

## Final Challenge - Kauã Raffaello

### PLANO DE TESTES - SISTEMA CINEMA APP

#### Apresentação

Este documento descreve a estratégia de testes para a aplicação Cinema, composta por uma API REST e interface web, que simula um sistema completo de reserva de ingressos de cinema para fins de avaliação técnica.

#### Objetivo

Garantir a qualidade do sistema Cinema através de testes automatizados que validam os endpoints da API e os fluxos end-to-end da interface web, assegurando a funcionalidade completa e a segurança da aplicação.

---

#### Resumo

O Cinema é uma aplicação completa de gerenciamento de cinema com dois perfis de usuário: **administrador** (com acesso total ao sistema, ainda em construção) e **usuário comum** (foco em reservas de ingressos). Este planejamento de testes contempla a validação de funcionalidades críticas através de automação com Robot Framework, cobrindo tanto a camada de API quanto a interface web. A hipótese deste teste é que todos os endpoints estejam funcionais e que os fluxos de negócio estejam operando corretamente conforme as regras estabelecidas.

#### Pessoas Envolvidas

Neste planejamento teve apenas um total de 1 QA envolvido.

**Kauã Raffaello (NoBugs)** - QA Engineer

---

#### Ambientação que será testado

URL da API: <http://localhost:3000/api/v1>

URL do Front-end: <http://localhost:3000>

#### Repositórios:

- Back-end: [GitHub - juniorschmitz/cinema-challenge-back: Back-end for the QE PB Final Challenge](#)

- Front-end: [GitHub - juniorschmitz/cinema-challenge-front: Front-end for the QE PB Final Challenge](#)  
[e.](#)

**Ferramentas:** Robot Framework, Browser Library, RequestsLibrary, GitHub, GitHub Actions.

**Dependências:** Token JWT de autenticação, MongoDB rodando, dados de seed carregados.

## Recursos e Ferramentas

**IDE:** VSCode

**Framework:** Robot Framework 6.1+

### Libraries Necessárias:

- **RequestsLibrary** - Para requisições HTTP à API
- **Browser Library** - Para testes web com Playwright
- **Collections** - Manipulação de dicionários e listas
- **String** - Validações de strings
- **BuiltIn** - Keywords nativas do Robot
- **OperatingSystem** - Manipulação de arquivos e sistema
- **DateTime** - Validações de data e hora

### Bibliotecas Python Customizadas:

- **DataGenerator** - Geração dinâmica de dados de teste
- **ApiHelper** - Helpers para operações de API
- **WebHelper** - Helpers para validações web

**Versionamento:** GitHub para controle de versão e CI/CD via GitHub Actions.

---

## Escopo

Endpoints de teste - API:

- ✓ [/api/v1/auth](#) - Autenticação (register, login, perfil)
- ✓ [/api/v1/users](#) - Gerenciamento de usuários
- ✓ [/api/v1/movies](#) - CRUD de filmes
- ✓ [/api/v1/theaters](#) - CRUD de salas/teatros
- ✓ [/api/v1/sessions](#) - Gerenciamento de sessões

## ✓ /api/v1/reservations - Sistema de reservas

### Fluxos de teste - Web:

- ✓ Autenticação (login, registro, logout)
- ✓ Navegação e busca de filmes
- ✓ Seleção de sessões e assentos
- ✓ Fluxo completo de reserva
- ✓ Gestão de reservas pessoais
- ✓ Painel administrativo

### Fora do escopo:

- Integrações externas de pagamento
- Testes de performance e carga
- Testes de segurança avançados (pentest)
- Validações de acessibilidade (WCAG)
- Testes em múltiplos navegadores (foco apenas em Chromium)

---

## Análise

Iremos focar em testes automatizados funcionais dos endpoints da API e fluxos end-to-end da interface web, priorizando cenários críticos de negócio.

### Riscos identificados:

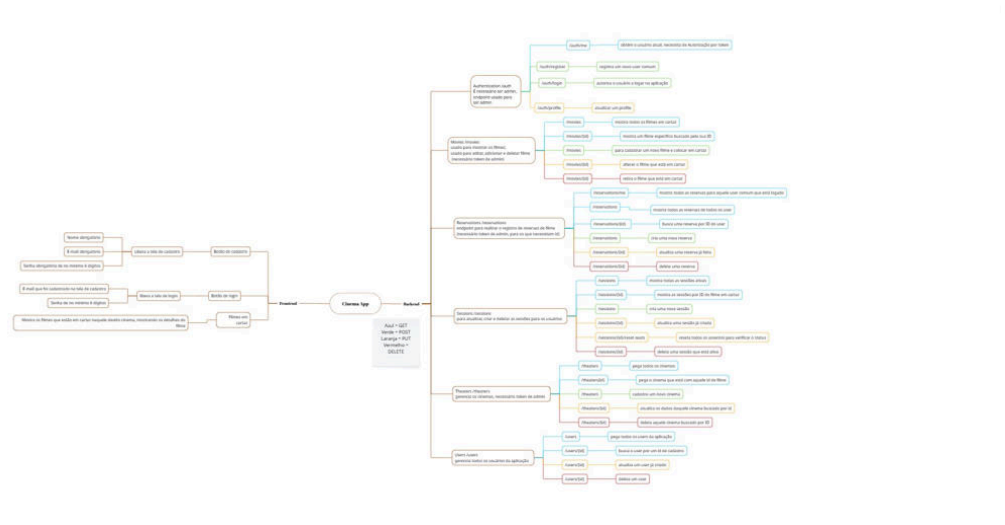
- API ou Front-end instáveis durante desenvolvimento
- Dependência de MongoDB rodando localmente
- Seeds de dados não carregados corretamente
- Conflitos de portas (3000, 5000, 27017)
- AWS apagão.
- Tempo limitado para implementação (9 dias)

### Técnicas Aplicadas


- ✓ **Teste de caixa-preta** - Validação funcional sem análise de código
- ✓ **Teste de API automatizado** - Validação de contratos e respostas REST
- ✓ **Teste funcional automatizado** - Validação de fluxos de negócio

- ✓ **Teste end-to-end (E2E)** - Jornadas completas de usuário
- ✓ **Teste de regressão automatizado** - Garantir que novas mudanças não quebrem funcionalidades
- ✓ **Padrões de projeto** - Page Objects (Web) e Service Objects (API)

Mapa Mental da Aplicação



Cronograma do Planejamento até a Entrega

Dia	Data	Atividade
Dia 1-2	21-22/10	 <b>Planejamento completo</b> do projeto + definição de escopo  Evolução com GenAI e <b>setup do ambiente</b> de testes
Dia 3	23/10	 <b>Estruturação do projeto no GitHub</b> + configuração do <b>Robot Framework</b>
Dia 4-5	24-25/10	 <b>Implementação dos testes de API</b> (cenários <b>P0</b> e <b>P1</b> )

<b>Dia 6-7</b>	26-27/10	 <b>Implementação dos testes Web E2E</b> (cenários P0)
<b>Dia 8</b>	28/10	 <b>Refinamento geral</b> , abertura de <b>issues</b> e configuração do <b>pipeline CI/CD</b>
<b>Dia 9</b>	29/10	 <b>Documentação final</b> (README, Wiki)
<b>Dia 10</b>	30/10	<b>Apresentação final</b> – duração: <b>6 minutos</b>

## CENÁRIOS DE TESTE PLANEJADOS

### 1. AUTENTICAÇÃO (AUTH)

#### 1.1 Criar usuário com dados válidos (POST)

**Pré-condição:** Nenhum usuário cadastrado com o mesmo e-mail.

**Ação:** Enviar POST para `/api/v1/auth/register` com dados válidos (name, email, password).

#### **Resultado esperado:**

- Usuário cadastrado com sucesso, HTTP 201
- Resposta retorna token JWT e dados do usuário
- Campos obrigatórios preenchidos corretamente

#### **Cenários de borda:**

- E-mail no limite de caracteres (254)
- Senha com caracteres especiais
- Nome com acentuação
- Campos extras no payload

#### **Mensagens de erro esperadas:**

- "E-mail já cadastrado"
- "Senha deve ter no mínimo 6 caracteres"
- "E-mail inválido"

### 1.2 Criar usuário com dados inválidos (POST)

**Pré-condição:** Tentativa de cadastro com dados inválidos.

**Ação:** Enviar POST com e-mail duplicado, senha curta ou campos obrigatórios faltando.

**Resultado esperado:**

- Usuário não cadastrado, HTTP 400
- Mensagem de erro clara e específica

**Cenários de borda:**

- E-mail sem @
- Senha vazia
- Nome com números ou caracteres especiais
- Payload vazio

### 1.3 Login com credenciais válidas (POST)

**Pré-condição:** Usuário previamente cadastrado.

**Ação:** POST `/api/v1/auth/login` com e-mail e senha corretos.

**Resultado esperado:**

- Login realizado, HTTP 200
- Token JWT válido retornado
- Dados do usuário incluídos na resposta

### 1.4 Login com credenciais inválidas (POST)

**Pré-condição:** Tentativa de login com dados incorretos.

**Ação:** POST com e-mail inexistente ou senha errada.

**Resultado esperado:**

- Login negado, HTTP 401
- Mensagem: "Credenciais inválidas"

### 1.5 Obter perfil do usuário autenticado (GET)

**Pré-condição:** Usuário autenticado com token válido.

**Ação:** GET `/api/v1/auth/me` com header Authorization.

**Resultado esperado:**

- Perfil retornado, HTTP 200

- Dados completos do usuário (exceto senha)

#### 1.6 Obter perfil sem autenticação (GET)

**Pré-condição:** Requisição sem token JWT.

**Ação:** GET `/api/v1/auth/me` sem header Authorization.

**Resultado esperado:**

- Acesso negado, HTTP 401
  - Mensagem: "Token não fornecido"
- 

## 2. FILMES (MOVIES)

#### 2.1 Listar todos os filmes (GET)

**Pré-condição:** Filmes cadastrados no sistema.

**Ação:** GET `/api/v1/movies` .

**Resultado esperado:**

- Lista retornada, HTTP 200
- Array com todos os filmes
- Estrutura correta (title, genre, duration, rating, etc)

**Cenários de borda:**

- Banco vazio (retornar array vazio)
- Paginação se houver muitos filmes
- Filtros por gênero

#### 2.2 Buscar filme por ID válido (GET)

**Pré-condição:** Filme existe no banco.

**Ação:** GET `/api/v1/movies/:id` .

**Resultado esperado:**

- Filme retornado, HTTP 200
- Dados completos incluindo sessões disponíveis

#### 2.3 Buscar filme por ID inválido (GET)

**Pré-condição:** ID não existe ou formato inválido.

**Ação:** GET `/api/v1/movies/id-invalido` .

**Resultado esperado:**

- Filme não encontrado, HTTP 404
- Mensagem: "Filme não encontrado"

**2.4 Criar filme como admin (POST)**

**Pré-condição:** Usuário autenticado com role 'admin'.

**Ação:** POST `/api/v1/movies` com dados válidos (title, genre, duration, rating, description).

**Resultado esperado:**

- Filme criado, HTTP 201
- ID gerado e retornado
- Dados salvos corretamente

**Cenários de borda:**

- Rating = 0 ou 5
- Duração muito longa (>300 min)
- Título duplicado
- Descrição vazia

**Mensagens de erro esperadas:**

- "Rating deve estar entre 0 e 5"
- "Duração deve ser maior que 0"
- "Título é obrigatório"

**2.5 Criar filme como usuário comum (POST)**

**Pré-condição:** Usuário com role 'user'.

**Ação:** POST `/api/v1/movies` .

**Resultado esperado:**

- Acesso negado, HTTP 403
- Mensagem: "Acesso restrito a administradores"

**2.6 Criar filme sem autenticação (POST)**

**Pré-condição:** Requisição sem token.

**Ação:** POST `/api/v1/movies` .



**Resultado esperado:**

- Não autorizado, HTTP 401
- Mensagem: "Token não fornecido"

**2.7 Atualizar filme como admin (PUT)**

**Pré-condição:** Filme existe, usuário é admin.

**Ação:** PUT `/api/v1/movies/:id` com novos dados.

**Resultado esperado:**

- Filme atualizado, HTTP 200
- Alterações refletidas no banco

**2.8 Deletar filme como admin (DELETE)**

**Pré-condição:** Filme existe sem sessões ativas.

**Ação:** DELETE `/api/v1/movies/:id`.

**Resultado esperado:**

- Filme removido, HTTP 200
  - Filme não aparece mais na listagem
- 

**3. SESSÕES (SESSIONS)****3.1 Listar todas as sessões (GET)**

**Pré-condição:** Sessões cadastradas.

**Ação:** GET `/api/v1/sessions`.

**Resultado esperado:**

- Lista de sessões, HTTP 200
- Dados incluem filme, teatro, horário, preço

**Cenários de borda:**

- Filtrar por filme
- Filtrar por data
- Ordenar por horário

**3.2 Buscar sessão por ID (GET)**

**Pré-condição:** Sessão existe.

**Ação:** GET `/api/v1/sessions/:id` .

**Resultado esperado:**

- Detalhes da sessão, HTTP 200
- Assentos disponíveis mostrados
- Informações completas do filme e teatro

**3.3 Criar sessão como admin (POST)**

**Pré-condição:** Admin autenticado, filme e teatro existem.

**Ação:** POST `/api/v1/sessions` com movie\_id, theater\_id, date, time, price.

**Resultado esperado:**

- Sessão criada, HTTP 201
- Validação de conflito de horário

**Cenários de borda:**

- Horário no passado
- Sessão duplicada (mesmo filme, teatro e horário)
- Preço = 0 ou negativo

**Mensagens de erro esperadas:**

- "Conflito de horário detectado"
- "Data/hora não pode ser no passado"
- "Preço deve ser maior que zero"

**3.4 Criar sessão com conflito de horário (POST)**

**Pré-condição:** Já existe sessão naquele teatro e horário.

**Ação:** POST com dados conflitantes.

**Resultado esperado:**

- Criação negada, HTTP 400
- Mensagem: "Já existe sessão agendada neste horário"

---

## 4. TEATROS (THEATERS)

**4.1 Listar todos os teatros (GET)**

**Pré-condição:** Teatros cadastrados.

**Ação:** GET `/api/v1/theaters`.

**Resultado esperado:**

- Lista de teatros, HTTP 200
- Dados incluem nome, capacidade, tipo

**4.2 Criar teatro como admin (POST)**

**Pré-condição:** Admin autenticado.

**Ação:** POST `/api/v1/theaters` com name, capacity, type.

**Resultado esperado:**

- Teatro criado, HTTP 201
- Capacidade validada (>0)

**Cenários de borda:**

- Capacidade = 0 ou negativa
- Nome duplicado
- Tipo inválido

**4.3 Atualizar capacidade do teatro (PUT)**

**Pré-condição:** Teatro existe.

**Ação:** PUT `/api/v1/theaters/:id` com nova capacidade.

**Resultado esperado:**

- Teatro atualizado, HTTP 200
- Nova capacidade refletida

---

## 5. RESERVAS (RESERVATIONS)

**5.1 Criar reserva para sessão disponível (POST)**

**Pré-condição:** Usuário autenticado, sessão existe, assentos disponíveis.

**Ação:** POST `/api/v1/reservations` com session\_id, seats (array de números).

**Resultado esperado:**

- Reserva criada, HTTP 201
- Assentos marcados como ocupados
- Total calculado corretamente

### **Cenários de borda:**

- Múltiplos assentos (2, 5, 10)
- Assento no limite da capacidade
- Último assento disponível

#### **5.2 Criar reserva para assento ocupado (POST)**

**Pré-condição:** Assento já reservado.

**Ação:** POST tentando reservar assento indisponível.

#### **Resultado esperado:**

- Reserva negada, HTTP 400
- Mensagem: "Assento X já está ocupado"

#### **5.3 Listar minhas reservas (GET)**

**Pré-condição:** Usuário autenticado com reservas.

**Ação:** GET `/api/v1/reservations/me`.

#### **Resultado esperado:**

- Lista de reservas do usuário, HTTP 200
- Ordenadas por data (mais recentes primeiro)
- Status de cada reserva (confirmada, cancelada)

#### **5.4 Listar todas reservas como admin (GET)**

**Pré-condição:** Admin autenticado.

**Ação:** GET `/api/v1/reservations`.

#### **Resultado esperado:**

- Todas as reservas do sistema, HTTP 200
- Filtros por status, data, filme

#### **5.5 Cancelar reserva própria (DELETE)**

**Pré-condição:** Usuário tem reserva ativa.

**Ação:** DELETE `/api/v1/reservations/:id`.

#### **Resultado esperado:**

- Reserva cancelada, HTTP 200
- Assentos liberados para nova reserva

- Status alterado para 'cancelada'

#### 5.6 Tentar cancelar reserva de outro usuário (DELETE)

**Pré-condição:** Reserva pertence a outro usuário.

**Ação:** DELETE `/api/v1/reservations/:id`.

**Resultado esperado:**

- Acesso negado, HTTP 403
  - Mensagem: "Você não tem permissão para cancelar esta reserva"
- 

## 6. TESTES WEB (END-TO-END)

### 6.1 Realizar login com sucesso

**Páginas:** Login → Home

**Ação:** Preencher e-mail e senha válidos, clicar em "Entrar".

**Resultado esperado:**

- Redirecionamento para a home
- Nome do usuário visível no header
- Menu "Minhas Reservas" disponível

### 6.2 Realizar login com credenciais inválidas

**Páginas:** Login

**Ação:** Preencher credenciais incorretas.

**Resultado esperado:**

- Mensagem de erro exibida: "E-mail ou senha incorretos"
- Permanecer na tela de login
- Campos não limpos automaticamente

### 6.3 Realizar logout

**Páginas:** Qualquer → Login

**Ação:** Clicar no botão "Sair" no menu.

**Resultado esperado:**

- Redirecionamento para login
- Token removido do localStorage

- Acesso a rotas protegidas negado

#### 6.4 Registro de novo usuário

**Páginas:** Register → Home

**Ação:** Preencher formulário de cadastro com dados válidos.

**Resultado esperado:**

- Cadastro realizado
- Login automático
- Redirecionamento para home

#### 6.5 Visualizar lista de filmes

**Páginas:** Home

**Ação:** Acessar página inicial.

**Resultado esperado:**

- Todos filmes exibidos em cards
- Informações visíveis: título, gênero, rating, duração
- Imagens carregadas corretamente

#### 6.6 Acessar detalhes de um filme

**Páginas:** Home → MovieDetail

**Ação:** Clicar em um card de filme.

**Resultado esperado:**

- Página de detalhes carregada
- Informações completas do filme
- Sessões disponíveis listadas
- Botão "Reservar" visível

#### 6.7 Buscar filme por nome

**Páginas:** Home

**Ação:** Digitar nome do filme no campo de busca.

**Resultado esperado:**

- Resultados filtrados em tempo real
- Apenas filmes correspondentes exibidos

- Mensagem "Nenhum filme encontrado" se não houver resultado

#### 6.8 FLUXO COMPLETO: Reservar ingresso do início ao fim

**Páginas:** Login → Home → MovieDetail → SeatSelection → Confirmation → Minhas Reservas

##### **Ação:**

1. Fazer login com usuário válido
2. Navegar até a home
3. Selecionar um filme
4. Escolher uma sessão disponível
5. Selecionar 2 assentos livres
6. Confirmar a reserva
7. Verificar reserva em "Minhas Reservas"

##### **Resultado esperado:**

- Fluxo completo sem erros
- Reserva criada e visível
- Assentos marcados como ocupados
- Total calculado corretamente
- E-mail de confirmação (se implementado)

#### 6.9 Visualizar minhas reservas

**Páginas:** Minhas Reservas

**Ação:** Acessar menu "Minhas Reservas".

##### **Resultado esperado:**

- Lista de todas as reservas do usuário
- Informações: filme, sessão, assentos, status
- Botão "Cancelar" para reservas ativas

#### 6.10 Cancelar uma reserva

**Páginas:** Minhas Reservas

**Ação:** Clicar em "Cancelar" em uma reserva ativa.

##### **Resultado esperado:**

- Modal de confirmação exibido
- Após confirmar, reserva removida/marcada como cancelada

- Mensagem de sucesso

#### 6.11 Acessar dashboard admin

**Páginas:** Admin Dashboard (apenas admin)

**Ação:** Login com usuário admin, acessar menu admin.

**Resultado esperado:**

- Dashboard visível apenas para admin
- Usuário comum não vê opção no menu
- Redirect se usuário comum tentar acessar URL direta

#### 6.12 Criar novo filme (Admin)

**Páginas:** Admin → Movies → Create

**Ação:** Preencher formulário de criação de filme.

**Resultado esperado:**

- Filme criado com sucesso
- Listado na página de gerenciamento
- Visível na home para usuários

#### 6.13 Editar filme existente (Admin)

**Páginas:** Admin → Movies → Edit

**Ação:** Alterar dados de um filme e salvar.

**Resultado esperado:**

- Alterações salvas
- Mudanças refletidas imediatamente
- Histórico de alteração (se implementado)

#### 6.14 Criar nova sessão (Admin)

**Páginas:** Admin → Sessions → Create

**Ação:** Preencher formulário: filme, teatro, data, hora, preço.

**Resultado esperado:**

- Sessão criada
  - Validação de conflito de horário
  - Sessão disponível para reserva
-



## PRIORIZAÇÃO DA EXECUÇÃO DOS CENÁRIOS

Cenário	Prioridade	Justificativa
<b>Criar usuário com dados válidos (API)</b>	 <b>P0 - Crítica</b>	Fluxo principal de onboarding; sem isso o usuário não acessa o sistema
<b>Login com credenciais válidas (API)</b>	 <b>P0 - Crítica</b>	Autenticação é requisito para 90% das funcionalidades
<b>Criar reserva para sessão disponível (API)</b>	 <b>P0 - Crítica</b>	Core business do sistema; falha impacta diretamente a receita
<b>Listar todos os filmes (API)</b>	 <b>P0 - Crítica</b>	Funcionalidade principal de navegação
<b>Fluxo completo de reserva (Web)</b>	 <b>P0 - Crítica</b>	Jornada end-to-end mais importante para o usuário
<b>Login web com sucesso</b>	 <b>P0 - Crítica</b>	Gateway para todas as funcionalidades web
<b>Criar filme como admin (API)</b>	 <b>P1 - Alta</b>	Essencial para operação do cinema, mas usado com menor frequência
<b>Listar minhas reservas (API/Web)</b>	 <b>P1 - Alta</b>	Usuário precisa consultar histórico; impacta experiência e satisfação
<b>Criar sessão como admin (API)</b>	 <b>P1 - Alta</b>	Sem sessões não há reservas; criação é pontual
<b>Buscar filme por ID (API)</b>	 <b>P1 - Alta</b>	Usado em detalhamento; importante, mas não bloqueia navegação
<b>Criar reserva para assento ocupado (API)</b>	 <b>P2 - Média</b>	Validação de regra de negócio; edge case relevante
<b>Cancelar reserva própria (API/Web)</b>	 <b>P2 - Média</b>	Funcionalidade secundária, mas esperada pelo usuário

Atualizar filme como admin (API)	 P2 - Média	Manutenção administrativa; não bloqueia operação principal
Acessar dashboard admin (Web)	 P2 - Média	Função administrativa; uso restrito
Criar sessão com conflito de horário (API)	 P3 - Baixa	Edge case; validação útil, mas situação rara
Deletar filme como admin (API)	 P3 - Baixa	Operação rara e de alto risco; execução pontual
Buscar filme por ID inválido (API)	 P3 - Baixa	Validação de erro; importante, mas não crítico

MATRIZ DE RISCO

Cenário	Impacto	Probabilidade	Classificação
Criar usuário com dados válidos	Alto	Alta	Crítico
Login com credenciais válidas	Alto	Alta	Crítico
Criar reserva (sessão disponível)	Alto	Alta	Crítico
Fluxo completo de reserva (Web)	Alto	Alta	Crítico
Listar todos os filmes	Alto	Alta	Crítico
Criar filme como admin	Alto	Média	Alto
Criar sessão como admin	Alto	Média	Alto
Listar minhas reservas	Médio	Alta	Alto
Cancelar reserva própria	Médio	Média	Médio

Criar reserva (assento ocupado)	Médio	Média	Médio
Buscar filme por ID inválido	Baixo	Baixa	Baixo
Deletar filme como admin	Médio	Baixa	Baixo

COBERTURA DE TESTES (Backend + Frontend)

Cenário	Endpoint / Fluxo	Cobertura
Criar usuário válido	POST /api/v1/auth/registe r	Cadastro completo; valida campos obrigatórios e formato de e-mail
Criar usuário inválido	POST /api/v1/auth/registe r	Validação de regras; testa e-mail duplicado, senha fraca, campos vazios
Login válido	POST /api/v1/auth/login	Autenticação JWT; valida token gerado e dados retornados
Login inválido	POST /api/v1/auth/login	Valida mensagens de erro específicas; credenciais incorretas
Obter perfil autenticado	GET /api/v1/auth/me	Valida retorno de dados do usuário; testa token no header
Obter perfil sem token	GET /api/v1/auth/me	Valida segurança; acesso negado sem autenticação
Listar filmes	GET /api/v1/movies	Listagem completa; valida estrutura de resposta e dados
Buscar filme por ID	GET /api/v1/movies/:id	Busca específica; valida dados completos e sessões relacionadas

<b>Criar filme (admin)</b>	POST /api/v1/movies	CRUD admin; valida permissões e criação com dados válidos
<b>Criar filme (user)</b>	POST /api/v1/movies	Valida controle de acesso; usuário comum não pode criar
<b>Atualizar filme (admin)</b>	PUT /api/v1/movies/:id	Edição de dados; valida persistência de alterações
<b>Deletar filme (admin)</b>	DELETE /api/v1/movies/:id	Remoção; valida que filme não aparece mais na listagem
<b>Listar sessões</b>	GET /api/v1/sessions	Lista sessões disponíveis; valida dados de filme, teatro e horário
<b>Buscar sessão por ID</b>	GET /api/v1/sessions/:id	Detalhes da sessão; valida assentos disponíveis
<b>Criar sessão (admin)</b>	POST /api/v1/sessions	Agendamento; valida conflito de horário e validações de data
<b>Listar teatros</b>	GET /api/v1/theaters	Lista salas; valida capacidade e tipos
<b>Criar teatro (admin)</b>	POST /api/v1/theaters	CRUD de salas; valida capacidade mínima
<b>Criar reserva válida</b>	POST /api/v1/reservations	Core business; valida criação, cálculo de total e ocupação de assentos
<b>Criar reserva (assento ocupado)</b>	POST /api/v1/reservations	Validação de regra; testa concorrência e mensagens de erro
<b>Listar minhas reservas</b>	GET /api/v1/reservations/me	Histórico pessoal; valida filtro por usuário autenticado

<b>Listar todas reservas (admin)</b>	GET /api/v1/reservations	Gestão admin; valida acesso total e filtros
<b>Cancelar reserva própria</b>	DELETE /api/v1/reservations/:id	Cancelamento; valida liberação de assentos e mudança de status
<b>Login Web</b>	Web: Login → Home	Fluxo de autenticação web; valida redirecionamento e sessão
<b>Registro Web</b>	Web: Register → Home	Onboarding web; valida formulário e login automático
<b>Visualizar filmes (web)</b>	Web: Home	Navegação principal; valida exibição de cards e dados
<b>Detalhes do filme (web)</b>	Web: Home → MovieDetail	Navegação de detalhes; valida informações completas e sessões
<b>Fluxo completo de reserva (E2E)</b>	Web: Login → Home → Detail → Seats → Confirm → MyReservations	Jornada ponta a ponta; valida experiência do usuário e integração total