**Descrição**

**Sistema 1**

O sistema 1 foi desenvolvido utilizando o micro-framework Flask, juntamente com a biblioteca SqlAlchemy para acesso ao banco de dados

Para solicitação de latitude e longitude do endereço recebido, utilizei a biblioteca Pika, responsável por adicionar mensagens ao RabbitMQ

**Sistema 2**

Para consumir as mensagens do RabbitMQ, utlizei a mesma biblioteca responsável por colocar mensagens no RabbitMQ utilizada no Sistema 1

Foi acesso ao Sistema 1, foi utilizado a biblioteca Requests

**Sistema de filas**

Foi utilizar o RabbitMQ em um Docker

**Banco de dados**

Durante os testes foi utilizado SQlite. Devido ao Flask utilizar SQLAlchemy, o tipo de banco pode ser alterado.

**Funcionamento**

O Sistema 1 apresenta duas rotas: Uma para cadastro de serviço e outra para atualização do serviço.

Endpoint 1: ./order

Endpoint 2: ./order\_update

Quando uma requisição chega ao endpoint /order, o sistema adiciona o pedido ao banco de dados e coloca um nova mensagem na fila “service” do RabbitMQ, contendo dados do endereço e id do serviço.

O sistema 2 escuta a fila “service” do RabbitMQ. Quando há mensagem, recebe o conteúdo e acessa a api MapQuest. MapQuest é uma api open source semelhante ao googlemaps. Preferi utilizar essa opção pois não possuo conta no googleCloud ( não quero colocar meu cartão de crédito lá ).

Finalizado o acesso a MapQuest, o sistema 2 envia ao sistema 1 pela rota “order\_update”, os dados retornados pela MapQuest.

Ao receber os dados do Sistema 2, o sistema 1 atualiza os dados do pedido acessando o banco de dados.