

# Pedidos

Sistema que recebe pedidos de clientes e guarda em base de dados para que possa ser feito relatórios de pedidos de seus clientes.

## Planejamento

Montagem do ambiente de desenvolvimento. Estimativa: 1 hora. Concluído em: 1 hora

- Baixar e instalação do Java v.21
- Configuração da IDE(IntelliJ Idea) e do banco(Postgres) de dados em local e Docker;

Desenvolvimento de Service, Controller, repositório e entidades. Estimativa: 6 horas. Concluído em 5 horas

- Desenvolvimento dos controllers;
- Desenvolvimento da camada de Entidades e repositório;
- Desenvolvimento da camada de Service

Consumo dos pedidos por messageira. Estimativa de 2 horas. Concluído em 3 horas

- Desenvolvimento do rabbit no docker e configurado para o projeto consumir;
- Criação da exchange, Queue e feito um binding com routing-key;

Deploy da aplicação. Estimativa: 3 horas. Concluído em 4 horas

- Configuração de conta na nuvem no Render e CloudAmqp por permitir conta free;
- Verificado se Base de dados estava instanciada;
- Deploy da aplicação;

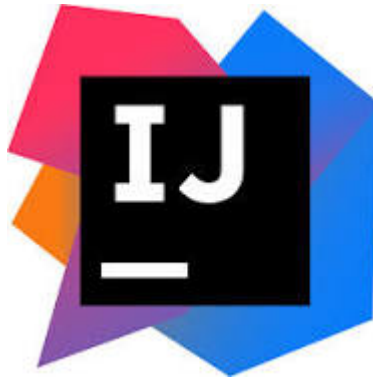
Documentação e teste manual da aplicação. Estimativa de 2 horas. Concluída em 2 horas

- Analise e escrita do escopo;
- Validações das funcionalidades;
- Gerações de evidências;

## Tecnologias Utilizadas

- Linguagem: Java 21
- IDE: IntelliJ Idea
- Framework: Spring Boot
- Banco de dados: PostgreSQL
- Mensageria: RabbitMQ
- Postman

- Link da aplicação: [Swagger](https://pedidos-0hqf.onrender.com/swagger-ui/index.html) <https://pedidos-0hqf.onrender.com/swagger-ui/index.html>



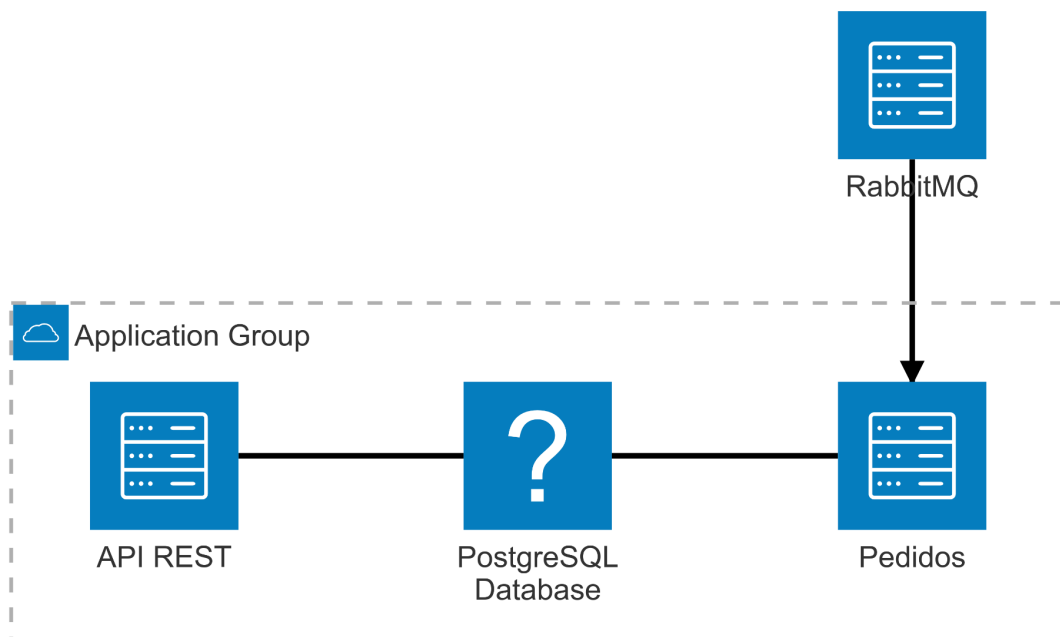
PostgreSQL



RabbitMQ

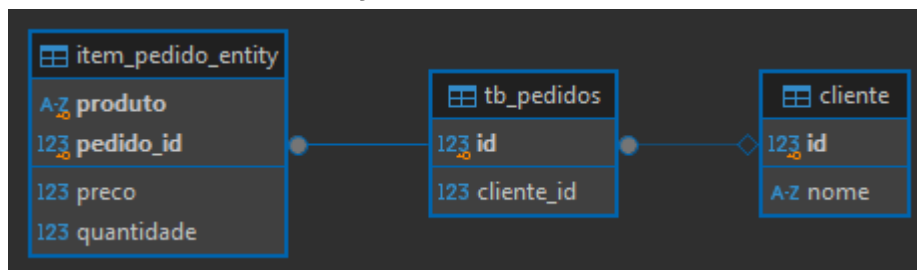


## Diagrama de arquitetura

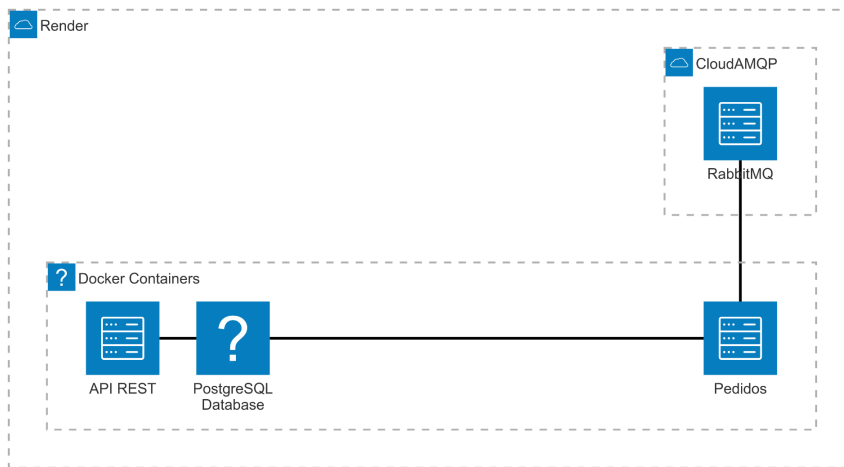


## Modelagem de dados

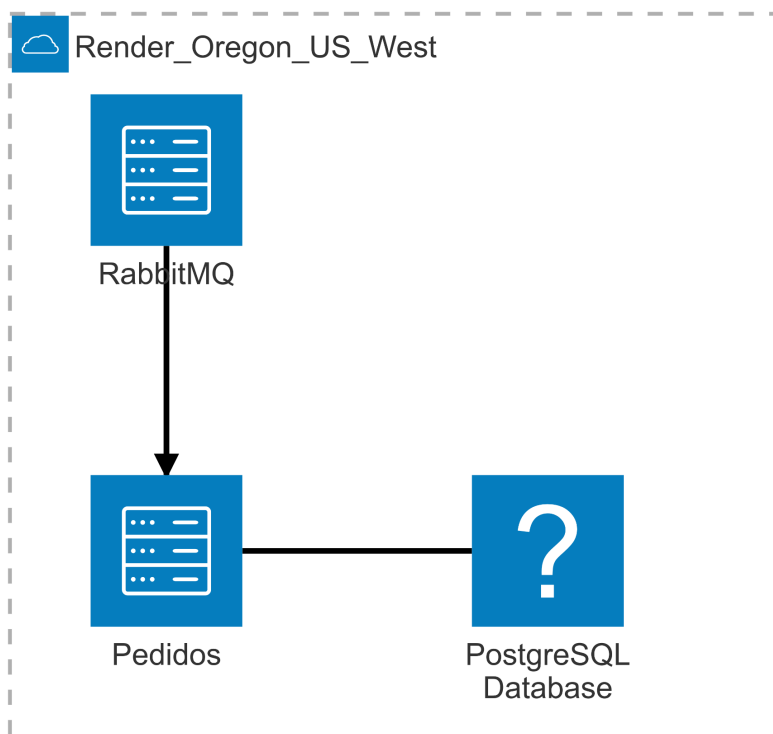
Por se tratar de uma aplicação simples foi criado apenas 3 tabelas



## Diagrama de implantação



## Diagrama de infra



### Descrição do Diagrama de Infraestrutura

A aplicação "Pedidos" está hospedada na Render na região de Oregon (US West).

A aplicação possui recursos de 0.1 CPU e 512 MB de memória.

A aplicação está registrada no repositório GitHub.

<https://github.com/WellingtonMJose/pedidos>

A URL da aplicação é <https://pedidos-0hqf.onrender.com>.

## Evidência de testes

Imagem da exchange criada, já com o binding na Queue

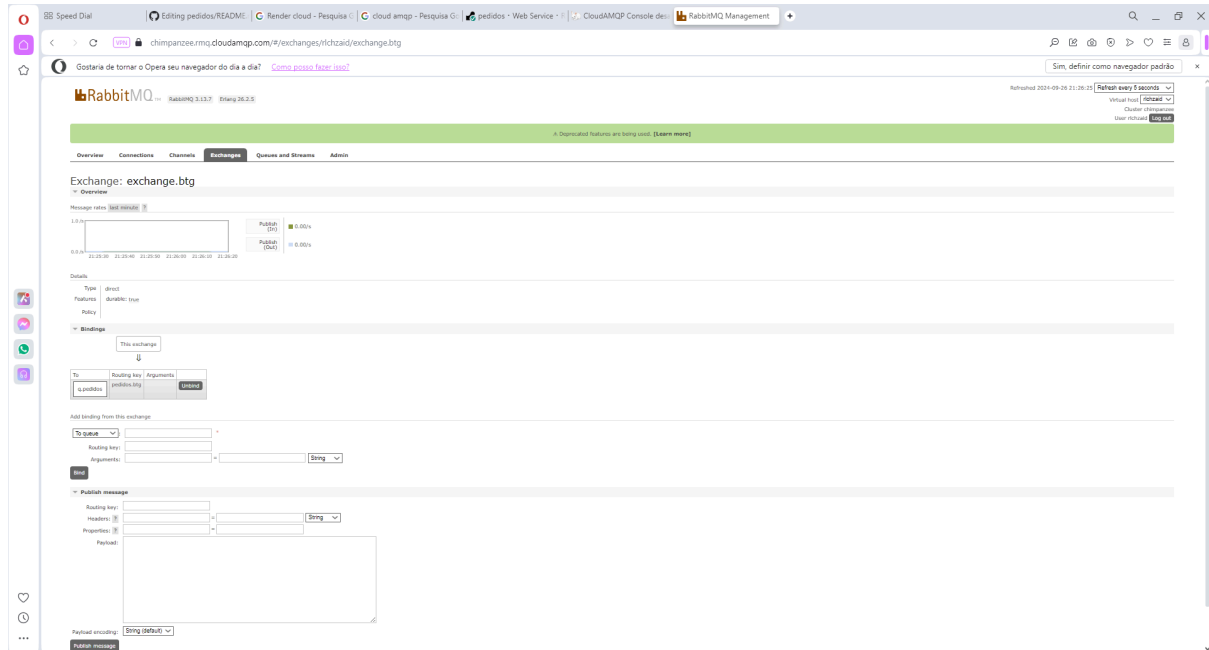


Imagem do envio de um pedido

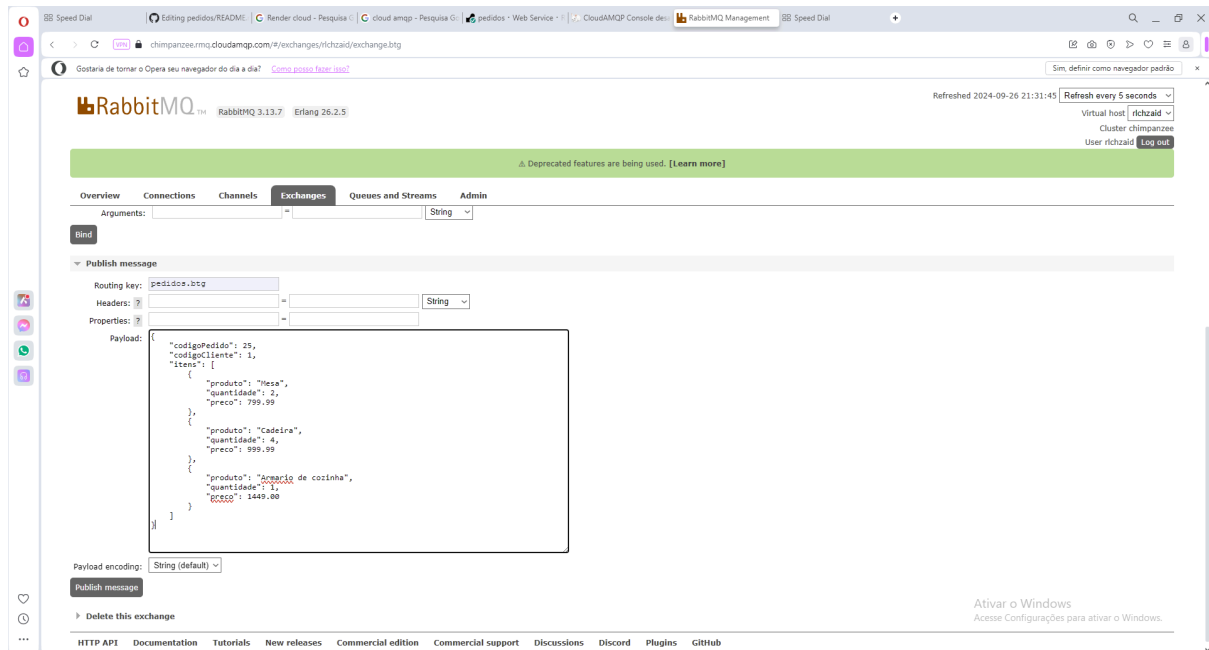
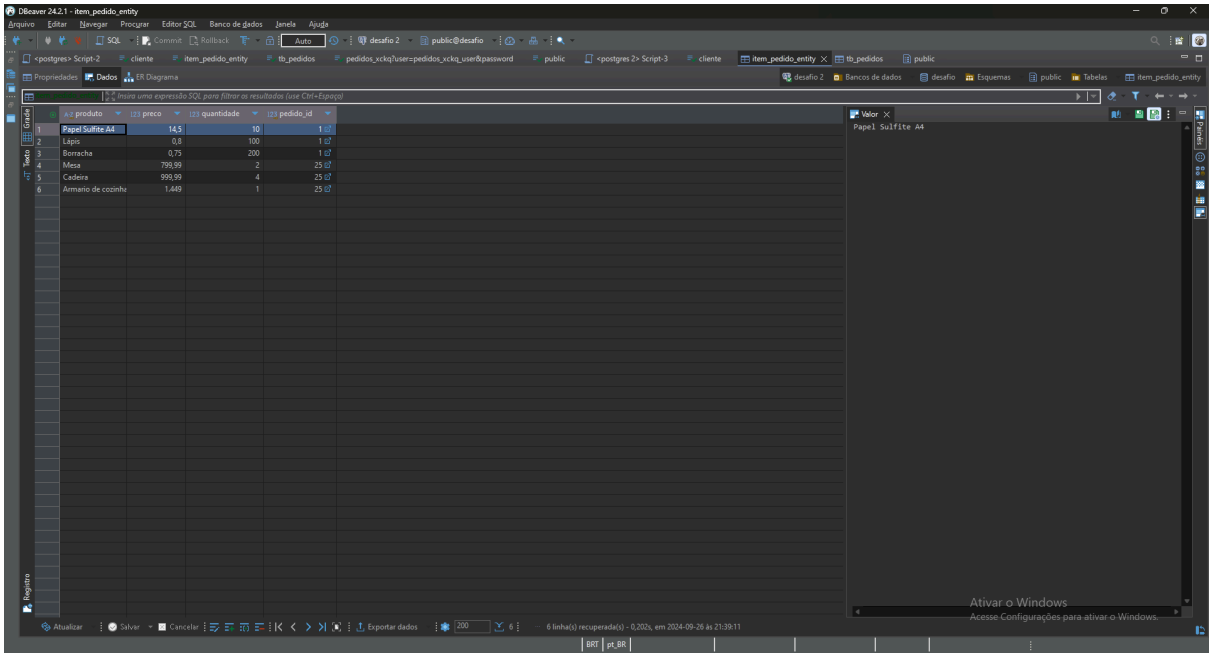
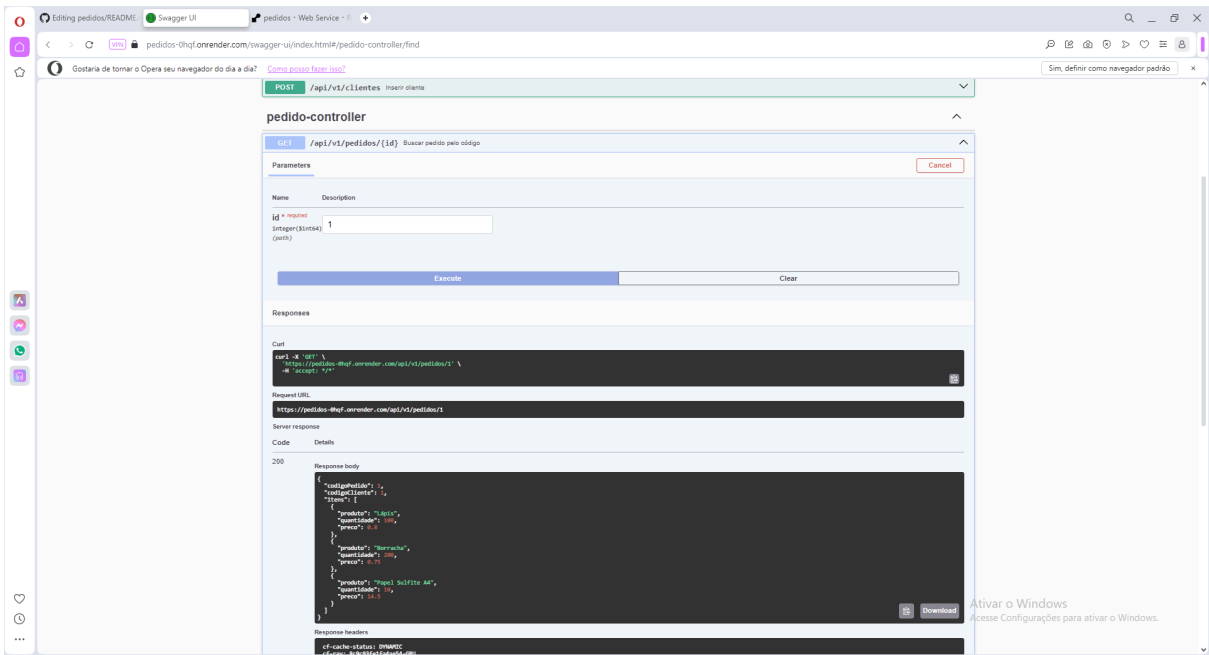


Imagem dos dados salvos na tabela de item\_pedido



	id	produto	preco	quantidade	pedido_id
1	1	Papel Sulfite A4	14.5	10	1
2	2	Lápis	0.8	100	1
3	3	Borracha	0.15	200	1
4	4	Mesa	799.99	2	25
5	5	Cadeira	999.99	4	25
6	6	Armário de cozinha	1.449	1	25

Imagem da busca do pedido via id



pedidos - Web Service

pedido-controller

GET /api/v1/pedidos/{id} Buscar pedido pelo id

Parameters

Name	Description
id	required integer (int64) (path)

Execute Clear

Responses

Curl

```
curl -X GET \
  https://pedidos-0hgf.onrender.com/api/v1/pedidos/1 \
  -H 'accept: */*'
```

Request URL

https://pedidos-0hgf.onrender.com/api/v1/pedidos/1

Server response

Code Details

200

Response body

```
{
  "codigoPedido": 1,
  "codigoCliente": 1,
  "items": [
    {
      "produto": "Borracha",
      "quantidade": 200,
      "preco": 0.15
    },
    {
      "produto": "Lápis",
      "quantidade": 100,
      "preco": 0.8
    },
    {
      "produto": "Papel Sulfite A4",
      "quantidade": 10,
      "preco": 14.5
    }
  ]
}
```

Response headers

```
cf-cache-status: DYNAMIC
cf-cache-status: DYNAMIC
```

## Imagem da busca de todos os pedidos do cliente via id

GET /api/v1/pedidos/cliente/{id} Listar todos pedidos pelo id do cliente

Parameters

Name	Description
id	required integer (int32) (path)

Execute Clear

Responses

200

```
{
  "id": 1,
  "cliente": {
    "id": 1,
    "nome": "Maria da Silva"
  },
  "items": [
    {
      "quantidade": 10,
      "preco": 10,
      "subtotal": 100
    },
    {
      "quantidade": 100,
      "preco": 1,
      "subtotal": 100
    },
    {
      "quantidade": 1000,
      "preco": 10,
      "subtotal": 10000
    }
  ],
  "totalFormatado": "R$ 970,00",
  "valorTotal": 10100
}
```

Response headers

```
cf-cache-status: DYNAMIC
cf-ray: 8b3b7f79e6a1d8d0
```

## Imagem da busca de quantidade de pedidos do cliente via id

GET /api/v1/pedidos/cliente/quantidade/{id} Buscar quantidade de pedidos feito pelo id do cliente

Parameters

Name	Description
id	required integer (int32) (path)

Execute Clear

Responses

200

```
{
  "total": 2
}
```

Response headers

```
cf-cache-status: DYNAMIC
cf-ray: 8b3b7f79e6a1d8d0
content-encoding: br
content-length: 20
content-type: application/json; charset=utf-8
date: Fri, 27 Sep 2024 15:43:13 GMT
etag: W/30211708-ncw
server: cloudflare
vary: Accept-Encoding
x-render-origin-server: Render
```

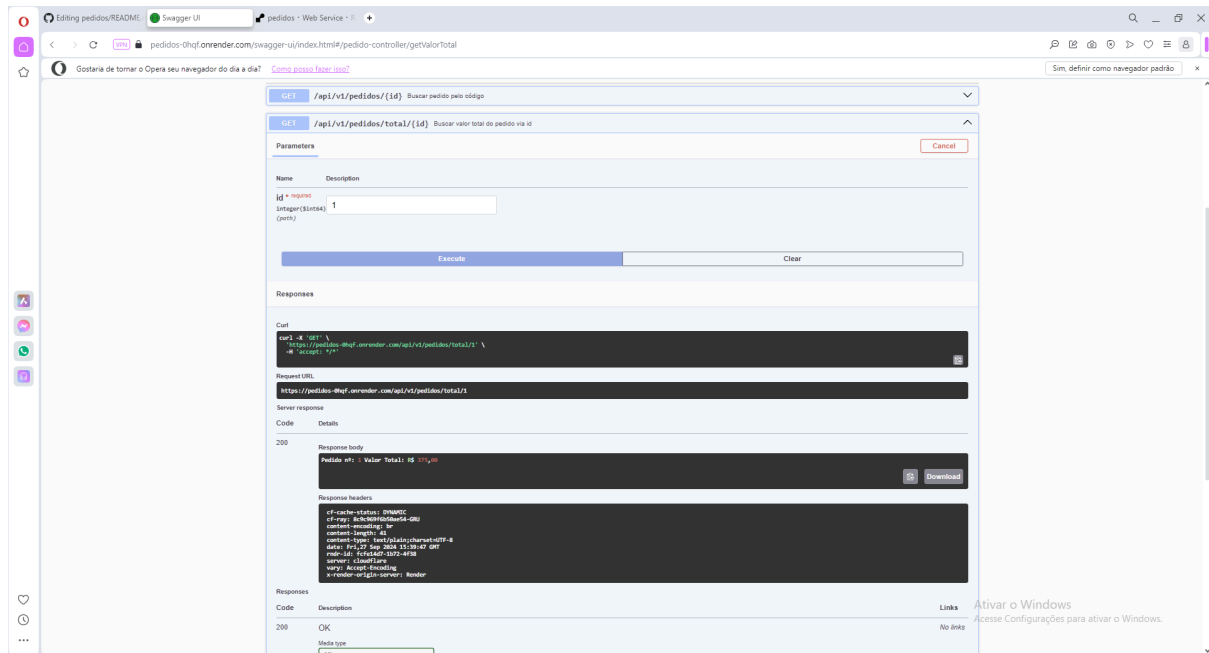
Responses

Code	Description	Links
200	OK	No links

Media type

\*/\*

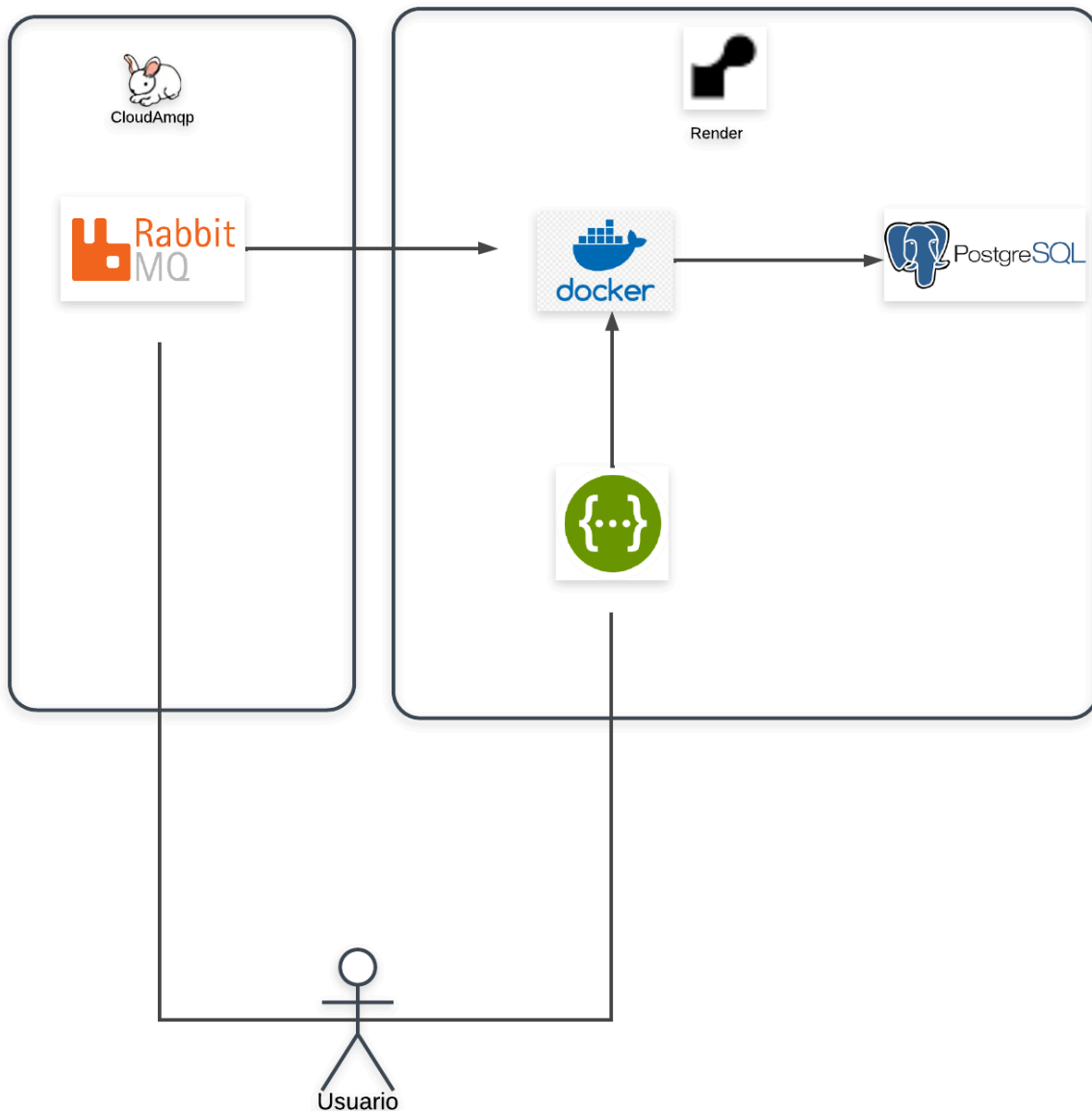
## Imagem da busca de valor total do pedido do cliente via id



A aplicação adota uma arquitetura monolítica, centralizando toda a lógica de negócio em um único aplicativo.

Essa escolha foi feita devido à simplicidade do projeto e ao baixo volume de transações esperado.





#### API REST:

Expor a funcionalidade da aplicação através de uma API REST.

Serviço de negócios: Contém a lógica de negócio da aplicação, incluindo o processamento de pedidos e a geração de relatórios.

Banco de dados: Armazena as informações dos pedidos.

Event-Driven:

Mensageria: Utilizado para comunicação assíncrona entre os componentes.

#### Obs:

Foi incluído um endpoint para auxiliar os testes, é um serviço com o método POST para salvar o cliente trazendo mais facilidades nos testes, não estava no escopo do projeto mas foi necessário pela praticidade.