Catálogo de Indicadores Gobernados en Gobierno de Navarra

Trabajo fin de máster

Máster Universitario en Ingeniería de Telecomunicación

ETS de Ingeniería Industrial, Informática y de Telecomunicación

Autor:

Andrés Santiago Díez

Tutores:

José Javier Astrain Escola

César Arriaga Egues

Convocatoria de febrero de 2024

Resumen

El Gobierno de Navarra impulsa una estrategia de gestión de datos para convertir la administración pública en un organismo que toma decisiones basadas en datos. Para ello se ha contratado *Cloud Pak for Data*, una plataforma de gobernanza e integración de datos de IBM en la que desplegar casos de uso alineados con esta misión. El primero de ellos es el Repositorio de Indicadores, un sistema de centralización y unificación de los KPI que facilita su descubrimiento y utilización. En este trabajo se describe el proceso de análisis, diseño e implementación de una solución a este caso de uso que resulta en tres productos de datos: un data mart en SQL Server, datasets publicados en un catálogo, y un glosario de negocio con las definiciones de los indicadores. Se detalla el modelo de datos diseñado y el proceso de ingeniería de datos para generar y cargar estos tres productos, principalmente dividido en una ETL y una carga a CP4D. Además, se realiza una labor de estudio de las posibilidades tecnológicas de la plataforma contratada y se describe el proceso del despliegue en producción de la solución.

1. Introducción

1.1. Interés y contexto en Gobierno de Navarra

En el Gobierno de Navarra existe un objetivo de ordenar todos los datos de los que disponen para convertir la administración pública en un organismo ejecutivo que actúe en base a datos, para llegar a 2030 siendo un gobierno digital [1]. Para ello, el Servicio de Avance Digital (de la Dirección General de Telecomunicaciones y Digitalización) puso en marcha una sección denominada la Oficina del Dato de Navarra, cuya misión es acercar a la administración a dicha meta.

Se vio la necesidad de elaborar un diccionario de datos de la administración pública. Las funciones principales de este glosario de datos son: inventariar los datos, facilitar su descubrimiento, contextualizarlos (relacionarlos entre ellos y con otros activos) y facilitar su acceso. Para ello, lo más conveniente es contratar una plataforma de gobierno del dato, ya que este tipo de plataformas incluyen diccionarios o glosarios.

Se realizó un estudio del estado del arte de plataformas de gestión de datos que incluyen herramientas de gobernanza de datos, y se lanzó a concurso. Entre las plataformas analizadas estuvieron: Anjana, Informatica e IBM *Cloud Pak for Data*. Las tres mencionadas son las que fueron probadas, pero algunas otras opciones fueron analizadas y descartadas en base a carencias que se fueron encontrando, siendo la última la escogida finalmente, y a partir de entonces referida dentro de Gobierno como 'Goberna'. Se escogió su versión *on-premise*, decisión tomada en base a la preferencia desde Gobierno de tener el software instalado en su propio CPD, y dentro de su propia red. Además, se está desarrollando un *Marketplace* que sirva como frontal del CP4D para los usuarios finales, con una interfaz más sencilla y algunas funcionalidades que no tiene la propia herramienta.

Para la implantación de proyectos de datos en esta plataforma, a mediados de 2022 se comenzó a formar un equipo llamado 'Datahub' en la empresa pública Tracasa Instrumental. Este equipo comenzó su participación desde el análisis de las herramientas comerciales de

gobierno del dato, y en la elaboración del pliego que se sacó a concurso. Desde la compra de *Cloud Pak for Data* (CP4D a partir de este momento), sus componentes han recibido formación en la herramienta y se han iniciado múltiples casos de uso para gobernar y publicar datos que sean útiles y accesibles para los usuarios de la administración pública.

En concreto, para la toma de decisiones basadas en datos, los indicadores son una de las mejores maneras de describir mediciones realizadas de una forma resumida y contextualizada. Este tipo de dato es común y se genera y utiliza de forma habitual en muchas áreas dentro de la administración pública, pero se hace de manera descoordinada, provocando que:

- Existan duplicidades o solapamientos en los análisis.
- Difieran las interpretaciones de los datos.
- Se desconozca la existencia de indicadores útiles generados por otros departamentos.

Como solución a estos problemas, se inicia un trabajo para crear un repositorio de indicadores que sea capaz de aunar y gobernar indicadores de diferentes fuentes de datos, y finalmente ponerlos a disposición de usuarios de la administración pública. Este sistema centralizará las fuentes de indicadores en el *Marketplace*, donde pueden ser encontrados, solicitados y finalmente utilizados, aportando valor a los usuarios de la administración pública. Además, compartirán un estándar de metadatos que los defina, y serán gobernados. El primer proveedor de indicadores para dicho proyecto será Nastat (Instituto de Estadística de Navarra), un organismo que genera un gran número de indicadores para el Gobierno de Navarra.

1.2. Objetivos del TFM

El objetivo principal de este TFM es materializar una solución para el repositorio de indicadores que pueda ayudar a dar un paso en la estrategia del Gobierno de Navarra de la gestión de sus datos. Para ello se comenzará describiendo los requisitos del caso de uso, y el alcance del trabajo irá desde el análisis de la viabilidad del proyecto hasta el despliegue del

sistema final, cubriendo el esquema análisis-diseño-implementación habitual en ingeniería de software.

Además, al ser uno de los primeros casos de uso iniciados en la plataforma Goberna (CP4D), como objetivo secundario se desea aprender lo máximo posible sobre la plataforma y sus posibilidades en cuanto a integración y gobernanza de datos para obtener el máximo valor posible de esta tecnología. Además, se intentarán estandarizar los métodos de trabajo para que en futuros casos de uso los desarrollos sean lo más homogéneos posible.

1.3. Metodología de gestión del caso de uso

Se realizan reuniones semanales de seguimiento del equipo Datahub con la Oficina del Dato de Navarra (ODaN), para controlar las tareas que se van realizando a nivel general. Para el seguimiento específico del caso de uso del repositorio de indicadores se programan reuniones bisemanales en las que se monitoriza el progreso y se toman decisiones sobre el diseño y desarrollo. Estas reuniones bisemanales cuentan con un representante de la ODaN, gestor del proyecto completo (de todos los casos de uso que se lanzan en CP4D), y con una representante de la Sección de Ingeniería de Datos de Avance Digital, que hace las veces de gestora del caso de uso y de *data steward* de negocio.

Algunas de las tareas a realizar se identifican conforme avanza el trabajo, otras en las propias reuniones bisemanales, donde se listan las prioridades funcionales del caso de uso para organizar de forma eficiente el tiempo dedicado. Estas tareas se compatibilizan con el resto de trabajo que surge en el equipo, principalmente de investigación y análisis de las capacidades técnicas de CP4D y de apoyo a otros casos de uso cuando es necesario.

Para la organización del trabajo en el equipo se utiliza Azure DevOps. Se utilizan principalmente 3 niveles de jerarquía: *Epics*, una por cada caso de uso, *Features*, que mapean a funcionalidades dentro de cada *Epic*, y *Stories*, que son los trabajos al nivel más específico, y que suelen ser tareas de hasta 20 horas. Además, se utilizan los *Service Spikes,* que pueden estar al nivel de las *Stories* o *Features* y que corresponden a periodos de tiempo de investigación o análisis. El último elemento utilizado son las *Indirects,* que sirven para registrar

horas invertidas en actividades no directamente productivas, como las reuniones semanales o la gestión de la metodología ágil [2].

En cuanto a la inclusión de nuevos proveedores de indicadores, en proyectos de datos es esencial comprender las necesidades del negocio, por lo que la comunicación con sus representantes es un punto esencial. Sin embargo, para el proveedor piloto, Nastat, no se establece una comunicación directa, ya que la representante de Ingeniería de Datos asume el rol de data steward de negocio. Para la inclusión de nuevos proveedores, lo esperable es una comunicación más directa con el negocio, ya sea Hacienda, Educación (proveedores que están en vista de incluirse) u otros departamentos, grupos u organismos.

Desde Tracasa, el principal responsable del desarrollo técnico y *data steward* técnico soy yo mismo, autor del TFM, tutorizado por un ingeniero de datos senior. Así, aunque en algún momento el desarrollo requiere la participación de otros componentes del equipo, la gran mayoría de la carga técnica (de análisis, diseño y de desarrollo) y de la organización del trabajo recae sobre mí.

El diseño e implementación del repositorio de indicadores no ha tenido más que dos fechas de entrega por varios factores, como la falta de experiencia con CP4D, la coordinación con el primer proveedor de datos, y la propia instalación on-prem de la plataforma, que han hecho que el desarrollo del proyecto completo fuese incierto, llegando en algún momento incluso a pararse (por ejemplo, en días en los que se estaba actualizando el software y no se podía implementar nada en CP4D). Se acordó para mediados de octubre tener una primera versión del data mart en un servidor de desarrollo de Gobierno, para que pudiesen ser accesibles en el desarrollo de cuadros de mando para presidencia.

Aparte de esta fecha, la ODaN tenía el compromiso, entre otros, de publicar antes del 1 de diciembre de 2023 diez datasets gobernados en la plataforma CP4D. Para cumplir con dicho objetivo, y al ser este el primer caso de uso que se vio con capacidad de ser desplegado en los entornos de producción, se puso esa fecha límite para poner en funcionamiento una primera versión del repositorio, a la que hace referencia este TFM.

1.4. Conceptos teóricos de base

1.4.1. Indicadores

Un indicador, comúnmente conocido como KPI (*key performance indicator*) es, en resumen, una forma de medir el rendimiento de una acción para alcanzar un cierto objetivo o la calidad de un aspecto de un negocio. Es decir, se trata de una métrica alineada con una cierta estrategia [3].

Esta es la principal diferencia entre una métrica y un indicador. Una métrica es cualquier aspecto cuantitativo o cualitativo que es medido; cuando cierta métrica resulta enriquecedora para el conocimiento del negocio acerca del progreso o resultado de una acción, se convierte en un indicador. La correcta elección de los KPIs utilizados en un negocio facilita que la toma de decisiones sea más rápida y certera, basándose en datos en vez de en sensaciones.

Un ejemplo de métrica cualitativa podría ser el estado de los tejados de las casas de un barrio. Para una empresa de reformas, esta métrica se convierte en un indicador que podría mostrar en qué barrios deberían publicitarse con más frecuencia. En general las métricas e indicadores cualitativos conllevan cierta subjetividad por la percepción de la persona que hace la valoración. Son especialmente útiles para comprender las necesidades y preferencias de los clientes de un negocio; por ejemplo, midiendo la satisfacción de los clientes después de usar un servicio web.

Las métricas cuantitativas son normalmente mediciones objetivas de la realidad que se obtienen realizando cálculos en base a datos de explotación. Un ejemplo de una métrica cuantitativa podría ser las adquisiciones de nacionalidad por parte de personas extranjeras en un territorio. Esta métrica se alinea con la estrategia o acción de la administración pública, por lo que se convierte en un KPI. Otro ejemplo de un KPI imprescindible en cualquier empresa es el ROI (*return on investment*), calculado en base a las cuentas económicas de la empresa.

Se entiende así que para que un indicador tenga valor en un negocio, sus datos son tan importantes como el contexto que lo rodea, sus *metadatos*. Por lo tanto, un indicador se

forma a partir de una métrica y unos metadatos, entre los que podrían encontrarse, por ejemplo, su definición, su fuente, su temática, su fórmula de cálculo, su cobertura temporal o la estrategia en la que se enmarca.

Además, un indicador normalmente viene desagregado. Por ejemplo, a nivel de definición y de interpretación, es diferente tener "Clientes que gastan más de 10€ en una compra" que "Clientes que gastan más de 10€ en una compra por rango de edad". En el segundo caso, para cada dato deberá especificarse el rango de edad (la desagregación) al que representa. La desagregación más común es la temporal, que facilita construir series temporales con el cambio de un indicador en el tiempo.

Estos metadatos y desagregaciones, que cambian de un indicador a otro, deben tenerse en cuenta en el diseño de un estándar que unifique indicadores de diferentes fuentes.

1.4.2. Gobernanza de datos

La gobernanza de datos se define como el conjunto de políticas y procedimientos que garantizan una gestión proactiva y efectiva de los datos corporativos. Requiere de la colaboración de distintos niveles y roles dentro de la organización para alinear los objetivos de negocio con los programas de gestión de datos [4].

En general las políticas y procedimientos se pueden enmarcar en una de estas tres categorías o principios:

- Accesibilidad de los datos. Se trata de convertir los datos en activos que sean fácilmente encontrables por los usuarios; para ello es necesario contextualizarlos correctamente, por medio de metadatos que los enmarquen de una forma unificada.
- Calidad de los datos. Imperativa para que las decisiones basadas en datos sean fiables y efectivas. Se suelen definir seis dimensiones de calidad de datos: completitud, conformidad, consistencia, precisión, singularidad e integridad [5].
- Seguridad de los datos. Es la definición de las limitaciones en el acceso a los datos: roles, permisos, mecanismos de seguridad, flujos de solicitud de acceso, anonimización, ofuscación y otros elementos para que se respeten las medidas de confidencialidad aplicables.

La gobernanza de datos requiere una colaboración estrecha entre los roles transversales de gobernanza, los equipos técnicos encargados de implementarla y los expertos de negocio que definen las políticas, reglas y procedimientos aplicables. Normalmente existe un comité de gobernanza de datos que comienza definiendo políticas a alto nivel y procedimientos de cómo aplicarlas, ya sean de calidad de datos, de control de acceso o de cualquier otro tipo. Después, junto con expertos de negocio, se definen reglas para aplicar las políticas a los negocios específicos. Por último, los equipos técnicos resuelven la implementación de estas reglas específicas a los datos físicos durante su ciclo de vida. Es de vital importancia definir y asignar roles adecuados para saber quién debe hacer qué y agilizar un proceso que, de otra manera, se puede complicar mucho. Entre los roles más comunes encontramos: el CDO (*chief data officer*), los *data owners,* los *data stewards*, los ingenieros de datos y los analistas.

1.4.3. Data Warehouse y Data Mart. Metodología Kimball.

Un Data Warehouse (almacén de datos) es una plataforma dedicada a un cierto ámbito de datos (de una organización o empresa), variante en el tiempo, integrada y no volátil, y el conjunto de medios y herramientas para obtener, procesar y gestionar los datos que lo conforman. Por ejemplo, un Data Warehouse en el que se almacenan datos curados del ámbito financiero de una empresa automovilística.

Un data mart es un subconjunto del Data Warehouse, de datos ya preparados, enfocados a un área específica del negocio y con un propósito único. Por ejemplo, un data mart dedicado al análisis de las ventas de coches de la misma empresa automovilística. En el ejemplo anterior, podría haber datos de sueldos, compras de material, alquileres de locales y mucho más, mientras que en el data mart específico sólo se analizan ventas de automóviles.

Los dos autores que más se escuchan cuando se habla de almacenes de datos suelen ser Bill Inmon y Ralph Kimball. El primero propone un enfoque top-down, en el que se define primero el Data Warehouse y después los Data Marts [6], mientras que el segundo defiende las ventajas de un diseño bottom-up [7], definiendo primero los Data Marts y después el Data Warehouse.

La principal ventaja del modelo Kimball es su menor complejidad en el diseño, que permite que un equipo más pequeño y en menor tiempo despliegue el caso de uso. La principal desventaja frente al enfoque Inmon es su mayor rigidez. Por tanto, Kimball es más adecuado para proyectos pequeños (como el repositorio de indicadores), mientras que Inmon sería mejor en proyectos de mayor envergadura, como un almacén para los datos de Hacienda.

El enfoque Kimball propone para los data marts un modelo de datos relacional en estrella. En él, una pieza central, la tabla de hechos, contiene uno o varios datos de explotación que representan un suceso. Por ejemplo, una venta en un supermercado. En esta venta, los "hechos" serían el precio total, el precio sin IVA y los descuentos aplicados. Normalmente los hechos son medidas cuantitativas sobre las que se realizan operaciones de agregación por dimensiones.

Alrededor de esa tabla de hechos se encuentran las dimensiones, que dan información contextual alrededor del hecho. Entre la tabla de hechos y las dimensiones hay claves foráneas que las relacionan, que se recomienda que sean datos de tipo entero (por eficiencia computacional) y que se denominan *surrogate keys* (claves subrogadas). Continuando el ejemplo anterior, las dimensiones podrían ser el empleado que realiza el cobro, el local en el que se realiza la compra, el momento exacto en el que sucede, y muchas más.

El modelo estrella sólo permite un nivel de relación (la tabla de hechos y sus dimensiones). Su contraparte sería un modelo "copo de nieve" (*snowflake*) en el que puede haber relaciones entre dimensiones, y que en general logran reducir la redundancia de datos, a costa de una mayor complejidad para su análisis. En la figura 1 se muestra la diferencia.

Kimball defiende el modelo estrella argumentando que las dimensiones son, en general, tablas muy pequeñas comparadas con las tablas de hechos y que por tanto, tratar de optimizarlas es irrelevante para el rendimiento de las bases de datos. Por ejemplo, una tabla de ventas puede tener millones de registros, pero la dimensión de productos sólo cientos (¿Va a tener algún impacto en el rendimiento desagregarla en "productos" - "tipo de producto" para evitar repetir unas pocas celdas varios cientos de veces?).

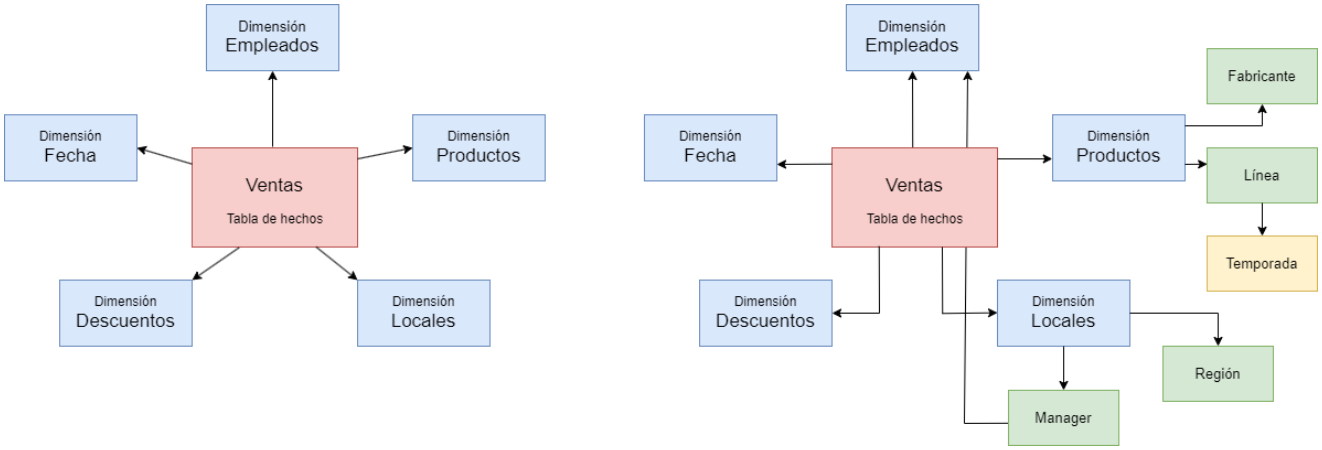

Figura 1: Diferencia entre modelo estrella y modelo copo de nieve

El proceso de extracción, transformación y carga (ETL)

Lo más habitual en almacenes de datos que terminan en data marts es encontrar esquemas que dan soporte a procesos de carga ETL. Además, es normal encontrar estos esquemas en la misma base de datos, muchas veces llamada *staging area* (almacenamiento temporal), a la que los usuarios y procesos finales no tienen acceso. A continuación se describen los tipos de esquema según su paso en la ETL junto con los procesos necesarios para generar las tablas que contienen.

Extracción:

En un esquema 'extracción' se encuentran copias exactas de las tablas originales, sin ningún tipo de modificación frente a los orígenes de datos. Estos datos en crudo aún deben ser tratados para su uso.

Transformación:

En los esquemas de 'transformación' (puede ser uno o varios, según sea necesario) se localizan los datos limpios y modelados en estrella. Es normal encontrar varias fases de transformación separadas en esquemas diferentes o en el mismo esquema. Las claves foráneas que unen dimensiones y hechos en este estado del proceso son denominadas *durable keys* (claves persistentes).

Es imprescindible que estas claves se mantengan en el tiempo, y por eso en ocasiones pueden diferir de las *natural keys* o *business keys*, que son claves utilizadas por el negocio. Por ejemplo, en un negocio puede darse una clave a un producto en función de su nombre. Ese nombre puede cambiar con el tiempo, por lo que no puede usarse como clave persistente. Estas claves en muchas ocasiones se pueden formar concatenando datos que identifiquen al elemento dimensional de forma unívoca. Para un producto podríamos utilizar "código del fabricante" concatenado a "código del producto en el origen".

Una de las claves de un almacén de datos bien diseñado es su capacidad de historificar los cambios que surgen en las dimensiones del negocio. Para ello existen las *slowly changing dimensions*, que pueden ser de varios tipos (denominadas SCD0, SCD1 y hasta 7 tipos). Habitualmente se habla de los tres primeros. El SCD0, que no permite cambios en los registros, el SCD1, que simplemente reescribe los valores almacenados, y el SCD2, que escribe un registro nuevo (una fila nueva) y almacena desde cuándo y hasta cuándo es válido dicho registro y su predecesor. Por ejemplo, si un empleado cambia su nombre de "Pablo" a "Jesús", para el análisis, al negocio le pueden interesar varias opciones:

1. Ver simplemente "Pablo" porque ya están familiarizados con el nombre.
2. Ver "Jesús" porque es su nombre actual.
3. Ver "Pablo" hasta el 2024 y a partir de entonces ver "Jesús" porque, aunque ambos nombres representan al mismo empleado, para comparar las ventas de "Pablos" vs "Jesuses" puede ser útil.

En el caso de las SCD2, es importante remarcar que la fila vieja y la nueva comparten clave persistente, pero tienen claves subrogadas diferentes. También es de relevancia saber que una dimensión puede contemplar diferentes tipos de SCD para cada campo. Por ejemplo, en la dimensión "empleados", se puede desear SCD0 para el identificador (no cambia nunca), SCD1 para el nombre (no se historifica el cambio, solo se cambia) y SCD2 para el puesto (hasta 2021 ingeniero senior y a partir de 2023 gestor de proyectos).

Aunque en este momento del proceso se utilizan las claves persistentes para relacionar las tablas, se deben generar para todas las dimensiones sus claves subrogadas, que, como ya se decía, normalmente son números enteros.

Carga:

Al final del paso (o pasos) de transformación tenemos un modelo relacional completo que está unido mediante las claves persistentes. El último paso antes de cargar el data mart final es el paso de carga. En este paso se sustituyen en la tabla de hechos las claves persistentes por las claves subrogadas, de tal manera que pasamos de tener códigos (habitualmente cadenas de texto, que con SCD2 pueden apuntar a varios registros) a identificadores de número entero (que sólo apuntan a un registro, el que sea válido en la fecha del hecho).

Por último, se copian las tablas del esquema de carga (a veces llamado precarga) al data mart, que normalmente se encuentra en otra base de datos. Podría parecer un paso redundante, pero la sustitución de las claves puede ser computacionalmente costosa, por lo que es mejor hacer el reemplazo en la base de datos *staging* para no cargar la base de datos de consulta, que puede tener usuarios o procesos activos que se verían perjudicados.

- o -

Finalmente, el data mart está cargado y preparado para ser consultado. El modelo relacional en estrella del almacén permite interpretarlo como un cubo OLAP (*Online Analytical Processing*). Estos cubos, también conocidos como hipercubos (por su número indefinido de dimensiones), permiten hacer agregaciones en los datos por medio de cortes dimensionales.

Es decir, si se tienen datos de compras y se cuenta con dimensiones como el sexo, el rango de edad y el local, estos pueden ser agregados para ver los ingresos totales por sexo, o por local y rango de edad. Esto lo hace valioso para los analistas. En el caso de Microsoft, el análisis de cubos se hace con un software llamado SSAS (SQL Server Analytical Engine), que es estándar en el Gobierno de Navarra, y permite enriquecer informes y ayuda en la toma de decisiones basadas en datos.

2. Requisitos del caso de uso

2.1. Requisitos funcionales

La implementación del repositorio de indicadores debe cubrir los siguientes requerimientos funcionales a alto nivel:

- Es capaz de federar múltiples fuentes de datos, manteniendo un estándar común de metadatos que defina los indicadores de forma unificada.
- Las definiciones (metadatos) son accesibles para todos los usuarios.
- Alberga indicadores públicos (para todos los usuarios) e internos (con restricción de acceso).
- Permite accesos automatizados, por ejemplo, para alimentar cuadros de mando en tiempo real.
- Facilita el descubrimiento de indicadores.
- Los indicadores quedan gobernados, midiendo su calidad, con control de acceso y controlando sus relaciones con activos de datos (como datasets, cuadros de mando, etc.).
- Los indicadores quedan publicados en CP4D para su compartición y uso.

Los requisitos solicitados para el repositorio de indicadores se materializan principalmente en tres productos de datos:

- Un data mart con los datos y las definiciones de los indicadores.
- Un conjunto de *datasets* publicados en CP4D (disponibles en el *Marketplace*)
- Un glosario de negocio con todas las definiciones de los indicadores y sus relaciones con los datasets.

La meta de este trabajo es diseñar e implementar un sistema que permita realizar cargas automatizadas de estos tres productos, que pueda unificar diversas fuentes de datos de indicadores y los gobierne usando la plataforma CP4D. En concreto, para la primera versión del repositorio se incluyen los datos de Nastat, que son públicos (en la primera versión no hay indicadores de uso interno). Como objetivos principales de una primera versión del repositorio de indicadores se definen:

- Diseño del modelo de datos del data mart.
- Diseño del modelo de datos en CP4D.
- Implementación del proceso de carga automatizado que genere los 3 productos mencionados.
- Inclusión de los indicadores facilitados por Nastat como proveedor piloto, al que se sumarán otros en fases futuras del caso de uso.
- Automatización del gobierno de datos aplicado en este caso de uso.
- Despliegue del proyecto en entornos de desarrollo, preproducción y producción.

En resumen, la primera fase del caso de uso, y el alcance de este trabajo, empieza con el análisis de los requerimientos y necesidades de negocio y termina con el despliegue del repositorio de los entornos de producción del Gobierno de Navarra.

2.2. Requisitos no funcionales

- En primer lugar, se debe cumplir la normativa vigente de protección de datos y seguridad de la información. La primera versión del repositorio solo incluye datos públicos de Nastat, pero en el futuro, cuando se incluyan datos de uso interno o privados, el sistema debe ser capaz de administrar el acceso para que únicamente los usuarios o cuentas de servicio con permiso puedan acceder a sus valores. Para los metadatos de indicadores no es necesario aplicar ninguna medida de seguridad.
- Además, en materia de disponibilidad, en la medida de lo posible se deben alcanzar estas premisas:
    - Ventanas diarias y horarias en que el producto/servicio estará disponible: Deberá estar disponible en horario de 8:00 a 15:00
    - Tiempo máximo de parada asumible: 3 días.
    - Número máximo de paradas asumibles por mes: 4.
- El sistema y todas sus partes deben estar instaladas *on-premise* en el CPD del Gobierno de Navarra.
- En medida de lo posible se deben utilizar las herramientas de integración de datos de

CP4D.

- La gobernanza de datos debe realizarse en CP4D.

2.3. Estado del arte

En este proyecto podemos dividir el trabajo principalmente en dos categorías: las partes relacionadas al procesamiento y carga de datos, y las relacionadas a la gobernanza de datos.

En el mercado existen múltiples herramientas que facilitan la gobernanza de datos, todas ellas con sus particularidades y similitudes. Entre los puntos comunes encontramos que la mayoría de herramientas permiten elaborar un glosario y relacionarlo a elementos internos (como *datasets* si la herramienta permite publicarlos) y externos (como un cuadro de mando en Tableau). Además, es habitual encontrar control de calidad, perfilado de datos y control de acceso de datasets. Algunos productos están dedicados exclusivamente para gobernanza del dato, como Anjana [8], y algunas otras plataformas incluyen también herramientas de integración de datos, como *Informatica* o CP4D [9].

También es común encontrar herramientas de virtualización de datos, que unifican el acceso a muchos tipos de fuentes de datos, como ficheros locales o almacenados en la nube y diferentes bases de datos. Algunas herramientas lo incorporan, y también existe software dedicado exclusivamente a virtualización, como Denodo. La virtualización permite independizar el gobierno del dato de su origen, por lo que es muy común en entornos de datos colaborativos, en los que cada grupo de trabajo almacena de forma diferente.

Para el procesamiento de datos se suele optar por, o bien herramientas gráficas como SSIS de Microsoft o DataStage de IBM, o por código, ya sea SQL y sus variantes (según el tipo de base de datos que se utilice), Python, con librerías como Pandas, Polars, SQLAlchemy, entre otras, Java o Escala. En entornos de *Big Data* es común encontrar Spark (o PySPark, su interfaz para Python), o para procesamiento en tiempo real Flink.

Para proyectos de datos en Gobierno de Navarra lo más común es utilizar SQL Server, ya que es una tecnología bien conocida, por la que se paga soporte y que, en definitiva, un estándar entre departamentos. Normalmente se utilizan procedimientos almacenados para transformaciones de datos, y se orquesta utilizando SSIS (SQL Server Integration Services).

Esta herramienta también facilita la extracción y carga de datos, y permite hacer transformaciones, pero los ingenieros de datos con experiencia normalmente son capaces de trabajar de forma más eficiente programando los procedimientos almacenados (en T-SQL).

Además, se cuenta con CP4D, que no sólo es una herramienta de gobernanza, sino que incluye Watson Studio, un servicio con varias alternativas de procesamiento de datos: *Datastage,* que se trata del software análogo a SSIS propietario de IBM, *Watson Pipelines*, para el orquestado de datos, y cuadernos Jupyter, que permiten programar en Python e incluyen acceso a un clúster de Spark (vía Pyspark) desde su entorno de ejecución.

Para análisis estadístico también existen cuadernos para R, y para el diseño y despliegue de modelos de aprendizaje automático e IA incluye AutoML y AutoAI, aunque estos tres paquetes no se utilizan para procesos de ingeniería de datos (son más propios de la ciencia de datos).

CP4D también incluye DB2, un sistema de gestión de bases de datos propietario de IBM, que también está hospedado en CP4D y que ofrece posibilidades similares a las de SQL Server, aunque utilizan diferentes técnicas de optimización, tipos de datos y dialecto SQL.

Para este caso de uso, las tecnologías principales usadas en el trabajo serán:

- SQL Server, para hospedar el data mart, que es estándar en Gobierno de Navarra y facilita la gestión de accesos, los cortafuegos, y los despliegues.
- Python para el orquestado del proceso de carga. El procesamiento se va a ejecutar en CP4D, por lo que debe ser bien de esta manera, o usando Watson Pipelines. Este es un software de orquestación con el que no se tiene experiencia, y en el que después de probarse se encontró alguna carencia y una complejidad mayor a la opción de código, por lo que terminó siendo descartado.
- Pandas para parte del procesamiento de datos en Python, debido a que se va a manejar un volumen de datos pequeño (no Big Data), por lo que no es necesario ni útil utilizar PySpark. Además, como el procesamiento es en CP4D, no se puede usar SSIS, y Datastage se descarta por su complejidad y porque, al estar todavía en desarrollo en su versión para CP4D, aún tiene algunos fallos que entorpecen el desarrollo.

- Procedimientos almacenados en SQL Server para parte de las transformaciones de datos, cuando sea más sencillo o eficiente que usando Pandas.
- Watson Knowledge Catalog para la gobernanza de datos, que es la mayor razón por la que se escogió CP4D.

También se hará uso de otras utilidades y paquetes de CP4D, que serán mencionadas en secciones posteriores.

3. Análisis

3.1. Estudio de viabilidad del caso de uso.

Durante los primeros meses de trabajo se realizó el estudio de viabilidad del proyecto, en el que participaron de forma activa (1) representantes de Ingeniería de Datos y la ODaN, (2) el equipo de Tracasa (principalmente el autor de este TFM y el jefe del equipo del desarrollo) y (3) especialistas en CP4D del departamento *Customer Success* de IBM (en menor medida, a modo de consultores).

La mayor parte del trabajo de este análisis se enfocó en estudiar qué posibilidades existen para publicar y gobernar los datos y metadatos de indicadores de diferentes fuentes. Se propusieron y valoraron 4 posibilidades:

1. Creación DM Repositorio de Indicadores e integración en Goberna: un data mart, alojado en SQL Server, que se integra con CP4D vía la virtualización de datos (para casos de consulta directa) y por medio de publicación de extractos de los datos como ficheros CSV. El metadatado se realiza en CP4D y se enlaza con datos publicados en la propia plataforma, lo que combina las novedades de este nuevo producto en términos de gobierno del dato con todas las ventajas que tiene SQL Server para los usuarios de Gobierno.

   Y estas ventajas son que, resumidamente, SQL Server es un sistema estándar en la administración pública, con el que hay experiencia de integraciones con cuadros de mando, que está perfectamente integrado con LDAP (por lo que la administración de usuarios es sencilla) y en general resulta un entorno conocido para los gestores de despliegue, lo que simplifica el proceso a nivel global.

   Además, tener un data mart para los datos resulta muy conveniente porque simplifica el trabajo de los analistas de datos.

2. Implementación de metadatos en Goberna y los datos en otros sistemas: los metadatos estarían definidos en CP4D para su consulta, y estarían enlazados a los datos. Los datos podrían alojarse de 3 formas, con sus ventajas y desventajas:

a. En una base de datos en el DataHub/DataLake gestionado por CP4D: muy conveniente pero este lago de datos todavía se encuentra en desarrollo, por lo que habría retrasado el proyecto hasta un año.
b. En DB2 (bases de datos relacionales en CP4D): muy útil porque DB2 es un módulo de la plataforma, por lo que la conectividad entre CP4D y las bases de datos es muy buena, y DB2 presume de un rendimiento muy alto. La principal desventaja es que, aunque ofrece unas posibilidades similares a las de SQL Server, a diferencia de este, no es conocido en Gobierno, por lo que su adopción conllevaría complicaciones.
c. En los sistemas origen de los proveedores: esta opción es la de menor carga de trabajo, pero genera dependencias con los proveedores, que en algunos casos no quieren realizar conexiones directas entre el CPD de Gobierno de Navarra con sus sistemas origen (este es el caso de Nastat, proveedor piloto del repositorio de indicadores).

En esta opción contempla publicar los datos sin seguir un modelo de data mart, sino que las fuentes de datos publicarían sus datos limpios, pero no unificados. Esto conlleva menos trabajo, pero no se alinea con uno de los objetivos principales del repositorio, ya que los indicadores podrían quedar centralizados (al menos sus metadatos) pero no estarían unificados.

3. Creación de una base de datos en SQL SERVER y una aplicación para gestionar el repositorio: esta opción se desestima porque una de las prioridades es alinear el proyecto con el gobierno de datos realizado en Goberna. Se trata de un software que comienza ahora a adoptarse en el Gobierno de Navarra y que se ha comprado precisamente para dar salida a casos de uso del estilo.
4. Publicación de la información en OpenData (CKAN) para consulta y para facilitar la descarga mediante API: en OpenData sólo se podrían publicar datos públicos; la solución debería poder implementar control de acceso, para poder publicar también indicadores de uso interno, por lo que esta opción queda descartada.

Tras desestimar las soluciones 3 y 4 y comparar las ventajas y desventajas de las dos primeras opciones, los gestores del proyecto decidieron, siguiendo el consejo de los

desarrolladores en Tracasa y de IBM, optar por la solución 1. Se inicia el trabajo para crear un data mart que unifique y centralice los indicadores y sus definiciones, alojado en un servidor del Gobierno de Navarra de SQL Server el cual se integra con la plataforma CP4D, donde se realiza su gobernanza.

3.2. Análisis de los datos del proveedor piloto.

El Instituto de Estadística de Navarra (Nastat) ha estado trabajando en la definición de un gran número de indicadores estadísticos, y ha creado una base de datos que incluye sus propios indicadores y otros generados por otras entidades, como el INE o EuroStat. De esta forma contribuye como un organismo central que sirve de proveedor de estadísticas y mediciones para todo el Gobierno de Navarra y sus ciudadanos. Para ponerlos a disposición de los usuarios finales, los publica en su página web (https://nastat.navarra.es/es/), junto con algunas visualizaciones como tablas con filtros dinámicos y cuadros de mando.

Los datos y definiciones de los indicadores que proporciona Nastat vienen en tres ficheros que irán actualizando periódicamente (como ellos lo consideren porque depende de cuándo realizan sus cálculos). Los ficheros son:

1. inventario_indicadores_datos.csv
2. indicadores_municipales_datos.csv
3. inventario_indicadores_definiciones.csv

Inicialmente el archivo de datos municipales se iba a poblar, pero cuando se proveyeron los datos finales para la primera versión del repositorio no se incluyó ningún indicador desagregado por municipios, por lo que este archivo está vacío en el primer despliegue. Aun así, el desarrollo se realiza contando con datos municipales (por lo que el sistema lo debe soportar, aunque no se utilice en el primer lanzamiento).

El CSV de datos de indicadores contiene los campos mostrados en la tabla 1.

Nombre del campo	Notas
id	Identificador interno en Nastat, números enteros. La mayoría por debajo de 1.500, algunos 5.XXX y algunos 10.XXX.
anio	Año de la medida
semestre	Semestre de la medida (0, 1 o 2). 0 cuando no aplica el semestre (se mide por trimestres o meses).
trimestre	Trimestre de la medida (0 al 4). 0 cuando no aplica el trimestre (se mide por semestres o meses).
mes	Mes de la medida (0 al 2). 0 cuando no aplica el mes (se mide por trimestres o semestres).
dato	Números de hasta 16 decimales.
fecha_creacion	Fecha de creación del indicador con formato dd/mm/yyyy.
fecha_actualizacion	Fecha de actualización del indicador con formato dd/mm/yyyy.

Tabla 1: Campos del archivo de datos de indicadores origen de Nastat

El fichero con datos municipales tiene los mismos campos y añade 'municipio' con el identificador del municipio correspondiente.

El CSV de definiciones contiene los campos de la tabla 2.

Nombre del campo	Notas
tema	Código de la primera jerarquía de clasificación temática en Nastat.
subtema	Código de la segunda clasificación temática.
ambito	Código de la tercera clasificación temática.
frecuencia	Código de la frecuencia de actualización (periodicidad de cálculo).
id	Identificador interno en Nastat, números enteros. La

	mayoría por debajo de 1.500, algunos 5.XXX y algunos 10.XXX.
nombre_indicador	Cadena de texto con el nombre del indicador.
territorio	Código del territorio (ES, ES31, EU, del INE).
desagregacion	Cadena de texto con la desagregación (si la hay, por ejemplo 'Municipios').
id_indicador_necesario	Lista de identificadores de indicadores con los que hay una dependencia.
calculo	Booleano si se realiza un cálculo para obtener el indicador.
unidad	Cadena de texto con la unidad de medida de la métrica (Ej.: habitantes, euros).
decimales	Número de decimales de la medida (hasta 16).
enlace	Enlace al dato publicado en la página de Nastat.
fuente	Cadena de texto con la fuente de datos (Ej.: Datos económicos de Nastat por medio del INE)

Tabla 2: campos del fichero de definiciones de indicadores origen de Nastat.

Además, se proporciona un Excel con las codificaciones y nombres de varios campos que aparecen en la tabla de definiciones, para hacer el intercambio de códigos por texto para usuarios finales. Con ese fichero se generan los siguientes CSV:

1. codificacion_tema.csv
2. codificacion_subtema.csv
3. codificacion_ambito_territorial.csv
4. codificacion_periodicidad.csv

Estos cuatro ficheros son estáticos, es decir, no se actualizan de forma periódica, sino por necesidad y al desplegar una nueva versión del repositorio de indicadores. Sólo contienen un campo con el código y otro con el nombre descriptivo.

3.3. Estándar de metadatos de indicadores en Goberna

Esta sección presenta el estándar de metadatos para indicadores desarrollado en Tracasa con la consultora Desidedatum. El estándar no lo ha desarrollado el autor de este TFM, sino que el trabajo realizado ha sido el de analizar los campos propuestos y asignar valores a todos los que sea posible, bajo la validación de la responsable del caso de uso en Ingeniería de Datos.

Por el momento basta con saber que se ha desarrollado un estándar para metadatar indicadores de forma unificada, para que cuando sean publicados en el Marketplace de Goberna, compartan una estructura común. Así, este estándar se verá reflejado en el frontal de CP4D y del Marketplace. En la tabla 3 se presenta la lista de metadatos y sus valores para los indicadores de Nastat. No se va a dar una descripción completa de todos los campos porque está fuera del alcance de este texto. Además, se han obviado algunos que son metadatos nativos en CP4D y no forman parte de esta parte del análisis.

Metadato	Campo en Nastat o Valor
Nombre	nombre_indicador
Descripción	-
Ámbito	tema
Sub-Ámbito	subtema
ID en Sistema Origen	id
Tipo de medida	“Cuantitativa”
Unidad Medida	unidad
Fórmulas de Cálculo	-
Periodicidad publicación	frecuencia
Ámbito temporal	frecuencia
Cobertura temporal	-
Cobertura geográfica	territorio
Dimensiones de desagregación	desagregacion
Normativa	-
Departamento	“Economía y Hacienda”
Dirección general	“No procede”
Unidad Organizativa	“(Nastat) Instituto Navarro de Estadística”
Nombre persona responsable	“Pablo Cebrián”
Email de contacto	“xxx@navarra.es”
Teléfono de contacto	“848XXXXXX”
Enlace	enlace

Público	“Sí”
Observaciones	-
Fuente datos origen	fuente
Ámbito poblacional	-
Fecha de modificación	-
Límites de publicación	-
Objetivo	-
Tipo de operación	-
Decimales de la medida	decimales
Código en el Plan de Estadística de Navarra	-
Código en el Plan Nacional de Estadística	-

Tabla 3: Campos del estándar de metadatos de indicadores y su correspondencia con Nastat

Se puede ver que hay varios campos que no se pueden completar con los datos que nos proporciona el primer proveedor de datos. Para fases futuras del proyecto, la responsable en Ingeniería de Datos solicitará a su responsable en Nastat que complete todos los campos que le sean posibles.

El desarrollo de este trabajo es bajo el rol de ingeniero de datos y de *data steward* técnico (encargado de la implementación de la gobernanza, no de su definición), por lo que la implementación debe permitir a futuro la carga de estos campos, pero no debe ser de estudiar el negocio a tanta profundidad como para hacer el trabajo del *data steward* de negocio, ya que esta responsabilidad recae sobre el representante del negocio. Así, en este trabajo se prepara un repositorio que pueda albergar estos metadatos, y si los campos no se pueden rellenar, quedarán como nulos.

3.4. *Data Fabric* y CP4D. Servicios y activos de CP4D utilizados en el proyecto.

Data Fabric es un concepto de solución de gestión de datos que ha crecido con fuerza en los últimos años, como predijo Gartner en el 2021 [8]. Se trata de una arquitectura que reúne múltiples servicios para unificar y facilitar el tratamiento de datos en una organización. Así, estos servicios, normalmente contratados en *cloud*, permiten virtualizar fuentes de datos, procesar dichos datos y finalmente ponerlos a disposición del consumidor del dato en forma de visualizaciones o con conexión directa a los datos, de una forma más ágil y efectiva. Además, los servicios de estas plataformas suelen ser escalables, por lo que se convierten en solución viable para casos de uso de *Big Data* y procesamiento en tiempo real.

Cloud Pak for Data es la plataforma de filosofía *data fabric* propietaria de IBM. La empresa ofrece este servicio cloud (como su propio nombre indica) pero también vende la solución en versión on-premise (opción por la que se opta en GdN). CP4D es, en resumen, una agrupación de servicios que corren en un clúster de OpenShift, una plataforma de contenedores basada en Kubernetes. Estos servicios se centran principalmente en: integración de datos, inteligencia artificial, gobernanza de datos y almacenamiento.

Así, si en CP4D se inicia un entorno de ejecución de cuadernos de trabajo Jupyter, por debajo lo que tenemos es un gestor que levanta contenedores que tienen Python y el resto de programas necesarios instalados. Y lo mismo para todos los demás servicios ofrecidos por la plataforma, ya sean bases de datos, virtualización, procesamiento y demás.

A continuación, se listan los componentes de CP4D que son utilizados en el caso de uso del repositorio de indicadores. La plataforma dispone de otros componentes, pero un listado completo queda fuera del interés de este trabajo.

Proyectos: son espacios de trabajo que contienen activos con los que realizar procesos de integración de datos (como montar conexiones o ejecutar procesos de ETL) y de gobernanza de datos (como controles de calidad o perfilado de datos). Los proyectos pueden exportarse e importarse, manteniendo muchos de sus activos para trasladar el trabajo, por ejemplo, de un entorno de desarrollo a uno de producción.

Trabajos: son ejecuciones que pueden lanzarse manualmente o programarse. Múltiples activos generan trabajos.

Los siguientes componentes son **activos** dentro de los proyectos.

Activos de datos: pueden ser ficheros de varios formatos o datos conectados (por ejemplo, una conexión a SQL Server). Pueden añadirse a un proyecto o generarse en el mismo. Pueden ser publicados en catálogos para ser compartidos con otros usuarios.

Cuadernos de trabajo (Jupyter): bloques de código en Python (o R, aunque no en este caso de uso) que son ejecutados de forma secuencial. Además de ejecución manual (como un cuaderno normal) pueden generar trabajos para ejecutarse

Interconexiones (pipelines): activos que se utilizan para orquestación de procesos utilizando programación gráfica (bloques con diferentes funciones interconectados entre sí). Son capaces de ejecutar trabajos de otros activos, como cuadernos Jupyter, entre otros.

Conjuntos de parámetros: son parejas clave - valor que sirven para configurar ejecuciones de trabajos. Por ejemplo, se puede definir un parámetro “entorno” con valor “producción” para configurar la ejecución de un pipeline.

Áreas de enriquecimiento de metadatos: espacios en los que se añaden activos de datos para su enriquecimiento. En ellos se definen parámetros de enriquecimiento de metadatos (configuración de análisis de calidad, perfilado de datos y asignación de artefactos de gobierno). Después se ejecuta un trabajo asociado a este activo y los datos añadidos al mismo se enriquecen.

Otros componentes globales (no forman parte de un proyecto) de CP4D que son necesarios para este trabajo son:

Volúmenes de almacenamiento: espacios de almacenamiento independientes que están apificados y a los que se pueden crear conexiones desde los proyectos. Pueden servir como *landing areas* para los datos de un proveedor.

Secretos: en general, son clave - valor, y pueden ser consultados vía API o por determinados procesos ejecutados por los usuarios que tengan acceso. Por ejemplo, para almacenar de forma global y segura las credenciales a una base de datos.

Con estas descripciones queda una idea general de qué elementos son necesarios para el caso de uso. En los apartados de diseño e implementación se dará mayor nivel de detalle para los componentes para los que sea necesario.

3.5. Aplicación de la gobernanza en CP4D. Artefactos de gobierno.

La gobernanza del dato es una práctica de gestión de datos en la que se definen procedimientos, políticas, roles y responsabilidades para tratar los datasets como activos valiosos en una organización. Eso es, asegurar la calidad de los datos, su disponibilidad, su seguridad, su acertado marco contextual y su facilidad para ser interpretados correctamente. En CP4D se dispone de una serie de objetos denominados artefactos de gobernanza [10] que facilitan la definición y aplicación de los procedimientos convenientes para alcanzar estos objetivos.

A continuación, se listan los artefactos que existen en la plataforma; para ilustrarlos se proponen algunos ejemplos de una prueba de concepto realizada con los datos públicos del catastro de Navarra.

Categorías: sirven para organizar los artefactos de gobierno en una estructura estilo directorio, por lo se puede tener categorías padre e hijo y almacenar artefactos al nivel que se desee. Por ejemplo, se puede tener la estructura “Hacienda> Catastro> Alfanumérico>…” y cualquiera de los niveles puede contener cualquier tipo de artefacto de gobierno. Implementan control de acceso para gestionar permisos en los artefactos que contienen.

Términos empresariales: permiten formalizar un vocabulario empresarial común entre los departamentos para describir sus activos de datos. Se pueden introducir relaciones entre los términos para desarrollar una ontología. De base existen relaciones de “parte de”, “tipo de”, sinonimia y relación general. Se les pueden asignar clases de datos que sirvan para asignar términos de negocio a columnas de datos que siguen el patrón de la clase de datos.

Clases de datos: en ellas se definen patrones con los que clasificar datos. Se pueden usar expresiones regulares, rangos, tipos de datos, datos de referencia/listas, y nombres de columna. Con las clases de datos podemos asignar automáticamente los términos de negocio adecuados. Por ejemplo, podemos crear una clase de datos para detectar datos de Bienes Inmuebles con un RegEx que describa el formato de su código.

Datos de referencia: recopilaciones que pueden ser usadas en clases de datos como método de coincidencia. Por ejemplo, se genera una lista de nombres de municipios navarros y una clase de datos "Municipio navarro" cuyo método de coincidencia usa dicha lista. Cuando una columna tenga muchos registros que coincidan con los elementos de la lista, se asignará esa clase de datos (y término si lo hubiese).

Clasificaciones: con ellas describimos características de los datos. Por defecto CP4D trae clasificaciones para describir la sensibilidad de los datos, pero la plataforma no está limitada a ese uso. Por ejemplo, podríamos describir activos de datos con clasificaciones "Preparado" y "Modelado" según la fase de transformación a la que pertenece el activo. Se pueden establecer relaciones de padre-hijo entre clasificaciones.

Reglas: existen reglas de dos tipos: reglas de gobernanza (que son sólo declarativas) y reglas de protección de datos (ambas accionables). Las reglas de protección de datos permiten ofuscar o controlar el acceso en base a condiciones con respecto a usuarios, grupos, clasificaciones, términos empresariales, clases de datos, etiquetas, propietarios del activo y nombres de columna. Además, en la primera versión de CP4D instalada, existían las reglas de soberanía, que además permiten controlar el acceso en base a la ubicación de los datos o del solicitante. Estos dos tipos de reglas no se organizan en categorías como el resto de artefactos. Por ejemplo, podría crearse una regla para enmascarar datos sobre el estado de un bien inmueble a usuarios fuera del departamento de Hacienda.

Políticas: describen a nivel general cómo administrar y proteger los datos. Agrupan reglas de protección de datos y de gobernanza, pero en sí mismas son meramente declarativas. Por ejemplo, directivas generales sobre cómo proteger datos personales en los activos de catastro.

Todos los artefactos de gobierno excepto las categorías y clasificaciones permiten establecer relaciones con clasificaciones y términos empresariales. Además, la plataforma permite crear relaciones adicionales entre artefactos de gobierno, entre activos, o entre un activo y un artefacto. Debe tenerse en cuenta que los tipos de relaciones que se crean son globales y que por tanto el interfaz de usuario nativo de CP4D puede quedar muy recargado si se genera un gran número de ellas. Estas relaciones facilitan la navegación entre activos y artefactos y son muy útiles para los usuarios finales.

Con estos artefactos surgen muchas posibilidades de aplicación de gobernanza de datos. Podemos resumir las prácticas de gobernanza en tres dimensiones:

PROTECCIÓN: control de acceso y ofuscación de datos: mediante reglas que se aplican usando el resto de artefactos. Podemos filtrar filas, enmascarar columnas o denegar/permitir el acceso a un activo. El control de acceso se puede administrar también por catálogo o directamente por activo en vez de usar reglas.

CALIDAD: métricas de calidad de un activo: CP4D analiza la calidad de un activo de datos teniendo en cuenta varias métricas que a priori no se pueden alterar (como la consistencia de mayúsculas/minúsculas, presencia de nulos, etc.). Existe una métrica denominada "violación de clases de datos" que es controlable porque la clase de datos la define el desarrollador y se puede asignar/desasignar manualmente. Por otro lado, las reglas de calidad son activos que se desarrollan en proyectos y que se lanzan por separado. Su resultado no se ve reflejado en la métrica de calidad en la versión instalada actualmente. Desde IBM aseguran que en versiones posteriores del producto se verá reflejado en la pestaña de calidad de los activos de datos, por lo que ese análisis de datos complementará al por defecto.

CONTEXTO: principalmente mediante relaciones entre activos y conceptos empresariales, por medio de la ontología desarrollada y el metadatado de los activos (en concreto sus relaciones con artefactos u otros activos). Con este contexto se facilita el descubrimiento de datos, lo que permite establecer sinergias entre negocios.

En el diagrama de la figura 2 se ilustran las relaciones de los artefactos de gobierno con las dimensiones de gobierno establecidas.

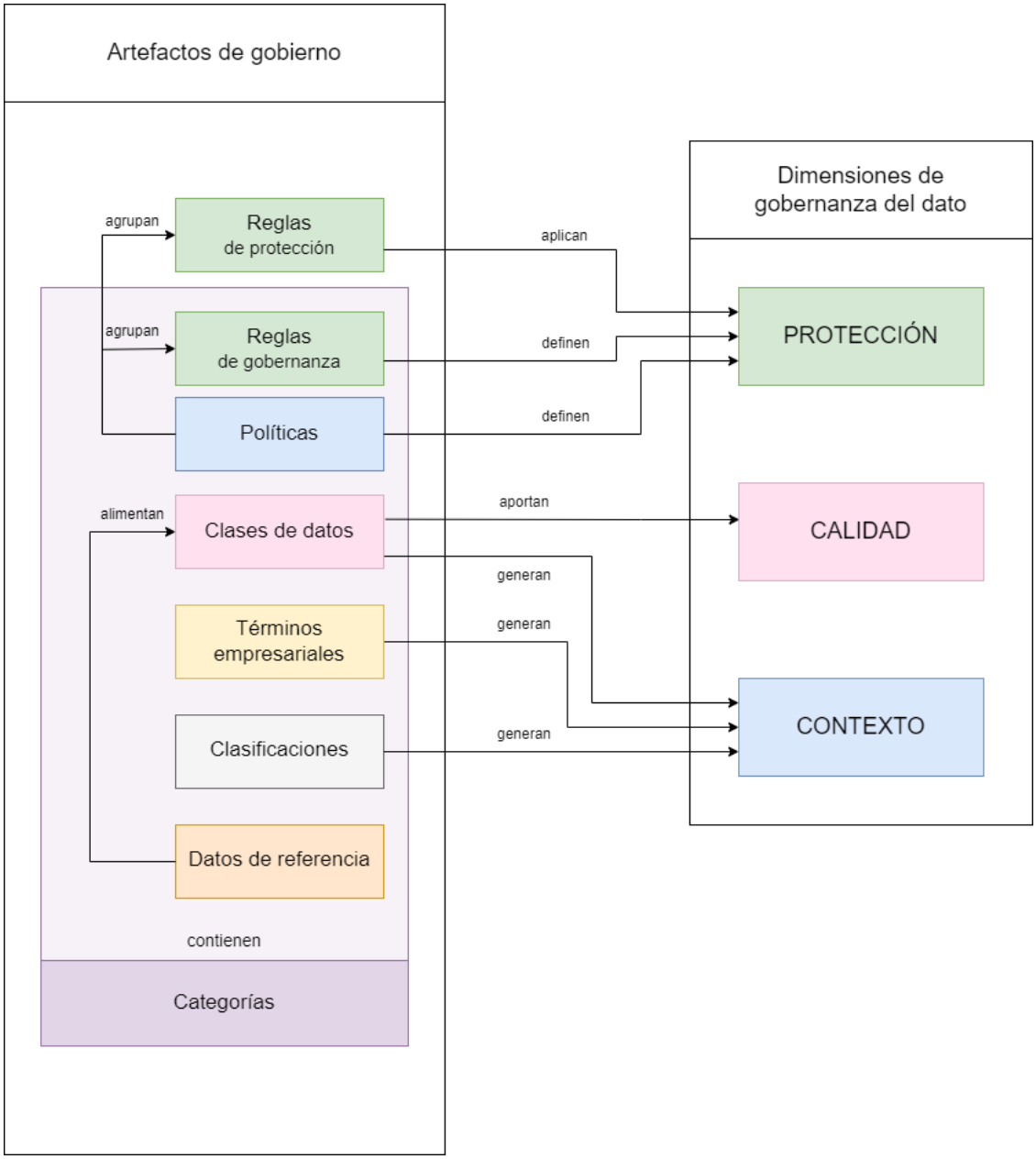

Figura 2: Artefactos de gobierno de CP4D y su relación con dimensiones de gobernanza

Para la primera versión del repositorio de indicadores, únicamente se van a generar términos empresariales y una categoría. El resto de artefactos serán utilizados en versiones futuras del repositorio, conforme el proyecto global entre en una fase más madura y se definan en mayor detalle los procedimientos de gobernanza que se deben poner en práctica.

Todos estos artefactos de gobernanza forman parte del servicio *Watson Knowledge Catalog* de CP4D, y tienen un rol de importancia en la gobernanza de datos junto con los

catálogos. Estos son lugares en los que se publican los activos de datos gobernados para su compartición con el usuario final. Desde ahí, un usuario con acceso a un cierto activo de datos puede llevarlo a un proyecto propio para realizar sus propios análisis y procesamientos.

3.6. Publicación de datos en CP4D

Para el caso de uso, publicar las definiciones (los metadatos) de los indicadores casi tanto o más que publicar sus datos en sí. No sólo porque son necesarias para interpretar correctamente los datos, sino porque una de las finalidades esenciales del proyecto es facilitar el descubrimiento de datos; tanto es así, que este caso de uso es también referido a veces como “Catálogo de Indicadores”.

Durante los primeros meses de trabajo se realizó formación en la plataforma. Con este conocimiento, para la publicación de indicadores en CP4D se propusieron y analizaron las siguientes opciones:

1. Publicar en un catálogo activos de datos con los valores de los indicadores junto con activos personalizados para las definiciones. Los activos personalizados son básicamente grupos de metadatos que aparecen como activos en catálogos de datos, aunque no contienen un dataset.
2. Publicar las definiciones de indicadores como términos de negocio (a los que se pueden añadir campos de metadatos) y los valores en activos de datos.
3. Publicar en activos de datos los valores y las definiciones juntos.
4. Publicar en activos de datos independientes los valores y las definiciones.

En la tabla 4 se valoran las opciones mostrando sus ventajas y desventajas.

Opción	Ventajas	Desventajas
1. Definiciones en atributos personalizados, valores en datasets	- Se construyen relaciones entre activos de datos y las definiciones de los indicadores que contienen. - Los indicadores sólo son visibles en el catálogo dedicado.	- El catálogo queda inundado de activos personalizados (sólo con Nastat más de 1000) y los activos de datos pierden visibilidad. - La descarga de datos no incluye las definiciones, tengo que obtenerlas de otro sitio.
2. Definiciones como términos, valores en datasets	- Se construyen relaciones entre activos de datos y las definiciones de los indicadores que contienen. - Los indicadores aparecen en la búsqueda global y también el activo en el que están publicados sus valores.	- Las definiciones de indicadores quedan al mismo nivel que el resto de términos de negocio (no están distinguidas como objetos diferentes). - La descarga de datos no incluye las definiciones, tengo que obtenerlas de otro sitio.
3. Valores y definiciones en datasets juntos	- El usuario descarga todos los datos y definiciones de golpe.	- No se construyen relaciones semánticas. - No hay búsqueda global de indicadores.
4. Valores y definiciones en datasets separados	- Es la publicación que se asemeja más al dataset que hay en el data mart de indicadores.	- No se construyen relaciones semánticas. - No hay búsqueda global de indicadores. - Para unir definiciones y valores el usuario necesita hacer un ligero procesado (join de las tablas).

Tabla 4: Opciones de publicación estudiadas. Ventajas y desventajas.

Además, la publicación de activos de datos se puede hacer de dos maneras:

1. Como datos conectados. Datasets cuyos datos se obtienen mediante conexiones a tiempo real al data mart. La ventaja de esta opción es que evita tener que generar

activos nuevos, simplemente se crea la conexión y conforme el data mart se actualiza, también lo hacen los activos publicados.

2. Como ficheros generados a partir de los datos del data mart. Este tipo de publicación es descargable, a diferencia de los datos conectados, y por tanto es muy cómodo para analistas, especialmente aquellos que están acostumbrados a trabajar con ficheros tipo CSV (son cómodos de importar en herramientas como Excel, que hoy en día son estándar).

Tras comparar las propuestas se llegó a la conclusión de que la opción más viable es la de publicar las definiciones de los indicadores como términos de negocio (opción 2), pero a su vez publicando los datos junto a las definiciones como activos de datos (opción 3). Esta decisión estuvo apoyada por el encargado del seguimiento del proyecto por parte de IBM.

Esto permite que un analista que está investigando cierto tema:

- Descubra indicadores que aporten conocimiento a su estudio por medio de una búsqueda global de un concepto. Por ejemplo, buscando “natalidad” encontraría un término “Índice de natalidad por comunidades autónomas (%)”.
- Encuentre activos de datos que contienen valores para los indicadores que necesita, por medio de las relaciones existentes entre términos y activos. Ahí, puede explorar el activo mediante la previsualización que ofrece el frontal de CP4D (o el del Marketplace, una vez esté disponible).
- Pueda llevar el activo a un proyecto para realizar su análisis o descargarlo para estudiarlo en sus herramientas propias.

Con la opción escogida se utilizan de forma efectiva las posibilidades de la plataforma para dar un servicio completo a usuarios analistas de la administración pública.

4. Diseño

4.1. Esquema general

En la figura 3 se muestra el ciclo de vida del dato a alto nivel, que comienza en las fuentes de información, pasa a un data mart y al Marketplace y finalmente es consumido por usuarios finales o sistemas terceros (como un cuadro de mando). De forma sintética, y siguiendo el orden desde los orígenes de información a la explotación y publicación de los datos, la propuesta se basa en:

(1) Automatizar al máximo la ingesta de las fuentes de datos proporcionadas por los proveedores del repositorio. En el caso de Nastat, ficheros de datos CSV.

(2) Modelar y consolidar los indicadores (datos y metadatos) mediante **procesos de ETL en Python y SQL Server. Orquestar** dichos procesos mediante **Python.**

(3) Controlar la calidad de los datos durante la ETL y una vez cargados en Goberna.

(4) Generar un glosario de negocio de indicadores usando las definiciones proporcionadas por los proveedores.

(5) Publicar el Repositorio de Indicadores en catálogo de Goberna, introduciendo **control de acceso** (por catálogo). Publicar en forma de **ficheros de datos.** A futuro, publicar en el Marketplace de Goberna y publicar por medio de conexiones virtualizadas a SQL Server.

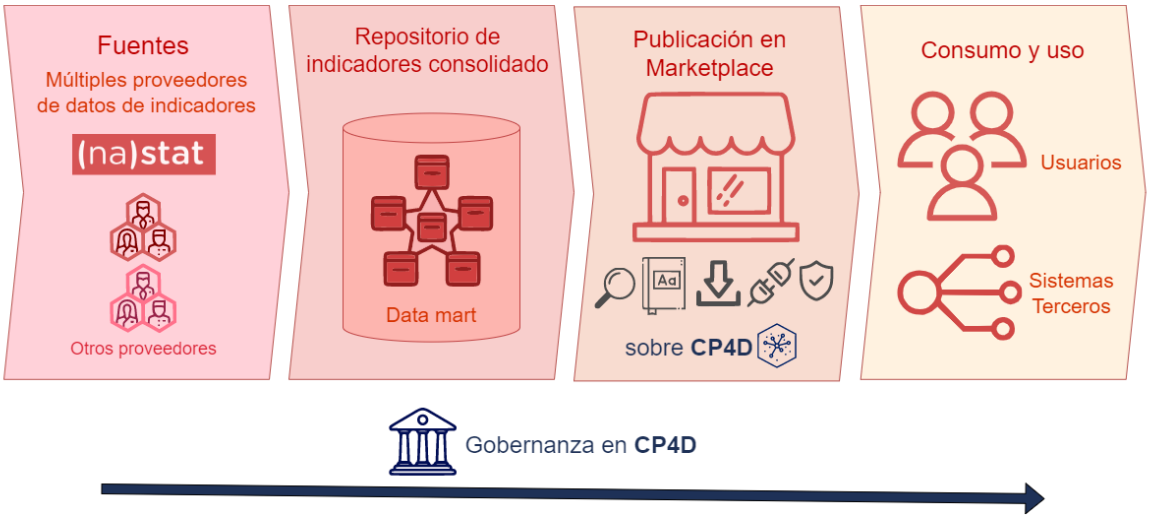

Figura 3: Ciclo de vida del dato a alto nivel en este caso de uso.

Para el diseño del modelo de datos del repositorio de indicadores en SQL Server se va a seguir la metodología Kimball. En ella se definen recomendaciones para construir un almacén de datos (*data warehouse*) en el que se puedan historificar datos de múltiples fuentes de forma limpia, escalable, segura y que facilite su interpretación y consulta.

4.2. Diseño lógico

En la figura 4 se muestra un diagrama del diseño lógico del ciclo de vida del dato en este caso de uso.

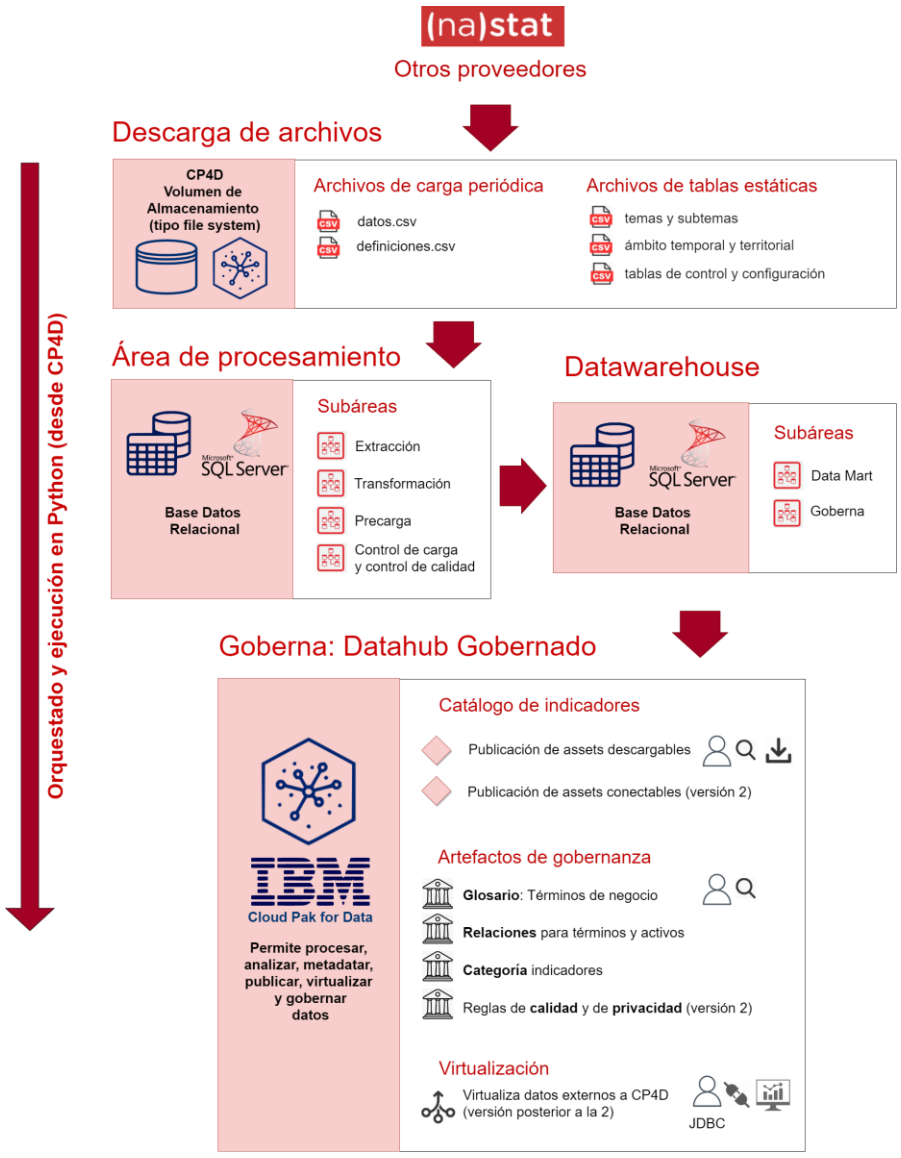

Figura 4: Diagrama del diseño lógico del caso de uso

A continuación, se narra dicho ciclo de vida paso a paso, siguiendo el esquema propuesto:

1. Los indicadores son generados en sus sistemas de origen. En el caso de Nastat, se crean dos ficheros CSV a partir de los cálculos realizados en su clúster, uno para datos sin desagregar y otro para datos desagregados por municipio. Se genera otro CSV a partir de los metadatos que componen la definición de los indicadores.
2. Estos ficheros se cargan en un volumen de almacenamiento de CP4D dedicado exclusivamente a este caso de uso. Inicialmente, la carga se realiza manualmente por la responsable de Ingeniería de Datos; más adelante, se hace de forma automatizada vía API desde el clúster de Nastat.
3. Se ejecuta la ETL: el proceso comienza con los archivos origen (CSV) y termina en el data mart.

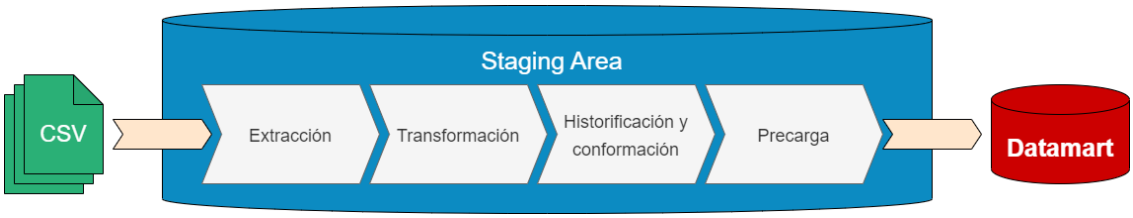

Figura 5: Diagrama de los pasos (y esquemas) de la ETL

i. Extracción: lectura de los ficheros depositados por el proveedor en el volumen y escritura de sus datos en tablas de extracción (sin aplicar ninguna transformación al dato o su esquema). Las extracciones son independientes para los datos de cada proveedor.

ii. Transformación: procesos de modelado de las dimensiones y de la tabla de hechos. En esta fase de modelado se estandarizan los campos utilizados para definir los indicadores. Las transformaciones son independientes para los datos de cada proveedor.

iii. Control de calidad de proveedor: a los datos modelados de cada proveedor se les pasa un control de calidad. En este trabajo se prepara la infraestructura para que, por medio de procedimientos almacenados, se definan condiciones de calidad que cuando no se pasen se impida la carga de los datos del proveedor. Por ejemplo, para impedir que cuando se incluya un indicador al que no hay datos asociados, no

se cargue en el repositorio. Sin embargo, estas reglas las debe definir un data steward, así para la primera salida sólo se proponen dos reglas que sirven de ejemplo.

iv. Historificación y conformación: SCD2 en la dimensión de definiciones de indicadores (para historificar los cambios que sufra la definición de cualquier indicador) y conformación de las definiciones de todos los proveedores. Conformación de los datos de todos los proveedores en una tabla de hechos común. Este paso, en el esquema habitual ETL, es una segunda fase de la transformación de datos. Solo pasan a esta fase los datos de proveedores que han pasado el control de calidad.

v. Precarga: cruce de las DK por SK en la tabla de hechos. Tablas finales para subir al data mart. Se realiza este paso para evitar aplicar lógica en la subida a la base de datos final.

vi. Carga: copia de las tablas finales en la base de datos del data mart, sin aplicar ninguna lógica en el proceso.

4. A partir del data mart se generan dos tablas que corresponden a datasets que se publicarán en un futuro en el Marketplace, virtualizando una conexión al SQL Server. Hasta entonces, sirven para generar los archivos que se publicarán en el catálogo de indicadores.
5. En el proyecto donde se ejecuta el proceso se generan los activos de datos y definiciones de indicadores en formato CSV. Para la primera versión del caso de uso, se genera un dataset por cada subtema al que correspondan indicadores con datos.
6. Para cada indicador se publica un término de negocio con todos sus metadatos (su definición completa). Estos términos se publican en una categoría denominada “Indicadores” que podrá ser subdividida en categorías para cada proveedor o temática en un futuro, por ejemplo: “Indicadores” - “Indicadores Nastat” o “Indicadores Económicos”.
7. Se inicia el proceso de enriquecimiento de metadatos: esto se realiza en las áreas de enriquecimiento de metadatos. Ahí se pasa un segundo control de calidad de CP4D (que está predefinido) y se realiza un perfilado de datos (estadísticas de datos). Se

asignan los términos de negocio de indicadores a los activos de datos a los que pertenecen. Es decir, al activo publicado “Indicadores de Cuentas Económicas” (uno de los temas definidos por Nastat), se le asignan todos los indicadores de cuentas económicas, formando así la relación entre el glosario de indicadores y los activos de datos publicados, y facilitando el descubrimiento de datos por medio de la búsqueda global.

8. Cuando finaliza, se publican los activos en un catálogo y se asignan metadatos de activos de datos (como el autor, la fuente, etc.). Finalmente, un usuario busca un concepto en el buscador global, encuentra una definición de un indicador, y después de leer su descripción, unidad, fuentes, y otros metadatos, entra en el dataset. Ahí decide utilizarlo: importando el activo de datos en un proyecto, o descargándolo para trabajar en local.

El sistema puede dividirse en 5 capas o bloques lógicos que forman el diseño completo:

1. **Orígenes de información.** Ficheros depositados por los diferentes proveedores de datos y metadatos de indicadores que consoliden el Repositorio de Indicadores. Por ahora Nastat.

2. **Capa de adquisición y procesamiento de datos.** Mecanismos y componentes software que cargarán la información en el Repositorio de Indicadores: procesos de extracción, transformación y carga (ETL), componentes de integración, etc. También en este apartado se tienen en cuenta mecanismos para el tratamiento de dicha información, desde procesos de depuración, limpieza y validación de los datos hasta procesos de enriquecimiento de metadatos.

3. **Capa de almacenamiento de datos.** Repositorios de información que contendrán los datos del Catálogo de Indicadores de Gobierno de Navarra y al que accederán los componentes de adquisición de datos (para cargar información), así como los componentes de publicación de datos (para generar ficheros finales o virtualizar los accesos al servidor y aplicar gobernanza).

4. **Capa de gobernanza y publicación de datos.** Conjunto de herramientas y procesos de calidad, enriquecimiento, seguridad que tienen lugar en Goberna, así como la publicación de los datos y metadatos en catálogo para el acceso final desde Marketplace.

5. **Capa de consumidores.** Conjunto personas con diferentes perfiles tanto de uso como de acceso a la información y que utilizarán la información obtenida para la toma de decisiones.

En la tabla 5 se resumen los componentes funcionales y productos software utilizados para el caso de uso, clasificados por la capa a la que pertenecen.

(2) Capa de adquisición y procesamiento de datos	
Orquestación, ingesta y procesamiento de datos	**Librería propia desarrollada en Python.** Orquestación y ejecución de la ETL usando librerías de procesamiento de datos. Procedimientos almacenados de SQL Server llamados desde la librería Python.
Control de Calidad	El control de calidad se implementará tanto en los propios procesos ETL usando procedimientos almacenados y lógica en Python, como en los procesos de enriquecimiento que se llevan a cabo en la plataforma mediante reglas de calidad y los controles por defecto del perfilado.
(3) Capa de almacenamiento de datos	
Volúmenes de Almacenamiento	Zona de carga de archivos propia de Goberna en la que los proveedores deben depositar los orígenes de información (ficheros CSV) para ser tratados.
Almacenamiento Intermedio (Staging Area)	En una base de datos dedicada en un SQL Server de Gobierno de Navarra denominada “RepositorioIndicadores_SA”. Esquemas para el control de los procesos de carga y la extracción, transformación y precarga de los datos. Las tablas específicas se listan en el diseño físico.
Almacenamiento Consolidado (Data Mart)	En una base de datos dedicada en un SQL Server de Gobierno de Navarra denominada “RepositorioIndicadores_DM”. Un esquema para el data mart, con la tabla de hechos (datos de los indicadores) y las dimensiones necesarias, así como el resultado del control de calidad llevado a cabo durante la ETL. Las tablas específicas se listan en el diseño físico.
(4) Capa de gobernanza y publicación de datos	
Búsqueda de indicadores	Usando los artefactos de gobierno y activos propios de Watson Knowledge Catalog en Goberna. Términos de negocio que recogen las definiciones de los indicadores, etiquetas y una categoría donde publicar los términos por el momento. A futuro, clases de datos para mejorar el control de calidad, clasificaciones para la sensibilidad de los

	datos.
Control de acceso	Mediante los grupos, usuarios, roles y permisos nativos de Goberna para el control de acceso a datos no públicos. También con la posibilidad de crear y utilizar reglas de protección de datos que ofrezcan una mayor granularidad a la hora de ofrecer o denegar datos al usuario final. Esta funcionalidad por el momento no es crítica; los datos de Nastat son públicos, por lo que el control de acceso se pospone a cuando se incluyan indicadores de uso interno (privados).
Publicación	Los activos de datos (sean datos conectados o ficheros) se publican en catálogos del Watson Knowledge Catalog dedicados a la compartición de indicadores.

Tabla 5: Componentes funcionales del caso de uso divididos por capa lógica

4.3. Diseño físico

4.3.1. Almacenamiento de datos

Capa conformada por los sistemas en los que se persisten los datos originales, los datos de diferentes etapas durante el proceso de ingeniería y los datos finales.

1. **Volumen de almacenamiento**

Lugar en el que los proveedores depositan los archivos origen de información. Los volúmenes de almacenamiento en Goberna son espacios transversales a los proyectos a los que se puede acceder vía API; son una opción segura en la que crear una zona de transferencia de archivos con el negocio, ya que así se evita dar permiso de acceso a los proyectos donde se ejecute el proceso, y al ser vía API la carga de archivos es automatizable.

2. **Base de datos Staging Area**

Denominada RepositorioIndicadores_SA, en esta base de datos se encuentran las tablas intermedias de la ETL. En ella se encuentran los siguientes esquemas:

Nombre del esquema	Contenido
dbo	No se utiliza.
control	Tablas de configuración del proceso de ingeniería y de su orquestado. Tablas de registro histórico de ejecución
goberna	Tablas de configuración del proceso de carga a Goberna, principalmente los activos a publicar con todos sus metadatos.
extraccion_COM	Tablas de extracción de archivos de datos comunes (no específicos de un proveedor). Tablas de extracciones históricas (datos de las extracciones anteriores).
extraccion_NST	Tablas de extracción de archivos de datos de Nastat (NST). Tablas de extracciones históricas (datos de las extracciones anteriores).
extraccion_ {código de proveedor}	N esquemas para los N proveedores de indicadores. Contienen tablas de extracción de archivos de datos de cada proveedor (XXX). Tablas de extracciones históricas (datos de las extracciones anteriores).
transformacion_COM	Tablas de transformación de datos para modelo de estrella (dimensiones y tablas de hechos) comunes (no específicas de un proveedor).
transformacion_NST	Tablas de transformación de datos para modelo de estrella (dimensiones y tabla de hecho) específicas de Nastat.
transformacion_ {código de proveedor}	N esquemas para los N proveedores de indicadores. Contienen las tablas de transformación de cada proveedor (XXX).
historificacion_ conformacion	Tablas de conformación, donde se unen los datos y metadatos de todos los proveedores. Además, historificación usando SCD2 para la dimensión de indicadores (metadatos).
precarga	Tabla de hechos con SKs de las dimensiones. Catálogo de indicadores (modelo de estrella entrelazado para publicar en Goberna).

Tabla 6: Esquemas de la base de datos de *Staging Area*

A continuación, se muestran las tablas que componen cada uno de los esquemas anteriores.

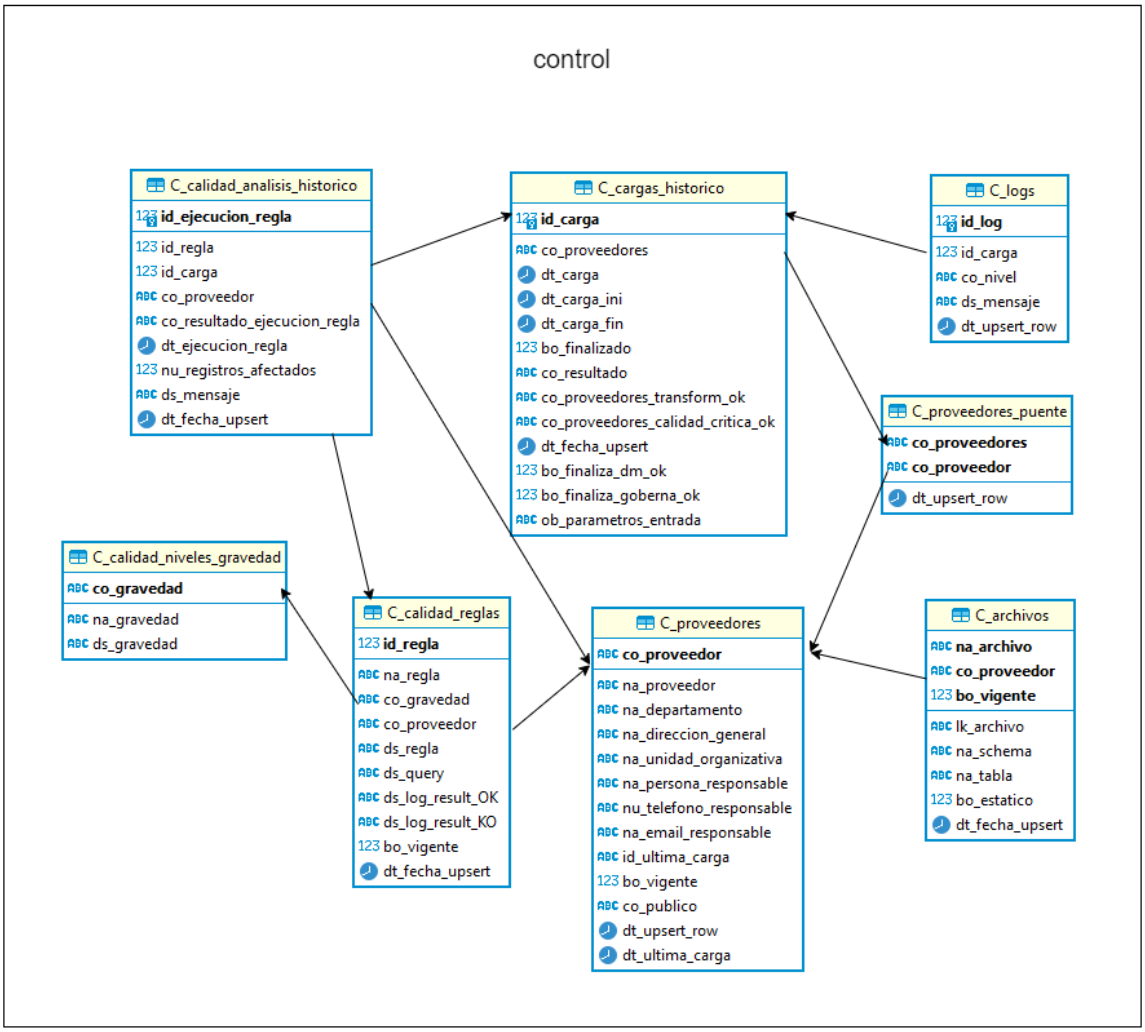

Figura 6: Modelo de datos del esquema ‘control’

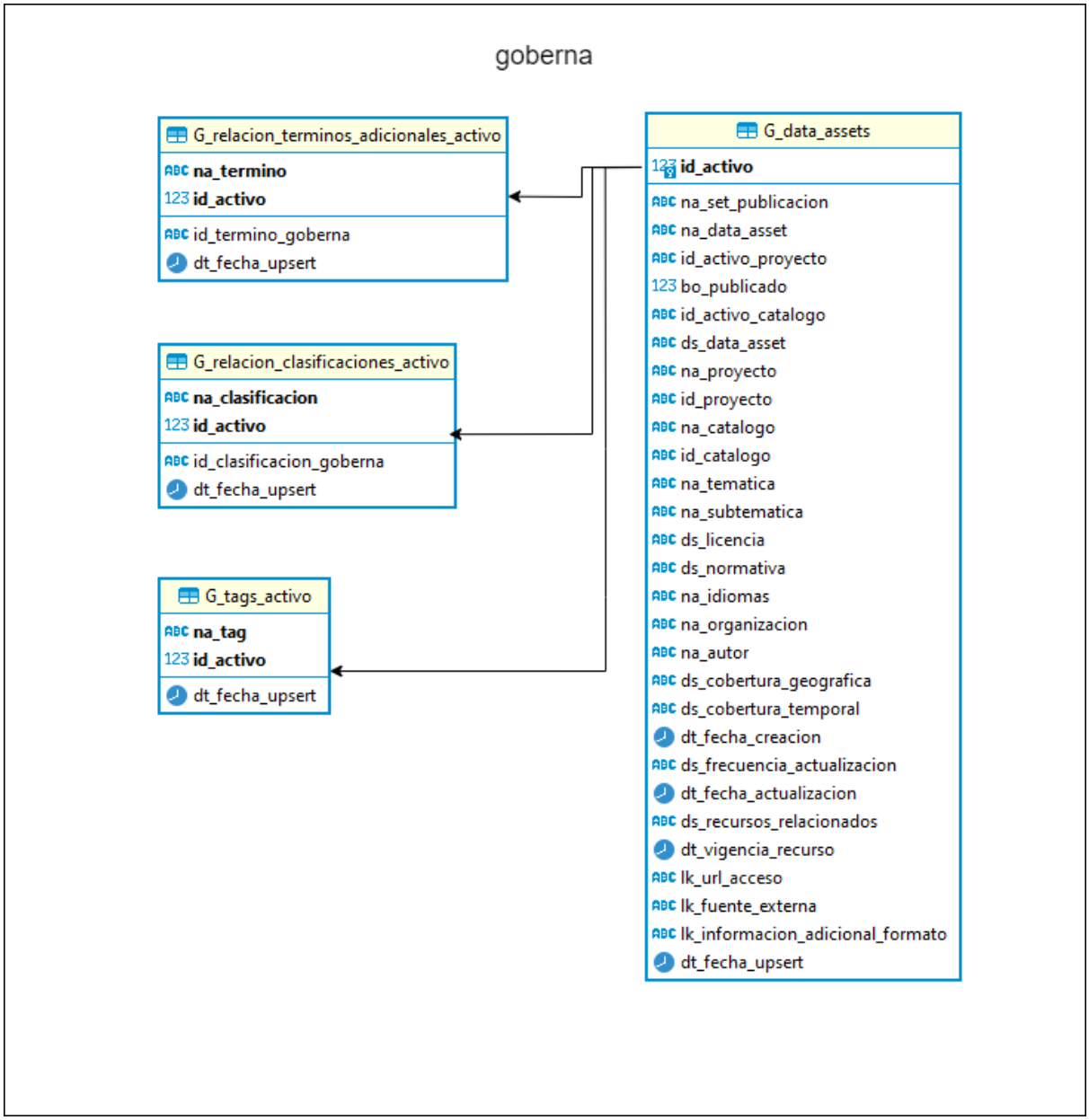

Figura 7: Modelo de datos del esquema ‘goberna’

Figura 8: Modelo de datos de los esquemas de extracción

Figura 9: Modelo de datos de los esquemas de transformación, historificación y precarga

3. Base de datos del data mart

Denominada RepositorioIndicadores_DM, contiene el modelo informacional del almacén de datos. Es a donde apuntarán las diferentes herramientas de análisis o reporte de la capa de explotación. Hospedada en el mismo SQL Server que el Staging Area. En ella se encuentran los siguientes esquemas:

Nombre del esquema	Contenido
dbo	No se utiliza.
datamart	La tabla de hechos (datos de indicadores) y sus dimensiones. Por ahora, Dim_indicadores (metadatos de los indicadores), Dim_fecha, Dim_municipios (para desagregaciones municipales de los datos). Además, una tabla C_calidad_analisis_historico donde consultar el resultado de la ejecución de reglas de calidad en la última carga del DM.
publicacion	Esquema adicional en el que hay dos tablas: Catalogo_Indicadores, que contiene las definiciones de los indicadores (pero no como dimensión, simplemente los metadatos) y Catalogo_Indicadores_Datos, que contiene los valores y las definiciones de los indicadores. Estas dos tablas servirán para generar los datasets que se publiquen en CP4D, tanto como ficheros como tablas virtualizadas.

Tabla 7: Esquemas de la base de datos de data mart

A continuación, se muestran las tablas que componen cada uno de los esquemas anteriores.

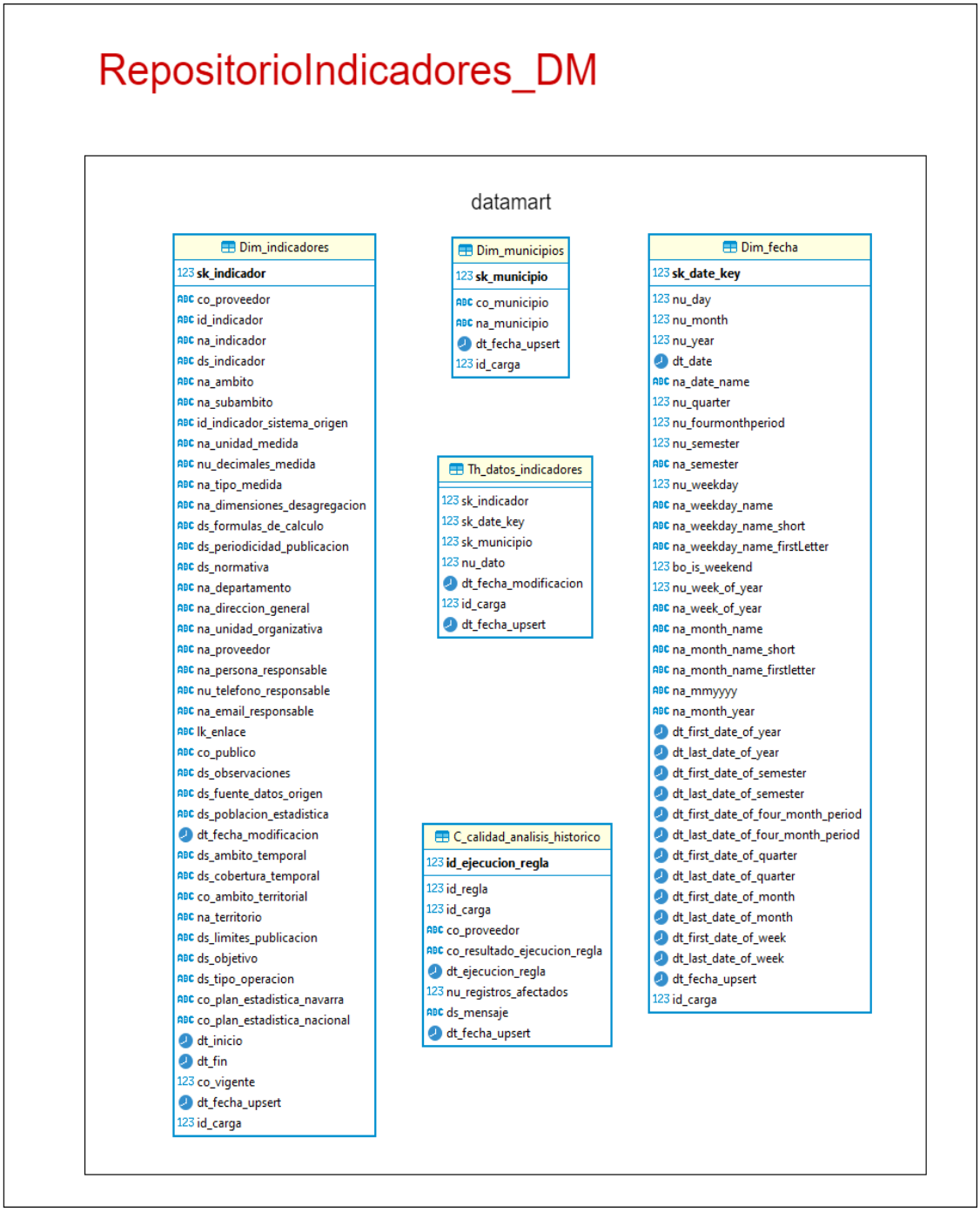

Figura 10: Modelo de datos del esquema ‘datamart’

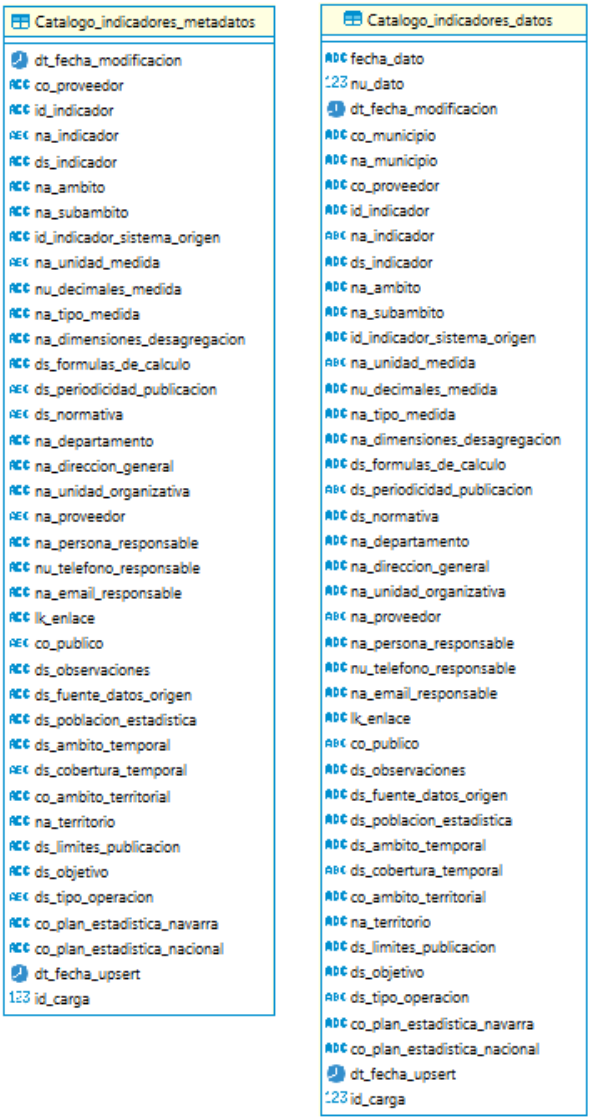

Figura 11: Modelo de datos del esquema ‘publicacion’

4. Almacenamiento de proyecto y catálogo

En el proyecto se almacenan los archivos de los activos de datos que luego son publicados en el catálogo. Los proyectos y catálogos tienen un espacio de almacenamiento reservado en el clúster de CP4D y los almacenan junto al resto de activos del proyecto. En principio para la primera versión son 32 activos de datos diferenciados por temática en formato CSV para facilitar a los usuarios finales su descarga y uso.

4.3.2. Control de acceso

Para la primera versión del caso de uso únicamente se cargan datos públicos, por lo que no es crítico un control de acceso al catálogo donde se publican. Aun así, de base CP4D realiza un control de acceso a los catálogos, así que, para permitir acceso a todos los usuarios, el gestor de despliegue debe crear un grupo de usuarios de "Visores mínimos" en el que incluir a todos, y en el catálogo habilitarle el acceso de "Visor" al grupo. En cuanto al proyecto donde se ejecuta el proceso de carga, ahí únicamente tiene acceso el grupo de "Gestores de despliegue".

El acceso a CP4D se controla por medio de un LDAP que el Gobierno de Navarra utiliza para gestionar la autorización de acceso a sus sistemas informáticos y que está integrado con la plataforma. Por el momento, en CP4D es necesario buscar los usuarios de LDAP y darles un permiso mínimo para que puedan comenzar a usar la plataforma, es decir, aún no se ha abierto a todo el personal de la administración pública.

El control de acceso a las bases de datos de SQL Server debe ser administrado por los gestores de implantación. Ellos crean un usuario de servicio para cada entorno (desarrollo, preproducción y producción) con el que realizar las conexiones del proceso de carga automático. Al resto de usuarios que requieran acceso, por la razón que sea, les puede ser concedido únicamente por medio de su usuario en el LDAP de la empresa (gestión de usuarios centralizada).

4.3.3. Conectividad entre componentes

Como se ha mencionado, el proceso de ingeniería de datos se orquesta y ejecuta mediante Python (en un cuaderno Jupyter) en un entorno levantado en CP4D (dentro de un proyecto). Por tanto, para el funcionamiento del proceso es necesario asegurar la conectividad con el resto de componentes desde este entorno. De las siguientes, las dos primeras conexiones se realizan dentro de CP4D y la tercera es de CP4D al exterior:

1. **Con el volumen de almacenamiento**

Para extraer los datos de los orígenes de información debemos realizar lecturas de ficheros CSV almacenados en un volumen de almacenamiento de CP4D. Para ello, es necesario crear una conexión con el volumen en el proyecto donde se ejecuta el proceso. Las conexiones son un tipo de activo de datos y requieren que al usuario que realiza una lectura por medio de una se le haya otorgado acceso al volumen.

Una vez creada la conexión, al levantar un entorno de cuadernos Jupyter, se genera un directorio en la raíz del entorno que está mapeado al volumen de almacenamiento. Así, la lectura de los ficheros se realiza como si el fichero fuese local al entorno.

2. **Con Watson Knowledge Catalog**

Este servicio está convenientemente *apificado*, por lo que la carga de los términos y activos de datos en el catálogo se realiza por medio de llamadas API desde Python. La autenticación vía API se hace con *bearer tokens.* Estos tokens se pueden solicitar usando el par usuario-clave de API o usuario-contraseña (menos recomendable).

Estas solicitudes son sólo necesarias para realizar pruebas en local, ya que los entornos levantados en CP4D contienen una variable de entorno con un bearer token funcional, por lo que la librería Python simplemente debe leer esa variable de entorno. El token generado tiene los mismos permisos que el usuario que levanta el entorno (los permisos varían según los roles que tenga asignados el usuario).

3. **Con las bases de datos Staging Area y Data Mart.**

La conexión en Python se realiza utilizando el driver de SQL Server para SQLAlchemy, el cual se utiliza como motor de query y se comunica con Pandas. Así, al realizar una lectura con SQLAlchemy, lo cargamos directamente en un dataframe de Pandas, y a la inversa, para escribir en una base de datos lo podemos hacer directamente con un dataframe usando el mismo motor.

Aunque tanto CP4D como los SQL Server están dentro de la red de Gobierno de Navarra, por motivos de seguridad existe un cortafuego que limita las conexiones que necesitan

realizarse. Por eso, es necesario que el gestor de implantación cree reglas que habiliten la conectividad entre el entorno de CP4D de no-producción (desarrollo y preproducción) y los servidores de DES y PRE, y la conectividad del entorno de producción con las bases de datos de PRO.

Las credenciales de los usuarios de servicio para las bases de datos se obtienen cada vez que se desea realizar una conexión desde el servicio de Vault (Cajas Fuertes) de CP4D, vía API desde la librería de Python. Esto evita *hardcodear* el usuario y la contraseña en el código. La autenticación en la API de secretos se realiza por medio del bearer token del entorno de ejecución.

5. Implementación

5.1. Metodología de implementación

Los procedimientos de gobernanza de datos son asesorados por Desidedatum, empresa contratada para asesorar al equipo técnico, junto con empleados de IBM que conocen a fondo las posibilidades en materia de gobernanza de datos de CP4D. La empresa contratada cuenta con amplia experiencia en materia de gobernanza pero no en concreto con esta plataforma, por lo que el análisis de cómo poner en práctica la teoría lo realiza el equipo de Tracasa.

El código desarrollado en Python por los ingenieros de datos se empaqueta en una librería independiente para cada caso de uso, por lo que el proceso de carga del repositorio de indicadores tiene su propia librería. Esta se sube a un repositorio Nexus hospedado en Tracasa y al que se otorga conectividad desde CP4D. Así, los entornos en los que se ejecuta el código pueden instalar la librería. El desarrollo de la librería se hace siempre con los principios SOLID en mente, para conseguir un software eficiente, fácil de mantener, limpio y reutilizable por otros casos de uso. Además, el equipo graba todo su trabajo (DDL de SQL, código Python, documentación, etc.) en repositorios GIT en el servidor de Azure DevOps de la empresa y en GesFuentes, un repositorio del Gobierno de Navarra.

El equipo de desarrollo todavía no cuenta con un pipeline de despliegue CI/CD (integración contínua y despliegue continuo), debido a la dificultad añadida que supone integrar esta metodología con CP4D. Sin embargo, el desarrollo del código en Python se realiza con la metodología DevOps siempre en mente, escribiendo un código idempotente y agnóstico al entorno en el que se despliega y ejecuta.

5.2. Proceso de carga

Como se ha mencionado en las secciones anteriores, el proceso de carga se orquesta y ejecuta en por medio de una librería desarrollada en Python, la cual se utiliza en cuadernos Jupyter de Python en entornos levantados en CP4D. En esta sección se hará una revisión a la librería implementada, comenzando con su arquitectura, su uso, el orquestado que realiza y

las utilidades implementadas para el proceso de carga. Después, se estudiarán los subprocesos de carga ETL y de carga a CP4D.

5.2.1. Librería de orquestado y ejecución

i) Estructura de la librería

En este apartado se revisará la estructura de la librería y se explicará brevemente la utilidad de cada uno de los directorios y módulos que la componen, comenzando por la raíz del proyecto.

Figura 12: Estructura de la librería Python

En la raíz de la librería, denominada 'gn_catalogo_indicadores', se encuentra:

- 'ejecucion': que contiene el código que orquesta el proceso.
- 'procesamiento_y_control': que contiene el código que realiza el procesamiento y control, es decir, principalmente los pasos del proceso.
- 'utils': donde se alojan los módulos que contienen las clases y funciones compartidas entre los pasos del proceso.
- '__init__.py': script que se ejecuta al importar la librería. En él se lee el archivo de configuración.
- 'config.yml': archivo de configuración estática. Aunque se puede editar en medio de la ejecución, en principio se usa para almacenar parámetros de la ejecución que son estáticos, por ejemplo, nombres de bases de datos, enlaces de servidores, tablas, peticiones SQL, enlaces de API y demás. Evita *hardcodear* muchos de estos parámetros en el código para que cuando sea necesario cambiar alguno sólo sea necesario cambiarlo en este fichero.

Figura 13: Contenido del directorio 'ejecucion'

El primero de los directorios inmediatamente superiores a la raíz es ejecución, donde se encuentra:

- ‘ejecutor_pipeline.py’: alberga la clase ‘EjecutorPipeline’, que es la clase utilizada para lanzar el proceso después de importar la librería. En la sección siguiente se explicará su funcionamiento a alto nivel y cómo se utiliza.

Figura 14: Contenido del directorio ‘procesamiento_y_control’

En ‘procesamiento_y_control’ se encuentra:

- ‘carga_config_control’: que contiene los módulos que extraen los datos de los ficheros de control a las tablas del esquema del mismo nombre.
- ‘carga_config_goberna’: igual que el anterior, pero para el esquema goberna.
- ‘clases_base’: con las clases utilizadas en el procesamiento.
- ‘pasos’: con los pasos del proceso completo.
- ‘pipeline.py’: en este archivo tenemos el pipeline, que es un diccionario que recoge los subprocesos y pasos, y que es donde se registra el orden de la ejecución.

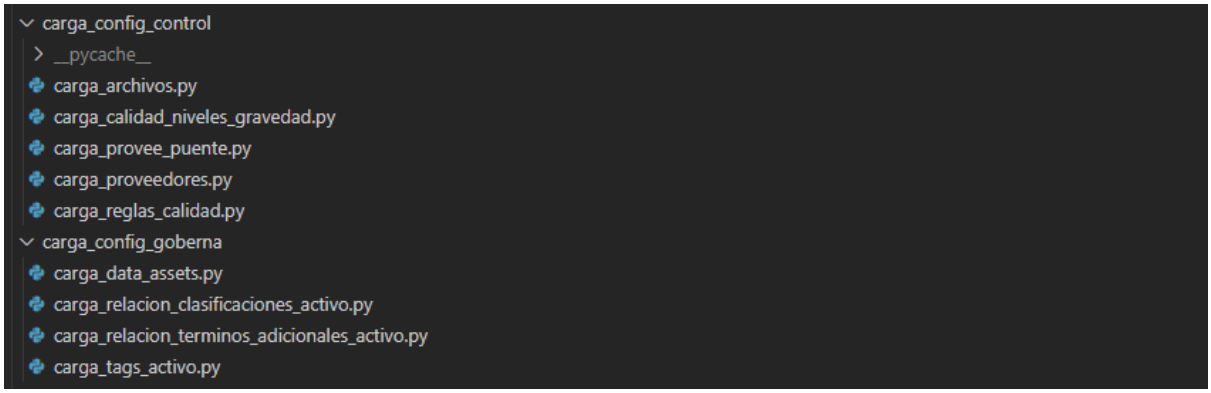

Figura 15: Contenido de los directorios ‘carga_config_control’ y ‘carga_config_goberna’

En los directorios ‘carga_config_goberna’ y ‘carga_config_control’, como se decía, se almacenan módulos para la carga de archivos estáticos de configuración, tanto del esquema “control” como 'goberna”. Cada módulo carga una única tabla (responsabilidad única). Esta

carga es necesario hacerla la primera vez que se ejecuta el proceso, ya que hay que poblar las tablas de control para poder ejecutar el proceso. En cargas posteriores no es necesario a menos que haya que actualizar algún parámetro de configuración.

Figura 16: Contenido del directorio 'clases_base'

En 'clases_base' se encuentra:

- 'asset_datos_subtema.py': con las propiedades o metadatos necesarios para publicar en el catálogo los activos de datos.
- 'parametros_ejecucion.py': que contiene una clase *dataclass* en la que se almacenan parámetros de la ejecución que, o bien se configuran al iniciar el proceso, o bien toman valor durante la ejecución, por ejemplo, los proveedores de indicadores que pasan el control de calidad o el entorno de ejecución, entre otros.
- 'paso_base.py': es la clase de la que heredan todos los pasos, implementa un gran número de métodos comunes y necesarios para muchos de los pasos, por ejemplo, de interacción con la base de datos o con Watson Knowledge Catalog; por ejemplo, ejecutar un comando SQL, leer un fichero CSV o publicar un término de negocio. La mayoría de los métodos funcionan como middleware entre los pasos y los métodos definidos en el código que se almacena en "utils".
- 'termino_empresarial_indicador.py': contiene un objeto en el que se almacenan todos los detalles de los indicadores de términos empresariales (los campos de metadatos que necesita, con sus identificadores internos). También dos métodos para transformar un JSON con una lista de términos en formato diccionario (que es como los representa la API de Watson Knowledge Catalog) a un dataframe de Pandas (que es muy útil para realizar transformaciones y es como se importan las definiciones de términos desde el data mart).

Figura 17: Contenido del directorio 'pasos'

En el directorio de pasos encontramos 8 subdirectorios numerados que corresponden a los subprocesos de la carga. Cada uno de ellos contiene módulos que se corresponden a un módulo cada uno. Los pasos son clases que heredan de 'paso_base' y que contienen, en general, poca inteligencia.

El primero, 'p00_inicio', inicia el proceso de carga leyendo de la base de datos para obtener el identificador de carga (un número entero con el que registrar los logs y marcar los registros para obtener trazabilidad) y la fecha de carga. También valida los proveedores y archivos a leer.

Los subprocesos 1-5 conforman la carga ETL, que se explica en el siguiente apartado. El subproceso 6, una sección más adelante, es la carga a CP4D. El último subproceso registra el final del proceso.

Figura 18: Contenido del directorio 'utils'

En el módulo utils encontramos lo siguiente:

- 'logger': contiene tres archivos, en primer lugar '__init__.py', que sirve para instanciar el logger y así utilizar la misma instancia en todos los módulos; en segundo lugar, un módulo 'CustomLogger.py' donde se encuentra el *logger* en sí, que es una clase que hereda de la clase 'logger' de la librería *Logging* de Python (nativa) e implementa un método que inserta los mensajes de log en la base de datos de *Staging Area*; y por

último ‘EmptyException.py’, que sirve para que sólo se muestre el último nivel de un error y no el traceback completo, lo cual es conveniente para los registros en la base de datos. Se implementan 5 niveles de log: *debug*, *info*, *warning*, *error* y *critical*. Por el momento *warning* y *critical* no se utilizan. *Debug* se utiliza para señalizar la entrada y salida de todos los métodos e instancias de clase. *Info* se usa para señalar pequeños hitos durante la carga, como la creación de activos de datos o actualización de un término de negocio. Por último, *error* se utiliza para señalizar fallos en el proceso de carga y normalmente su mensaje incluye el mensaje de la excepción que lo genera, o en el caso de un error de API el cuerpo de la respuesta. Este *logger* se reutiliza de una librería anterior.

- ‘config.py’: contiene los métodos para leer y escribir la configuración del ‘config.yml’. Como se ha mencionado, en general es configuración estática por lo que el método de escritura se utiliza poco, pero a veces es necesario. El método ‘get_config’ se usa mucho durante el procesamiento. Es una clase muy potente y que se reutiliza de una librería que se hizo anteriormente en el equipo.
- ‘cp4d_utils.py’: contiene el código para interactuar con CP4D, especialmente con la API del Watson Knowledge Catalog, vía una clase denominada ‘GobernaWKC’. Hay métodos para guardar, actualizar, eliminar, consultar artefactos de gobierno, para publicar activos, iniciar enriquecimientos y más. Es una clase muy potente y que será reutilizable entre casos de uso.
- ‘database.py’: contiene una clase ‘database’ con todos los métodos necesarios para interactuar con bases de datos, como leer tablas, ejecutar queries o procedimientos almacenados, truncar tablas y demás. En el resto de casos de uso del equipo se implementan módulos similares que a futuro se estandarizarán para realizar estas acciones de la forma más similar.
- ‘enums.py’: contiene enumeradores, muchos de ellos apuntan a configuraciones de ‘config.yml’ y sirven para que, al escribir una referencia a una configuración, no ocurran errores de escritura (al escribir una referencia almacenada como enumerador, reconoce si el texto existe o no y así se evitan errores de escritura).
- ‘requests.py’: contiene una clase ‘postman’ que se encarga de realizar las llamadas API. En el config.yml se almacenan todos los endpoints, con el método (GET, POST,

DELETE, PATCH, PUT), los *headers* que pide esa API (si pide *content-type* o no, si pide *bearer token*, etc.) y el número de reintentos que se debe permitir. Cuando se lanza una llamada a 'postman', este lee el config y se encarga de toda la inteligencia, devolviendo el cuerpo de la respuesta y el código. También gestiona los errores y en caso de fallo, reintenta el número de veces adecuado, esperando un cierto tiempo entre llamadas. Esta clase facilita mucho el desarrollo y se utilizará en casos de uso posteriores.

- 'secrets.py': contiene métodos para leer secretos de las cajas fuertes de CP4D. No almacena credenciales de ningún tipo, pero simplifica la obtención de secretos, sólo es necesario pasar a los métodos el nombre del secreto a leer. Estos métodos ya se usan en otros casos de uso.

ii) Orquestado y ejecución

Para orquestar y programar flujos de datos (procesos de ingeniería de datos) en Python es muy común utilizar la librería Airflow. Esta librería permite lanzar un proceso Python para que, a las horas programadas, se ejecute un cierto flujo de datos definido. Esto obliga a tener el proceso corriendo, lo cual es muy poco costoso en una máquina (el proceso en segundo plano no ocupa apenas RAM o tiempo de procesador). Sin embargo, en este caso de uso se deben realizar las ejecuciones en CP4D, por lo que tener un proceso activo de Airflow obliga a tener un entorno Python levantado, el cual reserva como mínimo un vCPU (virtual CPU, es la unidad de procesamiento en el clúster, y viene limitado por licenciamiento) y 2GB de RAM (para procesamientos costosos habría que reservar más). Esto lo hace completamente inviable porque se irían implementando nuevos procesos de ingeniería de datos y pronto se acabarían los recursos.

En la sección posterior se explicará cómo programar y ejecutar los procesos Python desde CP4D, por el momento se delega la tarea de programar las ejecuciones a la plataforma y se deja en manos de la librería el orquestado, es decir, el control del flujo de los datos.

Como se ha mencionado anteriormente, la clase encargada del orquestado y el punto de entrada a la librería (el lugar desde donde se inicia el proceso) es EjecutorPipeline. Al iniciar una instancia de la clase, se le deben introducir varios parámetros de configuración: el entorno, la selección de subprocesos (dentro de la librería llamados simplemente procesos),

pasos y proveedores, de archivos a cargar, la fecha de carga (para re-ejecuciones), si hay que cargar los ficheros de configuración, y si se debe impedir que se carguen al data mart los datos de proveedores que no cumplan los requerimientos del control de calidad.

```
dataflow = EjecutorPipeline(entorno="des",
                            seleccion_proveedores="",
                            seleccion_procesos="",
                            seleccion_pasos="",
                            seleccion_archivos_estaticos=None,
                            dt_carga = "9999-12-31",
                            bo_carga_goberna = False,
                            bo_carga_config = False,
                            bo_solo_proveedores_qa_para_dm = True)
```

Entonces se ejecuta el *__init__* y se valida que los procesos, proveedores y pasos existen y que el entorno es desarrollo, preproducción o producción. Además, se instancia el *pipeline* y se ejecutan los *__init__* de todos los pasos (en ese momento básicamente se registra un log de que el paso existe). Después se ejecuta el método 'run()' de la instancia del EjecutorPipeline y esto inicia el flujo de datos.

```
    dataflow.run()
```

El 'run()' del EjecutorPipeline lee los procesos que deben ejecutarse y los va pasando a 'ejecuta_proceso()' pasándole como argumento el código del proceso a ejecutar. Entonces, se recoge en el diccionario que contiene todos los pasos aquellos que corresponden a ese proceso y los va llamando (método '__call__()' de la clase de la que heredan todos los pasos) pasando como argumento el objeto 'parametros_ejecuccion' (que contiene, como se ha dicho antes, información necesaria para la ejecución).

Desde ahí se llama al método 'run()' que es un método declarado en la clase padre como un *abstract method* y que está implementado en cada uno de los pasos (cuando no está implementado se lanza un *NotImplementedError*). En ese método es donde se aloja la inteligencia del paso (las tareas que deben realizarse). Dichas tareas casi siempre se basan en el uso de métodos comunes entre todos los pasos que están implementados en la clase padre, y que normalmente actúan como *middleware* entre el paso y las clases de 'Database', 'Postman', 'GobernaWKC' y 'CustomLogger'. Para referenciar nombres de tablas, endpoints y demás, utiliza los enumeradores mencionados anteriormente (en el módulo 'enums.py'). Este flujo se encuentra resumido en el diagrama de la figura 19.

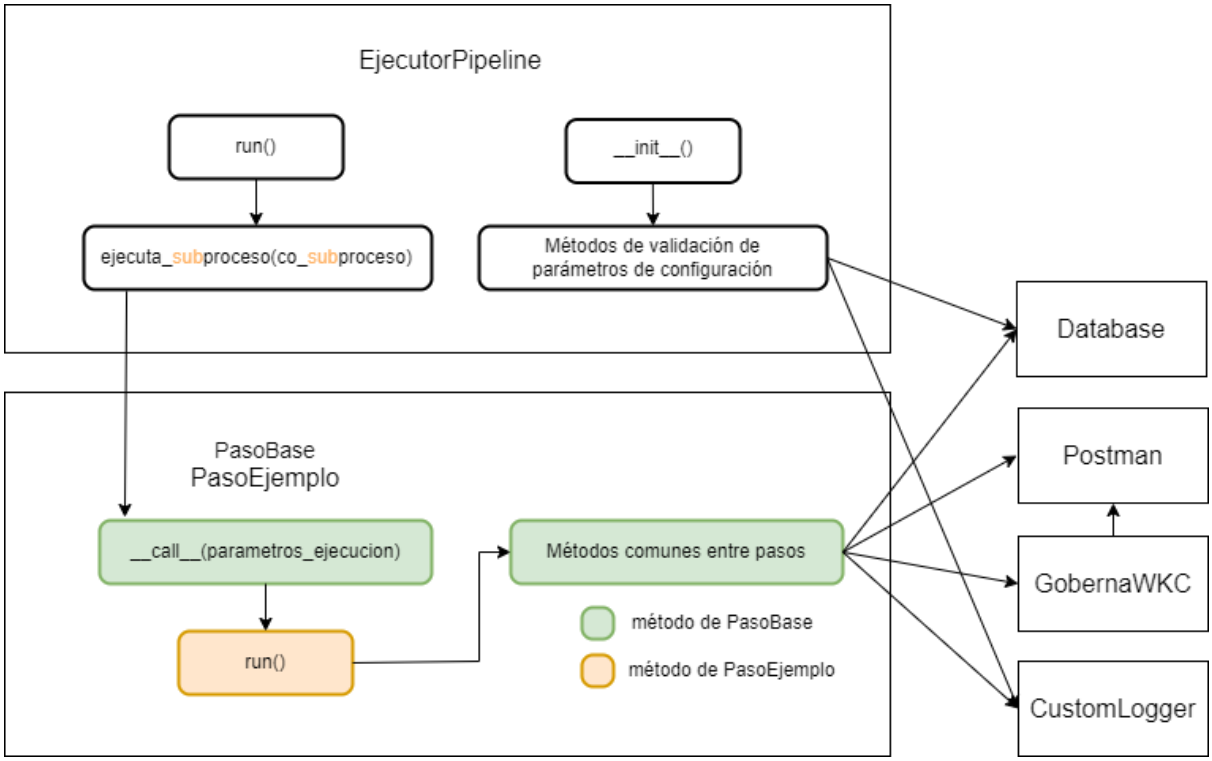

Figura 19: Diagrama de clases para la ejecución del proceso de ingeniería

5.2.2. Carga ETL

El proceso de ETL, que comienza desde la extracción de los ficheros CSV hasta la carga del data mart, se divide en pasos de responsabilidad única. Es decir, cada paso (y por tanto cada módulo Python del directorio de pasos) únicamente lee, modela y escribe una sola tabla. A continuación, se enumeran los subprocesos de la ETL y se explican las particularidades de la implementación de cada uno de ellos, sin entrar a gran detalle de las transformaciones que se han realizado para cargar cada tabla.

1) Extracción

Los pasos de extracción no deberían implementar ningún tipo de transformación o limpieza de los datos; se trata de hacer una simple copia de los datos origen en el sistema de información sobre el que se va a trabajar. Para cada proveedor debe haber extracciones independientes. Además, hay extracciones de tablas que contienen datos comunes entre proveedores, algo análogo a datos maestros. En este caso tenemos tablas del primer proveedor, Nastat, y una tabla de municipios que es transversal (no es de un proveedor).

Las extracciones se realizan leyendo los ficheros CSV desde Pandas con el método 'read_csv()', en general se permite que la librería y en algunos casos se especifica el tipo de la columna (particularmente para booleanos), y escribiendo los *dataframes* desde Pandas mediante el motor de SQLAlchemy (conector de SQL Server).

2) Transformación

La fase de transformación también es independiente para cada proveedor. Estos pasos, como en cualquier ETL que termina en *data mart*, deben realizar el modelado dimensional y la tabla de hechos. Se debe generar para cada proveedor una tabla de hechos con los valores de los indicadores y una dimensión de definición de indicadores.

En las dimensiones indicadores deben estar los campos de los metadatos del estándar descrito en la sección de análisis, aunque algunos de ellos no puedan ser cumplimentados porque no están definidos en los datos de origen. Las DKs se generan concatenando la cadena "dk#" con el código del indicador, que es el código del proveedor concatenado al ID del indicador en el sistema origen, por ejemplo "NST00213" para el indicador 213 de Nastat".

Además, hay dos dimensiones que son transversales, es decir, que no se generan a partir de un proveedor: la dimensión fecha y la dimensión municipios. Estas tablas se tratan como parte de un proveedor denominado 'común'. La dimensión municipios es prácticamente igual que la tabla extraída de municipios, con el campo para la DK que toma el formato dk#[código_municipio]. La dimensión fecha se genera a partir de un procedimiento almacenado. Sus DKs se generan concatenando la cadena "dk#" con el año, mes y día de la fecha. El resto de transformaciones se realizan en Python usando las utilidades de Pandas.

3) Control de calidad

El control de calidad de la ETL sirve para impedir que datos que no cumplan con unos requerimientos mínimos suban al *data mart*. Esto permite que, aunque un proveedor deposite datos mal generados, el resto de proveedores no se vean afectados y los productos finales sigan conteniendo el resto de indicadores.

Este control se realiza generando reglas de calidad mediante procedimientos almacenados que al ejecutarse devuelven "OK" o "KO" según si el resultado pasa o no pasa la regla. Además, se registra en la tabla de reglas de calidad si la regla es crítica o leve, en el caso de las leves para registrarlo, pero no impedir su carga, y en el caso de la crítica para no cargar los

indicadores del proveedor. Dichos procedimientos almacenados se inician desde Python, pero evidentemente la ejecución tiene lugar en el propio SQL Server.

4) Historificación y conformación

A partir de esta fase sólo se manipulan los datos de los proveedores que han pasado el control de calidad. Esta fase tiene dos objetivos: historificar las definiciones de indicadores (mediante SCD2) y conformar la dimensión final (que aúna las definiciones de todos los proveedores). El SCD2 se realiza mediante la instrucción *merge* de SQL Server, y se hace usando un procedimiento almacenado. Una vez se federan las definiciones de indicadores se generan también las SKs (que son enteros).

La historificación permite que haya varias filas con la misma DK que registran el mismo indicador que ha cambiado con el tiempo. Es importante señalar de qué fecha a qué fecha es válido un registro (y por simplicidad, un booleano que diga si el registro está vigente o no). Estas filas quedan identificadas por la SK, que siempre es diferente. En resumen, no puede haber dos registros con la misma SK, pero sí con la misma DK, cuando representen el mismo registro dimensional cambiando con el tiempo. Para evitar la ambigüedad, la tabla de hechos del data mart apunta usando las SKs, no las DKs (además es computacionalmente más eficiente porque las DKs son cadenas de texto y las SKs son enteros).

5) Precarga

El paso de precarga tiene la responsabilidad de sustituir las DKs por las SKs en la tabla de hechos. Esto permite que un hecho (en este caso de uso, un valor de un indicador) apunte a una sola definición de indicador. Este paso se realiza en el *Staging Area* para no cargar la base de datos final, a la que acceden los usuarios. Esto es una buena práctica, pero no siempre se hace. El cruce de las DKs por las SKs se realiza usando un procedimiento almacenado.

6) Carga a *data mart*

Sólo copia las dimensiones y la tabla de hechos finales en la base de datos *Data Mart* sin implementar ninguna inteligencia. La copia de las tablas se hace usando Pandas. Adicionalmente, se crea un esquema “publicacion” en el que se cargan los datasets que serán publicados en CP4D, que combinan los datos y las definiciones para ser cómodos de usar por los analistas cuando descarguen los ficheros del catálogo (para evitar que tengan que unir datasets de definiciones y valores).

Como se ha visto, gran parte del procesamiento de datos se realiza usando Pandas, aunque también se ha mencionado anteriormente que CP4D permite levantar entornos con Spark para utilizarse desde Python. Sin embargo, el volumen de datos que se trabaja es bastante reducido en este caso de uso (por el momento, la mayor parte del volumen viene de indicadores municipales, y se trata de una tabla de 6 columnas con 300.000 registros). Por lo tanto, Pandas es más que suficiente; en otros casos de uso con volúmenes grandes de datos sí es recomendable utilizar Spark, que es muy útil para casos de *Big Data*.

5.2.3. Carga a CP4D

Una vez creado el data mart es momento de publicar los artefactos de gobierno y datasets en el catálogo. Los pasos necesarios para dicha carga, ejecutados en la librería Python, utilizan principalmente la API de Watson Knowledge Catalog (Watson Data API). En la figura 20 se resumen los pasos necesarios para realizar esta carga.

Figura 20: Pasos de la carga de datasets y términos de negocio de indicadores a CP4D

1) Creación de activos de datos:

Se decide que los indicadores se van a publicar en datasets diferenciados por su temática. En concreto, por el campo de subámbito, que por el momento está heredado de los temas-subtemas de Nastat, y en el que encontramos 42 categorías como Nupcialidad, Climatología o Innovación. De las cuales por el momento sólo hay indicadores de 32 de ellas.

La librería Python lee la tabla del esquema "publicacion" que contiene los datos y definiciones de los indicadores juntos y usando Pandas genera ficheros CSV por cada

subámbito que contenga algún indicador. Una vez creados, están almacenados en el entorno de ejecución y hay que registrarlos como activos de datos. Esto se hace mediante la librería watson-studio-lib, que viene instalada por defecto en el entorno. Una vez registrados aparecen en el frontal de CP4D como activos normales y tienen un identificador con el que consultarlos, modificarlos o publicarlos mediante la API.

2) Publicación de términos de negocio

El segundo paso de la carga a Goberna consiste en publicar el glosario de negocio de indicadores, que como se ha mencionado, es un término de negocio por cada definición de un indicador. Se debe tener en cuenta que cada vez que se desee realizar la carga de las definiciones se tienen que tener en cuenta los artefactos que ya existen: no se desea eliminar el glosario de indicadores y cargarlo de 0 porque entonces se perderían las relaciones con otros artefactos o activos que se hayan creado de forma manual por los *data stewards* de la plataforma.

Por eso, para cargar los términos, primero se debe consultar qué indicadores existen y compararlos con los que existen en el data mart. Así, se realizan 3 acciones en la carga:

- Crear aquellos indicadores que aún no existen en el glosario.
- Eliminar aquellos indicadores que existen en el glosario y que no aparecen en el data mart (porque han sido eliminados del origen de información y por tanto no tendrán datos relacionados).
- Actualizar aquellos que existen en el glosario y en el data mart pero que cuyos metadatos han cambiado.

El flujo necesario para realizar estas acciones se resume en la figura 21.

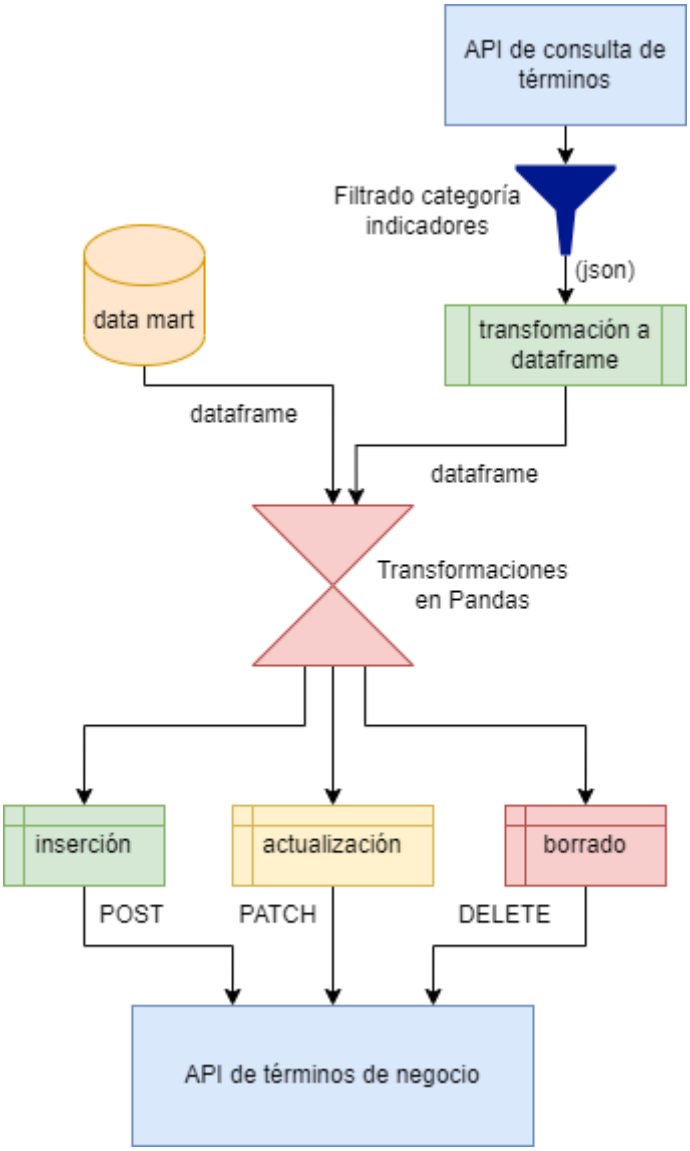

Figura 21: Diagrama de la carga de términos de negocio de indicadores a CP4D

Aquellos indicadores que existen y no han cambiado deben mantenerse inalterados. Para comparar lo que existe se consultan todos los términos del glosario. No se pueden consultar únicamente los de la categoría “indicadores” porque para hacer ese filtro se debe usar la API de búsqueda global y sus resultados no muestran los detalles (metadatos) de los indicadores. Después de leer todos se filtra por categoría y se descartan lo que no sean indicadores.

A continuación, se transforma el JSON de indicadores en un *dataframe* de Pandas, y se compara con el *dataframe* que se genera al leer la tabla “Catalogo_Indicadores” del esquema “precarga”. Por medio de transformaciones que comparan los indicadores actuales con los de la nueva carga se obtienen las filas nuevas, las filas a eliminar y las filas a actualizar en

dataframes separados. Se utilizan métodos diferentes para cada acción en la clase 'GobernaWKC', y métodos de petición API distintos para cada una. La comparación se hace utilizando el identificador que se les otorga durante la ETL, que concatena el proveedor con el ID del indicador en el sistema origen.

Los *dataframes* de filas nuevas y de actualización se transforman de vuelta al formato de JSON que reconoce la API y se cargan. Las filas nuevas se pueden cargar en *batches* de hasta 100 indicadores por lo que su carga es rápida. Las actualizaciones deben realizarse de una en una, usando el método PATCH de la misma API, por lo que es más lento, pero como las definiciones de indicadores no varían demasiado con el tiempo, en general no debería ralentizar la carga. Por último, se eliminan los indicadores que sobran de uno en uno.

3) Enriquecimiento de activos de datos

El proceso de enriquecimiento de metadatos se hace utilizando un activo de proyecto llamado Área de Enriquecimiento de Metadatos (MDEA). Este activo llega a la fase de despliegue y se configura vía API. En este paso los activos que se han creado en el primer paso son añadidos al MDEA utilizando la misma API y usando sus identificadores (que se obtienen de la creación).

Una vez añadidos se inicia el enriquecimiento de metadatos: se pasa un control de calidad (por el momento se deja con los valores por defecto de la plataforma) y se realiza un perfilado de los datos. Este perfilado coge una muestra de los datos y calcula estadísticas que pueden o no ser útiles, como el rango donde se concentran los datos de una columna numérica o el valor más repetido de una columna de texto.

Este proceso es asíncrono, es decir, se envía la petición de inicio del enriquecimiento vía API y no se espera a que haya terminado para devolver la respuesta (puede durar varios minutos). La respuesta contiene el identificador de la ejecución del trabajo, y por medio de otro *endpoint* de la API se consulta su estado cada 15 segundos hasta que haya terminado.

Una vez finalizado, se crean las relaciones entre los términos de negocio de indicadores y los activos de datos enriquecidos que los contienen. Se hace también con la Watson Data API y se añaden todos los términos de golpe para cada activo (una llamada API por activo). Al finalizar, los *datasets* están preparados para la publicación.

4) Publicación de activos de datos

La publicación de activos de datos es un proceso asíncrono que también se inicia con la Watson Data API. Se realiza una sola llamada para publicar todos los activos de golpe: la publicación se haría de uno en uno si se hiciese de forma normal desde el proyecto, pero al tenerlos agrupados en un MDEA, tenemos la posibilidad de publicar todos los términos del MDEA por lo que se ahorra algo de tiempo.

Una vez publicados los activos el proceso de carga finaliza, se *loggea* su éxito (o error, en caso de haber fallado en algún punto anterior) y el entorno de Python se destruye. En CP4D encontramos: 32 activos de datos publicados en catálogo y 1400 términos de negocio en la categoría “Indicadores”.

5.2.4. Ejecución del proceso en CP4D

La ejecución del proceso descrito en *Cloud Pak for Data* se realiza combinando varios componentes que ya se han mencionado a lo largo de este trabajo. El primero de ellos son los cuadernos Jupyter, que permiten ejecutar código Python en CP4D; para trabajar en un cuaderno se levanta un entorno con Python instalado en el que se debe instalar la librería desarrollada.

El acceso al repositorio Nexus de Tracasa en el que se encuentra la librería se controla por medio de autenticación con usuario y contraseña. Para obtenerlos se utilizan los secretos de CP4D, de igual forma que para el acceso al SQL Server (para evitar escribir credenciales en el código). Se agrega en el proyecto el módulo Python de la librería que sirve para consultar secretos, y este se añade al entorno Python, ‘secrets.py’. Una vez ahí se importan los métodos en Python por ruta y se puede realizar la consulta a los secretos. Con los secretos importados se instala la librería desde Nexus y ya está lista para ser utilizada.

En la sección anterior se cuenta que la clase que inicia la ejecución en Python pide varios parámetros de configuración, como el entorno o los subprocesos a ejecutar. Estos parámetros deben pasarse al cuaderno de trabajo de forma externa (en las ejecuciones programadas no va a haber nadie configurando a mano el proceso). Para programar la ejecución se crea un

trabajo del cuaderno, pero a este *job* no se le pueden pasar parámetros de forma directa. Sin embargo, en la versión en uso de la plataforma los *pipelines* sí pueden pasar parámetros de entrada a los trabajos de los cuadernos, haciendo que en ellos aparezcan como variables de entorno. Estos parámetros se almacenan y configuran en un *parameter set* dentro del proyecto.

Por esa razón, aunque los *pipelines* normalmente se utilizan para orquestación de procesos (lo cual se hace en la librería), en esa ocasión sólo funcionan como un puente entre los parámetros de configuración y la librería Python. A partir del *pipeline* se crea un trabajo que es programado para ser ejecutado a diario.

5.3. Despliegue

Los despliegues en los sistemas de Gobierno de Navarra conllevan ciertos procedimientos que, aunque en ocasiones pueden ser latosos, ofrecen un cierto nivel de confianza a los gestores de proyectos y administradores. Para ello existe la figura del gestor de despliegue, que utilizan usuarios que tienen acceso a los servidores de producción, a diferencia de los desarrolladores, y que se encargan de ejecutar todas las acciones necesarias para realizar el despliegue de bases de datos.

El despliegue del sistema comienza con la creación de las bases de datos y de todos sus esquemas, tablas, claves primarias/foráneas, procedimientos almacenados y cualquier otro elemento de SQL Server utilizado por el sistema final. Para ello se generan scripts de T-SQL con las DDL, los cuales se ejecutan en el servidor de desarrollo, preproducción y producción y aseguran que el modelo de datos y el funcionamiento de las bases de datos es idéntico. Además, se pide la creación de un usuario de servicio en todas las bases de datos que tiene permisos de lectura, escritura y *alter* (este último es necesario para truncar tablas, acción que se realiza en varios puntos durante los pasos de carga). Una vez creadas las bases de datos se debe desplegar el sistema en la plataforma.

Se han instalado dos entornos para la plataforma CP4D, uno no productivo (denominado en ocasiones no-pro) y otro productivo, al que acceden los usuarios y sistemas finales. En el primero se realiza el desarrollo, y ahí se ensayan los despliegues de producción en dos entornos simulados, desarrollo y preproducción; como no hay una separación física entre los dos entornos se buscan maneras de que coexistan los procesos. El despliegue en CP4D

requiere una serie de tareas, objetos y configuraciones, parte de las cuales realizan manualmente los gestores de despliegue. El resto se automatizan usando cuadernos Jupyter para ahorrar trabajo a los gestores. Las acciones necesarias son las de la tabla 8:

Acción	Tipo	Observaciones
Creación del catálogo	Manual	No se puede automatizar por limitación de la API. Permite crear el catálogo, pero no permite asignarle un espacio de almacenamiento en el clúster por lo que queda inutilizable. La creación debe ser manual por el interfaz de usuario.
Importación del proyecto	Manual	Sólo se puede hacer manualmente.
Creación del volumen de almacenamiento	Manual	Se puede crear vía API, pero los volúmenes son de tipo *Persistent Volume Claims* (PVC) en el openshift, y una vez creados no se puede liberar la memoria que reservan de forma sencilla. Por ello, y ya que esta es una tarea que sólo se hace una vez, es preferible que la realice un gestor de despliegue.
Ejecución del cuaderno de despliegue	Manual	Debe ser manual porque es lo que ejecuta la parte automática del despliegue.
Creación de la categoría	Automática	Creación de la categoría 'Indicadores' en la que se publican los términos de negocio.
Obtención del identificador del proyecto y catálogo del caso de uso	Automática	Se hace con consultas vía API, estos identificadores se almacenan como parámetros que son necesarios en la ejecución de la carga.
Creación de parámetros dentro del *parameter set*	Automática	En el parameter set existen varios parámetros que deben ser configurados, como el endpoint de la plataforma, el entorno de ejecución y varios identificadores. Esto se hace editando el fichero interno en el que se almacena el set de parámetros.
Configuración del MDEA	Automática	Debe configurarse el área de enriquecimiento de metadatos, que ya viene creada con la importación del proyecto pero no puede ser configurada hasta una vez creada la categoría.

Creación de la conexión al volumen	Automática	Para habilitar acceso a los archivos origen. Se hace vía API.
Creación de los secretos	Automática	Se crean vía API. Se crean vacíos porque no se puede almacenar credenciales en el código (y no son conocidas por los desarrolladores)
Creación de atributos personalizados de activos de datos	Automática	Se deben definir mediante un archivo JSON que se sube vía API. Permite que en los activos de datos publicados aparezcan campos de metadatos como el autor o la temática. Esto crea los campos, cumplimentarlos es posterior (al publicar cada activo).
Creación de atributos personalizados de términos de negocio	Automática	Similar al anterior pero no se hace por medio de un archivo, sino que los campos se definen en el cuerpo de la petición API.
Configuración de los secretos	Manual	Se insertan las credenciales en los secretos y se da acceso al grupo de LDAP 'Gestores de Despliegue' (por si el gestor cambia).
Control de acceso a catálogo	Manual	Habilitar control de acceso con rol de visor al catálogo de indicadores al grupo 'Usuarios mínimos' (grupo en el que se incluye a todos los usuarios y que permite ver datos públicos).
Control de acceso a proyecto	Manual	Habilitar control de acceso con rol de Administrador a 'Gestores de Despliegue'.
Ejecución de prueba del proceso de carga	Manual	Es manual porque se necesita comprobar que el proceso es completado con éxito antes de programarlo para ejecutar diariamente (de noche para en horario de oficina esté disponible). En esta primera ejecución se deben cargar los archivos estáticos (los que pueblan las tablas de configuración del proceso, que definen qué se publica, los proveedores que existen, la localización de los archivos en el volumen de almacenamiento, las reglas de calidad, etc.).
Verificación	Manual	Comprobar que las publicaciones de términos y activos son correctas (activos

		enriquecidos, descargables, relacionados a los términos de negocio, y estos están bien definidos).
Programación de la ejecución diaria	Manual	Es manual porque es tarea del gestor que no se debe delegar. Se programa el trabajo del pipeline de carga (el que contiene el cuaderno Jupyter en el que se instala gn_catalogo_indicadores).

Tabla 8: Acciones necesarias para el despliegue del caso de uso en producción

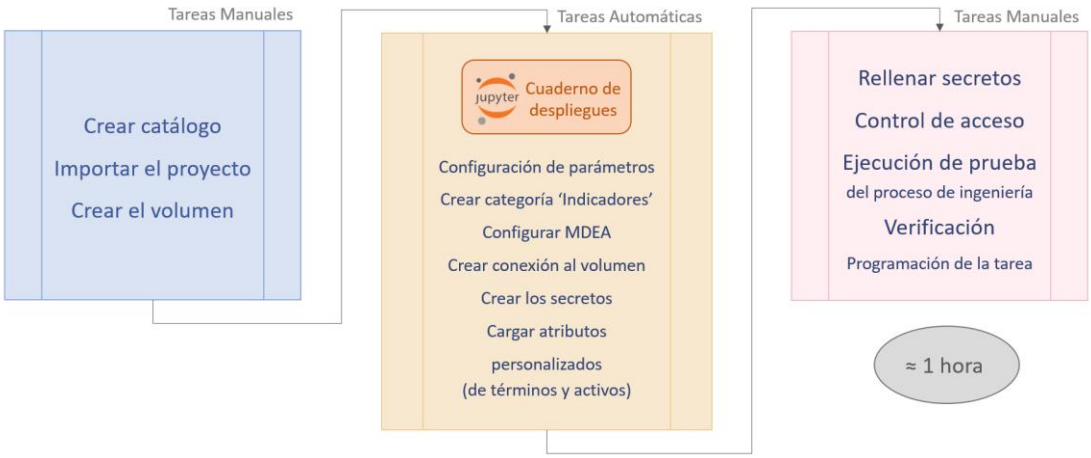

Figura 22: Secuencia de las acciones realizadas durante el despliegue

En la figura 22 se resumen los pasos mencionados. El proyecto que se importa (se genera primero en el entorno no productivo y se lleva al de producción) incluye varios activos: el módulo *secrets.py*, dos trabajos de pipelines, dos trabajos de cuadernos Jupyter, un MDEA, un conjunto de parámetros y dos archivos JSON con los atributos customizables de términos y activos de datos. El resto de activos se generan durante la ejecución del cuaderno de despliegue y durante el proceso de carga.

5.4. Monitorización de los procesos programados

Cuando el sistema está desplegado en producción, es importante monitorizar el estado de las ejecuciones. Para ello CP4D ofrece de base alertas básicas que, al entrar informan de las ejecuciones exitosas y fallidas que han tenido lugar desde el anterior *login*. Además, existe la

pestaña ‘Trabajos’ en la que se pueden filtrar las ejecuciones para comprobar el estado de cierto trabajo.

Figura 23: Captura de pantalla de las alertas de CP4D

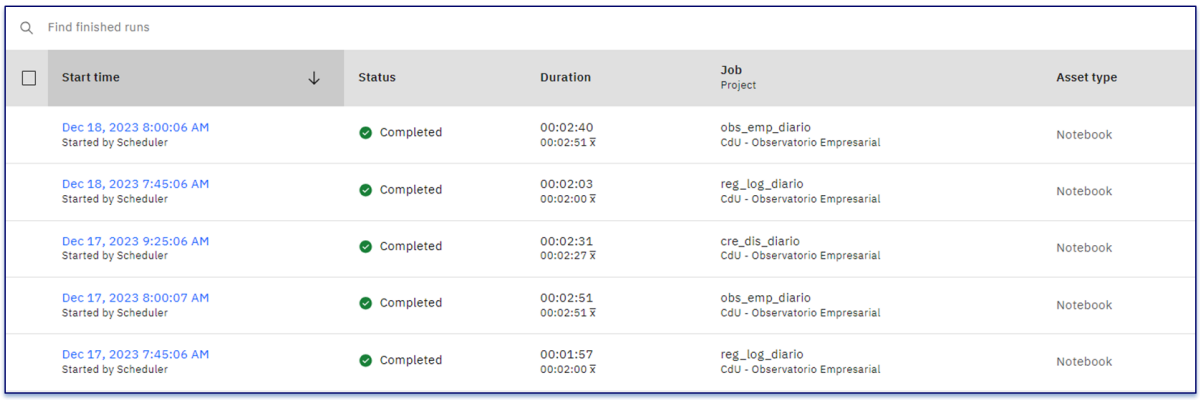

Figura 24: Captura de pantalla de la pestaña ‘Trabajos’ en CP4D

La situación no es idónea porque debe haber alguien encargado de ir comprobando las ejecuciones de forma casi manual (sí, CP4D informa, pero igualmente obliga a entrar, y los encargados de monitorizarlo no van a ser usuarios habituales). Para solucionarlo, CP4D ofrece la capacidad de integrarse con servidores SMTP a través de los cuales enviar correos electrónicos a ciertos grupos de usuarios en base a situaciones (estado de las ejecuciones, errores, información relevante, como activos publicados, etc.). El servidor SMTP necesario ya existe y se usa en otros sistemas en el Gobierno de Navarra. Esta integración, a enero de 2024, todavía está en cola porque los instaladores de la plataforma tienen otras tareas, así que por el momento no está disponible.

6. Resultados

La implementación y despliegue de este caso de uso resulta en tres productos de datos: un data mart con datos y valores de indicadores en modelo estrella, 32 activos de datos descargables (CSV) con definiciones y valores de indicadores separadas por temáticas, y un glosario de negocio con todas las definiciones de indicadores en un modelo estándar de metadatos, el cual forma relaciones con los activos de datos para facilitar el descubrimiento de indicadores para los usuarios de la administración pública. Además, dichos activos de datos publicados en catálogos de CP4D están gobernados: están contextualizados, tienen análisis de calidad, perfilado de datos, y control de acceso.

El proceso de ingeniería de datos (proceso de carga) para generar dichos productos es desplegado en el entorno de producción por un gestor de despliegue (y las bases de datos por un gestor de implantación); dicho despliegue tiene lugar después de dos ensayos en preproducción (el primero fallido) y toma alrededor de 1 hora. En el futuro la duración del despliegue se reducirá debido a la mayor experiencia de los gestores con la plataforma CP4D.

La duración del proceso de carga es de alrededor de 20 minutos cuando se lanza con nivel de log *debug*, porque se insertan en la base de datos cientos de mensajes relativos a la ejecución, lo cual genera muchas conexiones con la base de datos que consumen tiempo. Cuando se ejecuta este proceso con nivel de *error*, el tiempo se reduce a alrededor de 13 minutos (si el proceso no falla en principio no hay por qué registrar mensajes en la base de datos). La duración del proceso de carga sin tener en cuenta la escritura de logs varía en función de muchos factores.

En buena medida se debe a lo que se deba hacer con los términos de negocio; el peor caso posible sería tener que actualizar las definiciones de todos los términos, ya que se hace de uno en uno y esto es costoso, y el mejor caso es cuando no ha cambiado nada y no se deben insertar, actualizar ni eliminar términos de negocio. La duración también depende de factores externos al proceso de carga, como el estado de los SQL Servers (que afecta principalmente a la ETL) y del propio CP4D (que afecta a la carga a CP4D por los tiempos de respuesta de la API). En principio el entorno de Python no debería verse afectado por lo cargado que esté CP4D porque reserva recursos que no comparte con otros procesos.

La primera versión del caso de uso se considera un éxito por tres motivos:

- A nivel funcional cumple con los requerimientos iniciales: se trata de un sistema que permitirá la unificación y consolidación de fuentes variadas de indicadores en la administración pública. Estos objetivos se alinean con la estrategia de datos del Gobierno de Navarra.
- Permite cumplir los hitos señalados por la oficina del dato en relación a la publicación de activos de datos gobernados.
- Su análisis, diseño y desarrollo son experiencias muy valiosas para el equipo. Al ser el primer caso de uso desplegado, sirve para estandarizar las implementaciones, conocer la plataforma CP4D, aprender los procedimientos y tecnologías usadas en Gobierno de Navarra y tener una referencia de cómo realizar un despliegue.

Además de los éxitos a nivel interno, desde IBM consideraron que el trabajo realizado tiene mucho valor, y por ello se recibió una invitación para presentar el caso de uso en el IBM TechXchange Summit que tuvo lugar en Barcelona del 23 al 25 de enero de 2024. En el evento se tuvo la oportunidad de compartir el trabajo invertido en este caso de uso durante el pasado año, y de aprender de las experiencias de otros clientes y consultores de IBM sobre sus propios proyectos de datos.

7. Conclusión

En este trabajo de fin de máster se ha abordado la tarea de diseñar e implementar un repositorio de indicadores gobernado para el Gobierno de Navarra. El objetivo de este sistema es centralizar y unificar múltiples fuentes de indicadores, comenzando con Nastat como proveedor piloto, para ofrecer a los usuarios de la administración pública unos productos de datos que faciliten el descubrimiento y la utilización de datos de indicadores, y así facilitar la toma de decisiones basada en datos.

Para ello se ha realizado un análisis del problema a resolver y de las posibilidades tecnológicas en relación a procesamiento y gobernanza de datos de *Cloud Pak for Data.* Se resume el proyecto en tres productos de datos: un data mart en SQL Server con definiciones y valores de indicadores, activos de datos descargables publicados en catálogos de CP4D y un glosario de negocio (formado por términos de negocio) con las definiciones de los indicadores.

Se ha diseñado un modelo de datos que unifica las definiciones de indicadores, teniendo en cuenta el estándar de metadatos desarrollado para este caso de uso. Para resolver el proceso de ingeniería de datos se ha implementado una librería de Python que orquesta y ejecuta la ETL (carga del data mart) y la carga a CP4D (activos y términos), apoyándose de los objetos y servicios disponibles en la propia plataforma. Dicho proceso se ha desplegado en producción con éxito y está programado para ejecutarse diariamente.

La primera versión de este caso de uso es un éxito, no solo por cumplir con los requerimientos solicitados a nivel funcional, sino porque ha permitido alcanzar los hitos de la Oficina del Dato en materia de publicación de activos de datos y ha aportado una experiencia muy valiosa para Gobierno de Navarra y para el equipo de Tracasa.

8. Trabajo futuro

Como se puede intuir, el caso de uso del repositorio de indicadores no se trata de un trabajo único, sino que requiere incluir nuevos proveedores conforme estos se dispongan. A corto plazo se espera poder incluir indicadores de varios departamentos, como Educación o Hacienda, y también otros organismos como el Servicio Navarro de Empleo.

Para este último hay un caso de uso de cálculo de indicadores, por lo que hay una clara sinergia, y el data mart que se está desarrollando para ellos ya tiene los campos del estándar común de metadatos por lo que incluirlo en el repositorio será sencillo. Con cada inclusión de un nuevo proveedor se debe desarrollar su parte específica de la ETL (extracción y transformación). Además, se prevé que nuevos proveedores puedan incluir desagregaciones nuevas, por lo que requerirán crear dimensiones de desagregación y el modelo de datos se irá actualizando.

A corto plazo también se espera realizar publicaciones de los datos del data mart virtualizados en los catálogos (y el Marketplace). Esto permitirá centralizar el control de acceso en CP4D (no habrá que establecerlo en SQL Server). También permitirá utilizar la API de virtualización de CP4D para ofrecer un servicio de indicadores a demanda (aún se están estudiando las limitaciones de la API de virtualización en rendimiento y concurrencia).

En otra línea de trabajo se están desarrollando dos librerías Python que se utilizarán en las siguientes versiones del caso de uso. La primera, denominada dentro del equipo por el momento 'común', pretende estandarizar los modos de desarrollar el código, tanto temas de orquestado, como de peticiones API o accesos a bases de datos, logging y demás utilidades que son comunes entre los casos de uso. Esta librería servirá de apoyo a las librerías específicas de cada caso de uso, por lo que en la de indicadores algunos módulos quedarán obsoletos y se eliminarán.

La segunda librería reúne código para los despliegues. En el despliegue de la primera versión de este caso de uso se utilizó un cuaderno Jupyter con bastantes funciones que facilitan diversas tareas. Estas y otros métodos nuevos se reúnen en la librería de despliegues, que idealmente contendrá todas las funcionalidades que cualquier caso de uso nuevo necesite. Cuando estas dos librerías estén completas, se deberá refactorizar el código para utilizarlas. Esto permitirá que el caso de uso pueda ser mantenido por cualquier compañero del equipo sin necesidad de estudiar el código en profundidad.

9. Bibliografía

[1] *Estrategia Digital Navarra 2030*. (2020, October 1). Estrategia Digital Navarra 2030. https://www.navarra.es/documents/48192/5159030/EDN2030.pdf/5fc6912e-27fd-27e0-f245-6a20259f0c3c?t=1601551423827

[2] *Acerca de los elementos de trabajo y los tipos de elementos de trabajo - Azure Boards*. (2024, January 4). Microsoft Learn. https://learn.microsoft.com/es-es/azure/devops/boards/work-items/about-work-items?view=azure-devops&tabs=agile-process

[3] Hennigan, L. (2023, April 24). *What Is A KPI? Definition & Examples*. Forbes. https://www.forbes.com/advisor/business/what-is-a-kpi-definition-examples/

[4] Cheong, L. K., & Chang, V. (2007). The Need for Data Governance: A Case Study. *ACIS 2007 Proceedings*, 100.

Anjana Data Platform. (n.d.). Anjana Data. https://anjanadata.com/en/product/

[5] Olson, J. E. (2003). *Data Quality: The Accuracy Dimension*. Elsevier Science.

[6] Inmon, W. H. (2005). *Building the Data Warehouse*. Wiley.

[7] Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley.

[8] *Top Trends in Data and Analytics for 2021: Data Fabric Is the Foundation*. (2021, February 16). Gartner. https://www.gartner.com/en/documents/3996983

[9] *Top Informatica Competitors & Alternatives 2024*. (n.d.). Gartner. https://www.gartner.com/reviews/market/data-integration-tools/vendor/informatica/alternatives

[10] *Governance artifacts - IBM Knowledge Catalog*. (n.d.). IBM. https://www.ibm.com/docs/en/cloud-paks/cp-data/4.8.x?topic=governance-artifacts