Cinema App

Autor: Mário Queiroz

1. Apresentação

Este documento detalha o planejamento estratégico para os testes da aplicação **Cinema App**, cobrindo tanto o **Frontend** quanto o **Backend**. O objetivo é guiar a equipe de Qualidade na validação completa das funcionalidades, da interface do usuário à lógica de negócio no servidor, com base nas Histórias de Usuário.

Este plano servirá como a principal fonte de referência para a criação de cenários, execução de testes e reporte de resultados.

- **Aplicação-alvo**: Cinema App
- **Ambientes de Teste**:
    - **Frontend**: Navegador Chrome/Chromium.
    - **Backend**: Ambiente de desenvolvimento local com Node.js e MongoDB.
- **Ferramentas**:
    - **Frontend**: Ferramentas de desenvolvedor do navegador, Robot (para automação E2E).
    - **Backend**: Postman, Insomnia (para testes de API).
    - **Gestão**: Jira e Confluence.

2. Objetivo

O objetivo principal é **validar a qualidade, funcionalidade e usabilidade da aplicação Cinema App de ponta a ponta**, garantindo que a experiência do usuário seja coesa e que a integração entre o frontend e o backend funcione perfeitamente, conforme os critérios de aceitação das Histórias de Usuário.

Objetivos Secundários:

- Garantir uma experiência de usuário intuitiva e visualmente consistente em diferentes navegadores e tamanhos de tela.
- Validar a segurança e a integridade dos dados desde a entrada no frontend até o armazenamento no backend.
- Identificar e documentar defeitos funcionais, de usabilidade e de integração.
- Criar uma base de testes de regressão (manuais e automatizados) para garantir a

estabilidade da aplicação em futuras atualizações.

3. Escopo

3.1. Funcionalidades em Escopo:

- **Testes de Backend (API):**
  - Validação de todos os endpoints (`/auth`, `/users`, `/movies`, `/theaters`, `/sessions`, `/reservations`).
  - Testes de contrato (request/response).
  - Validação de regras de negócio no servidor.
  - Testes de segurança de rotas (autenticação e autorização por roles).
- **Testes de Frontend (UI/UX):**
  - Validação de todos os fluxos de usuário descritos nas histórias.
  - Testes de componentes visuais e interativos.
  - Testes de responsividade (desktop, tablet, mobile).
  - Testes de usabilidade e navegação.
- **Testes de Integração (E2E - Ponta a Ponta):**
  - Fluxos completos que simulam a jornada do usuário, como: Registro -> Login -> Seleção de Filme -> Escolha de Assentos -> Checkout -> Verificação em "Minhas Reservas".

3.2. Funcionalidades Fora do Escopo:

- Testes de performance, carga e estresse.
- Integração real com gateways de pagamento (o pagamento é simulado).
- Testes de compatibilidade com navegadores legados (ex: Internet Explorer).

4. Análise de Testes (Estratégia)

A estratégia será dividida em três camadas:

1. **Testes de API (Backend):** Foco na lógica de negócio, integridade dos dados e segurança. Serão os primeiros a serem executados para garantir que a "base" da aplicação está sólida.
2. **Testes de UI (Frontend):** Foco nos componentes visuais, interatividade e experiência do usuário. Valida se a "vitrine" da loja é funcional e agradável.
3. **Testes de Ponta a Ponta (E2E):** Foco na integração entre as camadas. Valida se um cliente consegue entrar na loja, escolher um produto, pagar e sair com ele, simulando o fluxo real.

5. Técnicas de Teste Aplicadas

- **Particionamento de Equivalência e Análise de Valor Limite:** Usadas tanto no

backend (para testar a API) quanto no frontend (para testar formulários).

- **Teste Baseado em Estado:** Para validar as transições de estado da aplicação (ex: Visitante -> Logado -> Admin).
- **Testes Exploratórios:** Sessões livres onde o testador explora a aplicação sem um roteiro, buscando encontrar falhas inesperadas em ambos os ambientes (front e back).
- **Teste de Usabilidade:** Avaliação da facilidade de uso da interface, clareza da navegação e feedback visual ao usuário.
- **Teste de contrato(API):** Garantir o pleno funcionamento de todos os endpoints, especificando tipo da requisição, envio e recebimento de dados.

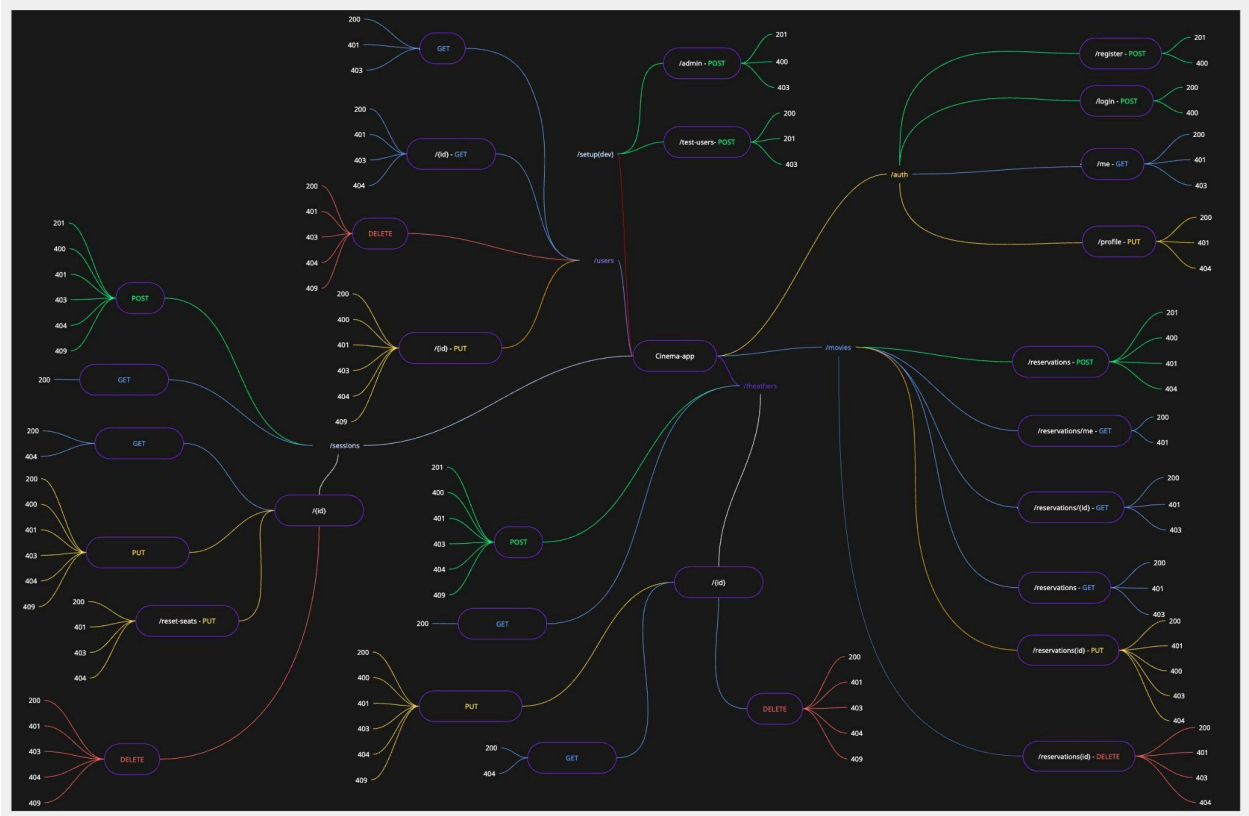
6. Cenários de Teste Planejados

7. Estratégia de Automação

- **Backend (API):** Utilizar o **Postman/Insomnia** para criar uma suíte de regressão automatizada que valide os contratos e regras de negócio da API. Executar essa suíte a cada nova build.
- **Frontend (E2E):** Utilizar Robot para automatizar os fluxos críticos do usuário (Happy Paths de registro, login e reserva). Esses testes garantem que as principais jornadas não foram quebradas.

8. Mapa Mental da Aplicação

A imagem a seguir, fornecida como base, ilustra a estrutura da aplicação Cinema-App e as funcionalidades a serem testadas.



9. Priorização da Execução dos Cenários

A execução seguirá uma priorização baseada na criticidade da funcionalidade para o negócio.

- **Prioridade Alta (Must Have):**
  - Cenários de "Happy Path" de todas as funcionalidades (/usuarios, /login, /produtos, /carrinho).
  - Testes de autenticação e autorização (US002-TC01, US003-TC02).
  - Validações críticas de negócio (e-mail único, nome de produto único).
  - "Happy Path" do fluxo de compra (US004-TC01, US004-TC09).
- **Prioridade Média (Should Have):**
  - Cenários de validação de dados (campos vazios, formatos inválidos, limites de senha).
  - Fluxos alternativos (ex: PUT criando um novo registro).
  - Restrições de negócio específicas (provedores de e-mail).
  - Validações de regras de negócio de carrinhos (US004-TC03, US004-TC05).
- **Prioridade Baixa (Could Have):**

- Cenários de teste para casos de borda menos comuns.
- Testes exploratórios para encontrar falhas não previstas.
- Cenário de cancelamento de compra (US004-TC10).

10. Matriz de Risco

11. Cobertura de Testes

12. Testes Automatizados com Robot

13. Testes automatizados na collection

14. Cronograma

ATIVIDADES	29/09	30/09	01/10	02/10	03/10	06/10	07/10	08/10	09/10	10/10
Criar Mapa mental										
Produção do plano de testes										
Criação dos scripts de teste										
Execução dos Testes										
Produção do relatório de issues e melhorias										
Apresentação dos resultados										