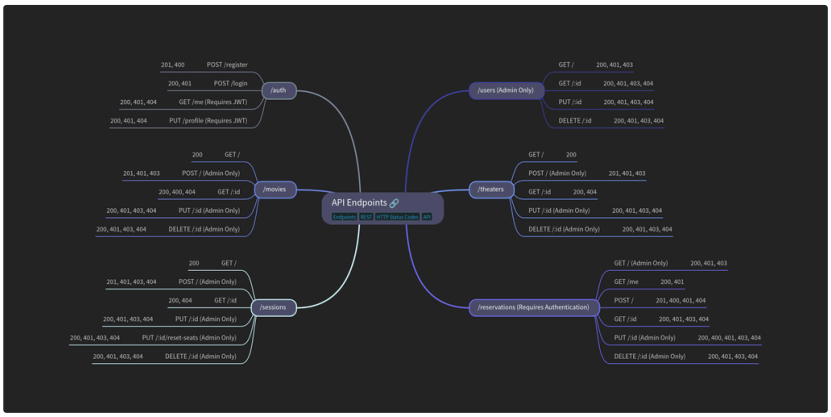
Plano de Testes: Cinema App API

1. Introdução

Este plano de testes visa garantir a qualidade da **Cinema App API**, uma API RESTful projetada para um sistema de reserva de ingressos de cinema. A API gerencia a autenticação de usuários, o catálogo de filmes, as salas de cinema, as sessões e as reservas.



(Mapa mental da API; incluíndo endpoints, requests, status code e rotas, compilados e organizados para melhor visualização e aproveitamento. Clique aqui para expandir.)

---

2. Objetivo Geral

Estabelecer uma metodologia para validar a qualidade da Cinema App API, garantindo que todas as funcionalidades implementadas estejam corretas, seguras e alinhadas com as histórias de usuário e os requisitos de negócio. O plano tem como propósito identificar falhas, validar regras de negócio e apoiar a entrega de um produto confiável.

2.1 Escopo dos Testes

Dentro do Escopo:

- **Testes Funcionais e de Integração:** Validação de todas as rotas da API, incluindo as operações de CRUD para Autenticação, Usuários, Filmes, Sessões e Reservas, assim como a interação entre esses módulos.
- **Regras de Negócio e Segurança:** Verificação das regras de negócio (ex: unicidade de e-mail, status dos assentos) , do fluxo de autenticação com token JWT e do controle de acesso por permissões (usuário comum vs. admin).

Fora do Escopo:

- Testes de interface do usuário (frontend), testes de performance e carga, testes de usabilidade e validação da infraestrutura (servidor, banco de dados).

---

3. Análise dos Testes

3.1 Base para Testes

A criação e priorização dos cenários de teste serão baseadas nos seguintes artefatos:

- **Documentação Swagger**: Conforme definido em `src/config/swagger.js`, usada para mapear rotas, parâmetros, e estruturas de requisição/resposta.
- **Histórias de Usuário**: O arquivo `USER-STORIES.md` fornece os critérios de aceitação e os fluxos esperados do ponto de vista do usuário.
- **Modelos de Dados**: A estrutura definida em `src/models` (User, Movie, Theater, Session, Reservation) orienta a criação de dados de teste válidos e inválidos.

3.2 Cobertura de Testes

A cobertura estimada visa atingir **100% de validação dos fluxos principais e alternativos** definidos nas Histórias de Usuário e rotas da API, com um foco rigoroso nas regras de negócio, autenticação e controle de permissões.

- **Testes Funcionais**: Validam o comportamento esperado das rotas (status code, corpo da resposta) para os fluxos de sucesso.
- **Testes Negativos**: Avaliam a resposta da API a entradas inválidas, como campos obrigatórios ausentes, dados mal formatados ou violação de regras de negócio.
- **Testes de Segurança**: Asseguram que rotas protegidas só possam ser acessadas por usuários autenticados e com a devida autorização (role `admin` vs. `user`). Validam o tratamento de tokens inválidos ou expirados.

- **Testes de Integração**: Verificam a interação entre diferentes módulos, como a tentativa de excluir um filme que possui sessões ativas ou cancelar uma reserva e verificar se os assentos voltam a ficar disponíveis.

---

4. Técnicas Aplicadas

- **Teste Funcional**: Execução de requisições via Postman para validar as funcionalidades de cada endpoint conforme a documentação Swagger.
- **Teste de Fluxo**: Validação de jornadas completas do usuário, como: registrar, fazer login, buscar um filme, ver sessões e criar uma reserva.
- **Validação de Limite**: Análise dos limites das regras de negócio, como o número mínimo de caracteres para senhas ou a seleção de um assento já ocupado.
- **Teste de Segurança**: Foco na autenticação (geração e validação de token) e autorização (acesso baseado em papéis de usuário).

---

5. Cenários de Teste Planejados

A seguir, são apresentados os cenários de teste definidos para cada funcionalidade principal da API. Eles foram elaborados com base nas especificações e no comportamento esperado da aplicação, visando validar as respostas da API, os códigos de status, a estrutura dos dados e as regras de negócio envolvidas.

5.1 Autenticação (US-AUTH)

User Story	Cenário	Ação	Resultado Esperado
US-AUTH-001	Registrar novo usuário com dados válidos	`POST /auth/register` com nome, email único e senha	Status 201, retorna dados do usuário e token
US-AUTH-001	Tentar registrar usuário com e-mail já existente	`POST /auth/register` com e-mail duplicado	Status 400, mensagem "User already exists"
US-AUTH-002	Realizar login com credenciais válidas	`POST /auth/login` com e-mail e senha corretos	Status 200, retorna dados do usuário e token
US-AUTH-002	Tentar realizar login com senha incorreta	`POST /auth/login` com senha errada	Status 401, mensagem "Invalid email or password"
US-AUTH-004	Obter perfil do usuário autenticado	`GET /auth/me` com token válido	Status 200, retorna dados do usuário logado
US-AUTH-004	Tentar obter perfil sem autenticação	`GET /auth/me` sem token de autorização	Status 401, mensagem de não autorizado
US-AUTH-004	Atualizar nome do perfil do usuário	`PUT /auth/profile` com um novo nome	Status 200, retorna dados atualizados e novo token

5.2 Filmes (US-MOVIE)

User Story	Cenário	Ação	Resultado Esperado
US-MOVIE-001	Listar todos os filmes (público)	`GET /movies`	Status 200, retorna lista de filmes
US-MOVIE-002	Obter detalhes de um filme por ID (público)	`GET /movies/{id}` com ID válido	Status 200, retorna dados do filme
US-MOVIE-002	Tentar obter filme com ID inexistente	`GET /movies/{id}` com ID inválido	Status 404, mensagem "Movie not found"
Admin	Criar novo filme (Admin)	`POST /movies` com dados válidos e token de admin	Status 201, retorna dados do filme criado
Admin	Tentar criar filme sem ser admin	`POST /movies` com token de usuário comum	Status 403, mensagem de não autorizado
Admin	Excluir um filme (Admin)	`DELETE /movies/{id}` com ID válido e token de admin	Status 200, mensagem "Movie removed"
Admin	Tentar excluir um filme como usuário comum	`DELETE /movies/{id}` com token de usuário comum	Status 403, mensagem de não autorizado

5.3 Sessões (US-SESSION)

User Story	Cenário	Ação	Resultado Esperado
US-SESSION-001	Listar todas as sessões (público)	`GET /sessions`	Status 200, retorna lista de sessões
US-SESSION-001	Filtrar sessões por filme	`GET /sessions?movie={movieId}`	Status 200, retorna sessões apenas para o filme
US-SESSION-001	Obter detalhes de uma sessão por ID	`GET /sessions/{id}` com ID válido	Status 200, retorna dados da sessão e assentos
Admin	Criar nova sessão (Admin)	`POST /sessions` com dados válidos e token de admin	Status 201, retorna sessão criada com assentos
Admin	Tentar criar uma sessão como usuário comum	`POST /sessions` com dados válidos e token de usuário comum	Status 403, mensagem de não autorizado
Admin	Tentar criar sessão para filme inexistente	`POST /sessions` com `movieId` inválido	Status 404, mensagem "Movie not found"
Admin	Resetar assentos de uma sessão (Admin)	`PUT /sessions/{id}/reset-seats` com token de admin	Status 200, assentos da sessão com status "available"

5.4 Reservas (US-RESERVE)

User Story	Cenário	Ação	Resultado Esperado
US-RESERVE-001	Criar reserva com assentos disponíveis	`POST /reservations` com token de usuário, `sessionId` e assentos "available"	Status 201, retorna dados da reserva
US-RESERVE-001	Tentar criar reserva para assento ocupado	`POST /reservations` com assentos "occupied"	Status 400, mensagem de assentos indisponíveis
US-RESERVE-001	Tentar criar reserva sem autenticação	`POST /reservations` sem token	Status 401, erro de não autorizado
US-RESERVE-003	Listar "Minhas Reservas"	`GET /reservations/my` com token de usuário	Status 200, retorna lista de reservas do usuário
Admin	Cancelar/Excluir uma reserva (Admin)	`DELETE /reservations/{id}` com token de admin	Status 200, mensagem "Reservation removed"
Admin	Validar se assentos ficam disponíveis após cancelamento	Após `DELETE /reservations/{id}`, verificar `GET /sessions/{sessionId}`	Status 200, os assentos da reserva cancelada devem estar como "available"

5.5 Salas (THEATERS)

User Story	Cenário	Ação	Resultado Esperado
N/A	Listar todas as salas (público)	`GET /theaters`	Status 200, retorna lista de salas
N/A	Obter detalhes de uma sala por ID (público)	`GET /theaters/{id}` com ID válido	Status 200, retorna dados da sala
Admin	Criar nova sala (Admin)	`POST /theaters` com dados válidos e token de admin	Status 201, retorna dados da sala criada
Admin	Tentar criar sala como usuário comum	`POST /theaters` com token de usuário comum	Status 403, acesso negado
Admin	Excluir uma sala (Admin)	`DELETE /theaters/{id}` com ID válido e token de admin	Status 200, mensagem "Theater removed"

5.6 Usuários (USERS)

User Story	Cenário	Ação	Resultado Esperado
Admin	Listar todos os usuários (Admin)	`GET /users` com token de admin	Status 200, retorna lista de usuários
Admin	Tentar listar usuários como usuário comum	`GET /users` com token de usuário comum	Status 403, acesso negado
Admin	Obter detalhes de um usuário por ID (Admin)	`GET /users/{id}` com token de admin	Status 200, retorna dados do usuário (sem senha)
Admin	Atualizar dados de um usuário (Admin)	`PUT /users/{id}` com novos dados e token de admin	Status 200, dados atualizados retornados
Admin	Excluir um usuário (Admin)	`DELETE /users/{id}` com token de admin	Status 200, mensagem "User removed"

---

6. Matriz de Risco

Risco	Probabilidade	Impacto	Mitigação
Acesso não autorizado a rotas de admin	Média	Alto	Testar todas as rotas de admin (`/users` , `POST /movies` , etc.) com um token de usuário comum para garantir que o acesso seja negado (Status 403).
Falha ao atualizar o status dos assentos	Média	Alto	Criar cenários que validem a mudança de status do assento para "occupied" após uma reserva e para "available" após o cancelamento.
Inconsistência de dados ao excluir entidades	Baixa	Alto	Testar a exclusão de um filme ou cinema que tenha sessões associadas. Embora não implementado, é um risco a ser registrado.
Tratamento falho de tokens expirados	Alta	Alto	Tentar acessar rotas protegidas (`/auth/me` , `/reservations/my`) com um token JWT expirado para garantir que a API retorne Status 401.
Criação de reserva com assentos conflitantes	Média	Alto	Simular duas requisições simultâneas para os mesmos assentos (se possível) ou validar que a API rejeita uma reserva para assentos já ocupados.
Validação fraca de dados de entrada	Média	Médio	Testar a criação de entidades (usuários, filmes) com campos obrigatórios ausentes ou com formatos inválidos (ex: email mal formatado).

---

7. Testes Candidatos à Automação

A seleção de testes para automação é uma etapa estratégica que visa otimizar o processo de QA, garantindo feedback rápido e confiável sobre a saúde da aplicação. Os cenários listados abaixo foram escolhidos com base em critérios de **criticidade, repetitividade e impacto no negócio**. Estes testes formam a espinha dorsal da nossa suíte de regressão, sendo executados frequentemente para validar que novas implementações não introduziram falhas em funcionalidades existentes. A automação desses fluxos permite que a equipe de testes manuais se concentre em cenários mais complexos e exploratórios.

Cenários Automatizados com Robot Framework:

Usuários e Autenticação

- Criar usuário com dados válidos.
- Tentar criar usuário com e-mail duplicado.
- Realizar login com credenciais válidas.
- Tentar realizar login com credenciais inválidas.
- Acessar rota protegida (`/auth/me`) com token válido.
- Tentar acessar rota protegida sem token.

Filmes e Sessões (Admin)

- Criar um novo filme com token de admin.
- Tentar criar um filme com token de usuário comum.
- Criar uma nova sessão para um filme e sala existentes.

Reservas (Fluxo do Usuário)

- Criar uma reserva para uma sessão com assentos disponíveis.
- Tentar criar uma reserva para uma sessão com assentos já ocupados.
- Listar as reservas do usuário logado.
- Cancelar uma reserva (como admin, para teste) e verificar se os assentos foram liberados.

Gerenciamento de Usuários (Admin)

- Listar todos os usuários com token de admin.
- Tentar listar todos os usuários com token de usuário comum.

Gerenciamento de Salas (Admin)

- Criar uma nova sala com token de admin.
- Tentar criar uma nova sala com token de usuário comum.

---

8. Estrutura de Testes

A automação com Robot Framework será organizada da seguinte forma para garantir manutenibilidade e reuso:

```
/cinema-api-tests
|
|--- tests/
|   |--- test_authentication.robot
|   |--- test_movies.robot
|   |--- test_sessions.robot
|   |--- test_reservations.robot
|   |--- test_permissions.robot
|
|--- keywords/
|   |--- auth_keywords.robot
|   |--- movie_keywords.robot
|   |--- session_keywords.robot
|   |--- reservation_keywords.robot
|
|--- resources/
|   |--- base.robot
|
|--- variables/
|   |--- variables.robot
```

- **tests/**: Contém os arquivos de teste com os cenários de alto nível.
- **keywords/**: Contém as implementações de palavras-chave reutilizáveis (ex: `Create User`, `Login`).
- **resources/**: Arquivos de configuração base, importando bibliotecas e keywords.
- **variables/**: Centraliza variáveis como URL base, credenciais de teste e IDs.

---

9. Ferramentas de Teste

- **Robot Framework**: Ferramenta principal para automação dos testes de regressão e dos fluxos críticos da API.
- **Postman**: Utilizado para testes manuais exploratórios, validação de novos endpoints e depuração rápida.
- **Swagger**: Fonte de documentação para consulta de rotas, parâmetros e modelos de dados durante a criação dos testes.
- **Jira**: Para registrar e rastrear bugs, vinculando-os aos cenários de teste correspondentes.
- **QAlity**: Para gerenciar e documentar os casos de teste manuais, registrar evidências e controlar os resultados.

---

10. Conclusão

Este plano de testes fornece uma abordagem estruturada para validar a Cinema App API, cobrindo os principais fluxos funcionais, cenários de erro, segurança e integração. A combinação de testes manuais exploratórios e uma suíte de testes automatizados com Robot Framework garantirá a detecção precoce de falhas e a entrega de um produto estável e de alta qualidade. A estrutura de automação proposta visa a escalabilidade e a fácil manutenção, permitindo que a equipe de QA acompanhe o desenvolvimento contínuo da API de forma eficiente.