Plano de Teste - Challenge Final

🗂️ Sumário

🗂️ Sumário | 📝 Apresentação | 🎯 Objetivo | 📦 Escopo | 🛠️ Ferramentas utilizadas | 🧠 Mapa mental da aplicação | 🔍 Análise | 📌 Requisitos | 📋 Cenários de teste planejados | 🚧 Erros e Melhorias | ⚠️ Matriz de risco | 🤖 Testes automatizados | 📈 Cobertura de testes da API (Manuais e Automatizados) | ✅ Conclusão

---

📝 Apresentação

A aplicação Cinema App foi desenvolvida para simular um sistema de reserva de ingressos de cinema com o principal objetivo de ser usada como material de estudo em testes de API e de interface web durante este Challenge Final. Em sua documentação, conta com os endpoints `/auth`, `/movies`, `/reservations`, `/sessions`, `/theaters` e `/users`. Esses recursos permitem que sejam realizadas explorações e análises em busca de possíveis erros para que os estagiários na área de QA na Compass UOL apliquem seus conhecimentos de forma prática quanto à qualidade de software e testes de APIs REST e interface web.

A aplicação Cinema App é composta por uma interface web (front-end) e uma API (back-end), cujos repositórios estão disponíveis no GitHub:

- **Front-end**: GitHub - juniorschmitz/cinema-challenge-front
- **Back-end**: GitHub - juniorschmitz/cinema-challenge-back

🎯 Objetivo

O objetivo principal da realização dos testes realizados na API Cinema App é garantir a qualidade da aplicação por meio da validação das regras de negócio e da identificação de erros e possíveis oportunidades de melhoria.

📦 Escopo

Dentro do escopo

Serão testadas as seguintes rotas e suas respectivas requisições, conforme documentado no Swagger da API Cinema App:

- `/auth`
  - Realizar login
  - Obter perfil do usuário atual
  - Cadastrar novo usuário
  - Atualizar perfil do usuário
- `/movies`
  - Obter todos os filmes
  - Obter detalhes de um filme por ID

    - Cadastrar novo filme (apenas administradores)
    - Atualizar filme (apenas administradores)
    - Excluir filme (apenas administradores)
- `/reservations`
    - Obter todas as reservas (apenas administradores)
    - Obter reservas do usuário atual
    - Obter detalhes da reserva por ID
    - Cadastrar nova reserva
    - Atualizar status da reserva (apenas administradores)
    - Excluir reserva (apenas administradores)
- `/sessions`
    - Obter todas as sessões
    - Obter detalhes da sessão por ID
    - Cadastrar nova sessão (apenas administradores)
    - Atualizar sessão (apenas administradores)
    - Excluir sessão (apenas administradores)
    - Resetar todos os assentos de uma sessão para status “disponível” (apenas administradores)
- `/theaters`
    - Obter todos os cinemas
    - Obter detalhes do cinema por ID
    - Cadastrar novo cinema (apenas administradores)
    - Atualizar cinema (apenas administradores)
    - Excluir cinema (apenas administradores)
- `/users`
    - Obter todos os usuários (apenas administradores)
    - Obter detalhes do usuário por ID (apenas administradores)
    - Atualizar usuário (apenas administradores)
    - Excluir usuário (apenas administradores)

Além das rotas, também estão incluídos no escopo:

- Testes manuais e automatizados das funcionalidades disponíveis
- Testes de interface web (front-end), com foco na usabilidade e experiência do usuário
- Validação de responsividade da interface em diferentes resoluções

Fora do escopo

Não serão testadas as seguintes rotas e suas respectivas requisições, conforme documentado no Swagger da API Cinema App:

- `/setup`
- `/` (API Info)

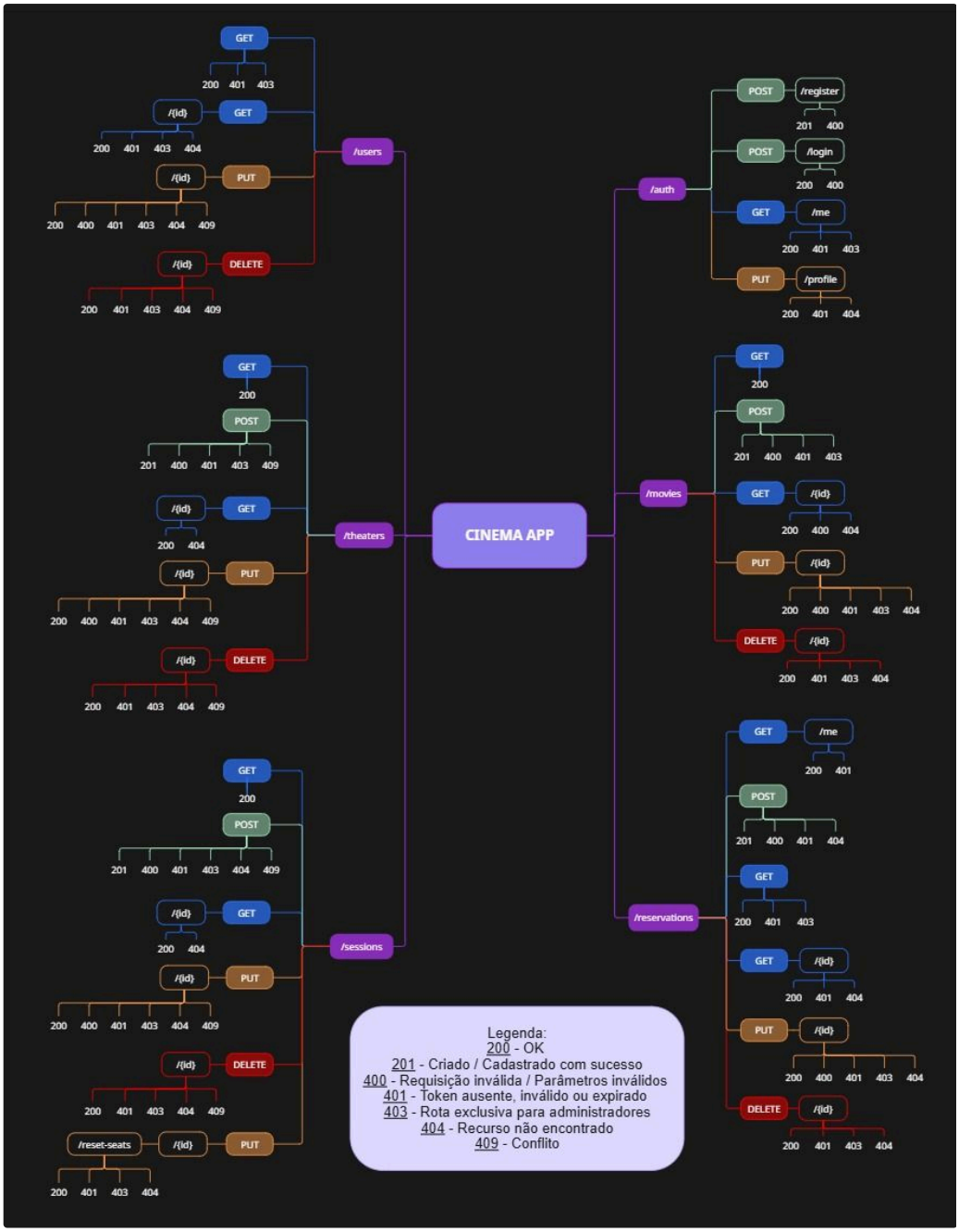
Além das rotas, também não estão incluídos no escopo:

- Testes de desempenho e carga
- Testes de segurança avançados
- Verificações de acessibilidade

🛠️ Ferramentas utilizadas

- Aplicação Cinema App:
  - **Front-end**: GitHub - juniorschmitz/cinema-challenge-front
  - **Back-end**: GitHub - juniorschmitz/cinema-challenge-back
- MongoDB
- Confluence
- Postman
- Jira
- Miro
- VS Code + Robot Framework
- QAlity no Jira

🧠 Mapa mental da aplicação



🔍 Análise

Cenários de teste

Os cenários foram gerados com base nos critérios de aceitação presentes nas histórias de usuário, cobrindo os principais fluxos funcionais e de interface. Além disso, foram considerados os happy paths descritos na documentação Swagger da API para garantir a integridade técnica das funcionalidades. Casos negativos e de exceção foram avaliados de forma exploratória, mas não foram formalizados neste documento para manter a objetividade e foco no comportamento esperado do sistema.

Técnicas aplicadas

- Testes baseados em cenários
  - Foi a principal técnica aplicada tendo como base os requisitos extraídos das User Stories e da documentação Swagger.
  - Abordagem permitiu construir casos de uso realistas que cobrem principalmente os fluxos essenciais.
- Testes exploratórios
  - Essa abordagem foi utilizada para identificar comportamentos inesperados ou inconsistências, especialmente em situações não cobertas diretamente pelos cenários documentados.
    Foi útil para validar exceções e a experiência do usuário.
- Partição de equivalência
  - Aplicada especialmente na validação de entradas nos campos de senha, garantindo que apenas senhas com mais de 6 caracteres fossem aceitas.
  - Permitiu dividir os dados de entrada em classes válidas e inválidas, otimizando o esforço de teste e garantindo maior cobertura de regras de negócio.

📌 Requisitos

Materiais utilizados para definição de requisitos:

- Histórias de usuário
  - Utilizadas como fonte principal, representando a visão e os objetivos do usuário final.
- Documentação Swagger
  - Utilizada como base para extrair os comportamentos esperados dos endpoints, especialmente quando as histórias de usuário estão ausentes ou não fornecem informações suficientes sobre as requisições e respostas esperadas.

`/auth`

- Base principal: US-AUTH-001, US-AUTH-002, US-AUTH-003 e US-AUTH-004.
- Complementado com requisitos técnicos extraídos do Swagger.

ID	Descrição	Origem
R1.1	Sistema deve permitir que usuário visitante cadastre uma conta informando nome, e-mail e senha.	US-AUTH-001
R1.2	Sistema deve validar o formato do e-mail, exigindo que ele contenha o caractere “@” e termine com “.com” após o domínio.	US-AUTH-001
R1.3	Sistema deve validar o formato da senha, exigindo que ela contenha 6 ou mais caracteres.	US-AUTH-001
R1.4	Sistema deve impedir cadastro de e-mail duplicados.	US-AUTH-001

R1.5	Após cadastro bem-sucedido, a interface deve redirecionar o usuário para a página de login e o sistema deve autenticar o acesso automaticamente.	US-AUTH-001
R1.6	Sistema deve permitir que usuário registrado faça login em sua conta informando e-mail e senha.	US-AUTH-002
R1.7	Sistema deve autenticar credenciais válidas no login.	US-AUTH-002
R1.8	Sistema deve manter sessão do usuário através de token JWT.	US-AUTH-002
R1.9	Após login bem-sucedido, a interface deve redirecionar o usuário para a página inicial.	US-AUTH-002
R1.10	Interface deve permitir que usuário logado possa fazer logout através do menu de navegação.	US-AUTH-003
R1.11	Após logout bem-sucedido, o sistema não deve permitir que rotas protegidas sejam acessadas.	US-AUTH-003
R1.12	Após logout bem-sucedido, sistema deve remover Token JWT do localStorage.	US-AUTH-003
R1.13	Sistema deve exibir nome do usuário, e-mail e função da conta no perfil do usuário logado.	US-AUTH-004
R1.14	Sistema deve permitir que usuário logado edite seu nome completo.	US-AUTH-004
R1.15	Interface deve indicar visualmente os campos que foram alterados.	US-AUTH-004
R1.16	Sistema deve confirmar sucesso após salvar alterações.	US-AUTH-004
R1.17	Interface deve exibir mensagem de confirmação após atualização bem-sucedida.	US-AUTH-004
R1.18	Página de perfil deve ser separada da página de reservas para melhor organização.	US-AUTH-004
R1.19	Sistema deve permitir que usuário atualize sua senha informando nome, senha atual e nova senha.	Swagger

⚠️ *Observação:* O formato que e-mail e senha devem ter informados nos requisitos R1.2 e R1.3 foram registrados de acordo com teste exploratório no front-end da aplicação, uma vez que não estavam especificados nas histórias de usuário.

`/movies`

- Base principal: US-HOME-001, US-MOVIE-001 e US-MOVIE-002.
- Complementado com requisitos técnicos extraídos do Swagger.

ID	Descrição	Origem
R2.1	Interface deve exibir banner com informações sobre o cinema na página inicial.	US-HOME-001
R2.2	Interface deve exibir uma seção destacada de "Filmes em Cartaz" com pôsteres em tamanho adequado.	US-HOME-001
R2.3	Layout da página inicial deve ser responsivo, se adaptando a diferentes tamanhos de tela.	US-HOME-001
R2.4	Interface deve ter links rápidos para principais áreas da aplicação na página inicial.	US-HOME-001
R2.5	Interface deve fornecer acesso fácil à navegação principal por meio do cabeçalho.	US-HOME-001
R2.6	Interface deve mostrar opções personalizadas no menu para usuários autenticados.	US-HOME-001
R2.7	Interface deve permitir que usuário visualize uma lista dos filmes em exibição com layout em grid.	US-MOVIE-001
R2.8	Interface deve permitir que filmes sejam exibidos com pôster grande e de alta qualidade.	US-MOVIE-001
R2.9	Interface deve permitir que cards de filmes mostrem título, classificação e gêneros.	US-MOVIE-001
R2.10	Interface deve permitir que cards incluam duração do filme e data de lançamento.	US-MOVIE-001
R2.11	Sistema deve adaptar layout para diferentes tamanhos de tela (responsivo) na lista de filmes.	US-MOVIE-001
R2.12	Interface permitir que usuário consiga acessar detalhes do filme com um clique.	US-MOVIE-001
R2.13	Detalhes do filme devem incluir sinopse, elenco, diretor, data de lançamento, duração.	US-MOVIE-002
R2.14	Interface deve mostrar pôster do filme na página de detalhes.	US-MOVIE-002
R2.15	Interface deve exibir horários de sessões disponíveis na página de detalhes.	US-MOVIE-002

R2.16	Interface deve permitir que usuário navegue para reserva a partir dos horários disponíveis.	US-MOVIE-002
R2.17	Sistema deve retornar todos os filmes, cadastrar novo filme, atualizar filme e excluir filme corretamente.	Swagger

`/reservations`

- Base principal: US-RESERVE-001, US-RESERVE-002 e US-RESERVE-003.
- Complementado com requisitos técnicos extraídos do Swagger.

ID	Descrição	Origem
R3.1	Interface deve permitir que usuário logado possa visualizar o layout de assentos do cinema.	US-RESERVE-001
R3.2	Assentos devem ser codificados por cores conforme disponibilidade.	US-RESERVE-001
R3.3	Sistema deve permitir que usuário selecione múltiplos assentos disponíveis.	US-RESERVE-001
R3.4	Sistema não deve permitir que usuário selecione assentos já reservados.	US-RESERVE-001
R3.5	Sistema deve mostrar o subtotal à medida que os assentos são selecionados.	US-RESERVE-001
R3.6	Interface deve redirecionar usuário para a página de checkout após selecionar os assentos.	US-RESERVE-002
R3.7	Interface deve permitir que resumo dos assentos selecionados sejam exibido na página de checkout.	US-RESERVE-002
R3.8	Sistema deve permitir que usuário visualize o valor total da compra.	US-RESERVE-002
R3.9	Sistema deve permitir que usuário selecione um método de pagamento (cartão de crédito, débito, PIX, transferência).	US-RESERVE-002
R3.10	Sistema deve processar o pagamento (simulado) e confirmar a reserva.	US-RESERVE-002
R3.11	Interface deve permitir que usuário receba confirmação visual do sucesso da reserva.	US-RESERVE-002
R3.12	Assentos selecionados devem ser marcados como ocupados.	US-RESERVE-002

R3.13	Interface deve permitir que usuário acesse a lista de suas reservas através do link "Minhas Reservas" no menu.	US-RESERVE-003
R3.14	Interface deve exibir reserva em formato de card com informações visuais claras.	US-RESERVE-003
R3.15	Detalhes devem incluir filme, data, horário, cinema, assentos, status e método de pagamento para cada reserva.	US-RESERVE-003
R3.16	Interface deve permitir que usuário visualize o pôster do filme associado à reserva.	US-RESERVE-003
R3.17	Sistema deve exibir indicadores visuais de status da reserva (confirmada, pendente, cancelada).	US-RESERVE-003
R3.18	Interface deve permitir que usuário acesse página dedicada de reservas separada das informações de perfil.	US-RESERVE-003
R3.19	Sistema deve retornar todas as reservas, retornar reservas do usuário logado, cadastrar nova reserva, atualizar status da reserva e excluir reserva corretamente.	Swagger

`/sessions`

- Base principal: US-SESSION-001.
- Complementado com requisitos técnicos extraídos do Swagger.

ID	Descrição	Origem
R4.1	Sistema deve permitir que usuário veja horários disponíveis para um filme selecionado.	US-SESSION-001
R4.2	Horários devem exibir data, hora, cinema e disponibilidade.	US-SESSION-001
R4.3	Interface deve permitir que usuário navegue para a seleção de assentos de um horário.	US-SESSION-001
R4.4	Sistema deve retornar todas as sessões de filmes, cadastrar nova sessão, atualizar sessão e excluir sessão corretamente.	Swagger
R4.5	Sistema deve resetar todos os assentos em uma sessão para atualizar status para “disponível”.	Swagger

`/theaters`

- Base principal: Documentação Swagger, como não há histórias de usuário.

ID	Descrição	Origem
R5.1	Detalhes do cinema deve incluir ID, nome, capacidade, tipo e data do cadastro.	Swagger
R5.2	Sistema deve retornar todos os cinemas, cadastrar novo cinema, atualizar cinema e excluir cinema corretamente.	Swagger

`/users`

- Base principal: Documentação Swagger, como não há histórias de usuário.

ID	Descrição	Origem
R6.1	Detalhes do usuário devem incluir ID, nome, e-mail, função e data do cadastro.	Swagger
R6.2	Sistema deve retornar todos os usuários, atualizar usuário e excluir usuário corretamente.	Swagger

Experiência do Usuário

- Base principal: US-NAV-001.

ID	Descrição	Origem
R7.1	Cabeçalho deve estar presente em todas as páginas com links para áreas principais.	US-NAV-001
R7.2	Menu deve ser responsivo e se adaptar a diferentes tamanhos de tela (versão móvel).	US-NAV-001
R7.3	Usuário logado deve ter acesso à seções personalizadas como "Minhas Reservas" e "Perfil".	US-NAV-001
R7.4	Breadcrumbs ou elementos de navegação devem indicar o caminho atual do usuário.	US-NAV-001
R7.5	Interface deve ter links diretos para retornar à página anterior quando apropriado.	US-NAV-001
R7.6	Feedback visual deve indicar a página atual no menu de navegação.	US-NAV-001

📋 Cenários de teste planejados

As tabelas abaixo, separadas por rotas e experiência do usuário, apresentam os cenários de teste planejados com base nos requisitos definidos a partir das histórias de usuário e da documentação Swagger. Também estão incluídas as informações sobre a execução dos testes manuais, indicando se foram realizados via Postman (para validação de requisições da API) ou via navegador (para testes de interface e experiência do usuário).

⚠️ *Observação:* Alguns requisitos foram validados por meio de cenários vinculados a rotas diferentes das originalmente previstas, uma vez que elas já contemplam adequadamente os fluxos esperados.

Autenticação

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade⚠️
CT01	Login com dados válidos	Usuário cadastrado e dados válidos	Postman	1. Criar requisição POST para `/auth/login` 2. Incluir e-mail e senha válidos no corpo da requisição 3. Verificar autenticação de credenciais válidas	Status code 200	Status code 200	✅ Passou	R1.6 e R1.7	Alta
CT02	Verificar redirecionamento para página inicial após login bem-sucedido	Usuário cadastrado e login bem-sucedido	Navegador	1. Realizar login em `/login` com dados válidos 2. Analisar redirecionamento para página inicial com conta logada	Redirecionamento de login para página inicial	Redirecionamento de login para página inicial	✅ Passou	R1.9	Média
CT03	Logout através do menu de	Usuário logado ou	Navegador	1. Estar logado em uma conta	Logout com sucesso	Logout com sucess	✅ Passou	R1.10 e R1.11	Alta

	navegação e acesso à áreas protegidas após logout	administrador		2. Clicar em “Sair” no canto direito do menu de navegação 3. Tentar acessar rotas protegidas	e acesso não permitido à rotas protegidas após desconexão	o e acesso não permitido à rotas protegidas após desconexão			
CT04	Verificar se sessão do usuário se mantém através de token JWT	Usuário logado ou administrador	Navegador	1. Realizar login 2. Abrir DevTools (F12) 3. Ir para aba “Aplicativo” 4. Ir em “Armazenamento local” (ou “localStorage”) no canto esquerdo 5. Selecionar link da aplicação 6. Excluir token e tentar entrar em rota protegida	Presença de Token JWT durante sessão ativa	Presença de Token JWT durante sessão ativa	✅ Passou	R1.8	Alta
CT05	Verificar remoção do token JWT do localStorage após logout bem-sucedido	Usuário desconectado	Navegador	1. Realizar logout 2. Abrir DevTools (F12) 3. Ir para aba “Aplicativo” 4. Verificar remoção de token JWT no “localStorage”	Sem presença do tolen JWT após logout	Sem presença do tolen JWT após logout	✅ Passou	R1.12	Alta

CT06	Verificar dados do perfil	Usuário logado ou administrador	Postman	1. Criar requisição GET para `/auth/me` 2. Adicionar token de autenticação no cabeçalho 3. Verificar dados retornados na resposta da requisição	Status code 200 e presença de nome do usuário, e-mail e função da conta nos detalhes	Status code 200 e presença de nome do usuário, e-mail e função da conta nos detalhes	✅ Passou	R1.13	Média
CT07	Cadastrar usuário com dados válidos	Usuário visitante e dados válidos	Postman	1. Criar requisição POST para `/auth/register` 2. Incluir nome, e-mail e senha válidos no corpo da requisição	Status code 201	Status code 201	✅ Passou	R1.1	Alta
CT08	Cadastrar usuário com formato de e-mail inválido	Usuário visitante e e-mail inválido	Postman	1. Criar requisição POST para `/auth/register` 2. Incluir nome e senha válidos e e-mail inválido no corpo da requisição	Status code 400	Status code 400	✅ Passou	R1.2	Média
CT09	Cadastrar usuário com formato de	Usuário visitante e senha inválida	Postman	1. Criar requisição POST para	Status code 400	Status code 400	✅ Passou	R1.3	Alta

	senha inválido			`/auth/register` 2. Incluir nome e e-mail válidos e senha inválida no corpo da requisição					
CT10	Cadastrar usuário com e-mail já utilizado	Usuário visitante e e-mail já utilizado	Postman	1. Criar requisição POST para `/auth/register` 2. Incluir nome e senha válidos e e-mail já utilizado no corpo da requisição	Status code 400	Status code 400	✅ Passou	R1.4	Alta
CT11	Verificar redirecionamento para página de login após cadastro bem-sucedido	Usuário cadastrado	Navegador	1. Realizar cadastro em `/register` com dados válidos 2. Analisar redirecionamento para página de login em conta recém-cadastrada	Redirecionamento de cadastro para página de login	Redirecionamento de cadastro para página inicial	❌ Falhou	R1.5	Baixa
CT12	Atualizar nome completo do usuário e verificar visualmente campos alterados	Usuário logado ou administrador	Navegador	1. Acessar “Perfil” no menu de navegação 2. Atualizar nome completo no campo “Nome	Alteração do nome com marcação visual de campos alterado	Alteração do nome com marcação visual de campos alterad	✅ Passou	R1.14, R1.15 e R1.17	Média

				Completo” em `/profile` 3. Verificar marcação visual dos campos alterados 4. Clicar em “Salvar Alterações” 5. Verificar mensagem de sucesso	s e mensagem de sucesso	os e mensagem de sucesso			
CT13	Atualizar senha com dados válidos	Usuário logado ou administrador	Postman	1. Criar requisição PUT para `/auth/profile` 2. Adicionar token de autenticação no cabeçalho 3. Incluir dados válidos para serem atualizados no corpo da requisição 4. Verificar confirmação de sucesso após salvar alterações no sistema	Status code 200	Status code 500	❌ Falhou	R1.19 e R1.16	Média
CT14	Verificar separação da página de perfil e da página de reservas	Usuário logado ou administrador	Navegador	1. Realizar login em `/login` com dados válidos 2. Acessar página de perfil pelo	Página de perfil e página de reservas	Página de perfil e página de reservas	✅ Passou	R1.18 e R3.18	Baixa

				menu de navegação 3. Acessar página de reservas pelo menu de navegação	separadas	separadas			

Movies

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade⚠️
CT15	Obter todos os filmes	Usuário visitante, logado ou administrador	Postman	1. Criar requisição GET para `/movies` 2. Incluir parâmetros válidos para filtrar filmes por título, gênero, ordem, limite ou página	Status code 200	Status code 200	✅ Passou	R2.17	Média
CT16	Verificar detalhes do filme	Usuário visitante, logado ou administrador	Postman	1. Criar requisição GET para `/movies/{id}` 2. Incluir ID do filme nos parâmetros 3. Verificar dados retornados na resposta da requisição	Status code 200 e presença de sinopse, elenco, diretor, data de lançamento, duração nos detalhes	Status code 200, porém não há elenco nos detalhes do filme	❌ Falhou	R2.13	Média

CT17	Verificar detalhes dos cards de filmes	Usuário visitante, logado ou administrador	Navegador	1. Acessar página principal ou seção “Filmes em Cartaz” 2. Verificar detalhes do card de um dos filmes	Cards de filmes devem ter título, classificação, gênero, duração do filme e data de lançamento e devem ser acessados facilmente	Cards de filmes devem ter título, classificação, gênero, duração do filme e data de lançamento e devem ser acessados facilmente	✅ Passou	R2.9, R2.10 e R2.12	Média
CT18	Verificar presença de horários de sessões disponíveis na página de detalhes	Usuário visitante, logado ou administrador	Navegador	1. Acessar página principal ou seção “Filmes em Cartaz” 2. Clicar em “Ver Detalhes” de um dos filmes 3. Verificar horários de sessões	Presença de horários de sessões disponíveis na página de detalhes	Presença de horários de sessões disponíveis na página de detalhes	✅ Passou	R2.15 e R4.1	Alta
CT19	Cadastrar filme com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição POST para `/movies` 2. Adicionar token de autenticação no cabeçalho 3. Incluir dados válidos no	Status code 201	Status code 201	✅ Passou	R2.17	Alta

				corpo da requisição					
CT20	Atualizar filme com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição PUT para `/movies/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do filme nos parâmetros 4. Incluir dados válidos para serem atualizados no corpo da requisição	Status code 200	Status code 200	✅ Passou	R2.17	Alta
CT21	Excluir filme com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição DELETE para `/movies/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do filme nos parâmetros	Status code 200	Status code 200	✅ Passou	R2.17	Média
CT22	Verificar navegação para reserva a partir de	Usuário visitante, logado ou	Navegador	1. Acessar página principal ou	Ir para reserva por meio de horários	Vai para reserva por meio de horários	✅ Passou	R2.16 e R4.3	Média

	horários disponíveis	administrador		seção “Filmes em Cartaz” 2. Clicar em “Ver Detalhes” de um dos filmes 3. Escolher horário de sessão em “Selecionar Assentos” 4. Acessar página de reserva	disponíveis com sucesso	disponíveis com sucesso			
CT23	Verificar banner com informações sobre o cinema na página inicial	Usuário visitante, logado ou administrador	Navegador	1. Acessar página inicial 2. Analisar banner com informações sobre cinema	Presença de banner sobre cinema na página inicial	Não há banner sobre cinema na página inicial	❌ Falhou	R2.1	Média
CT24	Verificar seção de “Filmes em Cartaz”	Usuário visitante, logado ou administrador	Navegador	1. Acessar página inicial 2. Clicar em “Filmes em Cartaz” no menu de navegação	Acessar “Filmes em Cartaz” com sucesso	Acesso de “Filmes em Cartaz” com sucesso	✅ Passou	R2.2	Alta
CT25	Verificar exibição dos pôsteres dos filmes	Usuário visitante, logado ou administrador	Navegador	1. Acessar página principal ou seção “Filmes em Cartaz” 2. Clicar em “Ver Detalhes” de um dos filmes 3. Analisar pôster	Pôsteres dos filmes devem ser grandes e com alta qualidade	Pôsteres dos filmes estão em tamanho pequeno e não aparecem	❌ Falhou	R2.8 e R2.14	Média
CT26	Verificar layout de	Usuário visitante,	Navegador	1. Acessar página	Filmes devem	Filmes são	✅ Passo	R2.7	Baixa

	lista de filmes em exibição	logado ou administrador		principal ou seção “Filmes em Cartaz” 2. Analisar layout da lista de filmes	ser exibidos com layout em grid	exibidos com layout em grid	u		

Reservations

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade ⚠️
CT27	Obter todas as reservas	Usuário administrador	Postman	1. Criar requisição GET para `/reservations` 2. Adicionar token de autenticação no cabeçalho 3. Incluir parâmetros válidos para filtrar por página e/ou limite	Status code 200	Status code 200	✅ Passou	R3.19	Média
CT28	Obter reservas do usuário atual	Usuário logado ou administrador	Postman	1. Criar requisição GET para `/reservations/me` 2. Adicionar token de autenticação no cabeçalho	Status code 200	Status code 200	✅ Passou	R3.19	Alta
CT29	Verificar detalhes da reserva	Usuário logado ou	Postman	1. Criar requisição GET para	Status code 200 e	Status code 200 e	✅ Passou	R3.15	Alta

		administrador		`/reservations/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID da reserva nos parâmetros 4. Verificar dados retornados na resposta da requisição	presença de filme, data, horário, cinema, assentos, status e método de pagamento nos detalhes	presença de filme, data, horário, cinema, assentos, status e método de pagamento nos detalhes			
CT30	Cadastrar reserva com dados válidos	Usuário logado ou administrador e dados válidos	Postman	1. Criar requisição POST para `/reservations` 2. Adicionar token de autenticação no cabeçalho 3. Incluir dados válidos no corpo da requisição	Status code 201	Status code 201	✅ Passou	R3.19	Alta
CT31	Cadastrar mais de um assento em uma reserva	Usuário logado ou administrador e dados válidos	Postman	1. Criar requisição POST para `/reservations` 2. Adicionar token de autenticação no cabeçalho 3. Incluir dados de mais de um	Status code 201	Status code 201	✅ Passou	R3.3	Alta

				assento no corpo da requisição					
CT32	Cadastrar assentos já reservados em uma reserva	Usuário logado ou administrador e assento já reservado	Postman	1. Criar requisição POST para `/reservations` 2. Adicionar token de autenticação no cabeçalho 3. Incluir dados de assento já reservado no corpo da requisição	Status code 400	Status code 400	✅ Passou	R3.4	Alta
CT33	Verificar layout de assentos do cinema com usuário logado	Usuário logado ou administrador	Navegador	1. Escolher horário em sessão de um filme 2. Analisar layout de assentos	Visualização correta do layout de assentos do cinema	Visualização correta do layout de assentos do cinema	✅ Passou	R3.1	Média
CT34	Verificar cores por disponibilidade e seleção dos assentos	Usuário logado ou administrador	Navegador	1. Escolher horário em sessão de um filme 2. Analisar cores por disponibilidade e seleção em layout de assentos	Assentos precisam estar com coloração correspondente a disponibilidade	Assentos estão com coloração correspondente a disponibilidade e seleção	✅ Passou	R3.2 e 3.12	Média

					e seleção				
CT35	Verificar subtotal ao que assentos são selecionados	Usuário logado ou administrador	Navegador	1. Escolher horário em sessão de um filme 2. Verificar subtotal ao escolher assentos	Presença do subtotal ao escolher assentos	Presença do subtotal ao escolher assentos	✅ Passou	R3.5	Média
CT36	Verificar redirecionamento para página de checkout após seleção de assentos	Usuário visitante ou logado	Navegador	1. Selecionar assentos em sessão de um filme 2. Clicar em “Continuar para Pagamento” 3. Analisar redirecionamento para página de checkout em `/checkout`	Redirecionamento da página de escolha de assentos para página de checkout	Redirecionamento da página de escolha de assentos para página de checkout	✅ Passou	R3.6	Alta
CT37	Verificar resumo dos assentos selecionados e valor total da compra na página de checkout	Usuário visitante, logado ou administrador	Navegador	1. Selecionar assentos em sessão de um filme 2. Clicar em “Continuar para Pagamento” 3. Verificar resumo dos assentos e total da compra na página de checkout em `/checkout`	Presença do resumo dos assentos selecionados e valor total da compra no checkout	Presença do resumo dos assentos selecionados e valor total da compra no checkout	✅ Passou	R3.7 e R3.8	Alta

CT38	Verificar seleção do método de pagamento, processamento do pagamento e confirmação da reserva	Usuário visitante, logado ou administrador	Navegador	1. Selecionar assentos em sessão de um filme 2. Clicar em “Continuar para Pagamento” 3. Escolher método de pagamento na página de checkout em `/checkout` 4. Clicar em “Finalizar Compra” 5. Verificar confirmação visual da reserva	Presença do método de pagamento na página de checkout, processamento de pagamento e confirmação da reserva mostrados com sucesso	Presença do método de pagamento na página de checkout, processamento de pagamento e confirmação da reserva mostrados com sucesso	✅ Passou	R3.9, R3.10 e R3.11	Alta
CT39	Acessar reservas pelo menu e visualizar como cards	Usuário logado ou administrador	Navegador	1. Clicar em “Minhas Reservas” no menu de navegação 2. Analisar se reservas estão presentes em formato de card	Acessar minhas reservas por meio do menu de navegação e visualizá-las em formato de card	Acessar minhas reservas por meio do menu de navegação e visualizá-las em formato de card	✅ Passou	R3.13, R3.14, R3.16 e R3.17	Média
CT40	Verificar indicadores visuais de status da	Usuário logado ou	Navegador	1. Clicar em “Minhas Reservas” no	Presença de indicador	Presença de indicador visual	✅ Passou	R3.17	Baixa

	reserva e pôster do filme	administrador		menu de navegação 2. Analisar se card possui indicador visual com status da reserva e pôster do filme	visual com status da reserva e pôster do filme	com status da reserva e espaço para pôster do filme			
CT41	Atualizar status da reserva com dados válidos	Usuário administrador	Postman	1. Criar requisição PUT para `/reservations/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID da reserva nos parâmetros 4. Incluir dados válidos para serem atualizados no corpo da requisição	Status code 200	Status code 200	✅ Passou	R3.19	Alta
CT42	Excluir reserva com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição DELETE para `/reservations/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID da reserva nos parâmetros	Status code 200	Status code 200	✅ Passou	R3.19	Média

Sessions

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade⚠️
CT43	Obter todas as sessões	Usuário visitante, logado ou administrador	Postman	1. Criar requisição GET para `/sessions` 2. Incluir parâmetros válidos para filtrar por filme, cinema, data, limite e/ou página	Status code 200	Status code 200	✅ Passou	R4.4	Média
CT44	Verificar detalhes dos horários	Usuário visitante, logado ou administrador	Postman	1. Criar requisição GET para `/sessions/{id}` 2. Incluir ID da sessão nos parâmetros 3. Verificar dados retornados na resposta da requisição	Status code 200 e presença de data, hora, cinema e disponibilidade nos detalhes	Status code 200 e presença de data, hora, cinema e disponibilidade nos detalhes	✅ Passou	R4.2	Média
CT45	Cadastrar sessão com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição POST para `/sessions` 2. Adicionar token de autenticação no cabeçalho 3. Incluir dados válidos no	Status code 201	Status code 201	✅ Passou	R4.4	Alta

				corpo da requisição					
CT46	Atualizar sessão com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição PUT para `/sessions/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID da sessão nos parâmetros 4. Incluir dados válidos para serem atualizados no corpo da requisição	Status code 200	Status code 200	✅ Passou	R4.4	Alta
CT47	Excluir sessão com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição DELETE para `/sessions/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID da sessão nos parâmetros	Status code 200	Status code 200	✅ Passou	R4.4	Média
CT48	Resetar assentos e verificar status dos assentos	Usuário administrador	Postman	1. Criar requisição PUT para `/sessions/{id}/reset-seats` 2. Adicionar token de	Status code 200	Status code 200	✅ Passou	R4.5	Média

				autenticação no cabeçalho 3. Incluir ID da sessão nos parâmetros					

Theaters

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade ⚠️
CT49	Obter todos os cinemas	Dados válidos	Postman	1. Criar requisição GET para `/theaters` 2. Incluir parâmetros válidos para filtrar por tipo do cinema, ordem, limite e/ou página	Status code 200	Status code 200	✅ Passou	R5.2	Média
CT50	Verificar detalhes do cinema	Dados válidos	Postman	1. Criar requisição GET para `/theaters/{id}` 2. Incluir ID do cinema nos parâmetros 3. Verificar dados retornados na resposta da requisição	Status code 200 e presença de ID, nome, capacidade, tipo e data do cadastro nos detalhes	Status code 200 e presença de ID, nome, capacidade, tipo e data do cadastro nos detalhes	✅ Passou	R5.1	Média
CT51	Cadastrar cinema com dados válidos	Usuário administrador e	Postman	1. Criar requisição POST para `/theaters`	Status code 201	Status code 201	✅ Passou	R5.2	Alta

		dados válidos		2. Adicionar token de autenticação no cabeçalho 3. Incluir dados válidos no corpo da requisição					
CT52	Atualizar cinema com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição PUT para `/theaters/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do cinema nos parâmetros 4. Incluir dados válidos para serem atualizados no corpo da requisição	Status code 200	Status code 200	✅ Passou	R5.2	Alta
CT53	Excluir cinema com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição DELETE para `/theaters/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do cinema nos parâmetros	Status code 200	Status code 200	✅ Passou	R5.2	Média

Users

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade⚠️
CT54	Obter todos os usuários	Usuário administrador	Postman	1. Criar requisição GET para `/users` 2. Adicionar token de autenticação no cabeçalho 3. Incluir parâmetros válidos para filtrar por página, limite e/ou função	Status code 200	Status code 200	✅ Passou	R6.2	Média
CT55	Verificar detalhes do usuário	Usuário administrador	Postman	1. Criar requisição GET para `/users/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do usuário nos parâmetros 4. Verificar dados retornados na resposta da requisição	Status code 200 e presença de ID, nome, e-mail, função e data do cadastro nos detalhes	Status code 200 e presença de ID, nome, e-mail, função e data do cadastro nos detalhes	✅ Passou	R6.1	Média
CT56	Atualizar usuário com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição PUT para `/users/{id}`	Status code 200	Status code 200	✅ Passou	R6.2	Média

				2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do usuário nos parâmetros 4. Incluir dados válidos para serem atualizados no corpo da requisição					
CT57	Excluir usuário com dados válidos	Usuário administrador e dados válidos	Postman	1. Criar requisição DELETE para `/users/{id}` 2. Adicionar token de autenticação no cabeçalho 3. Incluir ID do usuário nos parâmetros	Status code 200	Status code 200	✅ Passou	R6.2	Média

Experiência do Usuário

ID	Descrição	Pré-condições	Ferramenta utilizada	Passo a passo	Resultados esperados	Resultados obtidos	Status	Requisitos que cobre	Prioridade ⚠️
CT58	Verificar presença do cabeçalho contendo links rápidos para áreas principais nas páginas da aplicação	Usuário visitante, logado e administrador	Navegador	1. Analisar links rápidos para áreas principais no cabeçalho da aplicação em qualquer página para usuários	Presença de cabeçalho e links diretos, incluindo seções	Presença de cabeçalho e links diretos, incluindo seções	✅ Passou	R2.4, R2.5, R2.6, R7.1 e R7.3	Alta

				visitantes, logados e administrador	personalizadas para usuários autenticados	personalizadas para usuários autenticados			
CT59	Verificar responsividade do menu, página inicial e lista de filmes	Usuário visitante, logado ou administrador	Navegador	1. Testar menu, página inicial e seção “Filmes em Cartaz” da aplicação	Tela da aplicação se adapta a diferentes formatos	Tela da aplicação se adapta a diferentes formatos	✅ Passou	R2.3, R2.11 e R7.2	Alta
CT60	Verificar caminho atual do usuário pelos elementos de navegação	Usuário logado ou administrador	Navegador	1. Acessar alguma página da aplicação 2. Verificar elementos de navegação	Presença de elementos de navegação que mostrem caminho atual do usuário	Presença de elementos de navegação que mostrem caminho atual do usuário	✅ Passou	R7.4	Baixa
CT61	Verificar retorno para página anterior por links diretos	Usuário visitante, logado ou administrador	Navegador	1. Acessar alguma página da aplicação 2. Verificar se possui botão para voltar à página anterior	Presença de retorno à página anterior	Presença de retorno à página anterior	✅ Passou	R7.5	Média
CT62	Verificar indicação da página atual no menu por	Usuário visitante, logado ou	Navegador	1. Acessar alguma página da aplicação	Presença de indicação da	Presença de indicação da	✅ Passou	R7.6	Baixa

	feedback visual	administrador		2. Verificar indicação da página atual no menu	página atual no menu	página atual no menu			

Resumo

Tipo de teste	Executados	Passaram	Falharam
API (Postman)	35	33	2
Web (Navegador)	27	24	3
Total	62	57	5

🚧 Erros e Melhorias

Falhas encontradas e oportunidades de melhoria levantadas durante o processo de testes da API foram reportadas em tickets do Jira, a fim de garantir maior controle e melhor acompanhamento. Além disso, também é possível conferir a mesma documentação no documento “Relatório de erros e melhorias - Challenge Final” no repositório GitHub - Challenge Final.

⚠️ Matriz de risco

ID	Título	Probabilidade (1-3)	Impacto (1-3)	Risco
Risco 001	Falha na autenticação de usuários permite acessos não autorizados	2	3	6
Risco 002	Ausência de validação dos dados inseridos	2	3	6
Risco 003	Falha da API ou indisponibilidade do back-end	2	3	6
Risco 004	Tempo insuficiente para execução de todos os testes planejados	3	2	6
Risco 005	Interface e back-end exibem informações inconsistentes	2	3	6
Risco 006	Interface exibe elementos quebrados em telas menores (responsividade)	2	2	4

Risco 007	Botões ou links importantes não funcionam na interface	2	2	4
Risco 008	Sistema não fornece feedback visual após ações importantes	2	2	4
Risco 009	Documentação da API desatualizada ou incompleta	2	2	4
Risco 010	Token JWT expirado permite continuar acessando a API	1	3	3

🤖 Testes automatizados

A implementação e resultados dos testes automatizados podem ser visualizados no repositório GitHub - Challenge Final.

⚙️ API

Os principais testes de API que podem ser automatizados são os repetitivos e usados com frequência, os que possuem requisitos estáveis, os ligados a fluxos críticos do funcionamento da aplicação e os que apresentam resultados simples e previsíveis. Os cenários de teste criados que se encaixam nessa descrição e que foram automatizados são:

Critérios de automação	Cenários de teste	Descrição dos cenários	Justificativa para automação
Usados com frequência e fluxos críticos	CT01	Relacionado ao login	Primeiro contato dos usuários com a aplicação e necessário para que loja possa ser acessada
Fluxos críticos	CT07, CT10, CT19, CT30, CT31, CT32, CT45, CT51	Cadastro de usuários, filmes, reservas, sessões e cinemas	Um dos principais fluxos do sistema, já que usuários precisam conseguir cadastrar contas e aplicação se trata de um sistema para reserva de ingressos de cinema que necessita de filmes, reservas, sessões e cinemas para funcionar

Requisitos estáveis e fluxos críticos	CT08, CT09	Cadastro de usuários com padrões específicos de e-mail e senha	Importante para validação de regras de negócio, garantindo a segurança e conformidade dos dados de cadastro de usuários
Usados com frequência, fluxos críticos e resultados simples e previsíveis	CT06, CT15, CT16, CT28, CT29, CT43, CT44, CT49, CT50	Listagem e detalhes de usuários, filmes, reservas, sessões e cinemas	Importantes para que usuários consigam acessar detalhes importantes de seus perfis e de informações necessárias para reservarem ingressos

🖥️ Interface Web

Os testes de interface web mais indicados para automação são aqueles que simulam ações repetitivas do usuário e fazem parte dos fluxos principais da aplicação. Automatizar esses cenários garante agilidade na validação de funcionalidades essenciais e reduz o retrabalho causado por erros em pontos críticos da interface. Com base nesses critérios, foram automatizados os seguintes testes de frontend:

Critérios de automação	Cenários de teste	Descrição dos cenários	Justificativa para automação
Usados com frequência e fluxos críticos	CT02, CT03	Relacionado ao login e logout	Primeiro contato dos usuários com a aplicação, necessário para que loja possa ser acessada e conta possa ser desconectada
Fluxos críticos	CT24, CT34	Verificação de informações importantes para usuário conseguir completar ações	Importantes para que usuários consigam acessar informações necessárias para reservarem ingressos
Usados com frequência, fluxos críticos	CT22, CT36, CT38, CT39	Navegações e ações mais usadas pelos usuários na aplicação	Principais redirecionamentos e ações importantes para reserva de ingressos

Usados com frequência	CT58	Presença de cabeçalho para principais funcionalidades do site	Importante por oferecer acesso rápido e direto às funcionalidades e áreas mais importantes do sistema

Os testes relacionados aos fluxos principais da aplicação e à garantia de uma boa experiência do usuário foram priorizados na automação inicial. Os testes ainda não automatizados, como os mais voltados para funcionalidades administrativas ou fluxos menos críticos, poderão ser automatizados futuramente, conforme a evolução do projeto e a definição de novas prioridades.

📈 Cobertura de testes da API (Manuais e Automatizados)

Considerando apenas as rotas documentadas no Swagger que estão dentro do escopo dos testes:

Path Coverage (input)

Quantidade de endpoints na API REST: 16

Quantidade de endpoints testados de forma manual: 16

Total manual = 16/16 = 1 = 100%

Quantidade de endpoints testados de forma automatizada: 11

Total automatizada = 11/16 = 0,67 = 67%

Operator Coverage (input)

Quantidade de métodos na API REST: 30

Quantidade de métodos testados de forma manual: 30

Total manual = 30/30 = 1 = 100%

Quantidade de métodos testados de forma automatizada:

Total automatizada = 15/30 = 0,5 = 50%

✅ Conclusão

De modo geral, a grande maioria dos testes foi bem-sucedida. No entanto, os poucos casos de falha revelaram pontos de atenção importantes, principalmente por envolverem erros de código que precisam ser corrigidos e erros que afetam a experiência do usuário na utilização do sistema. Além disso, falhas também foram identificadas em partes voltadas à administração do sistema, como acesso à página administrativa na aplicação web, o que pode comprometer a eficiência da gestão.

Os testes manuais foram elaborados com foco em garantir uma cobertura ampla dos requisitos, conforme as User Stories e os happy path da documentação Swagger da API. Essa abordagem possibilitou validar os fluxos principais, contribuindo

para uma análise dos comportamentos esperados da aplicação. Os testes automatizados, por sua vez, priorizaram funcionalidades críticas e de execução recorrente. Isso permite a garantia de que as partes principais do sistema estejam sempre em correto funcionamento.

Além disso, foram realizados testes exploratórios e funcionais na interface web da aplicação, com foco na experiência do usuário. Verificações visuais, de usabilidade e de navegação permitiram avaliar como as funcionalidades se comportam na prática, identificando inconsistências e oportunidades de melhoria. Essa camada de validação reforçou a importância de testar não apenas os dados trocados na API, mas também como essas informações são apresentadas ao usuário final.

Em resumo, a execução dos testes na API e na interface web contribuiu significativamente para validar a aplicação, identificar riscos, verificar comportamentos esperados e definir ações prioritárias para evolução do sistema, promovendo uma experiência mais estável e eficiente para os usuários e administradores da plataforma.