Planejamento de Testes - Módulo LOGIN

1. Apresentação

Este documento detalha o planejamento de testes para o módulo de Login da API ServeRest, com o objetivo de validar os requisitos da User Story US 002. O plano segue a metodologia e o ferramental estabelecidos anteriormente, incluindo o uso do QALity no Jira, automação com Robot Framework, versionamento no Git e a infraestrutura de testes em AWS EC2.

2. Objetivo

O objetivo principal é garantir que o processo de autenticação da API ServeRest seja seguro, confiável e esteja em conformidade com os requisitos. O foco é validar a geração de tokens de autorização para usuários válidos e o tratamento adequado de tentativas de login inválidas.

3. Escopo

Não pode faltar:	• Testar o endpoint de autenticação (/login). • Validar cenários de sucesso (credenciais válidas) e falha (credenciais inválidas, usuário inexistente). • Assegurar a correta geração do token Bearer em casos de sucesso. • Validar os códigos de status de resposta (200, 401). • Gerenciar todos os casos de teste e execuções no QALity (Jira). • Automatizar os cenários de regressão com Robot Framework.
É bom ter:	• Validar a estrutura do token gerado (JWT). • Testar a validade do token (expiração em 10 minutos).
Fora do escopo:	• Teste do módulo de cadastro de usuários (pré-condição). • Testes de endpoints que consomem o token de autenticação. • Testes de performance do endpoint de login.

	• Fluxo de "Esqueci minha senha".

4. Análise

A estratégia 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

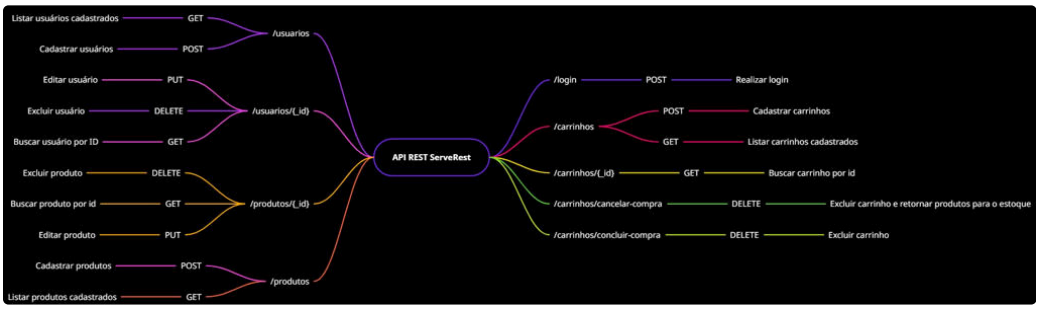
Ferramenta: Robot Framework, QALity, EC2.

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: Todos os casos de teste são diretamente derivados dos Critérios de Aceitação da US 002.
- Particionamento de Equivalência: Divisão das entradas em classes válidas (usuário cadastrado com senha correta) e inválidas (usuário não cadastrado, senha incorreta).
- Análise de Causa e Efeito: Análise das diferentes combinações de e-mail e senha e os resultados esperados (sucesso, falha 401, falha 400).

6. Mapa mental da aplicação



7. Cenários de Testes

- Cenário: Autenticação de Usuário - US002-CE1

ID do Caso de Teste	Descrição	Pré-Condições	Dados de Entrada (POST /login)	Resultado Esperado	Prioridade
US002-CE1-CT1	Autenticar com sucesso utilizando credenciais válidas	Usuário "fulano@qa.com" cadastrado com a senha "teste"	email: "fulano@qa.com", password: "teste"	Status 200 OK. Resposta contém um token de autorização Bearer não nulo.	Alta
US002-CE1-CT2	Tentar autenticar com senha inválida	Usuário "fulano@qa.com" cadastrado	email: "fulano@qa.com", password: "senhaerrada"	Status 401 Unauthorized. Mensagem de erro "Email e/ou senha inválidos".	Alta
US002-CE1-CT3	Tentar autenticar com um e-	E-mail "naoexiste@qa.com"	email: "naoexiste@qa.com",	Status 401 Unauthorized.	Alta

	mail não cadastrado	não está cadastrado	password: "qualquersenha"	Mensagem de erro "Email e/ou senha inválidos".	
US002-CE1-CT4	Tentar autenticar com o campo "email" vazio	N/A	email: "", password: "teste"	Status 400 Bad Request. Mensagem de erro "email não pode ficar em branco".	Média
US002-CE1-CT5	Tentar autenticar com o campo "password" vazio	N/A	email: "fulano@qa.com", password: ""	Status 400 Bad Request. Mensagem de erro "password não pode ficar em branco".	Média
US002-CE1-CT6	Validar tempo de expiração do token (10 minutos)	Token gerado no caso de teste CT1	1. Usar token em uma rota protegida (sucesso). 2. Aguardar 11 minutos.	1. Status 2xx. 3. Status 401 Unauthorized com mensagem de token	Média

			 3. Usar o mesmo token (falha).	expirado.	

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
Acesso não autorizado de usuários	Crítico	Baixa	• Testes rigorosos de cenários negativos (senha inválida, usuário inexistente, campos em branco).	• Invalidação imediata de todos os tokens de sessão ativos. • Forçar a redefinição de senha para todos os

			 • Implementação de mecanismos de segurança como proteção contra brute-force e hashing seguro de senhas. • Testes de segurança para tentar burlar a autenticação.	usuários. • Realizar uma análise forense para identificar a causa e o alcance da falha.
Falha na autenticação de usuários válidos	Alto	Média	• Cobertura completa do "caminho feliz" (login com sucesso) em testes manuais e automatizados. • Testes de regressão automatizados para	• Rollback da versão da API para uma versão estável anterior. • Aplicação de uma correção de emergência (hotfix). • Canal de

			garantir que novas alterações não quebrem a funcionalidade existente. • Monitoramento do serviço de autenticação em produção.	suporte para atender os usuários impactados.
Degradação de performance no login sob alta carga	Média	Média	• Realização de testes de desempenho (carga e estresse) no endpoint	/login. • Otimização das consultas ao banco de dados para busca de usuários. • Uso de estratégias de cache, se aplicável.
Geração de token inválido ou com permissões incorretas	Alto	Baixa	• Testes que validam não apenas a existência do token, mas também	• Invalidação de todos os tokens gerados. • Correção emergencial na

			sua estrutura e capacidade de ser usado em rotas protegidas. • Validação do tempo de expiração do token (Critério de Aceitação).	lógica de geração e assinatura do token.

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	2
Quantidade total de parâmetros nos métodos da API	2
Cobertura	100%

- **Parameter Value Coverage (input)**

Quantidade total de valores diferentes enviados	5
Quantidade total de valores que podem assumir.	5
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

Os seguintes casos são ideais para automação com Robot Framework, pois formam a base da suíte de regressão para a funcionalidade de login:

- US002-CE1-CT1: Autenticação com sucesso.
- US002-CE1-CT2: Tentativa com senha inválida.
- US002-CE1-CT3: Tentativa com e-mail não cadastrado.
- US002-CE1-CT4 e CT5: Validações de campos obrigatórios.

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 002 - [API] Login
- Documentação da API ServeRest (Swagger)