Certified Tech Developer
The Ultimate Degree

Especialización en Back End I

Examen Final

Esta evaluación será distinta a la anterior, ya que la deberán ir construyendo durante las clases siguientes. Esto no quiere decir que lo resolvamos durante los encuentros en vivo, pero con los temas que iremos abordando podrán completar los requisitos.

Este examen también debe ser resuelto en equipos de dos personas. Los integrantes deben ser los mismos que para el examen parcial. Se deberán seguir utilizando los repositorios que se utilizaron para el examen parcial, tanto para las configuraciones centralizadas como para el proyecto.

El profesor pondrá a disposición de los alumnos un repositorio que contendrá el microservicio serie-service, que tendrá embebida una base de datos no relacional MongoDB. Dicho microservicio deberá ser integrado al proyecto original.

Las tareas a realizar deberán dividirse equitativamente entre los integrantes del equipo.

Recibirán la nota vía mail en el transcurso de una semana. **¡Éxitos!**

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

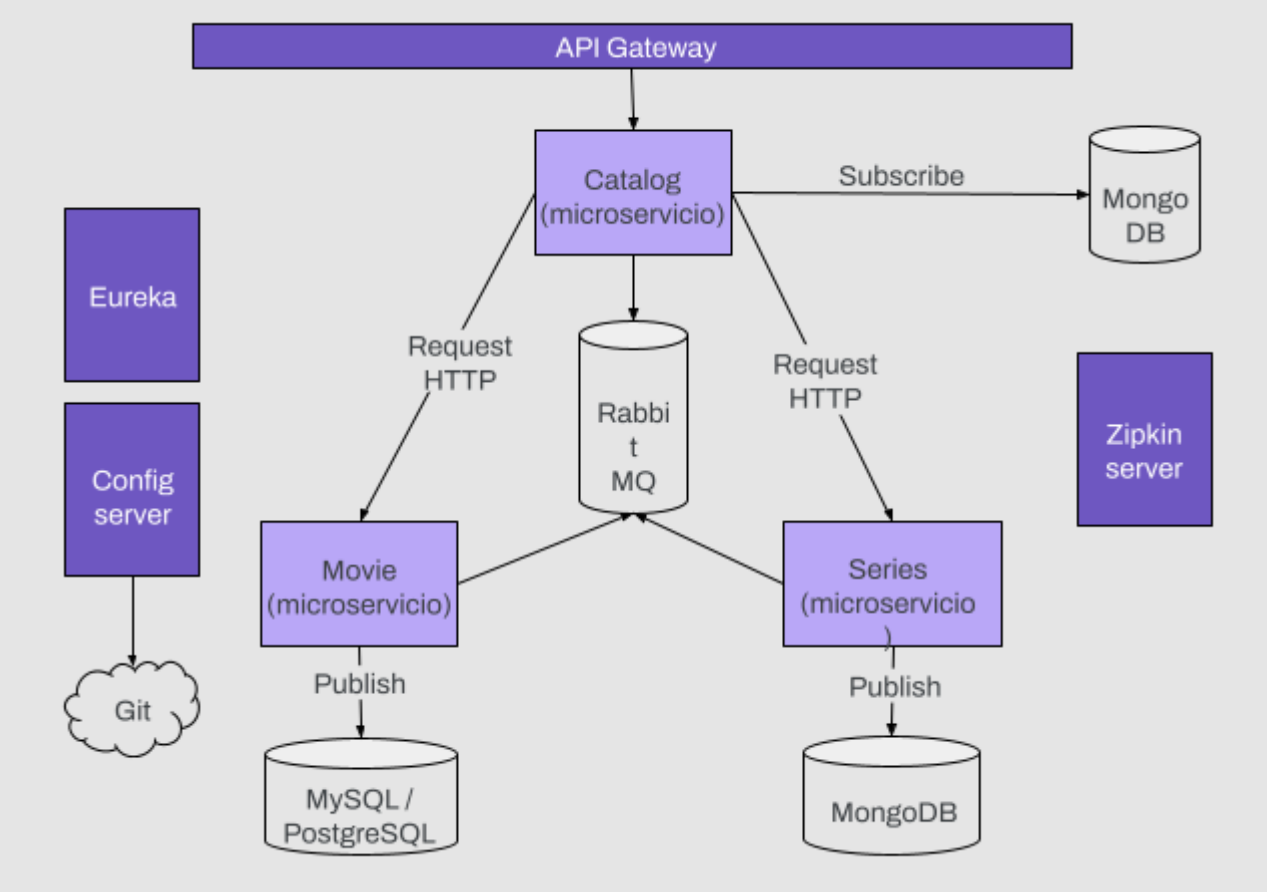
Catalog es una API REST que nos permite buscar por género, tanto películas como series. Además desde el microservicio catalog se deberán poder guardar películas y series.

La comunicación entre microservicio para crear películas debe realizarse con Feign. Al guardar una película, la misma se persistirá en su base de datos (movie-service), de acuerdo a lo solicitado en el examen parcial.

A diferencia de lo solicitado en el examen parcial, ahora, cuando se persiste una película en su base de datos, adicionalmente se debe enviar un mensaje a través de una cola de RabbitMQ a catalog-service, quien lo persistirá en su base de datos no relacional MongoDB. Lo mismo deberá ocurrir con las series (serie-service).

Cuando **Catalog** busca por género, lo hace directamente en su base de datos no relacional MongoDB.

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

Los microservicios de **película** y **serie** deben ser invocados cada vez que se carga una nueva película o serie y se debe persistir la información que proporcionan ambos microservicios en una base de datos no relacional de MongoDB con la siguiente estructura:

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

Consigna:

serie-service

- Configurar el nuevo servicio para su auto descubrimiento utilizando el nombre: **serie-service**.
- Centralizar la configuración de serie-service.
- Realizar la comunicación con Feign de serie-service y catalog-service para persistir en la base de datos de serie, cada serie creada.
- Agregar RabbitMQ y enviar un mensaje en el momento que se agregue una nueva serie.

movie-service

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

catalog-service

- Modificar la consulta por género, que ahora será directamente a su base de datos no relacional MongoDB.
- Actualizar catalog utilizando **Feign** para comunicarlo con el microservicio serie-service, y obtener desde catálogo las series filtradas por género, como así también utilizar la Feign para crearlas.
- Agregar RabbitMQ y escuchar los mensajes que envían movie-service y serie-service. En caso de recibir un mensaje de algún servicio, actualizar el listado correspondiente, ya sea de las películas o las series.

Spring Cloud: traceo utilizando Zipkin

- Configurar **Zipkin** en todos los microservicios para visualizar la trazabilidad.

Resiliencia - Resilence4j

- 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, implementar retry, un método fallback y configurar las reglas del circuito.
- Describir la solución de dicha implementación, planteando un supuesto escenario que requiera implementar este patrón, justificándolo en un comentario o en el Readme del repositorio perteneciente al proyecto.