*Maria Gimenez Costa, Camila Bernardez y Micaela Oliva*

Tecnología Digital VII: Ingeniería de Datos [Beiro]

12 de Julio del 2024

# Trabajo Práctico Grupal (2da Parte)

Arquitectura y flujo de datos

# Arquitectura

## Origen, cadencia, formato y volumen de los datos

La organización presentada en la primera entrega era una biblioteca con dos tipos de datos principales para ingestar: los libros que adquiere por compras o donaciones, y los eventos que surgen de transacciones web (registro de usuarios, reserva de libros, renovaciones, etc). Los primeros llegan cada una cantidad de tiempo determinada, mientras que los segundos arriban constantemente y necesitan ser atendidos con una frecuencia alta (por ejemplo, si se quiere dar de alta una nueva cuenta o reservar un préstamo).

En el siguiente cuadro detallamos los datos con los que trabaja la biblioteca, incluyendo el origen de los mismos, la cadencia con la que arriban y en qué volumen, y el formato en el que arriban y deberíamos leerlos. Además detallamos los procesos intermedios que hacen usos de estos datos, así como quienes tienen acceso a los mismos.

| Datos | Origen | Cadencia llegada | Cadencia carga | Formato | Volumen | Usos en procesos | Roles |
|---|---|---|---|---|---|---|---|
| Libros (adq) | Proveedores /donaciones | mensual | mensual [1] | xlsx + json | 252/253 [2] | Datos de origen para la API de ISBN<br>Lista de idiomas disponibles para un libro (API) | bibliotecario → permiso de lectura y carga de facturas, NO de escritura |
| Información de libros a partir de ISBN | APIs | mensual | mensual | json | 202/203 | Donde corresponde guardar un nuevo libro.<br>Categorizar libros por idioma y por edades.<br>Actualización de stock | bibliotecario → lectura de datos<br>escritura sobre categorización por edades (en caso de error) |
| Ejemplares (adq) | Proveedores /donaciones | mensual | mensual | xlsx + json | 50/51 existentes, 202/203 nuevos | Para calcular los ejemplares disponibles. | |
| Audiolibros (adq) | Proveedores | trimestral | trimestral | xlsx + json | 20 | Actualización de stock. | |
| Reservas | gestión reservas (web) | continua | inmediata | json | 1000 | Reservas pendientes.<br><br>Reservas activas | |
| Préstamos (inc. renovaciones) | gestión préstamos (web) | continua | inmediata | json | 1000 | préstamos activos → Para calcular ejemplares disponibles<br><br>Reserva activa + préstamo activo = aviso de solicitud | bibliotecario → escritura para cancelar una multa ya paga |

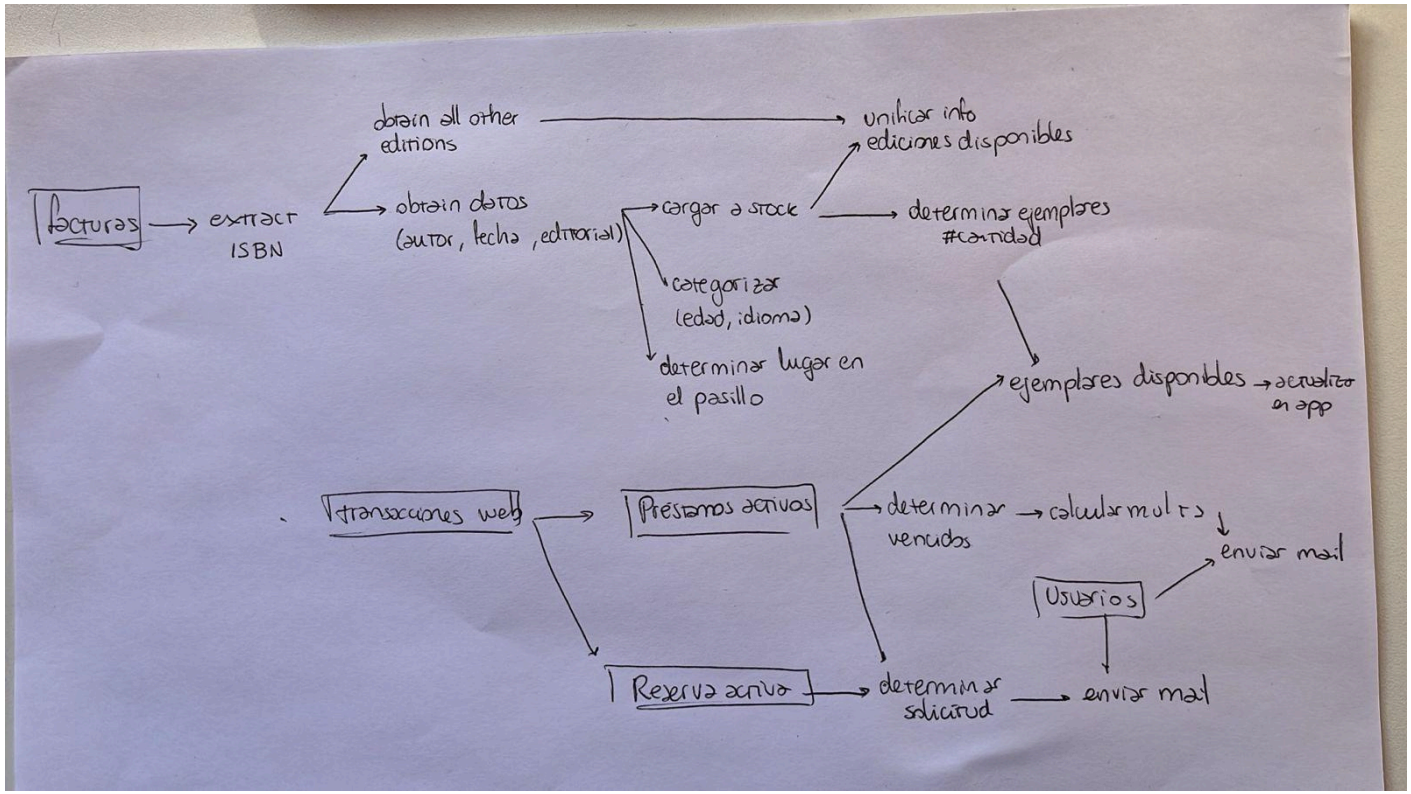
[1] compramos libros una vez al mes y los cargamos de una
[2] Los cálculos estimados están debajo del cuadro.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | préstamos activo → préstamos vencidos y multa correspondiente → mail | |
| Usuarios (creación) | registro | continua | 15 min | json | 2/3 | Actualización de Datos | usuario → crea su propia cuenta |
| Datos personales (act) | gestión cuentas (web) | continua | 15 min | json | 3% | Actualización de Datos | usuario → ve y edita sus propios datos |

Para calcular el volumen con el que arriban los datos realizamos las siguientes suposiciones:

- Libros: consideramos 10 proveedores que realizan entregas mensuales de 25 libros cada una. Estas compras representan el 99% de nuestra adquisición mensual de libros, mientras que el 1% restante proviene de donaciones. Entonces adquirimos 250 libros por compra, y entre 2 y 3 se adquieren via donaciones.
- Ejemplares: el 20% de los libros que obtenemos mensualmente son de libros que ya existen en nuestra colección, por lo que de los 252/253 que arriban hay 202 nuevos.
- Audiolibros: consideramos 2 proveedores que realizan entregas trimestrales de 10 audiolibros cada una, dejándonos un total de 20 audiolibros trimestrales.
- Reservas y préstamos: consideramos que tenemos 1000 usuarios activos. Algunos realizarán varias reservas y préstamos por mes, mientras que otros no harán ninguna por mucho tiempo, así que estimamos 1 reserva y 1 préstamo por mes por usuario.
- Usuarios: los usuarios pueden darse de alta en cualquier momento, pero es difícil estimar con qué cadencia llegan. Consideramos que tendremos un total de 5 nuevos usuarios cada 2 meses (una tasa de crecimiento anual del 3%), es decir entre 2 y 3 por mes.
- Actualizaciones de datos: consideramos que un 3% de los usuarios actualizaran sus datos de contacto una vez al año.

## Linaje de los datos




## Roles y permisos

Como se muestra en el cuadro, hay dos usuarios principales que operan sobre la base de datos. Por un lado están los bibliotecarios, encargados de gestionar por ejemplo la categorización por edades o las cancelaciones de las multas, pero no tienen permiso de acceso sobre cualquier tipo de dato. Por ejemplo, las cargas de los libros, ejemplares y similares se realiza de manera automática a partir de las facturas de compra. Solo puede intervenir un bibliotecario con permisos de administrador en caso de errores en la carga. Por el otro lado tenemos a los usuarios, quienes pueden gestionar ciertos datos como los personales referidos a su cuenta (e-mail, teléfono, etc).

# Flujo de carga de datos

## Carga de datos

Los datos utilizados son una mezcla de datos reales y los datos del faker. Por una parte, usamos el csv para extraer datos básicos como el ISBN, el título, los autores y la fecha de publicación. Pero luego muchos aspectos de nuestra base de datos deben ser simulados, como por ejemplo los géneros, los datos de los usuarios (nombre, dni, email, teléfonos, etc), los préstamos y muchos más. Para mantener la concordancia con nuestro esquema y lo definido en la primera parte del trabajo manejamos los DAGs de la siguiente manera:

1. Cargamos los idiomas y las editoriales primero. Luego cuando carguemos libros y audiolibros solo necesitaremos tomar algunos de estos al azar para asignarlos en los campos correspondientes.
2. Cargamos los autores antes que los libros a partir de lotes mensuales del csv. Mientras debemos revisar que:
    - Cada autor tenga un único id (se debe verificar entonces que este autor no exista en el esquema ni en el lote actual para no repetirlo)
    - Cada id se use en un sólo autor (se puede obtener el id más alto del esquema y luego incrementarlo dentro del lote)
3. Al tener los autores cargados, cada vez que leemos un libro del csv (del mismo lote que el que leemos para los autores) podemos ir al esquema a buscar el autor correspondiente y extraer el id para cargarlo en la tabla de escribio. La misma idea va para los audiolibros, que los extraemos a partir del mismo csv. Como los libros no se repiten en nuestro csv no necesitamos contrastar con el esquema ni nuestro lote
4. Luego cargamos los libros físicos a partir de los libros recién cargados. Como no hay repetidos sabemos que el lote será igual al anterior. Los digitales también salen de acá y por este motivo debimos romper parte de lo que establecimos en la entrega anterior. Habíamos explicado que los ISBN son únicos y ni siquiera se repiten entre ediciones físicos y digitales de un mismo libro, pero como estamos extrayendo todo desde un mismo dataset y la carga del ISBN se da primero, decidimos dejarles el mismo ISBN si llegara a darse que un libro está en ambas tablas. Reconocemos que había soluciones, pero debíamos complejizar de más la generación de datos y creemos que no era el objetivo principal de la entrega.
5. Cargamos los ejemplares a partir de los libros físicos siguiendo las mismas ideas que hasta ahora.
6. Generamos usuarios completamente random.

7. Generamos reservas a partir de samples de usuarios y libros físicos, considerando que un usuario puede reservar más de un libro y un libro puede ser reservado por más de un usuario (digamos que se respeta un orden de prioridad para entregar el préstamos)
8. Generamos préstamos a partir de samples de usuarios y ejemplares. La estructura es un poco diferente que para generar reservas ya que un ejemplar solo puede estar en préstamo a un único usuario.

Un problema que surgió con Faker es el tamaño de los datos que genera. Incluso luego de haber corrido varias veces el airflow, había ocasiones en las que sucedía que se nos iban de tamaño los nombres de los usuarios, editoriales, títulos (extraídos del csv), etc. Así que debimos ajustar el tamaño de algunos datos, lo cual quedará incongruente con las tablas creadas para la primera entrega. También cambiamos la fecha de publicación de los libros de date a numeric(4) porque la data extraída del csv eran años, y nos pareció poco útil agregar meses y días simulados con Faker.

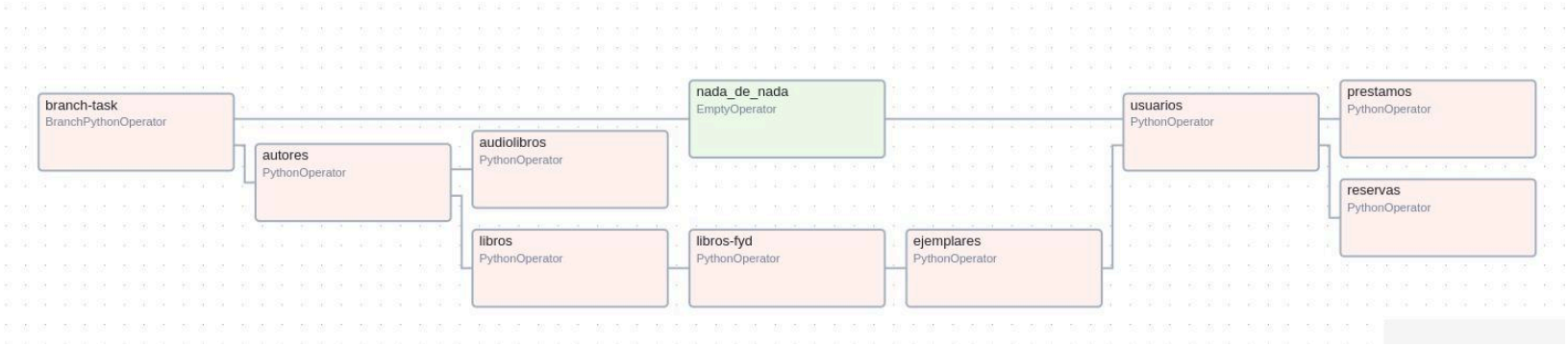
## Flujo de datos

Por una parte configuramos el DAG de fill_data_once donde se ejecuta una sola vez desde el 1ero de junio de 2024 que carga los datos iniciales de “idiomas” y “editoriales”.

Por el otro lado, fill_data se ejecuta de manera diaria, tal que los datos se llenen de una manera progresiva (utilizamos el catchup para ejecutar tareas pasadas que no se ejecutaron).

Como vamos a cargar varias actualizaciones iban a haber muchas ejecuciones si lo hacíamos minuto a minuto, por lo cual, para además no sobresaturar el sistema, decidimos no seguir la misma frecuencia y volumen que en el punto 1. Por ejemplo, los usuarios, los préstamos y las reservas se actualizarán de forma diaria.

Vamos a especificar un poco más acerca del DAG fill_data, ya que el de fill_data_once contiene un único nodo:




**Branching:**

Usando es_principio_de_mes evalua si la fecha es el primer día del mes. Si verdaderamente es el primero del mes, devolverá a los autores, en cambio cualquier otro día, devuelve nada_de_nada, que es un task vacío. Ese task vacío actúa como un placeholder para que se puedan ejecutar el resto de las tareas que no sean las del principio del mes.

**Tareas del principio de mes:**

La tareas task_autores, task_libros, task_libros_fyd, task_ejemplares y task_audiolibros se ejecutan únicamente cuando es el primer día de mes, para que la carga sea mensual (en vez de hacer un DAG mensual porque luego tenemos tareas diarias que dependen de estas mensuales)

Task_autores llama a la función obtener_autores, que maneja la generación e inserción de los datos necesarios. Las demás tareas mencionadas anteriormente siguen una idea similar.

**Tareas diarias:**

La tarea task_usuarios es responsable de generar datos de usuarios y almacenarlos en una base de datos, para eso utiliza la función obtener_usuarios. Esta tarea se ejecuta diariamente como parte del flujo de trabajo de generación de los datos en Airflow. Luego como las reservas y los préstamos dependen de que los usuarios hayan sido generados, task_reservas y task_préstamos (que generan e insertan los datos correspondientes) dependen de task_usuarios. También dependen de la creación de libros físicos y ejemplares de la otra rama del branching, pero

de una manera no directa. Para esto está el nodo nada_de_nada: siempre la ejecución debe iniciarse un primero de mes para que existan libros que reservar y prestar, pero luego mientras durante el resto del mes no se cargan, se siguen pudiendo generar reservas y préstamos.

## Dependencias

Branching:

**branch_task >> [task_autores, task_vacia]**

branch_task dirige el flujo hacia task_autores si es el primer día del mes, o hacia task_vacia en caso contrario, asegurando que la generación de los datos de los autores sea al principio del mes.

Flujo de datos mensual:

**task_autores >> task_libros >> task_libros_fyd >> task_ejemplares >> task_usuarios**

**task_autores >> task_audiolibros**

task_autores lleva a task_libros, luego a task_libros_fyd, seguido de task_ejemplares y finalmente a task_usuarios. Los datos de autores son utilizados en la generación de libros y audiolibros (task_libros y task_audiolibros respectivamente). Después de generar libros y audiolibros, task_libros_fyd se encarga de crear ejemplares físicos y digitales. Este paso utiliza los libros recién generados y es indispensable para tener ejemplares listos para préstamos y reservas. Con los libros físicos y digitales disponibles, task_ejemplares genera ejemplares específicos que están listos para ser utilizados en préstamos y reservas.

Flujo de datos diario:

**task_vacia >> task_usuarios**

Mientras se generan y preparan los datos de libros y ejemplares, task_usuarios genera datos de usuarios diariamente, asegurando que siempre haya usuarios disponibles para interactuar con el sistema de préstamos y reservas.

**task_usuarios >> task_reservas**

**task_usuarios >> task_prestamos**

Finalmente, task_prestamos y task_reservas dependen de que tanto usuarios como ejemplares estén disponibles. Estas tareas completan el flujo al generar datos de préstamos y reservas basados en los usuarios y ejemplares generados previamente.

De este modo, los DAGs no son una lista enlazada.

# Enriquecimiento

Planteamos dos transformaciones de los datos con sus respectivos casos de uso:

Obtención de deudores

La biblioteca necesita tener registro de quienes están en falta y no han devuelto sus libros al llegar la fecha de vencimiento para sus préstamos, a fin de obtener sus datos de contacto para cobrarles la multa correspondiente. Puede tener otros objetivos en mente, como conocer deudores seriales para bloquearles la posibilidad de sacar en préstamo, pero nos quedamos con lo anterior para nuestro caso de uso.

En la transformación debemos encontrar los préstamos que no tengan fecha de devolución registrada y que ya se haya pasado la fecha de vencimiento. Podría ser usada por el bibliotecario o por una aplicación que envíe correos a los deudores de manera automatizada.

La materialización elegida es una tabla, ya que son consultas que vamos a realizar más de una vez, en específico de manera diaria.

Detección de demanda insatisfecha

Esta transformación tiene un fin analítico: queremos encontrar que reservas son las que menos se satisfacen para definir qué libros comprar. La idea es si tenemos una cantidad x de copias disponibles de un libro (resultante de calcular cuántos están en préstamos) y una cantidad y de reservas activas, si y > x la diferencia entre ambas es la demanda extra que no podemos satisfacer con nuestros ejemplares actuales. Aquellos libros con las diferencias y-x más grandes serían los candidatos a comprar en las próximas órdenes.

La materialización elegida también es una tabla por las mismas razones, pero hay transformaciones intermedias que debemos realizar para calcular las reservas insatisfechas. En específico debemos calcular las reservas activas y las copias disponibles, y esta última a su vez requiere de

calcular los préstamos activos. Estas transformaciones intermedias podrían manejarse como CTEs, pero por comodidad preferimos definirlas en archivos separados y con materializaciones del tipo ephemeral. De esta manera la transformación no se almacena en ninguna tabla, sino que solo se crea para ser usada de manera transiente para el cálculo final de las reservas insatisfechas. En los archivos reservas_insatisfechas.sql y deudores.sql dejamos CTEs que podrían también ser materializaciones ephemeral, pero al ser la última transformación para calcular las cosas necesarias decidimos dejarlas, ya que se visualiza mejor de qué tablas estamos haciendo los select correspondientes.

En ambas transformaciones incluimos tests out-of-the-box, como por ejemplo garantizar el uniqueness de los ISBN o la no nulidad de varios campos. Además, incluimos tests singulares para garantizar:

- que la cantidad de reservas insatisfechas sea positiva estricta
- que los días de deuda sea mayor o igual a 1 (solo nos interesa incluir deudores, si es 0 no debería estar)
- que el email de los usuarios sea válido (es decir, formato ___@___.___)
- que el ISBN sea válido (13 dígitos y comenzado por 978)