

# **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": "12345678900",
    "telefone": "(61)999999999",
    "numero": "123",
    "bairro": "Centro",
    "cidade": "Brasília",
    "senha": "SenhaForte123",
    "nomelusuario": "arthur_sousa"
}'
```

Request URL

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

Server response

Code Details

201 Response body

Undocumented

```
{
    "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

#### Curl

```
curl -X 'POST' \
  'http://localhost:5159/api/auth/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "email": "arthur.sousa@example.com",
    "senha": "SenhaForte123"
}'
```



#### Request URL

```
http://localhost:5159/api/auth/login
```

#### Server response

Code	Details
200	Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWJtIiwiIjoiJhcndodXUcTuc291c2FAZXhhbXBsZS5jb28lCjUwM1IjolQXO0aHvYIGRIIFNvOXMhIiwiidXNlc5hbWUiOiJhcndodXUfc291c2EiLC1leHAiOjE3NjQwNTI0NDM9.PWQH741RM0sni0iDTeTOUzzCUi3EmBxmeuHF6pEDU",
  "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, 06 Mar 2025 22:34:03 GMT
server: Kestrel
transfer-encoding: chunked
```

### Responses

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) 1 (path)

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	<p>Response body</p> <pre>{   "id": 1,   "nomeCompleto": "Arthur de Sousa",   "email": "arthur.sousa@example.com",   "cpf": "12345678900",   "telefone": "(61) 999999999",   "numero": "123",   "bairro": "Centro",   "cidade": "Brasília",   "estado": "DF",   "nomesUsuario": "arthur_sousa",   "criadoEm": "2025-11-24T00:09:21.875056Z" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon, 24 Nov 2025 22:36:55 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200	OK	No links

## **DELETE /api/clientes/{id}**

Remove um cliente existente no sistema.  
Requer autenticação.

The screenshot shows a DELETE request for the endpoint /api/clientes/{id}. The 'id' parameter is set to 1. The interface includes sections for 'Responses', 'Curl', 'Request URL', 'Server response', and 'Code'. The 'Server response' section displays a successful 204 response with headers: access-control-allow-origin: \*, date: Mon, 24 Nov 2025 22:39:07 GMT, and server: Kestrel. The 'Code' section shows a 200 OK status with the message 'OK'.

## **PUT /api/clientes/{id}**

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

The screenshot shows a PUT request for the endpoint /api/clientes/{id} with id 2. The request body contains a JSON object with fields: id: 2, nomeCompleto: "Arthur de Sousa Silva", email: "arthur.silva@example.com", cpf: "12345678900", telefone: "61988887777", nomeUsuario: "arthur\_silva", numero: "456", bairro: "Asa Sul", cidade: "Brasília", estado: "DF", senhaHash: "NAO\_E\_ENVIADO\_SENHA\_PLANA\_AQUI", and criadoEm: "2025-11-24T19:48:10.438Z". The interface includes sections for 'Responses', 'Curl', 'Request URL', 'Server response', and 'Code'. The 'Server response' section displays a successful 204 response with headers: access-control-allow-origin: \*, date: Mon, 24 Nov 2025 22:45:07 GMT, and server: Kestrel. The 'Code' section shows a 200 OK status with the message 'OK'.

## 1.3 Alteração de Senha e Endereços

### PATCH /api/conta/senha

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

The screenshot shows a REST API testing interface with the following details:

- Method:** PATCH
- Endpoint:** /api/conta/senha
- Parameters:** No parameters
- Request body (required):** application/json
- Request Body Content:**

```
{  
    "senhaAtual": "SenhaForte123",  
    "novaSenha": "NovaSenhaAindaMaisForte456"  
}
```
- Buttons:** Execute (blue), 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:** (empty)

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

The screenshot shows a REST API testing interface. At the top, there's a green header bar with the method **POST** and the endpoint **/api/enderecos**. Below this, there are sections for **Parameters** (empty), **Request body required** (containing a JSON schema), and **Responses**.

**Request body required:**

```
{
  "apelido": "Casa",
  "cep": "70040900",
  "logradouro": "Esplanada dos Ministérios",
  "numero": "65",
  "complemento": "Bloco A",
  "bairro": "Zona Cívico-Administrativa",
  "cidade": "Brasília",
  "estado": "DF",
  "principal": true
}
```

**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": "65",
  "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}**

**Parameters**

Name	Description
<b>id</b> <small>* required</small>	integer(\$int32) <input type="text" value="2"/>

**Request body** required

application/json

```
{
  "apelido": "Trabalho",
  "cep": "70714000",
  "logradouro": "Setor Comercial Norte",
  "numero": "500",
  "complemento": "Sala 301",
  "bairro": "Asa Norte",
  "cidade": "Brasília",
  "estado": "DF",
  "principal": false
}
```

**Responses**

**Curl**

```
curl -X 'PUT' \
'http://localhost:5159/api/enderecos/2' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d $'{
  "apelido": "Trabalho",
  "cep": "70714000",
  "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**

Name	Description
<b>id</b> <small>* required</small>	integer(\$int32) <input type="text" value="2"/>

**Responses**

**Curl**

```
curl -X 'DELETE' \
'http://localhost:5159/api/enderecos/2' \
-H 'accept: */*'
```

**Request URL**

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

**Server response**

## 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
<small>Undocumented</small>	
	Response body
	<pre>"CPF inválido."</pre>
	<small>Response headers</small>
	 <a href="#">Download</a>

```
curl -X 'PUT' \
  'http://localhost:5159/api/clients/2' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 2,
    "nomeCompleto": "Arthur de Sousa Silva",
    "email": "arthur.silva@example.com",
    "cpf": "12345678900",
    "telefone": "+55988887777",
    "nomeUsuario": "arthur_silva",
    "numero": "456",
    "bairro": "Asa Sul",
    "cidade": "Brasília",
    "estado": "DF",
    "senhaHash": "MAQ_E_ENVIADO_SEMNA_PLANA_AQUI",
    "criadoEm": "2025-11-24T19:40:18.430Z"
}.'
```

Request URL  
<http://localhost:5159/api/clients/2>

Server response

Code	Details
400 Undocumented	Error: Bad Request Response body

```
Microsoft.AspNetCore.Http.BadHttpRequestException: Failed to read parameter "Clientes body" from the request body as JSON.  
--> System.Text.Json.JsonException: '' 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.Utf8JsonReader.ReadObjectHeader(Utf8JsonReader& json, ExceptionResource resource, Byte nextByte, ReadOnlySpan<byte> bytes)  
at System.Text.Json.Utf8JsonReader.ConsumeNextToken(Byte marker)  
at System.Text.Json.Utf8JsonHeader.ReadStringSegment()  
at System.Text.Json.Utf8JsonHeader.Read()  
at System.Text.Json.Serialization.JsonConverter`1.EndReadCore(Utf8JsonReader& reader, JsonSerializerOptions options, ReadStack& state)  
-- End of inner exception stack trace --  
at System.Text.Json.ThroughParser.ThrowWithPath(ReadStack& state, JsonHeaderException ex)  
at System.Text.Json.ThroughParser.ParseConverter`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.Metadata.JsonTypeInfo`1.DeserializeAsObjectAsync(Stream utf8Json, CancellationToken cancellationToken)  
at Microsoft.AspNetCore.Http.HttpRequestExtensions.ReadFromJsonAsync<T>(HttpRequest request, JsonTypeInfo jsonTypeInfo, CancellationToken cancellationToken)  
at Microsoft.AspNetCore.Http.HttpRequestExtensions.ReadFromJsonAsync<T>(HttpRequest request, Type bodyType, JsonTypeInfo jsonTypeInfo, CancellationToken cancellationToken)  
at Microsoft.AspNetCore.Http.HttpRequestExtensions.ReadFromJsonAsync<T>(HttpRequest request, Type bodyType, String parameterName, Boolean allowEmptyRequestBody, Boolean throwOnBadRequest, JsonTypeInfo jsonTypeInfo)  
-- End of inner exception stack trace --  
at Microsoft.AspNetCore.Http.RequestDelegateFactory.Log.InvalidJsonRequestBody(HttpContext httpContext, String parameterTypeName, String parameterName, Exception exception, Boolean shouldT  
hrow)  
at Microsoft.AspNetCore.Http.RequestDelegateFactory.<HandleRequestBodyAndCompileRequestDelegatesForJsonb>`1._TryReadBodyAsync<T>(HttpContext httpContext, Type bodyType, String parameterType  
eName, String parameterName, Boolean allowEmptyRequestBody, Boolean throwOnBadRequest, JsonTypeInfo jsonTypeInfo)  
at Microsoft.AspNetCore.Http.RequestDelegateFactory.<>c__DisplayClass10_2.<<HandleRequestBodyAndCompileRequestDelegatesForJsonb>`2.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 Microsoft.AspNetCore.SwaggerUI.SwaggerUIMiddleware.Invoke(HttpContext context)  
at Swashbuckle.AspNetCore.SwaggerUIHandlerMiddleware.Invoke(HttpContext context) -> (async+await+foreach+loop+deco  
Download
```

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": "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	Links
401 <i>Undocumented</i>	Error: Unauthorized	No links

**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	Links
200	Response body	No links

**Response body**

```
[
  {
    "id": 2,
    "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**

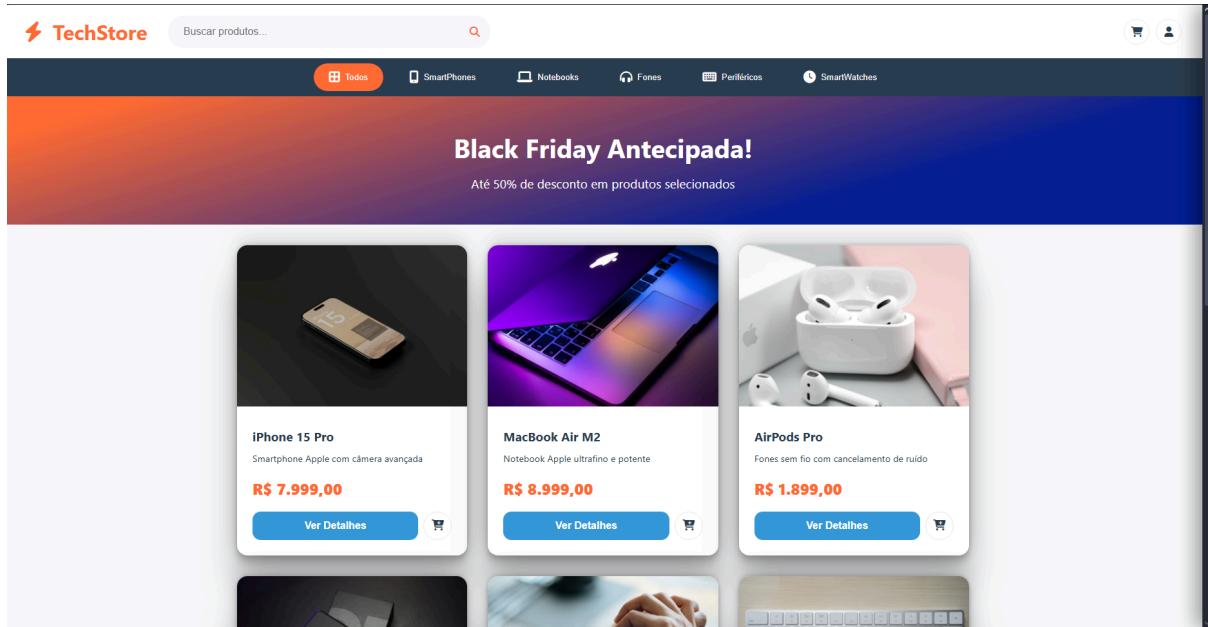
Code	Description	Links
200	OK	No links

### 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  
61999999999

Telefone  
61999999999

CEP  
Apenas números

Bairro

Cidade

Estado (UF)

Nome de usuário

Senha  
Digite sua senha

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

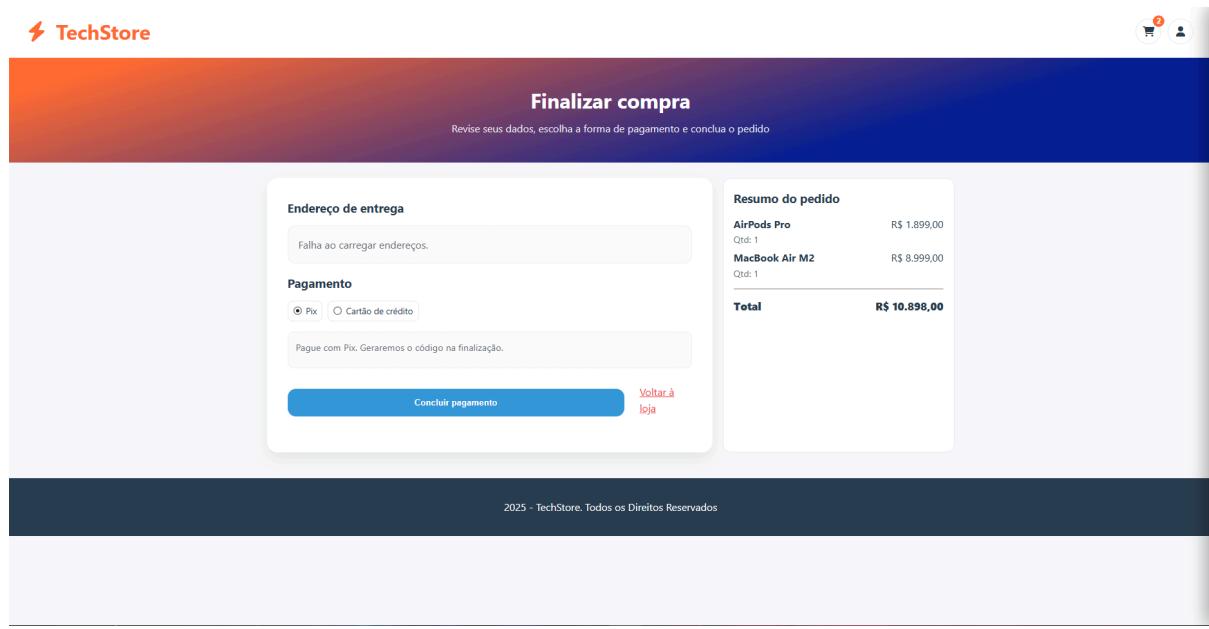
Cadastrar

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.



## 4. BANCO DE DADOS

Banco PostgreSQL em container Docker.

Gerenciado via Adminer.

Tabelas principais: clientes, produtos, categorias, enderecos, pedidos, itens\_pedido, pagamentos.

The screenshot shows the Adminer interface for the 'public' schema of the 'techstore\_db' database. The main area displays a table of tables with columns: Tabela (Table), Motor (Engine), Colação (Collation), Tamanho de dados (Data size), Tamanho de índice (Index size), Espaço Livre (Free space), Incremento Automático (Automatic increment), Registros (Registers), and Comentário (Comment). The 'products' table is highlighted, showing 57,344 rows and 16 columns. Below the table, there are buttons for 'Vaciar' (Empty), 'Duplicar' (Duplicate), 'Truncar' (Truncate), and 'Apagar' (Delete). The interface also includes sections for 'Criar tabela' (Create table), 'Criar visão' (Create view), 'Rotinas' (Procedures), 'Criar procedimento' (Create procedure), 'Criar função' (Create function), 'Sequências' (Sequences), 'Nome' (Name), 'Tipos definidos pelo usuário' (User-defined types), and 'Criar tipo' (Create type).

---

## **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>