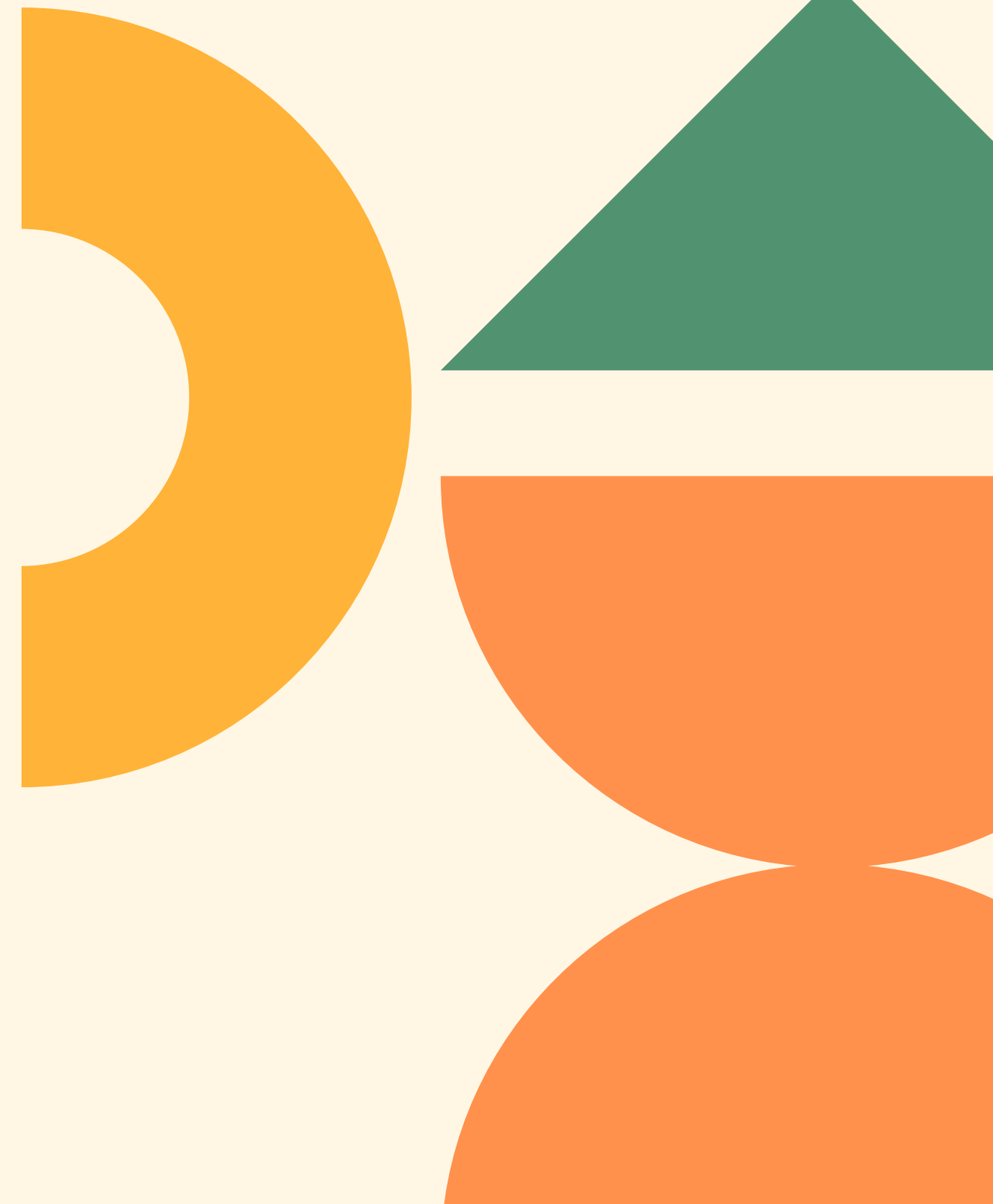




PROJETO LOCADORA DE FILMES

Autores: Ana Luíza Gonçalves Leite, Gabriel Lucas, João Victor Vasconcellos, Letícia Gomes S. Costa, Lucas Silva Tanure, Marcos Vinicius e Murilo Silva

- 01** OBJETIVO
- 02** ARQUITETURA E TECNOLOGIAS UTILIZADAS
- 03** FUNCIONALIDADES
- 04** CASOS DE TESTE IMPLEMENTADOS
- 05** COBERTURA DOS TESTES
- 06** ANÁLISE DETALHADA
- 07** CONCLUSÃO



OBJETIVO

- Desenvolver um sistema funcional para gestão de locadora de filmes
- Automatizar cadastros, locações, devoluções e relatórios
- Aplicar testes unitários com foco na qualidade do software
- Medir a cobertura dos testes com JaCoCo
- Validar a confiabilidade do sistema

ARQUITETURA E TECNOLOGIAS UTILIZADAS

- Linguagem: Java
- Banco de dados: SQLite
- Testes: JUnit 5 e Mockito
- Cobertura: JaCoCo (HTML Reports)
- Arquitetura modular:
 - model – entidades
 - service – regras de negócio
 - ui – entrada e saída no console

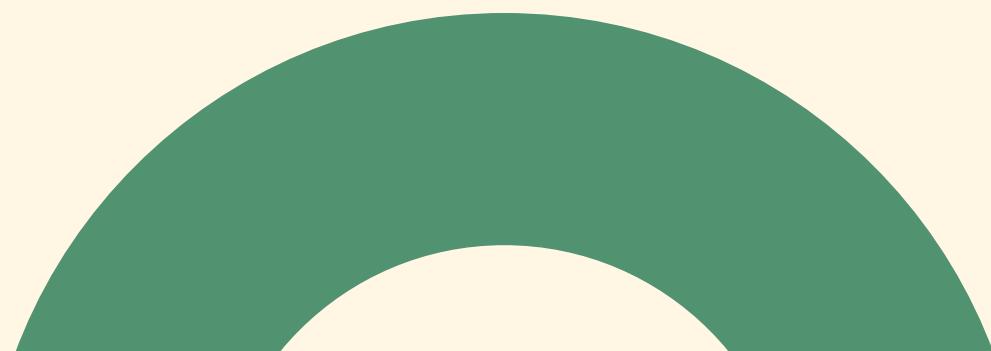
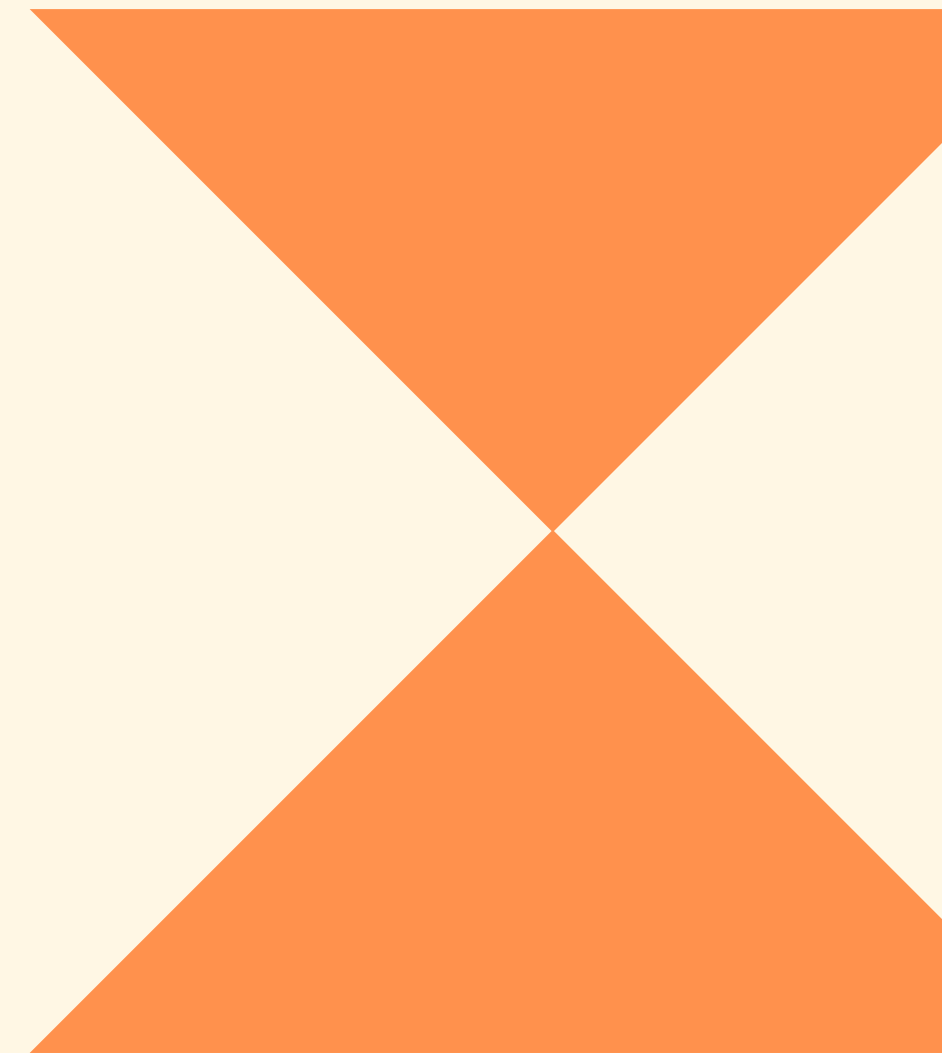
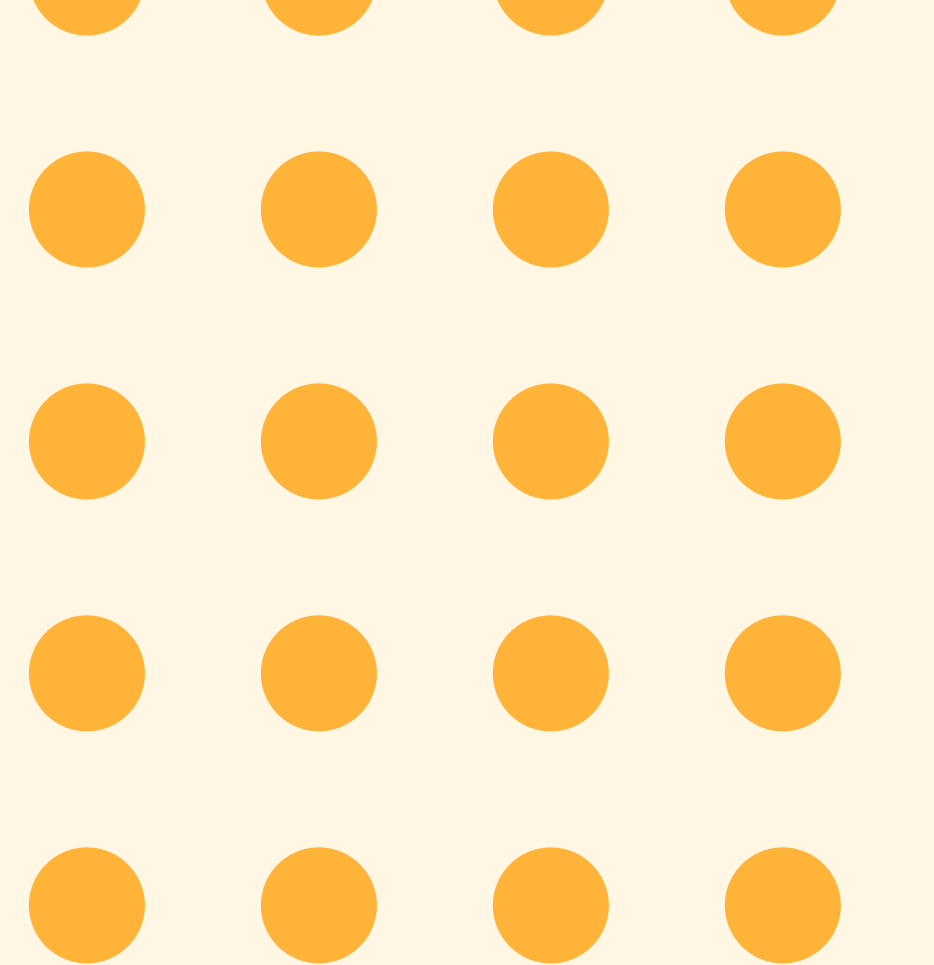


FUNCIONALIDADES

- Cadastro e edição de filmes e clientes
- Locações com controle de estoque e devoluções
- Envio de notificações (atrasos, promoções, avisos)
- Geração de relatórios por período e filmes mais alugados
- Persistência completa com SQLite

CASOS DE TESTE IMPLEMENTADOS




- Total de 52 testes unitários, divididos em:
 - ServiceTest – 26 testes
 - UiTest – 16 testes
 - ModelTest – 10 testes
- Testes de comportamento esperado, exceções, mocks e entrada simulada
- Validação de funcionalidades críticas e mensagens exibidas ao usuário



RESULTADOS COM JACOCO

Locadora Filmes

Locadora Filmes

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 com.seuprojeto.locadora.ui	<div><div></div></div>	53%	<div><div></div></div>	63%	16	35	68	147	1	11	0	2
 com.seuprojeto.locadora.service	<div><div></div></div>	89%	<div><div></div></div>	86%	8	57	34	312	1	31	0	4
 com.seuprojeto.locadora.model	<div><div></div></div>	82%	<div><div></div></div>	38%	22	103	22	158	10	85	0	7
Total	446 of 2.299	80%	44 of 129	65%	46	195	124	617	12	127	0	13



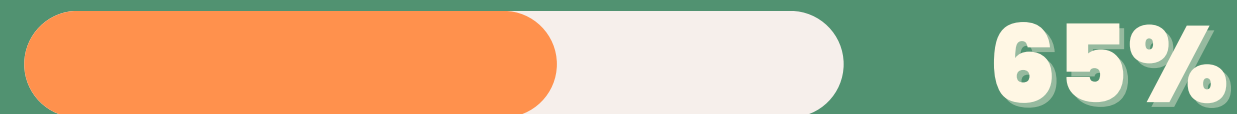
80%

de cobertura de linhas
(493 de 617 linhas)



78%

de cobertura de
instruções



65%

de cobertura de
ramificações lógicas



91%

dos métodos foram
testados (115 de 127)



100%

das classes foram
cobertas

COBERTURA DOS TESTES

ANÁLISE DETALHADA

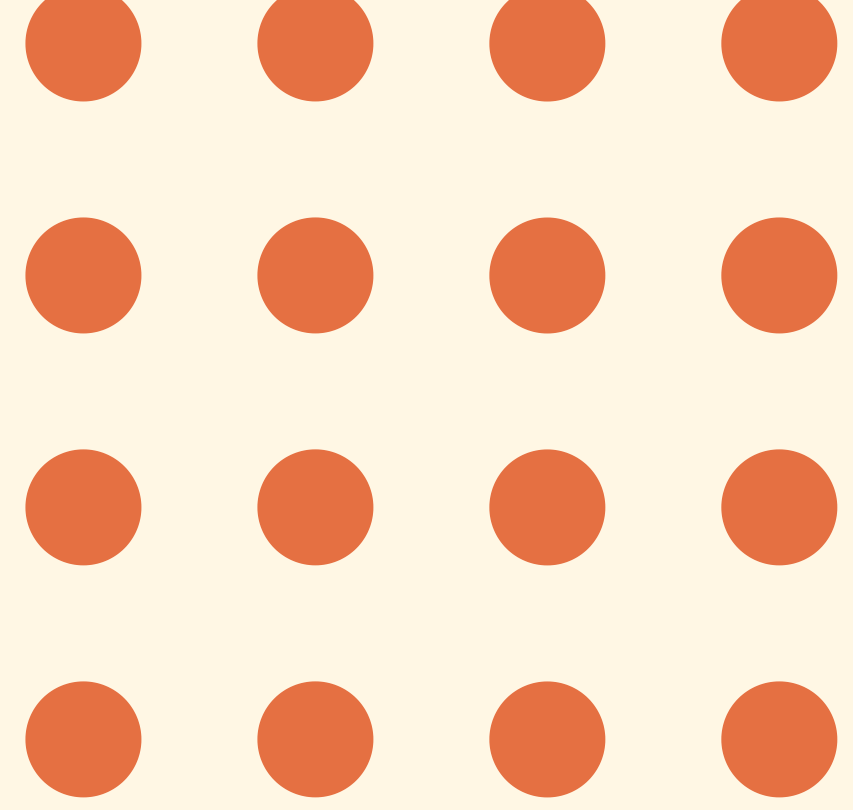
- Total de classes: 13
- Total de métodos: 127
- Total de ramificações: 129
- Cobertura atingida:
 - 110 métodos testados
 - 85 ramificações cobertas
 - Cobertura significativa nas regras de negócio e validações



CONCLUSÕES

- Sistema modular e bem estruturado
- Testes abrangentes cobrindo regras de negócio e interface
- Cobertura sólida que reforça a qualidade do código
- Ferramentas como JUnit, Mockito e JaCoCo facilitaram validação
- Projeto pronto para manutenção e expansão futura





OBRIGADO

