

# Ana - Plano de Teste



## Apresentação [🔗](#)

A API REST ServeRest é uma ferramenta gratuita que foi desenvolvida para simular uma loja virtual com o principal objetivo de ser usada como material de estudo em testes de API. Em sua documentação ([👉 ServeRest](#)), conta com os endpoints `/login`, `/usuarios`, `/produtos` e `/carrinhos`. Esses recursos permitem que sejam realizadas explorações e análises em busca de possíveis erros, sendo possível que interessados na área de qualidade de software e teste de APIs REST apliquem seus conhecimentos de forma prática.



## Objetivo [🔗](#)

O objetivo principal da realização dos testes realizados na API ServeRest é garantir a qualidade da aplicação por meio da validação das regras de negócio e da identificação erros e possíveis oportunidades de melhoria.



## Escopo [🔗](#)

Serão testados os seguintes recursos e suas respectivas requisições, conforme documentado na API ServeRest:

- `/login`
  - Realizar login
- `/usuarios`
  - Listar usuários cadastrados
  - Cadastrar usuário
  - Buscar usuário por ID
  - Excluir usuário
  - Editar usuário
- `/produtos`
  - Listar produtos cadastrados
  - Cadastrar produto
  - Buscar produto por ID
  - Excluir produto
  - Editar produto
- `/carrinhos`
  - Listar carrinhos cadastrados
  - Cadastrar carrinho
  - Buscar carrinho por ID
  - Excluir carrinho (quando compra foi concluída)
  - Excluir carrinho e retornar produtos para estoque (quando compra foi cancelada)

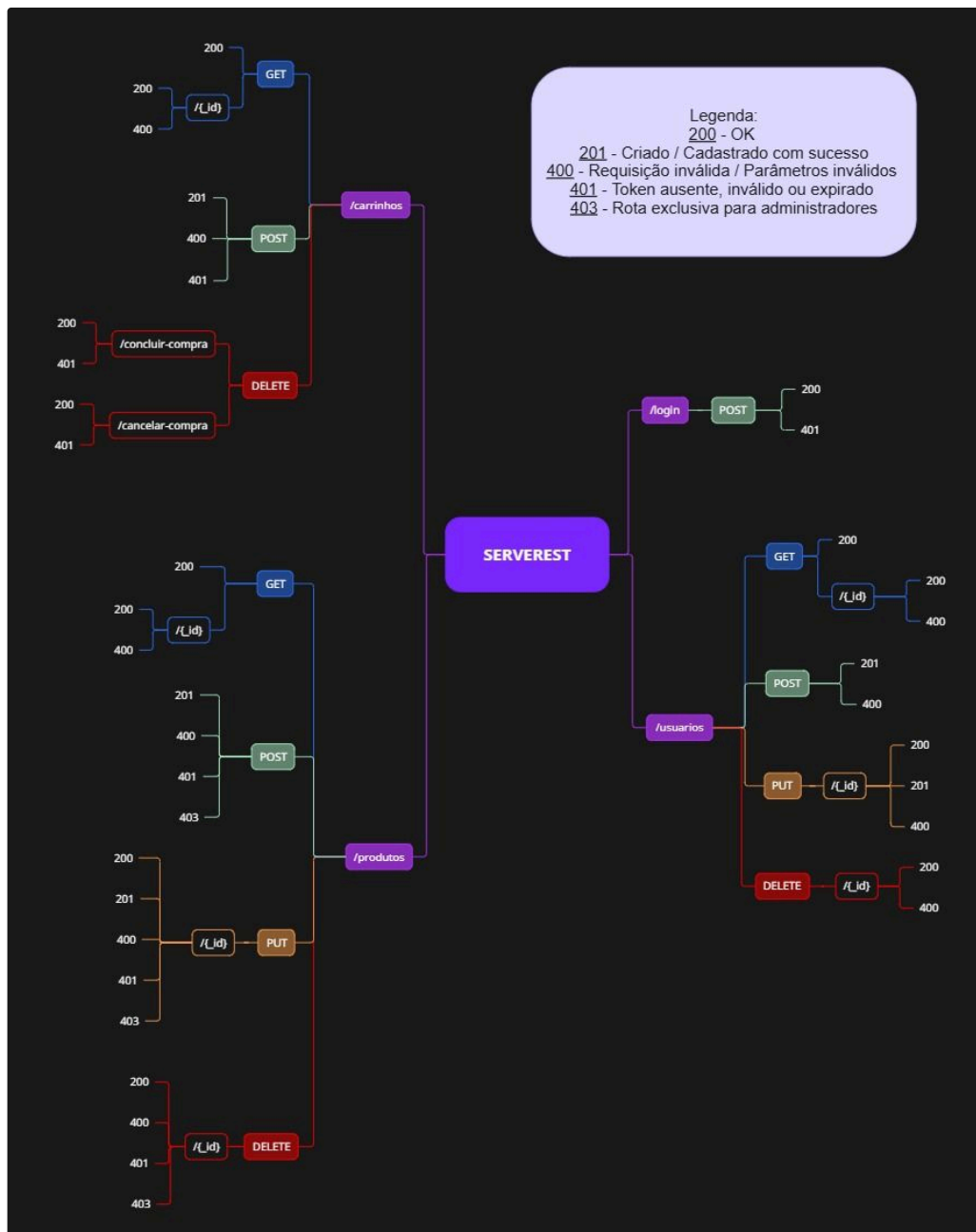


## Ferramentas utilizadas [🔗](#)

- Swagger da API ServeRest: [👉 ServeRest](#)
- Postman
- Jira
- Confluence

- Miro

## 🧠 Mapa mental da aplicação 🔗



Mapa mental da API ServeRest realizado na ferramenta Miro

## 🔍 Análise 🔗

### Cenários de teste 🔗

Os cenários de teste foram elaborados com base nos requisitos definidos nas User Stories dos endpoints `/login`, `/usuarios` e `/produtos`, bem como na documentação Swagger referente ao endpoint `/carrinhos`. Além disso, foram incluídos cenários de “happy path” para garantir que os fluxos principais de cada recurso estejam funcionando corretamente.

### Técnicas aplicadas 🔗

- Testes baseados em cenários
  - Foi a principal técnica aplicada tendo como base os requisitos extraídos das User Stories e da documentação Swagger.
  - Abordagem permitiu construir casos de uso realistas que cobrem tanto os fluxos principais quanto os alternativos.
- Testes exploratórios
  - Abordagem utilizada no endpoint `/carrinhos` analisando comportamentos através da documentação da API, já que ele que não possuía sua própria User Story.
- Partição de equivalência
  - Abordagem utilizada para validar entradas válidas e inválidas no campo de senha que precisava ter de 5 a 10 caracteres. Foram testadas diferentes classes de equivalência para garantir o comportamento correto com entradas dentro e fora do intervalo permitido.

## Requisitos [↗](#)

### `/login` [↗](#)

Baseados nos critérios de aceite da US 002: [API] Login.

ID	Descrição
R1.1	Sistema não deve permitir autenticação de usuários não cadastrados
R1.2	Sistema não deve permitir autenticação de usuário com senha inválida
R1.3	Sistema deve retornar status code 401 (Unauthorized) ao falhar autenticação
R1.4	Ao autenticar com sucesso, sistema deve ser gerado um token Bearer
R1.5	Cada token Bearer gerado deve ter validade de 10 minutos

### `/usuarios` [↗](#)

Baseados nos critérios de aceite da US 001 - [API] Usuários.

ID	Descrição
R2.1	Cada vendedor (usuário) deve ter os campos NOME, E-MAIL, PASSWORD e ADMINISTRADOR
R2.2	Sistema não deve permitir que ações sejam executadas para usuários inexistentes (exceto em PUT)
R2.3	Se ID informado no PUT for de um usuário inexistente, um novo usuário deve ser criado pelo sistema
R2.4	Sistema não deve permitir que seja criado um novo usuário com e-mail já utilizado em POST e PUT
R2.5	Sistema não deve permitir que sejam cadastrados usuários com e-mail gmail e hotmail
R2.6	E-mails devem seguir padrão válido de e-mail para cadastro (ex: nome@dominio.com)
R2.7	Senhas devem ter de 5 a 10 caracteres
R2.8	Sistema não deve permitir que usuário com carrinho cadastrado seja excluído

R2.9	Sistema deve listar usuários cadastrados, buscar usuários por ID, excluir usuários e editar usuários corretamente
------	---

## /produtos

Baseados nos critérios de aceite da US 003: [API] Produtos.



ID	Descrição
R3.1	Sistema não deve permitir que usuários não autenticados consigam realizar ações na rota
R3.2	Sistema não deve permitir que produtos dentro do carrinho sejam excluídos
R3.3	Se ID informado no momento do UPDATE for de um produto inexistente, um novo produto deve ser criado pelo sistema
R3.4	Sistema não deve permitir o cadastro em POST e PUT de produtos com nomes já utilizado
R3.5	Sistema deve listar produtos cadastrados, cadastrar produtos, buscar produtos por ID, excluir produtos e editar produtos corretamente

## /carrinhos

Baseados na documentação Swagger.

ID	Descrição
R4.1	Sistema não deve permitir que usuários não autenticados consigam realizar ações na rota
R4.2	Sistema deve cadastrar apenas um carrinho por usuário
R4.3	Sistema deve reduzir a quantidade no cadastro do produto que foi inserido em um carrinho com cadastro bem sucedido
R4.4	Sistema deve excluir um carrinho quando uma compra for concluída
R4.5	Sistema deve excluir o carrinho e atualizar o estoque dos produtos contidos nele quando uma compra for cancelada
R4.6	Sistema deve listar carrinhos cadastrados e buscar carrinhos por ID corretamente

## Cenários de teste planejados

ID	Descrição	Pré-condições	Passo a passo	Resultado esperado	Resultados obtidos	Status	Requisitos que cobre	Prioridade 
CT01	Login com usuário existente	Usuário cadastrado	1. Criar requisição POST para /login 2. Incluir e-mail e senha válidos no corpo da requisição	200 e Token Bearer do usuário	200 e Token Bearer do usuário	 Passou	R1.4	Alta

CT02	Login com usuário existente e senha incorreta	Usuário cadastra do	1. Criar requisição POST para <code>/login</code> 2. Incluir e-mail válido e senha inválida no corpo da requisição	401	401	✓ Passou	R1.2 e R1.3	Alta
CT03	Login com usuário inexistente	Nenhuma	1. Criar requisição POST para <code>/login</code> 2. Incluir e-mail e senha inválidos no corpo da requisição	401	401	✓ Passou	R1.1 e R1.3	Alta
CT04	Realizar alguma ação na API após validade do token de usuário	Token expirado	1. Criar requisição POST para <code>/login</code> 2. Incluir e-mail e senha válidos no corpo da requisição 3. Copar token retornado 4. Aguardar expiração do token	401	401	✓ Passou	R1.5	Alta
CT05	Listar usuários cadastrados	Parâmetros válidos	1. Criar requisição GET para <code>/usuarios</code> 2. Incluir nome, e-mail, senha, administrador e ID do usuário válidos no parâmetros da requisição	200	200	✓ Passou	R2.9	Média
CT06	Cadastrar usuário com os campos NOME, E-MAIL, PASSWORD e ADMINISTRADOR válidos	Dados válidos	1. Criar requisição POST para <code>/usuarios</code> 2. Incluir nome, e-mail, senha e administrador válidos no corpo da requisição	201 e ID do novo usuário cadastrado	201 e ID do novo usuário cadastrado	✓ Passou	R2.1	Alta
CT07	Buscar, excluir e editar usuário com ID válido	Parâmetros e dados válidos	1. Criar requisição GET para <code>/usuarios/{_id}</code> e incluir ID do usuário válido no parâmetros da requisição	200	200	✓ Passou	R2.9	Média

			<p>2. Criar requisição DELETE para <code>/usuarios/{_id}</code> e incluir ID do usuário válido no parâmetros da requisição</p> <p>3. Criar requisição PUT para <code>/usuarios/{_id}</code> e incluir ID do usuário válido no parâmetros e nome, e-mail, senha e administrador válidos no corpo da requisição</p>					
CT08	Buscar usuário com ID de usuário inexistente	Nenhuma	<p>1. Criar requisição GET para <code>/usuarios/{_id}</code></p> <p>2. Incluir ID do usuário que não existe no parâmetros da requisição</p>	400	400	✓ Passou	R2.2	Baixa
CT09	Deletar usuário com ID de usuário inexistente	Nenhuma	<p>1. Criar requisição DELETE para <code>/usuarios/{_id}</code></p> <p>2. Incluir ID do usuário que não existe no parâmetros da requisição</p>	200 ("Nenhum registro excluído")	200 ("Nenhum registro excluído")	✓ Passou	R2.2	Baixa
CT10	Atualizar usuário com ID de usuário inexistente	Nenhuma	<p>1. Criar requisição PUT para <code>/usuarios/{_id}</code></p> <p>2. Incluir ID do usuário que não existe no parâmetros e nome, e-mail, senha e administrador válidos no corpo da requisição</p>	201 e ID do novo usuário cadastrado	201 e ID do novo usuário cadastrado	✓ Passou	R2.3	Média

CT11	Criar usuário em POST com e-mail já utilizado	E-mail de usuário cadastrado	1. Criar requisição POST para <code>/usuarios</code> 2. Incluir e-mail já utilizado no corpo da requisição	400	400	✓ Passou	R2.4	Alta
CT12	Criar usuário com e-mail gmail e hotmail	E-mail gmail e e-mail hotmail	1. Criar requisição POST para <code>/usuarios</code> e incluir e-mail com @gmail.com no corpo da requisição 2. Criar requisição POST para <code>/usuarios</code> e incluir e-mail com @hotmail.com no corpo da requisição	400	200	✗ Falhou	R2.5	Média
CT13	Criar usuário com e-mail fora do padrão válido	Nenhuma	1. Criar requisição POST para <code>/usuarios</code> 2. Incluir e-mail fora do formato padrão nome@dominio.com no corpo da requisição	400	400	✓ Passou	R2.6	Média
CT14	Criar usuário com senha válida	Nenhuma	1. Criar requisição POST para <code>/usuarios</code> 2. Incluir password válida com 5 a 10 caracteres no corpo da requisição	201 e ID do novo usuário cadastrado	201 e ID do novo usuário cadastrado	✓ Passou	R2.7	Alta
CT15	Criar usuário com senha inválida curta	Nenhuma	1. Criar requisição POST para <code>/usuarios</code> 2. Incluir password inválida com menos de 5 caracteres no corpo da requisição	400	200	✗ Falhou	R2.7	Média
CT16	Criar usuário com senha inválida longa	Nenhuma	1. Criar requisição POST para <code>/usuarios</code>	400	200	✗ Falhou	R2.7	Média

			2. Incluir senha inválida com mais de 10 caracteres no corpo da requisição					
CT17	Excluir usuário com carrinho cadastrado	Usuário e carrinho cadastrados	1. Criar requisição DELETE para <code>/usuarios/{_id}</code> 2. Incluir ID do usuário que possui um carrinho cadastrado no parâmetros da requisição	400	400	✓ Passou	R2.8	Alta
CT18	Listar produtos cadastrados	Dados válidos	1. Criar requisição GET para <code>/produtos</code> 2. Incluir ID do produto, nome, preço, descrição e quantidade válidos no parâmetros da requisição	200	200	✓ Passou	R3.5	Média
CT19	Cadastrar produto válido	Dados válidos e usuário autenticado	1. Criar requisição POST para <code>/produtos</code> 2. Incluir nome, preço, descrição e quantidade no corpo da requisição 3. Adicionar token de autenticação no cabeçalho	201 e ID do produto cadastrado	201 e ID do produto cadastrado	✓ Passou	R3.5	Alta
CT20	Buscar, excluir e editar com ID de produto válido	Parâmetros e dados válidos e usuário autenticado	1. Criar requisição GET para <code>/produtos/{_id}</code> e incluir ID do produto válido no parâmetros da requisição 2. Criar requisição DELETE para <code>/produtos/{_id}</code> e incluir ID do produto válido no parâmetros da requisição	200	200	✓ Passou	R3.5	Média



			<p>3. Adicionar token de autenticação no cabeçalho</p> <p>4. Criar requisição PUT para <code>/produtos/{_id}</code> e incluir ID do produto válido no parâmetros e nome, preço, descrição e quantidade válidos no corpo da requisição</p> <p>5. Adicionar token de autenticação no cabeçalho</p>					
CT21	Cadastrar, excluir e editar produto com usuário não autenticado	Usuário não autenticado	<p>1. Criar requisição POST para <code>/produtos</code> e incluir nome, preço, descrição e quantidade no corpo da requisição</p> <p>2. Não adicionar token de autenticação no cabeçalho</p> <p>3. Criar requisição DELETE para <code>/produtos/{_id}</code> e incluir ID do produto válido no parâmetros da requisição</p> <p>4. Não adicionar token de autenticação no cabeçalho</p> <p>5. Criar requisição PUT para <code>/produtos/{_id}</code> e incluir ID do produto válido no parâmetros e nome, preço, descrição e quantidade válidos no corpo da requisição</p> <p>6. Não adicionar token de autenticação no cabeçalho</p>	401	401	✓ Passou	R3.1	Alta

CT22	Atualizar produto com ID de produto inexistente	Usuário autenticado	<ol style="list-style-type: none"> <li>1. Criar requisição PUT para <code>/produtos/{_id}</code></li> <li>2. Incluir ID do produto que não existe no parâmetros e nome, preço, descrição e quantidade válidos no corpo da requisição</li> <li>3. Adicionar token de autenticação no cabeçalho</li> </ol>	201 e ID do novo produto cadastrado	201 e ID do produto cadastrado	✓ Passou	R3.3	Alta
CT23	Criar produto em POST com nome já utilizado	Nenhuma	<ol style="list-style-type: none"> <li>1. Criar requisição POST para <code>/produtos</code> e incluir nome já utilizado no corpo da requisição</li> <li>2. Adicionar token de autenticação no cabeçalho</li> <li>3. Criar requisição PUT para <code>/produtos/{_id}</code> e incluir nome já utilizado no corpo da requisição</li> <li>4. Adicionar token de autenticação no cabeçalho</li> </ol>	400	400	✓ Passou	R3.4	Média
CT24	Excluir produto dentro do carrinho	Carrinho cadastrado e usuário autenticado	<ol style="list-style-type: none"> <li>1. Criar requisição DELETE para <code>/produtos/{_id}</code> e incluir ID de um produto que esteja no carrinho do usuário no parâmetros da requisição</li> <li>2. Adicionar token de autenticação no cabeçalho</li> </ol>	400	400	✓ Passou	R3.2	Alta
CT25	Listar carrinhos cadastrados	Dados válidos	<ol style="list-style-type: none"> <li>1. Criar requisição GET para <code>/carrinhos</code></li> <li>2. Incluir ID do carrinho, preço total,</li> </ol>	200	200	✓ Passou	R4.6	Média

			quantidade total e ID do usuário válidos no parâmetros da requisição					
CT26	Cadastrar carrinho com usuário válido	Usuário válido e autenticado	<ol style="list-style-type: none"> <li>1. Criar requisição POST para <code>/carrinhos</code></li> <li>2. Incluir ID e quantidade dos produtos no corpo da requisição</li> <li>3. Adicionar token de autenticação no cabeçalho</li> </ol>	201 e redução da quantidade do produto que foi inserido	201 e redução da quantidade do produto que foi inserido	✓ Passou	R4.3	Alta
CT27	Cadastrar carrinho com usuário que já possui um carrinho	Usuário válido com carrinho e autenticado	<ol style="list-style-type: none"> <li>1. Criar requisição POST para <code>/carrinhos</code></li> <li>2. Incluir ID de usuário que já possui um carrinho no corpo da requisição</li> <li>3. Adicionar token de autenticação no cabeçalho</li> </ol>	400	400	✓ Passou	R4.2	Alta
CT28	Buscar carrinho com ID válido	Parâmetro válido	<ol style="list-style-type: none"> <li>1. Criar requisição GET para <code>/carrinhos/{_id}</code></li> <li>2. Incluir ID de carrinho válido no parâmetros da requisição</li> </ol>	200	200	✓ Passou	R4.6	Média
CT29	Deletar carrinho ao concluir compra	Usuário válido com carrinho e autenticado	<ol style="list-style-type: none"> <li>1. Criar requisição DELETE para <code>/carrinhos/concluir-compra</code></li> <li>2. Rodar requisição</li> <li>3. Adicionar token de autenticação no cabeçalho</li> </ol>	200 e carrinho excluído	200	✓ Passou	R4.4	Média
CT30	Deletar carrinho ao cancelar compra	Usuário válido com carrinho e	<ol style="list-style-type: none"> <li>1. Criar requisição DELETE para <code>/carrinhos/cancelar-compra</code></li> <li>2. Rodar requisição</li> </ol>	200, carrinho excluído e reabastecimento	200	✓ Passou	R4.5	Alta

		autentica do	3. Adicionar token de autenticação no cabeçalho	do estoque dos produtos contidos nele				
CT31	Deletar carrinho ao concluir compra com usuário não autenticado	Usuário não autentica do	1. Criar requisição DELETE para /carrinhos/concluir -compra 2. Rodar requisição 3. Não adicionar token de autenticação no cabeçalho	401	401	✓ Passou	R4.1	Alta
CT32	Deletar carrinho ao cancelar compra com usuário não autenticado	Usuário não autentica do	1. Criar requisição DELETE para /carrinhos/cancelar -compra 2. Rodar requisição 3. Não adicionar token de autenticação no cabeçalho	401	401	✓ Passou	R4.1	Alta

## ! Matriz de risco [🔗](#)

ID	Título	Probabilidade (1-3)	Impacto (1-3)	Risco
Risco 001	Falha na autenticação de usuários permite acessos não autorizados	2	3	6
Risco 002	Ausência de validação dos dados inseridos	2	3	6
Risco 003	Queda da API	2	3	6
Risco 004	Tempo insuficiente para execução de todos os testes planejados	3	2	6
Risco 005	Documentação da API desatualizada	2	2	4
Risco 006	Sistema não trata erros corretamente (mensagens genéricas)	2	2	4
Risco 007	Falha no ambiente de teste	2	2	4

Risco 008	Dados inconsistentes ou ausentes na base usada para testes	2	2	4
Risco 009	Mudanças na API sem aviso prévio	1	3	3
Risco 010	Token expirado permite continuar acessando a API	1	3	3



## Cobertura de testes [🔗](#)

### Path Coverage (input) [🔗](#)

Quantidade de endpoints testados: 9

Quantidade de endpoints na API REST: 9

Total =  $9/9 = 1 = 100\%$

### Operator Coverage (input) [🔗](#)

Quantidade de métodos testados: 16

Quantidade de métodos na API REST: 16

Total =  $16/16 = 1 = 100\%$

### Parameter Coverage (input) [🔗](#)

Quantidade de parâmetros testados: 27

Quantidade de parâmetros na API REST: 27

Total =  $27/27 = 1 = 100\%$

### Content-Type Coverage (input e output) [🔗](#)

Quantidade de Content-Type testados: 16

Quantidade de métodos na API REST: 16 (há apenas a opção de JSON)

Total =  $16/16 = 1 = 100\%$

### Status Code Coverage (Output) [🔗](#)

Quantidade de status code testados: 33

Quantidade de status code na API REST: 38

Total =  $31/38 = 0,82 = 82\%$



## Testes candidatos a automação [🔗](#)

Os principais testes candidatos a automação são os repetitivos, os que possuem a possibilidade de adicionar arquivos de dados com valores diferentes e os que têm resultados mais simples e previsíveis. Os cenários de teste que se encaixam nessa descrição são:

- CT01, CT02 e CT03 - Relacionados ao login

- CT05, CT18 e CT25 - Relacionados com a listagem de usuários, produtos e carrinhos
- CT12, CT13, CT14, CT15 e CT16 - Relacionados com criação de usuários com padrões específicos de e-mail e senha