CENTRO UNIVERSITÁRIO UNIVEL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

# ATIVIDADE DE ROTEIROS DE ARQUITETURA DE SOFTWARE

MATHEUS HENRIQUE BUTKOSKI
GUSTAVO PERUZZO

Cascavel

29/09/202

CENTRO UNIVERSITÁRIO UNIVEL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

# MODELO DE RELATÓRIO LABORATÓRIO DE PADRÕES ARQUITETURAIS

**MATHEUS HENRIQUE BUTKOSKI GUSTAVO PERUZZO**

**Relatório apresentado como requisito parcial para obtenção de nota na disciplina "Arquitetura de Software e Padrões de Projetos" ministrada pelo Prof.Me. Ederson Schmeing.**

Cascavel
29/09/2022

# Sumário

# 1 Introdução

Neste trabalho, executamos os roteiros que foram disponibilizados na atividade. Dessa forma, no decorrer da experiência poderemos entender melhor sobre a arquitetura de microsserviços e subscribe, também aprenderemos sobre os programas e componentes de uma arquitetura.

## 1.1 Objetivos

O Objetivo do trabalho é recriar com sucesso os modelos de micro-livraria apresentados nos roteiros, a fim de compreender e entender as estruturas que compõem arquitetura de microsserviços e subscribe.

## 1.2 Material e Métodos

Para realizar o trabalho foram necessários alguns programas para rodar as arquiteturas. No primeiro roteiro, era necessário a instalação do Git e Node.js, que nós já possuíamos nas máquinas. Além disso, utilizamos o Visual Studio Code para realizar as alterações nos códigos e também para a utilização do Terminal. A parte mais complicada se tornou a instalação do Docker, visto que ele é um programa pesado, que exige diversos componentes, um deles era o WSL, que não tínhamos no computador, após várias tentativas e reinicializações de sistema, conseguimos enfim executar o programa corretamente.

# 2 Resultados e discussão

## 2.1 Roteiro de Microsserviços

Primeiramente, o roteiro nos pediu para baixarmos os arquivos e executar as linhas de códigos indicadas. Nessa primeira execução já vimos como basicamente o sistema funcionava, fizemos a execução do microsserviço através do npm run start e o sistema passou a rodar inicialmente em um localhost com porta 5000, foi possível visualizar toda a parte de front-end do sistema com os livros e seus respectivos títulos, autores e preço.

Passando para a segunda parte, criamos uma nova operação no sistema através de implementações de código indicadas pelo roteiro. Criamos uma nova função chamada SearchProductByID e ao fim de tudo criamos uma nova rota para testarmos a função e tudo ocorreu como indicado. Ao acessarmos a novo link com a nova rota, visualizamos a chamada feita na API, e nela estava o resultado da nova operação do back-end, mostrando as informações do livro que foi solicitado através da função criada.

Nessa próxima etapa foi a que mais encontramos dificuldades, visto que o Docker não é um programa tão simples, mas após algumas falhas e tentativas conseguimos por fim instalá-lo e rodá-lo corretamente. Nos foi pedido para criar um novo arquivo na raiz do projeto e após isso compilá-lo para a geração de uma imagem. Logo em seguida, implementamos um novo microsserviço chamado Shipping, que ao fim da execução do roteiro foi possível entendê-lo perfeitamente. Agora o front-end do sistema possuia interações e a aba de calcular frente nos retornava um valor aleatório quando incluíamos um número.

Compreendemos a facilidade que o Docker pode nos fornecer ao ser possível transportar microsserviços facilmente entre máquinas diferentes.

## 2.2 Roteiro de Publish/Subscribe

Primeiramente, deverá ser instalado Docker, caso você possua um sistema windows, deverá ser instalado o componente WSL para rodar o Docker corretamente. Após instalado, basta executar o Docker, e executar uma linha de comando dentro da pasta raíz do projeto para que o RabbitMQ possa ser instalado. Depois de rodar o comando e instalar o Rabbit, uma instância do Rabbit estará sendo executada localmente na sua máquina e podemos acessar através da porta 15672, ao acessá-lo, devemos fazer login para prosseguir, por padrão o acesso terá como usuário e senha a palavra guest.

Após feito tudo isso, devemos criar uma fila que irá representar os pedidos dos clientes indo até a aba “Queues”, na sessão “Add a new queue”, preencha o campo “name” como “orders” e clique na opção “lazy mode”. Com a fila criada, podemos inserir um json que criará um evento representando um pedido, para inserir o json devemos navegar até “Publish message” e copiar o json no campo “Payload”.

Como próximo passo, devemos ativar os 3 serviços disponíveis necessários para dar continuidade a finalização de um pedido, que são: orders-service, contact-service e shipping-service.

Para executar o orders-service que será responsável pelo processamento de pedidos, devemos ir até a pasta raíz do projeto e executá-lo através do Docker, após executado, podemos acessar um arquivo de logs por meio de comandos, nele irá conter todas as filas que foram inseridas no Rabbit.

O serviço contact-service será responsável por contactar o usuário através de um email, informando se o seu pedido foi aprovado ou não. Para executarmos, devemos ir até a raíz do projeto e inicializando-o através do Docker por linha de comando e após executado, será gerado um arquivo json com o conteúdo do email.

Para executar o shipping-service que será responsável pelo envio da mercadoria, basta executá-lo através do Docker e para visualizar as informações geradas, basta acessar o arquivo de logs executando através de linha de comando. Por fim, após finalizado, devemos encerrar e finalizar as aplicações através de um comando no Docker.

# 3 Conclusões

Com a execução do primeiro roteiro pudemos compreender mais como funciona a parte das requisições e modificações de microsserviços, assim como a criação de novas operações e interações para o sistema. Além disso, foi possível possuir o entendimento sobre as funcionalidades do Docker e como ele pode nos auxiliar nos microsserviços

Através do segundo roteiro podemos entender como funciona a arquitetura Publish/Subscribe que é um modelo assíncrono, no qual são executados 3 serviços diferentes e não acoplados, que são para poder finalizar um pedido. Na arquitetura Publish/Subscribe temos dois tipos de processos, produtores e consumidores, neste roteiro tivemos contato com os dois tipos de processos, pois o serviço de pagamento é tanto consumidor do evento de solicitação de pagamento como produtor de eventos para os demais processos do sistema.