

SERVEREST

Plano de Testes Refinado – API ServeRest

Propósito:


O presente documento descreve o planejamento e a organização dos testes para a **API ServeRest**, com foco na verificação de requisitos funcionais, validação de regras de negócio, cobertura de endpoints, e na expansão da automação de testes utilizando Robot Framework.

Este plano é uma continuação do [Challenge](#) realizado anteriormente.

Objetivos dos Testes:

- Garantir a qualidade da aplicação ServeRest, validando se os comportamentos da API estão aderentes aos critérios de aceite e à documentação Swagger.
- Expandir a suíte de testes automatizados para incluir regressão de bugs identificados e cenários críticos.
- Refinar a estratégia de testes com base nos aprendizados e *issues* da fase anterior.
- Gerenciar e rastrear a execução dos testes e defeitos utilizando Jira com o aplicativo QALity.

Referências:

- Documentação da API ServeRest:  [ServeRest](#)
- Plano de Testes Anterior: [Challenge](#)
- Feedback da fase anterior sobre testes automatizados no Postman.

Escopo dos Testes

- **Funcionalidades/Endpoints no Escopo:**
 - **Autenticação:** `/login`
 - **Gestão de Usuários:** `/usuarios` (CRUD completo)
 - **Gestão de Produtos:** `/produtos` (CRUD completo, considerando autenticação)
 - **Gestão de Carrinhos:** `/carrinhos` (operações de carrinho, considerando autenticação)
- **Funcionalidades/Aspectos Fora do Escopo:**
 - Testes de Performance (conforme plano anterior).
 - Testes de Usabilidade da documentação da API (foco na API em si).

- **Tipos de Testes a Serem Executados:**

- **Testes Funcionais:** Validar as funcionalidades dos endpoints.
- **Testes de Regressão:** Assegurar que as correções de bugs e novas implementações não reintroduziram defeitos.
- **Testes de Validação:** Verificar regras de negócio e validações de entrada (e-mails, senhas, dados obrigatórios etc.).
- **Testes de Segurança (Foco em Autenticação/Autorização):** Verificar o manuseio de tokens, acesso a rotas protegidas e expiração de tokens.
- **Testes Positivos, Negativos e Alternativos:** Conforme definido no plano anterior.

Estratégia de Testes Refinada

- **Abordagem Geral:**

- Combinação de testes manuais exploratórios e baseados em cenários (gerenciados via QALity/Jira).
- Automação de testes com Robot Framework para cenários de regressão, fluxos críticos e validações chave.

- **Foco da Rodada Atual de Testes:**

- **Automação de Regressão para Bugs Identificados:** Prioridade máxima para automatizar os cenários que resultaram nos BUG01, BUG02, BUG03, e BUG04 (CT03, CT04, CT05, CT08).
- **Validações de Entrada Mais Robustas:** Expandir os testes (manuais e automatizados) para cobrir uma gama maior de dados inválidos e de limite, especialmente para os campos que apresentaram falhas (e-mail, senha).
 - **Testes de Segurança da Autenticação:** Investigar a fundo o ciclo de vida do token de autenticação, incluindo testes específicos para a issue de não expiração (BUG04).
 - **Aumento da Cobertura de Respostas da API:** Dedicar atenção para criar cenários que validem diferentes códigos de status e mensagens de erro, visando melhorar a cobertura de respostas que estava em 73,7%.
 - **Reflexão sobre Feedback dos Testes Postman:** Os testes automatizados no Postman foram identificados como simples e com tendência a passar. Para a automação com Robot Framework, o foco será em criar asserções mais detalhadas e variadas, cobrindo não apenas o "caminho feliz", mas também cenários negativos e de contorno para uma validação mais profunda.

- **Critérios de Aceite Gerais para os Testes:**

- Todos os casos de teste planejados (manuais e automatizados) executados.
- Todos os bugs críticos e de alta prioridade corrigidos e retestados com sucesso.
- Cobertura de automação atingindo X% dos cenários críticos e de regressão definidos.
- Relatórios de teste gerados e aprovados.

Ambiente, Ferramentas e Recursos

- **Ambiente de Teste:** API ServeRest (URL base:  [ServeRest](#) (ou instância EC2 conforme Fase 4 do guia, se der tempo)

- **Ferramentas:**

- **Postman:** Para testes exploratórios manuais e validação rápida de endpoints.
- **Robot Framework (com RequestsLibrary, JSONLibrary):** Para automação dos testes de API.

- **Jira:** Para gerenciamento do projeto e rastreamento de issues.
- **QALity:** Para gerenciamento de casos de teste, planejamento de ciclos e execução de testes manuais.
- **GitHub:** Para controle de versão do plano de testes, scripts de automação e outros artefatos.
- **(Talvez) AWS EC2:** Para deploy da aplicação e/ou do ambiente de execução de testes.

Organização e Gerenciamento de Testes (Jira/QALity)

• Criação e Mapeamento de Casos de Teste no QALity:

- Os cenários de teste existentes CT01-CT17 e TA01-TA05 serão recriados ou mapeados como Casos de Teste no QALity.
- Novos casos de teste identificados nesta fase também serão documentados no QALity.

• Organização em Ciclos de Teste:

- Serão criados Ciclos de Teste específicos no QALity para agrupar os testes por funcionalidade (ex: Testes de Usuários, Testes de Login).

• Registro de Defeitos:

- Novos defeitos encontrados serão registrados como *issues* no Jira, vinculados aos respectivos Casos de Teste no QALity.
- Os *issues* já identificados (BUG01 a BUG04) serão gerenciados e terão seus status atualizados no Jira conforme as correções e retestes.

Cenários de Teste

• Cenários de Teste Manuais (Revisão e Execução via QALity):

- Os cenários CT01 a CT17 e TA01 a TA05 do plano de testes original formam a base.
- Estes serão revisados, atualizados se necessário, e executados manualmente com o suporte do QALity para registro de resultados.
- Foco especial nos cenários que ainda não foram automatizados ou que exigem exploração mais aprofundada.

• Cenários de Teste a Serem Automatizados com Robot Framework:

- A seleção de candidatos à automação será refinada com base na criticidade, frequência de execução e complexidade.

Prioridade 1: Testes de Regressão para Bugs Identificados (baseados nos CTs com status ISSUE e BUGs):

- RF_CT03_BUG01_Validacao_Email_Dominio : Tenta criar usuário com e-mail Gmail ou Hotmail, esperando a falha (ou validando o comportamento atual do bug).
- RF_CT04_BUG02_Validacao_Senha_Maior_10 : Tenta criar usuário com senha >10 caracteres, esperando a falha.
- RF_CT05_BUG03_Validacao_Senha_Menor_5 : Tenta criar usuário com senha <5 caracteres, esperando a falha.
- RF_CT08_BUG04_Expiracao_Token : Tenta realizar login ou acessar recurso protegido após o tempo esperado de expiração do token.

Prioridade 2: Fluxos Críticos (Caminho Feliz):

- RF_CT01_Criar_Usuario_Valido
- RF_CT06_Login_Valido
- RF_CT09_Criar_Produto_Autenticado

- RF_CT16_Criar_Carrinho_Valido

Prioridade 3: Outras Validações e Cenários Negativos Chave:

- RF_CT02_Criar_Usuario_Email_Duplicado
- RF_CT07_Login_Senha_Invalida (validar o comportamento esperado, seja 401 ou o atual do bug).
- RF_CT10_Criar_Produto_Nome_Duplicado
- RF_CT11_Excluir_Produto_Em_Carrinho
- RF_CT12_Criar_Produto_Nao_Autenticado
- RF_CT17_Criar_Multiplos_Carrinhos_Mesmo_Usuario

• Estrutura da Suíte de Automação Robot Framework:

◦ Suítes por Endpoint/Funcionalidade:

- usuarios.robot
- login.robot
- produtos.robot
- carrinhos.robot

◦ Keywords Reutilizáveis:

- common_api_keywords.robot (para setup de sessão, obtenção de token de autenticação, validações de status code e content-type padrão, etc.).

◦ Variáveis:

- api_environment_variables.py (para URLs base, credenciais de teste sensíveis se não gerenciadas de outra forma).
- Variáveis de suíte/teste para dados de teste comuns.

◦ Tradução das Asserções do Postman:

- As verificações existentes no Postman (tempo de resposta, Content-Type, códigos de status) serão traduzidas e expandidas em keywords e asserções do Robot Framework (ex: Response Should Have Status , Should Be Equal As Strings , Dictionary Should Contain Key , Get Value From Json).

Execução dos Testes [↗](#)

• Ordem de Execução:

1. Execução de testes manuais exploratórios para novas funcionalidades ou áreas de risco.
2. Execução dos ciclos de teste manuais planejados no QALity.
3. Execução da suíte de testes automatizados do Robot Framework como parte do ciclo de regressão, especialmente após correções de bugs ou novos deploys.

• Registro de Resultados:

- **Testes Manuais:** Resultados registrados diretamente no QALity.
- **Testes Automatizados:** Relatórios gerados pelo Robot Framework (log.html , report.html) serão arquivados.

Relatório de Issues (Bugs) [↗](#)

- **Issues Existentes:** Os issues BU601 a BU604 continuarão sendo rastreados no Jira. Casos de teste automatizados específicos serão criados para verificar sua correção.

- **Novos Bugs:** Qualquer novo defeito identificado durante os testes manuais ou automatizados será reportado como um novo *issue* no Jira, com informações detalhadas.

Cobertura de Testes (Reavaliação)

- A cobertura de testes será reavaliada ao final desta fase, com foco em:
 - **Cobertura de Requisitos:** Garantir que todos os requisitos e user stories definidos tenham casos de teste associados.
 - **Cobertura de Endpoints e Operações:** Manter 100% de cobertura para os paths e operações em escopo.
 - **Cobertura de Parâmetros:** Manter 100% de cobertura dos parâmetros definidos para os endpoints.
 - **Cobertura de Respostas:** Esforço para aumentar a cobertura de diferentes tipos de resposta, buscando superar os 73,7% anteriores.
 - **Cobertura de Automação:** Medir a porcentagem de cenários críticos e de regressão cobertos por testes automatizados.

Entregáveis

Este Plano de Testes Refinado e Organização (documento atualizado).

- Casos de Teste atualizados e registrados no QALity/Jira.
- Suíte de testes automatizados com Robot Framework (código-fonte).
- Relatórios de execução de testes (manuais e automatizados).
- Relatório de bugs atualizado no Jira.
- Relatório de Cobertura de Testes ao final da fase.

Checklist para me guiar

Fase 0: Preparação e Alinhamento

- ☒ Revisar o "Challenge" anterior e o feedback recebido, especialmente sobre os testes Postman.
- ☒ Garantir que a documentação da API ServeRest (Swagger) e os critérios de aceite estão acessíveis e compreendidos.
- ☒ Confirmar a lista de endpoints e funcionalidades NO ESCOPO (Login, Usuários, Produtos, Carrinhos).
- ☒ Confirmar a lista de funcionalidades e aspectos FORA DO ESCOPO (Performance, Usabilidade da documentação).
- ☒ Incorporar aprendizados da fase anterior na estratégia de automação (foco em asserções mais detalhadas e variadas no Robot).

Fase 1: Ambiente e Ferramentas

- ☒ Configurar e validar o ambiente de teste (URL da API ServeRest: `https://serverest.dev/` ou instância EC2, se aplicável).
- ☐ Garantir acesso e configuração das ferramentas:
 - Postman (para exploração e validação rápida).
 - Robot Framework (com `RequestsLibrary`, `JSONLibrary`).
 - Jira (para issues).
 - QALity (para casos de teste manuais e planejamento de ciclos).
 - GitHub (para versionamento do plano, scripts e artefatos).

Fase 2: Gerenciamento de Casos de Teste e Defeitos (Jira/QALity)

☐ Casos de Teste Manuais no QALity:

- Mapear/Recrutar os cenários de teste manuais existentes (CT01-CT17, TA01-TA05) no QALity.
- Revisar e atualizar estes cenários no QALity conforme necessário.
- Documentar NOVOS casos de teste manuais identificados durante a fase no QALity.

☐ Ciclos de Teste no QALity:

- Criar e organizar os Casos de Teste em Ciclos de Teste específicos no QALity (ex: "Testes de Usuários", "Testes de Login", "Regressão de Bugs").

☐ Gerenciamento de Defeitos no Jira:

- Garantir que os BUGs existentes (BUG01 a BUG04) estão corretamente registrados e gerenciados no Jira.
- Preparar o processo para registrar NOVOS defeitos no Jira, vinculando-os aos Casos de Teste no QALity quando aplicável.

Fase 3: Desenvolvimento da Automação com Robot Framework

☐ Estrutura da Suíte de Automação:

- Definir a estrutura de pastas e arquivos da suíte Robot Framework.
- Criar arquivos de suíte por endpoint/funcionalidade (`usuarios.robot` , `login.robot` , `produtos.robot` , `carrinhos.robot`).
- Criar arquivo para keywords reutilizáveis (`common_api_keywords.robot`).
- Implementar gerenciamento de variáveis (ex: `api_environment_variables.py` para URLs base, ou variáveis de suíte/teste).

☐ Desenvolvimento de Keywords Reutilizáveis:

- Desenvolver keyword para setup de sessão/obtenção de token.
- Desenvolver keywords para validações comuns (status code, content-type, estrutura JSON básica).

☐ Desenvolvimento de Scripts de Teste Automatizados (Prioridades):

• **Prioridade 1 (Regressão de Bugs):**

- ☐ RF_CT03_BUG01_Validacao_Email_Dominio
- ☐ RF_CT04_BUG02_Validacao_Senha_Maior_10
- ☐ RF_CT05_BUG03_Validacao_Senha_Menor_5
- ☐ RF_CT08_BUG04_Expiracao_Token (investigar e validar o comportamento do token)

• **Prioridade 2 (Fluxos Críticos - Caminho Feliz):**

- ☐ RF_CT01_Criar_Usuario_Valido
- ☐ RF_CT06_Login_Valido
- ☐ RF_CT09_Criar_Produto_Autenticado
- ☐ RF_CT16_Criar_Carrinho_Valido

• **Prioridade 3 (Outras Validações e Cenários Negativos Chave):**

- ☐ RF_CT02_Criar_Usuario_Email_Duplicado
- ☐ RF_CT07_Login_Senha_Invalida
- ☐ RF_CT10_Criar_Produto_Nome_Duplicado
- ☐ RF_CT11_Excluir_Produto_Em_Carrinho
- ☐ RF_CT12_Criar_Produto_Nao_Autenticado
- ☐ RF_CT17_Criar_Multiplos_Carrinhos_Mesmo_Usuario

☐ **Asserções e Validações Detalhadas:**

- Para cada script, traduzir e EXPANDIR as verificações do Postman, incluindo:
 - Validação de Status Code (`Response Should Have Status`).
 - Validação de Content-Type.
 - Validação de Tempo de Resposta (se relevante e dentro do escopo de testes funcionais).
 - Validação de chaves e valores específicos no JSON de resposta (`Dictionary Should Contain Key` , `Get Value From Json` + `Should Be Equal As Strings/Numbers` , etc.).
 - Validação de mensagens de erro específicas para cenários negativos.

☐ **Cobertura de Respostas da API:**

- Dedicar atenção especial para criar cenários que validem diferentes códigos de status (2xx, 4xx, 5xx) e mensagens de erro da API.

Fase 4: Testes Manuais Exploratórios e Baseados em Cenário

- ☐ Planejar sessões de testes manuais exploratórios para áreas de maior risco ou novas funcionalidades (se houver).
- ☐ Preparar-se para executar os ciclos de teste manuais planejados no QALity.
- ☐ Foco em validações de entrada robustas e regras de negócio complexas que podem ser difíceis de automatizar completamente ou que se beneficiam da intuição humana.

Fase 5: Revisão e Documentação Contínua

- ☐ Manter o Plano de Testes atualizado conforme o desenvolvimento avança e novas decisões são tomadas.
- ☐ Versionar todos os artefatos de teste (plano, scripts, etc.) no GitHub.
- ☐ Documentar quaisquer decisões de design ou desvios do plano original.