

Plano de Teste - ServeRest [Robot Framework / AWS] (compassuol.serverest.dev)

🔗 Challenge - Sprint 6/Semana 12 Plano de Teste Versão 1.1

@Pedro Afonso de Alencar Silva

📜 Histórico de Revisões 🔗

Histórico de Revisões		
Versão	Data	Descrição
1.0	30 de mai. de 2025	<ul style="list-style-type: none">• Versão Inicial
1.1	30 de mai. de 2025	<p>Atualizações gerais no plano de teste:</p> <ul style="list-style-type: none">• Seção 5 – Técnicas Aplicadas: Substituição do Postman por Robot Framework, execução migrada para AWS (EC2), ajustes nas descrições técnicas unificando testes manuais e automatizados, remoção de menções ao Postman e inclusão de tópicos como uso de token, dados dinâmicos e geração de logs.• Seção 7 – Cenários de Teste Planejados + Rastreabilidade: Remoção da coluna “Executado?”.• Seção 8 – Priorização dos Cenários: Reclassificação das prioridades, inclusão de novos cenários críticos (carrinho, compra, login) e remoção de cenários com menor relevância.• Seção 9 – Matriz de Risco: Atualização dos riscos conforme nova priorização, adição de risco de falha na finalização de compra, remoção de riscos de baixo impacto e revisão da classificação de impacto/probabilidade.• Seção 11 – Cobertura de Testes: Observação incluída indicando que a cobertura representa apenas o que foi testado ao menos uma vez e não cobre a execução final.• Seção 12 – Artefatos Gerados: Inclusão da subseção “Relatório do QAlity Plus”.

📋 Índice 🔗

1. 📄 Apresentação
2. 🎯 Objetivo
3. 💚 Escopo
4. 🔎 Análise
5. 🛠️ Técnicas Aplicadas
6. 🌐 Mapa Mental da API
7. 🌐 Cenários de Teste Planejados + Rastreabilidade
8. 📊 Priorização dos Cenários
9. ⚡ Matriz de Risco

10. Testes Candidatos à Automação
 11. Cobertura de Testes
 12. Artefatos Gerados
 13. Considerações sobre Itens Omitidos
-

Plano de Teste

1. Apresentação

O ServeRest é uma API REST gratuita que simula as **funcionalidades de uma loja virtual**, tendo como **principal objetivo** servir como **ambiente de estudos para testes de APIs**. Ela disponibiliza rotas para gerenciamento de **usuários, autenticação, produtos e carrinhos**, permitindo a execução de testes funcionais, de negócio e exploratórios.

2. Objetivo

Este plano de testes visa assegurar a conformidade da API ServeRest com as **regras de negócio estabelecidas nas User Stories**, complementando as definições técnicas presentes na documentação Swagger. O foco está em validar cenários reais e alternativos, inclusive aqueles não explicitamente descritos, garantindo maior **confiabilidade, rastreabilidade e qualidade** da aplicação.

Adicionalmente, o plano visa identificar cenários candidatos à **automação de testes**, apoiar a cobertura de testes baseada em risco e gerar evidências claras de execução conforme critérios de aceite.

3. Escopo

Dentro do Escopo:

Este plano de testes contempla a validação funcional e de regras de negócio dos seguintes módulos da API ServeRest:

OBS: As US inseridas no JIRA, são as mesmas definidas no Learning.

Type	Key	Resumo	Responsável	Prioridade	Status
	COMPB-3	US003 - [API] Gerenciamento de Produtos	Pedro Afonso de Alenc...	Medium	A FAZER
	COMPB-2	US002 - [API] Login	Pedro Afonso de Alenc...	Medium	A FAZER
	COMPB-1	US001 - [API] Cadastro de Usuários	Pedro Afonso de Alenc...	Medium	A FAZER

Sincronizado agora • 3 itens

- **Módulo Carrinhos** (Mesmo não contemplado nas User Stories);
- **Evidências de execução e testes candidatos à automação**, conforme critérios definidos no plano.

Fora do Escopo:

Os seguintes itens não serão validados neste plano de testes:

- **Testes exploratórios:** não incluídos nesta fase do plano de testes, por foco exclusivo em **testes roteirizados** baseados nas User Stories e Swagger.
- **Testes de carga, desempenho ou segurança** da API;
- Execução e validação em diferentes ambientes (como mobile ou frontend);
- **Infraestrutura técnica**, como banco de dados, escalabilidade, configuração de rede ou servidores;
- Interações externas ou simulações reais de fluxo completo de compra.

4. 🔎 Análise

A documentação técnica da API ServeRest, acessada via Swagger, descreve claramente os endpoints disponíveis, seus parâmetros, respostas esperadas e status HTTP. No entanto, ao analisar as User Stories do desafio, foi possível identificar regras de negócio e critérios que **não estão documentados no Swagger**, sendo essenciais para garantir a conformidade funcional e comportamental da aplicação.

Principais gaps identificados:

- **Restrições de e-mail:** O Swagger não menciona a proibição de e-mails dos domínios `gmail.com` e `hotmail.com`, mas essa regra aparece nos critérios de aceite da US001.
- **Validação de senhas:** O Swagger não impõe limites de 5 a 10 caracteres na senha, mas essa restrição é definida nas User Stories.
- **Comportamento do PUT em Usuários e Produtos:** Embora documentado parcialmente, as User Stories deixam claro que, se o ID não for encontrado, um novo cadastro deve ser feito — essa lógica não é explicitamente abordada no Swagger.
- **Autorização via token:** O Swagger apresenta o uso de token, mas não detalha o tempo de expiração (10 minutos), como descrito na US002.
- **Exclusão condicional de produtos:** A US003 define que não se pode excluir produtos vinculados a carrinhos, o que também não é detalhado tecnicamente no Swagger.
- **Módulo Carrinhos:** Está documentado no Swagger com rotas e respostas, mas **não está contemplado nas User Stories**.

Diante disso, a análise de testes será guiada tanto pelo Swagger quanto pelas User Stories, assegurando cobertura de requisitos **técnicos e de negócio**, inclusive para comportamentos não documentados.

5. 🛠️ Técnicas Aplicadas

- A abordagem de testes adotada neste plano foi baseada em **testes automatizados com Robot Framework**, com foco na validação das **regras de negócio descritas nas User Stories** e no comportamento técnico documentado no **Swagger da API ServeRest**.

Os testes seguiram **cenários roteirizados previamente (CT001 a CT024)**, sem execução de testes exploratórios livres.

📌 Técnicas de Teste Aplicadas

- **Caixa Preta:**
Todos os testes foram elaborados utilizando a técnica de caixa preta, **validando o comportamento da API com base nas entradas e saídas definidas nas User Stories** e documentação Swagger, sem acesso ao código-fonte da aplicação.
- **Particionamento de Equivalência (Parcial):**
A técnica foi aplicada **apenas aos valores previstos nos cenários planejados**, como e-mails válidos e inválidos definidos nas User Stories. **Não foram criadas classes adicionais de equivalência ou valores além do roteiro**.
- **Análise de Valor Limite (Parcial):**
A técnica foi aplicada **apenas aos testes de senhas**, com validação de limites mínimo (5) e máximo (10 caracteres), incluindo tentativas fora dos limites (4 e 11 caracteres). **Não foi estendida a outros parâmetros como preço, quantidade ou IDs**.
- **Cobertura de Códigos de Resposta HTTP (Parcial):**
Foram testados os status codes previstos em cada cenário planejado (ex: 200, 201, 400, 401). **Não foram criados testes adicionais para validar todos os status documentados no Swagger, nem cenários alternativos não previstos nas User Stories**.

🤖 Técnicas Planejadas para Automação:

- **Validação de Respostas:**
Validação automatizada de status codes, mensagens de retorno e estrutura do JSON com uso das `keywords` da `RequestsLibrary`.
- **Autenticação com Token:**
Execução de cenários com e sem autenticação, utilizando o token Bearer nos headers para simular perfis de usuário.

- **Fluxos Automatizados:**

Operações como cadastro, login, edição e exclusão de usuários e produtos foram roteirizadas com comandos automatizados, usando dados parametrizados.

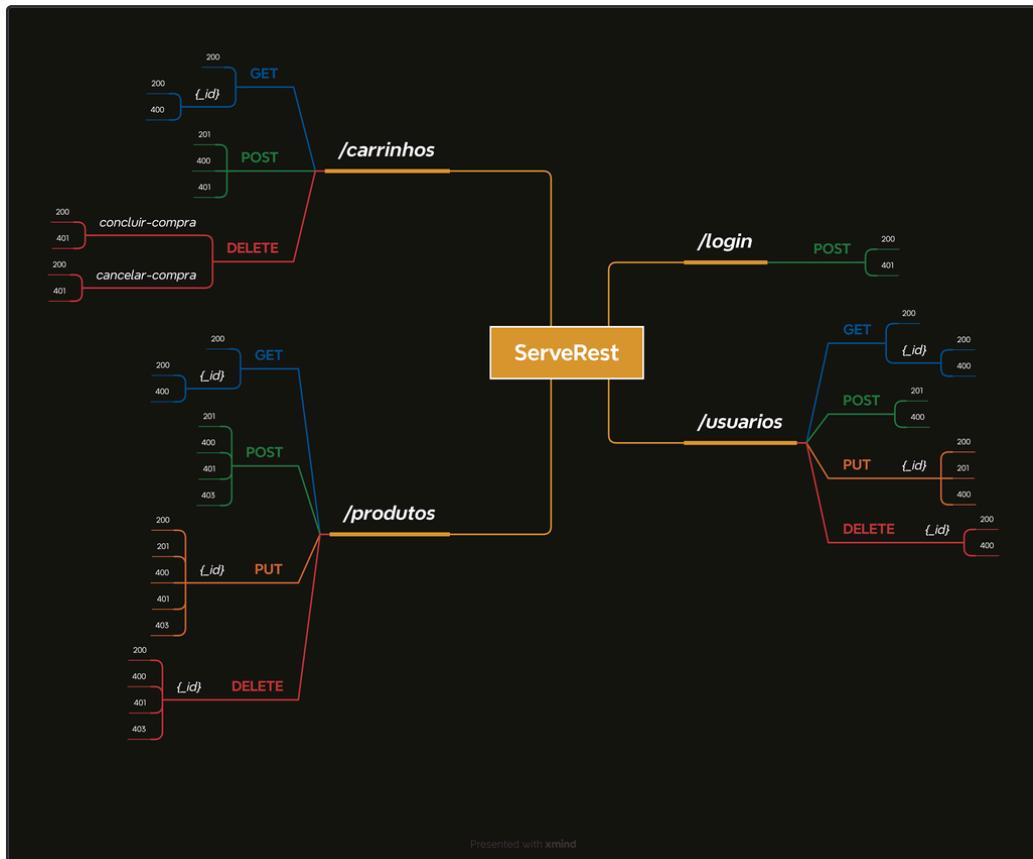
- **Massa de Dados Dinâmica:**

Testes com geração de dados randomizados e pré-configurados (como e-mails duplicados ou senhas inválidas), simulando variações negativas comuns.

- **Execução em Ambiente AWS:**

Toda a automação será executada em instância EC2 configurada com Robot Framework, com coleta dos logs (`log.html`, `report.html` e `output.xml`) ao final de cada execução via terminal.

6. Mapa Mental da API ↗



7. Cenários de Teste Planejados + Rastreabilidade ↗

Diponível no Jira: 24 Issues

ID	Funcionalida de	User Story	Cenário	Resultado Esperado	Result ado	Bug Relacionad o	Evidênci a
CT001	Usuários	US001	Criar usuário com dados válidos.	201 Created	Passou	—	
CT002	Usuários	US001	Criar usuário com e-mail já cadastrado.	400 E-mail já cadastrado	Passou	—	
CT003	Usuários	US001	Criar usuário com e-mail inválido (formato incorreto)	400 E-mail inválido	Passou	—	

CT004	Usuários	US001	Criar usuário com e-mail do domínio gmail.com e hotmail.com	400 E-mail não permitido	Falhou	1 Issues	
CT005	Usuários	US001	Criar usuário com senha de 4 caracteres	400 Senha fora dos limites	Falhou	1 Issues	
CT006	Usuários	US001	Criar usuário com senha de 11 caracteres	400 Senha fora dos limites	Falhou	1 Issues	
CT007	Usuários	US001	Editar usuário existente com dados válidos	200 Alterado com sucesso	Passou	—	
CT008	Usuários	US001	Editar usuário usando e-mail já existente	400 E-mail já cadastrado	Passou	—	
CT009	Login	US002	Login com dados válidos	200 OK + Token Bearer	Passou	—	
CT010	Login	US002	Login com senha incorreta	401 Unauthorized	Passou	—	
CT011	Login	US002	Login com usuário inexistente	401 Unauthorized	Passou	—	
CT012	Login	US002	Login com token expirado após 10 minutos	401 Token expirado	Passou	—	
CT013	Produtos	US003	Criar produto com token de admin	201 Created	Passou	—	
CT014	Produtos	US003	Criar produto sem token	401 Token ausente	Passou	—	
CT015	Produtos	US003	Criar produto com nome já existente	400 Nome duplicado	Passou	—	
CT016	Produtos	US003	PUT com ID inexistente (criação via PUT)	201 Created	Passou	—	
CT017	Produtos	US003	PUT com nome já existente	400 Nome duplicado	Passou	—	
CT018	Produtos	US003	Excluir produto vinculado a carrinho	400 Não é permitido excluir	Passou	—	
CT019	Carrinhos	-	Criar carrinho com token válido	201 Created	Passou	—	
CT020	Carrinhos	-	Criar carrinho com produto inexistente	400 Produto não encontrado	Passou	—	
CT021	Carrinhos	-	Criar carrinho com produto duplicado	400 Produto duplicado	Passou	—	

CT022	Carrinhos	-	Criar segundo carrinho para o mesmo usuário	400 Apenas um carrinho permitido	Passou	—	
CT023	Carrinhos	-	Cancelar carrinho (estoque deve ser devolvido)	200 Carrinho cancelado	Passou	—	
CT024	Carrinhos	-	Concluir compra (carrinho removido)	200 Carrinho excluído com sucesso	Passou	—	

8. Priorização dos Cenários

ID	Cenário	Prioridade	Justificativa
CT019	Criar carrinho com token válido	Alta	Funcionalidade principal de compra, diretamente ligada à receita.
CT023	Cancelar carrinho (estoque deve ser devolvido)	Alta	Impacta a experiência do usuário e a gestão de estoque, crucial para a disponibilidade de produtos.
CT024	Concluir compra (carrinho removido)	Alta	Finalização do fluxo de compra, essencial para a receita e satisfação do cliente.
CT001	Criar usuário com dados válidos	Alta	Fluxo essencial para novos usuários acessarem a plataforma.
CT009	Login com dados válidos	Alta	Funcionalidade básica para acesso de usuários existentes.
CT013	Criar produto com token de admin	Alta	Essencial para a gestão do catálogo de produtos por administradores.
CT002	Criar usuário com e-mail já existente	Média	Validação importante para integridade dos dados e experiência do usuário.
CT007	Editar usuário existente com dados válidos	Média	Permite ao usuário manter seus dados atualizados.
CT010	Login com senha incorreta	Média	Cenário comum, importante para a segurança e feedback ao usuário.
CT018	Excluir produto vinculado a carrinho	Média	Regra de negócio importante para evitar inconsistências em carrinhos ativos.
CT004	Criar usuário com e-mail do domínio gmail.com e hotmail.com	Baixa	Teste de regra de negócio específica (domínios não permitidos), menor impacto geral.
CT005	Criar usuário com senha de 4 caracteres	Baixa	Validação de limite de senha, importante, mas menos crítico que falhas de fluxo principal.

9. ! Matriz de Risco

Probabilidade	90%	Média	Média	<i>Cadastro com e-mail inválido ou duplicado</i>	Alta	Token expirado ou não gerado corretamente
	70%	Baixa	Média	<i>Cadastro com senha fora dos limites (muito curta ou longa)</i>	Alta	Alta
	50%	Baixa	Baixa	<i>Permitir nome duplicado + Cancelar carrinho sem devolver estoque</i>	Alta	Erro na finalização da compra (pagamento falha, carrinho não excluído)
	30%	Baixa	<i>Criar usuário com domínio de e-mail bloqueado (ex: gmail.com)</i>	Média	<i>Cadastro de produto sem autenticação</i>	Alta
	10%	Baixa	<i>Criar produto com preço negativo (validação)</i>	Baixa	Baixa	Média
		Muito Baixo	Baixo	Moderado	Alto	Muito Alto
Impacto						

10. ⚙️ Testes Candidatos à Automação

ID	Cenário	Justificativa
CT001	Criar usuário com dados válidos	Fluxo base de entrada no sistema, necessário para demais funcionalidades.
CT002	Criar usuário com e-mail já existente	Fluxo base de entrada no sistema, necessário para demais funcionalidades.
CT003	Criar usuário com e-mail inválido	Fluxo base de entrada no sistema, necessário para demais funcionalidades.
CT004	Criar usuário com e-mail do domínio gmail.com e hotmail.com	Fluxo base de entrada no sistema, necessário para demais funcionalidades.
CT005	Criar usuário com senha de 4 caracteres	Fluxo base de entrada no sistema, necessário para demais funcionalidades.
CT006	Criar usuário com senha de 11 caracteres	Fluxo base de entrada no sistema, necessário para demais funcionalidades.
CT007	Editar usuário existente com dados válidos	Manutenção de dados — importante mas não crítico.
CT008	Editar usuário usando e-mail já existente	Manutenção de dados — importante mas não crítico.
CT009	Login com dados válidos	Autenticação básica — fluxo essencial para acesso ao sistema.
CT010	Login com senha incorreta	Validação de segurança e feedback ao usuário.

CT011	Listar todos os usuários	Cenário de apoio à navegação — menor impacto direto na receita.
CT012	Atualizar usuário autenticado	Manutenção de dados — importante mas não crítico.
CT013	Criar produto com token de admin	Fluxo comum de cadastro de produtos — relevante para o catálogo.
CT014	Excluir usuário autenticado	Controle de remoção de dados — relevante para segurança e integridade.
CT015	Criar produto com nome já existente	Fluxo comum de cadastro de produtos — relevante para o catálogo.
CT016	Listar todos os produtos	Cenário de apoio à navegação — menor impacto direto na receita.
CT017	Buscar produto por ID	Cenário de apoio à navegação — menor impacto direto na receita.
CT018	Excluir produto vinculado a carrinho	Controle de remoção de dados — relevante para segurança e integridade.
CT019	Criar carrinho com token válido	Fluxo de compra, diretamente ligado à receita e integração entre módulos.
CT020	Adicionar produtos ao carrinho	Fluxo de compra, diretamente ligado à receita e integração entre módulos.
CT021	Cadastrar produto com sucesso	Fluxo comum de cadastro de produtos — relevante para o catálogo.
CT022	Cadastrar produto com preço negativo	Validação de regra técnica que evita falhas financeiras.
CT023	Cancelar carrinho (estoque deve ser devolvido)	Fluxo de compra, diretamente ligado à receita e integração entre módulos.
CT024	Concluir compra (carrinho removido)	Finalização do processo de venda — essencial para receita.

11. Cobertura de Testes

A cobertura de testes apresentada nesta seção foi calculada com base nos critérios definidos no artigo *Test Coverage Criteria for RESTful Web APIs*. Ela representa os **itens técnicos testados ao menos uma vez** durante a elaboração dos cenários e execução parcial da suíte.

Os valores não refletem ainda a **cobertura total de execução** (ex: todos os testes passaram/falharam), mas sim a verificação de que cada item foi contemplado **em pelo menos um cenário planejado ou automatizado**.

A cobertura de execução completa será apresentada separadamente no **Relatório do QAlity Plus - Test Management for Jira**.

A tabela abaixo resume os percentuais obtidos por tipo de critério técnico:

Critério	Total Documentado	Testado	Cobertura (%)	Observações

Path Coverage (input)	8 paths	8	100%	Todos os caminhos únicos testados (login, usuários, produtos, carrinhos).
Operator Coverage (input)	16 operações	16	100%	Todos os métodos GET, POST, PUT, DELETE testados ao menos uma vez.
Parameter Coverage (input)	18 parâmetros	14	78%	Todos parâmetros de Body testados; faltaram alguns parâmetros de query (ex: GET /usuarios).
Parameter Value Coverage (input)	12 valores esperados	6	50%	Testados apenas os valores previstos no roteiro; faltaram equivalência e limites adicionais.
Content-Type Coverage (input/output)	1 content-type	1	100%	Único content-type <code>application/json</code> testado; nenhum outro necessário/documentado.
Operation Flow Coverage (input)	4 fluxos possíveis	3	75%	Testados fluxos principais (POST → PUT, POST → DELETE, POST → GET); faltou alguma combinação.
Response Properties Body Coverage (output)	8 propriedades totais	4	50%	Validado apenas <code>message</code> , <code>authorization</code> ; faltaram validações de todos os campos retornados.
Status Code Coverage (output)	40 status codes totais	20	50%	Testados status esperados de cada cenário; faltaram erros alternativos e status documentados adicionais.

12. Artefatos Gerados

Os seguintes artefatos foram produzidos e fazem parte do desafio:

- **Plano de Testes** (este documento)
- **Relatório do QAlity Plus - Test Management for Jira:**

Test Cycles / Robot Framework + AWS

Robot Framework + AWS

Test Cases Overview

Add Test Cases to Test Cycle | All testcases | At assignee | Search for test case

Type	Test Case #	Status	Execution Assignee	Bugs	Actions
Q	COMPB-11 CT001 - Criar usuário com dados válidos	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-12 CT002 - Criar usuário com e-mail já existente	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-13 CT003 - Criar usuário com e-mail inválido (formato incorreto)	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-14 CT004 - Criar usuário com e-mail do domínio gmail.com e hotmail.com	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-15 CT005 - Criar usuário com senha de 4 caracteres	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-16 CT006 - Criar usuário com senha de 11 caracteres	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-17 CT007 - Editar usuário existente com dados válidos	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-18 CT008 - Editar usuário usando e-mail já existente	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-19 CT009 - Logon com dados válidos	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-20 CT010 - Logon com senha incorreta	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-21 CT011 - Logon com usuário inexistente	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-22 CT012 - Logon com token expirado após 10 minutos	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-23 CT013 - Criar produto com token de admin	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-24 CT014 - Criar produto sem token	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-25 CT015 - Criar produto com nome já existente	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-26 CT016 - PUT com ID inexistente (criação via PUT)	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-27 CT017 - PUT com nome já existente	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-28 CT018 - Excluir produto vinculado a carrinho	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-29 CT019 - Criar carrinho com token válido	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-30 CT020 - Criar carrinho com produto inexistente	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-31 CT021 - Criar carrinho com produto duplicado	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-32 CT022 - Criar segundo carrinho para o mesmo usuário	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-33 CT023 - Cancelar carrinho (estoque deve ser devolvido)	Unexecuted	Pedro Alfonso de ...	No issues found	
Q	COMPB-34 CT024 - Concluir compra (carrinho removido)	Unexecuted	Pedro Alfonso de ...	No issues found	

Test Cycles / Robot Framework + AWS

Robot Framework + AWS

Test Cases Overview

Project: Compass UOL PB

Version: No version

Description

Add description

Created: 30/May/2025

Due date: 02/Jun/2025

24 Test Cases in the Test Cycle

PASSSED	0.00% (0)
FAILED	0.00% (0)
BLOCKED	0.00% (0)
IN PROGRESS	0.00% (0)
UNRESOLVED	100.00% (24)

- Collection Postman com os cenários mapeados:

Swagger / User Stories

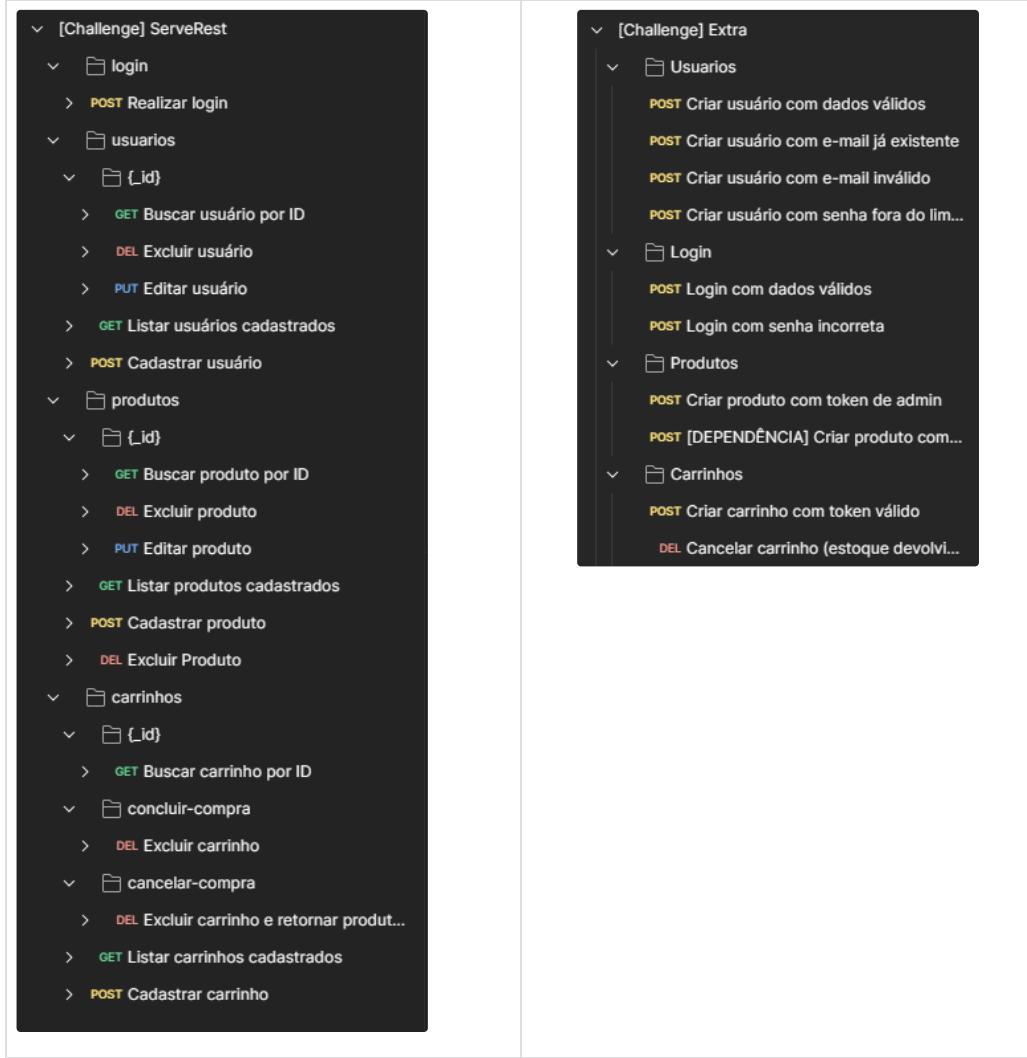
Automatização

[Challenge] ServeRest.postman_collection.json

[Challenge] Extra.postman_collection.json

[Challenge] Extra

[Challenge] ServeRest



13. ? Considerações sobre Itens Omitidos 🔗

Este plano foi adaptado para atender ao escopo e às exigências específicas do desafio proposto pela API ServeRest. Alguns tópicos tradicionalmente presentes em planos de teste mais robustos foram omitidos ou simplificados por não se aplicarem ao contexto deste projeto, conforme exemplos:

- **Glossário:** os termos utilizados são amplamente conhecidos no contexto de testes de APIs REST.
- **Infraestrutura e Responsabilidades da Equipe:** a infraestrutura já está fornecida e a execução será feita individualmente.
- **Fluxo de Trabalho, Comunicação e Cronograma:** não há múltiplos stakeholders, equipes envolvidas ou marcos formais no desafio.
- **Aprovações e Referências formais:** não há exigência de validações corporativas neste projeto prático.

Essa simplificação foi feita visando manter o plano de testes **focado, objetivo e funcional**, atendendo aos **entregáveis obrigatórios do desafio** e evitando complexidade desnecessária.