Plano de testes - API ServeRest

Apresentação do Plano de Testes

O presente documento descreve o planejamento de testes para a API ServeRest, com foco em **verificação de requisitos funcionais**, **validação de regras de negócio** e **cobertura de endpoints**. A API simula uma loja virtual, permitindo operações como autenticação, gestão de usuários, produtos e carrinhos.

Objetivo

Garantir a qualidade da aplicação ServeRest, validando se os comportamentos da API estão aderentes aos critérios de aceite definidos nas user stories e documentação Swagger. E para isso utilizei testes manuais e automáticos.

Escopo

Endpoints:

- [x] /usuarios
- [x] /login
- [x] /produtos
- [x] /carrinhos

- Validação de regras de negócio descritas nas user stories (ex: não permitir e-mails do Gmail/Hotmail).
- Testes positivos, negativos e alternativos.
- Scripts automatizados básicos na aba "Tests" do Postman.

Fora do escopo:

- [ ] Testes de performance

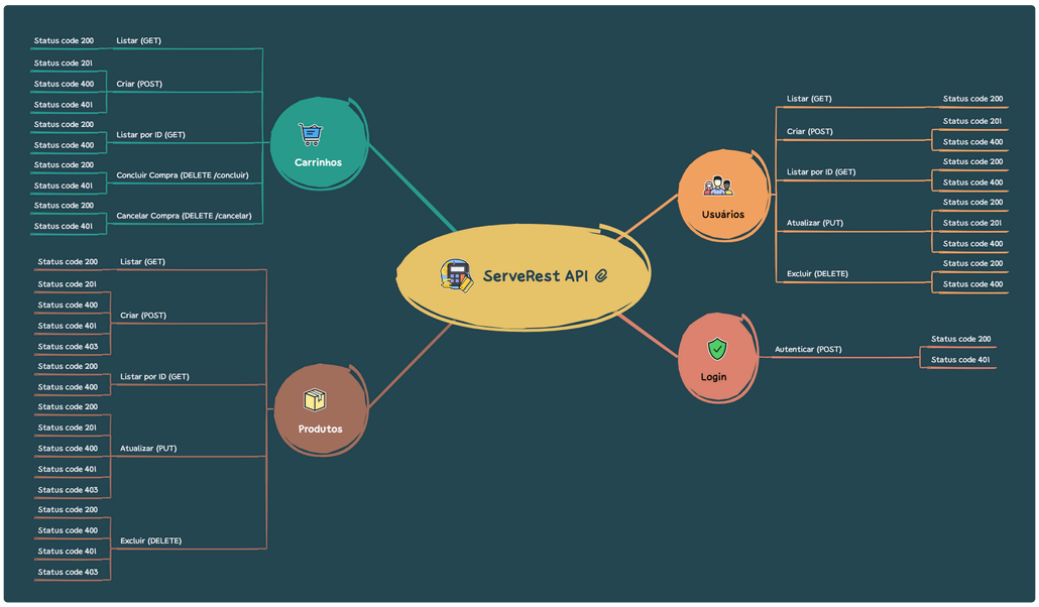
Análise

- **US001 – Usuários**: Validação total do CRUD + normas específicas de e-mail e senha.
- **US002 – Login**: Processo de autenticação com abordagem para senhas incorretas, tokens vencidos.
- **US003 – Produtos**: CRUD com autenticação e limitações como nomes repetidos.
- Interações com o carrinho para verificar as regras de exclusão condicional de itens.

Técnicas aplicadas

1. Análise de requisitos
2. Teste baseado em risco
3. Partição de equivalência → válido e inválido
4. Análise de Valor Limite → Ex: Senha com menos de 5 caracteres
5. Técnica de Fluxo Principal e Alternativo
6. Modelagem visual com mapa mental

Mapa mental do ServeRest



Representação visual os endpoints da API, suas requisições e possíveis respostas (status codes).

Cenários de teste planejados

ID	Recurso	Cenário	Tipo de cenário
CT01	Usuários	Criar usuário com dados válidos	Positivo
CT02	Usuários	Criar usuário com e-mail já utilizado	Negativo
CT03	Usuários	Criar usuário com e-mail Gmail ou/e Hotmail	Negativo
CT04	Usuários	Criar usuário com senha > 10	Negativo
CT05	Usuários	Criar usuário com senha < 5	Negativo
CT06	Login	Login com credenciais válidas	Positivo
CT07	Login	Login com senha inválida	Negativo
CT08	Login	Login após token expirado	Negativo
CT09	Produtos	Criar produto autenticado	Positivo
CT10	Produtos	Criar produto com nome duplicado	Negativo
CT11	Produtos	Tentar excluir produto que já estão em carrinho	Negativo
CT12	Produtos	Tentar criar produto sem autenticação	Negativo
CT13	Produtos	Tentar listar produtos sem autenticação	Negativo
CT14	Produtos	Tentar atualizar produto sem autenticação	Negativo
CT15	Produtos	Tentar excluir produto sem autenticaçã	Negativo
CT16	Carrinho	Criar carrinho com produtos válidos	Positivo
CT17	Carrinho	Criar dois carrinhos com mesmo usuário	Negativo

Mapeamento de cenários previstos

Priorização de Execução

Priorização da Execução dos Cenários de Teste

Cenários de Teste	Probabilidade de falha	Impacto para o négocio	Dependência de outras funcionalidades	Caminho crítico do sistema	Frequência de uso	Prioridade
Criar Usuário	Alto	Alto	Alto	Alto	Alto	59049
Login	Alto	Alto	Alto	Alto	Alto	59049
Carrinho	Médio	Médio	Alto	Alto	Alto	6561
Criar Produto	Baixo	Médio	Alto	Alto	Médio	729
PUT/DELETE	Médio	Médio	Médio	Médio	Médio	243
GET por ID	Baixo	Baixo	Baixo	Médio	Baixo	3
Fluxo alternativo	Baixo	Baixo	Baixo	Baixo	Baixo	1

- **Alta Prioridade**: Login, Criar Usuário
- **Média Prioridade**: Criar Produto, Carrinho, PUT/DELETE
- **Baixa Prioridade**: GET por ID, fluxos alternativos

Matriz de Risco

Matriz de risco validando a priorização acima

Cobertura de Testes

Tipo de cobertura	Descrição	Total	Cobertura	%
Cobertura de paths	URL + URN (Resource name). Ex.: /usuarios	9	9	100%
Cobertura de operações	Todos os métodos - GET, POST, PUT, DELETE...	16	16	100%
Cobertura de parâmetros	Todos os parâmetros definidos para o endpoint	39	39	100%
Cobertura de respostas	Diferentes tipos de resposta documentados	38	28	73,7%
		Média de cobertura		93,4%

Testes Candidatos à Automação

Critérios: Repetição frequente, facilidade de validação, rápida execução...

- Login com sucesso e falha
- CRUD completo de usuários
- Testes de produto com autenticação
- Verificação de status code e headers

Compilado de Resultados

/Usuários

CT01 – Criar usuário com dados válidos

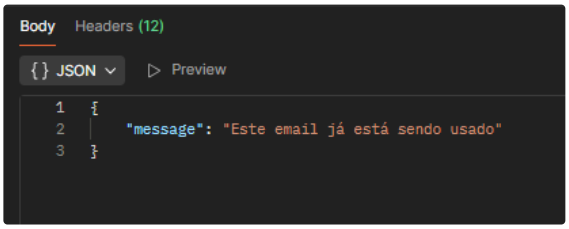
- **Descrição do teste**: Verifica o cadastro de um usuário com nome, e-mail e senha válidos.
- **Resultado esperado**: Código 201, retorno de mensagem de sucesso e ID do usuário.
- **Requisição executada**: POST /usuarios

```
{
  "nome": "Fulano Teste",
  "email": "fulano@qa.com",
  "password": "teste123",
  "administrador": "true"
}
```

Resultado obtido: 201 created

CT02 – Criar usuário com e-mail já utilizado

- **Descrição do teste:** Impede o cadastro duplicado de um e-mail já existente.
- **Resultado esperado:** Código 400, mensagem "Este email já está sendo usado".

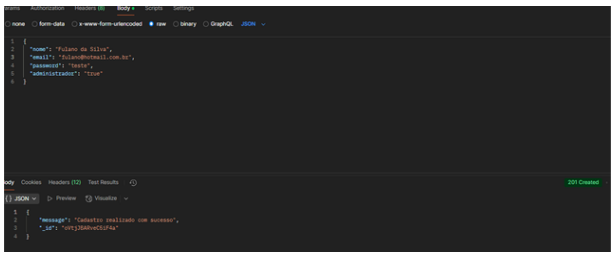


Resultado obtido: 400 Bad Request

CT03 – Criar usuário com e-mail Gmail ou Hotmail

- **Descrição do teste:** Verifica rejeição de e-mails com domínios proibidos.
- **Resultado esperado:** Código 400, erro de validação por domínio.
- **Requisição executada**:

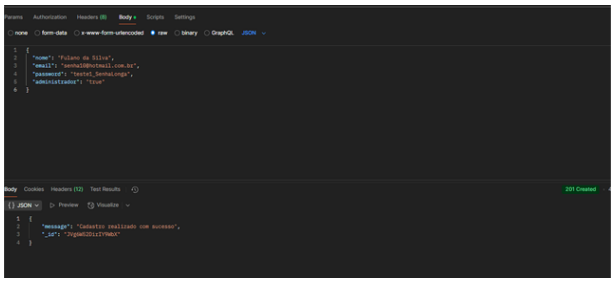
```
{
  "nome": "Fulano da Silva",
  "email": "tester@hotmail.com",
  "password": "teste123",
  "administrador": "false"
}
```



Resultado obtido: mesmo com o domínio “hotmail” retornou status 201 created

CT04 – Criar usuário com senha > 10

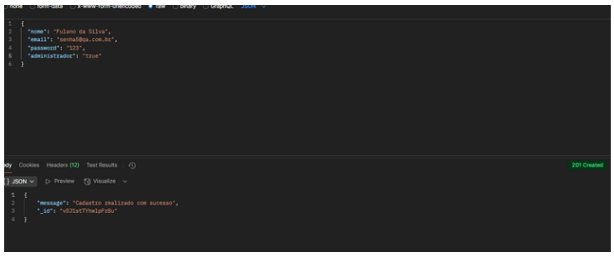
- **Descrição do teste:** Testa rejeição de senha com mais de 10 caracteres.
- **Resultado esperado:** Código 400, erro de validação.



Resultado obtido: mesmo com a senha com mais de 10 caracteres retornou status 201 created

CT05 – Criar usuário com senha < 5

- **Descrição do teste:** Testa rejeição de senha curta.
- **Resultado esperado:** Código 400, erro de validação.



Resultado obtido: mesmo com a senha com menos de 5 caracteres retornou status 201 created

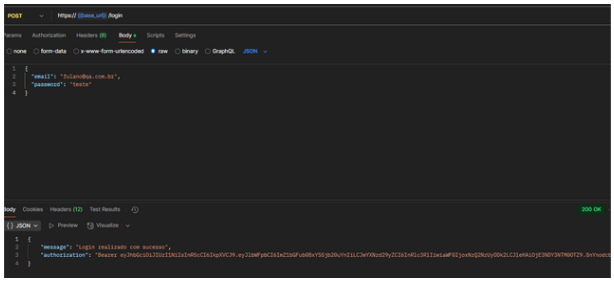
/Login

CT06 – Login com credenciais válidas

- **Descrição do teste:** Valida autenticação com usuário válido.
- **Resultado esperado:** Código 200, token retornado.
- **Requisição**: POST /login

```
{
  "email": "fulano@qa.com.br",
```

```
  "password": "teste"
}
```

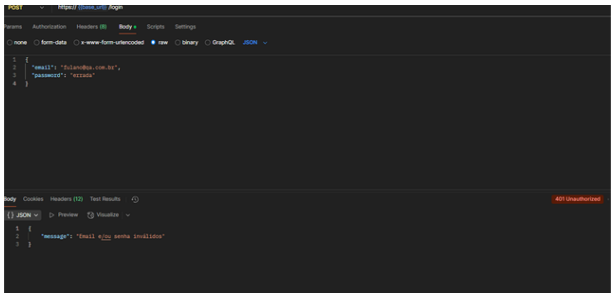


Resultado obtido**:** Código 200, token retornado.

CT07 – Login com senha inválida

- **Descrição do teste:** Garante erro ao usar senha incorreta.
- **Resultado esperado:** Código 401, mensagem de erro.
- **Requisição executada**:

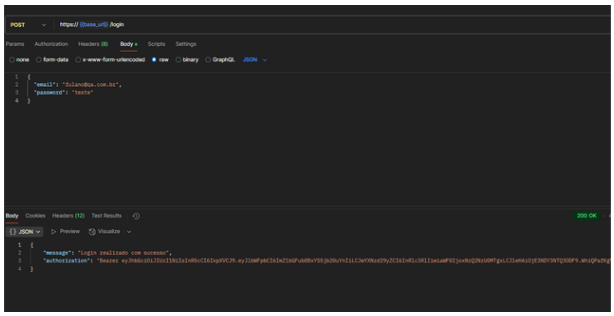
```
{
  "email": "maria@qa.com",
  "password": "errada"
}
```



CT08 – Login após token expirado

- **Descrição do teste:** Verifica se o sistema retorna 401 após o token expirar (10 min).
- **Resultado esperado:** Código 401, mensagem “Token expirado”.
- **Requisição executada:**
  GET /usuarios com header:

```
Authorization: Bearer {{token_expirado}}
```



Resultado obtido: Status 200 mesmo após o período determinado

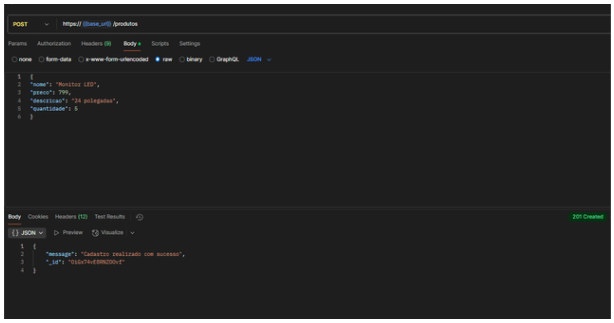
/Produtos

CT09 – Criar produto autenticado

- **Descrição do teste:** Verifica criação de produto com token válido.
- **Resultado esperado:** Código 201, ID e mensagem de sucesso.

POST /produtos Authorization: Bearer {{token}}

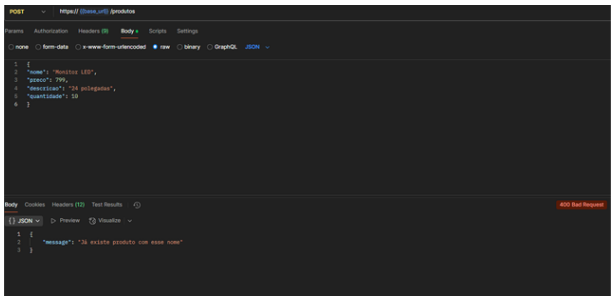
```
{
    "nome": "Monitor LED",
    "preco": 799,
    "descricao": "24 polegadas",
    "quantidade": 5
}
```



Resultado Obtido: Status 201

CT10 – Criar produto com nome duplicado

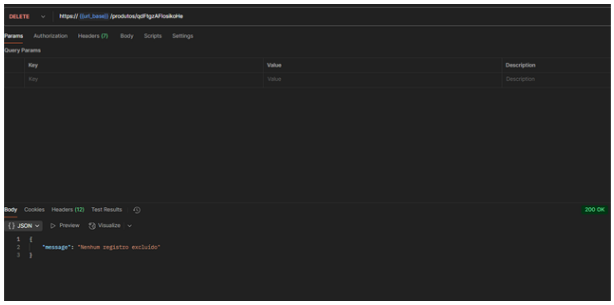
- **Descrição do teste:** Valida bloqueio de nomes duplicados.
- **Resultado esperado:** Código 400, mensagem "Produto já existe".



Resultado Obtido: 400 Bad Request - “Já existe produto com esse nome”

CT11 – Tentar excluir produto que está em carrinho

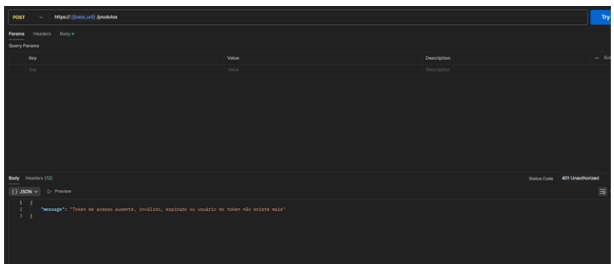
- **Descrição do teste:** Garante que a exclusão de produto em uso no carrinho não é permitida.
- **Resultado esperado:** Código 400, mensagem de erro.
- **Requisição executada:** `DELETE /produtos/{idProduto}` com produto vinculado a um carrinho



Retorno obtido status 200 - “Nenhum registro excluido”

CT12 – Criar produto sem autenticação

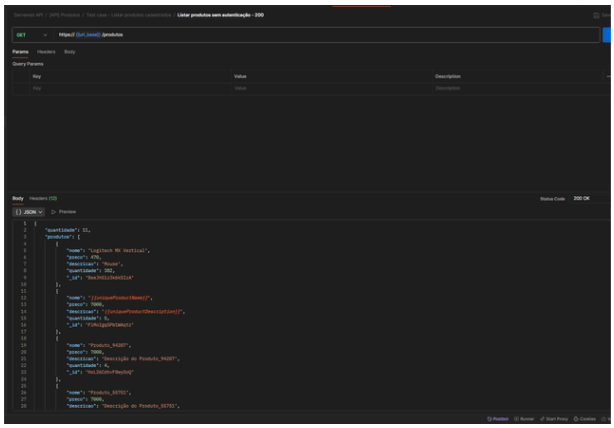
- **Descrição do teste**: Testa proteção de rota de criação de produto sem token.
- **Resultado esperado**: Código 401, mensagem “Token ausente”.
- **Requisição executada**: POST /produtos com payload válido, mas sem token.



Resultado obtido: 401, não autorizado

CT13 – Listar produtos sem autenticação

- **Descrição do teste**: Verifica se o endpoint de listagem exige token.
- **Resultado esperado**: Código 200 (se pública).
- **Requisição executada**: GET /produtos sem header de Authorization



Resultado obtido: status code 200 ok

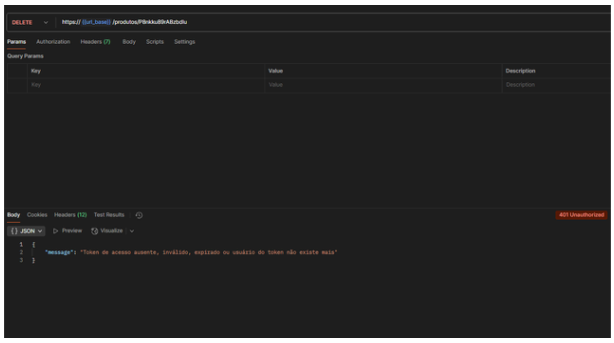
CT14 – Atualizar produto sem autenticação

- **Descrição do teste**: Garante que PUT sem token seja bloqueado.
- **Resultado esperado**: Código 401, mensagem “Token ausente”.
- **Requisição executada**: PUT /produtos/{id} sem header Authorization

Retorno obtido: 401, não autorizado

CT15 – Excluir produto sem autenticação

- **Descrição do teste**: Garante que DELETE sem token não seja aceito.
- **Resultado esperado**: Código 401.
- **Requisição executada**: DELETE /produtos/{id} sem token



Resultado obtido: Code 401 não autorizado

/Carrinho

CT16 - Criar carrinho com produtos válidos

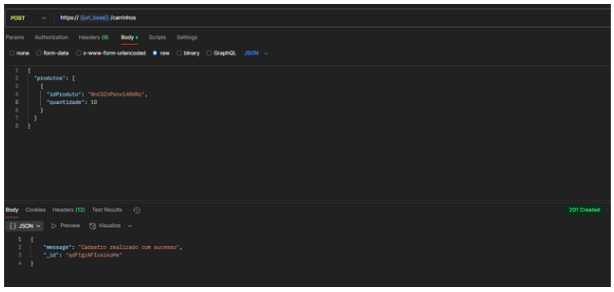
Descrição do teste: Valida o cadastro de um carrinho com token e produtos válidos.

Resultado esperado: Código 201, mensagem de sucesso.

Requisição executada: POST /carrinhos

```
{
  "produtos": [
```

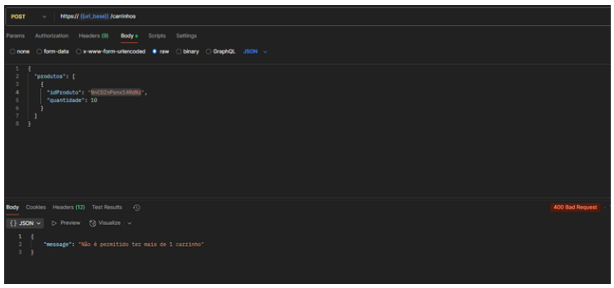
```
    {
        "idProduto": " {{id_produto_valido}}",
        "quantidade": 2
    }
  ]
}
```



Resultado obtido: 200 ok

CT17 – Criar dois carrinhos com mesmo usuário

- **Descrição do teste:** Garante que a API impede múltiplos carrinhos para o mesmo usuário.
- **Resultado esperado:** Código 400, mensagem “Não é permitido ter mais de 1 carrinho”.
- **Requisição executada:** Segundo `POST /carrinhos` usando mesmo token do CT16

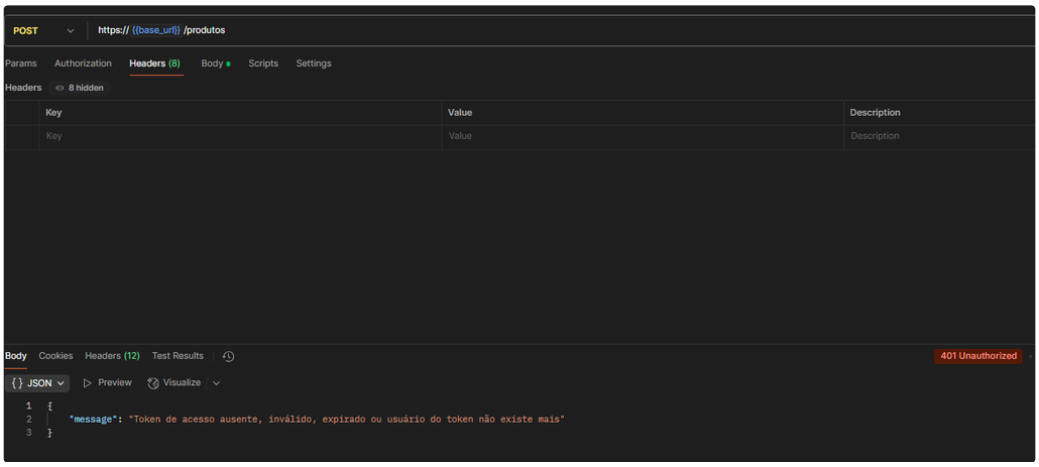


Resposta obtida: 400 Bad request - “Não é permitido ter mais de 1 carrinho”

Testes Alternativos

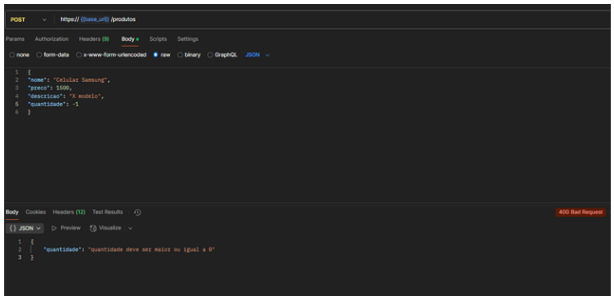
TA01 - Cadastrar produto com Token de autorização expirado

- **Descrição**: Verifica se consigo cadastrar mesmo com a autorização expirada.
- **Resultado:** esperado e Obtido - código 401, não autorizado.



TA02 - Cadastrar produto com quantidade negativa

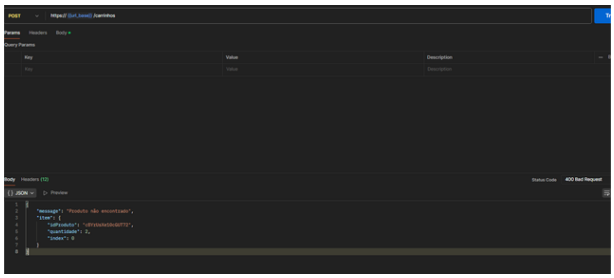
- **Descrição**: Verificar se consigo cadastrar produtos com a quantidade negativa
- **Resultado esperado e Obtido**: Código 400, bad request



Resultado obtido: 400 Bad Request “quantidade deve ser maior ou igual a 0”

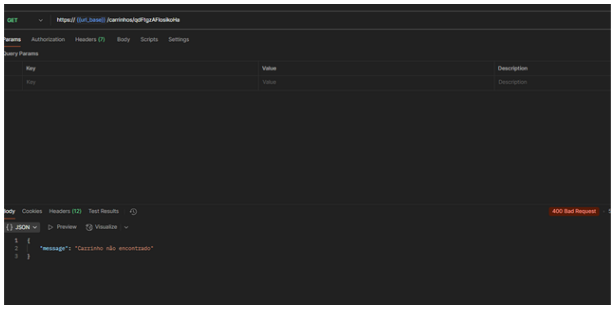
TA03 - Cadastrar carrinho sem produto existente

- **Descrição**: Verificar se consigo cadastrar carrinho sem produto existente
- **Resultado esperado e Obtido**: Código 400, bad request



TA04 - Listar carrinho com ID inválido

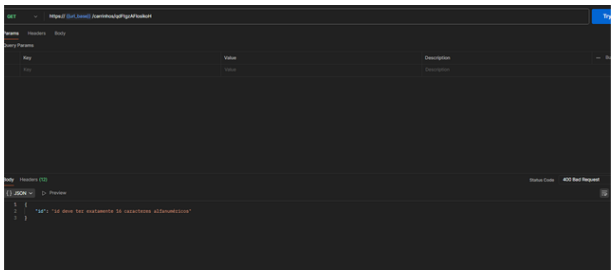
- **Descrição**: Verificar se consigo listar carrinho com ID inexistente
- **Resultado esperado e Obtido**: Código 400, bad request



Resposta obtida: Carrinho não encontrado code 400

TA05 - Listar carrinho com ID < 16 caracteres

- **Descrição**: Verificar se consigo listar carrinho com ID < 16 caracteres
- **Resultado esperado e Obtido**: Código 400, bad request



Resposta obtida: id deve ter exatamente 16 caracteres alfanuméricos - code 400

Issues encontrados

ID	Recurso	Cenário	Resultado testes
CT01	Usuários	Criar usuário com dados válidos	OK
CT02	Usuários	Criar usuário com e-mail já utilizado	OK
CT03	Usuários	Criar usuário com e-mail Gmail ou/e Hotmail	ISSUE
CT04	Usuários	Criar usuário com senha > 10	ISSUE
CT05	Usuários	Criar usuário com senha < 5	ISSUE
CT06	Login	Login com credenciais válidas	OK
CT07	Login	Login com senha inválida	OK
CT08	Login	Login após token expirado	ISSUE
CT09	Produtos	Criar produto autenticado	OK
CT10	Produtos	Criar produto com nome duplicado	OK
CT11	Produtos	Tentar excluir produto que já estão em carrinho	OK
CT12	Produtos	Tentar criar produto sem autenticação	OK
CT13	Produtos	Tentar listar produtos sem autenticação	OK
CT14	Produtos	Tentar atualizar produto sem autenticação	OK
CT15	Produtos	Tentar excluir produto sem autenticação	OK
CT16	Carrinho	Criar carrinho com produtos válidos	OK
CT17	Carrinho	Criar dois carrinhos com mesmo usuário	OK

Issues encontrados nos cenários de teste

Type	Key	Resumo	Responsável	Prioridade
	CHA01-4	[BUG04] Token Bearer não expira após 10 minutos de cadastro	Maria Eduarda ...	High
	CHA01-3	[BUG03] Validação de senha curta não é aplicada – aceita < 5 caracteres	Maria Eduarda ...	High
	CHA01-2	[BUG02] Validação de comprimento da senha não é aplicada – > 10	Maria Eduarda ...	Medi...
	CHA01-1	[BUG01] Validação incorreta de domínio de e-mail - Hotmail e Gmail permiti...	Maria Eduarda ...	High

4 itens

Sincronizado agora

Collection Postman

Testes manuais

Serverest API.... on.json
09 mai. 2025, 02:26 PM

Testes manuais + automatizado perfomance

Serverest API -...on.json
09 mai. 2025, 02:26 PM

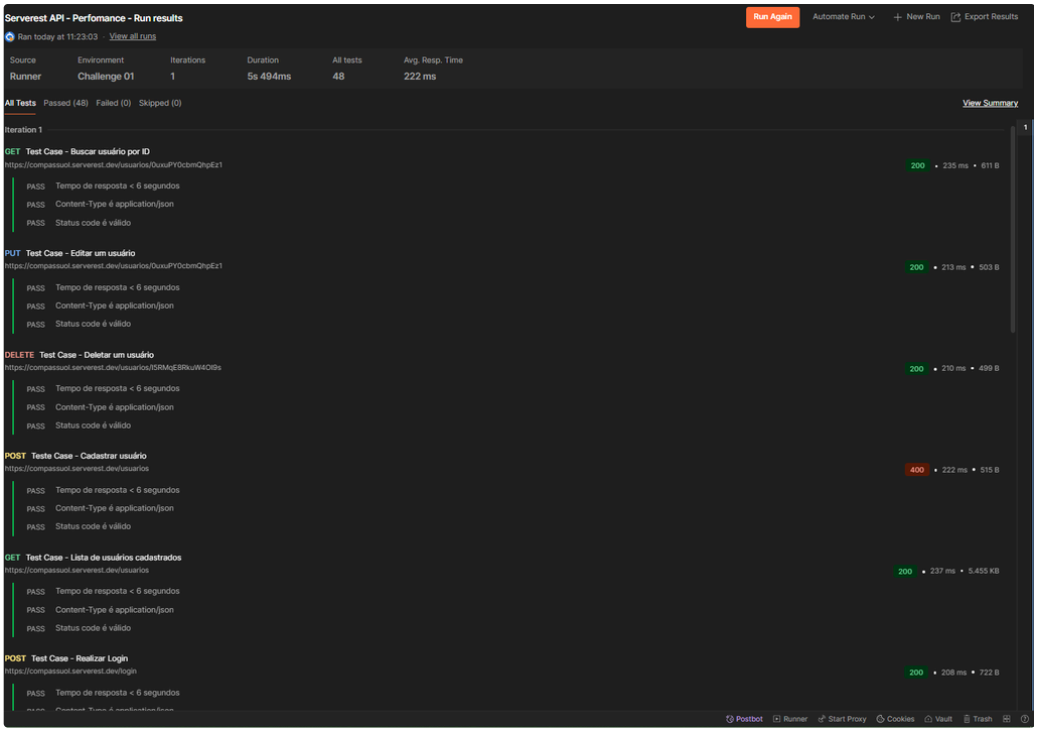
Automatização - Script + Resultado Runner

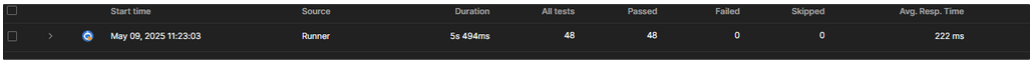
```
pm.test("Tempo de resposta < 6 segundos", function () {
    pm.expect(pm.response.responseTime).to.be.below(6000);
});

pm.test("Content-Type é application/json", function () {
    pm.response.to.have.header("Content-Type");
    pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Status code é válido", function () {
    const status = pm.response.code;
    pm.expect([200, 201, 204, 400, 401, 403, 404, 500]).to.include(status);
```

```
});
```


Serverest API - Perfomance - Run results
Run Again
Automate Run
New Run
Export Results
Ran today at 11:23:03 · View all runs
Source
Runner
Environment
Challenge 01
Iterations
1
Duration
5s 494ms
All tests
48
Avg. Resp. Time
222 ms
All Tests Passed (48) Failed (0) Skipped (0)
View Summary
Iteration 1
GET Test Case - Buscar usuário por ID
https://compassuol.serverest.dev/usuarios/0uxuPY0cbmQhpEz1
200 • 235 ms • 611 B
PASS Tempo de resposta < 6 segundos
PASS Content-Type é application/json
PASS Status code é válido
PUT Test Case - Editar um usuário
https://compassuol.serverest.dev/usuarios/0uxuPY0cbmQhpEz1
200 • 213 ms • 503 B
PASS Tempo de resposta < 6 segundos
PASS Content-Type é application/json
PASS Status code é válido
DELETE Test Case - Deletar um usuário
https://compassuol.serverest.dev/usuarios/I5RMqE8RkuW4OI0s
200 • 210 ms • 499 B
PASS Tempo de resposta < 6 segundos
PASS Content-Type é application/json
PASS Status code é válido
POST Teste Case - Cadastrar usuário
https://compassuol.serverest.dev/usuarios
400 • 222 ms • 515 B
PASS Tempo de resposta < 6 segundos
PASS Content-Type é application/json
PASS Status code é válido
GET Test Case - Lista de usuários cadastrados
https://compassuol.serverest.dev/usuarios
200 • 237 ms • 5.455 KB
PASS Tempo de resposta < 6 segundos
PASS Content-Type é application/json
PASS Status code é válido
POST Test Case - Realizar Login
https://compassuol.serverest.dev/login
200 • 208 ms • 722 B
PASS Tempo de resposta < 6 segundos
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Start time
Source
Duration
All tests
Passed
Failed
Skipped
Avg. Resp. Time
May 09, 2025 11:23:03
Runner
5s 494ms
48
48
0
0
222 ms