Curso Alura SpringCloud (Microservices)

Aula 1

O curso usa **spring boot versão 2.15** com spring cloud e **mariaDB 10.3**

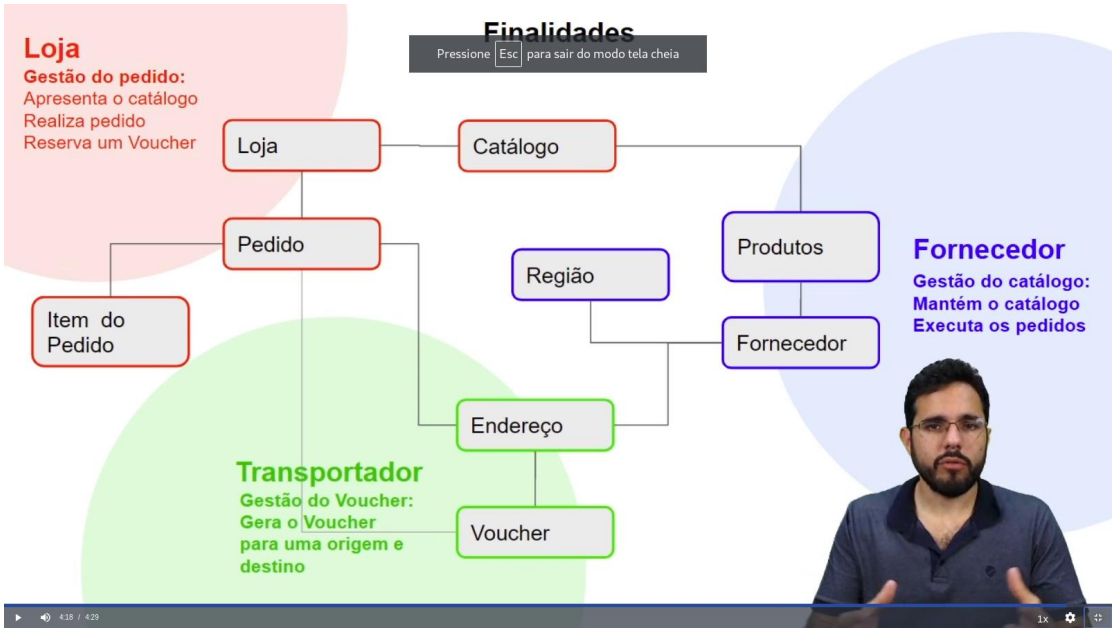
-> vou usar ultima versao de tudo com mongoDB

Neste caso, nesse microserviço nao precisamos do banco, já que ele não persistirá nada.

Slides:
https://docs.google.com/presentation/d/e/2PACX-1vQTuAjVqa3-k7wLlFXB2sHs9ZSeUJW8Zr11VoAUGJoF_gclEUK__VgBfeW7-Soifk9abCWDfiLIxHmj/pub?start=true&loop=false&delayms=1000&slide=id.p

* O microserviço é a implementação de um contexto (separado).

O foco principal dos microserviços é a separação da modelagem da aplicação em contextos coesos e **independentes** uns dos outros.



Cada uma dessas cores é um contexto diferente.

* Ele gerou o projeto a partir do STS (spring tools suite)
-> Usei o spring initialzr e já passei tanto o spring web quanto o mongoDB do spring data

AULA 02:

* Usa-se o **RestTemplate** para comunicação entre microserviços (passando o endpoint que queremos acessar).

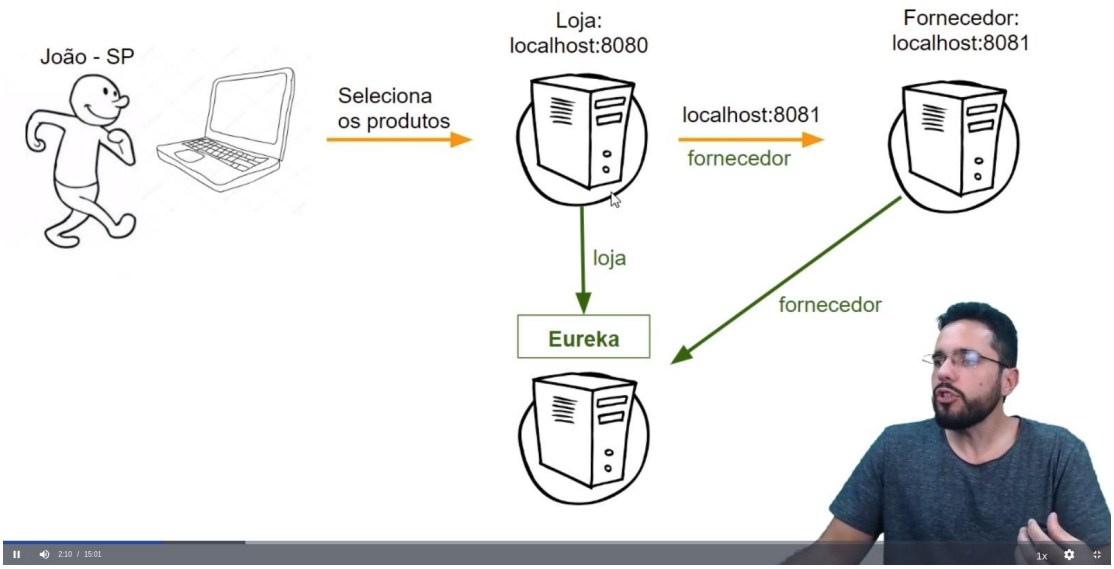
Ex.: RestTemplate client = new RestTemplate();
Client.exchange("http://localhost:**8081**/info", HttpMethod.GET, null, InfoDTO.class);
A porta é diferente, pois subiremos o outro microserviço em outra porta, claro.

Ao criar o segundo microserviço, usei o spring initialzr também, porém, com o MongoDB do spring data e o JPA.

Para trocar a porta, basta ir no **application.properties** ou criar um arquivo **application.yml**

Porém, em código, deixamos hardcoded o localhost:8081.

Para resolver isso, usamos um **service discovery (nesse caso o Eureka)**. Ou seja, nossos serviços vão se registrar no service discovery, com o nome e endereço. Com isso, o Eureka seria o intermediador entre os serviços.



Para isso, criamos um novo projeto para o Eureka (padrao).

Ao selecionar as dependencias, selecionamos apenas o **Eureka server.**

Apenas configuramos o application.properties para remover o servidor eureka de se tornar um serviço, já que ele é o proprio eureka.
E passamos uma anotação @EnableEurekaService no starter.
Alteramos a porta também, para a padrao do eureka (8761)

Quando passamos o endpoint http://localhost:8761/eureka/apps

Ele nos lista **todas** as aplicações que se encontram registradas no eureka.

Para adicionarmos apps no eureka, passamos via **application.yml** as configuracoes do eureka, além disso, precisamos passar o **spring.application.name** com o nome da aplicação para o eureka fazer a “conexão” pelo nome da aplicação.

Precisamos adicionar no pom o **eureka discovery client** ou no STS simplesmente clicar com o botao direito no projeto e selecionar spring > edit starter e colocar a dependencia do eureka.

Com isso, não usaremos mais o restTemplate hardcoded, precisamos configurar um restTemplate para usar o eureka na aplicação. Ou seja, precisamos editar o application (o arquivo de start dos projetos do spring boot).

Para ter a inteligencia de resolução por nome de serviço, colocamos a anotação **@LoadBalanced** no bean, que faz com que o eureka resolva o IP e porta a partir do nome dado.