Relatório de Execução de Testes – Cinema App API

Data de Execução: 26 de junho de 2025 **Ferramenta de Automação:** Robot Framework

1. Resumo da Execução

Os testes automatizados da API REST foram realizados com sucesso utilizando o **Robot Framework**. As principais bibliotecas empregadas foram:

- **RequestsLibrary:** para a execução de requisições HTTP (GET, POST, PUT, DELETE).
- **JSONLibrary:** para a extração de dados e validação de respostas em formato JSON.
- **Collections:** para a manipulação de estruturas de dados, como dicionários para a montagem de *request bodies* e *headers*.
- **DateTime:** para a geração de dados dinâmicos (timestamps), garantindo a unicidade em cenários de criação.

A estrutura de automação seguiu rigorosamente o modelo proposto, com uma organização modular que separa casos de teste, keywords de negócio, recursos e variáveis, garantindo alta legibilidade e manutenibilidade.

---

2. Limitações na Documentação

Durante a fase de implementação dos testes, foi identificado que a documentação da API via **Swagger** estava **incompleta** ou **inexistente** para as seguintes rotas críticas:

- **/sessions**
- **/reservations**

Essas rotas não apresentavam uma definição clara dos schemas de *request body* para as operações de `POST`, nem exemplos de uso. Essa lacuna impossibilitou a consulta direta à documentação pública como fonte primária para a criação dos testes destas funcionalidades.

---

3. Ação Técnica Realizada

Para contornar a limitação da documentação, foi adotada uma abordagem de **análise estática do código-fonte do backend**. Essa análise permitiu compreender a fundo o comportamento esperado da API, incluindo:

- **Métodos e Rotas:** Verificação dos endpoints disponíveis em `src/routes/sessionRoutes.js` e `src/routes/reservationRoutes.js`.
- **Parâmetros e Corpo da Requisição:** Análise dos respectivos *controllers* (ex: `sessionController.js`) e *models*(ex: `Session.js`, `Reservation.js`) para identificar os campos obrigatórios, tipos de dados e as validações internas.

Com base neste levantamento, foi possível **automatizar os testes das rotas** `/sessions` **e** `/reservations` **com total precisão**, garantindo a cobertura planejada mesmo sem o apoio da documentação oficial. Sendo este problema da documentação posteriormente corrigido e as rotas limitantes devidamente reportas com exatidão dentro das ferramentas utilizadas.

---

4. Evidências e Logs

Todos os testes foram executados e os **logs completos foram gerados com o Robot Framework**. Os relatórios detalhados, que servem como evidência da execução e dos resultados, podem ser consultados nos seguintes arquivos gerados na pasta de resultados:

- `log.html` (Log detalhado com cada passo da execução)

- `report.html` (Relatório gerencial com gráficos e resumo dos resultados)
- `output.xml` (Arquivo de saída com todos os dados brutos da execução)

O ciclo de testes completo pode ser analisado através do **QAlity** afim de obter um histórico completo da execução de testes realizadas.

4.1 Issues Encontradas

Durante o processo de desenvolvimento da automação, foram identificados os seguintes pontos que exigiram análise e correção. Eles foram registrados na ferramenta de gestão de projetos para garantir o rastreamento e a resolução formal pela equipe de desenvolvimento.

1. **Melhoria de Documentação: Revisão e Complemento do Swagger API [Resolvido]**

Descrição: Foi identificado que a documentação da API via Swagger, embora funcional, está incompleta em diversas rotas, notadamente em `/sessions` e `/reservations`. Para garantir que a documentação sirva como uma fonte única e confiável, é necessária uma revisão completa para assegurar que todos os endpoints possuam schemas de requisição e resposta, exemplos de uso e parâmetros devidamente documentados.

BC-262: SESSIONS - Documentação Swagger Incompleta Impossibilita Testes CONCLUÍDO

BC-263: RESERVATIONS - Documentação Swagger Incompleta impossibilita Testes CONCLUÍDO

2. **Inconsistência no Seed das Credenciais de Teste do Administrador**

Descrição: Foi identificada uma divergência entre as senhas do usuário administrador (`admin@example.com`) nos diferentes arquivos do projeto (documentação vs. scripts de seed/validação), o que causou uma falha inicial de `401 Unauthorized` na suíte de testes. Os usuários por seed não são repassados ao banco de dados com seus dados devidamente criptografados.

BC-264: Inconsistência no Seed das Credenciais de Teste do Administrador TO-DO [BUGS PRA CORRIGIR]

3. **Esclarecimento de Requisito: Ausência de Limite de Caracteres**

Descrição: Não há especificação ou validação de limite máximo de caracteres para campos de texto importantes, como `name` no modelo de usuário (`src/models/User.js`). A ausência desta regra de negócio pode causar inconsistências de dados e potenciais problemas de layout na interface.

BC-265: Definir e Implementar Limite de Caracteres em Campos de Texto TO-DO [BUGS PRA CORRIGIR]

4. **Falha na Documentação: Omissão da Variável de Ambiente** `JWT_SECRET`

Descrição: O arquivo `README.md` do backend omite a variável de ambiente obrigatória `JWT_SECRET` , que é indispensável para a geração de tokens de autenticação, conforme verificado no código-fonte (`src/utils/generateToken.js`). A ausência desta informação impede a correta configuração do ambiente por novos desenvolvedores.

BC-266: Adicionar Variável JWT_SECRET ao README do Backend TO-DO [BUGS PRA CORRIGIR]

---

5. Arquivo de Apoio – Excel

Para facilitar a consulta e o gerenciamento dos resultados em outras ferramentas, foi gerado um arquivo de apoio em formato Excel:

- Text_Execution_Report.xlsx

Este arquivo contém uma transposição dos dados da tabela acima, com todos os cenários, status de execução, mensagens de erro (quando aplicável em depuração) e referências para as evidências nos logs, simulando a integração com um sistema de planejamento como o QAlity.

---

6. Conclusão

A execução da suíte de testes automatizados foi concluída com **100% de sucesso**. Todos os cenários descritos no planejamento foram implementados, depurados e validados. A ausência de documentação em rotas importantes foi um desafio superado com sucesso através da análise técnica do código-fonte. Elementos da documentação apresentaram problemas e foram devidamente relatados para correção.

O projeto de automação resultante é **completo, robusto, confiável e pronto para ser utilizado em ciclos de regressão**, garantindo a qualidade contínua da Cinema App API.