Projeto e Arquitetura de Software
Microsserviços

Exercício: *Microsserviços*

• Criando um par de microsserviços independentes

Este exemplo será composto por dois microsserviços:
- Um que fornece valores de câmbio (*cambio*);
- Um que converte valores em uma moeda para outra (*conversao*).

Ambos expõem *endpoints* e atendem a requisições HTTP e retornam JSON como resposta, num estilo REST.

Como cliente pode-se usar o "Postman" ou outro software qualquer capaz de gerar requisições HTTP. A figura 1 apresenta a estrutura básica dos componentes do sistema:

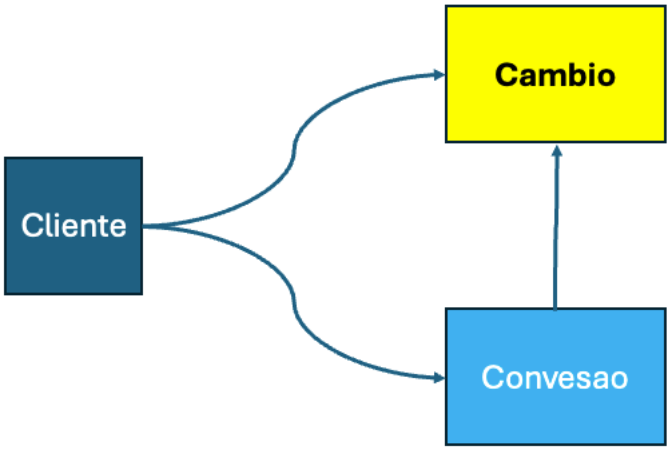

Figura 1 – estrutura básica do sistema de câmbio

Note que um cliente pode enviar requisições diretamente para o microsserviço de câmbio (para saber a cotação de uma moeda em outra), quanto para o microsserviço de conversão (para saber quanto vale uma certa quantia de moeda convertida em outra), o qual, para resolver esta questão, irá questionar o microsserviço de câmbio.

• Passo 1: criação do microsserviço de câmbio

A tecnologia de implementação será o Java com uso do *framework* Spring. Adicionalmente o mecanismo de *build* será realizado com Maven.

- Baixe o arquivo compactado com os códigos do microsserviço de câmbio do Moodle.
- Descompacte o arquivo.
- No Github crie um novo repositório privado Java e faça upload dos arquivos descompactados.
- Inicie um CodeSpaces no repositório.

Analise a estrutura do projeto:

- A arquitetura implementada foi baseada na Arquitetura Hexagonal.
- O 'Domínio' possui apenas elementos de domínio (entidades e serviços de negócio) e interfaces (portas).
- A 'Aplicacao' possui adaptadores para os elementos externos (REST e banco de dados).
- Há conversores (adapters) entre objetos de domínio e persistência (entity) e objetos de domínio e objetos de aplicação (DTO).
- Há um pacote 'main' que possui um componente de configuração para inicializar algumas dependências.

- A persistência é realizada com JPA com banco de dados H2.
- O serviço de câmbio será disponibilizado na porta 8000 (ver arquivo application.properties).
- O serviço responde ao endpoint "/currency-exchange/from/{from}/to/{to}" onde {from} indica a moeda de origem e {to} indica a moeda de destino. Ex.: "/currency-exchange/from/USD/to/BRL"

- Execute a classe de aplicação e torne a porta pública.
- Você pode testar o endpoint do serviço de câmbio por um navegador ou pelo Postman.
- Deixe o serviço executando para o próximo passo.

Passo 2: criando o microsserviço de conversão

- Baixe o arquivo compactado com os códigos do microsserviço de conversão do Moodle.
- Descompacte o arquivo.
- Abra uma nova janela ou em um novo navegador, e no Github crie um novo repositório privado Java e faça upload dos arquivos descompactados.
- Inicie um CodeSpaces no repositório.

Analise a estrutura do projeto:

- Este projeto foi desenvolvido de forma simplificada.
- O serviço expõe 2 endpoints que realizam a mesma função: recebem a moeda de origem {from}, a moeda de destino {to} e a quantidade {quantity} de dinheiro a ser convertido.
- O primeiro endpoint "/currency-conversion" faz a conversão dos valores.
  - Ele questiona o microsserviço de câmbio sobre o valor corrente da moeda para então poder fazer a conversão. Seu funcionamento é convencional. Ele recebe os parâmetros por "path variables". Para se comunicar de forma síncrona com o microsserviço de câmbio, usa uma instância de "RestTemplate".
  - Note que por ser uma comunicação síncrona a execução é suspensa até que o microsserviço demandado encaminhe a resposta.
  - O método "getForEntity" da classe "RestTemplate" é usado para retornar uma instância de "ResponseEntity". A classe "ResponseEntity" armazena uma série de informações sobre a comunicação em si. O método "getBody" dessa classe retorna o "corpo" da mensagem HTTP que se traduz na classe resposta.
  - O envio da solicitação por meio do método "getForEntity" exige a string com o "endpoint" destino, a classe que será retornada e um dicionário com os parâmetros que serão informados na solicitação.
  - É uma construção trabalhosa.
- O segundo endpoint "/currency-conversion-feign" faz exatamente a mesma coisa – solicita o câmbio para o microsserviço de câmbio e retorna a conversão – só que de uma forma mais simples.
  - Antes de tudo define-se a interface "CurrencyExchangeProxy" anotada com "@FeignClient".
  - A API OpenFeign é um gerador de clientes para o protocolo HTTP.
  - Uma classe concreta que implementa essa interface é gerada pelo SpringBoot e age como um "proxy", ou seja, faz o papel do microsserviço de câmbio, mas na verdade comunica-se com ele. Esta classe é injetada no "controller" para poder ser acessada pelo "endpoint".
  - O uso de "clientes proxy" na comunicação entre microsserviços simplifica bastante esta comunicação principalmente quando for o caso de se usar balanceamento de carga como será visto mais adiante. Para que este recurso funcione, porém é necessário a inclusão da dependência da API correspondente no arquivo POM.xml. Sugere-se o uso desta segunda abordagem.
- Copie a URL do servidor do primeiro microsserviço e ajuste o atributo 'URLmscambio' do Controller para esta URl.
- Execute a classe de aplicação.
- Você pode testar os endpoints do serviço de conversão por um navegador ou pelo Postman.

- ## Conclusão

Este roteiro mostrou como executar os microssserviços de câmbio e de conversão de moedas e como eles podem interagir de forma síncrona.