MBA Engenharia de Software

**Service Architecture / API / Mobile Architecture – Padrões de Resiliência**

# Implementando o Padrão Fallback

1. Importe para o Eclipse o projeto Maven **orderservice** que está localizado na pasta

**2-resiliencia**;

A importação deve ser feita da mesma forma que fizemos na primeira aula (importação de um projeto Maven);

1. Localize o arquivo **pom.xml**;
2. Adicione dentro da tag <dependencies> as dependências:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

1. Na classe **OrderServiceApplication.java** adicione a seguinte anotação:

@EnableCircuitBreaker

1. Salve o trabalho;
2. Execute a classe **OrderServiceApplication.java** (Clique com o botão da direita na classe, Run As >> Java Application);

Será exibido no console algumas informações a respeito da inicialização.

1. Abra um browser e acesse a seguinte URL: <http://localhost:8080/v1/orderservice/getPaymentoInfo/1> diversas vezes, você verá que o comportamento dessa chamada está com algum problema, em praticamente metade das chamadas, é retornado um erro.
2. Interrompa a execução da classe OrderServiceApplication.java;
3. Navegue pelas classes OrderServiceController.java e OrderService.java e examine o fluxo de execução da chamada;
4. Na classe OrderService.java você deve implementar um método de fallback para o método getPaymentInfo(Order order). Nesse novo método você irá retornar um objeto PaymentInfo, atribuindo na propriedade comments a seguinte informação: “No information available about payments!”;

Lembrando a anotação:

@HystrixCommand(fallbackMethod = "nome\_metodo\_fallback")

1. Verifique que na classe PaymentInfo temos um construtor que recebe um parâmetro e atribui a propriedade comments;
2. Salve o trabalho;
3. Execute a classe **OrderServiceApplication.java** (Clique com o botão da direita na classe, Run As >> Java Application);
4. Abra um browser e acesse a seguinte URL: <http://localhost:8080/v1/orderservice/getPaymentoInfo/1> com o método de fallback implementado não será mais apresentado erros para o usuário e sim a mensagem configurada no item 10.

# Implementando o Circuit Break

1. Execute a classe **OrderServiceApplication.java** (Clique com o botão da direita na classe, Run As >> Java Application);
2. Abra um browser e acesse a seguinte URL: <http://localhost:8080/v1/orderservice/getItems/10> verifique que o comportamento dessa chamada em alguns momentos demora demais para responder, podendo causar lentidão e degradação na aplicação;
3. Navegue pelas classes OrderServiceController.java e OrderService.java e examine o fluxo de execução da chamada;
4. Introduza no método getTotalItem(Order order) um circuit-break para limitar o tempo máximo desse método em 2 segundos;

@HystrixCommand(commandProperties={@HystrixProperty(name="execution.isolation.thread.timeoutInMilliseconds",value="TIMEOUT\_EM\_MILISSEGUNDOS")})

1. Também adicione um método de fallback para retornar o valor -1 no caso do tempo exceder os 2 segundos;
2. Salve o trabalho;
3. Execute a classe **OrderServiceApplication.java** (Clique com o botão da direita na classe, Run As >> Java Application);
4. Abra um browser e acesse a seguinte URL: <http://localhost:8080/v1/orderservice/getItems/10> verifique a mudança ocorrida após a implementação.

Veja o passo a passo e uma explicação desse exercício em:

<https://drive.google.com/file/d/1XtxWNQj9tfWXEGDJReCIgTBg81x_oJjx/view?usp=sharing>