

Documentação Mini-Autenticador REST

1. Sobre

A VR processa todos os dias diversas transações de Vale Refeição e Vale Alimentação, entre outras. De forma breve, as transações saem das maquininhas de cartão e chegam até uma de nossas aplicações, conhecida como *autorizador*, que realiza uma série de verificações e análises. Essas também são conhecidas como *regras de autorização*.

Ao final do processo, o autorizador toma uma decisão, aprovando ou não a transação:

- se aprovada, o valor da transação é debitado do saldo disponível do benefício, e informamos à maquininha que tudo ocorreu bem.
- senão, apenas informamos o que impede a transação de ser feita e o processo se encerra.

Sua tarefa será construir um *mini-autorizador*. Este será uma aplicação Spring Boot com interface totalmente REST que permita:

- a criação de cartões (todo cartão deverá ser criado com um saldo inicial de R\$500,00)
- a obtenção de saldo do cartão
- a autorização de transações realizadas usando os cartões previamente criados como meio de pagamento

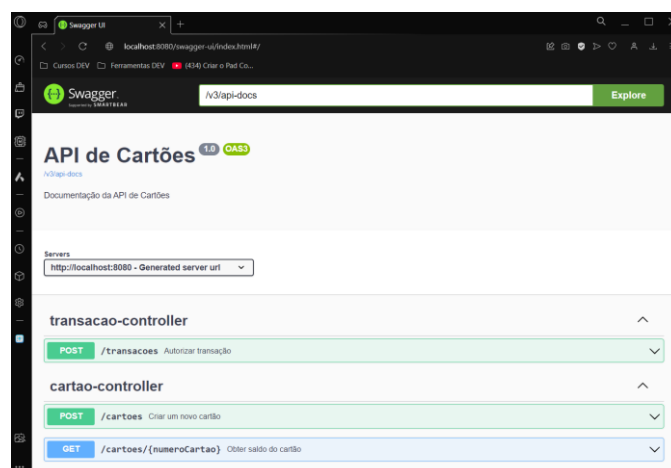
Uma transação pode ser autorizada se:

- o cartão existir
- a senha do cartão for a correta
- o cartão possuir saldo disponível

2. Demonstração das Funcionalidades

Para demonstração das funcionalidades está sendo usado o Swagger implementado na API, Acessando a url:

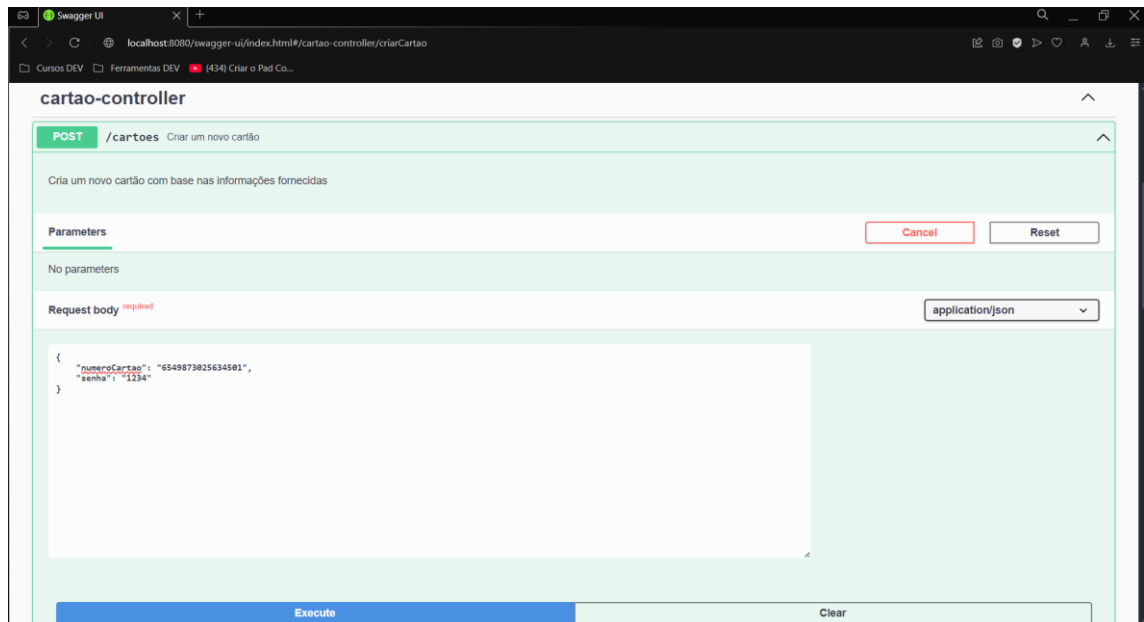
- <http://localhost:8080/swagger-ui/index.html#/>



Para Acessar é necessário utilizar login: **username** e senha: **password**.

2.1 Criar um novo cartão

2.1.1 Criar com sucesso



2.1.2 Erro ao criar, cartão já existe

cartao-controller

POST

/cartoes

Criar um novo cartão

Cria um novo cartão com base nas informações fornecidas

Parameters

Cancel

Reset

No parameters

Request body

required

application/json

```
{  "numeroCartao": "6549873025634501",  "senha": "1234"}  
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  "http://localhost:8080/cartoes" \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "numeroCartao": "6549873025634501",  "senha": "1234"}'  
```

Request URL

```
http://localhost:8080/cartoes
```

Server response

Code

Details

422

Error: response status is 422

Response body

```
{  "senha": "1234",  "numeroCartao": "6549873025634501"}  
```

Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate  connection: keep-alive  content-type: application/json  date: Wed, 13 Nov 2024 01:43:27 GMT  expires: 0  keep-alive: timeout=60  pragma: no-cache  transfer-encoding: chunked  x-content-type-options: nosniff  x-frame-options: DENY  x-xss-protection: 0  
```

2.2 Obter saldo do cartão

2.2.1 Saldo obtido com sucesso

GET

/cartoes/{numeroCartao}

Oblter saldo do cartão

Retorna o saldo do cartão com o número de cartão fornecido

Parameters

Cancel

Name	Description
numeroCartao * required	
string (path)	6549873025634501

ExecuteClear

Responses

Curl

curl -X 'GET' \n'http://localhost:8080/cartoes/6549873025634501' \n-H 'accept: */*'

Request URL

http://localhost:8080/cartoes/6549873025634501

Server response

Code	Details
200	<div><div>Response body</div><div>500</div><div>Download</div></div> <div><div>Response headers</div><div>cache-control: no-cache,no-store,max-age=0,must-revalidate\nconnection: keep-alive\ncontent-type: application/json\ndate: Wed,13 Nov 2024 01:46:10 GMT\nexpires: 0\nkeep-alive: timeout=60\npragma: no-cache\ntransfer-encoding: chunked\nx-content-type-options: nosniff\nx-frame-options: DENY\nx-xss-protection: 0</div></div>

2.2.2 Cartão não encontrado

GET

/cartoes/{numeroCartao}

Oblter saldo do cartão

Retorna o saldo do cartão com o número de cartão fornecido

Parameters

Cancel

Name	Description
numeroCartao * required	
string (path)	6549873025634502

ExecuteClear

Responses

Curl

curl -X 'GET' \n'http://localhost:8080/cartoes/6549873025634502' \n-H 'accept: */*'

Request URL

http://localhost:8080/cartoes/6549873025634502

Server response

Code	Details
404	<div><div>Error: response status is 404</div><div>Response headers</div><div>cache-control: no-cache,no-store,max-age=0,must-revalidate\nconnection: keep-alive\ncontent-length: 0\ncontent-type: text/plain;charset=UTF-8\ndate: Wed,13 Nov 2024 01:49:22 GMT\nexpires: 0\nkeep-alive: timeout=60\npragma: no-cache\nx-content-type-options: nosniff\nx-frame-options: DENY\nx-xss-protection: 0</div></div>

2.3 Autorizar transação

2.3.1 Transação autorizada com sucesso

POST

/transacoes

Autorizar transação

Autoriza uma transação com base nos dados fornecidos.

Parameters

Cancel

Reset

No parameters

Request body

required

application/json

{

"numeroCartao": "6549873025634501",

"senhaCartao": "1234",

"valor": "10.00"

}

Execute

Clear

Responses

Curl

curl -X 'POST' \

'http://localhost:8080/transacoes' \

-H 'accept: */*' \

-H 'Content-Type: application/json' \

-d '{

"numeroCartao": "6549873025634501",

"senhaCartao": "1234",

"valor": "10.00"

}'

Request URL

http://localhost:8080/transacoes

Server response

Code

Details

201

Response body

OK

Download

Response headers

cache-control: no-cache,no-store,max-age=0,must-revalidate

connection: keep-alive

content-length: 2

content-type: text/plain;charset=UTF-8

date: Wed,13 Nov 2024 02:11:59 GMT

expires: 0

keep-alive: timeout=60

pragma: no-cache

x-content-type-options: nosniff

x-frame-options: DENY

x-xss-protection: 0

Responses

2.3.2 Cartão não encontrado

POST

/transacoes Autorizar transação

Autoriza uma transação com base nos dados fornecidos.

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{  "numeroCartao": "6549873025634502",  "senhaCartao": "1234",  "valor": "10.00"}  
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  'http://localhost:8080/transacoes' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "numeroCartao": "6549873025634502",  "senhaCartao": "1234",  "valor": "10.00"}'  
```

Request URL

http://localhost:8080/transacoes

Server response

Code	Details
422	<div>Error: response status is 422</div> <div><div>Response body</div><div>CARTAO_INEXISTENTE</div><div>Download</div></div> <div><div>Response headers</div><div>cache-control: no-cache,no-store,max-age=0,must-revalidate connection: keep-alive content-length: 18 content-type: text/plain;charset=UTF-8 date: Wed, 13 Nov 2024 02:14:16 GMT expires: 0 keep-alive: timeout=60 pragma: no-cache x-content-type-options: nosniff x-frame-options: DENY x-xss-protection: 0</div></div>

2.3.3 Saldo insuficiente

POST

/transacoes Autorizar transação

Autoriza uma transação com base nos dados fornecidos.

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "numeroCartao": "6549873025634501",
  "senhaCartao": "1234",
  "valor": 1000.00
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/transacoes' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "numeroCartao": "6549873025634501",
    "senhaCartao": "1234",
    "valor": 1000.00
  }'
```

Request URL

http://localhost:8080/transacoes

Server response

Code	Details
422	<div>Error: response status is 422</div> <div><div>Response body</div><div><pre>SALDO_INSUFICIENTE</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><pre>cache-control: no-cache,no-store,max-age=0,must-revalidate connection: keep-alive content-length: 18 content-type: text/plain; charset=UTF-8 date: Wed, 13 Nov 2024 02:15:18 GMT expires: 0 keep-alive: timeout=60 pragma: no-cache x-content-type-options: nosniff x-frame-options: DENY</pre></div>

2.3.4 Senha inválida

POST

/transacoes Autorizar transação

⬆

Autoriza uma transação com base nos dados fornecidos.

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{  "numeroCartao": "6549873025634501",  "senhaCartao": "XXXX",  "valor": "10.00"}  
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  'http://localhost:8080/transacoes' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "numeroCartao": "6549873025634501",  "senhaCartao": "XXXX",  "valor": "10.00"}  '
```

Request URL

http://localhost:8080/transacoes

Server response

Code


Details

422

Error: response status is 422

Response body

SENHA_INVALIDA

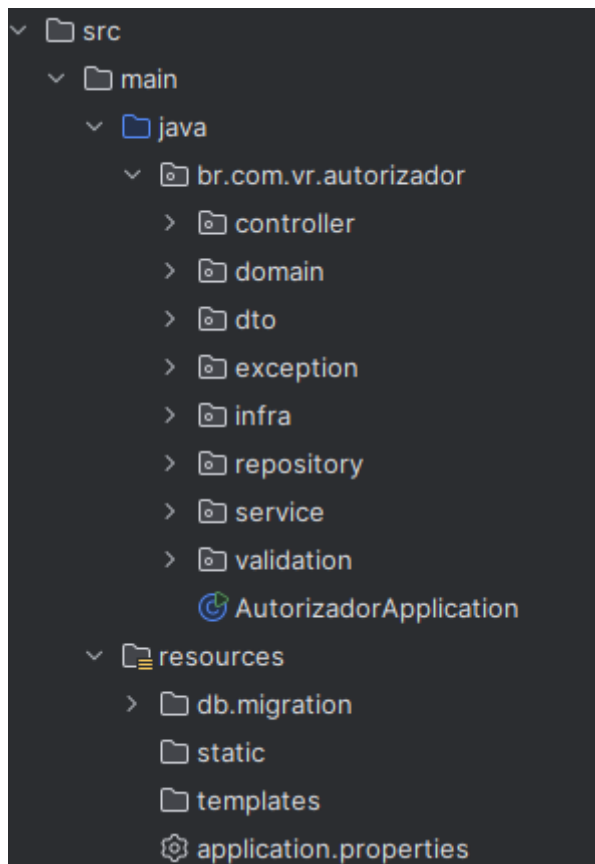
 Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate  connection: keep-alive  content-length: 14  content-type: text/plain;charset=UTF-8  date: Wed, 13 Nov 2024 02:16:07 GMT  expires: 0  keep-alive: timeout=60  pragma: no-cache  x-content-type-options: nosniff  x-frame-options: DENY
```


3. Como foi desenvolvido

Foram criados pacotes de acordo com a funcionalidades das classes













- controller – Classes responsáveis por lidar com as requisições http. Foram criadas duas Classes de Controle, Cartão controler e Transação controler.
- Domain – Classes de domínio do negócio, usada também para mapear a estrutura de tabelas do banco de dados. Para essa api foi criado apenas a classe Cartão.
- Dto – Foram criadas classes dto para passar de forma mais simples a informação, não exibindo as classes de domínio.
- Exception – Foram criadas classes personalizadas de exceção conforme a necessidade do negócio.
- Infra – pacote onde estão os arquivos de configuração de segurança e do Swagger.
- Repository – Classes responsáveis pela consulta e persistência das informações
- Service – Pacote onde estão as classes responsáveis pela regra de negócio.
- Validation – Classes de validação do cartão, Foi utilizado para validação o design Strategy, reduzindo a quantidade de if no código.
- O pacote db.migration estão os arquivos de migração da base de dados, Utilizado o flyway para isso.

4. Cobertura de testes

Para criação dos testes unitários foi utilizado bando de dados em memória H2.

Abaixo está a planilha da cobertura: Sendo focado nas classes que envolvem diretamente as funcionalidades.

Element	Class, % ^	Method, %	Line, %	Branch, %
✓  br.com.vr.autorizador	77% (14/18)	65% (23/35)	66% (48/72)	100% (4/4)
 AutorizadorApplication	0% (0/1)	0% (0/1)	0% (0/1)	100% (0/0)
>  infra	0% (0/2)	0% (0/5)	0% (0/17)	100% (0/0)
>  exception	80% (4/5)	40% (4/10)	50% (6/12)	100% (0/0)
>  repository	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
>  service	100% (1/1)	100% (3/3)	100% (13/13)	100% (0/0)
>  domain	100% (1/1)	100% (6/6)	100% (12/12)	100% (2/2)
>  controller	100% (2/2)	100% (3/3)	100% (5/5)	100% (0/0)
>  validation	100% (3/3)	100% (3/3)	100% (6/6)	100% (2/2)
>  dto	100% (3/3)	100% (4/4)	100% (6/6)	100% (0/0)

5. Como executar

Para executar basta ter instalado o Docker e ir ao diretório onde foi baixado o código e executar no prompt o comando:

- `docker-compose up --build`

Ao terminar de carregar já pode Fazer requests. Caso queira utilizar o Swagger utilizar endereço:

- <http://localhost:8080/swagger-ui/index.html#/>