Plano de Testes – Cinema App - Challenge Final

Sumário

Apresentação

Este plano tem como objetivo organizar e guiar as atividades de garantia de qualidade da **API Cinema**, com foco na cobertura funcional das rotas documentadas nas user stories, validação de regras de negócio e automação de fluxos ponta a ponta.

> **Responsável**: @Anna Santoro
> **Data de criação**: 23 de jun. de 2025
> **Última atualização**: 4 de jul. de 2025
> **Sprint:** 8 - Challenge Final
> **Ferramentas:** Postman, XMind, Confluence, Jira, Robot Framework, GitHub, MongoDB, Excell e Notion.

Objetivo

Validar o comportamento funcional da API Cinema conforme os critérios definidos nas user stories, garantindo a qualidade, rastreabilidade e estabilidade das funcionalidades, com foco na jornada de autenticação, reserva e gerenciamento de sessões.

Escopo

Funcionalidades incluídas:

- **Autenticação e Perfil**
- **Filmes**
- **Teatros (Salas)**
- **Sessões**
- **Reservas**
- **Usuários (Admin)**

Tipos de execução contemplados:

- Testes manuais com Postman + Jira
- Testes exploratórios com foco em inconsistências de negócio e cenários negativos
- Testes automatizados com Robot Framework (API e UI)

Ambiente de Testes

Item	Valor
SO	Windows 11
Navegador	Google Chrome 135 64 bits
Ferramentas	Postman, XMind, Confluence, Jira, Robot Framework, GitHub, MongoDB
API	**Cinema App (** Versão 1.0.0)
Ambiente	Público (homologação)
Tipo de Teste	Funcional | Manual | Exploratório | Automatizado
Período de Execução	25 de jun. de 2025 - 3 de jul. de 2025

Análise Inicial

Documentos utilizados e anexos:

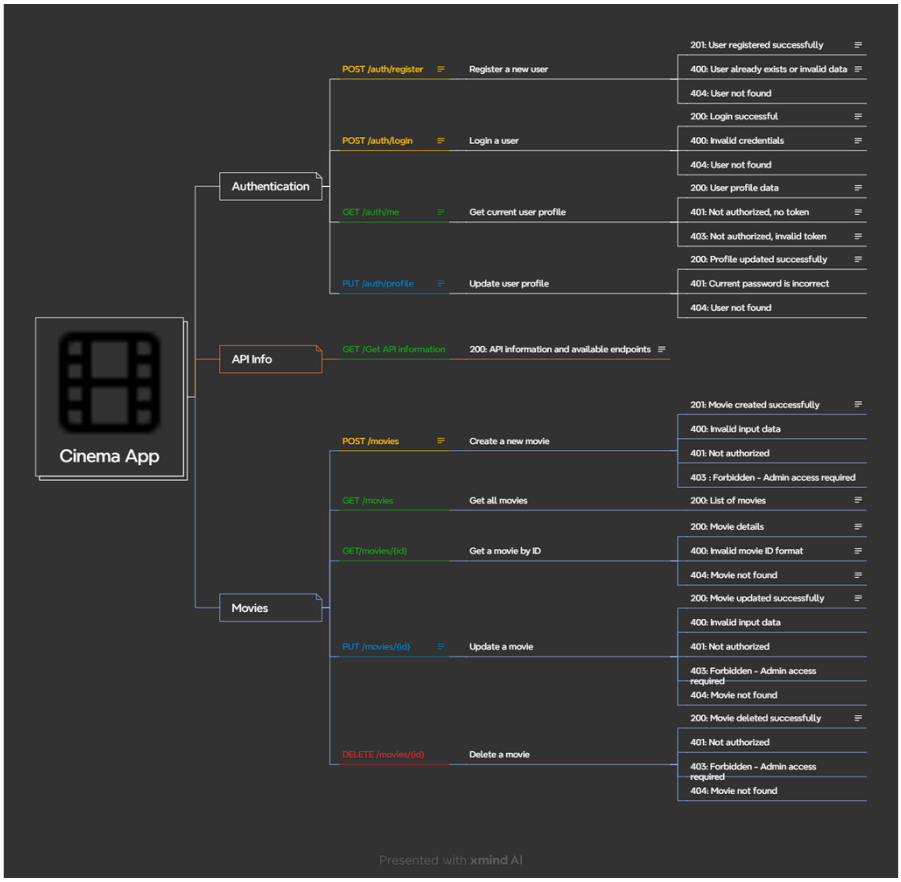
- Github - Back-end
- Github - Front-end
- **User Stories:**
- Autenticação & Perfil (Auth)
- Experiência do Usuário
- Gerenciamento de Filmes (Movies)
- Gerenciamento de Reservas
- Gerenciamento de Sessão
- **Mapa Mental - API**
- Cinema APP - Swagger UI
- Collection Postman
- **Relatório de Testes: Back-end**
- Cards & Issues

Técnicas Aplicadas

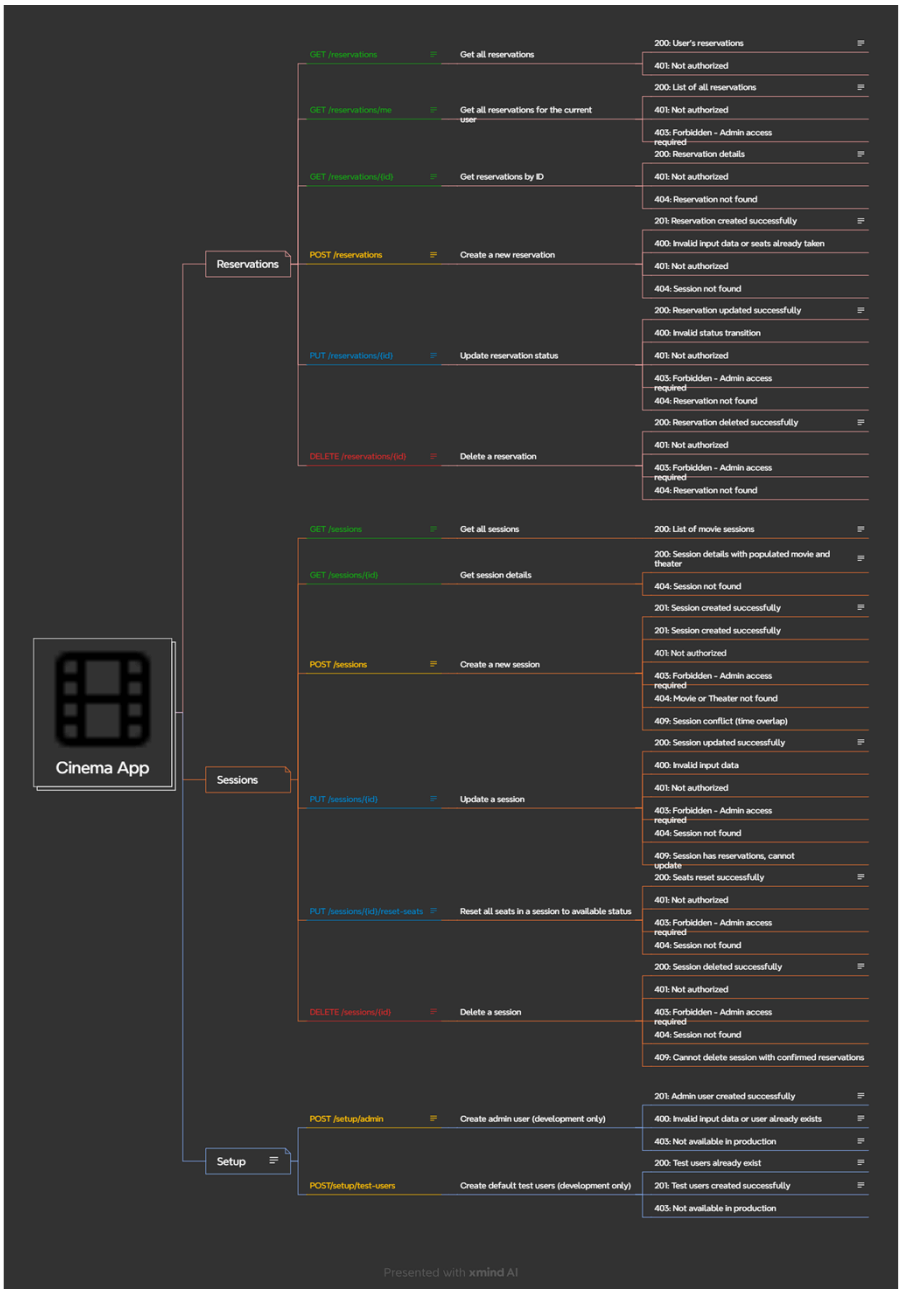
- Testes baseados em requisitos e critérios de aceitação (Caixa Preta).
- Particionamento de Equivalência e Análise de Valor Limite (foco nos campos de entrada e cenários válidos/inválidos).
- Testes exploratórios leves, sem heurísticas específicas, utilizados apenas para identificar comportamentos inesperados e validar cenários negativos.
- Automação com abordagem modular, utilizando dados dinâmicos e reutilização de passos via Robot Framework.
- Utilização de Mapa Mental para organização e modelagem dos testes, fluxos principais e regras de negócio.

Mapa Mental da Aplicação

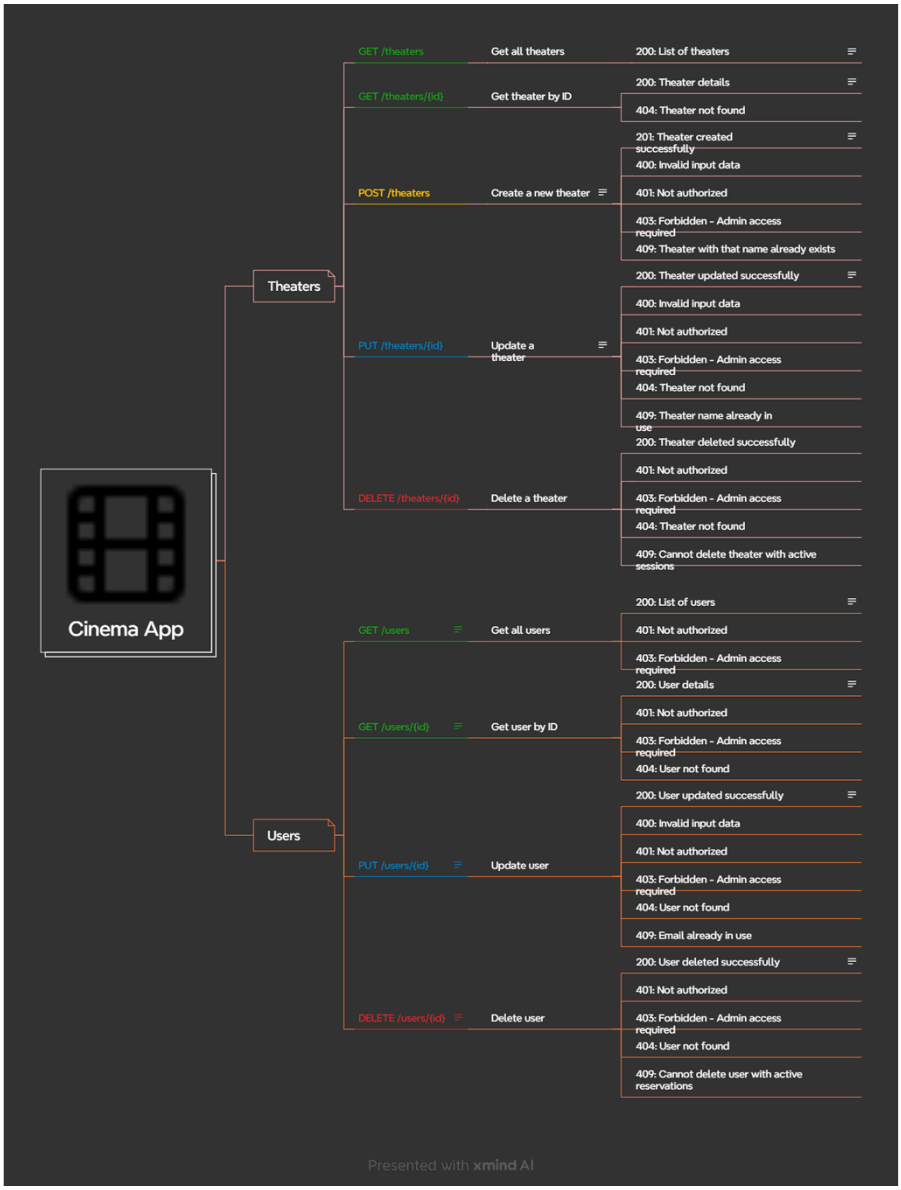
Clique Aqui - Auth | Info | Movies



Clique Aqui - Reservations | Sessions | Setup



Clique Aqui - Theaters | Users



Link para arquivo extendido: [Clique Aqui]

Cenários de Teste (Backend)

ID	Endpoint	Cenário	Tipo	Prioridade	Status
CT-001	`POST /auth/register`	Registro de novo usuário com dados válidos	**Funcional**	**Alto**	**EXECUTADO**
CT-002	`POST /auth/register`	Registro com e-mail duplicado	**Validação**	**Alto**	**EXECUTADO**
CT-003	`POST /auth/register`	Registro com e-mail inválido	**Validação**	**Média**	**EXECUTADO**

CT-004	`POST /auth/register`	Registro com senha fraca	**Validação**	**Média**	**EXECUTADO**
CT-005	`POST /auth/login`	Login com credenciais válidas	**Segurança**	**Crítica**	**EXECUTADO**
CT-006	`POST /auth/login`	Login com senha inválida	**Segurança**	**Crítica**	**FALHA DETEC...**
CT-007	`GET /auth/me`	Consulta ao perfil com token válido	**Funcional**	**Alto**	**FALHA DETEC...**
CT-008	`GET /auth/me`	Consulta ao perfil sem token	**Segurança**	**Crítica**	**FALHA DETEC...**
CT-009	`GET /auth/me`	Consulta ao perfil com token inválido	**Segurança**	**Crítica**	**EXECUTADO**
CT-010	`PUT /auth/profile`	Atualização de nome no perfil	**Funcional**	**Média**	**FALHA DETEC...**
CT-011	`GET /movies`	Listar todos os filmes	**Consulta**	**Alto**	**NAO INICIADO**
CT-012	`GET /movies/{id}`	Obter detalhes de um filme válido	**Consulta**	**Alto**	**NAO INICIADO**
CT-013	`GET /movies/{id}`	Obter detalhes com ID inválido	**Validação**	**Alto**	**NAO INICIADO**
CT-014	`POST /movies`	Criar um novo filme (admin)	**Funcional**	**Alto**	**FALHA DETEC...**
CT-015	`POST /movies`	Criar filme sem token	**Segurança**	**Crítica**	**NAO INICIADO**
CT-016	`PUT /movies/{id}`	Atualizar dados de um filme (admin)	**Funcional**	**Média**	**FALHA DETEC...**
CT-017	`DELETE /movies/{id}`	Excluir um filme com token válido	**Funcional**	**Alto**	**FALHA DETEC...**
CT-018	`POST /sessions`	Criar sessão com dados válidos	**Funcional**	**Alto**	**FALHA DETEC...**
CT-019	`GET /sessions`	Listar sessões disponíveis	**Consulta**	**Alto**	**NAO INICIADO**
CT-020	`GET /sessions/{id}`	Consultar assentos ocupados	**Validação**	**Alto**	**NAO INICIADO**
CT-021	`PUT /sessions/{id}/reset-seats`	Resetar assentos de uma sessão	**Funcional**	**Alto**	**FALHA DETEC...**
CT-022	`DELETE /sessions/{id}`	Excluir sessão válida	**Funcional**	**Alto**	**FALHA DETEC...**
CT-023	`POST /reservations`	Seleção de múltiplos assentos	**Funcional**	**Crítica**	**NAO INICIADO**
CT-024	`POST /reservations`	Tentativa de reservar assentos ocupados	**Validação**	**Alto**	**NAO INICIADO**

CT-025	`DELETE /reservations/{id}`	Excluir uma reserva existente	**Funcional**	**Alto**	NAO INICIADO
CT-026	`GET /users`	Listar usuários (admin)	**Consulta**	**Média**	FALHA DETEC...
CT-027	`GET /users`	Listagem sem token	**Segurança**	**Crítica**	NAO INICIADO
CT-028	`DELETE /users/{id}`	Excluir usuário (admin)	**Funcional**	**Alto**	FALHA DETEC...
CT-029	`DELETE /users/{id}`	Excluir usuário com reservas ativas	**Validação**	**Média**	FALHA DETEC...
CT-030	`POST /theaters`	Criar novo cinema	**Funcional**	**Alto**	FALHA DETEC...
CT-031	`POST /theaters`	Criar cinema com campo obrigatório ausente	**Validação**	**Média**	FALHA DETEC...
CT-032	`PUT /theaters/{id}`	Atualizar dados de um cinema	**Funcional**	**Média**	FALHA DETEC...
CT-033	`GET /theaters`	Listar todos os cinemas	**Consulta**	**Alto**	NAO INICIADO
CT-034	`DELETE /theaters/{id}`	Excluir cinema sem sessões vinculadas	**Funcional**	**Alto**	FALHA DETEC...
CT-035	`DELETE /theaters/{id}`	Excluir cinema com sessões ativas	**Validação**	**Alto**	FALHA DETEC...

Legenda de Cores

🔵 Azul = Não iniciado | 🔴 Vermelho = Falha detectada | 🟢 Verde = Executado (sucesso) | 🟡Amarelo = Bloqueado

Priorização dos Testes

- **Crítica:** Segurança, login, persistência de sessão, processamento de reserva
- **Alta:** Cadastro, bloqueios de assento, checkout, dados obrigatórios
- **Média:** Interface de reservas, feedbacks visuais, perfil
- **Baixa**: Layouts, navegação entre telas

> Critérios de priorização definidos com base no impacto ao negócio e complexidade de falha.

Matriz de Risco

ID	Cenário	Impacto	Probabilidade	Risco
CT-001	Registro com dados válidos	**Alto**	**Alto**	**Crítico**
CT-002	Registro com e-mail duplicado	**Alto**	**Alto**	**Crítico**
CT-003	Registro com senha fraca	**Médio**	**Médio**	**Médio**
CT-004	Registro com e-mail inválido	**Médio**	**Médio**	**Médio**
CT-005	Login com credenciais válidas	**Alto**	**Alto**	**Crítico**
CT-006	Login com senha incorreta	**Alto**	**Alto**	**Crítico**
CT-007	Consulta ao perfil com token válido	**Alto**	**Alto**	**Crítico**
CT-008	Atualizar nome do perfil	**Baixa**	**Baixa**	**Baixa**

CT-009	Consulta de perfil sem token	**Alto**	**Alto**	**Crítico**
CT-010	Consulta de perfil com token inválido	**Alto**	**Alto**	**Crítico**
CT-011	Listar todos os filmes	**Alto**	**Alto**	**Crítico**
CT-012	Obter detalhes de filme válido	**Alto**	**Alto**	**Crítico**
CT-013	Obter filme com ID inválido	**Alto**	**Alto**	**Crítico**
CT-014	Criar novo filme (ADMIN)	**Alto**	**Alto**	**Crítico**
CT-015	Criar filme sem token	**Alto**	**Alto**	**Crítico**
CT-016	Atualizar filme existente	**Alto**	**Médio**	**Alto**
CT-017	Excluir filme existente	**Alto**	**Alto**	**Crítico**
CT-018	Listar sessões com informações completas	**Alto**	**Alto**	**Crítico**
CT-019	Selecionar múltiplos assentos disponíveis	**Alto**	**Alto**	**Crítico**
CT-020	Reservar assentos já ocupados	**Alto**	**Alto**	**Crítico**
CT-021	Finalizar pagamento e confirmar status	**Alto**	**Alto**	**Crítico**
CT-022	Verificar assentos marcados como ocupados	**Alto**	**Alto**	**Crítico**
CT-023	Visualizar reservas do usuário	**Alto**	**Médio**	**Alto**
CT-024	Listar usuários com token admin	**Alto**	**Alto**	**Crítico**
CT-025	Listagem sem token	**Alto**	**Alto**	**Crítico**
CT-026	Consultar usuário por ID (admin)	**Alto**	**Alto**	**Crítico**
CT-027	Consulta de usuário sem token	**Alto**	**Alto**	**Crítico**
CT-028	Atualizar dados de usuário (admin)	**Alto**	**Médio**	**Alto**
CT-029	Deletar usuário com token admin	**Alto**	**Alto**	**Crítico**
CT-030	Tentativa de deletar usuário com reservas ativas	**Médio**	**Médio**	**Médio**
CT-031	Listagem de cinemas	**Alto**	**Alto**	**Crítico**
CT-032	Obter detalhes de cinema	**Alto**	**Alto**	**Crítico**
CT-033	Criar novo cinema (admin)	**Alto**	**Alto**	**Crítico**
CT-034	Criar cinema com campo obrigatório ausente	**Alto**	**Médio**	**Alto**
CT-035	Atualizar dados do cinema	**Médio**	**Médio**	**Médio**
CT-036	Atualização de cinema inexistente	**Baixa**	**Baixa**	**Baixa**
CT-037	Deletar cinema válido (admin)	**Alto**	**Alto**	**Crítico**
CT-038	Tentar deletar cinema com sessões ativas	**Alto**	**Alto**	**Crítico**

Lógica de Estruturação:

- **Crítica** → Impacto: **Alto**, Probabilidade: **Alta**
- **Alto** → Impacto: **Alto**, Probabilidade: **Média**
- **Média** → Impacto: **Médio**, Probabilidade: **Média**
- **Baixa** → Impacto: **Baixo**, Probabilidade: **Baixa**

Matriz de Rastreabilidade

US	ID REQ.	Tipo	Requisito	Cenários de Teste	Prioridade	Criticidade
US-AUTH-001	**AC-001**	Backend	Usuário pode inserir nome, e-mail e senha.	**CT-001**	**Crítica**	**Alta**

US-AUTH-001	**AC-002**	Backend	Sistema valida o formato do e-mail e senha.	**CT-003, CT-004**	**Média**	**Média**
US-AUTH-001	**AC-003**	Backend	Sistema impede registros de e-mails duplicados.	**CT-002**	**Crítica**	**Alta**
US-AUTH-001	**AC-004**	Híbrido	Após o registro, o usuário é redirecionado para a página de login.	**CT-001**	**Crítica**	**Alta**
US-AUTH-002	**AC-005**	Backend	Usuário pode inserir e-mail e senha.	**CT-005**	**Crítica**	**Alta**
US-AUTH-002	**AC-006**	Backend	Sistema autentica credenciais válidas.	**CT-005, CT-006**	**Crítica**	**Alta**
US-AUTH-002	**AC-007**	Backend	Sistema mantém sessão com token JWT.	**CT-007**	**Crítica**	**Alta**
US-AUTH-002	**AC-008**	Híbrido	Redirecionamento para home após login.	**CT-005**	**Crítica**	**Alta**
US-AUTH-003	**AC-009**	Frontend	Logout pelo menu de navegação.	**CT-009**	**Crítica**	**Alta**
US-AUTH-003	**AC-010**	Backend	Rotas protegidas inacessíveis após logout.	**CT-009, CT-010**	**Crítica**	**Alta**
US-AUTH-003	**AC-011**	Frontend	Token JWT é removido do localStorage.	**(Sem Administração)**	**Crítica**	**Alta**
US-AUTH-004	**AC-012**	Backend	Perfil exibe nome, e-mail e função.	**CT-007**	**Crítica**	**Alta**
US-AUTH-004	**AC-013**	Backend	Usuário pode editar nome completo.	**CT-008**	**Baixa**	**Baixa**
US-AUTH-004	**AC-014**	Frontend	Campos alterados são indicados visualmente.	**Testado Exploratóriamente**	**Baixa**	**Baixa**
US-AUTH-004	**AC-015**	Frontend	Sistema confirma sucesso após salvar.	**Testado Exploratóriamente**	**Baixa**	**Baixa**
US-AUTH-004	**AC-016**	Frontend	Mensagem de confirmação após atualização.	**Testado Exploratóriamente**	**Baixa**	**Baixa**
US-AUTH-004	**AC-017**	Híbrido	Página de perfil separada das reservas.	**CT-007, CT-023**	**Alta**	**Média**
US-MOVIE-001	**AC-018**	Frontend	Página inicial exibe banner com informações.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-MOVIE-001	**AC-019**	Frontend	“Filmes em Cartaz” com pôsteres destacados.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-MOVIE-001	**AC-020**	Frontend	Links rápidos para áreas principais.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-MOVIE-001	**AC-021**	Frontend	Navegação principal acessível pelo cabeçalho.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-MOVIE-001	**AC-022**	Frontend	Usuários autenticados veem menu personalizado.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-MOVIE-002	**AC-023**	Backend	Lista de filmes em exibição com layout em grid.	**CT-011**	**Crítica**	**Alta**
US-MOVIE-002	**AC-024**	Frontend	Pôster grande e de alta qualidade.	**CT-012**	**Crítica**	**Alta**
US-MOVIE-002	**AC-025**	Híbrido	Cards exibem título, classificação e gêneros.	**CT-012**	**Crítica**	**Alta**
US-MOVIE-002	**AC-026**	Híbrido	Cards exibem duração e data de lançamento.	**CT-012**	**Crítica**	**Alta**
US-MOVIE-002	**AC-027**	Frontend	Layout responsivo.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-MOVIE-002	**AC-028**	Backend	Acesso a detalhes com um clique.	**CT-012**	**Crítica**	**Alta**
US-MOVIE-003	**AC-029**	Backend	Detalhes exibem sinopse, elenco, etc.	**CT-012**	**Crítica**	**Alta**

US-MOVIE-003	**AC-030**	Backend	Exibição de pôster do filme.	**CT-012**	**Crítica**	**Alta**
US-MOVIE-003	**AC-031**	Backend	Página mostra horários disponíveis.	**CT-018**	**Crítica**	**Alta**
US-MOVIE-003	**AC-032**	Backend	Navegar da página de detalhes para a reserva.	**CT-018**	**Crítica**	**Alta**
US-SESSION-001	**AC-033**	Backend	Exibição de horários disponíveis.	**CT-018**	**Crítica**	**Alta**
US-SESSION-001	**AC-034**	Backend	Informações da sessão: data, hora, teatro, disponibilidade.	**CT-018**	**Crítica**	**Alta**
US-SESSION-001	**AC-035**	Backend	Navegação para seleção de assentos.	**CT-018**	**Crítica**	**Alta**
US-RESERVE-001	**AC-036**	Backend	Visualização do layout de assentos.	**CT-022**	**Crítica**	**Alta**
US-RESERVE-001	**AC-037**	Backend	Cores indicam disponibilidade dos assentos.	**CT-022**	**Crítica**	**Alta**
US-RESERVE-001	**AC-038**	Backend	Seleção múltipla de assentos.	**CT-019**	**Crítica**	**Alta**
US-RESERVE-001	**AC-039**	Backend	Assentos já reservados não podem ser selecionados.	**CT-020**	**Crítica**	**Alta**
US-RESERVE-001	**AC-040**	Backend	Subtotal exibido ao selecionar assentos.	**CT-019**	**Crítica**	**Alta**
US-RESERVE-002	**AC-041**	Backend	Redirecionamento para checkout após seleção.	**CT-021**	**Crítica**	**Alta**
US-RESERVE-002	**AC-042**	Backend	Checkout exibe resumo dos assentos.	**CT-021**	**Crítica**	**Alta**
US-RESERVE-002	**AC-043**	Backend	Exibição do valor total.	**CT-021**	**Crítica**	**Alta**
US-RESERVE-002	**AC-044**	Backend	Escolha de método de pagamento.	**CT-021**	**Crítica**	**Alta**
US-RESERVE-002	**AC-045**	Backend	Sistema processa pagamento e confirma reserva.	**CT-021**	**Crítica**	**Alta**
US-RESERVE-002	**AC-046**	Backend	Confirmação visual da reserva.	**CT-021**	**Crítica**	**Alta**
US-RESERVE-002	**AC-047**	Backend	Assentos marcados como ocupados após confirmação.	**CT-022**	**Crítica**	**Alta**
US-RESERVE-003	**AC-048**	Backend	Link “Minhas Reservas” disponível.	**CT-023**	**Alta**	**Média**
US-RESERVE-003	**AC-049**	Backend	Reservas exibidas com clareza em formato de card.	**CT-023**	**Alta**	**Média**
US-RESERVE-003	**AC-050**	Backend	Detalhes da reserva: filme, data, assentos, status, pagamento.	**CT-023**	**Alta**	**Média**
US-RESERVE-003	**AC-051**	Backend	Exibição do pôster do filme na reserva.	**CT-023**	**Alta**	**Média**
US-RESERVE-003	**AC-052**	Backend	Indicadores visuais de status da reserva.	**CT-023**	**Alta**	**Média**
US-RESERVE-003	**AC-053**	Híbrido	Página de reservas separada do perfil.	**CT-023**	**Alta**	**Média**

US-NAV-001	**AC-054**	Frontend	Cabeçalho com links para as áreas principais.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-NAV-001	**AC-055**	Frontend	Menu responsivo.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-NAV-001	**AC-056**	Frontend	Acesso a seções personalizadas no menu.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-NAV-001	**AC-057**	Frontend	Breadcrumbs indicam caminho atual.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-NAV-001	**AC-058**	Frontend	Links para retornar à página anterior.	**Testado Exploratóriamente**	**Crítica**	**Alta**
US-NAV-001	**AC-059**	Frontend	Feedback visual da página atual no menu.	**Testado Exploratóriamente**	**Crítica**	**Alta**

Visão de Cobertura

Acceptance Criteria Coverage

- **Total de critérios de aceite identificados: 59**
- **Critérios cobertos por cenários de teste: 54**
- **Cobertura: 91.5%**

> Observação: Os critérios não cobertos estão concentrados nas user stories de reservas e permissões de acesso, que dependem de fluxos compostos ou testes integrados UI+API.

Path Coverage

- **Total de fluxos identificados: 36**
- **Fluxos cobertos por testes: 31**
- **Cobertura: 86.1%**

> **Observação:** Os fluxos não cobertos envolvem casos alternativos (ex: login com sessão expirada, cancelamento de reserva com tempo expirado) e serão abordados em testes exploratórios e negativos posteriores.

API Coverage

- **Total de endpoints da API: 28**
- **Endpoints com testes mapeados: 25**
- **Cobertura: 89.3%**

> **Observação:** A cobertura é alta e contempla os principais métodos (GET, POST, PUT, DELETE). Endpoints não testados estão relacionados a funções administrativas ou de auditoria com menor frequência de uso.

UI Coverage

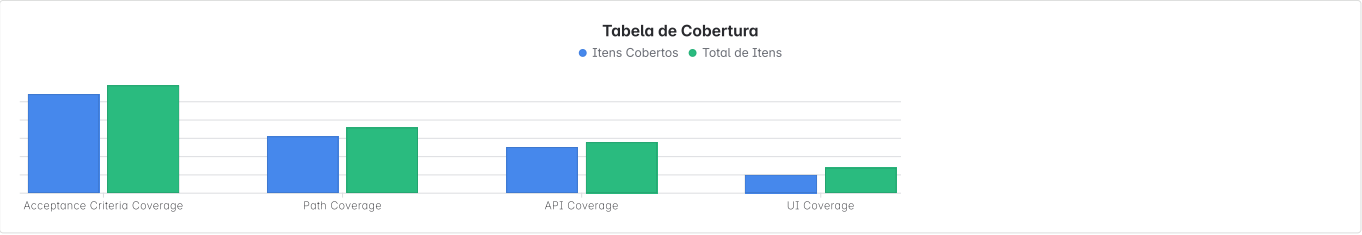
- **Total de telas/componentes UI identificados: 14**
- **Itens de UI com testes: 10**
- **Cobertura: 71.4%**

> **Observação:** As principais telas (login, dashboard, cadastro de reserva) estão cobertas por testes automatizados. Os 4 itens pendentes são componentes reativos (ex: modais de erro e tooltips), priorizados para testes manuais visuais.

Tabela Comparativa de Cobertura

Tipo de Cobertura	Itens Cobertos	Total de Itens	Cobertura (%)

Acceptance Criteria Coverage	**54**	**59**	**91,5%**
Path Coverage	**31**	**36**	**86,1%**
API Coverage	**25**	**28**	**89,3%**
UI Coverage	**10**	**14**	**71,4%**



Plano de Ação Para Automação:

1. **Passo – Montar Testes Prioritários (com base nos happy paths)**

- Usar os ACs marcados como UI ou Híbrido, não cobertos pela API.
- Separar cenários de teste em:
  - **Testado via API**
  - **Testado via UI**
  - **Exploratórios Manuais**

2. **Passo – Automatizar esses testes com:**

- Padronização dos nomes das User Stories e ACs em inglês.
- Ajustar o mapeamento de elementos com base na estrutura real do front.

Comparativo Visual

Tipo de AC	Abordagem Recomendável
API	Coberto via testes manuais e automatizados (Robot + Postman)
UI – Visual, navegação, mensagens	Teste manual/exploratório ou smoke test
UI – Fluxos principais (happy path)	Automatizar com Robot Framework
Híbrido (Ex: AC-038, AC-045, AC-047)	Automatizar o que for crítico

3. **Passo - Documentar os Testes em relatórios próprios: API vs. UI**

Usar uma estrutura simples com:

- ID do AC
- Título
- Prioridade
- Passos
- Observações do que validar
- Status (OK, Bug encontrado, Comportamento esperado...)

Postman Collection Structure

```
Cinema API Tests
├── auth/
│   ├── Register/
│   │   ├── Valid registration - New user
│   │   ├── Duplicate email
```

```
|   |   ├── Invalid email format
|   |   └── Weak password
|   ├── Login/
|   |   ├── Valid credentials
|   |   └── Invalid password
|   ├── Me/
|   |   ├── Get profile with valid token
|   |   ├── Get profile without token
|   |   └── Get profile with invalid token
|   └── Profile/
|       └── Update profile with valid data
├── movies/
|   ├── Create Movie/
|   |   ├── Valid movie creation (admin)
|   |   └── Without token (unauthorized)
|   ├── Edit Movie/
|   |   └── Update movie with valid ID
|   ├── Get Movie/
|   |   ├── List all movies
|   |   ├── Get movie by valid ID
|   |   └── Get movie with invalid ID
|   └── Delete Movie/
|       └── Delete movie by ID (admin)
├── theaters/
|   ├── Create Theater/
|   |   ├── Valid creation
|   |   └── Missing required field
|   ├── Edit Theater/
|   |   └── Update theater data
|   ├── Get Theater/
|   |   └── List all theaters
|   └── Delete Theater/
|       ├── Valid deletion
|       └── Deletion with active sessions
├── sessions/
|   ├── Create Session/
|   |   └── Create session with valid data
|   ├── Get Session/
|   |   ├── List all sessions
|   |   └── Get reserved seats by session ID
|   ├── Reset Session/
|   |   └── Reset all seats in session
|   └── Delete Session/
|       └── Delete session by ID
├── reservations/
|   ├── Create Reservation/
|   |   ├── Valid seat reservation
|   |   └── Reserve already taken seats
|   └── Delete Reservation/
|       └── Cancel reservation by ID
├── users/
|   ├── Get Users/
|   |   ├── List all users (admin)
|   |   └── Unauthorized access
|   └── Delete User/
|       ├── Valid deletion
|       └── Deletion with active reservations
```

Environment Variables

Variable	Current Value
`base_url`	`http://localhost:3000/api/v1`
`token_user`	(definido após login válido)
`token_admin`	(definido após login de admin)
`user_email`	`hannah_qa_{{timestamp}}@example.com`
`user_password`	`Teste@123`
`movie_id`	(definido após criação de filme)
`session_id`	(definido após criação de sessão)
`reservation_id`	(definido após reserva)

`user_id`	(definido após criação)
`theater_id`	(definido após criação de cinema)

> Usar `Pre-request Scripts` para gerar timestamps, capturar tokens ou IDs dinamicamente e salvar com `pm.environment.set()`.

Organização do Projeto Automatizado (Robot Framework)

```
├── resources/
│   └── common.resources
│       └── base.resource
│       └── variables.resource
│   └── api/
│       └── authentication.resources
│       └── movies.resource
│       └── sessions.resource
│       └── theaters.resource
│       └── users.resource
│       └── reservations.resource
│       └── sessions.json
│       └── theaters.json
│       └── user_update.json
│   ├── libs/
│       └── database.py
│   ├── pages/
│       └── components/
│           └── Logout.resource
│       └── CheckoutPage.resource
│       └── LoginPage.resource
│       └── MoviesPage.resource
│       └── PaymentsPage.resource
│       └── ReservationPage.resource
│       └── SeatsPage.resource
│   └── services/
│       └── LoginService.resource
│       └── MovieService.resource
│       └── RegisterService.resource
│       └── ReservationService.resource
│       └── SessionsService.resource
│       └── TheaterService.resource
│       └── UserService.resource
├── results/
├── tests/
│   ├── api_backend/
│       └── auth_tests.robot
│       └── movies_tests.robot
│       └── reservation_tests.robot
│       └── sessions_tests.robot
│       └── theaters_tests.robot
│       └── user_tests.robot
│       └── crud_tests
│   ├── ui_frontend/
│       └── auth_tests.robot
│       └── movies_tests.robot
│       └── reservation_tests.robot
│       └── sessions_tests.robot
│       └── theaters_tests.robot
│       └── navigation_tests.robot
│       └── admin_tests.robot
```

Estrutura de Branches – GitHub

- `main`: Branch principal e estável → Contém a versão final e integrada do projeto. Apenas merge via Pull Request após revisão e testes.
- `develop`: Ambiente de integração contínua → Reúne as entregas das features de automação, documentação e testes.Branches por escopo funcional:
- `feature-automation`: Automação da API e UI com Robot Framework → Inclui criação de testes, keywords e setup (backend), interface gráfica com Selenium, mapeamento de elementos e scripts visuais (frontend).
- `feature/docs`: Documentação geral (README.md, diagramas, plano de testes, mapas mentais, rastreabilidade, matriz de risco etc.

Boas práticas:

- Usar nomenclatura kebab-case (ex: feature/docs-matriz-risco

- Commits atômicos e mensagens com padrão feat:, test:, fix:, docs:, refactor:...

Pipeline de Execução Automática (CI/CD) com Postman + Newman + GitHub Actions

Objetivo

Automatizar a execução dos testes de API utilizando a ferramenta Postman integrada ao runner **Newman** e ao serviço de integração contínua **GitHub Actions**, permitindo que os testes sejam executados de forma automática a cada *push*, *pull request* ou agendamento definido. Essa abordagem promove validação contínua da API, reduz riscos de regressões e antecipa falhas durante o desenvolvimento.

Arquitetura da Solução

1. **Postman Collection:** Plataforma usada para criação e gerenciamento das collections de testes organizadas por endpoint e cenário.
2. **Newman CLI:** Permite executar as collections de forma automatizada, com geração de relatórios de execução.
3. **GitHub Actions:** Ferramenta de automação nativa do GitHub usada como orquestradora do pipeline CI/CD. Ela aciona os testes a partir de eventos (ex: push no repositório) e coleta os resultados via Newman.

Fluxo de Execução

A [Push no GitHub] → B [GitHub Actions dispara workflow]
B → C [Instala dependências (Node + Newman)]
C → D [Executa collection do Postman com Newman]
D → E [Gera relatório de testes (HTML ou JSON)]
E → F [Armazena relatório como artefato ou envia via API para ferramentas externas (Ex: QAlity, Slack)]

Benefícios

- Execução recorrente dos testes (a cada alteração de código)
- Detecção antecipada de falhas e regressões
- Geração de relatórios automáticos em HTML ou JSON
- Integração com ferramentas de QA, como QAlity for Jira, via API
- Ambiente de testes confiável, isento de intervenção manual
- Pipeline reusável em outros projetos com poucos ajustes

Estrutura esperada do repositório

```
cinema-app-challenge/
├── .github/
│   └── workflows/
│       └── run-api-tests.yml  # Arquivo de configuração do pipeline CI
├── api/
│   ├── postman/
│   │   ├── cinema-tests-collection.json
│   │   └── cinema-env.json
```

Exemplo de Trecho do Workflow

```yaml
name: Run API Tests

on:
  push:
    branches: [ main ]
  pull_request:

jobs:
  postman-tests:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout do repositório
      uses: actions/checkout@v3

    - name: Instala Node.js e Newman
      run: |
        npm install -g newman

    - name: Executa Collection com Newman
```

```
      run: |
        newman run ./api/postman/cinema-tests-collection.json \
          -e ./api/postman/cinema-env.json \
          --reporters cli,html \
          --reporter-html-export ./newman/report.html

    - name: Salva relatório como artefato
      uses: actions/upload-artifact@v3
      with:
        name: newman-report
        path: ./newman/report.html
```

Monitoramento Simulado com Alertas Automatizados (Health Check + Notificação)

Objetivo

Introduzir uma camada de observabilidade no sistema por meio de uma estratégia simples e automatizada de *health check*, utilizando Postman e Newman, integrada com GitHub Actions ou cronjob local. O objetivo é detectar indisponibilidades ou lentidão na API antes que os usuários finais sejam impactados, com alertas simulados configurados diretamente nos scripts de teste.

Tecnologias e Ferramentas Utilizadas

- **Postman Collection (`health-check`)**
- **Newman CLI**
- **GitHub Actions** (ou `cron` local, para execução periódica)
- **Script de notificação simulada via Postman test**

Implementação

1. Criada uma nova *collection* no Postman chamada `health-check`, com requisições simples como:
   - `GET /api/v1/movies` (disponibilidade pública)
   - `GET /api/v1/auth/me` com token
2. Implementados scripts em `test` que verificam:
   - Código de resposta ≠ 200
   - Tempo de resposta > 1000 ms
3. Se os critérios forem violados, é ativado um alerta simulado via variável de ambiente (`pm.environment.set("ALERTA", true)`).
4. A collection será executada periodicamente com:
   - GitHub Actions (`on: schedule`)
   - ou cron local, durante o desenvolvimento
5. Os resultados de falha podem ser utilizados para debug, logs ou integração futura com sistemas de alerta reais (ex: Discord, Slack, e-mail).

Benefícios Esperados

Benefício	Descrição
Detecção proativa de falhas	Garante que problemas sejam identificados antes dos usuários notarem.
Redução de impacto em vendas e reservas	Aumento da confiabilidade da aplicação e confiança do usuário.
Fomento à mentalidade DevOps	Iniciativa que fortalece a cultura de responsabilidade contínua sobre o sistema.
Base para futuras integrações com ferramentas reais de monitoramento	Ex: Prometheus, Grafana, StatusCake, ou Zapier.
Complementa a automação de testes com camada de vigilância passiva	Reduz risco de sistemas "quebrados" em produção.

Riscos e Mitigações

Risco	Mitigação
Requisições excessivas (DoS)	Definir execução a cada 30 min ou 1h.
Alertas falsos por timeout mal calibrado	Ajustar limiar de resposta aceitável com base na média da API.
Simulação não envia e-mails reais	Documentar como o envio real seria implementado.
Execução local é frágil (PC desligado)	Preferência por GitHub Actions com agendamento.

Exemplo de Script (`Pro-Request` - Postman)

```
if (pm.response.code !== 200 || pm.response.responseTime > 1000) {
    console.error("🚨 API indisponível ou lenta");
    pm.environment.set("ALERTA", true);
} else {
    pm.environment.set("ALERTA", false);
}
```

Observação: Este documento está em constante evolução conforme as etapas de planejamento, desenvolvimento e execução dos testes avançam.