

Hands-on Lab: Automação de Testes - Sistema de Biblioteca

Neste laboratório prático, você irá aplicar seus conhecimentos de automação de testes para validar tanto o **frontend** quanto o **backend** de um sistema de gerenciamento de biblioteca. O objetivo é criar uma suíte completa de testes automatizados que garanta a qualidade e confiabilidade da aplicação.

Objetivos de Aprendizagem

- Automatizar casos de teste de API (backend)
- Automatizar casos de teste de interface (frontend)
- Implementar validações de dados e regras de negócio
- Praticar boas práticas de automação de testes
- Documentar e reportar resultados dos testes

Contexto da Aplicação

O sistema de biblioteca possui as seguintes funcionalidades:

- **Autenticação:** Registro e login de usuários
- **Gerenciamento de Livros:** CRUD completo (Criar, Ler, Atualizar, Deletar)
- **Dashboard:** Visualização de estatísticas
- **Favoritos:** Sistema de marcação de livros favoritos
- **Detalhes:** Visualização completa de informações dos livros

Casos de Teste - Backend (API)

CT-API-001: Registro de Novo Usuário (Sucesso)

Endpoint: POST /registro

Objetivo: Validar criação de novo usuário com dados válidos

Pré-condições: Email não cadastrado no sistema

Dados de Entrada:

json

```
{  
  "nome": "Maria Silva",  
  "email": "maria.silva@teste.com",  
  "senha": "senha123"  
}
```

Validações Esperadas:

- Status code: 201
 - Resposta contém: mensagem: "Usuário criado com sucesso"
 - Resposta contém objeto usuario com id, nome e email
 - Campo senha não deve estar presente na resposta
 - Campo id deve ser um número inteiro positivo
-

CT-API-002: Registro com Email Duplicado (Falha)

Endpoint: POST /registro

Objetivo: Validar que sistema impede registro com email já existente

Pré-condições: Email "admin@biblioteca.com" já cadastrado

Dados de Entrada:

```
json
{
  "nome": "João Santos",
  "email": "admin@biblioteca.com",
  "senha": "senha456"
}
```

Validações Esperadas:

- Status code: 400
 - Resposta contém: mensagem: "Email já cadastrado"
-

CT-API-003: Login com Credenciais Válidas

Endpoint: POST /login

Objetivo: Validar autenticação bem-sucedida

Dados de Entrada:

```
json
{
  "email": "admin@biblioteca.com",
  "senha": "123456"
```

}

Validações Esperadas:

- Status code: 200
 - Resposta contém: mensagem: "Login realizado com sucesso"
 - Resposta contém objeto usuario sem campo senha
 - Tempo de resposta inferior a 2 segundos
-

CT-API-004: Login com Credenciais Inválidas

Endpoint: POST /login

Objetivo: Validar rejeição de credenciais incorretas

Dados de Entrada:

```
json
{
  "email": "admin@biblioteca.com",
  "senha": "senhaerrada"
}
```

Validações Esperadas:

- Status code: 401
 - Resposta contém: mensagem: "Email ou senha incorretos"
-

CT-API-005: Listar Todos os Livros

Endpoint: GET /livros

Objetivo: Validar retorno da lista completa de livros

Validações Esperadas:

- Status code: 200
- Resposta é um array
- Cada livro contém campos: id, nome, autor, paginas, descricao, imageUrl, dataCadastro
- Campo paginas é número inteiro positivo
- Campo dataCadastro está no formato ISO 8601[dev](#)

CT-API-006: Buscar Livro por ID (Existente)

Endpoint: GET /livros/1

Objetivo: Validar retorno de livro específico

Validações Esperadas:

- Status code: 200
 - Resposta contém livro com id: 1
 - Todos os campos obrigatórios estão presentes
 - Campo nome não está vazio
-

CT-API-007: Buscar Livro por ID (Inexistente)

Endpoint: GET /livros/9999

Objetivo: Validar tratamento de ID não encontrado

Validações Esperadas:

- Status code: 404
 - Resposta contém: mensagem: "Livro não encontrado"
-

CT-API-008: Adicionar Novo Livro

Endpoint: POST /livros

Objetivo: Validar criação de novo livro

Dados de Entrada:

```
json
{
  "nome": "Código Limpo",
  "autor": "Robert C. Martin",
  "paginas": 425,
  "descricao": "Manual de boas práticas",
  "imagemUrl": "https://exemplo.com/imagem.jpg"
}
```

Validações Esperadas:

- Status code: 201
 - Resposta contém livro criado com id gerado
 - Campo dataCadastro foi preenchido automaticamente
 - Valores correspondem aos dados enviados
-

CT-API-009: Atualizar Livro Existente

Endpoint: PUT /livros/1

Objetivo: Validar atualização de dados do livro

Dados de Entrada:

json

```
{  
  "nome": "Clean Code - Edição Atualizada",  
  "autor": "Robert C. Martin",  
  "paginas": 464,  
  "descricao": "Guia completo atualizado",  
  "imagemUrl": "https://exemplo.com/nova-imagem.jpg"  
}
```

Validações Esperadas:

- Status code: 200
 - Resposta contém livro atualizado
 - Campo id permanece inalterado
 - Novos valores foram aplicados
-

CT-API-010: Deletar Livro

Endpoint: DELETE /livros/2

Objetivo: Validar remoção de livro

Validações Esperadas:

- Status code: 200
- Resposta contém: mensagem: "Livro removido com sucesso"

-
- Requisição GET /livros/2 subsequente retorna 404
-

CT-API-011: Obter Estatísticas da Biblioteca

Endpoint: GET /estatisticas

Objetivo: Validar cálculo de estatísticas

Validações Esperadas:

- Status code: 200
 - Resposta contém: totalLivros, totalPaginas, totalUsuarios
 - Todos os valores são números inteiros não-negativos
 - totalPaginas é a soma de páginas de todos os livros
-

CT-API-012: Adicionar Livro aos Favoritos

Endpoint: POST /favoritos

Objetivo: Validar funcionalidade de favoritar livro

Dados de Entrada:

```
json
{
  "usuarioid": 1,
  "livroid": 1
}
```

Validações Esperadas:

- Status code: 201
 - Resposta contém: mensagem: "Livro adicionado aos favoritos"
-

CT-API-013: Listar Favoritos de Usuário

Endpoint: GET /favoritos/1

Objetivo: Validar retorno de livros favoritos

Validações Esperadas:

- Status code: 200
- Resposta é um array de livros

- Apenas livros favoritados pelo usuário 1 são retornados
-

Casos de Teste - Frontend (Interface)

CT-FE-001: Fluxo Completo de Registro

Objetivo: Validar criação de conta de novo usuário

Passos:

1. Acessar página /registro.html
2. Preencher campo "Nome" com "Carlos Oliveira"
3. Preencher campo "Email" com "carlos@teste.com"
4. Preencher campo "Senha" com "senha123"
5. Preencher campo "Confirmar Senha" com "senha123"
6. Clicar no botão "Registrar"

Validações:

- Alert com mensagem "Conta criada com sucesso!" é exibido
 - Redirecionamento para página /login.html
 - Dados não ficam salvos no formulário após registro
-

CT-FE-002: Validação de Senhas Não Correspondentes

Objetivo: Validar mensagem de erro quando senhas não coincidem

Passos:

1. Acessar /registro.html
2. Preencher todos os campos
3. Preencher "Senha" com "senha123"
4. Preencher "Confirmar Senha" com "senha456"
5. Clicar em "Registrar"

Validações:

- Alert "As senhas não coincidem!" é exibido
- Usuário permanece na página de registro
- Nenhuma requisição API é enviada

CT-FE-003: Login com Sucesso

Objetivo: Validar fluxo de autenticação bem-sucedida

Passos:

1. Acessar /login.html
2. Preencher "Email" com "admin@biblioteca.com"
3. Preencher "Senha" com "123456"
4. Clicar em "Entrar"

Validações:

- Alert "Login realizado com sucesso!" é exibido
 - Redirecionamento para /dashboard.html
 - Nome do usuário aparece no header
 - Dados do usuário salvos no localStorage
-

CT-FE-004: Login com Credenciais Inválidas

Objetivo: Validar tratamento de erro de autenticação

Passos:

1. Acessar /login.html
2. Preencher com credenciais inválidas
3. Clicar em "Entrar"

Validações:

- Alert com mensagem de erro é exibido
 - Usuário permanece na tela de login
 - Campos não são limpos automaticamente
-

CT-FE-005: Verificar Proteção de Rotas

Objetivo: Validar que páginas protegidas exigem autenticação

Passos:

1. Limpar localStorage

2. Tentar acessar diretamente /dashboard.html

Validações:

- Redirecionamento automático para /login.html
 - Mensagem ou comportamento indicando necessidade de login
-

CT-FE-006: Visualizar Dashboard com Estatísticas

Objetivo: Validar carregamento correto do dashboard

Pré-condições: Usuário autenticado

Passos:

1. Acessar /dashboard.html

Validações:

- Cards de estatísticas são exibidos
 - Valores numéricos estão formatados corretamente
 - Grid de "Últimos Livros Adicionados" é carregado
 - Máximo de 5 livros recentes são exibidos
 - Cada card de livro contém imagem, nome e autor
-

CT-FE-007: Adicionar Novo Livro

Objetivo: Validar formulário de cadastro de livro

Pré-condições: Usuário autenticado

Passos:

1. Acessar /livros.html
2. Preencher "Nome do Livro" com "O Hobbit"
3. Preencher "Autor" com "J.R.R. Tolkien"
4. Preencher "Número de Páginas" com "310"
5. Preencher "Descrição" com texto descriptivo
6. Preencher "URL da Imagem" com URL válida
7. Clicar em "Adicionar Livro"

Validações:

- Alert "Livro adicionado com sucesso!" é exibido
 - Formulário é limpo após submissão
 - Novo livro aparece na lista de livros
 - Página de livros é recarregada automaticamente
-

CT-FE-008: Validação de Campos Obrigatórios

Objetivo: Validar que campos obrigatórios são verificados

Passos:

1. Acessar /livros.html
2. Tentar submeter formulário vazio
3. Tentar submeter com apenas alguns campos preenchidos

Validações:

- Mensagens de validação HTML5 são exibidas
 - Formulário não é submetido
 - Nenhuma requisição é enviada ao servidor
-

CT-FE-009: Navegação Entre Páginas

Objetivo: Validar funcionamento dos links de navegação

Pré-condições: Usuário autenticado

Passos:

1. Acessar /dashboard.html
2. Clicar em cada botão da navegação

Validações:

- Botão "Dashboard" redireciona para /dashboard.html
 - Botão "Gerenciar Livros" redireciona para /livros.html
 - Botão "Meus Favoritos" redireciona para /favoritos.html
 - Transições ocorrem sem erros de console
-

CT-FE-010: Visualizar Detalhes de Livro

Objetivo: Validar página de detalhes do livro

Pré-condições: Usuário autenticado, livro com ID 1 existe

Passos:

1. Acessar /livros.html
2. Clicar em um card de livro

Validações:

- Redirecionamento para /detalhes.html?id=1
 - Imagem do livro é exibida
 - Todos os campos (nome, autor, páginas, descrição, data) são exibidos
 - Botões de ação estão visíveis e funcionais
-

CT-FE-011: Adicionar Livro aos Favoritos

Objetivo: Validar funcionalidade de favoritar

Pré-condições: Usuário autenticado, na página de detalhes

Passos:

1. Acessar /detalhes.html?id=1
2. Clicar em "Adicionar aos Favoritos"

Validações:

- Alert "Adicionado aos favoritos!" é exibido
 - Botão muda para "Remover dos Favoritos"
 - Ícone ou cor do botão é alterado
 - Livro aparece em /favoritos.html
-

CT-FE-012: Remover Livro dos Favoritos

Objetivo: Validar remoção de favorito

Pré-condições: Livro já está nos favoritos

Passos:

1. Acessar /detalhes.html?id=1
2. Clicar em "Remover dos Favoritos"

Validações:

- Alert "Removido dos favoritos!" é exibido
 - Botão volta para "Adicionar aos Favoritos"
 - Livro não aparece mais em /favoritos.html
-

CT-FE-013: Listar Livros Favoritos

Objetivo: Validar página de favoritos

Pré-condições: Usuário tem pelo menos 1 livro favoritado

Passos:

1. Acessar /favoritos.html

Validações:

- Grid de livros favoritos é exibido
 - Apenas livros favoritados aparecem
 - Se não há favoritos, mensagem "Você ainda não tem livros favoritos" é exibida
-

CT-FE-014: Deletar Livro com Confirmação

Objetivo: Validar exclusão de livro

Pré-condições: Na página de detalhes do livro

Passos:

1. Clicar em botão "Deletar Livro"
2. Confirmar ação no dialog

Validações:

- Dialog de confirmação é exibido
 - Após confirmar, alert "Livro deletado com sucesso!" aparece
 - Redirecionamento para /livros.html
 - Livro não aparece mais na lista
-

CT-FE-015: Cancelar Deleção de Livro

Objetivo: Validar cancelamento de exclusão

Passos:

1. Clicar em "Deletar Livro"
2. Cancelar ação no dialog

Validações:

- Dialog é fechado
 - Nenhuma ação é executada
 - Usuário permanece na página de detalhes
 - Livro continua existindo
-

CT-FE-016: Logout do Sistema

Objetivo: Validar funcionalidade de sair

Pré-condições: Usuário autenticado

Passos:

1. Clicar no botão "Sair"

Validações:

- Dados do localStorage são removidos
 - Redirecionamento para /login.html
 - Tentativa de acessar páginas protegidas redireciona para login
-

Ferramentas Sugeridas

Backend (API Testing):

- **Postman** (exploração manual)
- **Rest Assured** (Java)
- **SuperTest** (JavaScript)
- **RestAssured** (JavaScript)
- **Cypress** (JavaScript)
- **Playwright** (JavaScript)

Frontend (UI Testing):

- **Selenium WebDriver** (Java)

- **Cypress** (JavaScript)
- **Playwright** (JavaScript)

Entregáveis

1. **Código dos testes automatizados** (backend e frontend)
2. **Relatório de execução** com evidências (screenshots, logs)
3. **README** com instruções de execução dos testes

Dicas para Sucesso

1. Comece pelos casos mais simples (happy path)
2. Use dados de teste consistentes e isolados
3. Implemente waits adequados (explícitos, não fixos)
4. Capture evidências em caso de falha
5. Organize testes em suítes lógicas
6. Execute testes localmente antes de subir
7. Documente bugs encontrados durante automação