E-BANK

(Documentação API)

Equipe 4:

Igor Silva Sobral

Israel Davi Travassos Verissimo

João Lucas Dantas Nunes Gomes

Sumário

1. Sobre

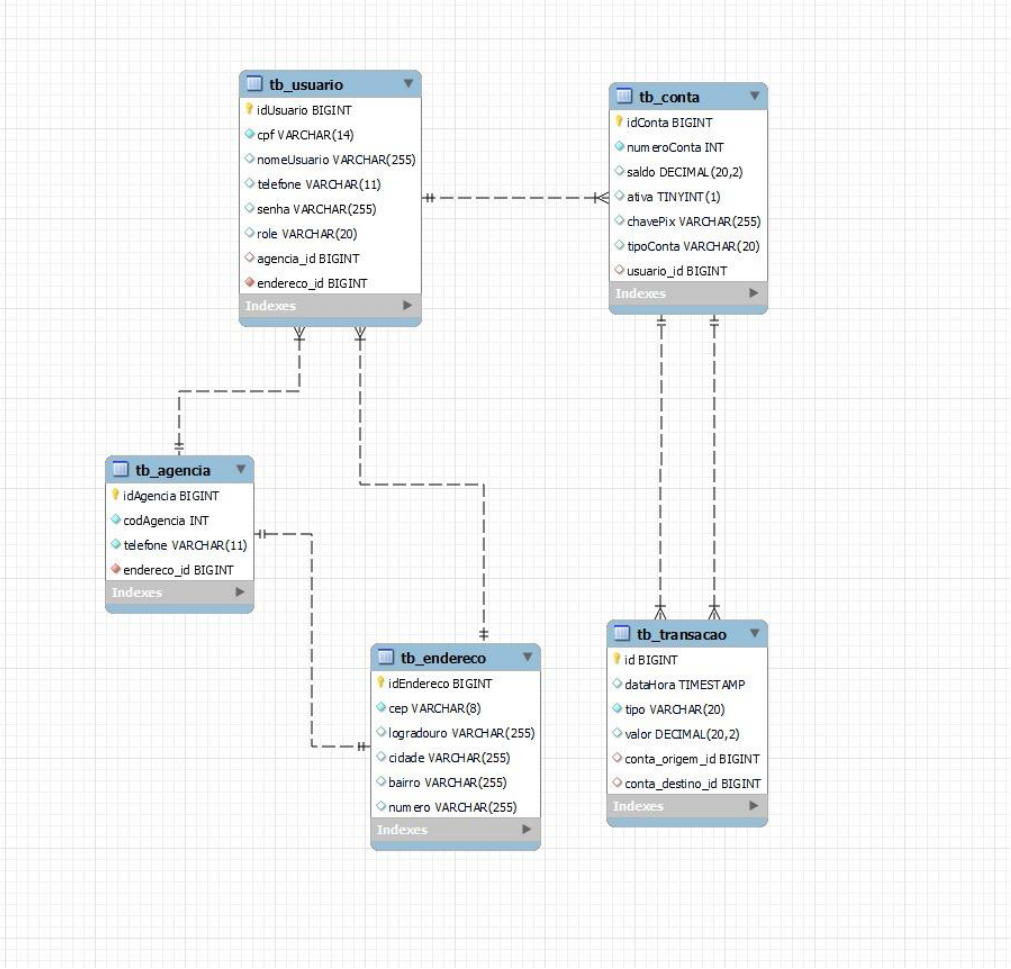
API para sistema bancário, consistindo em armazenar dados e simular operações comuns de bancos.

2. Introdução

Este documento fornecerá instruções para rápida integração aos serviços do eBank via HTTP (HTTP API).

3. Diagrama Entidade Relacionamento

Este tipo de diagrama é composto por entidades, atributos e relacionamentos, em que as entidades representam um objeto do mundo real e que possuem uma existência independente, segue abaixo a representação do projeto por meio do diagrama.

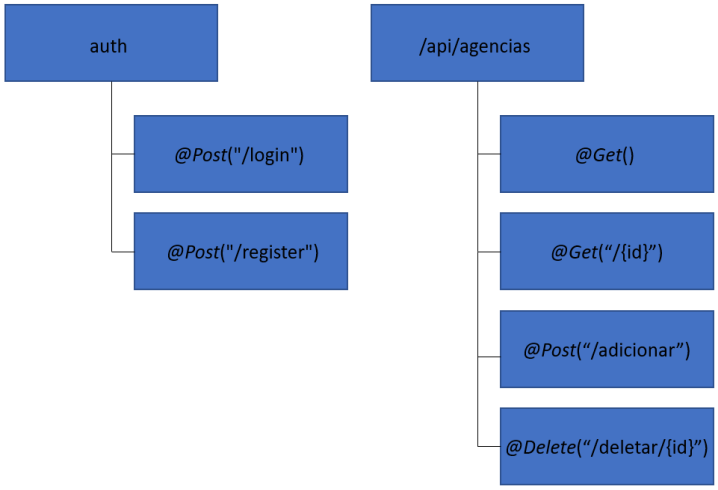


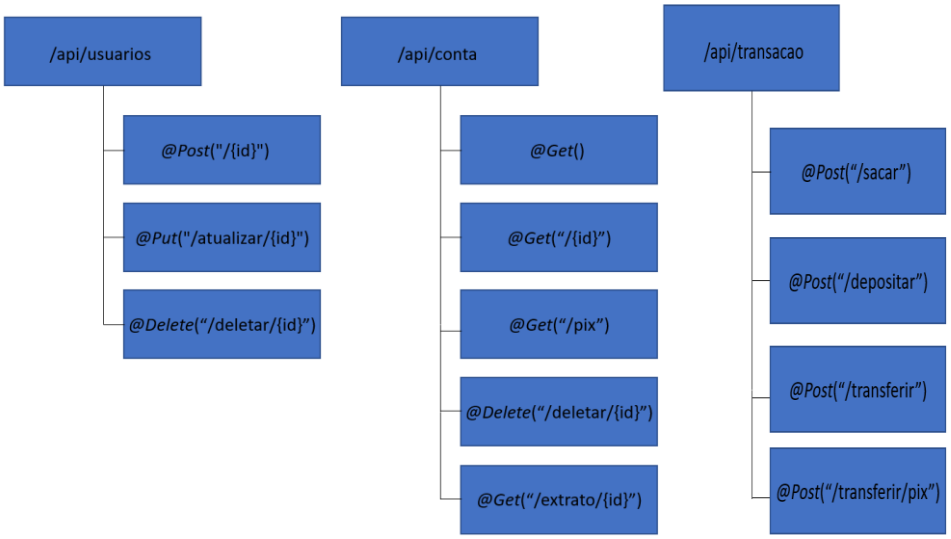
4. Premissas acordadas

Para combater qualquer tipo de problemas de acesso as rotas, definimos limitações de uso:

- Tirando o Login, todo *endpoint* acessado na aplicação exige um token de autenticação;
- Apenas um usuário do tipo ADMIN pode criar um novo usuário;
- Apenar um usuário do tipo ADMIN tem acesso aos endpoints de Agencia.

5. EndPoints do projeto

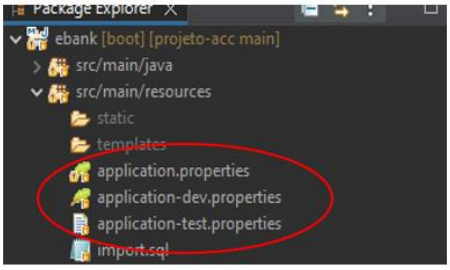




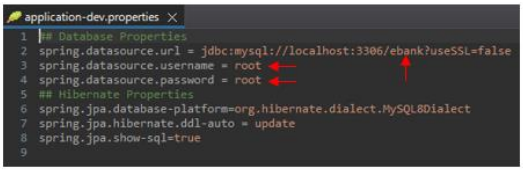
6. Como utilizar a API

6.1 Configurar Base de dados

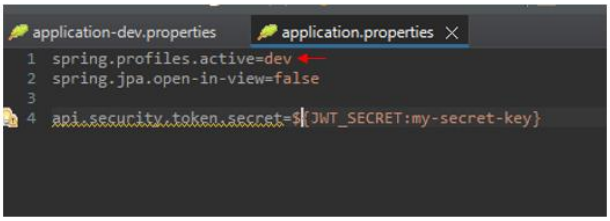
A API, apresenta duas formas de armazenamento de dados. Ela pode rodar na versão teste (H2) e na versão desenvolvimento (MySQL). Caso você escolha rodar a aplicação com MySQL deve configurar os seguintes campos da Figura II com os dados correspondentes ao datasource do Banco de dados. Além disso deve deixar a Figura III no campo destacado com dev.



(I)



(II)



(III)

Além disso, deve carregar a base de dados com alguns dados para facilitar na hora de testar todos os endpoints, segue abaixo um arquivo com código SQL para inserir dados nas tabelas:

data.sql

6.2 Autenticação

Alguns dados foram inseridos via SQL como mostra a Figura IV para se ter alguns dados de teste, é possível observar que as senhas são encriptadas no Banco de dados. Para realizar Login é necessário informar cpf e senha, vale ressaltar que o cpf precisa ser informado no formato (xxx.xxx.xxx-xx).

O retorno é um token, que será necessário para autenticação do usuário no sistema, e ele só conseguirá acessar as outras rotas informando o token.

Para Logar, será feita uma requisição do tipo POST para a url/auth/login

Request body:

```
{
    "cpf": "123.456.789-00",
    "senha": "1234"
}
```

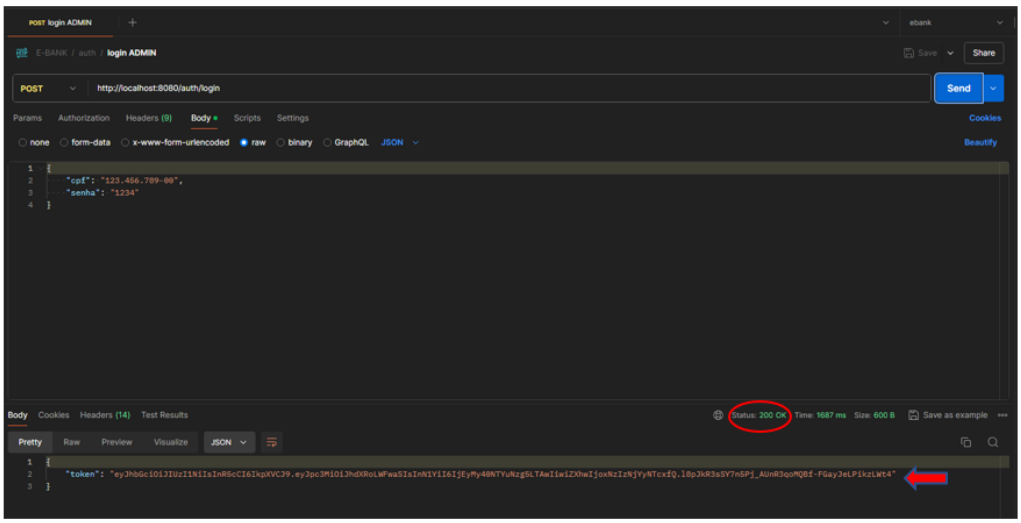
Respostas esperadas:

HTTP200: Sucesso (com body usado acima)

HTTP500: Usuário inexistente ou senha inválida

ROLE	AGENCIA_ID	ENDERECO_ID	ID_USUARIO	CPF	NOME_USUARIO	SENHA	TELEFONE
0	1	4	1	123.456.789-00	João Silva	$2a$12$N4g0VLAUu75chvfKd5mhuuzdmJhW85HZ3holFKguUiYJvfirwYHhq	1234445678
1	2	5	2	987.654.321-00	Maria Souza	$2a$12$N4g0VLAUu75chvfKd5mhuuzdmJhW85HZ3holFKguUiYJvfirwYHhq	1234885679

(IV)



(V)

6.3 Registro de usuário

Para Registrar um usuário, será feita uma requisição do tipo POST para a url/auth/register

Request body:

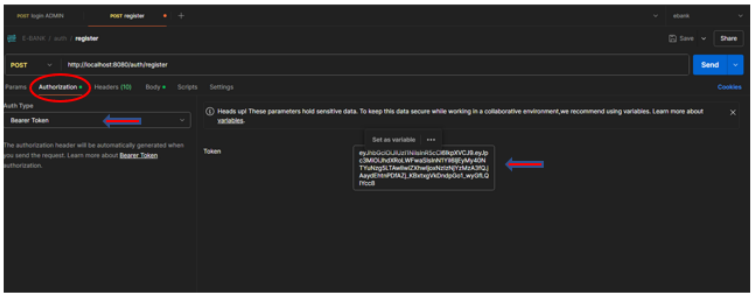
```
{
  "cpf": "987.655.341-00",
  "nomeUsuario": "Maria Oliveira",
  "telefone": "8398765432",
  "senha": "senhaSegura",
  "role": "USER",
  "idAgencia": 1,
  "endereco": {
    "cep": "56789-122",
    "logradouro": "Avenida Central",
    "cidade": "Cidade Nova",
    "bairro": "Centro",
    "numero": "789"
  },
  "contas": [
    {
      "saldo": 3000.00,
      "ativa": true,
      "chavePix": "chave-pix-nova",
      "tipoConta": "POUPANCA"
    }
  ]
}
```

Respostas esperadas:

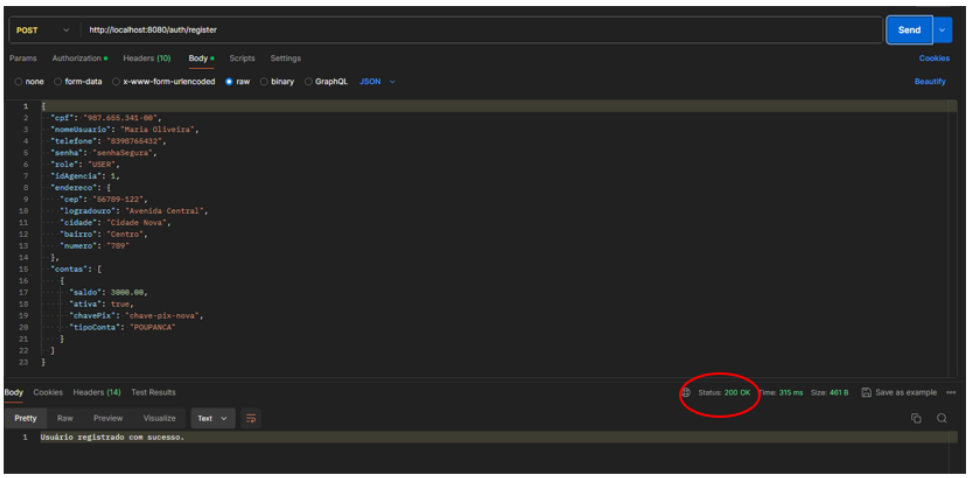
HTTP200: Sucesso (com body usado acima).

HTTP500: Se algum dado estiver incoerente com o esperado (cpf, cep, telefone, role, tipoConta).

HTTP403: Forbidden, caso um usuário não esteja logado ou um usuário do tipo USER tente cadastrar. (ROTA EXCLUSIVA DE USUÁRIOS DO TIPO ADMIN).



(VI)



(VII)

6.4 Listagem de Agências

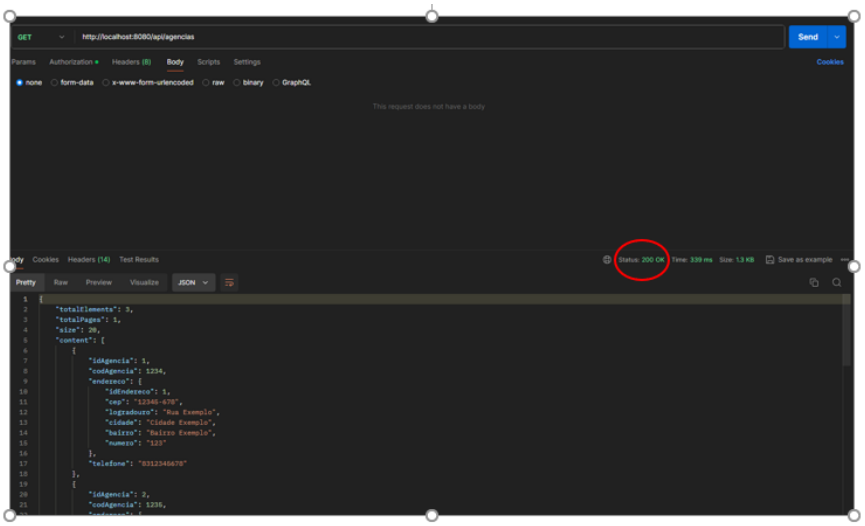
Para Listar as agências, será feita uma requisição do tipo GET para a url/api/agencias

Respostas esperadas:

HTTP200: Sucesso (Caso o usuário esteja logado e seja do tipo ADMIN).

HTTP403: Forbidden, caso esteja logado, mas seja do tipo USER.

HTTP401: Caso o usuário não esteja autenticado.



(VIII)

6.5 Postagem de Agência

Para Registrar um usuário, será feita uma requisição do tipo POST para a url/api/agencias/adicionar

Request body:

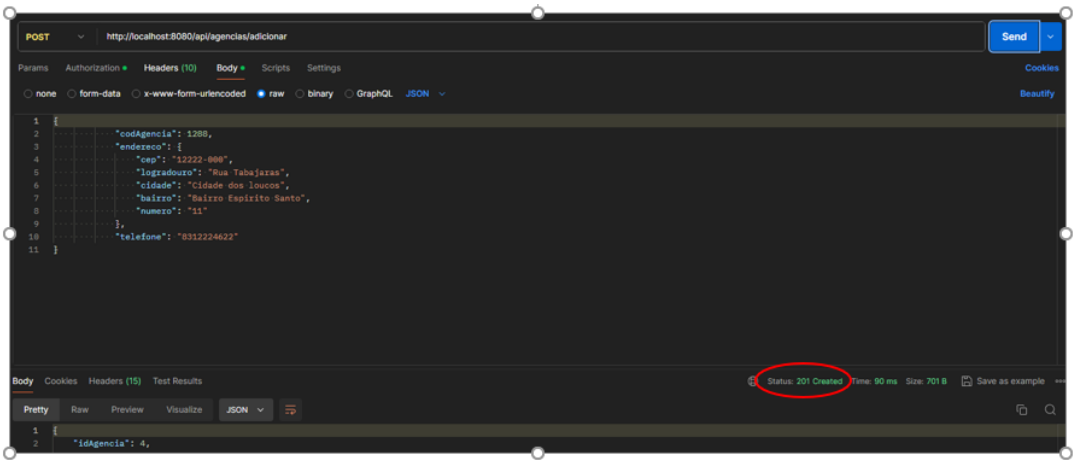
```
{
            "codAgencia": 1288,
            "endereco": {
                "cep": "12222-000",
                "logradouro": "Rua Tabajaras",
                "cidade": "Cidade dos loucos",
                "bairro": "Bairro Espirito Santo",
                "numero": "11"
            },
            "telefone": "8312224622"
}
```

Respostas esperadas:

HTTP201: Sucesso (com body usado acima).

HTTP500: Se algum dado estiver incoerente com o esperado (cep, telefone).

HTTP403: Forbidden, caso um usuário não esteja logado ou seja do tipo USER.



(IX)

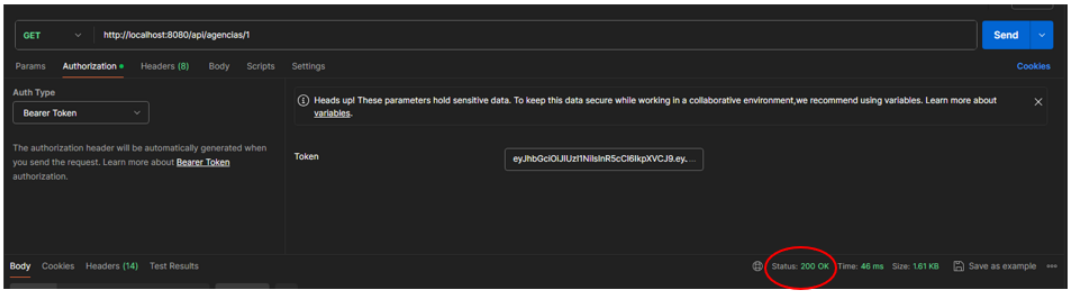
6.6 Agência por id

Para buscar uma agência, será feita uma requisição do tipo GET para a url/api/agencias/{id}

Respostas esperadas:

HTTP200: Sucesso

HTTP403: Forbidden, caso um usuário não esteja logado ou seja do tipo USER.



(X)

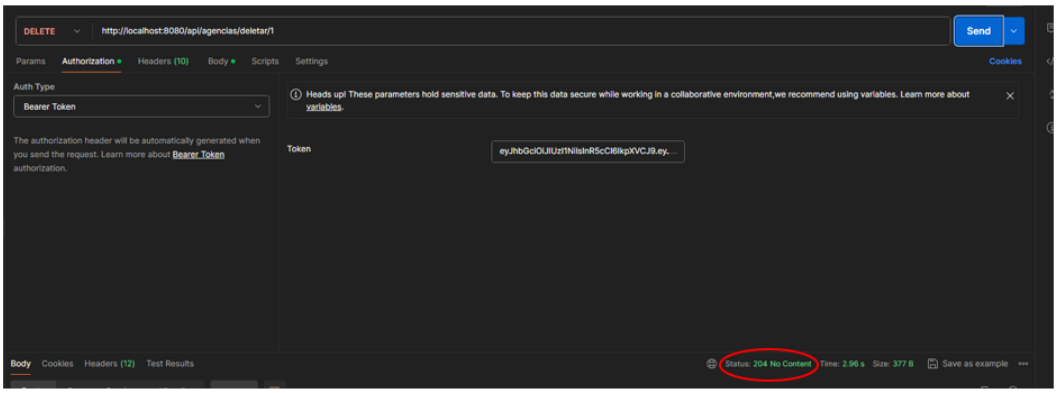
6.7 Delete agência

Para Deletar uma agência, será feita uma requisição do tipo DELETE para a url/api/agencias/deletar/{id}

Respostas esperadas:

HTTP204: Sucesso

HTTP403: Forbidden, caso um usuário não esteja logado ou seja do tipo USER.



(XI)

6.8 Usuário por id.

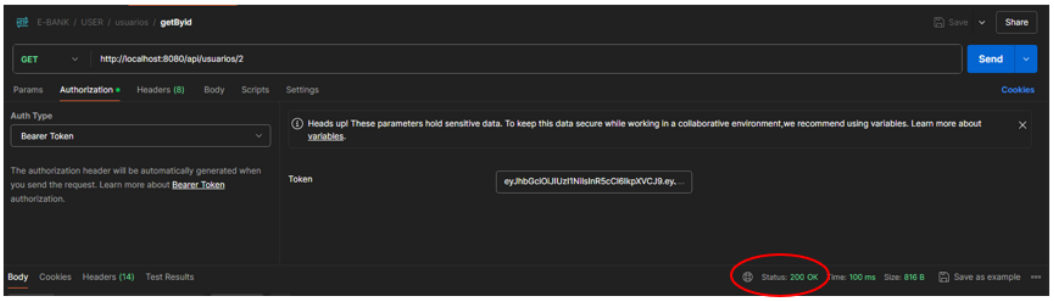
Para buscar um usuário, será feita uma requisição do tipo GET para a url/api/usuarios/{id}

Respostas esperadas:

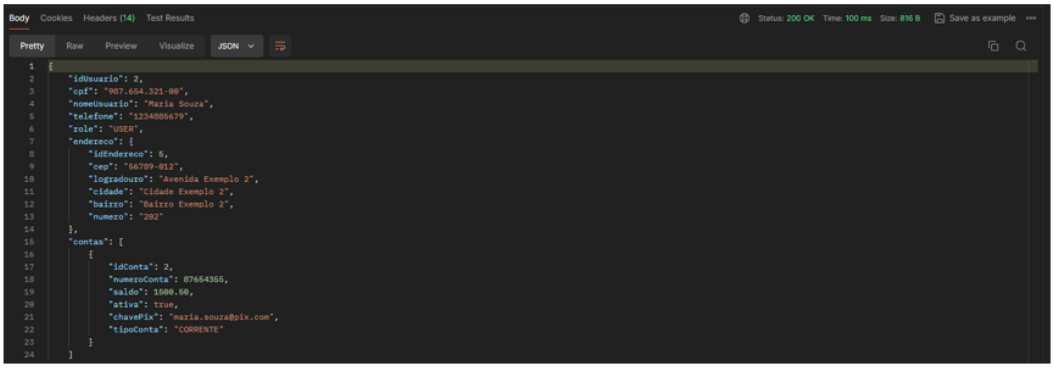
HTTP200: Sucesso

HTTP403: Forbidden, tente acessar sem login.

HTTP500: Acesso negado, caso tente acessar os dados de um usuário que não corresponda ao seu login.



(XII)



(XIII)

6.9 Atualizar usuário.

Para atualizar um usuário, será feita uma requisição do tipo PUT para a url/api/usuarios/atualizar/{id}

Request body:

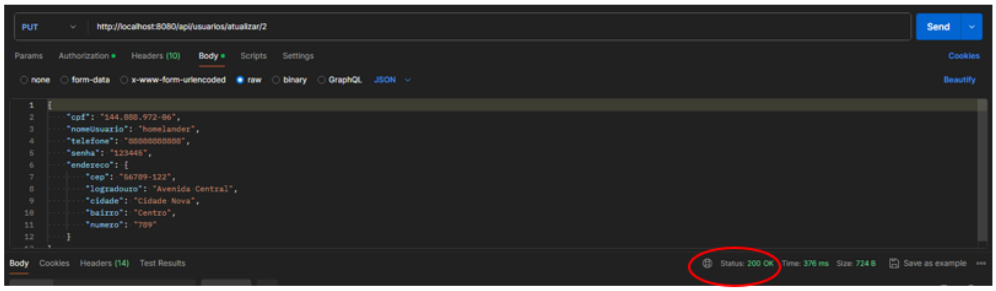
```
{
    "cpf": "144.888.972-06",
    "nomeUsuario": "homelander",
    "telefone": "88888888888",
    "senha": "123445",
    "endereco": {
        "cep": "56789-122",
        "logradouro": "Avenida Central",
        "cidade": "Cidade Nova",
        "bairro": "Centro",
        "numero": "789"
    }
}
```

Respostas esperadas:

HTTP200: Sucesso (Com o body acima)

HTTP403: Forbidden, tente acessar sem login.

HTTP500: Acesso negado, caso tente acessar os dados de um usuário que não corresponda ao seu login.



(XIV)

(XV)

6.10 Deletar usuário.

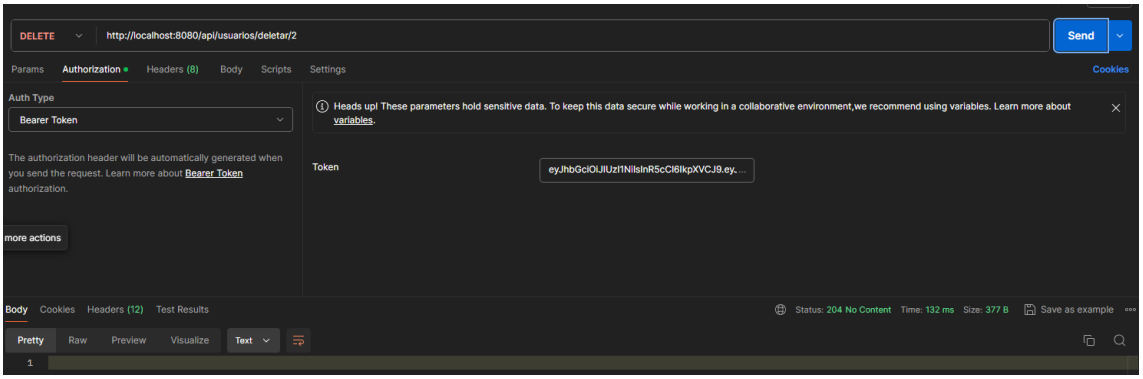
Para Deletar um usuário, será feita uma requisição do tipo DELETE para a url/api/usuarios/deletar/{id}

Respostas esperadas:

HTTP204: Sucesso

HTTP403: Forbidden, caso tente deletar um usuário sem estar logado.

HTTP500: Acesso negado, caso tente deletar um usuário diferente do logado.



6.11 Acessar Extrato.

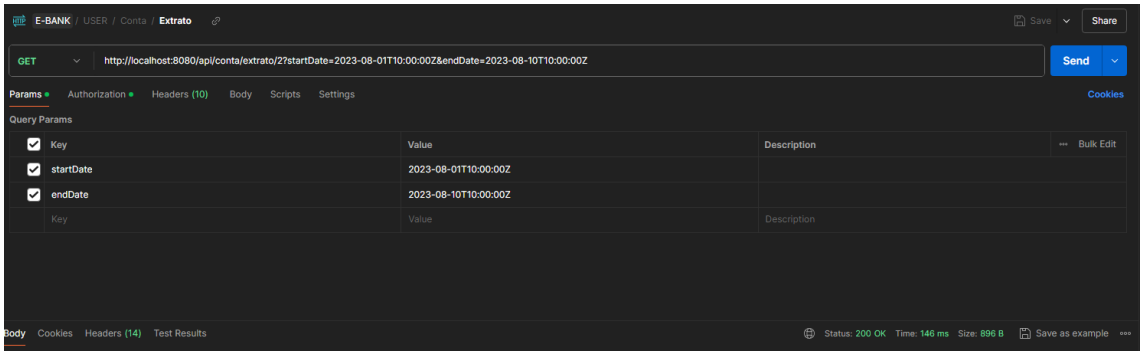
Para acessar o extrato, será feita uma requisição do tipo GET para a url/api/conta/extrato/{id}. (Utiliza Params)

Respostas esperadas:

HTTP200: Sucesso

HTTP403: Forbidden, caso um usuário tente acessar sem estar logado.

HTTP500: Acesso negado, caso um usuário tente acessar o extrato de outro usuário.



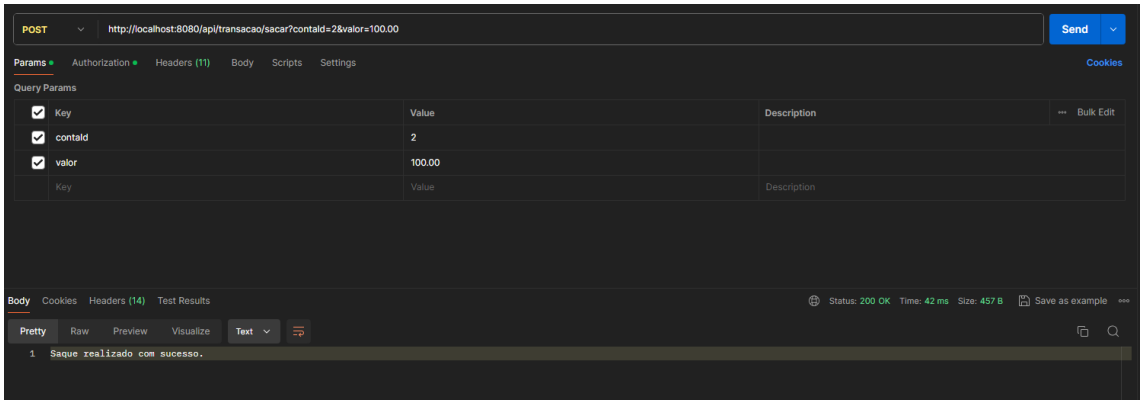
6.12 Realizar Saque.

Para realizar saque, será feita uma requisição do tipo POST para a url/api/transacao/sacar. (Utiliza Params)

Respostas esperadas:

HTTP20o: Sucesso (Retorna um comprovante)

HTTP403: Forbidden, caso um usuário tente acessar sem estar logado.



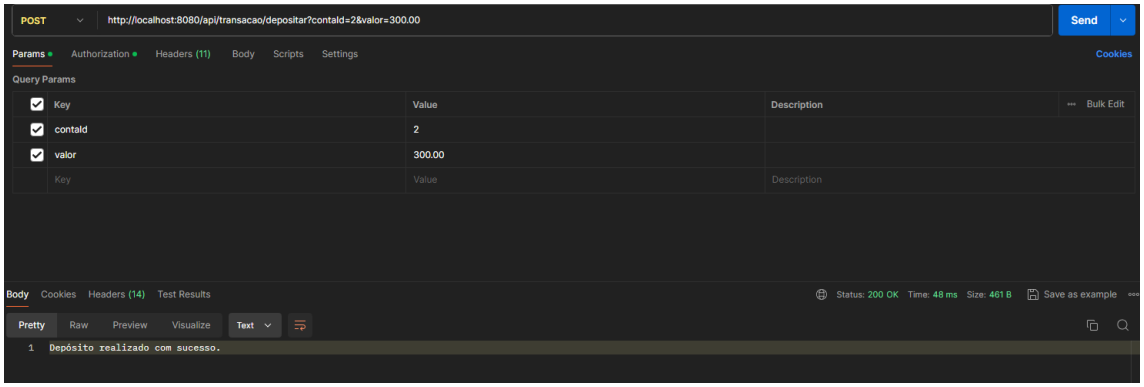
6.13 Realizar Depósito.

Para realizar depósito, será feita uma requisição do tipo POST para a url/api/transacao/depositar. (Utiliza Params)

Respostas esperadas:

HTTP200: Sucesso (Retorna um comprovante)

HTTP403: Forbidden, caso um usuário tente acessar sem estar logado.



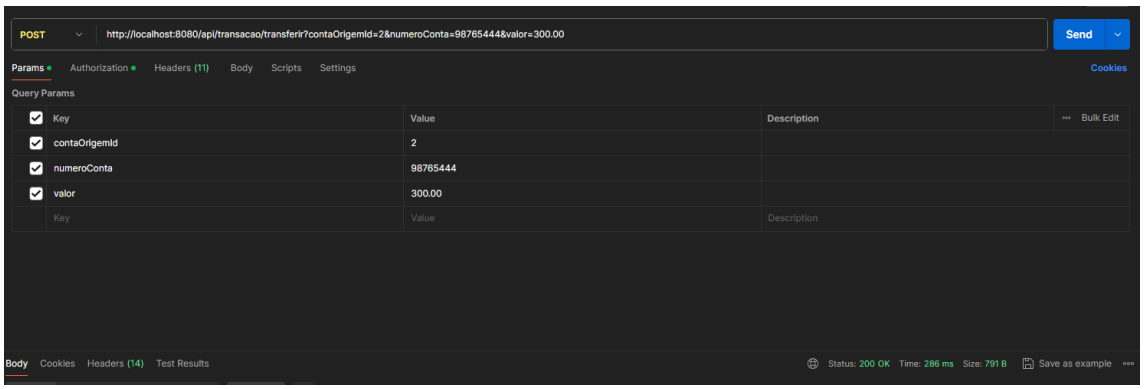
6.14 Realizar Transferência.

Para realizar transferência, será feita uma requisição do tipo POST para a url/api/transacao/transferir. (Utiliza Params)

Respostas esperadas:

HTTP200: Sucesso (Retorna um comprovante)

HTTP403: Forbidden, caso um usuário tente acessar sem estar logado.



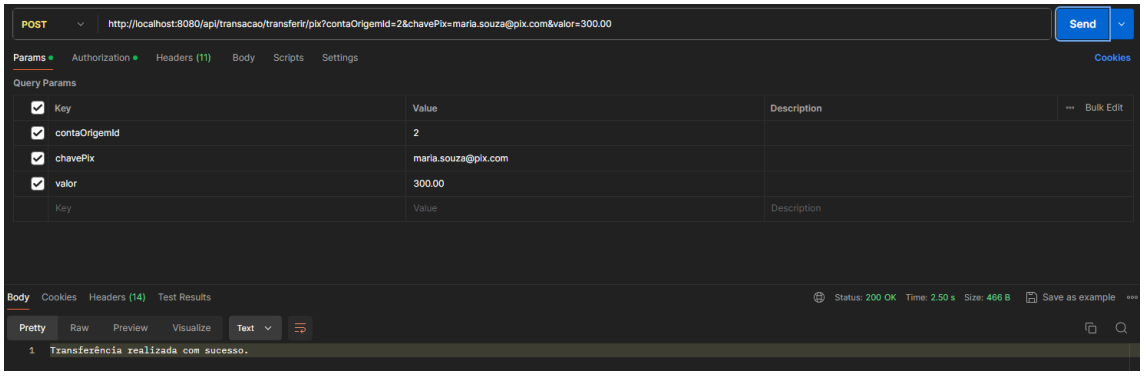
6.15 Realizar Pix.

Para realizar transferência, será feita uma requisição do tipo POST para a url/api/transacao/transferir/pix. (Utiliza Params)

Respostas esperadas:

HTTP200: Sucesso (Retorna um comprovante)

HTTP403: Forbidden, caso um usuário tente acessar sem estar logado.



7. Tecnologias Utilizadas

- Java 17
- Spring Boot
- Spring Security (JWT)
- MySQL
- H2
- Swagger
- Postman
- Jacoco

8. Jacoco

ebank

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.br.accenture.eBank.ebank.controllers		31%		n/a	14	24	39	57	14	24	0	4
com.br.accenture.eBank.ebank.services		89%		75%	11	51	11	146	7	43	0	4
com.br.accenture.eBank.ebank.dtos.agencia		69%		n/a	7	24	19	58	7	24	0	2
com.br.accenture.eBank.ebank.configs		90%		50%	2	21	7	75	0	19	0	4
com.br.accenture.eBank.ebank.exceptions		50%		n/a	2	5	2	7	2	5	0	3
com.br.accenture.eBank.ebank.dtos.conta		93%		n/a	1	25	4	54	1	25	1	4
com.br.accenture.eBank.ebank.controllers.auth		95%		100%	0	7	2	41	0	6	0	1
com.br.accenture.eBank.ebank.dtos.usuario		98%		n/a	2	56	3	136	2	56	0	4
com.br.accenture.eBank.ebank		37%		n/a	1	2	2	3	1	2	0	1
com.br.accenture.eBank.ebank.entities.enums.auth		89%		n/a	1	3	1	6	1	3	0	1
com.br.accenture.eBank.ebank.dtos.transacao		97%		75%	1	3	0	8	0	1	0	1
com.br.accenture.eBank.ebank.entities		100%		100%	0	79	0	181	0	75	0	5
com.br.accenture.eBank.ebank.configs.validation		100%		n/a	0	6	0	18	0	6	0	1
com.br.accenture.eBank.ebank.dtos.auth		100%		n/a	0	14	0	3	0	14	0	3
com.br.accenture.eBank.ebank.entities.enums		100%		n/a	0	2	0	8	0	2	0	2
com.br.accenture.eBank.ebank.dtos.endereco		100%		n/a	0	6	0	1	0	6	0	1
com.br.accenture.eBank.ebank.services.auth		100%		100%	0	3	0	5	0	2	0	1
Total	376 of 3.042	87%	7 of 36	80%	42	331	90	807	35	313	1	42