### MICROSERVICE E MENSAGERIA COM SPRING BOOT CLOUD E DOCKER ###

1. DISCOVERY SERVER EUREKA
   1. Criar projeto em Spring Initializr <https://start.spring.io/> e adicionar as dependências:
      1. Eureka Server
      2. Spring Security
   2. Em src/main/java/EurekaserverApplication adicionar a annotation
      1. @EnableEurekaServer;
   3. Em src/main/Resources alterar a extensão para application.yml e configura-lo
2. GATEWAY
   1. Criar projeto em Spring Initializr <https://start.spring.io/> e adicionar as dependências:
      1. Eureka Discovery Client
      2. OAuth2 Resource Server
   2. Configurar o arquivo pom.xml
      1. Adicionar <dependencyManagement>
      2. Alterar <dependency> spring-boot-starter-webflux
      3. Adicionar <dependency>org.springframework.cloud => spring-cloud-starter-gateway
      4. Adicionar <dependency>spring-cloud-starter-netflix-eureka-client
   3. Em src/main/Resources alterar a extensão para application.yml e configura-lo
   4. Em src/main/java/MscloudgatewayApplication adicionar a annotation
      1. @EnableEurekaClient
      2. @EnableDiscoveryClient;
   5. Criar em src/main/java/MscloudgatewayApplication:
      1. Função RouteLocator routes(RouteLocatorBuilder builder)
      2. Adicionar a annotation @Bean
      3. Referenciar as rotas dos microserviços
3. MICROSERVICE Clientes
   1. Criar projeto em Spring Initializr <https://start.spring.io/> e adicionar as dependências:
      1. Spring WEB
      2. Spring Data JPA
      3. Lombok
      4. Spring Boot DevTools
      5. Spring Boot Actuator
      6. H2 Database
      7. Eureka Discovery Client
   2. Configurar o arquivo pom.xml
   3. Em src/main/Resources alterar a extensão para application.yml e configura-lo
   4. Em src/main/java/EurekaserverApplication adicionar a annotation
      1. @EnableEurekaClient;
   5. Criar no pacote raiz => domain.Cliente
   6. Criar no pacote raiz => infra.repository.IClienteRepository
   7. Criar no pacote application => ClienteService
   8. Criar no pacote raiz => application.ClienteResource ou ClienteController
   9. Criar no pacote application => representation.ClienteSaveRequest ou dto.ClienteSaveRequest
   10. Realizar o registro da rota no microservice GATEWAY
4. MICROSERVICE Cartoes
   1. Criar projeto em Spring Initializr <https://start.spring.io/> e adicionar as dependências:
      1. Spring WEB
      2. Spring Data JPA
      3. Lombok
      4. Spring Boot DevTools
      5. Spring Boot Actuator
      6. H2 Database
      7. Eureka Discovery Client
      8. Spring for RabbitMQ
   2. Configurar o arquivo pom.xml
   3. Em src/main/Resources alterar a extensão para application.yml e configura-lo
   4. Em src/main/java/MscartoesApplication adicionar a annotation
      1. @EnableEurekaClient;
   5. Criar no pacote raiz => domain.Cartao
   6. Criar no pacote raiz domain => BandeiraCartao
   7. Criar no pacote raiz => infra.repository.ICartaoRepository
   8. Criar no pacote application => CartaoService
   9. Criar no pacote raiz => application.CartoesResource ou CartoesController
   10. Criar no pacote application => representation.CartaoSaveRequest ou dto.CartaoSaveRequest
   11. Realizar o registro da rota no microservice GATEWAY
5. MICROSERVICE AvaliadorCredito
   1. Criar projeto em Spring Initializr <https://start.spring.io/> e adicionar as dependências:
      1. Spring WEB
      2. Spring Data JPA
      3. Lombok
      4. Spring Boot DevTools
      5. Spring Boot Actuator
      6. Eureka Discovery Client
      7. OpenFeign
      8. Spring for RabbitMQ
   2. Configurar o arquivo pom.xml
   3. Em src/main/Resources alterar a extensão para application.yml e configura-lo
   4. Em src/main/java/MscartoesApplication adicionar a annotation
      1. @EnableEurekaClient;
      2. @EnableFeignClients
   5. Criar no pacote raiz => domain.SituacaoCliente
   6. Criar no pacote domain => DadosCliente
   7. Criar no pacote domain => CartaoCliente
   8. Criar no pacote application => AvaliadorCreditoService
   9. Criar no pacote raiz => infra.clients.ClienteResourceClient
   10. Criar no pacote raiz => application.AvaliadorCreditoController
   11. Realizar o registro da rota no microservice GATEWAY
6. MENSAGERIA RabbitMQ(Broker) COM DOCKER
7. Broker -> recebe uma mensagem via AMQP() e envia para um Exchange
8. Exchange -> recebe as mensagens e encaminha de acordo com os bindings(conexões) e os tipos declarados de Exchange
   1. Exchange Default -> recebe a solicitação de mensagem e encaminha para a fila
   2. Exchange direct ->
   3. Exchange fanout ->
   4. Exchange topic ->
   5. Exchange headers ->
9. Instalação RabbitMQ com Docker
   1. CLI -> docker run -it --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.11-management
   2. Localhost:15672 -> user: guest - pwd: guest
   3. Na aba Queues -< Add queue
      1. Name: emissão-cartoes
10. No mscartoes -> application.yml
    * + - referenciar a fila do RabbitMQ emissão-cartoes em mq:

queues:

emissão-cartoes: emissão-cartoes

* + - * referenciar em spring:

rabbitmq:

host: localhost

port: 5672

username: guest

password: guest

1. No mscartoes -> MscartoesApplication -> Annotation @EnableRabbit
2. No mscartoes -> adicionar a classe infra.mqueue.EmissaoCartaoSubscriber
3. Run nos projetos eureca/gateway/cartões
4. Para testar o RabbitMQ => Queues->emissão-cartoes->Publish message->Payload->Publish message
5. No msavaliadordecredito, repetir passos d, e
6. No msavaliadordecredito-> adicionar a classe domain.model.mqueue.SolicitacaoEmissaoCartaoPublisher
7. No msavaliadordecredito-> adicionar a classe infra.mqueue.DadosSolicitacaoEmissaoCartao
8. No msavaliadordecredito-> adicionar a classe config.MQConfig
9. No msavaliadordecredito->adicionar a classe domain.model.ProtocoloSolicitacaoCartao
10. No msavaliadordecredito->adicionar a classe application.ex.ErroSolicitacaoCartaoException
11. SEGURANÇA COM KEYCLOAK
12. Instalação com Docker

CLI => docker run --name keycloak -p 8081:8080 -e KEYCLOAK\_ADMIN=admin -e KEYCLOAK\_ADMIN\_PASSWORD=admin quay.io/keycloak/keycloak:20.0.3 start-dev

1. Para acessar: localhost:8081
2. Administration Console => user: admin pwd: admin
3. Create Realm
4. Users -> Create new user -> configurar novo usuario
5. No mscloudgateway -> application.yml
   1. Copier o endpoint da pagina do keycloak -> Realm settings->EndPoints-> copier a url de issuer na página que será aberta no navegador
   2. Adicionar security e colar em issuer-uri a url copiada do keycloak
6. Para testar-> no Insomnia, nova requisição->Authentication->OAuth2
   1. Enable
   2. Grant type -> url do keycloak => token\_endpoint
   3. Client ID->Client ID do Client em Keycloak
   4. Client Secret->Keycloak-> Clients-> escolher o cliente->Credentials-> copiar Client secret
   5. Fetch Tokens-> vai receber o token->Copiar
   6. Em Clientes->Salvar->Barer Token
7. Actuator, métricas e arquivo de log
   1. Em msclientes-> apllication.yml-> adicionar:

management:

endpoints:

web:

exposure:

include: '\*'

logging:

file:

name: ./logs/log-file.log

1. Documentação da API
   1. Adicionar dependency de <https://springdoc.org/>
      1. Acessar no microserviço em eureca -> localhost:XXXXX/Swagger-ui/index.html
2. DOCKER
   1. Build do projeto
      1. CLI -> ./mvnw clean package -DskipTests
      2. CLI-> cd target -> ls -> java -jar .\nome.jar => para rodar o projeto
      3. CLI -> ctrl+C => para a aplicação
   2. Na raiz do projeto criar um arquivo chamado Dockerfile
   3. CLI => docker build --tag cursoms-eureka .
   4. CLI => docker run --name nome\_container -p 8761:8761 nome\_imagem => criar e rodar o container
   5. CLI => docker container stop nome\_container
   6. CLI => docker run nome\_container => voltar a rodar o container ja criado
   7. CLI => docker container rm nome\_container => remove o container
   8. CLI => docker image rm nome\_image => deleta a imagem
   9. CLI => docker images => lista as imagens disponíveis
   10. CLI => docker build --tag cursoms-eureka .
   11. CLI => docker network create nome\_network => cria uma nova network
   12. CLI => docker network ls => visualiza as networks disponíveis
   13. CLI => docker run --name eureka -p 8761:8761 --network cursoms-network cursoms-eureka
   14. CLI => docker run --name cursoms-cartoes --network cursoms-network cursoms-cartoes