Relatório Técnico Desafio BTG Pactual

Gabriel dos Santos

São Paulo, 2023

Github

https://github.com/ga-santos/Desafio-Engenheiro-Software

Plano de trabalho

Tasks	Horas previstas	Horas realizadas
Criar a base dados no MongoDB com uma coleção (MongoDB Atlas)	1	1
Estabelecer conexão com um servidor do RabbitMQ que irá conter a fila (Docker)	1	1
Criar projeto da API REST em .NET e configurar o Swagger	2	0,5
(API REST) Estabelecer conexão com a base de dados Mongo	2	1
(API REST) Desenvolver consulta do valor total do pedido	5	0,5
(API REST) Desenvolver consulta da quantidade de pedidos por cliente	1	0,5
(API REST) Desenvolver consulta da lista de pedidos realizados por cliente	1	0,5
(API REST) Desenvolver testes funcionais	8	2
(API REST) Expor as consultas através de endpoints	2	0,5
Criar microserviço em .NET para publicar a mensagem na fila	2	
Criar um endpoint para publicar a mensagem na fila	1	0,5
Criar microserviço em .NET para consumir os dados da fila	2	0,5
(Microserviço Consumer) Estabelecer conexão com a base de dados Mongo	2	0,25
(Microserviço Consumer) Desenvolver inserção da mensagem na coleção	8	0,25
Documentação	4	4
TOTAL	**42**	**13**

- Houve uma significativa diminuição no número de horas para realização do projeto.
    - A diminuição mais significativa foi no desenvolvimento da API REST e isso se deu pela experiência de projetos passados.
    - Houve também uma diminuição significativa no desenvolvimento do Microserviço Consumidor e isso se deu devido que trechos do código puderam ser reaproveitados da API REST.
    - O item em vermelho foi descontinuado devido que foi possível efetuar a publicação dos pedidos na fila através da própria API REST.
- O MongoDB Cloud foi utilizado pela praticidade e oportunidade de conexão através da nuvem.
- O Docker foi utilizado para iniciar o RabbitMQ devido a facilidade dos containers. Optei por rodar o Docker localmente e o código do container está armazenado em uma pasta do projeto ao invés do dockerHub devido que não possuo tanto contato com o Docker.

Tecnologias utilizadas

- Linguagens: C#
- Versões: .NET 6.0
- IDE: Visual Studio 2022
- SO: Windows

- Base de dados: MongoDB Cloud Atlas
- Mensageiro: RabbitMQ
- Conteinerização: Docker

Instruções de uso

Base de dados

- A base de dados está no MongoDB Cloud Atlas.
- Para conexão, através do Robô 3T/Studio 3T ou outra IDE, utilizar os seguintes parâmetros:
    - Server: cluster0.jeofofc.mongodb.net
    - UserName: admin
    - Password: NEoJN3KEl0QzjTmI
    - SRV Service Name: mongodb

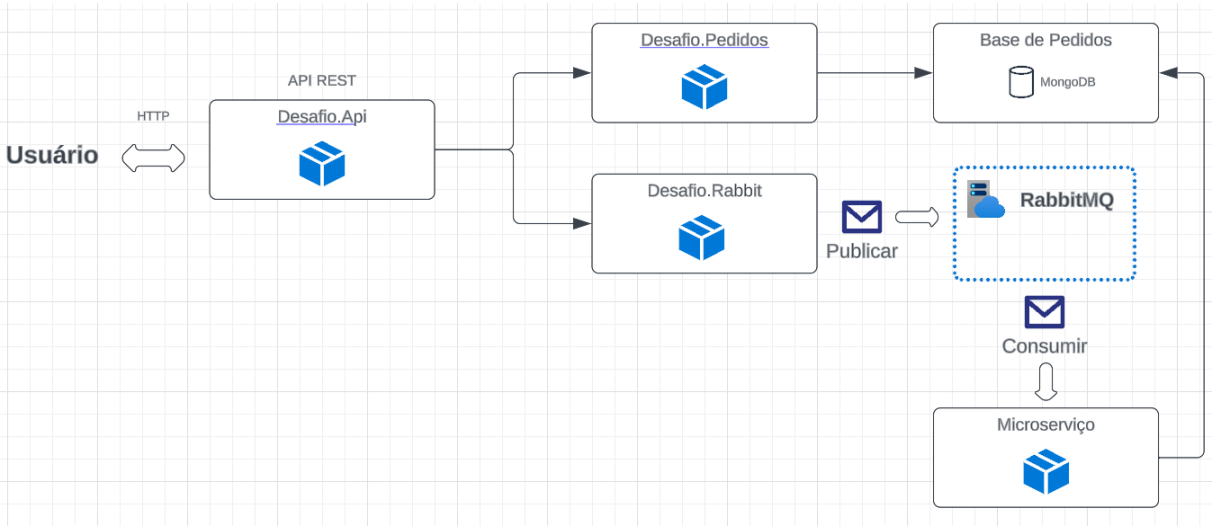
Mensageiro

- O RabbitMQ é iniciado a partir de um container.
- É necessário a instalação do Docker na máquina.
- O arquivo com as configurações está no Git (rabbit-mq/docker-compose.yml)
- Para iniciar o RabbitMQ **localmente**, rodar o comando “docker-compose up” na pasta em que está o arquivo.

API Rest e Microserviço Consumidor

- Executar as soluções **localmente** a partir do projeto .Api

Diagrama de arquitetura

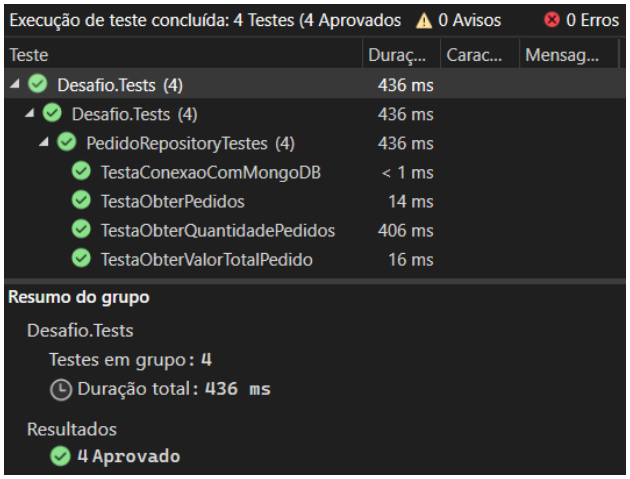


Modelagem da base de dados

Banco de dados não-relacional MongoDB

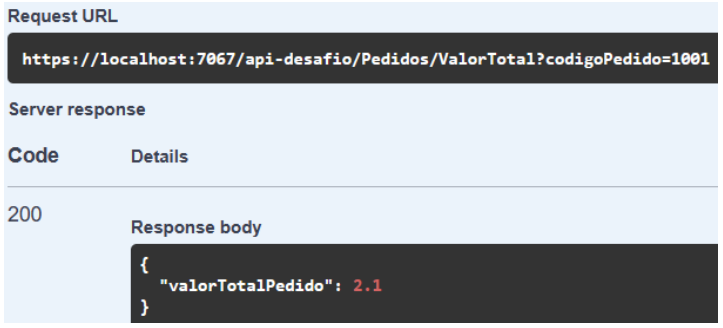
Evidência de Testes funcionais da aplicação

- Testes unitários

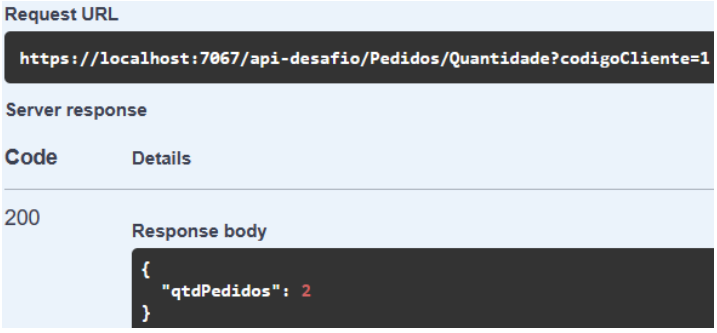


- Base MongoDB alimentada com pedidos

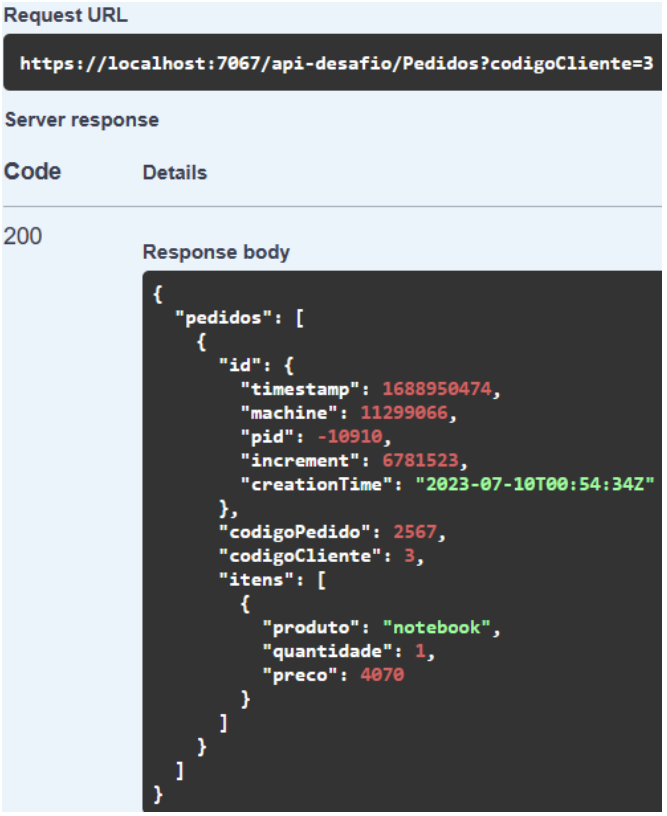
- Obter valor total do pedido



- Obter quantidade de pedidos por cliente



- Obter lista de pedidos por cliente



Referências utilizadas

https://www.rabbitmq.com/tutorials/tutorial-one-dotnet.html

https://github.com/VitorAraujoAlcantara/youtube-samples/blob/main/Rabbit.Publisher.Api/docker-compose.yml

https://www.c-sharpcorner.com/article/consuming-rabbitmq-messages-in-asp-net-core/