Casos de Teste

Software: Fake Store API

QA responsável: Elisama dos Santos Gregorio

---

1. Introdução

Este documento tem como objetivo apresentar a análise de qualidade da API FakeStore, fornecendo uma visão detalhada dos principais casos de teste aplicáveis aos seus endpoints, bem como antecipar riscos e validações essenciais. A proposta é assegurar a confiabilidade da aplicação desde os estágios iniciais, pensando não apenas no backend, mas também em uma futura integração com o frontend.

O foco do trabalho está na cobertura dos seguintes pontos:

- Casos de teste positivos e negativos
- Validações de campos obrigatórios e tipos de dados
- Validações de autenticação e permissões
- Considerações de performance e resposta
- Identificação de riscos relacionados à segurança e consistência de dados

2. Resumo da API FakeStore

A FakeStore API é uma API REST que simula um ambiente de e-commerce, com operações típicas como visualização de produtos, gerenciamento de carrinhos, login de usuários e autenticação básica.

Principais recursos:

- `/products`: Lista de produtos disponíveis na loja
- `/carts`: Operações com carrinhos de compra
- `/users`: Gerenciamento de usuários
- `/auth/login`: Login e geração de token

A API não requer autenticação para a maioria dos endpoints, com exceção do login que simula uma autenticação JWT simples.

3. Casos de Teste: Produtos (`/products`)

ID	Cenário	Tipo	Descrição	Validações
TC001	Listar todos os produtos	Positivo	Verifica se a API retorna uma lista de produtos corretamente	Status 200, corpo é array, contém campos como id, title, price, category
TC002	Buscar produto por ID existente	Positivo	Verifica se a API retorna corretamente os dados de um produto válido (ex: ID 1)	Status 200, id corresponde, dados completos
TC003	Buscar produto por ID inexistente	Negativo	Verifica comportamento ao buscar um ID que não existe (ex: ID 9999)	Status 404 ou objeto vazio, validação de mensagem
TC004	Buscar produto com ID inválido (string ou especial)	Negativo	Testa o endpoint com valores inválidos no ID (ex: "abc", "@$")	Status 400 ou tratamento adequado, mensagem clara

TC005	Criar produto com método não permitido	Negativo	Tenta fazer um POST em /products que é somente leitura	Status 405, mensagem indicando método não permitido
TC006	Verificar tipo de dados dos campos	Validação	Confirma que campos retornam com os tipos corretos: price como número, title como string, etc	Tipos corretos para todos os campos esperados
TC007	Performance - Tempo de resposta	Performance	Mede tempo de resposta da listagem completa dos produtos	< 2 segundos (exemplo de SLA)

4. Casos de Teste: Carrinho (/carts)

ID	Cenário	Tipo	Descrição	Validações
TC008	Listar todos os carrinhos	Positivo	Verifica se a API retorna a lista completa de carrinhos existentes	Status 200, corpo é array, contém campos como id, userId, date, products
TC009	Buscar carrinho por ID existente	Positivo	Verifica se é possível recuperar os dados de um carrinho específico	Status 200, id correto, lista de produtos com productId e quantity
TC010	Criar carrinho com dados válidos	Positivo	Simula um POST com userId e lista de produtos	Status 200 ou 201, confirmação de criação, ID gerado
TC011	Criar carrinho com payload inválido	Negativo	Tenta criar carrinho com dados ausentes ou malformados	Status 400, validação de campos obrigatórios
TC012	Atualizar carrinho com método PUT	Positivo	Verifica se é possível atualizar produtos no carrinho	Status 200, carrinho atualizado corretamente
TC013	Atualizar carrinho inexistente	Negativo	Envia PUT para um ID de carrinho que não existe	Status 404 ou erro coerente

TC014	Verificação de tipos de dados	Validação	Confirma que os campos retornados têm os tipos corretos (products como array, productId como inteiro, etc.)	Tipos corretos e esperados
TC015	Consistência de dados e segurança	Segurança	Verifica se não há possibilidade de manipulação incorreta (ex: adicionar produto inválido)	Restrições aplicadas, consistência nos dados criados

5. Casos de Teste: Usuários (`/users`)

ID	Cenário	Tipo	Descrição	Validações
TC016	Listar todos os usuários	Positivo	Verifica se a API retorna todos os usuários cadastrados	Status 200, corpo é array, contém campos como id, email, name, address
TC017	Criar usuário com dados válidos	Positivo	Envia um POST com nome, email, senha, endereço	Status 200 ou 201, id gerado, dados retornados corretamente
TC018	Criar usuário com campos invalidos	Negativo	Tenta criar usuário sem nome, email ou senha	Status 400, validação de obrigatoriedade
TC019	Verificar tipos de dados	Validação	Confirma tipos corretos em campos como email (string), address (objeto), zipcode (string)	Tipos consistentes
TC020	Verificar campos sensíveis expostos	Segurança	Garante que senhas ou tokens não estejam visíveis na resposta	Nenhuma exposição indevida

6. Casos de Teste: Login (`/auth/login`)

ID	Cenário	Tipo	Descrição	Validações

TC021	Login com credenciais válidas	Positivo	Envia usuário e senha válidos para login	Status 200, corpo contém token válido (string)
TC022	Login com usuário inválido	Negativo	Envia login com nome de usuário inexistente	Status 401 ou 403, mensagem de erro coerente
TC023	Verificar formato do token retornado	Validação	Verifica se o token retornado é string e segue o padrão JWT (3 partes separadas por ".")	Formato correto, token presente
TC024	Performance no login	Performance	Mede o tempo de resposta do login com credenciais válidas	< 1 segundo

7. Validações Técnicas Transversais

Validação	**Descrição**	**Endpoints Impactados**
Campos obrigatórios	Validação de presença de todos os campos essenciais (ex: email, password, productId) nas requisições	/users, /auth/login, /carts
Tipo de dado	Garantir que o tipo dos dados retornados e enviados esteja coerente com o esperado (ex: price como número, email como string)	Todos
Status HTTP adequado	Validação de que os códigos retornados (200, 400, 401, 404, 405) são apropriados para o cenário	Todos
Mensagens de erro claras	Erros devem ter mensagens compreensíveis e informativas, sem expor estrutura interna do sistema	Todos
Tempo de resposta (Performance)	Respostas devem estar dentro de um SLA razoável para garantir boa experiência (ex: < 2s)	Todos

Tratamento de entradas inválidas	Entradas com campos malformados, tipos errados, ou dados de ataque devem ser tratadas com respostas seguras	Todos

8. Riscos Identificados

Risco	Descrição	Impacto	Ação de Mitigação
Exposição de dados sensíveis	A API não deve retornar senhas, hashes ou dados pessoais além do necessário	Alto	Garantir mascaramento e exclusão de dados sensíveis nas respostas
Falta de autenticação em endpoints críticos	Endpoints como /carts e /users podem ser acessados sem token (risco de acesso indevido)	Alto	Implementar autenticação JWT obrigatória
Entrada maliciosa (SQL Injection / XSS)	Falta de validação pode abrir brechas de segurança	Alto	Aplicar sanitização e validação de entrada
Falta de limitação de requisições	Possível abuso por bots ou DoS	Médio	Implementar rate limiting
Inconsistência de dados entre entidades	Produtos no carrinho com IDs inválidos, usuários inexistentes etc.	Médio	Criar validação cruzada entre entidades relacionadas
Performance instável	Listagens grandes podem levar a lentidão sem paginação	Médio	Implementar paginação e cache

9. Conclusão

A análise da API FakeStore permitiu identificar pontos fortes e oportunidades de melhoria no que diz respeito à sua confiabilidade, segurança e preparação para integração com um frontend futuro.

Foram cobertos testes funcionais (positivos e negativos), validações de estrutura e tipos de dados, testes de autenticação, performance e segurança. Além disso, foram antecipados riscos relevantes como a exposição de dados sensíveis e a falta de autenticação em endpoints críticos.

Essa abordagem busca alinhar a qualidade desde o início do desenvolvimento, evitando retrabalho futuro e promovendo uma base mais sólida para escalabilidade. Os cenários levantados também servem como base para futuras automações de testes e monitoramentos em produção.

Próximos passos sugeridos:

- Implementar autenticação robusta em endpoints sensíveis.
- Automatizar os testes de regressão para os principais fluxos.
- Monitorar métricas de performance e segurança em tempo real.
- Adicionar paginação nos endpoints de listagem para garantir escalabilidade.