CESED – CENTRO DE ENSINO SUPERIOR E DESENVOLVIMENTO

ELABORAR PLANO DE TESTES E VALIDAÇÃO DE UM SOFTWARE

IMPLEMENTAÇÃO DE UMA BILHETERIA DO CINEMA

PROFESSOR(a): CARLOS DIEGO

ALEXANDRE FERNANDES DE OLIVEIRA BESSA

LUAN DOS SANTOS BARBOSA

LUIS VITOR GOMES ALVES DA SILVA

PEDRO HENRIQUE TEBERGES CAVALCANTI

RYANDRO DA SILVA TAVARES

IMPLEMENTAÇÃO DE UMA BILHETERIA DO CINEMA

CAMPINA GRANDE – PB

SETEMBRO - 2025

1. Análise de Requisitos de Teste

1.1 Identificação e Categorização dos Requisitos Funcionais

A tabela abaixo lista todos os requisitos funcionais que serão testados, indicando sua categoria e se serão validados.

ID Requisito	Descrição	Categoria	Será testado? (Sim/Não)
RF01	O sistema deve permitir adicionar um novo filme em uma das salas disponíveis.	Funcional	Sim
RF02	O sistema deve permitir remover ou atualizar um filme existente em uma sala.	Funcional	Sim
RF03	O sistema deve emitir um ingresso para o filme selecionado, gerando comprovante digital.	Funcional	Sim
RF04	O sistema deve emitir o status de cada sala.	Usabilidade	Sim
RF05	O sistema deve permitir filtrar filmes por nome parcial e/ou intervalo de datas de lançamento.	Funcional	Sim
RF06	O sistema deve verificar a validade de um ticket emitido, garantindo que não seja reutilizado	Segurança	Sim
RF07	O sistema deve salvar o estado atual das salas e dos filmes para manter os dados após o fechamento da aplicação.	Persistência	Sim

1.2 Matriz de Rastreabilidade

A tabela abaixo mostra a ligação entre cada requisito do sistema e os testes que serão feitos para verificar se tudo está funcionando corretamente. Dessa forma, dá pra garantir que todas as funções do sistema de cinema foram testadas e estão de acordo com o que foi planejado.

ID Requisito	ID Caso de Teste	Tipo de Teste	Descrição do Caso de Teste
RF01	CT01	Unitário	Adicionar filme válido (ISO e BR) em uma sala
RF01	CT01a1	Unitário	Adicionar filme com idade negativa (inválido)
RF01	CT01a2	Unitário	Adicionar filme com data inválida
RF01	CT01a3	Unitário	Adicionar filme com nome vazio (inválido)
RF01	CT01a4	Unitário	Adicionar filme com gênero vazio (inválido)
RF01	CT01b	Unitário	Adicionar filme quando já existe outro (substituição)
RF02	CT02	Unitário	Remover filme existente de uma sala
RF02	CT02a	Unitário	Tentar remover filme de uma sala vazia (nenhum filme presente)
RF03	CT03	Unitário	Emissão de ingresso normal com decremento de 1 no número de ingressos
RF03	CT03a	Unitário	Emissão de ingresso quando os ingressos estão zerados (deve lançar erro)
RF03	CT03b	Unitário	Emissão de ingresso considerando idade mínima do filme
RF03	CT03c	Unitário	Simulação de emissão de ingresso com idade maior que mínima (mesmo comportamento)
RF03	CT03d	Unitário	Limite de ingressos: decremento até 0 e erro na próxima emissão
RF04	CT04	Unitário	Verifica status inicial das salas (todas vazias, sem filmes)
RF04	CT04a	Unitário	Verifica status das salas após alterações (inclusão de filmes em algumas)
RF05	CT05	Unitário	Filtrar salas por nome e datas, retornando apenas as correspondentes
RF05	CT05a	Unitário	Nenhuma sala corresponde aos critérios de filtro

RF05	CT05b	Unitário	Retornar múltiplas salas que atendem aos critérios de filtro
RF06	CT06	Unitário	Emissão de ticket válido com decremento de ingressos e verificação da assinatura.
RF06	CT06a	Unitário	Verificação de ticket inválido com assinatura falsa.
RF06	CT06b	Integração	Teste de ticket já utilizado no sistema real (arquivo JSON).
RF06	CT06c	Unitário	Emissão de ticket quando os ingressos do filme estão zerados, esperando erro.
RF06	CT06d	Unitário	Emissão de ticket considerando a idade mínima exigida para o filme.
RF07	CT07	Integração	Testa salvar e restaurar o estado das salas com filmes, verificando a integridade dos dados.
RF07	CT07a	Integração	Testa integridade do estado após várias operações de adição e remoção de filmes.
RF07	CT07b	Sistema	Persistência do estado do sistema: salvar e carregar salas com filmes.
RF07	CT07c	Sistema	Fluxo completo do sistema (fim-a-fim): salvar estado, emitir ticket, verificar persistência.

2. Definição das Técnicas e Critérios de Teste

2.1 Técnicas de Teste

Para garantir a cobertura e a qualidade da aplicação de bilheteria, as seguintes técnicas de teste serão empregadas:

- **Teste de Unidade (Caixa Branca):** Validação dos componentes individuais (módulos, funções, classes) para verificar se funcionam conforme o esperado. Esta é a técnica primária demonstrada nos scripts de teste fornecidos (pytest).
- **Caixa Preta:** Validar o comportamento do sistema sem conhecer o código-fonte, focando nas entradas e saídas (ex: interação com a CLI).

- **Particionamento de Equivalência:** Testar classes de dados válidas e inválidas (ex: idades mínimas válidas ou inválidas, número de ingressos positivo vs. zero).
- **Análise de Valor Limite:** Testar os limites das restrições (ex: emitir o último ingresso disponível, número máximo de salas).
- **Teste de Criptografia e Segurança:** Foco específico na validação dos mecanismos de assinatura digital (RF06) e criptografia do estado do sistema (RF07).

2.2 Critérios de Aceitação

- Todos os casos de teste de unidade automatizados devem passar (80% de sucesso).
- Todos os requisitos funcionais (RF01-RF07) devem ser cobertos por pelo menos um caso de teste (unidade, integração ou manual).
- Nenhum defeito crítico (ex: falha na emissão de ingresso, falha na validação de ticket, perda de dados) deve permanecer sem correção.
- A cobertura de código (coverage) pelos testes de unidade deve ser monitorada e mantida acima de um limite pré-definido (ex: 80%).

2.3 Especificação dos Casos de Teste

Abaixo estão detalhados os casos de teste automatizados que validam requisitos críticos, conforme implementado nos scripts de teste do Pytest.

Caso de Teste	Requisito	Script / Função	Técnica	Dados de Entrada (Mock)	Ação	Resultado Esperado (Assert)
CT01	RF01	test_filmes.py ::	Unidade, Equivalên	Filme: "Filme	add_filme_to_sala	Filme criado na

		test_adicionar_filme_valido	cia	ISO", Data: "2025-12-31"	(...)	sala, ingressos=50, data convertida para ISO se necessário.
CT01 (Negativo)	RF01	test_filmes.py :: test_adicionar_filme_idade_negativa	Valor Limite	Idade: -5	add_filme_to_sala(...)	Deve levantar exceção ValueError.
CT03	RF03	test_ingressos.py :: test_emitir_ticket_mock	Unidade, Mock	Sala(numero=1), Mock da Private Key	issue_ticket(sala)	Retorna dict com assinatura válida; sala.filme.ingressos decrementa (49).
CT03a	RF03	test_ingressos.py :: test_emitir_ticket_sem_ingressos_mock	Valor Limite	sala.filme.ingressos = 0	issue_ticket(sala)	Deve levantar exceção ValueError impedindo a venda.
CT05	RF05	test_filtros.py :: test_filtrar_por_nome_e_data	Unidade	Lista de salas com filmes variados	filter_salas(..., nome="Aven")	Retorna apenas a sala contendo o filme "Aventura".
CT06	RF06	test_ticke	Segurança	Ticket	verify_tic	Retorna

		ts.py :: test_ticket_valido_mock	a, Criptografia	gerado validamente	ket_payload(ticket)	True.
CT06a	RF06	test_tickets.py :: test_ticket_invalido_mock	Segurança	Ticket com assinatura alterada manualmente ("deadbeef")	verify_ticket_payload(ticket)	Deve levantar ValueError ("Assinatura inválida").
CT06b	RF06	test_tickets.py :: test_ticket_ja_utilizado_mock	Segurança (Replay)	Ticket já presente no arquivo used_tickets	verify_ticket_payload(ticket)	Deve levantar ValueError ("Ticket já utilizado").
CT07	RF07	test_storage_crypto.py :: test_encrypt_decrypt	Persistência, Criptografia	Estado (dict) e Senha	encrypt_state -> decrypt_state	O estado descriptografado deve ser idêntico ao original.

3. Cronograma das Atividades de Teste

Planejamento temporal da execução dos testes, com marcos e responsáveis, baseado no mês de Setembro de 2025.

Fase	**Atividade**	**Responsável**	**Início**	**Término**
Planejamento	Preparar casos de	Alexandre	27/10/2025	07/11/2025

	teste, revisar requisitos, configurar ambiente	Fernandes e Pedro Henrique Cavalcanti		
Execução	Executar testes de unidade automatizados (Pytest)	Ryandro da Silva Tavares e Luis Vitor Gomes	10/11/2025	14/11/2025
Execução	Executar testes funcionais manuais (Fluxo de CLI)	Luis Vitor Gomes	17/11/2025	21/11/2025
Relato	Registrar defeitos e analisar resultados dos testes	Luan dos Santos e Pedro Henrique Cavalcanti	24/11/2025	28/11/2025
Encerramento	Revisão final do teste e relatório de cobertura e aceitação	Ryandro da Silva	29/12/2025	01/12/2025

4. Recursos Necessários

4.1 Ferramentas, Ambiente e Dados

Recurso	**Descrição**
Ferramenta	Python 3.13.x, pytest (framework de teste), cryptography (Biblioteca RSA para assinatura), pycryptodome (Biblioteca AES para criptografia de estado).
Ambiente	Ambiente de desenvolvimento local (Windows) com Python e bibliotecas instaladas. Execução via Terminal (CLI).
Dados	Dados de teste simulados (mock data) definidos diretamente nos scripts de teste (ex: test_models.py, test_storage_crypto.py).

4.2 Funções da Equipe

Função	**Responsável**	**Atribuições**
Planejador de Testes	Alexandre Fernandes e Pedro Henrique Cavalcanti	Montar o plano, definir cronograma e critérios, revisar requisitos.
Executor de Testes (Automatizados)	Ryandro da Silva Tavares e Luis Vitor Gomes	Desenvolver e manter os scripts de teste de unidade (pytest), executar a suíte de

		testes.
Executor de Testes (Funcionais)	Luis Vitor Gomes	Realizar testes manuais exploratórios e baseados em casos de uso na CLI.
Analista de Defeitos	Luan dos Santos e Pedro Henrique Cavalcanti	Registrar, classificar e rastrear falhas encontradas (Bugs) no sistema.
Gerente de Testes	Ryandro da Silva Tavares	Coordenar a equipe, gerar o relatório final de testes e aprovar a versão.

5. Relatórios de Limitações e Riscos

5.1 Limitações Técnicas e Operacionais

- **Foco em Testes de Unidade:** Os scripts fornecidos validam a lógica interna (unidades) com sucesso, mas não cobrem testes de integração (interação entre módulos) ou testes de sistema (o fluxo completo da CLI).
- **Ausência de Testes de Carga:** Os dados de teste são de volume baixo (ex: 1 sala, 2 ingressos). O sistema não foi validado sob estresse (ex: múltiplas emissões simultâneas ou um grande número de salas).
- **Ambiente de Teste:** O teste ocorre em ambiente de desenvolvimento local, que pode diferir do ambiente de produção (se houver).
- **Segurança:** Os testes de segurança validam a implementação da criptografia, mas não cobrem vulnerabilidades de segurança da aplicação (ex: injection na CLI, análise de dependências).

5.2 Estratégias de Mitigação

- **Testes Exploratórios Manuais:** A equipe (especialmente Luis Vitor) deve focar em testes manuais exploratórios para cobrir os fluxos de caso de uso (RF01-RF05) que não são validados pelos testes de unidade.
- **Testes de Integração Futuros:** Recomenda-se o desenvolvimento de testes de integração que simulem a entrada do usuário na CLI e verifiquem o estado do sistema (ex: usando subprocess ou click.testing).

- **Monitoramento em Produção:** Na ausência de testes de carga, o sistema deve ser monitorado de perto após o lançamento para identificar gargalos de performance.

6. Conclusão

Este plano de testes, atualizado com base nos scripts de teste de unidade desenvolvidos, estabelece como, quando e por quem os testes do Sistema de Bilheteria serão realizados. A combinação de testes de unidade automatizados (focados em lógica e segurança) e testes funcionais manuais (focados no fluxo do usuário) visa garantir que todos os requisitos funcionais e de segurança sejam validados antes da entrega final.