# Requisitos do Sistema

## Requisitos Funcionais (User Stories)

Os requisitos funcionais foram definidos utilizando o formato de User Story, focando no valor entregue ao usuário final. A tabela abaixo apresenta as principais histórias de usuário identificadas para o projeto "Leitura Descomplicada":

|  |  |  |  |
| --- | --- | --- | --- |
| ID | User Story | Prioridade | Status |
| RF01 | Como um novo usuário, eu quero me cadastrar na plataforma fornecendo nome, email e senha para que possa acessar as funcionalidades personalizadas. | Alta | Concluído |
| RF02 | Como um usuário cadastrado, eu quero fazer login utilizando meu email e senha para que possa acessar meu dashboard e recomendações. | Alta | Concluído |
| RF03 | Como um usuário logado, eu quero responder a um quiz sobre minhas preferências de leitura (gêneros, temas) para que possa receber recomendações de livros personalizadas. | Alta | Concluído |
| RF04 | Como um usuário logado, eu quero visualizar um carrossel com as capas e títulos dos livros recomendados com base nas minhas respostas do quiz para que possa descobrir novas leituras. | Alta | Concluído |
| RF05 | Como um usuário logado, eu quero acessar um dashboard que mostre meus dados (nome) e talvez estatísticas básicas de leitura ou status do quiz para que possa acompanhar meu perfil. | Média | Concluído |
| RF06 | Como um visitante ou usuário logado, eu quero enviar um formulário de feedback com título e mensagem para que possa dar sugestões ou reportar problemas. | Média | Concluído |
| RF07 |  | Alta | Concluído |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | User Story | Prioridade | Status |
|  | Como um usuário logado, eu quero que minhas respostas do quiz sejam salvas no banco de dados para que minhas preferências sejam lembradas. |  |  |
| RF08 | Como um usuário logado, eu quero que meu gênero favorito (informado no cadastro ou quiz) seja registrado para futuras personalizações. | Média | Concluído |

## Requisitos Não Funcionais

Além das funcionalidades, o sistema deve atender aos seguintes requisitos de qualidade:

|  |  |
| --- | --- |
| Categoria | Requisito |
| Usabilidade | A interface deve ser intuitiva e fácil de navegar, especialmente para o público jovem acostumado a apps. |
|  | O fluxo de cadastro, login e resposta ao quiz deve ser simples e direto. |
| Desempenho | O tempo de carregamento das páginas principais (Home, Dashboard, Quiz) deve ser considerado rápido (idealmente < 3 segundos). |
|  | As consultas ao banco de dados para login e exibição de dados no dashboard devem ser eficientes. |
| Segurança | As senhas dos usuários devem ser armazenadas de forma segura no banco de dados (ex: hash). |
|  | Validações básicas devem ser implementadas tanto no frontend quanto no backend para evitar dados inválidos. |
| Confiabilidade | O sistema deve estar disponível durante o período de avaliação. |
|  | As funcionalidades principais (login, cadastro, quiz, dashboard) devem operar sem erros críticos. |
| Manutenibilidade | O código deve seguir uma estrutura organizada (como a divisão em routes , controllers , models ). |

|  |  |
| --- | --- |
| Categoria | Requisito |
|  | Comentários devem ser utilizados no código para explicar partes complexas ou lógicas importantes. |

## Casos de Uso Principais

Os casos de uso descrevem as interações mais importantes entre o usuário e o sistema.

[INSERIR DIAGRAMA: Diagrama de Caso de Uso (UML). Deve mostrar um ator "Usuário" (ou "Visitante" e "Usuário Logado") interagindo com os casos de uso: "Cadastrar-se", "Fazer Login", "Realizar Quiz", "Ver Recomendações", "Acessar Dashboard", "Enviar Feedback".]

Descrição Textual Simplificada:

* + - UC01 - Cadastrar Usuário: Um visitante acessa a página de cadastro, preenche nome, email e senha, e submete o formulário. O sistema valida os dados e cria um novo registro de usuário no banco de dados.
    - UC02 - Autenticar Usuário: Um usuário cadastrado acessa a página de login,

informa email e senha. O sistema verifica as credenciais no banco de dados e, se válidas, redireciona o usuário para o dashboard.

* + - UC03 - Realizar Quiz: Um usuário logado acessa a página do quiz, responde às

perguntas sobre preferências e submete as respostas. O sistema salva as respostas associadas ao usuário no banco de dados.

* + - UC04 - Visualizar Dashboard: Um usuário logado acessa sua área pessoal

(dashboard). O sistema busca os dados do usuário (e talvez dados do quiz/leitura) no banco de dados e os exibe na tela.

* + - UC05 - Enviar Feedback: Um visitante ou usuário logado acessa a página de

contato/feedback, preenche título e mensagem, e envia. O sistema salva o feedback no banco de dados.

# Arquitetura do Sistema

## Visão Geral da Arquitetura

O projeto "Leitura Descomplicada" foi desenvolvido seguindo uma arquitetura de aplicação web monolítica, composta por três camadas principais:

1. Frontend (Cliente): Interface com o usuário, construída com HTML, CSS e JavaScript puro, executada no navegador do usuário. Responsável por exibir as informações e capturar as interações do usuário.
2. Backend (Servidor/API): Construído com Node.js e o framework Express.js.

Responsável por receber as requisições do frontend, aplicar as regras de negócio, interagir com o banco de dados e retornar as respostas para o cliente. Segue um padrão próximo ao MVC (Model-View-Controller), com separação de responsabilidades em rotas, controladores e modelos.

1. Banco de Dados (Persistência): Utiliza o MySQL como sistema de gerenciamento

de banco de dados relacional (SGBDR) para armazenar todos os dados da aplicação (usuários, quizzes, formulários, livros, etc.).

[INSERIR DIAGRAMA: Diagrama de Arquitetura. ESSENCIAL! Deve mostrar visualmente as camadas: Navegador (HTML/CSS/JS) -> Servidor Node.js/Express (app.js, routes, controllers, models) -> Banco de Dados MySQL. Indicar os protocolos de comunicação (HTTP entre cliente e servidor, SQL entre servidor e BD). Se o BD roda em VM, indicar isso também.]

## Tecnologias Utilizadas

A tabela abaixo detalha as principais tecnologias empregadas em cada camada do sistema:

|  |  |  |  |
| --- | --- | --- | --- |
| Camada | Tecnologia | Versão (Aprox.) | Justificativa Técnica |
| Frontend | HTML5 | - | Linguagem de marcação padrão para estruturação de páginas web. |
| Frontend | CSS3 | - | Linguagem para estilização visual das páginas web. |
| Frontend | JavaScript (ES6+) | - | Linguagem para adicionar interatividade e dinamismo no lado do cliente (navegador). |

|  |  |  |  |
| --- | --- | --- | --- |
| Camada | Tecnologia | Versão (Aprox.) | Justificativa Técnica |
| Backend | Node.js |  | Ambiente de execução JavaScript no servidor, permitindo usar JS no backend. Requisito. |
| Backend | Express.js |  | Framework minimalista e flexível para Node.js, facilitando a criação de APIs web. |
| Backend | mysql2 |  | Driver Node.js para conexão e interação com o banco de dados MySQL. |
| Backend | dotenv |  | Módulo para carregar variáveis de ambiente de arquivos .env . |
| Banco de Dados | MySQL |  | SGBDR robusto, popular e requisito do curso para armazenamento de dados relacionais. |
| Ambiente BD | Linux (VM) |  | Ambiente simulado para hospedagem do banco de dados, conforme requisito. |
| Virtualização | VirtualBox |  | Software para criação e gerenciamento da Máquina Virtual Linux. |
| Controle Versão | Git | - | Sistema de controle de versão distribuído padrão. |
| Repositório | GitHub | - | Plataforma de hospedagem para repositórios Git. |
| Gerenc. Tarefas | Trello | - | Ferramenta visual para gerenciamento de tarefas no estilo Kanban. |

## Ambiente de Desenvolvimento e Produção

* + - Ambiente de Desenvolvimento: O desenvolvimento foi realizado localmente na máquina do desenvolvedor, utilizando VSCode como editor de código, Git para controle de versão, Node.js/Express rodando localmente e conectando-se ao MySQL também local ou na VM configurada no VirtualBox. Arquivos

.env.dev

foram usados para configurar variáveis de ambiente locais.

* + - Ambiente de Produção (Simulado): Para fins de projeto e demonstração, o ambiente de "produção" consiste na aplicação Node.js rodando localmente, mas conectando-se ao banco de dados MySQL hospedado na Máquina Virtual Linux configurada no VirtualBox. Variáveis de ambiente para produção podem ser

gerenciadas no arquivo .

.env

## Integração com VM Linux

A API Node.js, executada na máquina host do desenvolvedor, estabelece conexão com o servidor MySQL que está rodando dentro da Máquina Virtual Linux. Essa conexão é feita através da rede configurada no VirtualBox (geralmente NAT com redirecionamento de portas ou Rede Interna/Host-Only com IP fixo para a VM). As credenciais de acesso ao banco de dados (usuário, senha, nome do banco, host/IP da VM, porta do MySQL) são

.env

gerenciadas de forma segura através de variáveis de ambiente (arquivos e

.env.dev ), não sendo expostas diretamente no código-fonte.

# Modelagem de Dados

## Modelo Conceitual

O modelo conceitual representa as principais entidades de informação do sistema e como elas se relacionam de forma abstrata. Para o "Leitura Descomplicada", as entidades centrais são:

* + - Usuário: Representa a pessoa que utiliza a plataforma, com seus dados de cadastro e preferências.
    - Livro: Representa uma obra literária com título, autor, gênero, etc. (Usado para recomendações).
    - Questionario: Armazena as respostas do quiz de preferências de um usuário.
    - Formulario: Guarda as mensagens de feedback enviadas pelos usuários.

Relacionamentos Principais: \* Um Usuário PODE responder a UM Questionario. \* Um Usuário PODE enviar VÁRIOS Formularios de feedback. \* Um Questionario PERTENCE a UM Usuário. \* Um Formulario PERTENCE a UM Usuário. \* (Implicitamente) Livros são recomendados a Usuários com base nas respostas do Questionario.

[INSERIR DIAGRAMA: Diagrama Conceitual Simplificado. Caixas para Usuário, Livro, Questionario, Formulario e linhas indicando os relacionamentos descritos acima.]

## Modelo Lógico

O modelo lógico traduz o modelo conceitual em uma estrutura de tabelas relacionais, definindo colunas, tipos de dados e chaves.

[INSERIR DIAGRAMA: Diagrama Entidade-Relacionamento (DER) Lógico. ESSENCIAL! Use uma ferramenta como MySQL Workbench ou draw.io para criar o DER mostrando as

livro

tabelas usuario , formulario , questionario , com suas colunas, tipos de dados,

PKs (chaves primárias) e FKs (chaves estrangeiras) e a cardinalidade dos relacionamentos (1:1, 1:N). Exporte como imagem.]

## Dicionário de Dados

A tabela abaixo detalha cada campo das tabelas do banco de dados:

Tabela:

usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coluna | Tipo | PK | FK | Descrição |
| id | INT | Sim | Não | Identificador único do usuário (auto- increment) |
| nome | VARCHAR(50) | Não | Não | Nome do usuário |
| email | VARCHAR(100) | Não | Não | Email do usuário (usado para login) |
| senha | VARCHAR(40) | Não | Não | Senha do usuário () |
| generoFavorito | VARCHAR(20) | Não | Não | Gênero literário preferido do usuário |

Tabela:

formulario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coluna | Tipo | PK | FK | Descrição |
| id | INT | Sim | Não | Identificador único do formulário (auto-increment) |
| fkUsuario | INT | Sim | Sim | Chave estrangeira referenciando  usuario(id) |
| classeFormulario | VARCHAR(20) | Não | Não | Tipo/Categoria do formulário (ex: 'Feedback', 'Sugestão') |
| titulo | VARCHAR(50) | Não | Não | Título da mensagem de feedback |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coluna | Tipo | PK | FK | Descrição |
| mensagem | VARCHAR(1000) | Não | Não | Conteúdo da mensagem de feedback |

Tabela:

questionario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coluna | Tipo | PK | FK | Descrição |
| id | INT | Sim | Não | Identificador único do questionário (auto- increment) |
| fkUsuario | INT | Sim | Sim | Chave estrangeira referenciando usuario(id) |
| genero1 | VARCHAR(30) | Não | Não | Primeira preferência de gênero do usuário |
| genero2 | VARCHAR(30) | Não | Não | Segunda preferência de gênero do usuário |

Tabela:

livro

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coluna | Tipo | PK | FK | Descrição |
| idLivro | INT | Sim | Não | Identificador único do livro (auto-increment) |
| nome | VARCHAR(70) | Não | Não | Título do livro |
| autor | VARCHAR(70) | Não | Não | Autor do livro |
| genero | VARCHAR(45) | Não | Não | Gênero principal do livro |
| qtdPag | INT | Não | Não | Quantidade de páginas do livro |
| urlCapa | VARCHAR(300) | Não | Não | URL da imagem da capa do livro |

## Scripts SQL (Criação)

Os scripts SQL utilizados para criar a estrutura do banco de dados são apresentados abaixo:

CREATE DATABASE leituraDescomplicada; USE leituraDescomplicada;

CREATE TABLE usuario(

id INT PRIMARY KEY AUTO\_INCREMENT, nome VARCHAR(50),

email VARCHAR(100), senha VARCHAR(40),

generoFavorito VARCHAR(20)

);

CREATE TABLE formulario( id INT AUTO\_INCREMENT,

fkUsuario INT, classeFormulario VARCHAR(20), titulo VARCHAR(50),

mensagem VARCHAR(1000),

CONSTRAINT fk\_usuario\_formulario FOREIGN KEY (fkUsuario) REFERENCES

usuario(id),

PRIMARY KEY (id, fkUsuario)

);

CREATE TABLE questionario( id INT AUTO\_INCREMENT,

fkUsuario INT, genero1 VARCHAR(30), genero2 VARCHAR(30),

CONSTRAINT fk\_usuario\_questionario FOREIGN KEY (fkUsuario) REFERENCES

usuario(id),

PRIMARY KEY (id, fkUsuario)

);

CREATE TABLE livro (

idLivro INT PRIMARY KEY AUTO\_INCREMENT, nome VARCHAR(70),

autor VARCHAR(70),

genero VARCHAR(45), qtdPag INT,

urlCapa VARCHAR(300)

);

*-- Scripts de INSERT (Exemplo para popular a tabela livro)*

INSERT INTO livro (nome, autor, genero, qtdPag, urlCapa) VALUES ('O Pequeno Príncipe', 'Antoine de Saint-Exupéry', 'Fábula', 96, 'https:// exemplo.com/capa\_pequeno\_principe.jpg'),

('Dom Casmurro', 'Machado de Assis', 'Realismo', 256, 'https://exemplo.com/ capa\_dom\_casmurro.jpg');

*-- Adicionar mais INSERTs conforme necessário*

# Implementação Frontend

## Estrutura de Arquivos

Os arquivos do frontend estão organizados dentro da pasta

no diretório

Site -

API/web-data-viz . A estrutura segue um padrão comum para aplicações web estáticas:

public

public/

├── css/ # Arquivos de estilo CSS

│ ├── style.css

│ └── ... (outros arquivos CSS específicos de páginas)

├── js/ # Arquivos JavaScript

│ ├── script.js

│ └── ... (outros arquivos JS específicos)

├── assets/ # Recursos como fontes, etc.

│ └── fonts/

├── img/ # Imagens utilizadas na interface

│ ├── logo.png

│ └── ...

├── index.html # Página inicial (Home)

├── login.html # Página de Login

├── cadastro.html # Página de Cadastro

├── quiz.html # Página do Quiz

├── dashboard.html # Página do Dashboard do Usuário

├── formulario.html # Página do Formulário de Feedback

└── ... (outras páginas HTML, se houver)

[INSERIR IMAGEM: Opcional: Screenshot da estrutura de pastas do diretório no VSCode.]

public

## Design System (Guia de Estilo Simplificado)

Embora um Design System formal não tenha sido o foco, alguns elementos visuais foram padronizados para manter a consistência:

* + - Paleta de Cores:
      * Cor Primária: [Cor Principal]
      * Cor Secundária: [Cor Secundária]
      * Cor de Destaque/Ação: [Cor de Ação]
      * Cores Neutras (Texto, Fundo): [Cores Neutras] [INSERIR IMAGEM: Pequenos quadrados com as cores principais da paleta e seus códigos HEX.]
    - Tipografia:
      * Fonte Principal (Títulos): [Fonte Títulos]
      * Fonte Secundária (Corpo de Texto): [Fonte Texto] [INSERIR TEXTO: Exemplos curtos de texto usando as fontes definidas.]
    - Componentes:
      * Botões:
      * Cards (Livros):
      * Formulários: [INSERIR IMAGEM: Screenshots de exemplos dos principais componentes reutilizados (botões, cards, inputs).]

## Telas Principais

A seguir, uma visão geral das telas principais da aplicação:

* + - Tela Inicial ( index.html ): Apresentação do projeto, chamadas para ação (Login/ Cadastro). [INSERIR SCREENSHOT: Tela Inicial ( index.html )]
    - Tela de Login ( login.html ): Formulário para autenticação do usuário. [INSERIR SCREENSHOT: Tela de Login ( login.html )]
    - Tela de Cadastro ( cadastro.html ): Formulário para registro de novos usuários. [INSERIR SCREENSHOT: Tela de Cadastro ( cadastro.html )]
    - Tela do Quiz ( quiz.html ): Perguntas interativas sobre preferências de leitura. [INSERIR SCREENSHOT: Tela do Quiz ( quiz.html )]
    - Tela do Dashboard ( dashboard.html ): Área pessoal do usuário logado, exibindo informações e recomendações (carrossel). [INSERIR SCREENSHOT: Tela do Dashboard ( dashboard.html )]
    - Tela do Formulário ( formulario.html ): Interface para envio de feedback.

[INSERIR SCREENSHOT: Tela do Formulário ( formulario.html )]

# Implementação Backend (JavaScript/Node.js)

## Estrutura da API

O backend, desenvolvido em Node.js com Express, segue uma estrutura organizada para separar responsabilidades, localizada em Site - API/web-data-viz/src :

* + - app.js : Arquivo principal que inicializa o servidor Express, configura middlewares (como CORS, urlencoded) e define as rotas principais.
    - src/routes/ : Contém os arquivos que definem os endpoints para cada recurso (usuários, questionários, formulários, etc.). Cada arquivo de rota utiliza um

express.Router() .

* + - src/controllers/ : Contém a lógica de negócio para cada rota. Os controladores recebem as requisições das rotas, interagem com os modelos (para acesso ao banco de dados) e enviam as respostas.

src/models/

* + - : Responsável pela interação com o banco de dados. Contém as

funções que executam as queries SQL (SELECT, INSERT, UPDATE, DELETE)

utilizando o driver .

mysql2

src/database/

(arquivo

config.js

: Contém a configuração da conexão com o banco de dados que lê variáveis de ambiente).

[INSERIR DIAGRAMA: Opcional: Diagrama simples mostrando o fluxo de uma requisição: Rota -> Controller -> Model -> Banco de Dados e o retorno.] [INSERIR IMAGEM: Opcional:

src

Screenshot da estrutura de pastas do diretório no VSCode.]

## Endpoints da API

A tabela abaixo lista os principais endpoints implementados na API:

|  |  |  |  |
| --- | --- | --- | --- |
| Método | URL | Controller | Descrição |
| POST | /usuarios/cadastrar | usuarioController | Registra um novo usuário. |
| POST | /usuarios/autenticar | usuarioController | Autentica um usuário existente (login). |
| GET | /usuarios/:idUsuario | usuarioController |  |
| POST | /questionarios/cadastrar | questionarioController | Salva as respostas do quiz de um usuário. |
| GET | /questionarios/:idUsuario | questionarioController |  |
| POST | /formularios/cadastrar | formularioController | Salva uma mensagem de feedback. |
| GET | /livros | livroController |  |
| GET | /dashboards/ dados/:idUsuario | dashboardController | Busca dados agregados para o |

|  |  |  |  |
| --- | --- | --- | --- |
| Método | URL | Controller | Descrição |
|  |  |  | dashboard do usuário. |

## Validação de Formulários (Backend)

As validações no backend são cruciais para garantir a integridade dos dados antes de persistir no banco de dados. As principais validações implementadas nos controllers incluem:

* + - Cadastro de Usuário: Verificação se campos obrigatórios (nome, email, senha) foram enviados.
    - Login: Verificação se email e senha foram fornecidos.
    - Quiz: Verificação se as respostas e o ID do usuário foram enviados.
    - Formulário: Verificação se título e mensagem foram enviados.

## Algoritmos e Lógica Principal

* + - Autenticação ( usuarioController.autenticar ): Recebe email e senha, consulta o banco de dados ( usuarioModel.autenticar ) para verificar se existe um usuário com essas credenciais. Retorna sucesso ou falha.
    - Cadastro de Quiz ( questionarioController.cadastrar ): Recebe as respostas do

questionarioModel.cadastrar

quiz e o ID do usuário, chama o dados na tabela questionario .

para inserir os

* + - Busca de Dados do Dashboard ( dashboardController.buscarDados ): Recebe o ID do usuário, consulta o banco de dados ( dashboardModel.buscarDados ) para obter informações relevantes (ex: nome do usuário, talvez contagem de livros lidos ou status do quiz) e retorna esses dados para o frontend.
    - Lógica de Recomendação (Implícita): A lógica para gerar as recomendações

baseadas no quiz parece estar principalmente no frontend ou não totalmente implementada no backend analisado.

[INSERIR DIAGRAMA/CÓDIGO: Fluxograma simples para o processo de autenticação ou cadastro do quiz.] [INSERIR DIAGRAMA/CÓDIGO: Bloco de código relevante do

dashboardController

questionarioController

ou com comentários.]

## Aplicação de Conceitos Matemáticos

Conceitos matemáticos básicos são aplicados em:

* + - Processamento do Quiz: Embora a lógica exata não esteja detalhada no backend, potencialmente poderia envolver contagens ou pontuações simples para determinar preferências.
    - Dashboard: Cálculos para exibir estatísticas, como porcentagens de progresso,

médias ou contagens totais (ex: total de feedbacks enviados - se implementado).

# Aspectos Socioemocionais

## Origem do Interesse pelo Tema (Linha da Vida)

Meu interesse pelo tema da leitura e suas dificuldades surgiu no começo do ensino médio. Nessa época, eu tinha certa dificuldade em socializar e encontrei nos livros uma forma de conexão e aprendizado. Um marco foi a leitura do livro "Como Fazer Amigos e Influenciar Pessoas", de Dale Carnegie, que pertencia ao meu irmão. Essa experiência mostrou o poder transformador da leitura, não apenas para adquirir conhecimento, mas também para o desenvolvimento pessoal e interpessoal, motivando-me a criar algo que pudesse facilitar esse acesso a outros.

## Valores Pessoais Refletidos no Projeto

O projeto "Leitura Descomplicada" reflete alguns valores pessoais que considero importantes:

* + - Resiliência: Assim como a leitura pode exigir persistência para superar partes desinteressantes ou complexas, o desenvolvimento do projeto exigiu resiliência para enfrentar desafios técnicos e manter a motivação.
    - Intelectualidade e Aprendizado Contínuo: Acredito no poder do conhecimento

adquirido através dos livros, mesmo os de ficção. O projeto busca ser uma ferramenta que estimule essa busca constante por aprendizado.

* + - Empatia: A leitura nos permite ver o mundo pelos olhos de outros, sejam autores

ou personagens. Desenvolver a plataforma foi um exercício de empatia ao tentar entender as dificuldades e necessidades do público-alvo, colocando-me no lugar de quem luta para criar o hábito de ler.

## Maior Dificuldade Enfrentada

A maior dificuldade durante o desenvolvimento foi encontrar a melhor forma de transmitir a mensagem central do projeto: convencer os usuários de que dedicar tempo à leitura é uma ação valiosa e recompensadora, especialmente no contexto atual de rotinas corridas e excesso de informações rápidas. Traduzir essa mensagem em funcionalidades engajadoras e uma interface atraente foi um desafio constante.

## Maior Superação Alcançada

Considero como maior superação a capacidade de construir uma aplicação web completa (full-stack), desde o backend com Node.js e banco de dados até o frontend interativo. Além disso, definir uma identidade visual coesa para o site foi um marco importante. Particularmente, o desenvolvimento do frontend representou uma grande superação, pois eu não me considerava muito habilidoso nessa área e tinha pouca experiência prévia com CSS antes de iniciar este projeto.

## Agradecimentos

Gostaria de expressar minha gratidão a algumas pessoas que foram fundamentais durante esta jornada:

* + - Ao meu irmão, por todo o apoio e ajuda constante em meus projetos.
    - Ao meu barbeiro, pelas valiosas dicas e insights sobre design e experiência do usuário (UI/UX).
    - Ao Igor, pela disposição em ouvir minhas ideias e oferecer opiniões construtivas que contribuíram significativamente para a construção deste trabalho.

# Testes

[PLACEHOLDER: SEÇÃO 9 - TESTES] \* Tipos de Testes: (Ex: Testes manuais de funcionalidade, Testes de usabilidade informais). \* Procedimentos: (Ex: Naveguei por todas as telas, tentei cadastrar/logar com dados válidos e inválidos, respondi ao quiz, verifiquei o dashboard). \* Resultados: (Ex: Funcionalidades principais operando conforme esperado, alguns bugs menores de layout corrigidos, etc.).

# Conclusão

[PLACEHOLDER: SEÇÃO 10 - CONCLUSÃO] \* Resumo dos Resultados: (Ex: O projeto atingiu os objetivos de criar uma plataforma funcional para incentivo à leitura,

implementando as tecnologias X, Y, Z). \* Lições Aprendidas: (Técnicas: aprofundamento em Node.js, CSS, modelagem de dados; Pessoais: gerenciamento de tempo, resiliência, importância do design). \* Trabalhos Futuros: (Ex: Implementar gamificação, integração com APIs externas, melhorar algoritmo de recomendação, adicionar funcionalidades sociais).

# Referências Bibliográficas

[PLACEHOLDER: SEÇÃO 11 - REFERÊNCIAS BIBLIOGRÁFICAS] \* Turban, E., Rainer Jr, R. K.,

Potter, R. E. (2007). *Administração de Tecnologia da Informação: Teoria e Prática*. LTC. \* Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business. \* Project Management Institute. (2017). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* – Sixth Edition. PMI. \* [Link para documentação do Node.js] \* [Link para documentação do Express.js] \* [Link para documentação do MySQL] \* [Link para tutorial ou artigo específico que ajudou] \* [PDFs das aulas: 02.SP1.Aula02 Documentacao Parte Um..., 03.SP1.Aula03 Documentacao Parte Dois...]