Planejamento de Teste - Módulo USUÁRIOS

1. Apresentação

Este documento apresenta o planejamento de testes aprimorado para o módulo de Usuários da API ServeRest. O objetivo é garantir a qualidade e a conformidade da aplicação com os requisitos da User Story US 001 - [API] Usuários. Este plano integra as diretrizes atualizadas de automação com Robot Framework, gerenciamento de testes via QALity no Jira, versionamento com Git e a utilização de uma infraestrutura em nuvem (AWS EC2) para os ambientes de teste e aplicação.

2. Objetivo

O objetivo principal é validar a qualidade e a conformidade do módulo de Usuários, identificando defeitos, reduzindo riscos e assegurando que a API atenda aos critérios de aceitação funcionais e não funcionais. A estratégia combina testes manuais exploratórios com uma suíte de testes automatizados para garantir agilidade e cobertura de regressão.

3. Escopo

Não pode faltar:	• Testar todos os endpoints (CRUD) do módulo de Usuários (CRIAR, ATUALIZAR, LISTAR, DELETAR). • Preparar massa de dados para cobrir os cenários de teste, incluindo a geração dinâmica de dados para a automação. • Realizar uma rodada de testes manuais exploratórios para validação inicial e descoberta de defeitos. • Desenvolver testes automatizados com Robot Framework para os cenários críticos e de regressão. • Gerenciar todo o ciclo de testes (casos de teste, execução, defeitos) no QALity dentro do Jira. • Versionar o plano de testes e o código de automação no Git.
É bom ter:	• Validações de responses com JSON Schema para garantir a estrutura dos dados.

	• Testes de desempenho (carga e estresse) para o módulo de Usuários, executados no ambiente EC2.
Fora do escopo:	• Testes de interface do usuário (frontend). • Testes de integração com sistemas externos não especificados. • Testes de usabilidade. • Testes de outros módulos da API ServeRest (Login, Produtos, Carrinhos) neste plano.

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.

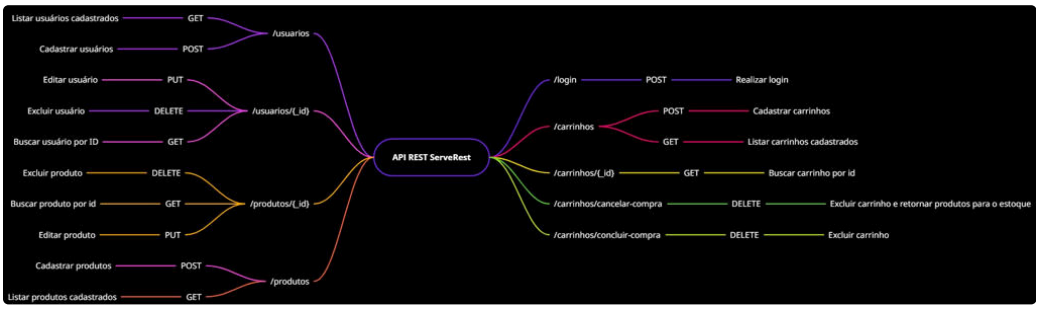
Objetivo: Executar a suíte de testes automatizados a partir da instância EC2 dedicada, garantindo que novas alterações não introduziram defeitos. Os resultados serão reportados e gerenciados no QALity.

5. Técnicas aplicadas

- Testes Baseados em Riscos: Priorização dos testes com base no impacto e probabilidade de falha, focando em funcionalidades críticas como criação de usuário e validações de segurança.

- Testes Exploratórios e Baseados em Sessão: Para complementar a cobertura dos casos de teste e descobrir defeitos não previstos.
- Técnicas de Design de Teste:
  - Particionamento de Equivalência: Para otimizar a seleção de dados de teste (ex: e-mails válidos/inválidos, provedores permitidos/não permitidos).
  - Análise de Valor Limite: Foco nos limites das regras de negócio (ex: senhas com 4, 5, 10 e 11 caracteres).
  - Teste de Tabela de Decisão: Para validar as regras complexas, como o comportamento do PUT para criar ou atualizar um usuário.

6. Mapa mental da aplicação



7. Cenários de Testes

- Cenário: Cadastro de Novo Usuário - US001-CE1

ID do Caso de Teste	Descrição	Dados de Entrada	Resultado Esperado	Prioridade
US001-CE1-CT1	Cadastrar usuário comum com dados válidos	Nome, E-mail (válido e não utilizado), Password (entre 5-10 chars), Administrador: false	Status 201 Created. Usuário cadastrado com sucesso.	Alta
US001-CE1-CT2	Cadastrar usuário administrador com dados válidos	Nome, E-mail (válido e não utilizado), Password (entre 5-10	Status 201 Created. Usuário cadastrado	Alta

		chars), Administrador: true	com sucesso.	
US001-CE1-CT3	Tentar cadastrar usuário com e-mail já utilizado	Nome, E-mail (já existente), Password, Administrador	Status 400 Bad Request. Mensagem de erro "Este email já está sendo usado".	Alta
US001-CE1-CT4	Tentar cadastrar usuário com e-mail de provedor não permitido (gmail)	E-mail: teste@gmail.com	Status 400 Bad Request. Mensagem de erro indicando provedor não permitido.	Alta
US001-CE1-CT5	Tentar cadastrar usuário com e-mail de provedor não permitido (hotmail)	E-mail: teste@hotmail.com	Status 400 Bad Request. Mensagem de erro indicando provedor não permitido.	Alta
US001-CE1-CT6	Tentar cadastrar usuário com senha menor que o limite (4 chars)	Password: "1234"	Status 400 Bad Request. Mensagem de erro sobre o tamanho da senha.	Média
US001-CE1-CT7	Tentar cadastrar usuário com senha maior	Password: "12345678901"	Status 400 Bad Request. Mensagem de erro sobre	Média

	que o limite (11 chars)		o tamanho da senha.	
US001-CE1-CT8	Tentar cadastrar usuário com campo nome vazio	Nome: ""	Status 400 Bad Request. Mensagem de erro indicando campo obrigatório.	Média
US001-CE1-CT9	Tentar cadastrar usuário com campo email vazio	Email: ““	Status 400 Bad Request. Mensagem de erro indicando campo obrigatório.	Média
US001-CE1-CT10	Tentar cadastrar usuário com campo password vazio	Password: ““	Status 400 Bad Request. Mensagem de erro indicando campo obrigatório.	Média
US001-CE1-CT11	Tentar cadastrar usuário com campo admnistrador vazio	Administrador: ““	Status 400 Bad Request. Mensagem de erro indicando campo obrigatório.	Média
US001-CE1-CT12	Tentar cadastrar usuário com campo administrado	Administrador: “string“	Status 400 Bad Request. Mensagem de erro indicando o	Média

	r não booleano		parâmetro não booleano.	
US001-CE1-CT13	Tentar cadastrar usuário com campo email sem dominio	Email: "teste2@"	Status 400 Bad Request. Mensagem de erro indicando email em formato incorreto.	Média
US001-CE1-CT14	Tentar cadastrar usuário com campo email sem .com.br	Email: "teste2@dominio"	Status 400 Bad Request. Mensagem de erro indicando email em formato incorreto.	Média
US001-CE1-CT15	Tentar cadastrar usuário com campo email sem @	Email: "teste2dominio.com.br"	Status 400 Bad Request. Mensagem de erro indicando email em formato incorreto.	Média

- Cenário: Listagem e Busca de Usuários - US001-CE2

ID do Caso de Teste	Descrição	Dados de Entrada	Resultado Esperado	Prioridade

US001-CE2-CT1	Listar todos os usuários cadastrados	N/A (GET /usuarios)	Status 200 OK. Lista de usuários retornada.	Alta
US001-CE2-CT2	Buscar usuário por ID válido	ID de um usuário existente	Status 200 OK. Dados do usuário retornados.	Alta
US001-CE2-CT3	Buscar usuário por ID inexistente	ID de um usuário não existente	Status 400 Bad Request. Mensagem "Usuário não encontrado".	Alta

- Cenário: Atualização de Usuário - US001-CE3

ID do Caso de Teste	Descrição	Dados de Entrada	Resultado Esperado	Prioridade
US001-CE3-CT1	Atualizar dados de um usuário existente com sucesso	ID existente, dados válidos para atualização	Status 200 OK. Mensagem "Registro alterado com sucesso".	Alta
US001-CE3-CT2	Tentar atualizar usuário para um e-mail já utilizado por outro	ID existente, E-mail de outro usuário	Status 400 Bad Request. Mensagem "Este email já está sendo usado".	Alta
US001-CE3-CT3	Criar novo usuário via PUT com um ID que não existe	ID inexistente, dados válidos para	Status 201 Created. Mensagem "Cadastro realizado	Alta

		um novo usuário	com sucesso".	
US001-CE3-CT4	Tentar criar novo usuário via PUT com e-mail já existente	ID inexistente, E-mail já utilizado	Status 400 Bad Request. Mensagem "Este email já está sendo usado".	Alta

- Cenário: Exclusão de Usuário - US001-CE4

ID do Caso de Teste	Descrição	Dados de Entrada	Resultado Esperado	Prioridade
US001-CE4-CT1	Excluir usuário existente com sucesso	ID de um usuário existente (sem carrinho)	Status 200 OK. Mensagem "Registro excluído com sucesso".	Alta
US001-CE4-CT2	Tentar excluir usuário inexistente	ID de um usuário não existente	Status 200 OK (com mensagem "Nenhum registro excluído"). Observação: validar comportamento esperado.	Alta
US001-CE4-CT3	Tentar excluir usuário com carrinho associado	ID de usuário com carrinho	Status 400 Bad Request. Mensagem "Não é permitido excluir	Alta

			usuário com 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
Falha no cadastro de usuário	Alta	Média	Testes abrangentes de validação de entrada e unicidade de e-mail. Automação de testes de regressão.	Rollback da versão da API. Correção de emergência.
Exposição de dados sensíveis de usuário	Alta	Baixa	Testes de segurança (injeção, autenticação /autorização)	Notificação de usuários afetados. Análise forense.

			. Criptografia de dados em trânsito e em repouso.	
Lentidão na listagem de usuários	Média	Média	Testes de desempenho (carga e estresse). Otimização de consultas ao banco de dados.	Otimização de infraestrutura (escalabilidade EC2).
Falha na exclusão de usuário com carrinho associado	Média	Baixa	Testes de integridade referencial. Validação de regras de negócio na API.	Correção manual de dados.

10. Cobertura de testes

- **Path Coverage (input)**

Testes Automatizados	5
Quantidade de Endpoints	5
Cobertura	100%

- **Operator Coverage (input)**

Quantidade de operações da API estão automatizados	5
Quantidade total de operações da API REST	5
Cobertura	100%

- **Parameter Coverage (input)**

Quantidade total de parâmetros cobertos na suítes de testes	5
Quantidade total de parâmetros nos métodos da API	5
Cobertura	100%

- **Parameter Value Coverage (input)**

Quantidade total de valores diferentes enviados	20
Quantidade total de valores que podem assumir.	20
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%

- **Operation Flow (input)**

Fluxos Possíveis	2
Fluxos Automatizados	2
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	6

Número de propriedades da resposta que os testes estão cobrindo	6
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

Com base na estratégia, os seguintes casos de teste são os principais candidatos para automação com Robot Framework após a validação manual, por representarem o fluxo principal (caminho feliz) e as validações de negócio mais críticas:

- Testes de Caminho Feliz:
  - US001-CE1-CT1: Cadastro de usuário comum.
  - US001-CE2-CT2: Listar usuários.
  - US001-CE2-CT3: Buscar usuário por ID.
  - US001-CE3-CT1: Atualizar usuário existente.

    - US001-CE4-CT1: Excluir usuário.
- Testes de Regras de Negócio Críticas:
    - US001-CE1-CT3: Tentativa de cadastro com e-mail duplicado.
    - US001-CE3-CT3: Criação de usuário via PUT (comportamento específico).
    - US001-CE4-CT3: Tentativa de exclusão de usuário com dependência (carrinho).

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
- User Story: US 001 - [API] Usuários