Testes manuais API

Resumo da Execução

Item	Descrição
Projeto	API - Cinema Challenge
Período de Execução	26 e 27 de Junho de 2025
Versão da API	v1
Ambiente de Teste	`http://localhost:3000/api/v1` (Local)
Responsável pela Execução	Maria Eduarda Pedreira Correia
Tipo de Teste	Teste Funcional Exploratório e de Regressão
Ferramenta	Postman
Resultado Geral	**Reprovado (Bugs e equivocos na documentação)**
Status da Automação CI	**Falhando (Conforme Esperado)**

Este relatório detalha os resultados do ciclo de testes executado na API do projeto "Cinema Challenge". O objetivo foi realizar uma validação funcional geral, cobrindo os principais fluxos da aplicação. Embora parte significativa das funcionalidades opere corretamente, foram identificados **defeitos críticos e de alta prioridade**, incluindo inconsistências graves no tratamento de erros, que justificam o resultado **Reprovado** para este ciclo.

Escopo e Estratégia de Teste

O escopo do teste abrangeu os principais endpoints da API, focando na validação de funcionalidades de Autenticação, Usuários, Filmes e Sessões. A estratégia de teste adotada incluiu:

- **Testes de Caminho Feliz:** Validação dos fluxos principais e operações bem-sucedidas (CRUD).
- **Testes de Caminho Alternativo e Negativo:** Verificação de como a API responde a dados inválidos, IDs inexistentes e cenários de erro.
- **Testes de Autorização:** Checagem das regras de acesso, diferenciando as permissões entre perfis de `admin` e usuários comuns.

Métricas da Execução (Postman Runner)

Métrica	Quantidade
Total de Requisições Executadas	62
Total de Asserções (Testes)	114
Asserções com Sucesso	107
Asserções com Falha	**7**
Taxa de Sucesso das Asserções	93.86%

Observação: A taxa de sucesso isolada pode ser enganosa. As 7 falhas de asserção, quando investigadas, apontaram para os defeitos de maior impacto documentados abaixo.

Defeitos Encontrados

Foram identificados e reportados os seguintes defeitos.

ID	Endpoint	Resumo do Defeito	Criticidade
BUG-01	`PUT /auth/profile`	**(Erro 500)** A tentativa de alterar a senha do usuário retorna "Illegal arguments: string, undefined", impedindo uma funcionalidade crítica do perfil.	**Crítica**
BUG-02	`POST /sessions`	A API permite a criação de sessões conflitantes (mesmo filme, sala e horário), retornando `201 Created` em vez do esperado `409 Conflict`.	**Alta**
BUG-03	`GET /users`	A resposta da listagem de usuários não inclui o objeto `pagination`, divergindo da	**Média**

		especificação no Swagger e quebrando contratos com consumidores da API.	
BUG-04	`PUT /auth/profile`	Ao usar um token de usuário que foi deletado, a API retorna `403 Forbidden` em vez de `404 Not Found`, dificultando a identificação da causa do erro.	**Média**
BUG-05	`DELETE /sessions/{id}`	Para um ID com formato inválido (ex: `invalido123`), a API retorna `400 Bad Request` em vez do esperado `404 Not Found`, sendo inconsistente com o tratamento de outras rotas.	**Baixa**
BUG-06	`POST /auth/login`	A resposta para credenciais inválidas é `401 Unauthorized` (o que é semanticamente correto), mas a documentação Swagger especifica `400 Bad Request`.	**Baixa**
BUG-07	`DELETE /sessions/{id}`	A mensagem de sucesso ao deletar uma sessão é `"Session removed"`, mas a	**Baixa**

		documentação especifica `"Session deleted successfully"`.	

Recomendações

O ciclo de testes foi eficaz em identificar defeitos significativos que afetam a estabilidade, segurança e usabilidade da API. O erro 500 (BUG-01) e a falha na validação de conflito (BUG-02) são impeditivos para a aprovação da versão atual.

Recomendações Imediatas:

1. **Correção Prioritária:** A equipe de desenvolvimento deve focar na resolução dos bugs de criticidade **Crítica** e **Alta** (BUG-01 e BUG-02).
2. **Alinhamento com a Documentação:** É crucial revisar e alinhar todas as respostas de erro e sucesso da API com o que está definido no Swagger. Inconsistências (BUG-03, BUG-04, BUG-05, BUG-06, BUG-07) geram retrabalho e dificultam a integração com outras aplicações.
3. **Reforçar o Tratamento de Erros:** A lógica de tratamento de erros deve ser padronizada, especialmente na diferenciação entre recursos não encontrados (`404`), acesso proibido (`403`) e requisições inválidas (`400`).

Evolução do Processo: Pipeline de CI com GitHub Actions

O processo de teste foi elevado a um novo patamar com a implementação de um workflow de Integração Contínua (CI) no GitHub Actions, garantindo validação automática da API a cada alteração no código. A seguir, uma análise detalhada da implementação.

O workflow está configurado para executar os testes da API de forma totalmente automatizada, utilizando as ferramentas e práticas mais recentes da Postman.

- **Ferramenta:** A automação utiliza o **Postman CLI**, a interface de linha de comando oficial da Postman, que se integra diretamente com a plataforma em nuvem.
- **Ambiente de Execução:** O job é executado em um ambiente `windows-latest` , demonstrando a portabilidade da solução.
- **Gatilho:** O workflow é acionado a cada `push` , garantindo feedback rápido para a equipe de desenvolvimento.
- **Execução:** O comando `postman collection run` é utilizado de forma exemplar:
  - **Collection e Environment por ID:** A collection (`44552321-fa...`) e o environment (`44552321-6d...`) são referenciados por seus IDs, garantindo que a pipeline execute sempre a versão mais recente disponível na nuvem do Postman.
  - **Variável de Ambiente:** A URL da API (`url_base`) é sobrescrita para apontar para o ambiente de homologação (`https://cinema-challenge-back.onrender.com/api/v1`), validando um

ambiente real e publicado.

- **Geração de Relatórios:** Múltiplos relatórios (`cli` , `json` , `html`) são gerados, fornecendo informações tanto para visualização rápida no console quanto para análise detalhada no relatório HTML.

Código do Workflow Implementado

```
name: Automated API tests using Postman CLI

on: push

jobs:
  automated-api-tests:
    runs-on: windows-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
- name: Install Postman CLI
  run: |
    powershell.exe -NoProfile -InputFormat None -ExecutionPolicy AllSigned -Command "[S

- name: Login to Postman CLI
  run: postman login --with-api-key ${{ secrets.POSTMAN_API_KEY }}

- name: Run API tests
  continue-on-error: true
  run: |
    postman collection run "44552321-fa4258bc-23d4-446d-9485-5f547f0e5d82" -e "4455232

- name: Upload Test Report
  uses: actions/upload-artifact@v4
  with:
    name: api-test-report
    path: results/report.html
```

Recomendações Finais

A automação da execução dos testes no pipeline de CI/CD eleva o projeto a um novo patamar de qualidade. A estratégia agora deve ser usar essa ferramenta como um motor para a correção e melhoria contínua.

Recomendações:

1. **Manter o Build "Quebrado" (Não Desativar os Testes):** A pipeline falhando é o estado correto até que os bugs sejam corrigidos. A equipe deve se comprometer a não mesclar código novo enquanto a pipeline estiver vermelha.
2. **Implementar um "Quality Gate" (Portão de Qualidade):** Configure o GitHub para **bloquear o merge de Pull Requests** na branch `main` ou `develop` caso o workflow de testes falhe. Isso força a correção dos bugs antes que eles avancem no ciclo de desenvolvimento.
3. **Foco em "Verdejar" a Pipeline:** A prioridade máxima do time de desenvolvimento deve ser corrigir os defeitos reportados (BUG-01 a BUG-07) em ordem de criticidade. Cada correção deve ser validada por uma nova execução da pipeline, com o objetivo de fazê-la passar (ficar "verde").

4. **Enriquecer a Suíte de Testes:** À medida que novos endpoints forem desenvolvidos ou os existentes alterados, os testes correspondentes devem ser adicionados à collection do Postman. Isso garante que a "rede de segurança" cresça junto com o produto.