Plano de Testes: Aplicação Cinema (Challenge AWS & AI for QE)

Testador: Guilherme Hepp

Aplicação: Cinema App (Front-end e Back-end)

---

1. Apresentação

Este documento detalha a estratégia e o planejamento de testes para a aplicação "Cinema App", parte do Challenge Técnico do PB AWS & AI for QE. O plano abrange os módulos de Autenticação, Filmes, Cinemas, Sessões e Reservas, focando em testes funcionais de API (back-end) e testes de interface de usuário E2E (front-end).

2. Objetivo

O principal objetivo deste plano de testes é validar a implementação da aplicação Cinema App, garantindo sua funcionalidade, confiabilidade e segurança em ambas as camadas (API e Web).

Objetivos Específicos:

- Verificar se todos os endpoints da API (Auth, Movies, Theaters, Sessions, Reservations) se comportam conforme o esperado, incluindo regras de autorização.
- Validar os principais fluxos de usuário na interface web (Login, Cadastro, Atualização de Perfil, e o fluxo E2E de Reserva).
- Identificar e documentar defeitos, inconsistências e vulnerabilidades de segurança (ex: permissões de Admin vs. User).
- Garantir a correta integração entre os módulos (ex: um usuário precisa de um token válido para ver seu perfil; uma reserva depende de uma sessão e assentos).
- Servir como base para a suíte de testes automatizados em Robot Framework.

3. Escopo

3.1. Dentro do Escopo

- **Testes de API (Back-end):**
    - Testes funcionais de todos os endpoints das APIs de /auth, /movies, /theaters, /sessions e /reservations.
    - Validação de operações CRUD (Create, Read, Update, Delete) para cada módulo.
    - Testes de "caminho feliz" (fluxos de sucesso) e "caminho triste" (fluxos de exceção e erro 4xx/5xx).
    - Validação das regras de autenticação (Bearer Token) e autorização (rotas exclusivas de Admin vs. User).
- **Testes de UI (Front-end):**

    - Teste de fluxos de usuário: Login, Cadastro de Usuário e Atualização de Perfil.
    - Teste de funcionalidades da página de filmes: Barra de Busca e Filtro de Gênero.
    - Teste E2E (End-to-End) do fluxo crítico de reserva: Login -> Selecionar Filme -> Selecionar Sessão -> Selecionar Assentos -> Página de Pagamento -> Confirmação da Reserva.

3.2. Fora do Escopo

- Testes de Performance (Carga e Stress).
- Testes de Usabilidade e Design (UX/UI).
- Testes de compatibilidade em múltiplos navegadores (foco no Chrome/Playwright).

4. Análise

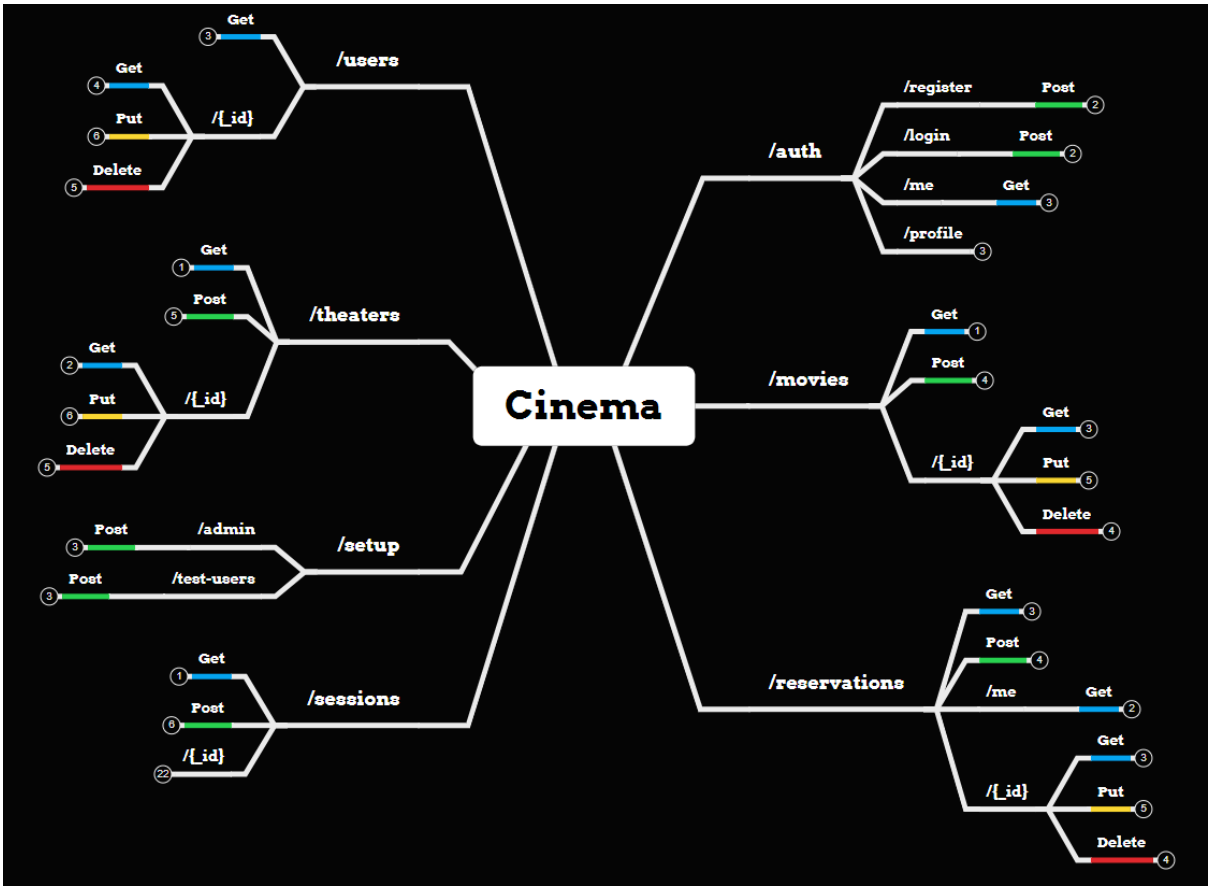
A estratégia de teste foi definida a partir da análise exploratória manual da API e do front-end. A análise revelou a existência de dois perfis de usuário (admin e user) com permissões distintas, sendo este um ponto crítico de segurança a ser validado. O fluxo E2E de reserva (desde o login até a confirmação) foi identificado como o fluxo de negócio mais crítico da aplicação web.

5. Técnicas Aplicadas

- **Testes Funcionais:** Validação de cada funcionalidade (API e Web) contra seus requisitos implícitos.
- **Testes Baseados em Cenários (E2E):** Criação de fluxos de ponta a ponta que simulam o uso real (ex: o fluxo completo de reserva no front-end e o teste de integração na API).
- **Testes de Segurança (Básicos):** Foco na validação de autorização, garantindo que um token de user não possa acessar endpoints de admin (ex: POST /movies). Durante esta fase, foi descoberta a falha de segurança BUG-03.
- **Testes Exploratórios:** Utilizados para descobrir bugs e melhorias não documentadas (BUG-01 a BUG-04 e MEL-01 a MEL-03).

6. Mapa Mental da Aplicação

O mapa mental foi a base para a criação dos cenários de teste, garantindo a cobertura visual de todos os endpoints da API (Auth, Movies, Theaters, Sessions, Users, Reservations) e seus respectivos métodos HTTP, parâmetros de entrada e códigos de resposta esperados.



7. Cenários de Teste Planejados (Cobertura da Automação)

A seguir, a lista consolidada dos principais cenários que foram automatizados.

7.1. Cenários de API

- **CT-API-01 (Auth):** Realizar login com sucesso (Admin).
- **CT-API-02 (Auth):** Realizar login com sucesso (User).
- **CT-API-03 (Auth):** Validar busca de perfil (/auth/me) com token válido.
- **CT-API-04 (Admin):** Criar, atualizar e deletar Filmes (CRUD).
- **CT-API-05 (Admin):** Criar, atualizar e deletar Cinemas (CRUD).
- **CT-API-06 (Admin):** Criar, atualizar e deletar Sessões (CRUD).
- **CT-API-07 (Admin):** Listar todos os usuários.
- **CT-API-08 (Security):** (User) Tentar acessar rota de Admin (ex: GET /users) e falhar (403).

- **CT-API-09 (Integration):** Fluxo E2E de API (Admin cria Filme/Cinema/Sessão -> User loga -> User cria Reserva).
- **CT-API-10 (Security - BUG-03):** (User) Tentar resetar assentos de sessão e (incorretamente) ter sucesso.

7.2. Cenários de Web (Front-end)

- **CT-WEB-01 (Homepage):** Carregar a Homepage e verificar os elementos principais.
- **CT-WEB-02 (Login):** Realizar login com sucesso.
- **CT-WEB-03 (Login):** Tentar login com senha inválida (Validar msg de erro - BUG-02).
- **CT-WEB-04 (Signup):** Realizar cadastro de novo usuário com sucesso.
- **CT-WEB-05 (Profile):** Acessar a página de perfil e atualizar o nome do usuário.
- **CT-WEB-06 (Movies):** Usar a barra de busca de filmes e validar o resultado.
- **CT-WEB-07 (Movies):** Usar o filtro de gênero e validar o resultado.
- **CT-WEB-08 (E2E - Reserva):** Fluxo completo: Login > Clicar "Filmes em Cartaz" > Clicar "Ver Detalhes" > Clicar "Selecionar Assentos" > Selecionar 2 assentos > Clicar "Continuar para Pagamento" > Selecionar "PIX" > Clicar "Finalizar Compra" > Validar "Reserva Confirmada!".
- **CT-WEB-09 (Reservations):** Validar navegação para a página "Minhas Reservas".

8. Priorização da Execução

- **P1 - Crítico:** Fluxos essenciais de login (API/Web) e o fluxo E2E de reserva (Web), que valida a principal funcionalidade da aplicação. Testes de segurança de autorização (Admin vs User).
- **P2 - Alto:** Testes de CRUD da API para todos os módulos (Filmes, Cinemas, Sessões) e os demais fluxos do front-end (Busca, Filtro, Perfil).
- **P3 - Médio:** Testes de caminhos negativos (ex: login com senha errada) e validação de todas as funcionalidades de usuário (ex: "Minhas Reservas").

9. Matriz de Risco

Risco Identificado	Probabilidade	Impacto	Estratégia de Mitigação
Acesso não autorizado a rotas de Admin	Média	**Crítico**	Testes de API exaustivos (CT-API-08, CT-API-10) usando um token de user em todos os endpoints de admin (CRUDs e GET /users) e validando o retorno 403 Forbidden.
Falha no fluxo E2E de Reserva	Média	Alto	Criação de um teste E2E robusto (CT-WEB-08) e um teste de integração de API (CT-API-09) que validam o fluxo completo de reserva, garantindo o isolamento dos dados.
Conflito de dados (Assentos Ocupados)	Alta	Médio	**(Problema Resolvido)** O teste E2E cria seus próprios dados (Filme, Cinema, Sessão) a cada execução, garantindo uma sessão "limpa" e evitando falhas por assentos já ocupados (400 Bad Request).

10. 🐞 Issues: Bugs e Melhorias Identificadas

Esta seção documenta os defeitos e sugestões de melhoria identificados durante os testes exploratórios e de automação.

10.1. Bugs Encontrados 🐛

ID	Título	Prioridade	Tipo
BUG-01	**[API/UX]** Endpoint de resetar assentos retorna mensagem de erro "falso positivo" para Admin	**Baixa**	API
BUG-02	**[Front-end]** Alerta de falha no login não é acessível/capturável via automação	**Média**	Testabilidade
BUG-03	**[Segurança]** Falha de Autorização: Usuário comum pode resetar assentos (endpoint de Admin)	**Crítica**	API / Lógica
BUG-04	**[Front-end]** Página de mudança de senha não é funcional	**Média**	Front-end

Detalhes dos Bugs:

- **BUG-01:** O front-end exibe "Erro ao resetar assentos [...] verifique se você tem permissões de admin" mesmo quando o admin reseta os assentos com sucesso (API retorna 200 OK).
- **BUG-02:** O alerta de "E-mail ou senha inválidos" no login desaparece muito rápido e não foi capturável pelas ferramentas de automação padrão, exigindo depuração avançada (Pause Execution) para ser identificado.
- **BUG-03 (Crítico):** Um usuário logado com a role user consegue enviar uma requisição PUT para o endpoint /api/v1/sessions/{id}/reset-seats (que deveria ser restrito ao admin) e a API (incorretamente) processa o pedido.
- **BUG-04:** A funcionalidade de "Mudar Senha" na página de perfil existe visualmente, mas não é funcional (não envia requisição).

10.2. Melhorias Sugeridas ✨

ID	Título	Prioridade	Tipo
MEL-01	Implementar interface de Admin para gerenciamento de filmes/sessões	**Alta**	Front-end
MEL-02	Adicionar funcionalidade de "Cancelar Reserva"	**Média**	Front-end / API

Detalhes das Melhorias:

- **MEL-01:** A aplicação não possui interface de administrador. Todas as ações de admin (criar filme, criar sessão) só podem ser feitas via API, o que impede o uso da aplicação por um admin não-técnico.
- **MEL-02:** A página "Minhas Reservas" é apenas informativa. Seria uma melhoria de UX permitir que o usuário cancele sua própria reserva.

11. Estratégia de Automação

- **Ferramentas:**
  - **API:** Robot Framework + RequestsLibrary
  - **Web:** Robot Framework + Browser Library (Playwright)
- **Padrões Aplicados:**
  - **API (Service Objects):** Criação de keywords personalizadas em arquivos .resource (ex: auth_keyword.resource, movies_keyword.resource) para abstrair a complexidade da RequestsLibrary e tornar os testes legíveis.
  - **Web (Page Object Model):** Separação de responsabilidades em arquivos de pages (contendo localizadores e keywords de ação, ex: LoginPage.resource, SessionPage.resource) e arquivos de tests.
- **Gerenciamento de Dados:**
  - Uso de um arquivo global env.resource (ou global_variables.resource) para armazenar dados sensíveis e URLs (BASE_URL, ADMIN_EMAIL, USER_EMAIL), facilitando a manutenção.
  - **Isolamento de Teste:** Os testes E2E (API e Web) são autossuficientes, criando seus próprios dados (novos filmes, sessões) para garantir que rodem de forma consistente e evitem falhas por dados "sujos" (ex: assentos ocupados).
- **Boas Práticas:**
  - Uso de Setup e Teardown (Setup Test Session, Start Session) para garantir um estado limpo para cada suíte de teste.
  - Uso de seletores robustos na Browser library (ex: role=, text=, placeholder=) em vez de XPaths frágeis.
  - Validações explícitas em múltiplas camadas (Status Code na API, elementos visíveis na Web).