Teste Exploratório - Cinema App

1. Informações Gerais

- **Aplicação:** Cinema App
- **Tester:** @Thais Nogueira
- **Data da Sessão:** 25 de jun. de 2025
- **Ambiente:**
  - Backend: `http://localhost:3000/api/v1`
  - Frontend: `http://localhost:3002/`
  - Navegador: Google Chrome (última versão)
  - Sistema Operacional: Windows 10 64 bits

2. Heurísticas Utilizadas

Heurística	Aplicação no Contexto
ALTER-FACE	Validação de campos nas requisições, estados esperados nas respostas da API
CHIQUE	Verificação de campos obrigatórios, fluxo correto, bloqueios e mensagens apropriadas
GOLDILOCKS	Testes com dados mínimos, excessivos e ideais (ex: nome de filme vazio, muito longo)
CRUD	Criação, leitura, atualização e exclusão de recursos em todos os módulos

3. Técnicas Aplicadas

Particionamento de Equivalência: Entradas válidas e inválidas para todos os campos de requisição
Análise de Valor Limite: Nomes, senhas, quantidade de assentos, horários de sessões
Testes Negativos: Dados inválidos, endpoints acessados sem autorização, manipulação de payloads

Testes Baseados em Cenários: Fluxos ponta-a-ponta envolvendo múltiplos endpoints (ex: criar filme → criar sessão → fazer reserva)

Exploratórios: Manipulações criativas dos dados e das chamadas da API para revelar comportamentos inesperados

4. Notas de Sessão por Área

4.1 Autenticação

Charter:
Explorar o fluxo de autenticação e manipulação de tokens,

com foco em validação de campos e segurança,

para descobrir falhas de validação e vulnerabilidades de acesso.

Exploração Visual da Requisição e Resposta
Heurística: ALTER-FACE
Descrição Detalhada:

- Verifiquei se os campos obrigatórios (e-mail e senha) aparecem corretamente no payload de requisição.
- Analisei se as mensagens de retorno da API são claras, específicas e padronizadas, tanto para sucesso quanto para erro.
- No caso de sucesso no login, confirmei que o token JWT está presente no corpo da resposta e segue o padrão esperado
- Testei o tempo de resposta da API em diferentes condições (dados válidos, inválidos, token ausente), observando se há lentidão ou inconsistências.
- Validei se o status HTTP retornado está coerente com o tipo de resposta

Validação de Campos de Entrada
Técnica: Particionamento de Equivalência + Valor Limite
Heurística: CHIQUE

- Ambos os campos vazios.
- Cadastro com E-mail inválido
- Cadastro com e-mail existente
- Senha abaixo do mínimo permitido
- Cadastro com E-mail e senha corretos.
- Login com dados validos
- Login com e-mail nao cadastrado
- Login com campos vazios

Testes de Manipulação de Token

- Sem token: 401 esperado.
- Token válido: Acesso permitido.
- Token expirado: 401 esperado.

4.2 Filmes

Charter:

Explorar o gerenciamento de filmes

com foco em validação de campos, regras de negócio e operações CRUD,

para descobrir inconsistências de dados, falhas de validação e problemas de fluxo.

Exploração Visual das Respostas
Heurística: ALTER-FACE

- Verifiquei se os campos "Título" e "Duração" estão presentes e corretos nas respostas da API.
- Confirmei se as mensagens de retorno da API são padronizadas, claras e adequadas.
- Analisei se o status HTTP retornado está coerente
- Validei se o frontend reflete corretamente as alterações feitas via API, garantindo consistência visual.

Validação de Campos de Entrada
Técnica: Particionamento de Equivalência + Valor Limite
Heurística: CHIQUE, GOLDILOCKS

- Título vazio.
- Duração negativa
- Dados corretos.

Testes CRUD

- Criação, leitura, atualização e exclusão de filmes.
- Tentativa de remover filme em exibição resultou em erro apropriado.

4.3 Reservas

Charter: Explorar o fluxo de reservas

com foco em regras de negócio, integridade dos dados e manipulação de entradas

para descobrir falhas críticas em disponibilidade de assentos, reservas inválidas e inconsistências no frontend.

Exploração Visual das Respostas
Descrição Detalhada:

- Validei se os campos "Reserva", "Assentos" e "Sessão" estão corretamente presentes nas respostas da API.
- Analisei a clareza e padronização das mensagens de retorno, incluindo mensagens de erro para tentativas inválidas
- Confirmei a consistência dos status HTTP retornados em cada cenário.
- Observei o tempo de resposta da API ao criar, cancelar ou manipular reservas.
- Verifiquei se as alterações de reservas via API se refletem corretamente na visualização do frontend (ocupação de assentos).

Validação de Campos de Entrada

- Assentos ocupados.
- Reservas para sessões inexistentes.
- Reserva sem autenticaçao
- Dados corretos.
- Atualizaçao de dados

Testes de Manipulação de Dados

- Tentativas de burlar regras de reserva via API.
- Impacto correto refletido no frontend após manipulações.

4.3 Sessões

Charter: Explorar o cadastro e gerenciamento de sessões

com foco em integridade de dados, vínculos entre entidades e regras de negócio,

para descobrir falhas em dependências, validação de horários e inconsistências no fluxo.

Exploração Visual das Respostas

Heurística: ALTER-FACE

- Validação dos campos de sessão, horário, filme e teatro.

Validação de Campos de Entrada

Técnica: Particionamento de Equivalência + Valor Limite

- Horários inválidos (passado, sobreposição).
- Filmes inexistentes vinculados.
- Dados corretos.

Testes CRUD

- Criação, leitura, atualização e exclusão de sessões.
- Tentativa de remover sessões com reservas ativas

4.4 Teatros

Charter:

Explorar o gerenciamento de teatros

com foco em validação de campos, integridade dos dados e restrições de dependência,

para descobrir inconsistências e falhas nas regras de cadastro, edição e exclusão.

Exploração Visual das Respostas

- Verifiquei se os campos "Nome" e "Capacidade" aparecem corretamente nas respostas da API.
- Validei a clareza e padronização das mensagens de retorno, tanto para sucesso quanto para erros
- Confirmei se os status HTTP retornados correspondem às operações realizadas.
- Assegurei que as alterações feitas via API sejam refletidas corretamente na listagem do frontend.

Validação de Campos de Entrada

- Nome vazio ou muito longo.
- Capacidade: zero, negativa, muito alta.
- Dados corretos.

Testes CRUD

- Criação, leitura, atualização e exclusão de teatros.
- Tentativa de excluir teatro com sessões ativas gerou erro apropriado.

---

5. Report de bugs

Type	Key	Resumo	Prioridade	Categorias
Bug	CA-26	API- DELETE/reservations/id- Mensagem divergente ao deletar reserva entre API e...	Low	melhoria
Bug	CA-24	API POST/auth/profile- Erro 500 ao atualizar perfil de usuário - Funcionalidade não...	High	funcionalidade
Bug	CA-23	API- DELETE/user/id - Mensagem divergente ao deletar usuário	Low	melhoria
Bug	CA-22	API- POST/theaters -Inconsistência no Status Code ao tentar criar teatro com nom...	Medium	melhoria
Bug	CA-21	Front- PUT/session/id/reset-seats -Usuário comum consegue "desmarcar" assento...	High	funcionalidade

Type	Key	Resumo	Prioridade	Categorias
🐞	CA-20	Front- PUT/session/id/reset-seats -Reset de assentos como admin exibe mensage...	= Medium	funcionalidade
🐞	CA-19	API-DELETE/sessions/id - API permite deletar sessão com reserva ativa	^ High	
🐞	CA-18	API- DELETE/sessions/id -Documentação Swagger informa retorno incorreto para ...	⌄ Low	melhoria
🐞	CA-17	API-POST/session - API permite criar sessão com data/hora no passado	^ High	funcionalidade
🐞	CA-16	API- POST/sessions - API permite criar sessões duplicadas para mesmo horário e ...	^ High	funcionalidade
🐞	CA-15	API - PUT/reservations- Documentação Swagger não especifica status aceitos na ...	⌄ Low	melhoria

Sincronizado agora • 25 itens

Abaixo estão listados os bugs identificados durante a execução da sessão exploratória:

- CA-1: API - POST /auth/register - Status incorreto ao cadastrar e-mail já existente
- CA-2: API - POST /auth/register - Mensagem de erro não documentada para senha abaixo do mínimo permitido
- CA-3: API - POST /auth/login - Documentação Swagger do login apresenta status incorreto para credenciais inválidas
- CA-4: API - POST /auth/login - Status e mensagem não documentados ao realizar login com campos vazios
- CA-5: API - GET /auth/me - Mensagem de erro não documentada ao tentar obter perfil do usuário sem autenticação
- CA-6: Front - Mensagem de erro no Login desaparece rápido demais, dificultando visualização
- CA-7: Front - Mensagem de erro genérica no cadastro ao inserir senha abaixo do mínimo permitido
- CA-8: API - POST /movies - mensagem de erro não documentada ao tentar criar filme com título em branco
- CA-9: API - POST /movies - Permite criar filme com duração negativa
- CA-10: API - DELETE /movies/{id} - Mensagem de sucesso ao deletar filme está diferente do Swagger
- CA-11: API - POST /sessions - Documentação Swagger imprecisa - Campos movie e theater no cadastro de sessão aceitam apenas IDs
- CA-12: API - DELETE /movies/id - Mensagem de erro no Swagger diferente ao tentar deletar filme sem permissão de administrador
- CA-13: API - POST /reservation - Mensagem de erro diferente documentada no Swagger ao tentar criar reserva com assento já ocupado
- CA-14: API - POST /reservation - Documentação Swagger imprecisa - Campo session no cadastro de reserva exige ID
- CA-15: API - PUT /reservations - Documentação Swagger não especifica status aceitos na atualização de reserva
- CA-16: API - POST /sessions - API permite criar sessões duplicadas para mesmo horário e sala
- CA-17: API - POST /session - API permite criar sessão com data/hora no passado
- CA-18: API - DELETE /sessions/id - Documentação Swagger informa retorno incorreto para exclusão de sessão
- CA-19: API - DELETE /sessions/id - API permite deletar sessão com reserva ativa
- CA-20: Front - PUT /session/id/reset-seats - Reset de assentos como admin exibe mensagem de erro no Front-end, mesmo com operação concluída
- CA-21: Front - PUT /session/id/reset-seats - Usuário comum consegue "desmarcar" assento na interface ao clicar em "Liberar Assentos"
- CA-22: API - POST /theaters - Inconsistência no Status Code ao tentar criar teatro com nome duplicado
- CA-23: API - DELETE /user/id - Mensagem divergente ao deletar usuário
- CA-24: API - POST /auth/profile - Erro 500 ao atualizar perfil de usuário - Funcionalidade não finalizada está exposta na documentação Swagger
- CA-26: API - DELETE /reservations/id - Mensagem divergente ao deletar reserva entre API e documentação Swagger

6. Resumo Final da Sessão

✔ Foram encontrados 26 bugs distribuídos entre o Frontend e a API.

✔ Problemas críticos de segurança e validação foram detectados, principalmente em autenticação, reservas e sessões.

✔ Diversas inconsistências entre a documentação Swagger e o comportamento real da API foram evidenciadas.

✔ Fluxos positivos básicos foram confirmados para cadastros e reservas quando as entradas estão corretas.

✔ Mensagens de erro e status HTTP carecem de padronização e precisam ser alinhados com a documentação.

✔ A experiência do usuário no Frontend apresenta pontos de melhoria relacionados a mensagens e usabilidade.

7. Sugestões e Melhorias

Melhorias Técnicas:

- Reforçar as validações no backend, especialmente para senhas, e-mails e dados de sessões.
- Corrigir as inconsistências documentais no Swagger para refletir o comportamento real da API.
- Implementar restrições adicionais para impedir reservas ou sessões com dados inválidos ou duplicados.
- Garantir que o backend rejeite tokens manipulados de forma robusta.
- Revisar o mecanismo de expiração e renovação de tokens.

Melhorias no Frontend:

- Tornar as mensagens de erro mais claras, visíveis e duradouras, principalmente no fluxo de login e cadastro.
- Garantir o bloqueio adequado de funcionalidades para usuários não autenticados ou sem permissão.
- Revisar os fluxos de reserva e liberação de assentos, evitando manipulações indesejadas.