Certified Tech Developer
The Ultimate Degree

Especialização em Back End I

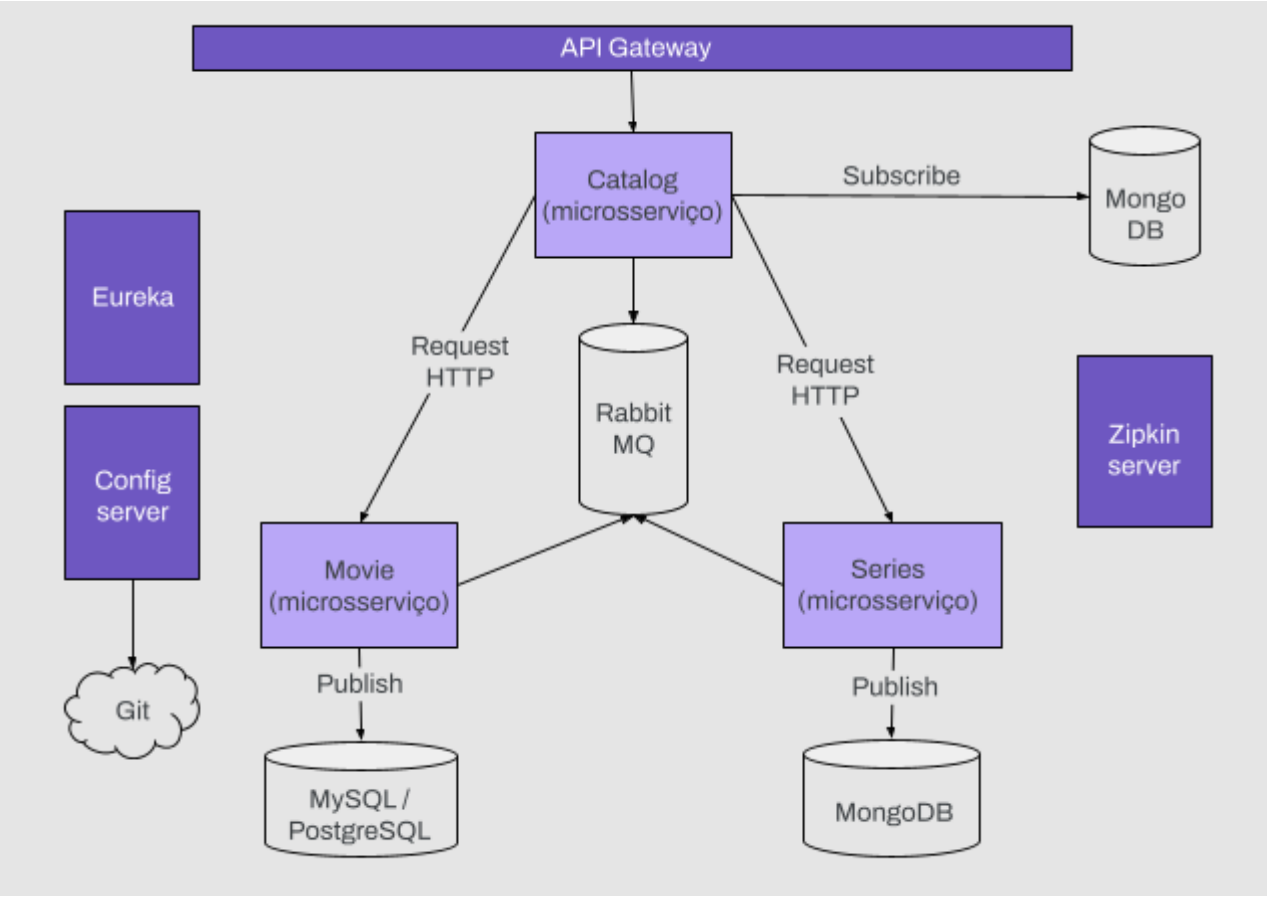
Exame final

Esta avaliação será diferente da anterior, pois a construiremos durante as aulas seguintes. Isto não significa que a resolveremos durante as reuniões ao vivo, mas com os tópicos que cobriremos você poderá completar as exigências. Sucesso!

Contextualização

O projeto consiste em 3 microsserviços: Filme, Série e Catálogo. O catálogo é um microsserviço que lê informações de Filmes e Séries a fim de enviar um catálogo ao cliente. O catálogo recebe uma mensagem toda vez que um filme ou uma série são lançados e os persiste em um banco de dados MongoDB não-relacional. Quando recebe uma solicitação do cliente, ele pesquisa o banco de dados e responde.

Vejamos um diagrama básico dos microsserviços:



A seguir, veremos os detalhes dos microsserviços.

movie-service

O microsserviço gerencia as operações no cinema. Cada filme tem como atributo:

- id
- name
- genre
- urlStream

serie-service

O microsserviço gerencia as operações da série. Cada série tem os seguintes atributos:

- id
- name

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

catalog-service

O microsserviço tem como objetivo invocar os filmes e séries de microsserviços **Movies** e **Series**. Estes microsserviços devem ser invocados toda vez que um novo filme ou série é carregado e as informações fornecidas por ambos os microsserviços devem persistir em um banco de dados MongoDB não-relacional com a seguinte estrutura:

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

Enunciado

serie-service

- Criar microsserviços em série.
- Configure Eureka para o novo serviço e use o nome: serie-service.
- Configurar o roteamento no gateway para o novo serviço e adicionar segurança com a OAuth.
- Configurar a configuração do servidor para obter a configuração de um repositório Git.
- Criar uma API que nos permita:
    - Obter uma lista de séries por gênero. Endpoint: **/series/{genre}** [GET].
    - Adicionar uma nova série. Endpoint: **/series** [POST].
- Persistência: adicionar a dependência e implementar o MongoRepository para persistir a série.
- Adicionar RabbitMQ e enviar uma mensagem quando uma nova série for adicionada.

.

movie-service

- Adicione persistência: use o MySQL para persistir nos filmes.
- Configurar o roteamento no gateway para adicionar segurança com o OAuth.
- Adicionar RabbitMQ e enviar uma mensagem quando um novo filme for adicionado.

catalog-service

- Atualizar o catálogo usando **Feign** para adicionar a este serviço a busca de séries por gênero (serie-sevice) e adicioná-las à resposta do endpoint **/catalog/{genre}**.
- Acrescente persistência: depois de obter os filmes e séries por gênero, persista-os em MongoDB.
- Adicionar RabbitMQ e ouvir as mensagens enviadas por movie-service e serie-service. No caso de receber uma mensagem de qualquer serviço, atualize a lista correspondente, seja filmes ou séries.

Spring Cloud: rastreamento utilizando Zipkin

- Criar projeto e configurar o servidor Zipkin para receber mensagens de microsserviços. Adicionar Zipkin UI para visualizar os traços.
- Configure o Zipkin em cada microsserviço.
- Visualizar a comunicação entre microsserviços a partir da interface fornecida pela Zipkin UI.
- Implantação: todos os microsserviços devem ser implantados em dockers.

Resiliência - Resilence4J

- Do projeto acima, selecione um dos serviços (de preferência aquele que você acha que será mais utilizado) e adapte-o para ser tolerante a falhas.
- Para fazer isso, você precisará:
    - Definir o esquema de resiliência. Por exemplo: duplicação de redundância, retry, balanceamento de carga, tempos de warm-up, regras do circuito.
    - Modifique o código do seu projeto - aplicando qualquer uma das três tecnologias mencionadas acima - para que o esquema definido seja aplicado dentro do serviço selecionado.
- Como mínimo, o serviço deve ter:
    - Dupla redundância.
    - Regras do circuito (você pode criar um serviço que retorna ativo/inativo dependendo da memória disponível, uso do processador, exceções).
    - Descrição da solução de redundância, justificativa (um comentário no código).