendo desde medidas cada segundo hasta cada 15 minutos. Como resultado, se decidió unificar la

frecuencia de recogida de los datos, haciendo un filtrado en base al tiempo, obteniendo las telemetrías de los últimos 10 minutos.

- **Veracidad**: Con respecto a la dimensión veracidad, como en casos anteriores, los datos son guardados y protegidos por el responsable del piloto. Los datos son accesibles de forma abierta, sin necesidad de emplear credenciales. Debido a que el piloto sufrió varias instancias de denegación de servicio, decidieron limitar la frecuencia de las peticiones a los *endpoints* de la API temporalmente. Por último, podemos asegurar que la fuente de datos es segura debido al uso de la versión segura del protocolo HTTP.

4.5.2. Flujos de datos

Job

La ejecución de la *job* (figura 31) consta únicamente del bloque de ejecución de la transformación, contando como parámetro la frecuencia de repetición del proceso. No se requiere ningún parámetro adicional dado que la API no cuenta con ningún tipo de identificación o autenticación de usuarios. También se define el caso de error, enviando una alarma mediante un correo electrónico. El periodo de repetición se ha ajustado a 10 minutos teniendo en cuenta la frecuencia de generación de datos para los tres *endpoints*.

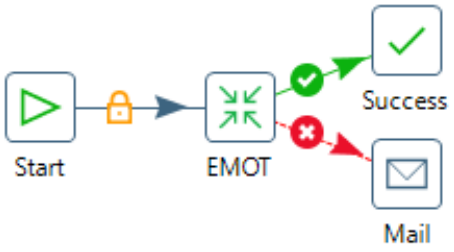

Figura 31: Job piloto 05a

Transformación

La transformación asociada a este piloto cuenta con tres líneas de datos (Figuras 32, 33 y 34). Cada una de ellas tiene como fuente de datos los *endpoints* mencionados anteriormente. Los procesamientos llevados a cabo en los tres casos son muy similares, pero se procederá a su descripción de forma individual explicando las particularidades de cada uno de ellos.

La extracción de datos se lleva a cabo con peticiones HTTP a tres *endpoints* diferentes de la API. Cada uno de ellos se corresponde con un *dataset* dentro del piloto (estaciones de carga de vehículo eléctrico, generación fotovoltaica y datos de edificio). Esta recolección no permite

el filtrado temporal de las muestras recogidas, por lo que se lleva a cabo en la transformación, como un paso más de la misma.

Para la línea de datos sobre producción fotovoltaica del edificio, comenzamos con la sustitución de las variables pasadas desde el *job*, como son las credenciales y la frecuencia de muestreo. Posteriormente se lleva a cabo la petición HTTP al *endpoint* correspondiente de la API. Después, se hace la extracción de los campos y subcampos relevantes de la respuesta en formato JSON y se procede al filtrado temporal de los mismos, teniendo un horizonte temporal pasado que se especifica como parámetro en el *job*, como se ha mencionado previamente. Después, se calcula el *calendar_id* y se lleva a cabo un pivotaje de los datos para dotarlos de una estructura conveniente por columnas. A partir de este punto, se separan los metadatos y los datos dinámicos a través de la unión de los datos tratados y la extracción del listado de variables de nuestra tabla del DWH. Finalmente se exportan los datos tanto al DWH como al flujo de datos en tiempo real de *Kafka* en el *topic digibuild.05a.pv*.

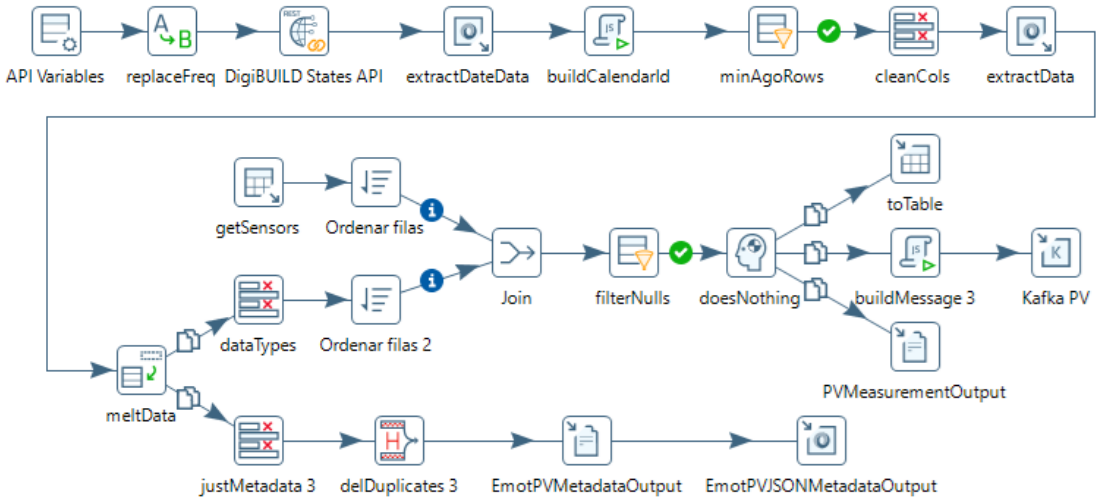

Figura 32: ETL piloto 05a - Línea producción PV

En el segundo flujo (Figura 34) comienza de forma similar a la línea anterior: se sustituye el valor de los argumentos pasados desde el *job* y se realiza la petición a la API de EMOT. Después, se extraen los campos y subcampos pertinentes y se filtra temporalmente y calculando el *calendar_id*. A continuación, se separan los metadatos de las series temporales y estos últimos sufren un pivotaje y se concatenan los campos de nombre de la torre y nombre del sensor para formar un identificador único. Por último, son filtrados de la misma forma mencionada para la línea anterior y son introducidos en el DWH y en el *topic digibuild.05a.building* del *broker*.

Por último, el flujo sobre datos de estaciones de carga de vehículo eléctrico (Figura 34) se recogen las variables del *job* y se extraen los datos de vehículo eléctrico de la API, como en las líneas anteriores. Posteriormente, se extraen los campos y subcampos del JSON de respuesta obtenido y se procede a la sustitución de variable de texto con valores “on” y “off” por 1 y 0 respectivamente. Después se calcula el *calendar_id*, se pivotan los datos y se concatenan dos de

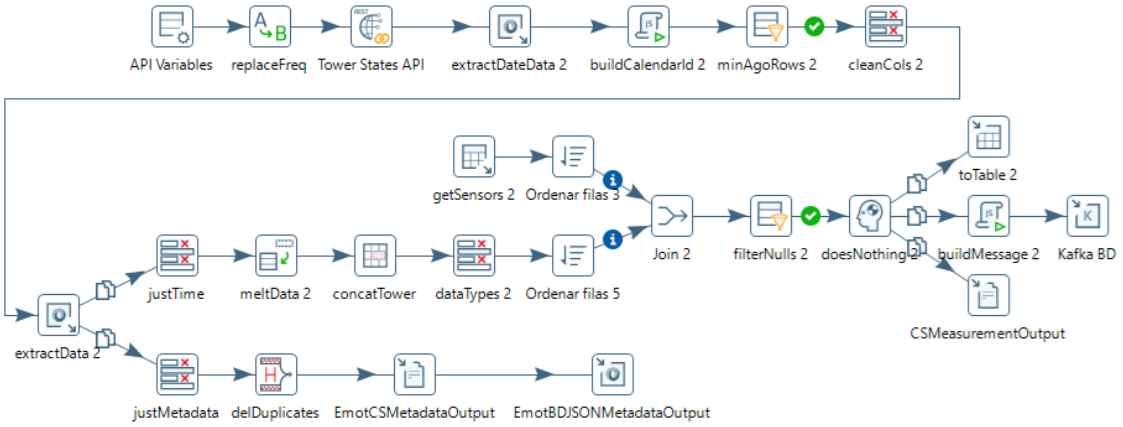

Figura 33: ETL piloto 05a - Línea consumo edificio

los campos extraídos para formar el identificador único de los sensores y dotar a los datos de la estructura conveniente para su almacenamiento. Para finalizar, tras separar los metadatos del flujo de datos dinámicos, éstos últimos sufren un filtrado a partir del listado obtenido con una petición SQL de la tabla del piloto de EMOT en el DWH y se vuelcan los datos tanto en el DWH como en el *topic digibuild.05a.ev* de *Kafka*.

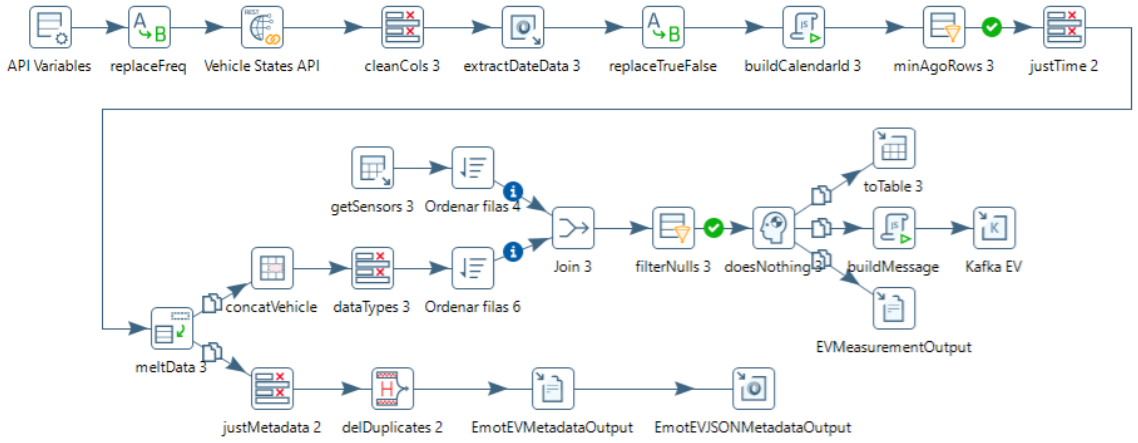

Figura 34: ETL piloto 05a - Línea vehículo eléctrico

4.5.3. Salidas

En este apartado procedemos a explicar las salidas de las diferentes líneas que conforman los flujos de datos del piloto de EMOT.

En el momento de desarrollo de las ETLs para el piloto 05a no se poseía la información para

el acceso a los *endpoints* de los flujos de datos del edificio ni de producción fotovoltaica. Es por esto que se mostrará la salida del flujo de datos de las estaciones de carga. La figura 35 nos muestra un extracto de dicha salida. En ella se pueden observar columnas con una estructura similar a lo expuesto en anteriores casos, como un identificador de medida, en este caso el número de la estación de carga. Además, del valor de las medidas y el instante temporal en el que han sido tomadas con el formato definido al comienzo del capítulo de esta memoria.

tower_id	plug1	power1	plug2	power2	calendar_id
39	59	0	60	0	20231114120228
55	85	0	<null>	<null>	20231114120228
39	59	0	60	0	20231114120229
55	85	0	<null>	<null>	20231114120229
39	59	0	60	0	20231114120230
55	85	0	<null>	<null>	20231114120230
39	59	0	60	0	20231114120231
55	85	0	<null>	<null>	20231114120231
39	59	0	60	0	20231114120232
55	85	0	<null>	<null>	20231114120232
39	59	0	60	0	20231114120233
55	85	0	<null>	<null>	20231114120233
39	59	0	60	0	20231114120234
55	85	0	<null>	<null>	20231114120234
39	59	0	60	0	20231114120235
55	85	0	<null>	<null>	20231114120235
39	59	0	60	0	20231114120236
55	85	0	<null>	<null>	20231114120236
39	59	0	60	0	20231114120237

Figura 35: Piloto 05a: Línea CS - Salida de datos dinámicos

4.6. ETL del piloto 05b - FOCCHI

El piloto de FOCCHI se desarrolla en un edificio de las oficinas centrales de la fábrica de FOCCHI en Rimini, Italia. Este piloto emplea dos sistemas principales de BMS para recopilar datos de sensores ambientales (temperatura, humedad, luminancia, presencia, etc.) y medidores de energía. Dada la fragmentación de estos sistemas.

Los objetivos principales del piloto incluyen la homogeneización de los servicios de monitoreo para enfrentar estos desafíos, optimizando el uso de energía y proporcionando a los gerentes de Recursos Humanos y de Instalaciones análisis para verificar y asegurar un mayor nivel de bienestar para los ocupantes .

4.6.1. Dimensiones del *big data*

Tras analizar la información sobre el piloto 05, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: Se pretende generar una mejora en la comodidad en el acceso a la información gracias a la homogeneización de los datos, así como mejoras en la visualización y, por tanto, en el control ambiental y consumo gracias a los *Digital Twins*, además de poder producir perfiles de comportamiento en los trabajadores de las oficinas.
- **Variedad**: Se cuenta con multitud de fuentes de datos diferentes dentro de varios edificios, conformando los siete *datasets* disponibles. Existen dos sistemas de gestión de edificios (BMS) principales que recopilan datos dinámicos e históricos de consumo energético, producción de energía fotovoltaica, parámetros ambientales internos y externos, y retroalimentación de usuarios.

  En total se alcanzan unas 350 variables diferentes. Estos *datasets* no son accesibles desde una misma interfaz. La extracción de datos de este piloto se centra principalmente en 4 de estas interfaces. Estas interfaces son accedidas empleando los procedimientos que se describen a continuación.
- **Volumen**: La cantidad de datos generados en este piloto es bastante grande, pero no se especifica por parte de los responsables del mismo de forma concreta. Las interfaces de extracción de los datos tienen frecuencias de muestreo elevadas y variadas, proporcionando datos con una granularidad desde cada segundo hasta cada hora. Gracias al conocimiento de la frecuencia y la cantidad de variables podemos estimar que no será uno de los pilotos que menos volumen de datos produzca dentro de DigiBUILD.
- **Velocidad**: La generación de datos en cada uno de los flujos tienen frecuencias muy variables, como hemos mencionado en la dimensión anterior recorren desde el segundo hasta la hora. En algunos de ellos, como el que emplea MQTT, se generan prácticamente cada segundo, mientras que la generación de los ficheros CSV en la línea de *Google Drive* se generan diariamente. Es por esto que se ha decidido unificar la frecuencia de la extracción de los datos en todas las interfaces, ajustándola a 1 vez al día.

- **Veracidad**: Como en anteriores ocasiones, los datos provienen de servidores custodiados y mantenidos por el responsable del piloto. Su equipo son los únicos con capacidades de gestionar los accesos a los datos y, por consiguiente, velar por su correcto funcionamiento y capacidad de medición. Además, los accesos a las diferentes interfaces se llevan cabo empleando pórtalos estándar seguros, como HTTPS o MQTTS.

4.6.2. Flujos de datos

Job

El *job* correspondiente al piloto 05b (Figura 36) está formado por dos bloques de ejecución de *scripts* externos para la extracción de datos para varios de los flujos que se explicarán a continuación. Posteriormente, se llama a la transformación, pasándole como parámetros la clave de la API y los credenciales de dichos flujos, para la API y el *broker*. Observamos en la figura que si en alguno de estos pasos se produce algún tipo de error, se informará mediante un *e-mail* a los administradores configurados.

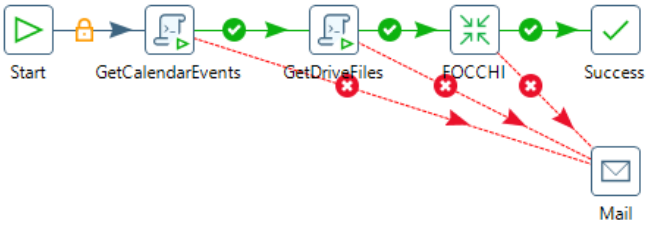

Figura 36: Job piloto 05b

Transformación

La primera interfaz de datos (Figura 37) se corresponde con la API que proporciona datos de generación fotovoltaica. La transformación comienza con la obtención de los parámetros pasados desde el *job*, siendo estos la frecuencia de ejecución y los credenciales de acceso a las APIs. En esta interfaz se extraen los datos de dos *endpoints* de una API, la cual requiere autenticación que mencionamos. Tras las llamadas a la API se extraen los campos y subcampos del JSON que son relevantes y se unifican los flujos. Después, Se procede a formatear los datos aplicando filtrado de valores nulos, haciendo un pivotaje y formando los campos que nos interesan mediante concatenación de diferentes columnas del flujo. También, se procede a la generación del parámetro temporal *calendar_id*. A continuación, se separan los datos dinámicos de los metadatos y se filtran las variables necesarias mediante una petición SQL a la tabla del DWH donde se almacena el listado de variables de FOCCHI, así como con la unión de los flujos y el filtrado de las filas que no encuentren una coincidencia entre la lista de puntos del DWH y

las variables extraídas de la API. Finalmente, los datos temporales se vuelcan en el DWH. De esta interfaz no se exporta ninguna variables al *broker Kafka*.

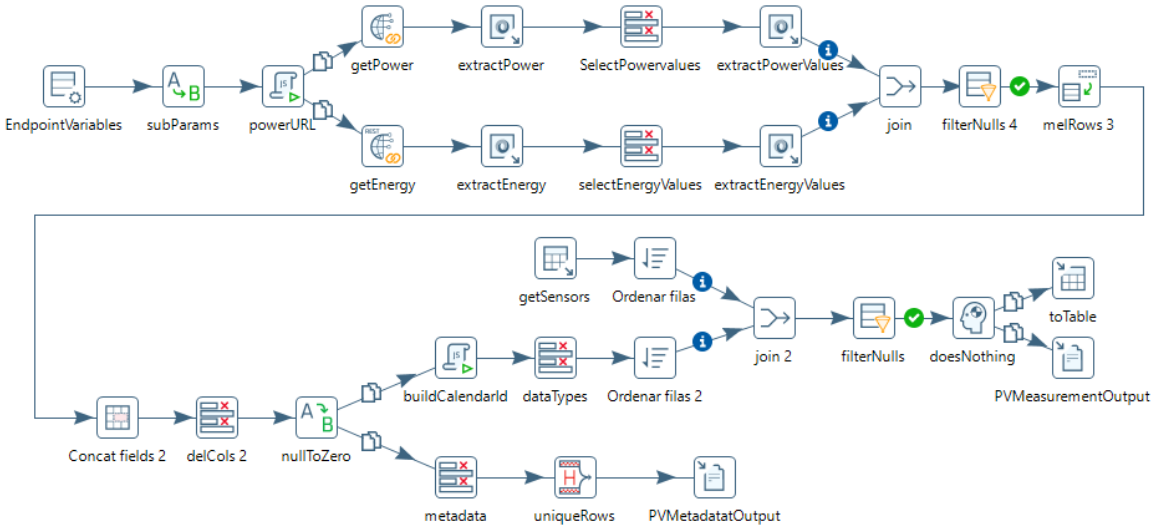

Figura 37: ETL piloto 05b - Línea generación PV

Para el segundo flujo (Figura 38), extraemos los mismos argumentos que pasa el *job*. Después, extraemos los datos de otra API que requiere autenticación y ésta genera un *token* de sesión al hacer una consulta a un *endpoint* y autenticarnos. Tras la obtención de este *token*, procedemos a la extracción de los datos en un *endpoint* diferente. Ahora se procede al formateo de los datos originales extrayendo los campos relevantes de la respuesta de la API y se transforman los valores de las variables *booleanas* a un equivalente numérico, se forman los campos relevantes mediante la concatenación de diferentes columnas y se lleva a cabo un pivotaje para poder tener una estructura de los datos en columnas. Tras ello, se separa la parte de metadatos del flujo. Finalmente, procedemos al filtrado de los puntos que se han definido en el DWH y su posterior exportación al mismo DWH y a en *Kafka* en los *topics* de las variables en datos de edificio (*digibuild.05b.building*) y de calidad de aire (*digibuild.05b.airquality*). El filtrado se hace tras obtener un listado de sensores relevantes del piloto y cruzarlo con los sensores obtenidos en la API.

El tercer flujo (Figura 39) consiste en una fuente de datos formada por un *broker* MQTT donde se publican, haciendo uso de un *token* para la autenticación, los mensajes con contenido de la misma índole que en la *API* anterior. Tras dicha extracción, obtenemos una respuesta con datos en formato JSON y se aplican filtros consecutivos para la obtención de los parámetros específicos dependiendo del tipo de sensor que los haya generado embebidos en subcampos de la estructura JSON. En cada sub-flujo que hemos generado se aplica un pivotaje de los datos para conseguir una estructura pura en columnas y se unifican todos ellos. Después, generamos el parámetro temporal *calendar_id*. Posteriormente, se filtran los puntos registrados en el DWH mediante el cruzamiento del listado de sensores almacenados en el DWH y los obtenidos del *broker*. Por último, se exportan al mencionado DWH y en *Kafka* en el *topic digibuild.05b.building*. Cabe destacar que en este flujo no extraemos ningún metadato.

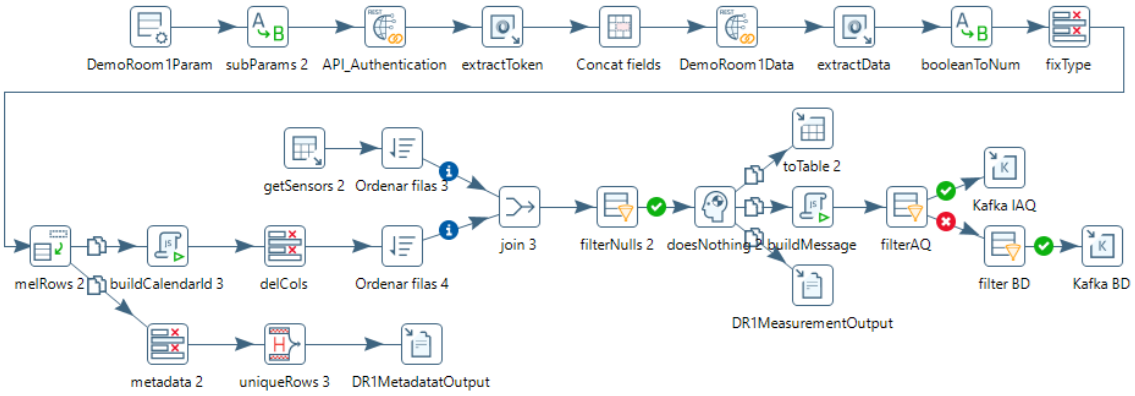

Figura 38: ETL piloto 05b - Línea sala 1

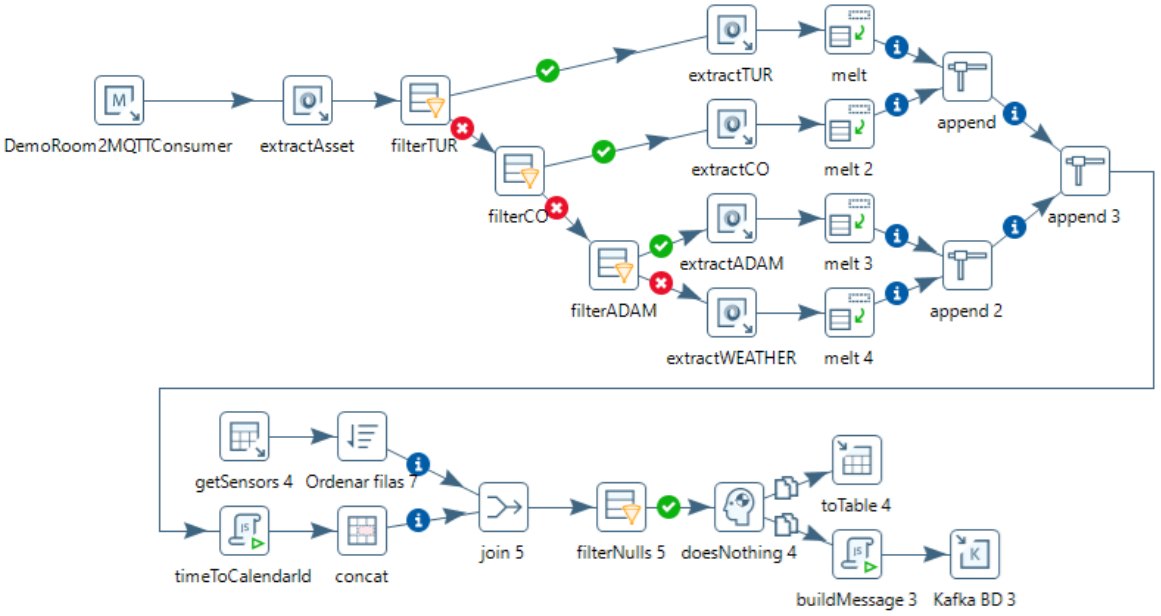

Figura 39: ETL piloto 05b - Línea sala 2

En el cuarto y último flujo (Figura 40), se extraen los datos mediante un *script* que se ejecuta en el *job* mediante una autenticación en la API de *Google Drive*. Los datos extraídos en formato CSV contienen información de consumos energéticos de diferentes partes del edificio de aplicación. Tras la importación de los datos, se eliminan las columnas no relevantes para el proyecto y se transforma el separador decimal de "," a ".". Después se genera el parámetro *calendar_id* y se pivota la tabla de datos resultante. Como últimos pasos del flujo se procede a filtrar únicamente los sensores registrados y se exportan los datos limpios y formateados al DWH y al *topic* de *Kafka digibuild.05b.building*.

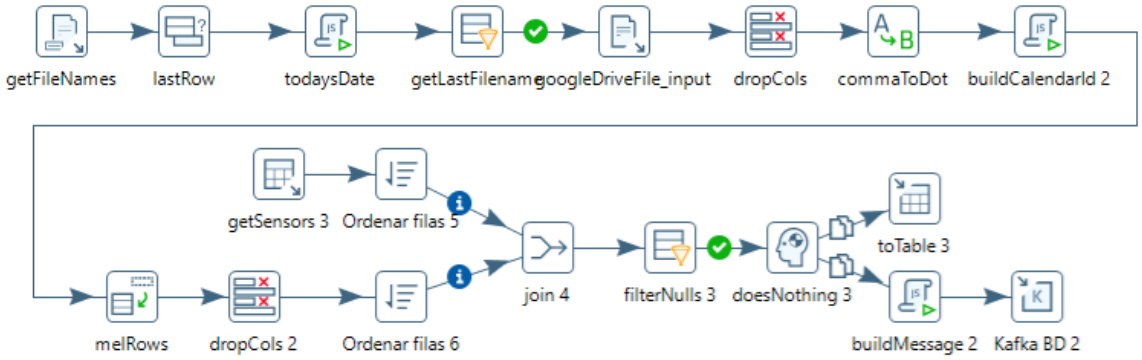

Figura 40: ETL piloto 05b - Línea Google Drive

Salidas

En este apartado procedemos a explicar las salidas de las diferentes líneas que conforman los flujos de datos del piloto de FOCCHI. De nuevo, cabe destacar que las figuras presentadas son muestras parciales del proceso de desarrollo de las ETLs y no enseña el formato final que se ha elegido en el procesamiento de los datos generados.

En la figura 41 se observa una muestra de datos dinámicos extraída de la interfaz de la línea de generación fotovoltaica del piloto 05b. En ella se aprecian las columnas de *Type* y *sensor_ID*, empleadas como identificador de las variables extraídas de energía y potencia, una columna que contiene el valor de la medida (todas ellas son ceros debido a que son en hora de la madrugada) y, por último, la variable temporal.

Type	sensor_ID	value	calendar_id
Power	2672996	0.0	20231001000000
Energy	2672996	0.0	20231001000000
Power	2672996	0.0	20231001001500
Energy	2672996	0.0	20231001001500
Power	2672996	0.0	20231001003000
Energy	2672996	0.0	20231001003000
Power	2672996	0.0	20231001004500
Energy	2672996	0.0	20231001004500
Power	2672996	0.0	20231001010000
Energy	2672996	0.0	20231001010000
Power	2672996	0.0	20231001011500
Energy	2672996	0.0	20231001011500
Power	2672996	0.0	20231001013000
Energy	2672996	0.0	20231001013000
Power	2672996	0.0	20231001014500
Energy	2672996	0.0	20231001014500
Power	2672996	0.0	20231001020000

Figura 41: Piloto 05a: Línea generación PV - Salida de datos dinámicos

La segunda muestra de las salidas de este piloto (Figura 42) se corresponde con los datos dinámicos de la línea de la sala 2. En ella se aprecian salidas de tres tipos de dispositivos diferentes. Uno de estos tipos mide consumos energéticos de los *fancoils* de la sala bajo medida,

así como un identificador de los mismos y el correspondiente parámetro temporal en el que fueron tomadas estas medidas. Otro tipo de sensor mide temperatura y humedad, mientras que el tercero genera mediciones de dióxido de carbono, manteniendo la columna de identificador de medidor y de temporalidad.

assetName	temperature	humidity	co2	V0	V1	V2	V3	V4	V5	V6	V7	calendar_id
ADAM5				1689	1645	1525	3981	2700	64592	1055	32775	20231122094200
ADAM5				1697	1645	1525	3981	2645	64576	1055	32775	20231122094216
ADAM5				1697	1645	1525	3981	2635	64570	1055	32775	20231122094230
ADAM5				1689	1645	1525	3973	2670	64581	1055	32775	20231122094240
ADAM5				1697	1645	1525	3973	2685	64587	1050	32774	20231122094310
assetName	temperature	humidity	co2	V0	V1	V2	V3	V4	V5	V6	V7	calendar_id
TUR3	213.0	542.0										20231122094152
TUR10	218.0	553.0										20231122094152
TUR13	220.0	548.0										20231122094152
TUR4	228.0	527.0										20231122094153
CO6			690.8125									20231122094154

Figura 42: Piloto 05a: Línea Sala 2 - Salida de datos dinámicos

4.7. ETL del piloto 06 - HERON

El piloto HERON es liderado por la empresa HERON, dedicada a la producción, suministro y comercialización de electricidad. El demostrador dentro del proyecto consiste en múltiples apartamentos que forman parte del mismo edificio, incluyendo un punto de carga individual.

El principal objetivo del piloto es promover edificios libres de carbono y gestionar la flexibilidad de la demanda agregada de múltiples viviendas. La armonización de datos y la interoperabilidad son aspectos clave a considerar debido a la heterogeneidad de los datos que se generan, incluyendo medidores inteligentes de electricidad del edificio, datos de sensores, carga de vehículos eléctricos y activos de generación renovable.

4.7.1. Dimensiones del *big data*

Tras analizar la información sobre HERON, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: En HERON podemos generar análisis de patrones de consumo energético de los convivientes del edificio, algoritmos de balanceo de carga relativo a los consumos o respuesta flexible de la demanda, así como modelos de predicción de esta demanda a partir de los datos que obtenemos del propio piloto y que nos ayudarían a alcanzar los objetivos propuestos.
- **Variedad**: El piloto 06 está conformado por las diferentes fuentes de datos generadas en un único edificio. Este piloto cuenta con 7 diferentes conjuntos de datos diferentes, los cuales cubren datos de enchufes y medidores inteligentes, generación de energías renovables diversas, mediciones de CO_2 y vehículo eléctrico.

  En este caso de uso sólo se extraen datos de una única interfaz. Además, la generación de datos en casa punto de este piloto es del orden de minutos. Por ambas razones, se ha establecido que la extracción de datos se lleve a cabo automáticamente cada diez minutos.
- **Volumen**: En HERON se llegan a producir anualmente un volumen de datos entorno a los 2,5 GB. Estos datos se encuentran originalmente en diferentes formatos de representación (Tabla 1).
- **Velocidad**: Las frecuencias de muestreo de los diferentes *datasets* varían desde los 30 segundos hasta cada varias horas. En nuestro caso, implementaremos la recolección de los datos relativos a la potencia y energía eléctrica en enchufes inteligentes, lo que limita esta variedad de frecuencias a únicamente los 30 segundos.
- **Veracidad**: Como se ha mencionado, en el piloto 06 contamos solamente con una API para la adquisición de los datos desarrollada por el propio piloto. La extracción de los datos requiere hacer una petición HTTPS, enviando los credenciales en el cuerpo de la misma, se obtiene un *token* para nuestra autenticación en futuras peticiones. Además, el mismo piloto es quien gestiona y almacena sus propios datos, así como los sensores y

dispositivos dedicados a la generación de los mismos. Por estos motivos, se considera que tanto su procedencia como su valor son confiables en mayor o menor medida.

4.7.2. Flujos de datos

Job

La estructura del *job* desarrollado (figura 43) consiste únicamente en la llamada a la ETL. Se pasan como parámetros los credenciales para la identificación del usuario al extraer los datos de la API del piloto, así como la frecuencia de ejecución del *job*. Se define además la alarma vía *e-mail* en caso de que se produzca un error en la ejecución.

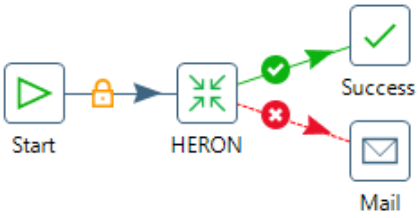

Figura 43: Job piloto 06

Transformación

La transformación asociada a este piloto se puede ver en la figura 44. En ella, se comienza extrayendo los argumentos pasados a través del *job*, como los credenciales o la periodicidad de ejecución del propio *job*. Después, se envía una petición a uno de los *endpoints* de la API autenticándonos y de esta forma obtenemos un *token* de sesión válido para la extracción de los datos. Una vez obtenido el *token*, lo extraemos del campo JSON correspondiente en la respuesta y generamos una nueva URL para solicitar un listado de dispositivos. Una vez obtenida la respuesta con el listado, el flujo se separa en metadatos y datos dinámicos.

En relación a los metadatos, se lleva a cabo una petición con un nuevo *URL* creado a partir d elos dispositivos obtenidos para preguntar a un nuevo *endpoint* por información sobre dichos dispositivos.

Los datos dinámicos son obtenidos en respuesta a una petición al *endpoint* pertinente y extraídos los campos y subcampos de interés de la estructura JSON. Cada una de las fases de los *smart meters* de las diferentes variables energéticas registradas hacen que sea necesario dividir el flujo. Posteriormente, en el flujo de energía de retorno se eliminan los valores nulos dado que no todos los dispositivos lo miden y en el subflujo de relés se transforman los valores de *Booleanos* "On" y "Off" a numéricos "1" y "0". Después, se pivotan los datos y se unen, generando de esta forma un único flujo con estructura en columnas. A continuación, se genera el parámetro

temporal *calendar_id*. Por último, se filtran los puntos relevantes cruzando el listado de variables definidas en el DWH y los obtenidos de la API y se exportan al mismo DWH, así como a los *topics digibuild.06.building* y *digibuild.06.ev* respectivamente del *broker Kafka*.

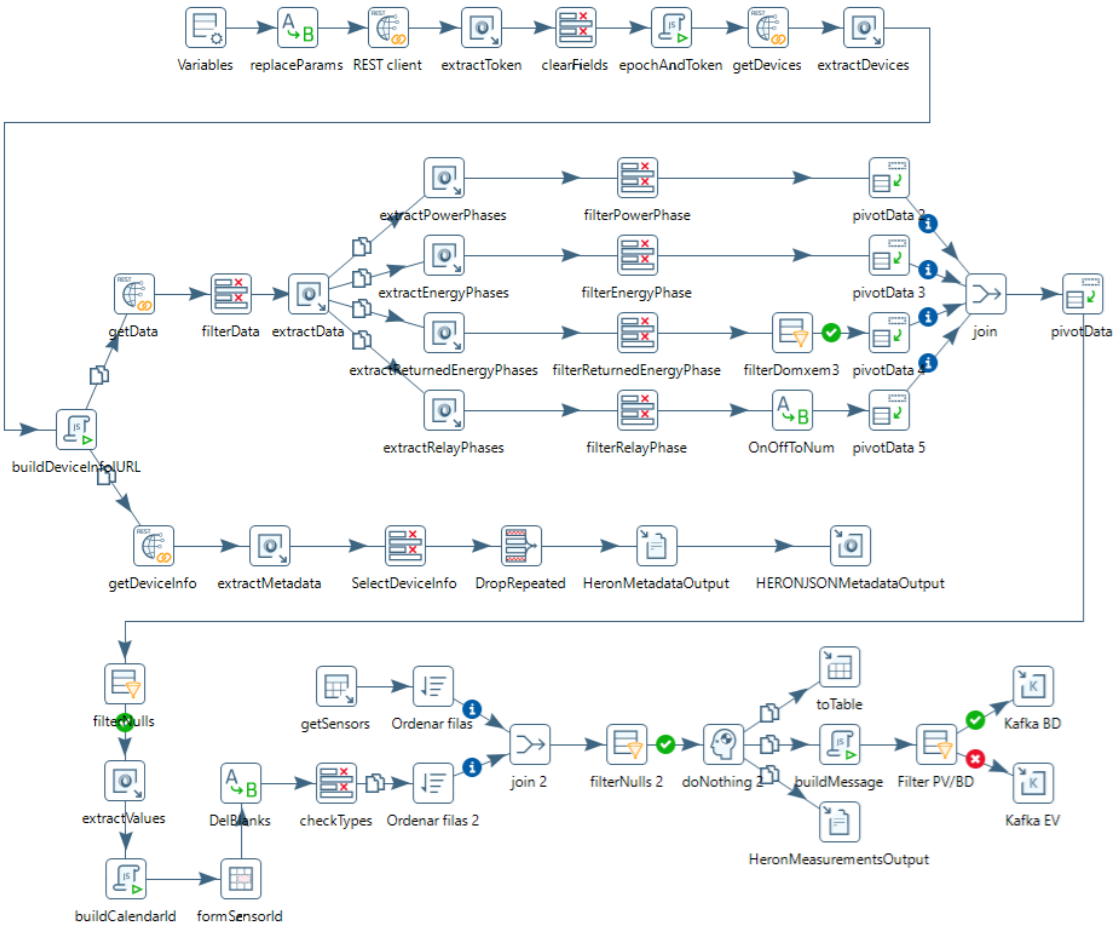

Figura 44: ETL piloto 06

Salidas

En este apartado procedemos a explicar las salidas de las diferentes líneas que conforman los flujos de metadatos y datos dinámicos del piloto de HERON.

Para la salida de metadatos (Figura 45) observamos una tabla con información sobre los dispositivos que miden las variables energéticas en el edificio de HERON. Estas variables son el tipo de dispositivo dentro del fabricante en forma numérica o de texto, el nombre de los mismos, un identificador del apartamento en el que se encuentran, su fecha de registro y las magnitudes que mide cada uno de ellos.

En contraparte, tenemos los datos dinámicos, cuya muestra podemos ver en la figura 46. En

device_id	device_type	device_type_text	device_name	home_id	registered_at	measurements
domxplug-C15499	4	Shelly Plug	286-A/C	286	2023-11-10T09:38:08.801Z	["energy","power","relay"]
domxplug-286D94	4	Shelly Plug	286-WM	286	2023-11-10T09:37:36.982Z	["energy","power","relay"]
domxem3-485519C920CD	2	3-phase EM	287-meter-EV	287	2023-11-10T09:37:04.165Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
domxem3-349454717657	2	3-phase EM	286-meter	286	2023-11-10T09:36:30.126Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
domxem3-349454718E54	2	3-phase EM	268-meter-EV	268	2023-10-09T07:43:49.362Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
shellyplug-s-7C87CEBA3E4C	5	Shelly Plug S	279-TV	279	2023-10-09T07:42:35.404Z	["energy","power","relay"]
domxplug-2C907E	4	Shelly Plug	279-A/C	279	2023-10-09T07:41:52.448Z	["energy","power","relay"]
domxplug-2CAB64	4	Shelly Plug	279-DW	279	2023-10-09T07:40:53.666Z	["energy","power","relay"]
domxem3-244CAB436349	2	3-phase EM	279-meter	279	2023-10-09T07:39:50.952Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
domxem3-3494546ED8F9	2	3-phase EM	283-meter-EV	283	2023-09-26T07:41:52.219Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
shellyplug-s-C8C9A3A5CA8F	5	Shelly Plug S	284-TV	284	2023-09-13T18:04:39.047Z	["energy","power","relay"]
domxplug-C19AEE	4	Shelly Plug	284-DW	284	2023-09-13T18:04:04.350Z	["energy","power","relay"]
domxem3-244CAB4356D1	2	3-phase EM	285-meter-EV	285	2023-09-13T18:03:05.346Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
domxem3-244CAB436229	2	3-phase EM	284-meter	284	2023-09-13T18:01:16.468Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
domxem3-244CAB435699	2	3-phase EM	273-meter-EV	273	2023-03-30T08:36:51.672Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]
shellyplug-s-C1AE2F	5	Shelly Plug S	272-TV	272	2023-03-30T08:30:28.184Z	["energy","power","relay"]
domxplug-286A3F	4	Shelly Plug	272-DW	272	2023-03-30T08:30:02.932Z	["energy","power","relay"]
domxem3-485519C9C37E	2	3-phase EM	272-meter	272	2023-03-29T09:56:08.527Z	["current","energy","pf","power","returned_energy","total","total_returned","voltage","relay"]

Figura 45: Piloto 06 - Salida de metadatos

ella podemos ver tres tipos de datos diferentes. El primero es el identificador de los puntos de medida, formado por la concatenación de las columnas de identificador de sensor, de la magnitud y de la fase que mide. El segundo es el valor de la medición, siempre en formato numérico aunque los valores del relé sean en texto. Por último, el parámetro temporal que asigna el instante en el que las medidas fueron tomadas por los dispositivos identificados.

calendar_id	device_id	measurement	phase	value
20231112230016	domxplug-C15499	energy	0	0
20231112230029	domxplug-C15499	relay	0	on
20231112230029	domxplug-C15499	power	0	0
20231112230029	domxplug-C15499	energy	0	0
20231112230059	domxplug-C15499	relay	0	on
20231112230059	domxplug-C15499	power	0	0
20231112230059	domxplug-C15499	energy	0	0
20231112230117	domxplug-C15499	energy	0	0
20231112230129	domxplug-C15499	relay	0	on
20231112230129	domxplug-C15499	power	0	0
20231112230129	domxplug-C15499	energy	0	0
20231112230159	domxplug-C15499	relay	0	on
20231112230159	domxplug-C15499	power	0	0
20231112230159	domxplug-C15499	energy	0	0

Figura 46: Piloto 06 - Salida de datos dinámicos

4.8. ETL del piloto 07 - FVH

El piloto FVH se centra en la gestión digital basada en datos de las instalaciones un edificio de reciente construcción de la ciudad de Helsinki. Conforme a la estrategia de la ciudad, Helsinki cuenta con un Sistema de Gestión de Energía de Edificios (BEMS), pero carece de interoperabilidad total. Por ello, DigiBUILD busca implementar técnicas de integración de datos, asegurando privacidad, seguridad e interoperabilidad semántica para la automatización de edificios.

Como objetivos fijados, este piloto pretende mejorar el perfilado de usuarios y el confort, así como entender la resiliencia frente al cambio climático.

4.8.1. Dimensiones del *big data*

Tras analizar la información sobre FVH, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: La integración del piloto en el proyecto permitirá una gestión unificada de los datos del mismo, así como del aprovechamiento de la existencia del BEMS como parte de la estrategia de digitalización de la ciudad de Helsinki.
- **Variedad**: El piloto cuenta con una gran cantidad de sensores (más de 5000). Todos ellos han sido recolectados desde el año 2022. Estos sensores forman parte de 5 *datasets*, los cuales contienen información sobre consumos energéticos, producción fotovoltaica y parámetros de confort.

  La única interfaz definida para este piloto se basa en *InfluxDBv2*. *Pentaho* no dispone de un *plug-in* en el *marketplace* para la versión 2 de *InfluxDB*. Por ello, se ha acordado con el piloto la publicación diaria de los valores almacenados en la base de datos en un servidor FTP. Además, los metadatos se almacenan en un fichero CSV adicional en el servidor. Por ello, pese a que la frecuencia de muestreo de las magnitudes es de 15 minutos, la extracción e ingesta de datos se lleva a cabo diariamente.
- **Volumen**: El demostrador de FVH genera anualmente un volumen de datos previsto en torno a los 3 GB.
- **Velocidad**: El periodo de generación de datos por parte de los medidores en las variables de interes para el proyecto dentro del piloto 07 es de 15 minutos.
- **Veracidad**: En este piloto, los datos generados por los sensores y medidores están a cargo de mantenimiento y extracción del piloto y su responsable técnico para le proyecto. En este piloto tienen un nivel de control de las telemetrías generadas más estricta que en los demás. Además, todas las extracciones de los datos las llevamos a cabo mediante protocolos estándares seguros. Por estas razones, se consideran fiables y veraces los datos extraídos.

4.8.2. Flujos de datos

Job

El *job* se ejecutará diariamente para descargar el fichero CSV comprimido del servidor FTP (tanto el de datos dinámicos como el de metadatos), se descomprimen los datos y se redistribuye el fichero para la siguiente ejecución. En el *job* no es necesaria la definición de ningún parámetro, dado que los credenciales van dentro del *script* de descarga y no se requiere la utilización de la frecuencia de ejecución del *job* (figura 47).

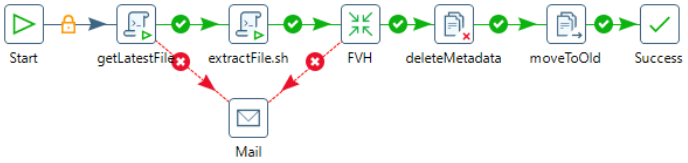

Figura 47: Job piloto 07

Transformación

La ETL de este piloto (Figura 48) consta de un único flujo. En este caso, los datos importados en el flujo están separados de los metadatos del piloto, descargados en un fichero aparte pero, al no ser relevantes, no se cargan en el flujo.

En el flujo de datos dinámicos se abre el fichero descargado del servidor FTP, se eliminan las columnas que no tienen información relevante y se calcula el *calendar_id*. Tras estos pasos se procede al envío de los datos dinámicos tratados al DWH y al *broker* de *Kafka* en los *topics* *digibuild.07.building* y *digibuild.07.airquality*.

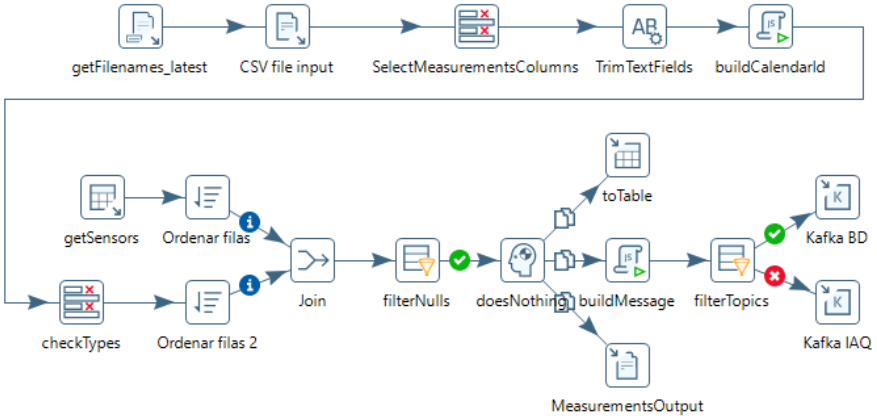

Figura 48: ETL piloto 07

Salidas

En este apartado procedemos a explicar las salidas de las diferentes líneas que conforman los flujos de metadatos y datos dinámicos del piloto 07.

La salida de metadatos en esta ETL está recogida en la figura 49. En ella se observan columnas sobre los elementos de medida del piloto, tanto sobre sensores como medidas. Estos metadatos comprenden el nombre y la descripción del sensor, la unidad de medida de la magnitud y el nombre de la misma, así como la localización en términos de piso, sección y habitación en el edificio. Por último, también existe un identificador del punto de medida.

name	description	unit	category	Floor	FloorSection	Room	datapointid
EdilD_2819184_P	SÃ¤hkÃ¶ 2819184	kWh	electricity	<null>	<null>	<null>	134625
29249_01_CUM	LÃ¤mpÃ¶ 29249	kWh	heating	<null>	<null>	<null>	134626

Figura 49: Piloto 07 - Salida de metadatos

Finalmente, se expone en la figura 50 la salida de datos dinámicos de la ETL del piloto FVH. En ella se aprecian las tres columnas necesarias marcadas por la estructura definida en la base de dadtos del DWH: el identificador del punto, el valor y la variable temporal *calendar_id*.

value	datapointid	calendar_id
157.7	134625	20220701000000
213.78	134625	20220701010000
254.81	134625	20220701020000
288.56	134625	20220701030000
311.41	134625	20220701040000
362.16	134625	20220701050000
363.2	134625	20220701060000
363.0	134625	20220701070000
355.1	134625	20220701080000
370.36	134625	20220701090000
366.73	134625	20220701100000
354.57	134625	20220701110000
352.57	134625	20220701120000
327.57	134625	20220701130000
313.8	134625	20220701140000
288.23	134625	20220701150000
283.84	134625	20220701160000
288.25	134625	20220701170000
258.76	134625	20220701180000

Figura 50: Piloto 07 - Salida de datos dinámicos

4.9. ETL del piloto 08 - IEECP

El piloto IEECP se lleva a cabo en un conjunto de edificios seleccionados estratégicamente en diferentes ubicaciones europeas. Estos edificios varían en tamaño, antigüedad y uso, lo que proporciona una amplia gama de contextos para la implementación y evaluación de las soluciones de gestión energética y de edificios.

El piloto 08 tiene como objetivo principal demostrar la viabilidad y los beneficios de la integración de soluciones de gestión energética y de edificios en diferentes contextos europeos. Este piloto se enfoca en la recopilación, análisis y utilización de datos energéticos y de edificios para mejorar la eficiencia energética, reducir el consumo de energía y las emisiones de CO2, y optimizar la operación y el mantenimiento de los sistemas de gestión de edificios.

4.9.1. Dimensiones del *big data*

Tras analizar la información sobre el piloto de IEECP, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: El valor que se obtiene de los datos tan variados extraidos del piloto varáin desde la previsión de consumos en edificios de oficinas mediante la generación de modelos de ML, el contraste de confort en los diferentes escenarios hasta el monitoreo continuo para desarrollo y la mejora de un modelo de negocio basado en una aplicación de pago para confort.
- **Variedad**: Este piloto está formado por diferentes edificios, los cuales cuentan con sensores de calidad de aire y medidores de energía. Las instalaciones que conforman este piloto cuentan con los siguientes dispositivos medidores: 30 sensores de calidad de aire interior, 30 dispositivos medidores de energía, 6 dispositivos medidores de energía centralizados y 10 locales.

  En IEECP se cuenta con 2 interfaces diferentes par ala extracción de los datos. Una de ellas se corresponde con los dispositivos de *Netatmo*, los cuales se emplean en la medición de calidad de aire interior. La otra interfaz de datos se implementa a través del protocolo MQTT y se recogen mediciones de energía.
- **Volumen**: IEECP ha estimado que su demostrador genera entorno a los 720 MB de datos anualmente en diferentes formatos de representación según su fuente de origen.
- **Velocidad**: La ingesta de datos en este piloto se ha establecido con un periodo de repetición de 15 minutos pese a que se obtengan entre cada minuto y cada 5. Esta decisión ha sido tomada en base a la frecuencia con la que se generan las medidas en ambos flujos que permita no ejecutar la ETL para un procesamiento de un lote de datos pequeño y menos óptimo.
- **Veracidad**: En el piloto 08, el responsable del mismo piloto es el encargado del mantenimiento y el control del acceso a los datos. Además, tiene la tarea del mantenimiento de los sensores y medidores que generan las telemetrías. Cabe mencionar que tanto en este como

en todos los pilotos, los responsables y responsables técnicos de los mismos han definido rangos de valores entre los que se espera que se reciban los datos generados.

4.9.2. Flujos de datos

Job

El *job* desarrollado para esta transformación (figura 51) sólo está formado por el bloque de ejecución de la propia ETL. Además, el único parámetro que se requiere es la frecuencia de ejecución del propio *job*, que ha sido ajustada a 15 minutos. Adicionalmente, se ha definido el flujo en caso de error, generando una alarma que se notifica por correo electrónico.

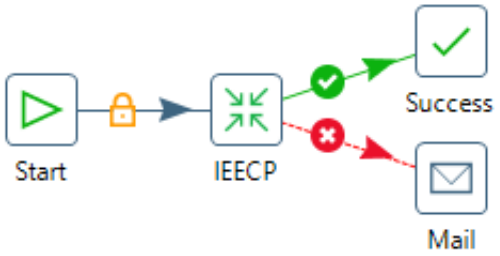

Figura 51: Job piloto 08

Transformación

En el flujo que se corresponde con la interfaz del protocolo MQTT (figura 52) se llevan a cabo los siguientes pasos. Tras la ingesta de los datos, asignamos el instante actual como variable temporal de la medida recibida del cliente MQTT y formamos la variable *calendar_id*. Después, se transforman los valores *Booleanos*(*On* y *Off*) en numéricos (1 y 0) para seguir la estructura de datos definida para nuestra base de datos. Por último, se cruzan las entradas de variables recibidas con la lista de puntos almacenada en la base de datos del DWH obteniendo sólo los relevantes y se exportan los datos formateados y limpios al *data warehouse* únicamente.

En la interfaz de *Netatmo* (Figura 53) se lleva a cabo en primera instancia la conformación de las URLs necesarias a partir de un fichero con los credenciales de autenticación provisto por parte del responsable técnico del piloto. A continuación, se procede a obtener un *token* de sesión gracias a uno de los *endpoints* de la API que emplea el mecanismo *Refresh Token*.

El mecanismo de *refresh token* o *token* de actualización es una técnica de seguridad utilizada en sistemas de autenticación para permitir a los usuarios mantener su sesión activa sin tener que volver a autenticarse constantemente. El proceso de autenticación en este mecanismo consiste en llevar a cabo una autenticación inicial por parte del usuario, donde el servidor verifica sus

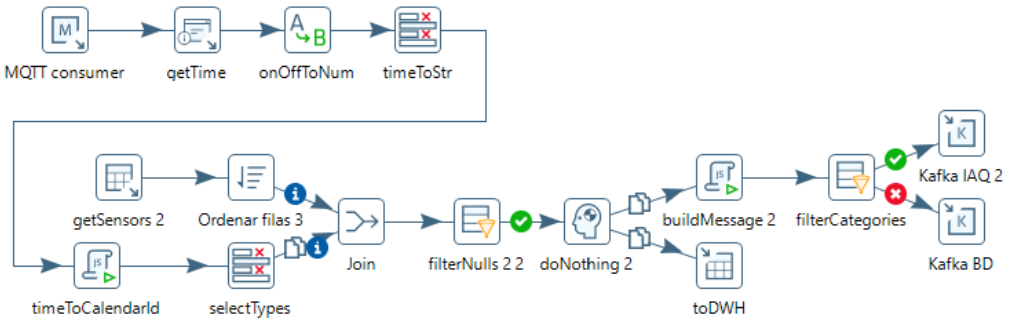

Figura 52: ETL piloto 08 - Línea MQTT

credenciales y genera tanto un *access token* como un *refresh token*. Cada vez que se quiere acceder a un recurso protegido dentro de la API, se manda en la cabecera el *access token* y puede hacer uso de él hasta que éste expira. Una vez el *access token* caduca, se emplea el *refresh token* para solicitar otro *access token* al servidor, evitando tener que autenticarse de nuevo con los credenciales. Este mecanismo mejora la seguridad de los datos en servidores dada la corta vida útil de los *access tokens*, así como un mayor control de la revocabilidad de los permisos y sesiones activas y, finalmente, para la experiencia del usuario, evitando tener que identificarse constantemente.

Tras la obtención del *token*, procedemos a la extracción del listado de estaciones disponibles en la API. Después, se extrae dicha lista del objeto JSON recibido y es a partir de aquí cuando se separan los metadatos del flujo. Respecto a los datos dinámicos, se extraen a partir de una nueva petición al *endpoint* indicado. Una vez con los datos en el flujo, separamos las columnas que contienen múltiples datos concatenados y juntamos las que necesitamos que formen un único tipo de dato. Después, se lleva a cabo un pivotaje para obtener la estructura por columnas que buscamos, se filtran las muestras vacías y se calcula el *calendar_id*. Finalmente, se filtran los puntos medidos mediante un cruce con la lista de variables almacenadas en la tabla del piloto del DWH, obteniendo así únicamente los relevantes para el proyecto. Esta exportación se lleva a cabo tanto hacia el DWH como al *broker* en el *topic digibuild.08.airquality*.

Salidas

En este apartado procedemos a explicar las salidas de las dos líneas que conforman los flujos de datos dinámicos del piloto 08.

Para la primera línea, la correspondiente con el cliente MQTT, podemos ver en la figura 54 el formato de los datos previo a la trasposición de las columnas que contienen las medidas. Cabe destacar que la situación final consistirá en que la columna de identificación de las variables sea la concatenación de la columna *device_id* y el nombre de la magnitud correspondiente además de una columna con el correspondiente valor medido. Por último y, como en todos los pilotos y flujos, la columna de la variable temporal *calendar_id*.

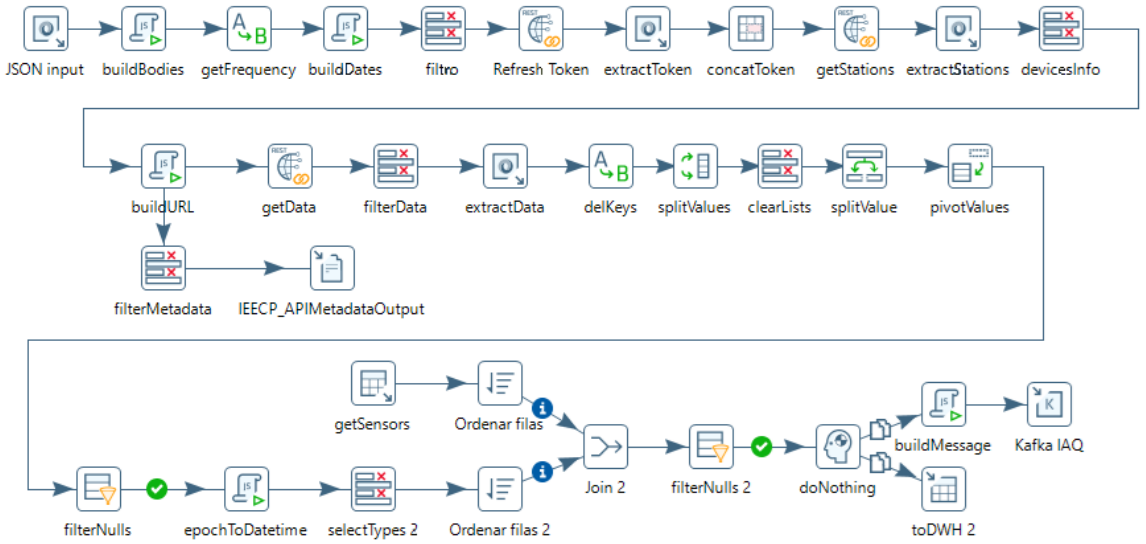

Figura 53: ETL piloto 08 - Línea Netatmo

device_id	calendar_id	temperature	min_temp	max_temp	humidity	min_hum	max_hum	co2	min_co2	max_co2	pressure	min_pressure	max_pressure
70:ee:50:83:38:42	20231024121500	26.6	26.5	26.7	55.0	55.0	56.0	1187.0	1173.0	1210.0	1020.1	1020.0	1020.0
70:ee:50:83:38:42	20231024124500	26.8	26.6	26.9	55.0	54.0	55.0	1238.0	1189.0	1286.0	1019.8	1019.7	1019.7
70:ee:50:83:38:42	20231024131500	27.1	27.0	27.1	54.0	54.0	54.0	1269.0	1241.0	1318.0	1019.6	1019.5	1019.5
70:ee:50:83:38:42	20231024134500	27.1	27.1	27.2	53.0	53.0	54.0	1250.0	1189.0	1318.0	1019.5	1019.4	1019.4
70:ee:50:83:38:42	20231024141500	27.2	27.1	27.2	53.0	53.0	53.0	1214.0	1198.0	1224.0	1019.2	1019.1	1019.1
70:ee:50:83:38:42	20231024144500	27.3	27.3	27.3	52.0	52.0	53.0	1246.0	1204.0	1280.0	1019.0	1018.8	1018.8
70:ee:50:83:38:42	20231024151500	26.4	25.7	27.1	51.0	50.0	52.0	1127.0	993.0	1235.0	1018.8	1018.8	1018.8
70:ee:50:83:38:42	20231024154500	25.2	24.7	25.6	49.0	48.0	49.0	952.0	904.0	992.0	1018.8	1018.7	1018.7
70:ee:50:83:38:42	20231024161500	24.7	24.4	25.1	48.0	48.0	48.0	840.0	804.0	870.0	1018.7	1018.6	1018.6
70:ee:50:83:38:42	20231024164500	25.8	25.4	26.0	49.0	48.0	49.0	826.0	818.0	835.0	1018.6	1018.5	1018.5
70:ee:50:83:38:42	20231024171500	26.1	26.1	26.2	50.0	49.0	50.0	809.0	768.0	875.0	1018.7	1018.6	1018.6
70:ee:50:83:38:42	20231024174500	26.0	26.0	26.1	50.0	50.0	51.0	732.0	703.0	754.0	1018.7	1018.6	1018.6
70:ee:50:83:38:42	20231024181500	25.9	25.9	25.9	51.0	51.0	51.0	700.0	686.0	715.0	1019.0	1019.0	1019.0
70:ee:50:83:38:42	20231024184500	25.8	25.8	25.9	52.0	51.0	52.0	664.0	648.0	689.0	1019.0	1019.0	1019.0
70:ee:50:83:38:42	20231024191500	25.7	25.7	25.8	52.0	52.0	53.0	622.0	604.0	639.0	1019.0	1018.9	1018.9
70:ee:50:83:38:42	20231024194500	25.7	25.7	25.7	53.0	53.0	53.0	606.0	581.0	626.0	1019.1	1019.0	1019.0
70:ee:50:83:38:42	20231024201500	25.7	25.6	25.7	53.0	53.0	53.0	583.0	574.0	590.0	1019.2	1019.1	1019.1
70:ee:50:83:38:42	20231024204500	25.6	25.6	25.6	53.0	53.0	53.0	558.0	539.0	581.0	1019.1	1019.0	1019.0

Figura 54: Piloto 08: Línea MQTT - Salida de datos dinámicos

En el caso de la línea de la interfaz *Netatmo* (Figura 55), se generará una columna que identifique el punto de medida con la concatenación de las columnas *Device, Measure* y *Number*, el valor de la medida se corresponderá con *Message* en formato numérico y, de nuevo, la temporalidad aportada por el *calendar_id*.

Message	Device	PL	Measure	Number
off	shellies	db-pilot8-ieecp-pl1	relay	0
off	shellies	db-pilot8-ieecp-pl2	relay	0
off	shellies	db-pilot8-ieecp-pl1	relay	0
off	shellies	db-pilot8-ieecp-pl2	relay	0
off	shellies	db-pilot8-ieecp-pl1	relay	0
off	shellies	db-pilot8-ieecp-pl2	relay	0
off	shellies	db-pilot8-ieecp-pl1	relay	0
off	shellies	db-pilot8-ieecp-pl2	relay	0
off	shellies	db-pilot8-ieecp-pl1	relay	0
off	shellies	db-pilot8-ieecp-pl2	relay	0
off	shellies	db-pilot8-ieecp-pl1	relay	0

Figura 55: Piloto 08: Línea Netatmo - Salida de datos dinámicos

4.10. ETL del piloto 09 - NTUA

Por último, el piloto NTUA está implementado en un conjunto de edificios universitarios dentro del campus de la Universidad Técnica Nacional de Atenas. Estos edificios son representativos de las instalaciones educativas típicas y proporcionan un entorno adecuado para probar y validar tecnologías avanzadas de gestión energética. Proporciona un entorno rico y diversificado para la evaluación de tecnologías avanzadas de gestión energética, abarcando una amplia gama de sistemas y dispositivos que trabajan juntos para optimizar el uso de energía y mejorar la sostenibilidad de las operaciones del campus universitario.

Entre los objetivos de este demostrador destacamos la implementar tecnologías avanzadas de gestión energética así como su monitoreo y análisis para reducir su consumo energéticos. Además, de la valoración de la viabilidad de inversiones y beneficios obtenidos de mejoras que supondrían inversiones en las soluciones de eficiencia energética.

4.10.1. Dimensiones del *big data*

Tras analizar la información sobre el piloto 09, podemos identificar en las cinco dimensiones del *big data* la siguiente información:

- **Valor**: Gracias a la explotación del monitoreo continuo de los datos, pueden llevarse a cabo controles de consumos más exhaustivos que permitan detectar fallas en el uso de las instalaciones o equipos, así como llevar a cabo análisis con gran detalle de las potenciales inversiones en equipos y su rentabilidad en el corto y largo plazo.
- **Variedad**: Este piloto está formado por un único edificio y cuenta con un numero bastante limitado de mediciones. Se recogen datos en diferentes 2 *datasets*: sobre consumos energéticos, siendo aire acondicionado, luz y ventilación, y sobre parámetros de confort, siendo temperatura y humedad relativa.
- **Volumen**: La estimación del volumen de datos generado en este demostrador asciende a los 1,5 GB anuales. En comparación con los demás demostradores podemos observar que es uno de los que más cantidad de datos genera.
- **Velocidad**: Los sensores y medidores operando en las instalaciones recogen datos en periodos que varían entre los 5 minutos y la media hora en función de la variable física que midan.

  Para la extracción de los datos de los diferentes *datasets* se ha migrado la base de datos de una *InfluxDB* a una *PostgreSQL* y se vuelcan los datos dos veces al día. Así, la extracción de los datos se lleva a cabo mediante una petición SQL a una gran cantidad de tablas y donde se especifica el periodo temporal de las mediciones que se desean recuperar. La generación de datos se lleva a cabo cada 10 minutos aproximadamente, pero el volcado en la base de datos a través de la cual tenemos accesibles los mismos se hace dos veces al día, limitando de esta forma el número de veces que hacemos las peticiones.

- **Veracidad**: La veracidad de los datos extraídos en este piloto está asegurada dado que en el proceso de petición y respuesta se emplea el protocolo *TLS*. Además, el acceso a la base de datos de *PostgreSQL* que consultamos requiere de una autenticación.

4.10.2. Flujos de datos

Job

El *job* correspondiente a esta transformación (figura 56) cuenta únicamente con el bloque de ejecución de la propia ETL. A esta ETL no se le pasa ningún parámetro, dado que la identificación se lleva a cabo en la definición de la conexión con la base de datos y le frecuencia de ejecución del *job* necesita ser un literal dentro de las peticiones SQL. En este *job* también se implementan las alarmas por correo electrónico en caso de error de ejecución en la transformación.

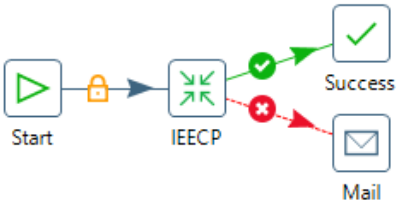

Figura 56: Job piloto 09

Transformación

En la transformación para el piloto 09 (figura 57) comenzamos con dos flujos que realizan peticiones SQL a la base de datos del piloto. Esto se debe a la existencia de diferentes columnas en cada caso, lo que evita que ambos flujos puedan ser unificados sin ningún formateo previo. En la primera, las peticiones se realizan a las diferentes tablas que contienen datos sobre consumos energéticos de aire acondicionado y de luz. En la segunda extraemos datos sobre magnitudes ambientales, como la temperatura y la humedad. En ambas líneas se lleva a cabo un pivotaje de los datos para poder extraer los metadatos. Con los dos flujos de los datos dinámicos, se unifican y se procede a realizar un pivotaje. Después, se unen dos de las columnas para formar el identificador de los sensores y se construye el *calendar_id*. Para finalizar, filtramos los puntos relevantes que almacenamos en la tabla del piloto dentro de nuestra base de datos del DWH y se exportan los datos al DWH. En este caso no se requiere de exportación a ningún *topic* del *broker*.

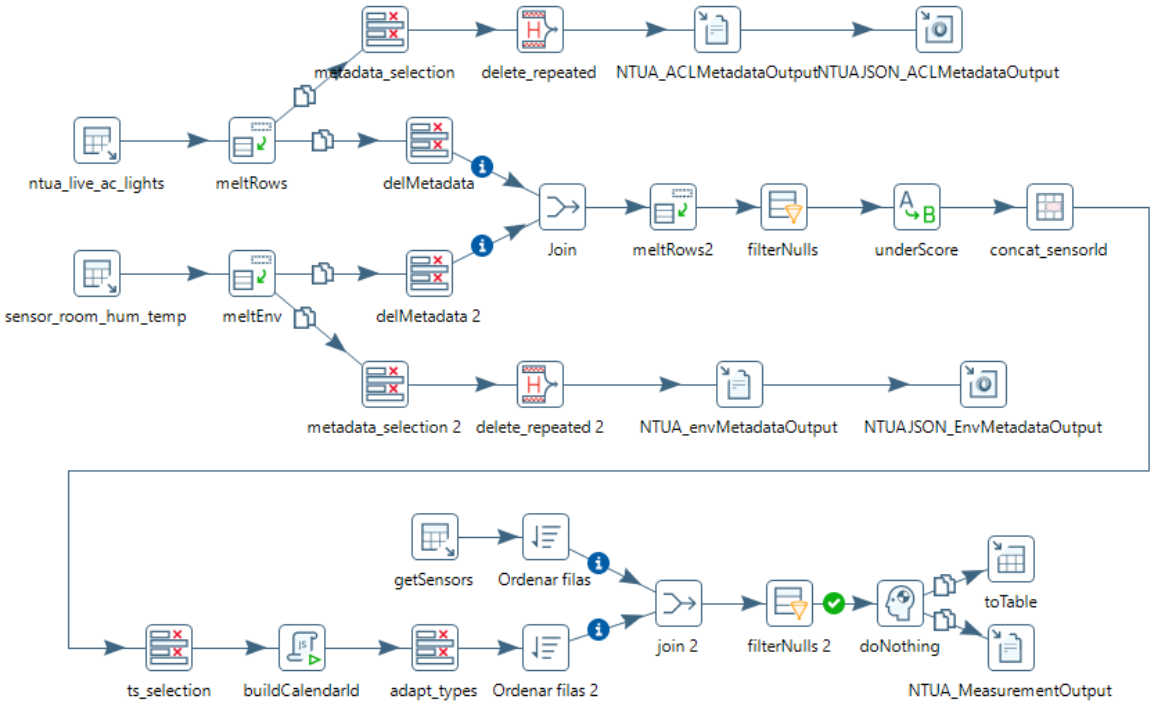

Figura 57: ETL piloto 09

Salidas

En este apartado procedemos a explicar las salidas de las tablas de las cuales extraemos los datos dinámicos del piloto de NTUA.

Comenzando por la figura 58, donde vemos los datos de la tabla de consumos de energía de los aires acondicionados y de la iluminación de diferentes salas de los edificios del demostrador. Podemos identificar las columnas de nombre de objeto y medida como las que se han usado de forma concatenada para identificar el punto de medida. Además, las columnas de valor de las medidas y de temporalidad como en los demás casos expuestos.

De forma equivalente, en la figura 59 observamos una muestra de datos de la tabla que almacena datos sobre magnitudes de calidad de aire. Las columnas que se observan son iguales que en el caso anterior y tienen el mismo tratamiento en la salida de la propia ETL.

object_name	measurement	value	calendar_id
AC DATA ROOM	kw_l1	0,3286	20231121000000
AC DATA ROOM	kwh_l1	31296,1	20231121000000
AC DATA ROOM	v_l1_n	237,7	20231121000000
AC DATA ROOM	a_l1_a	2,243	20231121000000
AC DATA ROOM	kva_l1	0,3315	20231121000000
AC DATA ROOM	kvarh_l1	2107,6	20231121000000
LIGHTS ROOM 24.25.26	kw_l1	0,0	20231121000000
LIGHTS ROOM 24.25.26	kwh_l1	7369,5	20231121000000
LIGHTS ROOM 24.25.26	v_l1_n	238,6	20231121000000
LIGHTS ROOM 24.25.26	a_l1_a	0,0	20231121000000
LIGHTS ROOM 24.25.26	kva_l1	0,0	20231121000000
LIGHTS ROOM 24.25.26	kvarh_l1	2345,4	20231121000000
AC ROOM 24	kw_l1	0,001	20231121000000
AC ROOM 24	kwh_l1	2853,1	20231121000000
AC ROOM 24	v_l1_n	237,4	20231121000000
AC ROOM 24	a_l1_a	0,075	20231121000000
AC ROOM 24	kva_l1	0,018	20231121000000
AC ROOM 24	kvarh_l1	1561,4	20231121000000
LIGHTS ROOM 24.25.26	kw_l1	0,0	20231121000000

Figura 58: Piloto 09: Tabla Aire Acondicionado e Iluminación - Salida de datos dinámicos

SENSOR ROOM 24	temperature	21,0	20231121000000
SENSOR ROOM 24	humidity	0,0	20231121000000
SENSOR ROOM 28	temperature	22,1	20231121000000
SENSOR ROOM 28	humidity	0,0	20231121000000
SENSOR ROOM 29	temperature	21,3	20231121000000
SENSOR ROOM 29	humidity	0,0	20231121000000

Figura 59: Piloto 09: Tabla Calidad Aire - Salida de datos dinámicos

4.11. ETL de la API meteorológica

Esta ETL emplea como entrada de datos una de las fuentes externas del proyecto. En este caso se utiliza la API meteorológica de *WeatherBit*. El motivo de la elección de este servicio es por la disponibilidad de una licencia por parte de nuestra organización.

Los pilotos cuyos servicios y/o gemelos digitales requieren de información meteorológica proporcionada por esta API son UCL (01), EDF (02), IASI&SITTA (03), VEOLIA FASA (04a), VEOLIA RVENA (04b), EMOT (05a) y HERON (06).

4.11.1. Dimensiones del *big data*

- **Valor**: El valor que se extrae en este caso es más directo y sencillo que en las interfaces de los pilotos. Esto se debe a que principalmente se usa para servicios de predicción de generación fotovoltaica o de consumos de calefacción o por los gemelos digitales de algunos de dichos pilotos.
- **Variedad**: Adicionalmente, una ventaja con respecto a otros servicios de predicción meteorológica es que esta API proporciona datos sobre una gran variedad de variables meteorológicas (más de 20), entre ellas, varios tipos de radiación solar, las cuales afectan directamente a los datos de los pilotos sobre la generación fotovoltaica y, por consiguiente, son datos muy relevantes para los servicios de predicción de este tipo de energía.

  De las diversas magnitudes que proporciona ese servicio, las más importantes son las que extraemos para el caso de aplicación, que son apenas 10. Estas variables son la temperatura, la presión, la velocidad y dirección del viento, la humedad, la cobertura del cielo y las radiaciones difusa, directa, global y estimada. Cada vez que se ejecuta la transformación se hace una llamada pidiendo las previsiones a 10 días con frecuencia horaria de los pilotos que requieren datos meteorológicos
- **Volumen**: Las características que nos concede el acceso a este servicio son 25.000 llamadas por día (25 llamadas por segundo), predicción meteorológica diaria con horizonte temporal de 16 días, predicción meteorológica horaria con horizonte temporal de 10 días y predicción meteorológica minutal con horizonte temporal de 1 hora.
- **Velocidad**: La API de la que extraemos datos en esta ETL proporciona, de la forma en la que la usamos en nuestro caso de uso, predicciones meteorologías con frecuencia horaria. Con motivo de no sobrecargar la máquina en la que corren los procesos que ejecutan las transformaciones, se a considerado que una frecuencia de ejecución diaria es suficiente para cubrir las necesidades de extracción de estos datos.
- **Veracidad**: La ejecución de las peticiones a la API a través del bloque que proporciona *Pentaho* nos asegura el uso del protocolo seguro *TLS* que, junto con la autenticación del usuario para el uso de la API podemos considerar una extracción de datos veraces.

4.11.2. Flujos de datos

Job

El *job* desarrollado para la transformación de *WeatherBit* (Figura 60) consta únicamente de la llamada a la ETL. No se requiere de ningún parámetro del que haga uso la transformación. La repetición de la ejecución del *job* está configurada para ser llevada a cabo diariamente.

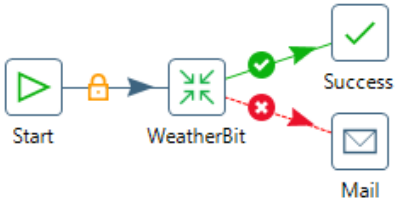

Figura 60: Job API meteorológica

Transformación

La transformación implementada (figura 61) cuenta con un único flujo de datos, aunque dentro del mismo se tratan simultáneamente los datos de las peticiones de los diferentes pilotos que las requieren. Específicamente, los pilotos que requieren de esta información meteorológica son los pilotos 01, 02, 03, 04 (ambos), 05a y 06, como se ha mencionado en el apartado anterior.

La extracción de los datos meteorológicos se lleva a cabo a través de peticiones HTTP al *endpoint* correspondiente de la API. Para estas peticiones se utiliza la localización geográfica de los diferentes pilotos interesados, devolviendo los datos en formato JSON.

Tras la obtención de los datos de predicción meteorológica en formato JSON de cada uno de los pilotos se formatean los datos extrayéndolos de subcampos del JSON original, haciendo un pivotajes y calculando el parámetro *calendar_id*. Por último, se exportan las predicciones obtenidas y formateadas al DWH. Además, se emplea el campo *city_name* para poder discernir a qué *topic* de *Kafka* publicarlo. Estos *topics* son los siguientes:

4.11.3. Salidas del flujo

En la figura 62 podemos ver la salida de la ETL. En la primera columna se especifica el nombre de la ciudad a la que se corresponden los datos de la predicción. EN la segunda columna se observa el mensaje que se va a publicar en *Kafka*, con los campos de magnitud, *calendar_id* (como en los pilotos) y el valor.

Además, estas predicciones obtenidas a través de la API se almacenarán en el DWH con el

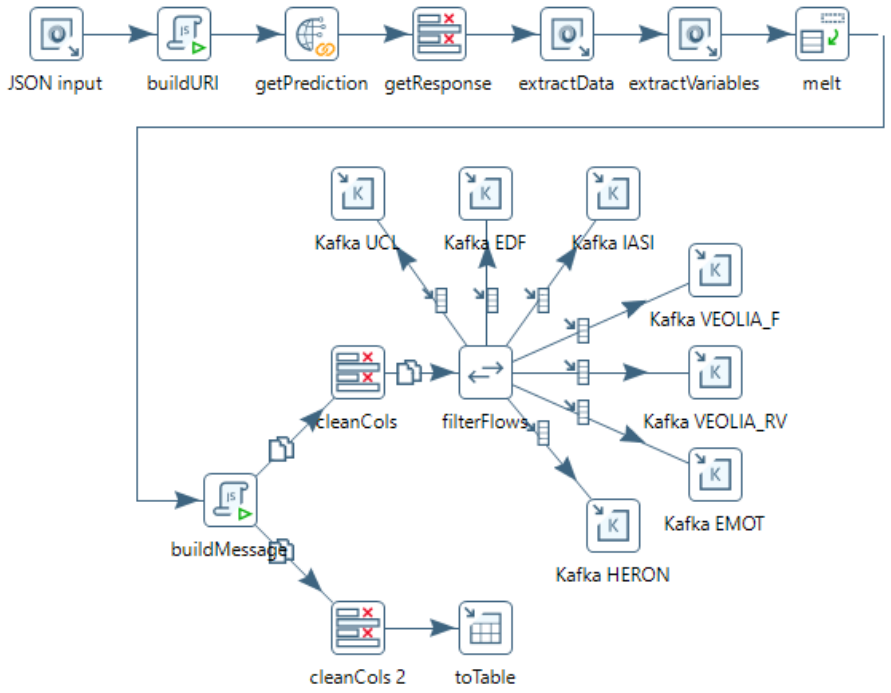

Figura 61: ETL API meteorológica

Piloto	***Topic***
01 - UCL	digibuild.01.weather
02 - EDF	digibuild.02.weather
03 - IASI&SITTA	digibuild.03.weather
04a - VEOLIA FASA	digibuild.04a.weather
04b - VEOLIA RVENA	digibuild.04b.weather
05a - EMOT	digibuild.05a.weather
06 - HERON	digibuild.06.weather

Tabla 3: *Topics* de predicción meteorológica

#	city_name	message	key
1	Blackheath	{"measurement_id":"Blackheath_temperature","calendar_id":"202403130800","value":10.7}	key
2	Blackheath	{"measurement_id":"Blackheath_pressure","calendar_id":"202403130800","value":1010.5}	key
3	Blackheath	{"measurement_id":"Blackheath_wind_spd","calendar_id":"202403130800","value":6.69}	key
4	Blackheath	{"measurement_id":"Blackheath_wind_dir","calendar_id":"202403130800","value":215}	key
5	Blackheath	{"measurement_id":"Blackheath_humidity","calendar_id":"202403130800","value":90}	key
6	Blackheath	{"measurement_id":"Blackheath_clouds","calendar_id":"202403130800","value":96}	key
7	Blackheath	{"measurement_id":"Blackheath_difusse_rad","calendar_id":"202403130800","value":66.33}	key
8	Blackheath	{"measurement_id":"Blackheath_direct_rad","calendar_id":"202403130800","value":589.81}	key
9	Blackheath	{"measurement_id":"Blackheath_global_rad","calendar_id":"202403130800","value":207.31}	key
10	Blackheath	{"measurement_id":"Blackheath_estimated_rad","calendar_id":"202403130800","value":95.58316}	key

Figura 62: Salida de Kafka API meteorológica

mismo formato que se almacenan los datos de las ETLs de los pilotos: [*calendar_id*, *sensor_id*, valor].

#	calendar_id	measurement_id	f_value
1	202403130800	temperature	10.7
2	202403130800	pressure	1010.5
3	202403130800	wind_spd	6.69
4	202403130800	wind_dir	215.0
5	202403130800	humidity	90.0
6	202403130800	clouds	96.0
7	202403130800	difusse_rad	66.33
8	202403130800	direct_rad	589.81
9	202403130800	global_rad	207.31
10	202403130800	estimated_rad	95.58316

Figura 63: Salida de DWH API meteorológica

Capítulo 5

Metodologías de Calidad de Datos

En este capítulo de la memoria se aborda el tema de la calidad de datos. En DigiBUILD es una parte importante tanto la evaluación como la mejora de la calidad de los datos que se extraen de los pilotos. La calidad de datos se define como las condiciones de los datos en base a diferentes atributos que pueden ser evaluaciones objetivas, si su evaluación no se basa en interpretaciones personales, o subjetivas, si por el contrario, sí que emplean juicios personales influenciados por el contexto del caso de aplicación. Además, las evaluaciones objetivas pueden ser dependientes o independientes de la tarea de la que se extraen los datos, según si su evaluación depende del contexto o el caso de uso en el que se empleen o no. Algunas métricas objetivas dependientes pueden ser normas o regulaciones de la empresa o del gobierno, restricciones del gestor de la base de datos, etc [41, 42].

5.1. Dimensiones bajo análisis

Existen multitud de métricas o dimensiones para de calidad de datos [41, 42], pero las seleccionadas en el proyecto [28] para la evaluación final de los *datasets* disponibles de cada piloto son las siguientes:

- **Precisión**: la medida en la que los datos reflejan un comportamiento veraz. Implica que los datos son consistentes con la realidad y están libres de errores. En definitiva, que los valores de las variables están dentro del rango esperado que se ha definido en el DWH.
- **Completitud**: la medida en que el volumen de datos no es falta y es de suficiente amplitud y profundidad para la tarea en cuestión. Esto quiere decir que los datos son continuos y no existen huecos temporales en los mismos.
- **Fiabilidad**: medida de completitud y precisión conjuntas.
- **Consistencia**: la medida en la que los datos son uniformes a lo largo del tiempo y en los puntos del sistema donde se almacenen.

- **Relevancia**: la medida en la que los datos son aplicables y útiles para la tarea en cuestión. Está relacionado con la correlación del contenido de los datos y las áreas de interés de los usuarios finales de los servicios y gemelos digitales del proyecto. En definitiva, es la implicación directa del inventariado de los datos y la selección de los *datasets* más relevantes de cada uno de los pilotos.
- **Accesibilidad**: la medida en la que los datos están disponibles o sean fácil y rápidamente accesibles. Esto implica que en los pilotos se implementen interfaces con protocolos de comunicaciones y de acceso libres o no existan problemas de accesibilidad a la información que requerimos.
- **Actualidad**: la medida en que los datos están suficientemente actualizados para la tarea en cuestión. Esto está directamente relacionado con la frecuencia de ejecución de los *jobs*. En algunos de los casos expuestos el cuello de botella se encuentra en el periodo de publicación de los datos en las interfaces definidas por los pilotos y no en la capacidad de computación en nuestro lado.

A continuación, podemos observar una tabla (Tabla 4) en la que clasificamos las dimensiones de calidad de datos que sometemos bajo estudio según si son objetivas o subjetivas y dependientes o independientes.

	Objetivo	***Subjetivo***
Independiente	Precisión Completitud	
Dependiente	Actualidad Consistencia Accsibilidad	Relevancia Fiabilidad

Tabla 4: Clasificación dimensiones calidad de datos

Se ha decidido implementar la evaluación de la calidad de datos sobre la extracción de datos que llevan a cabo las ETLs. De esta forma, se añade un flujo adicional a las figuras relativas a las ETLs de los pilotos explicadas en el capítulo 4 de la memoria para implementar el cálculo de diferentes parámetros de calidad de datos para obtener métricas que permitan identificar si hay algún problema con la ingesta de estos datos dinámicos. En este caso no estamos evaluando *datasets* completos, sino los datos obtenidos en cada ejecución de las transformaciones, por ello, tanto la selección de dimensiones como su planteamiento es diferente a los descritos en el párrafo anterior y deben adaptarse a la evaluación de series temporales [28].

La implementación se ha llevado a cabo dentro de las ETLs se puede observar en la figura 65. Este flujo se encuentra embebido en todas las ETLs correspondientes a la ingestión de datos de los pilotos, en el punto donde los datos tienen el formato *sensor_id*, *calendar_id*, *f_value*.

Las métricas que aplican en el caso de la ingestión de datos en series temporales y su definición son las siguientes [43]:

- **Completitud**: Mide la proporción de los sensores de los que se están recibiendo datos con

respecto al total de los sensores registrados en el DWH y se calcula de la siguiente forma.

$$\text{Completitud} = \frac{\#\text{variables}_{\text{adquiridas}}}{\#\text{variables}_{\text{registradas}}}$$

- **Precisión**: Mide qué proporción de valores recibidos de las fuentes de datos son precisos con respecto al total. Se consideran valores precisos a los que se encuentran en el rango de valores esperado para cada medida especificado en el DWH para cada uno de los sensores registrados. Su definición ha sido especificada por cada uno de los responsables técnicos de cada piloto. En caso de ser una medida incremental solo se comprueba el mínimo.

$$\text{Precisión} = \frac{\#[(\text{Valor} > \text{Mínimo}_{\text{variables}}) \ \ \& \ \ (\text{Valor} < \text{Máximo}_{\text{variables}})]}{\#\text{Valores}_{\text{adquiridos}}}$$

- **Consistencia**: Mide qué proporción de valores recibidos de las fuentes de datos son consistentes con respecto al total. Se consideran valores precisos a los que se encuentran en el rango comprendido entre la media menos la desviación típica (STD) y la media más la STD (figura 64) del conjunto de valores adquiridos por sensor.

$$\text{Consistencia} = \frac{\#[(\text{Valor} > (\text{Media} - \text{std})_{\text{adq/variable}}) \ \ \& \ \ (\text{Valor} < (\text{Media} + \text{std})_{\text{adq/variable}})]}{\#\text{Valores}_{\text{adquiridos}}}$$

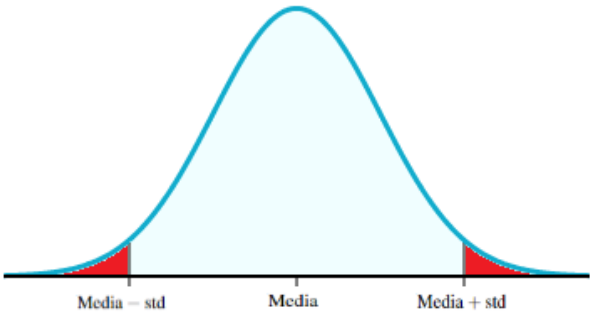

Figura 64: Consistencia en una distribución genérica

- **Fiabilidad**:

$$\text{Fiabilidad} = \text{Completitud} \cdot \text{Precisión}$$

En realidad, el punto de vista de la calidad de datos dentro de DigiBUILD implica dos objetivos: evaluar los flujos y corregirlos. En este TFM se desarrolla la parte de evaluación de los datos extraídos en una iteración de las transformaciones. La evaluación de los datos ya almacenados y su corrección está fuera de los alcances de este trabajo.

5.2. Implementación y resultados

A continuación procedemos a describir la implementación del flujo de evaluación de la calidad de datos en la extracción de datos dinámicos en las ETLs de los pilotos de forma

detallada. En la figura 65 podemos observar esta implementación. En el flujo de la parte superior se encuentran los datos que se han consumido en la ejecución de la ETL. En el primer paso los valores se ordenan por *sensor_id* y se calculan el número total de valores obtenidos y los sensores de los que se han extraído y, agrupando por sensor, la media y desviación típica de cada uno de ellos. En el flujo inferior obtenemos el listado de sensores definidos en el DWH y calculamos cuántos son. A continuación se unen ambos flujos empleando como clave común el *sensor_id*. Después asignamos a cada medida un indicador sobre si son precisos (se encuentran dentro del rango de valores definido en el DWH) y/o consistentes (se encuentran dentro del rango de valores de media±STD de los datos extraídos) en función del sensor que las haya generado y se cuentan los casos positivos. En este momento tenemos todas la componentes para calcular las métricas de calidad de datos, así que se procede a su computación.

Figura 65: Implementación de Data Quality en ELTs

El desarrollo de este flujo se implementa en todas las ETLs. La no inclusión del mismo en las figuras de los procesos de extracción se debe al deseo de no ensuciar en cierta manera la visibilidad de lo que se exponía en las secciones correspondientes.

En el estado actual del proyecto, la aplicación de los cálculos definidos está pensada para conocer el estado, por el momento, inicial, de los datos que se extraen en cada iteración llevada a cabo y, en caso de que se detecte una degradación de cualquiera de ellos, aplicar, en el futuro, metodologías de relleno y corrección de errores mediante algoritmos de ML.

Capítulo 6

Base de datos para de datos contextuales y dinámicos: Data Warehouse y Knowledge Graph

Este capítulo está destinado a la explicación del diseño, sus motivos y las características del componente del *data-lake* destinado al almacenamiento de los datos dinámicos o de series temporales, en una base de datos del *Data Warehouse* (DWH). Como se ha explicado previamente, en el capítulo 3, sobre la arquitectura, el DWH empleará como entradas los datos previamente tratados de todos los procesos ETL (capítulo 4) con el fin de almacenarlos y hacerlos accesibles a las capas superiores del sistema. Estos datos se emplearán para tanto los servicios como los gemelos digitales.

Como se ha mencionado en capítulos anteriores de la memoria, de algunas ETLs se extraen metadatos o datos contextuales que revelan información del entorno de las medidas o sobre los propios elementos que las toman. Es por esto que se hará mención sobre su población en el DWH y su sincronización con los datos dinámicos. Cabe destacar que el trabajo de recolección de metadatos, no ha sido trabajo del autor de este TFM, si no de un miembro del consorcio de DigiBUILD.

Cabe aclarar que el alcance del proyecto no engloba la creación de dicho *Data Warehouse*, sino su utilización y la definición de una base de datos dentro del mismo para poder almacenar los datos provenientes de las ETLs. El trabajo de creación y mantenimiento del DWH es autoría del centro tecnológico CARTIF [44], donde se ha desarrollado este trabajo de fin de máster.

Debido a que DigiBUILD es un proyecto donde el *big data* es la piedra angular, contamos con volúmenes de datos muy altos. Esto hará que el almacenamiento de los mismos sea un pilar importante a la hora del diseño del DWH. Con este tamaño de almacenamiento tan grande que requiere el sistema, el tiempo de respuesta del sistema de gestión de bases de datos es también un punto crítico. Teniendo en cuenta los puntos más sensibles a tener en cuenta para el diseño de la estructura de las tablas de la base de datos, comenzaremos con la explicación de las acciones llevadas a cabo.

6.1. DWH: Bases de datos relacionales y no relacionales

Las bases de datos pueden clasificarse, principalmente en dos grandes tipos: las bases de datos relacionales (SQL) o las no relacionales (No-SQL).

Las bases de datos relacionales almacenan los datos en tablas, estas tablas comparten la información, creando de esta forma relaciones entre ellas. Cada una de estas tablas tiene una columna que almacena un identificador único para cada entrada de la tabla, conocido como clave primaria, que puede ser referenciada en las otras tablas que conformen el esquema de la base de datos para poder dar lugar a las relaciones mencionadas. El uso de esta clave primaria en otra tabla la convierte en clave foránea de la misma. Las ventajas de este tipo de bases de datos son las siguientes:

- Cumple con el estándar *ACID* [11]: Esto asegura la fiabilidad de las transacciones en entornos de procesamiento de datos. *ACID* es un acrónimo que se refiere a Atomicidad (cada transacción se ejecuta por completo o no se ejecuta en absoluto), Coherencia (cada transacción lleva la base de datos de un estado válido a otro también válido), Aislamiento (las transacciones se ejecutan como si fueran la única en el sistema, aun cuando se ejecutan concurrentemente), y Durabilidad (los resultados de una transacción se mantienen incluso en caso de fallos del sistema). Este conjunto de propiedades garantiza que las bases de datos mantengan la integridad de los datos y soporten transacciones seguras y estables.
- Precisión de los datos: Esto pretende evitar la duplicidad de los datos haciendo que no se almacene información de forma redundante.
- Normalización: Ayuda a que los datos se almacenen de forma organizada intentando que no se generen anomalías o intentando evitarlas en la medida de lo posible.
- Facilidad en su implementación: Dado que existen multitud de herramientas y el lenguaje SQL están tan extendidos, se encuentran en un estado de optimización elevado en los gestores de bases de datos.

Las bases de datos no relacionales, por el contrario, no cuentan con una estructura rígida para le almacenamiento de los datos. Se diseñan teniendo en cuenta su flexibilidad y gran escalabilidad horizontal. Se pueden categorizar en diferentes tipos, cada uno de ellos con fortalezas y debilidades específicas. Las ventajas de este tipo de bases de datos complementan las de las relacionales, siendo principalmente la flexibilidad y la escalabilidad.

Nuestros datos han sido tratados y procesados en las ETLs que se han definido, por lo que ha cuentan con una estructura estándar. Además, al manejar tales volúmenes de datos (debido a que trabajamos con una gran cantidad de pilotos y multitud de variables y fuentes de datos), se requiere reducir al máximo posible la redundancia de información y, en definitiva su espacio en disco. Es por esto que se ha decidido trabajar con bases de datos relacionales. En concreto, trabajamos con bases de datos *PostgreSQL* [36].

En la fase de desarrollo del proyecto, previa a la fase de producción, se ha decidido definir un esquema para cada uno de los pilotos de forma independiente, aunque alojados en la misma

máquina. Además, el tipo de esquema elegido para la estructura de la base de datos ha sido el esquema en estrella.

6.2. DWH: Esquema en estrella

El esquema en estrella es una de las configuraciones más empleadas en las arquitecturas de bases de datos en multitud de sistemas en multitud de campos diferentes. Esta configuración esta formada por dos tipos de tablas:

- **Tabla de hechos:** Es el núcleo de la estructura de tablas y almacena las claves foráneas de las demás tablas, además de otras variables cuantitativas. Es conveniente este tipo de topología ya que permite la aplicación de diferentes cálculos a estas variables cuantitativas de una forma directa y más sencilla. Esta tabla de hechos almacena instancias de las observaciones que, en nuestro caso, sería cada medición tomada por uno de los sensores en un instante temporal concreto.
- **Tablas de dimensiones:** Cada una de estas tablas almacena información de los campos (columnas) de la tabla de hechos, donde se referencia su clave primaria. Su tamaño es mucho más reducido que el de la tabla de hechos.

Esta topología permite generar estructuras de datos y peticiones más simples, además de ser mucho más fácil de comprender por usuarios ajenos. Como se ha mencionado, facilita operaciones sobre los datos, como la agregación, y además, el mantenimiento.

En la figura 66 podemos observar un ejemplo de esta topología de base de datos donde, en el centro se encuentra la tabla de hechos y las tablas de dimensiones están conectadas con ella.

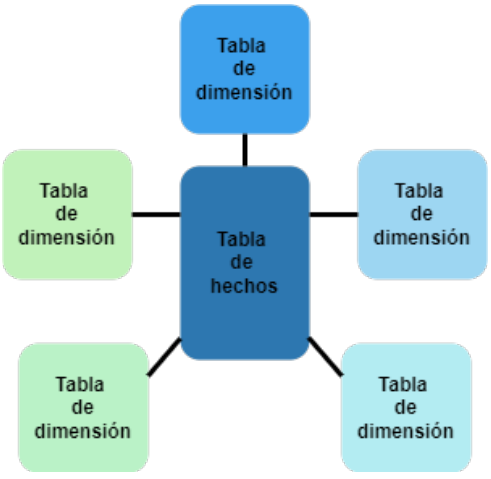

Figura 66: Esquema en estrella

La aplicación de esta estructura de base de datos en nuestra base de datos ha sido respaldada por las recomendaciones de desarrolladores de *software* para gestión de *big data* como puede ser

el caso de *Microsoft Power BI* [45]. Es sus guías de recomendaciones sobre la estructuración de los datos sugieren el empleo de esta topología o, en su defecto, la configuración en "copo de nieve", donde las dimensiones pueden estar formadas por varias tablas, no una única. Es una configuración similar a la explicada anteriormente, pero con una complejidad un poco superior.

Una característica que pretendemos explotar al máximo es la normalización, que se ha mencionado como ventaja de las bases de datos relacionales. Esta normalización consiste en la extracción de los datos al máximo posible de la tabla de hechos, dejando en cada una de sus entradas únicamente las claves foráneas de todas las dimensiones definidas y el menor número de columnas con datos en ella

6.3. Definición de la base de datos dinámica del Data Warehouse

Para concluir con este capítulo se procede a describir la estructura definida para los diferentes esquemas propuestos en nuestra base de datos de *PostgreSQL*.

Para la fase de desarrollo del proyecto se ha decidido implementar une esquema individual para cada uno de los pilotos que conforman el proyecto de DigiBUILD. Cada uno de estos esquemas estará alimentados directamente por la respectiva ETL como se ha descrito en capítulos anteriores. En total se definirán 10 esquemas diferentes.

En la figura 67 podemos ver cómo definimos todos los esquemas. La tabla de hechos, llamada *f_tsData* está formada por cuatro campos o columnas:

- ***ts_id*:** Es la clave primaria de la tabla de hechos, representa de forma única cada entrada de datos en la misma. Es un entero autoincremental.
- ***calendar_id*:** Es la clave foránea de la dimensión temporal. Representa el instante temporal en el que se ha tomado la medida por parte de un sensor concreto. Sigue la estructura explicada en la sección 4.1. Es un tipo de dato entero grande.
- ***sensor_id*:** Es la clave foránea de la dimensión sensor. Identifica únicamente al sensor que toma la medida registrada. Es una cadena de caracteres.
- ***value*:** Este campo de la tabla de hechos es el único que no es una clave foránea y representa el dato como tal. Es la cantidad medida de las deferentes magnitudes físicas que mide cada uno de los sensores de los pilotos.

Como podemos observar en dicha figura, también existen tres dimensiones de los datos. Estas dimensiones son: *d_calendar*, que identifica el momento de la toma de datos de cada entrada de la tabla de hechos; *d_sensor*, que identifica cada uno de los sensores de los que se recoge datos en cada uno de los pilotos; y *d_sensor_type*, que define qué tipo de sensor es cada sensor definido en la anterior tabla.

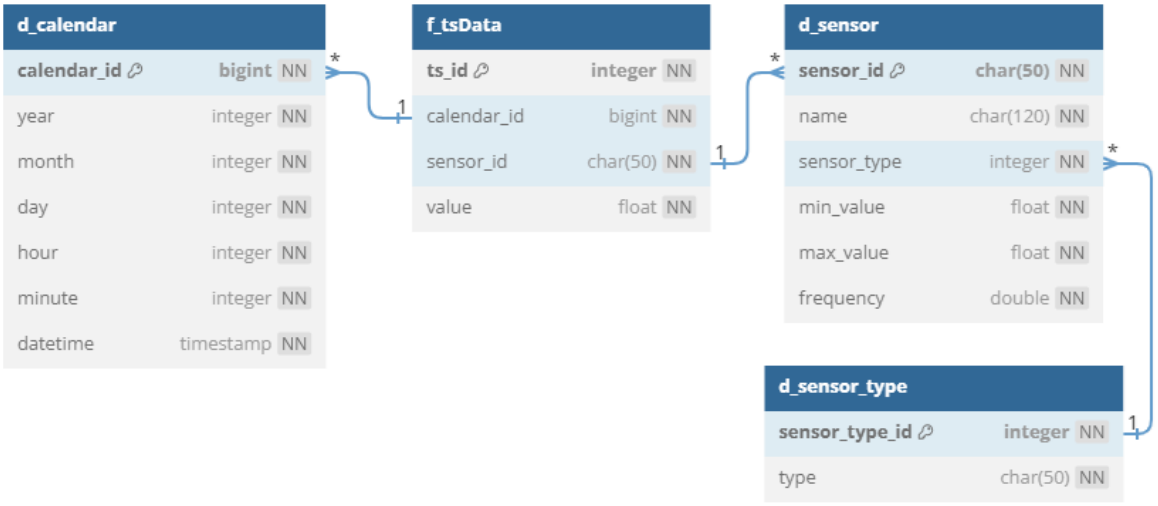

Figura 67: Diagrama de tablas del DWH

La tabla *d_calendar* está formada por las siguientes columnas:

- ***calendar_id*:** Es la clave primaria de esta tabla e identifica unívocamente cada instante temporal con una granularidad de minutos. Su formato está descrito en la sección 4.1. El formato de datos es un entero.
- ***year*:** Se corresponde con el año de la medida. Es un entero de cuatro cifras.
- ***month*:** Se corresponde con el mes de la medida. Es un entero de dos cifras que comprende del 01 al 12.
- ***day*:** Se corresponde con el día del mes de la medida. Es un entero de dos cifras que comprende del 01 al 31.
- ***hour*:** Se corresponde con la hora de la medida. Es un entero de dos dígitos que comprende el rango entre 00 y 23.
- ***minute*:** Se corresponde con los minutos en los que se tomó la medida. Es un entero de dos dígitos definido entre 00 y 59.
- ***timestamp*:** Se corresponde con la fecha y hora de la medida. Tiene el formato *timestamp* de *PostgreSQL*.

La tabla *d_sensor* está formada por las siguientes columnas:

- ***sensor_id*:** Se corresponde con la clave primaria de la tabla de sensores. Identifica únicamente a cada uno de los sensores de cada piloto. Es un tipo de dato de cadena de caracteres.
- ***name*:** Esta columna es el nombre del sensor. Es una cadena de caracteres.
- ***sensor_type*:** Este campo se corresponde con la clave foránea de la dimensión tipo de sensor. Identifica qué magnitud física o lógica mide el sensor.

- ***min_value*:** Se corresponde con el valor mínimo del rango medible esperado tiene cada sensor. Esta variable es un número decimal.
- ***max_value*:** Se corresponde con el valor máximo del rango medible esperado del sensor. En caso de ser magnitudes acumuladas su valor será *null*. Esta variables es de tipo decimal.
- ***frequency*:** Esta variable describe la frecuencia con la que toma muestras el sensor. Esta columna almacena datos de tipo doble.

Por último, la tabla *d_sensor_type* está formada por las columnas de *sensor_type_id*, que es la clave primaria de la tabla e identifica unívocamente cada uno de los tipos de sensor que hay en formato de número entero; y la variable *type*, que describe la magnitud medida y es de tipo cadena de caracteres.

Para disminuir el tiempo de respuesta por parte de *PostgreSQL*, es decir, reducción de la carga computacional, cuando se lleva a cabo un petición o *query* a los esquemas implementando herencia. En los casos de manejo de tablas de tamaño considerable los retardos pueden llegar a ser tan notorios que generen fallos y errores en la gestión de los datos. Esta herencia consiste en la asignación automática de columnas y valores a tablas "hijas" para poder modelar de una mejor manera las relaciones de diferenciación entre entidades. El concepto de herencia en *PostgreSQL* es bastante prominente y permite implementarlo. Tanto es así que existe una extensión de la base de datos llamada *TimeScale* [46] que permite llevar a cabo la herencia de forma mucho más sencilla con la implementación de la entidad "hipertabla". En el futuro se estudiará la migración del DWH a esta extensión mencionada para poder mejorar la cantidad de recursos empleada para atender a estas peticiones.

En este caso,la herencia se ha implementado en la tabla *d_calendar*, donde se han creado tablas "hijas" mensuales que han heredado las columnas de la tabla de calendario definida en 67. Gracias a esto, cuando se lleve a cabo una consulta, *PostgreSQL* podrá acceder de una manera más rápida y sin requerir de tanto tiempo de procesamiento.

6.4. DWH: Propuestas para la mejora de la gestión de grandes volúmenes de datos

Hasta ahora se ha mencionado alguna consideración que se ha tenido en cuenta para cumplir con la gestión de un gran volumen de datos, como la normalización, la herencia o la arquitectura en estrella o de copo de nieve. En esta sección se propondrán otras medidas que se tendrán en cuenta para mejorar el rendimiento de las consultas y mejorar el tiempo de respuesta del sistema.

Pre-filtrado y calidad de datos

La primera medida a tener en cuenta es la reducción del duplicado de información almacenada al máximo posible. Esto se está llevando a cabo mediante el pre-filtrado que se implementa en

las *ETL*s. Esto no evita que en dos ejecuciones diferentes que extraigan datos superpuestos a ejecuciones anteriores no almacenen información previamente guardada.

Una solución a esto podría ser hacer una búsqueda previa a la inserción de los datos ya existentes dentro de la misma ETL, lo que podría llevar a aumentar considerablemente la duración de la ejecución del propio proceso, o modificar la arquitectura actual del DWH haciendo, por ejemplo, que el identificador del sensor y el instante en el que se ha tomado la medida sean claves primarias. Existen muchas otras modificaciones de la arquitectura que podrían ser estudiadas e implementadas, pero la más inmediata sería la que se acaba de explicar.

Gestión automática de herencias

Actualmente, en la definición de las diferentes bases de datos de cada uno de los pilotos se están implementando herencias con respecto a la variable temporal de las telemetrías. Esta definición se las herencias se lleva a cabo de forma manual, separando las entradas de la tabla de hecho por meses.

Se han mencionado con anterioridad que estamos empleando una base de datos *PostgreSQL*. Pero una mejora en la gestión de un volumen elevado de datos sin necesidad de hacer un gran despliegue de medios en la migración sería el empleo de *TimescaleDB*.

TimescaleDB [46] es una extensión de *PostgreSQL* diseñada específicamente para la gestión del almacenamiento de series de datos temporales. Añade una capa de abstracción sobre las tablas normales de *PostgreSQL*, facilitando la gestión de grandes conjuntos de datos temporales mediante la automatización de muchas de las tareas asociadas con el diseño, la inserción y la consulta de estos datos.

Aunque se esté llevando a cabo la definición de las herencias de forma manual, esta extensión permite un proceso de definición y manejo mucho más simple y optimizado que lo implementado actualmente.

Indexación

La indexación permite mejorar el desempeño de la base de datos reduciendo el número de acciones que realiza, como por ejemplo, la cantidad de búsquedas en disco para completar una petición. En todo tipo de aplicaciones se busca la mayor inmediatez posible a la hora de satisfacer las peticiones obtenidas pero, en el caso de sistemas de *big data* es un punto mucho más crítico debido a la extensión que pueden alcanzar las tablas.

La creación de índices permite reducir el número de consultas y, por tanto, de tiempo necesario para satisfacer la petición. En las situaciones donde no existen los índices, la tabla al completo es analizada en búsqueda de entradas que satisfagan las consultas realizadas, mientras que si se ha definido previamente un índice, el número de filas que se consultan se reduce

considerablemente.

Los índices no requieren de ninguna actualización ni gestión posterior a su creación, el propio sistema de base de datos se encarga de actualizar automáticamente la información de las nuevas entradas [47].

Particionado de los datos y procesamiento distribuido y paralelo

La utilización de *frameworks* y aplicaciones que permiten la gestión de bases de datos distribuidas en diferentes máquinas. El empleo de estas utilidades puede ser tanto por aumento del tiempo de respuesta como por seguridad en el almacenamiento de los datos.

Existen *frameworks* altamente depurados, como *Apache Hadoop* [48], que permite la gestión y el procesamiento de grandes volúmenes de datos a través de diferentes *clusters*, así como un escalado horizontal automático para adaptarse de forma dinámica a las necesidades del sistema. *Hadoop* también tiene flexibilidad en cuanto al tipo de datos que almacena, permitiendo los estructurados, semi-estructurados y no estructurados.

Las características a destacar a mayores de lo que se ha mencionado son la capacidad del procesamiento de los datos en paralelo a través de los diferentes *clusters*, un negociador de recursos que permite asignar y despachar las tareas de una forma optimizada. Tiene resistencia a fallos gracias a copas de los datos que le permiten recuperación de la información entre los diferentes *clusters*.

Actualmente, su empleo está muy extendido en sistemas de compañías tan grandes como *Meta* o *Google*, por lo que podemos suponer que la aportación respecto a la gestión de *big data* es considerable y fácilmente implementable en DigiBUILD

Cubos OLAP Mondrian de Pentaho

Pentaho implementa una herramienta de código abierto conocida como *OLAP (Online Analytic Processing) Mondrian* [49]. Se emplea principalmente para la creación de entidades conocidas como cubos OLAP, que son estructuras de datos multidimensionales que permiten el almacenamiento de grandes volúmenes de datos y la aplicación eficiente de métricas agregadas.

Gracias a la definición de estos cubos se pueden llevar a cabo análisis interactivos complejos que pueden definirse de forma gráfica sobre los datos que contengan. Además, permiten llevar a cabo operaciones como filtrados, agrupaciones, *drill-down* y resúmenes con el fin de permitirnos obtener la información más relevante de los datos en cada caso.

Los cubos OLAP están diseñados para proporcionar un rendimiento óptimo en consultas analíticas complejas, incluso sobre grandes volúmenes de datos. Esto es idóneo para un sistema de almacenamiento que requiera de gestión *big data*. Esta optimización la consigue gracias a

pre-procesado de los datos, compresión de los datos y optimización automática de las consultas.

Finalmente, cabe mencionar que cuenta con la integración de herramientas de BI, como la creación de informes y *dashboards* que incluyen cálculos de métricas avanzadas.

6.5. Metadatos y su sincronización

En el contexto de DigiBUILD, los metadatos juegan un papel crucial en la gestión y utilización eficiente de los datos relacionados con la digitalización de edificios. Los metadatos proporcionan información esencial sobre los datos dinámicos recopilados, permitiendo una mejor comprensión, organización, seguimiento y búsqueda de los mismos. Estos incluyen detalles como la fuente de los datos, el tipo de datos, las condiciones bajo las cuales se recopilaron, y cualquier otra información relevante que describa el contexto y las características de los mismos. En proyectos **big data** como en el que se desarrolla este TFM, donde la integración y el análisis de datos de múltiples fuentes y formatos son fundamentales para lograr los objetivos del proyecto, los metadatos son la piedra angular.

6.5.1. Elementos del data lake

En el *data lake* que se genera en DigiBUILD se manejan cantidades de datos dinámicos muy altas, como se ha expuesto en el capítulo 4. Es por eso que la unión de estos datos dinámicos con los metadatos debe llevarse a cabo de forma cuidado. Para ello, en este proyecto se ha propuesto la definición de estructuras de datos comunes que permitan este unión.

El *data lake* definido previamente a este trabajo por componentes ajenos al TFM se compone de varios bloques principales, cada uno con funciones específicas. La arquitectura se divide en tres componentes de *Data Warehouse* y dos componentes de *Knowledge Graph*:

- **Servicios de Catálogo de Datos**: Es la interfaz del *data lake* con otras aplicaciones aguas arriba del sistema. Esta interfaz facilita el acceso y la gestión a los datos.
- **Base de Datos de Series Temporales**: En este componente se alojan los datos que se extraen de las ETLs y de bases de datos de históricos. En el podemos almacenar además datos en tiempo real y es uno de los componentes más críticos del *data lake*.
- **Almacén de *Data Marts***: Es un almacén que contiene estructuras conocidas como *Data Marts*, que son combinaciones de datos estáticos y dinámicos que funcionan a modo de informes o *reports* facilitando el análisis y los servicios de BI.

- ***Graph DB***: Almacena grafos semánticos en formato RDF o *Resource Description Framework* utilizando archivos de tríadas TTL o *Turtle*. Estos grafos nos permiten conectar los datos dinámicos del DWH con datos estáticos del almacenamiento de objetos usando tecnologías semánticas.

- ***Almacenamiento de Objetos***: Guarda información estática en forma de archivos de datos abiertos. Estos datos incluyen modelos de información de edificios, geometría y otros contextos estáticos relevantes para el análisis.

6.5.2. Mecanismos de sincronización

En el núcleo de las estructuras del *data lake*, los datos dinámicos se vinculan a los datos estáticos mediante grafos semánticos. Estos grafos se forman siguiendo una ontología desarrollada en el proyecto DigiBUILD, la cual reutiliza conceptos de las ontologías *REC* y *Brick*.

Las ontologías son marcos de trabajo que se emplean para definir estructuras de conocimiento que se utilizan para organizar y categorizar información, modelar relaciones conceptuales y mejorar la gestión del conocimiento.

La ontología definida para DigiBUILD consta de cinco módulos:

- **Módulo Central**: Contiene tipos de datos esenciales que conectan los tipos de datos de los otros módulos.
- **Módulo de Edificación**: Incluye datos espaciales sobre los pilotos, sus sistemas de distribución y los activos que abarcan, en definitiva, la información espacial.
- **Módulo Contextual**: Contiene tipos de datos relacionados con agentes, vehículos y recursos externos.
- **Módulo Cuantitativo**: Comprende tipos de datos cuantitativos categorizados en grupos de energía, operación, confort, economía, desempeño y medio ambiente.
- **Módulo Temporal**: Incluye tipos de metadatos para datos de series temporales y sus relaciones semánticas con tipos de datos del módulo cuantitativo.

Todos los enlaces semánticos se producen a través del módulo central y se ha establecido que cada fuente de datos dinámicos se representa mediante un objeto de clase “Punto” y se conecta con una “Referencia de Serie Temporal” a una entrada de la base de datos.

6.6. Interfaz para compartición de datos

Como parte de la hipótesis de este trabajo, se ha explicado cómo se plantea un paradigma diferente en el diseño del sistema de DigiBUILD, opuesto a los sistemas desarrollados con anterioridad bajo el nombre de enfoque de silo. Este nuevo paradigma permitiría que los edificios inteligentes puedan gestionar sus datos y compartirlos con las partes interesadas de forma integral en el sistema de una forma más accesible y controlada para el público objetivo en cada caso de uso.

Anteriormente, se ha explicado que los diferentes pilotos que forman parte del consorcio se engloban dentro del *big data*. Estos datos se extraen, formatean y almacenan para hacer uso de ellos. Una vez almacenados, cada piloto tiene asociados unos servicios, los cuales hacen uso de dichos datos para ofrecer un valor añadido según su objetivo. Es en este punto donde nos enfocaremos en esta sección. En la arquitectura (capítulo 3) las funcionalidades que satisface el desarrollo de esta sección están asociadas al bloque *APIs for Learning Services* de la capa *framework* de servicio.

Estas funcionalidades se construyen sobre los bloques funcionales del *data lake*, dado que hacen uso de los datos tanto estáticos como dinámicos que han sido previamente extraídos, tratados y almacenados, así como dotados de la sincronización necesaria para explotar la información que contienen. Estas capas superiores giran entorno a los gemelos digitales y los servicios de inteligencia artificial desarrollados.

El desarrollo llevado a cabo correspondiente a esta sección de la memoria se trata de una interfaz o API para hacer uso de los diferentes servicios de inteligencia artificial. El desarrollo de estos servicios y los modelos asociados no forman parte de este TFM.

Este acceso a los servicios se lleva a cabo mediante una API, como se ha mencionado. El desarrollo de esta interfaz se ha llevado a cabo en *Python*, utilizando la librería *FastAPI*. Esta interfaz se encuentra en fase de desarrollo todavía, por lo que los modelos y sus métricas se almacenan en local y las predicciones meteorológicas son muestras también en local. En el momento del despliegue de la interfaz, se almacenarán tanto los modelos como sus evaluaciones en una base de datos y las predicciones meteorológicas se obtendrán mediante una llamada a un servicio público externo, como el de *Open Meteo*, previamente mencionado.

A continuación, se procede a la explicación de las diferentes funcionalidades que permite esta interfaz.

6.6.1. *Endpoints*

En la API desarrollada podemos encontrar los siguientes *endpoints* o funciones que realiza la interfaz (Figura 68). En esta figura generada automáticamente por la librería de *Python* empleada, podemos ver en el título el nombre de dicha librería (*FastAPI*). A continuación, se observan dos categorías en las que hemos dividido los tres *endpoints*:

- ***Base Endpoints*:** Que se emplean para conocer información sobre los modelos disponibles.
- ***Model Inference*:** Que se emplea para poder utilizar los modelos de predicción energética.

Para poder interactuar con la API podemos hacerlo a través de peticiones HTTP con los parámetros que cada uno de los *endpoints* requiere o bien hacerlo a través de la documentación que genera *FastAPI*.

En las siguientes secciones se explicarán las funciones de cada uno de los *endpoints* indicando

sus entradas y salidas.

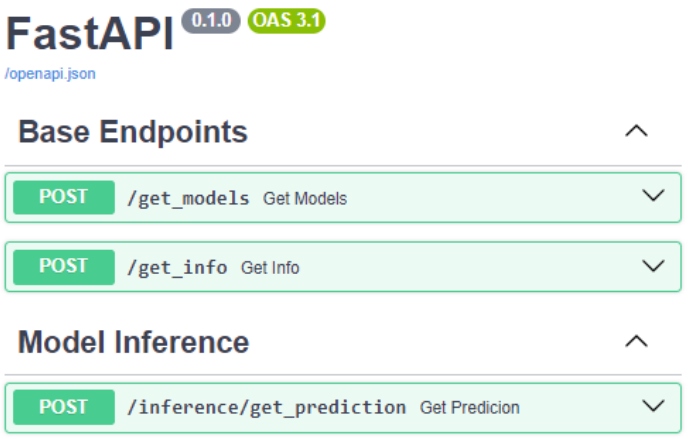

Figura 68: Endpoints definidos en la API

get_models

Este *endpoint* permite obtener información sobre los modelos de aprendizaje automático generados para el piloto que se desee. En la figura 69 podemos ver que el método HTTP que requiere es un *POST* y que tiene un único parámetro de entrada.

Este parámetro de entrada es el nombre del piloto como cadena de texto y es opcional. En caso de no conocerlo, o no introducir un nombre, ser devuelve una lista con todos los modelos que almacena la API. Podemos ejecutar una llamada a este *endpoint* directamente desde la documentación de la API. En caso de introducir un nombre incorrecto o no reconocido la API devolverá un error.

La respuesta de este *endpoint* es en formato JSON. En la figura 70 podemos apreciar los diferentes campos con información que obtenemos de cada uno de los modelos. Estos campos son:

- ***pilot_name*:** El nombre del piloto como cadena de caracteres.
- ***variable*:** El nombre de la magnitud que predice, como potencia o energía en formato cadena de caracteres.
- ***frequency*:** Frecuencia de los datos con los que ha sido entrenado el modelo, como puede ser horario o cada media hora. Este parámetro es una cadena de caracteres
- ***identifier*:** Es un identificador que ayuda a distinguir las diferentes instancias de los modelos dentro de un mismo piloto. Por ejemplo, los diferentes cargadores de vehículo eléctrico. Este parámetro también es una cadena de caracteres.
- ***algorithm*:** Es parámetro es el tipo de algoritmo que emplea el modelo al que identifica, como *decisiontree* o *randomforest* en nuestros modelos.

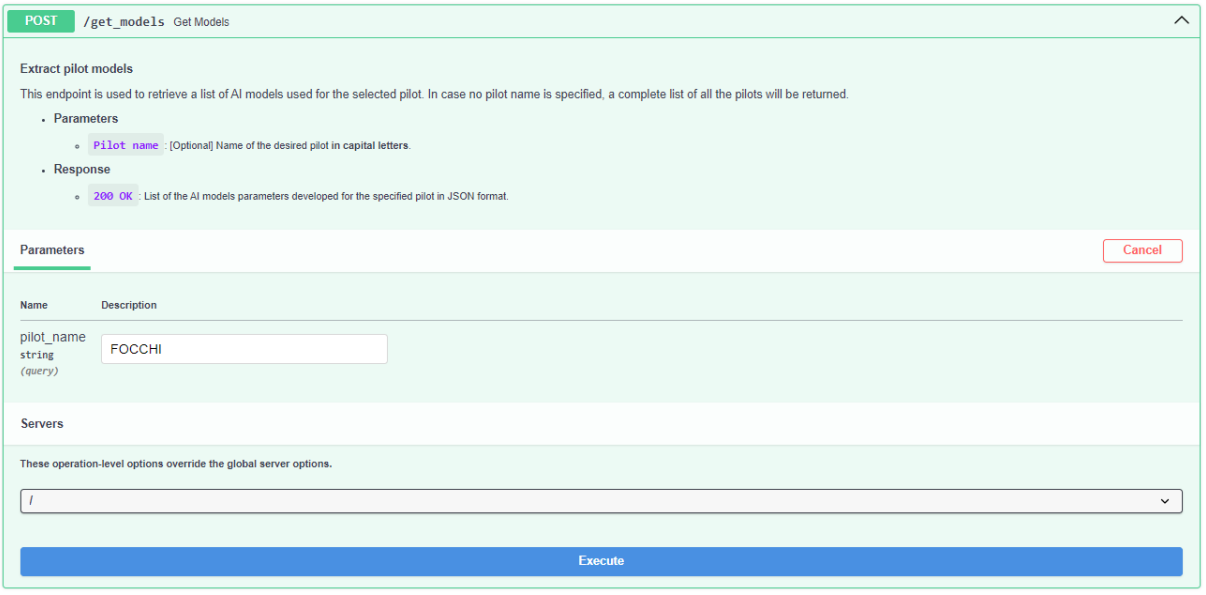

Figura 69: Endpoint description: get_models

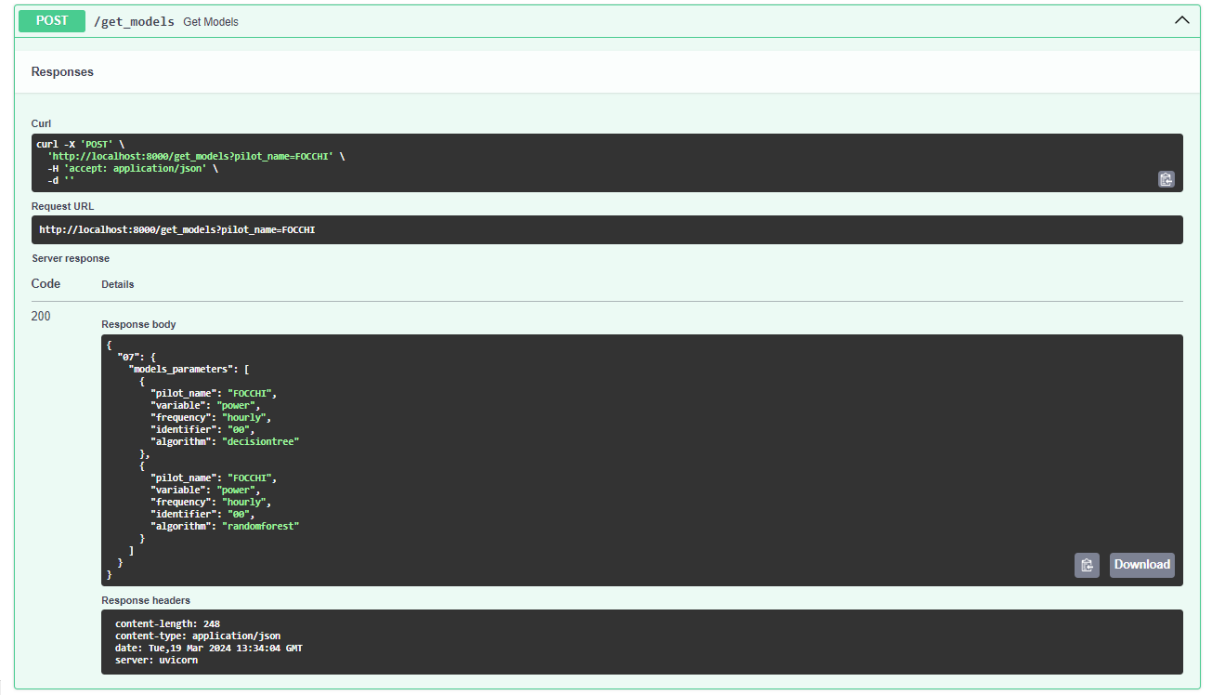

Figura 70: Endpoint response: get_models

get_info

En esta ocasión, este *endpoint* se utiliza para conocer información del modelo que deseemos. Esta información se refiere a los diferentes hiperparámetros con los que ha sido entrenado, el tiempo de entrenamiento y las métricas de evaluación del propio modelo. A partir de la información proporcionada por el anterior *endpoint*, podemos utilizar este.

En la figura 71 podemos ver una descripción de este servicio. Principalmente proporciona una lista de parámetros sobre el modelo especificado. En caso de que no se especifique alguno de los parámetros opcionales, se mostrará la información de todos los modelos referentes al piloto indicado en la petición. El nombre el piloto sí que es un campo obligatorio.

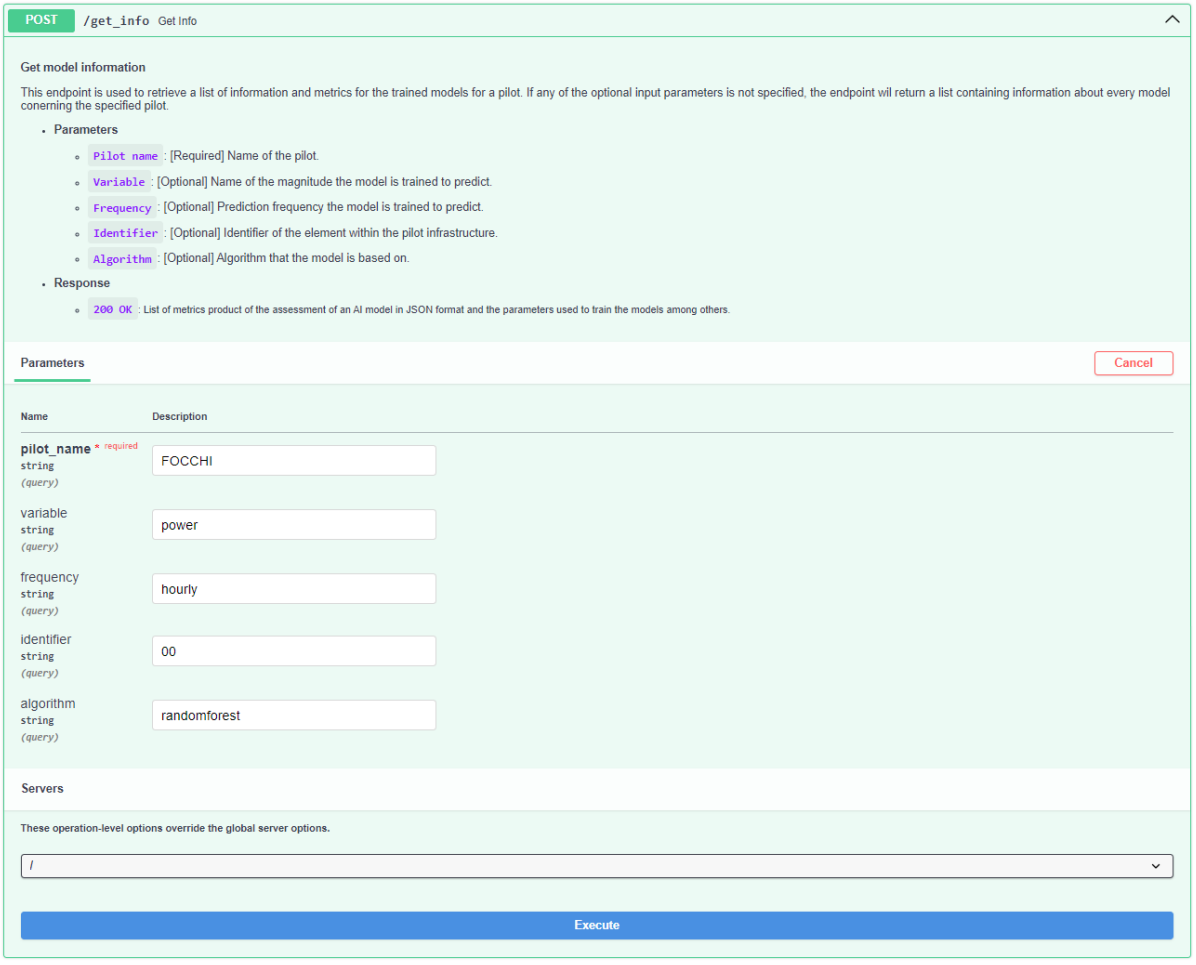

Figura 71: Endpoint description: get_info

La respuesta recbida en caso de hacer una petición a este *endpoint* será una lista en formato JSON con la siguiente información.

- **Parámetros anteriores:** Información similar extraída en el *endpoint* explicado anteriormente.
- ***test_size*:** Porcentaje de muestras empleadas para el test en el entrenamiento del modelo.
- ***n_estimators*:** Número de árboles de decisión empleados en el algoritmo *random forest*.
- ***criterion*:** Criterio de medición de la calidad de una división en cada uno de los nodos del árbol. Se emplea tanto en el algoritmo *decision tree* como en el *random forest*.
- ***splitter*:** Estrategia para la división en cada uno de los nodos del árbol en el algoritmo *decision tree*.

- ***training_time*:** Tiempo empleado en el entrenamiento del modelo final medido en segundos.
- **Métricas de evaluación:** Conjunto de las diferentes métricas de evaluación del modelo por el que se ha preguntado a la API

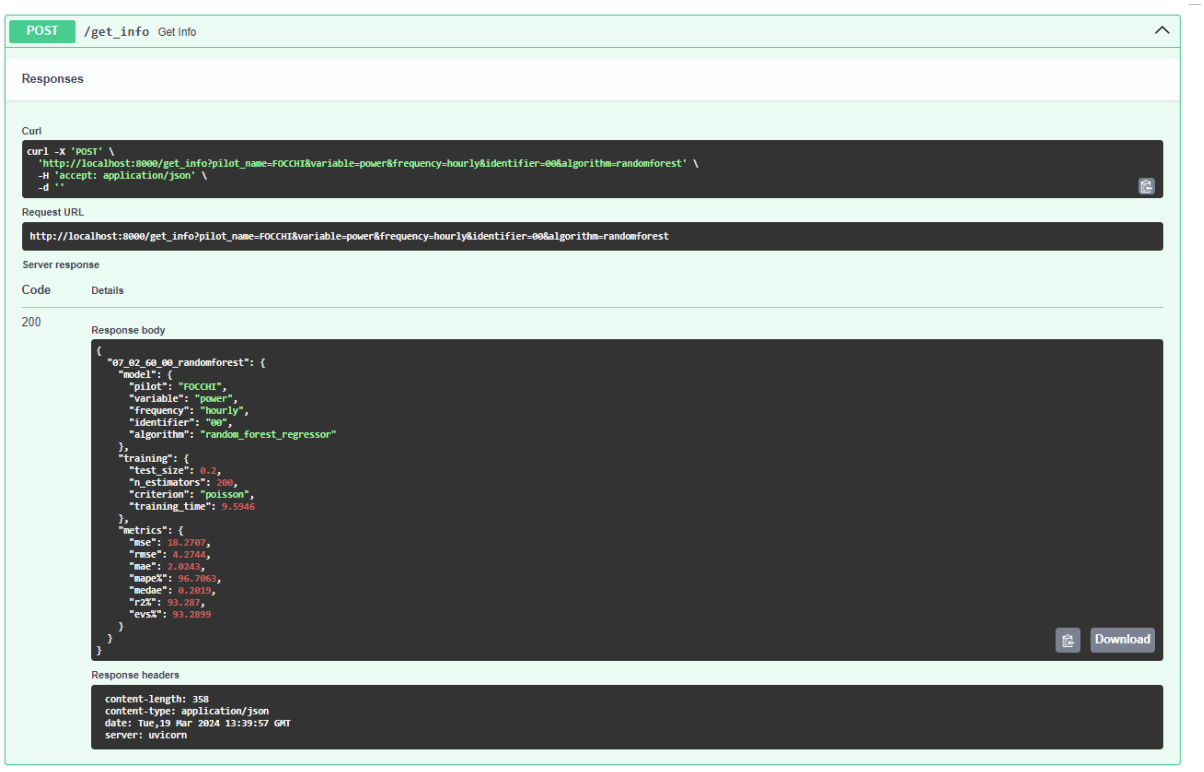

Figura 72: Endpoint response: get_info

get_prediction

Para finalizar, y el *endpoint* principal de la API, tenemos la función de predicción. Con esta llamada a la API, indicando los mismos parámetros que en anterior *endpoint*, podemos obtener una predicción de la variable basándose en una predicción meteorológica. El horizonte temporal de esta predicción se escala automáticamente en función de la granularidad o frecuencia de las muestras con las que se haya entrenado cada modelo. En la figura 73 podemos ver la descripción que proporciona la documentación de la API, así como los diferentes parámetros y su obligatoriedad.

Estas predicciones, una vez se despliegue la API se consultarán con una llamada a un servicio externo de meteorología, pero actualmente, por no saturar el servicio con las pruebas que estamos llevando a cabo, se ha empleado una predicción almacenada localmente.

Para el piloto 04b, tenemos muestras de datos cada media hora, mientras que para el piloto 05b, tenemos muestras de datos horarias. Es por esto, que la API es capaz de identificar en cada

modelo, la frecuencia de las muestras con las que se ha entrenado el algoritmo y adaptar el horizonte temporal de la predicción. de la siguiente forma: para datos horarios se predicen los siguientes 7 días, para datos cada media hora los siguientes 3 días y para datos quince-minutales únicamente el próximo día.

En caso de no especificar el algoritmo que deseamos emplear para llevar a cabo la inferencia, se utilizará aquel que tenga mejores métricas. Este criterio ha sido subjetivo, y sólo tiene en cuenta la media aritmética del MSE y del MAE.

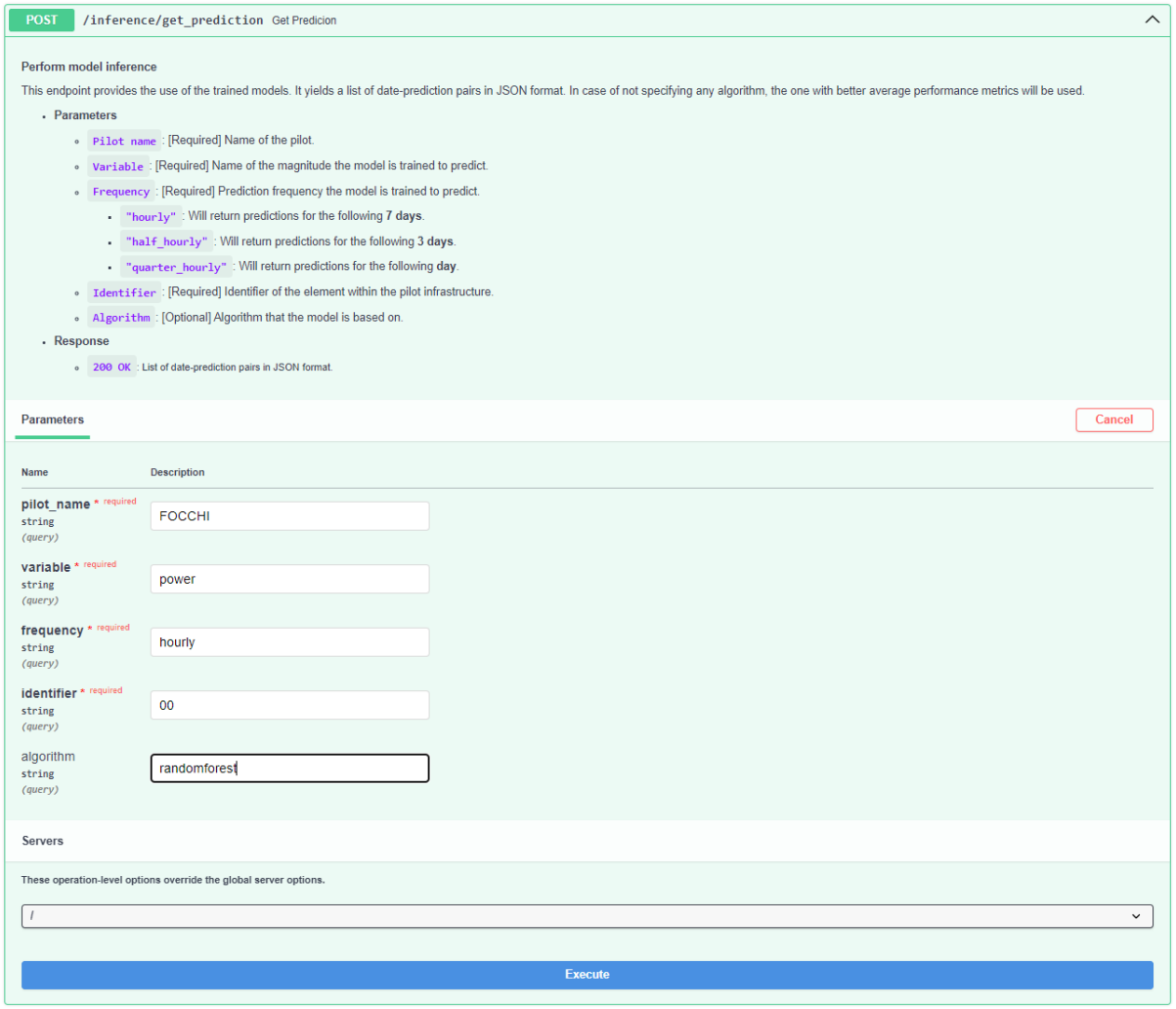

Figura 73: Endpoint description: get_prediction

La respuesta que obtenemos de este *endpoint* se puede apreciar en la figura 74. Podemos ver cómo la API devuelve un JSON con una lista llamada "*predictions*" con elementos formados por un campo de fecha en formato "AAAA-MM-DD hh:mm:ss" y el valor de la predicción.

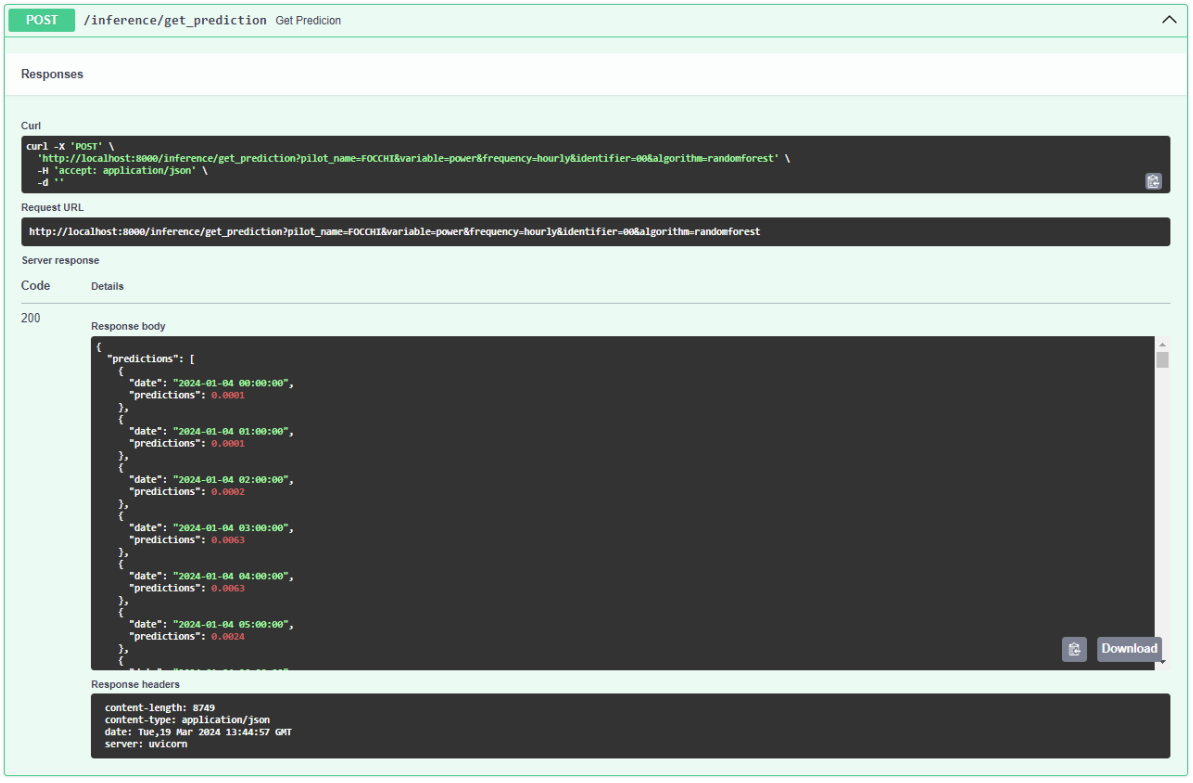

Figura 74: Endpoint response: get_predicion

Capítulo 7

Conclusiones

En este último capítulo de la memoria se incluye una recopilación de las ideas y conclusiones a las que se ha llegado tras la realización del presente proyecto.

La elaboración y el desarrollo de este trabajo de fin de máster ha supuesto una oportunidad de formación y/o profundización en los temas principales que toca. Los temas en los que se ha llevado a cabo investigación han sido muy diversos: formación en ingeniería y sistemas de energía para el conocimiento del entorno del proyecto, formación en los diferentes protocolos de acceso y de seguridad en las fuentes de datos, especialización en procesos y funcionalidades de tratamiento de datos en *Python*, formación sobre sistemas *big data* y el uso de *PDI* para la definición de las ETLs y los *jobs* asociados, formación sobre el despliegue y funcionamiento de *Apache Kafka*, investigación en definición de estructuras de bases de datos y técnicas de gestión *big data* en las mismas.

Durante el proceso de las tareas que se han llevado a cabo, se ha interactuado con otras tareas de DigiBUILD que no se engloban este TFM, pero ha sido necesario y beneficioso para poder tomar decisiones sobre diferentes cuestiones. Comenzando por la base de nuestra hipótesis, hemos estado en contacto con los responsables técnicos de los pilotos y con los propios pilotos para conocer en primera mano el estado, la relevancia, la accesibilidad y los sistemas que guardan los diferentes *datasets* enumerados en los documentos de DigiBUILD para poder adecuar dichos sistemas o los procedimientos y protocolos de acceso a los recursos para poder utilizarlos en *Petaho*. Iniciados los procesos de conexión y extracción en las fuentes de datos, se han tenido que modificar los planes iniciales como migraciones de los sistemas de almacenamiento, actualización de protocolos o similares. En cada uno de los pilotos se han conformado los formatos de los identificadores de los sensores de forma diferente en cada instancia en función de la disponibilidad de información. En adición, se han analizado los casos de uso de cada uno de los pilotos para agrupar los servicios y definir un listado de *topics* para la exportación de datos a *Kafka* de la forma más organizada y centralizada.

Como consecuencia del desarrollo de este proyecto dentro de DigiBUILD, se han podido plasmar los desarrollos e ideas generadas durante la realización en dos contribuciones en dos importantes congresos internacionales sobre eficiencia energética y energías renovables. Uno de

estos congresos es SpliTech[1]. Es un congreso internacional donde se exponen los desarrollos más novedosos sobre tecnologías inteligentes y sostenibles. En concreto, se proponen soluciones en los campos del internet de las cosas, eficiencia energética, *smart cities* o salud inteligente. El otro congreso es el taller internacional del IEEE sobre la metrología (o el estudio científico de las mediciones) para entornos donde vivimos o *MetroLivEnv*[2]. En él se exponen los desarrollos más punteros en relación a los avances en metrología, incluyendo diseños en sistemas a lo largo de todo el ciclo de vida de los edificios, como por ejemplo, diagnósticos, sistemas de monitoreo IoT, gemelos digitales o calidad de aire entre otros. Las contribuciones se han desarrollado con los compañeros de la *University College London*, con los que CARTIF comparte el desarrollo de tareas relacionadas con las capas del sistema en las que se desarrolla este trabajo. Las referencias (pendientes de publicación) son las siguientes:

- J. L. Hernández, D. Arévalo, S. Martín, K. Katsigarakis, G. N. Lilis, D. Rovas, I. de Miguel, "Using Extraction, Transformation and Loading procedures for digitalisation of buildings", *9th International Conference on Smart and Sustainable Technologies (SpliTech 2024)*, Split-Bol (Croacia), 25-28 de junio de 2024.
- J. L. Hernández, D. Arévalo, S. Martín, K. Katsigarakis, G. N. Lilis, D. Rovas, I. de Miguel, "Connection of dynamic and static data: A data lake for building digitalisation", *2024 IEEE International Workshop on Metrology for Living Environment (IEEE MetroLivEnv 2024)*, Chania, Creta (Grecia), 12-14 de junio de 2024.

[1] `https://splitech.org/`
[2] `https://www.metrolivenv.org/`

CAPÍTULO 7. CONCLUSIONES

Glosario

API Application Programming Interface | Interfaz de Programación de la Aplicación. 3, 4, 6, 8, 10, 15, 18, 19, 28–34, 36, 37, 42, 44–46, 50–52, 55–57, 63, 64, 71, 72, 89, 90, 93, 94

BBDD Databases | Bases de Datos. 36

BEM Building Energy Model | Model Energético del Edificio. 44

BEMS Building Energy Management System | Sistema de Gestión de Energía para Edificios. 18, 20, 40, 59

BI Business Intelligence | Inteligencia de Negocio. 9, 87

BIM Building Information Model | Modelo de Información del Edificio. 44

BMS Building Monitoring Systems | Sistemas de Monitoreo de Edificio. 18, 23, 25, 49

CINEA European Climate, Infrastructure and Environment Executive Agency | Agencia Ejecutiva del Clima, Infraestructura y Medioambiente Europea. 1

CKAN Comprehensive Knowledge Archive Network | Red de Archivos de Conocimiento Integral. 14

CSV Comma Separated Values | Valores Separados por Comas. 18, 19, 36, 37, 40, 41, 44, 49, 52, 59, 60

DBL Digital Building Logbook | Registro Digital del Edificio. 36

DEMS District Energy Managment System | Sistema de Gestión de Energía para Distritos. 18, 20, 40

DigiBUILD High-Quality Data-Driven Services for a Digital Built Environment towards a Climate-Neutral Building Stock | Servicios Basados en Datos de Alta Calidad para Entorno Digital Construido hacia un Conjunto de Edificios Respetuosos con el Medioambiente. I, 1–5, 8, 9, 11, 13, 14, 23, 44, 49, 59, 75, 77, 79, 82, 86–88, 96

DL Deep Learning | Aprendizaje Profundo. 11

DWH Data Warehouse | Almacén de Datos. 3, 4, 7, 19, 20, 25, 30, 31, 33, 34, 37, 41, 42, 46, 47, 50–52, 57, 60, 61, 63, 64, 68, 72, 75, 77–79, 84, 85, 87

Glosario

EC European Comission | Comisión Europea. 1, 14

EMS Energy Monitoring Systems | Sistemas de Monitoreo de Energía. 18, 23

EPC Energy Performance Certificate | Certificado de Eficiencia Energética. 36

ETL Extraction, transformation and Load | Extracción, transformación y carga. 6, 7, 9, 17, 19–24, 41, 47, 53, 56, 60–63, 68, 69, 71–73, 76–80, 82, 85, 87, 96

EV Electrical Vehicle Charge | Carga de Vehículo Eléctrico. 18, 21, 44

FTP File Transfer Protocol | Protocolo de Transferencia de Ficheros. 41, 59, 60

GUI Graphical User Interface | Interfaz Gráfica de Usuario. 8

HTTP HyperText Transfer Protocol | Protocolo de Transferencia de HiperTexto. 19, 29, 36, 45, 46, 50, 55, 72, 89, 90

IA Artifitial Intelligence | Inteligencia Artificial. 2, 3, 8, 11, 15

IAQ Indoor Air QUality | Calidad de Aire Interior. 44

IDS-RAM International Data Spaces-Reference Architecture Model | Modelo de Arquitectura de Referencia-Espacios de Datos Internacionales. 13

IDSA International Data Spaces Association | Asociación Internacional de Espacios de Datos. 13

IEEE Institute of Electrical and Electronics Engineers | Instituto de Ingenieros Eléctricos y Electrónicos. 97

IoT Internet of the Things | Internet de las Cosas. 5, 14, 97

JSON JavaScript Object Notation | Notación de Objeto JavaScript. 18, 19, 24, 25, 30, 33, 37, 42, 46, 50, 51, 56, 64, 72, 90, 92, 94

KPI Key Performance Indicator | Indicador Clave de Rendimiento. 23

MAC Media Access Control | Control de Acceso al Medio. 38

MAE Mean Absolute Error | Error Absoluto Medio. 94

ML Machine Learning | Aprendizaje Automático. 11, 62, 78

MQTT Message Queuing Telemetry Transport | Transporte de Mensajes Encolados de Telemetría. 18, 19, 23, 24, 44, 49–51, 62–64

MSE Mean Squared Error | Error Cuadrático Medio. 94

OLAP Online Analytical Processing | Procesado Analítico Online. 9, 86

PDI Pentaho Data Integration. 9

PV Photovoltaic Generation | Generación Fotovoltaica. 18, 21, 44

RDF Resource Description Framework | Marco de Descripción de Recursos. 87

SFTP Safe File Transfer Protocol | Protocolo Seguro de Transferencia de Ficheros. 18, 19, 40, 41

SQL Structured Query Language | Lenguaje Estructurado de Peticiones. 19, 25, 30, 31, 36, 40, 41, 47, 50, 67, 68, 80

SSH Secure Shell | Shell Seguro. 19

STD Standard Deviation | Desviación Típica. 77, 78

TCP/IP Transmission Control Protocol/Internet Protocol | Protocolo de Control de la Transmisión/Protocolo de Internet. 16

TFM - | Trabajo de Fin de Máster. 1–6, 8, 11, 19, 77, 79, 87, 89, 96

TLS Tansport Layer Security | Seguridad de la Capa de Transporte. 36, 37

URL Uniform Resource Locator | Localizador Uniforme de Recursos. 30, 56, 63

Bibliografía

[1] Comisión Europea, "High-Quality Data-Driven Services for a Digital Built Environment towards a Climate-Neutral Building Stock - Grant Agreement 101069658." https://doi.org/10.3030/101069658, 17 junio 2022. Accedido: 27 de mayo de 2024.

[2] Consorcio Proyecto DigiBUILD, "DigiBUILD - Página web." https://digibuild-project.eu/, 2023. Accedido: 27 de mayo de 2024.

[3] Agencia Ejecutiva del Clima, Infraestructura y Medioambiente Europea, Comisión Europea, "CINEA - Página web." https://cinea.ec.europa.eu/index_en, noviembre de 2023. Accedido: 27 de mayo de 2024.

[4] Comisión Europea, "Comisión Europea - Página web." https://commission.europa.eu/index_en, noviembre de 2023. Accedido: 27 de mayo de 2024.

[5] Agencia Ejecutiva del Clima, Infraestructura y Medioambiente Europea, "Programa de investigación "Energy Use (Horizon Europe)" - Página web." https://cinea.ec.europa.eu/programmes/horizon-europe/energy-use-horizon-europe_en, noviembre de 2023. Accedido: 27 de mayo de 2024.

[6] Joint Research Centre, "Bauhaus Initiative - Página web." https://new-european-bauhaus.europa.eu/about/about-initiative_en, julio de 2023. Accedido: 27 de mayo de 2024.

[7] Consorcio DigiBUILD, Euroheat & Power, "Communication, dissemination, and capacity-building activities (progress and report plan)," Tech. Rep. D7.3, Euroheat & Power, noviembre 2023.

[8] V. Mayer-Schönberger y K. Cukier, *Big data : la revolución de los datos masivos*. Turner Libros, junio 2013.

[9] I. Lee, "Big Data: dimensions, evolution, impacts, and challenges," *Business Horizons*, vol. 60, pp. 293–303, enero 2017.

[10] Hitachi Vantara LLC, "Pentaho Data Integration - Página web." https://www.hitachivantara.com/pentaho/pentaho-plus-platform.html. Accedido: 27 de mayo de 2024.

[11] Comité técnico ISO/IEC JTC 1, *Modelo OSI TP: Estándar ACID (ISO/IEC 10026-1:1998)*, ch. 4. Organización de Estandarización Internacional - ISO, octubre 1998. Accessed: 27 May 2024.

[12] Consorcio DigiBUILD, Ethnicon Metsovion Polytechnion, "DigiBUILD architecture towards an Energy Efficient Building Data Space," Tech. Rep. D1.5, Ethnicon Metsovion Polytechnion, abril 2023.

[13] International Data Spaces Association, "Reference Architecture Model," Tech. Rep. IDS-RAM 4.0, International Data Spaces Association, abril 2019.

[14] GAIA-X, "GAIA-X - Framework," Tech. Rep. 22.04, GAIA-X, abril 2022.

[15] FIWARE Foundation, "FIWARE - Página web." https://www.fiware.org/. Accedido: 27 de mayo de 2024.

[16] Comisión Europea, "Modular Big Data Applications for Holistic Energy Services in Buildings - Grant Agreement 101000158." https://doi.org/10.3030/101000158, 7 septiembre 2020. Accedido: 27 de mayo de 2024.

[17] Consorcio MATRYCS, Comisión Europea, "MATRYCS - Página web." https://matrycs.eu/. Accedido: 27 de mayo de 2024.

[18] Comisión Europea, "BD4NRG: Big Data for Next Generation Energy - Grant Agreement 872613." https://doi.org/10.3030/872613, 26 noviembre 2020. Accedido: 27 de mayo de 2024.

[19] Consorcio BD4NRG, Comisión Europea, "BD4NRG - Página web." https://www.bd4nrg.eu/. Accedido: 27 de mayo de 2024.

[20] Comisión Europea, "Harmonised Building Information Speedway for Energy-Efficient Renovation - Grant Agreement 820553." https://doi.org/10.3030/820553, 17 agosto 2018. Accedido: 27 de mayo de 2024.

[21] Consorcio BIM-SPEED, Comisión Europea, "BIM-SPEED - Página web." https://www.bim-speed.eu/en. Accedido: 27 de mayo de 2024.

[22] Comisión Europea, "BIM2TWIN: Optimal Construction Management & Production Control - Grant Agreement 958398." https://doi.org/10.3030/958398, 24 septiembre 2020. Accedido: 27 de mayo de 2024.

[23] Consorcio BIM2TWIN, Comisión Europea, "BIM2TWIN - Página web." https://bim2twin.eu/. Accedido: 27 de mayo de 2024.

[24] Comisión Europea, "Buildings as a Service (Ecosystem) - Grant Agreement 288409." https://cordis.europa.eu/project/id/288409/es, mayo 2012. Accedido: 27 de mayo de 2024.

[25] Consorcio BaaS, Comisión Europea, "Proyecto BaaS - Página web." https://www.baas-project.eu/. Accedido: 27 de mayo de 2024.

[26] Comisión Europea, “Smart Transition of EU cities towards a new concept of smart Life and Economy - Grant Agreement.” https://doi.org/10.3030/731297, 26 octubre 2016. Accedido: 27 de mayo de 2024.

[27] Consorcio mySMARTLife, Comisisón Europea, “mySMARTLife - Página web.” https://www.mysmartlife.eu/mysmartlife/. Accedido: 27 de mayo de 2024.

[28] Consorcio DigiBUILD, Fundación CARTIF, “Dynamic data ingestion and data quality methodologies,” Tech. Rep. D2.2, Fundación CARTIF, 11 2023.

[29] MQTT Org., “MQTT: The Standard for IoT Messaging - Página web.” https://mqtt.org/, 2022. Accedido: 27 de mayo de 2024.

[30] Comité Técnico ISO/IEC JTC 1/SC 22, “The JSON data interchange syntax,” Tech. Rep. ISO/IEC 21778:2017, Organización de Estandarización Internacional - ISO, noviembre 2017.

[31] InfluxData, Inc., “InfluxDBv3 Real-time Analytic Platform - Página web.” https://www.influxdata.com/products/influxdb-overview/. Accedido: 27 de mayo de 2024.

[32] Y. Shafranovich, “Common Format and MIME Type for Comma-Separated Values (CSV) Files.” RFC 4180, octubre 2005.

[33] REBEX ČR, “Estándares SFTP - Página web.” https://www.sftp.net/specification. Accedido: 27 de mayo de 2024.

[34] Google LLC, “Google Workspace: Google Calendar API Documentation.” https://developers.google.com/calendar/api/guides/overview?hl=es-419, marzo 2024. Accedido: 27 de mayo de 2024.

[35] Google LLC, “Google Workspace: Google Drive API Documentation.” https://developers.google.com/drive/api/guides/about-sdk?hl=es-419, marzo 2024. Accedido: 27 de mayo de 2024.

[36] PostgreSQL Organization, “PostgreSQL - Página web.” https://www.postgresql.org/, 2024. Accedido: 27 de mayo de 2024.

[37] Confluent, Inc., “Documentación Apache Kafka - Página web.” https://docs.confluent.io/kafka/overview.html. Accedido: 27 de mayo de 2024.

[38] Conduktor, Inc., “Convención de nombres en topics de Kafka - Página web.” https://www.conduktor.io/kafka/kafka-topics-naming-convention/, 2024. Accedido: 27 de mayo de 2024.

[39] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, y L. Stewart, “HTTP Authentication: Basic and Digest Access Authentication,” Tech. Rep. RFC 2617, Network Working Group, junio 1999.

[40] InfluxData, Inc., “InfluxDB Project - Repositorio de GitHub.” https://github.com/influxdata/influxdb, abril 2013. Accedido: 27 de mayo de 2024.

[41] L. L. Pipino, Y. W. Lee, y R. Y. Wang, “Data quality assessment,” *Communications of the ACM*, vol. 45, pp. 211–218, abril 2022.

[42] C. J. Fox, A. Levitin, y T. C. Redman, “The notion of data and its quality dimensions,” *Information Processing and Management*, vol. 30, pp. 9–19, enero 1994.

[43] F. Sidi, P. H. Shariat Panahy, L. S. Affendey, M. A. Jabar, H. Ibrahim, y A. Mustapha, “Data quality: A survey of data quality dimensions,” in *2012 International Conference on Information Retrieval & Knowledge Management*, pp. 300–304, marzo 2012.

[44] CARTIF, “Centro Tecnológico CARTIF - Página web.” https://www.cartif.es/. Accedido: 27 de mayo de 2024.

[45] Microsoft Power BI, “Understand Star Schema and the importance for power BI.” https://learn.microsoft.com/en-us/power-bi/guidance/star-schema, frebrero 2023. Accedido: 27 de mayo de 2024.

[46] Timescale, Inc., “ Timescale: PostgreSQL ++ for time series and events - Página web.” https://www.timescale.com/, 2024. Accedido: 27 de mayo de 2024.

[47] PostgreSQL Organization, “PostgreSQL Documentation: Chapter 11. Indexes.” https://www.postgresql.org/docs/current/indexes.html. Accedido: 27 de mayo de 2024.

[48] The Apache Software Foundation, “Apache Hadoop - Página web.” https://hadoop.apache.org/. Accedido: 27 de mayo de 2024.

[49] Pentaho y J. Hyde, “Pentaho: Mondrian Documentation.” https://mondrian.pentaho.com/documentation/olap.php. Accedido: 27 de mayo de 2024.