Faculdade de Tecnologia de Americana “Ministro Ralph Biasi”
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Aldair Marino Junior
Layo Levi Tito da Silva
Matheus Hansen Raizer

BookBuds

Americana, SP
2025

Aldair Marino Junior

Layo Levi Tito da Silva

Matheus Hansen Raizer

BookBuds

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na área de concentração em Desenvolvimento de Sistemas.

Orientador: Prof. Me. Jonas Bodê

Este trabalho corresponde à versão final do Trabalho de Conclusão de Curso apresentado por Aldair Marino Junior, Layo Levi Tito da Silva e Matheus Hansen Raizer, e orientado pelo Prof. Me. Jonas Bodê.

Americana, SP

2025

FICHA CATALOGRÁFICA – BIBLIOTECA FATEC AMERICANA MINISTRO RALPH BIASI- CEETEPS DADOS INTERNACIONAIS DE CATALOGAÇÃO-NA-FONTE

MARINO, Aldair Junior

BookBuds. / Aldair Junior Marino, Layo Levi Tito Silva, Matheus Hansen Raizer – Americana, 2025.

110f.

Relatório técnico (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana Ministro Ralph Biasi – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Jonas Bodê

1. Análise de dados 2. Desenvolvimento de software 3. Livro. I. MARINO, Aldair Junior, II. SILVA, Layo Levi Tito, III. RAIZER, Matheus Hansen IV. BODÊ, Jonas V. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana Ministro Ralph Biasi

CDU: 681516
681.3.05
003(02)

Elaborada pelo autor por meio de sistema automático gerador de ficha catalográfica da Fatec de Americana Ministro Ralph Biasi.

Aldair Marino Junior
Layo Levi Tito da Silva
Matheus Ransen Raizer

BookBuds

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana Ministro Ralph Biasi.
Área de concentração: Análise e Desenvolvimento de Sistemas.

Americana, 2 de dezembro de 2025.

Banca Examinadora:

Jonas Bodê
Mestre
Fatec Americana "Ministro Ralph Biasi"

Antonio Alfredo Lacerda
Especialista
Fatec Americana "Ministro Ralph Biasi"

Paula da Fonte Sanches
Mestre
Fatec Americana "Ministro Ralph Biasi"

A Deus, sopro da vida e da sabedoria. Às nossas famílias e cônjuges, alicerces invisíveis que, com paciência e afeto, transformaram nossos dias difíceis em esperança. Aos colegas, pela parceria desde os primórdios dos estudos. Este trabalho é fruto do amor incondicional que, silenciosamente, nos guiou até aqui.

Agradecimentos

Agradecemos primeiramente a Deus, pela vida, oportunidade e capacitação que nos sustentaram durante toda a nossa jornada acadêmica. À Faculdade de Tecnologia de Americana, pela estrutura oferecida para o nosso desenvolvimento, e ao nosso orientador, Prof. Me. Jonas Bodê, pela paciência e sabedoria ao nos guiar na elaboração deste relatório técnico. Estendemos nossa profunda gratidão aos nossos pais, avós, irmãos e companheiros. O amor incondicional, os conselhos, os sacrifícios e o apoio constante de vocês foram o alicerce que tornou este estudo possível.

"Quem mal lê, mal ouve, mal fala, mal vê."
(Monteiro Lobato).

RESUMO

A leitura é fundamental para o desenvolvimento de qualquer indivíduo, fomentando a aquisição de competências cognitivas essenciais. No cenário contemporâneo, a sociedade encontra-se imersa em um fluxo constante de informações que se alinham a uma cultura de imediatismo e gratificação rápida. Consequentemente, hábitos de leitura tornam-se cada vez mais escassos no Brasil, visto que esta prática precisa agora competir pela atenção do usuário contra os estímulos algorítmicos das redes sociais. O objetivo principal deste projeto consiste em mitigar esse declínio através da integração da literatura com a tecnologia, por meio do desenvolvimento do "BookBuds", um Aplicativo Web Progressivo (PWA). Este sistema permite aos usuários organizarem suas estantes virtuais, acompanhar o progresso de leitura e interagir socialmente através de resenhas e comentários. No que tange à metodologia, o projeto adotou uma abordagem hipotético-dedutiva, apoiada tanto por pesquisa bibliográfica quanto por coleta de dados em campo para validar o problema central. O processo de desenvolvimento aderiu ao framework ágil Scrum e utilizou o padrão de Arquitetura Limpa. Os resultados obtidos na pesquisa de campo indicam uma alta aceitação da solução proposta, com uma maioria significativa dos respondentes validando a necessidade de uma ferramenta digital de incentivo para manter a leitura. Conclui-se que o software desenvolvido demonstra viabilidade técnica e relevância social, provando que ferramentas digitais, frequentemente vistas como distrações, podem ser efetivamente reaproveitadas para fomentar hábitos de leitura e ampliar o engajamento cultural.

Palavras-Chave: Livros; Rede Social; Biblioteca.

Abstract

Reading is fundamental for the development of any individual, fostering the acquisition of essential cognitive skills. In the contemporary scenario, society is immersed in a constant flow of information that aligns with a culture of immediacy and rapid gratification. Consequently, reading habits are becoming increasingly scarce in Brazil, as this practice must now compete for user attention against the algorithmic stimuli of social networks. The main objective of this project is to mitigate this decline by integrating literature with technology through the development of "BookBuds," a Progressive Web App (PWA). This system allows users to organize their virtual bookshelves, track reading progress and interact socially through reviews and comments. Regarding methodology, the project adopted a hypothetical-deductive approach, supported by both bibliographic research and field data collection to validate the central problem. The development process adhered to the Scrum agile framework and utilized the Clean Architecture pattern. The results obtained from the field research indicate a high acceptance of the proposed solution, with a significant majority of respondents validating the need for a digital incentive tool to maintain reading. In conclusion, the developed software demonstrates technical viability and social relevance, proving that digital tools, often seen as distractions, can be effectively repurposed to foster reading habits and enhance cultural engagement.

Keywords: *Books; Social Network; Library.*

Lista de Ilustrações

Lista de gráficos

Lista de quadros

Lista de siglas

ACM	Association for Computing Machinery
APA	American Psychological Association
API	Application Programming Interface
BABOK	Business Analysis Body of Knowledge
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DER	Diagrama de Entidade-Relacionamento
DOM	Document Object Model
GIT	Global Information Tracker
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
I/O	Input/Output
IIBA	International Institute of Business Analysis
IPL	Instituto Pró-Livro
JS	JavaScript
JSON	JavaScript Object Notation
LGPD	Lei Geral de Proteção de Dados Pessoais
MER	Modelo Entidade-Relacionamento
MVC	Model-View-Controller
MVP	Minimum Viable Product
N/A	Not Applicable
NoSQL	Not Only SQL
NPM	Node Package Manager
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PWA	Progressive Web App
RAM	Random Access Memory
REST	Representational State Transfer
RF	Requisito Funcional
RNF	Requisito Não Funcional
SDK	Software Development Kit
SQL	Structured Query Language

TCC	Trabalho de Conclusão de Curso
TIC	Tecnologias da Informação e Comunicação
UI	User Interface
UML	Unified Modeling Language
USP	Universidade de São Paulo
UX	User Experience
VS	Code Visual Studio Code
WWW	World Wide Web
XP	Experience Points
XSS	Cross-Site Scripting

SUMÁRIO

1 INTRODUÇÃO

A leitura desempenha um papel crucial no desenvolvimento cognitivo e social do indivíduo, sendo fundamental para a aquisição de competências críticas e culturais. No entanto, na sociedade contemporânea, marcada pelo imediatismo e pela onipresença das redes sociais, observa-se um declínio progressivo no hábito de ler, especialmente entre os jovens. A falta de ferramentas que integrem a experiência de leitura ao universo digital de forma engajadora e organizada cria uma lacuna que desestimula a manutenção desse hábito.

Esse cenário de desestímulo é corroborado por indicadores alarmantes. A 6ª edição da pesquisa "Retratos da Leitura no Brasil" aponta uma redução de aproximadamente sete milhões de leitores no país entre 2019 e 2024. Paralelamente, a capacidade de foco profundo enfrenta desafios na era digital; estudos indicam que o tempo médio de atenção sustentada em telas caiu de 150 segundos em 2004 para apenas 75 segundos em 2012. Essa fragmentação da atenção, impulsionada pela cultura do imediatismo, cria uma barreira significativa para a leitura de obras extensas, exigindo novas estratégias de engajamento.

É neste cenário que surge a necessidade de soluções tecnológicas capazes de reconectar o leitor ao livro através de mecanismos de incentivo e gestão pessoal. A relevância deste projeto reside na utilização da tecnologia não como uma distração, mas como uma aliada no fomento à cultura. Ao propor uma ferramenta que organiza e socializa a leitura, busca-se mitigar os efeitos da dispersão digital, oferecendo uma plataforma que atenda às necessidades do leitor moderno.

Nesse contexto, o objetivo geral do projeto BookBuds é o desenvolvimento de um aplicativo mobile que permita aos usuários registrarem, acompanhar e organizar suas leituras de forma prática e personalizada, visando incentivar o hábito da leitura que se tornou escasso no Brasil e facilitar o registro e controle de obras lidas, em andamento ou planejadas.

Dentre os objetivos específicos, a solução busca oferecer diversas possibilidades para os usuários, tais como:

a) Criação e edição de um perfil próprio;

b) Registro de status de leitura para seus livros (lendo, lido, quero ler, abandonei);
c) Interface de fácil usabilidade e intuitiva;
d) Pesquisa de livros;
e) Possibilidade de interação entre usuários através de comentários e publicações;
f) Registro de histórico de leituras.

A implementação desta solução projeta benefícios que transcendem a funcionalidade técnica. No âmbito educacional e cognitivo, espera-se que o acompanhamento de progresso auxilie na recuperação do foco e na constância da leitura. Socialmente, a ferramenta visa democratizar o debate literário, criando comunidades digitais que estimulam a troca de conhecimentos. Culturalmente, o projeto busca ressignificar o uso do smartphone, transformando-o de um potencial distrator em um facilitador de acesso à cultura, fortalecendo a valorização da literatura.

Para o desenvolvimento, o trabalho adota o método hipotético-dedutivo como abordagem principal e se baseia em pesquisa bibliográfica para embasar o problema da queda do hábito de leitura. Além disso, propõe o desenvolvimento de um sistema como estudo de caso aplicado, podendo futuramente incluir pesquisas de campo para validação do impacto do aplicativo junto ao público-alvo.

Este trabalho está estruturado em quatro capítulos principais. O primeiro capítulo apresenta esta introdução. O segundo capítulo aborda o Referencial Teórico e Tecnológico, fundamentando a importância da leitura e detalhando as tecnologias utilizadas, como Node.js e metodologias ágeis. O terceiro capítulo descreve o Desenvolvimento, apresentando a engenharia de requisitos, a arquitetura do sistema, os diagramas UML e os fluxos de navegação. Por fim, o quarto capítulo traz as Considerações Finais, analisando os resultados alcançados frente aos objetivos propostos.

2 REFERENCIAL TEÓRICO E TECNOLÓGICO

2.1 A IMPORTÂNCIA DA LEITURA

A leitura se apresenta como uma prática fundamental para o desenvolvimento intelectual, social e cultural do indivíduo. “A leitura do mundo precede a leitura da palavra” (Freire, 1989, p. 11). Essa frase demonstra a importância do ato de ler, que, aliado ao seu processo de desenvolvimento, impacta diretamente a capacidade de compreensão e de interação com a realidade. Para o autor, ler ultrapassa a simples decodificação de letras, exercendo um papel político e libertador, sobretudo em contextos de exclusão social. De forma complementar, Dr. Seuss afirma: “Quanto mais você lê, mais coisas saberá. Quanto mais você aprende, a mais lugares irá!”[1] (Geisel, 1978), indicando que leitura e aprendizado estão intimamente ligados como ferramentas de crescimento e ampliação dos horizontes dos indivíduos.

Sob uma perspectiva mais técnica, a leitura contribui para o desenvolvimento mental e cognitivo, sendo especialmente eficaz quando promovida desde os primeiros anos de vida. “O simples ato de ler um livro para uma criança pequena tem demonstrado, repetidamente, ter um poder notável”[2] (Rodriguez et al., 2009, p. 677, tradução nossa).

Além disso, a neurociência comprova que a leitura transforma a estrutura cerebral. O processo de alfabetização recicla áreas originalmente destinadas ao reconhecimento visual de objetos e rostos. Segundo Dehaene (2012, p. 22)

> A leitura transforma o cérebro. Ao aprendermos a ler, reciclamos áreas cerebrais originalmente destinadas ao reconhecimento visual de objetos e rostos. Essa reciclagem neuronal dá origem ao que chamo de ‘circuito da leitura’, que passa a integrar automaticamente sons, grafemas e significados.

Tais constatações enfatizam o impacto da leitura no desenvolvimento de habilidades cognitivas e na formação integral do indivíduo.

---

1 Original: “The more that you read, the more things you will know. The more that you learn, the more places you’ll go!”

2 Original: “The humble act of reading a book to a young child has repeatedly been found to have remarkable power”.

2.2 LEITURA DIGITAL

Com o avanço das Tecnologias da Informação e Comunicação (TICs), emergiram novas modalidades de consumo literário, expandindo a prática da leitura para além dos suportes físicos tradicionais. No ecossistema digital, a leitura incorpora novas linguagens, formatos e dinâmicas. A transição progressiva dos meios físicos para os recursos digitais apresenta tanto benefícios quanto desafios, configurando um cenário de irreversibilidade tecnológica que demanda, por parte da sociedade e das instituições, uma adaptação estratégica às novas tendências de consumo.

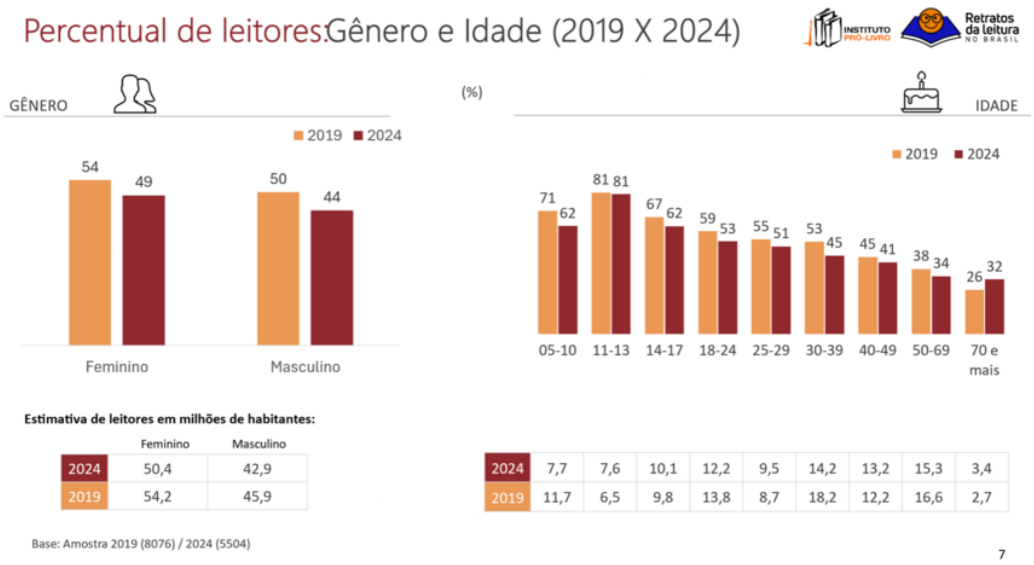
Nesse contexto, a facilidade de distribuição digital traz à tona desafios éticos e econômicos. A pirataria permanece prevalente no consumo de livros digitais, notadamente no meio universitário. Segundo Depexe e Neres (2023), esse fenômeno evidencia a urgência de políticas que promovam e facilitem o acesso legal a obras digitais, tornando as alternativas oficiais mais atrativas e acessíveis ao público acadêmico.

Em termos quantitativos, dados indicam que o contingente total de leitores em países como Estados Unidos, França e Brasil manteve-se estável na última década. Contudo, observa-se uma mudança qualitativa nos hábitos de consumo: houve uma retração significativa na demanda por obras físicas e bibliotecas tradicionais, contrastada pelo crescimento do interesse por livros digitais e audiolivros. Conforme aponta Souza (2022), o público jovem lidera o engajamento com essas novas mídias, sinalizando a necessidade imperativa de reestruturação das instituições culturais para atender às demandas contemporâneas da sociedade da informação.

2.3 LEITURA EM TEMPOS DE IMEDIATISMO

Em consonância com a crescente popularidade das redes sociais e o fortalecimento da cultura do imediatismo, observa-se uma retração no hábito da leitura no Brasil. Dados da 6ª edição da pesquisa "Retratos da Leitura", realizada pelo Instituto Pró-Livro (2024), apontam uma redução de aproximadamente sete milhões de leitores no país entre 2019 e 2024. Conforme demonstrado na Figura 1, os indicadores não sugerem uma reversão dessa tendência, mas sim o seu agravamento em diversos estratos demográficos.

Gráfico 1 – Percentual de leitores: Gênero e Idade (2019 x 2024)



Fonte: Instituto Pró-Livro, 2024

A análise desse cenário é corroborada por Boto (2025), da Universidade de São Paulo (USP), que atribui tal declínio à intensificação das atividades digitais. Segundo a autora, o ambiente online torna-se mais atrativo por oferecer sistemas de recompensas imediatas, característica intrínseca à sociedade contemporânea, marcada pela busca constante por prazer instantâneo e estímulos rápidos.

Essa perspectiva é reforçada por Mark (2023 apud APA, 2024), destacando como a interação contínua com a internet e dispositivos digitais impacta a capacidade cognitiva de foco. Dados apresentados pela autora revelam uma queda drástica na média de atenção sustentada: em 2004, o tempo médio era de 150 segundos (dois minutos e meio); em 2012, esse intervalo reduziu-se para apenas 75 segundos.

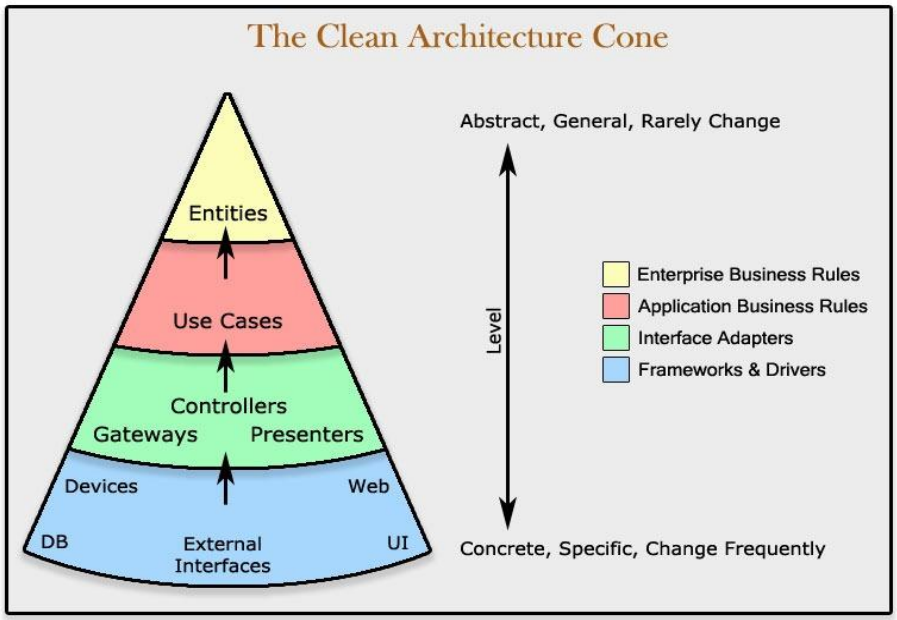
Portanto, evidencia-se que a redução do hábito da leitura no Brasil possui uma correlação direta com a onipresença das tecnologias digitais e a consolidação de um modelo de consumo de informações fragmentado, típico da atual era da informação.

2.4 ARQUITETURA

2.4.1 Arquitetura limpa

Os modelos apresentados abaixo visam padronizar a arquitetura do software, de forma a criar uma estrutura semântica que favoreça a reutilização e a independência dos elementos tecnológicos utilizados. O objetivo é manter um design com consistência lógica, facilitando o entendimento e a comunicação do código, além de promover um sistema mais robusto, testável e de fácil manutenção.

Figura 1 – Arquitetura de software (*Clean Architeture*)



Fonte: Hipólito Júnior, 2018

Organizado em camadas e seguindo o fluxo indicado pelas setas (conhecido como Regra da Dependência), o modelo proposto por Robert C. Martin estabelece uma separação clara das responsabilidades entre as partes do sistema. A seguir, são detalhadas as funções de cada camada:

Frameworks & *Drivers* (*External Interfaces*): "A camada mais externa é geralmente composta por frameworks e ferramentas, como o banco de dados e o *framework web*" (Martin, 2017, p. 192). Essa camada representa tudo o que está fora do domínio da aplicação. Por depender de tecnologias específicas, deve ser tratada como substituível. O sistema deve ser capaz de continuar funcionando mesmo diante da substituição de algum desses componentes, o que favorece a flexibilidade e a adaptação a novos recursos tecnológicos.

Interface Adapter (*Controllers*): "Esta camada contém adaptadores que convertem os dados do formato mais conveniente para os casos de uso e entidades, para o formato mais conveniente para alguma agência externa, como o banco de dados ou a web" (Martin, 2017, p. 192). Ela atua como uma ponte entre a lógica do sistema e o mundo exterior, incluindo interfaces de usuário, *Application Programming Interfaces* (APIs) e bancos de dados. Seu papel é garantir que os dados cheguem de forma limpa e no formato esperado à camada de *Use Cases*, permitindo que a lógica do domínio permaneça isolada de tecnologias externas.

Application Business Rules (*Use Cases*): "Casos de uso orquestram o fluxo de dados entre entidades e adaptadores, e direcionam as entidades para utilizar as regras de negócio amplas, com o objetivo de cumprir os objetivos definidos do caso de uso" (Martin, 2017, p. 191). Os casos de uso descrevem o que o sistema faz, mas não se preocupam com a forma como os dados são exibidos ou armazenados. Essa separação garante que as regras específicas da aplicação estejam centralizadas, facilitando a manutenção e o teste dos comportamentos esperados do sistema.

Enterprise Business Rules (*Entities*): "Entidades encapsulam regras de negócio amplas da empresa. Uma entidade pode ser um objeto com métodos ou um conjunto de estruturas de dados e funções" (Martin, 2017, p. 191). Essas entidades contêm regras de negócio que são independentes da aplicação e estão relacionadas ao domínio central da organização. Por representarem o núcleo mais estável e

reutilizável do sistema, as entidades podem ser reaproveitadas em diferentes contextos ou aplicações, mesmo fora do ambiente atual.

Regra da Dependência (Setas): "A regra fundamental que faz essa arquitetura funcionar é a Regra da Dependência. Ela determina que as dependências no código-fonte devem apontar apenas para o interior" (Martin, 2017, p. 191). Também chamada de fluxo de controle, essa regra assegura que as camadas de nível mais alto, como as *Entities*, não dependam das camadas mais baixas, como frameworks e ferramentas externas. Isso reforça a sustentabilidade da arquitetura, pois permite que decisões de alto nível não sejam impactadas por alterações em detalhes de implementação, promovendo maior estabilidade a longo prazo.

2.4.2 Models View Controler (MVC)

O MVC é um padrão de arquitetura de software frequentemente utilizado no desenvolvimento de sistemas por conta da sua capacidade de organização e estruturação de aplicações que facilita a manutenção e reutilização do código. Como dito por Sommerville (2011, p. 183), “O padrão *Model-View-Controller* separa a aplicação em três componentes principais, de forma que a lógica de apresentação fique independente da lógica de negócio, facilitando a manutenção e a evolução do sistema.”.

O primeiro componente é o Modelo, ele é responsável pela manipulação dos dados e da lógica por trás dos negócios, ele representa as regras pelas quais o sistema funciona. De acordo com Larman (2004), o modelo deve refletir o domínio do problema e manter-se independente de qualquer interface, garantindo baixo acoplamento.

A Visão é o segundo componente e é a camada com a qual o usuário interage diretamente, ela é responsável pela apresentação visual, diferente da primeira camada, a Visão não contém lógica de negócios, apenas exibe os dados fornecidos pelo Modelo.

A terceira e última camada se denomina Controlador e atua como um intermediário entre as outras duas camadas. Ele recebe as requisições do usuário e as encaminha para o Modelo, que realiza o processamento e posteriormente utiliza a Visão para mostrar ao usuário o resultado.

Este padrão tem uma sinergia e relevância especial com sistemas que lidam com mudanças constantes e interfaces dinâmicas (características típicas em aplicações atuais) por conta da sua arquitetura organizada e de fácil manutenção.

2.5 METODOLOGIAS ÁGEIS

O escritor Oscar Wilde, no século XIX, argumentou que "o descontentamento é o primeiro passo na evolução de um homem ou de uma nação" (Wilde, 2007). Sob essa ótica, traça-se um paralelo com a evolução da Engenharia de Software: a necessidade de adaptação constante imposta pelo mercado atual colide com a rigidez das abordagens pretéritas. O imediatismo tornou-se parte do cotidiano, exigindo que as metodologias de desenvolvimento acompanhassem esse ritmo.

Historicamente, o desenvolvimento de *software* apoiava-se em metodologias tradicionais, como o Modelo em Cascata ou o *Project Management Body of Knowledge* (PMBOK). Conforme definem Dennis, Wixom e Tegarden (2014), o modelo em cascata é a estrutura clássica onde o ciclo de vida do desenvolvimento avança sequencialmente por quatro fases rígidas: planejamento, análise, projeto e implementação. Embora fundamentais em sua época, a estrutura linear e preditiva desses modelos mostrou-se pouco aderente à volatilidade dos requisitos modernos.

No cenário contemporâneo, a mudança é a única constante. Requisitos que hoje são prioridade, amanhã podem perder sua prioridade ou serem descartados. Nesse contexto de alta velocidade, consolidou-se o paradigma Ágil. O marco dessa transição ocorreu em 2001, com a reunião de 17 especialistas que buscavam alternativas aos modelos rígidos, resultando na publicação do "Manifesto Ágil" (Beck et al., 2001). O documento estabeleceu quatro valores fundamentais que priorizam:

a) Indivíduos e interações mais que processos e ferramentas;
b) Software em funcionamento mais que documentação abrangente;
c) Colaboração com o cliente mais que negociação de contratos;
d) Responder a mudanças mais que seguir um plano.

Nota-se, portanto, que a agilidade transcende a nomenclatura; trata-se de uma aplicação prática de princípios. As metodologias ágeis operam por meio de desenvolvimento incremental, no qual o produto é entregue em partes progressivas e funcionais. O *feedback* constante do cliente torna-se o motor do projeto, permitindo que as mudanças necessárias sejam feitas em tempo hábil. Isso garante flexibilidade e maior alinhamento entre o que é construído e o que é necessitado.

A superioridade dessa abordagem em cenários incertos é estatisticamente comprovada. Segundo Mishra e Alzoubi (2023), "as taxas de conclusão bem-sucedida para iniciativas ágeis são de 40%, enquanto apenas 15% dos projetos em cascata foram concluídos com sucesso". Tais dados ratificam a razão pela qual as metodologias ágeis se tornaram o padrão dominante no contexto tecnológico atual.

2.5.1 Scrum

Dentre as metodologias ágeis, o Scrum destaca-se como um *framework* iterativo e incremental para o gerenciamento de projetos complexos. Enquanto metodologias tradicionais frequentemente se baseiam em processos preditivos (como os descritos no PMBOK), o Scrum fundamenta-se nos valores e princípios do Manifesto Ágil (Beck et al., 2001). A estrutura prioriza a flexibilidade e a adaptação contínua, focando na colaboração entre pessoas e na entrega de valor. O modelo reduz a burocracia excessiva em favor do dinamismo, operando através de planejamentos curtos, *feedbacks* empíricos e adaptação constante às necessidades do cliente.

Figura 2 – Os quatro valores do Manifesto Ágil

Fonte: ScrumPath+ (2025)

A opção pela utilização do Scrum neste projeto justifica-se pela sua capacidade de organização de prioridades e controle de tarefas através de ciclos curtos. A metodologia demonstrou-se adequada para mitigar riscos e garantir o alinhamento constante entre o desenvolvimento e os requisitos do sistema.

O *framework* foi formalizado por Jeff Sutherland e Ken Schwaber na década de 1990, inspirados no artigo "*The New New Product Development Game*" de Takeuchi e Nonaka (1986), publicado na *Harvard Business Review*. Segundo Schwaber e Sutherland (2020), o Scrum sustenta-se em três pilares fundamentais da teoria de controle de processos empíricos:

- Transparência: Garante que os aspectos significativos do processo estejam visíveis aos responsáveis pelos resultados.
- Inspeção: Preconiza a verificação frequente dos artefatos e do progresso em direção ao objetivo para detectar variações indesejadas.
- Adaptação: Determina que, caso a inspeção revele problemas, o processo ou o material sendo processado deve ser ajustado o mais rápido possível para minimizar desvios.

Conforme detalhado no Guia do Scrum (2020), a estrutura é composta por artefatos, eventos e papéis. Os três artefatos principais são: o *Product Backlog* (lista priorizada de requisitos do produto); o *Sprint Backlog* (conjunto de itens selecionados para execução no ciclo atual); e o Incremento (a soma de todos os itens do Backlog completados durante a Sprint).

O fluxo de trabalho é regido por eventos temporais definidos. O ciclo inicia-se com o Planejamento, onde define-se o objetivo e o escopo da iteração. Durante a execução, que dura geralmente de duas a quatro semanas, ocorrem reuniões diárias de alinhamento. Ao final do ciclo, realiza-se a Revisão para inspeção do incremento produzido, seguida da Retrospectiva, focada na melhoria contínua do processo de trabalho da equipe (Schwaber; Sutherland, 2020).

Quanto aos papéis, Schwaber e Sutherland (2020) definem responsabilidades claras. O Product Owner (Dono do Produto) é responsável por maximizar o valor do produto e gerenciar o Backlog, atuando como ponto central entre a equipe técnica e as partes interessadas. Já o Scrum *Master* atua como um líder servidor e facilitador; diferentemente de um gerente de projetos tradicional, seu foco é remover impedimentos, garantir que o framework seja compreendido e seguido, e assegurar

que a equipe de desenvolvimento mantenha-se produtiva e livre de interrupções externas.

2.6 REQUISITOS

Os requisitos constituem o alicerce contratual e técnico para o desenvolvimento de qualquer projeto de software. Eles representam a materialização das necessidades dos *stakeholders* e dos objetivos de negócio, servindo como a especificação formal do que deve ser construído.

Segundo Sommerville e Sawyer (1997, p. 12):

> Requisitos são definidos durante os estágios iniciais de um desenvolvimento de sistema como uma especificação do que deve ser implementado. Eles são descrições de como o sistema deve se comportar, ou de uma propriedade ou atributo do sistema. Eles podem ser uma restrição ao processo de desenvolvimento do sistema.

De forma complementar, o PMBOK Guide (Project Management Institute, 2017, p. 158) afirma que "Requisitos incluem condições ou capacidades que são requeridas que estejam presentes em um produto, serviço ou resultado para satisfazer um acordo ou outra especificação formalmente imposta".

Para fins de modelagem e documentação, a literatura técnica estabelece uma distinção fundamental na categorização dos requisitos, conforme Vazquez e Simões (2016, p. 58), “A classificação mais utilizada agrupa os requisitos em duas categorias: funcionais e não funcionais.”. Essa taxonomia é essencial para a organização da especificação, orientando de forma precisa as etapas subsequentes de arquitetura, implementação e testes.

2.6.1 Requisitos Funcionais

Conforme Wiegers e Beatty (2013, p. 599) " Requisito funcional: uma descrição de um comportamento que o sistema exibirá sob condições específicas."[3]. Dessa forma, os Requisitos Funcionais são a especificação direta das ações, capacidades de gerenciamento de informação e interações que o sistema executará.

Corroborando essa perspectiva, o BABOK Guide (International Institute of Business Analysis, 2015, p. 16) estabelece que tais requisitos "[...] descrevem as capacidades que uma solução deve possuir em termos do comportamento e das informações que a solução irá gerenciar."[4].

2.6.2 Requisitos não Funcionais

Em contrapartida, os Requisitos Não Funcionais referem-se aos atributos de qualidade e restrições do software. Wiegers e Beatty (2013, p. 600) definem este tipo de requisito como "uma descrição de uma propriedade ou característica que um sistema deve apresentar ou uma restrição que ele deve respeitar"[5]. De forma complementar, o BABOK Guide (International Institute of Business Analysis, 2015, p. 158) estabelece que eles "descrevem as condições ambientais ou qualidades necessárias para que o produto seja eficaz"[6].

Portanto, esta categoria está intrinsecamente ligada à mensuração da qualidade da aplicação. Ao definirem declarações sobre os atributos do sistema, estes requisitos estabelecem os padrões críticos de desempenho, segurança, usabilidade e confiabilidade, delimitando as fronteiras técnicas essenciais para a aceitação final da solução.

---

[3] Original: “Functional requirement: A description of a behavior that a system will exhibit under specific conditions.”

[4] Original: “[...] describe the capabilities a solution must have in terms of the behaviour and information the solution will manage.”

[5] Original: “A description of a property or characteristic that a system must exhibit or a constraint that it must respect.”

[6] Original: “[…] describe the environmental conditions or qualities required for the product to be effective.”

2.6.3 Levantamento de Requisitos

A etapa de Elicitação de Requisitos atua como o ponto de partida para a engenharia do software. Segundo Wiegers e Beatty (2013, p. 599), "O processo de descoberta dos requisitos de um sistema por meio da comunicação com clientes, usuários do sistema e outros que tenham interesse no desenvolvimento do sistema"[7]. A obtenção das diversas categorias de requisitos ocorre através deste processo de natureza investigativa, que busca mapear e compreender profundamente as necessidades que a solução deve atender. Trata-se, portanto, de uma atividade analítica e iterativa, servindo como o alicerce sobre o qual todo o entendimento lógico e técnico do sistema é construído.

2.7 DIAGRAMAS UML

A Linguagem de Modelagem Unificada (UML) é uma linguagem padrão para a visualização, especificação, construção e documentação de artefatos de sistemas de software. Segundo Sommerville (2010), a UML tornou-se a linguagem padrão para a modelagem de sistemas orientados a objetos, permitindo que engenheiros de software comuniquem a estrutura e o comportamento do sistema de forma clara e padronizada antes da implementação.

2.7.1 Diagrama de Caso de Uso

O Diagrama de Casos de Uso é uma técnica de modelagem baseada em cenários que descreve as interações entre os usuários (atores) e o sistema para alcançar um objetivo específico. De acordo com Pressman e Maxim (2016), os casos de uso capturam os requisitos funcionais do ponto de vista do usuário, descrevendo "quem" interage com o sistema e "o que" o sistema deve fazer para atender a essa interação, sem detalhar a lógica interna de como isso será implementado.

---

[7] Original: “The process of discovering the requirements for a system by communication with customers, system users, and others who have an interest in the development of the system.”

2.7.2 Diagrama de Entidade-Relacionamento

O Diagrama de Entidade-Relacionamento (DER), ou Modelo Entidade-Relacionamento (MER), é uma ferramenta utilizada para a modelagem de dados de alto nível. Segundo Dennis, Wixom e Tegarden (2014), o DER ilustra as principais "entidades" (pessoas, objetos ou conceitos) do sistema e como elas se relacionam entre si, servindo como base para o design do banco de dados relacional. Pressman e Maxim (2016) complementam que este diagrama é fundamental para estabelecer a arquitetura da informação, definindo chaves primárias, chaves estrangeiras e a cardinalidade das relações.

2.7.3 Diagrama de Classe

O Diagrama de Classes é o principal bloco de construção da modelagem orientada a objetos. Ele representa a estrutura estática do sistema, detalhando as classes, seus atributos (propriedades), métodos (comportamentos) e os relacionamentos entre elas, como herança, agregação e associação. Sommerville (2010) define que este diagrama funciona como um "gabarito" para a criação de objetos durante a execução do sistema, descrevendo não apenas o que cada objeto sabe (dados), mas o que ele faz (operações).

2.8 PROTOTIPAÇÃO

A prototipagem constitui uma etapa estratégica no ciclo de desenvolvimento de software, atuando como um mecanismo de redução de incertezas. Segundo Sommerville (2007, p. 139), " um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções.". Sob essa ótica, a construção de versões preliminares permite a validação tangível de conceitos e a exploração de diferentes alternativas de design. É nesta fase que os requisitos elicitados são confrontados com a realidade de uso, permitindo ajustes antes do investimento na codificação final.

No contexto da engenharia de software, Pressman e Maxim (2016) ressaltam que a prototipação ocorre em um espectro de fidelidade, onde os protótipos variam desde

modelos em papel de baixa fidelidade até versões operacionais de alta fidelidade, permitindo diferentes níveis de validação junto ao usuário.

2.8.1 Baixa Fidelidade

A prototipagem de baixa fidelidade caracteriza-se como uma estratégia ágil e econômica para a visualização preliminar do software. O objetivo central desta abordagem é permitir alterações estruturais rápidas com custo de refatoração próximo a zero.

Tal definição fundamenta-se em Rudd, Stern e Isensee (1996, p. 77), que estabelecem que "protótipos de baixa fidelidade são geralmente limitados em funcionalidade e interação. Eles são construídos para representar conceitos, alternativas de design e layouts de tela."[8].

Essa perspectiva é corroborada por Rettig (1994, p. 22), que enfatiza a natureza iterativa do processo: "A prototipagem de baixa fidelidade [...] permite iterar rapidamente. Como o custo de construção de um protótipo é muito baixo, é viável descartá-lo e experimentar uma ideia diferente."[9]. Dessa forma, a técnica fomenta a exploração criativa sem o apego técnico inerente a implementações mais complexas.

2.8.2 Alta Fidelidade

Em contrapartida, a prototipagem de alta fidelidade tem como objetivo oferecer uma representação visual e funcional concreta da solução proposta. Embora ainda sujeita a refinamentos, essa abordagem preserva a vasta maioria dos elementos estéticos e fluxos de navegação previstos para a versão final do software.

---

[8] Original: "Low-fidelity prototypes are generally limited in function and interaction. They are constructed to depict concepts, design alternatives, and screen layouts."

[9] Original: "Lo-fi prototyping [...] allows you to iterate quickly. Because the cost of building a prototype is so low, you can afford to throw it away and try a different idea."

Tais características são descritas por Stone et al. (2004, p. 244), ao afirmarem que

> "protótipos de alta fidelidade [...] representam a funcionalidade central da interface de usuário do produto. Eles são totalmente interativos e simulam a jornada do usuário pelo sistema. São precisos em relação a cores, fontes e layout, proporcionando ao usuário uma sensação real do produto final."[10].

Dessa forma, garante-se uma validação precisa da experiência do usuário antes do investimento na codificação complexa do sistema.

2.9 TECNOLOGIAS UTILIZADAS

Para a fundamentação tecnológica deste projeto, as ferramentas selecionadas foram organizadas em cinco categorias distintas, conforme sua aplicação no ciclo de desenvolvimento:

a) Linguagens de Programação e Marcação, que compõem a base da interface;
b) Ambiente de Desenvolvimento e Versionamento, essenciais para a codificação e controle de alterações;
c) *Back-end* e Infraestrutura de Dados, responsáveis pela lógica do servidor e persistência;
d) Design e Prototipagem, focados na experiência do usuário;
e) Serviços de Integração, que permitem a comunicação com fontes de dados externas.

[10] Original: “High-fidelity prototypes [...] represent the core functionality of the product's user interface. They are fully interactive, and simulate the user's journey through the system. They are precise regarding colour, fonts, and layout, and give the user a true feeling of the final product.”

2.9.1 Linguagens de Programação e Marcação

2.9.1.1 HyperText Markup Language HTML

Para a estruturação do conteúdo e a definição da hierarquia semântica das interfaces, utilizou-se o HTML. Considerada o alicerce da World Wide Web (WWW), a linguagem atua como "o bloco de construção mais básico da web", sendo a responsável direta por definir "o significado e a estrutura do conteúdo" apresentado ao navegador (Mozilla, 2025).

Sua aplicação no projeto foi fundamental para organizar os elementos de texto, mídia e formulários de maneira lógica, estabelecendo a base necessária para a posterior estilização e interatividade.

2.9.1.2 Cascading Style Sheets (CSS)

A definição da identidade visual e a formatação da interface foram implementadas através do CSS. Atuando na camada de apresentação, esta tecnologia permite dissociar o conteúdo estrutural de seu design visual. Conforme define a Mozilla (2025), trata-se de uma "linguagem de folhas de estilo utilizada para descrever a apresentação de um documento escrito em uma linguagem de marcação como o HTML".

No âmbito deste projeto, o uso do CSS foi determinante para assegurar a consistência estética, o controle tipográfico e a responsividade do layout em diferentes dispositivos.

2.9.1.3 JavaScript

Para a implementação da lógica de funcionamento e interatividade, empregou-se a linguagem JavaScript. Atuando como a camada de comportamento da aplicação, ela é a responsável por transformar interfaces estáticas em ambientes dinâmicos. Conforme a definição da Mozilla (2025), trata-se da linguagem que "permite a você implementar itens complexos em páginas web", operando como o motor fundamental "toda vez que uma página da web faz mais do que simplesmente mostrar a você informação estática". No contexto deste projeto, o JavaScript foi utilizado para manipular o DOM, validar formulários e gerenciar eventos de usuário em tempo real.

2.9.2 Ambiente de Desenvolvimento e Versionamento

2.9.2.1 Visual Studio Code

Para o desenvolvimento do código-fonte, optou-se pelo Visual Studio Code (VS Code). Desenvolvida pela Microsoft, a ferramenta é descrita oficialmente como “um editor leve, porém poderoso, que roda no desktop e está disponível para Windows, macOS e Linux"[11] (Microsoft, 2025, tradução nossa).

Sua escolha justifica-se pela arquitetura robusta que integra nativamente recursos essenciais como depuração, realce de sintaxe inteligente e controle de versão Git, otimizando a produtividade sem a sobrecarga de ambientes mais complexos.

2.9.2.2 GitHub

Para a hospedagem remota dos repositórios e gestão do ciclo de vida do desenvolvimento, empregou-se o GitHub. A plataforma atua como um serviço baseado em nuvem que centraliza as operações do Git, sendo definida oficialmente como um ambiente de "de hospedagem de código para controle de versão e colaboração. Ele permite que você e outras pessoas trabalhem juntos em projetos de qualquer lugar.”[12] (GitHub, 2025, tradução nossa).

2.9.2.3 Git

O gerenciamento da evolução do código-fonte e a integridade do histórico de desenvolvimento foram assegurados pelo Git. Trata-se do padrão industrial para controle de versão distribuído, definido oficialmente como um sistema de código aberto projetado para "lidar com tudo, desde projetos pequenos a muito grandes, com velocidade e eficiência"[13] (GIT, 2025, tradução nossa). Sua implementação foi crucial

---

[11] Original: "Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux."

[12] Original: "GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere."

[13] Original: "Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."

para permitir o rastreamento granular de alterações, a reversão segura para estados anteriores e o isolamento de novas funcionalidades através de ramificações (branches), garantindo que o desenvolvimento paralelo não comprometesse a estabilidade da base de código principal.

2.9.2.3 Node Package Manager (NPM)

O gerenciamento das dependências e a integração de bibliotecas externas foram realizados através do npm (Node Package Manager). Atuando como o gerenciador de pacotes padrão para o ambiente Node.js, ele conecta o projeto a um vasto ecossistema de código aberto. A documentação oficial o descreve como o maior registro de software do mundo. Desenvolvedores de código aberto de todos os continentes utilizam o npm para compartilhar e obter pacotes."[14] (NPM, 2025, tradução nossa). No contexto deste trabalho, o npm foi essencial para assegurar a instalação versionada e a manutenção escalável dos módulos necessários para a aplicação.

[14] Original: "npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages."

2.9.3 Back-end e Infraestrutura de Dados

2.9.3.1 Node.js

A infraestrutura de back-end foi desenvolvida sobre o Node.js, uma tecnologia que estende a capacidade do JavaScript para além dos navegadores web. Este ambiente de execução permite a construção de servidores de alto desempenho e escalabilidade, sendo definido oficialmente como um "ambiente de execução JavaScript de código aberto e multiplataforma"[15] (OpenJS Foundation, 2025, tradução nossa). Sua adoção no projeto justifica-se pela capacidade de unificar a linguagem de desenvolvimento entre o cliente e o servidor, além de sua eficiência no processamento de requisições simultâneas.

2.9.3.2 Cloud FireStore

Para a camada de persistência e gerenciamento de dados, adotou-se o Cloud Firestore. Integrante do ecossistema Firebase, esta solução apresenta-se como um banco de dados Not Only SQL (NoSQL) orientado a documentos, projetado para oferecer alto desempenho em aplicações modernas. Conforme a documentação do Google (2025, tradução nossa), trata-se de “um banco de dados flexível e escalável para desenvolvimento web, mobile e de servidor"[16]. Sua escolha para este projeto fundamenta-se na capacidade nativa de sincronização em tempo real, permitindo que os dados sejam atualizados instantaneamente entre os clientes conectados sem a necessidade de gerenciamento complexo de sockets ou infraestrutura de servidores dedicados.

[15] Original: "Node.js is an open-source, cross-platform JavaScript runtime environment."
[16] Original: "Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud."

2.9.4 Design e Prototipagem (UI/UX)

2.9.4.1 Quant Ux

Para a etapa de design de interface e validação, a ferramenta selecionada foi o Quant-UX, devido à sua capacidade especializada em integrar a criação visual com testes de usabilidade baseados em dados. Diferenciando-se de editores convencionais, a plataforma foca na obtenção de métricas quantitativas, sendo descrita como "uma ferramenta de design e prototipagem que permite testar rapidamente ideias de projeto. A ferramenta fornece mapas de calor, registros de interação e gravações de tela."[17] (Quant-UX, 2025, tradução nossa). Essa arquitetura permitiu que a prototipagem não fosse apenas um exercício estético, mas uma base sólida para a análise empírica da interação do usuário.

2.9.5 Serviços de Integração (APIs)

2.9.5.1 Google Books

A integração de dados bibliográficos externos foi implementada através da Google Books API. Esta interface permite a realização de consultas programáticas diretamente ao índice global do Google Livros, facilitando o enriquecimento do catálogo da aplicação sem a necessidade de cadastro manual exaustivo. Conforme descrito na documentação oficial (Google Developers, 2025, tradução nossa), a ferramenta possibilita " realizar pesquisas em texto completo e obter informações sobre a obra, condições de visualização e disponibilidade do e-book"[18]. No contexto do sistema, sua utilização foi essencial para automatizar a captura de metadados, garantindo a precisão e a padronização das informações literárias exibidas aos usuários

[17] Original: "Quant-UX is a prototyping and design tool that allows you to quickly test your design ideas. It provides you with heat maps, interaction logs, and screen recordings."

[18] Original: "The Google Books API is easy to use to retrieve information about the books in Google Books. You can perform full-text searches and retrieve book information, viewability and eBook availability."

3 DESENVOLVIMENTO

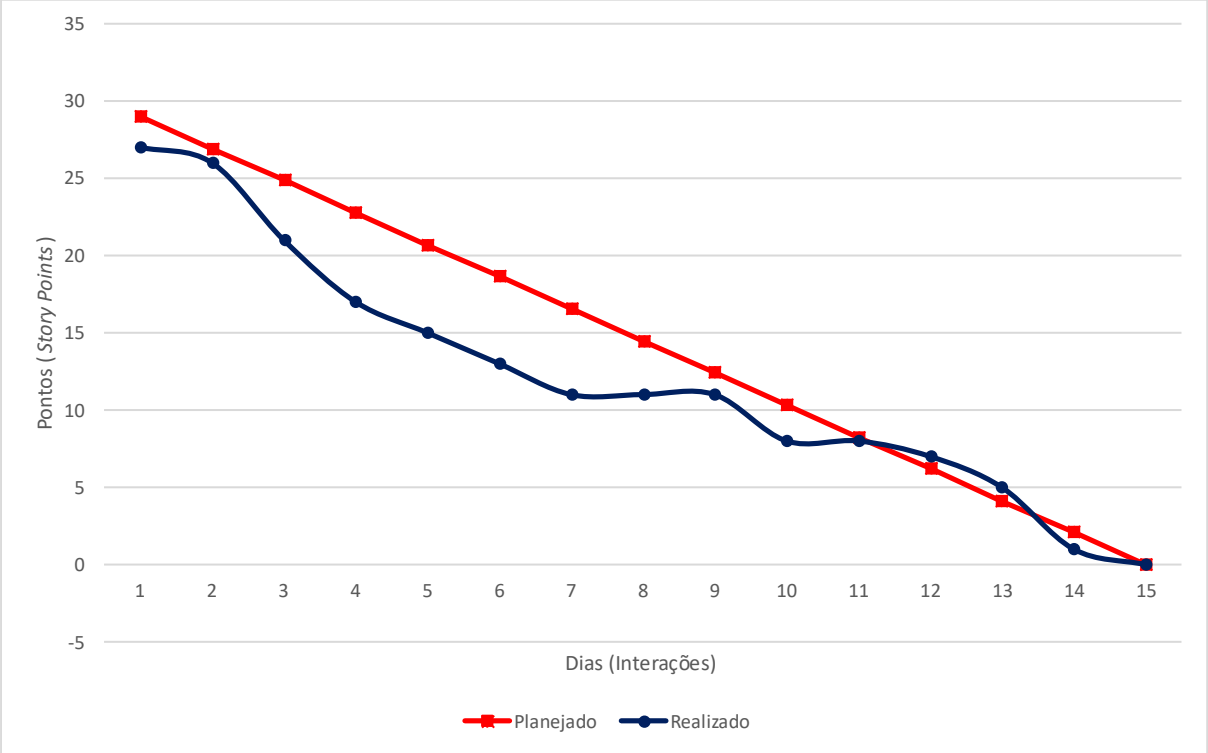
3.1 PROCESSO DE DESENVOLVIMENTO

Adotou-se a metodologia ágil Scrum no desenvolvimento do projeto, o método permitiu a divisão do trabalho em ciclos menores, o que consequentemente resultou em entregas mais contínuas e maior identificação de problemas. Dividimos o projeto em 4 sprints de curta duração e no decorrer delas definimos as prioridades e organizamos o backlog para que assim o essencial fosse priorizado, mantendo uma relação de pontos x dias, sendo os pontos valores de importância definido para objetivos definidos no início da sprint, ao longo de 15 dias, com uma pausa de 15 dias. A abordagem escolhida possibilitou maior controle sobre as evoluções do projeto e facilitou para que adaptações fossem realizadas conforme necessário, o Scrum facilitou a organização do fluxo de trabalho e o cumprimento de prazos. É importante destacar que, utilizou-se desta metodologia como controle geral do desenvolvimento, porém, o software também foi aprimorado e alterado em momentos separados, dos quais, não se encaixariam totalmente nas sprints em si, logo, está subseção diz somente sobre o escopo geral, e não minucioso. Todas os dados se encontram no Apêndice C.

3.1.1 Primeira sprint

A primeira Sprint ocorreu em agosto de 2025, onde focamos na fundação e estruturação fundamental do projeto. Destacou-se majoritariamente na criação das telas no protótipo de alta fidelidade, realizamos o levantamento de requisitos, quais ferramentas e recursos seriam utilizados e começamos os diagramas, especificamente, o de caso de uso. Em uma análise de desempenho, o gráfico de *Burndown* abixo demonstra que foi planejado 29 pontos a serem cumpridos e que a equipe manteve um constante ritmo de entrega.

Gráfico 2 – Burndown (primeira sprint)

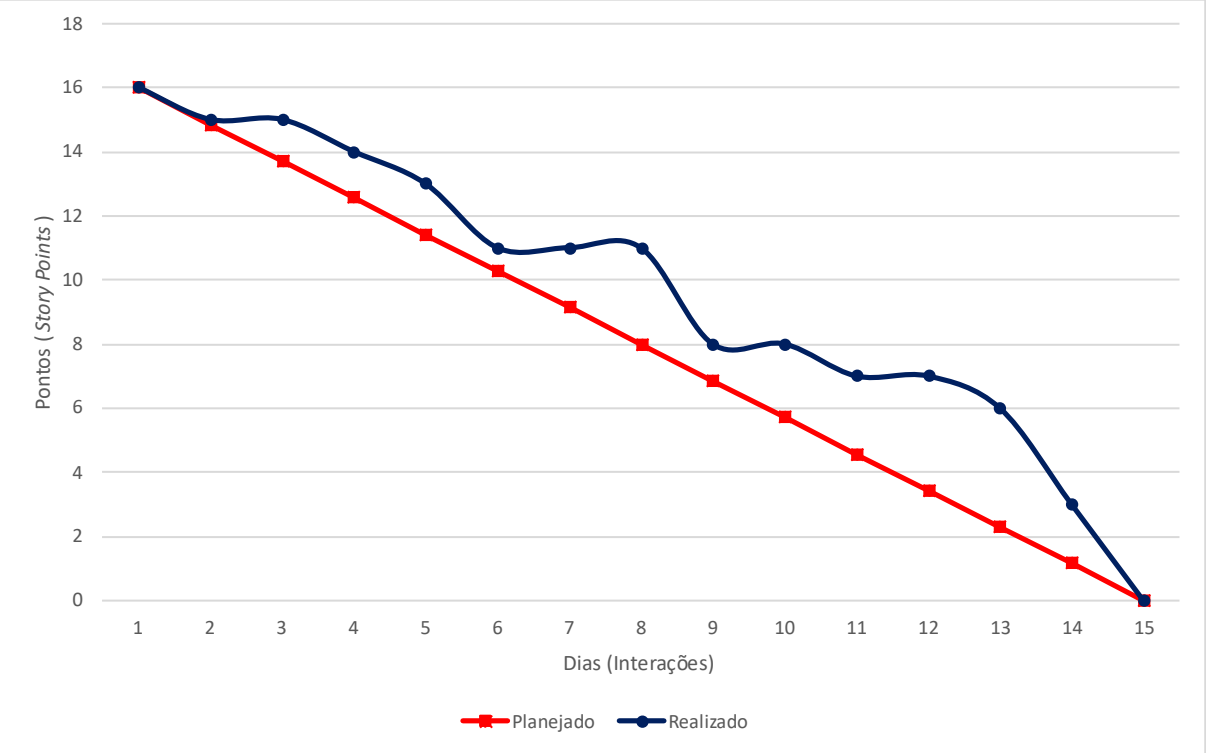


Fonte: Elaborado pelos autores (2025)

3.1.2 Segunda sprint

Em setembro de 2025 foi realizado a segunda sprint do projeto, onde planejamos 16 pontos a serem trabalhados. Analisando o desempenho percebemos que a curva de realização ficou próxima a linha ideal, assim, concluímos que a equipe teve uma precisão maior nas atividades desse período. Essa porção do projeto foi dedicada principalmente ao desenvolvimento inicial *front-end*, realizamos o desenvolvimento base de algumas telas principais e efetuamos ajustes visuais em partes que utilizavam Javascript.

Gráfico 3 – Burndown (segunda sprint)

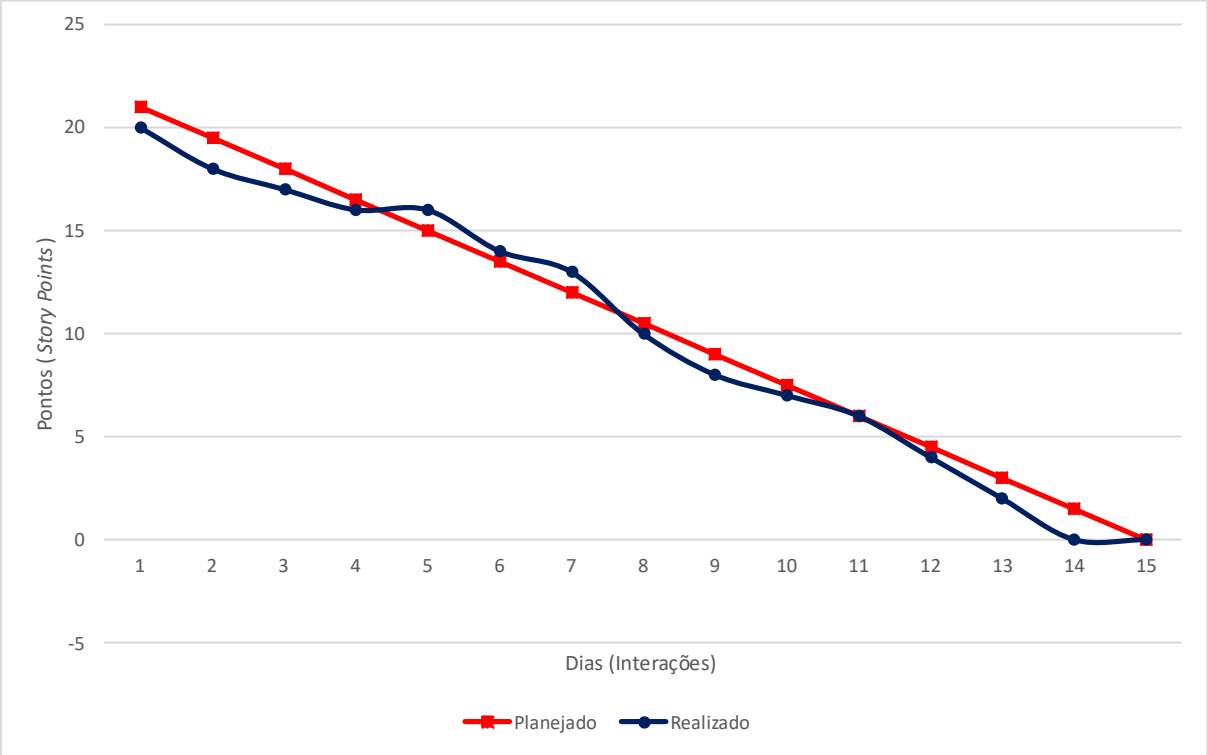


Fonte: Elaborado pelos autores (2025)

3.1.3 Terceira sprint

Com o planejamento inicial de 21 pontos e tendo seu período entre setembro e outubro, a terceira e penúltima Sprint teve foco na escolha de qual API seria utilizada e consequentemente na implementação da escolhida, a Google Books API. A análise de desempenho demonstra através do *Burndown* uma queda acentuada nos primeiros dias em relação aos pontos, o que gerou um leve atraso no meio da Sprint. Dedicamos nossa atenção para a aplicação das heurísticas de Nielsen em nosso projeto, continuamos a produção de diagramas, dessa vez, o de classe, inicializamos animações produzidas em Javascript e CSS e seguimos com a produção do *back-end* e front-end.

Gráfico 4 – Burndown (terceira sprint)

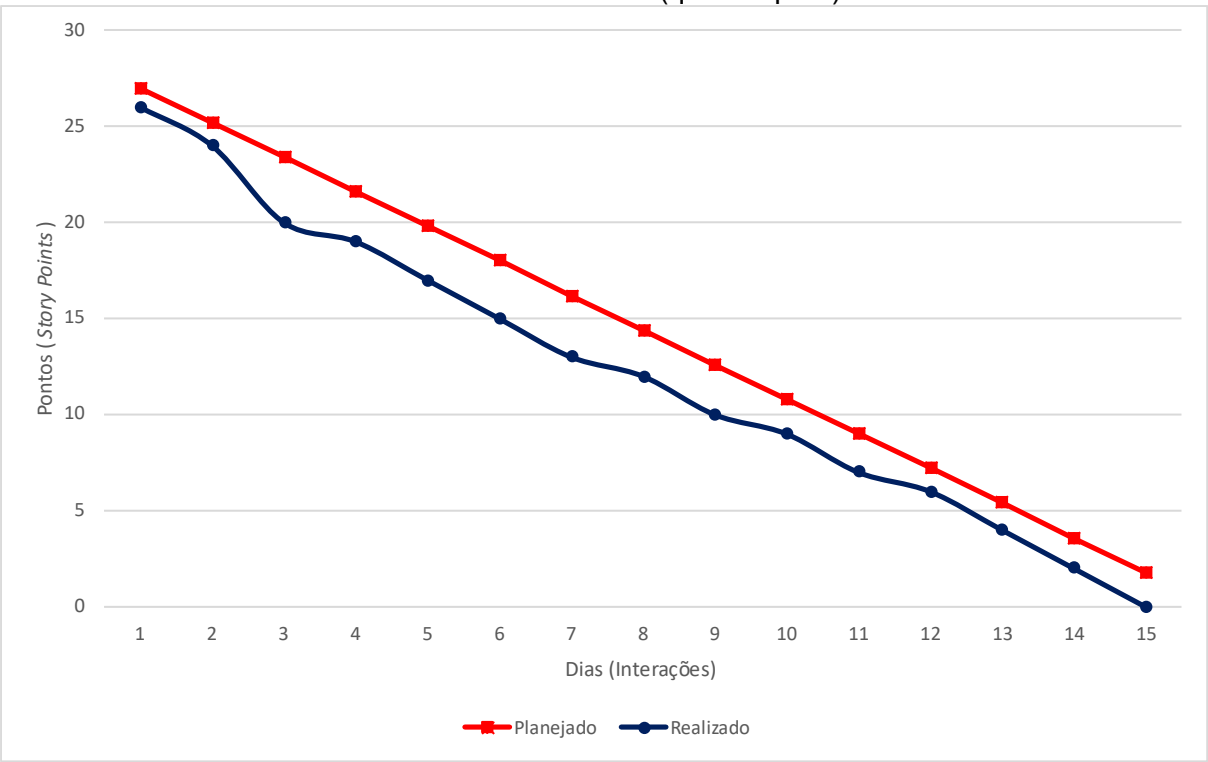


Fonte: Elaborado pelos autores (2025)

3.1.4 Quarta sprint

Durante outubro e novembro de 2025 foi realizada a 4 e última sprint, realizamos os testes de software e focamos no refinamento e finalização das telas e da programação (*back-end e front-end*). Na análise de desempenho, percebe-se que a equipe manteve um ritmo constante de entrega, especialmente na reta final onde foram eliminados os últimos 10 pontos rapidamente. As animações em Javascript foram finalizadas e assim totalizou 27 pontos, conforme visto no gráfico 4.

Gráfico 5 – Burndown (quarta sprint)



Fonte: Elaborado pelos autores (2025)

3.2 ANÁLISE DE REQUISITOS

3.2.1 Levantamento de requisitos

Para realizar a elicitação de requisitos, foi primeiramente projetado uma entrevista semi-estruturada de forma qualitativa, como descrito no Apêndice A, e uma pesquisa social quantitativa, como descrita no Apêndice B, de forma a recolher dados regionais e primeiras impressões para iniciar-se o desenvolvimento do software

Posteriormente, são realizadas reuniões ao longo da evolução do software afim de identificar novos requisitos e/ou remover requisitos não aplicáveis.

3.2.2 Requisitos funcionais

Quadro 1 – Requisitos Funcionais do Sistema

Identificação	**Requisito Funcional**	**Prioridade**
RF001	Acessar API	Essencial
RF002	Acessar banco de dados	Essencial
RF003	Avaliar livro	Essencial
RF004	Comentar livro	Essencial
RF005	Avaliar livro	Essencial
RF006	Cadastrar usuário	Essencial
RF007	Login de usuário	Essencial
RF008	Pesquisar livros (Catálogo)	Essencial
RF009	Filtrar pesquisa	Importante
RF010	Registrar livro lido	Importante
RF011	Avaliar de resenha	Importante
RF012	Borrar Spoliers	Importante
RF013	Redefinir senha	Importante
RF014	Comentar comentários	Desejável
RF015	Editar perfil	Desejável
RF016	Chat entre usuários	Desejável
RF017	Filtragem de resenha	Desejável
RF018	Registro de status do livro	Desejável

Fonte: Elaborado pelos autores (2025)

3.2.3 Requisitos não funcionais

Quadro 2 – Requisitos não Funcionais do Sistema

Identificação	Categoria	Requisito Não Funcional	Prioridade
RNF001	Desempenho	Carregar telas em < 2s	Essencial
RNF002	Segurança	Atender à LGPD	Essencial
RNF003	Infraestrutura	Sincronizar dados na nuvem	Essencial
RNF004	Distribuição	Compatibilidade Android	Essencial
RNF005	Hardware/Soft.	Compatibilidade de Hardware	Essencial
RNF006	Confiabilidade	Atualizar catálogo em tempo real	Importante
RNF007	Segurança	Criptografar dados sensíveis	Importante
RNF008	Portabilidade	Adaptar a telas (Responsividade)	Importante
RNF009	Confiabilidade	Recuperar conexão automaticamente	Importante
RNF010	Confiabilidade	Garantir disponibilidade (99%)	Importante
RNF011	Usabilidade	Seguir Heurísticas de Nielsen	Importante
RNF012	Desempenho	Otimizar resposta tátil	Desejável
RNF013	Desempenho	Economizar bateria e RAM	Desejável
RNF014	Desempenho	Escalar para muitos usuários	Desejável
RNF015	Usabilidade	Oferecer acessibilidade visual	Desejável
RNF016	Padrões	Recomendar via histórico	Desejável

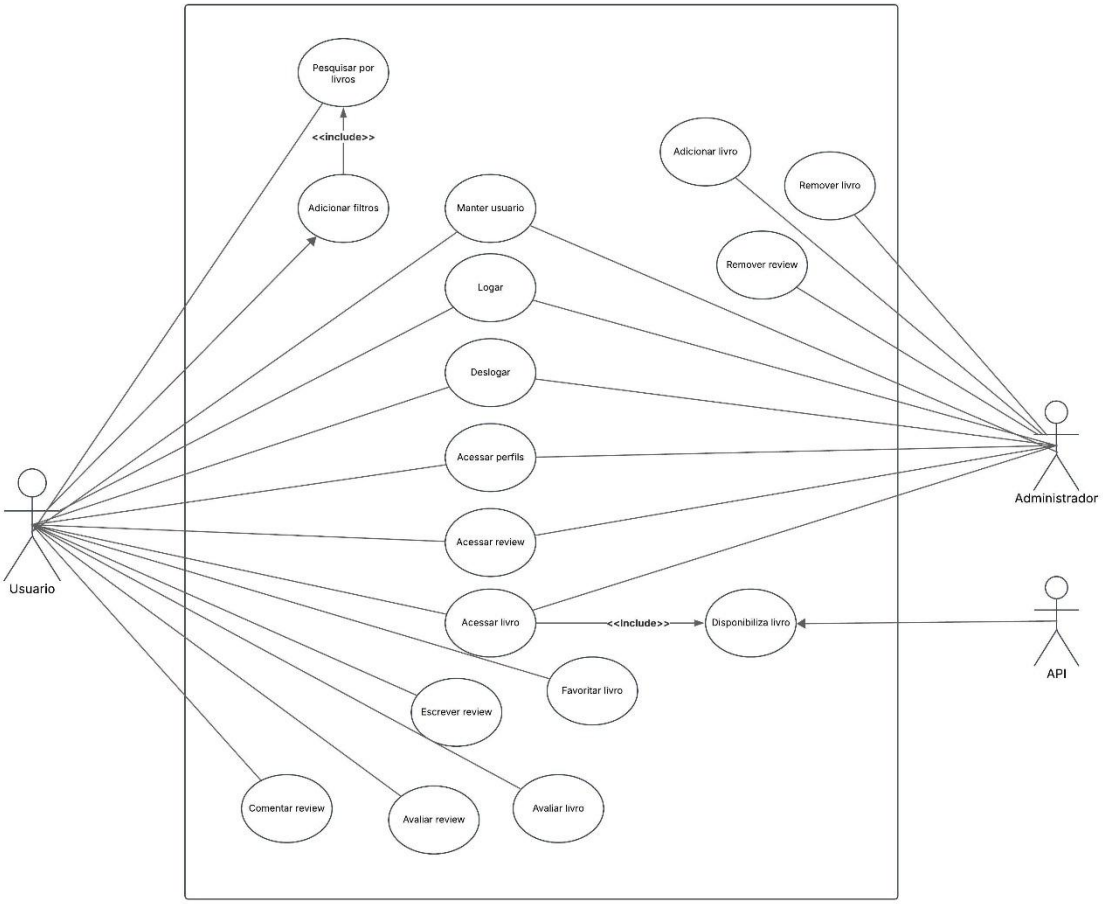
Fonte: Elaborado pelos autores (2025)

3.2.4 Diagrama de casos de uso

Após a definição textual dos requisitos funcionais e não funcionais apresentados nos Quadros 1 e 2, torna-se necessário visualizar como os atores interagem com essas funcionalidades. O diagrama de casos de uso ilustra o escopo do sistema, detalhando as permissões de acesso e as dependências entre os processos.

A Figura 4 apresenta a modelagem dessas interações, destacando os papéis do usuário comum e as integrações externas necessárias para o funcionamento da aplicação.

Figura 3 – Diagrama de caso de uso



Fonte: Elaborado pelos autores (2025)

Conforme observado, o ator principal é o Usuário, que interage com as funcionalidades centrais do aplicativo, como "Pesquisar livros", "Avaliar livro" e "Registrar livro lido", atendendo diretamente aos requisitos essenciais RF008, RF003 e RF010. Nota-se também a representação da comunicação com a Google Books API, que é fundamental para o caso de uso de pesquisa e obtenção de metadados das obras.

O diagrama também evidencia que certas funcionalidades, como "Comentar livro" e "Editar perfil", exigem que o usuário esteja previamente autenticado (Logado), garantindo a segurança e a personalização dos dados conforme estipulado nas regras de negócio do projeto.

3.3 DECISÕES TÉCNICAS JUSTIFICADAS

A definição da *stack* tecnológica e arquitetônica, baseou-se na análise de critérios como escalabilidade, manutenibilidade, curva de aprendizado da equipe, custo de infraestrutura e experiencias previas.

3.3.1 Estratégia de Distribuição: Progressive Web App (PWA)

A opção pelo desenvolvimento de um PWA, em detrimento de aplicações nativas, fundamenta-se na necessidade de portabilidade e resiliência. Tecnicamente, essa abordagem viabiliza a execução da aplicação em múltiplos sistemas operacionais (Android, iOS, Windows) a partir de uma única base de código, otimizando os recursos de desenvolvimento. Além disso, a implementação de *Service Workers* permite interceptar requisições de rede e gerenciar o cache local. Essa capacidade é crítica para atender aos Requisitos Não Funcionais de disponibilidade, garantindo que o sistema ofereça funcionalidades essenciais de consulta e navegação mesmo em cenários de conectividade intermitente ou inexistente.

3.3.2 Persistência de Dados: Abordagem NoSQL com Cloud Firestore

Para a camada de persistência, selecionou-se o Cloud Firestore, um banco de dados NoSQL orientado a documentos. Esta decisão diverge do modelo relacional tradicional devido à natureza variável dos metadados literários e à exigência de sincronização em tempo real. A arquitetura do Firestore permite a atualização instantânea da interface do cliente através de listeners ativos, eliminando a latência de requisição típica de APIs RESTful convencionais. Tal característica atende diretamente à necessidade de dinamicidade no feed social e na atualização de status de leitura entre dispositivos, sem a sobrecarga de infraestrutura de servidores dedicados.

3.3.3 Ambiente de Execução Unificado: Node.js

A utilização do Node.js na infraestrutura de *back-end* justifica-se pela sua arquitetura orientada a eventos e pelo modelo de I/O não bloqueante. Dado que a aplicação realiza intensivas operações de entrada e saída (como consultas à API externa e

leitura de banco de dados), o Node.js oferece desempenho superior no tratamento de múltiplas conexões simultâneas. Adicionalmente, a unificação da linguagem JavaScript tanto no cliente quanto no servidor reduz a complexidade cognitiva do desenvolvimento e facilita o compartilhamento de lógica e validações entre as camadas.

3.3.4 Integração de Dados Externos: Google Books API

Para assegurar a integridade e a padronização dos dados bibliográficos, implementou-se a integração com a Google Books API. A decisão de consumir uma fonte externa de dados justifica-se pela inviabilidade técnica e operacional de manutenção manual de um catálogo literário abrangente. O uso da API permite o enriquecimento automático da aplicação com metadados precisos (título, autoria, sinopse e capas), garantindo que o Requisito Funcional de pesquisa e catálogo seja atendido com dados de alta confiabilidade desde o primeiro acesso do usuário.

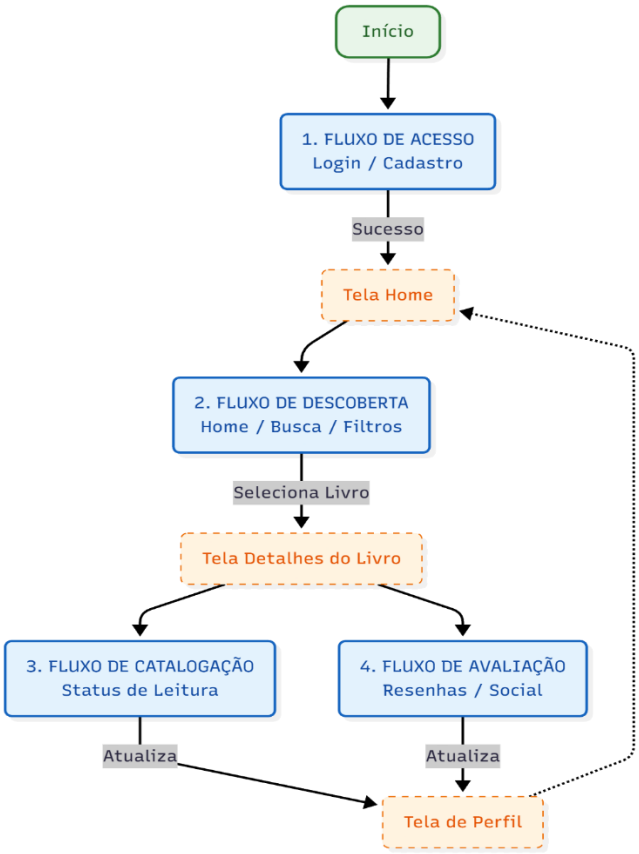
3.4 FLUXO DE USUÁRIOS

Para garantir uma compreensão holística da interação entre o usuário e o sistema, os processos de navegação foram mapeados através de diagramas de fluxo, usando desta representação visual para validar a lógica de negócios e assegurar que a jornada do usuário seja contínua e livre de impasses técnicos.

3.4.1 Fluxo de alto nível

O diagrama de nível macro ilustra a interdependência entre os módulos principais, demonstrando como o usuário transita desde a autenticação até as funcionalidades centrais de descoberta, catalogação e avaliação, culminando na atualização do perfil pessoal.

Figura 4 – Fluxo macro de usuário



Fonte: Elaborado pelos autores (2025)

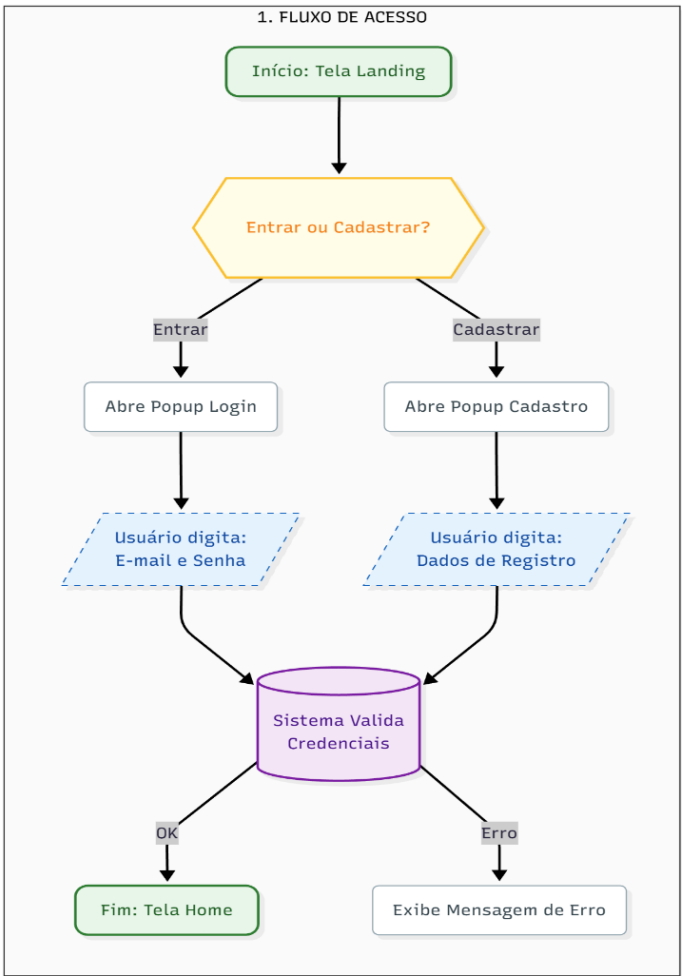
Nota-se como a arquitetura de informação privilegia uma estrutura hierárquica onde a "Tela Home" atua como o hub central de distribuição para os demais fluxos.

3.4.2 Fluxos específicos

Para uma análise técnica aprofundada, os macro-processos foram decompostos em fluxos detalhados, evidenciando as decisões do sistema e as interações com o banco de dados e APIs externas.

a) Fluxo de acesso: o primeiro contato do usuário, detalha as rotinas de validação de credenciais e registro de novos usuários:

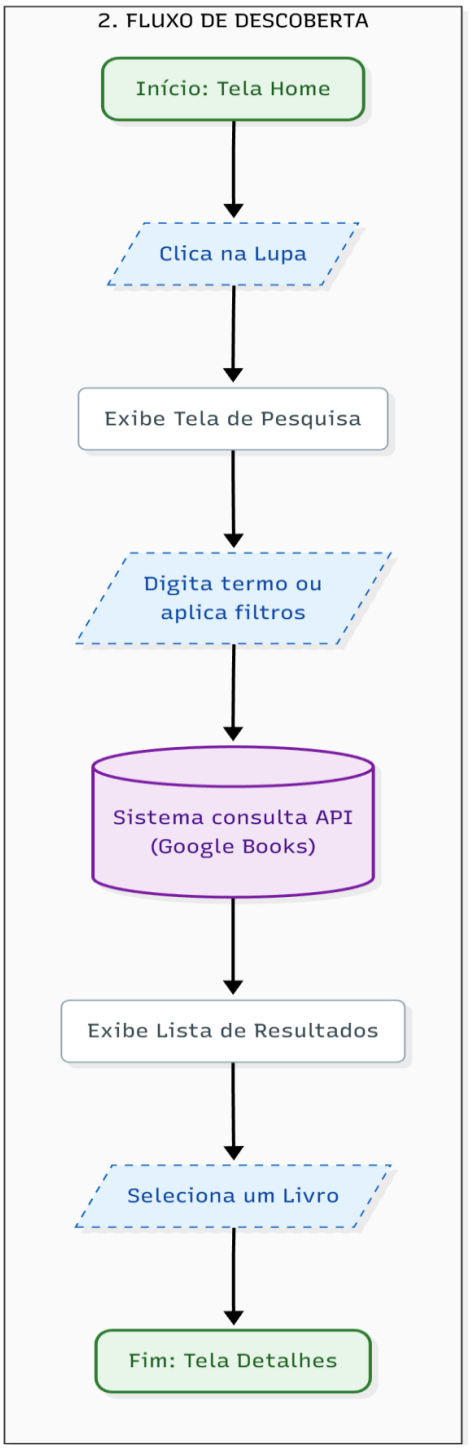
Figura 5 – Fluxo de acesso



Fonte: Elaborado pelos autores (2025)

b) Fluxo de descoberta: a sequência de ações desde a solicitação do usuário até a renderização da lista de resultados, evidenciando o consumo de dados externos:

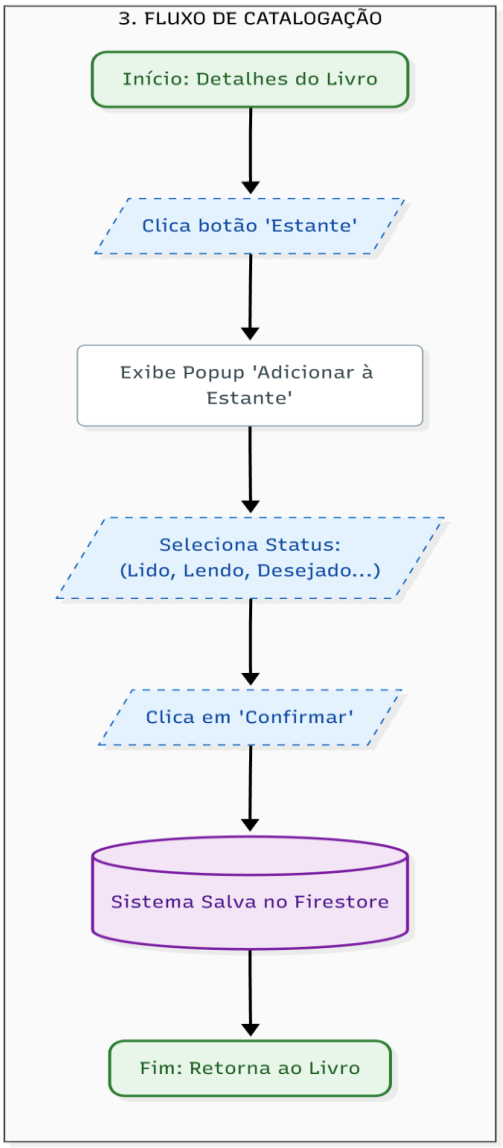
Figura 6 – Fluxo de descoberta



Fonte: Elaborado pelos autores (2025)

c) Fluxo de catalogação: a interação com o banco de dados Cloud Firestore para a persistência do status de leitura selecionado pelo usuário:

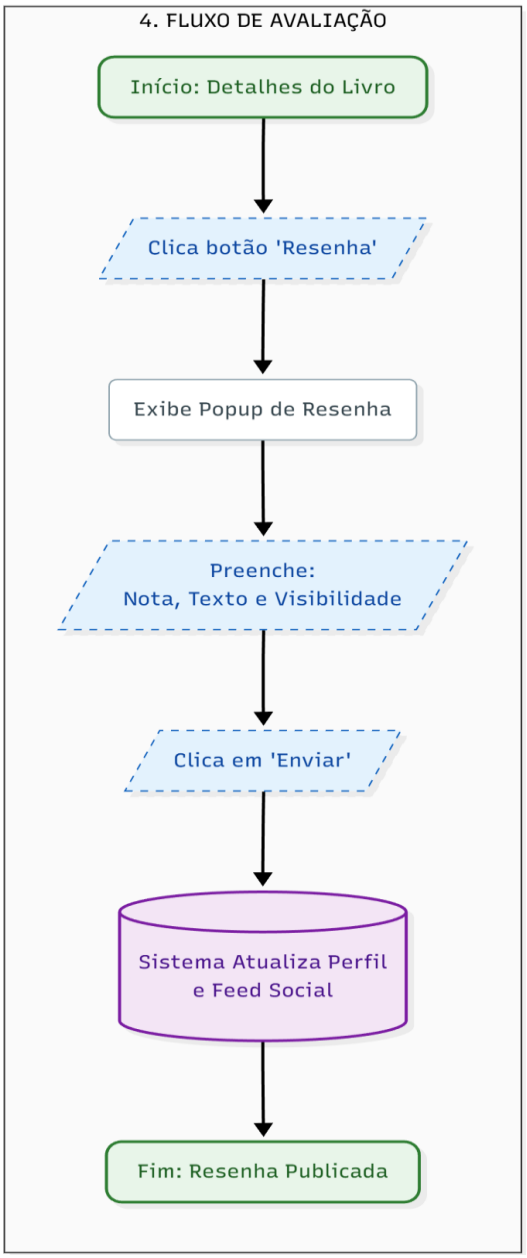
Figura 7 – Fluxo de catalogação



Fonte: Elaborado pelos autores (2025)

d) Fluxo de avaliação: o processo de criação de resenhas, validação de entrada de dados e a consequente atualização do *feed* social e das estatísticas de perfil:

Figura 8 – Fluxo de avaliação



Fonte: Elaborado pelos autores (2025)

3.5 IMPLEMENTAÇÃO DA PERSISTÊNCIA DE DADOS

Para a camada de persistência de dados do projeto BookBuds, optou-se pela utilização do Cloud Firestore, um banco de dados NoSQL orientado a documentos, oferecido pela plataforma Google Firebase. Diferentemente dos bancos de dados relacionais tradicionais (SQL) que organizam dados em tabelas e linhas rígidas, o Firestore armazena dados em documentos, que são organizados em coleções.

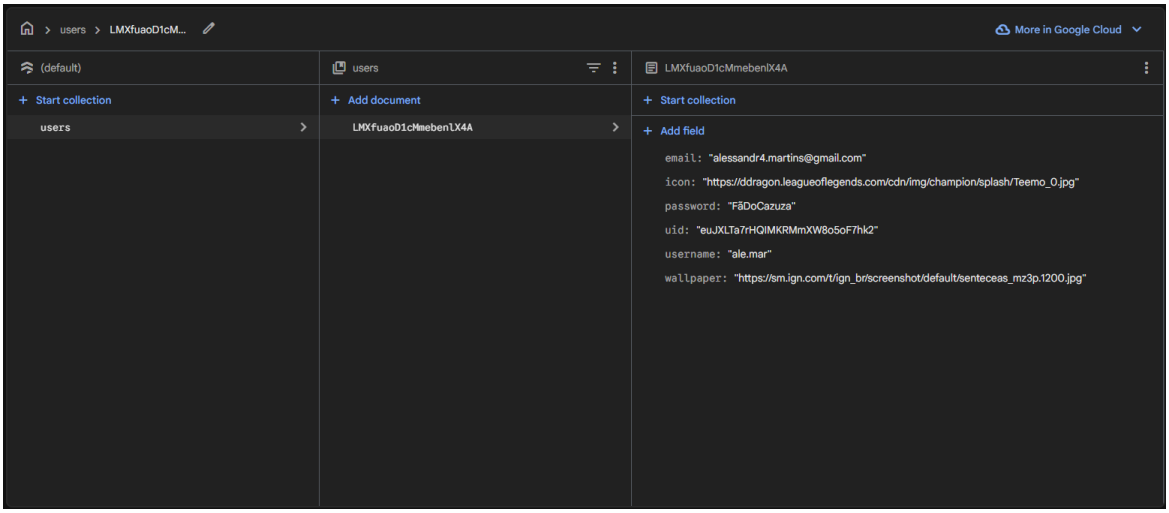
Essa estrutura oferece flexibilidade e escalabilidade, permitindo que os objetos definidos na modelagem do sistema sejam mapeados de forma direta para o banco de dados.

3.5.1 Estrutura de Coleções e Documentos

A organização interna do Firestore reflete a hierarquia de classes do sistema. Conforme observado na captura de tela da implementação real (Figura 9), os dados dos usuários são armazenados na coleção raiz denominada users.

Cada documento dentro desta coleção representa um objeto único da classe User. O identificador do documento (neste caso, LMXfuao...) atua como a chave primária única, muitas vezes sincronizada com o UID gerado pelo serviço de Autenticação (Firebase Auth).

Figura 9 – Visualização do documento de usuário no Console do Firestore



Fonte: Elaborado pelos autores (2025)

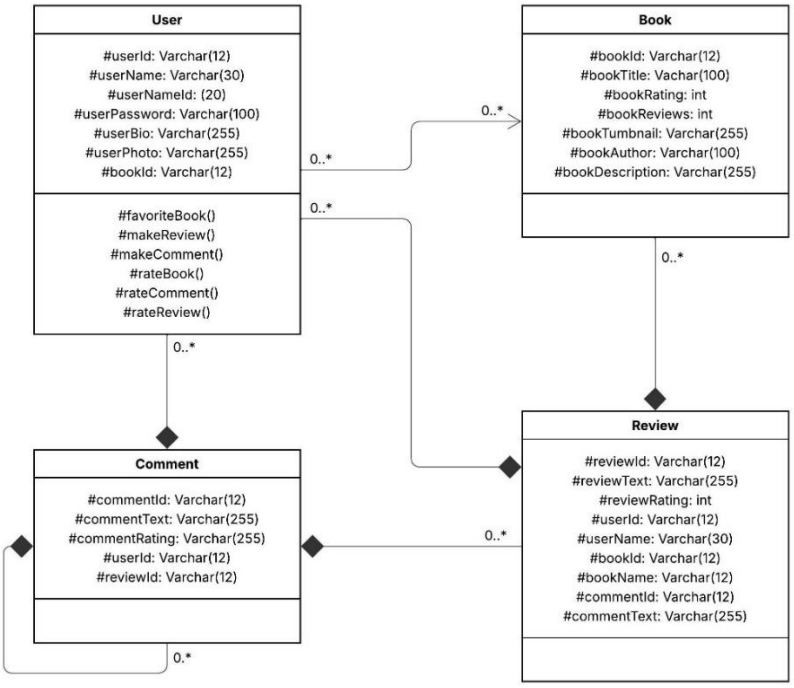
No exemplo acima, observa-se a materialização dos atributos da classe:

a) Campos de Texto: email, password e username são armazenados como strings;
b) Recursos Visuais: Os atributos icon e wallpaper armazenam as URLs das imagens (hospedadas externamente ou no Firebase Storage), permitindo que o front-end carregue a foto de perfil e o fundo personalizado do usuário;
c) Identificação: O campo uid vincula este documento de dados às credenciais de login do usuário.

3.5.2 Mapeamento do Diagrama de Classes para o NoSQL

A estrutura adotada no Firestore é derivada diretamente da modelagem orientada a objetos definida na fase de projeto. A Figura 10 apresenta o Diagrama de Classes da Sprint 4, que serviu de base para a criação dos documentos no banco.

Figura 10 – Diagrama de Classe do Sistema



Fonte: Elaborado pelos autores (2025)

A relação entre o diagrama e o banco de dados ocorre da seguinte maneira:

a) Classe User: Conforme detalhado na Figura A, os atributos definidos no diagrama (como userName, userPassword, userPhoto) foram convertidos diretamente para chaves no documento JSON do Firestore (username, password, icon);
b) Relacionamentos (Associações): O diagrama indica que um User pode ter múltiplos Review e Comment (relacionamento 0..*). No Firestore, essa relação não é feita através de "joins" complexos como no SQL, mas sim através de referências;
c) Ao criar uma Review (classe à direita no diagrama), o sistema salva o userId dentro do documento da review. Isso permite que o sistema faça uma consulta simples: "Busque todas as Reviews onde o campo userId seja igual ao ID do usuário logado";
d) Classe Book: De forma similar, os livros possuem sua própria coleção books. O diagrama prevê atributos como bookTitle e bookAuthor. No Firestore, esses dados são persistidos para garantir que, ao carregar uma resenha, o aplicativo tenha acesso rápido aos dados do livro sem precisar consultar a API do Google Books repetidamente.

Dessa forma, o Cloud Firestore atua não apenas como um repositório de dados, mas como um espelho do estado da aplicação, garantindo que as interações definidas nos métodos da classe (como #makeReview() ou #favoriteBook()) resultem na criação ou atualização imediata dos documentos correspondentes na nuvem.

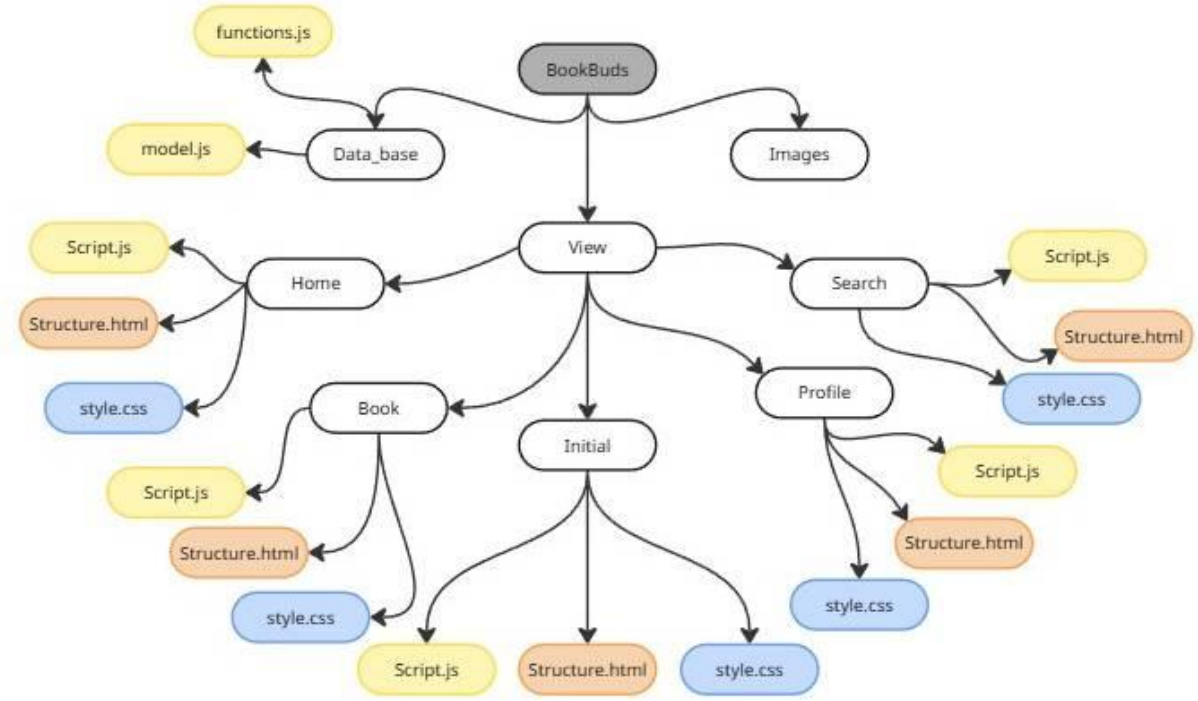
3.6 ARQUITETURA

A arquitetura foi pensada em dois níveis de abstração distintos, mas complementares, sendo eles, a arquitetura da solução, que define a infraestrutura e a integração entre serviços distribuídos, e a arquitetura de software, que detalha a organização interna do código-fonte e os padrões de projeto aplicados.

3.6.1 Distribuição dos diretórios

Além das divisões conceituais e teóricas que se apresentam como uma interpretação, é necessário demonstrar como a arquitetura física em si se apresenta.

Figura 11 – Arquitetura física do sistema



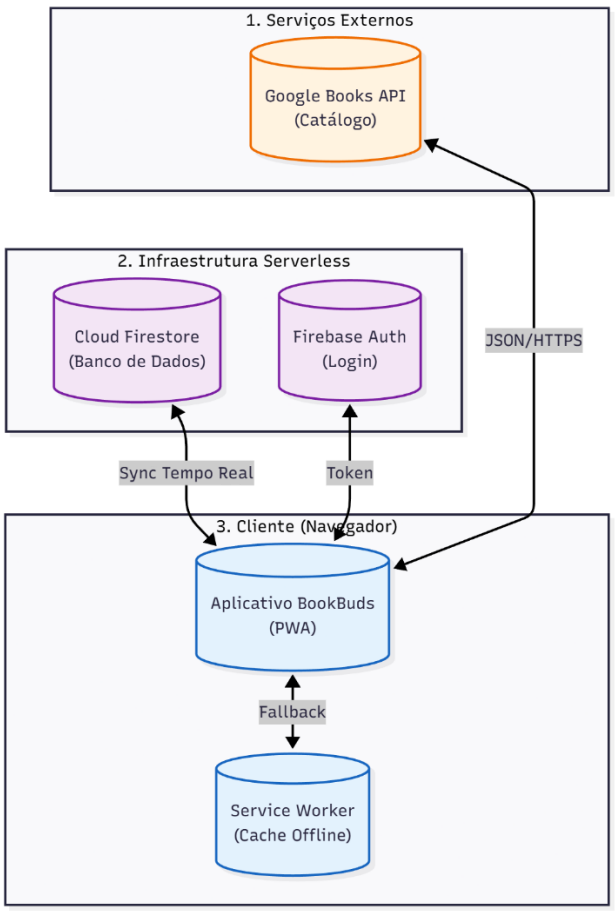
Fonte: Elaborado pelos autores (2025)

3.6.2 Arquitetura da solução

A solução foi concebida sob um modelo sem servidor dedicado, chamado *Serverless*, e utilizando uma abordagem centrada no cliente, chamado *Client-Side Rendering*.

Além disso, aplicativo opera como um PWA, rodando diretamente no dispositivo do usuário, o que permite o acesso offline e o uso de recursos nativos.

Figura 12 – Arquitetura da solução (simplificada)



Fonte: Elaborado pelos autores (2025)

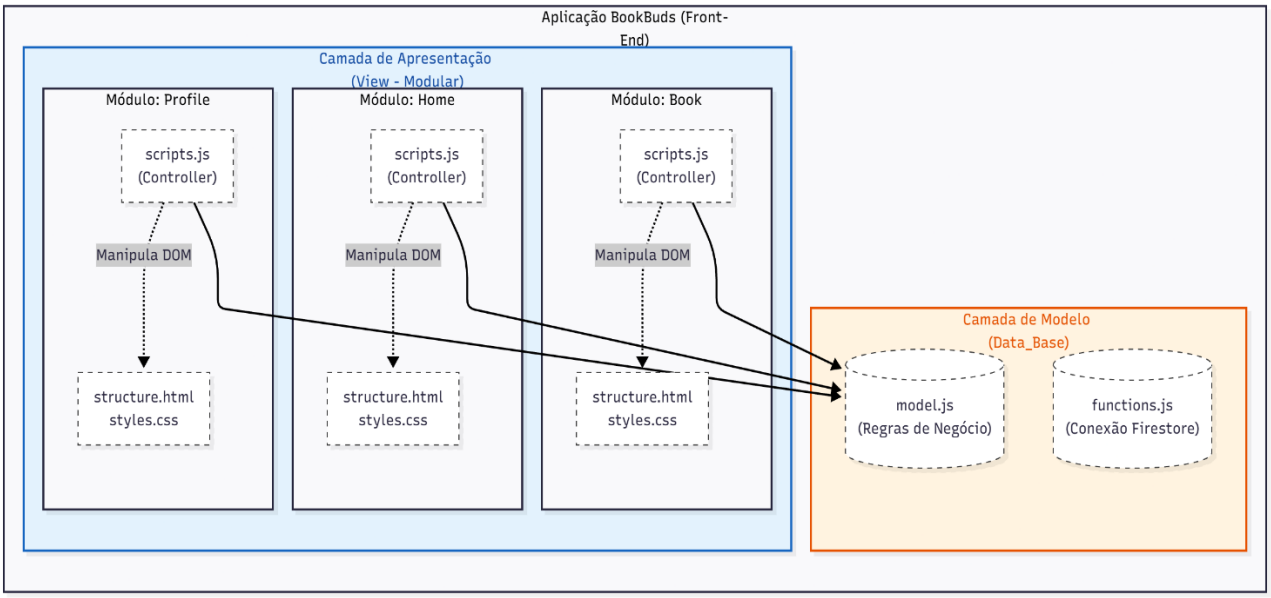
Já a comunicação de dados ocorre de forma descentralizada, identificando-se principalmente em dois processos:

a) Persistência e Autenticação: o aplicativo conecta-se diretamente ao Google Firebase (Cloud Firestore e Authentication) para gerenciar usuários e salvar o histórico de leituras em tempo real;
b) Catálogo de Obras: Para a obtenção de metadados de livros, o front-end consome diretamente a Google Books API, garantindo um catálogo vasto sem a necessidade de manutenção manual de banco de dados.

3.6.3 Arquitetura de software

Para garantir a manutenibilidade, escalabilidade e a organização lógica do código-fonte, a arquitetura do sistema foi estruturada seguindo o padrão de arquitetura em camadas baseada em módulos e fortemente inspirada no padrão de projeto MVC. Diferente de abordagens monolíticas onde arquivos de diferentes naturezas se misturam, optou-se pela separação de responsabilidades, onde cada componente funcional do sistema possui seus arquivos de estrutura, estilo e comportamento isolados.

Figura 13 – Arquitetura do software (simplificada)



Fonte: Elaborado pelos autores (2025)

A organização do projeto divide-se fundamentalmente em duas grandes camadas lógicas:

a) Camada de modelo e persistência: representada no diretório Data_Base, esta camada centraliza toda a lógica de acesso aos dados e regras de negócio globais. Arquivos como functions.js e model.js atuam como a interface de comunicação com o banco de dados Cloud Firestore. Tem como função abstrair

a complexidade das chamadas à API e ao banco de dados, garantindo que a interface do usuário não manipule dados diretamente;

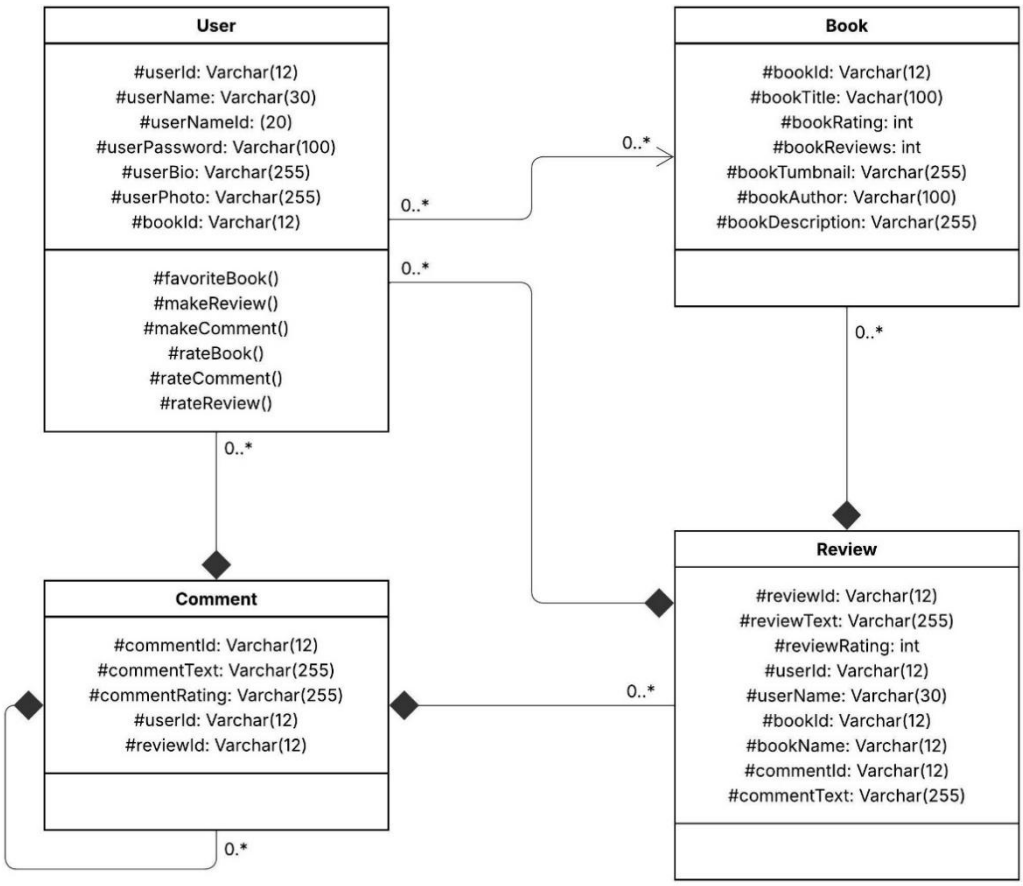
b) Camada de Apresentação e Controle: camada visual foi organizada sob o diretório View, utilizando uma estratégia de Empacotamento por Funcionalidade (Package by Feature). Em vez de separar todos os HTMLs em uma pasta e todos os CSSs em outra, cada módulo do sistema (ex: Home, Book, Profile, Search) reside em seu próprio diretório contendo:

   - View (Visualização): os arquivos structure.html e styles.css definem a estrutura semântica e a estética da interface. Eles são passivos e aguardam instruções;
   - Controller (Controle): o arquivo scripts.js presente em cada módulo atua como o Controlador. Ele intercepta as ações do usuário (cliques, inputs), solicita dados à camada de Modelo (Data_Base) e, com base na resposta, manipula o DOM para atualizar a View.

3.6.4 Modelagem de classes

Uma vez definida a arquitetura modular e as camadas de responsabilidade, faz-se necessário detalhar a estrutura estática dos dados que transitam pelo sistema. O Diagrama de Classes, apresentado na Figura 13, ilustra as principais entidades do BookBuds, seus atributos e os relacionamentos que sustentam as regras de negócio da camada de modelo (Data_Base).

Figura 14 – Diagrama de classe do sistema



Fonte: Elaborado pelo autor (2025)

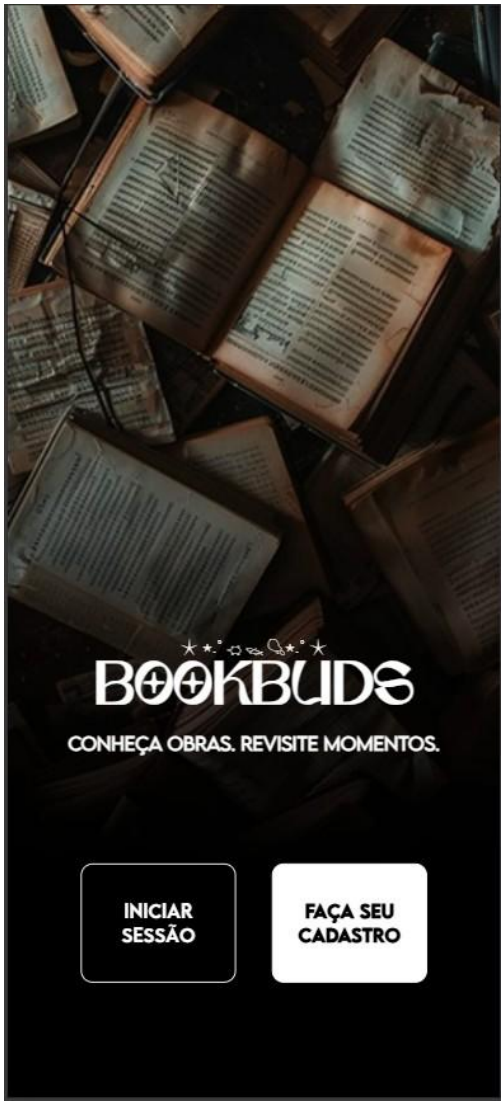
A modelagem destaca a classe User como elemento central, relacionando-se diretamente com as funcionalidades de interação. Observa-se que as classes Book, Review e Comment possuem atributos tipados (como Varchar e Int) que refletem o mapeamento direto para a persistência no banco de dados. Vale ressaltar a relação de composição entre Review e Comment, onde um comentário depende existencialmente de uma avaliação prévia, e o auto-relacionamento na classe Comment, que permite a criação de respostas (*threads*) dentro da plataforma.

3.7 APRESENTAÇÃO DA INTERFACE

Com base na arquitetura e nos requisitos definidos, foi desenvolvida a interface do BookBuds priorizando a usabilidade e a experiência do usuário. As telas a seguir demonstram a materialização dos fluxos desenhados anteriormente.

A Tela Inicial atua como o ponto de entrada da aplicação. Projetada com um design minimalista, ela cumpre a função de direcionar o usuário para as rotas de autenticação, apresentando de forma clara os botões de "Iniciar Sessão" e "Faça seu Cadastro".

Figura 15 – Tela Inicial do Sistema



Fonte: Elaborado pelos autores (2025)

Para o processo de autenticação, optou-se pelo uso de modais (*pop-ups*) sobrepostos, evitando o redirecionamento de páginas e mantendo o usuário contextualizado.

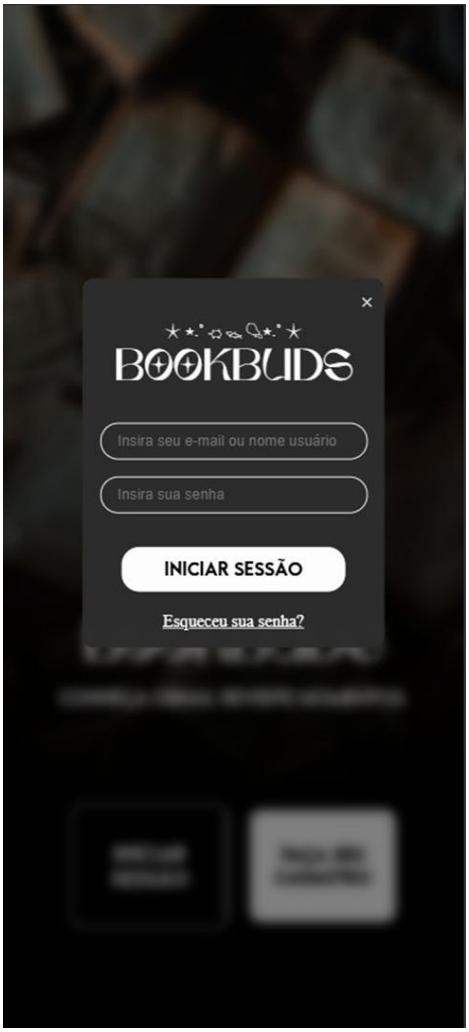
O Pop-up de Cadastro coleta os dados essenciais para o registro e fornece acesso aos termos legais da plataforma.

Figura 16 – Pop-up de Cadastro

Fonte: Elaborado pelos autores (2025)

Já o Pop-up de Login simplifica o acesso de usuários recorrentes, solicitando apenas as credenciais básicas e oferecendo o fluxo de recuperação de senha ("Esqueceu sua senha?") caso necessário.

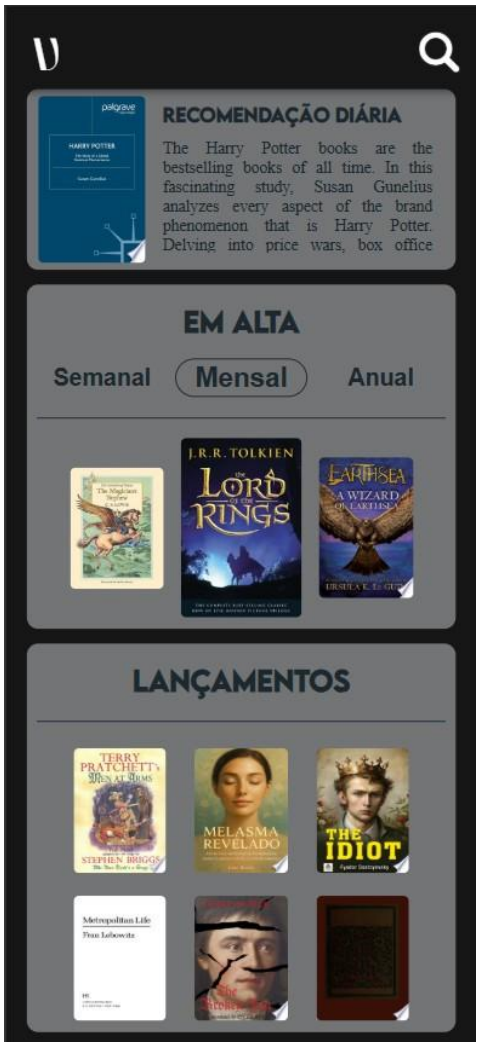
Figura 17 – Pop-up de Login



Fonte: Elaborado pelos autores (2025)

Após a autenticação, o usuário é direcionado para a Tela Home. Esta interface foca na descoberta e recomendação de conteúdo, apresentando seções dinâmicas como "Recomendação Diária", obras "Em Alta" e "Lançamentos", incentivando a exploração do catálogo.

Figura 18 – Tela Home



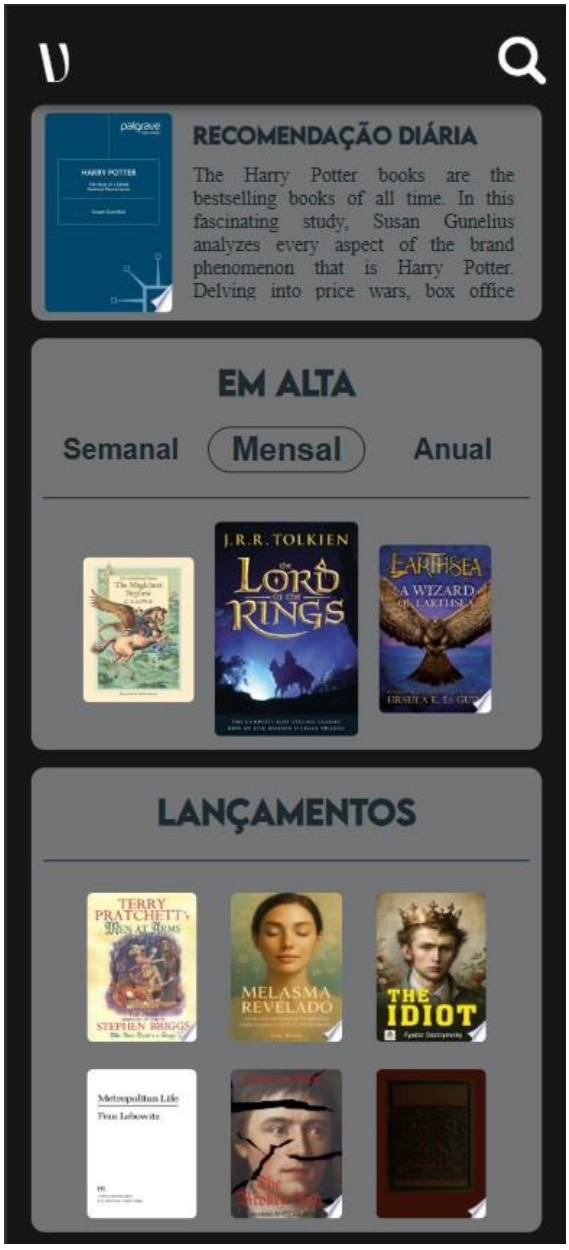
[19]

Fonte: Elaborado pelos autores (2025)

[19] As capas de livros exibidas nessa e em quaisquer outras interfaces são de propriedade de suas respectivas editoras e utilizadas aqui apenas para fins de demonstração da funcionalidade do sistema via API.

Para buscas específicas, a Tela de Pesquisa permite localizar obras por título ou autor. A interface conta com um campo de busca destacado e uma aba de filtros para refinar os resultados, facilitando a navegação em grandes volumes de dados.

Figura 19 – Tela de Pesquisa



Fonte: Elaborado pelos autores (2025)

Ao selecionar uma obra, o usuário visualiza a Tela de Detalhes do Livro . Esta tela centraliza as informações da obra (capa, sinopse, autor) e as ações principais do sistema: adicionar à "Estante" (gerenciamento de status de leitura) e escrever uma "Resenha”.

Figura 20 – Tela de Detalhes do Livro

Fonte: Elaborado pelos autores (2025)

Por fim, a Tela de Perfil (Figura 20) consolida a identidade do usuário na plataforma. Nela, é possível visualizar e editar informações pessoais, acompanhar métricas sociais (seguidores/seguindo) e acessar rapidamente a biblioteca de livros favoritos e o histórico de resenhas publicadas.

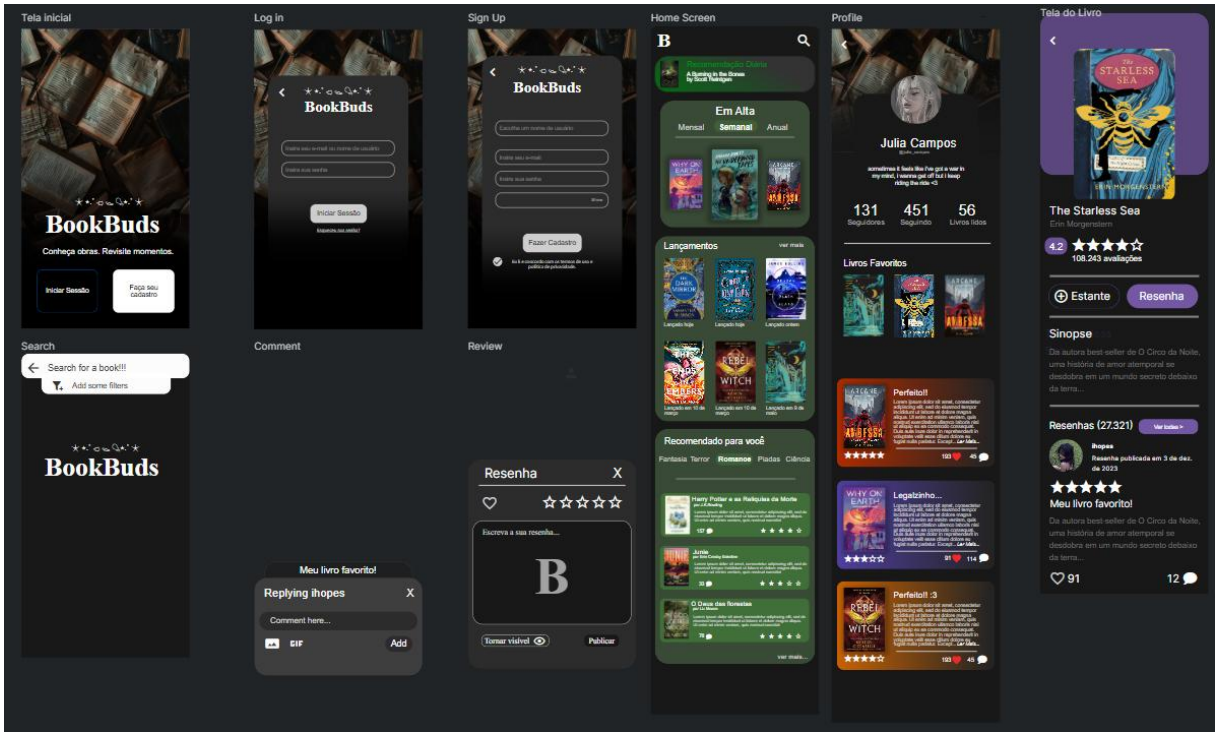
Figura 21 – Tela de Perfil

Fonte: Elaborado pelos autores (2025)

3.8 ANÁLISE COMPARATIVA: PROTOTIPAÇÃO VS. PRODUTO FINAL

A etapa de prototipação de alta fidelidade desempenhou um papel crucial na definição da identidade visual e na arquitetura de informação do *software*. Ao comparar o design idealizado inicialmente:

Figura 22 – Telas da prototipação de alta fidelidade



Fonte: Elaborado pelos autores

Com a interface final implementada (Figuras 15 a 21), observa-se uma alta aderência aos conceitos estéticos propostos, com ajustes estratégicos focados em usabilidade e viabilidade técnica.

a) Identidade Visual e Consistência O produto final manteve a essência "dark mode" do protótipo, preservando a paleta de cores escura que favorece a leitura prolongada e destaca as capas dos livros. A tipografia do logotipo "BookBuds" e a disposição dos elementos centrais na Tela Home permaneceram fiéis ao planejado, garantindo que a identidade da marca fosse preservada. O layout de grid para os livros "Em Alta" e "Lançamentos" foi

implementado com sucesso, mantendo a hierarquia visual que guia o olhar do usuário;

b) Adaptações de Usabilidade (UX) Houve refinamentos notáveis na experiência do usuário durante a implementação:

- Navegação e Acesso: Na Tela Inicial, o protótipo original apresentava um fundo mais complexo e botões menores. A versão final (Figura 15) optou por uma abordagem mais limpa e minimalista, com botões de ação ("Iniciar Sessão" e "Faça seu Cadastro") maiores e mais contrastantes, facilitando o toque em dispositivos móveis, conforme as boas práticas de Mobile First;
- Fluxo de Autenticação: Os modais de Login e Sign Up (Figuras 16 e 17) evoluíram para garantir maior clareza. Enquanto o protótipo sugeria campos de entrada mais estilizados, a implementação final priorizou inputs padrões de formulário web, o que melhora a acessibilidade e o preenchimento automático pelos navegadores.

c) Evolução Funcional Algumas funcionalidades foram readequadas para atender às limitações técnicas e prazos do MVP (Minimum Viable Product):

- Tela do Livro: No protótipo, a tela de detalhes do livro apresentava uma disposição de elementos focada puramente na estética. Na versão final (Figura 20), a disposição foi otimizada para destacar a sinopse e as ações principais ("Estante" e "Resenha"), garantindo que o usuário tenha acesso imediato às funções mais críticas do sistema;
- Interação Social: A interface de "Comment" e "Review" foi simplificada na implementação. O protótipo previa interações complexas com GIFs e respostas aninhadas. A versão entregue foca na funcionalidade central de avaliação (estrelas e texto), assegurando a estabilidade da persistência de dados no Firestore sem comprometer a performance.

Em suma, a comparação evidencia que o BookBuds não apenas atingiu o nível de qualidade visual proposto na fase de design, mas também amadureceu em termos de

engenharia de software, entregando uma interface que equilibra a estética desejada com a funcionalidade necessária para uma aplicação PWA robusta.

3.9 ANÁLISE DE RISCOS TÉCNICOS

A fim de garantir a confiabilidade, a segurança e a longevidade do software, foi realizada uma análise de riscos técnicos detalhada. Dada a natureza da arquitetura escolhida uma aplicação *Client-Side* (PWA) apoiada por infraestrutura *Serverless* e dependente de serviços externos, surgem desafios específicos que transcendem o desenvolvimento de software tradicional.

Esta análise visa antecipar potenciais falhas e definir estratégias de mitigação proativas. Os riscos foram mapeados e categorizados em três pilares fundamentais: Arquitetura e Infraestrutura, Comunicação Externa e Dados, e Segurança e Experiência do Usuário.

3.9.1 Riscos de Arquitetura e Infraestrutura

A escolha por uma arquitetura sem middleware (API proprietária) e o uso de um banco de dados NoSQL (Firestore) trazem agilidade ao desenvolvimento, mas impõem desafios quanto à consistência dos dados e à centralização das regras de negócio. O Quadro 3 detalha os riscos inerentes à estrutura lógica e física do sistema.

Quadro 3 – Análise de riscos da arquitetura e infraestrutura

Risco Identificado	Probabilidade	Impacto	Descrição	Estratégia de Mitigação
Acoplamento Front-End/Regras de Negócio	Alta	Médio	A ausência de uma API proprietária concentra a lógica de negócios no cliente (navegador), dificultando a replicação das regras para outras plataformas futuras.	Migrar validações críticas para *Firebase Cloud Functions*, centralizando a lógica em ambiente seguro *serverless*.
Limites do Plano Gratuito (Firestore)	Média	Alto	O plano *Spark* possui cotas rígidas de leitura/escrita. Testes intensivos ou loops de requisição podem tirar o app do ar temporariamente.	Implementar paginação em listas e otimizar consultas para evitar leituras desnecessárias de coleções inteiras.
Conflitos de Sincronização Offline	Alta	Baixo	Alterações simultâneas em dados (ex: perfil) em dispositivos offline podem gerar inconsistência ao reconectar.	Utilizar a resolução de conflitos nativa do SDK do Firestore ("last write wins") e notificar o usuário sobre o estado da conexão.
Inconsistência de Dados (NoSQL)	Média	Médio	A flexibilidade do NoSQL (schemaless) permite que dados antigos e novos coexistam com estruturas diferentes, podendo quebrar a aplicação.	Manter rigor na camada de *Model* (arquivo model.js) para garantir a padronização dos objetos antes da persistência.

Fonte: Elaborado pelos autores (2025)

3.9.2 Riscos de Comunicação Externa e Dados

O sistema opera como uma interface de agregação, dependendo inteiramente da Google Books API para a exibição de conteúdo. Essa dependência externa cria um cenário de risco quanto à soberania e disponibilidade dos dados, conforme apresentado no Quadro 4.

Quadro 4 – Análise de riscos de comunição externa e dados

Risco Identificado	Probabilidade	Impacto	Descrição	Estratégia de Mitigação
Dependência Crítica da API (Google Books)	Alta	Alto	O sistema não possui a posse dos dados do catálogo; atua apenas como interface de visualização. Se a API falhar, o conteúdo desaparece.	Implementar padrão *"Cache on Read"*: salvar no Firestore os dados essenciais (título, autor, capa) sempre que um usuário interagir com um livro.
Indisponibilidade ou Mudança na API	Baixa	Alto	Interrupção do serviço ou alterações abruptas (*breaking changes*) na estrutura da API do Google.	Tratamento de erros robusto no *Controller* para falha graciosa (*graceful degradation*) e exibição de mensagens amigáveis ao usuário.
Dados Incompletos na Fonte	Alta	Médio	A API pode retornar registros com campos nulos (sem capa ou sinopse), quebrando o layout da aplicação.	Implementar imagens de *fallback* (capa padrão) e verificações condicionais na renderização dos componentes visuais.
Inconsistência de Dados (NoSQL)	Média	Médio	A flexibilidade do NoSQL (schemaless) permite que dados antigos e novos coexistam com estruturas diferentes, podendo quebrar a aplicação.	Manter rigor na camada de *Model* (arquivo model.js) para garantir a padronização dos objetos antes da persistência.

Fonte: Elaborado pelos autores (2025)

3.9.3 Riscos de Segurança e Experiência do Usuário (UX)

Por se tratar de uma aplicação que roda majoritariamente no navegador do cliente (*Client-Side*), a exposição de chaves de acesso e a performance de renderização tornam-se pontos críticos. O Quadro 5 aborda os riscos relacionados à proteção da aplicação e à fluidez da interface.

Quadro 5 – Análise de riscos de segurança e experiencia

Risco Identificado	Probabilidade	Impacto	Descrição	Estratégia de Mitigação
Bloqueio de Renderização (DOM)	Média	Médio	Manipulação direta do DOM via JavaScript puro para pop-ups pode causar travamentos (*jank*) em dispositivos de entrada.	Utilizar tags nativas como <dialog> e priorizar animações via CSS (transform) para liberar a *main thread* do processador.
Vulnerabilidade em Regras de Segurança	Alta	Crítico	Regras de banco de dados permissivas (allow read, write: if true) expõem dados de usuários a exclusão ou roubo.	Configurar *Firestore Security Rules* estritas, validando a autenticação (request.auth.uid) para operações de escrita.
Exposição de Chaves de API	Alta	Médio	Chaves de API visíveis no código cliente (*Client-Side*) podem ser usadas indevidamente por terceiros.	Restringir o uso da chave no console da Google Cloud Platform apenas para requisições originadas do domínio do PWA.
Cross-Site Scripting (XSS)	Média	Alto	Injeção de scripts maliciosos através de campos de texto (resenhas/comentários).	Sanitizar rigorosamente todas as entradas de usuário antes da renderização, evitando o uso de innerHTML com dados não tratados.

Fonte: Elaborado pelos autores (2025)

3.10 ANÁLISE DE RESULTADOS

A análise dos resultados do projeto *BookBuds* foi conduzida sob três perspectivas fundamentais: a eficiência do processo de desenvolvimento (metodologia), a conformidade técnica do software (testes funcionais) e a validação da proposta de valor junto ao público-alvo.

3.10.1 Desempenho do Processo de Desenvolvimento (Scrum)

A adoção da metodologia ágil Scrum permitiu um controle granular sobre o progresso do projeto, dividido em quatro Sprints de 15 dias. A análise dos dados e gráfico, na seção 3.1 e no Apêndice D, demonstra uma evolução consistente da equipe:

a) Curva de Aprendizado e Ritmo: Na Primeira Sprint, focada na estruturação e prototipagem de alta fidelidade, a equipe planejou e executou 29 pontos, mantendo um ritmo constante. Já na Segunda Sprint, dedicada ao *front-end* e lógica inicial, a curva de realização aproximou-se significativamente da linha ideal, indicando um aumento na precisão do planejamento e na maturidade técnica da equipe;
b) Gestão de Complexidade: A Terceira Sprint apresentou o maior desafio técnico com a implementação da *Google Books API*. O gráfico de *Burndown* (Gráfico 4) revela uma queda acentuada inicial seguida de um leve atraso, reflexo da complexidade de integração de serviços externos, mas que foi recuperada ao final do ciclo;
c) Finalização: A Quarta Sprint focou no refinamento e testes, apresentando uma eliminação rápida de pontos na reta final, totalizando 27 pontos entregues e confirmando a estabilidade do fluxo de trabalho na conclusão do MVP (*Minimum Viable Product*).

3.10.2 Validação Técnica e Funcional

Os testes de software, detalhados nos testes de software, encontrado no Apêndice C, foram conduzidos manualmente na última fase do desenvolvimento para assegurar a qualidade das funcionalidades essenciais.

a) Funcionalidades Validadas (Êxito): O núcleo da aplicação demonstrou robustez nas operações críticas para a experiência do usuário:
   - Acesso e Navegação: Os fluxos de "Iniciar Sessão" (Login), "Realizar Cadastro" e a navegação entre as telas principais (Home, Livro, Perfil) funcionaram conforme o esperado;
   - Integração e Pesquisa: A funcionalidade de "Clicar em um livro" e "Pesquisar um livro" operou corretamente, validando a integração com a API externa e o roteamento interno;
   - Interação Social: Recursos vitais para a proposta de rede social, como "Resenhar um livro", "Favoritar" e "Sair de pop-ups", apresentaram comportamento estável e correto;
   - Tratamento de Erros: O sistema respondeu adequadamente a tentativas de uso indevido, como "Pesquisar sem nada escrito" ou "Iniciar Sessão sem dados", exibindo as mensagens de erro esperadas.

b) Limitações e Falhas Identificadas: Os testes também revelaram limitações funcionais que devem ser priorizadas em ciclos futuros de manutenção (Trabalhos Futuros):
   - Gestão de Estante: A funcionalidade crítica de "Adicionar um livro à estante" falhou nos testes ("Nada acontece"), impedindo a catalogação personalizada neste momento;
   - Edição e Recuperação: Ações de "Editar o perfil" e o fluxo de recuperação de senha ("Esqueceu sua senha?") não geraram resposta do sistema;
   - Interface (UI): Foram identificados *bugs* visuais na avaliação por estrelas (seleção total indesejada) e na ausência de resposta ao clicar nos botões de filtros de pesquisa e links legais (Termos/Política).

3.10.3 Validação da Hipótese e Aceitação de Mercado

A pesquisa quantitativa, no Apêndice B, forneceu dados empíricos que validam a premissa central do projeto.

a) Confirmação do Problema: A pesquisa confirmou que, embora 64,5% dos usuários atribuam nota máxima à importância da leitura, existe um descompasso prático, onde 58,1% leram apenas 1 a 2 livros no último ano. Crucialmente, 83,9% dos entrevistados confirmaram sentir dificuldade explicita em manter o hábito de leitura, validando a "dor" que o *BookBuds* se propõe a resolver;
b) Viabilidade da Solução: A aceitação da proposta foi expressiva. 96,8% dos participantes afirmaram acreditar que uma rede social de incentivo ajudaria a manter o hábito de leitura. Dado que 87,1% dos respondentes atualmente não utilizam nenhum aplicativo para gerenciar leituras, o *BookBuds* encontra um mercado desassistido e receptivo à solução tecnológica desenvolvida.

3.10.4 Conclusão dos Resultados

Os resultados indicam que o projeto foi bem-sucedido em construir uma base tecnológica sólida (PWA/*Serverless*) e em validar sua relevância social. Embora os testes de software (Apêndice C) tenham apontado falhas pontuais em funcionalidades secundárias e de gestão de estante que requerem correção, o núcleo da aplicação (Descoberta, Pesquisa, Socialização e Autenticação) está operacional. A alta taxa de aceitação potencial (96,8%) sugere que, após os refinamentos técnicos apontados, a aplicação possui alta viabilidade de adoção real.

4 CONSIDERAÇÕES FINAIS

O presente trabalho atingiu seu objetivo principal ao desenvolver e apresentar o BookBuds, uma solução tecnológica sob a forma de um PWA destinada a mitigar o declínio do hábito de leitura no contexto da sociedade do imediatismo. A proposta de unir a gestão de acervo literário com funcionalidades de rede social demonstrou-se não apenas viável tecnicamente, mas socialmente necessária.

Através da fundamentação teórica, confirmou-se que a "economia da atenção" e a fragmentação do foco são barreiras reais para a leitura profunda. No entanto, a pesquisa de campo realizada validou a hipótese central do projeto: embora 83,9% dos entrevistados sintam dificuldade em manter o hábito de leitura, 96,8% acreditam que uma ferramenta de incentivo social seria eficaz para reverter esse cenário. Isso comprova que o problema não é a falta de interesse pela leitura, mas a ausência de ferramentas adequadas que se integrem à rotina digital dos usuários.

Do ponto de vista técnico, a adoção da arquitetura baseada em camadas (inspirada no padrão MVC e Clean Architecture) e a utilização de uma infraestrutura *serverless* (Firebase/Firestore) provaram ser escolhas acertadas. A integração com a Google Books API solucionou o desafio crítico de popular o catálogo, garantindo que o usuário tenha acesso imediato a milhões de títulos sem a necessidade de cadastro manual, atendendo aos requisitos de usabilidade e desempenho estabelecidos.

Contudo, como em todo ciclo de desenvolvimento de software, foram identificadas limitações. Os testes de software apontaram que funcionalidades secundárias, como a redefinição de senha e a edição avançada de perfil, ainda necessitam de refinamento para garantir a experiência completa do usuário. Além disso, a validação qualitativa (entrevistas) revelou um desejo latente por recursos de gamificação mais profundos, como barras de progresso visual e sistemas de recompensa, que não foram implementados nesta primeira versão (MVP).

Como propostas para trabalhos futuros, sugere-se:

a) Implementação de Gamificação: Desenvolvimento de níveis de usuário e conquistas para aumentar a retenção e a motivação;

b) Refinamento de Algoritmos: Utilização de Inteligência Artificial para aprimorar o sistema de recomendações com base no histórico de leitura (RNF016);
c) Expansão Social: Implementação de chat em tempo real e clubes do livro virtuais dentro da plataforma;
d) Correção e Melhoria Contínua: Ajuste das funcionalidades que apresentaram falhas nos testes finais e otimização da responsividade para uma gama maior de dispositivos.

Conclui-se, portanto, que o BookBuds cumpre seu papel acadêmico e social. Mais do que um repositório de livros, o sistema apresenta-se como um ecossistema de incentivo, provando que a tecnologia, frequentemente apontada como a causa do afastamento dos livros, pode ser a principal aliada na reconexão das pessoas com a literatura. Além de que, apesar das limitações encontradas, o ciclo iterativo demonstrou que a integração entre bibliotecas digitais, rede social e tecnologia web modernas é tecnicamente viável e alinhada às necessidades reais dos usuários.

REFERÊNCIAS BIBLIOGRÁFICAS

BECK, Kent et al. **Manifesto for Agile Software Development**. 2001. Disponível em: https://agilemanifesto.org. Acesso em: 21 nov. 2025.

BOTO, Ana. Cultura digital e imediatismo: impactos na leitura. **Revista USP**, São Paulo, n. 126, p. 45-60, 2025.

DEHAENE, Stanislas. **Os neurônios da leitura**: como a ciência explica o nosso cérebro leitor. Porto Alegre: Penso, 2012.

DENNIS, Alan; WIXOM, Barbara; TEGARDEN, David. **Systems analysis and design**. 6. ed. Hoboken: Wiley, 2014.

DENNIS, Alan; WIXOM, Barbara; TEGARDEN, David. **Systems analysis and design**: an object-oriented approach with UML. 5. ed. Hoboken: John Wiley & Sons, 2015.

DEPEXE, Marcelo; NERES, João. Pirataria de livros digitais no meio acadêmico: desafios e perspectivas. **Revista Brasileira de Educação**, Rio de Janeiro, v. 28, e280045, 2023.

FREIRE, Paulo. **A importância do ato de ler**: em três artigos que se completam. 23. ed. São Paulo: Cortez, 1989.

GEISEL, Theodor Seuss. **I can read with my eyes shut!** New York: Random House, 1978.

GIT. **About Git**. [S. l.], 2025. Disponível em: https://git-scm.com/about. Acesso em: 21 nov. 2025.

GITHUB. **About GitHub**. [S. l.], 2025. Disponível em: https://github.com/about. Acesso em: 21 nov. 2025.

GOOGLE. **Cloud Firestore**. [S. l.], 2025. Disponível em: https://firebase.google.com/docs/firestore. Acesso em: 21 nov. 2025.

GOOGLE DEVELOPERS. **Google Books APIs**: overview. [S. l.], 2025. Disponível em: https://developers.google.com/books. Acesso em: 21 nov. 2025.

HIPÓLITO JÚNIOR, José. **Arquitetura de software**: princípios e práticas. São Paulo: Novatec, 2018.

INSTITUTO PRÓ-LIVRO. **Retratos da leitura no Brasil**: 6ª edição. São Paulo: IPL, 2024. Disponível em: https://www.prolivro.org.br/pesquisas/retratos-da-leitura/. Acesso em: 21 nov. 2025.

INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS. **A guide to the business analysis body of knowledge (BABOK Guide)**. 3. ed. Toronto: IIBA, 2015.

LARMAN, Craig. **Applying UML and patterns**: an introduction to object-oriented analysis and design and iterative development. 3. ed. Upper Saddle River: Prentice Hall, 2004.

MARK, Gloria. **Attention span**: a groundbreaking way to restore balance, happiness and productivity. New York: Hanover Square Press, 2023.

MARTIN, Robert C. **Clean architecture**: a craftsman's guide to software structure and design. Boston: Prentice Hall, 2017.

MICROSOFT. **Visual Studio Code**: code editing. Redefined. [S. l.], 2025. Disponível em: https://code.visualstudio.com/. Acesso em: 21 nov. 2025.

MISHRA, Prakash; ALZOUBI, Omar. Success rates of agile vs. waterfall projects. **Journal of Software Engineering**, [S. l.], v. 12, n. 2, p. 1-12, 2023.

MOZILLA. **CSS**: Cascading Style Sheets. [S. l.]: MDN Web Docs, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS. Acesso em: 21 nov. 2025.

MOZILLA. **HTML**: HyperText Markup Language. [S. l.]: MDN Web Docs, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML. Acesso em: 21 nov. 2025.
MOZILLA. **JavaScript**. [S. l.]: MDN Web Docs, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 21 nov. 2025.

NPM. **About npm**. [S. l.], 2025. Disponível em: https://docs.npmjs.com/about-npm. Acesso em: 21 nov. 2025.

OPENJS FOUNDATION. **Node.js**: about. [S. l.], 2025. Disponível em: https://nodejs.org/en/about/. Acesso em: 21 nov. 2025.

PRESSMAN, Roger; MAXIM, Bruce. **Software engineering**: a practitioner's approach. 8. ed. New York: McGraw-Hill, 2016.

PROJECT MANAGEMENT INSTITUTE. **A guide to the project management body of knowledge (PMBOK Guide)**. 6. ed. Newtown Square: PMI, 2017.

QUANT-UX. **Quant-UX**: prototype, test & learn. [S. l.], 2025. Disponível em: https://quant-ux.com/. Acesso em: 21 nov. 2025.

RETTIG, Marc. Prototyping for user interface design. **Communications of the ACM**, New York, v. 37, n. 4, p. 21–27, 1994.

RODRIGUEZ, Ethel et al. The formative role of home literacy experiences across the first three years of life. **Applied Developmental Science**, [S. l.], v. 13, n. 4, p. 174–187, 2009.

RUDD, Jim; STERN, Scott; ISENSEE, Jay. Low-fidelity prototyping. **Interactions**, New York, v. 3, n. 5, p. 76–85, 1996.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**. [S. l.], nov. 2020. Disponível em: https://scrumguides.org. Acesso em: 21 nov. 2025.

SCRUMPATH+. **Os quatro valores do Manifesto Ágil**. [S. l.], [20--?]. 1 imagem. Disponível em: https://www.scrumpath.com. Acesso em: 21 nov. 2025.

SOMMERVILLE, Ian. **Software engineering**. 9. ed. Boston: Addison-Wesley, 2010.

SOMMERVILLE, Ian. **Software engineering**. 9. ed. Boston: Addison-Wesley, 2011.

SOMMERVILLE, Ian; SAWYER, Steve. **Requirements engineering**: a good practice guide. Chichester: Wiley, 1997.

SOUZA, Maria. Mudanças nos hábitos de leitura na era digital. **Revista Educação & Sociedade**, Campinas, v. 43, e25678, 2022.

STONE, Debbie et al. **User interface design and evaluation**. São Francisco: Morgan Kaufmann, 2005.

TAKEUCHI, Hirotaka; NONAKA, Ikujiro. The new new product development game. **Harvard Business Review**, [S. l.], v. 64, n. 1, p. 137-146, jan./fev. 1986.

VAZQUEZ, Carlos; SIMÕES, André. **Engenharia de requisitos**: software orientado ao negócio. Rio de Janeiro: Brasport, 2016.

WIEGERS, Karl; BEATTY, Joy. **Software requirements**. 3. ed. Redmond: Microsoft Press, 2013.

WILDE, Oscar. **The soul of man under socialism**. London: Penguin Classics, 2007.

GLOSSÁRIO

Back-end: Parte da aplicação responsável pela lógica de negócio, processamento de dados e comunicação com o banco de dados, operando no lado do servidor.

Backlog: Lista ordenada de tudo o que é conhecido ser necessário no produto; é a única origem dos requisitos para qualquer mudança a ser feita no produto.

Branches: Ramificações em sistemas de controle de versão (como o Git) que permitem o desenvolvimento de funcionalidades de forma paralela e isolada do código principal.

Burndown: Gráfico utilizado em metodologias ágeis para representar visualmente o trabalho restante versus o tempo disponível, permitindo acompanhar o progresso da equipe.

Cache on Read: Estratégia de otimização onde os dados são salvos em cache local ou banco de dados no momento em que são lidos de uma fonte externa, reduzindo a necessidade de novas consultas.

Client-Side Rendering: Técnica de desenvolvimento web onde a renderização do conteúdo da página ocorre no navegador do usuário, utilizando JavaScript, em vez de no servidor.

Crowdsourcing: Modelo de produção que utiliza conhecimentos e serviços coletivos, geralmente obtidos através da internet; no contexto do trabalho, refere-se à colaboração dos usuários em resenhas e avaliações.

Dom (Document Object Model): Interface de programação para documentos HTML e XML que representa a página de forma que programas possam alterar a estrutura, estilo e conteúdo do documento.

Elicitação: Processo de descoberta, revisão, documentação e entendimento das necessidades e restrições dos usuários para o desenvolvimento de um sistema.

Fallback: Mecanismo de contingência ou alternativa de segurança; utilizado no sistema para exibir conteúdos padrão (como capas de livros) quando a informação original não está disponível.

Framework: Estrutura de suporte definida, em que um outro projeto de software pode ser organizado e desenvolvido; fornece funcionalidades genéricas para facilitar o desenvolvimento.

Front-end: Camada da aplicação com a qual o usuário interage diretamente; compreende a interface gráfica, design e experiência do usuário.

Gamificação: Aplicação de elementos e mecânicas de design de jogos em contextos de não-jogos, visando aumentar o engajamento e a motivação dos usuários.

Graceful Degradation: Conceito de design que garante que o sistema continue funcionando, ainda que com funcionalidades reduzidas, em caso de falhas em componentes ou serviços externos.

Heurísticas de Nielsen: Conjunto de dez princípios gerais para design de interação utilizados para avaliar a usabilidade de interfaces de usuário.

Incremento: No Scrum, é a soma de todos os itens do Backlog do Produto completados durante uma Sprint e o valor dos incrementos de todas as Sprints anteriores.

Jank: Termo técnico utilizado para descrever travamentos visuais ou engasgos na renderização de interfaces web, geralmente causados por processamento excessivo na main thread.

Listeners: Mecanismo utilizado em bancos de dados em tempo real (como Firestore) que "ouve" alterações nos dados e atualiza a interface do usuário instantaneamente sem necessidade de recarregar a página.

Main Thread: A linha principal de execução do navegador onde o JavaScript é processado e a renderização da página ocorre; o bloqueio desta linha causa travamentos na interface.

Middleware: Software que atua como uma ponte entre um sistema operacional ou banco de dados e as aplicações, especialmente em uma rede.

Package by Feature: Estratégia de organização de arquitetura de software onde os arquivos são agrupados por funcionalidade (ex: Pasta "Login" contendo HTML, CSS e JS do login) em vez de por tipo de arquivo.

Product Owner: Papel no Scrum responsável por maximizar o valor do produto resultante do trabalho do Time de Desenvolvimento e gerenciar o Backlog do Produto.

Sanitizar: Processo de limpeza e validação de dados de entrada em um sistema para prevenir injeção de códigos maliciosos ou falhas de segurança.

Scrum Master: Papel no Scrum responsável por promover e suportar o Scrum, ajudando a todos a entenderem a teoria, práticas, regras e valores da metodologia.

Serverless: Modelo de execução de computação em nuvem onde o provedor de nuvem gerencia dinamicamente a alocação de recursos da máquina, permitindo que os desenvolvedores foquem apenas no código.

Service Workers Scripts: que o navegador executa em segundo plano, separados da página web, possibilitando funcionalidades como notificações push e funcionamento offline (cache).

Sprint: Um período de tempo fixo (time-box) durante o qual um "Pronto", utilizável e potencialmente liberável incremento de produto é criado no Scrum.

Stakeholders: Partes interessadas; indivíduos, grupos ou organizações que podem afetar, ser afetados ou perceberem-se afetados por uma decisão, atividade ou resultado de um projeto.

Story Points: Unidade de medida utilizada em metodologias ágeis para estimar o esforço total necessário para implementar um item do backlog ou uma funcionalidade.

Threads: No contexto de fóruns e comentários, refere-se a uma sequência de respostas conectadas a uma mensagem original, formando uma linha de discussão.

APÊNDICE A – TRANSCRIÇÃO DA ENTREVISTA QUALITATIVA

1 METODOLOGIA

Entrevista semiestruturada realizada com 31 participantes, porém selecionado os 10 participantes de diferentes faixas etárias e hábitos de leitura, mais relevantes, focada em identificar barreiras de entrada, problemas de usabilidade em soluções atuais e validação de funcionalidades propostas.

2 ENTREVISTADOS

2.1 JOVEM UNIVERSITÁRIO, SOFRE COM DESATENÇÃO.

a) Hábitos: Leitura obrigatória (acadêmica); Baixa frequência de leitura por lazer; Conflito de atenção com smartphones;
b) Dores: Déficit de atenção; Interrupções constantes por notificações; Abandono de livros pela metade;
c) Organização: Inexistente/Caótica; Acúmulo de capturas de tela (prints) não gerenciadas;
d) Social: Consumidor passivo de conteúdo rápido (TikTok); Percepção de superficialidade nas redes atuais;
e) Necessidade: Gamificação; Visualização de progresso (Barra de XP); Incentivo visual para conclusão.

2.2 LEITOR ÁVIDO (POWER USER), ORGANIZADO, USUÁRIO DE TECNOLOGIA.

a) Hábitos: Leitura frequente (4-5 livros/mês); Híbrido (Físico e Digital);
b) Dores: Gestão manual trabalhosa de acervo volumoso; Dificuldade em rastrear leituras passadas e futuras;
c) Organização: Planilhas complexas; Frustração com entrada manual de dados (título, autor, sinopse);
d) Social: Consumidor ativo de críticas; Valorização da opinião de pares (crowdsourcing);
e) Necessidade: Automação de Catálogo; Integração com API (Google Books); Cadastro simplificado em um clique.

2.3 USUÁRIO SOCIÁVEL, COMUNICATIVO, SENSÍVEL A SPOILERS.

a) Hábitos: Leitura de Sagas/Séries; Ficção e Fantasia; Leitura em maratona;
b) Dores: Medo de revelações do enredo (Spoilers); Frustração com performance de apps concorrentes (travamentos);
c) Organização: Uso de apps concorrentes, porém insatisfeito com a usabilidade;
d) Social: Alta interação; Desejo de debate; Receio de ler comentários não moderados;
e) Necessidade: Proteção contra Spoilers (Blur/Borrar); Segurança na navegação de resenhas; Performance do sistema.

2.4 PROFISSIONAL CORPORATIVO, LEITURA EM TRÂNSITO (MOBILE).

a) Hábitos: Leitura fragmentada (intervalos curtos); Uso predominante de mobile em deslocamento;
b) Dores: Perda de contexto (esquece onde parou); Perda de referências (indicações recebidas verbalmente);
c) Organização: Improvisada (Galeria de fotos); Dificuldade de recuperação da informação;
d) Social: Compartilhamento ativo (boca a boca), mas sem registro digital;
e) Necessidade: Desempenho (Carregamento < 2s); Sincronização em Nuvem; Funcionamento Offline/Baixa conexão.

2.5 IDOSO OU COM DIFICULDADE VISUAL, TRADICIONALISTA.

a) Hábitos: Preferência por suporte físico; Migração forçada para tablets (necessidade de ampliação de fonte);
b) Dores: Barreiras tecnológicas; Interface complexa; Baixa legibilidade (contraste/tamanho);
c) Organização: Analógica (Papel); Aversão a sistemas burocráticos;
d) Social: Interação intergeracional (família/netos); Linguagem acessível;
e) Necessidade: Acessibilidade Visual; Usabilidade simplificada (Heurísticas de Nielsen); Prevenção de erros.

2.6 O "RELEITOR" NOSTÁLGICO.

a) Hábitos: Releitura de clássicos/infância; Baixa adesão a lançamentos;
b) Dores: Falha de memória sobre experiências passadas; Dificuldade em registrar sentimentos de leituras antigas;
c) Organização: Acervo físico desorganizado
d) Social: Baixa exposição; Desejo de privacidade (evitar "timeline" pública);
e) Necessidade: Privacidade de dados; Controle de visibilidade das resenhas (Público vs. Privado/Diário).

2.7 O "CAÇADOR DE RECOMENDAÇÕES" (INDECISO).

a) Hábitos: Tempo de escolha superior ao tempo de leitura; Abandono frequente por desinteresse;
b) Dores: Dificuldade de descoberta (Curadoria); Desconfiança de listas de "Mais Vendidos"
c) Organização: Nenhuma.;
d) Social: Ceticismo com influenciadores pagos; Busca por autenticidade;
e) Necessidade: Filtros Avançados (Gênero, Tamanho, Tema); Recomendação Assertiva.

2.8 USUÁRIO CASUAL E "ESQUECIDO" (PROBLEMA DE ACESSO).

a) Hábitos: Leitura sazonal (Férias); Picos de intensidade seguidos de hiatos;
b) Dores: Gestão de credenciais (Esquecimento de senhas); Barreira de reentrada no app;
c) Organização: Memória (falha);
d) Social: Participação esporádica condicionada à facilidade de acesso;
e) Necessidade: Facilidade de Login/Recuperação (Autenticação simples); Retenção de usuário.

2.9 LEITOR DE NICHO (TÉCNICO/ACADÊMICO).

a) Hábitos: Leitura utilitária/Estudo; Foco em absorção de conteúdo técnico;
b) Dores: Falta de ferramentas para síntese; Necessidade de registro detalhado de aprendizado;
c) Organização: Arquivos locais (Desktop);
d) Social: Interesse em curadoria profissional/técnica; Networking via leitura;
e) Necessidade: Ferramentas de Resenha estendida (Edição/Formatação); Avaliação técnica.

2.10 O "CRÍTICO DE INTERFACE" (USUÁRIO EXIGENTE).

a) Hábitos: Leitura digital (E-books); Consumo visual;
b) Dores: Rejeição estética a interfaces datadas; Fadiga visual com layouts poluídos;
c) Organização: Uso de concorrentes (Goodreads) por falta de opção, apesar da crítica à UI;
d) Social: Valorização da identidade visual do livro (Capa) como fator de decisão;
e) Necessidade: Interface Moderna (UI/UX); Modo Noturno; Valorização da arte (Capa).

APÊNDICE B - RESULTADOS DA PESQUISA QUANTITATIVA

A pesquisa foi realizada através de formulário online (Google Forms), obtendo um total de 31 respostas. O objetivo foi validar a relevância do problema (dificuldade de leitura) e a aceitação da solução proposta (rede social de incentivo).

Abaixo são apresentados os gráficos resultantes da coleta de dados:

1 PERFIL E HÁBITOS DE LEITURA

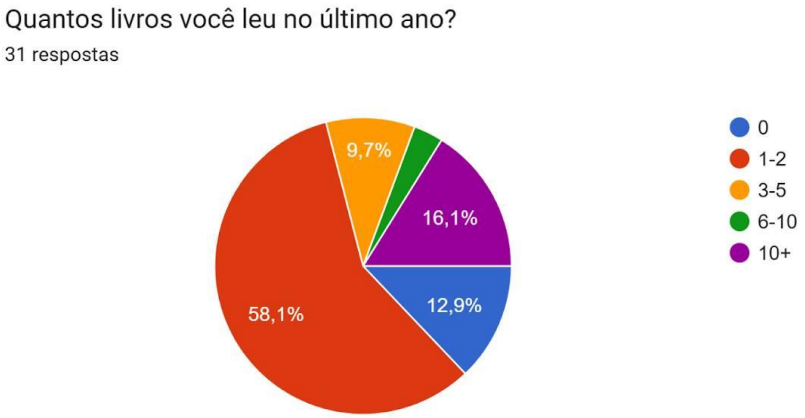
Inicialmente, buscou-se entender a relação do público com a leitura. Conforme observado no Gráfico 1, a maioria absoluta dos entrevistados (64,5%) atribui nota máxima à importância da leitura em suas vidas.

Gráfico 1 – Importância da leitura para o usuário

Fonte: Elaborado pelos autores (2025)

No entanto, ao confrontar a importância dada com a prática real, nota-se um descompasso. O Gráfico 2 revela que 58,1% dos respondentes leram apenas entre 1 e 2 livros no último ano, indicando um baixo volume de leitura.

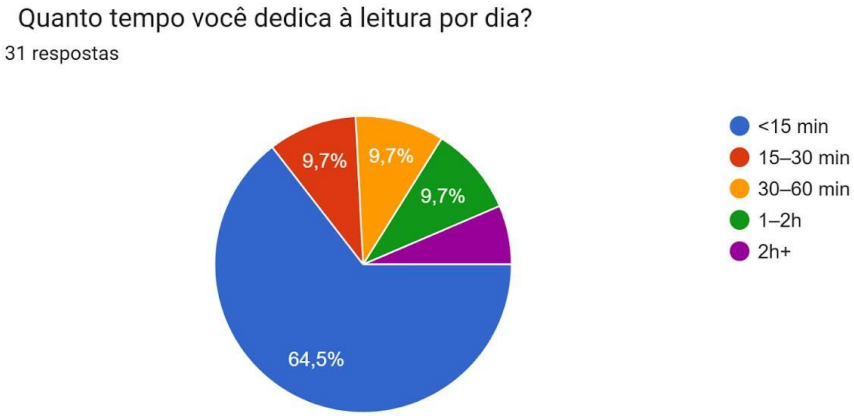
Gráfico 2 – Quantidade de livros lidos no último ano



Fonte: Elaborado pelos autores (2025)

Esse baixo volume é corroborado pelo tempo diário dedicado à atividade. O Gráfico 3 demonstra que a grande maioria (64,5%) dedica menos de 15 minutos por dia para ler, o que reforça a tese do "imediatismo" e falta de tempo discutida no referencial teórico.

Gráfico 3 – Tempo dedicado à leitura por dia

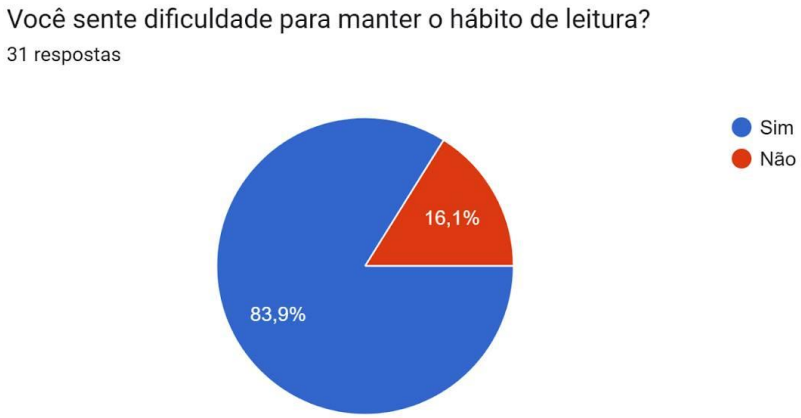


Fonte: Elaborado pelos autores (2025)

2 VALIDAÇÃO DO PROBLEMA E OPORTUNIDADE DE MERCADO

Ao serem questionados sobre a manutenção do hábito, o problema central do projeto foi validado. O Gráfico 4 mostra que 83,9% dos entrevistados sentem dificuldade explícita em manter a constância na leitura.

Gráfico 4 – Dificuldade em manter o hábito



Fonte: Elaborado pelos autores (2025)

Apesar dessa dificuldade, o mercado atual não parece estar atendendo a essa demanda de forma eficaz. O Gráfico 5 revela que 87,1% dos usuários não utilizam nenhum aplicativo para gerenciar suas leituras, sugerindo que as soluções existentes são desconhecidas ou inadequadas para este público.

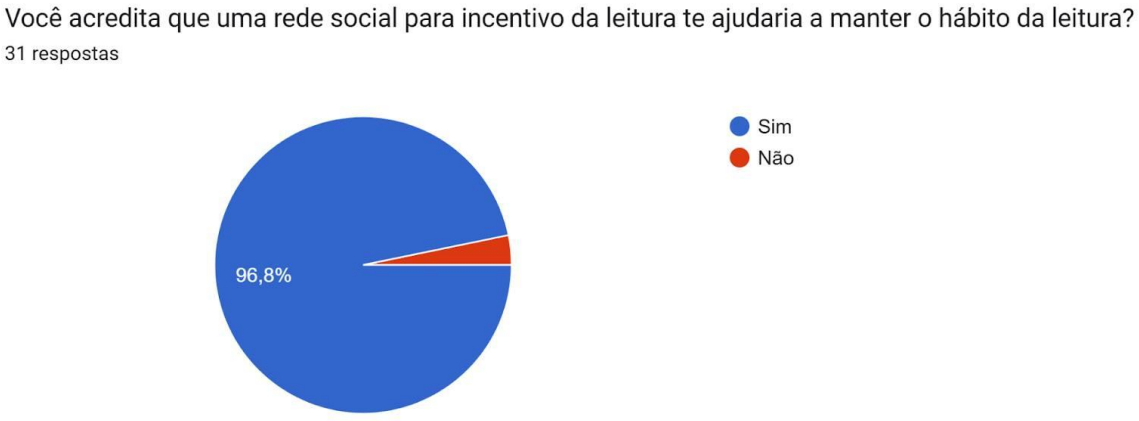
Gráfico 5 – Uso atual de aplicativos de gestão de leitura

Fonte: Elaborado pelos autores (2025)

3 ACEITAÇÃO DA SOLUÇÃO PROPOSTA

Por fim, foi apresentada a proposta de valor do sistema BookBuds. O resultado, apresentado no Gráfico 6, demonstra uma validação quase unânime: 96,8% dos participantes acreditam que uma rede social focada em incentivo ajudaria a manter o hábito, confirmando a viabilidade e o desejo pelo produto desenvolvido.

Gráfico 6 – Aceitação de uma rede social de incentivo



Fonte: Elaborado pelos autores (2025)

APÊNDICE C – TESTES DE SOFTWARE

Os testes de software tiveram seu desenvolvimento realizado na última sprint do projeto, onde cada funcionalidade foi testada e seus respectivos resultados. Eles foram conduzidos manualmente a fim de testar minuciosamente cada funcionalidade presente. Dessa forma, garantimos a qualidade e o funcionamento correto do sistema antes da sua versão final, conseguimos uma satisfação maior por parte dos usuários e asseguramos que o projeto está dentro dos padrões esperados.

Quadro 1 – Testes de software

TESTE	**PASSOS**	**RESULTADO ESPERADO**	**RESULTADO**
Iniciar Sessão	Clicar no botão "Iniciar Sessão"; Inserir seus dados; Clicar em "Iniciar Sessão".	Ir para a tela inicial com login feito.	Aprovado.
Realizar Cadastro	Clicar no botão "Faça seu cadastro"; Inserir seus dados; Clicar em "Cadastrar".	Ir para a tela inicial com login feito e receber a nova conta no banco de dados.	Aprovado.
Colocar nome de usuário ou e-mail e senha incorretamente	Clicar no botão de Iniciar Sessão, colocar dados errados e clicar em iniciar sessão.	O usuário deve receber uma notícia que seu login está errado.	Aprovado.
Fechar Pop-up da tela de iniciar sessão/cadastro	Clicar no botão de X após entrar em um dos pop-ups da tela de login.	Voltar para a tela de login.	Aprovado.
Esqueceu sua senha?	Clicar no texto sublinhado "Esqueceu sua senha?" após clicar no botão de "Iniciar Sessão".	Redirecionar o usuário para uma página de redefinição de senha.	Deu errado. (Nada acontece).

TESTE	PASSOS	RESULTADO ESPERADO	RESULTADO
Termos de serviço	Clicar no texto sublinhado "Termos de Serviço" após clicar no botão de "Faça seu cadastro".	Redirecionar o usuário para uma página mostrando os termos de serviço.	Reprovado. (Nada acontece).
Política de privacidade	Clicar no texto sublinhado "Política de Privacidade" após clicar no botão de "Faça seu cadastro".	Redirecionar o usuário para uma página mostrando as políticas de privacidade.	Reprovado. (Nada acontece).
Cadastro sem dados	Clicar no botão de "Faça seu cadastro", não colocar nenhum dado e clicar em cadastrar.	Ao clicar no botão para registrar, uma notícia que nada está escrito deve aparecer.	Reprovado. (Nada acontece).
Iniciar Sessão sem dados	Clicar no botão de "Iniciar sessão", não colocar nenhum dado e clicar em iniciar sessão.	Ao clicar no botão para iniciar sessão, uma notícia que nada está escrito deve aparecer.	Aprovado.
Mudar a periodicidade no "Em Alta"	Clicar em algum dos botões na sessão "Em Alta".	Mudar os livros em exibição de acordo com a periodicidade escolhida.	Aprovado.
Clicar em um livro	Clicar na imagem de um livro.	Ir para a página daquele livro.	Aprovado.
Mudar o gênero literário nos "Recomendados para você"	Clicar em algum dos botões na sessão "Recomendados para você".	Mudar os livros em exibição de acordo com o gênero literário escolhido.	Aprovado.

TESTE	PASSOS	RESULTADO ESPERADO	RESULTADO
Clicar no botão "Estante" na página de um livro	Clicar no botão "Estante".	Abrir o pop-up para adicionar o livro à estante.	Aprovado.
Clicar no botão "Resenha" na página de um livro	Clicar no botão "Resenha".	Abrir o pop-up para resenhar o livro.	Aprovado.
Clicar para voltar da página de um livro	Clicar no botão com o símbolo de seta para a esquerda.	Voltar para a tela inicial.	Aprovado.
Clicar para pesquisar um livro	Clicar no botão com o símbolo de uma lupa na tela inicial.	Ir para a tela de pesquisa.	Aprovado.
Pesquisar sem nada escrito	Na tela de pesquisa, clicar para pesquisar sem digitar nenhum caractere.	Uma mensagem de erro deve aparecer para o usuário.	Aprovado.
Adicionar filtros a leitura	Na tela de pesquisa, clicar no botão com o símbolo de um filtro.	Filtros devem aparecer para o usuário selecionar.	Reprovado. (Nada acontece).
Voltar da página de pesquisa para a tela inicial	Clicar no botão com o símbolo de seta para a esquerda.	Voltar para a tela inicial.	Aprovado.
Acessar o perfil do usuário	Clicar no botão com o símbolo de um "V" na tela inicial no canto superior esquerdo.	O usuário deve ir para a tela de seu perfil.	Aprovado.
Clicar para editar o perfil	Na tela do perfil, clicar no botão com o símbolo de lápis.	O usuário deve ser capaz de editar seu perfil (Nome, foto de perfil).	Reprovado. (Nada acontece).
Adicionar um livro à estante	No pop-up da estante na tela do livro, o usuário deve selecionar o	O livro deve aparecer na	Reprovado. (Nada acontece).

TESTE	estado do livro e clicar no botão de "Confirmar". **PASSOS**	biblioteca do usuário. **RESULTADO ESPERADO**	**RESULTADO**
Clicar para sair do pop-up da estante	No pop-up da estante na tela do livro, o usuário deve clicar no botão com o símbolo de "X".	O usuário deve voltar para a tela de livro normalmente.	Aprovado.
Resenhar um livro	Clicar no botão de resenha, clicar na caixa de texto e digitar a resenha.	O usuário deve ser capaz de escrever sua resenha dentro da caixa de texto.	Aprovado.
Favoritar um livro	Clicar no botão de resenha, clicar no botão com símbolo de coração.	O coração deve ficar vermelho simbolizando que o livro foi marcado.	Aprovado.
Dar nota a um livro	Clicar no botão de resenha, clicar nas estrelas de acordo com a nota desejada.	As estrelas devem ficar amarelas de acordo com a nota que o usuário escolheu.	Reprovado. (As estrelas todas ficam marcadas).
Clicar para sair do pop-up da resenha	No pop-up da resenha na tela do livro, o usuário deve clicar no botão com o símbolo de "X".	O usuário deve voltar para a tela de livro normalmente.	Aprovado.

Fonte: Elaborado pelos autores

APÊNDICE D – DADOS DAS SPRINTS

1 PRIMEIRA SPRINT

a) Planejamento:

Quadro 1 – Planejamento da primeira sprint

Atividade	**Tempo (em dias)**	**Pontos**
Criação de tela dos livros	2	2
Criação de tela inicial	1	1
Criação de tela de home	3	2
Criação de tela de review	1	1
Criação de tela de pesquisa	1	1
Criação de tela de perfil	2	2
Criação de tela de iniciar sessão	1	1
Criação de tela de cadastro	1	1
Levantamento de requisitos	2	3
Diagrama de caso de uso	2	4
Definição de recursos e ferramentas utilizadas	3	5
Apresentação em slide	1	4
Criação de tela dos status do livro	1	1
Criação de tela de mensagem do livro	1	1
Total	22	29

Fonte: Elaborado pelos autores (2025)

b) Dados do gráfico:

Quadro 2 – Dados do gráfico de Burndown da primeira sprint

Dia	Data	Planejado	Feito (Entrada)	Realizado
1	05/08/2025	29	2	27
2	06/08/2025	26,9285714	1	26
3	07/08/2025	24,8571429	5	21
4	08/08/2025	22,7857143	4	17
5	09/08/2025	20,7142857	2	15
6	10/08/2025	18,6428571	2	13
7	11/08/2025	16,5714286	2	11
8	12/08/2025	14,5	0	11
9	13/08/2025	12,4285714	0	11
10	14/08/2025	10,3571429	3	8
11	15/08/2025	8,28571429	0	8
12	16/08/2025	6,21428571	1	7
13	17/08/2025	4,14285714	2	5
14	18/08/2025	2,07142857	4	1
15	19/08/2025	-7,994E-15	1	0

Fonte: Elaborado pelos autores (2025)

c) Retrospectiva (simplificada):

Quadro 3 – Retrospectiva da primeira sprint

O que deu certo?	**O que deu errado?**	**Ações de melhorias**
Os protótipos iniciais das telas foram criados	Pequenos conflitos de edição com o site escolhido	N/A.
Foi criado os requisitos e ferramentas na documentação do projeto	N/A.	Melhorar requisitos e ferramentas
Foi feito o diagrama de caso de uso	N/A.	Melhorar ações do usuário no diagrama
Foi finalizado a prototipação das telas e novas telas foram adicionadas	N/A.	N/A.
Finalização da documentação inicial do projeto	N/A.	Melhorar diagramação e sumário da documentação
Realização dos slides para apresentação sob o modelo	N/A.	N/A.

Fonte: Elaborado pelos autores (2025)

2 SEGUNDA SPRINT

a) Planejamento:

Quadro 4 – Planejamento da segunda sprint

Atividade	**Tempo (em dias)**	**Pontos (1-5)**
Escolha do meio de desenvolvimento mobile	1	2
Ajustes visuais da tela Início	1	1
Ajustes visuais do popup de login	1	1
Ajustes visuais do popup de registro	1	1
Finalização da estrutura da tela Home	3	3
Desenvolvimento visual base da tela Perfil	2	3
Desenvolvimento visual base da tela Livro	2	3
Desenvolvimento visual base da tela Buscar	1	1
Atualizar a documentação	1	1
Total	13	16

Fonte: Elaborado pelos autores (2025)

b) Dados do gráfico:

Quadro 5 – Dados do gráfico de Burndown da segunda sprint

Dia	Data	Planejado	Feito (Entrada)	Realizado
1	31/08/2025	16	0	16
2	01/09/2025	14,8571429	1	15
3	02/09/2025	13,7142857	0	15
4	03/09/2025	12,5714286	1	14
5	04/09/2025	11,4285714	1	13
6	05/09/2025	10,2857143	2	11
7	06/09/2025	9,14285714	0	11
8	07/09/2025	8	0	11
9	08/09/2025	6,85714286	3	8
10	09/09/2025	5,71428571	0	8
11	10/09/2025	4,57142857	1	7
12	11/09/2025	3,42857143	0	7
13	12/09/2025	2,28571429	1	6
14	13/09/2025	1,14285714	3	3
15	14/09/2025	5,3291E-15	3	0

Fonte: Elaborado pelos autores (2025)

c) Retrospectiva (simplificada):

Quadro 6 – Retrospectiva da segunda sprint

O que deu certo?	**O que deu errado?**	**Ações de melhorias**
Decisão da implementação mobile	Falta de experiências com as ferramentas escolhidas	N/A.
Implementação do PWA	N/A.	Aprofundar em seu funcionamento e ferramentas
Desenvolvimento base do front-end por completo	N/A.	Começar o detalhamento visual
Atualização da documentação	N/A.	N/A.

Fonte: Elaborado pelos autores (2025)

3 TERCEIRA SPRINT

a) Planejamento:

Quadro 7 – Planejamento da terceira sprint

Atividade	**Tempo (em dias)**	**Pontos (1-5)**
Escolha da API	1	2
Implementação da API	2	5
Inicialização das animações em JS e CSS	3	5
Aplicação das Heuristicas de Nielsen	5	5
Produção do diagrama de classe	1	2
Atualização da documentação	2	2
Total	14	21

Fonte: Elaborado pelos autores (2025)

b) Dados do gráfico:

Quadro 8 – Dados do gráfico de Burndown da terceira sprint

Dia	**Data**	**Planejado**	**Feito (Entrada)**	**Realizado**
1	26/09/2025	21	1	20
2	27/09/2025	19,5	2	18
3	28/09/2025	18	1	17
4	29/09/2025	16,5	1	16
5	30/09/2025	15	0	16
6	01/10/2025	13,5	2	14
7	02/10/2025	12	1	13
8	03/10/2025	10,5	3	10
9	04/10/2025	9	2	8
10	05/10/2025	7,5	1	7
11	06/10/2025	6	1	6
12	07/10/2025	4,5	2	4
13	08/10/2025	3	2	2
14	09/10/2025	1,5	2	0
15	10/10/2025	0	0	0

Fonte: Elaborado pelos autores (2025)

c) Retrospectiva (simplificada):

Quadro 9 – Retrospectiva da terceira sprint

O que deu certo?	**O que deu errado?**	**Ações de melhorias**
Decidimos qual API usar e conseguimos implementar ela no nosso código.	A API que usamos não permite edição nas fotos dos livros.	Melhorar algumas interações da API com o código.
Fizemos animações e melhorias visuais através do JS	As dimensões dos containers precisaram ser ajustadas.	Polir o código.
Fizemos slides e atualizamos a documentação do projeto.	Algumas animações não conseguiram ser realizadas pelo JS e foram deixadas de lado por agora.	N/A.
Nos atentamos as heurísticas de Nielsen, verificando quais delas estavam sendo aplicadas no nosso projeto.	N/A.	N/A.
Fizemos o diagrama de classe do projeto.	N/A.	N/A.

Fonte: Elaborado pelos autores (2025)

4 QUARTA SPRINT

a) Planejamento:

Quadro 10 – Planejamento da primeira sprint

Atividade	**Tempo (em dias)**	**Pontos (1-5)**
Finalização das telas	5	4
Finalização do back-end	8	5
Acabar as animações feitas com JS	3	5
Finalização do front-end	4	5
Acabar a documentação do projeto	1	2
Fizemos os testes de software	2	6
Total	23	27

Fonte: Elaborado pelos autores (2025)

b) Dados do gráfico:

Quadro 11 – Dados do gráfico de Burndown da quarta sprint

Dia	**Data**	**Planejado**	**Feito (Entrada)**	**Realizado**
1	22/10/2025	27	1	26
2	23/10/2025	25,2	2	24
3	24/10/2025	23,4	4	20
4	25/10/2025	21,6	1	19
5	26/10/2025	19,8	2	17
6	27/10/2025	18	2	15
7	28/10/2025	16,2	2	13
8	29/10/2025	14,4	1	12
9	30/10/2025	12,6	2	10
10	31/10/2025	10,8	1	9
11	01/11/2025	9	2	7
12	02/11/2025	7,2	1	6
13	03/11/2025	5,4	2	4
14	04/11/2025	3,6	2	2
15	05/11/2025	1,8	2	0

Fonte: Elaborado pelos autores (2025)

c) Retrospectiva:

Quadro 12 – Retrospectiva da quarta sprint

O que deu certo?	**O que deu errado?**	**Ações de melhorias**
Finalizamos as telas e o código propriamente dito.	Algumas animações do JS não funcionando como deveriam.	Algumas chamadas de API podem ser otimizadas.
Fizemos integração da API com outros botões.	N/A.	N/A.
Fizemos os slides e finalizamos a documentação do projeto.	N/A.	N/A.
Mudamos as dimensões do código para que tudo se encaixe melhor.	N/A.	N/A.

Fonte: Elaborado pelos autores (2025)