

Planejamento de testes - API ServeRest

Apresentação 🔗

1 Apresentação | 2 Objetivo | 3 Escopo: | 4 Análise: | 5 Técnicas aplicadas | 6 Mapa Mental da aplicação | 7 Cenários de testes | 7.1 Usuário | 7.2 Login | 7.3 Produtos | 7.4 Carrinho | 8 Matriz de risco | 9 Cobertura de testes | 10 Reporte dos Bugs | 11 Testes candidatos para automação | 12 Cronograma

Objetivo @

Tem como objetivo validar o funcionamento das rotas da API ServeRest, garantindo que os respectivos fluxos estejam de acordo com as User Stories US001, US002 e US003 e as regras de negócio e garantir a qualidade da aplicação antes da entrega.

Escopo: @

- Testes com base nas User Stories, nas regras de negócio e no Swagger: As funcionalidades de Usuários, login e produtos
- Testes exploratórios e com base no Swagger: Carrinho

Análise: @

Baseado nos critérios de aceite das user stories e no Swagger:

US 001: [API] Usuários

Get - Testar a busca de um usuário com ID inexistente → erro;

Delete - Testar deletar um usuário com ID inexistente → erro;

Testar deletar um usuário que possui carrinho → erro;

Post - Testar a criação de um usuário com email já utilizado, ou sendo gmail ou hotmail → erro;

Testar criar uma senha com menos de 5 caracteres e maiores que 10 → erro;

Put - Testar atualizar um usuário com ID inexistente → Deve criar um novo.

US 002: [API] Login

Post - Testar um usuário não cadastrado tentando se autenticar → erro;

Testar um usuário cadastrado porém com senha inválida → erro;

Testar quanto tempo dura o token de autenticação, do qual tem que durar 10 minutos;

US 003: [API] Produtos

- Se um usuário que não é cadastrado tenta adicionar, atualizar, buscar ou excluir algum produto ightarrow erro

Get - Testar buscar um produto por um ID inválido → erro;

Post - O cadastro de um novo produto com um nome já existente → erro;

Delete - Ao tentar excluir um produto, porém ele está no carrinho → erro;

Put - Ao informar um ID inexistente de um produto, um novo deve ser criado;

O produto criado não deve ter o mesmo nome de algum produto já cadastrado.

[API] Carrinho

Get - Buscar carrinho por ID inválido;

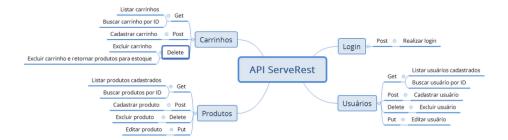
Post - Testar cadastrar mais de um carrinho por usuário;

Delete - Testar a exclusão de um carrinho com produtos nele;

Técnicas aplicadas @

- Caixa preta: os testes foram feitos com base nas entradas e saídas da API, sem olhar o código por trás.
- Particionamento de equivalência: usei para testar e-mails e senhas com dados corretos e incorretos.
- Valor-limite: apliquei para verificar se o sistema aceita senhas no tamanho mínimo e máximo permitido.
- Baseado em requisitos: criei os testes seguindo o que foi pedido nas User Stories e critérios de aceitação.
- Baseado no Swagger: usei a documentação da API para conferir se os métodos, rotas e respostas estavam corretos.
- Testes negativos: incluí casos de erro, como tentar logar com usuário não cadastrado.
- Exploratórios: fiz alguns testes livres para tentar encontrar falhas que não estavam nos requisitos.
- Teste End-to-End (E2E): testei alguns fluxos completos da API, como fazer login com usuário válido, buscar produtos para colocar no carrinho, cadastrar produtos no carrinho, para garantir que tudo funcione em sequência.

Mapa Mental da aplicação 🖉



Cenários de testes @

Usuário @

· CRUD de cadastro de vendedores.

ID	Cenário	Pré- requisitos	Método	Passos	Prioridade de teste	Resultado Esperado
CT-001	Criar usuário válido	E-mail ainda não cadastrado	POST	Enviar nome, e- mail, senha (5-10 caracteres), como adm	Alta	201 Created + ID
CT-002	Criar usuário com e-mail já usado	Criar um usuário com e-mail antes	POST	Enviar mesmo e- mail	Alta	400 + mensagem de e-mail duplicado
CT-003	Criar usuário com e-mail gmail/hotm ail	Nenhum	POST	Enviar e- mail @gmail.com ou @hotmail.c	Média	400 + mensagem de domínio não permitido
CT-004	Criar usuário com senha < 5 ou > 10	Nenhum	POST	Enviar senha com 4 ou 11 caracteres	Alta	400 + mensagem de senha inválida
CT-005	Buscar usuário com ID inválido	ID inexistente	GET	GET /usuarios/ {id}	Média	+ mensagem de não encontrado
CT-006	Atualizar usuário com ID inexistente	ID não cadastrado	PUT	Enviar PUT com dados + ID inválido	Baixa	201 Created (novo usuário criado)
CT-007	Deletar usuário com ID inexistente	ID não cadastrado	DELETE	Enviar DELETE /usuarios/ {id}	Baixa	200 OK + msg de não encontrado ou erro

Login €

ID	Cenário	Pré- requisitos	Método	Passos	Prioridade de teste	Resultado Esperado
CT-008	Login com usuário válido	Usuário já criado	POST	Enviar e- mail e senha corretos	Alta	200 OK + token válido

CT-009	Login com usuário não cadastrado	E-mail não existe	POST	Enviar e- mail inexistente	Alta	401 Unauthorize d
CT-010	Login com senha incorreta	Usuário criado com senha diferente	POST	Enviar senha incorreta	Alta	401 Unauthorize d
CT-011	Verificar validade do token (10 min)	Token gerado após login	POST	Aguardar 10 minutos e usar o token	Média	Token expira após 10 min

Produtos *𝒞*

ID	Cenário	Pré- requisitos	Método	Passos	Prioridade de teste	Resultado Esperado
CT-012	Cadastrar produto com nome único	Usuário autenticado	POST	Enviar nome novo + detalhes	Alta	201 Created
CT-013	Cadastrar produto com nome já usado	Produto com mesmo nome já criado	POST	Enviar mesmo nome	Alta	400 + mensagem de nome duplicado
CT-014	Ação em produto sem autenticaçã o	Não gerar token	POST	Tentar criar/editar/ deletar sem token	Alta	401 Unauthorize d
CT-015	Buscar produto com ID inválido	Produto com ID não existente	GET	GET /produtos/ {id}	Alta	+ mensagem de não encontrado
CT-016	Deletar produto que está no carrinho	Produto usado em carrinho	DELETE	DELETE /produtos/ {id}	Alta	400 ou erro com dependênci a
CT-017	Atualizar produto com ID inexistente	ID inválido	PUT	PUT /produtos/ {id} com novos dados	Média	Criar novo produto

CT-018	Atualizar	Nome já	PUT	PUT com	Alta	400 +
	produto	usado por		nome		mensagem
	com nome	outro		duplicado		de nome
	já existente	produto				duplicado

Carrinho €

ID	Cenário	Pré- requisitos	Método	Passos	Prioridade de teste	Resultado Esperado
CT-019	Criar mais de um carrinho por usuário	Usuário já com carrinho	POST	Enviar novo carrinho para o mesmo usuário	Baixa	400 ou erro de duplicidade
CT-020	Buscar carrinho com ID inválido	ID de carrinho não existe	GET	GET /carrinhos /{id}	Média	404 Not Found
CT-021	Excluir carrinho (cancelar compras) com produtos	Carrinho não vazio	DELETE	DELETE /carrinhos /{id}	Média	Deve remover, e os produtos retornarem para o estoque

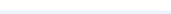
Matriz de risco 🛭

ID	Cenário	Probabilidade	Impacto	Nível de Risco	Justificativa
CT-001	Criar usuário válido	Alta	Alta	Crítico	Funcionalidad e essencial para o uso do sistema
CT-002	Criar usuário com e-mail já usado	Alta	Alta	Crítico	Viola a regra de unicidade, causa falhas no fluxo de cadastro
CT-003	Criar usuário com e-mail gmail/hotmail	Alta	Média	Alto	Gera erro por domínio inválido, impacto no cadastro
CT-004	Criar usuário com senha inválida	Alta	Média	Alto	Pode permitir cadastro inadequado

					ou falhar sem aviso
CT-005	Buscar usuário com ID inválido	Média	Média	Médio	Não impede uso do sistema, mas deve retornar mensagem adequada
CT-006	Atualizar usuário com ID inexistente	Baixa	Média	Baixo	Causa comportament o incomum, mas não crítico
CT-007	Deletar usuário com ID inexistente	Baixa	Média	Baixo	Não impacta negativamente diretamente
CT-008	Login com usuário válido	Alta	Alta	Crítico	Garante acesso às demais funcionalidade s
CT-009	Login com usuário não cadastrado	Alta	Alta	Crítico	Segurança e bloqueio de acessos indevidos
CT-010	Login com senha incorreta	Alta	Alta	Crítico	Segurança e validação de credenciais
CT-011	Verificar validade do token	Média	Média	Médio	Importante para segurança e controle de sessão
CT-012	Cadastrar produto com nome único	Alta	Alta	Crítico	Essencial para operação de vendas
CT-013	Cadastrar produto com nome já usado	Alta	Alta	Crítico	Pode gerar duplicidade e confusão nos estoques
CT-014	Ação em produto sem autenticação	Alta	Alta	Crítico	Segurança total comprometida

CT-015	Buscar produto com ID inválido	Média	Média	Médio	Precisa retornar erro claro, sem falhas no sistema
CT-016	Deletar produto que está no carrinho	Alta	Alta	Crítico	Pode afetar fluxo de vendas e integridade do carrinho
CT-017	Atualizar produto com ID inexistente	Baixa	Média	Ваіхо	Apenas gera criação ao invés de update; menos crítico
CT-018	Atualizar produto com nome já existente	Alta	Alta	Crítico	Viola regra de nome único, quebra integridade dos dados
CT-019	Criar mais de um carrinho por usuário	Baixa	Média	Baixo	Não afeta gravemente o sistema, mas é um caso negativo
CT-020	Buscar carrinho com ID inválido	Média	Média	Médio	Não bloqueia operação, mas precisa retorno adequado
CT-021	Excluir carrinho com produtos	Média	Alta	Alto	Afeta estoque e fluxo de compras

Cobertura de testes 🕝



Operações cobertas:

• POST /usuarios (CT-001, CT-002, CT-003, CT-004)

Operation coverage = (Operações testadas / Operações totais) * 100

- GET /usuarios/{id} (CT-005)
- PUT /usuarios/{id} (CT-006)
- DELETE /usuarios/{id} (CT-007)
- POST /login (CT-008, CT-009, CT-010, CT-011)
- POST /produtos (CT-012, CT-013, CT-014)
- GET /produtos/{id} (CT-015)

- DELETE /produtos/{id} (CT-016)
- PUT /produtos/{id} (CT-017, CT-018)
- POST /carrinhos (CT-019)
- GET /carrinhos/{id} (CT-020)
- DELETE /carrinhos/{id} (CT-021)

Resultado:

Operation Coverage = (12 / 12) * 100 = 100%

Reporte dos Bugs @



Testes candidatos para automação ${\mathscr O}$

ID	Cenário	Motivo para automação
CT-001	Criar usuário válido	Alta repetição e tem verificação com facilidade
CT-002	Criar usuário com e-mail já usado	Validação crítica de regra de negócio
CT-004	Criar usuário com senha inválida	Validação de valor-limite
CT-008	Login com usuário válido	Fluxo essencial e base para outros testes
CT-009	Login com usuário não cadastrado	Validação de segurança
CT-012	Cadastrar produto com nome único	Base para testes de carrinho

CT-014	Ação sem autenticação	Cobertura de segurança
CT-019	Criar mais de um carrinho por usuário	Fluxo que pode ser facilmente repetido
CT-021	Excluir carrinho com produtos	Valida comportamento de fluxo completo

Cronograma 🖉

