UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Modern Data Stack para analizar la situación epidemiológica

Presentado por José Daniel Ballester Delgado
en Universidad de Burgos — 5 de julio de 2022
Tutores: José Ignacio Santos Martín y José Manuel Galán Ordax

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. José Ignacio Santos Martín y D. José Manuel Galán Ordax, profesores del departamento de Ingeniería Civil, área de Ingeniería de Organización.

Exponen:

Que el alumno D. José Daniel Ballester Delgado, con DNI 71565431B, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Modern Data Stack para analizar la situación epidemiológica.

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

En Burgos, 5 de julio de 2022

Vº. Bº. del Tutor:

SANTOS MARTIN JOSE IGNACIO - 09331511C

Firmado digitalmente por SANTOS MARTIN JOSE IGNACIO - 09331511C
Fecha: 2022.07.05 10:28:02 +02'00'

D. José Ignacio Santos Martín

Vº. Bº. del co-tutor:

GALAN ORDAX JOSE MANUEL - 71121824M

Firmado digitalmente por GALAN ORDAX JOSE MANUEL - 71121824M
Fecha: 2022.07.05 10:31:15 +02'00'

D. José Manuel Galán Ordax

Resumen

La gran diversidad de fuentes de información en internet provoca desconfianza a la hora de consultar datos, ya que en muchas ocasiones es falsa o no está verificada. Por tanto este trabajo recopilará información actualizada sobre la situación de la pandemia de las principales fuentes oficiales, y brindará una opción visual e interactiva que proporciona el acceso a esta información veraz, sobre el estado de la pandemia causada por el Covid-19 en España.

Y todo ello se desarrollará utilizando un nuevo paradigma a la hora de construir la arquitectura que da soporte al ciclo de vida de los datos, el Modern Data Stack, que haciendo uso de la tecnología en la nube, trae muchas ventajas con respecto a la arquitectura tradicional.

Descriptores

Análisis de datos, Covid-19, Business Intelligence, Data Warehouse, KPI, Nube, OLAP, ELT, Modern Data Stack, Analytics engineer.

Abstract

The great diversity of information sources on the internet causes mistrust when consulting data, since it is often false or unverified. Therefore, this work will compile updated information of the pandemic's situation, from the main official sources and will provide a visual and interactive option that will give access to this truthful information, about the state of the pandemic caused by Covid-19 in Spain.

And all this will be developed using a new architecture paradigm that supports the data life cycle, the Modern Data Stack, which, by making use of cloud technology, brings many advantages in comparison to the traditional architecture.

Keywords

Data analysis, Covid-19, Business Intelligence, Data Warehouse, KPI, Cloud, OLAP, ELT, Modern Data Stack, Analytics engineer

Índice general

Índice de figuras

Índice de tablas

Introducción

La gran diversidad de fuentes de información en internet provoca desconfianza a la hora de consultar datos, ya que en muchas ocasiones la información es falsa o no está verificada.

Este trabajo tiene como objetivo brindar una solución visual e interactiva que facilite el acceso a información de la situación del Covid-19 en España.

Como respuesta a este problema se prevé desarrollar un cuadro de mandos que recopile los principales datos de las fuentes oficiales encargadas de publicar información sobre la situación actual del Covid-19 en el país, de forma que el usuario tenga una opción centralizada y fiable para consultar la información que desea y poder observar insights sobre ésta.

Asimismo, otra novedad que aporta con respecto a las alternativas más clásicas, es la interacción del usuario con los datos, permitiendo una modificación dinámica de la visualización mediante el uso de filtros, de manera que permita obtener información específica y más detallada sobre un dato o un conjunto de datos.

Estos análisis son complicados con los sistemas transaccionales tradicionales (OLTP), porque la información está dispersa en muchas tablas y no están dirigidas al análisis.

Debido a estos problemas, se ha introducido el uso de Business Intelligence y los sistemas analíticos (OLAP), donde la información se integra en un repositorio optimizado, para responder preguntas estratégicas.

El Business Intelligence (BI) se conoce como la capacidad de transformar los datos en información y la información en conocimiento. Se logra a través del almacenamiento y procesamiento de grandes cantidades de datos para que la información pueda ser analizada.

Otro aspecto importante, es que en este TFG se pretende utilizar el Modern Data Stack, un nuevo paradigma a la hora de montar la arquitectura que da soporte al ciclo de vida de los datos, gracias a las ventajas que tiene montar esta estructura en la nube.

El término "data stack" se origina en "technology stack", la combinación de diferentes tecnologías, que los ingenieros de software combinan para crear productos y servicios.

Los data stack están diseñados específicamente para respaldar el almacenamiento, la administración y el acceso a los datos y se usan para aprovechar los datos en la toma de decisiones estratégicas.

El Modern Data Stack (MDS) consiste en un conjunto flexible de tecnologías que ayudan a almacenar, administrar y aprender de los datos.

La nube ha transformado la industria del software y la forma en que las empresas construyen sus productos. Hoy, podemos configurar un technology stack en una fracción del tiempo menor y de forma más económica. Y no sorprende que estas transformaciones hayan allanado el camino para el Modern Data Stack (MDS) y un cambio de paradigma en este tipo de soluciones.

1.1. Estructura de la memoria

La memoria está formada por la siguiente estructura:

- **Introducción:** concisa presentación del problema a solucionar junto con la resolución sugerida. Se incluye la estructura de la memoria y los materiales adjuntos.
- **Objetivos del proyecto:** definición de los objetivos que pretende alcanzar el proyecto.
- **Conceptos teóricos:** breve desarrollo de los conceptos teóricos fundamentales para la correcta interpretación del resultado planteado.
- **Técnicas y herramientas:** exposición de métodos y herramientas empleadas para organización y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** muestra de fases relevantes a destacar durante el transcurso de la elaboración del proyecto.
- **Trabajos relacionados:** estado del arte en el campo de las arquitecturas de datos para el análisis de datos mediante reporting.
- **Conclusiones y líneas de trabajo futuras:** conclusiones extraídas de los resultados obtenidos a partir de la realización del proyecto y opciones de mejora o ampliación de la resolución planteada.

1.2. Materiales adjuntos

- **Anexos:**
  - **Plan del proyecto software:** organización y potencialidad del proyecto.
  - **Especificación de requisitos del software:** se incluye la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
  - **Especificación de diseño:** se expone la fase de diseño; el ámbito del software, el diseño de datos, el diseño procedimental y el diseño arquitectónico.
  - **Manual del programador:** comprende los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
  - **Manual de usuario:** guía de usuario para el buen uso y empleo adecuado de la aplicación.
- **Repositorio Github:** contiene la memoria y los anexos, además del archivo del cuadro de mandos, los accesos los sercivios y herramientas cloud utlilizados y los scripts de dbt, Snowflake y Phyton del proyecto.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

2.1. Objetivos generales

- Desarrollar una arquitectura de datos que recopile la información de las principales fuentes oficiales encargadas de publicar datos sobre el estado de la pandemia en España.
- Almacenar toda la información recopilada de forma organizada y accesible en la nube.
- Realizar la extración, carga y transformación de los datos automáticamente en la nube.
- Proporcionar un acceso visual a la información a través de representaciones gráficas.
- Facilitar al usuario la interacción con los datos mediante la aplicación de filtros para la selección de la información.

2.2. Objetivos técnicos

- Usar Snowflake como data warehouse analítico en la nube para el almacenamiento de datos.

- Hacer uso de la herramienta DBT la transformación de los datos directamente en una datawarehouse en la nube.
- Usar la herramienta Fivetran para extración y carga de datos automática.
- Realizar la visualización de datos con la aplicación PowerBI.
- Utilizar la herramienta GitHub para realizar el control de versiones.
- Emplear la metodología ágil Scrum para la organización del proyecto (y el desarrollo del software).
- Usar la plataforma ZenHub para la gestión del proyecto.

2.3. Objetivos personales

- Realizar una aportación a la modernización del Business Intelligence y los sistemas OLAP
- Profundizar en el Modern Data Stack (MDS) mediante la utilización de tecnologías que ayudan a extraer, cargar, transformar y visualizar los datos
- Examinar metodologías, herramientas y aplicaciones actuales.
- Ampliar mis conocimientos en el modelado de datos en sistemas OLAP
- Profundizar en las transformaciones en la nube del nuevo método Extract-Load-Transform (ELT)

Conceptos teóricos

Apartado que sintetiza los conceptos teóricos del dominio de conocimiento del proyecto, necesarios para su comprensión y desarrollo.

3.1. Business Intelligence

El Business Intelligence (BI) o Inteligencia Empresarial [14] comprende las aplicaciones, las herramientas y las infraestructuras, así como el conjunto de desarrollos necesarios para obtener información conveniente para las empresas, mediante la conversión de datos, de forma que posibilita el análisis del funcionamiento de las empresas con el objetivo de progresar y potenciar las determinaciones y productividad de las mismas.

Fundamentalmente la ventaja más importante que proporciona el Business Intelligence, es que posibilita la aportación de información beneficiosa a las empresas, de manera que facilita la identificación de los factores que impactan en los mercados y negocios, y por tanto, a su rendimiento.

El Business Application Research Center (BARC) mediante la realización del BI Survey, aportó una lista que contiene los beneficios fundamentales aplicados a la inteligencia empresarial:

- Preparar informes, examinar y organizar de forma más ágil y exacta.
- Progresar en la toma de decisiones.
- Aumentar la eficacia operativa.
- Mejorar la satisfacción y respuesta tanto de los clientes como de los empleados.

- Disminuir los costes.
- Acrecentar los ingresos.
- Producir estrategias más determinadas al futuro.
- Adquirir datos de mayor calidad y obtener ventajas competitivas superiores.

La principal función de Business Intelligence, es la obtención de datos para su análisis, valoración de riesgos y toma de decisiones.

Para ello, Business Intelligence dispone de herramientas de *reporting (reporting tools)*.

Las herramientas de *reporting* se usan para la representación de la información manejada, de forma que se prensenta de una manera clara y sencilla.

El objetivo de estas herramientas es proporcionar, tanto la información deseada, como su análisis, puntualizando en los detalles requeridos por cada tipo de usuario. Así, los usuarios pueden obtener información eficiente representada mediante tablas, gráficos y cuadros, entre otras.

3.2. *Data Warehouse*

Un *Data Warehouse* [5] es una base de datos que proporciona el almacenamiento de datos procedentes de distintas fuentes, con el objetivo de incorporar y depurar la información para posteriormente ser analizada en base a diversos factores. Por ello, es una plataforma usada en los sistemas de Business Intelligence.

Los Data Warehouse ejercen de repositorio central, ya que la información procede de distintas fuentes, como sistemas transaccionales u otras bases de datos vinculadas.

De esta forma, las empresas pueden disponer de una percepción de la totalidad de la información que se desea, afianzando las comprobaciones correspondientes.

3.3. KPI

Un KPI [3] es un indicador clave de rendimiento o indicador clave de desempeño. Este indicador trata de mostrar las variables, factores y unidades de medida para producir análisis.

Para el correcto funcionamiento, el indicador debe seguir una serie de propiedades:

- Accesible. Los fines deben ser realistas y alcanzables.
- Evaluable. Un KPI debe poder ser evaluado.
- Significativo. Se deben seleccionar los datos más relevantes.
- Constante. El indicador debe ser sometido a análisis constantes.
- Preciso. Debe seleccionar información precisa.

3.4. Data lake

Un data lake [13] es un repositorio centralizado, que se encarga de almacenar, procesar y proteger grandes cantidades de datos estructurados, semiestructurados o sin estructurar. Este repositorio permite almacenar datos en su formato nativo y procesar cualquier tipo de datos, sin tener en cuenta los límites de dimensión.

Un data lake ofrece una plataforma fiable de manera que facilita a las empresas realizar diversas funciones: transmitir cualquier dato desde cualquier sistema y a cualquier velocidad, además de almacenar cualquier tipo o volumen de datos.

3.5. *Data Mart*

Un *Data Mart* [18] es una versión característica de base de datos, dirigida en una sección o área de negocios en particular.

Este almacén de datos es caracterizado por su distribución óptima, ya que permite el acceso a datos específicos de diversas áreas de forma simple, realizando el análisis de la información en función de distintos criterios.

Para ello ejerce diversas labores, como:

- La planificación y disposición de la información para su posterior examinación.
- La realización de señalizadores clave de rendimiento (KPI).
- La elaboración de los informes para una formación automática e instantánea.
- La valoración de los datos, apreciando su ejecución de acuerdo a las finalidades del sector.

Además, los *Data Marts* ofrecen una serie de beneficios y ventajas:

- Agilidad y rapidez en la realización de consultas.
- Disponibilidad de consultas simples SQL y/o MDX.
- Comprobación inmediata de la información.
- Manejo de volúmenes reducidos de datos.

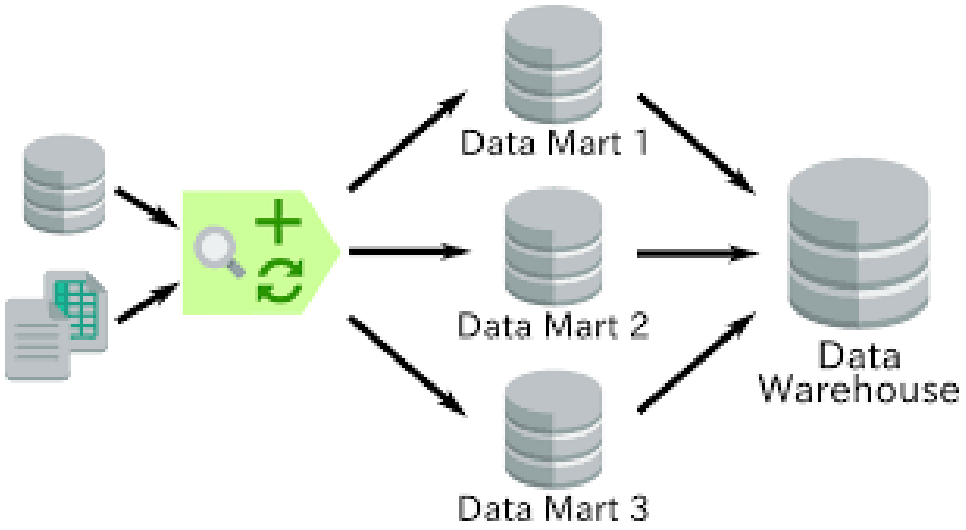

Figura 3.1: Ejemplo Data Mart.

Para la creación del *Data Mart*, se debe disponer de una organización válida para poder realizar el análisis de la información. Por lo tanto, se debe determinar la base de datos OLTP o OLAP a usar.

3.6. OLTP *(Online Transactional Processing)*

El procesamiento de transacciones en línea, es un tipo de procesamiento que proporciona y coordina las aplicaciones transaccionales para el acceso de información , restablecimiento y tratamiento de transacciones. Así, los sistemas OLTP son bases de datos empleadas para el procedimiento de transacciones.

Las transacciones producen procesos que deben ser comprobados y validados con un *commit* o invalidados con un *rollback*, en los que se incluyen diversas operaciones de inserción, modificación o borrado de datos.

Las funciones que caracterizan estos sistemas son los siguientes:

- En estos sistemas, la entrada a los datos está simplificada para aquellas operaciones reiteradas de lectura y escritura.
- En función del nivel de aplicación, los datos se organizan de distintas maneras.
- La representación de los datos no es siempre uniforme.
- El historial de los datos se encuentra reducido a la muestra de la información vigente.
- La información se restablece en tiempo real.
- Ausencia de variación e inflexibilidad del esquema.
- Elaboración de informes sencillos.
- Grandes cantidades de datos de entrada.
- La información se encuentra resumida de manera especifica.

3.7. OLAP *(Online Analytical Processing)*

El procesamiento analítico en línea, consiste en la interpretación de grandes volúmenes de datos, con el objetivo de obtener información relevante y realizar su análisis complejo.

Las funciones que caracterizan estos sistemas son los siguientes:

- La entrada de datos, suele ser sólo de lectura, de manera que la operación más usada es la de consulta, y en pocas ocasiones se realizan operaciones de inserción, actualización o borrado de datos.

- La representación de los datos es uniforme.
- Los datos se organizan en función de las distintas áreas.
- Incluye un historial de la información a largo plazo, generalmente de dos a cinco años.
- Representa los datos de forma resumida.
- Realiza el análisis de grandes volúmenes de datos.
- Ejerce la adjunción de datos.
- Estas bases de datos tienden a obtener la información de los sistemas operacionales presentes, a través de procedimientos de extracción, transformación y carga (ETL).

Tabla de hechos

Una tabla de hechos [32] es la tabla central de un esquema dimensional contenida tanto en los esquemas de estrella, como los esquemas de copo de nieve, en la que se almacenenan distintas medidas.

Estas medidas se obtienen de la intersección de las dimensiones que la definen, procedentes de la tabla de dimensiones, y que están relacionadas con la tabla de hechos.

Tabla de dimensiones

La tabla de dimensiones [33] contiene aquellos atributos usados para la restricción y agrupación de los datos pertenecientes a una tabla de hechos en la ejecución de las consultas de datos.

Las tablas de dimensiones facilitan información sobre los datos de la tabla de hechos mediante un análisis. Por tanto, las tablas de dimensiones son las que disponen de los metadatos de los hechos. Y las tablas de hechos son las que contiene dichos datos.

Modelo estrella

El modelo estrella [21] es un tipo de esquema de base de datos racional, compuesto por una sola tabla de hechos central, envuelta por varias tablas de dimensiones.

Las diferentes ramificaciones que realizan la conexión con las tablas, señalan la relación de muchos a uno, entre la tabla de hechos y las distintas tablas de dimensiones.

Por ello, se trata de una estructura más sencilla al contener solo una única tabla de hechos central, que dispone de los datos requeridos para el análisis, junto con sus múltiples conexiones a varias tablas de dimensión.

De esta forma todos los datos vinculados están incluidos en una única tabla, y solamente esa tabla mantiene relación con otras tablas.

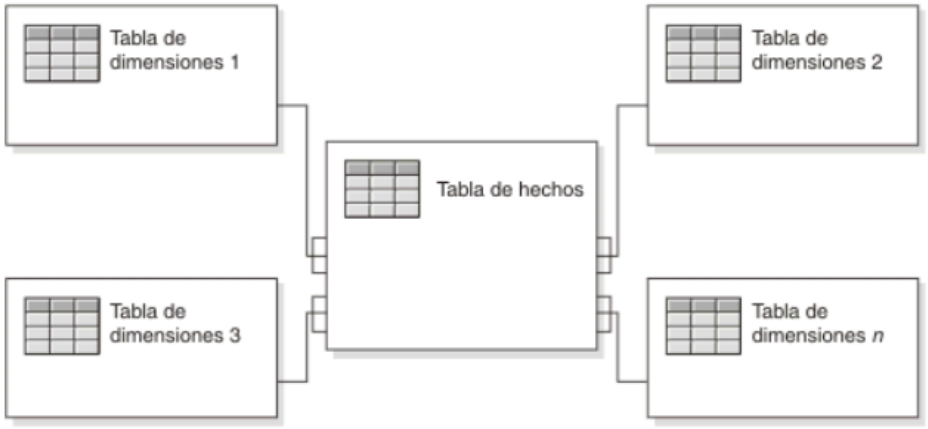

Figura 3.2: Ejemplo de modelo estrella.

Al tratarse de un esquema de estructura sencilla, permite el acceso a los datos de manera rápida, ya que posibilita la aplicación de una base de datos multidimensional, a través del uso de una base de datos relacional.

Este modelo, al vincular solamente la tabla de hechos con las tablas de dimensiones, implica que las operaciones de lectura de la información de la base de datos sea más rápida.

Por lo tanto, el modelo estrella resulta más sencillo que el modelo de copo de nieve.

Modelo copo de nieve

El modelo copo de nieve [20] es un tipo de esquema de base de datos, en el que una tabla de hechos está vinculada a muchas tablas de dimensiones, que pueden estar vinculadas a otras tablas de dimensiones, mediante una relación de muchos a uno.

Además, cada tabla de dimensión interpreta un nivel de clasificación.

Normalmente, las diversas tablas de un modelo de copo de nieve se encuentran normalizadas en el tercer formulario de normalización.

Así, el modelo de copo de nieve puede estar compuesto por diversas tablas de dimensiones y cada tabla tabla de dimension puede estar compuesta también por varios niveles.

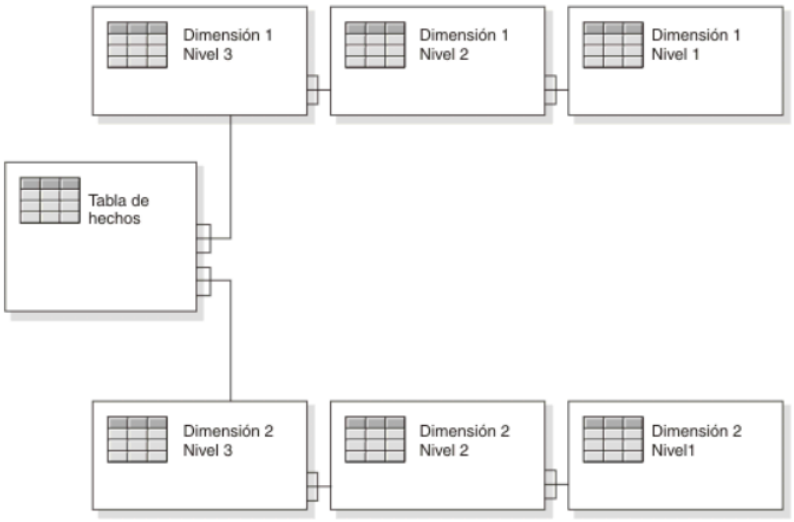

Figura 3.3: Ejemplo de modelo copo de nieve.

Por lo tanto, este modelo trata de vincular tanto la tabla de hechos, como de vincular las tablas de dimensiones entre otras tablas, sin tener conexión con la tabla de hechos, de forma que la tablas de dimensiones puedan estar divididas en otras subtablas.

En el modelo de copo de nieve, al disponer de más relaciones entre tablas, provoca que la consulta de datos sea más complicada.

3.8. Diferencias entre OLTP y OLAP

A continuación, se incluye una tabla que contiene las principales diferencias entre OLTP y OLAP [2] :

	OLTP	**OLAP**
Definición	Sistema transaccional en línea que administra los cambios de la base de datos.	Sistema de restauración y análisis de datos en línea.
Función	Insercción, actualización, y borrado de datos.	Extracción de datos para su análisis.
Tipo de usuario	Operacional.	Analista
Transacción	Transacciones cortas.	Transacciones largas.
Tiempo	Tiempo de procesamiento de transacción menor.	Tiempo de procesamiento de transacción mayor.
Modelo de datos	Relacional.	Multidimensional.
Consultas	Consultas sencillas.	Consultas complejas.
Normalización	Tablas normalizadas (3NF).	Tablas no normalizadas.
Integridad	Limitación de integridad.	Integridad no afectada.

Tabla 3.1: Diferencias entre OLTP y OLAP.

Por lo tanto, las principales ventajas por las que se caracteriza OLAP son las siguientes:

- Rapidez de la ejecución de las consultas.
- Disponibilidad de datos para realizar consultas.
- Reducción de tiempo en cáculos y elaboración de informes.
- Reducción de tiempo en el procesamiento de consultas de usuario.

Los análisis que se hacen en reporting son complicados con los sistemas transaccionales tradicionales (OLTP), porque la información está dispersa en muchas tablas y no están dirigidas al análisis.

Debido a estos problemas, se introdujeron el uso de Business Intelligence y los sistemas analíticos (OLAP), donde la información se integra en un repositorio optimizado para responder preguntas estratégicas.

3.9. ETL

Los procesos ETL (*Extract, Transform and Loading*) [4] consisten en la extracción, transformación y carga de datos, de manera que siguen tres fases:

- Extracción: obtención, análisis, interpretación y translado de datos desde diferentes fuentes internas y externas.
- Transformación: filtrado, depuración y agrupación de datos.
- Carga: carga y actualización de datos en otras bases de datos, data mart, o data warehouse, entre otros sistemas.

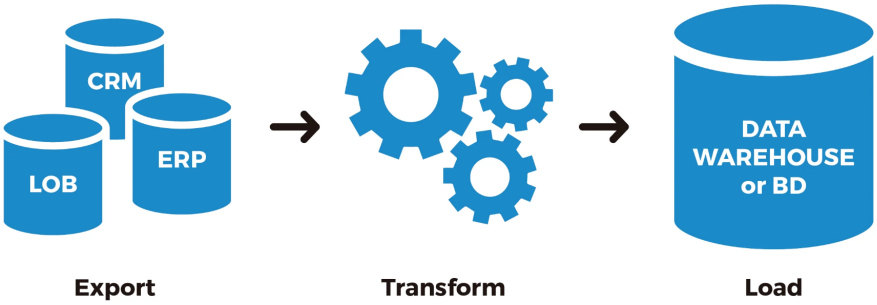

Figura 3.4: Ejemplo proceso ETL.

Así, mediante estos procesos se alcanza la consistencia entre las apliaciones y los sistemas.

3.10. *Modern Data Stack*

Los Modern Data Stack [28]toman ventaja de los data warehouses en la nube, ya que traen mejoras en seguridad y elasticidad, y principalmente en

la el almacenamiento y procesamiento de grandes volúmenes de datos a gran velocidad y un coste muy bajo.

Los data warehouses solían ser un gran cuello de botella. Se usaba principalmente bases de datos relacionales basadas en filas como sus data warehouses, lo que no se adaptaba bien a las cargas de trabajo de análisis de datos, ya que distribuye los datos relacionados en varios discos o servidores. Incluso con tecnologías como Hadoop tardaban horas en ejecutarse y eran muy complicados de escribir y mantener.

Además, debido al limitado poder de procesado de las antiguas data warehouses los ingenieros de datos solían hacer el trabajo de transformación antes de cargar los datos, lo que dio lugar al término ETL (*Extract-Transform-Load*).

Ahora, con el avance de los data warehouse en la nube, gracias a su alto rendimiento, los ingenieros de datos pueden ejecutar consultas a escala de petabytes en minutos.

Con un *Modern Data Stack*, se pueden cargar los datos en el data warehouse en minutos y la transformación de datos se puede manejar de manera mucho más efectiva allí que en alguna capa de procesamiento externa (ELT, *Extract-Load-Transform*).

Los princiapes beneficios de un Modern Data Stack por lo tanto son:

- Modularidad: debido a que los Modern Data Stack consisten en varias tecnologías con puntos de conexión general, se pueden cambiar partes del stack a medida que las necesidades cambian, lo que ayuda a evitar el vendor lock-in.

- Velocidad: debido a los límites de procesamiento de las antiguas data warehouses, las pipelines podían llegar a tardar horas en ejecutarse, pero ahora con el Modern Data Stack y sus recursos de cómputo elásticos, ese trabajo puede hacerse en minutos.

- Coste: los costes de la tecnología en la nube son normalmente significativamente más baratos que su contraparte on-premise. Un data warehouse antiguo tendría que pagar por un servidor todo el tiempo, aparte de ser costoso y disponer de dificultades en su escalabilidad, al contrario que con un data warehouse en la nube, donde solo pagas por lo que usas y puedes escalar para cargas de trabajo masivas cuando sea necesario.

3.11. ELT

Los procesos ELT (*Extract, load and transform*) [17] consisten en la extracción, carga y transformación de datos.

De manera que se diferencian de los procesos ETL en que los datos no se transforman al obtener los datos, sino que se guardan antes de ser procesados, es decir son almacenados en su formato inicial.

Esta nueva perspectiva esta compuesta por:

Modern ingestion

La ingestión de datos [24] comprende las fases de extracción de datos desde la fuente y la carga de estos mismo en el destino, la E y L del proceso ETL.

Modern Storage

El almacenamiento de datos [24] se realiza mediante almacenes de datos en la nube, entre los que se incluyen *Snowflake*.

Actualmente, estos almacenes de datos modernos, presentan una serie de mejoras respecto a los anteriores, ya que permiten la ausencia de configuraciones de *hardware*, y se caracterizan por su disponibilidad, rapidez y menor coste.

Modern transformation

La transformación moderna de datos [24] convierten, limpian y agregan los datos entre otras funciones. Y todas estas acciones se realizan directamente sobre el data warehouse analítico.

3.12. Diferencias entre ETL Y ELT

A pesar de que hay diferencias entre ETL y ELT, ambos cumplen con el mismo ojetivo, tener unos datos listos para el análisis y toma de decisioes,

Implementar un proceso ETL es más complejo, ya que requiere más esfuerzo en el diseño y ejecución, pero puede ofrecer más beneficios a largo plazo ya que es más económico y requiere menos recursos y tiempo.

A continuación, se incluye una tabla que contiene las principales diferencias entre los procesos ETL y ELT [22] :

	ETL	ELT
Orden de los procesos	Los datos se transforman antes de ser cargados en el sistema receptor.	Los datos se extraen y se cargan en el sistema receptor sin ser transformados. Estos datos son transformados en el sistema de destino.
Enfoque	Variación de datos, encubrimiento de datos, normalización, y unión entre tablas.	Carga en almacenamiento de datos. División de la carga de transformación y realización de las transformaciones en los almacenes.
Privacidad	La información reservada se puede registrar antes de su carga en el sistema receptor.	Los datos se cargan sin un previo procesamiento. El proceso de encubrimiento se realiza en el sistema receptor.
Mantenimiento	Los procesos de transformación y administración de las modificaciones de datos pueden exigir más gastos.	Mantenimiento acometido en el almacén de datos en el que se producen las transformaciones.
Latencia	Generalmente latencia más alta.	Generalmente latencia más baja.
Flexibilidad de datos	Reglas y lógica personalizada.	Soluciones generalizadas.
Flexibilidad de análisis	Tanto los casos de uso como los modelos de informen deben ser establecidos previamente.	Los datos pueden ser añadidos en cualquier momento.
Escala de datos	Sin sistemas de procesamiento distribuidos y escalables puede ocurrir un cuello de botella por ETL.	Más escalable, con menor procesamiento.

Tabla 3.2: Diferencias entre ETL y ELT.

3.13. Analytics engineers

Los analytics engineers[9] aportan datos limpios a los usuarios finales, modelan los datos de forma que empoderan a los usuarios para contestar

sus propias preguntas. Para ello dedican su tiempo a transformar, testear, desplegar y documentar datos, aplicando las mejores prácticas de la ingeniería del software como control de versiones e integración continua a la base de código. A continuación, se va a analizar la tendencia del mercado que hace que esté en alza este nuevo rol en los equipos de datos modernos.

El equipo de datos tradicional

Para entender bien, como surgió este rol, tenemos que retrotraernos al equipo de datos tradicional, donde había un ingeniero de datos, el cual se encargaba de la extracción de datos de la base de datos, y posteriormente su transformación y carga en el almacén de datos respectivo.

También se podía encontrar el analista de datos encargado de la creación de los informes de los datos.

Con el tiempo surgieron múltiples cambios en el ámbito de las herramientas de datos, entre los que se incluyen:

- Los almacenes de datos en la nube.
- Los servicos de data pipelines.
- Las herramientas de inteligencia empresarial.

Debido a estos cambios y evoluciones en las herramientas de datos, también surgieron cambios en los puestos que se ocupan de estas herramientas.

El equipo de datos moderno

Hoy en día dentro de un equipo de datos moderno, lo primero que se quiere contratar es una persona que domine todo el data stack, una persona que sea capaz de configurar una herramienta de ingestión de datos, mantener un data warehouse limpio, escribir transformaciones complejas y hacer reporting sobre unos datos limpios.

Este rol no es ni un ingeniero de datos, ni un analista de datos, sino algo intermedio, lo que denominamos analytics engineer, el cual está encargado de facilitar los datos definidos, transformados, testeados, documentados, y revisados.

Actualmente, se pueden distinguir tres cargos:

- Ingeniero de datos.
- Ingeniero de análisis.
- Analista de datos.

Una vez el equipo de datos empieza a crecer, se contratan a ingenieros de datos y analistas de datos, que se podrán están más especializados en unas tareas más concretas que en el equipo de datos tradicional, aunque a veces las líneas entre los roles pueden no estar claras ya que un analytics engineer puede hacer gran parte del trabajo de un ingeniero de datos y de un analista de datos.

A continuación, se incluye una tabla que contiene las principales tareas de los 3 cargos :

Ingeniero de datos	**Analytics engineer**	**Analista de datos.**
Integraciones de datos personalizadas	Proporcionar datos depurados y transformados preparados para el análisis	Deep insights
Administrar pipelines	Aplicar las mejores prácticas de ingeniería de software al código	Trabajar con usuarios comerciales para entender los requisitos de datos
Implementar endpoints de aprendizaje automático	Proporcionar la documentación y las definiciones de los datos	Elaborar cuadros de mando
Producción y mantenimiento de la plataforma de datos	Enseñar a los usuarios comerciales cómo emplear las herramientas de visualización de datos	Forecasting
Optimización del rendimiento del almacenamiento de datos		

Tabla 3.3: Equipo de datos moderno.

Técnicas de la ingenieria del sofware que aplican los analytics engineers

Tristan Handy [26] expuso una serie de métodos que los analytics engineers utilizan, que son propios de ingenieros del software, que antes de la aparición de este rol no se utilizaban en los equipos de datos tradicionales:

1. Control de versiones del código:
    - Anteriormente, los diferentes cargos como, los analistas e ingenieros de datos, guardaban sus cometidos en sus equipos personales o, en ocasiones disponían de un repositorio en el que podían almacenar sus cambios. Cuando se producía un cambio de código que interrumpía su correcto funcionamiento, surgía un gran problema ya que no sabían de dónde procedía el fallo.
    - Por ello, actualmente, los distintos cargos realizan sus códigos de análisis en repositorios *git* compartidos. De esta forma, colaboran de manera que realizan las revisiones de código pertinentes, antes de compartirlo con los demás cargos.
2. Código con garantía de calidad.
    - Anteriormente, no se disponía de pruebas en las transformaciones de datos.
    - Por ello, hoy en día, los equipos disponen de tests en las transformaciones de datos, de forma que se identifican los errores antes de que se comparta el resultado.
3. Código modular.
    - Anteriormente, los informes contenían las mismas consultas y reprodrucían las mismas fórmulas.
    - Actualmente, los distintos cargos cooperan mediante el uso de una base de código unificada, de forma que unos cargos pueden utilizar el trabajo de otros, optimizando su tiempo.
4. Uso de entornos.
    - Anteriormente, no se utilizaban entornos de desarrollos en los que se realizaban las pruebas correspondientes antes de la producción, sino que se probaba en el momento de la producción.

- Actualmente, los equipos disponen de software que prueban y revisan las modificaciones, pudiendo ver los cambios producidos antes de la producción.

5. Código diseñado para la mantenibilidad.

   - Anteriormente, mayoritariamente la actualización de nuevos datos producían el borrado de los modelos de datos ya existentes.
   - Por ello, actualmente, debido al modelado de datos modular los equipos ofrecen la seguridad pertinente para que no se produzcan cambios indeseados en los datos de origen, de manera que solo se produzcan las actualizaciones de los nuevos cambios.

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.

4.1. Gestión del proyecto

Scrum

[39] Es un proceso empleado en los proyectos para el desarrollo ágil. Consiste en la realización de entregas incompletas del proyecto mediante *sprints*, con la finalidad de acelerar el desarrollo de la entrega final del proyecto.

De esta manera las entregas parciales serán comprobadas y se irán desarrollando dentro del plazo fijado.

GitHub

[8] Es un servicio consistente en una nube que almacena un sistema de control de versiones (VCS) reconocido como Git. Por tanto, ofrece todas las funciones de Git, tales como documentación, revisión de código, wiki, gestión de tareas, etc.

Github Desktop

[1] Es una aplicación que permite la interacción con GitHub a través de una interfaz gráfica que sustituye su uso mediante línea de comandos o de buscadores web.

Esta aplicación facilita la actualización de los cambios que se van produciendo en el proyecto en el repositorio de GitHub.

ZenHub

[40] Es una plataforma de gestión de proyectos que permite su visualización, la organización de sprints y las herramientas para la elaboración de informes. Para ello ofrece un tablero canvas que equivale un issue de GitHub. Nos permite asignar a cada tarea una estimación mediante story points, priorizarla según la posición en la lista, asignar un responsable y a que sprint pertence.

4.2. Herramientas

Microsoft Power BI

[37] Es una herramienta que posibilita la conexión a distintas fuentes de datos locales o en la nube. De forma que se puede ajustar la información fácilmente, permitiendo producir tableros de control en tiempo real, e informes que incluyen la información óptima para la mejora del desarrollo de los resultados.

Así, *Microsoft Power BI* permite unir diferentes fuentes de datos, modelizar y analizar datos para después, presentarlos a través de paneles e informes; y que así puedan ser consultados de una manera fácil, atractiva e intuitiva.

Por ello, es una herramienta usada para la fase de análisis y creación de paneles e informes, que permite realizar diversos gráficos que ayudan a visualizar los datos y sacar insights de forma cómoda y flexible gracias a la posibilidad de aplicar filtros dinámicos.

Snowflake

[7] Es un data warehouse analítico en la nube que utiliza un repositorio de datos centralizado para datos persistentes, accesible desde todos los nodos del data warehouse, y que cuando va a procesar las consultas cada cluster almacena una porción de los datos localmente par procesarlos en pararelo.

La arquitectura de Snowflake tiene 3 capas:

- Database storage: en esta capa están los datos, una vez están en la nube, Snowflake los reorganiza en su propio formato para mejorar el rendimiento.

- Query processing: esta capa reliza las consultas mediante almacenes virtuales, cada uno de ellos es un cluster de varios nodos que trabajan en parelelo.
- Cloud services: son varios servicios que coordinan las actividades de Snowflake.

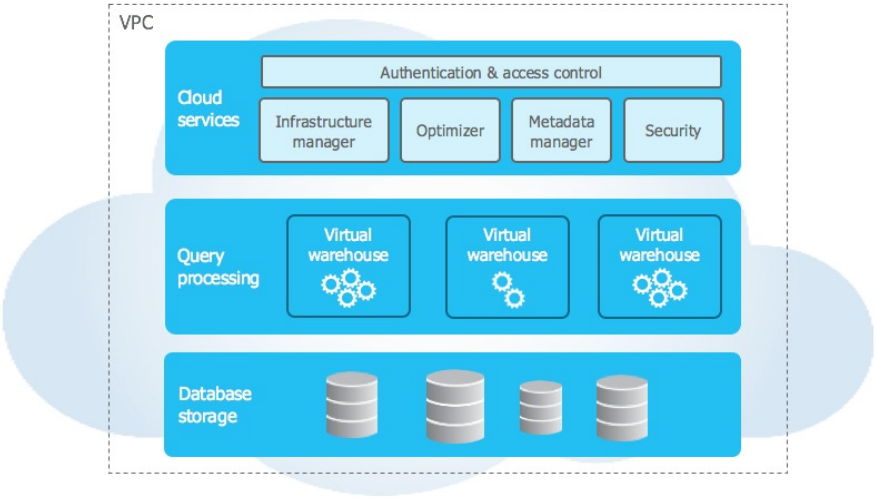

Figura 4.1: Arquitectura de Snowflake.

dbt

[16] Es una herramienta de línea de comandos que permite desarrollar colaborativamente código de analítica. Es la herramienta que se encarga de las trasformaciones, la T del ELT, siguiendo las mejores prácticas de la ingeniería del software como la modularidad, portabilidad, la integración y distribución continua, la documentación, el control de versiones, y el testeo de cada modelo, permitiendo construir data pipelines robustos.

En su flujo de trabajo para las transformaciones se generan automáticamente grafos de dependencia y su ejecución se puede programar y automatizar en su entorno en la nube.

Dbt depende de SQL para transformar los datos, no le queda más remedio ya que, al estar transformando los datos directamente en la data warehouse en la nube, está delegando todo el trabajo pesado en su motor. Aun así, dbt agrega funcionalidad de código como por ejemplo bucles, que no hay en SQL, gracias al template jinja, lo que permite escribir código más eficiente.

Fivetran

[31] Es una herramienta que permite a las compañías extraer datos de varias fuentes, y cargarlas en uno distino, normalemnte una data warehouse en la nube.

Estas tareas de integración están automatizadas y usan conectores de datos preconstruidos que hay que configurar para que se conecten a las fuentes y el destino de los datos.

Amazon S3

[30] Conocido también como *Amazon Simple Storage Service* es un servicio que ofrece el almacenamiento de elementos a partir de una interfaz de servicio web.

Se puede almacenar cualquier tipo de objeto, por lo que se le puede dar varios usos, pero en este caso será usado como un data lake.

Python

[27] Es un lenguaje de alto nivel de programación que permite el desarrollo de cualquier tipo de aplicación.

Lo que permite diferenciar este lenguaje de otros es que se considera un lenguaje interpretado, de forma que para la ejecución de las aplicaciones no se efectúa la compilación, sino que se ejecutan mediante un programa interpretador.

Os

[15] Es una biblioteca de Python que proporciona distintas funcionalidades sujetas al Sistema operativo.

Entre estas funcionalidades se incluyen: la creación de carpetas, el registro de contenidos de las carpetas, entre otras.

Requests

[38] Es una biblioteca de Python que se encarga de facilitar a los usuarios las solicitudes HTTP.

Boto3

[29] Es una biblioteca de Python que proporciona la agregación de servicios AWS, de manera que facilita al usuario la creación, el uso y la configuración de los servicios.

Google cloud

[34] Es un espacio virtual por el que se puede realizar una serie de tareas que antes requerían de hardware o software y que ahora usan la nube de Google como única forma de acceso, almacenamiento y gestión de datos.

Cloud functions

[10] Es un servicio que sirve para crear aplicaciones sin servidores dentro de Google Cloud, dando respuesta a la demanda de eventos que puedan ocurrir en cualquier sitio. Lo positivo de este servicio es que abonarás solo por lo que uses, osea el tiempo que tu código se esté ejecutando, por lo tanto es buena opción para proyectos pequeños. En este caso lo hemos utilizado para realizar un script de Phyton [27] que extrae desde URLs de fuentes oficiales del estado, .csv con los datos que queremos actulizar diariamente de la pandemia, para llevarlos a Amazon S3, nuestro data lake, dónde reunimos los datos antes de cargarlos en nuestra data warehouse.

Cloud sheduler

[11] Es un programador de trabajos cron administrado. Permite programar practiamente cualquier trabajo, desde trabajos por lotes y de macrodatos hasta operaciones de infraestructura de nube y mucho más. Es el servicio que usamos para que se ejecute nuestro scripts de phyton de forma diaría.

Pub|sub

[12] Este servicio permite que otros servicios se comuniquen de forma asíncrona, se usa para canalizaciones de integración de datos y estadísticas de transmisión a fin de transferir y distribuir datos. Es igual de efectivo que un middleware orientado a la mensajería para la integración de servicio o como una cola a fin de paralelizar tareas. Lo usamos para que se pueda comunicar Cloud Scheduler con el script de python programado en Cloud functions.

4.3. Documentación

LaTeX

LaTeX [35] es un sistema de elaboración de textos determinado para la creación de documentos escritos compuestos por una elevada condición tipográfica.

Overleaf

Overleaf [36] es un editor colaborador con LaTeX constituido en la nube que permite escribir, editar y publicar documentos científicos, facilitando su uso sin necesidad de instalaciones y permitiendo la cooperación múltiple en tiempo real.

Zotero

Zotero [23] es un programa de *software* libre encargado de gestionar, tanto las referencias bibliográficas, como las citas bibliográficas, de forma que permite fácilmente la recuperación y organización de las referencias.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto. Incluye la exposición del ciclo de vida utilizado, y los detalles de mayor relevancia de las fases de análisis, diseño e implementación.

5.1. Inicio del proyecto

El tema del proyecto surgió gracias a una charla que asistí en mi trabajo. En ella hablaron sobre un nuevo paradigma, a la hora de montar la arquitectura que da soporte al ciclo de vida de los datos, el Modern Data Stack, y de cómo las empresas españolas están empezando a cambiar su antigua arquitectura on premise, por esta nueva en la nube, gracias a las varias ventajas que conlleva su implementación.

Como no había tenido todavía la oportunidad de trabajar en un proyecto que utilizase estas nuevas tecnologías me pareció una buena idea aprender a usarlas, ya que me parecen interesantes y útiles de cara al futuro.

Por esas fechas, a principios de este año 2022, estábamos batiendo récords de incidencia de contagios, por lo tanto, pensé que podría ser muy útil tener un cuadro de mandos donde se tengan centralizados los datos más importantes sobre la pandemia, de forma que se pueda acceder de forma fácil e interactuar con los datos.

Juntando ambos factores, decidí que iba a montar una arquitectura para el ciclo de vida de los datos del COVID-19, utilizando este nuevo paradigma, el Modern Data Stack. Ya que se trata de un tema de actualidad de gran

interés social, y de una tecnología en pleno auge, que es presente y futuro de las arquitecturas de datos.

5.2. Metodologías

Con respecto a la gestión del proyecto se ha utilizado una metodología ágil, precisamente SCRUM, sobre la cual se trabajó en la asignatura gestión de proyectos. Aunque por razones obvias, como que esta metodología de trabajo se aplica para trabajo en equipo, y este trabajo ha sido desarrollado por una sola persona, no se ha podido aplicar por completo, pero si se ha intentado aplicar la filosofía ágil, dentro de los posible.

La división del desarrollo estaba marcada por sprints, estimando mediante Story points la carga de trabajo de cada uno de ellos, la mayoría de ellos han tenido una duración de una semana, excepto el primero y el último, cuya duración ha sido de solo 1 semana. Para llevar a cabo está gestión se ha hecho uso de una herramienta llamada Zenhub, ya que gracias a ella se ha podido usar un tablero canvas para estimar y priorizar las tareas.

Para dar finalizado el sprint, se realizaban las siguientes tareas:

- Revisión del sprint: en esta tarea se revisaban las distintas tareas correspondiente a cada sprint y se miraba si se habían completado los objetivos del sprint. También se analizaban los problemas que ha surgido, para aprender de ellos y que no supongan inconvenientes en los próximos sprints.
- Planificación siguiente sprint: posteriormente, y una vez finalizada la revisión del sprint, se planifica el sprint que va a continuación, se marcan las tareas y objetivos de éste y mediante los story points previamente mencionados se hace la estimación del esfuerzo que va a conllevar cada tarea.

5.3. Formación

Para la realización del proyecto se ha necesitado la investigación sobre varias herramientas de las cuales no tenía conocimiento. A continuación, se va a dar una breve explicación sobre cada una de ellas.

Snowflake

Para el almacenamiento de los datos se ha utilizado Snowflake, un data warehouse analítico en la nube.

Para aprender a usarlo, se ha realizado un curso estimado en unas 10 horas, que ofrece de manera gratuita el propio Snowflake, con el que aprendes a usar las principales características de Snowflake, en el cual recibes una insignia tras finalizarlo.

dbt

Para la transformación de datos se ha utilizado dbt, una herramienta que permite transformar los datos en sus datawarehouses de forma eficaz.

Al igual que en el apartado anterior, se han realizado varios cursos que ofrece de forma gratuita la propia herramienta, uno con los fundamentos principales, de duración de 5 horas que al finalizar te da una insignia, y también varios cursillos más avanzados, de duración total de 10 horas entre todos.

PowerBI

Para realizar el cuadro de mandos y los gráficos, que es lo que verá el usuario se ha utilizado PowerBI.

Yo ya había trabajado con otra herramienta de visualización de datos, pero como nunca había trabajado con PowerBI, tuve que estudiarme su documentación y leer varios artículos para aprenderme sus características únicas y poder usarla sin problemas.

5.4. Analisis

La primera etapa, consistió en un análisis de los requisitos del proyecto, que, en este caso, al ser un proyecto centrado en los datos y su análisis, coincide con los KPI que queremos medir.

Una vez tenía los KPI, tuve que buscar los datos necesarios para calcular los KPI en los datos abiertos que ponen a disposición las distintas instituciones del estado. Tuve que descartar algún KPI, como la incidencia acumulada en 14 días en las residencias, ya que no había datos con granularidad diaria, y tuve que conformarme con unos de granularidad semanal, donde los cálculos serían más limitados. También tuve que descartar otros KPI, como el número de contagios dependiendo del ámbito (social, familiar, laboral, centro educativo, etc.), porque no encontré en los datos abiertos información sobre ellos.

5.5. Diseño

Una parte importante del trabajo fue el diseño, en concreto el modelado de los datos. Sabiendo que se trata de un proyecto de analítica de datos, tuve bastante claro desde un principio, que iba a usar la tecnología OLAP ya que es mucho más rápida que su contraposición OLTP, a la hora de ejecutar secuencias SQL de tipo SELECT, y que iba a usar la metodología kimball para el diseño del data warehouse, aplicando un modelado en estrella, utilizando este modelado en vez de uno de copo de nieve. Ya que este, al estar más normalizado, aunque disminuye lo que ocupa el almacenamiento, aumenta el número de joins que se tiene que hacer en las consultas y por ello, el tiempo de respuesta.

Así que el siguiente paso, fue realizar el modelado de datos, siguiendo las reglas previamente mencionadas. Teniendo en mente los KPI que queríamos calcular, y debido a que estos se dividían en 3 grandes grupos (casos, hospitales, residencias), se realizó un modelado por separado de ellos, aunque luego en PowerBI se juntará todo en el mismo modelo sin problema, a través de sus tablas comunes. Para cada una de estas divisiones se realizó un modelo conceptual, lógico y físico.

Esta ha sido una de las partes más demandantes del proyecto, ya que a diferencia del modelado en tercera forma normal de las bases de datos convencionales, dónde se busca optimizar el almacenamiento y se prohíbe cualquier redundancia en los datos, cuando diseñas una data warehouse tienes que encontrar un balance entre optimizar el almacenamiento, no desnormalizando excesivamente las tablas, y el tiempo de respuesta de las consultas, aunque ahora gracias a las data warehouse en la nube, va perdiendo importancia la optimización de almacenamiento.

Resolución de problemas

Hubo varios problemas debidos a la complejidad de los modelos, ya que había tablas con diferente granularidad que se necesitaban en el cálculo de un mismo KPI (Para calcular la IA a 14 días cada 100.000 habitantes, se necesitaban la tabla de casos a nivel diario, y la tabla de población que viene a nivel anual), tablas con distinta granularidad que se relacionaban con la misma dimensión (la tabla de casos se relacionaba con la dimensión calendario con granularidad diaria, la tabla de población se necesitaba relacionar con la dimensión de calendario con granularidad anual, y la tabla de residencias que también se necesitaba relacionar con la dimensión calendario con granularidad semanal), y según donde mirases había partidarios de aplicar distintos tipos de soluciones. Por lo tanto, dediqué bastante tiempo diseñando el modelo de

datos, ya que se me ocurrían varias posibilidades y no estaba seguro de cual era mejor para mi caso.

Para resolver los problemas de granularidad opté por distintas soluciones dependiendo de la situación. La solución fácil para los problemas de granularidad, es agregar los datos hasta llegar al nivel mínimo de granularidad que tienen estás tablas que se relacionan entre sí. En el caso que he comentado en el párrafo anterior relacionado con la dimensión calendario, sería a nivel de año, pero así perdiríamos mucho poder de análisis, y en casos como el mío, donde tenemos menos de 3 años de datos no tiene ningún sentido. Otra solución que se suele plantear, es que se cree una tabla para cada granularidad necesitada, en este caso, una tabla de calendario con granularidad en años, otra en semanas y otra en días. Esta solución tiene el inconveniente de que repite datos, por lo que aumenta, lo que ocupa el almacenamiento de los mismos, pero podría ser una solución viable en nuestro caso de la tabla de residencias, que tiene granularidad semanal.

No obstante, para la tabla de residencias opté por otra solución que me parece igual de buena, pero sin el inconveniente de repetir datos y lo que esto conlleva. La solución fue crear una fecha ficticia para semana, que contenía el primer día de esa semana en la tabla de residencias. Ahora tenemos dos tablas con la misma granularidad que podemos relacionar sin problemas, y como la tabla residencias no va a realizar ningún calculo en días, ni va a estar relacionada en ningún KPI con otra tabla de hechos con granularidad de días (esto sí pasa con la tabla de población), está fecha ficticia va a estar oculta para el usuario y no va a provocar ningún problema en los cálculos. Este tipo de solución se aplicará también en esta tabla de residencias con respecto a la dimensión de provincias, ya que el resto de las tablas se relacionan con granularidad de provincia, pero la tabla residencias tiene granularidad de comunidad autónoma. Al igual que se creó una fecha ficticia para residencias, se va a crear una provincia ficticia que va a representar la comunidad autónoma, que en este caso será la capital, y ella será la que relacione la tabla con la dimensión provincias.

Pero esta solución que hemos dado para provincias, no sirve para resolver el problema de granularidad de población, ya que en este caso al compartir KPI con tablas que tienen granularidad de día, como la de casos, cuando filtremos un día especifico y queramos saber la IA a 14 días cada 100.000 habitantes, si seleccionamos por ejemplo el 5 de mayo, y tenemos asignada la fecha ficticia de población de cada año, al 1 de enero, cuando vaya a calcular la población para el cálculo de ese KPI lo dividirá por 0 y dará un cálculo erróneo. Tampoco podemos utilizar la posible solución que comenté antes de hacer una tabla

calendario para la granularidad de año, ya que entonces necesitaríamos duplicar los filtros en el cuadro de mandos, porque tenemos KPIs que utilizan ambas tablas, y por ellos necesitamos que tengan el mismo filtrado.

Por tanto, la única solución viable que encontré, fue que la dimensión calendario tuviera un campo año y que se relacionase con él la tabla población. En primera instancia esto no parece recomendable (ya que se salta dos recomendaciones típicas de modelado, como que no hay que unir tablas con distintas granularidad, o que hay que evitar las relaciones muchos a muchos entre una dimensión y una tabla de hechos) porque puede provocar resultados inesperados. Pero en nuestro caso es la mejor solución, lo que va a ocasionar es que cuando se filtre por un día del año, va a devolver el valor de ese año para todos los días de ese año, y en este caso, es justo lo que queremos, al tener datos anuales de la población, debido a que la población de 2022 es la que necesitamos en cualquier día de ese año. Un resultado inesperado podría darse en otros casos, por ejemplo, cuando tienes las ventas de un año, ahí no quieres que las ventas de un día sean las correspondientes al año completo, que es el sumatorio de las ventas de los días de un año , cosa que no pasa en nuestro caso con la población.

El mayor contratiempo del desarrollo de este proyecto se dio por un fallo en el modelado de datos, ya que se trató la población como una dimensión, que contenía los campos de grupo de edad, género, provincia y año, todos comunes con la tabla de hecho de casos (menos el campo año, pero que si estaba contenido en el campo fecha de la tabla casos), y se relacionó ambas tablas entre si mediante un campo compuesto por esos campos. Lo realicé de esa forma en un principio, porque no me parecía que población fuese una tabla de hechos, ya que su información, no era de ninguna utilidad para mis análisis si no iba de la mano de la tabla de casos.

Pero más tarde en el desarrollo, me di cuenta de este error, porque que cuando realizaba filtros muy restrictivos, los cálculos empezaban a ser imprecisos en los KPI que utilizaban ambas tabla. Esto se debía, a que ambas tablas se estaban filtrando entre sí en situaciones no deseadas, por ejemplo si restringías el cálculo a un día en concreto, el cálculo de la IA a 14 días cada 100.000 habitantes se realizaba solo sobre la porción población que había tenido al menos un caso ese día, cuando el cálculo debía realizarse sobre todo el conjunto de la población que cumpliese las condiciones filtrado, independientemente de si esa fracción de población tenía algún caso en la tabla de casos, o no. Del modo que estaba diseñado el modelo de datos, no se podía evitar este problema, ya que se necesitaba que la tabla de población filtrase el grupo de edad y genero

para la tabla de casos y que la tabla casos, filtrase a través de la dimensión calendario y provincia, a la tabla de población.

Entonces tuve que cambiar el modelado, hice que la población fuese otra tabla de hecho, en vez una dimensión, la cual se relacionaba directamente con las dimensiones calendario y provincia. Además de una nueva llamada demografía que iba a contener los campos de género y grupo de edad, que contenía anteriormente población. Ahora las tablas de casos y población no estaban conectadas entre sí, pero al estar conectadas a las mismas dimensiones que se quieren filtrar, se va realizar el cálculo correctamente. Viéndolo ahora con perspectiva, me parece bastante claro que población debe ser una tabla de hechos, pero cuando realicé el modelado me surgieron dudas y terminé eligiendo una opción incorrecta que me hizo perder bastante tiempo, ya que conllevó modificar mucho código en varios scripts y crear unos cuantos scripts nuevos.

Por último, tras tener diseñado el modelo de datos, hice un diseño de unos prototipos de como quería que quedasen las hojas y los gráficos del cuadro de mando, que más tarde tendría que desarrollar en PowerBI.

5.6. Desarrollo

Tras terminar la fase de diseño, prosegí con la fase de desarrollo, la cual consistió en varias fases:

Preparación de Snowflake

Lo primero que se hice fue preparar manualmente el data warehouse, Snowflake, donde se iban a ingresar los datos, se crearon las tablas y se ingresaron los datos de forma manual, aunque el objetivo final es que se ingresen los datos que se actualizan frecuentemente de manera automática, decidí que esto se haría al final del proyecto, y que al principio se harían tareas más prioritarias, y me conformaré al principio con está ingesta manual para tener una base de la que partir

Configuración dbt

El siguiente paso fue la configuración de dbt, donde tuve que conectarlo con Snowflake para poder transformar los datos, también se conectó con mi repositorio de Github para llevar a cabo el control de versiones. A parte configuré un entorno de desarrollo donde realizar los cambios en el código, que venía acompañado de su propia rama de Github y que irá creando los distintos

modelos en una base de datos nueva en Snowflake, diferenciada de la base de datos que contiene los datos crudos, sin transformaciones.

Transformaciones con dbt

Este fue otra de las partes más importantes del trabajo, donde invertí bastante tiempo a crear una veintena de scripts organizados en carpetas a distintos niveles, que realizan las innumerables transformaciones que se realizan para poder convertir los datos crudos que recibo en el modelado que he diseñado anteriormente, de forma que esté listo para realizar los cálculos requeridos por los KPIs de forma eficiente. Cada script realiza distintas transformaciones, pero están organizados en 3 capas:

- Staging: en ella se cargan los datos desde la fuente, que en mi caso es Snowflake y se renombran los campos a nombres comunes y entendibles. Ésta tiene varias carpetas en las que está organizada, una para cada tabla de hechos que son cargadas desde una fuente externa (casos, residencias, hospitales y población). A parte, cada una de estas carpetas tiene un fichero de configuración .yml para las fuentes de datos y otro para los scripts dónde se les puede documentar y testear.
- Temporal: en ella se realizan la mayoría de las transformaciones necesarias para obtener los modelos que hemos diseñado y por tanto tiene la mayoría del código. Al igual que la capa de Staging está organizada en varias carpetas, una para cada tabla de hechos y otra llamada máster para las dimensiones comunes que comparten varias tablas de hecho (calendario, demografía y provincias). Al igual que la capa de staging tiene un fichero de configuración por cada carpeta dónde documentar y testear los scripts.
- Core: en ella es dónde está la versión final de los modelos, y coincide exactamente con lo diseñado en el modelado de datos, en sus scripts se hacen pequeñas modificaciones finales, como joins o renombrar los campos de las tablas, para que estén preparados para su explotación en la herramienta de reporting, PowerBI. Tiene la misma organización en carpetas que la capa anterior, en tablas de hecho, y la carpeta máster. Como sucedía en las dos capas anteriores esta capa tiene también para documentar y testear los scripts de cada carpeta.

```
{{
    config(
        unique_key="ID_POBLACION"
    )
}}

-- depends_on: {{ ref('dim_calendario') }}

with fechas as(
    SELECT
        0 as min_fecha,
        1 as max_fecha
)

{%- call statement('fechas', fetch_result=True) -%}

    SELECT
        MIN("AÑO"),
        MAX("AÑO")
    FROM {{ ref('dim_calendario')}}

{%- endcall -%}

{%- set min_fecha = load_result('fechas')['data'][0][0] -%}

{%- set max_fecha = load_result('fechas')['data'][0][1] -%}

select
    LEFT(PROVINCIAS,2)::integer AS ID_PROVINCIA,
    FECHA,
    SUM(TOTAL) AS TOTAL,
    LEFT(SEXO,1)||case
        when SPLIT_PART(EDAD,' ',1)= '0-4' or SPLIT_PART(EDAD,' ',1)='5-9' then '0-9'
        {% for number in range(1,8) %}
        when SPLIT_PART(EDAD,' ',1)= '{{number}}0-{{number}}4' or SPLIT_PART(EDAD,' ',1)='{{number}}5-{{number}}9' then '{{number}}0-{{number}}9'
        {% endfor %}
        else '80+'
    end AS ID_DEMOGRAFIA,
    ID_PROVINCIA||ID_DEMOGRAFIA||FECHA AS ID_POBLACION
from {{ ref('stg_fact_poblacion')}}
where provincias <> 'TOTAL ESPAÃ‘A' and EDAD <> 'TOTAL EDADES'
and procedencia ='TOTAL' and SEXO <> 'Ambos sexos'
and FECHA>={{ min_fecha }} and FECHA<={{ max_fecha }}GROUP BY ID_DEMOGRAFIA, ID_PROVINCIA, FECHA
```

Figura 5.1: Transformación modelo tmp-fact-poblacion.

Testeo del código con dbt

Una vez terminado el código de las transformaciones, el siguiente paso es testear el código, tarea que se puede con el propio dbt, con los ficheros de configuración comentados en el apartado anterior, con un comando dbt test podemos correr los más de 40 tests que se he preparado, para asegurarme que las transformaciones se han hecho como esperaba y no hay ningún error. Los test más utilizados han sido comprobar que un campo no tenga ningún valor null o que la clave primaria de una tabla no se repita.

Documentación del código con dbt

Después del testeo y tener seguro que el código funciona bien, lo que hice fue documentar el código, al igual que con el testeo. Está tarea fue facilitada también por los ficheros de configuración .yml, dónde podemos poner descripciones en cada modelo, y en cada campo del modelo que queramos, con el comando dbt docs generate se forma una documentación, que aparte de contener las descripciones incluidas en el fichero de documentación, contiene el código dbt de cada modelo y su equivalencia en SQL puro. También contiene las dependencias de cada modelo con el resto, y varios gráficos DAG (Grafo Acíclico Dirigido),

uno por cada modelo ,mostrando sus dependencias y otro general, para todo el proyecto.

Figura 5.2: Gráfico DAG de fact-residencias.

Creación del cuadro de mando en PowerBI

Tras tener todas las transformaciones listas, lo siguiente fue conectar PowerBI a Snowflake para poder acceder a los modelos transformados, una vez conectados hay que relacionarlos entre sí, indicar la cardinalidad y la dirección de filtrado entre ellos.

Una vez tengo todos los modelos conectados en un único modelo, procedo a crear los gráficos, filtros y paneles, según lo tengo diseñado en el prototipo. Durante esta fase encuentro un comportamiento inesperado con algún KPI, el cual describo en detalle en la fase de diseño.

Una vez terminado, subo el cuadro de mandos a la nube para que pueda ser compartido mediante un enlace y ser accedido por el público.

Figura 5.3: Información general del cuadro de mandos.

Automatización de la extracción y carga de datos

Los datos abiertos que he utilizado se encuentran disponibles para su descarga a través de un url en las páginas públicas de distintas instituciones del estado en forma de .csv, estos son los que se actualizan frecuentemente y los que nos va a interesar tener actualizados, al contrario que otros como la población o las provincias, que o no cambian, o se actualizan cada año.

Para automatizar la extracción y carga de estos datos en principio pensé en utilizar una herramienta que pudiera extraer los datos desde la url de descarga y cargarla en Snowflake, pero después de hacer una búsqueda exhaustiva no encontré ninguna herramienta que fuera capaz de hacer esto, ya que no tenían ningún conector (en muchos casos entre más de sus 200 conectores disponibles) preparado para extraer datos desde una url, cosa que me pareció extraña, pero entendible ya que normalmente en una arquitectura de datos, los datos se extraen de bases de datos OLTP, servicios de almacenamientos de objetos (como Amazon S3 de Amazon Web Services), CRMs, ficheros locales, SAP, aplicaciones como LinkedIn, Twitter, Instagram, etc., pero no me imagino a muchas empresas cargando sus datos desde una url.

Como no iba a ser posible hacerlo de la manera que había pensado, lo que se me ocurrió es que primero, desde un script programado en Python iba a extraer los archivos desde la url y llevarlos a un Data Lake montado en Amazon s3, que se ejecutase de forma diaria, y de esa manera iba a almacenar en Amazon s3 los datos actualizados. Luego con una herramienta como Fivetran iba a extraer

los datos de Amazon s3 y cargarlos en las tablas de Snowflakes de datos crudos correspondientes. Para ello tuve que configurar Amazon s3 para dar un acceso a Fivetran y que pudiera acceder a los datos, creando un IAM, que es un servicio web que ayuda a controlar de forma segura el acceso a los recursos de AWS. Para poder llevarlo a cabo, cree una política y un rol exclusivo para Fivetran, por último, tuve que configurar en Fivetran el acceso al S3 que acaba de crear y también el acceso a Snowflake para que pudiese cargar los datos. Fivetran se configuró para que realizase este proceso cada 24 horas.

Este proceso se pudo completar sin problema para los datos de los casos, pero presentó problemas para las tablas de hospitales y residencias. Para los casos no hubo ningún problema ya que el url desde donde se descargaba el .csv era siempre el mismo. Para residencias había inconvenientes ya que el nombre cambiaba, e incluía la fecha de la última actualización del archivo. Pero se pudo modificar el script de Python para que incluyera en la url la fecha actual, y que en el caso de que no pudiera cargar el archivo porque en esa fecha no se había actualizado, tampoco lo hiciese en Amazon s3. El mayor problema vino con los datos de residencias, ya que la url cambiaba cada vez que se actualizaba, con un número aleatorio del que no se puede identificar ningún patrón, por lo que no podía saber cuál sería la url de descarga del archivo cuando lo actualizasen. Por tanto, no podía automatizar su extracción, e intenté contactar con el IMSERSO, para ver si podían decirme algo acerca del nombre del archivo que se usa en la url, pero no tuve respuesta. De este modo, los datos de las residencias se quedaron sin extracción automática y se tenían que extraer y cargar en Amazon S3 de forma manual, para que Fivetran pudiese hacer la carga automática en Snowflake. Además, la ejecución de este script de Phyton está tambien automatizado, gracias a los servicios de Google Cloud, que hemos descrito en el apartado de técnicas y herramientas.

Trabajos relacionados

Este apartado es un estado del arte. Incluye un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Al empezar con el TFG busqué trabajos relacionados con el mío para ver cómo estaba el estado del arte, pero como la mayoría de los trabajos de este tipo los hacen empresas con datos sensibles no pude encontrar apenas. Lo que sí encontré fueron TFGs de otras universidades públicos, los cuales me sirvieron para poder compararlos con lo que yo tenía intención de hacer.

6.1. Análisis y visualización de datos abiertos de carácter informativo para el alumnado de la Universidad de Valladolid

Se trata de un trabajo de fin de grado realizado por Nadia Pérez Faúndez en el año 2021 [19].

Su proyecto se centra en la disposición de datos abiertos con el fin de informar a los estudiantes universitarios a través de una aplicación.

Aunque el proceso de desarrollo no es el mismo que el planteado en este proyecto, ya que usa la arquitectura tradicional con ETL, se puede ver como el objetivo final se asemeja, ya que también pretende la extracción de datos fiables de fuentes oficiales, en este caso de distintas universidades públicas de Castilla y León, con la finalidad de proporcionar la visualización de la información obtenida facilitando al usuario el acceso e interacción con los datos.

6.2. Aplicación de herramientas de Business Intelligence en datos del entorno de Salud

Se trata de un trabajo de fin de grado realizado por Uxue Ayechu Abendaño en el año 2017 [6].

Su proyecto se centra en la explotación de la información sanitaria de Navarra a través del Business Intelligence con el objetivo de ayudar a los sanitarios a partir de los datos obtendidos, con su visualización a través de la herramienta Tableau.

En su caso, se usa la arquitectura tradicional con ETL, a diferencia de este proyecto, en cuyo proceso de desarrollo usa el Modern Data Stack, estando todo almacenado en la nube y usando los procesos ELT.

6.3. Business Intelligence y el análisis predictivo: COVID 19

Se trata de un trabajo de fin de grado realizado por Javier López Navas en el año 2020 [25].

Su proyecto se centra en elaborar un estudio de Business Intelligence del Covid-19 a través del uso de la herramienta Power Bi de Microsoft, mediante la interpretación de datos.

En este caso se emplea la misma herramienta para la visualización de la información, aunque la principal diferencia es en el proceso de desarrollo, ya que en este proyecto se usa la arquitectura tradicional con ETL y en el proceso de desarrollo de este proyecto se usa el Modern Data Stack, estando todo almacenado en la nube y usando los procesos ELT.

6.4. Mejoras de mi proyecto con respecto a los analizados en este apartado

Gracias a utilizar este nuevo paradigma de arquitectura de datos, mi proyecto tiene unas ventajas que no tienen proyectos anteriores:

- Mayor velocidad de procesamiento en los datos.
- No necesitar instalaciones al usar servicios en la nube.

- Implementar practicas de Ingeniería del sofware gracias a dbt como: modularidad, portabilidad, integación, distribución continua, documentación, control de versiones y testeo.
- Actualización diaria automática de los datos en todas las fases de nuestra arquitectura, con su correspondiente testeo.
- Gracias al enfoque ELT en lugar de ETL obtenemos menos latencia, mayor escalabilidad y menor procesamiento.

Conclusiones y Líneas de trabajo futuras

Conclusiones derivadas del desarrollo del proyecto.

7.1. Conclusiones

Una vez finalizado el proyecto, se pueden considerar una serie de conclusiones basadas en su evolución:

- Se ha podido comprobar que los objetivos generales se han cumplido de manera satisfactoria, ya que finalmente se ha logrado el desarrollo de una arquitectura de datos que recopila la información de las principales fuentes oficiales sobre el estado de la pandemia en España, mediante su extracción, carga y transformación en la nube. Y, además, de disponer toda la información recopilada almacenada en la nube, se ha facilitado al usuario la interacción con los datos.
- El proyecto ha requerido el uso de nuevos conocimientos sobre distintos conceptos y diversas herramientas. Respecto a los nuevos conceptos aprendidos puedo considerar que, en los que más he tenido que profundizar, son aquellos conceptos relacionados con el nuevo paradigma *modern data stack*, entre los que se incluyen ELT y *analytics engineer*. El aprendizaje de todos estos conceptos ha resultado muy útil debido al aumento de uso de esta nueva arquitectura que están teniendo las empresas españolas, gracias a las ventajas que ofrece.
- La evolución del proyecto ha sido controlada a través del desarrollo ágil de Scrum a partir de GitHub, lo que ha facilitado la organización y

aceleración de la entrega final del proyecto. Gracias a la elaboración del proyecto en sprints se ha podido observar el incremento de valor y, a su vez corregir los errores localizados en sprints anteriores.

7.2. Líneas de trabajo futuras

Algunas de las líneas de trabajo futuras que se podrían considerar son:

- Conseguir una automatización completa en la extracción de datos.
- Implementar algún análisis predictivo, mediante técnicas estadísticas de modelización, aprendizaje automático y minería de datos.
- Conseguir una versión premium de PowerBI para poder hacer público la URL del cuadro de mandos.
- Añadir más gráficos para la visualización de datos con nuevos KPIs en la herramienta empleada *Microsoft Power BI*.

Bibliografía

[1] Comenzar con github desktop. https://docs.github.com/es/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop, May 2022.

[2] Diferencias entre OLTP y OLAP. https://pc-solucion.es/2018/04/23/diferencias-entre-oltp-y-olap/#:~:text=OLAP%20es%20un%20sistema%20en,de%20base%20de%20datos%20online, April 2022.

[3] ¿qué es un kpi y para qué sirve? https://www.isdi.education/es/blog/que-es-un-kpi-y-para-que-sirve, April 2022.

[4] ¿qué son los procesos etl? https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/qu-son-los-procesos-etl, May 2022.

[5] Mike A. Data warehouse: ¿qué es y cómo utilizarlo? https://datascientest.com/es/data-warehouse-que-es-y-como-utilizarlo, April 2022.

[6] Uxue Ayechu Abendaño. Aplicación de herramientas de business intelligence en datos del entorno de salud. https://academica-e.unavarra.es/xmlui/bitstream/handle/2454/24611/TFG-Uxue%20Ayechu%20Abendano.pdf?sequence=1&isAllowed=y, March 2022.

[7] Daniel Alcón. ¿qué es snowflake? https://www.paradigmadigital.com/dev/que-es-snowflake/, May 2022.

[8] Gustavo B. ¿qué es github y cómo usarlo? https://www.hostinger.es/tutoriales/que-es-github, May 2022.

[9] CLAIRE CARROLL. *What is analytics engineering?* https://www.getdbt.com/what-is-analytics-engineering/, April 2022.

[10] Devoteam G Cloud. ¿qué es google cloud functions y para qué sirve? https://gcloud.devoteam.com/es/blog/que-es-google-cloud-functions-y-para-que-sirve/, June 2022.

[11] Google Cloud. Cloud scheduler. https://cloud.google.com/scheduler?hl=es-419, June 2022.

[12] Google Cloud. ¿qué es pub/sub? https://cloud.google.com/pubsub/docs/overview?hl=es-419, June 2022.

[13] Google Cloud. ¿qué es un data lake? https://cloud.google.com/learn/what-is-a-data-lake?hl=es-419, June 2022.

[14] IBERDROLA CORPORATIVA. QuÉ es business intelligence. https://www.iberdrola.com/innovacion/que-es-business-intelligence, April 2022.

[15] Covantec. Manipulación de archivos. https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion7/archivos.html#:~:text=El%20m%C3%B3dulo%20os%20de%20Python,%2C%20finalizar%20un%20proceso%2C%20etc., May 2022.

[16] James Densmore. dbt core. https://www.getdbt.com/product/what-is-dbt/, May 2022.

[17] Nuria Emilio. ¿qué es elt y cuáles son sus diferencias con etl? https://blog.bismart.com/que-es-elt-diferencias-etl, May 2022.

[18] Conexión ESAN. ¿qué es data mart y por qué es importante implementarlo en tu empresa? https://www.esan.edu.pe/conexion-esan/que-es-data-mart-y-por-que-es-importante-implementarlo-en-tu-empresa, April 2022.

[19] Nadia Pérez Faúndez. Análisis y visualización de datos abiertos de carácter informativo para el alumnado de la universidad de valladolid. https://uvadoc.uva.es/bitstream/handle/10324/50024/TFG-G5203.pdf;jsessionid=D5CF3DB92740A350ACC059CFC4CE6809?sequence=1, March 2022.

[20] IBM. Esquemas de copo de nieve. https://www.ibm.com/docs/es/ida/9.1.2?topic=schemas-snowflake, April 2022.

[21] IBM. Esquemas de estrella. https://www.ibm.com/docs/es/ida/9.1.2?topic=schemas-star, April 2022.

[22] John Kutay. Etl vs elt: Key differences and latest trends. https://www.striim.com/blog/etl-vs-elt-2/, May 2022.

[23] Librarian. Zotero, gestor de referencias bibliográficas. https://colaboratorio.net/librarian/program/2016/zotero-gestor-de-referencias-bibliograficas/, May 2022.

[24] Hlynur Magnusson. The amazing rise of the old-school modern data stack. https://medium.com/globant/the-amazing-rise-of-the-old-school-modern-data-stack-d12f3f915c51, May 2022.

[25] Javier López Navas. Business intelligence y el análisis predictivo: Covid 19. https://academica-e.unavarra.es/xmlui/bitstream/handle/2454/39107/TFG%20-%20Javier%20L%c3%b3pez%20Navas.pdf?sequence=1&isAllowed=y, March 2022.

[26] MILA PAGE. Why does it exist? https://www.getdbt.com/analytics-engineering/why/, May 2022.

[27] Santander. Python: qué es y por qué deberías aprender a utilizarlo. https://www.becas-santander.com/es/blog/python-que-es.html, May 2022.

[28] The Metabase Team. The modern data stack. https://www.metabase.com/blog/The-Modern-Data-Stack, April 2022.

[29] Unipython. Aws sdk para python (boto3). https://unipython.com/aws-sdk-para-python-boto3/#:~:text=%C2%BFQu%C3%A9%20es%20Boto3%3F,manejar%20y%20configurar%20dichos%20servicios., May 2022.

[30] Wikipedia. Amazon s3. https://en.wikipedia.org/wiki/Amazon_S3, May 2022.

[31] Wikipedia. Fivetran. https://en.wikipedia.org/wiki/Fivetran, May 2022.

[32] Wikipedia. Google cloud. https://es.wikipedia.org/wiki/Tabla_de_hechos, June 2022.

[33] Wikipedia. Google cloud. https://es.wikipedia.org/wiki/Tabla_de_dimensi%C3%B3n, June 2022.

[34] Wikipedia. Google cloud. https://es.wikipedia.org/wiki/Google_Cloud, June 2022.

[35] Wikipedia. Latex. https://es.wikipedia.org/wiki/LaTeX, May 2022.

[36] Wikipedia. Overleaf. https://en.wikipedia.org/wiki/Overleaf, May 2022.

[37] Wikipedia. Power bi. https://es.wikipedia.org/wiki/Power_BI, May 2022.

[38] Wikipedia. Requests (software). https://en.wikipedia.org/wiki/Requests_(software), May 2022.

[39] Wikipedia. Scrum (desarrollo de software). https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software), May 2022.

[40] ZenHub. Software para kanban, zenhub. https://www.getapp.es/software/110953/zenhub#:~:text=ZenHub%20es%20una%20soluci%C3%B3n%20de,de%20creaci%C3%B3n%20de%20informes%20pr%C3%A1cticos., May 2022.