

CENTRO UNIVERSITÁRIO UNIVERSIDADE NOVE DE JULHO

Bacharelado em Ciência da Computação

DOCUMENTAÇÃO DO PROJETO – 3º SEMESTRE

Sistema Interativo com Integração em Blockchain para Registro de Dados

Beatriz Kelly Lopes Rocha – 924113660

Felipe Gabriel Bozzo de Araujo – 924116434

Maria Eduarda R.B.R dos Santos – 925101989

Raquel Santos Vieira – 924107157

Sabrina Caseiro – 924105555

Tainá Souza Alves – 924114166

São Paulo - SP, 2025

CENTRO UNIVERSITÁRIO UNIVERSIDADE NOVE DE JULHO	1
2. Fundamentação Teórica	4
2.1 Blockchain Simulada com Decks e Cartas	4
2.2 APIs RESTful	4
2.3 Arquitetura Web: Front-end e Back-end	4
2.4 Banco de Dados NoSQL (MongoDB)	5
2.5 Segurança e Integridade dos Dados	5
6. Desenvolvimento Back-end	9
7.1 Users	10
7.2 Transacao	10
7.3 Categoria	11
7.4 Deck	11
7.5 Estrutura Relacional no Contexto NoSQL	11
8 Integração com Blockchain	13
8.1 Modelo de Blockchain Simulada	13
8.2 Funcionamento da Integração	13
8.3 Benefícios da Simulação	14
Prisma ORM	14

Objetivo do Projeto

Este projeto tem caráter educacional e visa demonstrar uma interface interativa que utiliza uma estrutura de blockchain representada por decks e cartas, garantindo a segurança dos dados e o acompanhamento eficaz dos processos. O sistema completo inclui design visual, front-end com React e Next.js, back-end utilizando Node.js, banco de dados NoSQL (MongoDB) com Prisma ORM e uma API que integra as funcionalidades, incluindo a comunicação com a blockchain simulada. Todos os componentes estão organizados e versionados em um repositório no GitHub.

Link do repositório do projeto: [inserir link]

1. Introdução

O avanço das tecnologias digitais tem impulsionado a busca por soluções cada vez mais seguras e transparentes para o gerenciamento de dados. Nesse contexto, a tecnologia blockchain destaca-se por sua capacidade de registrar informações de forma imutável, descentralizada e auditável.

Este projeto propõe o desenvolvimento de um sistema interativo que integra a tecnologia blockchain simulada, representada por decks e cartas, a uma aplicação web funcional. Os usuários podem visualizar e registrar dados de maneira prática e confiável, simulando os princípios de rastreabilidade e integridade presentes na blockchain tradicional.

A solução abrange desde a criação da identidade visual, passando pelo desenvolvimento front-end (React + Next.js), back-end (Node.js), banco de dados (MongoDB) e a integração com a lógica de blockchain. O projeto tem como objetivo demonstrar, de forma didática, como a blockchain pode ser aplicada fora do mercado financeiro, servindo para ambientes acadêmicos, corporativos e educacionais.

Esta documentação apresenta as etapas do desenvolvimento, as decisões técnicas adotadas e os resultados obtidos.

2. Fundamentação Teórica

O desenvolvimento de sistemas modernos exige o domínio de conceitos fundamentais. Este projeto adota uma arquitetura baseada em front-end, back-end, banco de dados NoSQL e blockchain simulada, integrada por meio de uma API. A seguir, são apresentados os principais fundamentos teóricos:

2.1 Blockchain Simulada com Decks e Cartas

O conceito inicial da blockchain simulada foi implementado em Python, mas posteriormente adaptado para a stack atual. O modelo utiliza decks e cartas como abstrações visuais e lógicas da blockchain tradicional. Cada carta representa um bloco de informações, e cada deck representa uma cadeia encadeada, mantendo o conceito de rastreabilidade e integridade dos dados.

2.2 APIs RESTful

As APIs RESTful são responsáveis pela comunicação entre as camadas do sistema. Utilizando o protocolo HTTP, elas permitem operações como criação, leitura, atualização e exclusão de dados (CRUD), além de promover a integração entre o front-end, o banco de dados e a lógica de blockchain.

2.3 Arquitetura Web: Front-end e Back-end

O front-end é desenvolvido em React e Next.js, oferecendo uma interface dinâmica e responsiva. O back-end, desenvolvido em Node.js, gerencia a lógica de negócio, a comunicação com o banco de dados MongoDB via Prisma e a integração com a blockchain simulada.

2.4 Banco de Dados NoSQL (MongoDB)

Diferente dos bancos relacionais, o MongoDB utiliza uma estrutura orientada a documentos. Os dados são armazenados em coleções, e cada documento possui um formato flexível em JSON. Essa abordagem oferece escalabilidade, flexibilidade e fácil integração com aplicações modernas.

2.5 Segurança e Integridade dos Dados

A segurança é garantida pela utilização de uma estrutura blockchain simulada, onde cada carta (bloco) possui um identificador único e um hash que conecta as cartas sequencialmente. Esse mecanismo assegura que qualquer alteração em uma carta invalida toda a sequência, preservando a integridade dos dados.

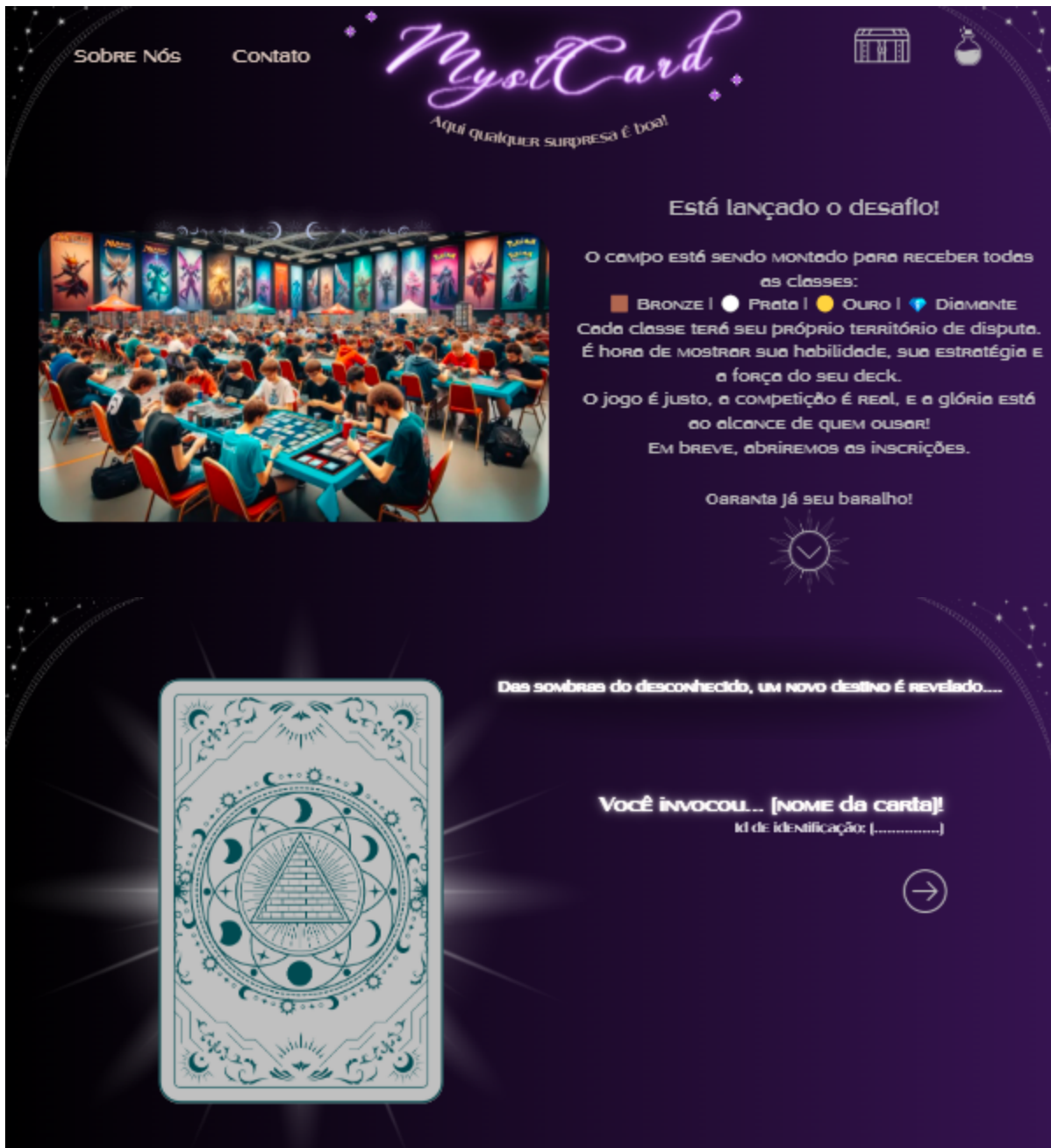
3. Tecnologias Utilizadas

- Design: Canva, para prototipagem e identidade visual.
- Front-end: React.js e Next.js, utilizando rotas dinâmicas e hooks para controle de estado.
- Back-end: Node.js, com desenvolvimento de API RESTful.
- ORM: Prisma, para interação com MongoDB.
- Banco de Dados: MongoDB (NoSQL).
- Blockchain Simulada: Estrutura baseada em decks e cartas com validação via hash.
- Versionamento: Git e GitHub.

4. Design e Wireframe

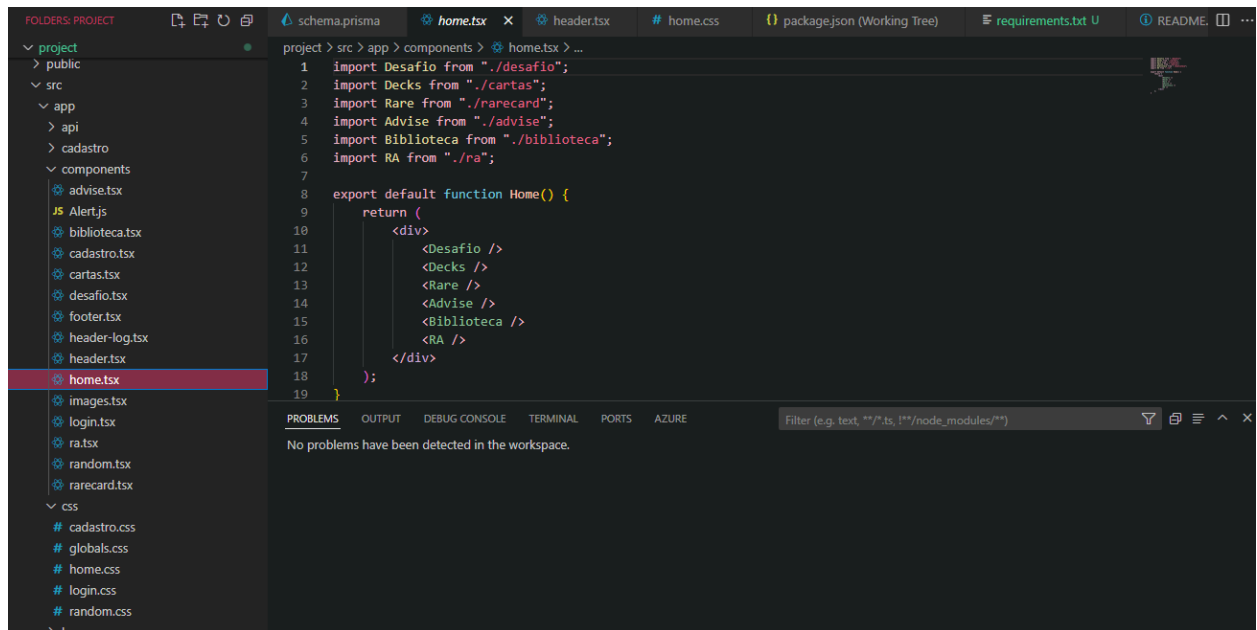
O design prioriza a usabilidade e a clareza das informações, com interface moderna e intuitiva. Os wireframes e protótipos estão disponíveis no repositório do projeto.

Imagens retiradas do design



5. Desenvolvimento Front-end

A camada de front-end foi desenvolvida utilizando React com Next.js, implementando páginas dinâmicas, hooks e consumo da API. O layout é responsivo e segue os princípios de design centrado no usuário.

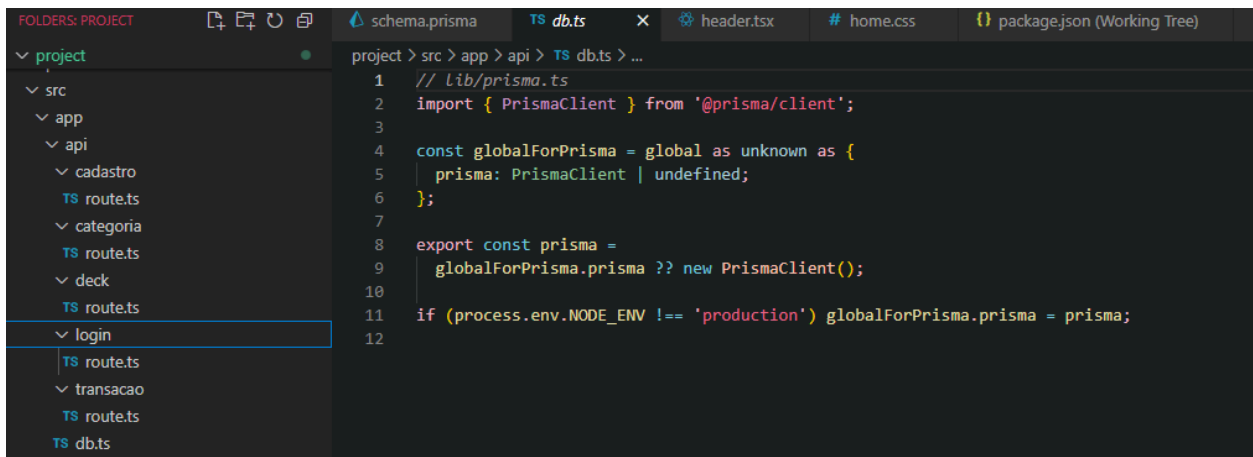


6. Desenvolvimento Back-end

O back-end foi desenvolvido em Node.js com TypeScript, garantindo maior segurança e qualidade no código por meio da tipagem estática. A principal responsabilidade do back-end é expor uma API RESTful que gerencia a lógica do sistema e a comunicação com o banco de dados MongoDB.

Utilizando o Prisma ORM, a API oferece rotas organizadas para realizar operações CRUD nas entidades do sistema, como usuários, transações, decks e categorias. Além das operações básicas, a API conta com endpoints específicos para manipular e validar a blockchain simulada, assegurando a integridade e rastreabilidade dos dados.

Essa estrutura modular e tipada permite que o front-end interaja de forma eficiente e segura com o back-end, facilitando a manutenção, escalabilidade e a implementação de novas funcionalidades.



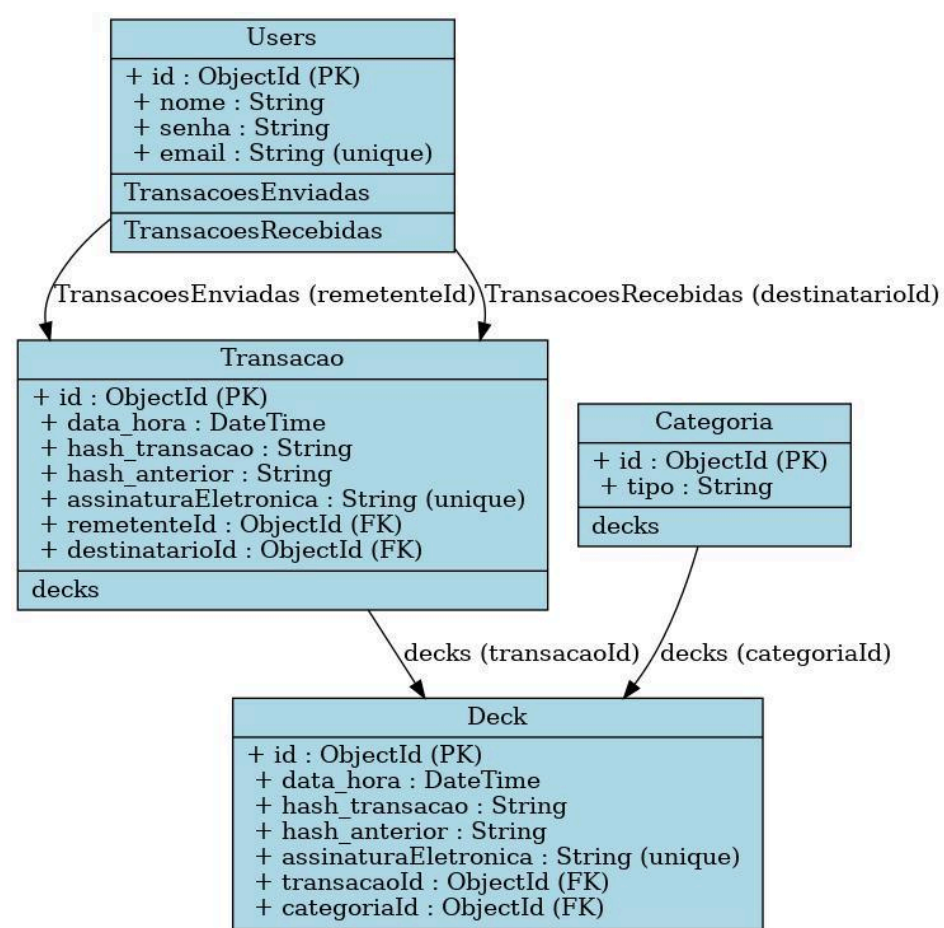
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'src', 'app', and 'api'. The 'api' folder is expanded, showing files like 'cadastro', 'categoria', 'deck', 'login', 'transacao', and 'db.ts'. The 'db.ts' file is selected. The code editor shows the following TypeScript code:

```
1 // lib/prisma.ts
2 import { PrismaClient } from '@prisma/client';
3
4 const globalForPrisma = global as unknown as {
5   prisma: PrismaClient | undefined;
6 };
7
8 export const prisma =
9   globalForPrisma.prisma ?? new PrismaClient();
10
11 if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma;
12
```

7. Banco de Dados NoSQL

O sistema foi desenvolvido utilizando um banco de dados NoSQL, especificamente o MongoDB, cuja modelagem é realizada por meio do Prisma ORM. O MongoDB armazena os dados em documentos no formato BSON (uma extensão do JSON), organizados em coleções. Esse modelo não relacional proporciona maior flexibilidade, escalabilidade horizontal e facilidade na manipulação de dados semi-estruturados.

Diagrama vertical do seu banco NoSQL (MongoDB)



A modelagem adotada reflete as entidades principais do sistema, suas relações e regras de integridade, conforme descrito a seguir:

7.1 Users

Coleção responsável por armazenar os dados dos usuários do sistema. Cada documento possui um identificador único (*id*), nome (*nome*), e-mail (*email*) e senha (*senha*). O e-mail é definido como campo único, garantindo que não haja duplicidade de cadastros. Além disso, há referências para as transações em que o usuário atuou como remetente e como destinatário.

7.2 Transacao

Representa o conceito de uma transação entre dois usuários. Cada transação contém um hash exclusivo (*hash_transacao*), um hash do elemento anterior (*hash_anterior*) para garantir a imutabilidade e rastreabilidade dos dados, além de uma assinatura eletrônica única (*assinaturaEletronica*). Está associada diretamente a um remetente e a um destinatário, ambos vinculados a registros da coleção *Users*. Uma transação também pode agrupar múltiplos decks relacionados.

7.3 Categoria

Define a categorização dos decks, com o campo *tipo* descrevendo a natureza ou grupo ao qual o deck pertence. Cada categoria pode estar associada a diversos decks.

7.4 Deck

Representa um conjunto de informações dentro do sistema, relacionado a uma transação específica e classificado por uma categoria. Cada deck possui também informações de integridade como *hash_transacao*, *hash_anterior* e *assinaturaEletronica*, reforçando o compromisso com a segurança e rastreabilidade dos dados.

7.5 Estrutura Relacional no Contexto NoSQL

Embora o MongoDB seja um banco de dados NoSQL e não utilize relações da mesma forma que bancos relacionais, o Prisma permite definir relações lógicas entre os documentos. Essas relações são implementadas por meio de referências utilizando identificadores (ObjectId), o que garante a integridade e consistência dos dados na aplicação.

O diagrama do banco de dados foi representado utilizando uma abordagem conceitual, destacando as coleções (*Users*, *Transacao*, *Deck* e *Categoria*) e suas inter-relações. Esse diagrama está disponível no repositório do projeto e serve como guia para a compreensão da arquitetura de dados da aplicação.

A representação desse modelo está ilustrada no diagrama de dados (Figura 1).

8 Integração com Blockchain

A integração com a blockchain simulada constitui um dos pilares do sistema, proporcionando segurança e rastreabilidade aos dados registrados. Originalmente implementada em Python, a estrutura foi adaptada para a stack atual utilizando Node.js, promovendo maior interoperabilidade e escalabilidade.

8.1 Modelo de Blockchain Simulada

A blockchain simulada é representada por uma estrutura de decks e cartas, onde cada carta equivale a um bloco contendo informações específicas. Cada bloco (carta) possui um identificador único e um hash criptográfico que referencia o bloco anterior, formando uma cadeia encadeada. Essa cadeia assegura a imutabilidade dos dados, pois qualquer modificação em um bloco comprometeria a integridade de toda a sequência.

8.2 Funcionamento da Integração

A API desenvolvida em Node.js é responsável por gerenciar a criação, validação e

consulta dos decks e cartas, realizando as seguintes funções principais:

- **Criação de cartas:** Registro de novos blocos contendo dados validados, vinculados ao hash do bloco anterior para manter a continuidade da cadeia.
- **Validação da cadeia:** Verificação da integridade da sequência de cartas por meio da comparação dos hashes, garantindo que não houve alterações indevidas.
- **Consulta dos dados:** Permite o acesso às informações armazenadas na blockchain simulada, fornecendo rastreabilidade completa para os usuários.

8.3 Benefícios da Simulação

Embora não utilize uma blockchain pública ou distribuída, a simulação adotada neste projeto proporciona uma compreensão prática dos conceitos fundamentais da tecnologia blockchain, como imutabilidade, segurança por meio de hashes e encadeamento dos blocos.

Esta abordagem facilita a aplicação didática e permite o uso da tecnologia em ambientes controlados, como acadêmicos ou corporativos, onde a blockchain real poderia ser excessivamente complexa e custosa.

8.4 Prisma ORM

Prisma é um ORM moderno usado para facilitar a comunicação entre a aplicação Node.js e o banco de dados MongoDB. Ele permite manipular os dados de forma simples e segura, gerando consultas tipadas e otimizadas. Com Prisma, é possível

definir o modelo dos dados e as relações entre coleções, garantindo a integridade e facilitando operações CRUD no banco NoSQL. Essa integração torna o desenvolvimento do back-end mais eficiente e confiável.

9 Conclusão

O sistema desenvolvido demonstra, de forma funcional e didática, os princípios da tecnologia blockchain aplicados fora do contexto financeiro. Através da representação por decks e cartas, foi possível implementar um modelo que garante integridade, segurança e rastreabilidade dos dados, utilizando tecnologias modernas e de fácil escalabilidade.

10 Referências

CASSINO, F. *Blockchain: fundamentos e aplicações*. 2. ed. São Paulo: Editora Tech, 2021.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. Tese (Doutorado em Ciência da Computação) – University of California, Irvine, 2000.

REACT. React – A JavaScript library for building user interfaces. Disponível em: <https://reactjs.org/>. Acesso em: 29 maio 2025.

NEXT.JS. Next.js – The React Framework. Disponível em: <https://nextjs.org/>. Acesso em: 29 maio 2025.

PRISMA. Prisma – Next-generation Node.js and TypeScript ORM. Disponível em: <https://www.prisma.io/>. Acesso em: 29 maio 2025.

MONGODB. MongoDB – The database for modern applications. Disponível em:

<https://www.mongodb.com/>. Acesso em: 29 maio 2025.

NODE.JS. Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Disponível em: <https://nodejs.org/>. Acesso em: 29 maio 2025.

CANVA. Canva – Design, gráfico e prototipagem. Disponível em:

<https://www.canva.com/>. Acesso em: 29 maio 2025.

Agradecimentos

Gostaríamos de expressar nossa sincera gratidão a todos que contribuíram para a realização deste projeto. Agradecemos especialmente aos nossos professores e orientadores pelo apoio, orientação e incentivo ao longo do desenvolvimento. Também agradecemos aos colegas e familiares pelo suporte e motivação durante essa jornada acadêmica.

Este trabalho foi possível graças ao esforço coletivo e à colaboração de todos, que de alguma forma colaboraram para o sucesso deste projeto.