Teste Prático

Jitterbit

ROCK YOUR WORKFLOW

Instruções: Preencha as informações com os seus dados e faça o **Teste Prático** de comunicação entre microserviços conforme instruções abaixo, ao final informe neste material qual é o endereço do github com o código fonte desenvolvido.

Importante: O Teste Prático tem o objetivo do candidato demonstrar as habilidades de realizar a comunicação entre microservices, aplicando os conceitos de comunicação assíncrona, mesmo que o candidato não tenha experiência com microserviços este teste é uma oportunidade para aprender a implementar e demonstrar a sua capacidade em resolver problemas e a demonstrar os seus conhecimentos com desenvolvimento de software.

Outro objetivo importante deste teste é aplicar os conceitos de arquitetura de microservices que temos dentro da Jitterbit e que dará a oportunidade do candidato se preparar para os cenários reais de microservices com alta escalabilidade e alta disponibilidade.

Qualquer dúvida referente ao teste não deixe de pedir ajuda!

Desejamos a você boa sorte!

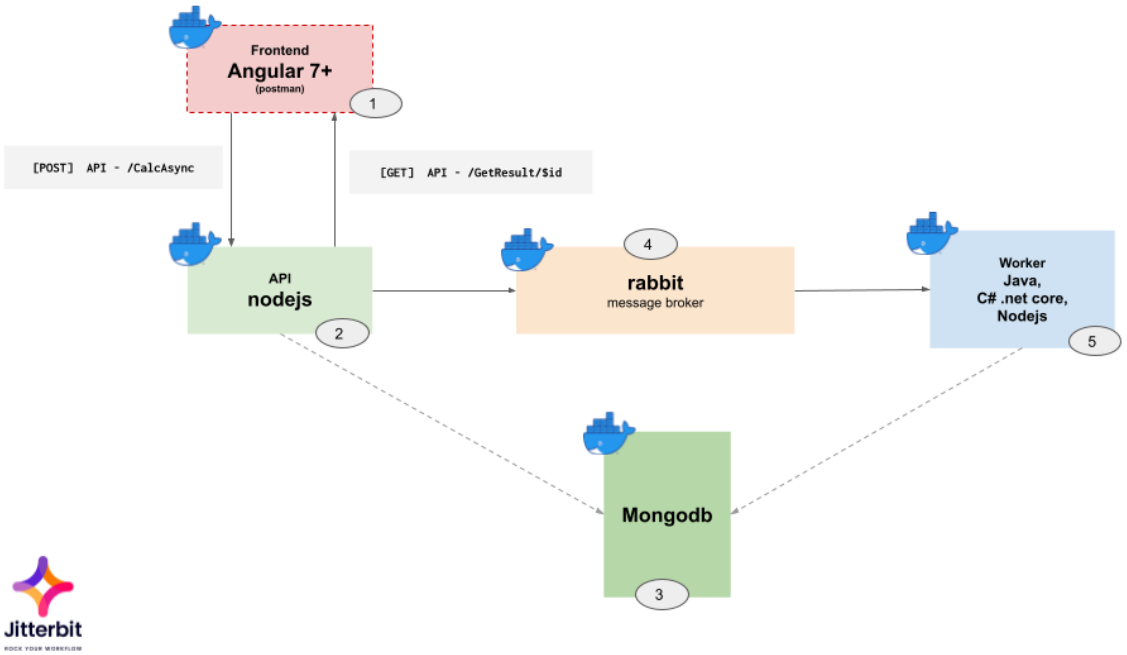
Nome do candidato(a):	Dieter Marno Araújo dos Santos
Telefone:	+5551999315862
Linkedin:	https://www.linkedin.com/in/dietermarno/
Data de realização:	29/06/2023

Informe o endereço github dos projetos realizados:

Repositório Backend:	https://github.com/dietermarno/jitterbit-test.git
Repositório Frontend:	https://github.com/dietermarno/jitterbit-test.git

Instruções para o desenvolvimento:

- Utilizar GIT para gerenciar o Projeto
- Subir o código fonte para o seu repository Github
- Recomendamos utilizar Docker para cada aplicação (dockerfiles separados)
- Opcional - Desejável a aplicação das seguintes técnicas: (não obrigatórias)
    - TDD - Test Driven Development
    - DDD - Domain-Driven Design
    - ODM - Object Data Model
- Será utilizado neste teste as seguintes tecnologias e suas referências:
    - NodeJS;
    - C#;
    - Angular 7+;
    - MongoDB
    - RabittMQ
- A arquitetura do projeto se baseia em comunicação de forma assíncrona entre os microservices onde o objetivo é criar uma calculadora simples de única operação do tipo soma, onde:



Análise do desenho: Observe o desenho da arquitetura proposta, cada número no desenho é descrito nos passos abaixo:

1. **Frontend**: A tela**(1)** que fará a interação com o usuário é bem simples, precisa permitir que o usuário informe dois números e clique em um botão que irá fazer a chamada de uma api(/CalcAsync) transportando os números informados e os armazenando, após fazer a operação**(2)** irá retornar os dados

da identificação do registro para que o usuário consiga consultar(/GetResult/$id) o resultado da soma e saber se já foi concluída ou se ainda está pendente o resultado.

2. **API:** A API**(2)** irá armazenar um novo registro com os dois números que vieram do front**(1)** no banco de dados**(3)** com o status “pending”, após isso irá enviar uma mensagem(json) com a referência do registro armazenado para o Message Broker**(4)**.

3. **Worker**: A última etapa será uma aplicação Worker**(5)** que tem o objetivo de realizar a operação de soma, ela irá se conectar ao Message Broker**(4)** e irá consumir a mensagem que foi enviada pela API**(2)**.
   Depois, o Worker irá consultar o registro da soma no banco de dados**(3)** e atualizar o resultado da soma e também alterar o status para “done“ no banco de dados

MongoDB

O mongodb será o nosso banco de dados que terá o objetivo de armazenar os dados de soma.

- Referências em C#
  - Doc MongoDB C#/.NET Driver
  - Package MongoDB.Driver
  - Exemplos implementação

- Referências NodeJS
  - Doc MongoDB Node Driver
  - Package mongoose

- Estrutura de exemplo da collection, podendo seguir ou não o seguinte schema:
  - _id: ObjectId
  - number1: number
  - number2: number
  - result: number
  - calculationSatus: pending | error | done

RabbitMQ

O RabbitMQ tem o objetivo de ser o nosso message broker que irá fazer a comunicação assíncrona entre as aplicações:

- Tutorial C#
- Tutorial NodeJS
- RabbitMQ - Docker
- RabbitMQ - Serviço online - CloudAMQP

Entrega

O código desenvolvido deve estar commitado em um repositório público do GitHub do candidato e no **READ.ME** deve conter todas as informações e instruções para executar o projeto.

O que será avaliado
Cumprir todos os requisitos não quer dizer que um excelente código foi desenvolvido.

Queremos avaliar a capacidade de lógica, entrega e principalmente a capacidade de resolução de problemas.

Queremos entender como o código foi desenvolvido e como ele está organizado, quais os design-patterns foram implementados, quais técnicas foram utilizadas e o quão resiliente o código desenvolvido é.

Desejamos sucesso e boa sorte!