UNIVERSIDADE SÃO JUDAS TADEU

SISTEMAS DE INFORMAÇÃO

GESTÃO E QUALIDADE DE SOFTWARE

JOÃO VITOR GOMES PEREIRA - 82329432

MATEUS HENRIQUE SALVADOR - 82323463

FELIPE CARDOSO SILVA – 82326693

PROJETO A3 – BOOKSCORE

PROF.º DOCENTE – ROBSON CALVETTI

SÃO PAULO

2025

SUMÁRIO

1 – PLANEJAMENTO DE TESTE DE SOFTWARE

O planejamento de testes é uma etapa estratégica essencial para garantir a qualidade e a confiabilidade de qualquer sistema de software. Nesse processo, definimos o que será testado, como será testado e quando cada atividade ocorrerá, alinhando expectativas entre equipe de desenvolvimento, testes e stakeholders.

No BookScore, o planejamento de testes organiza desde a criação de casos de uso funcional até a verificação de requisitos não-funcionais como desempenho e usabilidade

1.1 Cronograma de Atividades

Semana	Fase	Atividade	Duração Estimada	Responsável
1	Planejamento	Definir escopo dos testes, objetivos, critérios de entrada/saída	2 dias	Mateus/João
1	Planejamento	Levantar casos de uso e funcionalidades principais	2 dias	Mateus/Felipe
2	Análise de Testes	Identificar tipos de testes (funcional, usabilidade, segurança etc.)	1 dia	Mateus
2	Análise de Testes	Elaborar plano de testes (documento)	2 dias	Mateus
2	Análise de Testes	Definir ambiente de testes (dados, usuários, livros fictícios)	2 dias	Mateus/Felipe

3	Projeto de Casos de Teste	Criar casos de teste para cada funcionalidade (login, cadastro, etc.)	3 dias	Mateus
3	Projeto de Casos de Teste	Criar matriz de rastreabilidade (casos x requisitos)	1 dia	Mateus
4	Implementação dos Testes	Preparar scripts de teste (se automatizado) ou formulários manuais	3 dias	Mateus/João
4	Implementação dos Testes	Revisar e validar os casos de teste com o time	2 dias	Mateus/João
5	Execução dos Testes	Executar testes funcionais (CRUD, navegação, ranking, etc.)	1 semana	Felipe
6	Execução dos Testes	Executar testes de usabilidade e testes exploratórios	2 dias	Mateus/Usuários
6	Execução dos Testes	Executar testes de regressão (após correções)	2 dias	João
7	Registro e Análise de Erros	Documentar bugs encontrados, priorizar e repassar para correção	3 dias	Mateus/João
7	Registro e Análise de Erros	Retestar funcionalidades corrigidas	2 dias	Felipe
8	Encerramento	Gerar relatório final de testes	1 dia	Equipe completa

8	Encerramento	Reunião de lições aprendidas	1 dia	Equipe completa

1.2 Alocação de Recursos

Semana	Atividade	Recurso Alocado	Carga Horária Estimada
1	Definir escopo dos testes, objetivos, critérios de entrada/saída	QA / Dev	16
1	Levantar casos de uso e funcionalidades principais	QA / Dev	16
2	Identificar tipos de testes (funcional, usabilidade, segurança etc.)	QA	8
2	Elaborar plano de testes (documento)	QA	16
2	Definir ambiente de testes (dados, usuários, livros fictícios)	QA / DevOps	16
3	Criar casos de teste para cada funcionalidade	QA	24
3	Criar matriz de rastreabilidade (casos x requisitos)	QA	8
4	Preparar scripts de teste ou formulários manuais	QA / Dev	24
4	Revisar e validar os casos de teste com o time	QA / Dev	16
5	Executar testes funcionais (CRUD, navegação, ranking, etc.)	QA	40
6	Executar testes de usabilidade e testes exploratórios	QA / Usuários	16
6	Executar testes de regressão (após correções)	QA	16
7	Documentar bugs encontrados, priorizar e repassar para correção	QA / Dev	24
7	Retestar funcionalidades corrigidas	QA	16
8	Gerar relatório final de testes	QA	8
8	Reunião de lições aprendidas (retrospectiva)	Equipe completa	8

1.3 Marcos do Projeto

Nome do Marco	Descrição	Semana Prevista	Critério de Conclusão
Início do Planejamento	Início da definição do escopo, objetivos e critérios de testes.	Semana 1	Documento de escopo e objetivos aprovado.
Finalização do Plano de Testes	Plano de testes documentado com tipos de testes, ambiente e dados definidos.	Semana 2	Plano de testes revisado e validado pela equipe.
Casos de Teste Criados	Todos os casos de teste documentados e revisados.	Semana 3	Casos de teste aprovados e prontos para execução.
Início da Execução dos Testes	Início da execução dos testes funcionais e de usabilidade.	Semana 5	Ambiente de testes pronto e execução iniciada.
Encerramento da Execução dos Testes	Todos os testes executados, incluindo regressão e retestes.	Semana 7	Todos os testes executados e bugs críticos resolvidos.
Relatório Final de Testes	Geração do relatório final com resultados e cobertura de testes.	Semana 8	Relatório entregue e reunião de encerramento realizada.

2 - PLANO DE PROJETO

2.1. Introdução

O BookScore é uma aplicação desktop Java, desenvolvida como parte do Projeto A3 do curso de Sistemas de Informação. Ela tem como objetivo proporcionar uma plataforma para que leitores possam cadastrar livros, avaliá-los (nota de 0 a 10), escrever reviews e consultar um ranking dos títulos mais bem avaliados. Há ainda um módulo administrativo para gerenciamento de contas (criar, editar e excluir usuários).

Este Plano de Projeto documenta, de forma estruturada, todas as diretrizes e informações necessárias para o desenvolvimento, implementação e entrega do BookScore, seguindo práticas ágeis (Scrum) e garantindo visibilidade clara sobre escopo, recursos e cronograma.

2.2. Planejamento do Projeto

2.2.1. Metodologia de Trabalho

Abordagem Ágil (Scrum): Dividimos todo o desenvolvimento em 4 sprints quinzenais (cada sprint com duração de 2 semanas), priorizando entregas incrementais e validações constantes.

Reuniões de Sprint: Ao início de cada sprint, faremos uma Sprint Planning (planejamento detalhado das tarefas). Ao final, uma Sprint Review (demonstração das funcionalidades entregues) e Sprint Retrospective (lições aprendidas e melhorias para o próximo ciclo).

Kanban Board: Usaremos um quadro Kanban (GitHub Projects, Trello ou equivalente) com colunas: Backlog, Em Andamento, Em Revisão/Testes e Concluído.

2.2. Cronograma Geral

Sprint	Período	Objetivo Principal	Marcos / Entregas
Sprint 1	Semanas 1 e 2	Estrutura do projeto + Autenticação (Login/Cadastro)	Estrutura MVC, telas de Login e Cadastro, validações básicas, conexão com BD
Sprint 2	Semanas 3 e 4	Cadastro e Avaliação de Livros	Model e Controller de Livro/Avaliação, telas de cadastro e avaliação, mensagens
Sprint 3	Semanas 5 e 6	Ranking e Navegação	Tela de Ranking, lógica de ordenação, botões 'Voltar', refatoração de código
Sprint 4	Semanas 7 e 8	Módulo Admin e Finalização	Login Admin, gerenciamento de usuários, testes fim a fim, versão final

2.3. Escopo

2.3.1. Funcionalidades Principais (Dentro do Escopo)

1. Autenticação de Usuário (Login/Cadastro)
    - Tela de Login (e-mail + senha)
    - Tela de Cadastro (nome, e-mail, username, sexo, senha, confirmação de senha, até três gêneros de preferência)
    - Validações de formato, duplicidade de e-mail/username e correspondência de senhas.
    - Redirecionamento ao menu principal (usuário comum ou admin).

2. Menu Principal (Usuário Comum)
    - Botão “Avaliar Livro”
    - Botão “Cadastrar Livro”
    - Botão “Exibir Ranking”
    - Botão “Sair”

3. Avaliação de Livros
    - Seleção de livro existente
    - Nota de 0 a 10
    - Review (texto opcional)
    - Feedback de sucesso/erro

4. Cadastro de Livros
    - Informar título, autor, gênero (romance / ficção / técnico) e ano (opcional)
    - Verificar duplicidade de cadastro antes de inserir
    - Feedback de sucesso/erro

5. Ranking de Livros
    - Listagem de livros ordenada por:
        - Nota média (decrescente)
        - Número de avaliações (decrescente)
        - Nome do livro (ordem alfabética)
    - Botão “Voltar” para retornar ao menu principal

6. Logout com Confirmação
    - Ao clicar em “Sair”, exibir confirmação (“Deseja realmente sair?”)
    - Se confirmado, voltar à tela de Login

7. Módulo Admin
    - Login especial: username = admin e senha = admin
    - Menu principal (idem ao de usuário comum) + opção “Manter Usuários”

8. Tela de Gerenciamento de Usuários:
    - Criar Usuário (mesmos campos do cadastro normal)
    - Editar Usuário (buscar por username, alterar dados)
    - Excluir Usuário (buscar por username, confirmar exclusão)
    - Botão “Voltar” para retornar ao menu admin

9. Banco de Dados
    - Tabelas principais:
        - Usuario (id, nome, email, username, sexo, senha, preferências…)
        - Livro (id, título, autor, gênero, ano, etc.)
        - Avaliacao (id, id_usuario, id_livro, nota, review, data)

2.3.2. Exclusões (Fora do Escopo)

- Integração com redes sociais para login (OAuth).
- Versão mobile ou web (apenas desktop Java).
- Internacionalização (aplicação a apenas português brasileiro).
- Backup automatizado de banco de dados via arquitetura.
- Funcionalidades de comentários/votação em reviews.

2.4. Recursos

2.4.1. Humano

- João Vitor – Analista de Requisitos e Testes Manuais
- Felipe Cardoso – Desenvolvedor Back-end (Java, JDBC, lógica de negócio)
- Mateus Henrique – Desenvolvedor Front-end (Swing/AWT)

Obs: Cada membro atuará em todas as fases (análise, design, codificação, testes e documentação), mas com responsabilidades principais distribuídas conforme descrito.

2.4.2. Tecnológicos

- IDE Java: NetBeans
- JDK 17: Para compilação e execução
- Banco de Dados: MySQL local
- Controle de Versão: Git + GitHub
- Documentação: Microsoft Word

2.4.3. Materiais e Outros

- Acesso à internet (para GitHub e/ou referência de bibliotecas)
- Computador compatível com Java e IDE instalada

2.5. Estimativas de Projeto

2.5.1. Macroestimativa por Sprint

Sprint	Duração	Esforço Total (horas)	Observações
Sprint 1	2 semanas	70	Estrutura MVC, telas de Login/Cadastro, validações
Sprint 2	2 semanas	80	CRUD de Livro e Avaliação, telas, mensagens
Sprint 3	2 semanas	60	Tela de Ranking, ordenação, navegação
Sprint 4	2 semanas	70	Admin, testes, versão final
Total	8 semanas	280	Média de 35 h/semana por membro

2.5.2. Detalhamento de Tarefas e Esforço (Sprint 1 - Exemplo)

Tarefa	Responsável	Estimativa	Descrição
Definir estrutura de pastas (MVC)	Felipe	4h	Criar pacotes model, view, banco, resources
Desenvolver classe Usuario.java (Model)	João	6h	Atributos, getters/setters, construtores
Criar tela TelaLogin.java (View)	Mateus	8h	Layout com campos e botões, validações básicas
Criar tela TelaCadastroUsuario.java (View)	Mateus	8h	Campos: nome, e-mail, username, senha, etc.
Implementar CntrlUsuarios.java	Felipe	10h	Lógica de login, cadastro, validações
Configurar conexão JDBC e criar script SQL	João	6h	Criação de tabela 'usuario', conexão com BD
Testes manuais Login/Cadastro	João e Mateus	6h	Executar fluxos, validar funcionamento
Configurar GitHub + primeiro commit	Felipe	4h	Repositório, README.md

Documentação parcial (Plano de Projeto, Requisitos)	Todos	8h	Redigir e revisar documentos iniciais

3 – DOCUMENTO DE REQUISITOS

3.1. Introdução

Este Documento de Requisitos descreve, de forma detalhada, as funcionalidades que o BookScore deve oferecer, bem como as restrições, premissas e critérios de aceitação. Apresentando de forma clara e objetiva todas as necessidades que o sistema BookScore deve atender, garantindo alinhamento entre equipe de desenvolvimento e partes interessadas (professor, avaliadores, cliente fictício). Ele serve de base para o desenvolvimento, testes e validação do sistema.

3.2. Requisitos Funcionais (RF)

RF001 – Autenticação de Usuário

- Descrição: O sistema deve permitir que o usuário faça login usando e-mail (ou username) e senha.
- Entrada: E-mail (ou username) e senha.
- Processamento: Verificar credenciais no banco de dados (usuario), correspondência exata de senha.
- Saída: Se válido, redireciona ao menu principal (usuário comum ou admin); se inválido, exibe mensagem “E-mail/Usuário ou senha incorretos”.

RF002 – Cadastro de Novo Usuário

- Descrição: O sistema deve permitir que um usuário crie uma nova conta.
- Entrada: Nome, e-mail, username, sexo, senha, confirmação de senha, até três gêneros de preferência.
- Processamento:
- Validar formatos (e-mail válido, username não vazio).
- Verificar duplicidade de e-mail e username.
- Confirmar que “senha” e “confirmação de senha” coincidem.
- Inserir um novo registro na tabela usuario.
- Saída: Se bem-sucedido, mensagem “Cadastro realizado com sucesso” e redireciona ao login. Se falhar, exibir mensagem de erro.

RF003 – Logout

- Descrição: O usuário deve poder sair do sistema a partir do menu principal, apontando para a tela de login.
- Entrada: Clique no botão “Sair” no menu principal.
- Processamento: Solicitar confirmação (“Deseja realmente sair?”).
- Saída: Se confirmado, encerrar sessão atual e exibir tela de login; senão, retornar ao menu principal.

RF004 – Cadastro de Livros

- Descrição: Usuários autenticados devem poder cadastrar um novo livro.
- Entrada: Título, autor, gênero (romance/ficção/técnico) e ano de publicação (opcional).
- Processamento:
- Verificar se as informações mínimas (título, autor, gênero) não estão vazias.
- Checar duplicidade (título + autor).
- Inserir novo registro na tabela livro.
- Saída: Mensagem “Livro cadastrado com sucesso” ou mensagem de erro (duplicidade, campo inválido etc.).

RF005 – Avaliação de Livros

- Descrição: Usuários autenticados devem avaliar livros já cadastrados, atribuindo nota e review opcional.
- Entrada: Seleção de livro (lista dropdown ou pesquisa), nota (0 a 10), review (texto opcional).
- Processamento:
- Validar se nota está no intervalo [0,10].
- Inserir registro na tabela avaliacao, vinculando id_usuario e id_livro.
- Saída: Mensagem “Avaliação registrada com sucesso” ou mensagem de erro (erro de banco, nota inválida etc.).

RF006 – Exibir Ranking de Livros

- Descrição: Usuários acessam um ranking que lista os livros mais bem avaliados.
- Entrada: Ação “Exibir Ranking” no menu principal.
- Processamento:
- Consultar tabela avaliacao, calcular média de notas por id_livro.
- Ordenar por média (decrescente), em caso de empate ordenar por número de avaliações (decrescente) e, ainda empatado, ordenar por título (ordem alfabética).
- Exibir resultados em uma tabela (colunas: posição, título, autor, média, número de avaliações).
- Saída: Tela de Ranking com lista ordenada; botão “Voltar” para menu principal.

RF007 – Gestão de Usuários (Módulo Admin)

- Pré-condição: Usuário autenticado como “admin”.
    - Sub-RF007.1 – Criar Usuário
        - Entrada: Mesmos campos do RF002.
        - Processamento: Igual ao RF002, mas realizado a partir da tela ‘Gerenciar Usuários’.
        - Saída: Mensagem de sucesso/erro.
    - Sub-RF007.2 – Editar Usuário
        - Entrada: Username do usuário a ser editado.
        - Processamento:
            - Buscar usuário no banco (tabela usuario).
            - Exibir dados em campos editáveis.
            - Validar alterações (mesma lógica de RF002).
            - Atualizar registro.
        - Saída: Mensagem “Usuário atualizado com sucesso” ou mensagem de erro.

Sub-RF007.3 – Excluir Usuário

- Entrada: Username do usuário a ser excluído.
- Processamento:
    - Confirmar ação (“Deseja realmente excluir este usuário?”).
    - Deletar da tabela usuario (e, opcionalmente, todas as avaliações vinculadas).
- Saída: Mensagem “Usuário excluído com sucesso” ou mensagem de erro.

RF008 – Mensagens de Feedback

- Descrição: Todas as operações (login, cadastro, avaliação, ranking, CRUD de usuários) devem exibir mensagens claras de sucesso ou erro, indicando o que ocorreu.
- Requisito Específico: Utilizar métodos utilitários (classe Utils.java ou similar) para exibir JOptionPane com legendas padrão (ex: “Sucesso”, “Erro”, “Atenção”).

3.3. Requisitos Não Funcionais (RNF)

RNF001 – Usabilidade

- Interfaces devem ser intuitivas: botões com legendas claras (“Login”, “Cadastrar”, “Enviar Avaliação”, “Voltar”).
- Todos os formulários obrigatórios devem possuir validações visuais (mensagens de erro próximas ao campo).
- Botão “Voltar” presente em todas as telas (exceto Login)

RNF002 – Desempenho

- Operações de CRUD devem responder em até 2 segundos (considerando banco local).
- Geração de ranking (consulta e ordenação) não deve ultrapassar 3 segundos para até 1 000 registros de avaliações.

RNF003 – Confiabilidade / Robustez

- Tratamento de exceções em todas as operações de banco de dados (ex: falha de conexão, erros SQL).
- Se o banco de dados ficar indisponível, exibir mensagem adequada (“Erro interno: banco de dados indisponível.”).
- Salvamento atômico das transações de cadastro (usar transações JDBC ou controle de commit/rollback).

RNF004 – Segurança

- Senha armazenada de forma simples (como texto) não é ideal, mas para este escopo acadêmico, o armazenamento pode ser em texto plano no banco (observação: ambiente controlado, sem dados sensíveis).
- Caso deseje melhorar, documentar criptografia de senha (ex: hash MD5/BCrypt) como recomendação futura.

RNF005 – Manutenibilidade

- Padrão MVC garante separação de responsabilidades: código de apresentação desacoplado da lógica de negócio e do acesso a dados (embora Controllers façam CRUD diretamente).
- Comentários em métodos críticos (login, ranking, validações).
- Documentação mínima das classes públicas.

RNF006 – Compatibilidade

- Aplicação deve rodar em Windows, Linux ou macOS, desde que Java 17 esteja instalado.
- Testado em ambientes Windows 10 (64 bits)

3.4. Restrições

- Linguagem e Plataforma: Projetado exclusivamente em Java (JDK 17).
- Banco de Dados: Uso de MySQL ou SQLite local; não há suporte a servidores remotos nessa fase.
- Interface Gráfica: Swing/AWT (sem JavaFX ou frameworks web).
- Usuário Admin: Credenciais fixas: `username = admin` e `senha = admin`.
- Gêneros de Livro: Limitados a três opções fixas (romance, ficção, técnico) no cadastro.
- Avaliação: Nota inteira de 0 a 10 (sem casas decimais).

3.4. Premissas

- O usuário trabalha em ambiente com Java (JDK) e IDE configurada corretamente.
- O banco de dados será inicializado antes do primeiro Login (tabelas criadas por script SQL ou pela própria aplicação no primeiro acesso).
- Não há necessidade de autenticação via internet (login offline, tudo local).
- O módulo admin não precisa de níveis adicionais de permissão — apenas “admin” versus “usuário comum”.

4 - PLANEJAMENTO DE TESTES

4.1. Plano de Testes

4.1.1. Introdução

O Plano de Testes documenta em detalhes como iremos verificar e validar o BookScore antes da entrega final. Ele consolida os objetivos de qualidade, define métodos e critérios de aceitação, e orienta a equipe nos esforços de garantia de qualidade, assegurando que o sistema atenda aos requisitos funcionais e não-funcionais.

4.1.2 Escopo

Este plano abrange exclusivamente os fluxos do usuário comum em ambiente desktop Java/Swing:

- Autenticação (login & logout)
- Cadastro de usuário
- Cadastro de livro
- Avaliação de livro
- Exibição de ranking
- Navegação entre telas e feedback ao usuário
- Funcionalidades administrativas (CRUD de usuários pelo admin)

Fora de escopo nesta fase:

- Integrações externas (APIs de redes sociais, backup em nuvem)

4.1.3 Objetivos

1. Validar cada requisito funcional (RF001–RF006) segundo critérios de aceitação bem definidos.
2. Confirmar o atendimento aos requisitos não-funcionais (RNF001–RNF003) de usabilidade, desempenho e confiabilidade.
3. Detectar e classificar defeitos (functionalidades quebradas, mensagens incorretas, erros de banco).
4. Garantir que os fluxos críticos sejam executados sem interrupções e dentro de tempos de resposta aceitáveis.

4.1.4 Requisitos a serem testados

RF / RNF	Descrição	Critério de Sucesso
RF001	Cadastro de usuário	Novo usuário criado no BD e recebe confirmação
RF002	Login e logout	Autenticação bem-sucedida / logout retorna ao login
RF003	Cadastro de livro	Livro salvo no BD com todos os dados corretos
RF004	Avaliação de livro	Avaliação gravada com nota e comentário (se houver)
RF005	Exibição de ranking	Ranking exibido conforme critérios de ordenação
RF006	Navegação e mensagens de feedback	Botão 'Voltar' funciona; pop-ups claros
RNF001	Usabilidade	Fluxos manuais sem confusão
RNF002	Desempenho	CRUD em ≤2 s; ranking em ≤3 s (500 avaliações)
RNF003	Confiabilidade	Sistema não trava em falhas de SQL

4.1.5 Estratégias e Tipos de Testes

Testes Manuais Funcionais

- Cobertura: 100 % dos casos de uso do usuário comum.
- Procedimento: Seguir roteiros passo a passo; documentar cada resultado e captura de tela.

Testes Unitários (Caixa-Branca)

- Ferramenta: JUnit 5.
- Foco: Métodos de validação em UsuarioController, LivroController e AvaliacaoController.
- Meta: Cobertura mínima de 80% de linhas de código nos controllers.

Testes de Integração

- Escopo: Fluxo completo do front-end ao banco SQLite.
- Ambientes: Windows 10 e Ubuntu 22.04 para verificar compatibilidade.

Testes de Usabilidade

- Método: Sessões de walkthrough com 3 usuários-teste; tempo médio de cada fluxo; coleta de feedback qualitativo.
- Critério: Nenhum usuário-teste deve ficar mais de 30 s perdido em uma tela.

Testes de Desempenho

- Medição: Logging interno de timestamps para cada operação CRUD e ranking.
- Critério: CRUD → ≤ 2 s; Ranking → ≤ 3 s com base em 500 registros de avaliação.

Teste de Regressão

- Reexecutar todos os testes manuais e unitários a cada correção de bug.

4.1.6 Ferramentas

- IDE: Eclipse ou NetBeans
- Banco de Dados: SQLite + SQLite Browser
- Teste Unitário: JUnit 5
- Controle de Versão: Git + GitHub Actions (para build e testes automatizados)
- Registro de Defeitos: Planilha Google Sheets ou Excel
- Captura de Tela: Greenshot ou ferramenta equivalente

4.1.7 Recursos Humanos e Materiais

Recurso	Responsável	Descrição
Ambiente Windows 10	Toda a equipe	Configuração Java 17 + IDE
Ambiente Ubuntu 22.04	Toda a equipe	Configuração idêntica
Banco de Teste	Mateus	Script de dados e arquivo .db
Criação de TC	João	Redação e organização de casos
Execução de Testes	Felipe	Documentação de resultados
JUnit Scripts	Mateus	Desenvolvimento e manutenção

4.1.8 Cronograma das Atividades

Nº	Atividade	Responsável	Data Início	Data Fim
M1	Configuração de ambientes de teste	Todos	08/06/2025	09/06/2025
M2	Elaboração de casos de teste	João	10/06/2025	12/06/2025
M3	Implementação de testes unitários	Mateus	11/06/2025	15/06/2025
M4	Execução de testes manuais	Felipe	13/06/2025	17/06/2025
M5	Testes de integração	Todos	16/06/2025	18/06/2025
M6	Testes de desempenho e usabilidade	Felipe	19/06/2025	21/06/2025
M7	Consolidação e relatório de resultados	João	22/06/2025	23/06/2025

4.1.9 Marcos do Projeto

Marco	Data	Critério de Conclusão
M1: Ambiente Pronto	09/06/2025	Ambientes Windows e Ubuntu configurados
M2: TC Documentados	12/06/2025	100% dos casos de uso mapeados em casos de teste
M3: Unitários Concluídos	15/06/2025	Cobertura ≥80% nos controllers
M4: Manuais Concluídos	17/06/2025	Todos os casos manuais executados
M5: Integração Validada	18/06/2025	Fluxos ponta-a-ponta sem falhas críticas
M6: Desempenho Aprovado	21/06/2025	Todos os tempos de resposta dentro do esperado
M7: Relatório Finalizado	23/06/2025	Documento de testes entregue e revisado

4.2. Casos de Teste

ID	Descrição	Pré-condição	Passos Principais	Resultado Esperado
TC01	Cadastro de usuário com dados válidos	App aberto	1. Clicar em 'Cadastrar' 2. Preencher campos 3. Enviar	Mensagem Sucesso + retorno ao login
TC02	Cadastro de usuário com e-mail duplicado	Usuário criado	Mesmos passos acima, com e-mail repetido	Mensagem 'E-mail já existe'
TC03	Login com credenciais corretas	Usuário cadastrado	1. Preencher login e senha 2. Entrar	Tela inicial exibida
TC04	Login com senha incorreta	Usuário cadastrado	Mesmos passos acima, com senha inválida	Mensagem 'Usuário ou senha incorretos'
TC05	Cadastro de livro válido	Usuário logado	1. Clicar em 'Cadastrar Livro' 2. Preencher campos 3. Salvar	Mensagem Sucesso
TC06	Cadastro de livro duplicado	Livro existente	Mesmos passos acima, com dados existentes	Mensagem 'Livro já cadastrado'
TC07	Avaliação de livro válida	Livro cadastrado	1. Acessar 'Avaliar Livro' 2. Selecionar livro e nota 3. Enviar	Mensagem sucesso
TC08	Avaliação com nota fora do intervalo	Livro cadastrado	Mesmos passos acima, com nota inválida	Mensagem 'Nota inválida. Informe 0-10'
TC09	Exibição de ranking com dados	Ao menos 1 avaliação	Clicar em 'Exibir Ranking'	Ranking correto + botão Voltar
TC10	Logout com confirmação	Usuário logado	1. Clicar em 'Sair' 2. Confirmar	Retorna à tela de login

4.3. Roteiro de Testes

Preparação do Ambiente

- Instalar Java 17 e IDE nos dois SO’s.
- Importar projeto do GitHub.
- Carregar banco de testes (script SQL).

Execução de Testes Manuais

- Seguir rigorosamente cada caso de teste (TC01–TC10).
- Registrar em planilha: data, hora, SO, resultado e anomalias.
- Capturar tela de cada passo de falha.

Execução de Testes Unitários

- Rodar suíte JUnit no IDE / CI.
- Verificar relatórios de cobertura.
- Corrigir falhas e reexecutar.

Testes de Integração

- Simular fluxo completo (login→cadastro→avaliação→ranking→logout).
- Confirmar persistência no SQLite Browser.

Testes de Desempenho e Usabilidade

- Medir tempos com logging interno e cronômetro manual.
- Realizar walkthroughs com 3 usuários, coletar feedback.

Análise de Resultados e Correções

- Compilar defeitos em planilha, priorizar e atribuir correções.
- Validar correções com reexecução dos casos afetados.

Encerramento de Testes

- Consolidar relatório final.
- Arquivar artefatos de teste no repositório (/tests).
- Apresentar resultados ao professor em reunião de validação.

5 - GESTÃO DE CONFIGURAÇÃO DE SOFTWARE

A gestão de configuração de software é um processo essencial dentro da engenharia de software que garante o controle e a rastreabilidade de todas as modificações realizadas nos artefatos de um projeto. No contexto do aplicativo BookScore, esse processo é fundamental para manter a consistência, a integridade e a qualidade do software ao longo de seu ciclo de vida.

5.1. Itens de Configuração (SCI - Software Configuration Items)

No projeto BookScore, os principais itens de configuração incluem:

- Código-fonte Java das funcionalidades do sistema (classes de modelo, controle e interface).
- Arquivos de configuração e propriedades (ex: config.properties).
- Scripts SQL para criação e população do banco de dados.
- Documentos de requisitos e planos de teste.
- Relatórios e arquivos auxiliares (diagramas UML, protótipos, etc).

Todos os SCIs são nomeados e organizados de forma padronizada, com versão e data de modificação incluídas nos metadados dos arquivos.

5.2. Controle de Versão

O projeto utiliza o sistema de controle de versão Git, com repositório hospedado no GitHub. As boas práticas adotadas incluem:

- Criação de branches para funcionalidades novas (feature/), correções (fix/) e testes (test/).
- Uso de commits com mensagens claras e padronizadas.
- Pull requests para revisão de código antes da integração à branch principal (main).
- Marcação de versões usando "tags" para releases significativos.

5.3. Controle de Alterações

As alterações em qualquer artefato passam por um processo de aprovação simples:

- As tarefas são abertas como issues no GitHub ou em lista de controle.
- Cada modificação é documentada e associada a uma tarefa.
- Após implementada, é feita uma revisão e os testes são executados.

Esse processo visa evitar falhas, melhorar a rastreabilidade e garantir que as alterações sigam um fluxo controlado.

5.4. Gestão de Impacto

Antes de aplicar qualquer modificação, são avaliadas as dependências entre os módulos do sistema:

- Uma alteração na lógica de avaliação, por exemplo, pode impactar o ranking e os relatórios.
- Avalia-se quem são os membros impactados e quais partes do sistema devem ser reavaliadas.
- São aplicados testes regressivos para garantir que a funcionalidade existente não foi comprometida.

5.5. Auditoria de Configuração

1. A auditoria é realizada através de:
2. Revisão manual de código antes da aprovação de pull requests.
3. Checklist para verificar aderência a padrões definidos (nomenclatura, comentários, tratamento de exceções).
4. Verificação de documentação associada a cada alteração.

5.6. Relatório de Status

A cada commit no repositório, é gerado um histórico automático que mostra:

- O que foi alterado;
- Quem realizou a mudança;
- Em que data e hora;
- Referência à issue ou tarefa resolvida.

Essas informações são usadas em reuniões de acompanhamento e também para documentar a evolução do projeto no repositório.

6 - REPOSITÓRIO DE GESTÃO DE CONFIGURAÇÃO DE SOFTWARE

6.1 Introdução

O repositório de gestão de configuração centraliza todos os Itens de Configuração de Software (SCIs), permitindo identificar, versionar, controlar alterações e auditar cada artefato do projeto, conforme as práticas de SCM.

.

6.2 Estrutura do Repositório

- Plataforma: GitHub, repositório bookscore-a3.
    - Organização de Branches:
    - main – código estável homologado.
    - develop – integração contínua de funcionalidades validadas.
    - feature/<ID> – branches isoladas para cada tarefa/backlog item.
    - release/<versão> – preparar versão candidata para produção.
- Tags de Versão:
    - v1.0, v1.1, etc., correspondendo a releases aprovadas.

6.3 Controle de Versão

Cada SCI (código-fonte, scripts SQL, protótipos Figma, documentação) recebe commits atômicos com mensagens padronizadas ([RF004] CadastroLivro), formando changesets para cada requisito

Além disso, o Git armazena todas as versões dos arquivos e permite reconstruir qualquer release a partir dos diffs entre commits.

6.4 Controle de Alterações

Pull Requests (PRs): todo novo conjunto de alterações é submetido como PR, submetido a revisão de colegas antes do merge, garantindo que apenas código aprovado entre nos branches principais.

Engineering Change Orders (ECOs): para mudanças críticas, documentamos uma ECO em docs/changes/ECO-<n>.md, detalhando o escopo, autor e data da alteração.

6.5 Auditoria de Configuração

Revisões técnicas são realizadas em cada PR, verificando consistência entre SCIs e aderência aos padrões.

Auditoria Formal: semestral, com checklist de perguntas (ECO aplicada, revisão técnica, metadados corretos em cada SCI) para assegurar qualidade e rastreabilidade.

6.6 Relatórios de Status de Configuração

- O repositório mantém um Configuration Status Report (CSR) em docs/reports/CSR-YYYYMMDD.md, registrando:
    1. Novos SCIs (arquivos adicionados).
    2. Alterações aprovadas (PRs fechadas).
    3. Tags e versões criadas.
    4. Auditorias realizadas e seus resultados.

6.7 Ferramentas de Suporte

- Git/GitHub: versionamento distribuído e colaboração.
- GitHub Actions: automação de build, testes e geração de CSR.
- SQLite Browser: visualização de scripts e dados de teste.
- Editor Markdown: para documentação de ECOs e relatórios.

Este repositório, apoiado pelos processos de SCM, garante que cada entrega do BookScore seja rastreável, auditável e sujeita a controles automáticos, assegurando a qualidade e a capacidade de evolução contínua do software.

7 - CONCLUSÃO

A documentação ora apresentada constitui a base estratégica e operacional para transformar o BookScore, originalmente um protótipo acadêmico de avaliação de livros, em uma solução madura, confiável e de fácil manutenção. Com este material, o projeto contará com:

- Visão Clara e Compartilhada: O levantamento de requisitos, os casos de uso detalhados e o backlog estruturado garantem que todos os envolvidos entendam as funcionalidades prioritárias e os critérios de sucesso.
- Qualidade Assegurada: O plano de testes completo, com cronograma, recursos alocados e marcos definidos, permite validar de forma sistemática tanto os requisitos funcionais quanto os não funcionais (desempenho, usabilidade e confiabilidade), reduzindo drasticamente o risco de falhas em produção.
- Governança de Mudanças: O repositório de configuração, apoiado por práticas de controle de versão e pull requests no GitHub, assegura rastreabilidade total de cada alteração no código, protótipos e documentação, facilitando auditorias, auditorias de conformidade e manutenção evolutiva.
- Agilidade e Transparência: A adoção do Scrum, com sprints quinzenais e entregas incrementais, permite entregar valor de forma contínua, incorporar feedbacks de usuários reais e reajustar o escopo em tempo hábil, alinhando-se às dinâmicas de mercado e prioridades de negócio.

Em conjunto, esses artefatos fornecem o guia definitivo para implementação, avaliação e escalonamento do BookScore, elevando o projeto do nível acadêmico ao patamar corporativo.

Com este embasamento, o BookScore está pronto para avançar à fase de implementação, seguro de que cada etapa estará respaldada por processos comprovados e pelas melhores práticas de Engenharia de Software.

8 – REFERÊNCIAS

- https://link-intersystems.com/blog/2013/07/20/the-mvc-pattern-implemented-with-java-swing/
- https://www.scrumstudy.com/article/scrum-methodology-overview
- https://wildart.github.io/MISG5020/standards/ISO-IEC-IEEE-29119-1.pdf
- Engenharia de Software - Uma abordagem profissional / PRESSMAN, Roger; MAXIM, Bruce.
- Engenharia de Software / SOMMERVILLE, Ian
- Engenharia de software: teoria e prática / PFLEEGER, Shari Lawrence
- Conteúdos apresentados na aula de Gestão e Qualidade de Software, ministradas pelo professor Robson Calvetti.