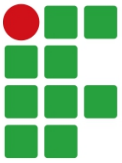


INSTITUTO FEDERAL
São Paulo
Campus Campinas

Gabriela Fernandes Almeida Costa

Backend para Site de Organização Pessoal de Leitura: Desenvolvimento e Otimização de Recursos

CAMPINAS
2024

Gabriela Fernandes Almeida Costa

Backend para Site de Organização Pessoal de Leitura: Desenvolvimento e Otimização de Recursos

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do diploma do Curso Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Campus Campinas.

Orientador: Prof. Dra. Bianca Maria Pedrosa

CAMPINAS
2024

Ficha Catalográfica
Instituto Federal de São Paulo – Campus Campinas
Biblioteca “Pedro Augusto Pinheiro Fantinatti”
Tatiane Salles - CRB8/8946

Costa, Gabriela Fernandes Almeida
C837b Backend para site organização pessoal de leitura: desenvolvimento e otimização de recursos / Gabriela Fernandes Almeida Costa.
– Campinas, SP: [s.n.], 2024.
49 f. : il.

Orientadora: Dra. Bianca Maria Pedrosa
Trabalho de Conclusão de Curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, 2024.

1. Node.js (Programa de computador). 2. Banco de dados da web. 3. Incentivo à leitura. 4. Livros e leitura. 5. Bibliotecas particulares. I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas, Curso de Análise e Desenvolvimento de Sistemas. II. Título.

ATA N.º 24/2024 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Especialização

Na presente data, realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado"Backend para Site de Organização Pessoal de Leitura: Desenvolvimento e Otimização de Recursos", apresentado(a) pelo(a) aluno(a) Gabriela Fernandes Almeida do **CURSO DE TECNÓLOGO EM AN'SLISE DE SISTEMAS** (campus Campinas). Os trabalhos foram iniciados às 16:30 h pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	Instituição	Presença (Sim/Não)
Bianca Maria Pedrosa (Presidente/Orientador)	IFSP	sim
Eliana Alves Moreira (avaliador 1)	IFSP	sim
Júlio César Pedroso (avaliador 2)	IFSP	sim

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do(a) candidato(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo(a) aluno(a), tendo sido atribuído o seguinte resultado:

[X] Aprovado(a) [ ] Reprovado(a)

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino em nome dos demais membros da banca examinadora.

IFSP Campus Campinas - 14/12/2023

Documento assinado eletronicamente por:

- **Bianca Maria Pedrosa**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 06/12/2024 08:49:42.
- **Eliana Alves Moreira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 06/12/2024 10:10:40.
- **Julio Cesar Pedroso**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 12/12/2024 10:04:56.

Este documento foi emitido pelo SUAP em 06/12/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 856736
Código de Autenticação: 46ad7d89ec

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema backend voltado à organização pessoal de leituras, com o objetivo de facilitar a gestão de livros e incentivar hábitos de leitura contínuos e enriquecedores. A solução proposta utiliza a arquitetura MVC (Model-View-Controller), Node.js e Express para a implementação da lógica de negócios, enquanto o Firebase é empregado para o gerenciamento do banco de dados (Firestore), armazenamento de arquivos (Storage) e autenticação de usuários. O desenvolvimento foi realizado seguindo o modelo incremental, organizado em ciclos para assegurar a evolução gradual e contínua do sistema. Cada ciclo abordou funcionalidades específicas, como registro de usuários, cadastro de livros, avaliação de leituras e testes automatizados com Jest. Esses testes garantiram a confiabilidade e a robustez do sistema, validando os principais fluxos de uso. A integração com o Firebase permitiu gerenciar dados como usuários e livros, armazenar arquivos como capas de livros e fotos de perfil, e implementar recursos de autenticação para assegurar que cada funcionalidade estivesse vinculada a um usuário específico. Durante o desenvolvimento, foram priorizadas boas práticas de programação, incluindo a separação de responsabilidades entre os componentes do sistema e a validação de dados nas operações realizadas. O sistema resultante atende às necessidades dos usuários ao centralizar a gestão de leituras, proporcionando um ambiente organizado e intuitivo. Assim, ele contribui para a formação de hábitos de leitura mais consistentes e para o acompanhamento do progresso pessoal, incentivando a reflexão sobre o conteúdo lido.

Palavras-chave: organização de leituras, backend, arquitetura MVC, Firebase, Node.js.

ABSTRACT

This work presents the development of a backend system aimed at personal reading organization, with the goal of facilitating book management and encouraging continuous and enriching reading habits. The proposed solution uses the MVC (Model-View-Controller) architecture, Node.js, and Express for implementing business logic, while Firebase is employed for database management (Firestore), file storage (Storage), and user authentication. The development followed an incremental model, organized into cycles to ensure the gradual and continuous evolution of the system. Each cycle addressed specific functionalities, such as user registration, book management, reading evaluation, and automated testing using Jest. These tests ensured the system's reliability and robustness, validating the main usage flows. The integration with Firebase enabled the management of data such as users and books, the storage of files like book covers and profile pictures, and the implementation of authentication features to ensure that each functionality is linked to a specific user. During development, best programming practices were prioritized, including the separation of responsibilities between system components and data validation in operations. The resulting system meets user needs by centralizing reading management, providing an organized and intuitive environment. Thus, it contributes to the formation of more consistent reading habits and the tracking of personal progress, encouraging reflection on the content read.

Keywords: reading organization, backend, MVC architecture, Firebase, Node.js.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Programming Interface*
APM	*Application Performance Monitoring*
APP	*Application*
BaaS	*Backend-as-a-Service*
CLI	*Command Line Interface*
CRUD	*Create Read Update Delete*
DOM	*Document Object Mode*l
ePUB	*Electronic Publication*
ISBN	*International Standard Book Number*
JSON	*JavaScript Object Notation*
MVC	*Model View Controller*
NoSQL	*Not Only Structured Query Language*
PDF	*Portable Document Format*
REST	*Representational State Transfer*
SaaS	*Software-as-a-Service*
SDK	*Software Development Kit*
TDD	*Test Driven Development*
URL	*Uniform Resource Locator*
XP	*Extreme Programming*

Sumário

1 INTRODUÇÃO

Na era contemporânea, a leitura está gradualmente conquistando um reconhecimento mais abrangente, não apenas como uma atividade acadêmica, mas também como uma fonte de entretenimento e enriquecimento intelectual em diversos domínios da vida. Com o avanço acelerado da tecnologia, o hábito de ler está se expandindo para além dos tradicionais livros físicos, abraçando uma variedade de formatos digitais, como *PDFs* e *ePUBs*. Essa diversificação oferece uma acessibilidade sem precedentes, permitindo que as pessoas leiam em dispositivos eletrônicos, como *tablets*, *smartphones* e *e-readers*, adaptando-se aos estilos de vida modernos e às demandas de uma sociedade cada vez mais conectada digitalmente.

Com essa evolução, surgem novos desafios para os leitores na organização e no gerenciamento de suas leituras. Muitos enfrentam dificuldades específicas, como a falta de uma plataforma centralizada para armazenar informações sobre livros, problemas para estabelecer e acompanhar metas de leitura, e a complexidade de manter uma biblioteca virtual atualizada. Além disso, os leitores frequentemente lidam com a ausência de ferramentas eficazes para monitorar o progresso de suas leituras e gerenciar grandes volumes de informações de forma segura e eficiente.

Nesse contexto, a criação de uma aplicação dedicada à organização pessoal de leitura torna-se cada vez mais pertinente. Essa aplicação visa atender à crescente demanda por uma plataforma centralizada e personalizada, onde os usuários possam adicionar e editar informações sobre os livros que estão lendo, atualizar seu progresso e estabelecer metas de leitura. Além de simplificar a gestão das leituras, essas ferramentas desempenham um papel significativo como um estímulo adicional para a prática da leitura, transformando-a de uma atividade isolada em um hábito contínuo e enriquecedor.

Os desafios enfrentados pelos leitores no gerenciamento de suas bibliotecas pessoais incluem a necessidade de uma lógica de programação eficiente para o processamento de dados, questões relacionadas à escalabilidade dos sistemas e o processamento seguro e eficiente de grandes volumes de dados. Além disso, a entrega de uma experiência personalizada é um ponto crucial a ser abordado.

O principal objetivo deste trabalho é desenvolver o *backend* de uma aplicação de organização de leitura pessoal, utilizando tecnologias como Node.js e os serviços de nuvem da Google, reunidos na plataforma Firebase. Essa aplicação permitirá aos usuários cadastrar seus livros, estabelecer metas de leitura, armazenar arquivos digitais e acompanhar o

progresso das leituras de forma eficiente. Pretende-se criar uma estrutura robusta e escalável, garantindo a segurança e integridade dos dados dos usuários.

Os resultados deste trabalho serão avaliados com base na análise funcional do *backend*, verificando se as funcionalidades implementadas atendem aos requisitos definidos e oferecem uma experiência consistente aos usuários. Será dado destaque à eficiência na manipulação de dados e à segurança no armazenamento das informações dos usuários, garantindo que a aplicação cumpra seu objetivo de facilitar a organização pessoal de leitura de forma confiável e acessível.

2 JUSTIFICATIVA

No contexto contemporâneo, a prática da leitura é reconhecida, cada vez mais, como uma atividade essencial para o desenvolvimento intelectual e pessoal. Com o avanço da tecnologia e as mudanças no estilo de vida, a maneira como as pessoas abordam a leitura também está se transformando. Apesar da profusão de recursos digitais disponíveis, muitos enfrentam desafios para organizar e gerenciar suas leituras de maneira eficaz.

Segundo uma pesquisa divulgada pela CNN, "73% dos entrevistados afirmaram que leram mais durante a pandemia, explorando uma maior diversidade de gêneros literários, e reconheceram que a prática da leitura contribuiu de alguma forma para a melhoria da qualidade de vida" (Failla, citada pela CNN Brasil, 2024). Essa constatação ressalta a relevância de iniciativas voltadas para a promoção e facilitação do acesso à leitura, especialmente em um contexto marcado por mudanças e desafios como o que vivemos atualmente.

Considerando o crescente uso de dispositivos digitais para acessar conteúdo de leitura, destacado em uma pesquisa do Instituto Pró-Livro, que revela que 61% da população prefere a flexibilidade oferecida por aplicativos de leitura (Fonseca, 2024), torna-se cada vez mais importante desenvolver soluções que facilitem a organização e o acompanhamento das leituras. No entanto, muitos leitores enfrentam desafios ao gerenciar suas bibliotecas pessoais, estabelecer metas de leitura e acompanhar seu progresso ao longo do tempo.

Ao oferecer um *web-app* digital dedicado à organização de leituras, o objetivo é proporcionar aos usuários uma experiência mais eficiente e personalizada. Por meio dessa plataforma digital, os usuários poderão catalogar seus livros, estabelecer metas de leitura, armazenar notas e comentários e acompanhar seu progresso de forma intuitiva e acessível. Assim, o desenvolvimento deste projeto não apenas atende a uma demanda existente por soluções de organização de leituras, mas também contribui para promover e facilitar o acesso à leitura em um mundo cada vez mais digitalizado.

O termo *backend* refere-se à parte do sistema que lida com a lógica e o armazenamento de dados, ou seja, a parte não visível para o usuário final. Este projeto, tecnicamente, tem o foco no *backend*, visando o desenvolvimento de conhecimentos sólidos em linguagens de programação, bancos de dados, servidores e outras ferramentas essenciais para a construção de sistemas robustos e eficientes.

3 OBJETIVOS

3.1 Objetivo Geral

Desenvolver o *backend* de um *web-app* organizador de leitura, visando facilitar a organização e o acompanhamento das leituras dos usuários.

3.2 Objetivos Específicos

a) Compreender o contexto e os hábitos de leitura dos usuários: Investigar como os usuários organizam e gerenciam suas leituras atualmente, identificando oportunidades para melhoria.
b) Pesquisar e selecionar as tecnologias adequadas para o desenvolvimento do *backend* do site organizador de leitura.
c) Projetar e implementar um banco de dados eficiente para armazenar informações sobre os livros, metas de leitura e progresso dos usuários.
d) Desenvolver os recursos necessários para que os usuários possam cadastrar, editar e excluir livros, estabelecer metas de leitura e acompanhar seu progresso.
e) Implementar um sistema de autenticação seguro para garantir a privacidade e segurança dos dados dos usuários.
f) Testar e validar o sistema de *backend* para garantir sua funcionalidade e usabilidade.
g) Realizar ajustes e melhorias com base nas avaliações dos testes.
h) Documentar o processo de desenvolvimento e as decisões tomadas ao longo do projeto.

4 FUNDAMENTAÇÃO TEÓRICA

Esta seção tem por objetivo apresentar os conceitos essenciais que fornecem a base teórica necessária para compreender os termos e assuntos recorrentes ao longo do trabalho. Serão abordados temas como arquitetura de *software web*, bancos de dados, APIs, além de aspectos relacionados à gestão de bibliotecas virtuais e práticas de organização pessoal de leitura. Essa compreensão prévia é fundamental para a análise e a implementação eficaz do backend da aplicação proposta.

4.1 Leitura e sites de organização de leitura

A leitura é essencial para o desenvolvimento pessoal, e a tecnologia tem alterado sua prática, com o surgimento de sites de organização de leitura. Essas plataformas auxiliam os leitores na gestão de suas leituras, oferecendo funcionalidades variadas.

4.1.1 Importância da leitura

A leitura, conforme observado por Krug (2015, p. 2), desempenha um papel fundamental na formação do indivíduo, influenciando sua capacidade de análise da sociedade, do cotidiano e, de forma peculiar, ampliando e diversificando visões e interpretações sobre o mundo e a própria vida. Essa prática transcende a mera aquisição de conhecimento, constituindo-se em uma jornada de descoberta, enriquecimento e expansão das fronteiras do pensamento. Ao absorverem livros, artigos, ensaios e outros materiais escritos, os leitores são conduzidos a explorar novas ideias, conceitos e perspectivas, o que não apenas amplia seus horizontes intelectuais, mas também fortalece habilidades cognitivas fundamentais, como análise, interpretação e síntese de informações.

Um dos aspectos mais marcantes da leitura é sua capacidade de promover a empatia e a compreensão mútua. Ao se colocarem na pele dos personagens e vivenciarem suas experiências, os leitores desenvolvem uma maior sensibilidade para as complexidades da condição humana. Essa empatia cultivada pela leitura pode ter um efeito transformador, incentivando a tolerância, o respeito e a compaixão em relação aos outros.

Além disso, a leitura desempenha um papel fundamental no estímulo à criatividade. Ao serem expostos a uma variedade de estilos literários, gêneros e narrativas, os leitores são inspirados a pensar de maneira mais inventiva e original. A capacidade de imaginar mundos

alternativos, resolver problemas complexos e explorar novas ideias é crucial para o progresso humano e a inovação em todas as áreas da vida.

Outro aspecto relevante da leitura é sua influência positiva na comunicação e no domínio da linguagem. A exposição a uma ampla gama de vocabulário, gramática e estilos de escrita contribui para o desenvolvimento de habilidades linguísticas sólidas. Isso não apenas melhora a capacidade de expressão escrita e verbal, mas também aumenta a confiança e a eficácia na comunicação interpessoal.

Além dos benefícios intelectuais, a leitura também desempenha um papel vital na promoção do bem-estar emocional. Muitas pessoas encontram conforto, inspiração e alívio do estresse ao se imergirem um bom livro. A leitura pode ser uma poderosa forma de escapismo, proporcionando momentos de relaxamento e entretenimento em meio às pressões do dia a dia.

Em suma, a importância da leitura vai muito além da mera aquisição de conhecimento. Ela é um veículo para o crescimento pessoal, o enriquecimento cultural, o desenvolvimento da criatividade e a promoção do bem-estar emocional. Cultivar o hábito da leitura não apenas amplia nossos horizontes, mas também enriquece nossas vidas de maneiras profundas e significativas.

> Percebe-se, dessa forma, quão importante é a habilidade de leitura, que ultrapassa os limites da decodificação, efetivando-se como ação, que prepara leitores capazes de participarem da sociedade na qual estão inseridos e, acima de tudo, exercendo o direito e o dever de transformá-la. (Krug, 2015, p. 6).

4.2 TRABALHOS CORRELATOS

Nesta seção serão explorados alguns sites que tem como propósito facilitar a organização da leitura. Existem diversos sites e aplicativos com o objetivo de permitir que o usuário organize suas leituras, e dentre todos foram escolhidos o Skoob, Goodreads e o Maratona para serem abordados no trabalho.

4.2.1 GOODREADS

Goodreads, lançado em janeiro de 2007, é um site para leitores e recomendações de livros, que tem como objetivo ajudar as pessoas a encontrar e compartilhar livros, além de aproveitar melhor a leitura (Goodreads, 2019).

Oferecendo como opções de cadastro próprio e autenticação via redes sociais/terceiros, que é uma forma de autenticação amplamente utilizada. Na parte do cadastro, o *Goodreads*

exige apenas nome, e-mail e senha para a criação da conta; além disso, ele permite que o usuário exclua sua conta a qualquer momento nas configurações.

Embora os detalhes específicos do banco de dados do Goodreads não estejam disponíveis publicamente, é possível, através de análises dos conteúdos disponíveis dos livros na página, pressupor as possíveis abordagens de armazenamento de dados.

As informações dos livros que são armazenadas são: título, autores, sinopses, capas, data de publicação, classificações, notas, resenhas e gêneros. Para os usuários, são armazenados: os nomes, endereços de e-mail, preferências de leitura, listas de desejos, listas de leitura e estatísticas de leitura. Além disso, permite que o usuário tenha uma meta de livros para ler por ano, sendo parte de um desafio que vários leitores podem participar.

O *Goodreads* permite, como formas de interações sociais, amizades entre usuários, grupos de leitura e discussões sobre livros. Alguns dos dados de atividades que podem ser armazenados são: histórico de leitura, livros marcados como lidos e estantes personalizadas.

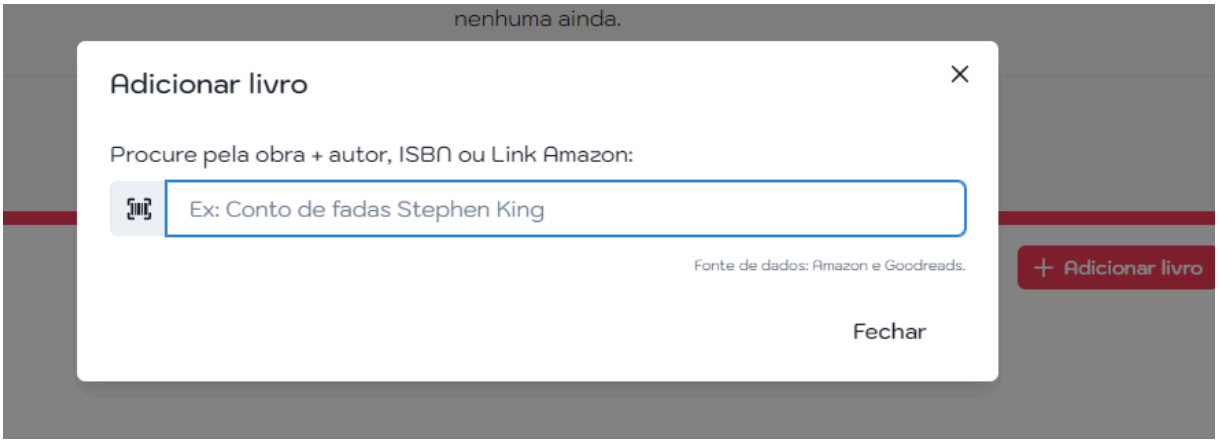
4.2.2 Maratona

O Maratona é um site com um foco maior em maratonas de leituras e leituras coletivas, que oferece diversas ferramentas que ajudam os usuários nesses quesitos (Maratona.app(2024)).

Oferecendo como opções de cadastro e login utilizar uma conta do Google ou uma conta da *Twitch*, o Maratona pede apenas o Gmail nos dados do cadastro. O *login* com uma conta do Google vem sendo muito adotado nas aplicações devido à sua popularidade e integração com diversos serviços *online* e também devido à sua facilidade de uso, especialmente em dispositivos móveis e navegadores que já estão conectados à conta. Já o login com uma conta da *Twitch* tem como benefícios uma comunidade integrada para os usuários da plataforma, mas isso é relevante apenas para os usuários que estão ativamente na plataforma, não sendo útil ou podendo ser confuso e adicionar uma complexidade para aqueles que não são familiares com a plataforma.

O fato de o Maratona fazer uso dos bancos de dados da Amazon e *Goodreads*, como é possível ver na figura 1, para obter informações sobre os livros, sugere que eles podem estar utilizando APIs fornecidas por essas plataformas para acessar dados de catálogo, resenhas e outras informações relacionadas aos livros.

Figura 1 - Fonte de Dados do Maratona.app



Fonte: Maratona.app(2024)

Ao integrar os bancos de dados da Amazon e *Goodreads*, o Maratona pode oferecer aos usuários uma vasta seleção de livros para escolher, permitindo também que os usuários tenham acesso a outras informações, como sinopses, gêneros, quantidade de páginas, entre outras coisas. Porém, isso também pode causar alguns problemas, como uma limitação no controle da precisão e atualização dos dados dos livros em comparação com uma solução de banco de dados proprietária, o risco causado por uma dependência de uma API externa, onde qualquer interrupção ou mudança pode afetar a disponibilidade e funcionalidade do Maratona, o que pode prejudicar a experiência do usuário.

4.2.3 Skoob

É a maior rede social para leitores do Brasil. Funciona como uma estante virtual, onde se pode colocar não só os livros que já leu, como aqueles que ainda deseja ler. É proposto que o site ofereça para o usuário uma forma organizada de registrar suas leituras, para que o usuário não se perca durante elas. Além disso, oferece também a possibilidade de compartilhar suas opiniões, fazer trocas de livros, participar de sorteios, entre outras coisas (Skoob).

Como é possível observar na figura 2, o Skoob oferece como opções de *login* e cadastro o email ou a conta do Facebook, solicitando nome, apelido, sexo, estado, email e senha como dados de cadastro. O *login* com email é prático, por todos terem um endereço de email, permitir a autenticação de dois fatores e também permitir que os usuários tenham total controle sobre suas contas, mudanças de senhas e de configurações de segurança, mas existe o risco de o usuário cometer erros de digitação ao inserir o endereço de email. Já o *login* com

Facebook possui uma facilidade de uso ao eliminar a necessidade de criar e lembrar novas credenciais, mas alguns usuários podem se preocupar com questões de privacidade ao vincular a conta do Facebook a outros serviços, e também existe a possibilidade de que nem todos os usuários têm ou desejam usar o Facebook.

Figura 2 - Página de Login e Cadastro Skoob

Fonte: Skoob(2024)

Embora os detalhes específicos do banco de dados do Skoob não estejam disponíveis publicamente, é possível, através de análises dos conteúdos disponíveis dos livros na página, pressupor as possíveis abordagens de armazenamento de dados.

Um dos aspectos importantes do Skoob é a sua capacidade de armazenar grandes volumes de dados relacionados aos livros, incluindo informações como título, autor, sinopse, capa, avaliações dos usuários, entre outros. Considerando a natureza escalável e de alta disponibilidade que uma plataforma como o Skoob requer, é necessário que eles utilizem um banco de dados robusto e escalável. Porém, um dos pontos negativos sobre a forma como o Skoob armazena os dados de livros é a possibilidade de se armazenar dados duplicados, ou seja, o mesmo livro é armazenado várias vezes com mudanças mínimas nos dados.

As plataformas Skoob, *Goodreads* e Maratona.app oferecem funcionalidades distintas para seus usuários, baseadas em diferentes escolhas de bancos de dados e arquiteturas de backend. O Skoob, por exemplo, foca em redes sociais voltadas para leitores, permitindo que os usuários compartilhem resenhas, montem prateleiras virtuais e interajam em comunidades de leitura. O *Goodreads* oferece funcionalidades semelhantes, mas com uma abordagem global e integrada à Amazon, oferecendo recomendações e listas personalizadas de leitura. Já

o Maratona.app se concentra na organização de metas literárias e acompanhamento do progresso de leitura, priorizando uma experiência mais minimalista e voltada ao gerenciamento pessoal. Essas plataformas serviram de referência para identificar funcionalidades e abordagens técnicas relevantes, permitindo uma análise comparativa de aspectos como a escolha de banco de dados, escalabilidade, desempenho e interação com o usuário, elementos essenciais para definir as diretrizes do trabalho.

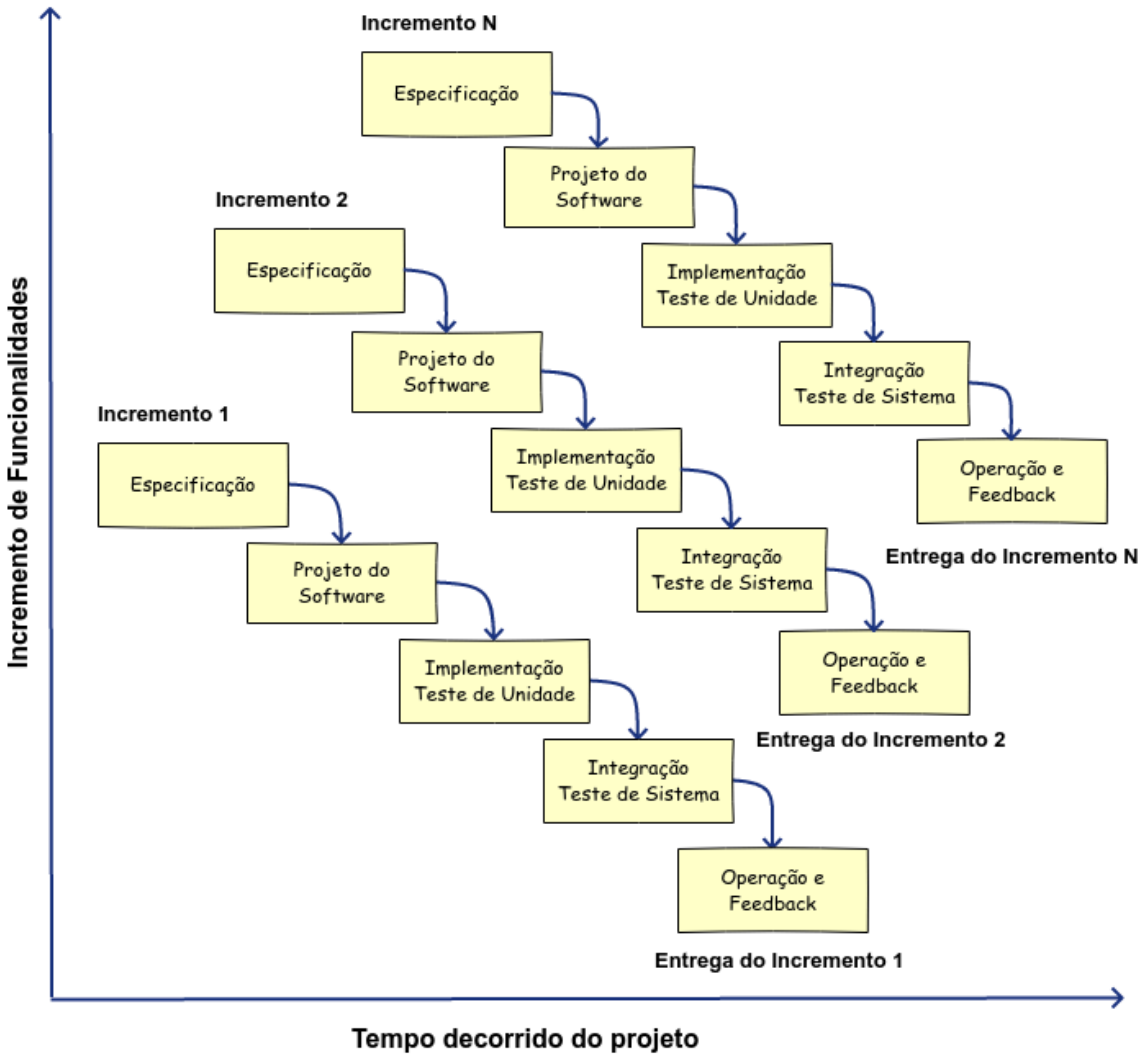
4.3 Desenvolvimento back-end

4.3.1 Modelo de desenvolvimento

Para o desenvolvimento do *backend* de uma aplicação *web* dedicada à organização pessoal de leitura, será adotado o modelo de desenvolvimento incremental. Este modelo é escolhido devido à sua capacidade de fornecer resultados tangíveis em estágios intermediários do projeto, permitindo ajustes e melhorias contínuas por trabalhar de forma incremental, ou seja, pequenas partes do software são entregues a cada vez. “Os modelos incrementais oferecem uma base melhor para criar um processo adaptável caso as alterações possam ser gerenciadas de forma inteligente.” (Pressman, 2021, p. 154-155).

Os modelos de processo incremental focam em produzir rapidamente versões operacionais do software e se adequam melhor às mudanças que podem ocorrer durante o desenvolvimento. A figura 3 abaixo, apresenta um modelo do fluxo de desenvolvimento utilizando o processo incremental.

Figura 3 - Modelo de Desenvolvimento Incremental



Fonte: Medium (2019)

4.3.2 Arquitetura de software e o padrão MVC

A arquitetura de *software* é um componente essencial no desenvolvimento de sistemas, pois define a estrutura e os componentes necessários para o funcionamento do *software*, além das interações entre eles. Ao longo do tempo, surgiram diferentes padrões de arquitetura que permitem organizar e otimizar a construção de aplicações, cada um com uma abordagem única para resolver problemas recorrentes no desenvolvimento de *software*.

Entre os diversos padrões, um dos mais amplamente adotados no desenvolvimento de sistemas, especialmente em aplicações *web* e móveis, é o padrão *MVC* (*Model-View-Controller*). Criado em 1979 por Trygve Reenskaug, o *MVC* busca dividir uma aplicação em três camadas distintas, cada uma com uma responsabilidade bem definida,

facilitando a manutenção, o desenvolvimento e a escalabilidade do sistema (Devmedia, 2024a).

O Padrão MVC divide uma aplicação em três componentes:

- ***View*:** é responsável por apresentar os dados ao usuário e solicitar interações (como cliques ou entradas de texto). Ela não manipula diretamente os dados, mas exibe informações provenientes do *Model*.
- ***Model*:** representa a camada de dados e as regras de negócio. No backend, o *Model* é responsável por gerenciar informações, acessar e manipular bancos de dados, além de implementar regras específicas, como validações e cálculos necessários para a aplicação.
- ***Controller*:** é o responsável por gerenciar as requisições e respostas do sistema. O *Controller* atua como intermediário entre as requisições feitas pelo cliente (ex.: APIs RESTful) e o *Model*, direcionando as operações e garantindo que as informações sejam processadas corretamente.

No backend, a ausência de uma camada de *View* significa que o *Controller* retorna respostas diretamente para os clientes da API (como *front-ends* ou outras aplicações). Assim, a separação de responsabilidades entre o *Controller* e o *Model* é essencial para manter a estrutura do sistema organizada e escalável.

Os benefícios de adotar o padrão MVC no desenvolvimento *backend* são:

- **Manutenção simplificada:** A separação entre o *Controller* e o *Model* permite que alterações na lógica de negócios ou no armazenamento de dados não impactem diretamente o gerenciamento de rotas e requisições, e vice-versa.
- **Organização do Código:** A estrutura em camadas permite que os desenvolvedores identifiquem rapidamente onde cada parte da funcionalidade está implementada, reduzindo a complexidade do código.
- **Reutilização de lógica de negócio:** Como o *Model* centraliza as operações relacionadas aos dados, ele pode ser reutilizado em diferentes partes do sistema, como APIs REST ou serviços internos.

Embora o padrão *MVC* seja amplamente utilizado, sua implementação em projetos exclusivamente *backend* requer atenção a alguns desafios:

- **Sobrecarregar o Controller:** é importante evitar que o *Controller* execute lógica de negócios ou manipulação de dados diretamente. Todas as operações devem ser delegadas ao *Model* para garantir a separação de responsabilidades;

- **Boas práticas na modelagem de dados:** o *Model* deve ser projetado de forma que a lógica de negócio e as interações com o banco de dados sejam otimizadas, promovendo um desempenho eficiente e uma estrutura clara;
- **Contribuições do MVC para o Desenvolvimento *Backend*:** no contexto deste projeto, o padrão MVC desempenha um papel fundamental na organização e manutenção do *backend*. Com a separação clara entre o *Controller* e o *Model*, é possível garantir que o código seja limpo, escalável e preparado para futuras evoluções, como a integração com *front-ends* ou a adição de novas funcionalidades.

4.3.3 API DE DESENVOLVIMENTO

4.3.3.1 API REST

Conforme apresentado pelo Red Hat (2024) uma API é um conjunto de definições e protocolos que facilita o desenvolvimento e a integração de aplicativos, estabelecendo um contrato entre o provedor e o consumidor de informações. No contexto das APIs, destaca-se a API RESTful, que segue as restrições do estilo de arquitetura REST, criado por Roy Fielding, cientista da computação norte-americano. Uma API RESTful permite a integração com serviços *web*, seguindo os princípios de REST, o que proporciona uma comunicação padronizada e eficiente entre sistemas distribuídos.

4.3.3.2 NODE.JS

O Node.js é uma escolha robusta para o desenvolvimento *backend* devido à sua natureza como um ambiente baseado em *JavaScript*. Funcionando como um *runtime*, ele oferece um conjunto de APIs que possibilitam a execução de *JavaScript* fora do contexto do navegador *web*.

Uma das características mais marcantes do Node.js é sua abordagem assíncrona, o que significa que ele pode lidar com tarefas demoradas sem bloquear a execução da aplicação, delegando esses processos para segundo plano (Pessôa, 2024).

Além disso, o Node.js permite o uso de uma única linguagem de programação para lidar com requisições entre cliente e servidor. Suas aplicações podem abranger uma variedade

de casos de uso, incluindo APIs REST, *web scraping*, *chatbots*, IoT, servidores *web*, aplicações *desktop*, entre outros.

As principais características do Node.js são:

- **Multiplataforma:** Possibilita o desenvolvimento de aplicativos para diversas plataformas, desde desktop até móveis, e sites *SaaS* (*Software-as-a-Service*).
- **Multi-paradigma:** Oferece suporte a diferentes paradigmas de programação, incluindo Orientação a Objetos, Funcional, Imperativo e Orientado a Eventos;
- **Código aberto:** Como plataforma de código aberto, o Node.js permite acesso ao seu código-fonte, o que possibilita personalização e contribuições diretas para a comunidade;
- **Escalabilidade:** Foi projetado com foco na construção de aplicações *web* escaláveis, proporcionando flexibilidade para atender a demandas crescentes.

Essas características tornam o Node.js uma escolha popular e poderosa para o desenvolvimento de aplicações backend em uma ampla gama de cenários e necessidades.

4.3.4 Express

O *Express* é um dos *frameworks* mais populares para desenvolvimento de *backend* em Node.js, sendo amplamente utilizado para facilitar o gerenciamento de servidores *web* e o desenvolvimento de APIs. Enquanto o Node.js oferece um ambiente de execução para *JavaScript* fora dos navegadores, ele não suporta diretamente tarefas como gerenciar diferentes rotas, lidar com requisições *HTTP*, como *GET*, *POST* e *DELETE*, servir arquivos estáticos ou apresentar respostas dinâmicas por meio de templates. O *Express* atua para suprir essas limitações, permitindo que os desenvolvedores configurem rotas para diferentes *URLs* e verbos *HTTP*, e definam outras configurações importantes, como a porta de conexão e a localização dos modelos para renderização dinâmica de respostas.

O *Express* também fornece uma abordagem flexível para a aplicação de *middlewares* (Mozilla Developer Network, 2024), que são funções executadas na sequência de processamento das requisições, permitindo a personalização de funcionalidades e o tratamento de requisitos específicos do sistema. Além de possibilitar a adição de *middleware* de terceiros, o *Express* permite que os desenvolvedores criem seus próprios pacotes de *middlewares* para resolver problemas particulares no desenvolvimento, promovendo uma arquitetura modular e escalável. A abstração fornecida pelo *Express* simplifica a configuração de servidores *web*,

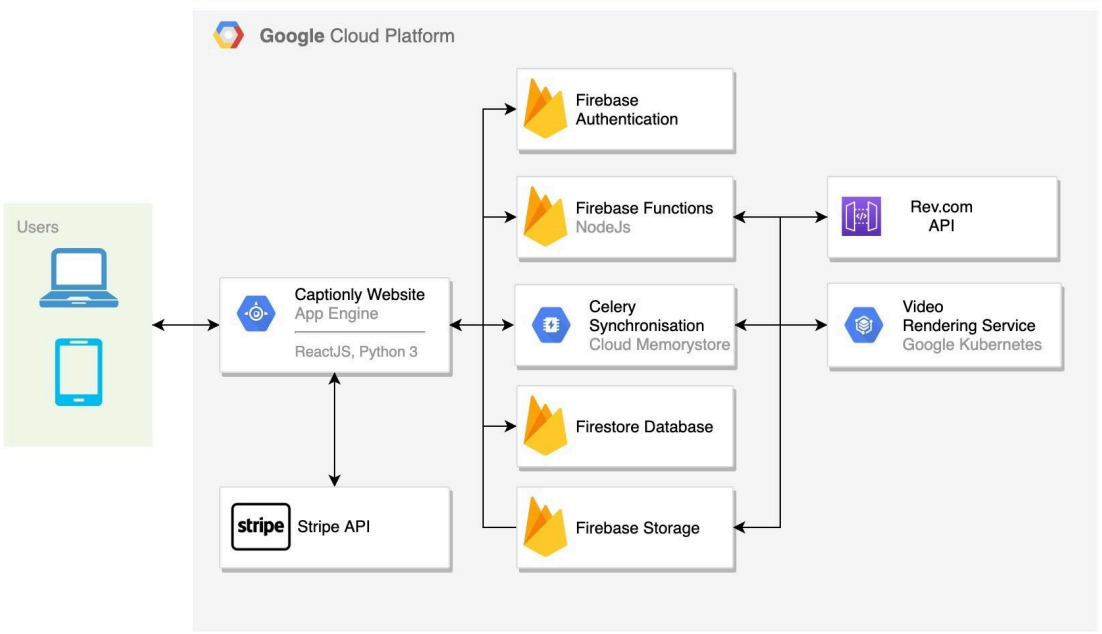
facilitando a criação de aplicações robustas e altamente personalizáveis, onde o servidor pode ouvir requisições e retornar respostas relevantes com eficiência.

4.3.5 Firebase

O *Firebase* é uma plataforma de *BaaS* (*backend as a service*) desenvolvida pelo Google, que oferece uma infraestrutura pronta para o uso aos desenvolvedores de aplicativos (Ribeiro, 2023). Ao optar pelo *Firebase*, os desenvolvedores podem se concentrar na criação da aplicação em si, sem a necessidade de lidar com a complexidade da configuração e manutenção de servidores.

Essencialmente, o *Firebase* é como uma caixa de ferramentas completa para o desenvolvimento multiplataforma, oferecendo uma variedade de recursos, como é apresentado na figura 4, para aprimorar e expandir aplicativos de forma eficiente. Com o *Firebase*, não é preciso se preocupar com hospedagens, configuração ou segurança, pois a plataforma já cuida desses aspectos (*I K G* Sudiartha *et al.*, 2020).

Figura 4 - Arquitetura do Firebase



Fonte: Google Cloud Platform[1].

[1] Google Cloud Plataform: https://cloud.google.com/

Abaixo estão destacados os principais tópicos do Firebase que serão abordados neste trabalho, delineando suas características e benefícios essenciais para o desenvolvimento de aplicativos modernos e eficientes:

- **Auth:** O Firebase *Authentication* é uma solução abrangente para autenticar usuários em aplicativos, oferecendo suporte a várias formas de autenticação. É possível implementar a autenticação utilizando tanto a FirebaseUI como também através da *SDK* do Firebase *Authentication*, e ele se integra com outros serviços do Firebase e permite personalização por meio de *back-ends* personalizados.
- **Firebase Firestore:** O Firebase *Firestore* é um banco de dados hospedado na nuvem. Os dados são armazenados como *JSON* e sincronizados em tempo real para cada cliente conectado. Todos os clientes compartilham uma instância do *Firestore* e recebem automaticamente atualizações com os dados mais recentes.
- **Cloud Storage:** Os *SDKs* do Firebase para *Cloud Storage* permitem que os desenvolvedores realizem operações de *upload* e *download* diretamente dos clientes, proporcionando a capacidade de retomar operações interrompidas em caso de conexão de rede fraca. Os arquivos são armazenados em um *bucket* do Google *Cloud Storage*, garantindo acesso flexível tanto pelo Firebase quanto pelo Google *Cloud*, possibilitando também o processamento do lado do servidor, como filtragem de imagens ou transcodificação de vídeo. A integração com o Firebase *Authentication* permite a identificação de usuários e o controle de acesso a arquivos de forma granular, seja tornando-os públicos ou privados conforme necessário. Essa integração oferece escalonamento automático no *Cloud Storage*, eliminando a necessidade de migração para outros provedores e proporcionando diversas vantagens em termos de armazenamento e segurança.

4.4 Banco de dados

Um banco de dados *NoSQL* (*Not Only SQL*) é uma estrutura de armazenamento de dados que difere do modelo relacional, baseado em tabelas, encontrado na maioria dos sistemas de banco de dados convencionais (Microsoft Learn, 2024). Os bancos de dados *NoSQL* podem ser representados como pares chave/valor simples, documentos *JSON* ou até mesmo como um grafo composto por bordas e vértices. Os bancos de dados NoSQL revolucionaram a forma como armazenamos e gerenciamos dados em aplicações web

modernas, como consequência muitas das soluções para desenvolvimento web em nuvem, tais como o Firebase, oferecem este tipo de banco de dados.

4.5 TESTES DE *BACKEND*

No contexto do desenvolvimento de *software*, os testes de *backend* desempenham um papel fundamental na garantia da qualidade, confiabilidade e desempenho das aplicações. Ao testar o *backend* de um sistema, os desenvolvedores verificam se os diversos componentes e funcionalidades estão operando conforme o esperado, assegurando que a aplicação funcione corretamente e atenda aos requisitos estabelecidos.

Os testes unitários, também conhecidos como testes de unidade, constituem uma prática importante no desenvolvimento de software, permitindo que os desenvolvedores avaliem a qualidade e a funcionalidade de pequenas unidades de código (Coodesh, 2024c). Esses testes concentram-se em verificar o comportamento de funções, métodos ou classes individualmente, garantindo que cada componente do sistema opere conforme o esperado. Ao isolar partes específicas do código para teste, os desenvolvedores podem identificar e corrigir erros de forma eficiente, melhorando a estabilidade e a confiabilidade do *software* como um todo.

O principal objetivo dos testes unitários é assegurar que cada parte do código funcione corretamente de forma independente, sem depender de outros componentes do sistema. Isso permite que os desenvolvedores identifiquem e resolvam problemas de forma rápida e eficaz, antes mesmo que eles se tornem mais complexos e impactem o funcionamento do sistema como um todo.

Além disso, os testes unitários desempenham um papel fundamental na prevenção de regressões, garantindo que alterações feitas no código não introduzam novos *bugs* ou afetem o funcionamento de partes já existentes do *software*. Isso é especialmente importante em ambientes de desenvolvimento ágil, onde as mudanças no código são frequentes e rápidas.

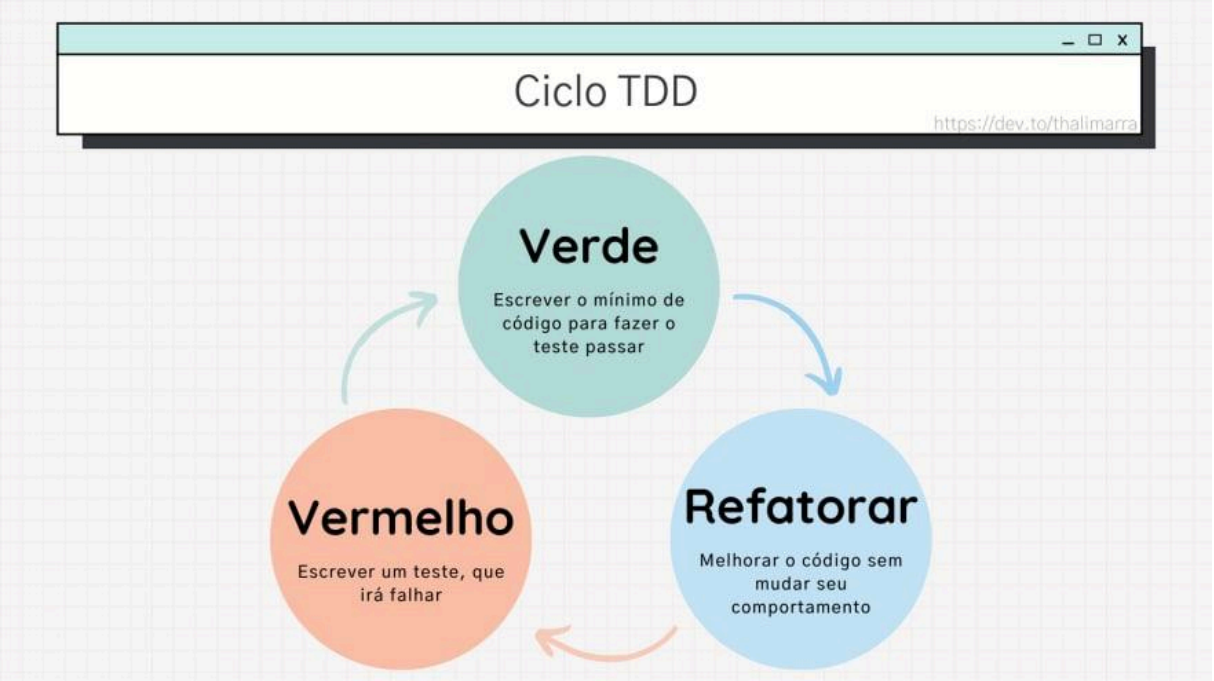
Uma metodologia amplamente adotada para auxiliar no desenvolvimento de *software* é o TDD (*Test Driven Development*), que teve origem na metodologia XP[2] (*Extreme Programming*) e pode ser aplicado em diversas abordagens de desenvolvimento. O TDD é reconhecido como uma maneira eficaz de separar o projeto lógico do físico. Nele, os testes são escritos antes da implementação do sistema, permitindo a identificação precoce de

[2]A extreme programming (XP) é uma metodologia ágil que prioriza feedback contínuo, entregas incrementais e comunicação constante, adaptando-se a requisitos em rápida mudança (Sydle, 2024).

problemas durante a implementação. Após a implementação da funcionalidade, o código é refatorado sem alterar seu comportamento, conforme ilustrado na Figura 5.

Esses testes não apenas auxiliam na compreensão do projeto, mas também esclarecem as expectativas em relação ao código (Devmedia, 2024b).

Figura 5 - Fluxograma TDD



Fonte: Marra (2021)

Assim como um carro requer testes não apenas em suas partes individuais, mas também em sua integração como um todo, um sistema de *software* necessita de testes abrangentes para garantir seu funcionamento adequado em todos os níveis. Isso reforça a importância dos testes unitários e do TDD no desenvolvimento de *software* de alta qualidade e confiabilidade.

4.5.1 Jest

O Jest é um *framework* de testes em *JavaScript* com foco em simplicidade e eficiência, amplamente utilizado no ecossistema de desenvolvimento para garantir a qualidade do código. Criado pelo Facebook, o Jest inicialmente foi pensado para uso com o *React*, mas se consolidou como uma ferramenta versátil, sendo também eficaz em testes de *backend* desenvolvidos com Node.js e *Express*.

No contexto de aplicações *backend*, o Jest é amplamente empregado para testes unitários, permitindo verificar o funcionamento correto de funções e módulos individualmente (Coodesh, 2024b). Esses testes são essenciais para minimizar erros, reduzir retrabalho e aumentar a confiabilidade do sistema. O *Jest* permite a execução de testes em paralelo, o que contribui para resultados rápidos e consistentes, além de facilitar a geração de relatórios de cobertura de código, prática que beneficia a manutenção e qualidade da aplicação. Devido a essas características, Jest se tornou uma ferramenta popular em projetos *JavaScript* e é utilizada por empresas de grande porte, como Facebook, Twitter, Airbnb e Spotify.

5 METODOLOGIA

Esta seção apresenta a metodologia utilizada para o desenvolvimento do projeto, que visa criar o *backend* de uma aplicação *web* dedicada à organização pessoal de leitura. Adotou-se a metodologia de desenvolvimento incremental, que consiste em uma série de etapas, que constituem um ciclo, que serão seguidas, utilizando um conjunto específico de ferramentas e abordagens para facilitar o processo de desenvolvimento do *backend* da aplicação.

5.1 Primeiro ciclo

Durante o primeiro ciclo foram analisadas plataformas que possuem funcionalidades similares às funcionalidades que são desejadas, e através dessa análise os requisitos do trabalho foram criados. Além disso, também foram definidas quais ferramentas serão utilizadas no decorrer do projeto.

5.1.1 Análise de trabalhos correlatos

Utilizando as aplicações Goodreads, Maratona e Skoob, foi feita uma análise e comparação entre as funcionalidades do banco de dados disponíveis nos sites. A partir desta análise, foi gerado o quadro abaixo que foi utilizado como base para o desenvolvimento das funcionalidades do trabalho.

Quadro 1 - Comparativo das Aplicações Estudadas

Plataforma	Cadastrar Livro	Adicionar Livro a Lista de Leitura	Atualizar Livro	Deletar Livro	Cadastro Usuário	Excluir Conta	Criar Metas de Leitura
Goodreads	Não	Sim	Sim	Não	Sim	Sim	Não
Maratona	Não	Sim	Sim	Não	Sim	Sim	Sim
Skoob	Sim	Sim	Sim	Não	Sim	Sim	Sim

Fonte: Elaborado pela Autora (2024)

5.1.2 Levantamento e análise de requisitos

Com as funcionalidades identificadas, foram levantados os seguintes requisitos funcionais e não funcionais:

Quadro 2 - Requisitos Funcionais do Sistema

Código	**Descrição**
RF01	Implementar *login* e registro de usuários.
RF02	Gerar e gerenciar tokens de acesso para autenticação API.
RF03	Implementar *logout*.
RF04	Cadastrar livros.
RF05	Editar e excluir livros cadastrados.
RF06	Criar leituras para livros cadastrados.
RF07	Editar e excluir leituras.
RF08	Criar metas de leitura com quantidade de livros e de páginas lidas.
RF09	Editar e excluir metas de leitura.
RF10	Visualizar o progresso da meta de leitura em tempo real, conforme as leituras são atualizadas.
RF11	Permitir que os usuários avaliem livros com notas e comentários.

Fonte: Elaborado pela Autora (2024)

Quadro 3 - Requisitos Não Funcionais do Sistema

Código	**Descrição**
RNF01	Armazenar os dados de forma segura no Firebase.
RNF02	Usar *tokens* de acesso para autenticação API e proteger *endpoints* contra acesso não autorizado.
RNF03	Otimizar *queries* no Firebase para garantir tempo de resposta rápido.
RNF04	Arquitetura do *backend* deve ser capaz de lidar com um grande número de usuários e dados.
RNF05	Usar código limpo e bem documentado.
RNF06	Implementar testes para garantir a qualidade do código.

Fonte: Elaborado pela Autora (2024)

5.1.3 Ferramentas e tecnologias utilizadas

A seguir serão apresentadas as principais ferramentas e tecnologias utilizadas no desenvolvimento da aplicação.

O Firebase, que é uma plataforma de desenvolvimento de aplicativos móveis e *web*, oferecido pelo Google, possui uma ampla gama de ferramentas e serviços para simplificar o desenvolvimento, o crescimento e a escalabilidade de aplicativos. Para o desenvolvimento desse trabalho será utilizado os serviços de: Firebase *Firestore* para o banco de dados, Firebase *Storage* para armazenamento de arquivos e imagens e o Firebase *Authentication* para auxiliar nos processos de *login*, cadastro e autenticação.

O GitHub é uma plataforma baseada em nuvem onde se pode armazenar, compartilhar e trabalhar em conjunto com outras pessoas para escrever código. Seu trabalho colaborativo utiliza o Git, *software* de código aberto, que é um sistema de controle de versão que rastreia alterações em arquivos de forma inteligente.(Github[3]). O Github será utilizado para armazenamento e controle de versionamento do trabalho.

O Node.js foi usado para o desenvolvimento do *backend* do trabalho, por sua base possuir *JavaScript*, funcionando como um ambiente de execução que permite rodar *JavaScript* fora dos navegadores, proporcionando um conjunto de APIs para esse objetivo e sua característica assíncrona permite lidar com tarefas demoradas sem bloquear a execução da aplicação, garantindo eficiência e desempenho.

Complementando o Node.js, o *framework Express* foi escolhido para gerenciar a estrutura das rotas e *endpoints* da aplicação. O *Express* foi utilizado para a criação de APIs RESTful, permitindo que as rotas fossem configuradas de forma organizada e mantendo o código do backend simples e modular. A facilidade na implementação de *middlewares* e a ampla comunidade que suporta o *Express* tornam-no uma escolha confiável para o desenvolvimento de *backends*, especialmente em aplicações que exigem uma arquitetura leve, ágil e de fácil manutenção.

O Insomnia foi utilizado como ferramenta de teste para a API desenvolvida durante o projeto, destacando-se por sua interface intuitiva e funcionalidades práticas, que facilitam a criação, envio e organização de requisições HTTP. Durante o desenvolvimento, o Insomnia permitiu validar as rotas criadas no *backend*, testando *endpoints*, enviando dados no formato apropriado e verificando as respostas da aplicação. Sua capacidade de lidar com diferentes

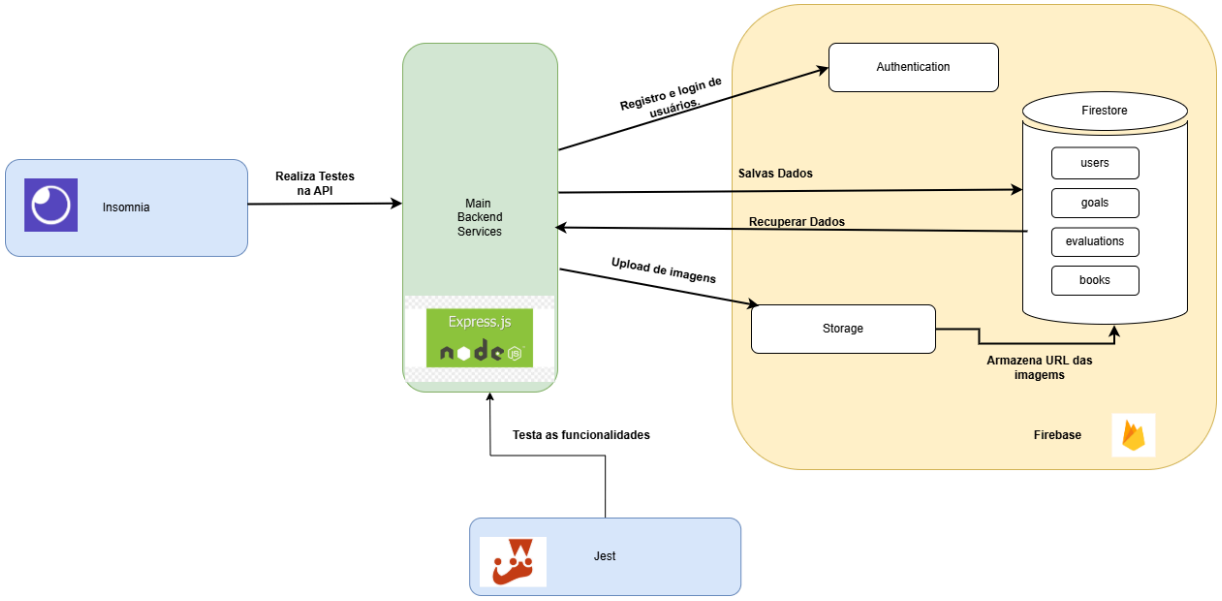
[3] Github. Disponível em: https://github.com/

tipos de requisições, como POST, GET, PUT e DELETE, foi essencial para assegurar o funcionamento correto da aplicação e corrigir possíveis erros antes da conclusão do trabalho.

Já o Jest foi utilizado para garantir a qualidade e funcionalidade do *backend* do trabalho, por se tratar de um *framework* robusto em *JavaScript* que simplifica o processo de criação de testes unitários e de integração. Ele possibilitou verificar o comportamento correto das funções e módulos de forma isolada, assegurando a qualidade e consistência do código. Suas funcionalidades de execução paralela e geração de relatórios de cobertura de código contribuem para um desenvolvimento mais confiável e eficiente, características essenciais para o bom desempenho e manutenção da aplicação.

A figura 6 apresenta um diagrama representando a arquitetura das ferramentas utilizadas no projeto.

Figura 6 - Visão Geral da Arquitetura do Sistema



Fonte: Elaborado pela Autora (2024)

5.2 Segundo ciclo

O segundo ciclo inclui o início da estruturação do *backend*, começando com a modelagem do banco de dados, definição das tabelas, atributos e relacionamentos, com base nos requisitos definidos no ciclo anterior.

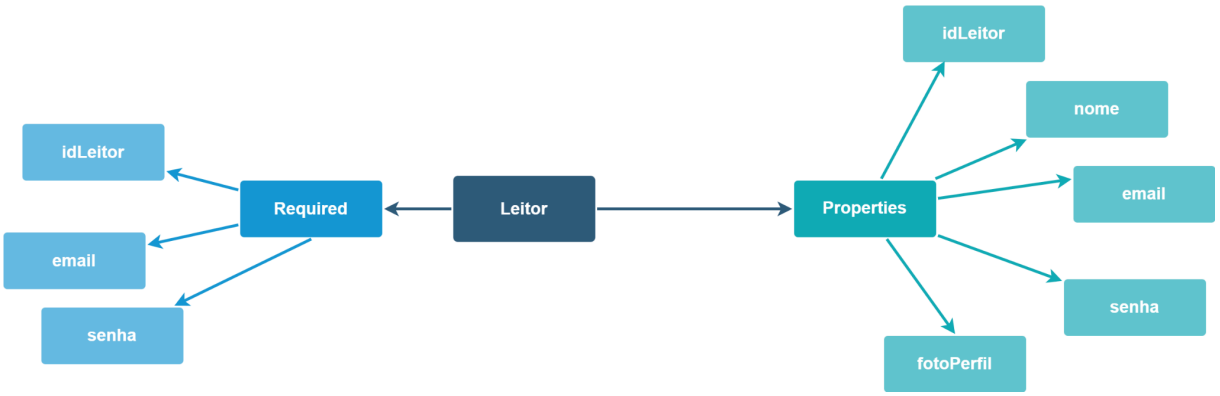
5.2.1 Banco de dados

Dado o escopo abrangente do sistema e a necessidade de armazenamento de informações, como *URLs* que redirecionam para arquivos e imagens externos, optou-se pelo Firebase *Firestore* devido a sua capacidade de fornecer atualizações em tempo real e sincronização de dados entre dispositivos. Essa escolha é essencial para o funcionamento eficiente da aplicação, garantindo eficiência e confiabilidade no armazenamento e recuperação de dados. O Firebase *Firestore* é um banco de dados *NoSQL* baseado em documentos, onde os dados são armazenados como *JSON* e sincronizados em tempo real.

As coleções e seus campos foram projetados para armazenar as informações necessárias para a aplicação e suas funcionalidades. Abaixo está uma visão geral das principais coleções (ou *nodes*) e seus respectivos documentos. Cada coleção e documento foi pensado para refletir diretamente as funcionalidades da aplicação, garantindo consistências e facilidade de acesso aos dados.

A figura 7 apresenta o modelo conceitual do objeto "Leitor" no sistema. Esse leitor terá um identificador único, além de nome, email e senha que deverão ser obrigatoriamente preenchidos pelo usuário no momento do cadastro, além da possibilidade de adicionar uma foto de perfil, caso queira.

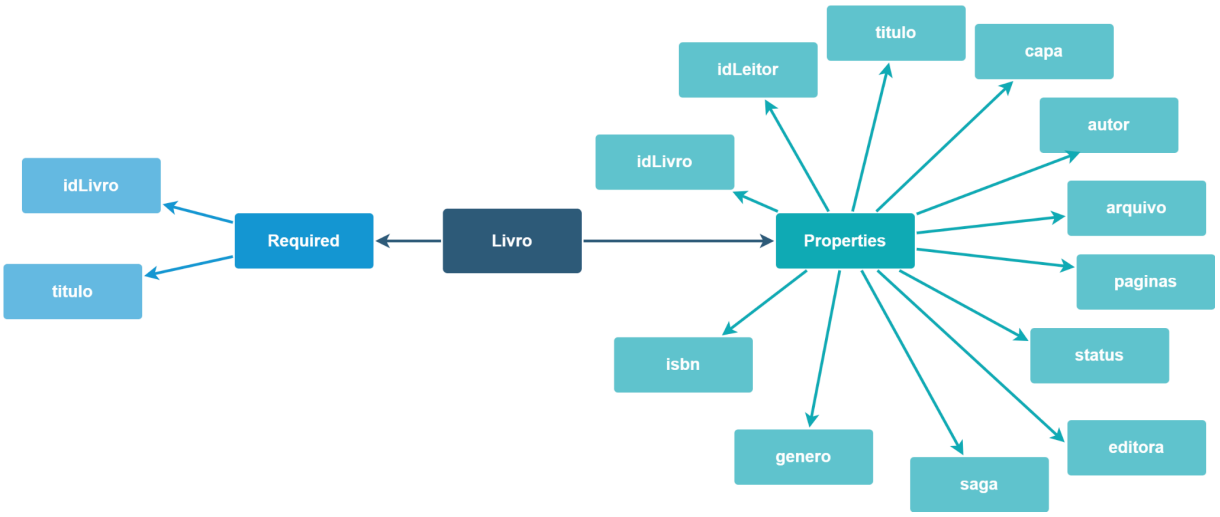
Figura 7 - Schema - Leitor



Fonte: Elaborado pela Autora (2024)

A figura 8 apresenta o modelo conceitual do objeto "Livro" no sistema. Além de um identificador único que será gerado pelo banco de dados, para o livro ser cadastrado é armazenado o título do livro, autor, páginas, editora, se faz parte de uma saga de livros, o ISBN e também uma imagem da capa do livro. É possível fazer o *upload* do arquivo do livro e definir um *status* para o livro, como por exemplo: ler, lido ou lendo.

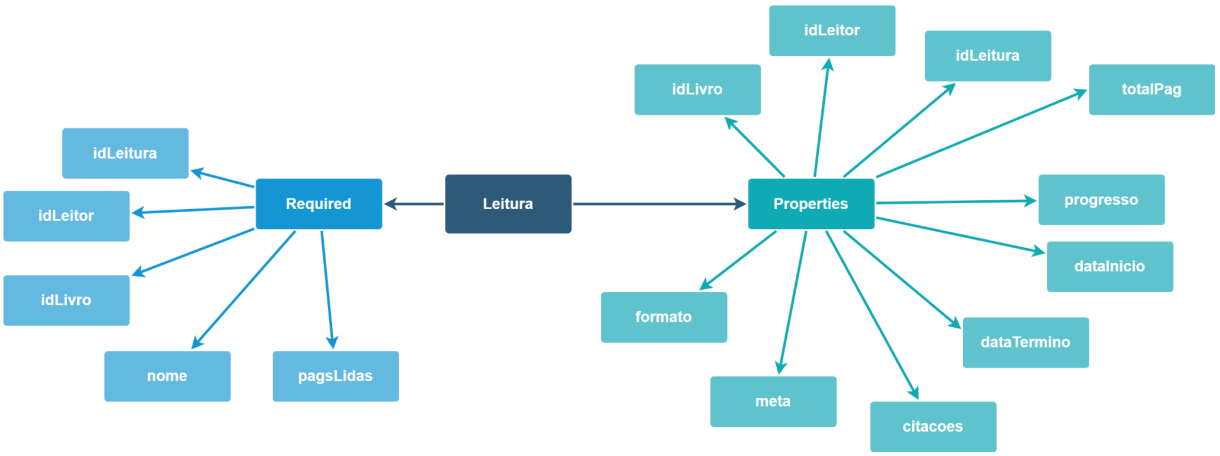
Figura 8 - Schema - Livro



Fonte: Elaborado pela Autora (2024)

A figura 9 apresenta o modelo conceitual do objeto "Leitura" no sistema. O identificador único da leitura será gerado pelo banco de dados e o único campo obrigatório para que ocorra a gravação do livro no banco de dados é o nome do livro, para que o banco possa consultar e gravar o *id* do livro, e a quantidade de páginas lidas, que poderá ser atualizada conforme ele prossiga com a leitura. O total de páginas do livro, a data de início e término da leitura, o formato do livro, se ele faz parte de alguma meta de leitura e adicionar o arquivo do livro. Progresso da leitura é atualizado conforme o usuário atualiza a quantidade de páginas lidas, utilizando elas no cálculo junto com o total de páginas do livro e retornando a porcentagem do progresso.

Figura 9 - Schema - Leitura



Fonte: Elaborado pela Autora (2024)

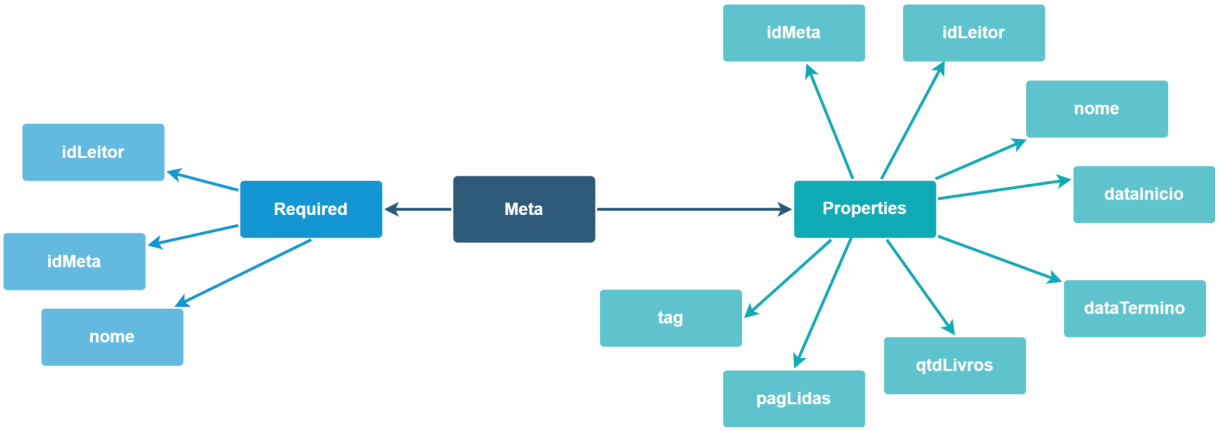
A figura 10 apresenta o modelo conceitual do objeto “Avaliação” no sistema. O banco de dados irá gerar um identificador único para cada avaliação, que também conterá o *id* da leitura que a avaliação estará relacionada. O usuário precisa atribuir uma nota ao livro lido para poder registrar a avaliação, e também poderá adicionar comentários avaliativos sobre o livro caso queira.

Figura 10 - Schema - Avaliação

Fonte: Elaborado pela Autora (2024)

A figura 11 apresenta o modelo conceitual do objeto “Meta” no sistema. Caso o usuário tenha interesse, ele poderá criar metas de leitura, cada meta terá seu identificador único gerado pelo banco de dados, possuirá o *id* do leitor para relacionar a cada meta ao seu respectivo leitor, cada meta obrigatoriamente tem que possuir um nome. Ela poderá ter também uma data de início e término, a quantidade de livros que o usuário quer que faça parte dessa meta, uma *tag* para a meta, para que ela possa ser adicionada às leituras do usuário e a quantidade de páginas lidas na meta.

Figura 11 - Schema - Meta



Fonte: Elaborado pela Autora (2024)

Para possibilitar que o usuário possa adicionar uma foto de perfil, capas para os livros e salvar os arquivos dos livros caso queira, será utilizado o Firebase *Storage* para esse armazenamento e uma referência a onde está salvo no *Storage* será armazenado no banco de dados, para que sempre que forem solicitadas essas informações serão carregadas para o usuário.

5.3 Terceiro ciclo

No terceiro ciclo, além da definição da arquitetura para a estruturação do *backend*, foi iniciado o desenvolvimento da aplicação, configurando o ambiente com as principais dependências e bibliotecas, como Node.js e *Express*, essenciais para a criação das rotas e a organização da aplicação.

5.3.1 Arquitetura do backend

A arquitetura do *backend* é baseada no padrão *MVC (Model-View-Controller)*, que promove uma separação clara entre a lógica de apresentação, a lógica de negócios e a manipulação de dados.

Neste trabalho, o *Model* é implementado para realizar operações específicas, como salvar e recuperar dados no *Firestore*, além de interagir com o Firebase *Storage* para gerenciar arquivos. Todas as interações com serviços externos do Firebase são abstraídas dentro do *Model*, garantindo que o *Controller* permaneça focado em processar requisições e enviar respostas ao cliente. O *Controller*, por sua vez, é responsável por:

- Receber as requisições HTTP;
- Validar as entradas recebidas (ex.: verificando campos obrigatórios ou tipos de dados);
- Invocar as funções adequadas no *Model* para processar os dados;
- Retornar respostas claras e bem estruturadas para o cliente, como mensagens de sucesso ou erros.

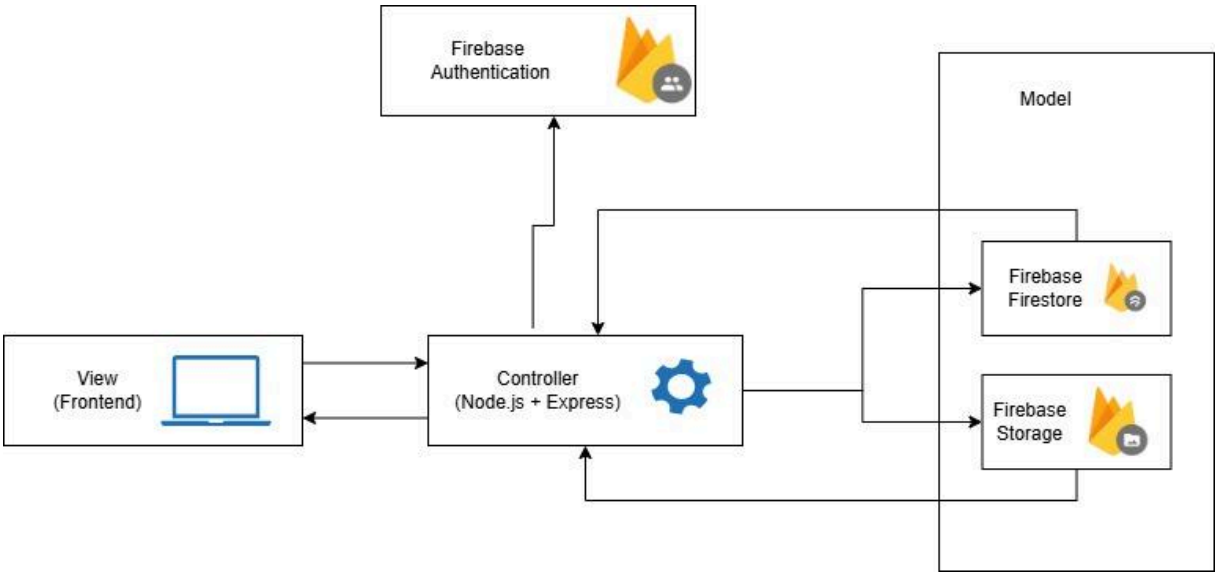
Embora a *View* geralmente represente a interface com o usuário, neste trabalho, ela não foi implementada diretamente, pois o foco está no *backend*. Ainda sim, a arquitetura é preparada para integrar uma *View* que faça uso das APIs desenvolvidas.

A figura 12, abaixo, ilustra essa arquitetura, destacando o fluxo de dados:

- **Cliente:** é representado pela *View,* envia uma requisição ao servidor (*Controller*).

- **C*ontroller*:** valida as entradas, aciona o *Model* e, caso necessário, interage com o *Firestore* ou o *Storage* para realizar a operação.

Figura 12 - Arquitetura MVC



Fonte: Elaborado pela Autora (2024)

Após processar os dados, o *Controller* envia a resposta para o cliente, informando o status da operação. Por exemplo, na funcionalidade de cadastro de livros, o fluxo ocorre da seguinte forma:

- O *Controller* recebe as informações enviadas pelo cliente, como título, autor e arquivos de imagem ou *e-book*.
- Ele valida os dados e delega ao *Model* a tarefa de armazenar as informações no *Firestore* e no Firebase *Storage*.
- O *Model* organiza as informações e interage com os serviços do Firebase, garantindo a integridade dos dados.
- O *Controller* retorna uma resposta confirmando a conclusão da operação ou informando erros, se houver.

Essa separação clara de responsabilidades entre *Model* e *Controller*, aliada à integração com os serviços do Firebase, oferece uma solução robusta e escalável para o gerenciamento de dados no *backend* da aplicação.

5.3.2 Desenvolvimento

Após a definição da arquitetura, deu-se início ao desenvolvimento do *backend* com a implementação das funcionalidades principais. Inicialmente, foi criado um CRUD para livros, estabelecendo uma estrutura básica e essencial para o gerenciamento completo dos registros de livros no sistema. Esse desenvolvimento seguiu o padrão arquitetural *MVC*, que proporciona uma separação clara entre as responsabilidades, garantindo organização e facilitando a manutenção do código.

No padrão *MVC*, o *Model* é responsável pelo armazenamento e manipulação dos dados dos livros, incluindo a interação com o *Firestore*, onde as informações são efetivamente armazenadas. O *Controller* gerencia as requisições dos clientes, processa a lógica de negócio e coordena as operações de manipulação de dados no *Model*. Já o *Router* define as rotas RESTful correspondentes a cada operação CRUD, associando cada rota a uma ação específica no Controller, o que garante que requisições HTTP (como GET, POST, PUT e DELETE) sejam tratadas de maneira organizada.

No Firebase *Storage*, foram criadas pastas específicas para armazenar as imagens de perfil dos usuários, as capas dos livros e, caso o usuário deseje, o arquivo do próprio livro. Para a integração com o banco de dados, foi configurada uma conexão com o *Firestore* para armazenar os dados dos livros. Algumas funções fornecidas pelo Firebase foram utilizadas para facilitar o envio dos dados ao Firebase *Storage* e ao *Firestore*, permitindo a manipulação e o armazenamento de dados de forma segura e eficiente.

5.4 Quarto ciclo

No quarto ciclo, foi implementado o sistema de autenticação, *login* e cadastro de usuários, utilizando o Firebase *Authentication*. Esse sistema atende aos requisitos da aplicação, permitindo que o usuário salve e edite uma foto de perfil, caso deseje. A imagem de perfil é armazenada no Firebase *Storage*, e uma *URL* de referência é gerada e salva no banco de dados.

Após a implementação da autenticação, foram realizadas adaptações nas funções relacionadas aos livros para que o *ID* do usuário seja armazenado junto aos dados do livro, permitindo que cada usuário visualize apenas os livros cadastrados por ele. Além disso, foi decidido unificar as funcionalidades de leitura e de gerenciamento dos livros em um único

banco de dados, o que possibilita centralizar todas as informações de leitura diretamente no banco de dados de livros, facilitando a organização e consulta dos dados pelo usuário.

Além disso, foram criadas pastas para armazenar as imagens e arquivos dos livros agrupados por usuário, garantindo que cada usuário tenha acesso apenas aos arquivos que ele próprio armazenou no banco de dados.

Também foram implementados testes automatizados para as funcionalidades de gerenciamento de livros, utilizando o *framework* Jest, com o objetivo de assegurar a qualidade e o funcionamento adequado do código. Além disso, foi configurada uma *pipeline* de testes no GitHub, utilizando o GitHub *Actions*. Dessa forma, a cada movimentação entre as *branches*, os testes são executados automaticamente para garantir que futuras implementações não impactem negativamente as funcionalidades existentes.

O quadro 4 mostra um caso de uso que foi utilizado nos testes para garantir que todas as funcionalidades dos livros estivessem sendo executadas como esperado, após as mudanças na estrutura do código. O teste inclui o cenário de cadastro e atualização de um livro no sistema, as entradas simulando o comportamento do usuário e o resultado esperado para verificar corretamente a operação de cada funcionalidade, sendo assim possível verificar tanto a integridade das funcionalidades quanto a aderência aos requisitos da aplicação.

Quadro 4 - Caso de Teste de Livros

Caso de Teste: CT1
Cenário: Cadastrar e Atualizar um livro
Entradas: ● Título do Livro: “O Senhor dos Anéis” ● Autor: “J.R.R. Tolkien” ● Gênero: “Fantasia” ● Arquivo *PDF* do Livro ● Imagem de capa do livro
Resultado Esperado: ● O livro é criado no banco de dados com todas as informações fornecidas, e seu *ID* é retornado; ● A imagem de capa e o arquivo *PDF* são enviados ao Firebase *Storage*, e os *URLs* correspondentes são armazenados no banco; ● Ao consultar o livro, todos os dados inseridos, incluindo o *link* para a capa e o arquivo, estão acessíveis.

Fonte: Elaborado pela Autora (2024)

5.5 QUINTO CICLO

No quinto ciclo, foram concluídas as funcionalidades previstas para o *backend*, finalizando as implementações de metas de leitura e avaliações de livros. A funcionalidade de metas permite que o usuário defina objetivos de leitura com prazos e monitoramento do progresso, enquanto a de avaliações permite ao usuário registrar e consultar suas avaliações sobre os livros lidos. Ambas as funcionalidades foram integradas ao banco de dados, possibilitando uma experiência de uso completa e personalizada para o usuário.

Além disso, foram criados e implementados testes automatizados em Jest para garantir a estabilidade das novas funcionalidades e verificar a integração correta com o sistema existente. Esses testes foram estruturados para assegurar a qualidade e prevenir regressões, garantindo que o sistema funcione conforme o esperado, mesmo com a adição das novas funcionalidades.

O quadro 5 apresenta um caso de uso aplicado nos testes para assegurar que todas as funcionalidades de metas funcionem conforme o esperado. O teste abrange o cenário de criação e atualização de uma meta no sistema, com entradas que simulam o comportamento do usuário e o resultado esperado para validar cada operação. Dessa forma, é possível verificar tanto a integridade das funcionalidades quanto a aderência aos requisitos da aplicação.

Quadro 5 - Caso de Teste de Meta de Leitura

Caso de Teste: CT2
Cenário: Criação e Atualização de Meta de Leitura
Entradas: ● Nome da Meta: “Meta de Leitura de Ficção” ● Total de livros: 5 ● Data de Início: Data atual ● Lista de Livros: *IDs* de cinco livros previamente cadastrados no sistema
Resultado Esperado: ● Meta é criada no banco de dados com todos os campos preenchidos corretamente; ● O progresso da meta aumenta conforme os livros associados são marcados como livros; ● Quando todos os livros são lidos, o *status* da meta atualiza para “Concluída” e o progresso atinge 100%;

Fonte: Elaborado pela Autora (2024)

Com as implementações técnicas finalizadas, o ciclo encerrou-se com a revisão e finalização do texto da monografia, detalhando todas as etapas de desenvolvimento e as decisões técnicas tomadas ao longo do projeto.

6 RESULTADOS

A aplicação desenvolvida conforme as etapas descritas na metodologia, resultou em um *backend* funcional que atende os requisitos propostos. Neste capítulo, são apresentados os resultados obtidos ao final das atividades, evidenciando a eficácia e as funcionalidades implementadas ao longo do desenvolvimento.

A implementação das funcionalidades principais, como o CRUD de livros, metas de leitura e avaliações, possibilita uma experiência de uso completa e funcional, permitindo que cada usuário gerencie seu progresso de leitura de forma independente e personalizada. O sistema de autenticação, aliado ao armazenamento seguro de arquivos no Firebase *Storage*, garantiu que cada usuário tivesse acesso apenas aos seus próprios dados e arquivos, atendendo às diretrizes de segurança e privacidade.

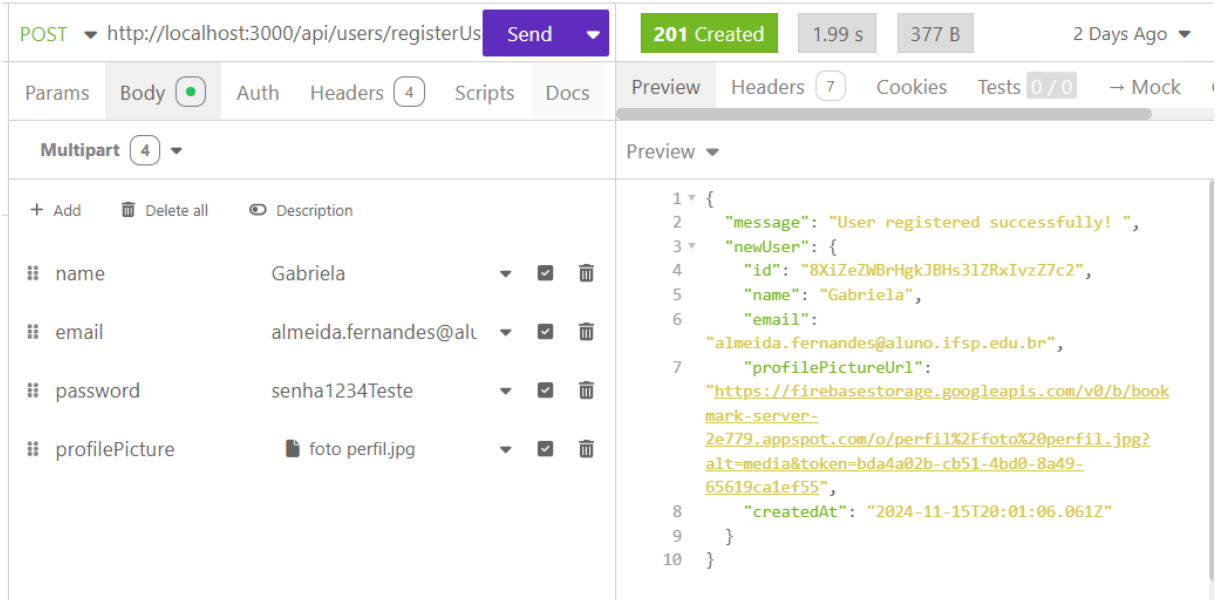
Para assegurar a funcionalidade do *backend* e a correta comunicação entre as rotas e o banco de dados, foram realizados testes com o Insomnia, uma aplicação para testes de APIs, especialmente usada para desenvolver e testar requisições HTTP e REST. Esse processo permitiu validar que as operações essenciais, como criação, atualização, recuperação e exclusão de registros, funcionam conforme o esperado, garantindo que o sistema esteja preparado para lidar com os dados que forem registrados através da aplicação.

Abaixo, são apresentados exemplos de testes para algumas das funcionalidades da aplicação.

6.1 Testes para a funcionalidade de usuários

Os testes para a funcionalidade de usuários abordaram o cadastro, autenticação, edição e exclusão da conta do usuário. A figura 13 abaixo apresenta um teste de cadastro de usuário com foto de perfil, onde o sistema retorna a URL da imagem armazenada e as informações de usuário correspondentes.

Figura 13 - Teste Cadastro de Usuário Insomnia

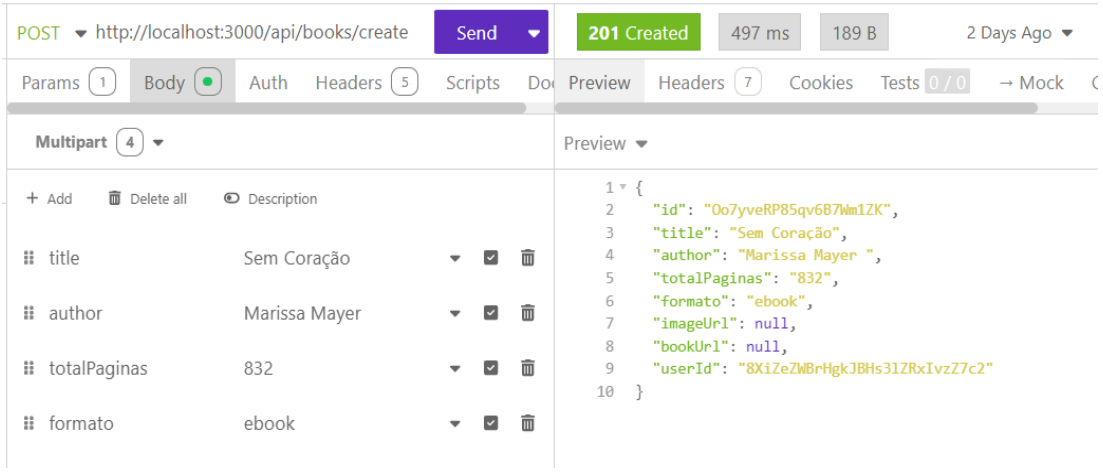


Fonte: Elaborado pela Autora (2024)

6.2 Testes para a funcionalidade de livros

As funcionalidades de gerenciamento de livros incluem a criação, atualização, exclusão e visualização de registros. A figura 14 abaixo, exemplifica a criação de um livro, validando a resposta com o código de *status* esperado e os dados do livro inserido.

Figura 14 - Teste Cadastro de Livro Insomnia

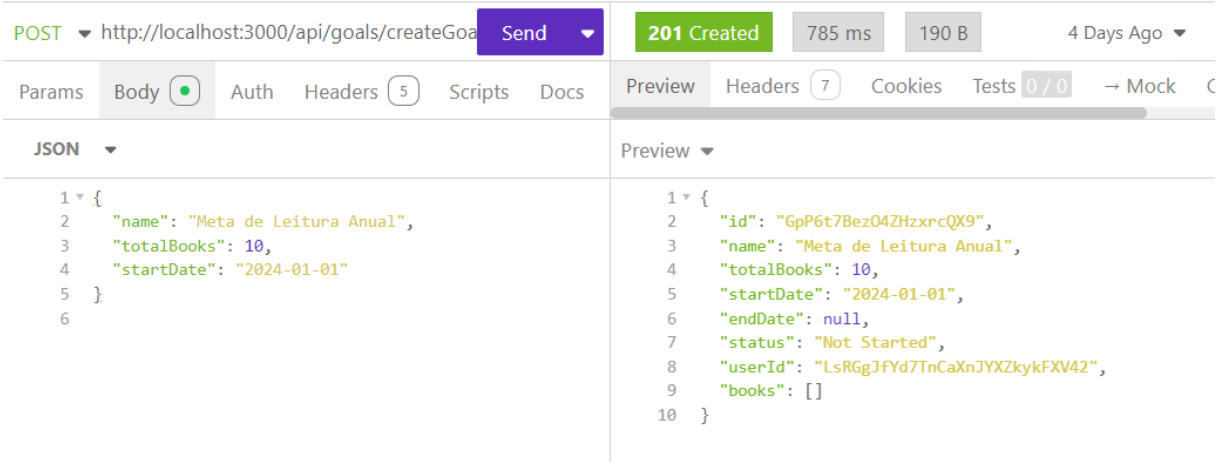


Fonte: Elaborado pela Autora (2024)

6.3 Testes para a Funcionalidade de Metas

As operações de criação, atualização e visualização de metas foram verificadas para garantir que o sistema respondesse conforme o esperado. Na figura 15 abaixo, é possível observar um teste de criação de uma nova meta, onde a aplicação retorna a resposta com os dados inseridos.

Figura 15 - Teste Cadastro de Meta Insomnia



Fonte: Elaborado pela Autora (2024)

Além disso, os testes unitários com Jest e combinados com a integração com o Github *Actions*, estabeleceram uma base sólida para o controle de qualidade do código. Os testes permitiram validar a execução correta de cada funcionalidade, enquanto a execução automatizada nas movimentações entre as *branches* assegurou que o código permanecesse funcional e consistente em todas as etapas do desenvolvimento.

Esses resultados obtidos demonstram que o sistema atende aos objetivos iniciais, que envolviam a compreensão do contexto e dos hábitos de leitura dos usuários, a seleção das tecnologias mais adequadas, e a implementação de funcionalidades essenciais para o gerenciamento de leituras. O *backend* foi desenvolvido com uma estrutura flexível e organizada, incluindo um banco de dados eficiente que suporta o cadastro, edição e exclusão de livros, o estabelecimento de metas de leitura, e o acompanhamento do progresso do usuário, bem como um sistema de autenticação seguro para proteger a privacidade dos dados.

Essa base sólida também permite o crescimento futuro, possibilitando a adição de novas funcionalidades ou a integração com uma interface *frontend*, ampliando a praticidade e o controle proporcionado ao usuário e agregando valor à experiência de leitura. E a documentação do desenvolvimento reflete as decisões tomadas, proporcionando um suporte valioso para futuras expansões.

7 CONCLUSÃO

A aplicação desenvolvida para o gerenciamento de leituras se propôs a fornecer aos usuários uma plataforma para organizar, acompanhar e avaliar suas leituras, com funcionalidades que facilitam o registro de livros, a criação de metas e a realização de avaliações pessoais. Para alcançar esses objetivos, foi desenvolvido um *backend* robusto, estruturado no padrão *MVC* e integrado aos serviços do Firebase para autenticação, armazenamento de dados e gerenciamento de arquivos.

A partir de um levantamento das principais necessidades dos usuários e de funcionalidades essenciais, foram definidas e implementadas funções como o CRUD de livros, o sistema de autenticação para usuários e o armazenamento de arquivos, além de metas de leitura e avaliações, garantindo uma experiência completa para o usuário. A utilização de testes unitários com Jest e a integração com GitHub *Actions* permitiram a verificação contínua do código, mantendo a qualidade e funcionalidade do sistema durante o desenvolvimento.

Ao disponibilizar recursos que possibilitam o acompanhamento detalhado de leituras e o gerenciamento de metas, o sistema oferece ao usuário uma ferramenta para controle e registro do seu processo, promovendo organização e incentivando o hábito da leitura. A estrutura desenvolvida, com segurança e organização de dados, oferece uma base sólida para futuras expansões, incluindo a integração com uma interface *frontend*.

Com a conclusão das funcionalidades e testes, o trabalho atingiu seus objetivos fornecendo aos usuários um sistema que facilita o acompanhamento de suas leituras e proporciona uma experiência eficiente e personalizada. Além disso, o projeto permitiu que a autora aprofundasse seus conhecimentos em desenvolvimento *backend*, aprimorando habilidades técnicas em Node.js, *Express* e Firebase, aplicando práticas de testes e automação para controle de qualidade.

REFERÊNCIAS

BANGARE, S. L. *et al*. Using Node.Js to Build High Speed and Scalable Backend Database Server. **International Journal of Research in Advent Technology**, 2016. 4 p. Disponível em: https://www.researchgate.net/profile/Sunil-Bangare-2/publication/301788361_Using_NodeJs_to_Build_High_Speed_and_Scalable_Backend_Database_Server/links/57285d6c08aee491cb416ad6/Using-NodeJs-to-Build-High-Speed-and-Scalable-Backend-Database-Server.pdf . Acesso em: 10 nov. 2024.

COODESH. **O que é arquitetura MVC?**. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-arquitetura-mvc/ . Acesso em: 23 nov. 2024a.

COODESH. **O que é Jest?**. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-jest/ . Acesso em: 09 nov. 2024b.

COODESH. **O que são testes unitários.** Disponível em: https://coodesh.com/blog/dicionario/o-que-sao-testes-unitarios>. Acesso em: 24 mar. 2024c.

DEVMEDIA. **Introdução ao Padrão MVC: Primeiros passos na Arquitetura MVC**. Disponível em: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308 . Acesso em: 23 nov. 2024a.

DEVMEDIA. **TDD: fundamentos do desenvolvimento orientado a testes**. 2013. Disponível em: https://www.devmedia.com.br/tdd-fundamentos-do-desenvolvimento-orientado-a-testes/28151. Acesso em: 24 mar. 2024b.

DIAS, Ricardo. **O Modelo Incremental**. MEDIUM, 2019. Disponível em: https://medium.com/contexto-delimitado/o-modelo-incremental-b41fc06cac04. Acesso em: 23 mar. 2024.

FAILLA, Zoara. **Influência das redes sociais nos hábitos de leitura aumentou, diz pesquisa.** CNN Brasil, São Paulo, 2022. [Entrevista concedida a] Amanda Garcia. 2022. Disponível em: https://www.cnnbrasil.com.br/nacional/influencia-das-redes-sociais-nos-habitos-de-leitura-aumentou-diz-pesquisa/ . Acesso em: 21 mar. 2024.

GITHUB. **Sobre o GitHub e ao Git**. Disponível em: https://docs.github.com/pt/get-started/start-your-journey/about-github-and-git. Acesso em: 26 mar. 2024.

GOODREADS. Disponível em: https://www.goodreads.com/ . Acesso em: 23 mar. 2024

KRUG, Flávia S. A Importância da Leitura na Formação do Leitor. **Revista de Educação do Ideau**, Uruguai, v.10, n. 22. Disponível em: https://www.caxias.ideau.com.br/wp-content/files_mf/d4ec50fa8dff16815b9bf525976d2b5c277_1.pdf . Acesso em: 21 mar. 2024.

MARATONA.APP. Disponível em: https://maratona.app/ . Acesso em: 23 mar. de 2024.

MEDIUM. **O Modelo Incremental**. Disponível em: https://medium.com/contexto-delimitado/o-modelo-incremental-b41fc06cac04 . Acesso em: 14 nov. 2024.

MICROSOFT LEARN. **Dados Não Relacionais e NoSQL**. Disponível em: https://learn.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data. Acesso em: 24 mar. 2024.

MOZILLA DEVELOPER NETWORK.. **Introdução Express/Node**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introduction#veja_tamb%C3%A9m . Acesso em: 11 nov. 2024.

PESSÔA, Camila. **Node.JS: Definição, Características, Vantagens e Usos Possíveis**. Alura, 2022. Disponível em https://www.alura.com.br/artigos/node-js-definicao-caracteristicas-vantagens-usos. Acesso em: 23 mar. 2024.

PRESSMAN, R. S.; MAXIM, B. Modelos de processo. In: PRESSMAN, R. S.; MAXIM, B. **Engenharia de software: uma abordagem profissional**. 9. ed. Porto Alegre: AMGH, 2021. p.154-155.

RED HAT. **O que é uma API REST?**. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 23 mar. 2024.

RIBEIRO. A. L. S. **O que é Firebase? Para que serve, principais características e um guia dessa ferramenta Google**. Alura, 2023. Disponível em: https://www.alura.com.br/artigos/firebase. Acesso em: 25 mar. 2024.

SKOOB. Disponível em: https://www.skoob.com.br/ . Acesso em: 23 mar. 2024.

SUDIARTHA, I K G *et al*. Data Structure Comparison Between MySql Relational Database and Firebase Database NoSql on Mobile Based Tourist Tracking Application. *In:* **International Conference on Science and Technology.** 2019. Journal of Physics: Conference Series, 2020. 8 p. Disponível em: https://iopscience.iop.org/article/10.1088/1742-6596/1569/3/032092/pdf. Acesso em: 26 ago. 2024.

SYDLE. **Extreme programming: o que é e como funciona?.** Disponível em: https://www.sydle.com/br/blog/extreme-programming-602ee205da4d096809438c9c . Acesso em: 08 dez. 2024.