

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
- Containerização: 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.jeofoc.mongodb.net
 - UserName: admin
 - Password: NEOJN3KEl0QzjTml
 - SRV Service Name: mongodb

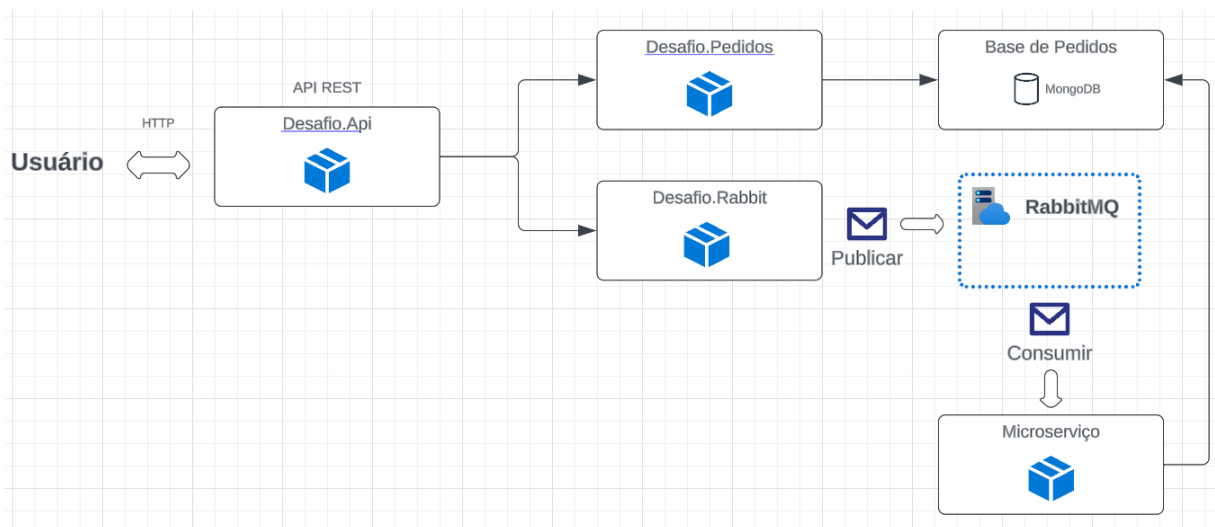
Mensageiro

- 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

Execução de teste concluída: 4 Testes (4 Aprovados) 0 Avisos 0 Erros			
Teste	Duraç...	Carac...	Mensag...
Desafio.Tests (4)	436 ms		
Desafio.Tests (4)	436 ms		
PedidoRepositoryTests (4)	436 ms		
TestaConexaoComMongoDB	< 1 ms		
TestaObterPedidos	14 ms		
TestaObterQuantidadePedidos	406 ms		
TestaObterValorTotalPedido	16 ms		
Resumo do grupo			
Desafio.Tests			
Testes em grupo: 4			
Duração total: 436 ms			
Resultados			
4 Aprovado			

- Base MongoDB alimentada com pedidos

```
_id: ObjectId('64a6b38db107bfc52af92aca')
codigoPedido: 1001
codigoCliente: 1
▸ itens: Array
```

```
_id: ObjectId('64ab5634ac68fad562677a52')
codigoPedido: 123
codigoCliente: 1
▸ itens: Array
```

```
_id: ObjectId('64ab56caac68fad562677a53')
codigoPedido: 2567
codigoCliente: 3
```

- Obter valor total do pedido

Request URL	
https://localhost:7067/api-desafio/Pedidos/ValorTotal?codigoPedido=1001	
Server response	
Code	Details
200	<div>Response body<div><pre>{ "valorTotalPedido": 2.1 }</pre></div></div>

- Obter quantidade de pedidos por cliente

Request URL	
<code>https://localhost:7067/api-desafio/Pedidos/Quantidade?codigoCliente=1</code>	
Server response	
Code	Details
200	Response body <pre>{ "qtdPedidos": 2 }</pre>

- Obter lista de pedidos por cliente

Request URL	
<code>https://localhost:7067/api-desafio/Pedidos?codigoCliente=3</code>	
Server response	
Code	Details
200	Response body <pre>{ "pedidos": [{ "id": { "timestamp": 1688950474, "machine": 11299066, "pid": -10910, "increment": 6781523, "creationTime": "2023-07-10T00:54:34Z" }, "codigoPedido": 2567, "codigoCliente": 3, "itens": [{ "produto": "notebook", "quantidade": 1, "preco": 4070 }] }] }</pre>

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/>