Desafio Técnico – Analista de Qualidade | Suzano

✅ **Entregas exigidas no desafio:**

- Mapeamento de casos de teste
- Aplicação de validações positivas, negativas, obrigatoriedade, tipo, integridade, segurança e performance
- Automação completa de ao menos um endpoint
- Antecipação de testes da interface frontend

✅ **Visão Geral das Funcionalidades**

Endpoint		Operações Cobertas
/auth/login		POST
/products		GET, POST, PUT, DELETE
/carts		GET, POST, PUT, DELETE
/users		GET, POST, PUT, DELETE

🧪 **Total de cenários de teste documentados**: **43**

Cada endpoint foi analisado com foco em:

- Cobertura funcional (fluxos de sucesso)
- Cobertura de exceções (erros esperados)
- Validações de integridade, tipo de dado e obrigatoriedade
- Segurança (XSS, SQL injection, dados sensíveis)
- Performance (tempo de resposta, carga leve)

📋 Casos de Teste Criados – API FakeStore

🔐 Endpoint: /auth/login – POST

ID	Cenário	Tipo	Esperado
01	Login com credenciais válidas	Positivo	200
02	Login com senha incorreta	Negativo	401
03	Login com usuário inexistente	Negativo	404
04	Login com campos em branco	Negativo	400
05	Login com tipos de dados inválidos	Negativo	400
06	Login com payload malformado	Negativo	400
07	Tentativa de SQL Injection	Segurança	401 ou 400
08	Verificar exposição de dados sensíveis	Segurança	Nenhum dado sensível
09	Tempo de resposta inferior a 500ms	Performance	SLA mantido

🛍️ Endpoint: /products

GET /products

ID	Cenário	Tipo	Esperado
10	Listar todos os produtos	Positivo	200 + array de produtos
11	Buscar produto por ID válido	Positivo	200 + produto correto
12	Buscar produto inexistente	Negativo	404
13	Buscar produto com ID inválido	Negativo	400

POST /products

ID	Cenário	Tipo	Esperado
14	Criar produto com dados válidos	Positivo	201
15	Criar sem campo obrigatório	Negativo	400
16	Criar com tipo incorreto	Negativo	400
17	Criar com valores inválidos	Negativo	400

PUT e DELETE /products/{id}

ID	Cenário	Tipo	Esperado
18	Atualizar produto com dados válidos	Positivo	200
19	Atualizar produto inexistente	Negativo	404
20	Deletar produto existente	Positivo	204
21	Deletar produto inexistente	Negativo	404

🛒 Endpoint: /carts

GET /carts

ID	Cenário	Tipo	Esperado
22	Listar todos os carrinhos	Positivo	200
23	Buscar carrinho por ID válido	Positivo	200
24	Buscar carrinho com ID inválido	Negativo	400

POST /carts

ID	Cenário	Tipo	Esperado
25	Criar carrinho com dados válidos	Positivo	201
26	Criar carrinho sem userId	Negativo	400
27	Criar com products vazio ou inválido	Negativo	400
28	Criar com data malformada	Negativo	400

PUT e DELETE /carts/{id}

ID	Cenário	Tipo	Esperado
29	Atualizar carrinho válido	Positivo	200
30	Atualizar carrinho inexistente	Negativo	404
31	Deletar carrinho válido	Positivo	204
32	Deletar carrinho inexistente	Negativo	404

🦍 Endpoint: /users

GET /users

ID	Cenário	Tipo	Esperado
33	Listar todos os usuários	Positivo	200
34	Buscar usuário por ID válido	Positivo	200
35	Buscar usuário com ID inválido	Negativo	400

POST /users

ID	Cenário	Tipo	Esperado
36	Criar usuário com dados válidos	Positivo	201
37	Criar sem campo obrigatório (username, password)	Negativo	400
38	Criar com tipos inválidos	Negativo	400
39	Criar com email malformado	Negativo	400

PUT e DELETE /users/{id}

ID	Cenário	Tipo	Esperado
40	Atualizar usuário existente	Positivo	200
41	Atualizar com dados inválidos	Negativo	400
42	Deletar usuário existente	Positivo	204
43	Deletar usuário inexistente	Negativo	404

🛡️ Validações (Segurança e Performance)

🔐 Validações de Segurança

🔒 Autenticação e Autorização

- Implementar proteção contra ataques de força bruta (limitação de tentativas de login)
- Verificar expiração adequada de tokens JWT
- Implementar autorização baseada em papéis (RBAC) para endpoints sensíveis
- Testar a revogação de tokens após logout

🔐 Proteção de Dados

- Verificar criptografia de dados em trânsito (TLS 1.3)
- Validar criptografia de dados em bases sensíveis

- Garantir mascaramento de dados sensíveis nos logs (ex: senha, CPF, e-mail)

🛡️ Headers de Segurança

- Verificar presença e configuração dos headers:
    - Content-Security-Policy
    - X-XSS-Protection
    - Strict-Transport-Security
- Verificar configuração segura de CORS (origens confiáveis, métodos permitidos)

✋ Validação de Entrada

- Validação de tipos de dados em todos os parâmetros
- Validação de limites de tamanho para uploads de arquivos
- Sanitização de entrada para evitar injeção de código (XSS, SQL Injection)

📈 Auditoria e Logging

- Logs devem registrar eventos de segurança: login, alterações de permissões, falhas
- Implementar rastreamento de comportamento

⚙️ Validações de Performance Adicionais

🔄 Testes de Carga

- Carga sustentada: 100 requisições/segundo durante 30 minutos
- Pico de tráfego: 500 requisições/segundo por 1 minuto
- Teste de recuperação: simular sobrecarga e avaliar recuperação

📊 Monitoramento de Recursos

- Avaliar uso de CPU e memória durante carga
- Medir tempo de resposta de operações no banco de dados
- Validar eficácia do cache (memória/disco)

Estratégia de Teste

A estratégia de testes adotada neste desafio foi dividida entre uma abordagem com foco em modularidade, clareza, reuso e escalabilidade. Todas as validações foram construídas com base em análise de contrato da API e critérios de boas práticas de qualidade.

🔍 Casos de Testes

- Casos criados com base nas operações disponíveis em cada endpoint (GET, POST, PUT, DELETE)
- Divisão clara entre casos positivos, negativos, de segurança e performance
- Organização orientada por recurso, com rastreabilidade dos objetivos e respostas esperadas
- Consideração de limites de dados, combinações inválidas, tipos incorretos e falhas comuns
- Tabela de documentação por endpoint estruturada para facilitar leitura, execução e auditoria

 Testes Automatizados (Cypress)

- Endpoint automatizado: **/products**.
- Estrutura modular
- Validações automatizadas de:
    - Status HTTP
    - Conteúdo do corpo da resposta
    - Validação de mensagens de erro e comportamento para entradas inválidas

 Integração Contínua (CI)

- Pipeline implementado via **GitHub Actions**:
    - Execução automatizada dos testes a cada push
    - Visibilidade dos resultados da automação diretamente no repositório
    - Validação automatizada do endpoint /products antes de deploys

 Arquitetura e Escalabilidade

- Separação de responsabilidades entre camadas do projeto
- Facilidade de expansão para novos endpoints com padrões replicáveis

- Base sólida para implementação futura de testes E2E com interface

🧠 Antecipação de Testes para o Futuro Frontend

Antecipação de Cenários de Interface Web

🔐 Fluxo de Autenticação

- Login/logout com diferentes tipos de usuários
- Recuperação de senha
- Registro de novos usuários
- Mensagens de erro claras para credenciais inválidas

🛍️ Catálogo de Produtos

- Listagem com paginação e filtros
- Visualização detalhada de produtos
- Busca com sugestões e autocomplete
- Exibição de produtos indisponíveis ou em promoção

🛒 Carrinho de Compras

- Adição e remoção de itens do carrinho
- Atualização de quantidades
- Cálculo de subtotais e totais
- Persistência do carrinho entre sessões

👤 Gestão de Usuários

- Edição de perfil
- Visualização de histórico de pedidos
- Gerenciamento de endereços
- Configurações de privacidade e notificações

✅ Validações Funcionais e de Usabilidade Sugeridas

⚙️ Validações Funcionais

Consistência de Dados

- Verificar se dados submetidos são adequadamente persistidos e recuperados
- Garantir que alterações em um componente se reflitam em outros relacionados
- Validar cálculos (subtotais, impostos, descontos,frete) entre frontend e backend

Gerenciamento de Estado

- Testar comportamento após refresh da página
- Verificar sincronização entre múltiplas abas abertas
- Garantir consistência entre dispositivos/sessões distintas

Responsividade a Ações

- Validar feedback visual e textual imediato após ações
- Testar comportamento durante operações longas (loading)
- Verificar tratamento de falhas de rede e timeouts

Compatibilidade

- Executar testes em diferentes navegadores modernos
- Verificar exibição e funcionalidade em variados tamanhos de tela
- Validar comportamento em conexões lentas ou instáveis

🎨 Validações de Usabilidade

Acessibilidade

- Conformidade com padrões WCAG
- Navegação completa via teclado
- Compatibilidade com leitores de tela (ex: NVDA, VoiceOver)
- Contraste de cores apropriado e legível

Facilidade de Uso

- Clareza nos fluxos de navegação
- Intuitividade nos formulários e interações
- Feedback visual e textual adequado para ações (confirmações, erros)
- Consistência visual e funcional entre diferentes telas

Performance Percebida

- Tempo de carregamento inicial otimizado
- Responsividade fluida durante interações
- Uso eficiente de imagens, assets e fontes
- Implementação de skeletons (carregamento visual progressivo)

Design Responsivo

- Adaptação fluida a diversos tamanhos e proporções de tela
- Compatibilidade com orientação retrato/paisagem
- Elementos touch-friendly em dispositivos móveis
- Verificação de comportamento com zoom ativado

💻 Abordagem e Considerações sobre a Tecnologia Escolhida

A escolha do Cypress para a automação foi motivada por diversos fatores estratégicos e técnicos:

- **Facilidade de uso e aprendizado:** a sintaxe simples e declarativa torna o framework acessível para criação rápida e eficaz de testes.
- **Integração com APIs:** Cypress permite chamadas diretas a endpoints REST
- **Ambiente de execução controlado:** facilita o isolamento dos testes e reprodutibilidade dos resultados.
- **Suporte à criação de comandos customizados:** essencial para aplicar reuso e padronização nas interações com a API.
- **Ecossistema maduro e bem documentado:** com suporte da comunidade, bibliotecas e plugins úteis.

A estrutura modular adotada permite escalar a automação facilmente para outros endpoints da FakeStoreAPI ou qualquer outro sistema. A separação clara entre comandos, fixtures e arquivos de teste favorece a manutenibilidade a longo prazo.

🧭 Dificuldades Encontradas e Como Foram Contornadas

Desafio	Solução Adotada
Documentação superficial em alguns pontos da FakeStoreAPI	Uso da documentação oficial + testes exploratórios
Respostas sem mensagens padronizadas (casos de erro)	Validação focada em status HTTP e estrutura mínima do payload
Ausência de frontend para simular jornada completa	Antecipação de testes de interface com base nos endpoints e regras observadas

📎 Link do Repositório

O repositório com toda a automação implementada no desafio técnico está disponível em:

📎 **GitHub:** https://github.com/phillipe77/suzano-qa-challenge