Especialización en Back End I

Examen final

Contextualización

El proyecto consiste de tres microservicios: **Movie**, **Series** y **Catalog**.

Anteriormente, teníamos el microservicio Catalog que se comunicaba con el microservicio Movie, para obtener las películas. Se nos suma un nuevo requerimiento del usuario que desea manejar dos modos de consulta **“online”** y **“offline”**.

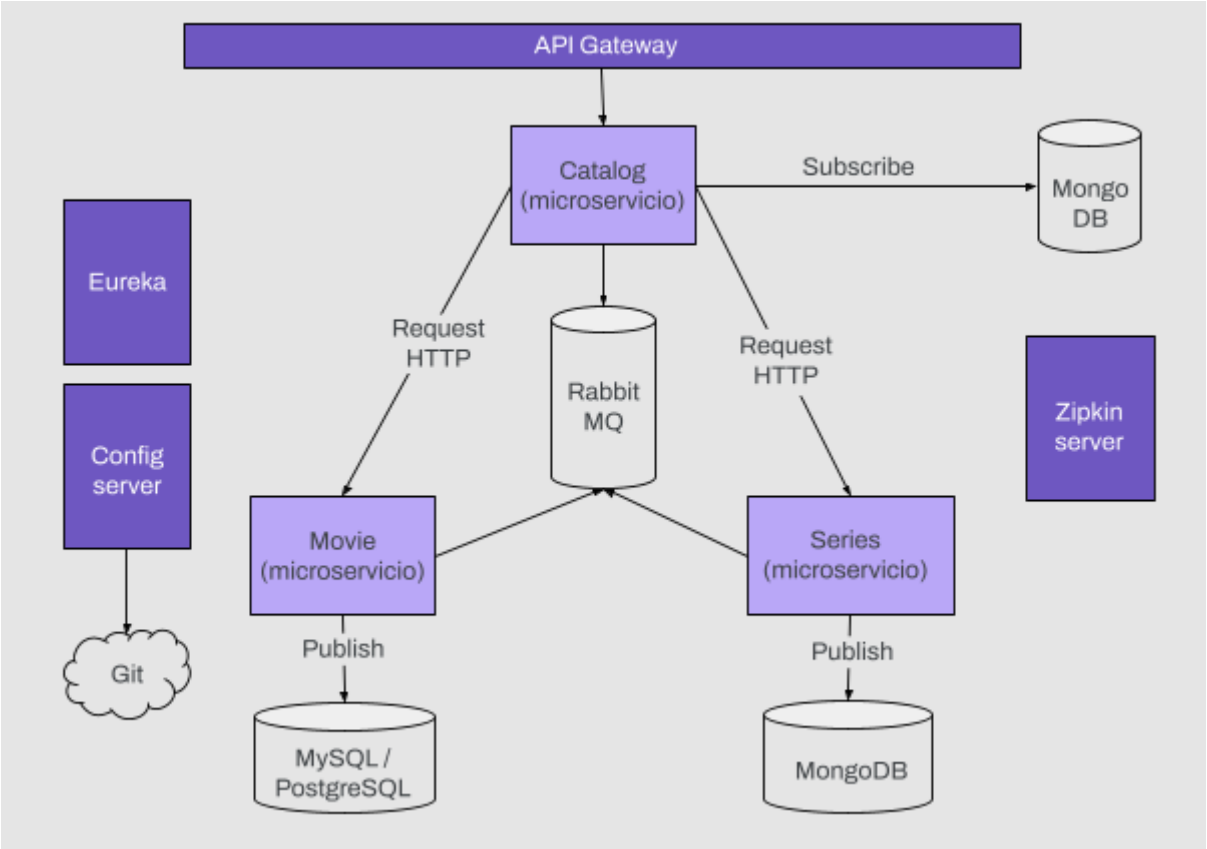
Es decir, vamos a mantener nuestra comunicación “sincrónica” entre los microservicios Catalog y Movie, y además vamos a conectar Catalog con Serie, para obtener en una misma respuesta, las películas y series con el mismo Género.

El modo offline, en vez de ir a buscar información de Movie y Serie en vivo, osea online, va a buscar en su base de datos No relacional.

El microservicio Catalog va persistir en su base de datos NO RELACIONAL, las películas y series. Y va recibir las “novedades” de nuevas Movies y nuevas Series, a través de una cola de mensajes RabbitMQ. Es decir, vamos a tener dos mensajes. **“NewMovie”, “NewSerie**”. Cada mensaje con la estructura completa de su entidad relacionada.

Cuando vayamos a invocar Catalog para obtener por género tanto películas, como series, va invocar su base de datos nada más.

Veamos un diagrama base de los microservicios:



A continuación, veremos la información detallada de los microservicios.

movie-service

El microservicio gestiona las operaciones sobre las películas. Cada película tiene como atributo:

- id
- name
- genre
- urlStream

serie-service

El microservicio gestiona las operaciones sobre las series. Cada serie tiene como atributos:

- id
- name
- genre
- seasons
    - id

        - seasonNumber
        - chapters
            id
            name
            number
            urlStream

catalog-service

Persistir la información que proporcionan ambos microservicios en una base de datos no relacional de MongoDB con la siguiente estructura: **(este tiene que ser el nuevo RESPONSE de catalog cuando se consulta por género)**

- genre
    - movies
        - id
        - name
        - genre
        - urlStream
    - series
        - id
        - name
        - genre
        - seasons
            - id
            - seasonNumber
            - chapters
                - id
                - name
                - number
                - urlStream

Consigna

serie-service

- Crear microservicio serie.
- Configurar Eureka para el nuevo servicio y utilizar el nombre: **serie-service**.
- Configurar el ruteo en el gateway para el nuevo servicio.
- Configurar Server config para obtener la configuración desde un repositorio Git.

- Crear API que nos permita:
    - Obtener un listado de series por género. Endpoint: **/series/{genre}** [GET]
    - Agregar una nueva serie. Endpoint: **/series** [POST]
- Persistencia: agregar la dependencia e implementar MongoRepository para persistir las series.
- Agregar RabbitMQ y enviar un mensaje en el momento que se agregue una nueva serie. (NewSerie)

movie-service

- Agregar RabbitMQ y enviar un mensaje en el momento que se agregue una nueva película. (NewMovie)

Spring Cloud: traceo utilizando Zipkin

- Configurar Zipkin en cada microservicio.
- Visualizar las comunicaciones entre los microservicios desde la interfaz que nos da **Zipkin UI**.

Resiliencia - Resilence4J

- Del proyecto anterior, se debe seleccionar uno de los servicios (preferentemente el que consideres que será más utilizado) y adaptarlo para que el mismo sea tolerante a fallos.
- Para lo anterior deberás:
    - **Definir esquema de resiliencia**. retry and fallback, reglas de circuito.
    - Descripción de la solución de redundancia, justificación (un comentario en el código).

Testing - RestAssured/MockMvc

- Usando esta librería, en la carpeta de test del microservicio Catalog, hacer un test para dar de alta una Película, y una serie con un género inventado, y obtener la respuesta de forma online.
- Poner un delay de 10 segundos, y obtener la misma respuesta de forma offline