

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.

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5159/api/auth/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "nomeCompleto": "Arthur de Sousa",
    "email": "arthur.sousa@example.com",
    "cpf": "12345678909",
    "telefone": "61999999999",
    "numero": "123",
    "bairro": "Centro",
    "cidade": "Brasília",
    "estado": "DF",
    "senha": "SenhaForte123",
    "nomeUsuario": "arthur_sousa"
  }'
```

Request URL

http://localhost:5159/api/auth/register

Server response

Code

Details

201

Undocumented

Response body

```
{
  "id": 1,
  "nomeCompleto": "Arthur de Sousa",
  "email": "arthur.sousa@example.com"
}
```

Download

Response headers

```
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Mon, 24 Nov 2025 20:00:21 GMT
location: /api/clientes/1
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

POST /api/auth/login

Realiza autenticação do usuário com email e senha.

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

Responses		
<div>Curl<pre>curl -X 'POST' \ http://localhost:5159/api/auth/login' \ -H 'accept: */*' \ -H 'Content-Type: application/json' \ -d '{ "email": "arthur.sousa@example.com", "senha": "SenhaForte123" }'</pre></div>		
<div>Request URL<pre>http://localhost:5159/api/auth/login</pre></div>		
<div>Server response</div>		
Code	Details	
200	<div><div>Response body</div><pre>{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IWRXPSJ9.ejZkdWl0IExiwlZWIhaWw1OThcnRodXuc29ic2FAZXhhbXBzSSJjb2B1LCUvZWllIjoiaXQ3bWVYIGRlIFNvdXNhTiwidXN1cmShbmU1OIJhcncodXJfc29lc2E1LClleHA1OjE3Mjc0Mw.TIRNDw8.PRWq7418MDmSeriesIOtTeTOUZZCUIJEabxneuFGpEDU", "nomeCompleto": "Arthur de Sousa", "email": "arthur.sousa@example.com" }</pre><div><div>Download</div></div></div> <div><div>Response headers</div><pre>access-control-allow-origin: * content-type: application/json; charset=utf-8 date: Mon, 24 Nov 2025 22:34:03 GMT server: Kestrel transfer-encoding: chunked</pre></div>	
<div>Responses</div>		
Code	Description	Links
200	OK	No links

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.

Name

Description

id required

integer(\$int32)

(path)

1

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5159/api/clientes/1' \
  -H 'accept: */*'

```

Request URL

```
http://localhost:5159/api/clientes/1

```

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "nomeCompleto": "Arthur de Sousa",
  "email": "arthur.sousa@example.com",
  "cpf": "12345678909",
  "telefone": "61999999999",
  "numero": "123",
  "bairro": "Centro",
  "cidade": "Brasília",
  "estado": "DF",
  "nomeUsuario": "arthur_sousa",
  "criadoem": "2025-11-24T20:09:21.8750556Z"
}

```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 24 Nov 2025 22:36:55 GMT
server: Kestrel
transfer-encoding: chunked

```

Responses

Code

Description

Links

200

OK

No links

DELETE /api/clientes/{id}

Remove um cliente existente no sistema.
Requer autenticação.

DELETE /api/clientes/{id}

Parameters

Cancel

Name	Description
id <small>* required</small>	
integer(\$int32)	
(path)	

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5159/api/clientes/1' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5159/api/clientes/1
```

Server response

Code	Details
204	

Response headers

```
access-control-allow-origin: *
date: Mon, 24 Nov 2025 22:39:07 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	OK	No links

PUT /api/clientes/{id}

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

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5159/api/clientes/2' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 2,
    "nomeCompleto": "Arthur de Sousa Silva",
    "email": "arthur.silva@example.com",
    "cpf": "12345678909",
    "telefone": "61988887777",
    "nomeUsuario": "arthur_silva",
    "numero": "456",
    "bairro": "Aaa Sui",
    "cidade": "Brasilia",
    "estado": "DF",
    "senhaHash": "1W0_E_ENVIADO_SENHA_PLANA_AQUI",
    "criadoEm": "2025-11-24T19:40:10.430Z"
  }'
```

Request URL

```
http://localhost:5159/api/clientes/2
```

Server response

Code	Details
204	

Response headers

```
access-control-allow-origin: *
date: Mon, 24 Nov 2025 22:45:07 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	OK	No links

1.3 Alteração de Senha e Endereços

PATCH /api/conta/senha

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

PATCH /api/conta/senha

Parameters

CancelReset

No parameters

Request body required

application/json

Edit ValueSchema

```
{  "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

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

POST /api/enderecos

Parameters

CancelReset

No parameters

Request body required

application/json

Edit Value | Schema

```
{  "apelido": "Casa",  "cep": "70040900",  "logradouro": "Esplanada dos Ministérios",  "numero": "0",  "complemento": "Bloco A",  "bairro": "Zona Cívico-Administrativa",  "cidade": "Brasília",  "estado": "DF",  "principal": true}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  'http://localhost:5159/api/enderecos' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "apelido": "Casa",  "cep": "70040900",  "logradouro": "Esplanada dos Ministérios",  "numero": "0",  "complemento": "Bloco A",  "bairro": "Zona Cívico-Administrativa",  "cidade": "Brasília",  "estado": "DF",  "principal": true}'
```

Request URL

```
http://localhost:5159/api/enderecos
```

Server response

Code

Details

PUT

/api/enderecos/{id}

Cancel

Reset

Parameters

Name	Description
id * required	
integer(\$int32)	2
(path)	

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": "Brasilia",  "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": "Brasilia",  "estado": "DF",  "principal": false}'
```

Request URL

http://localhost:5159/api/enderecos/2

Server response

DELETE

/api/enderecos/{id}

Cancel

Parameters

Name	Description
id * required	
integer(\$int32)	2
(path)	

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**

Server response

Code	Details
400	Error: Bad Request

Response body

```
"CPF inválido."
```

Download

Curl

```
curl -X 'PUT' \
  'http://localhost:5159/api/clientes/2' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 2,
    "nomeCompleto": "Arthur de Sousa Silva",
    "email": "arthur.silva@example.com",
    "cpf": "12345678909",
    "telefone": "61988887777",
    "nomeSobrinho": "arthur_silva",
    "numero": "456",
    "bairro": "Asa Sul",
    "cidade": "Brasília",
    "estado": "DF",
    "senhaHash": "NAO_E_ENVIADO_SENHA_PLANA_AQUI",
    "criadoEm": "2025-11-24T19:48:10.430Z"
  }.'
```

Request URL

http://localhost:5159/api/clientes/2

Server response

Code	Details
400	Error: Bad Request

Response body

```
Microsoft.AspNetCore.Http.BadRequestException: Failed to read parameter "Cliente body" from the request body as JSON.
--> System.Text.Json.JsonReaderException: ' ' is invalid after a single JSON value. Expected end of data. Path: $ | LineNumber: 13 | BytePositionInLine: 3.
--> System.Text.Json.JsonReaderException: ' ' is invalid after a single JSON value. Expected end of data. LineNumber: 13 | BytePositionInLine: 3.
at System.Text.Json.ThrowHelper.ThrowJsonReaderException(Utf8JsonReader& json, ExceptionResource resource, Byte nextByte, ReadOnlySpan`1 bytes)
at System.Text.Json.Utf8JsonReader.ConsumeNextToken(Byte marker)
at System.Text.Json.Utf8JsonReader.ReadSingleSegment()
at System.Text.Json.Utf8JsonReader.Read()
at System.Text.Json.Serialization.JsonConverter`1.ReadCore(Utf8JsonReader& reader, JsonSerializerOptions options, ReadStack& state)
--- End of inner exception stack trace ---
at System.Text.Json.ThrowHelper.ThrowWithPath(ReadStack& state, JsonReaderException ex)
at System.Text.Json.Serialization.JsonConverter`1.ReadCore(Utf8JsonReader& reader, JsonSerializerOptions options, ReadStack& state)
at System.Text.Json.Serialization.Metadata.JsonTypeInfo`1.ContinueDeserialize(ReadBufferState& bufferState, JsonReaderState& jsonReaderState, ReadStack& readStack)
at System.Text.Json.Serialization.Metadata.JsonTypeInfo`1.DeserializeAsync(Stream utf8Json, CancellationToken cancellationToken)
at System.Text.Json.Serialization.JsonTypeInfo`1.DeserializeObjectAsync(Stream utf8Json, CancellationToken cancellationToken)
at Microsoft.AspNetCore.Http.HttpRequestExtensions.ReadFromJsonAsync(HttpRequest request, JsonTypeInfo jsonTypeInfo, CancellationToken cancellationToken)
at Microsoft.AspNetCore.Http.RequestDelegateFactory.HandleRequestBodyAndCompileRequestDelegateForJsonAsync[T](HttpContext httpContext, Type bodyType, String parameterType, String parameterName, Boolean allowEmptyRequestBody, Boolean throwOnBadRequest, JsonTypeInfo jsonTypeInfo)
--- End of inner exception stack trace ---
at Microsoft.AspNetCore.Http.RequestDelegateFactory.Log.InvalidJsonRequestBody(HttpContext httpContext, String parameterType, String parameterName, Exception exception, Boolean shouldThrow)
at Microsoft.AspNetCore.Http.RequestDelegateFactory.HandleRequestBodyAndCompileRequestDelegateForJsonAsync[T](HttpContext httpContext, Type bodyType, String parameterType, String parameterName, Boolean allowEmptyRequestBody, Boolean throwOnBadRequest, JsonTypeInfo jsonTypeInfo)
at Microsoft.AspNetCore.Http.RequestDelegateFactory.<>c__DisplayClass102_2.<HandleRequestBodyAndCompileRequestDelegateForJsonAsync>d.MoveNext()
--- End of stack trace from previous location ---
at Microsoft.AspNetCore.Authorization.AuthorizationMiddleware.Invoke(HttpContext context)
at Microsoft.AspNetCore.Authentication.AuthenticationMiddleware.Invoke(HttpContext context)
at Swashbuckle.AspNetCore.SwaggerUI.SwaggerUIMiddleware.Invoke(HttpContext httpContext)
at Swashbuckle.AspNetCore.Swagger.SwaggerMiddleware.Invoke(HttpContext httpContext, ISwaggerProvider swaggerProvider)

Response headers
```

```
access-control-allow-origin: *
content-type: text/plain; charset=utf-8
date: Mon, 24 Nov 2025 22:41:50 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
------	-------------	-------

```
{
  "senhaAtual": "SenhaForte123",
  "novaSenha": "NovaSenhaIndaMaisForte456"
}
```

Execute

Clear

Responses

Curl

```
curl -X 'PATCH' \
  'http://localhost:5159/api/conta/senha' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "senhaAtual": "SenhaForte123",
    "novaSenha": "NovaSenhaIndaMaisForte456"
  }'
```

Request URL

```
http://localhost:5159/api/conta/senha
```

Server response

Code	Details
401	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 -X 'GET' \
  'http://localhost:5159/api/clientes' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5159/api/clientes
```

Server response

Code	Details
200	

Response body

```
[
  {
    "id": 1,
    "nomeCompleto": "Arthur de Sousa Silva",
    "email": "arthur.sousa@example.com",
    "nomeUsuario": "arthur_sousa",
    "criadoEm": "2025-11-24T22:48:25.238Z"
  }
]
```

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

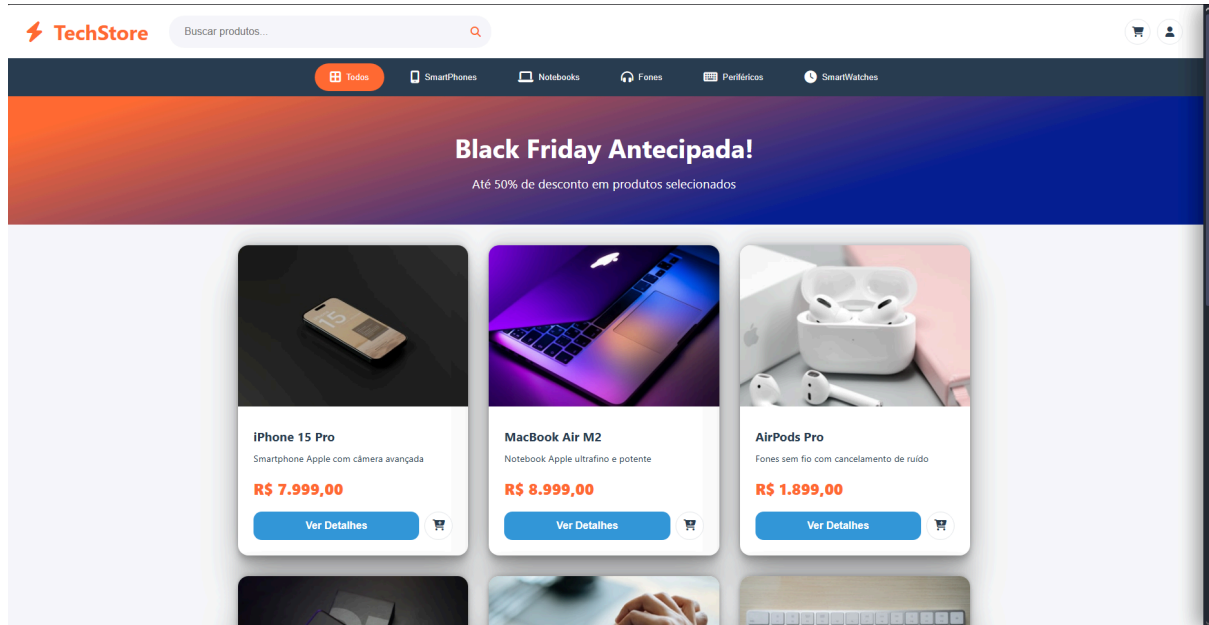
3. EVIDÊNCIAS DO FRONTEND

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.

TechStore

Entrar

Acesse sua conta para continuar

Faça login na TechStore

E-mail
ex: seuemail@gmail.com

Senha
Digite sua senha

- Pelo menos 8 caracteres
- Pelo menos 1 letra maiúscula (A-Z)
- Pelo menos 1 letra minúscula (a-z)
- Pelo menos 1 número (0-9)

Entrar

Ainda não tem conta? [Crie sua conta](#)

2025 - TechStore. Todos os Direitos Reservados

3.3 Tela de cadastro – register.html

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

Criar conta

Seja cliente TechStore e aproveite as melhores ofertas

Cadastro

Nome Completo

E-mail
ex: meuemail@gmail.com

CPF
somente números

Telefone
61999999999

CEP
Apenas números

Bairro

Cidade

Estado (UF)

Nome de usuário

Senha

- No mínimo 8 caracteres
- No mínimo 1 letra maiúscula
- No mínimo 1 letra minúscula
- No mínimo 1 número

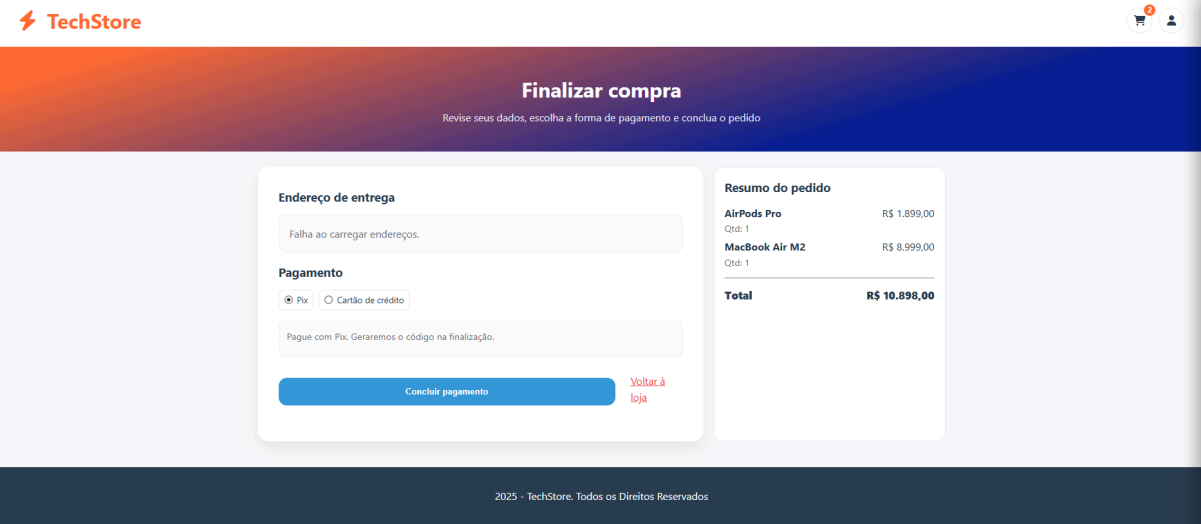
Cadastrar

Já tem conta? [Entrar](#)

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.



TechStore

Finalizar compra
Revise seus dados, escolha a forma de pagamento e conclua o pedido

Endereço de entrega

Falha ao carregar endereços.

Pagamento

☒ Pix ☐ Cartão de crédito

Pague com Pix. Geraremos o código na finalização.

Concluir pagamento [Voltar à loja](#)

Resumo do pedido

AirPods Pro R\$ 1.899,00
Qtd: 1

MacBook Air M2 R\$ 8.999,00
Qtd: 1

Total **R\$ 10.898,00**

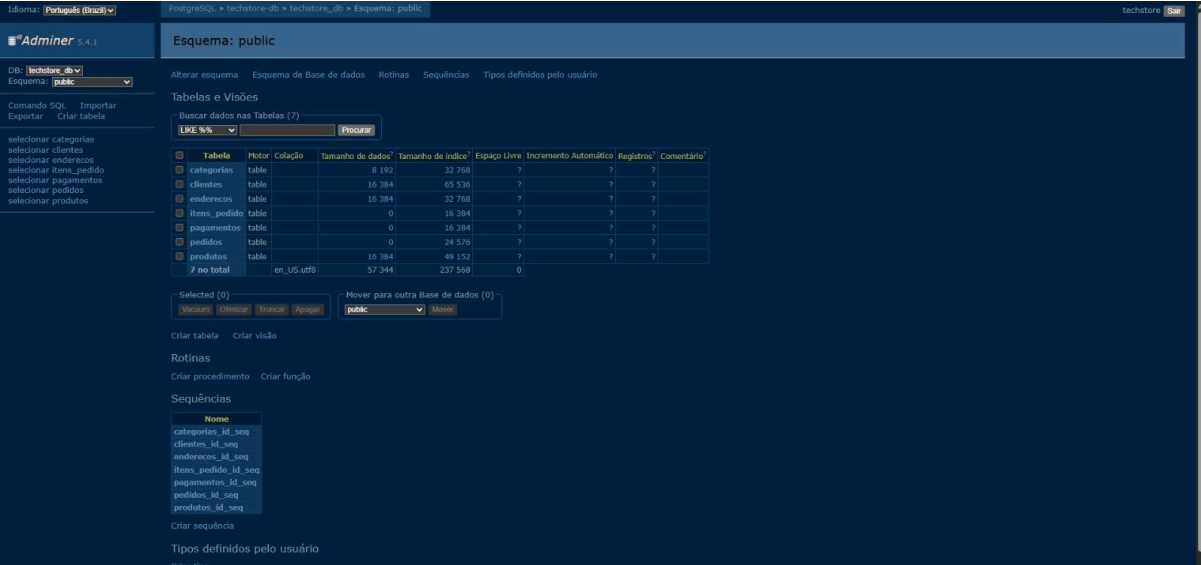
2025 - TechStore. Todos os Direitos Reservados

4. BANCO DE DADOS

Banco PostgreSQL em container Docker.

Gerenciado via Adminer.

Tabelas principais: clientes, produtos, categorias, enderecos, pedidos, itens_pedido, pagamentos.



Idioma: Português (Brasil) PostgreSQL - techstore_db - techstore_db - Esquema: public techstore

Adminer 5.4.1

DB: techstore_db Esquema: public

Comando SQL Importar Exportar Criar tabela

selecionar categorias
selecionar clientes
selecionar enderecos
selecionar itens_pedido
selecionar pagamentos
selecionar pedidos
selecionar produtos

Esquema: public

Alterar esquema Esquema de Base de dados Rotinas Sequências Tipos definidos pelo usuário

Tabelas e Visões

Buscar dados nas Tabelas (7)

LIKE % % Procurar

Tabela	Motor	Colação	Tamanho de dados	Tamanho de índice	Espaço Livre	Incremento Automático	Registros	Comentário
categorias	table		8.192	32.768	?	?	?	
clientes	table		16.384	65.536	?	?	?	
enderecos	table		16.384	32.768	?	?	?	
itens_pedido	table		0	16.384	?	?	?	
pagamentos	table		0	16.384	?	?	?	
pedidos	table		0	24.576	?	?	?	
produtos	table		16.384	49.152	?	?	?	
7 no total	en_US.utf8		57.344	237.568	0			

Select (0) Mover para outra Base de dados (0)

Nome: techstore Nome: public

Criar tabela Criar visão

Rotinas

Criar procedimento Criar função

Sequências

Nome
categorias_id_seq
clientes_id_seq
enderecos_id_seq
itens_pedido_id_seq
pagamentos_id_seq
pedidos_id_seq
produtos_id_seq

Criar sequência

Tipos definidos pelo usuário

Criar tipo

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>