Planejamento de Testes - Módulo Carrinho

1. Apresentação

Este documento detalha o planejamento de testes para a funcionalidade de criação de Carrinhos de Compras da API ServeRest, conforme especificado na User Story US 004. O plano se concentra em validar as regras de negócio críticas, como a gestão de estoque, a unicidade de carrinho por usuário e a necessidade de autenticação para realizar a operação.

2. Objetivo

O objetivo principal é garantir que a funcionalidade de cadastro de carrinhos seja robusta, segura e funcional. Os testes visam validar que apenas usuários autenticados possam criar carrinhos, que as validações de produto e estoque funcionem corretamente e que a criação de um carrinho tenha o impacto esperado no inventário de produtos.

3. Escopo

Não pode faltar:	• Testar o endpoint de cadastro de carrinhos (POST /carrinhos). • Validar a camada de autorização (necessidade de token Bearer). • Testar todas as regras de negócio: unicidade de carrinho por usuário, produto existente, estoque suficiente e produto duplicado na requisição. • Validar o efeito colateral: verificar se o estoque do produto é decrementado corretamente após o sucesso da operação. • Gerenciar todos os testes e execuções no QALity (Jira). • Automatizar os cenários de regressão com Robot Framework. É bom ter (Escopo secundário)
É bom ter:	• Testes de carga no endpoint de cadastro de carrinho para verificar a concorrência no acesso ao estoque.

Fora do escopo:	• Testes das demais operações do CRUD de Carrinhos (GET, DELETE, etc.). • Testes dos módulos de Usuários, Login e Produtos (são pré-condições). • Testes do fluxo de finalização de compra (checkout).

4. Análise

A estratégia de teste será dividida em fases para garantir uma abordagem estruturada e eficiente:

Fase 1: Testes Manuais e Exploratórios

Ferramenta: Postman.

Objetivo: Realizar a primeira rodada de execução baseada nos casos de teste definidos. O foco é a validação funcional inicial, a descoberta de defeitos óbvios e a compreensão aprofundada do comportamento da API, indo além da documentação (Swagger).

Fase 2: Refinamento e Desenvolvimento da Automação

Ferramenta: Robot Framework.

Objetivo: Após a validação manual, os testes candidatos à automação serão desenvolvidos. Esta fase foca em criar uma suíte de testes robusta que possa ser executada rapidamente para garantir a regressão.

Fase 3: Execução Automatizada e Relatórios

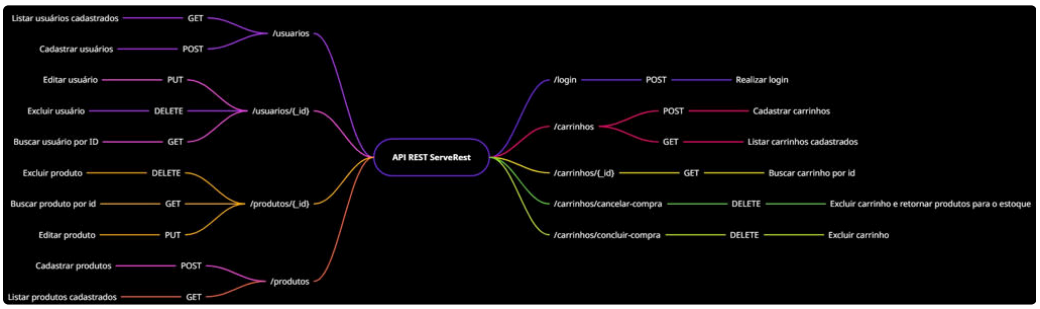
Ferramenta: Robot Framework, QALity, EC2.

Objetivo: Executar a suíte de testes automatizados a partir da instância EC2 dedicada, garantindo que novas alterações não introduziram defeitos. Os resultados serão reportados e gerenciados no QALity.

5. Técnicas aplicadas

- Testes Baseados em Requisitos: Mapeamento direto dos Critérios de Aceitação para os casos de teste.
- Teste de Transição de Estado: Validação da mudança de estado do estoque de um produto após a criação de um carrinho.
- Teste Baseado em Cenários: Simulação de fluxos realistas do usuário, como tentar adicionar um produto sem estoque ou criar um segundo carrinho.

6. Mapa mental da aplicação



7. Cenários de Testes

- Cenário: Criação de Carrinho de Compras - US004-CE1

ID do Caso de Teste	Descrição	Pré-Condições	Dados de Entrada (POST /carrinhos)	Resultado Esperado	Prioridade
US004-CE1-CT1	Criar um carrinho com sucesso	Usuário autenticado (sem carrinho); Produto com estoque > 0.	Token válido, corpo da requisição com produto e quantidade válidos.	Status 201 Created. Mensagem "Cadastro realizado com sucesso".	Alta
US004-CE1-CT2	Tentar criar carrinho sem autenticação	N/A	POST /carrinhos sem token de autorização.	Status 401 Unauthorized.	Alta
US004-CE1-CT3	Tentar criar um segundo carrinho para o mesmo usuário	Usuário autenticado já possui um carrinho.	Token válido do mesmo usuário.	Status 400 Bad Request. Mensagem "Não é permitido ter mais	Alta

				de 1 carrinho".	

8. Priorização da execução dos cenários de teste

A priorização da execução dos cenários de teste será baseada em uma combinação de fatores, incluindo a criticidade da funcionalidade, o risco associado, a frequência de uso esperada e a dependência de outros módulos. A prioridade de cada caso de teste já está indicada nas tabelas de cenários (Alta, Média, Baixa).

- Prioridade Alta: Casos de teste que cobrem funcionalidades críticas, cenários de alto risco (segurança, falhas que impedem o uso principal do sistema) e fluxos de usuário principais. Estes serão executados primeiro e com maior frequência.
- Prioridade Média: Casos de teste que cobrem funcionalidades importantes, cenários de risco moderado e fluxos alternativos. Serão executados após os de alta prioridade.
- Prioridade Baixa: Casos de teste que cobrem funcionalidades menos críticas, cenários de baixo risco ou casos de borda menos prováveis. Serão executados quando houver tempo e recursos disponíveis.

9. Matriz de Risco

Risco	Impacto	Probabilidade	Mitigação	Contigência
Inconsistência de estoque (vender produto indisponível)	Crítico	Média	• Testes automatizados rigorosos que validam o decremento do estoque (US004-CE1-CT2). • Testes de validação de	• Processo de reconciliação de inventário. • Cancelamento manual de compras com itens sem estoque e comunicaç

			estoque insuficiente (US004-CE1-CT6).	ão com o cliente.
Usuário consegue criar múltiplos carrinhos	Média	Baixa	• Testes específicos para a regra de negócio de unicidade de carrinho por usuário (US004-CE1-CT4).	• Script para limpeza de carrinhos duplicados no banco de dados, mantendo o mais recente.
Falha na autorização permite criação de carrinho anônimo	Alto	Baixa	• Cobertura de testes de segurança que tentam criar carrinhos sem um token válido (US004-CE1-CT3).	• Invalidação de sessões/tokens. • Análise de logs para identificar ações não autorizadas e limpeza de dados órfãos.
Falha na exclusão de usuário com carrinho associado	Média	Baixa	Testes de integridade referencial. Validação de regras de	Correção manual de dados.

			negócio na API.	

10. Cobertura de testes

- **Path Coverage (input)**

Testes Automatizados	1
Quantidade de Endpoints	1
Cobertura	100%

- **Operator Coverage (input)**

Quantidade de operações da API estão automatizados	1
Quantidade total de operações da API REST	1
Cobertura	100%

- **Parameter Coverage (input)**

Quantidade total de parâmetros cobertos na suítes de testes	3
Quantidade total de parâmetros nos métodos da API	3
Cobertura	100%

- **Content-Type Coverage (input e output)**

Quantidade total de content-type em cada operação cobertos pela suíte de testes	1
Quantidade total de content-type em todas as operações da API	1
Cobertura	100%

- **Response Properties Body Coverage (Output)**

Número total de todas as propriedades de todos os objetos que pode ser obtido na resposta da API	2
Número de propriedades da resposta que os testes estão cobrindo	2
Cobertura	100%

- **Status Code Coverage (Output)**

Status codes da API	3
Status codes cobertos na API	3
Cobertura	100%

11. Ferramentas, Ambiente e Infraestrutura

- Gerenciamento e Versionamento:
  - Jira com QALity: Para gestão completa dos casos de teste, planejamento de ciclos e reporte de defeitos.
  - Git: Para controle de versão de toda a documentação (Plano de Testes) e do código-fonte da automação.
- Ferramentas de Teste:
  - Postman: Utilizado para os testes manuais e exploratórios.
  - Robot Framework: Ferramenta principal para o desenvolvimento dos testes automatizados.
- Infraestrutura e Ambiente de Teste:
  - O ambiente será configurado em duas instâncias EC2 na AWS para simular um ambiente mais realista.
  - Configuração: Amazon Linux com Node.js e NPM para hospedar a API ServeRest.

12. Testes candidatos a automação

- US004-CE1-CT1 e CT2: Fluxo completo de sucesso, incluindo a validação do estoque (essencial para automação).
- US004-CE1-CT3: Validação de autorização.
- US004-CE1-CT4: Regra de negócio de unicidade de carrinho.

- US004-CE1-CT5: Validação de produto inexistente.
- US004-CE1-CT6: Validação de estoque insuficiente.

13. Cronograma

Atividade	Início	Término
Planejamento de Testes	8 de set. de 2025	9 de set. de 2025
Sessão de Testes	9 de set. de 2025	11 de set. de 2025
Relatório de Testes	12 de set. de 2025	12 de set. de 2025

14. Referências

- Test Coverage Criteria for RESTful Web APIs - Alberto Martin-Lopez, Sergio Segura e Antonio Ruiz-Cortés
- ISTQB Certified Tester Foundation Level (CTFL) 4.0 Syllabus
- ISO/IEC 25010:2011 – System and Software Quality Models
- ISO/IEC/IEEE 29119 – Software Testing
- IEEE 829 – Test Documentation
- Documentação da API ServeRest (Swagger)
- User Story: US 004 - [API] Carrinho