

Gateway de Pagamentos

Api desenvolvida em **Java 21** com **Spring Boot 3.5.6**, simulando um sistema de pagamentos com validação externa.

Pré-requisitos

- Clone o repositório:

```
git clone https://github.com/Alhexx/paymengateway
```

Configuração do Backend (Spring Boot)

Comandos para a execução:

1. Configuração de um `.env`:
 - Crie um arquivo `.env` e preencha com:

```
# Banco de dados
SPRING_DATASOURCE_USERNAME=postgres
SPRING_DATASOURCE_PASSWORD=postgres
SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/paymentgateway

# JWT
SECURITY_JWT_SECRET=secret-de-testes-para-localhostt

# URL externa
EXTERNAL_PAYMENT_VALIDATOR_URL=https://zsy6tx7aql.execute-api.sa-east-1.amazonaws.com/authorizer
```

2. Execute o Docker compose para ter um container do postgres já configurado para a aplicação:

```
docker compose up -d --build
```

Apos a inicialização do serviço:

- Acesse <http://localhost:8080/swagger-ui/index.html#/> para visualizar a documentação interativa das rotas.

Sugestão de fluxo para testes

1 Criação de usuários:

- Pela rota `/users`
 - Recomendada a criação de no mínimo 2 usuários para testar todas as funcionalidades:

```
{
  "name": "umberto",
  "email": "umberto@gmail.com",
  "cpf": "57994580017",
  "password": "teste123!"
}
```

```
{
  "name": "doisberto",
  "email": "doisberto@gmail.com",
  "cpf": "57831242066",
  "password": "teste123!"
}
```

2 Login com doisberto:

- Pela rota `/auth`

```
{
  "emailOrCpf": "57831242066",
  "password": "teste123!"
}
```

- Após o login clique no cadeado do swagger e copie o token retornado para lá

3 Deposito para doisberto

- Pela rota `/accounts/deposit`

```
{
  "amount": 200.01
}
```

4 Login com umberto:

- Pela rota `/auth`

```
{
  "emailOrCpf": "57994580017",
```

```
"password": "teste123!"  
}
```

- Após o login clique no cadeado do swagger e copie o token retornado para lá

5 Criação de cobranças:

- Pela rota `/billings`
 - Umberto vai criar uma cobrança para o Doisberto
 - Criar pelo menos 2

```
{  
  "value": 43.5,  
  "receiverCPF": "57831242066",  
  "description": "cobrança qualquer" # Detalhe obrigatório, ninguém paga algo sem  
  saber o que é  
}
```

```
{  
  "value": 150.5,  
  "receiverCPF": "57831242066",  
  "description": "cobrança qualquer" # Detalhe obrigatório, ninguém paga algo sem  
  saber o que é  
}
```

5 Login com doisberto:

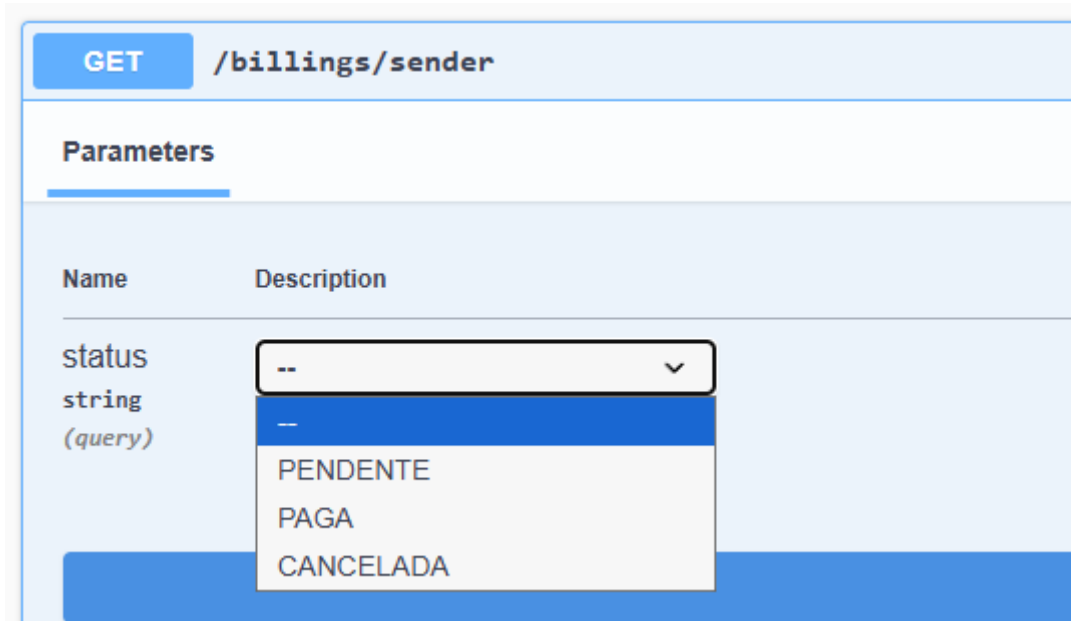
- Pela rota `/auth`

```
{  
  "emailOrCpf": "57831242066",  
  "password": "teste123!"  
}
```

- Após o login clique no cadeado do swagger e copie o token retornado para lá

7 Listagem de cobranças

- Pela rota `/billings/sender`
 - Listagem de cobranças por criadas e seus status

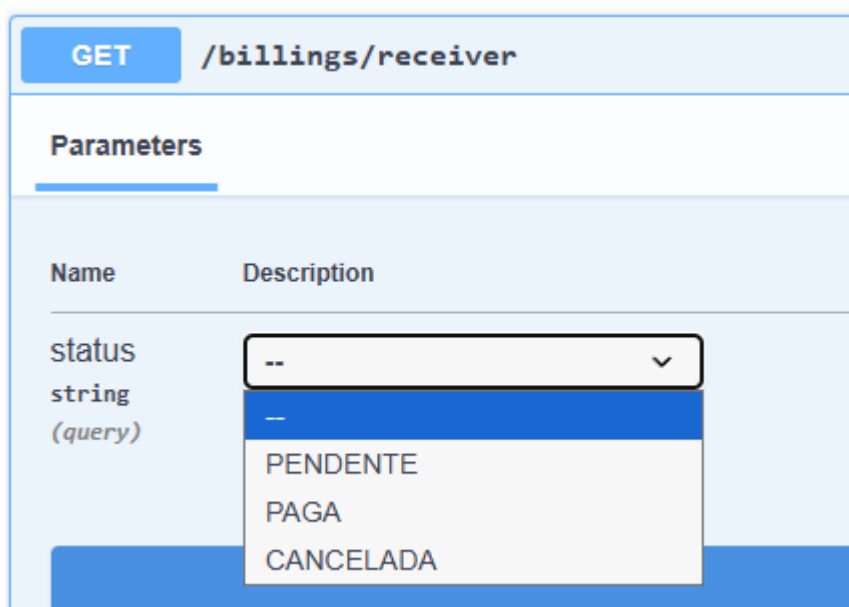


GET `/billings/sender`

Parameters

Name	Description
status	<input type="text" value="--"/>
string	—
(query)	PENDENTE PAGA CANCELADA

- Pela rota `/billings/receiver`
 - Listagem de cobranças por recebidas e seus status



GET `/billings/receiver`

Parameters

Name	Description
status	<input type="text" value="--"/>
string	—
(query)	PENDENTE PAGA CANCELADA

8 Pagamento de cobranças

- Pela rota `/billings/pay`
 - Pagamento por cartão

```
{
  "billingId": 2,
  "paymentMethod": "CARTAO",
  "cardNumber": "4111111111111111",
  "cvv": "123",
  "cardExpiryDate": "2027-05"
}
```

- Pagamento por saldo

```
{
  "billingId": 1,
  "paymentMethod": "SALDO"
}
```

9 Cancelar pagamento

- Pela rota `/billings/{billing_id}/cancel`
 - Informe o ID da cobrança (tivemos uma com cartão e outra com saldo)

The screenshot shows a REST client interface with a blue header bar. On the left, a blue button labeled 'GET' is next to the URL `/billings/{billing_id}/cancel`. Below the header, the word 'Parameters' is underlined. A table with two columns, 'Name' and 'Description', is shown. The first row has 'Name' as `billing_id` (with a red asterisk and 'required' text) and 'Description' as `integer($int32)` (with '(path)' below it). To the right of the description is a text input field containing the number '2'. At the bottom right of the interface is a blue button labeled 'Execute'.

Name	Description
<code>billing_id</code> * required	<code>integer(\$int32)</code> (path)

Observações

- O fluxo é apenas uma sugestão e a api pode ser testada livremente
- Api consta com validação de CPF
- Api consta com tratamento de exceções, esperadas e inesperadas
- JWT implementado, token expirando 10 minutos, por padrão
- Arquitetura baseada em camadas inspirada em DDD
- Commits e escrita do código em inglês, mas erros todos em português

Testes Unitários

- Foco dos testes foi em apenas algumas funcionalidades dos services, sem foco em chegar a 100% de coverage

paymengateway

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.challenge.paymengateway.application.service	<div><div></div></div>	55%	<div><div></div></div>	36%	33	49	62	159	14	26	0	4
com.challenge.paymengateway.config.security	<div><div></div></div>	13%	<div><div></div></div>	14%	23	26	62	73	16	19	1	4
com.challenge.paymengateway.application.dto	<div><div></div></div>	66%	<div><div></div></div>	0%	25	45	34	77	18	38	3	12
com.challenge.paymengateway.application.controller	<div><div></div></div>	19%	<div><div></div></div>	0%	10	14	20	32	9	13	0	4
com.challenge.paymengateway.common.exceptions	<div><div></div></div>	4%	<div><div></div></div>	n/a	6	7	15	16	6	7	1	2
com.challenge.paymengateway.application.model	<div><div></div></div>	77%	<div><div></div></div>	50%	12	46	14	75	10	44	0	4
com.challenge.paymengateway.common.components	<div><div></div></div>	12%	<div><div></div></div>	0%	5	6	9	12	3	4	2	3
com.challenge.paymengateway.common.utils	<div><div></div></div>	85%	<div><div></div></div>	59%	9	13	3	15	1	2	0	1
com.challenge.paymengateway		37%		n/a	1	2	2	3	1	2	0	1
com.challenge.paymengateway.config	<div><div></div></div>	100%		n/a	0	12	0	37	0	12	0	4
com.challenge.paymengateway.common.enums	<div><div></div></div>	100%		n/a	0	2	0	7	0	2	0	2
Total	920 of 2,042	54%	72 of 106	32%	124	222	221	506	78	169	7	41