

Gateway de Pagamentos

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

Arquitetura do projeto

```
+---src
|   +---main
|   |   +---java
|   |   |   \---com
|   |   |       \---challenge
|   |   |           \---paymengateway
|   |   |               |   PaymengatewayApplication.java # Classe principal do Spring Boot
|   |   |               |
|   |   |               +---application
|   |   |                   |   +---controller # Recebe requisições HTTP e retorna respostas
|   |   |                   |   +---dto # Objetos de transferência de dados
|   |   |                   |   +---model # Entidades do domínio (User, Account, Billing,
|   |   |                   |       Payment)
|   |   |                   |   +---repository # Acesso e manipulação de dados no banco
|   |   |                   |   \---service # Lógica de negócio e regras do sistema
|   |   |                   |
|   |   |                   +---common
|   |   |                       |   +---components # Componentes auxiliares (ex: PaymentValidator)
|   |   |                       |   +---enums # Enums do domínio (PaymentMethods, StatusCobranca)
|   |   |                       |   +---exceptions # Tratamento de exceções globais e específicas
|   |   |                       |   \---utils # Funções utilitárias (ex: CPFUtils)
|   |   |                       |
|   |   |                       \---config
|   |   |                           +---security # JWT, filtros e serviços de autenticação
|   |   |                           +---SwaggerConfig.java # Configuração do Swagger
|   |   |                           +---CorsConfig.java # Configuração de CORS
|   |   |                           \---WebClientConfig.java # Configuração de chamadas HTTP externas
```

- A título de curiosidade, para gerar essa árvore basta usar o comando `tree /F /A`

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#/`](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 Depósito 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:

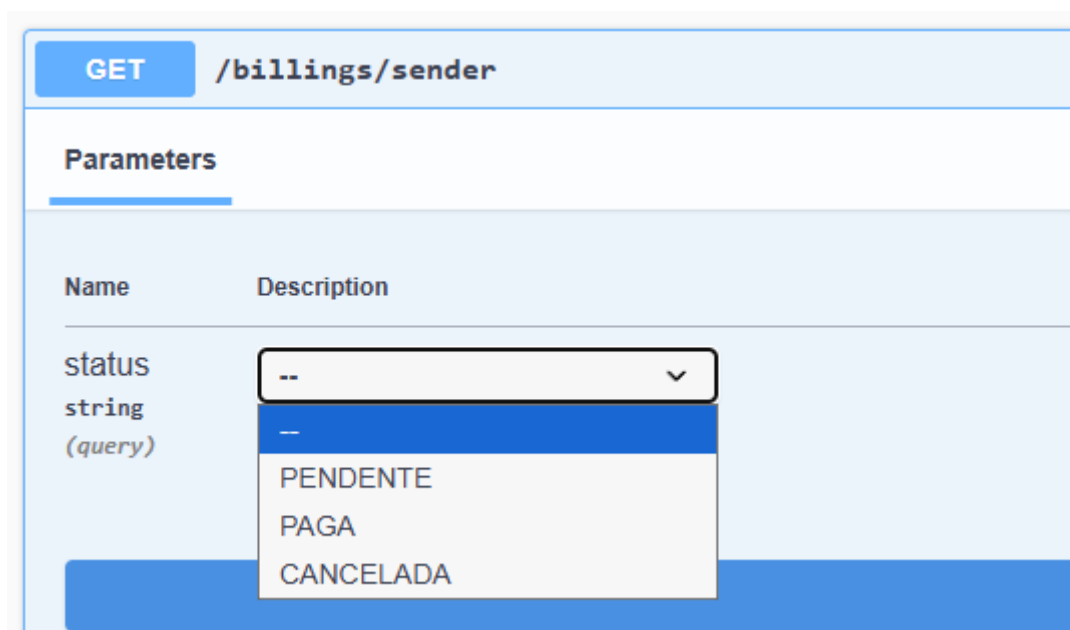
- Pela rota `/auth`

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

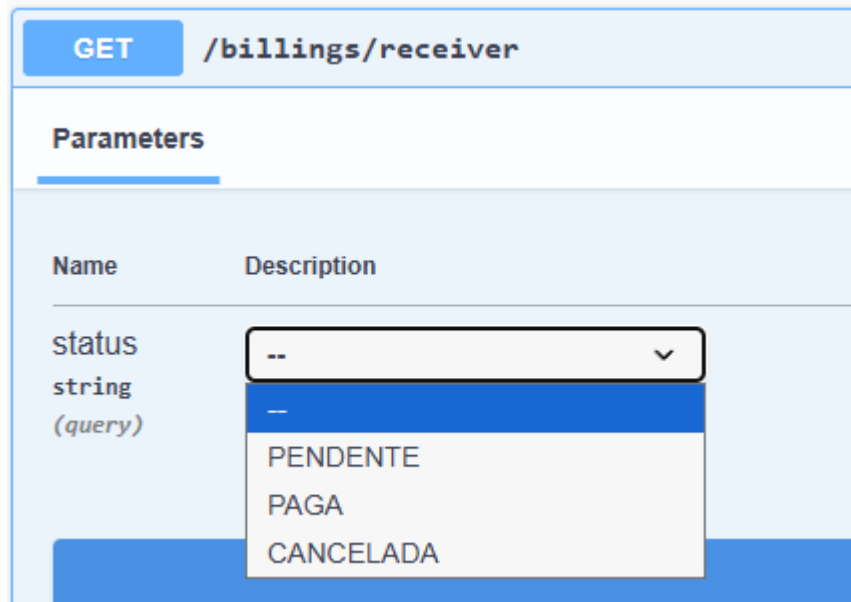
- Após o login clique no cadeado do swagger e copie o token retornado para lá

7 Listagem de cobranças

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



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



The screenshot shows a REST client interface for a GET request to the endpoint `/billings/receiver`. Under the 'Parameters' tab, there is a table with two columns: 'Name' and 'Description'. The 'Name' column contains the text 'status' followed by 'string' and '(query)' on separate lines. The 'Description' column contains a dropdown menu. The dropdown menu is currently open, showing a list of options: '--' (selected), '-', 'PENDENTE', 'PAGA', and 'CANCELADA'.

Name	Description
status string (query)	-- - PENDENTE PAGA CANCELADA

8 Pagamento de cobranças

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

```
{
  "billingId": 2,
  "paymentMethod": "CARTAO",
  "cardNumber": "411111111111111",
  "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)

GET

/billings/{billing_id}/cancel

Parameters

Name	Description
billing_id * required	
integer(\$int32)	
(path)	

2

Execute

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	Ctxty	Missed	Lines	Missed	Methods	Missed	Classes
com.challenge.paymengateway.application.service	<div></div>	55%	<div></div>	36%	33	49	62	159	14	26	0	4
com.challenge.paymengateway.config.security	<div></div>	13%	<div></div>	14%	23	26	62	73	16	19	1	4
com.challenge.paymengateway.application.dto	<div></div>	66%	<div></div>	0%	25	45	34	77	18	38	3	12
com.challenge.paymengateway.application.controller	<div></div>	19%	<div></div>	0%	10	14	20	32	9	13	0	4
com.challenge.paymengateway.common.exceptions	<div></div>	4%	<div></div>	n/a	6	7	15	16	6	7	1	2
com.challenge.paymengateway.application.model	<div></div>	77%	<div></div>	50%	12	46	14	75	10	44	0	4
com.challenge.paymengateway.common.components	<div></div>	12%	<div></div>	0%	5	6	9	12	3	4	2	3
com.challenge.paymengateway.common.utils	<div></div>	85%	<div></div>	59%	9	13	3	15	1	2	0	1
com.challenge.paymengateway	<div></div>	37%	<div></div>	n/a	1	2	2	3	1	2	0	1
com.challenge.paymengateway.config	<div></div>	100%	<div></div>	n/a	0	12	0	37	0	12	0	4
com.challenge.paymengateway.common.enums	<div></div>	100%	<div></div>	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