# Relatório de Testes – Cinema App - Challenge Final (Back-end)

**Descrição Geral**
Durante a execução dos testes manuais e automatizados da aplicação (API + UI), foram encontrados diversos impedimentos técnicos que afetaram diretamente a cobertura de testes e a confiabilidade da solução entregue. Os principais problemas envolvem a instabilidade da aplicação, endpoints ausentes ou não funcionais, além de falhas críticas no comportamento da API, que comprometem tanto os testes quanto a lógica de negócio.

**Cenários e Requisitos Afetados**

- Requisições com perfil **admin** (ex: criação de filmes, sessões e usuários) **não puderam ser validadas** devido à falha na autenticação ou indisponibilidade de rotas esperadas.
- **Cenários de UI híbrida**, que dependem de dados da API (como sessões e reservas), ficaram bloqueados por ausência de resposta ou inconsistência no fluxo.
- Testes manuais executados com Robot Framework funcionaram, mas ao testar os **mesmos endpoints via Postman, muitas rotas retornaram 404 (Not Found)**, indicando que o path da API **não está disponível via ambiente externo**, embora o backend esteja funcional em testes automatizados.
- **Casos de testes negativos e de validação**, como duplicação de e-mails ou tokens inválidos, apresentaram comportamento inconsistente com o esperado — alguns testes passaram mesmo quando deveriam falhar, indicando problemas de tratamento interno.
- Destaca-se o caso do endpoint **PUT /users/{id}**, que deveria atualizar um usuário existente, mas **em vez disso cria um novo usuário com os mesmos dados**, resultando em duplicação e violando regras básicas de integridade de dados.
- Outros cenários dependentes de persistência (ex: reservas associadas ao usuário) também foram comprometidos.

**Gravidade:** Crítica

**Risco de Negócio:** A impossibilidade de autenticar ou simular um usuário admin inviabiliza todo o conjunto de operações administrativas (criação/edição/exclusão de filmes, sessões e usuários), o que compromete funcionalidades essenciais ao funcionamento do cinema, como:

- Controle de exibição de sessões;
- Gestão da base de filmes;
- Validação de reservas, pagamentos e moderação de usuários.

Essa fragilidade na base técnica pode gerar **inconsistência nas regras de negócio**, **falhas operacionais em produção** e um alto risco de **erros em ações sensíveis**, como alterações de dados ou transações.

**Riscos Operacionais**

- Impossibilidade de garantir a unicidade de registros no banco de dados.
- Ambiente de testes inconsistente entre ferramentas (Postman vs Robot), dificultando a automação confiável.
- Incapacidade de avançar na construção de pipelines CI/CD com Newman, já que muitos fluxos esperados não estão disponíveis para consumo via API.

**Impactos Diretos**

- Cobertura de testes comprometida nos principais fluxos de negócios.

- Impossibilidade de validar regras críticas, como atualização, segurança e autenticação.
- Automação bloqueada por ausência ou falha em rotas fundamentais.
- Dados inconsistentes inseridos no sistema, sem validação adequada.
- Requisições que “passam” nos testes mas executam ações erradas, como a duplicação de usuários existentes.
- Endpoints críticos ausentes ou retornando `404`, mesmo com dados e tokens corretos.
- Inexistência de uma interface de administração, comprometendo as ações de controle via front-end.

**Impactos Diretos:**

- Quebra na execução da suíte de testes (API e UI);
- Cobertura de testes comprometida por impossibilidade de validar cenários fundamentais;
- Dificuldade em aplicar boas práticas como isolamento de dados, rastreabilidade e automação contínua (CI/CD);
- Perda de confiabilidade do sistema como um todo, mesmo para testes que retornaram "sucesso";
- Tempo adicional gasto na investigação dos defeitos e revalidação dos cenários.

---

## Próximos Passos e Sugestões de Melhoria

Com base na análise dos testes, seguem sugestões construtivas para evolução do projeto:

1. **Estabilização dos Endpoints Existentes**
   Revisar a cobertura do código back-end e garantir que todos os endpoints mencionados nas user stories e ACs estejam devidamente implementados, com os retornos esperados e lógica consistente. Corrigir esse desvio melhora a interoperabilidade com ferramentas externas e a escalabilidade da aplicação.
2. **Correção de Comportamentos Críticas em Requisições CRUD**
   Revisar métodos HTTP em uso, especialmente nos verbos PUT/POST, para garantir que a semântica REST esteja sendo aplicada corretamente. Um PUT deve atualizar recursos existentes, não duplicá-los.
3. **Implementação da Interface de Administração no Front-end**
   A área “Administração”, mencionada nos fluxos de UI, deve ser mapeada e implementada corretamente. Hoje, o erro 404 impede o acesso e bloqueia toda a trilha de testes UI administrativos.
4. **Validação de Tokens e Controle de Acesso**
   Atualmente, não há forma prática de simular ou testar ações de usuário com papel ADMIN, pois o sistema não gera ou permite promover um usuário a esse nível. Recomenda-se implementar um fluxo de **promoção de usuário a admin**, ou disponibilizar um seed com credenciais administrativas válidas para testes.
5. **Documentação Técnica dos Endpoints (API Reference)**
   Faltam informações precisas sobre os parâmetros esperados, códigos de resposta, comportamento dos endpoints e headers necessários. A ausência dessa documentação gera tentativa e erro, reduz a produtividade e aumenta a chance de erros.
6. **Automatização de Testes Prioritários via CI/CD**
   Mesmo com as falhas encontradas, o planejamento já prevê um pipeline automatizado com Postman + Newman + GitHub Actions. Após correções estruturais, esse pipeline poderá ser ativado para garantir a execução contínua e confiável dos testes regressivos.
7. **Foco em Qualidade de Código e Logs de Erro**
   Hoje, vários endpoints falham silenciosamente ou retornam códigos genéricos. A implementação de mensagens claras de erro (com logs rastreáveis) e uma gestão de exceptions mais robusta ajudará no diagnóstico e manutenção do sistema.

## Conclusão

Este projeto de testes permitiu aplicar e consolidar práticas essenciais de QA, como testes funcionais, negativos, regressivos e automação via Postman. Apesar das limitações encontradas na aplicação — como endpoints instáveis, funcionalidades incompletas e falhas no controle de autenticação — foi possível identificar riscos reais que impactariam diretamente o negócio se fossem ignorados.

As evidências coletadas reforçam a importância de uma base técnica estável e da colaboração entre times para garantir qualidade contínua. Mesmo em um ambiente fictício, os testes mostraram como falhas no backend comprometem a experiência no frontend e geram riscos operacionais.

O relatório também apresenta sugestões viáveis de melhoria e inovações simples com potencial de agregar valor ao produto e ao processo de desenvolvimento. Com isso, o projeto cumpriu seu papel tanto na validação da aplicação quanto no aprimoramento das práticas de QA.