DOCUMENTO DE EVIDÊNCIAS
PROJETO TECHSTORE

Integrantes: Arthur de Sousa e Bhrenno Borges
Disciplina: Desenvolvimento de Sistemas
Data: 25/11/2025

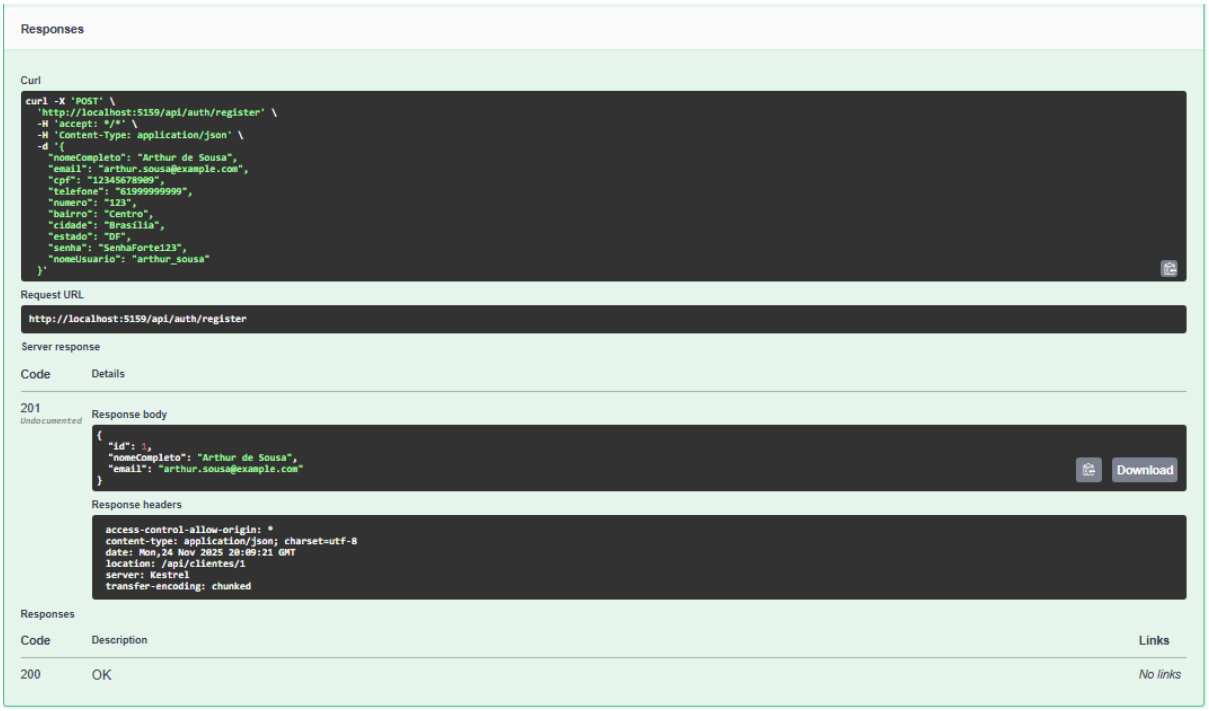
---

1. EVIDÊNCIAS DO BACKEND – SWAGGER

1.1 Autenticação (Registro e Login)

A API possui endpoints de registro e login, responsáveis pela criação de usuários e autenticação via JWT.

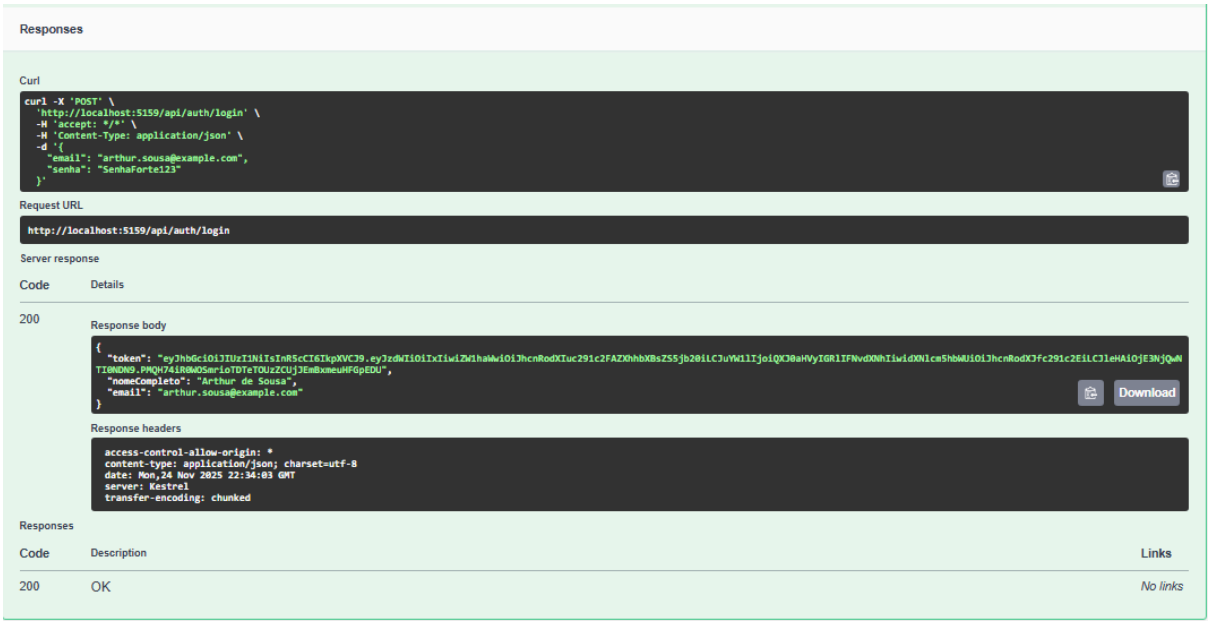
POST /api/auth/register
Essa operação registra um novo usuário no sistema enviando dados como nome, email, CPF, telefone, endereço, senha e nome de usuário.
Se os dados forem válidos, retorna **200 OK** e salva o cliente no banco de dados.



POST /api/auth/login

Realiza autenticação do usuário com email e senha.

Se correto, retorna um **token JWT**, usado em chamadas protegidas da API.

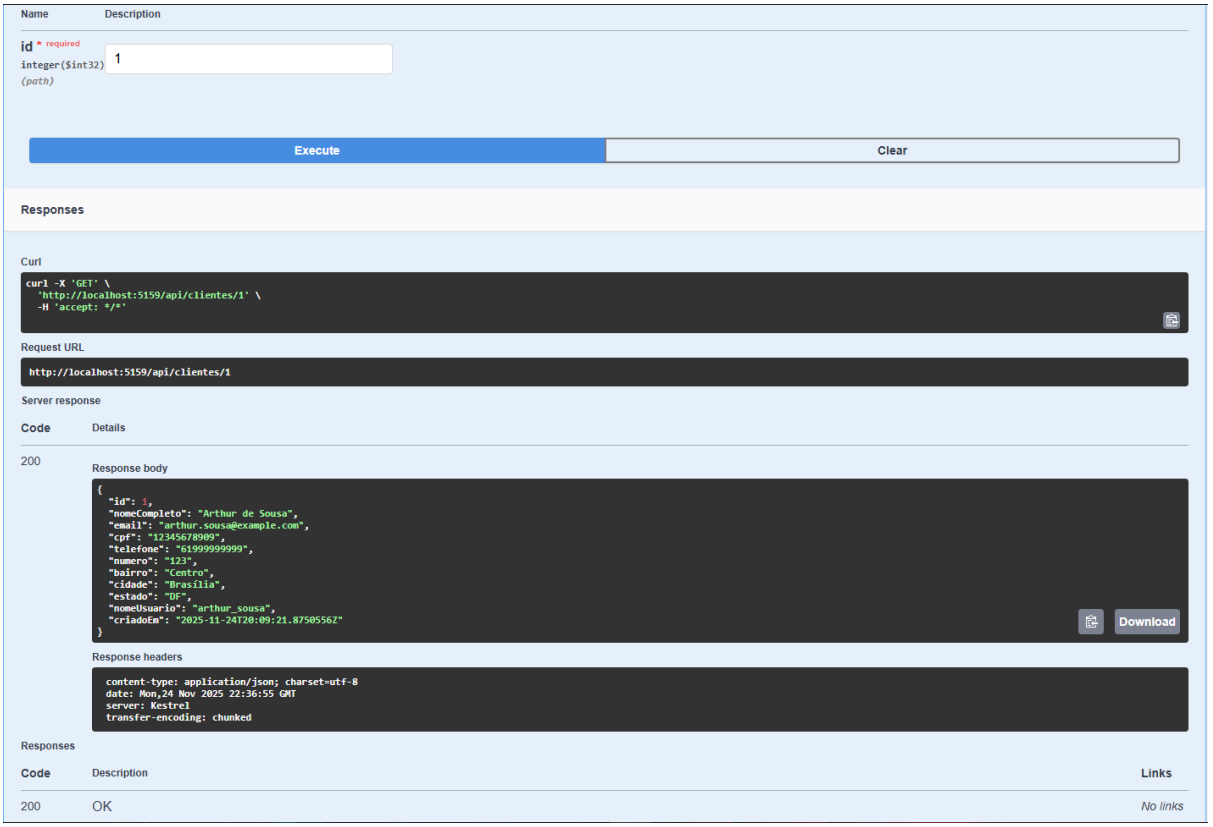


1.2 CRUD de Clientes

A API permite buscar, atualizar e deletar clientes cadastrados.

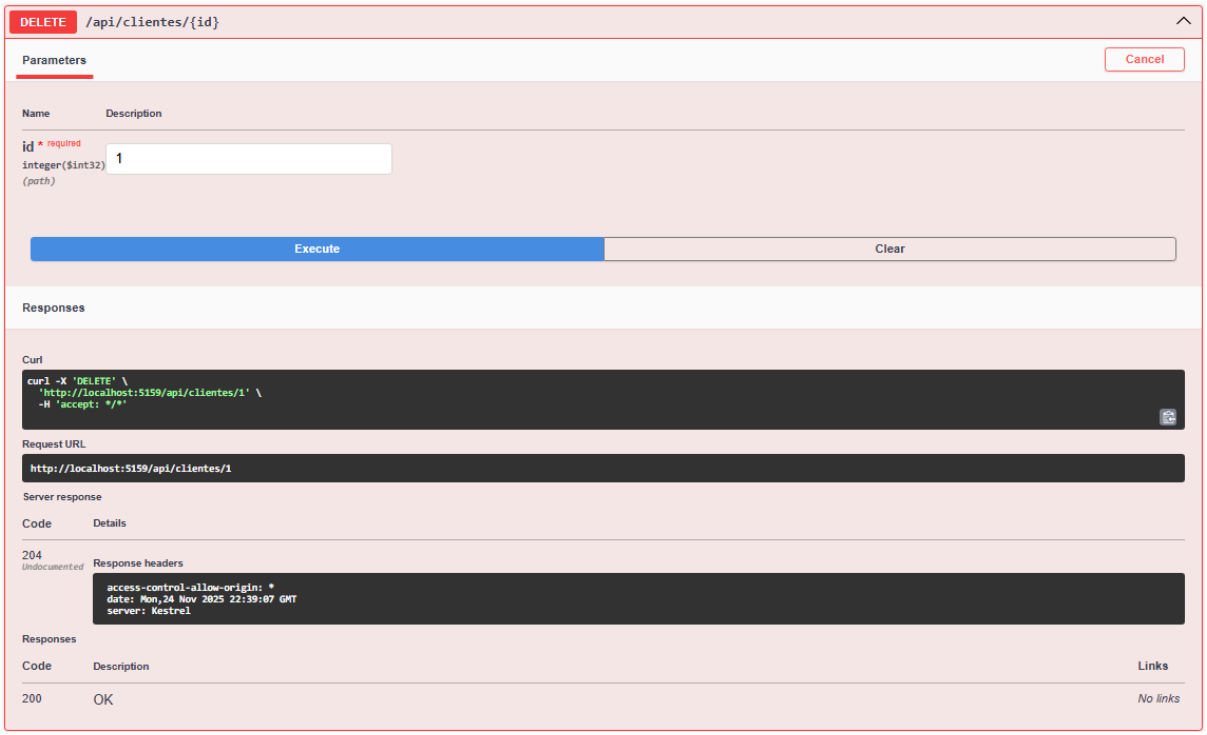
GET /api/clientes/{id}

Retorna os dados de um cliente específico.
Requer token Bearer no header Authorization.



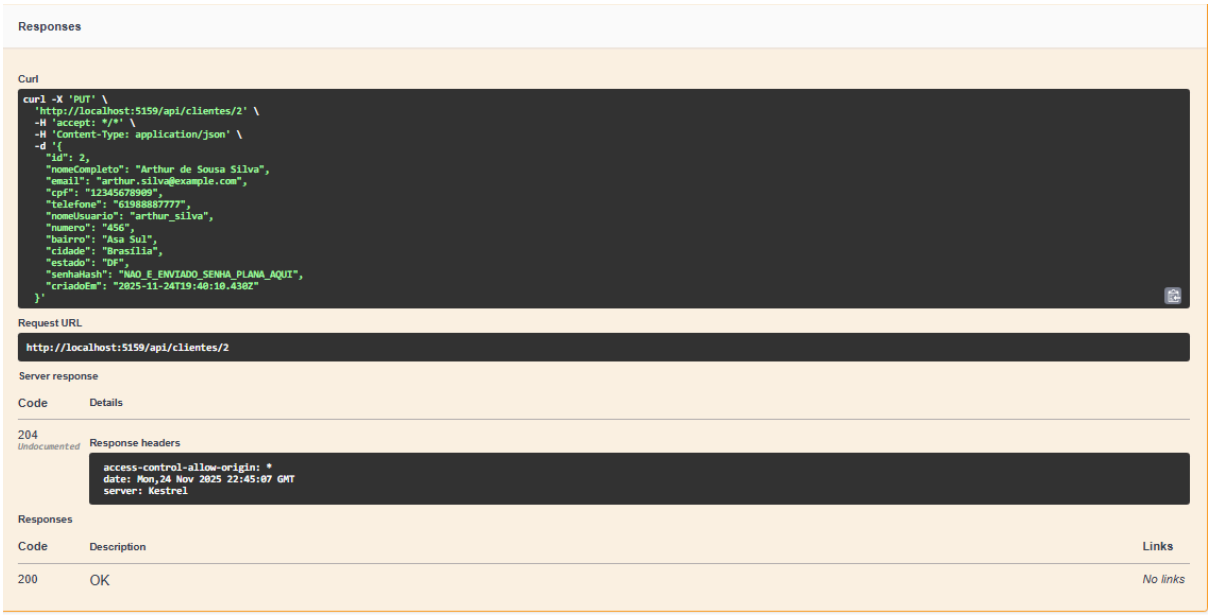
DELETE /api/clientes/{id}

Remove um cliente existente no sistema.
Requer autenticação.



PUT /api/clientes/{id}

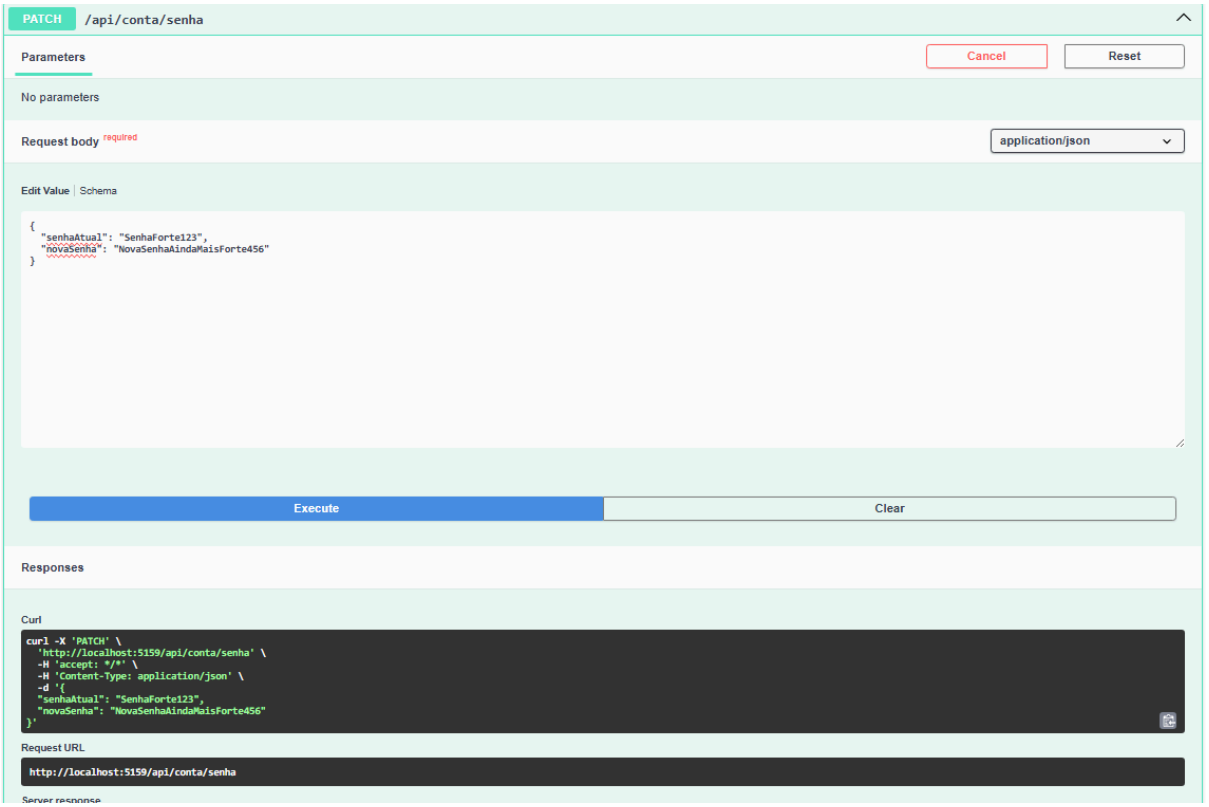
Atualiza todos os campos do cliente informado.
Envia objeto completo no corpo do JSON.



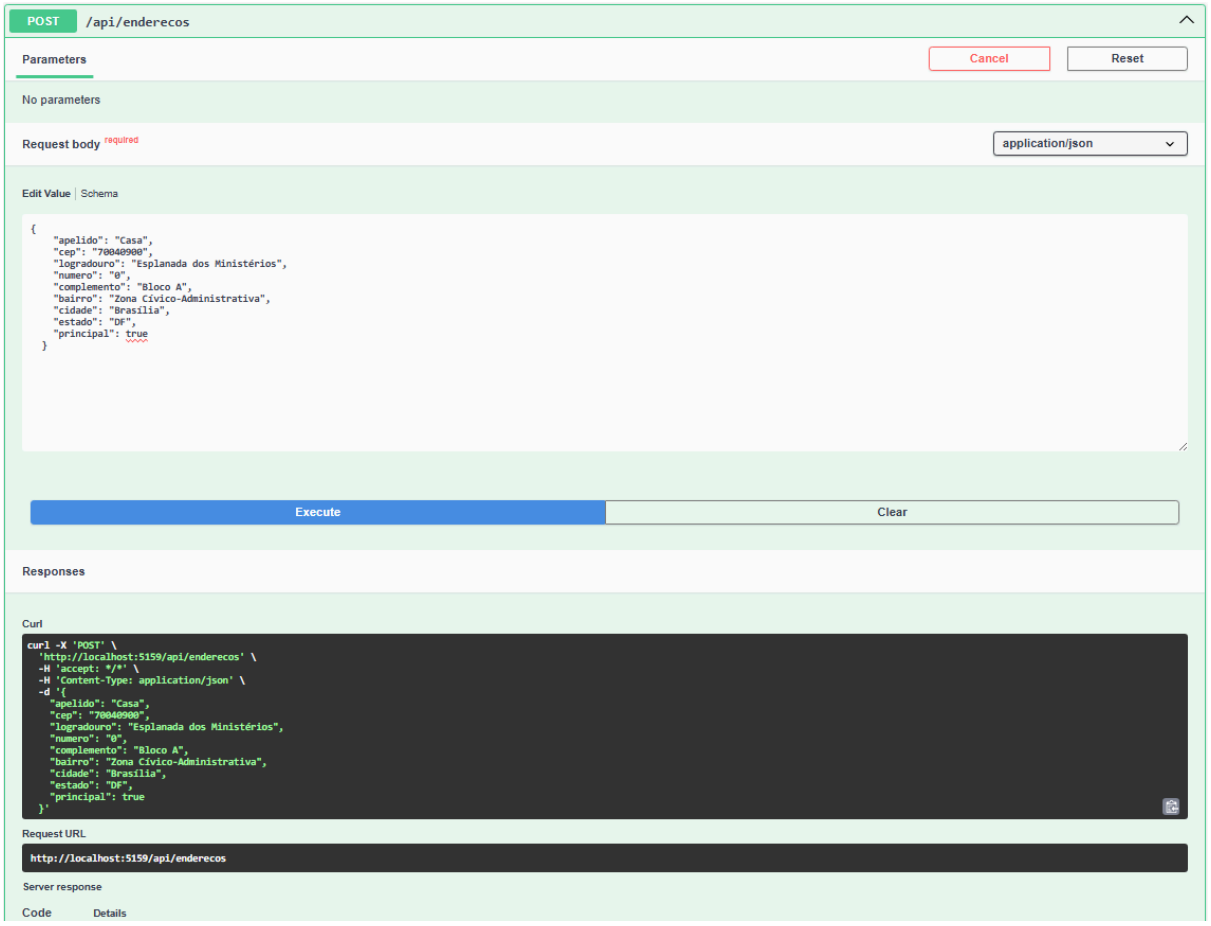
1.3 Alteração de Senha e Endereços

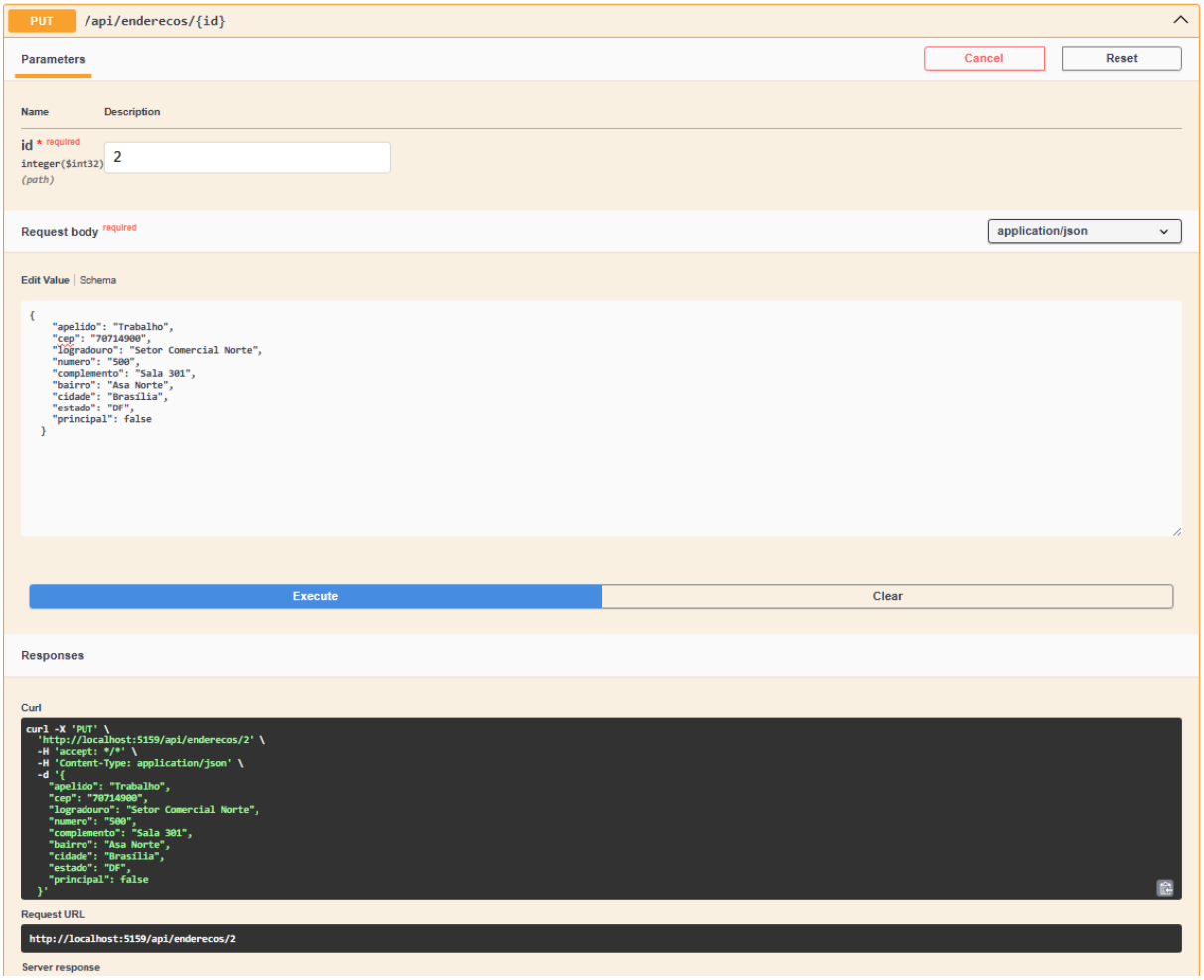
PATCH /api/conta/senha

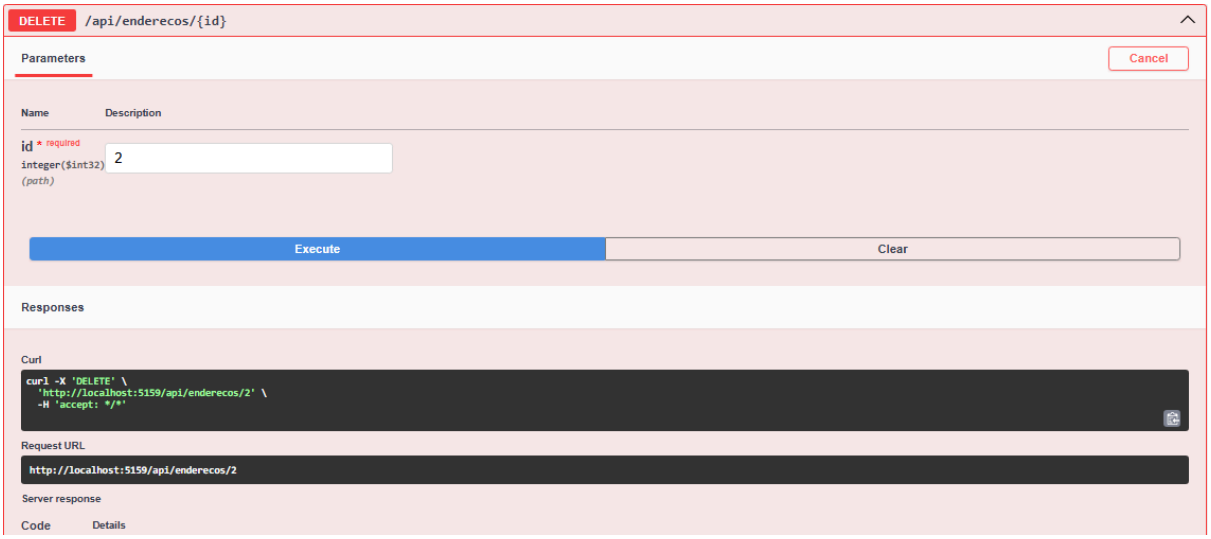
Permite trocar a senha do usuário autenticado enviando `senhaAtual` e `novaSenha`.



POST /api/enderecos, **PUT /api/enderecos/{id}**, **DELETE /api/enderecos/{id}**
CRUD completo de endereços associados a um cliente.




PUT
/api/enderecos/{id}
Parameters
Cancel
Reset
Name
Description
id * required
integer($int32)
(path)
2
Request body required
application/json
Edit Value | Schema
{
"apelido": "Trabalho",
"cep": "70714900",
"logradouro": "Setor Comercial Norte",
"numero": "500",
"complemento": "Sala 301",
"bairro": "Asa Norte",
"cidade": "Brasília",
"estado": "DF",
"principal": false
}
Execute
Clear
Responses
Curl
curl -X 'PUT' \
'http://localhost:5159/api/enderecos/2' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"apelido": "Trabalho",
"cep": "70714900",
"logradouro": "Setor Comercial Norte",
"numero": "500",
"complemento": "Sala 301",
"bairro": "Asa Norte",
"cidade": "Brasília",
"estado": "DF",
"principal": false
}'
Request URL
http://localhost:5159/api/enderecos/2
Server response

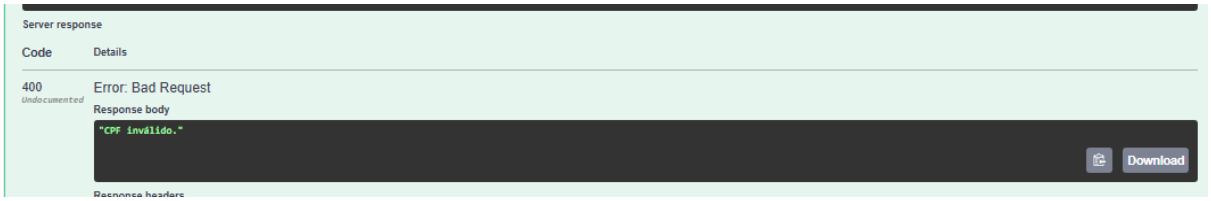

DELETE
/api/enderecos/{id}
Parameters
Cancel
Name
Description
id * required
integer($int32)
(path)
2
Execute
Clear
Responses
Curl
curl -X 'DELETE' \
'http://localhost:5159/api/enderecos/2' \
-H 'accept: */*'
Request URL
http://localhost:5159/api/enderecos/2
Server response
Code
Details

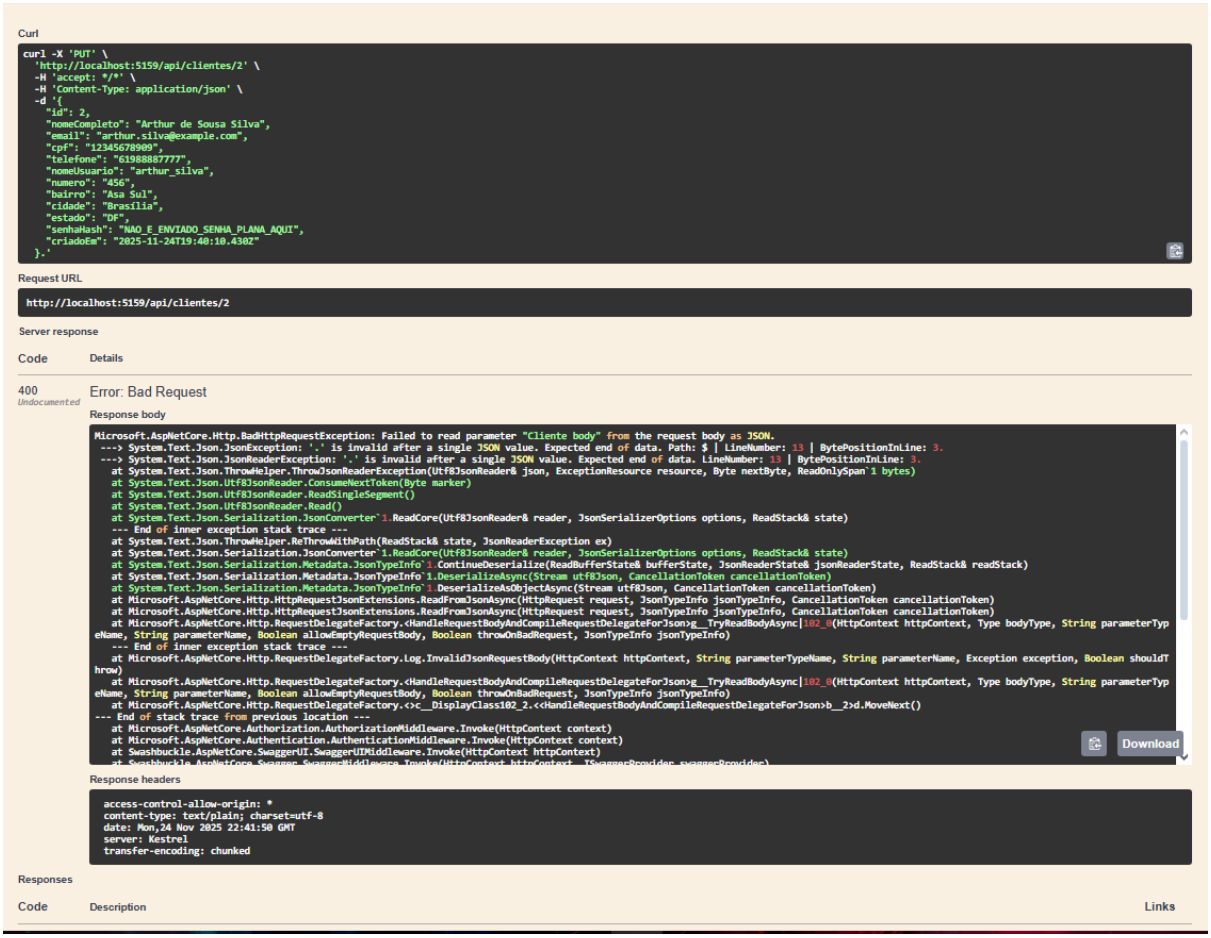
2. TRATAMENTO DE ERROS E TESTES

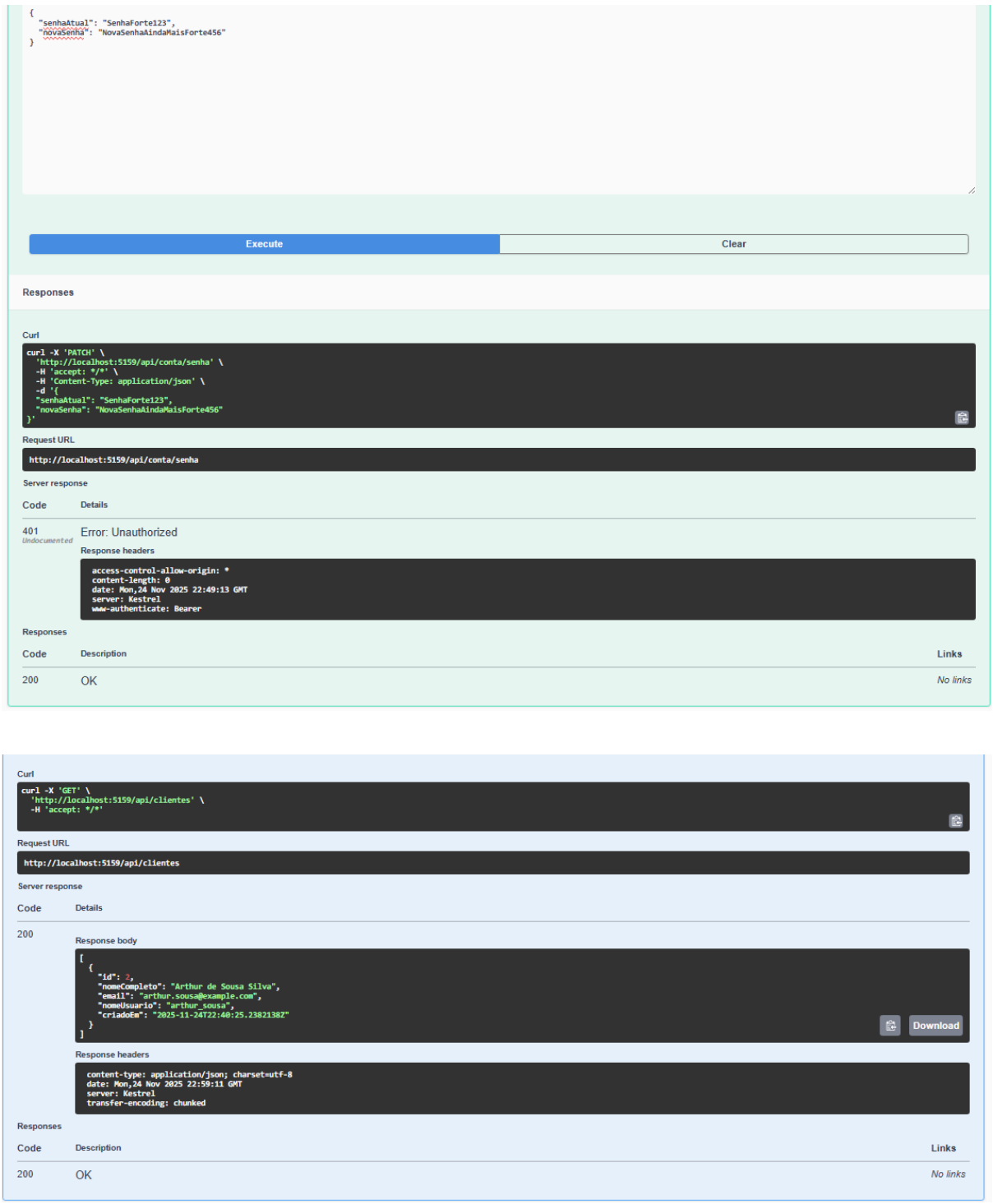
A API implementa mensagens de erro e validações, retornando diferentes códigos HTTP.

Exemplos evidenciados:

- CPF inválido → **400 Bad Request**
- JSON mal formatado → **400**
- Endpoint protegido sem token → **401 Unauthorized**
- Requisição correta com autenticação → **200 OK**





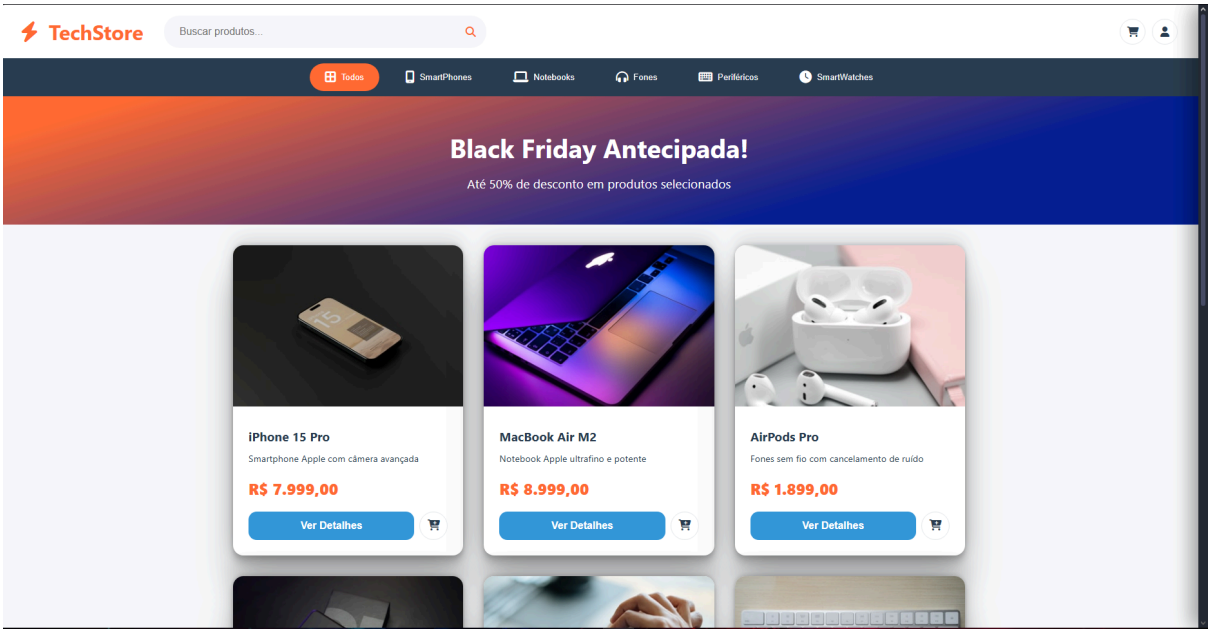

{
"senhaAtual": "SenhaForte123",
"novaSenha": "NovaSenhaAindaMaisForte456"
}
Execute
Clear
Responses
Curl
curl -X 'PATCH' \
'http://localhost:5159/api/conta/senha' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"senhaAtual": "SenhaForte123",
"novaSenha": "NovaSenhaAindaMaisForte456"
}'
Request URL
http://localhost:5159/api/conta/senha
Server response
Code
Details
401
Undocumented
Error: Unauthorized
Response headers
access-control-allow-origin: *
content-length: 0
date: Mon,24 Nov 2025 22:49:13 GMT
server: Kestrel
www-authenticate: Bearer
Responses
Code
Description
Links
200
OK
No links

Curl
curl -X 'GET' \
'http://localhost:5159/api/clientes' \
-H 'accept: */*'
Request URL
http://localhost:5159/api/clientes
Server response
Code
Details
200
Response body
[
{
"id": 2,
"nomeCompleto": "Arthur de Sousa Silva",
"email": "arthur.sousa@example.com",
"nomeUsuario": "arthur_sousa",
"criadoEm": "2025-11-24T22:40:25.2382138Z"
}
]
Download
Response headers
content-type: application/json; charset=utf-8
date: Mon,24 Nov 2025 22:59:11 GMT
server: Kestrel
transfer-encoding: chunked
Responses
Code
Description
Links
200
OK
No links

3. EVIDÊNCIAS DO FRONTEND

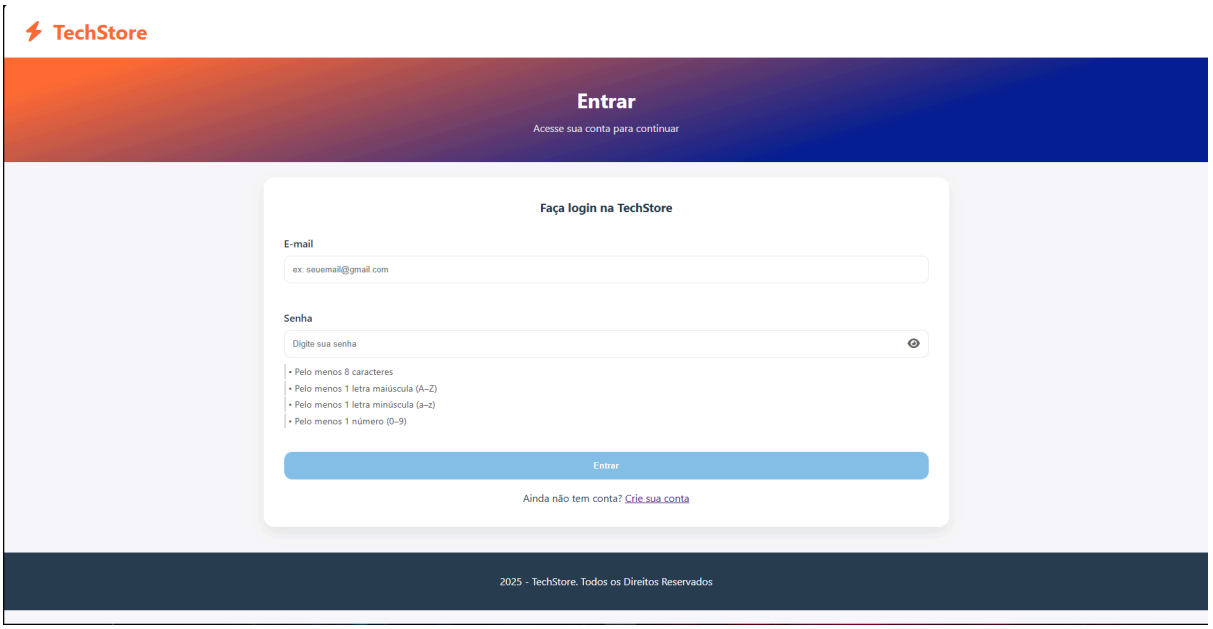
3.1 Página inicial – `index.html`

Tela inicial que exibe produtos, filtro por categorias, barra de busca e ícone do usuário/carrinho.
Os produtos são carregados dinamicamente via JavaScript.
O carrinho usa LocalStorage.



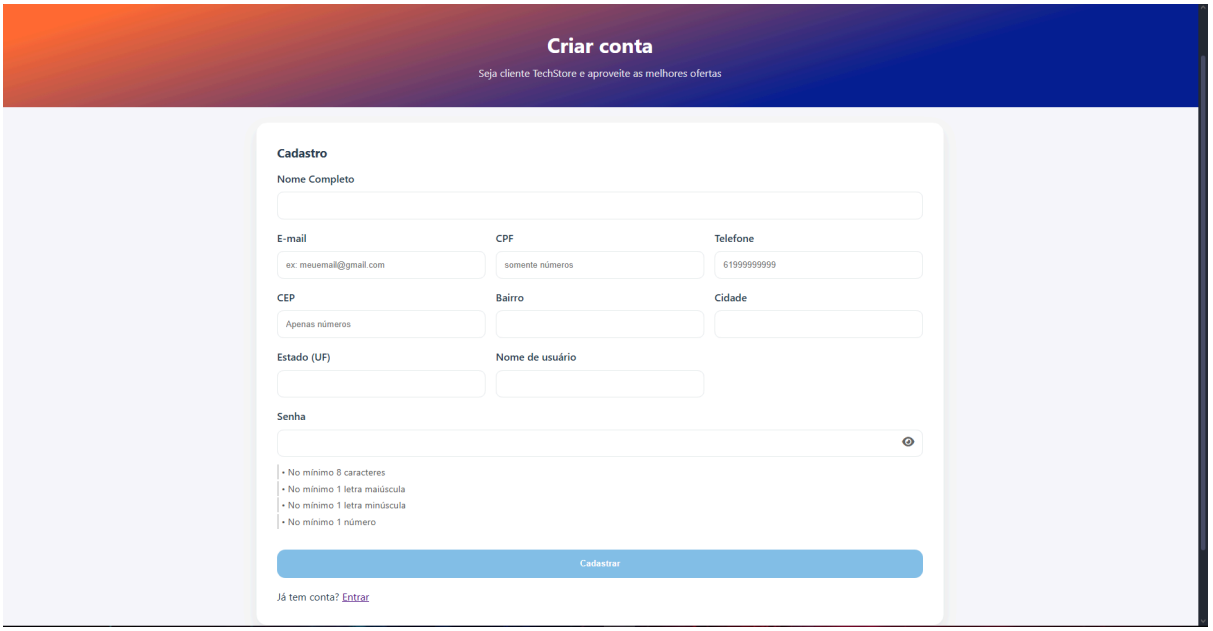
3.2 Tela de login – `login.html`

Formulário com email e senha, validação visual de força da senha e botão mostrar/ocultar. Integra com POST `/api/auth/login` e armazena o token JWT no navegador.



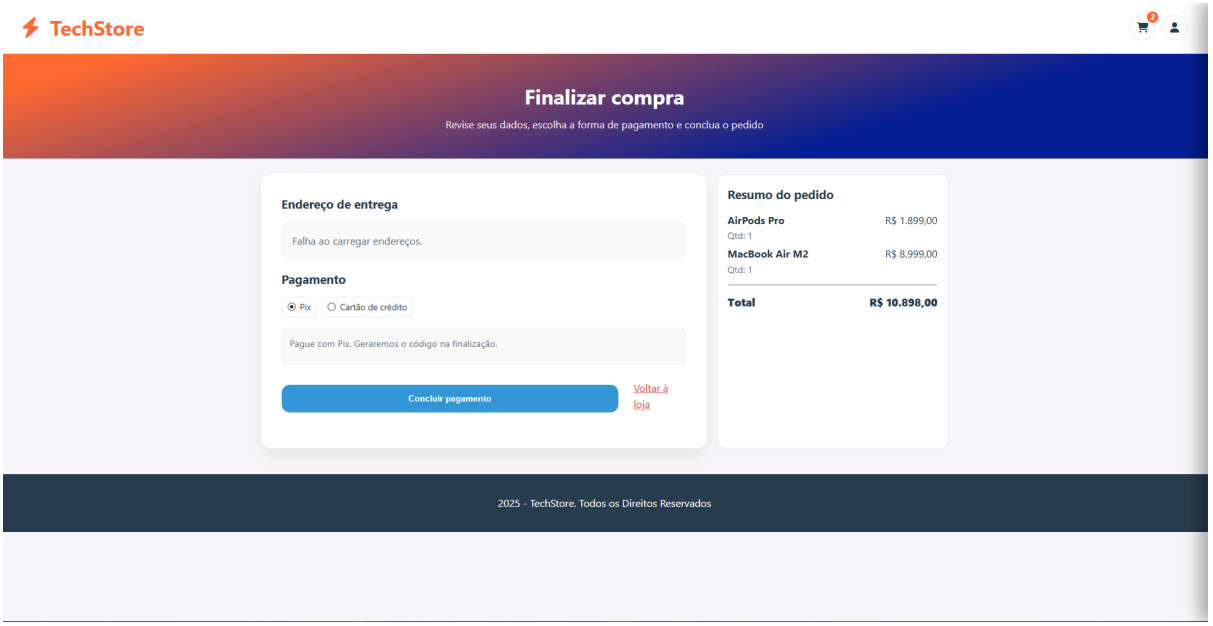
3.3 Tela de cadastro – `register.html`

Formulário com validação de CPF, telefone e senha, máscara nos campos e checklist de validação.



3.4 Tela de checkout – `checkout.html`

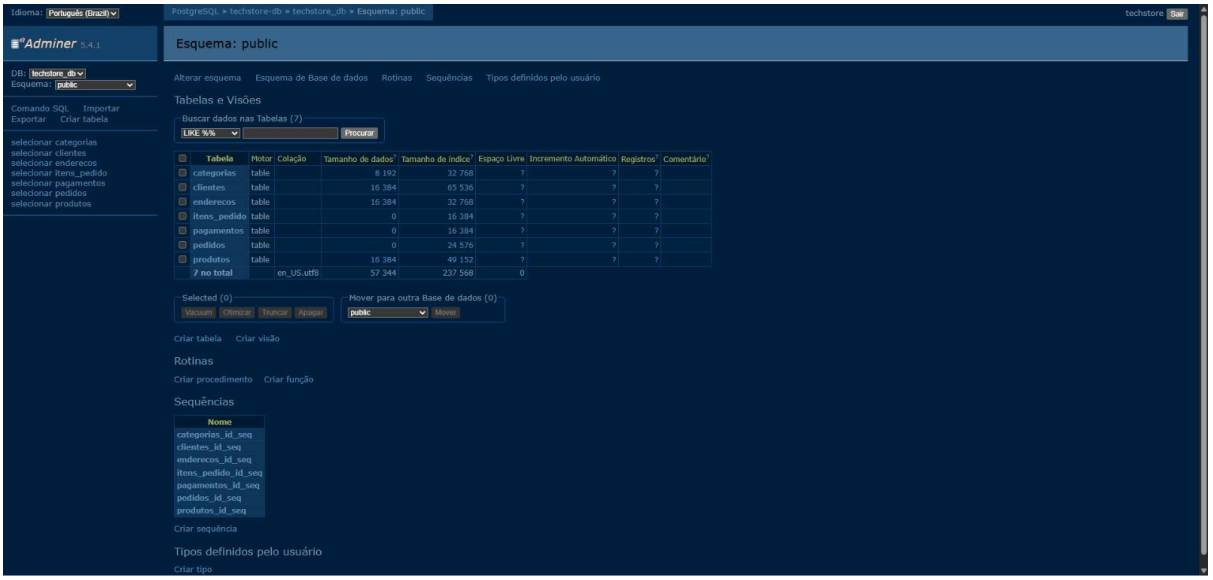
Rota protegida: redireciona se não estiver logado.
Exibe itens do carrinho, endereço carregado da API via GET `/api/enderecos`, métodos de pagamento (Pix ou cartão) e integração com POST `/api/checkout`.



---

4. BANCO DE DADOS

Banco PostgreSQL em container Docker.
Gerenciado via Adminer.
Tabelas principais: clientes, produtos, categorias, enderecos, pedidos, itens_pedido, pagamentos.



---

5. CONCLUSÃO

As evidências comprovam o bom funcionamento da API, segurança com JWT, banco Docker, integração com frontend, checkout funcional e tratamento de erros, cumprindo os requisitos do projeto.

6. REPOSITÓRIO DO GITHUB

Link do repositório:
https://github.com/BhrennoBorges/ProjetoFinal-DesenvolvimentoDeSistemas