Plano de Testes - Cinema

1. Identificação do Plano de Testes

- Nome do projeto: Cinema
- Responsável: Cassia Yumi Iwamoto Basso
- Data de criação: 23/06/2025

Data	Revisão
26/06/2025	Adicionado número dos casos de testes

2. Apresentação

Este documento apresenta o planejamento de testes da API Cinema, que é uma RESTful API para reservas de tickets de cinema ticket criada com Node.js, Express, and MongoDB e uma interface Web responsiva.

Serão realizados testes tanto no Backend quanto no Frontend:

- O backend será testado utilizando o Postman, com foco nos endpoints, autenticação JWT, consistência dos dados e validações de entradas, de acordo com as histórias de usuário apresentadas.
- O frontend utilizará o Robot Framework, com a finalidade de garantir que os fluxos funcionais principais (login, registro, seleção de assentos, e outras funcionalidades) atendam às necessidades dos usuários.

3. Escopo dos Testes

O escopo inclui as funcionalidades descritas nas histórias de usuário, em especial:

✔️**Autenticação (registro, login, logout, perfil).**

✔️**Navegação e visualização de filmes.**

✔️**Horários de sessão.**

✔️**Reservas de assentos e checkout.**

✔️**Consulta das reservas realizadas.**

✔️**Navegabilidade da interface.**

❌ Fora do escopo: Testes de integração externa de pagamento, Envio de e-mails, Teste de carga, Funcionalidades administrativas não acessíveis.

4. Funcionalidades a serem testadas

✅Registro de usuário (US-AUTH-001)

✅Login/logout (US-AUTH-002 e US-AUTH-003)

✅Gerenciamento de perfil (US-AUTH-004)

✅Home e exibição de filmes (US-HOME-001, US-MOVIE-001)

✅Detalhes de filmes (US-MOVIE-002)

✅Sessões (US-SESSION-001)

✅Seleção e reserva de assentos (US-RESERVE-001, US-RESERVE-002)

✅Histórico de reservas (US-RESERVE-003)

✅Navegação geral da aplicação (US-NAV-001)

5. Estratégia de Testes

Backend (Postman):

Serão aplicados os seguintes testes:

- Testes de métodos: GET, POST, PUT e DELETE.
- Testes de autenticação com JWT.
- Teste de validação de dados (campos obrigatórios, tipos).
- Testes de fluxo de reserva (selecionar assentos, checkout, bloquear assentos reservados).

Em relação à autenticação e controle e acesso serão validados o perfil admin para rotas protegidas, verificação de restrições de acesso sem token e testes com token expirado ou ausente.

Frontend (Robot Framework):

Serão aplicados os seguintes testes:

- Testes manuais.
- Validação de navegação entre telas.
- Verificação visual de mensagens, feedbacks e responsividade.
- Fluxos de usuário: registro, login, perfil, reserva, logout.
- Teste de interface para módulos “Filmes em cartaz”, “Detalhes do filme”, “Minhas reservas”.

Será realizado a análise da cobertura com base no Path Coverage, utilizando-se o seguinte critério:

- Total de URIs disponíveis: X
- URIs testadas: Y
- **Cobertura**: Path Coverage= (Y / X) * 100%

Será realizada a análise dos riscos, com foco em:

- Falhas em login, seleção de filmes ou processo de reserva.
- Utilização de matriz de risco para priorização dos testes.

Serão analisados os testes candidatos à automação com base em repetição, criticidade e estabilidade da funcionalidade.

Os dados de testes utilizados poderão ser usados da seguinte forma:

- Usuários com diferentes permissões (visitante, usuário autenticado, administrador)
- Filmes com diferentes classificações, datas e gêneros
- Tokens válidos e inválidos
- Carrinhos com múltiplos produtos ou sessões simultâneas

6. Critérios de Aceitação dos Testes

🟢Todos os endpoints devem retornar códigos HTTP apropriados;

🟢Validações de dados devem estar presentes (ex: formato de e-mail, campos obrigatórios);

🟢 Interface deve exibir mensagens claras em casos de erro ou sucesso;

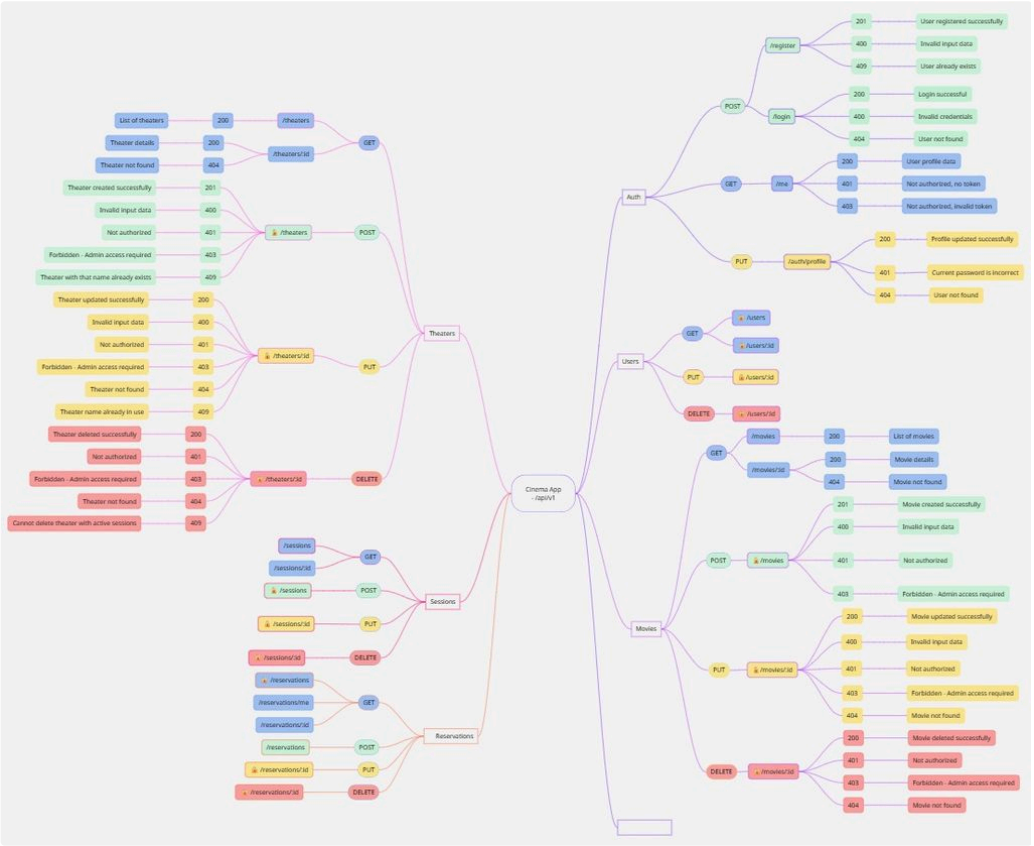
🟢Tokens

7. Técnicas aplicadas

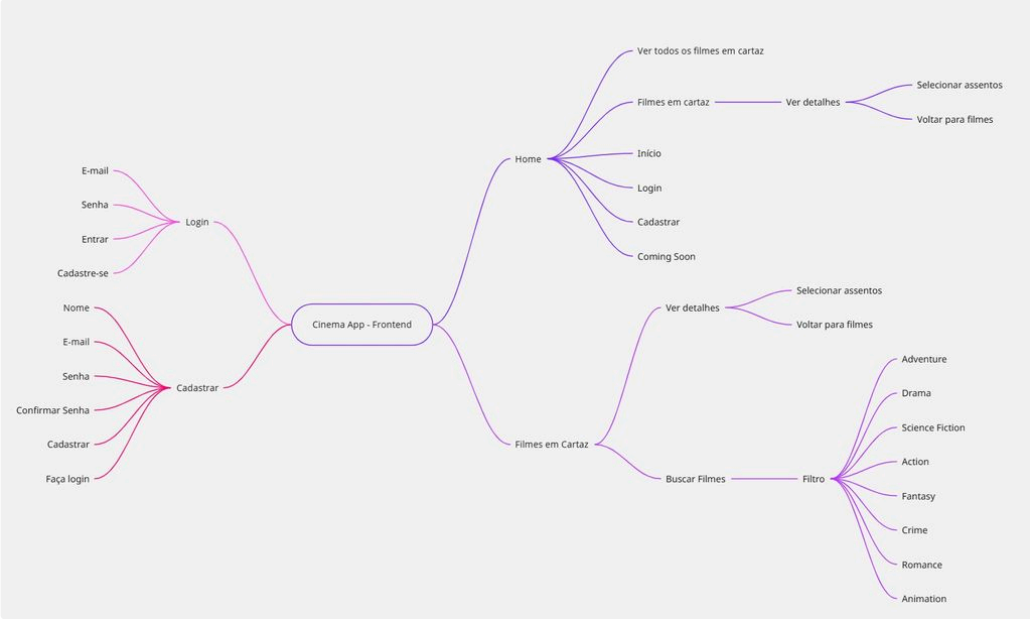
Com base na documentação da API e nas users stories disponibilizadas, foram selecionadas as seguintes técnicas de teste:

- Partição de Equivalência: testes de entradas válidas e inválidas;
- Análise de Valor Limite: verificação de limites mínimos e máximos das entradas;
- Teste de Valores Nulos e Vazios
- Testes baseados em regras de negócio: ex: reserva só com login, bloqueio de assentos;
- Testes exploratórios: visando encontrar comportamento não previstos nos requisitos ou na documentação;
- Técnica de transição de estados: especialmente quanto ao token, autenticação e navegação da sessão.

8. Mapa Mental da aplicação



Mapa mental - backend



Mapa mental - Frontend

9. Cenários de Testes Planejados

Com base nas users stories apresentadas e na documentação da API ServeRest foram planejados os seguintes cenários de testes:

1️⃣US-AUTH-001

Cenário de Teste US-AUTH-001

Descrição: Realizar o cadastro de um novo usuário com sucesso no Cinema App.

Pré-condições:

1. O usuário ainda **não está cadastrado** na base de dados.
2. O formulário de registro está disponível e acessível.
3. O endpoint `/users` (método POST) está ativo e funcional.

Condições:

- Campos obrigatórios: `nome`, `email`, `senha`;
- O sistema **deve validar** o formato do e-mail;
- O sistema **não deve permitir** registro com e-mails duplicados;
- O sistema **deve validar** a complexidade mínima da senha (por exemplo: mínimo 6 caracteres);
- Em caso de sucesso, o sistema **deve criar o usuário** e redirecionar para a página de login (no frontend);
- O backend **deve retornar HTTP 201 Created** em caso de sucesso;
- O backend **deve retornar HTTP 400 Bad Request** em caso de erros de validação.

Passo a passo:

1. Acessar a tela de **cadastro** no frontend;
2. Preencher os campos obrigatórios com dados válidos e enviar o formulário;
3. Verificar se a resposta da API é `201 Created`;
4. Validar que o usuário foi de fato persistido no banco;
5. Verificar se a interface redireciona para a tela de login;

6. Realizar novas tentativas com dados inválidos e verificar o comportamento do sistema.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
CIN-1 Registro com dados válidos	Cadastro com nome, e-mail e senha corretos	Preencher todos os campos e enviar formulário	Nome: Ana SilvaEmail: ana@teste.comSenha: Abc12345	HTTP 201 Created; redirecionamento para login	Frontend/API	1.0
CIN-2 Registro com e-mail já cadastrado	Testar bloqueio de e-mails duplicados	Enviar formulário com e-mail já existente	Email: ana@teste.com	HTTP 400 com mensagem "E-mail já cadastrado"	Frontend/API	1.0
CIN-3 Registro com e-mail inválido	Verificar validação de formato de e-mail	Enviar e-mail inválido no formulário	Email: teste.com	HTTP 400 com mensagem "E-mail inválido"	Frontend	1.0
CIN-4 Registro com senha fraca	Verificar se senha com menos de 6 caracteres é rejeitada	Enviar senha com poucos caracteres	Senha: 123	HTTP 400 com mensagem "Senha muito curta"	Frontend/API	1.0
CIN-5 Registro com nome vazio	Validação de campo obrigatório (nome)	Enviar formulário sem preencher o nome	Nome: ""	HTTP 400 com mensagem "Campo obrigatório"	Frontend/API	1.0
CIN-6 Registro com campos em branco	Testar envio de todos os campos vazios	Submeter formulário em branco	Nome, Email e Senha vazios	HTTP 400 com múltiplas mensagens de erro	Frontend	1.0
CIN-8 Resposta da API	Validar se resposta contém ID e e-mail após cadastro	Observar resposta JSON	-	JSON com propriedades `id`, `email`	API	1.0

2️⃣US-AUTH-002

Cenário de Teste US-AUTH-002

Descrição: Realizar autenticação com sucesso no Cinema App.

Pré-condição: Usuário previamente cadastrado e validado no sistema com nome, e-mail e senha corretos.

Condições:

- Visitantes (usuários não cadastrados) **não devem conseguir autenticar**;
- Usuários cadastrados com **senha incorreta não devem conseguir autenticar**;
- Falhas de autenticação devem retornar **HTTP 401 - Unauthorized**;
- Usuários **existentes com senha correta** devem ser autenticados com sucesso;
- A autenticação bem-sucedida deve **retornar um token do tipo Bearer**;

Passo a passo:

1. Enviar uma requisição HTTP **POST** para o endpoint `/login` com `email` e `password`;
2. Verificar se o **status da resposta** é `200 OK` (para sucesso) ou `401 Unauthorized` (para falha);
3. Validar a presença de um **token JWT** no corpo da resposta em caso de sucesso;
4. Confirmar que, após o login, o usuário é redirecionado à página inicial e vê o menu personalizado.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
CIN-10 Login com credenciais válidas	Verifica login com e-mail e senha corretos	Acessar login → Inserir credenciais → Clicar em Entrar	E-mail: ana@teste.com Senha: Teste123	Usuário é redirecionado para a página inicial	Frontend/API	1.0
Login com senha incorreta	Verifica resposta para senha inválida	Inserir e-mail correto com senha errada e tentar logar	E-mail: ana@teste.com Senha: Errada123	Mensagem "Credenciais inválidas" é exibida	Frontend/API	1.0
Login com campo vazio	Validação de formulário de login com campos vazios	Deixar campos em branco e submeter o formulário	E-mail:"" Senha:""	Validação impede envio; mensagens de campo obrigatório são exibidas	Frontend	1.0
Logout de usuário autenticado	Verifica se logout limpa sessão e	Fazer login → Clicar em Logout no menu	Sessão ativa com token válido	Token removido do localStorage; rotas	Frontend	1.0

	restringe acesso			protegidas não acessíveis		
Acesso ao perfil autenticado	Verifica se usuário logado acessa página de perfil	Fazer login → Navegar até página de perfil	Token JWT válido	Nome, e-mail e tipo de usuário são exibidos	Frontend/API	1.0
Atualização do nome no perfil	Testa edição de nome do usuário	Ir à página de perfil → Alterar nome → Salvar mudanças	Novo nome: Ana Oliveira	Nome é atualizado e mensagem de sucesso é exibida	Frontend/API	1.0
Acesso ao perfil sem autenticação	Tenta acessar /perfil sem estar logado	Acessar URL diretamente sem token JWT	-	Redirecionamento para login ou mensagem de acesso negado	Frontend/API	1.0

3️⃣US -AUTH-003

Cenário de Teste US-AUTH-003

Descrição: Encerrar a sessão de um usuário autenticado no Cinema App.

Pré-condições:

1. Usuário deve estar **logado/autenticado** com um token JWT válido armazenado no navegador (`localStorage` ou `sessionStorage`);
2. O menu de navegação com a opção de logout deve estar visível.

Condições:

- O botão ou link de logout deve estar disponível após o login;
- Ao acionar o logout, o **token deve ser removido** do `localStorage`;
- Após o logout, o **usuário não deve conseguir acessar rotas protegidas**;
- O sistema pode redirecionar o usuário para a **página de login ou home pública**;
- Nenhuma requisição futura deve conter o token anterior;
- A sessão deve ser considerada encerrada para efeitos de autorização.

Passo a passo:

1. Realizar login com e-mail e senha válidos;
2. Aguardar carregamento da interface autenticada;
3. Clicar na opção **"Logout"** no menu superior ou lateral;
4. Verificar se o token JWT foi removido do navegador;
5. Tentar acessar uma rota protegida, como `/perfil` ou `/minhas-reservas`;
6. Confirmar que o sistema impede o acesso (ex: redireciona para login ou mostra mensagem de acesso negado).

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Logout visível após login	Verificar se botão de logout aparece após autenticação	Fazer login e observar o menu	-	Opção de logout visível no menu	Frontend	1.0
Logout remove token	Verificar se token é removido ao sair	Logar → Clicar em Logout	-	`localStorage` não deve conter o token	Frontend	1.0
Acesso a rota protegida após logout	Verificar se usuário é impedido de acessar rotas privadas	Logout → Acessar `/perfil`	-	Redirecionamento para login ou mensagem de acesso negado	Frontend/API	1.0
Redirecionamento após	Verificar para onde o	Logout após login	-	Página inicial ou de login é	Frontend	1.0

logout	usuário é levado após logout			carregada		
Navegação limpa após logout	Verificar se menu personalizado não aparece mais	Logout → Verificar opções no menu	-	Menu padrão para visitantes é exibido	Frontend	1.0
Requisições sem token após logout	Garantir que token não é enviado após sair	Logout → Monitorar requisições subsequentes	-	Headers não contêm o token antigo	Frontend/API	1.0

4️⃣US-AUTH-004

Cenário de Teste US-AUTH-004

Descrição: Permitir que o usuário logado visualize e edite suas informações de perfil no Cinema App.

Pré-condições:

1. O usuário deve estar **logado e autenticado**;
2. A rota `/perfil` ou interface de perfil deve estar acessível apenas para usuários autenticados;
3. O backend deve prover um endpoint (ex: `GET /users/profile` e `PUT /users/profile`) que retorne e atualize os dados do usuário.

Condições:

- A página de perfil deve exibir: **nome, e-mail e tipo de conta** (usuário ou admin);
- O nome completo do usuário pode ser **editado**;
- O e-mail e o tipo de conta **não podem ser editados**;
- O sistema deve indicar visualmente os campos alterados (ex: mudança de cor ou borda);
- Após salvar, deve ser exibida uma **mensagem de sucesso**;
- O frontend deve atualizar os dados exibidos após alteração bem-sucedida;
- A edição de perfil deve estar disponível **apenas para o próprio usuário** autenticado.

Passo a passo:

1. Realizar login com um usuário válido;
2. Navegar até a página de **Perfil** através do menu;
3. Verificar se os dados (nome, e-mail, tipo) são exibidos corretamente;
4. Alterar o nome e clicar em **Salvar**;
5. Verificar se a resposta da API (`PUT /users/profile`) é `200 OK`;
6. Verificar se o nome foi atualizado corretamente no frontend e backend;
7. Verificar se a mensagem de sucesso é exibida;
8. Tentar alterar e-mail ou tipo e verificar se os campos estão desabilitados ou bloqueados.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Visualizar dados do perfil	Verificar se os dados do usuário são exibidos corretamente	Logar → Acessar página de perfil	Usuário autenticado	Nome, e-mail e tipo de conta visíveis	Frontend/API	1.0
Editar nome com sucesso	Alterar nome e salvar alterações	Alterar nome no campo e clicar em salvar	Novo nome: João Silva	Nome atualizado; resposta HTTP 200; mensagem de sucesso exibida	Frontend/API	1.0
Campo e-mail não editável	Verificar que o campo de e-mail está desabilitado	Tentar editar o e-mail diretamente na interface	-	Campo bloqueado ou em modo somente leitura	Frontend	1.0
Confirmação visual de alterações	Verificar se campos alterados mudam visualmente	Modificar o campo "nome"	-	Campo com destaque (ex: borda azul, ícone de modificação)	Frontend	1.0
Mensagem de sucesso após salvar	Confirmar feedback visual após atualização	Salvar alterações	-	Mensagem "Perfil atualizado com sucesso" exibida	Frontend	1.0
Tentar editar sem autenticação	Verificar segurança de rota protegida	Acessar `/perfil` sem estar logado	-	Redirecionamento para login ou erro HTTP 401	Frontend/API	1.0
Atualização inválida (nome vazio)	Tentar salvar nome em branco	Apagar conteúdo do campo "nome" e salvar	Nome: ""	Validação bloqueia ação; mensagem "Campo obrigatório"	Frontend/API	1.0

Acesso ao perfil de outro usuário	Testar acesso com token de outro usuário	Tentar editar perfil de outro usuário via token alternado	Token de usuário X para ID de usuário Y	HTTP 403 Forbidden ou redirecionamento	API	1.0

5️⃣US-HOME-001

Cenário de Teste US-HOME-001

Descrição: Verificar se a página inicial apresenta uma visão geral clara e atrativa para o usuário, com foco em conteúdo visual, navegação e responsividade.

Pré-condições:

1. A aplicação está acessível via navegador;
2. O frontend está carregando corretamente os dados da API (filmes em cartaz, usuário autenticado ou não);
3. Backend disponibiliza os filmes em cartaz.

Condições:

- Um **banner principal** deve ser exibido com informações do cinema;
- A seção **"Filmes em Cartaz"** deve mostrar pôsteres em tamanho adequado e com qualidade visual;
- O layout deve ser **responsivo**, adaptando-se a diferentes tamanhos de tela (mobile, tablet, desktop);
- A página deve conter **links de navegação rápidos** para áreas como login, reservas, perfil, etc.;
- O cabeçalho da página deve permitir **navegação principal clara**;
- **Usuários logados** devem ver opções personalizadas como “Minhas Reservas”, “Perfil” ou “Logout”.

Passo a passo:

1. Acessar a **home page** da aplicação (`/`);
2. Observar a presença do banner institucional;
3. Verificar a seção de **filmes em cartaz** com pôsteres organizados;
4. Redimensionar a tela ou abrir em dispositivo móvel para verificar responsividade;
5. Observar se os **links rápidos** para áreas principais estão visíveis e funcionais;
6. Fazer login como usuário e verificar se o **menu muda dinamicamente**.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Exibir banner principal	Verificar se banner institucional aparece na home	Acessar página inicial	-	Banner visível com texto/imagem informativa sobre o cinema	Frontend	1.0

Exibir seção Filmes em Cartaz	Verificar seção de filmes com pôsteres visuais	Rolagem até a seção de filmes	-	Grid com pôsteres, títulos e informações básicas	Frontend/API	1.0
Layout responsivo	Validar adaptação da home em telas diferentes	Redimensionar janela ou abrir em celular	-	Elementos se ajustam; layout legível e funcional	Frontend	1.0
Exibir links de navegação principais	Verificar links como login, reservas, perfil	Observar o cabeçalho	-	Links rápidos presentes e funcionais	Frontend	1.0
Menu para visitantes	Verificar opções disponíveis antes de login	Acessar como visitante	-	Exibição de login, registro, filmes; sem “Minhas Reservas” ou “Perfil”	Frontend	1.0
Menu para usuários autenticados	Verificar menu após login	Login → Voltar à home	Usuário válido	Menu mostra “Minhas Reservas”, “Perfil”, “Logout”	Frontend	1.0
Navegação funcional a partir da home	Verificar se os links da home levam para as seções corretas	Clicar nos botões/links da página inicial	-	Redirecionamentos corretos e sem erro 404	Frontend	1.0
Carregamento da home sem autenticação	Validar exibição para visitantes	Abrir página inicial sem login	-	Home carregada normalmente com conteúdo público	Frontend/API	1.0

6️⃣US-MOVIE-001

Cenário de Teste US-MOVIE-001

Descrição: Permitir que visitantes ou usuários autenticados visualizem a lista de filmes disponíveis, com informações claras e navegação acessível.

Pré-condições:

1. O backend deve estar populado com filmes disponíveis na rota (ex: `GET /movies`);
2. A home ou a página dedicada a “Filmes em Cartaz” deve estar acessível;

Condições:

- A lista de filmes deve ser exibida em **layout em grid** com boa organização visual;
- Cada card deve apresentar um **pôster de alta qualidade**, o **título**, a **classificação indicativa** e os **gêneros** do filme;
- Deve incluir **duração** e **data de lançamento** nos cards ou em tooltips/popups;
- O layout da lista de filmes deve ser **responsivo** (adaptável a diferentes tamanhos de tela);
- Cada card deve ser **clicável**, levando o usuário à página de detalhes do respectivo filme.

Passo a passo:

1. Acessar a página de **filmes em exibição** (na home ou em rota dedicada `/filmes`);
2. Observar a disposição dos cards em grid;
3. Verificar se os cards contêm as informações mínimas esperadas;
4. Clicar sobre um card e verificar se o redirecionamento para a página de detalhes ocorre;
5. Testar o comportamento da página em resoluções diferentes.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Exibição de grid de filmes	Verificar se os filmes aparecem organizados em grid	Acessar página de filmes	-	Lista de filmes exibida com espaçamento e alinhamento adequados	Frontend/API	1.0
Exibir pôster grande e visível	Validar que cada card exibe imagem de qualidade	Observar cards na página	-	Imagens visíveis, proporcionais e sem quebra	Frontend	1.0
Exibir título, classificação, gêneros	Verificar campos básicos de cada filme	Analisar informações visíveis nos cards	-	Título, classificação indicativa e gêneros corretamente exibidos	Frontend	1.0
Exibir duração e data de lançamento	Verificar informações adicionais de cada filme	Observar detalhes nos cards ou tooltip	-	Duração e data visíveis no card ou acessíveis via mouseover	Frontend	1.0
Card de filme é clicável	Verificar se clique no card leva à página de detalhes	Clicar em qualquer card	-	Navegação para página de detalhes do filme correspondente	Frontend	1.0

Acesso por visitante	Validar exibição da lista sem login	Acessar página de filmes sem estar autenticado	-	Lista de filmes visível normalmente	Frontend/API	1.0
Carregamento a partir da API	Verificar que os dados vêm da API `/movies`	Abrir devtools e observar requisição	-	Requisição `GET /movies` com retorno JSON válido	API	1.0

7️⃣US-MOVIE-002

Cenário de Teste US-MOVIE-002

Descrição: Permitir que visitantes ou usuários autenticados vejam informações detalhadas sobre um filme, com opção de seguir para a reserva.

Pré-condições:

1. O backend deve disponibilizar os dados detalhados do filme via rota (ex: `GET /movies/:id`);
2. Deve haver sessões cadastradas no sistema para o filme em questão;
3. A página de detalhes deve estar acessível a partir da lista de filmes ou via rota direta.

Condições:

- A página deve exibir: **sinopse, elenco, diretor, duração e data de lançamento**;
- O pôster do filme deve estar visível de forma destacada;
- Devem ser exibidos os **horários de sessão disponíveis** para aquele filme;
- O usuário deve conseguir **clicar em um horário para iniciar a reserva de assentos**;
- A página deve ser acessível tanto por visitantes quanto por usuários autenticados.

Passo a passo:

1. Acessar a lista de filmes e clicar em um card;
2. Ser redirecionado para a **página de detalhes** do filme selecionado;
3. Verificar a presença dos campos: sinopse, elenco, diretor, duração, lançamento;
4. Confirmar que o pôster está visível;
5. Verificar a lista de **sessões disponíveis** (com datas e horários);
6. Clicar em uma sessão e verificar se é redirecionado para a seleção de assentos.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Exibir detalhes do filme	Verificar campos principais do filme	Acessar página de detalhes	ID de filme válido	Sinopse, elenco, diretor, duração e lançamento visíveis	Frontend/API	1.0

Exibir pôster destacado	Verificar visibilidade do pôster do filme	Observar a imagem na página	-	Pôster exibido em tamanho adequado e carregando corretamente	Frontend	1.0
Exibir horários de sessão	Verificar sessões disponíveis para o filme	Observar seção "Horários"	-	Lista com datas, horários, sala e disponibilidade	Frontend/API	1.0
Acessar página de reserva pela sessão	Validar link do horário para reservar	Clicar em um horário	-	Redirecionamento para seleção de assentos	Frontend	1.0
Acesso com ID de filme inválido	Testar erro ao acessar um filme inexistente	Acessar `/filmes/abc123invalido`	ID inválido	Página de erro ou mensagem "Filme não encontrado"	Frontend/API	1.0
Carregamento de dados da API	Confirmar chamada `GET /movies/:id` e `GET /sessions?movieId=...`	Abrir devtools na página de detalhes	-	Requisições feitas com sucesso, resposta com dados relevantes	API	1.0

8️⃣US-SESSION-001

Cenário de Teste US-SESSION-001

Descrição: Permitir que visitantes e usuários autenticados visualizem os horários disponíveis de exibição para um filme específico e naveguem até a seleção de assentos.

Pré-condições:

1. O backend deve disponibilizar as sessões cadastradas com associação ao filme via rota (ex: `GET /sessions?movieId=...`);
2. O usuário deve acessar a **página de detalhes do filme**;
3. Deve haver pelo menos uma sessão futura cadastrada no banco.

Condições:

- Os horários de sessão devem exibir: **data, hora, teatro (sala)** e **disponibilidade de assentos**;
- A exibição deve ser clara, com layout que facilite comparação de opções;
- Os horários devem estar ordenados por data e hora;
- Cada item de horário deve ser **clicável** e redirecionar para a tela de **seleção de assentos**;
- A funcionalidade deve estar disponível tanto para visitantes quanto para usuários autenticados (reserva exige login posteriormente).

Passo a passo:

1. Acessar a página de detalhes de um filme com sessões disponíveis;
2. Observar a lista de **horários de sessão** exibida na tela;
3. Verificar se os campos de data, hora, sala e disponibilidade estão visíveis;
4. Clicar em um dos horários disponíveis;
5. Validar o redirecionamento para a **página de seleção de assentos**.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Exibir lista de sessões disponíveis	Verificar se sessões futuras aparecem na página	Acessar página de detalhes do filme	Filme com sessões	Sessões listadas com data, hora, sala e disponibilidade	Frontend/API	1.0
Sessões ordenadas cronologicamente	Validar ordenação das sessões por data e hora	Observar ordem da lista	-	Sessões ordenadas do mais próximo para o mais distante	Frontend	1.0
Visualização de sala e disponibilidade	Verificar se sala e número de assentos disponíveis são exibidos	Observar campos da sessão	-	Campo "Sala" e "Disponibilidade" visíveis	Frontend	1.0
Link para seleção de assentos	Verificar redirecionamento ao clicar em um horário	Clicar em um botão de horário	-	Usuário é levado à tela de seleção de assentos	Frontend	1.0
Sessão sem disponibilidade	Testar se sessões lotadas são desativadas ou indicadas corretamente	Visualizar sessão com todos assentos ocupados	Sessão lotada	Sessão marcada como indisponível ou botão desativado	Frontend/API	1.0
Carregamento da API `/sessions`	Validar que sessões vêm do backend com movieId	Acessar página → abrir devtools	-	Requisição `GET /sessions?movieId=xyz` retorna JSON com sessões	API	1.0

9️⃣US-RESERVE-001

Cenário de Teste US-RESERVE-001

Descrição: Permitir que o usuário logado visualize o mapa de assentos e selecione as posições disponíveis para reservar em uma sessão específica.

Pré-condições:

1. O usuário deve estar **logado** com token válido;
2. Deve existir uma **sessão com assentos disponíveis** no sistema;
3. O frontend deve exibir um **layout interativo** do mapa de assentos;
4. Backend deve disponibilizar as informações de assentos via rota (ex: `GET /sessions/:id/seats`).

Condições:

- O sistema deve exibir o **layout do teatro** com os assentos organizados visualmente;
- Assentos devem estar **codificados por cor**, indicando:
  - Disponível
  - Selecionado
  - Reservado
- O usuário pode **selecionar múltiplos assentos disponíveis**;
- **Assentos já reservados não podem ser selecionados**;
- O sistema deve calcular e exibir o **subtotal da compra** com base nos assentos selecionados;
- O botão de "Próximo" ou "Continuar" deve ser ativado somente após seleção válida.

Passo a passo:

1. Fazer login com um usuário válido;
2. Acessar uma sessão disponível via horários de exibição;
3. Verificar o **layout gráfico** dos assentos do teatro;
4. Observar a **legenda de cores** para status dos assentos;
5. Selecionar múltiplos assentos disponíveis;
6. Verificar se os assentos são destacados e se o subtotal aparece;
7. Tentar clicar em assentos já reservados e verificar o bloqueio;
8. Confirmar que é possível prosseguir apenas com seleção válida.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Visualizar layout de assentos	Verificar exibição do mapa gráfico da sala	Acessar página de reserva após escolher sessão	Sessão com layout ativo	Grid de assentos visível, organizado por fileira	Frontend/API	1.0
Legenda de cores	Validar se há legenda clara para	Observar layout de assentos	-	Cores distintas para disponível,	Frontend	1.0

	status dos assentos			selecionado e reservado		
Selecionar assento disponível	Verificar se assento pode ser selecionado	Clicar em assento disponível	-	Assento muda de cor para “selecionado”	Frontend	1.0
Selecionar múltiplos assentos	Verificar seleção de mais de um assento	Selecionar 2 ou mais assentos	-	Todos os assentos ficam destacados e adicionados ao subtotal	Frontend	1.0
Impedir seleção de assento reservado	Testar clique em assentos já ocupados	Clicar em assento marcado como “reservado”	Assento indisponível	Nada acontece ou mensagem “assento já reservado”	Frontend	1.0
Calcular subtotal dinamicamente	Verificar atualização do valor total da reserva	Selecionar e desselecionar assentos	Assento: R$ 20,00	Subtotal atualizado corretamente com base na seleção	Frontend	1.0
Acesso sem login	Testar tentativa de acesso à tela de assentos sem estar autenticado	Acessar `/reserva/:id` sem estar logado	-	Redirecionamento para login ou erro HTTP 401	Frontend/API	1.0
Interação com API de assentos	Validar requisição `GET /sessions/:id/seats`	Abrir devtools e observar a chamada	ID da sessão	Resposta com assentos, status e posições	API	1.0

🔟- US-RESERVE-002

Cenário de Teste US-RESERVE-002

Descrição: Permitir que o usuário finalize a compra de ingressos após selecionar os assentos, processando o pagamento e confirmando a reserva.

Pré-condições:

1. O usuário deve estar **logado**;
2. O usuário deve ter **selecionado um ou mais assentos disponíveis**;
3. A aplicação deve redirecionar o usuário para a página de **checkout** após a seleção;
4. O backend deve aceitar simulação de pagamento e registrar reserva.

Condições:

- A tela de checkout deve mostrar um **resumo claro dos assentos selecionados** (fileira, número, quantidade);
- O valor total da compra deve ser **calculado e exibido** corretamente;
- O usuário deve poder escolher o **método de pagamento** (cartão, débito, PIX, transferência);
- Ao confirmar a compra, o sistema deve **processar o pagamento (simulado)**;
- O sistema deve exibir **confirmação visual de sucesso** (mensagem ou tela de sucesso);
- Os **assentos selecionados devem ser marcados como ocupados** na base de dados e na próxima visualização da sessão.

Passo a passo:

1. Fazer login e selecionar assentos em uma sessão;
2. Ser redirecionado para a página de **checkout**;
3. Verificar os assentos selecionados e o valor total exibido;
4. Escolher um método de pagamento e confirmar a compra;
5. Verificar se o sistema retorna status de sucesso;
6. Visualizar mensagem de confirmação na interface;
7. Voltar à página da sessão e verificar que os assentos escolhidos agora estão ocupados.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Redirecionamento após seleção	Verificar se o usuário vai para o checkout após escolher assentos	Selecionar assentos → clicar em “Próximo”	2 assentos	Usuário vai para `/checkout` com os dados carregados	Frontend	1.0
Exibir resumo dos assentos	Validar apresentação das poltronas selecionadas	Observar o resumo do pedido na página de checkout	Assentos A2, A3	Resumo exibe corretamente fileiras, quantidade e numeração	Frontend	1.0
Exibir valor total	Verificar cálculo total com base nos assentos	Selecionar múltiplos assentos	R$ 20,00 cada	Total exibido = soma dos preços	Frontend	1.0
Selecionar método de pagamento	Validar interface de seleção de pagamento	Escolher cartão de crédito ou PIX	PIX	Campo de método ativo e selecionável	Frontend	1.0
Confirmar reserva	Verificar envio e resposta de	Confirmar checkout	Assentos + método	Requisição enviada com sucesso;	Frontend/API	1.0

	confirmação da reserva			resposta HTTP 201 ou equivalente		
Confirmação visual após pagamento	Verificar mensagem de sucesso após concluir reserva	Observar tela ou modal pós-checkout	-	Mensagem como "Reserva confirmada com sucesso"	Frontend	1.0
Atualização do status dos assentos	Verificar que os assentos são marcados como ocupados após o checkout	Voltar à mesma sessão após compra	-	Assentos aparecem como "reservados"	Frontend/API	1.0
Erro ao tentar reservar assentos ocupados	Testar tentativa de reservar assentos já ocupados	Forçar reserva duplicada (ex: outra aba)	Assento já reservado	API retorna erro 400/409; frontend mostra mensagem "assento indisponível"	API	1.0
Requisição para API de reserva	Validar chamada `POST /reservas` ou similar	Confirmar pedido no checkout	Corpo da reserva	JSON enviado corretamente com sessão, assentos, total e método	API	1.0

1️⃣1️⃣-US-RESERVE-003

Cenário de Teste US-RESERVE-003

Descrição: Permitir que o usuário autenticado visualize seu histórico de reservas com detalhes claros e organizados.

Pré-condições:

1. O usuário deve estar **logado** com sessão ativa (token válido);
2. O backend deve expor um endpoint para listar reservas por usuário (ex: `GET /reservas/minhas`);
3. O frontend deve disponibilizar a página **Minhas Reservas**, acessível via menu.

Condições:

- O menu de navegação deve conter um item “Minhas Reservas”, visível apenas para usuários logados;
- A página de reservas deve apresentar as informações em **cards visuais**;
- Cada card deve conter:
  - Título do filme
  - Data e horário da sessão

    - Cinema (sala ou unidade)
    - Lista de assentos reservados
    - Status da reserva (ex: confirmada, pendente, cancelada)
    - Método de pagamento
    - **Pôster do filme**
- Os **status das reservas** devem estar destacados com **indicadores visuais** (cor, ícone, selo, etc.);
- A página deve ser separada da tela de perfil e não acessível por visitantes.

Passo a passo:

1. Fazer login como usuário que possui reservas ativas;
2. Acessar o menu e clicar em “Minhas Reservas”;
3. Verificar se a rota leva para a página de histórico de reservas;
4. Observar se as reservas são exibidas corretamente em cards;
5. Validar presença e legibilidade de todas as informações relevantes;
6. Observar se o status está visualmente destacado (por cor ou selo);
7. Confirmar que pôsteres dos filmes são exibidos;
8. Validar o comportamento da tela para usuários **sem reservas**.

🔗 Casos de Teste Associados:

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Link “Minhas Reservas” visível	Verificar presença do link após login	Fazer login e abrir o menu	-	Link “Minhas Reservas” visível no menu	Frontend	1.0
Acesso à rota de reservas	Validar que a página é carregada ao clicar no link	Clicar em “Minhas Reservas”	-	Redirecionamento para `/reservas`	Frontend	1.0
Exibir cards de reservas	Verificar se reservas são exibidas em cards	Acessar página com histórico	Reservas registradas	Cards exibidos com organização visual clara	Frontend/API	1.0
Exibir detalhes da reserva	Validar campos como filme, data, hora, cinema, assentos e pagamento	Observar os dados de cada card	-	Todos os dados visíveis e bem formatados	Frontend	1.0
Exibir pôster do filme	Verificar se imagem do pôster é	Observar visual de cada card	-	Imagem do pôster exibida sem erro	Frontend	1.0

	carregada em cada reserva					
Status da reserva com destaque visual	Verificar destaque de status	Observar selo, cor ou ícone em cada card	Confirmada, Pendente	Indicação visual clara (ex: verde para confirmada, cinza para cancelada)	Frontend	1.0
Usuário sem reservas	Verificar comportamento quando não há reservas registradas	Fazer login com usuário sem reservas	-	Mensagem “Você ainda não fez nenhuma reserva” exibida	Frontend	1.0
Requisição para listar reservas	Validar chamada `GET /reservas/minhas`	Acessar página e abrir devtools	Token válido	Requisição com token; resposta com JSON contendo histórico de reservas	API	1.0

1️⃣2️⃣-US-NAV-001

Cenário de Teste US-NAV-001

Descrição: Garantir que o usuário, autenticado ou visitante, consiga navegar facilmente entre as páginas do Cinema App, com clareza visual, feedback e responsividade.

Pré-condições:

1. O sistema está acessível via navegador com interface carregada;
2. O usuário pode estar logado ou não;
3. As páginas da aplicação estão implementadas (ex: Home, Filmes, Perfil, Reservas, Login, etc.).

Condições:

- O **cabeçalho (header)** deve estar presente em todas as páginas visíveis;
- O **menu de navegação** deve conter os principais links (ex: Home, Filmes, Login/Logout, Minhas Reservas, Perfil);
- O menu deve ser **responsivo**, adaptando-se a dispositivos móveis;
- **Usuários logados** devem ver opções adicionais como “Minhas Reservas” e “Perfil”;
- **Breadcrumbs ou indicadores de caminho atual** devem ser visíveis em páginas internas;
- Deve haver **feedback visual claro** de qual página está ativa (ex: link destacado);
- Em telas com fluxo mais profundo, deve haver **link de retorno** para a página anterior.

Passo a passo:

1. Acessar a aplicação como visitante e verificar o cabeçalho em diferentes páginas;
2. Fazer login como usuário e verificar se o menu é atualizado com opções extras;
3. Verificar o comportamento do menu em dispositivos móveis (versão responsiva);

4. Acessar páginas internas (ex: detalhes do filme, checkout, perfil) e verificar breadcrumbs ou caminhos;
5. Observar qual item do menu está realçado como ativo;
6. Navegar para uma página interna e voltar usando o link de retorno (back link ou botão);
7. Testar acessibilidade visual (ex: contraste e clareza dos elementos ativos).

🔗 **Casos de Teste Associados:**

Test Name	Description	Test Step	Test Data	Expected Result	Component	Version
Cabeçalho presente em todas as páginas	Verificar visibilidade do header ao navegar entre rotas	Acessar várias rotas (home, login, perfil...)	-	Cabeçalho visível e funcional em todas as páginas	Frontend	1.0
Menu com links principais	Validar presença de links como Home, Filmes, Login, etc.	Observar o menu em qualquer página	-	Links principais visíveis no cabeçalho	Frontend	1.0
Menu responsivo	Testar menu em dispositivo móvel ou tela reduzida	Redimensionar janela / usar simulador mobile	-	Menu adaptado com botão hambúrguer ou equivalente	Frontend	1.0
Menu personalizado para usuário logado	Verificar se opções como "Minhas Reservas" e "Perfil" aparecem	Fazer login e abrir o menu	Usuário logado	Opções extras visíveis no menu após login	Frontend	1.0
Feedback visual da página ativa	Verificar destaque do link da página atual no menu	Navegar entre páginas	-	Link da página atual com cor diferente, sublinhado ou outra indicação visual	Frontend	1.0
Link de retorno para página anterior	Testar presença de botão/link de “voltar” em páginas internas	Acessar detalhes do filme ou checkout	-	Link funcional para voltar à página anterior (ex: lista de filmes)	Frontend	1.0

Navegação com teclado (acessibilidade)	Verificar usabilidade apenas com tab e enter	Usar teclado para navegar entre menus	-	Links acessíveis e utilizáveis sem mouse	Frontend	1.0

10. Matriz de Risco

Para a elaboração da matriz de risco, foram utilizadas as seguintes escalas:

- Probabilidade de falha: Alta/Média/Baixa.
- Impacto da falha na experiência do usuário ou funcionamento do sistema: Alto/Médio/Baixo.
- Riscos Associados
- Prioridade de teste: combinação de probabilidade, impacto e riscos

Rota	Probabilidade	Impacto	Riscos Associados	Prioridade
POST /users (Registro)	Média	Alta	• Validação incorreta de dados • Permitir e-mails duplicados • Falha no cadastro e retorno de erro	**Alta**
POST /login (Autenticação)	Alta	Alta	• Credenciais inválidas aceitas • Falha na geração ou armazenamento do token JWT • Ataques de brute force	**Alta**
GET /users/profile	Média	Média	• Exposição de dados incorretos • Falha no carregamento do perfil • Problemas de autorização	**Média**
PUT /users/profile	Média	Média	• Atualização incorreta dos dados • Falta de validação • Não confirmação de sucesso ao usuário	**Média**
GET / (Página inicial)	Baixa	Média	• Falha na exibição do banner ou filmes em cartaz • Problemas de responsividade	**Média**
GET /movies	Média	Alta	• Lista incompleta ou incorreta • Imagens com baixa qualidade • Falha ao carregar lista	**Alta**
GET /movies/:id	Média	Alta	• Dados incompletos ou errados • Falha ao carregar horários de sessões • Navegação quebrada	**Alta**
GET /sessions?movieId=	Média	Alta	• Horários desatualizados ou incorretos • Sessões com disponibilidade errada	**Alta**

GET /sessions/:id/seats	Alta	Alta	• Assentos incorretamente marcados como disponíveis ou ocupados • Layout confuso ou inacessível	**Alta**
POST /checkout ou /reservas	Alta	Crítica	• Pagamento não processado ou duplicado • Overbooking de assentos • Falha na confirmação da reserva	**Alta**
GET /reservas/minhas	Baixa	Média	• Histórico incompleto ou errado • Falha na exibição dos status ou dos métodos de pagamento	**Média**
GET /nav/header (frontend)	Baixa	Baixa	• Links quebrados • Problemas de responsividade • Feedback visual confuso	**Baixa**

11. Priorização da Execução dos Cenários de Teste

Considerando a realização dos testes com o Robot Framework e com base na documentação da API e, principalmente, as users stories apresentadas, bem como o fluxo da API e a matriz de risco apresentada no item anterior, os cenários de teste foram priorizados da seguinte forma:

Prioridade	Rotas / Funcionalidades	Justificativa	Recomendações
🔴 **Alta**	• POST /login (Autenticação)	Risco crítico: bloqueia acesso à aplicação, falha impacta todos os usuários	Executar primeiro, testar fluxos JWT e validação
	• POST /users (Registro)	Essencial para aquisição de novos usuários e dados corretos	Testes automáticos e de borda rigorosos
	• GET /movies (Lista de filmes)	Sem lista de filmes, usuário não pode explorar nem reservar	Testes funcionais
	• GET /movies/:id (Detalhes do filme)	Essencial para tomada de decisão de compra	Testar detalhamento, links para sessões
	• GET /sessions?movieId= (Horários)	Horários corretos são vitais para reserva	Testar atualização e disponibilidade
	• GET /sessions/:id/seats (Assentos)	Assentos incorretos impactam diretamente a experiência e reserva	Validar layout, cores e seleção
	• POST /checkout (Finalização de reserva)	Crítico para garantir compra sem erros, evita overbooking	Testar métodos de pagamento e confirmações
🟡**Média**	• GET /users/profile (Perfil)	Mantém dados do usuário atualizados e corretos	Testar leitura e controle de acesso
	• PUT /users/profile (Editar perfil)	Importante para atualização de dados pessoais	Validar mensagens e persistência

	• GET /reservas/minhas (Histórico de reservas)	Facilita acompanhamento, mas não bloqueia outras funções	Testes funcionais e visual
	• GET / (Página inicial)	Impacta percepção da aplicação, mas não impede funcionamento principal	Testes visuais e responsividade
🟢**Baixa**	• GET /nav/header (Navegação e menu)	Afeta usabilidade e navegação, porém com impacto menor	Testes de interface e responsividade

---

12. Cobertura de testes

A cobertura de testes foi medida considerando o Path Coverage, que "verifica a cobertura da suíte de testes de acordo com os endpoints que a API possui" (CREMA, Nayara. Como verificar a cobertura de testes de APIs REST, Medium).

🧮 Fórmula de Path Coverage

Path Coverage (%) = (Número de URIs testadas / Número total de URIs disponíveis) × 100

📌 URIs disponíveis na API Cinema

Módulo	Método HTTP	URI	Descrição
Usuários	POST	/users	Registro de novo usuário
	POST	/login	Autenticação/login
	POST	/logout	Logout do usuário
	GET	/users/profile	Visualizar perfil do usuário autenticado
	PUT	/users/profile	Atualizar dados do perfil
	GET	/users/:id	Buscar usuário por ID (possível admin)
	DELETE	/users/:id	Deletar usuário (possível admin)
Filmes	GET	/movies	Listar filmes disponíveis
	GET	/movies/:id	Visualizar detalhes do filme
	POST	/movies	Criar novo filme (admin)
	PUT	/movies/:id	Atualizar filme (admin)
	DELETE	/movies/:id	Remover filme (admin)
Sessões	GET	/sessions	Listar todas sessões (com filtros)
	GET	/sessions/:id	Detalhes de uma sessão
	GET	/sessions?movieId=xxx	Horários para filme específico

	GET	/sessions/:id/seats	Layout e status dos assentos da sessão
Reservas	POST	/reservations/checkout	Finalizar compra/reserva
	GET	/reservations/my	Listar reservas do usuário logado
	GET	/reservations/:id	Detalhes da reserva específica
	DELETE	/reservations/:id	Cancelar reserva
Navegação / UI	GET	/home	Dados para página inicial (banner, filmes)
	GET	/nav/header	Dados do menu e navegação

Total de URIs disponíveis: 22

✅ URIs testadas

URI	Status
`/usuarios`	Testada
`/usuarios/:id`	Testada
`/login`	Testada
`/produtos`	Testada
`/produtos/:id`	Testada
`/carrinhos`	Testada
`/carrinhos/:id`	Testada
`/carrinhos/cancelar-compra`	Testada
`/carrinhos/concluir-compra`	Testada

Total de URIs testadas: 9

📊 Resultado

`Path Coverage = (8 / 8) × 100 = 100%`

---

13. Testes candidatos a automação

Os testes candidatos à automação podem ser agrupados com base em diferentes tipos de validação e execução de requisições consecutivas.

Considerando o resultado dos testes manuais realizados no Postman, foram selecionados os casos de testes candidatos a automação:

Caso de Teste	Descrição	Justificativa para Automação
CT-Login-Valido	Testar login com credenciais válidas	Fluxo crítico, repetitivo, fundamental para acesso
CT-Login-Invalido	Testar login com credenciais inválidas	Regressão comum, previne acesso indevido
CT-Registro-Usuario	Testar cadastro de usuário com dados válidos	Processo obrigatório, validação automática facilita
CT-Registro-EmailDuplicado	Testar cadastro com email já cadastrado	Verificação de duplicidade, validação crítica
CT-Logout	Testar logout e invalidação do token	Teste essencial de segurança e sessão
CT-Visualizar-Perfil	Visualizar dados do perfil do usuário	Teste frequente e base para personalização
CT-Editar-Perfil	Atualizar dados do perfil e confirmar sucesso	Fluxo repetido, evita falhas na atualização
CT-Listar-Filmes	Listar filmes disponíveis com layout correto	Alta frequência e impacto visual
CT-Detalhes-Filme	Visualizar detalhes completos do filme	Facilita decisão do usuário, importante para UX
CT-Listar-Horarios	Consultar horários de sessões para um filme	Informação dinâmica e essencial para reservas
CT-Visualizar-Assentos	Visualizar layout e status dos assentos da sessão	Base para seleção e bloqueio, evita erros de disponibilidade
CT-Selecionar-Assentos	Selecionar múltiplos assentos disponíveis	Fluxo importante para compra e UX
CT-Checkout-Reserva	Finalizar compra e confirmar reserva	Processo crítico e sensível, garante fluxo completo
CT-Listar-Reservas-Usuario	Visualizar histórico de reservas do usuário	Função usada frequentemente, importante para confiança
CT-Navegacao-Menu-Responsivo	Verificar menu em diferentes resoluções	Garantir boa experiência em dispositivos variados