UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

EDOUARD JEAN ALEXANDRE

DESENVOLVIMENTO DE UM SISTEMA WEB PARA GERENCIAMENTO DE EMPRÉSTIMOS E VENDAS DE FERRAMENTAS E COMPONENTES EM LABORATÓRIOS DE HARDWARE.

TUCURUÍ
2024

EDOUARD JEAN ALEXANDRE

DESENVOLVIMENTO DE UM SISTEMA WEB PARA GERENCIAMENTO DE EMPRÉSTIMOS E VENDAS DE FERRAMENTAS E COMPONENTES EM LABORATÓRIOS DE HARDWARE

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia de Computação, do Campus Universitário de Tucuruí, da Universidade Federal do Pará, como requisito parcial para obtenção do título de Bacharelado em Engenharia de Computação.

Orientador: Dr. Marco José de Sousa
Coorientador: Dr. Bruno Merlin

TUCURUÍ
2024

EDOUARD JEAN ALEXANDRE

DESENVOLVIMENTO DE UM SISTEMA WEB PARA GERENCIAMENTO DE EMPRÉSTIMOS E VENDAS DE FERRAMENTAS E COMPONENTES EM LABORATÓRIOS DE HARDWARE

Este Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharelado em Engenharia de Computação, e aprovado em sua forma final pela Faculdade de Engenharia de Computação, do Campus Universitário de Tucuruí, da Universidade Federal do Pará.

Data da aprovação: ____/____/_____

Conceito: _____

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br MARCO JOSE DE SOUSA ROCHA
Data: 27/10/2025 15:00:08-0300
Verifique em https://validar.iti.gov.br

_____
Dr. Marco José de Sousa (UFPA/FECOMP)
Orientador

_____
Dr. Bruno Merlin (UFPA/FECOMP)
Coorientador

Documento assinado digitalmente
gov.br OTAVIO NOURA TEIXEIRA
Data: 30/10/2025 23:49:45-0300
Verifique em https://validar.iti.gov.br

Documento assinado digitalmente
gov.br BRUNO MERLIN
Data: 27/10/2025 19:43:41-0300
Verifique em https://validar.iti.gov.br

_____
Dr. Otávio Noura Teixeira (UFPA/FECOMP)
Examinador Interno

Documento assinado digitalmente
gov.br HELENO FULBER
Data: 27/10/2025 15:34:04-0300
Verifique em https://validar.iti.gov.br

_____
Dr. Heleno Fulber (UFPA/FECOMP)
Examinador Interno

À minha falecida tia e madrinha Liliane Evelyne Anicet.

AGRADECIMENTOS

Primeiramente à Deus por ter olhado para mim durante toda a minha vida e principalmente durante a minha graduação, colocando as pessoas certas no meu caminho. Mas também por ter me dado força e paciência para enfrentar desafios difíceis como a pandemia que ocorreu nos semestres mais pesados do curso assim como a perda de entes queridos quando eu menos esperava.

Aos meus pais e aos meus familiares próximos pelos ensinamentos, mas também pelo apoio quando eu mais precisei. Uma menção especial para minha mãe Maria Célia que sempre teve paciência e cuidado comigo. Obrigado por aturar um "velho rabugento" no dia a dia.

À pessoa que me recebeu na sua casa quando eu tinha apenas 8 dias e que desde lá até a hora do seu falecimento em 2024 nunca deixou de cuidar, de orar por mim e de me apoiar mesmo quando ninguém acreditava mais em mim, a minha querida segunda mãe, tia do meu pai e também minha madrinha de crisma Evelyne. Obrigado por tudo, prometi à senhora que lhe mostraria meu diploma no nosso próximo encontro infelizmente os planos de Deus foram outros. Espero que lá de cima esteja orgulhosa de mim.

Ao todo o corpo de docentes da Universidade Federal do Pará (UFPA) do Campus da cidade de Tucuruí, especialmente aos meus orientadores os professores Marco José de Souza e Bruno Merlin sem os quais finalizar este trabalho não teria sido possível. Encontrei vários profissionais de diversos lugares diferentes (até mesmo de nacionalidades diferentes), mas todos com o mesmo objetivo, passar o seu conhecimento para as gerações futuras. Obrigado a todos vocês pelos seus ensinamentos durante todo o curso. Levarei todo esse aprendizado comigo e o usarei na minha futura carreira profissional.

E por fim, mas não menos importante, aos amigos que fiz no caminho desde que entrei em 2019 na universidade. Com certeza sem vocês não teria chegado até aqui, até porque ninguém neste mundo vence sozinho. Conheci muitas pessoas boas e com certeza não me esquecerei de vocês.

Estou muito grato a todos vocês, e espero que possamos compartilhar novas aventuras no futuro. Muito obrigado!

“Design não é apenas como algo parece, mas como funciona.”

STEVE JOBS

RESUMO

O gerenciamento manual de empréstimos e vendas de ferramentas e componentes em laboratórios de hardware apresenta desafios como perda de prazos, falhas na comunicação e falta de rastreabilidade. Para solucionar esses problemas, este trabalho propõe o desenvolvimento de um sistema web responsivo, seguro e automatizado, construído com Ionic React (frontend) e Node.js (backend), que utiliza um banco de dados MySQL para armazenamento de dados. O sistema implementa autenticação por Json Web Token (JWT) com diferenciação de tipos de usuários, notificações push via Service Worker, envio de e-mails automatizados com OAuth2 e verificações diárias de atrasos através de cron jobs. A arquitetura monolítica do backend, concentrada em um único arquivo, foi escolhida para simplificar a manutenção, enquanto o frontend utiliza componentes reutilizáveis em TypeScript (TSX) para garantir consistência e responsividade em dispositivos móveis e desktop. Testes realizados demonstraram que a solução tem o potencial de reduzir o tempo de processamento de empréstimos, reduzir falhas humanas e melhorar a experiência do usuário com interfaces intuitivas. Conclui-se que o sistema atende aos requisitos de segurança, usabilidade e automação, portanto ele pode ser expandido para outros laboratórios ou integrado a tecnologias como QR Code para controle de inventário.

Palavras-chave: Sistemas web, Ionic React, Node.js, MySQL, autenticação JWT, notificações push, e-mails automatizados, laboratórios de hardware.

ABSTRACT

Manually managing loans and sales of tools and components in hardware laboratories presents challenges such as missed deadlines, communication failures, and lack of traceability. To address these issues, this work proposes the development of a responsive, secure, and automated web system built with Ionic React (frontend) and Node.js (backend), using a MySQL database for data storage. The system implements Json Web Token (JWT) authentication with user type differentiation, *push* notifications via Service Worker, automated email sending with OAuth2, and daily overdue checks via cron jobs. The backend's monolithic architecture, concentrated in a single file, was chosen to simplify maintenance, while the frontend uses reusable TypeScript (TSX) components to ensure consistency and responsiveness across mobile and desktop devices. Tests have demonstrated that the solution has the potential to reduce loan processing time, reduce human error, and improve the user experience with intuitive interfaces. The system meets security, usability, and automation requirements and can be expanded to other laboratories or integrated with technologies such as QR codes for inventory control.

Keywords: Web systems, Ionic React, Node.js, MySQL, JWT authentication, *push* notifications, automated emails, hardware laboratories.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

JSX	JavaScript XML
TSX	Junção de TipeScript e JavaScript XML
JWT	JSON Web Token
JSON	JavaScript Object Notation
HTTPS	HyperText Transfer Protocol Secure (Protocolo de Transferência de Hipertexto Seguro)
Admin	Administrador (tipo de usuário)
PK	Primary Key (Chave Primária)
FK	Foreign Key (Chave Estrangeira)
I/O	Input/Output (Entrada/Saída)
API	Application Programming Interface (Interface de Programação de Aplicações)
NPM	Node Package Manager (Gerenciador de Pacotes)
MiB	Mebibyte (unidade de medida de armazenamento; 1 MiB = 1.048.576 bytes)
ms	Milissegundo (unidade de tempo equivalente a 1 milésimo de segundo)
SSL	Secure Sockets Layer (camada de soquete seguro)
idle	sem uso no momento
CPU	Central Processing Unit (Unidade Central de Processamento)
URL	Uniform Resource Locator (Localizador Uniforme de Recursos)
RFID	Radio Frequency Identification (Identificação por Radiofrequência)
Apps	Aplicativos
IP	Internet Protocol (Protocolo de Internet)

LISTA DE SIMBOLOS


PK — Primary Key (Chave Primária)


FK — Foreign Key (Chave Estrangeira)


Coluna auto-gerada (AUTO_INCREMENT)

◇ Relacionamento opcional (0 ou mais registros)


Relacionamento obrigatório (1 ou mais registros)


Relacionamento de muitos para um (N:1)

SUMÁRIO

INTRODUÇÃO

O gerenciamento de ferramentas e componentes em laboratórios acadêmicos é, tradicionalmente, realizado de maneira manual, por meio de planilhas ou registros em papel. Esse processo, embora simples, apresenta sérias limitações, como a dificuldade em acompanhar a disponibilidade dos itens, o risco de perda de informações e a falta de transparência no controle de empréstimos e devoluções. Além disso, a ausência de um sistema informatizado compromete o uso consciente dos recursos, que em consequência prejudica tanto os alunos quanto os administradores.

Diante desse cenário, surgiu a necessidade de desenvolver uma solução tecnológica que permitisse automatizar esses processos, para tornar o acesso mais seguro, confiável e acessível. Para atender a essa demanda, foi desenvolvido um sistema web responsivo voltado ao gerenciamento de empréstimos e vendas de itens do laboratório, que integra autenticação de usuários, automação de tarefas administrativas e controle eficiente do inventário.

O sistema proposto busca atender simultaneamente às necessidades dos alunos, que utilizam os itens para atividades acadêmicas, e dos administradores, que são responsáveis pela gestão do acervo. O uso de tecnologias modernas como Ionic React para o frontend, Node.js para o backend e MySQL para armazenamento garante portabilidade, escalabilidade e segurança.

Assim, este trabalho tem como foco apresentar o processo de concepção, desenvolvimento e validação do sistema, afim de descrever suas funcionalidades, arquitetura e contribuições práticas para a comunidade acadêmica.

1.1 Objetivo Geral

Desenvolver um sistema web responsivo para facilitar o empréstimo e a venda de itens em um laboratório acadêmico, com autenticação segura, automação de processos e controle eficaz do inventário.

1.2 Objetivos Específicos

1. Implementar um sistema de cadastro e login de usuários com verificação de e-mail.
2. Criar uma interface intuitiva e responsiva, adaptada para desktop e dispositivos móveis.

3. Desenvolver um sistema de crédito digital, recarregáveis.
4. Implementar notificações push e e-mails automáticos para alertar usuários.
5. Criar um painel administrativo para controle de transações, itens e usuários.
6. Automatizar verificações diárias de prazos e devoluções.
7. Garantir a segurança do acesso, autenticação e transmissão de dados.

2 METODOLOGIA DE DESENVOLVIMENTO

Diferente de trabalhos de caráter científico, este projeto adota uma abordagem tecnológica, voltada ao desenvolvimento prático de uma solução de software. A metodologia escolhida organiza o trabalho em etapas bem definidas, que incluem levantamento de requisitos, análise, projeto, implementação e validação.

2.1 Levantamento de Requisitos

A definição das funcionalidades partiu da análise do funcionamento atual do laboratório e da consulta aos responsáveis pela gestão dos itens. A partir disso, foram definidos os requisitos funcionais, como cadastro, login, empréstimo e devolução, e os não funcionais, como usabilidade, segurança e compatibilidade entre dispositivos.

2.2 Tecnologias Usadas

As aplicações Web, também conhecidas como WebApps, são sistemas acessados por meio de navegadores e que funcionam a partir do uso da infraestrutura da internet. Elas dispensam a instalação local de programas e oferecem funcionalidades dinâmicas e interativas semelhantes às dos aplicativos tradicionais, sendo muito utilizadas em serviços bancários, redes sociais, sistemas de gestão, entre outros (W3C, 2023).

Essas aplicações normalmente adotam a arquitetura **cliente-servidor**, sendo compostas por duas partes principais: frontend e backend. O frontend é a camada responsável pela interface com o usuário e é executado diretamente no navegador web do cliente. Ele é construído a partir de linguagens como HTML, CSS e JavaScript, e pode ser desenvolvido com o auxílio de frameworks modernos, como React, Angular, Vue ou Svelte (Figura 1).

Figura 1 - Os mais Populares Frameworks (Frontend)

Fonte: Angular / React / Vue / Svelte (WILKYWAY, 2022)

Já o backend é o sistema que roda no servidor. Ele é responsável por processar requisições, acessar o banco de dados, aplicar regras de negócio e enviar respostas ao cliente.

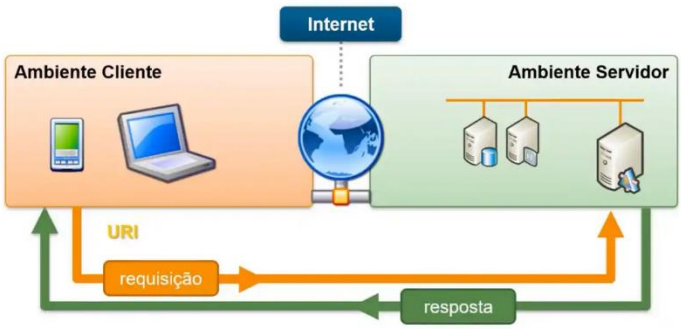
O backend opera de forma contínua, com tecnologias capazes de responder ao protocolo HTTP (como Node.js, Python, PHP, entre outros mostrado na Figura 2) e geralmente se conecta a um Sistema Gerenciador de Banco de Dados (SGBD), como MySQL, PostgreSQL ou MongoDB.

Figura 2 – Os mais Populares Frameworks (Backend)

Fonte: c-metric.com (RAMI, 2024)

A Figura 3 a seguir ilustra essa separação entre cliente e servidor e como eles se comunicam por meio da internet.

Figura 3 - Comunicação Cliente-Servidor



Fonte: CLIENT X SERVER (Filipe Gonçalves, 2024)

2.2.1 Frontend: Ionic + React

O framework Ionic é conhecido por facilitar a criação de aplicações web **responsivas** que funcionam tanto em dispositivos móveis quanto em desktops. O Ionic fornece uma biblioteca de componentes prontos, otimizados para performance e com aparência nativa, o que acelera o processo de desenvolvimento (IONICFRAMEWORK, 2025).

Para a construção da interface, o Ionic pode ser integrado ao React, uma biblioteca JavaScript[1] criada pelo Facebook, que permite o desenvolvimento de interfaces de usuário reativas e "componentizadas" (FACEBOOK, 2025). O React é amplamente utilizado na indústria de software por sua eficiência e pela facilidade de reutilização de componentes. Entre os frameworks alternativos que também oferecem recursos semelhantes estão o Angular, o Vue.js e o Svelte, todos utilizados na criação de aplicações web modernas e escaláveis como mostrado na Figura 4.

A implementação pode ser feita com o uso de TSX, uma extensão que combina TypeScript (uma linguagem de programação baseada em JavaScript, com tipagem estática que ajuda a evitar erros em tempo de desenvolvimento) com JSX (uma sintaxe que permite escrever código semelhante a **HTML**[2] dentro do JavaScript, que facilita a construção de componentes visuais) (MICROSOFT, 2025; META, 2025). Essa abordagem melhora a legibilidade do código e proporciona maior segurança no processo de desenvolvimento, além de facilitar a manutenção a longo prazo.

Figura 4 - Comparativo Entre os Frameworks

	React	Angular	Vue.js	Svelte
Tipo	Bibiloteca (UI)	Framework completo	Framework progressivo	Compllador / Framework inovador
Criado por	Meta (Facebook)	Googie	Evan You (ex-Google)	Rich Harris (Vercel)
Ano de lancamento	2013	2010	2014	2016
Curvo de aprendiizado	Moderada	Grande	Baixa (fácil para iniolentes)	Muito baixa
Tamanho da comunidade	Muito grande	Grande	Grande	Excelente (sem virtual DOM)
Desempenho	Muito bom	Bom	Muito bom	Excelente (sem virtual DOM)
Binding (dados)	Unidirecional	Bidirecional	Bidirecional (ou unidirecional)	Reativo por padrão
DOM	Virtual DOM	Virtual DOM	Virtual DOM	Compila para DOM nativo
Escalabiildade	Aita (com libs éxtemas)	Alta (estrutura completa já inciutlã)	Boa (com Vuex, Vue Router, etc	Boa (mas ainda em amaduracimento)
Ferramentas nativas	Requer libs externas (etz. Requje)	Inciui tudo (Rx JS. CLI, DI, etc.)	CLI, Vuex. Vue Router	Menos dependencia externas
Ecossistema	Rico, mas fragmentado	Integrado e completo	Rico e bem integrado	Mais encuto, mas promissor
Popularidade (2524)	Altissima	Alta	Alta	Crescendo muito rapido

Fonte: Autoria Própria

[1] Linguagem de programação amplamente utilizada no desenvolvimento *web*, que permite criar funcionalidades dinâmicas em páginas e aplicações (MDN WEB DOCS, 2025).

[2] Linguagem de marcação padrão utilizada para estruturar conteúdos em páginas da web, como textos, links e imagens (W3C, 2025).

2.2.2 Backend: Node.js

O Node.js é um ambiente de execução de código JavaScript do lado do servidor, construído sobre o motor V8 do Google Chrome. Sua principal característica é a arquitetura baseada em event loop e I/O não bloqueante[3], o que permite lidar de forma eficiente com múltiplas conexões simultâneas, que o torna particularmente adequado para aplicações web em tempo real e serviços de rede escaláveis (DAHL, 2009; TILKOV; VELMURUGAN, 2010).

Diferentemente de plataformas tradicionais como PHP, Java ou .NET, que seguem modelos multithread, o Node.js utiliza um modelo single-thread orientado a eventos[4]. Isso reduz o consumo de recursos do servidor e aumenta o desempenho em cenários de alta concorrência (HERRON, 2016).

Uma de suas principais vantagens é a vasta quantidade de bibliotecas disponíveis através do Node Package Manager (NPM)[5], que fornece suporte para funcionalidades como autenticação, envio de e-mails, manipulação de banco de dados e integração com APIs externas. Além disso, sua integração com frameworks como Express.js facilita o desenvolvimento de rotas, middlewares e a criação de APIs RESTful (BROWN, 2020).

2.2.3 Banco de Dados: MySQL (SQL vs. NoSQL)

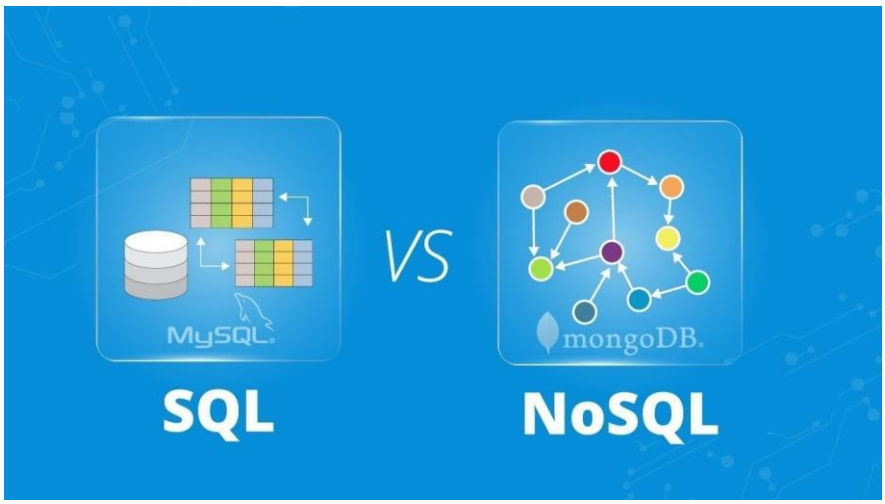
Os bancos de dados desempenham papel central no armazenamento e gerenciamento de informações em sistemas computacionais. Entre os principais modelos existentes, destacam-se os bancos relacionais (SQL) e os bancos não relacionais (NoSQL) (Figura 4).

[3] Modelo de programação onde operações de entrada/saída não impedem a execução de outras tarefas. O sistema continua com o processo de outras operações enquanto aguarda a conclusão das operações de I/O.

[4] Arquitetura onde uma única thread principal gerencia múltiplas operações através de um mecanismo de eventos, callbacks e operações assíncronas, diferentemente do modelo tradicional baseado em múltiplas threads.

[5] Gerenciador de pacotes padrão para o Node.js, que proporciona acesso a mais de um milhão de bibliotecas e ferramentas de desenvolvimento (HERRON, 2016).

Figura 5 - SQL vs NoSQL



Fonte: Edureka (KAPPAGANTULA, 2025)

Os bancos de dados relacionais utilizam a linguagem **SQL** (Structured Query Language) como padrão para a manipulação de dados estruturados. Nesse modelo, as informações são organizadas em tabelas relacionadas entre si, o que permite realizar consultas complexas, aplicar regras de integridade e garantir consistência por meio de transações e restrições. Além disso, sistemas relacionais como o **MySQL** oferecem suporte ao conjunto de propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), que assegura confiabilidade e segurança nas operações (ELMASRI; NAVATHE, 2020; ORACLE, 2025).

Por outro lado, os bancos **NoSQL** (Not Only SQL) surgiram para atender aplicações que lidam com grandes volumes de dados não estruturados ou semi-estruturados, que fornecem maior flexibilidade e escalabilidade horizontal. Diferentemente dos relacionais, não exigem esquemas fixos e podem adotar diferentes estruturas, como documentos, colunas, pares chave-valor ou grafos. Exemplos comuns incluem MongoDB, Cassandra e Redis, amplamente utilizados em cenários de dados variáveis e em tempo real (PRITCHARD, 2021).

2.2.4 Segurança da Conexão (HTTPS)

Para garantir uma comunicação segura entre os usuários e o servidor, o sistema utiliza o protocolo **HTTPS** (Hypertext Transfer Protocol Secure). Essa tecnologia emprega criptografia **TLS** (Transport Layer Security) para proteger os dados transmitidos, afim de evitar que sejam interceptados ou modificados por terceiros. Isso garante a confidencialidade e integridade das informações sensíveis trocadas, como senhas e dados pessoais. Além disso, a utilização de um certificado **SSL** (Secure Sockets Layer), obtido após a aquisição de um

domínio próprio, permite autenticar o servidor e transmitir confiança aos usuários, afim de evitar ataques como o "*man-in-the-middle*[6]" (MITM) (MOZILLA, 2024).

2.2.5 Segurança de Autenticação (JWT)

A autenticação do sistema foi implementada com base em JWT (JSON Web Token), um padrão aberto (RFC 7519) que permite a transmissão segura de informações entre partes como um objeto JSON[7] assinado digitalmente (AUTH0, 2025). Os tokens JWT são amplamente utilizados em aplicações modernas por sua leveza, portabilidade e por dispensarem o uso de sessões no servidor, o que os torna ideais para arquiteturas como SPA[8] (Single Page Applications) e APIs REST[9].

O motivo principal da escolha pelo JWT foi sua facilidade de integração com sistemas baseados em frontend e backend desacoplados, como o utilizado neste projeto. Diferentemente de sistemas tradicionais que utilizam sessões com cookies armazenados no servidor, o JWT é gerado no login e armazenado no cliente (geralmente no localStorage ou sessionStorage), sendo enviado em cada requisição protegida, o que facilita a escalabilidade e a descentralização da autenticação como demonstrado na Figura 6. Outras abordagens, como **OAuth2** e **SAML**, também são comuns, especialmente em sistemas corporativos ou integrações com redes sociais, mas apresentam maior complexidade de implementação.

Para garantir o armazenamento seguro das senhas dos usuários, foi adotado o algoritmo **bcrypt**, uma função de hash[10] criptográfico baseada em salting [11]e custo computacional ajustável, que dificulta significativamente ataques de força bruta[12] (OWASP, 2025). Mesmo que a base de dados seja comprometida, o bcrypt torna impraticável a recuperação das senhas originais sem acesso à chave de hashing e ao salt utilizado.

---

[6] Ataque em que um invasor intercepta a comunicação entre duas partes, assim ele ganha a possibilidade de ler, modificar ou inserir dados sem que os envolvidos percebam (MOZILLA, 2025).

[7] Formato leve de intercâmbio de dados, baseado em texto, utilizado para representar objetos e estruturas de forma legível tanto por humanos quanto por máquinas (ECMA INTERNATIONAL, 2025).

[8] Modelo de aplicação web que carrega uma única página HTML e atualiza dinamicamente seu conteúdo conforme a interação do usuário, sem necessidade de recarregar toda a página (MDN WEB DOCS, 2025).

[9] Conjunto de princípios de arquitetura para criação de interfaces de comunicação entre sistemas, baseadas em requisições HTTP e recursos identificados por URLs (FIELDING, 2000).

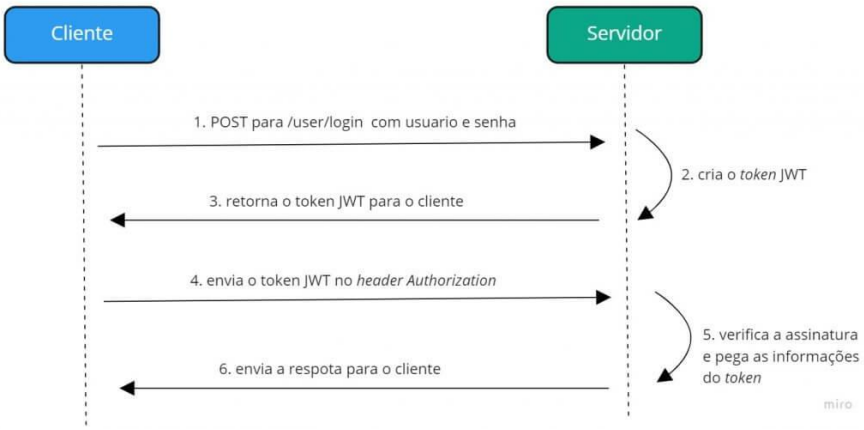
[10] Função criptográfica que converte uma entrada de dados em uma sequência fixa de caracteres, de forma irreversível, portanto ela serve como uma espécie de "impressão digital" da informação.

[11] Técnica de segurança que consiste em adicionar um valor aleatório (denominado "salt") à senha antes de aplicar a função de hash. Isso garante que senhas idênticas resultem em hashes diferentes, que dificultam ataques de força bruta e o uso de tabelas pré-computadas (*rainbow tables*).

[12] Método de ataque em que todas as combinações possíveis de caracteres são testadas até encontrar a senha correta.

Além disso, foi implementado um **middleware de autenticação** no backend, responsável por interceptar as requisições e verificar a validade dos tokens JWT antes de permitir o acesso às rotas protegidas. Essa abordagem adiciona uma camada extra de segurança e garante que apenas usuários autenticados possam acessar determinados recursos da aplicação.

Figura 6 - Funcionamento do token JWT



Fonte: *Boteco Digital* (ARAMBURU, 2022)

2.2.6 Envio de E-mails Automatizados (OAuth2)

O sistema utiliza o protocolo OAuth2 (Authorization Framework) para permitir a autenticação segura com provedores externos confiáveis, como Google e GitHub. O OAuth2 é um padrão amplamente adotado para delegação de acesso, isso permite que os usuários façam login sem precisar criar uma conta nova no sistema, já que utiliza suas credenciais já existentes (IETF, 2012). Essa abordagem melhora a usabilidade, reduz o atrito no processo de cadastro e evita o armazenamento local de senhas sensíveis.

A escolha pelo OAuth2 foi motivada, sobretudo, pela facilidade de integração com APIs modernas de provedores populares, especialmente do Google, que disponibiliza bibliotecas e documentação extensiva para esse fim. Além disso, o protocolo é compatível com padrões de segurança atualizados e já incorpora práticas como o uso de tokens temporários e escopos de permissão.

Embora o OAuth2 seja a solução adotada neste projeto, existem outras abordagens possíveis para autenticação federada, como o **SAML** (Security Assertion Markup Language), mais comum em ambientes corporativos, ou o OpenID Connect, que é uma camada de identidade construída sobre o próprio OAuth2 (OKTA, 2025). Alternativamente, o sistema também poderia utilizar o método tradicional baseado em login com e-mail e senha,

armazenados localmente, porém essa opção exigiria um cuidado maior com a segurança das credenciais e aumentaria a complexidade de manutenção.

Além da autenticação, o sistema implementa um serviço de envio automatizado de e-mails, que informa os usuários sobre transações realizadas, prazos de devolução de itens, e outros eventos importantes. Esse serviço é integrado a uma API de e-mails (como o Nodemailer com Gmail), e opera a partir de rotinas do backend disparadas por eventos específicos, para garantir que os usuários estejam sempre atualizados quanto às ações realizadas na plataforma (NODEMAILER, 2025).

2.2.7 Notificações Push (Service Workers)

A aplicação utiliza Service Workers, uma tecnologia que permite a execução de scripts em segundo plano no navegador, mesmo quando a aba da aplicação está fechada (MOZILLA, 2025). Esses scripts intermediários atuam como proxies entre o navegador e a rede, que possibilitam às funcionalidades como cache offline, sincronização em segundo plano e, principalmente, o envio de notificações push ao usuário.

No contexto deste projeto, os Service Workers são utilizados para notificar os usuários sobre prazos de devolução, confirmações de transações e alertas importantes referentes ao uso do sistema. Essa funcionalidade é especialmente útil quando o usuário não está com a aplicação aberta, mas precisa ser avisado sobre ações urgentes, como a proximidade do vencimento de um empréstimo. Com isso, melhora-se a comunicação entre o sistema e os usuários, para garantir o engajamento mesmo fora do ambiente da aplicação ativa.

Embora existam outras formas de alertar o usuário (como envio de e-mails ou atualizações dentro da interface do app), as notificações push via Service Workers oferecem uma experiência mais rápida e integrada, similar à dos aplicativos nativos, sem exigir interação direta do usuário no momento do disparo. Essa abordagem é bastante comum em **Progressive Web Apps** (PWAs), que visam oferecer uma experiência próxima à de apps instaláveis, diretamente pelo navegador (GOOGLE DEVELOPERS, 2024).

2.2.8 Tarefas Agendadas (Cron Jobs)

O sistema implementa cron jobs a partir da biblioteca node-cron para realizar verificações automáticas e enviar notificações e e-mails em horários predefinidos. Essa funcionalidade permite, por exemplo, alertar usuários sobre devoluções pendentes e atualizar status de empréstimos, que garantem um controle eficiente do inventário.

Essas tecnologias e conceitos formam a base do sistema desenvolvido, que proporcionam um ambiente seguro, automatizado e acessível para o gerenciamento de empréstimos e vendas no laboratório acadêmico.

2.3 Processo de Desenvolvimento

O projeto foi desenvolvido de forma iterativa e incremental, que permite validar funcionalidades parciais ao longo do processo. Inicialmente foram implementados os módulos de cadastro e login, seguidos pela estrutura de gerenciamento de itens, pelas funções administrativas, e por fim pelas demais funcionalidades do sistema de acordo com a prioridade.

- Para **alunos**: cadastro/login, compra com crédito digital, empréstimo de itens, carrinho, edição de perfil (com saldo), notificações, etc.
- Para **administradores**: cadastro de novos itens, aprovação/rejeição de pedidos de empréstimo ou fundos, gestão de usuários, relatórios, etc.

2.4 Estrutura do Sistema

O sistema foi projetado em três camadas principais:

1. Camada de Apresentação (Frontend)
    - Responsável pela interface do usuário.
    - Desenvolvida em Ionic React, possibilita navegação responsiva em desktop e dispositivos móveis.
    - Inclui páginas de login, cadastro, perfil do aluno, tela do administrador e catálogo de ferramentas.
    - Se comunica com a API do backend via requisições HTTP (REST).
2. Camada de Lógica de Negócio (Backend)
    - Implementada em Node.js com Express.js.
    - Gerencia regras de negócio, autenticação, registro de empréstimos, controle de saldo virtual e envio de notificações.
    - Utiliza JWT (JSON Web Token) para autenticação segura.
    - Faz a validação de dados antes de interagir com o banco de dados.
3. Camada de Dados (Banco de Dados)
    - Utiliza MySQL para armazenar informações persistentes.

3 ANÁLISE E PROJETO DO SISTEMA

Este capítulo apresenta o projeto técnico do sistema proposto, que inclui a definição dos requisitos funcionais e não funcionais, a modelagem de casos de uso, a modelagem de dados e a arquitetura adotada. O objetivo é detalhar a estrutura lógica que norteou a implementação, que garante clareza e consistência entre o planejamento e o desenvolvimento realizado.

3.1 Requisitos Funcionais

Os requisitos funcionais (Tabela 1) representam as ações e serviços que o sistema deve oferecer para atender às necessidades dos usuários (alunos e administradores).

Tabela 1 - Requisitos Funcionais

ID	Requisito	Atores Envolvidos	Prioridade
RF01	Cadastro de usuários com dados pessoais e senha	Aluno	Alta
RF02	Validação de e-mail após cadastro	Aluno	Média
RF03	Login com autenticação de credenciais	Aluno, Admin	Alta
RF04	Redefinição de senha por e-mail	Aluno, Admin	Alta
RF05	Controle de permissões (Aluno/Admin)	Aluno, Admin	Alta
RF06	Cadastro de item com imagem, descrição, estado, quantidade, preço	Admin	Alta
RF07	Edição de item	Admin	Média
RF08	Remoção de item	Admin	Alta
RF09	Visualização de itens disponíveis	Aluno, Admin	Alta
RF10	Solicitação de empréstimo de itens	Aluno	Alta
RF11	Confirmar devolução de empréstimo	Admin	Alta
RF12	Definição de quantidade e visualização da data de devolução para empréstimos	Aluno	Média
RF13	Aprovação ou rejeição de empréstimos	Admin	Alta
RF14	Consulta de histórico de empréstimos	Aluno, Admin	Alta
RF15	Upload opcional de foto na devolução	Admin	Baixa
RF16	Avaliação do estado da ferramenta devolvida	Admin	Alta

3.2 Requisitos Não Funcionais

Os requisitos não funcionais (Tabela 2) asseguram qualidade, segurança e usabilidade ao sistema.

Tabela 2 - Requisitos Não-Funcionais

ID	Categoria	Descrição
RNF01	Segurança	Uso de JWT para autenticação e proteção de rotas
RNF02	Segurança	Criptografia de senhas com bcrypt
RNF03	Segurança	Conexão segura via HTTPS com domínio próprio
RNF04	Usabilidade	Interface responsiva para diferentes dispositivos
RNF05	Desempenho	Respostas rápidas em ações comuns do sistema
RNF06	Confiabilidade	Persistência de dados via banco de dados relacional (MySQL)
RNF07	Escalabilidade	Código organizado para possível migração futura para microserviços
RNF08	Automatização	Uso de cron jobs (`node-cron`) para tarefas agendadas
RNF09	Comunicação	Envio de e-mails automatizados (ex: redefinição de senha, notificações)
RNF10	Compatibilidade	Suporte aos dispositivos móveis

3.3 Modelagem do Sistema

O sistema foi modelado tendo em consideração os papéis de **aluno** e **administrador**.

- **Aluno** pode:
  - cadastrar-se e autenticar-se;
  - consultar itens disponíveis;
  - solicitar empréstimos e realizar devoluções;
  - efetuar compras;
  - solicitar recarga de saldo;
  - visualizar histórico de empréstimos e compras;
  - editar dados pessoas (e-mail, celular, senha).
- **Administrador** pode:
  - gerenciar cadastros de alunos;
  - controlar o estoque (adicionar, editar ou remover itens);
  - aprovar ou recusar solicitações de empréstimo e de recarga de saldo;

- gerar relatórios de movimentação.

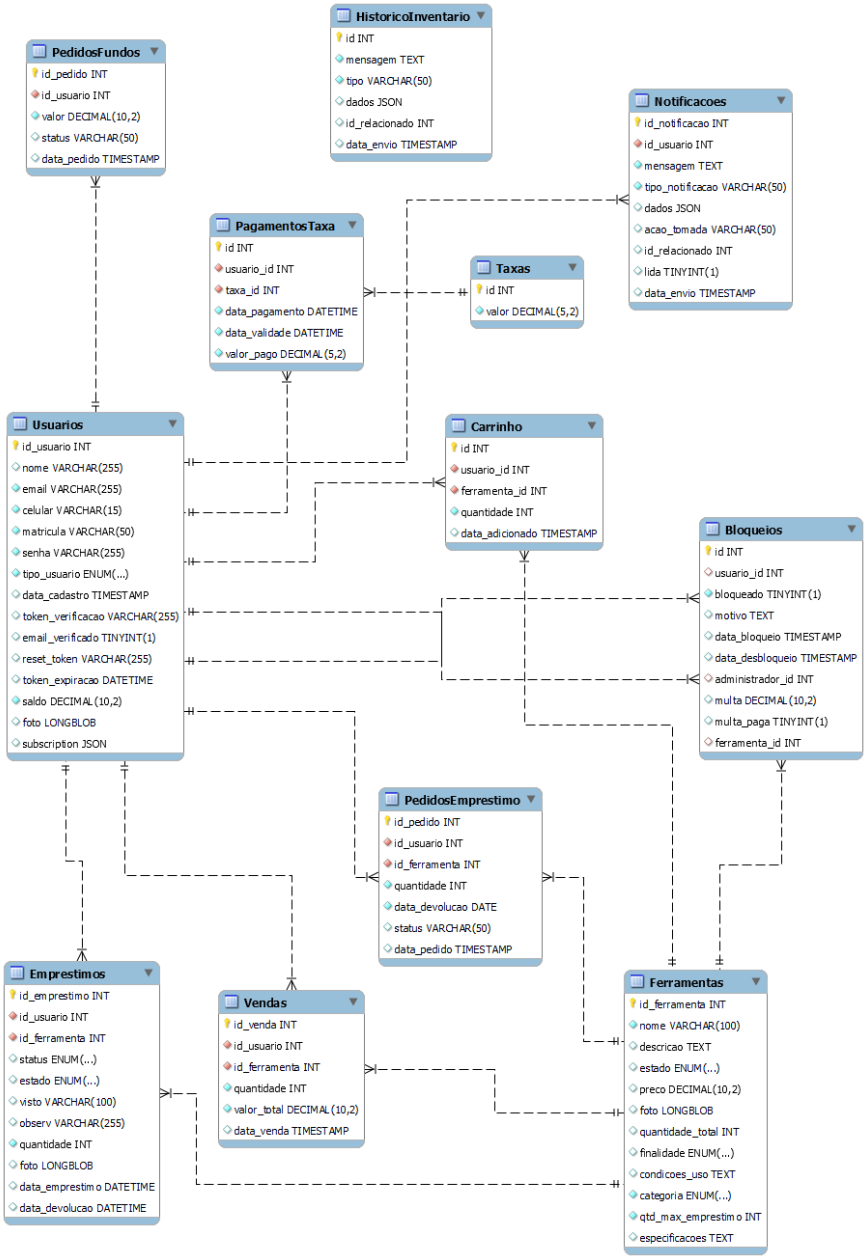
3.4 Modelagem de Dados

A base de dados foi estruturada em tabelas principais que representam os elementos centrais do sistema:

- **Usuários**: guarda informações pessoais, credenciais e saldo virtual.
- **Ferramentas**: armazena dados dos equipamentos, que incluem nome, quantidade e disponibilidade (empréstimo ou venda).
- **Emprestimos:** Registra os empréstimos realizados pelos usuários.
- **PedidosEmprestimo:** Controla os pedidos de solicitação de empréstimos antes da aprovação.
- **Carrinho:** Guarda temporariamente os itens que o usuário deseja adquirir ou emprestar.
- **Vendas:** Registra os itens adquiridos por compra direta com crédito.
- **PedidosFundos:** Armazena os pedidos de recarga de saldo de crédito realizados pelos usuários.
- **Notificacoes:** Salva o envio de alertas e mensagens automatizadas aos usuários, para informá-los sobre eventos importantes.

O Diagrama **Entidade-Relacionamento (DER)** completo, que ilustra as tabelas e suas relações, está disponível na (Figura 7).

Figura 7 - Diagrama Entidade Relacionamento



Fonte: Autoria Própria

As descrições detalhadas de cada tabela, que incluem campos, tipos de dados e restrições, encontram-se no APÊNDICE B – ESTRUTURAS DETALHADAS DAS TABELAS.

3.5 Arquitetura

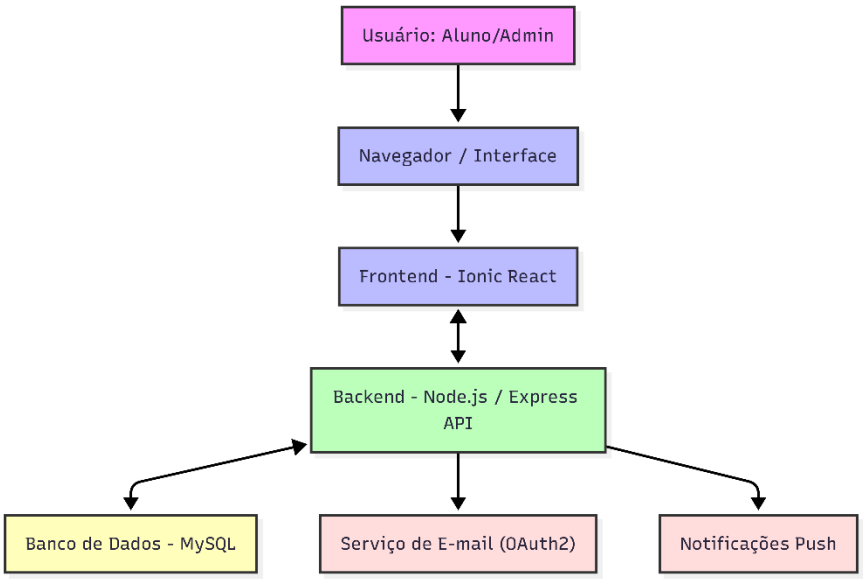
O sistema adota uma arquitetura **cliente-servidor** (Figura 8) baseada em três camadas:

1. **Frontend (Cliente)**
    - Desenvolvido em Ionic React.

- Responsável pela interface gráfica e interação do usuário.
- Comunicação com o backend via API REST.

2. **Backend (Servidor de Aplicação)**
   - Implementado em Node.js.
   - Processa as regras de negócio, autenticação e comunicação com o banco.
   - Utiliza JWT para autenticação segura.
3. **Banco de Dados (Persistência)**
   - Implementado em **MySQL**.
   - Armazena usuários, itens, transações e notificações.
   - Garante integridade e consistência dos dados.

Figura 8 - Diagrama da Arquitetura



Fonte: Autoria Própria

3.6 Protótipos de Tela

Antes de iniciar a implementação do sistema, foram desenvolvidos protótipos de tela com o objetivo de planejar visualmente a navegação e a disposição dos elementos principais da interface. Essa etapa foi essencial para alinhar os objetivos do projeto com as necessidades dos dois tipos de usuários*:* alunos e admin.

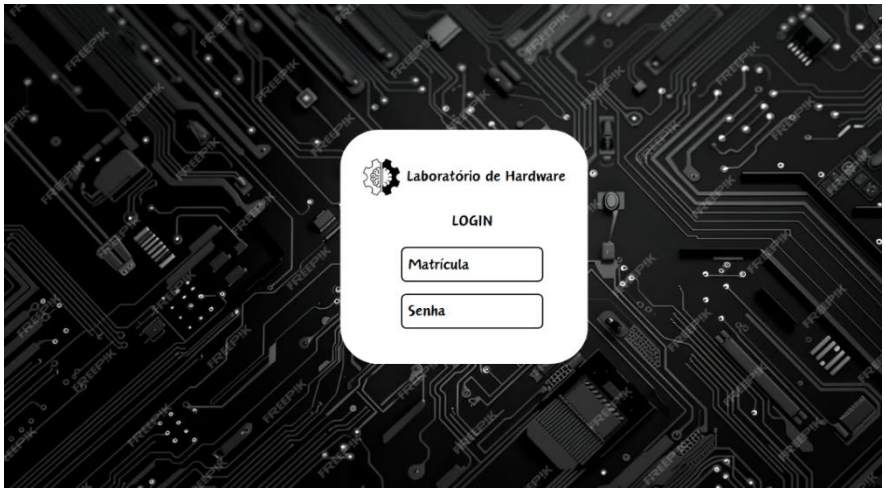
Foram criadas as telas principais que representam:

- **Login** e **Cadastro** de usuários (Figura 9 e Figura 10).

Tela principal do aluno, para consulta e solicitação de itens (Figura 11).

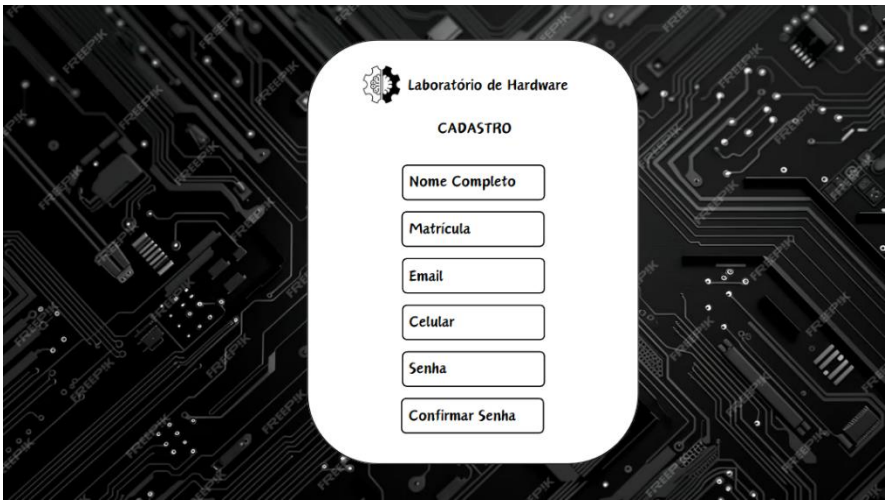
- **Tela do administrador**, com acesso ao gerenciamento de itens e usuários (Figura 12).

Figura 9 - Tela de login (protótipo inicial)



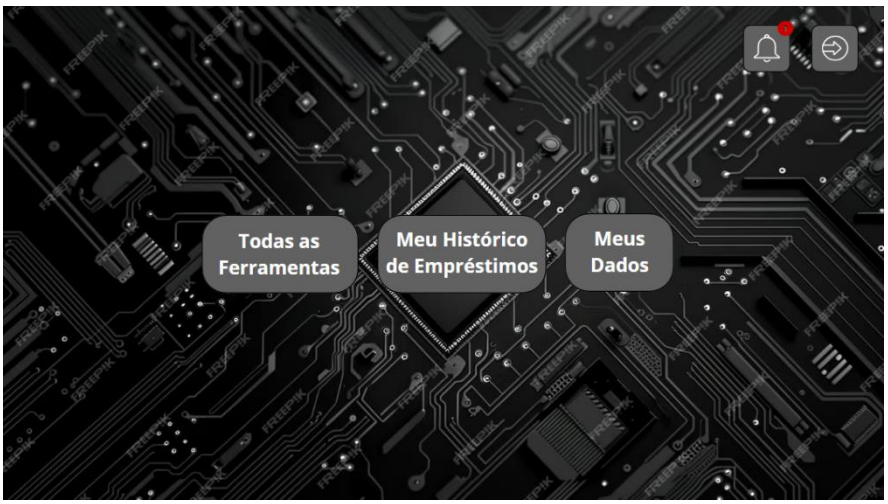
Fonte: Autoria Própria

Figura 10 - Tela de cadastro (protótipo inicial)



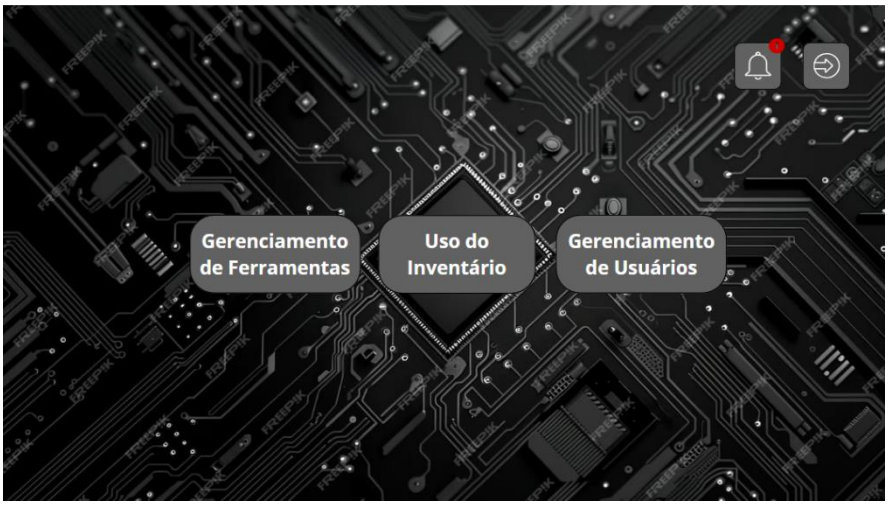
Fonte: Autoria Própria

Figura 11 - Tela principal do aluno (protótipo inicial)



Fonte: Autoria Própria

Figura 12 – Página principal do admin (protótipo inicial)



Fonte: Autoria Própria

Esses protótipos foram apresentados aos responsáveis pelo laboratório para validação antes da implementação. A partir do feedback, ajustes de layout e organização foram realizados, para garantir maior aderência às necessidades do usuário final.

Além dessas, outras telas foram desenvolvidas, como o gerenciamento de usuários, avaliação de devoluções, estatísticas administrativas e histórico de inventário, que totalizam 18 protótipos. Muitas dessas telas evoluíram ao longo do desenvolvimento, e algumas mudanças ocorreram devido a necessidades práticas de usabilidade. As demais telas estão listadas no APÊNDICE A – PROTÓTIPOS DE TELA**,** junto com suas respectivas figuras.

4 IMPLEMENTAÇÃO

A implementação do sistema consistiu em transformar a arquitetura planejada (descrita no Capítulo 3) em uma aplicação funcional, que incorpora as funcionalidades priorizadas durante o projeto. O desenvolvimento foi conduzido de forma incremental, com integração contínua entre frontend, backend e banco de dados, para permitir a validação prática das soluções propostas. Além disso, ajustes e melhorias foram aplicados ao longo do processo para atender às necessidades identificadas durante os testes de usabilidade e viabilidade técnica.

Na sequência, são apresentados os principais aspectos da implementação, a começar pela arquitetura concretizada e, posteriormente, os detalhes das funcionalidades desenvolvidas, os casos de uso efetivamente implementados, bem como os desafios enfrentados e as soluções adotadas.

4.1 Arquitetura Implementada

A implementação do sistema foi realizada de forma incremental a partir da arquitetura em camadas definida na fase de projeto, composta por frontend (Ionic React), backend (Node.js/Express) e banco de dados (MySQL).

No frontend, foram construídas interfaces responsivas que seguem os protótipos definidos no Canva. O uso do Ionic permitiu a criação de telas adaptadas tanto para dispositivos móveis quanto para desktop, afim de garantir uma boa experiência do usuário.

No backend, foi desenvolvida uma API REST responsável por gerenciar autenticação (JWT), transações de empréstimo e compra, envio de notificações por e-mail e push, além de verificações automáticas de prazos.

O banco de dados MySQL armazena todas as informações de usuários, itens, transações e notificações, para garantir integridade e consistência dos dados. As demais funcionalidades foram construídas de acordo com sua prioridade e relevância percebida para o uso prático do sistema.

4.2 Casos de Uso Implementados

Para validar a implementação e demonstrar as principais funcionalidades do sistema, foram definidos e detalhados casos de uso que representam as interações mais relevantes entre os usuários (alunos e administradores) e a aplicação. Esses casos de uso descrevem os objetivos

de cada ação, os fluxos principais e alternativos, bem como as condições necessárias para sua execução.

O objetivo dessa seção é evidenciar como os requisitos levantados no projeto foram traduzidos em operações práticas dentro da plataforma, afim de assegurar que o sistema atende às demandas de usabilidade, controle e segurança propostas inicialmente

4.2.1 Cadastro de Usuário

Ator Principal: Aluno

Objetivo: Realizar o cadastro no sistema para acessar funcionalidades.

Pré-condição:

- O usuário deve ter um e-mail válido.

Fluxo Principal:

1. O aluno acessa a tela de cadastro.
2. O sistema solicita dados como nome, matrícula, e-mail, celular e senha.
3. O aluno preenche as informações e confirma.
4. O sistema envia um e-mail de verificação.
5. O aluno confirma o e-mail e passa a ter acesso ao sistema.

Fluxo Alternativo:

- Se o e-mail já estiver cadastrado, o sistema exibe uma mensagem de erro.

4.2.2 Login no Sistema

Ator Principal: Aluno / Administrador

Objetivo: Permitir que o usuário acesse a plataforma.

Pré-condições:

- O usuário deve estar cadastrado e com o e-mail verificado.

Fluxo Principal:

1. O usuário acessa a tela de login.
2. O sistema solicita matrícula/e-mail e senha.
3. O usuário insere as credenciais.
4. O sistema valida os dados e cria um token JWT de autenticação.
5. O usuário é direcionado para a tela inicial correspondente ao seu perfil (Aluno/Admin).

Fluxo Alternativo:

- Caso a senha esteja incorreta, o sistema exibe mensagem de erro.

4.2.3 Solicitar Empréstimo

Ator Principal: Aluno

Objetivo: Solicitar empréstimo de ferramentas.

Pré-condições:

- O usuário deve estar autenticado.
- Não pode estar bloqueado ou com pendências.

Fluxo Principal:

1. O aluno acessa a listagem de itens disponíveis.
2. Seleciona a ferramenta desejada e a quantidade.
3. O sistema calcula a data prevista de devolução.
4. O aluno confirma a solicitação.
5. O sistema registra o pedido e envia notificação ao administrador.

Fluxo Alternativo:

- Caso o item não esteja disponível em estoque, o sistema informa indisponibilidade.

4.2.4 Aprovar/Rejeitar Solicitação de Empréstimo

Ator Principal: Administrador

Objetivo: Validar solicitações de empréstimo.

Pré-condição:

- O aluno deve ter enviado uma solicitação de empréstimo.

Fluxo Principal:

1. O administrador acessa a lista de solicitações pendentes.
2. Seleciona uma solicitação.
3. Avalia a disponibilidade do item e o histórico do aluno.
4. Aprova ou rejeita a solicitação.
5. O sistema envia notificação ao aluno com a decisão.

Fluxo Alternativo:

- Após 3 dias sem resposta a solicitação é automaticamente recusada.

4.2.5 Compra de Ferramentas/Componentes

Ator Principal: Aluno

Objetivo: Adquirir itens do laboratório com o uso do crédito digital.

Pré-condição:

- O aluno deve possuir saldo suficiente.

Fluxo Principal:

1. O aluno acessa a tela “Todos os Itens”.
2. Adiciona os itens desejados ao carrinho.
3. Confirma a compra.
4. O sistema valida saldo, atualiza estoque e debita o valor do crédito digital.
5. O sistema registra a transação no histórico.

Fluxo Alternativo:

- Se o saldo for insuficiente, o sistema bloqueia a compra e informa o usuário.

4.2.6 Notificações de Devolução

Ator Principal: Sistema

Objetivo: Avisar automaticamente o aluno sobre prazos de devolução.

Pré-condição:

- O aluno deve possuir itens emprestados com prazo ativo.

Fluxo Principal:

1. O cron job verifica diariamente os empréstimos em aberto.
2. Identifica itens com prazo de devolução próximo ou vencido.
3. Envia notificação push e/ou e-mail ao aluno.

Fluxo Alternativo:

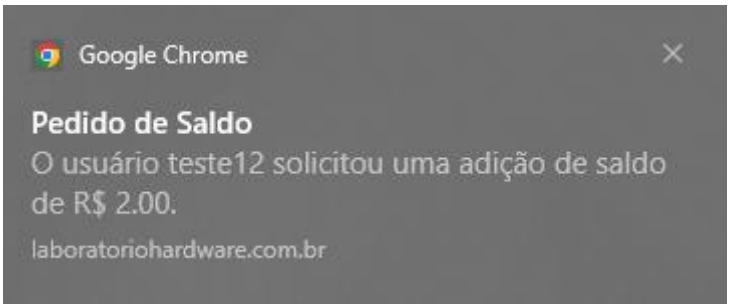
- Caso o aluno devolva antes do prazo, nenhuma notificação é enviada.

4.3 Dificuldades Técnicas e Soluções

Durante o desenvolvimento, diversas dificuldades técnicas foram enfrentadas e solucionadas. A implementação de notificações push (Figura 13), o acesso à câmera em diferentes dispositivos (webcam para desktop e câmera nativa em celulares/tablets), e a manipulação de blobs de imagem exigiram soluções específicas. A página de edição de itens,

que utiliza o mesmo formulário da criação, apresentou desafios relacionados ao carregamento e atualização correta dos dados pré-existentes.

Figura 13 - Exemplo de Notificação Push



Fonte: Autoria Própria

Outros pontos que demandaram atenção incluem:

- A utilização do componente *IonAccordionGroup* para organizar o menu administrativo, que apresentou problemas de responsividade inicialmente;
- A conversão de imagens armazenadas no banco (blobs) em formato base64 para exibição no frontend;
- A execução de tarefas agendadas com node-cron para verificação diária de atrasos;
- O envio automatizado de e-mails para confirmação de cadastro e avisos de devolução, com o nodemailer e autenticação OAuth2;
- A integração e configuração de HTTPS, bem como a publicação do frontend e backend no servidor com o auxílio do PM2[13], após tentativas frustradas em hospedagens convencionais como Vercel, Netlify, Railway.

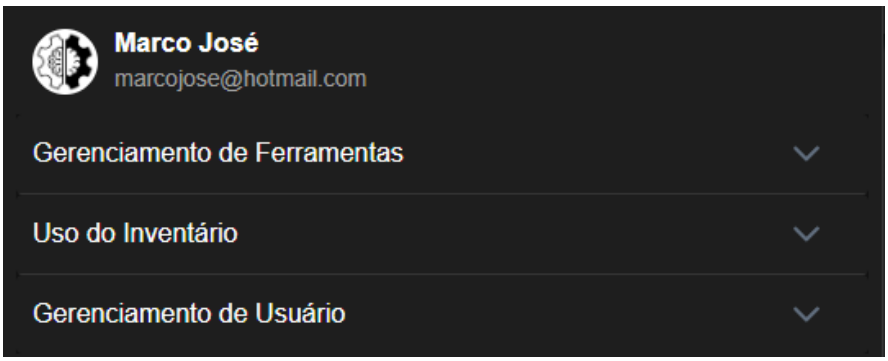
4.4 Evoluções em Relação ao Projeto Inicial

Durante o processo, diversos ajustes e melhorias foram realizados em relação aos protótipos iniciais. A navegação entre páginas foi substituída por um menu lateral personalizado para cada tipo de usuário, para tornar a experiência mais fluida. O menu lateral para *admin* é separado em grupos (Figura 14) e ao selecionar um deles revela as páginas relacionadas ao mesmo (Figura 15). Ao contrário do tipo *aluno* devido à quantidade menor de páginas (Figura 16). A ideia de criar dois carrinhos separados (um para compras e outro para empréstimos) foi descartada, manteve-se apenas o carrinho de compras; os empréstimos passaram a ser

[13] Gerenciador de processos Node.js que permitiu o monitoramento do uso de CPU, memória, latência de eventos e estabilidade da aplicação em execução no servidor (PM2, 2025).

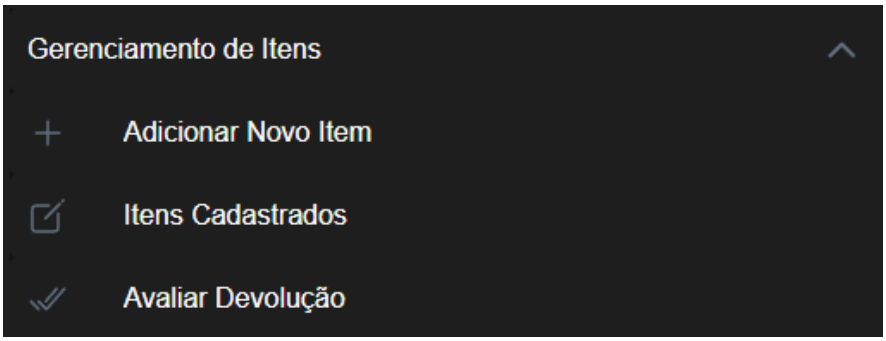
realizados item por item. A funcionalidade de listar ferramentas também foi renomeada para “Todos os Itens”, visto que o inventário abrange não apenas ferramentas, mas também componentes eletrônicos.

Figura 14 - Menu Lateral Admin Fechado (implementação final)



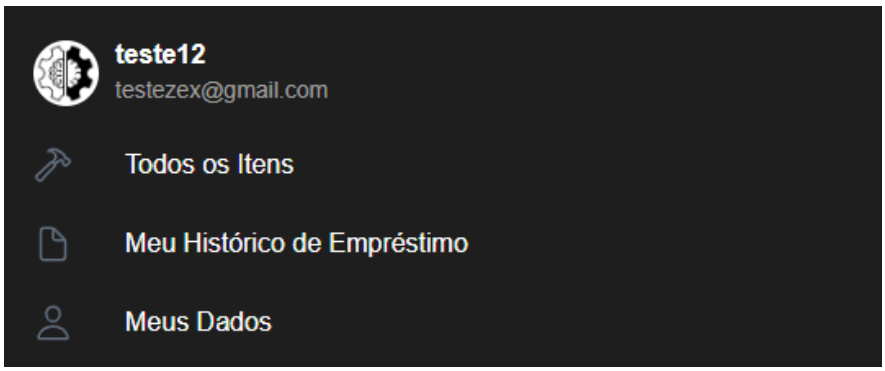
Fonte: Autoria Própria

Figura 15 - Menu Lateral Admin (implementação final uma aba aberta)



Fonte: Autoria Própria

Figura 16 - Menu Lateral Aluno (implementação final)



Fonte: Autoria Própria

Além das funcionalidades planejadas, outras foram incorporadas ao longo do desenvolvimento, como:

- Um ícone fixo de saldo virtual visível para o usuário;
- Uma tabela de bloqueios de usuários por atraso;
- Um sistema de assinatura mensal para habilitar empréstimos;

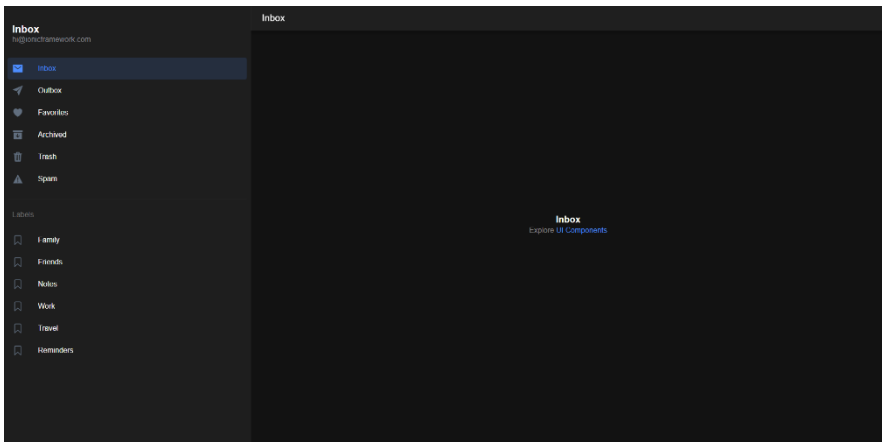
- Filtros e ordenações para facilitar a navegação;
- Exibição de foto de perfil do usuário;
- Consulta de histórico por item (permite ao administrador verificar quem já utilizou determinada ferramenta).

A estrutura do banco de dados também foi expandida durante a implementação. O projeto iniciou com seis tabelas principais (Usuarios, Ferramentas, Emprestimos, Vendas, Transacoes_Moedas e Notificacoes), mas ao longo do desenvolvimento foram adicionadas outras, como PedidosEmprestimo, PedidosFundos, Taxas e Bloqueios. A tabela Transacoes_Moedas foi removida por não ser mais necessária. Algumas dessas novas tabelas foram criadas para atender necessidades específicas que surgiram ao longo da implementação e representam melhorias ao projeto original.

O desenvolvimento foi realizado através do ambiente de testes ionic serve, que permite visualizar em tempo real as alterações feitas na interface. Isso possibilitou a realização de testes contínuos de usabilidade, responsividade e integração com o backend, com correções sendo feitas imediatamente após a identificação de falhas. Foram utilizados tokens JWT para autenticação, upload de arquivos via multer, service workers para notificações, e múltiplos recursos da API do navegador (como acesso à câmera e permissões).

Diversas bibliotecas externas e pacotes do NPM foram utilizados para viabilizar o projeto, que inclui axios (requisições HTTP), bcrypt (hash de senhas), jsonwebtoken, node-cron, nodemailer, multer, mysql2, react-toastify, react-router-dom, entre outros. A base para a estrutura da interface foi adaptada a partir de um exemplo oficial do Ionic com menu lateral (Figura 17).

Figura 17 - Modelo oficial do Ionic (página com menu lateral)



Fonte: Autoria Própria

Embora grande parte do desenvolvimento tenha sido baseada em tentativa e erro, a criação das regras de empréstimo e devolução foi inspirada nas normas do Serviço de Empréstimo da Biblioteca Pública do Paraná (BPP), sendo adaptadas à realidade do sistema. A documentação técnica oficial do Ionic Framework e do Node.js também foram referências complementares durante a implementação.

5 RESULTADOS

Nesta seção são apresentados os principais resultados obtidos com o desenvolvimento do sistema, afim de destacar suas funcionalidades, diferenciais técnicos e usabilidade.

5.1 Funcionalidades Implementadas

O sistema foi concluído com foco em dois perfis de usuários: *alunos* e *administradores*, cada um com recursos específicos.

Para Alunos:

- Solicitar empréstimos de itens disponíveis;
- Consultar histórico de empréstimos com informações de datas e status;
- Receber notificações automáticas de prazos (push e e-mail);
- Adicionar itens ao carrinho para múltiplos pedidos;
- Comprar itens com o saldo virtual;
- Visualizar saldo e taxa mensal de empréstimos acumulados;
- Navegar por uma interface intuitiva com ícones funcionais (notificação, carrinho, saldo, logout).

Para Administradores:

- Aprovar ou rejeitar solicitações de empréstimos e recargas de saldo;
- Cadastrar, editar, bloquear/desbloquear e remover itens;
- Acompanhar estoque em tempo real e identificar atrasos;
- Acessar painel administrativo com estatísticas e gráficos;
- Consultar histórico completo de qualquer item ou aluno;
- Enviar mensagens de lembrete para alunos em atraso;
- Gerenciar usuários e aplicar bloqueios quando necessário.

5.2 Diferenciais Técnicos

O sistema apresenta diversos recursos técnicos que o destacam de soluções básicas:

- **Sistema de notificações integrado**: envio automático via push e e-mail;
- **Verificação automática de atrasos**: cron job (node-cron) roda diariamente para identificar devoluções pendentes;
- **Segurança no envio de e-mails**: autenticação com OAuth2 via Gmail, para evitar exposição de credenciais;

- **Proteção da comunicação**: uso de HTTPS para todas as interações entre cliente e servidor.

5.3 Testes

Durante o desenvolvimento do sistema, foram realizados testes com foco em três aspectos principais: usabilidade, segurança e performance. Os testes buscaram garantir que o sistema fosse intuitivo para o usuário final, seguro em relação a acessos indevidos e eficiente mesmo em cenários com múltiplas requisições.

5.3.1 Testes de Usabilidade

O objetivo dos testes de usabilidade foi verificar se o sistema era fácil de usar e compreensível para o público-alvo.

- Foram realizados testes exploratórios com usuários não familiarizados com o sistema;
- Os participantes receberam tarefas específicas (ex: realizar um empréstimo, consultar disponibilidade);
- Foram observados tempo de execução, dificuldades encontradas e sugestões de melhoria.

Resultado: O feedback indicou que a interface era clara, mas foi necessário aumentar a visibilidade de algumas mensagens de confirmação.

5.3.2 Testes de Segurança

Para validar a segurança do sistema, foram verificados:

- Controle de acesso: testes manuais para garantir que usuários sem permissão não pudessem acessar rotas protegidas;
- Autenticação: validação do uso de tokens JWT;
- Testes de injeção de código simples nos campos de entrada para evitar vulnerabilidades.

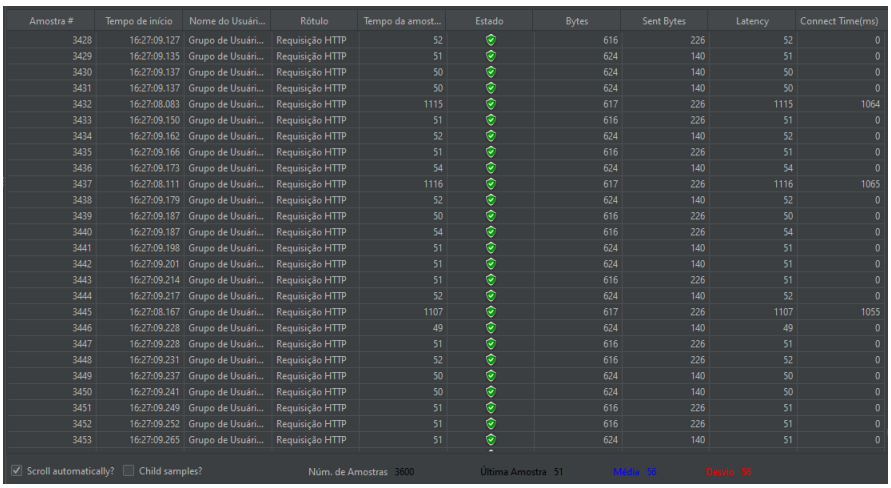
Resultado: O sistema respondeu adequadamente a tentativas de acesso indevido e campos inválidos foram tratados com validação.

5.3.3 Testes de Performance

O desempenho do sistema foi avaliado com foco nas rotas mais críticas, como as responsáveis por envio de notificações e manipulação do histórico do usuário.

Primeiramente, foram realizados testes de carga com o Apache JMeter, para simular múltiplos acessos simultâneos à aplicação (Figura 18). A configuração incluiu 180 usuários virtuais que realizam requisições a cada 5 segundos, que totalizam 3600 amostras. A média de tempo de resposta foi de aproximadamente 56 ms, com desvio padrão também de 56 ms, que indica um comportamento estável. A Tabela 3 resume os parâmetros utilizados e os principais resultados do teste.

Figura 18 - Simulação de Múltiplos Acessos



Fonte: Autoria Própria

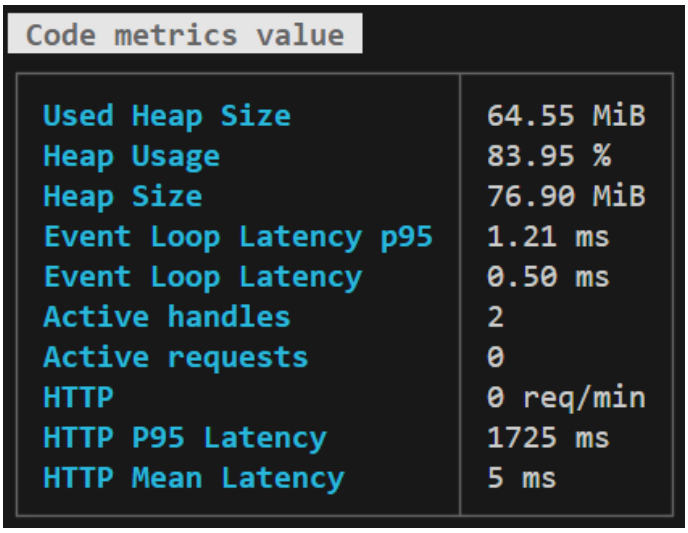
Tabela 3 - Teste de Carga com Apache JMeter

Parâmetro	Valor
Número de Usuários Virtuais	180 threads
Tempo de Inicialização	5 segundos
Número de Iterações	10
Total de Requisições	3600 amostras
Tempo da Última Requisição	51 ms
Tempo Médio de Resposta	56 ms
Desvio Padrão (latência)	56 ms

Em seguida, utilizou-se o PM2 para monitoramento em tempo real do uso de CPU, memória e latência da aplicação (Figura 19). Observou-se um uso de heap de aproximadamente 84%, com média de latência de event loop de 0.50 ms. Apesar da média de latência HTTP ser

de 5 ms, o valor P95[14] chegou a 1725 ms, que evidencia a presença de alguns picos sob carga intensa. A Tabela 4 apresenta os principais dados coletados.

Figura 19 - Valores Medidos (PM2)



Fonte: Autoria Própria

Tabela 4 - Métricas do Servidor

Métrica	Valor
Heap Size Alocada	76.90 MiB
Heap Size Utilizada	64.55 MiB
Uso de Heap (%)	83.95%
Latência da Event Loop (Média)	0.50 ms
Event Loop P95	1.21 ms
Latência HTTP Média	5 ms
Latência HTTP P95	1725 ms
Requisições HTTP por Minuto	0 (em idle)

Resultado: De forma geral, o sistema manteve um tempo de resposta aceitável (**< 150 ms**) até **180 requisições simultâneas**. Acima desse ponto, foram identificados sinais de lentidão em algumas amostras, o que indica um limite de escalabilidade atual, que pode ser otimizado em versões futuras com melhorias na infraestrutura ou ajustes no código.

5.3.4 Ferramentas

Durante o desenvolvimento e a validação do sistema, diversas ferramentas foram utilizadas para facilitar a programação, organização, testes e análise de desempenho:

[14] Medida estatística que indica que 95% das requisições foram respondidas em um tempo igual ou inferior a 1725 ms. Os 5% mais lentos (geralmente sob condições extremas de carga) levaram mais tempo que isso, portanto isso explica a grande diferença em relação à média (5 ms). É uma métrica crucial para entender a experiência real dos usuários.

- **Visual Studio Code (VS Code)**: editor de código-fonte utilizado como ambiente principal de desenvolvimento. Suporta extensões que facilitaram a escrita de código JavaScript, integração com o GitHub e execução de testes locais (MICROSOFT, 2025);
- **MySQL Workbench**: ferramenta gráfica para modelagem, administração e execução de consultas no banco de dados MySQL, usada para criação e verificação das estruturas de tabelas e dados persistentes do sistema (ORACLE, 2025);
- **Ferramentas de desenvolvedor do navegador (DevTools)**: utilizadas para testes das rotas da API, especialmente através do console, que permitem a visualização de requisições HTTP, respostas do servidor e mensagens de erro, enquanto o servidor do Ionic estava ativo (GOOGLE, 2025);
- **GitHub**: plataforma de versionamento usada para armazenar, documentar e controlar o histórico de desenvolvimento do projeto, além de facilitar a atualização dos arquivos com sua conexão com o VS Code (GITHUB, 2025).
- **Apache JMeter**: ferramenta de simulação de carga utilizada para realizar testes de desempenho, afim de medir o tempo de resposta sob múltiplas requisições simultâneas (APACHE SOFTWARE FOUNDATION, 2025);
- **Canva**: ferramenta gráfica usada para a criação de diagramas, protótipos de telas e imagens explicativas utilizadas na apresentação visual do projeto (CANVA, 2025).

Essas ferramentas foram essenciais para garantir qualidade no desenvolvimento, facilitar a depuração de erros, realizar testes de forma sistemática e produzir materiais de apoio visuais.

5.4 Telas do Sistema

O sistema conta com uma interface moderna e intuitiva, projetada para facilitar o uso tanto por alunos quanto por administradores. A seguir, são apresentados prints das principais telas do sistema nas versões mobile e desktop, com breves descrições de sua usabilidade.

5.4.1 Telas de Login

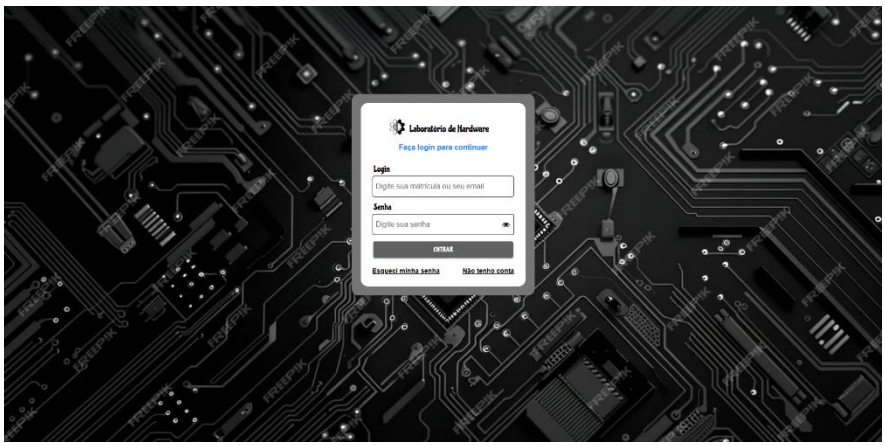
A tela de Login (Figura 20 e Figura 21) foi implementada com foco em segurança e usabilidade. Diferente de um simples formulário, ela incorpora mecanismos de validação de

entradas para evitar injeções de código (SQL/JavaScript) através do uso de expressões regulares e sanitização antes do envio ao backend.

Após a submissão, os dados são processados pela API, que retorna um token JWT. Esse token é armazenado em localStorage e decodificado no frontend para diferenciar automaticamente o tipo de usuário, para redirecionar o aluno para sua página inicial e o administrador para o painel de controle.

Foram adicionados ainda recursos de usabilidade: exibição/ocultação da senha digitada, mensagens claras de erro em caso de credenciais incorretas ou e-mail não verificado, e links rápidos para recuperação de senha e criação de nova conta. Esses cuidados visam equilibrar segurança e praticidade para o usuário final.

Figura 20 - Tela de Login (desktop)



Fonte: Autoria Própria

Figura 21 - Tela de Login (mobile)

Fonte: Autoria Própria

5.4.2 Tela de Cadastro

A tela de Cadastro (Figura 22 e Figura 23) foi implementada com foco em segurança, validação de dados e experiência do usuário.

como fraca, média ou forte de acordo com critérios como uso de letras maiúsculas, minúsculas, números e caracteres especiais. Além do layout responsivo provido pelo Ionic, foram adicionadas diversas camadas de validação tanto no frontend quanto na comunicação com o backend.

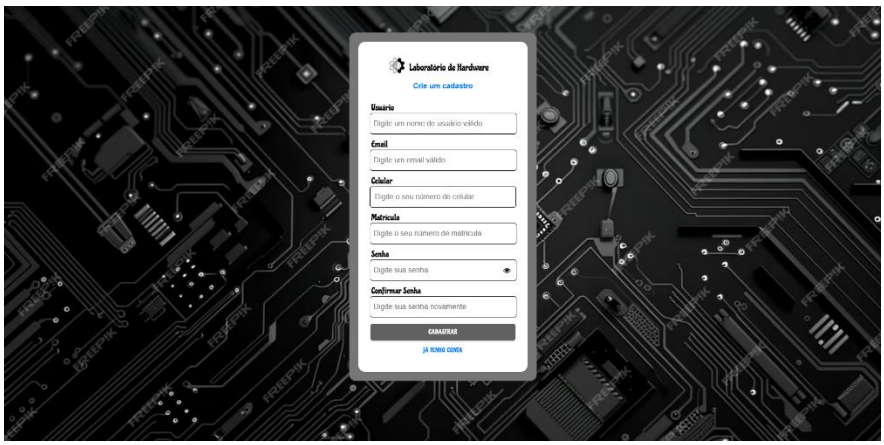
Do ponto de vista técnico, a tela incorpora:

- **Validação em tempo real** dos campos de entrada (nome, e-mail, celular, matrícula e senha), com mensagens de erro exibidas de forma dinâmica.
- **Máscara de entrada para celular**, com a biblioteca ReactInputMask, para garantir que o dado seja digitado no formato esperado.
- **Verificação da força da senha**, classificada
- **Sanitização de dados e prevenção contra injeções**, por meio de funções que detectam padrões maliciosos (como SQL Injection ou scripts) antes de enviar os dados ao backend.
- **Feedback imediato ao usuário** através da biblioteca react-toastify, que exibe notificações visuais que confirmam sucesso ou informam falhas durante o processo.
- **Proteção contra falhas do backend**, com tratamento explícito de erros e exibição de mensagens claras em caso de inconsistências na resposta da API.

Após a validação e sanitização, os dados são enviados via requisição POST para o endpoint /api/cadastro. Em caso de sucesso, o usuário recebe a instrução para verificar o e-mail de confirmação e é redirecionado automaticamente para a tela de login.

Esse conjunto de técnicas garante não apenas a usabilidade da tela, mas também um nível adicional de segurança, essencial em sistemas que lidam com dados pessoais e credenciais.

Figura 22 - Tela de Cadastro (desktop)



Fonte: Autoria Própria

Figura 23 - Tela de Cadastro (mobile)

Fonte: Autoria Própria

5.4.3 Tela Inicial Aluno

O corpo da página é dividido em três seções A tela inicial do sistema (Figura 24 e Figura 25) foi desenvolvida para ser o ponto central de navegação, ela combina usabilidade e acesso rápido às principais funcionalidades do sistema.

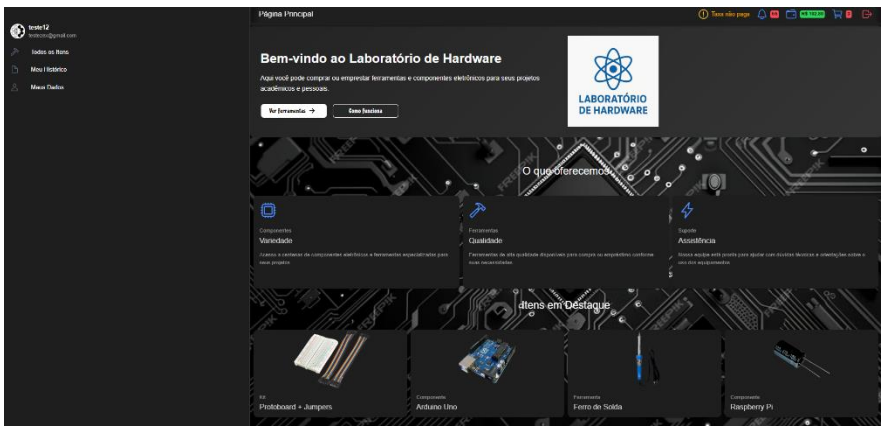
No cabeçalho superior, além do menu lateral (implementado via IonMenuButton), são exibidos ícones funcionais como: status de bloqueio (BlockedStatus), taxa acumulada (TaxaStatus), notificações (NotificationIcon), saldo virtual (SaldoIcon), carrinho de compras (CartButton) e logout (LogoutButton). Esses componentes reutilizáveis foram projetados para fornecer ao aluno informações imediatas sobre sua situação no sistema, sem a necessidade de navegar para outras páginas.

principais:

- **Banner de boas-vindas**: apresenta uma mensagem introdutória e botões de ação rápidos, implementados com IonButton, que permitem acesso imediato às páginas de itens disponíveis e à seção "Como Funciona". A disposição do banner se adapta entre colunas (desktop) e formato vertical (mobile), que garante a responsividade.
- **Seção de destaques**: utiliza IonCard para representar visualmente os principais diferenciais do laboratório (componentes, ferramentas e suporte). Cada card é acompanhado de um ícone representativo do pacote ionicons, para reforçar a clareza visual.
- **Itens em destaque**: exibe exemplos de ferramentas e componentes disponíveis (como kits de prototipagem, Arduino, ferro de solda e capacitores). Cada item é carregado em um IonCard clicável que redireciona o usuário à respectiva página de detalhes por meio do routerLink.

O design responsivo é alcançado com IonGrid, IonRow e IonCol, que organizam o layout em colunas no desktop e em blocos verticais no mobile. Essa estrutura, combinada ao uso de imagens (IonImg) e ícones, torna a tela inicial visual, informativa e de fácil navegação.

Figura 24 - Tela Inicial do Aluno (desktop)



Fonte: Autoria Própria

Figura 25 - Tela Inicial do Aluno (mobile)

Fonte: Autoria Própria

5.4.4 Tela: Perfil do Aluno

A tela de Perfil do Aluno (Figura 26 e Figura 27) foi projetada para concentrar as informações pessoais e funcionalidades de autogestão do usuário dentro do sistema. Sua implementação priorizou segurança dos dados, usabilidade e integração direta com o backend.

O cabeçalho segue o mesmo padrão visual de consistência adotado em todo o sistema, para trazer os ícones funcionais (BlockedStatus, TaxaStatus, NotificationIcon, SaldoIcon, CartButton e LogoutButton), acessíveis em qualquer página.

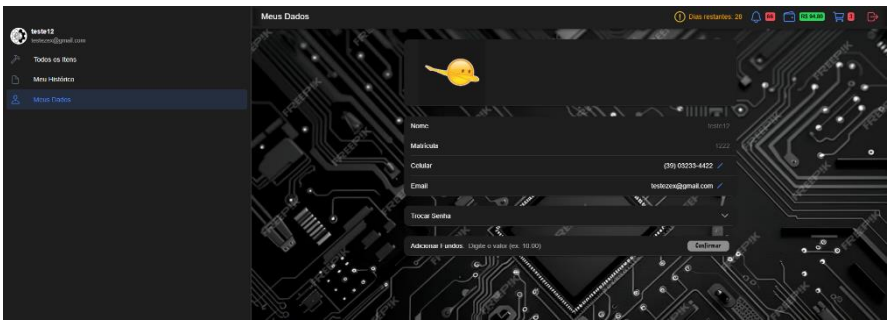
Entre os principais recursos implementados destacam-se:

- **Gerenciamento de dados pessoais**: Nome e matrícula são exibidos de forma apenas informativa, enquanto e-mail e celular podem ser editados. Essa edição é controlada por editMode e tempData, que garantem que apenas campos selecionados sejam temporariamente alterados antes de envio. O backend recebe as atualizações via requisições PUT, sempre precedidas de validação contra injeção de código malicioso.
- **Alteração de foto de perfil**: O usuário pode atualizar sua imagem clicando sobre a foto exibida. A implementação inclui seleção via input file com pré-visualização (FileReader) e validações de tipo e tamanho (somente imagens de até 5 MB). A foto é enviada ao backend com autenticação via token JWT, e feedback imediato é fornecido por Toast.

- **Troca de senha**: O fluxo de alteração de senha exige a confirmação da senha atual antes da definição da nova, validada pelo backend com endpoint dedicado. Foram adicionados mecanismos de segurança como detecção de padrões de injeção e máscara para exibição/ocultação de senhas. Em caso de falhas (senha incorreta, campos divergentes), o usuário recebe retorno imediato por Toast ou Note.
- **Gerenciamento de saldo virtual**: A tela permite solicitar créditos para a carteira digital, utilizada em compras. O valor digitado passa por validação de formato e envio ao backend, que gera uma solicitação vinculada ao usuário. Caso validada, são exibidas as instruções de pagamento via Pix e prazos para atualização do saldo.
- **Camada de segurança contra ataques**: Todas as entradas de dados (nome, e-mail, celular, senha, crédito) passam por verificações de padrões de injeção (hasInjection) e sanitização (sanitizeInput), que previnem SQL Injection, XSS e path traversal.

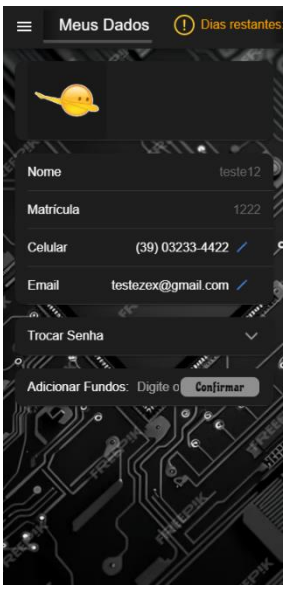
Assim, a tela de perfil combina **funcionalidades administrativas pessoais** (alterar senha, atualizar dados, solicitar fundos) com controles de segurança, que oferecem ao aluno autonomia sobre seu cadastro e ao mesmo tempo para preservar a integridade do sistema.

Figura 26 - Tela Perfil do Aluno (desktop)



Fonte: Autoria Própria

Figura 27 - Tela Perfil do Aluno (mobile)



Fonte: Autoria Própria

5.4.5 Tela de Empréstimo

A Tela de Empréstimo (Figura 28 e Figura 29) foi implementada como ponto crítico do sistema, pois consolida regras de negócio, verificações de segurança e interação direta com o backend para efetivar a solicitação de retirada de itens.

Do ponto de vista técnico, a construção incluiu os seguintes aspectos:

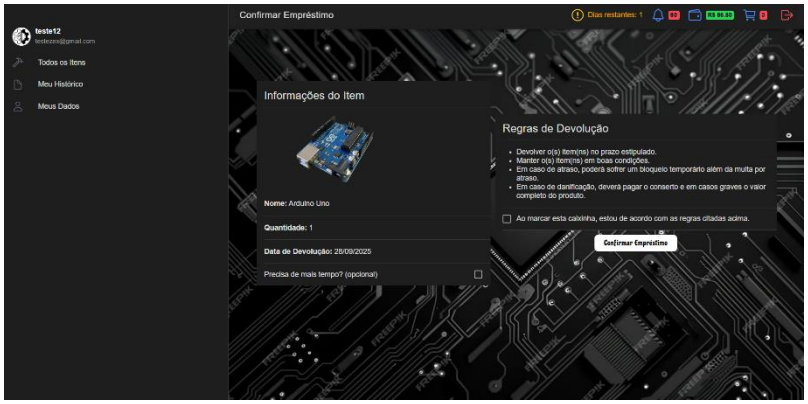
- **Carregamento dinâmico dos dados do item**: Ao acessar a tela, o sistema busca os detalhes do item selecionado via requisição GET /api/ferramenta/:id e exibe informações como foto, nome e quantidade solicitada. Esse processo é acompanhado de tratamento de estados (loading e error) com IonSpinner e mensagens de fallback em caso de falha na comunicação.
- **Cálculo da data de devolução**: Por padrão, a devolução é definida para 7 dias após a retirada. O usuário pode solicitar dias adicionais (até 7 extras), o que recalcula a data de devolução automaticamente por meio da função calcularDataDevolucao, formatada com a biblioteca moment.
- **Regras de devolução com aceite obrigatório**: O fluxo de confirmação só é liberado se o usuário marcar explicitamente o checkbox de aceite das regras. Isso garante concordância explícita com condições como devolução no prazo, conservação do item e penalidades em caso de atraso ou dano.
- **Integração com bloqueio e taxa mensal**: Antes da liberação, o sistema verifica via backend se o usuário está bloqueado (/api/usuario/:id/status-bloqueio) ou se

a taxa mensal não está em dia (*/api/taxa-status/:id*). Caso qualquer condição esteja pendente, a interface exibe a mensagem “Função Bloqueada”, e impede a solicitação.

- **Confirmação de empréstimo**: Após validações, o botão dispara a requisição POST /api/send-notif-emprestimo, que envia ao backend os dados do empréstimo (usuário, item, quantidade, data de devolução). Além de registrar a transação, o backend dispara notificações automáticas. O sucesso é comunicado ao usuário por meio de IonToast e redirecionamento automático para a página inicial.
- **Segurança e consistência**: Todas as operações de empréstimo exigem autenticação via token JWT armazenado em localStorage, para garantir que apenas usuários logados possam realizar solicitações.

Essa tela é, portanto, um exemplo de implementação completa de regra de negócio dentro da aplicação, pois conecta a interface, as regras de uso e o backend, para oferecer ao aluno uma experiência intuitiva, mas com rígido controle administrativo.

Figura 28 - Tela de Empréstimo (desktop)



Fonte: Autoria Própria

Figura 29 - Tela de Empréstimo (mobile)

Fonte: Autoria Própria

5.4.6 Tela de Histórico do Aluno

A tela de histórico do aluno (Figura 30 e Figura 31) foi desenvolvida para consolidar em um único ponto todas as transações realizadas pelo aluno; tanto empréstimos quanto compras. O objetivo principal é fornecer transparência e rastreabilidade das interações do usuário com o sistema, ao mesmo tempo em que possibilita filtros e buscas dinâmicas.

Do ponto de vista técnico, sua implementação envolveu os seguintes aspectos:

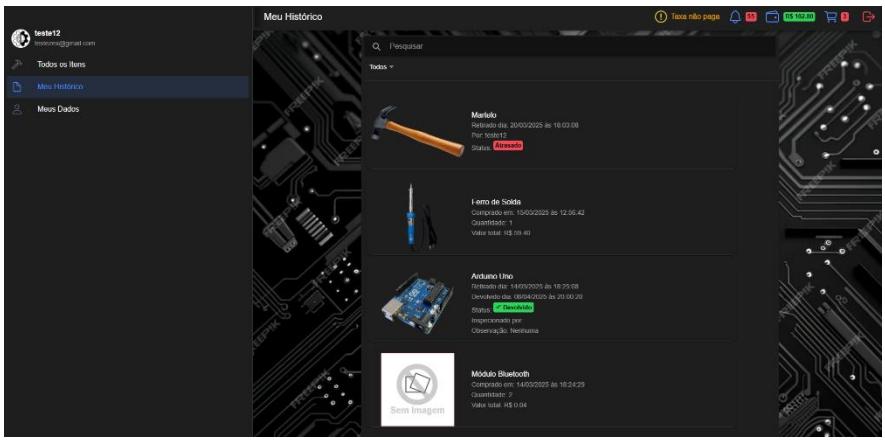
- **Integração com múltiplas fontes de dados**: O sistema realiza, de forma paralela (Promise.allSettled), requisições ao backend para recuperar os empréstimos (/api/loans) e compras (/api/compras). Os resultados são armazenados em arrays distintos (loans e compras), que posteriormente são unificados para renderização conjunta.
- **Unificação e ordenação cronológica**: Após carregar os dados, os registros de empréstimos e compras são mesclados em um único array. Cada item recebe um campo de data padronizado (data_evento) para permitir ordenação uniforme, e garantir que as transações sejam exibidas do mais recente para o mais antigo, independentemente do tipo.
- **Sistema de filtros e busca**: A interface inclui uma barra de pesquisa (IonSearchbar) e um seletor de filtros (IonSelect), que possibilitam refinar os resultados de acordo com o nome do item, tipo de transação (empréstimo ou compra) ou status específico (devolvido, pendente, atrasado). Essa abordagem

melhora a **usabilidade** ao permitir que o usuário encontre rapidamente informações relevantes.

- **Tratamento de status e usabilidade visual**: Para os empréstimos, o campo status é mapeado em badges visuais (IonBadge) com ícones distintos, como alerta para pendências, verde para devoluções e vermelho para itens atrasados ou danificados. Essa padronização facilita a **interpretação rápida** da situação de cada item.
- **Detalhamento contextualizado**: Os empréstimos exibem informações como datas de retirada e devolução, inspetor responsável (visto) e observações registradas pelo administrador. Já as compras mostram quantidade adquirida, valor total e data da transação.
- **Gestão de estados de carregamento e falhas**: Foram incluídos indicadores de carregamento (IonSpinner), mensagens de erro com botão de "Tentar novamente" e mensagens de vazio ("Histórico não encontrado"). Essa estratégia garante **resiliência e clareza** na interação, mesmo em cenários de falha de rede.
- **Segurança e autenticação**: Todas as requisições ao backend utilizam JWT armazenado em localStorage como cabeçalho de autenticação (Authorization: Bearer token), que assegura que apenas usuários autenticados possam acessar seus históricos.

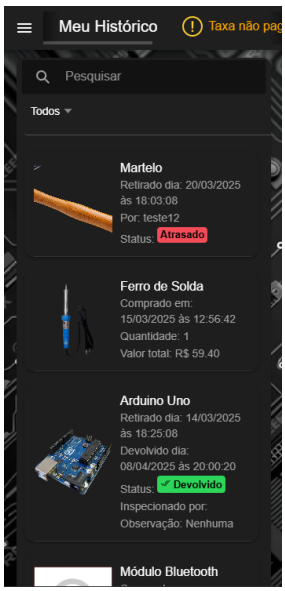
Assim, a tela de histórico não é apenas um repositório estático de registros, mas sim um painel dinâmico e interativo, que oferece visão consolidada das atividades do aluno com alto nível de usabilidade e segurança.

Figura 30 - Tela de Histórico de Aluno (desktop)



Fonte: Autoria Própria

Figura 31 - Tela de Histórico de Aluno (mobile)



Fonte: Autoria Própria

5.4.7 Tela de Adição de Item

A tela de Adição de Item (Figura 32 e Figura 33) foi projetada para que administradores possam **registrar novos materiais e ferramentas** no sistema, e assim garantir a correta catalogação e organização do acervo. As principais características são:

- **Formulário completo de cadastro**

  A interface utiliza múltiplos campos (IonInput, IonTextarea, IonSelect) para coletar dados do item, tais que:

  - Nome e descrição detalhada
  - Estado de conservação (bom, regular, danificado, indisponível)
  - Categoria (ferramentas elétricas, manuais, componentes eletrônicos etc.)
  - Finalidade (empréstimo ou venda)
  - Quantidade total disponível e limite máximo por empréstimo
  - Preço unitário, formatado em moeda brasileira

- **Upload e captura de imagens**

  A tela disponibiliza duas opções para associar uma foto ao item:

  - **Captura por câmera** (CameraComponent) quando o sistema detecta acesso via dispositivo móvel.
  - **Seleção de arquivo local** através de input de imagem.

  Ambas as opções incluem validação de formato (apenas imagens) e limite de tamanho (máximo 5MB) para evitar sobrecarga do sistema.

- **Segurança contra injeções**

  Antes do envio dos dados, todos os campos são validados por funções específicas:

  - hasInjection: detecta padrões suspeitos (tags <script>, comandos SQL, tentativas de path traversal, funções de execução de código como eval).
  - sanitizeInput: remove caracteres e padrões maliciosos, para garantir que os dados sejam tratados antes de serem persistidos no banco.

  Caso seja identificado risco de injeção, o sistema exibe um alerta de segurança (IonAlert) e bloqueia a submissão.

- **Validações adicionais de integridade**

  O sistema impede que o cadastro seja realizado se:

  - O campo nome ou descrição estiver vazio.
  - O estado, categoria ou finalidade não forem selecionados.
  - As quantidades (quantidade_total e qtd_max_emprestimo) forem menores que 1.
  - O preço não estiver definido corretamente.

- **Tratamento de preço**

  O valor monetário é convertido automaticamente para o padrão BRL (Real) com a formatação toLocaleString. Antes do envio ao servidor, o valor é sanitizado e convertido em número decimal adequado para persistência no banco.

- **Envio de dados ao backend**

  Após validação, os dados são empacotados em um FormData (necessário para incluir arquivos) e enviados à API (/api/addferramenta) com o método *POST* com autenticação via JWT. Em caso de sucesso, o administrador recebe um *toast* de confirmação e é redirecionado para a listagem de ferramentas cadastradas.

- **Tratamento robusto de erros**

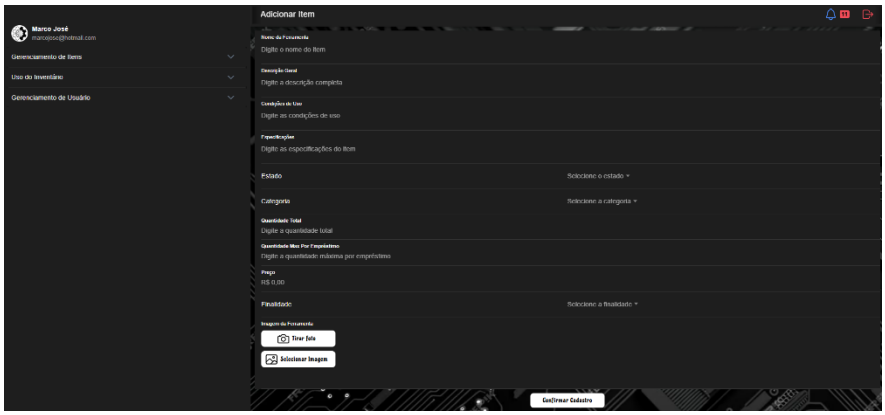
  Foram incluídos diferentes níveis de tratamento de erro:

  - **Erros de validação** → exibem mensagens específicas para cada campo.
  - **Erros de conexão** → informa que o servidor não pôde ser acessado.
  - **Erros inesperados** → mensagem genérica para evitar vazamento de informações sensíveis.

Essa tela é essencial para a manutenção e crescimento do acervo do laboratório, pois possibilita o **cadastro seguro, validado e documentado de novos itens**, para reduzir o risco

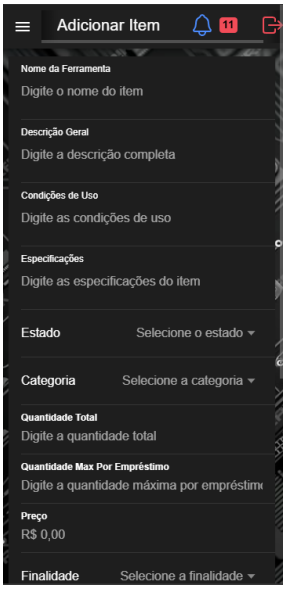
de inconsistências no inventário e proteger o sistema contra ataques de injeção e entradas maliciosas.

Figura 32 - Tela de Adição de Item (desktop)



Fonte: Autoria Própria

Figura 33 - Tela de Adição de Item (mobile)



Fonte: Autoria Própria

Essa tela é essencial para a administração do acervo de item, pois ela garante que os itens sejam cadastrados com todos os detalhes necessários para posterior exibição e gestão no sistema.

5.4.8 Tela Detalhes da Notificação

A tela detalhes da Notificação foi implementada para fornecer informações completas sobre eventos importantes no sistema, como solicitações de empréstimo, pedidos de adição de fundos ou aplicação de multas. Além de exibir os dados associados à notificação, ela permite a tomada de ações imediatas, que varia de acordo com o tipo de usuário (aluno ou administrador).

Funcionalidades principais:

- **Carregamento dinâmico de notificações**

  Através de requisições à API (/api/notificacoes/detalhes/:id), os detalhes de cada notificação são carregados automaticamente, isso inclui:

  - Mensagem descritiva
  - Data de envio
  - Tipo de notificação (empréstimo, fundos ou bloqueio)
  - Dados relacionados (quantidade, valor, ferramenta associada, data de devolução etc.)

- **Exibição de informações adicionais**

  Caso a notificação esteja vinculada a um **empréstimo**, são exibidos dados da ferramenta (nome, foto, quantidade solicitada e prazo de devolução).

  Para notificações de **fundos**, é mostrado o valor solicitado e o status da aprovação. Já para notificações de **bloqueio**, é informado se há multa pendente ou já quitada.

- **Ações disponíveis conforme perfil**
  - **Administrador:**
    - Pode **aceitar** ou **recusar** solicitações de empréstimo e fundos (Figura 36 e Figura 37).
    - A ação de aceitar um empréstimo gera um registro automático na tabela de empréstimos do sistema, com data de entrega e status inicial como "pendente" (Figura 34 e Figura 35).
    - Em solicitações de fundos, o valor é creditado diretamente no saldo do aluno.
  - **Aluno:**
    - Pode **cancelar** um pedido enviado ainda em status "pendente".
    - Em casos de multas, pode efetuar o **pagamento** diretamente a partir da notificação, para liberar novamente o acesso ao sistema.

- **Tratamento de status**

O sistema atualiza o status da solicitação em tempo real (aceito, recusado, cancelado, pendente, pago), que garante rastreabilidade. Isso é feito com chamadas adicionais às rotas /api/pedidos-emprestimo, /api/pedidos-fundos e /api/bloqueios.

- **Interface adaptada ao perfil do usuário**

  O cabeçalho da página exibe botões distintos para alunos e administradores:

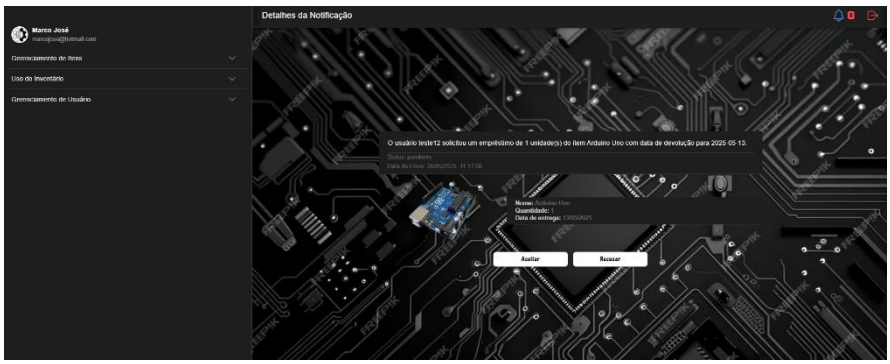
  - **Alunos**: ícones de notificações, saldo, carrinho, taxa de uso e logout.
  - **Administradores**: opções de gerenciamento e logout.

- **Segurança e controle de sessão**

  Caso o token do usuário seja inválido ou expirado, o sistema redireciona para a página de login. Além disso, o acesso às rotas é protegido, para evitar que usuários não autorizados manipulem notificações que não lhes pertencem.

Essa tela cumpre um papel essencial no fluxo de comunicação e decisão dentro do sistema, que funciona como a ponte entre as ações automáticas de notificação e a interação ativa do usuário. Ela garante que nenhum pedido fique sem resposta e que todas as ações realizadas (aceitação, recusa, cancelamento, pagamento) sejam registradas de forma transparente e vinculadas ao histórico do aluno ou do administrador.

Figura 34 - Tela Detalhes da Notificação tipo ‘Empréstimo’ (desktop)

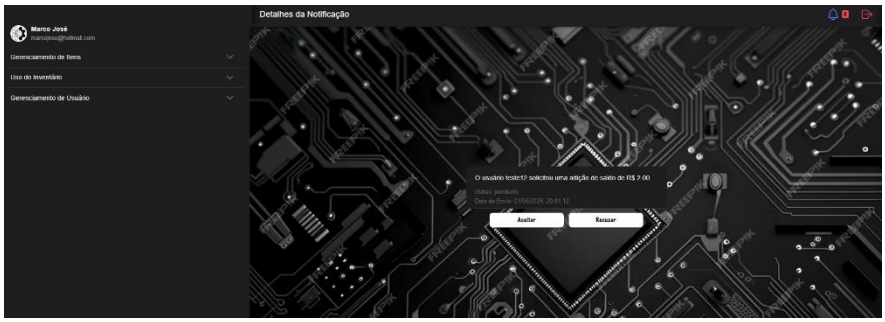


Fonte: Autoria Própria

Figura 35 - Tela Detalhes da Notificação tipo ‘Empréstimo’ (mobile)

Fonte: Autoria Própria

Figura 36 - Tela Detalhes da Notificação tipo 'Fundo' (desktop)



Fonte: Autoria Própria

Figura 37 - Tela Detalhes da Notificação tipo 'Fundo' (mobile)

Fonte: Autoria Própria

5.4.9 Tela Carrinho

A tela **Carrinho** (Figura 38 e Figura 39) foi projetada para centralizar todos os itens selecionados pelo aluno antes da conclusão de uma compra. Nela, o usuário tem acesso a informações detalhadas sobre cada ferramenta adicionada ao carrinho, além de opções de gerenciamento e finalização da transação.

Funcionalidades principais:

- **Listagem de itens do carrinho**
  Os itens adicionados ao carrinho são carregados a partir da API (/api/carrinho/:usuarioId) e exibidos com:
  - Nome e foto da ferramenta
  - Preço unitário
  - Quantidade selecionada
  - Valor total parcial do item
- **Controle de quantidade**
  O usuário pode alterar a quantidade de itens diretamente no carrinho, sem ultrapassar os limites de estoque disponíveis, que são buscados em tempo real pela rota /api/ferramenta/:id.
  - Quantidade mínima: 1
  - Quantidade máxima: vinculada ao estoque atual da ferramenta
- **Remoção de itens**
  Qualquer item pode ser removido individualmente do carrinho através da rota /api/carrinho/remover/:id.
- **Cálculo dinâmico do valor total**
  O preço total é recalculado automaticamente conforme a quantidade de itens é alterada ou itens são removidos.
- **Validação de saldo**
  Antes de finalizar a compra, o sistema realiza uma verificação do saldo do aluno, obtido pela rota /api/user/saldo/:usuarioId.
  - Caso o saldo seja **inferior ao valor total**, a compra é bloqueada e o usuário é informado.
  - Caso seja suficiente, a transação pode prosseguir.
- **Finalização da compra**

Ao clicar em *Finalizar Compra*, o sistema confirma a intenção do usuário e, em seguida, processa a transação pela rota /api/comprar/:usuarioId.

- O saldo do aluno é atualizado imediatamente.
- Os itens são removidos do carrinho.
- Uma mensagem de sucesso é exibida via IonToast.

- **Integração com o cabeçalho do sistema**

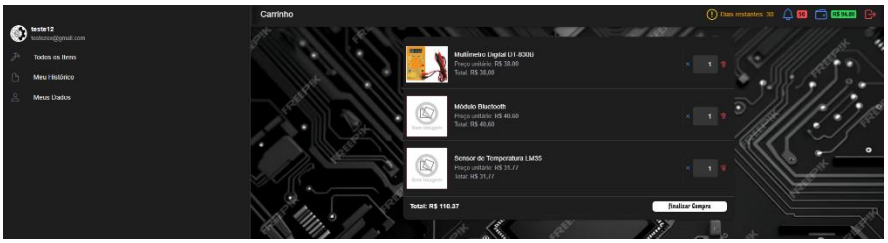
  Assim como em outras telas, o cabeçalho contém botões de acesso rápido, tais que:

  - Status de bloqueio
  - Taxas aplicáveis
  - Ícone de notificações
  - Saldo atualizado (em tempo real após cada compra)
  - Logout

Essa tela desempenha papel fundamental na parte de **monetização e gestão de recursos dentro do sistema**, pois conecta o fluxo de compras ao gerenciamento de saldo virtual dos alunos. Sua implementação garante que:

- Nenhuma compra seja realizada sem validação de saldo.
- O aluno possa gerenciar suas escolhas de forma prática (aumentar, reduzir ou remover itens).
- O processo de compra seja finalizado de forma clara e com feedback imediato de sucesso ou erro.

Figura 38 - Tela de Carrinho (desktop)



Fonte: Autoria Própria

Figura 39 - Tela de Carrinho (mobile)

Fonte: Autoria Própria

5.4.10 Tela de Gerenciamento de Usuários

A tela de **Gerenciamento de Usuários** (Figura 40 e Figura 41) foi projetada exclusivamente para os administradores, ela permite o controle completo dos usuários cadastrados no sistema. Essa tela centraliza funções relacionadas ao acompanhamento, bloqueio, desbloqueio, exclusão e envio de notificações ou lembretes, além de oferecer mecanismos de busca e filtragem.

Funcionalidades principais:

- **Listagem de usuários**

  Exibe todos os usuários cadastrados, com destaque em:

  - Foto de perfil
  - Nome
  - Matrícula
  - Status atual (Ativo ou Bloqueado)

- **Busca e filtros**

  Inclui barra de pesquisa que permite localizar usuários tanto por nome quanto por matrícula. Além disso, o administrador pode filtrar especificamente usuários **bloqueados** ou visualizar todos.

- **Bloqueio e desbloqueio de usuários**

  - O bloqueio é feito de forma contextualizada: o administrador informa um **motivo** e pode opcionalmente incluir uma **multa**.

    - Usuários bloqueados podem ser **desbloqueados** a qualquer momento, para restabelecer seu o acesso ao sistema.
    - Todo o processo é registrado com o ID do administrador responsável.
- **Exclusão de usuários**

  Permite remover definitivamente um usuário da base. Antes da exclusão, é exibido um alerta de confirmação para evitar remoções acidentais.
- **Envio de lembretes por e-mail**

  O administrador pode enviar mensagens personalizadas diretamente para o e-mail do usuário selecionado, pelo próprio sistema.
- **Gerenciamento de taxa de empréstimo**

  A tela também permite **alterar a taxa de empréstimo** (valor em R$ aplicado sobre operações de empréstimo).
    - O valor atual da taxa é exibido no cabeçalho da tela.
    - Um modal é aberto para que o administrador insira a nova taxa, que é então persistida no banco de dados.
- **Feedback em tempo real**

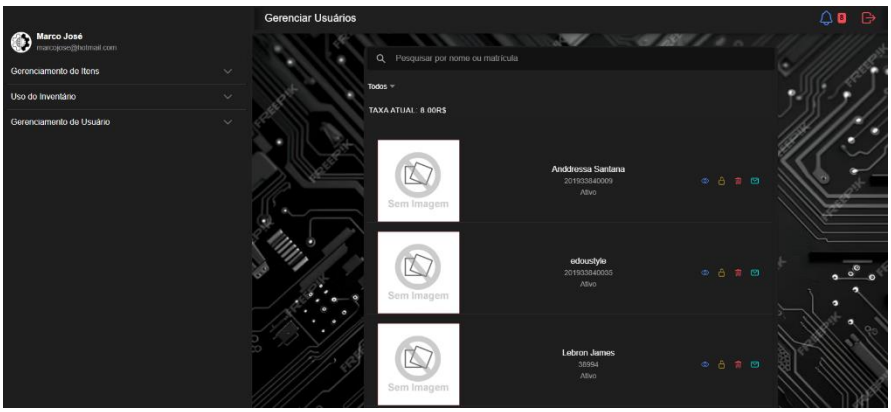
  Todas as ações administrativas (bloqueio, desbloqueio, exclusão ou envio de lembretes) retornam mensagens de sucesso ou erro por meio de alertas (IonAlert) e notificações (Toast).

A tela de **Gerenciamento de Usuários** representa a camada de **controle organizacional do sistema**. Ela garante que os administradores possam:

- Restringir o acesso de alunos inadimplentes ou que descumpriram regras.
- Enviar lembretes de forma rápida para melhorar a comunicação.
- Alterar parâmetros do sistema, como a taxa de empréstimo, sem necessidade de intervenção no código.
- Manter a base de usuários organizada, com a remoção de cadastros desatualizados.

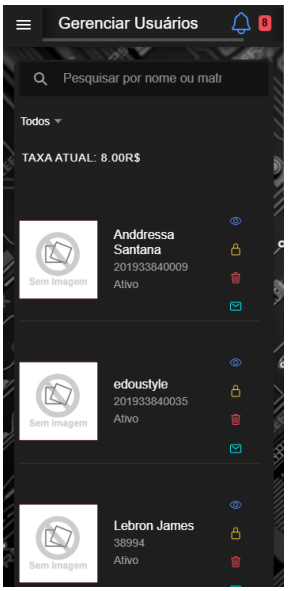
Assim, a funcionalidade fortalece a segurança, a disciplina no uso do laboratório e a governança do sistema, o que a consolida como uma das partes mais estratégicas da aplicação.

Figura 40 - Tela Gerenciar Usuários (desktop)



Fonte: Autoria Própria

Figura 41 - Tela Gerenciar Usuários (mobile)



Fonte: Autoria Própria

5.4.11 Tela de Estatísticas

A tela de estatísticas (Figura 42 e Figura 43) foi desenvolvida com o objetivo de fornecer aos administradores uma visão completa e detalhada do uso do sistema. Essa funcionalidade concentra dados sobre empréstimos, vendas, usuários e movimentações financeiras, para transformá-los em relatórios e gráficos interativos.

Funcionalidades principais:

- **Filtro por mês/ano**

  O administrador pode selecionar o período de interesse (últimos 12 meses) para visualizar as estatísticas correspondentes.

- **Itens mais emprestados e mais vendidos**

A tela apresenta listas das ferramentas e componentes com maior saída, afim de identificar quais são mais demandados pelos alunos.

- **Total do mês**
  Exibe de forma resumida a quantidade total de empréstimos e vendas realizadas no período selecionado.
- **Clientes mais ativos**
  Lista os usuários que mais utilizaram o sistema, com a consideração tanto das compras quanto dos empréstimos, afim de identificar padrões de engajamento.
- **Itens com baixo estoque**
  Fornece um alerta sobre ferramentas e componentes em quantidade reduzida, para auxiliar o administrador a planejar reposições.
- **Receita do mês e receita histórica**
  Mostra o valor total arrecadado no mês corrente e também apresenta a evolução da receita ao longo do tempo em forma de gráfico de linha.

A tela utiliza gráficos interativos (baseados no Chart.js) para tornar os dados mais compreensíveis:

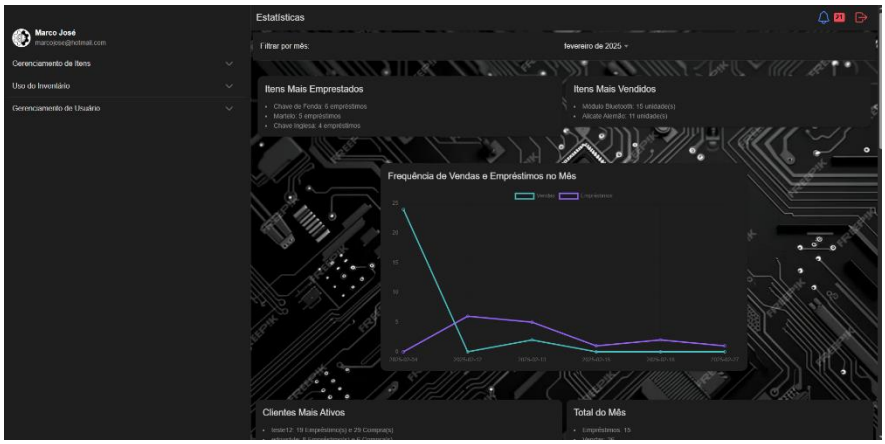
- **Gráfico de linha – Frequência de vendas e empréstimos**: mostra a evolução diária no mês selecionado.
- **Gráfico de barras – Vendas e empréstimos por categoria**: exibe quais tipos de ferramentas ou componentes têm maior procura.
- **Gráfico de barras – Clientes mais ativos**: destaca os usuários que mais interagem com o sistema.
- **Gráfico de linha – Receita mensal**: apresenta a evolução do faturamento em meses consecutivos, para identificar tendências.

A tela de **Estatísticas** funciona como um **painel de gestão estratégica**. Ela possibilita que os administradores:

- Tomem decisões baseadas em dados sobre estoque e compras.
- Identifiquem quais ferramentas precisam ser priorizadas em futuras aquisições.
- Avaliem a movimentação financeira do sistema e a sustentabilidade do modelo de empréstimos e vendas.
- Reconheçam os alunos mais ativos e incentivem boas práticas de utilização do laboratório.

Com isso, a tela fortalece o caráter analítico e administrativo do sistema, para aproximá-lo de um modelo profissional de gestão de recursos laboratoriais.

Figura 42 - Tela de Estatísticas (desktop)



Fonte: Autoria Própria

Figura 43 - Tela de Estatísticas (mobile)

Fonte: Autoria Própria

As demais telas implementas estão listadas no APÊNDICE C – TELAS IMPLEMENTADAS**,** junto com suas respectivas figuras.

6 CONCLUSÃO E TRABALHOS FUTUROS

O sistema desenvolvido atingiu plenamente a proposta de modernizar o controle de empréstimos e vendas no laboratório acadêmico, que substitui processos manuais por uma solução informatizada, responsiva e segura. A automação trouxe maior organização ao inventário, que reduzem erros e facilitam o trabalho dos administradores.

6.1 Contribuições

A aplicação foi construída em Ionic React, o que garante acessibilidade em diferentes dispositivos (desktop, tablets e smartphones) e proporciona uma experiência de uso consistente. Entre os principais avanços tecnológicos, destacam-se:

- **Autenticação e segurança**: implementação de login com verificação de e-mail, controle de sessão via JWT e comunicação protegida por HTTPS.
- **Gestão financeira integrada**: sistema de crédito funcional, com recarga e controle de saldo individual.
- **Notificações automatizadas**: envio de alertas por e-mail e push para devoluções, confirmações e avisos administrativos.
- **Painel administrativo completo**: controle de usuários, itens, estoque e monitoramento das transações em tempo real.
- **Controle de prazos**: verificações diárias automáticas para identificar devoluções em atraso.
- **Relatórios e estatísticas**: geração automática de relatórios mensais e exibição de gráficos interativos sobre vendas, empréstimos, entrada de fundos e movimentações gerais.

Em comparação ao processo anterior, que dependia de registros manuais, o sistema representa um salto tecnológico significativo. Ele não apenas melhora a eficiência do laboratório, como também amplia a confiabilidade das operações e fornece ferramentas de análise que não existiam antes.

6.2 Dificuldades

Durante o desenvolvimento do projeto, diversos desafios técnicos foram enfrentados. A integração com o OAuth2 para o envio seguro de e-mails exigiu atenção especial à segurança e permissões da conta. A hospedagem do site num servidor online assim como a configuração do

domínio HTTPS e seu redirecionamento para o IP do servidor também foi uma etapa complexa, que envolveu ajustes no DNS e certificados SSL.

A automatização das tarefas diárias por meio de cron jobs, o tratamento de sessões e o gerenciamento simultâneo de ações no frontend e backend (como em compras, empréstimos e notificações em tempo real) também representaram desafios relevantes, pois foram necessários testes rigorosos e refinamento contínuo da lógica do sistema.

6.3 Trabalhos Futuros

Apesar da completude da solução atual, há diversas possibilidades de evolução para o sistema, como:

- **Desenvolvimento de um aplicativo nativo** para Android/iOS, para aproveitar a base construída com Ionic e assim expandir para funcionalidades offline;
- **Integração com tecnologia QR Code** para controle físico de empréstimos, para tornar o processo mais ágil e seguro;
- **Sistema de gamificação**, que recompensaria alunos com bom histórico de uso e devoluções pontuais.

REFERÊNCIAS

W3C. *Media Queries – CSS Techniques for Responsive Design*. 2023 Disponível em: https://www.w3.org/TR/css3-mediaqueries/. Acesso em: 27 abr. 2025.

WILKYWAY. 프론트엔드 프레임워크 비교 (Angular / React / Vue / Svelte). Wilkyway (Tistory), 1 nov. 2022. Disponível em: https://streamls.tistory.com/entry/%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C-%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC-%EB%B9%84%EA%B5%90-Angular-React-Vue-Svelte . Acesso em: 29 ago. 2025.

IONIC. *Ionic Framework Documentation*. Disponível em: https://ionicframework.com/docs. Acesso em: 27 abr. 2025.

GONCALVES, Felipe. *CLIENT X SERVER – entendendo a comunicação na internet*. DIO, 31 jan. 2024. Disponível em: https://www.dio.me/articles/client-x-server-entendendo-a-comunicacao-na-internet. Acesso em: 22 ago. 2025.

MDN WEB DOCS. JavaScript. Mozilla Foundation, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 29 ago. 2025.

FACEBOOK. *React – A JavaScript library for building user interfaces*. Disponível em: https://reactjs.org. Acesso em: 27 abr. 2025

W3C. HTML Standard. World Wide Web Consortium, 2025. Disponível em: https://html.spec.whatwg.org/. Acesso em: 29 ago. 2025.

MICROSOFT. TypeScript: JavaScript With Syntax For Types. Disponível em: https://www.typescriptlang.org/ . Acesso em: 22 ago. 2025.

META. Introducing JSX. React Documentation. Disponível em: https://react.dev/learn/writing-markup-with-jsx . Acesso em: 22 ago. 2025.

DAHL, Ryan. *Node.js*. 2009. Disponível em: https://nodejs.org/. Acesso em: 29 ago. 2025.

TILKOV, Stefan; VELMURUGAN, Steve. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, v. 14, n. 6, p. 80-83, 2010.

HERRON, Andrew. *Node.js Web Development*. 4. ed. Birmingham: Packt Publishing, 2016.

BROWN, Ethan. *Web Development with Node and Express: Leveraging the JavaScript Stack*. 2. ed. Sebastopol: O'Reilly Media, 2020.

KAPPAGANTULA. *SQL vs NoSQL Key Differences - MySQL vs MongoDB*. Disponível em: https://www.edureka.co/blog/sql-vs-nosql-db/. Publicado em: 21 fev. 2025. Acesso em: 22 ago. 2025.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de Banco de Dados*. 7. ed. São Paulo: Pearson, 2020.

ORACLE. *MySQL Documentation*. Redwood City: Oracle Corporation, 2025. Disponível em: https://dev.mysql.com/doc/ . Acesso em: 22 ago. 2025.

PRITCHARD, Stephen. *SQL vs NoSQL Databases: What's the Difference?*. 2021. Disponível em: https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL . Acesso em: 22 ago. 2025.

MOZILLA. *Man-in-the-middle (MITM)*. Mozilla Foundation, 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/MITM. Acesso em: 29 ago. 2025.

MOZILLA. *Certificado de segurança de site*. [S. l.], 2024. Disponível em: https://support.mozilla.org/pt-BR/kb/certificado-de-seguranca-de-site. Acesso em: 23 ago. 2025.

ECMA INTERNATIONAL. The JSON Data Interchange Syntax (ECMA-404). Disponível em: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/. Acesso em: 29 ago. 2025.

AUTH0. *Introduction to JSON Web Tokens*. Auth0 Docs, 2025. Disponível em: https://auth0.com/intro-to-jwt/ . Acesso em: 22 ago. 2025.

MDN WEB DOCS. JavaScript. Mozilla Foundation, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 29 ago. 2025.

FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Tese (Doutorado em Filosofia) – University of California, Irvine, 2000. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/. Acesso em: 29 ago. 2025.

OWASP. *Autenticação segura em aplicações web*. Open Web Application Security Project. Disponível em: https://owasp.org. Acesso em: 27 abr. 2025.

ARAMBURU, Rodrigo. *JWT – JSON Web Token em PHP*. Boteco Digital, 19 jun. 2022. Disponível em: https://www.botecodigital.dev.br/php/jwt-json-web-token-em-php/. Acesso em: 22 ago. 2025.

IETF. *The OAuth 2.0 Authorization Framework. RFC 6749*, out. 2012. Disponível em: https://tools.ietf.org/html/rfc6749. Acesso em: 23 ago. 2025.

OKTA. *OAuth 2.0 and OpenID Connect*. Okta Developer Documentation, 2025. Disponível em: https://developer.okta.com/docs/concepts/oauth-openid/ . Acesso em: 22 ago. 2025.

NODEMAILER. *Nodemailer: Send e-mails from Node.js*. 2025. Disponível em: https://nodemailer.com/ . Acesso em: 22 ago. 2025.

MOZILLA. *Service Workers: an introduction*. 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API . Acesso em: 22 ago. 2025.

GOOGLE DEVELOPERS. *Progressive Web Apps*. 2024. Disponível em: https://developers.google.com/web/progressive-web-apps . Acesso em: 22 ago. 2025.

BIBLIOTECA PÚBLICA DO PARANÁ (BPP). *Serviço de Empréstimo*. Disponível em: https://www.bpp.pr.gov.br/Pagina/Servico-de-Emprestimo. Acesso em: 23 ago. 2025.

NODE.JS. *Node.js Documentation*. Disponível em: https://nodejs.org/en/docs. Acesso em: 27 abr. 2025.

PM2. *PM2 – Advanced Node.js process manager*. Disponível em: https://pm2.keymetrics.io/. Acesso em: 22 ago. 2025.

MICROSOFT. *Visual Studio Code*. Disponível em: https://code.visualstudio.com/ . Acesso em: 22 ago. 2025.

ORACLE. *MySQL Workbench*. Disponível em: https://www.mysql.com/products/workbench/ . Acesso em: 22 ago. 2025.

GOOGLE. *Chrome DevTools*. Disponível em: https://developer.chrome.com/docs/devtools/. Acesso em: 22 ago. 2025.

GITHUB. *GitHub*. Disponível em: https://github.com/. Acesso em: 22 ago. 2025.

APACHE SOFTWARE FOUNDATION. *Apache JMeter*. Disponível em: https://jmeter.apache.org/. Acesso em: 22 ago. 2025.

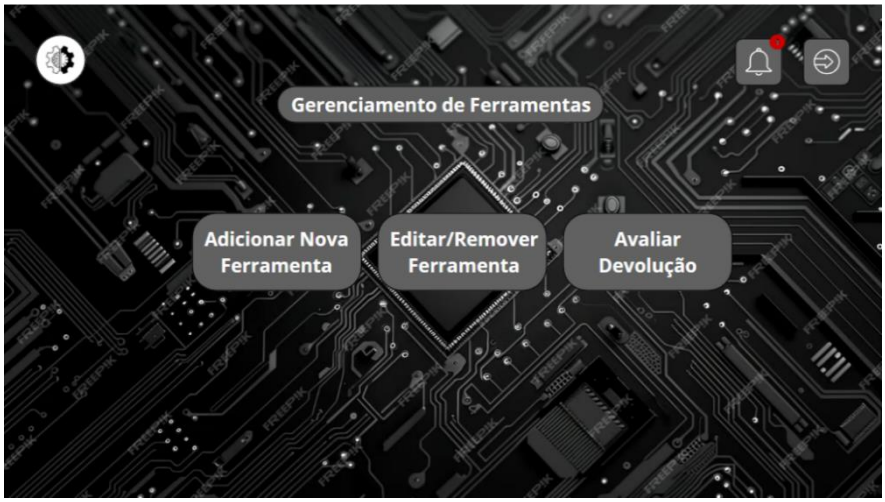
CANVA. *Canva*. Disponível em: https://www.canva.com/. Acesso em: 22 ago. 2025.

MYSQL. *MySQL 8.0 Reference Manual*. Disponível em: https://dev.mysql.com/doc/. Acesso em: 27 abr. 2025.

EXPRESSJS. *Express.js – Web framework for Node.js*. Disponível em: https://expressjs.com. Acesso em: 27 abr. 2025.

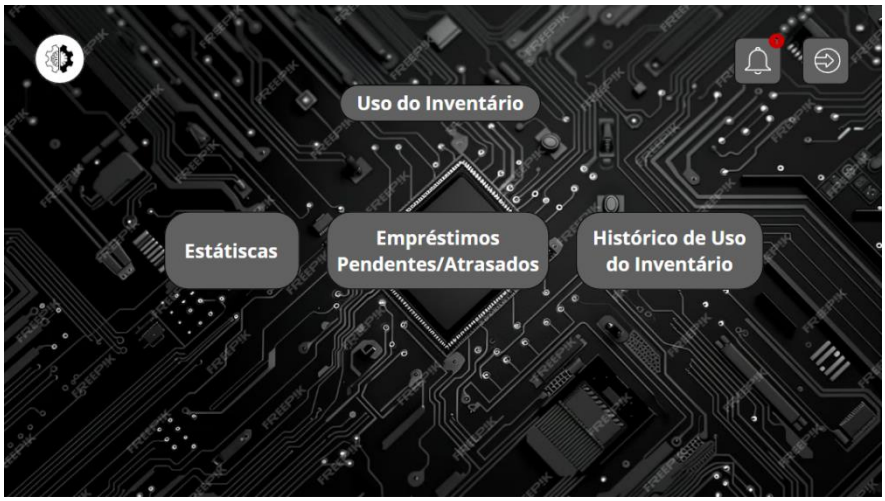
APÊNDICE A – PROTÓTIPOS DE TELA

Figura 44 – Tela Gerenciamento de Ferramentas (protótipo inicial)



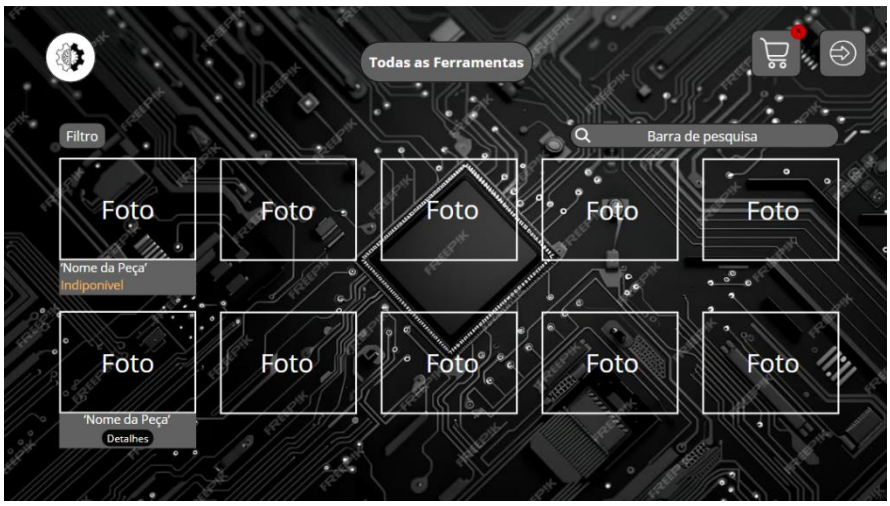
Fonte: Autoria Própria

Figura 45 – Tela Uso do inventário (protótipo inicial)



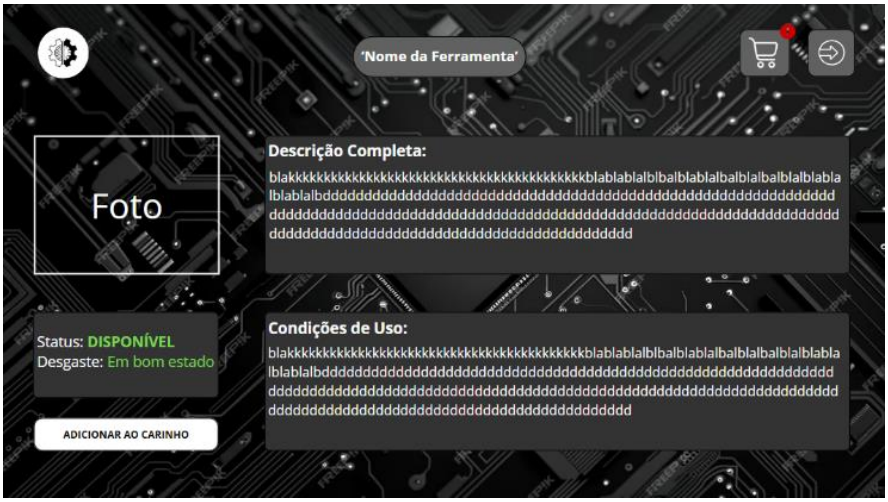
Fonte: Autoria Própria

Figura 46 – Tela Todas as Ferramentas (protótipo inicial)



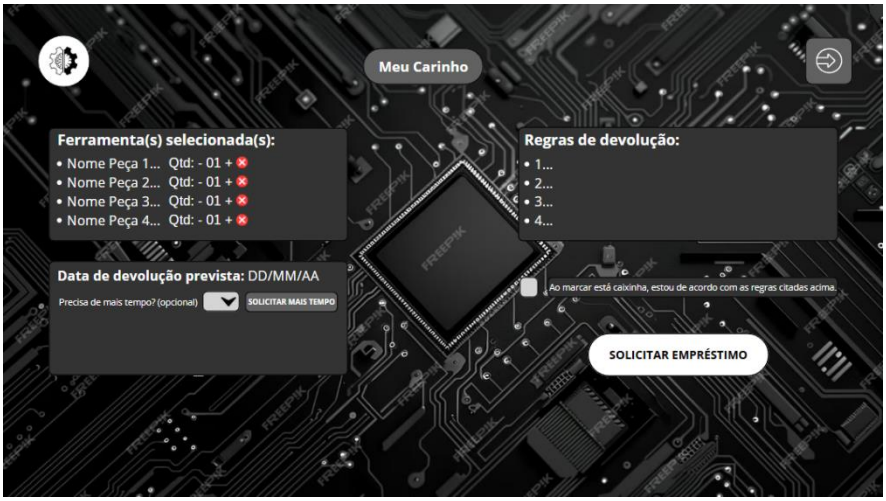
Fonte: Autoria Própria

Figura 47 - Tela Detalhes Ferramenta disponível (protótipo inicial)



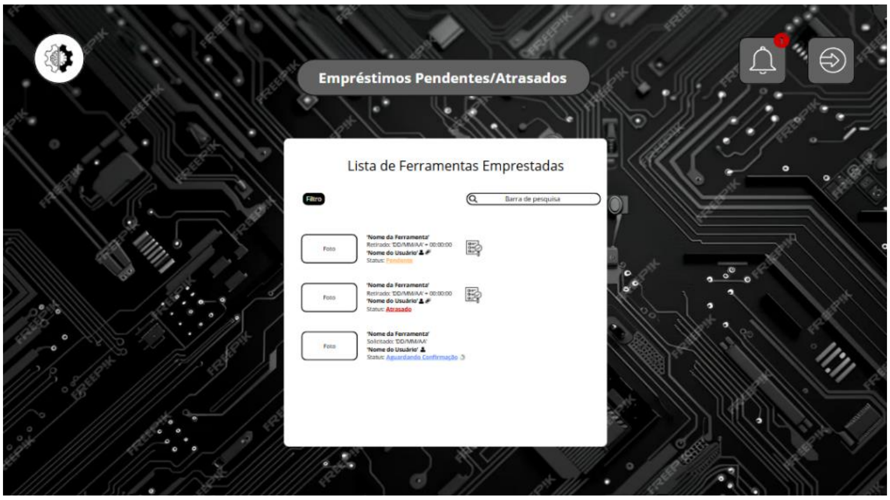
Fonte: Autoria Própria

Figura 48 - Tela Meu Carrinho (protótipo inicial)



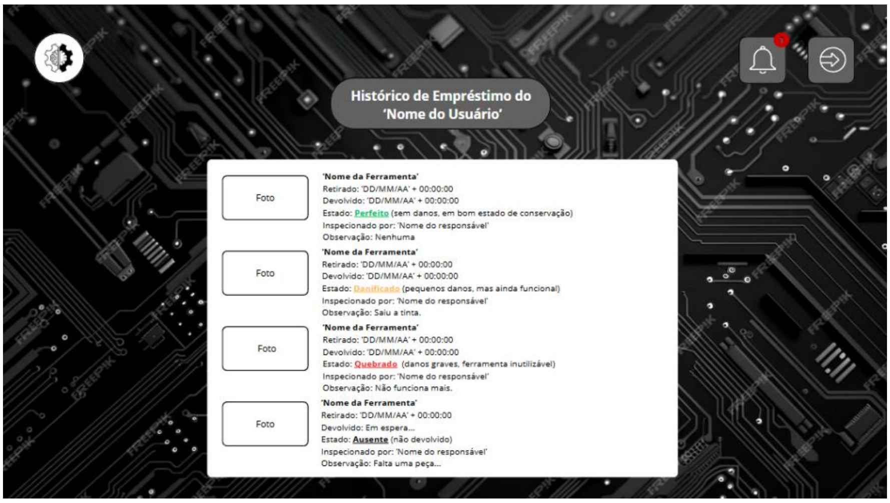
Fonte: Autoria Própria

Figura 49 – Tela Pendentes/Atrasados (protótipo inicial)



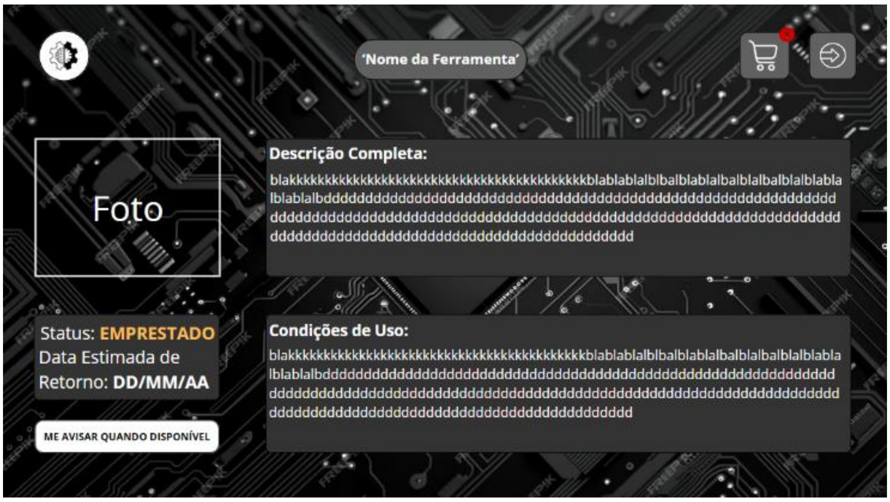
Fonte: Autoria Própria

Figura 50 - Tela Histórico de Empréstimo Aluno (protótipo inicial)



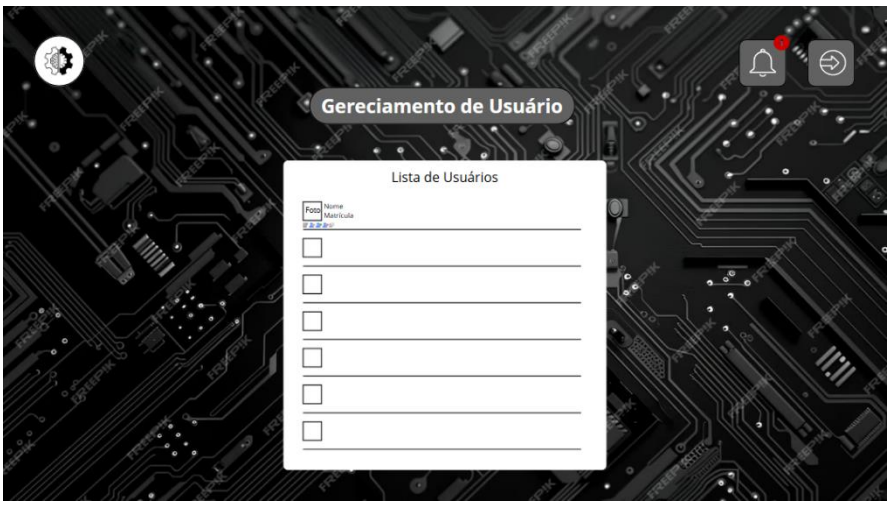
Fonte: Autoria Própria

Figura 51 – Tela Detalhes Ferramenta Emprestado (protótipo inicial)



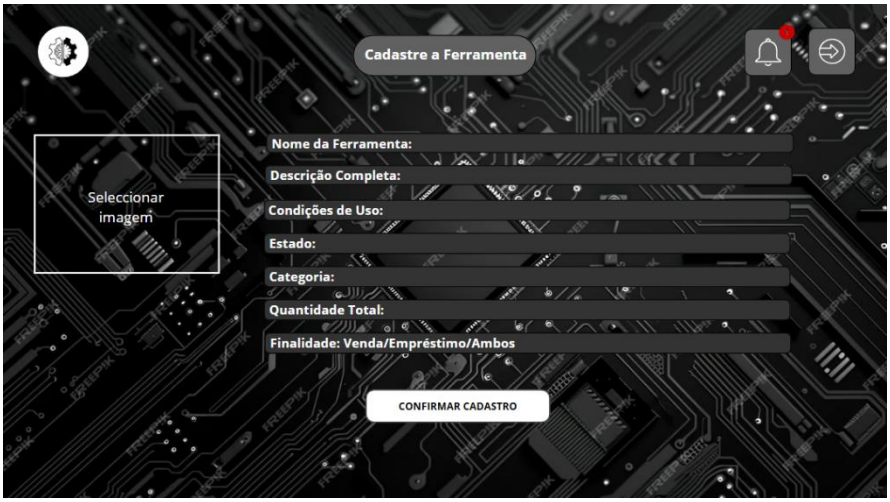
Fonte: Autoria Própria

Figura 52 – Tela Gerenciamento de Usuário (protótipo inicial)



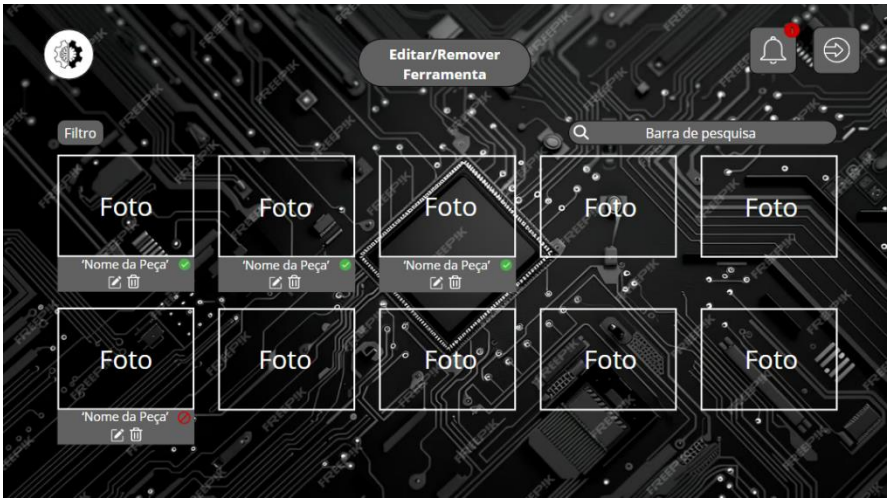
Fonte: Autoria Própria

Figura 53 – Tela Cadastrar a Ferramenta (protótipo inicial)



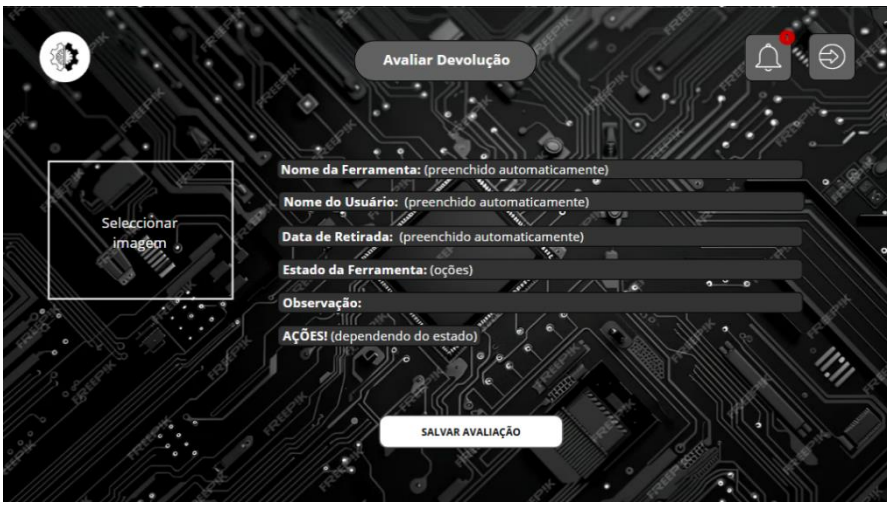
Fonte: Autoria Própria

Figura 54 - Tela Editar/Remover Ferramenta (protótipo inicial)



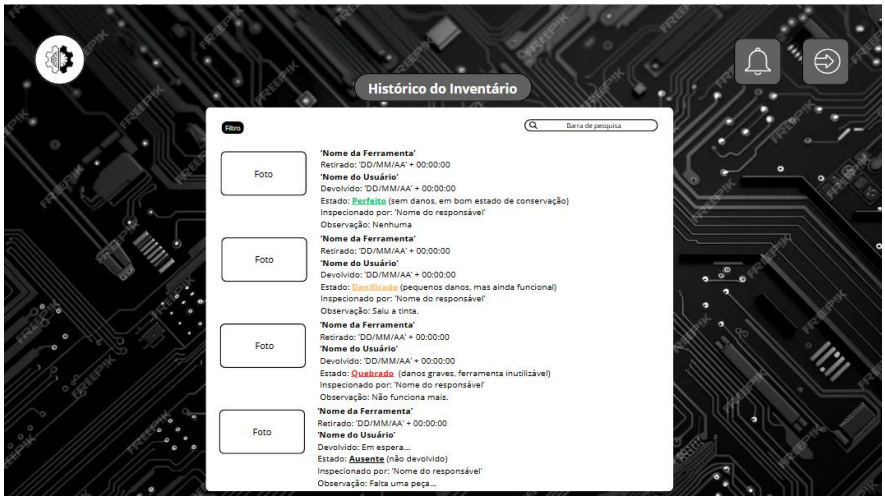
Fonte: Autoria Própria

Figura 55 - Tela Avaliar Devolução (protótipo inicial)



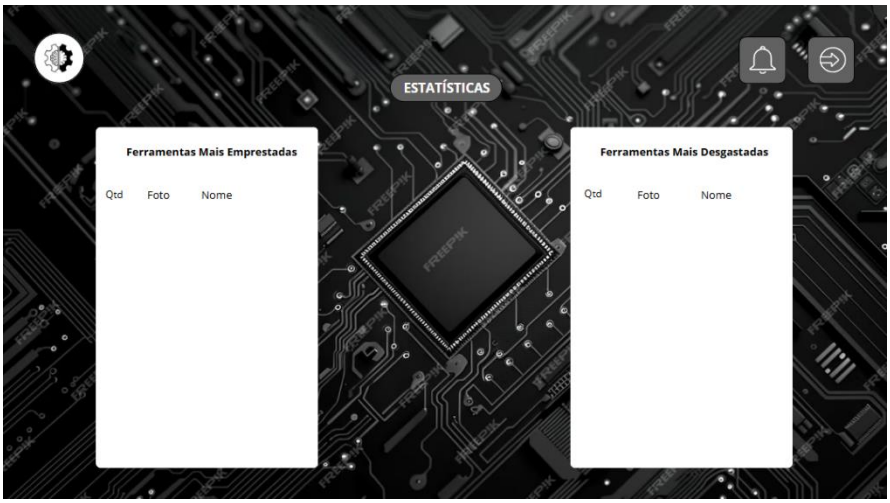
Fonte: Autoria Própria

Figura 56 - Tela Histórico do Inventário (protótipo inicial)



Fonte: Autoria Própria

Figura 57 - Tela Estatística (protótipo inicial)



Fonte: Autoria Própria

APÊNDICE B – ESTRUTURAS DETALHADAS DAS TABELAS

Tabela 5 - ‘Usuarios’

Campo	Tipo de Dado	Restrição	Descrição
id_usuario	INT	PK, Auto Increment	Identificador único do usuário
nome	VARCHAR(255)	NOT NULL	Nome completo do usuário
email	VARCHAR(255)	NOT NULL, Único	E-mail do usuário
celular	VARCHAR(15)	NOT NULL	Número de telefone
matricula	VARCHAR(50)	NOT NULL	Número de matrícula acadêmica
senha	VARCHAR(255)	NOT NULL	Senha criptografada
tipo_usuario	ENUM('Aluno', 'Admin')	NOT NULL	Define o perfil de acesso do usuário
data_cadastro	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data de criação da conta
saldo	DECIMAL(10,2)	DEFAULT 0.00	Crédito disponíveis
email_verificado	TINYINT(1)	DEFAULT 0	Indica se o email foi verificado (0/1)

Tabela 6 - ‘Ferramentas’

Campo	Tipo de Dado	Restrição	Descrição
id_ferramenta	INT	PK, Auto Increment	Identificador único da ferramenta
nome	VARCHAR(100)	NOT NULL	Nome da ferramenta
descricao	TEXT	NOT NULL	Descrição detalhada
estado	ENUM('Novo', 'Usado', 'Danificado')	NOT NULL	Estado da ferramenta
preco	DECIMAL(10,2)	DEFAULT 0.00	Preço de venda, se aplicável
quantidade_total	INT	NOT NULL	Quantidade em estoque
finalidade	ENUM('Venda', 'Emprestimo')	NOT NULL	Finalidade principal da ferramenta
condicoes_uso	TEXT	(Opcional)	Condições de uso recomendadas
categoria	ENUM(...)	(Opcional)	Categoria de classificação
qtd_max_emprestimo	INT	DEFAULT 1	Quantidade máxima por empréstimo
especificacoes	TEXT	(Opcional)	Outras especificações técnicas

Tabela 7 - ‘Emprestimos’

Campo	Tipo de Dado	Restrição	Descrição

id_emprestimo	INT	PK, Auto Increment	Identificador único do empréstimo
id_usuario	INT	FK (Usuarios)	Usuário que realizou o empréstimo
id_ferramenta	INT	FK (Ferramentas)	Ferramenta emprestada
status	ENUM(...)	NOT NULL	Status atual do empréstimo
estado	ENUM(...)	NOT NULL	Estado da ferramenta após devolução
quantidade	INT	NOT NULL	Quantidade emprestada
foto	LONGBLOB	(Opcional)	Foto de devolução
data_emprestimo	DATETIME	NOT NULL	Data do empréstimo
data_devolucao	DATETIME	(Opcional)	Data prevista para devolução

Tabela 8 - 'PedidosEmprestimo'

Campo	Tipo de Dado	Restrição	Descrição
id_pedido	INT	PK, Auto Increment	Identificador do pedido de empréstimo
id_usuario	INT	FK (Usuarios)	Usuário solicitante
id_ferramenta	INT	FK (Ferramentas)	Ferramenta solicitada
quantidade	INT	NOT NULL	Quantidade solicitada
data_devolucao	DATE	(Opcional)	Data prevista de devolução
status	VARCHAR(50)	NOT NULL	Status do pedido
data_pedido	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data da solicitação

Tabela 9 - 'Carrinho'

Campo	Tipo de Dado	Restrição	Descrição
id	INT	PK, Auto Increment	Identificador do item no carrinho
usuario_id	INT	FK (Usuarios)	Usuário dono do carrinho
ferramenta_id	INT	FK (Ferramentas)	Ferramenta adicionada
quantidade	INT	NOT NULL	Quantidade selecionada
data_adicionado	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data de adição ao carrinho

Tabela 10 - 'Vendas'

Campo	Tipo de Dado	Restrição	Descrição
id_venda	INT	PK, Auto Increment	Identificador único da venda
id_usuario	INT	FK (Usuarios)	Usuário que realizou a compra
id_ferramenta	INT	FK (Ferramentas)	Ferramenta adquirida

quantidade	INT	NOT NULL	Quantidade adquirida
valor_total	DECIMAL(10,2)	NOT NULL	Valor total da compra
data_venda	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data da venda

Tabela 11 - ‘PedidosFundos’

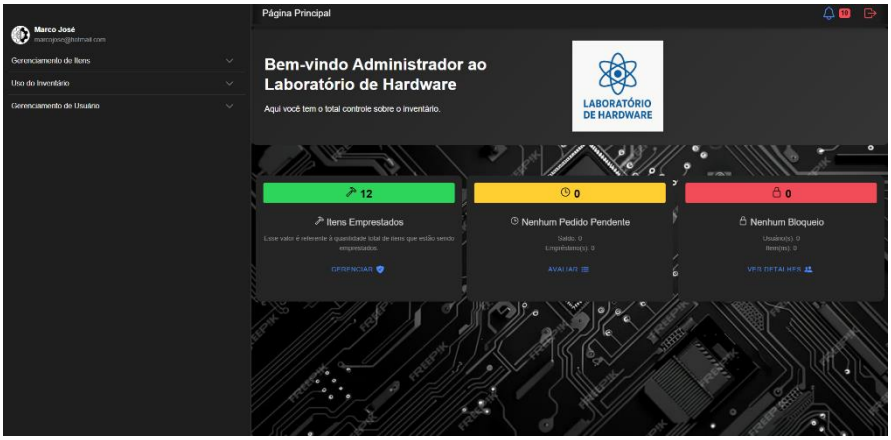
Campo	Tipo de Dado	Restrição	Descrição
id_pedido_fundos	INT	PK, Auto Increment	Identificador único do pedido de fundos
id_usuario	INT	FK (Usuarios)	Usuário que realizou o pedido de recarga
valor	DECIMAL(10,2)	NOT NULL	Valor solicitado para recarga
status	ENUM('Pendente', 'Aprovado', 'Recusado')	NOT NULL	Status do pedido de recarga
data_pedido	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data e hora em que o pedido foi realizado
data_aprovacao	TIMESTAMP	(Opcional)	Data e hora da aprovação ou recusa
observacao	TEXT	(Opcional)	Observações feitas pelo administrador

Tabela 12 - ‘Notificacoes’

Campo	Tipo de Dado	Restrição	Descrição
id_notificacao	INT	PK, Auto Increment	Identificador único da notificação
id_usuario	INT	FK (Usuarios)	Usuário destinatário da notificação
titulo	VARCHAR(255)	NOT NULL	Título da notificação
mensagem	TEXT	NOT NULL	Corpo da mensagem da notificação
tipo	ENUM('Emprestimo', 'Venda', 'Sistema', 'Alerta')	NOT NULL	Categoria da notificação
lida	TINYINT(1)	DEFAULT 0	Indica se a notificação foi lida (0/1)
data_envio	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data e hora de envio

APÊNDICE C – TELAS IMPLEMENTADAS

Figura 58 - Tela Inicial Admin (desktop)

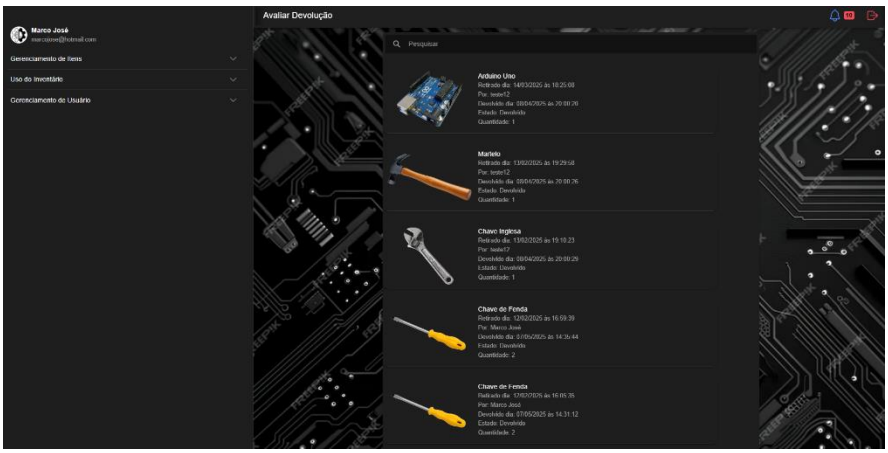


Fonte: Autoria Própria

Figura 59 - Tela Inicial Admin (mobile)

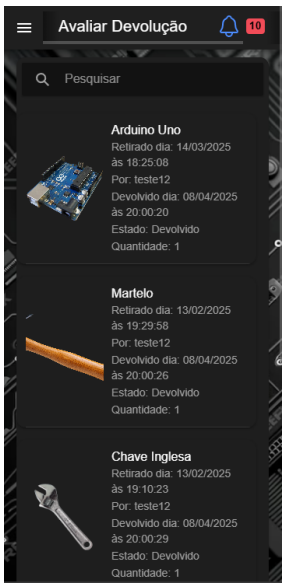
Fonte: Autoria Própria

Figura 60 - Tela Avaliar Devolução (desktop)



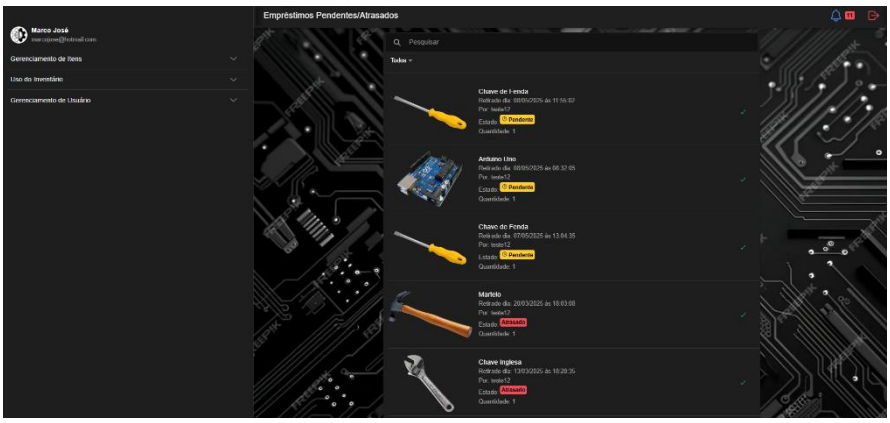
Fonte: Autoria Própria

Figura 61 - Tela Avaliar Devolução (mobile)



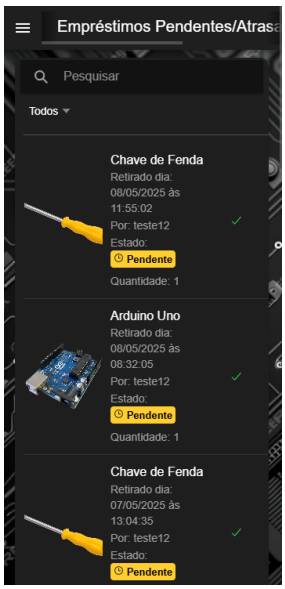
Fonte: Autoria Própria

Figura 62 - Tela Empréstimos Pendentes/Atrasados (desktop)



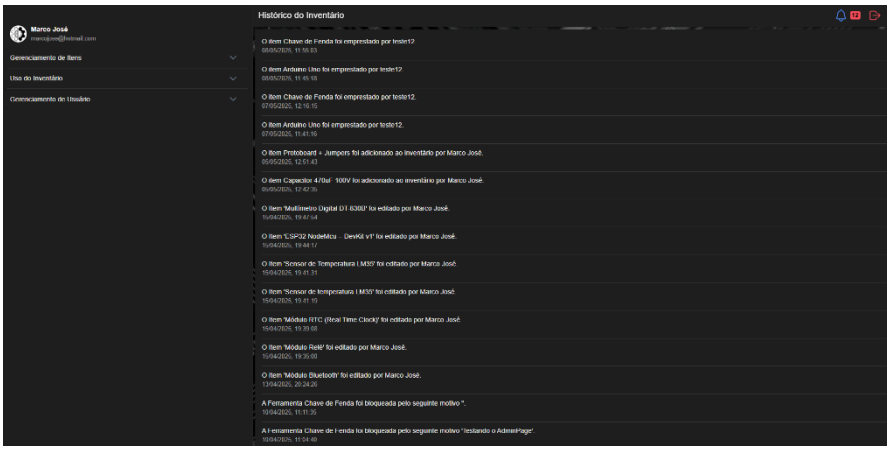
Fonte: Autoria Própria

Figura 63 - Tela Empréstimos Pendentes/Atrasados (mobile)



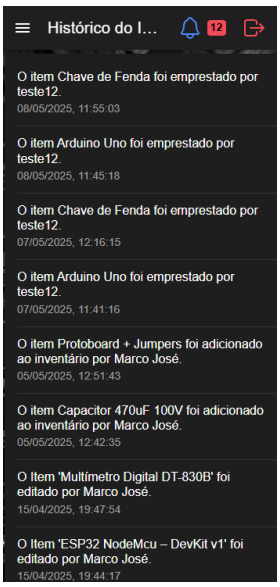
Fonte: Autoria Própria

Figura 64 - Tela Histórico do Inventário (desktop)



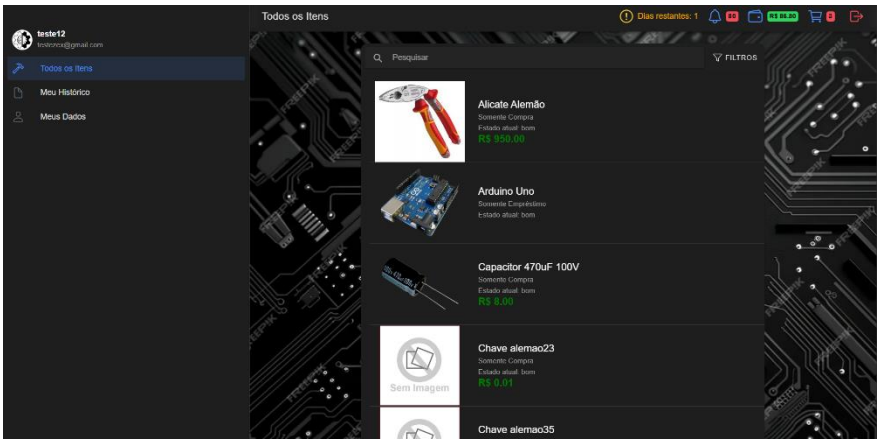
Fonte: Autoria Própria

Figura 65 - Histórico do Inventário (mobile)



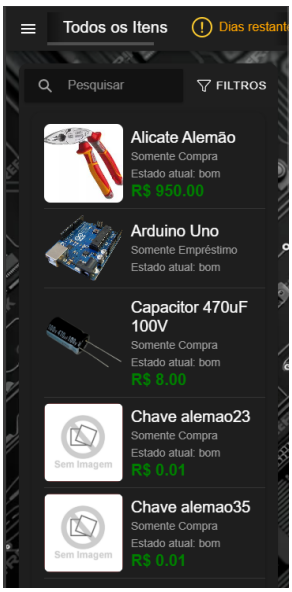
Fonte: Autoria Própria

Figura 66 - Tela Todos os Itens (desktop)



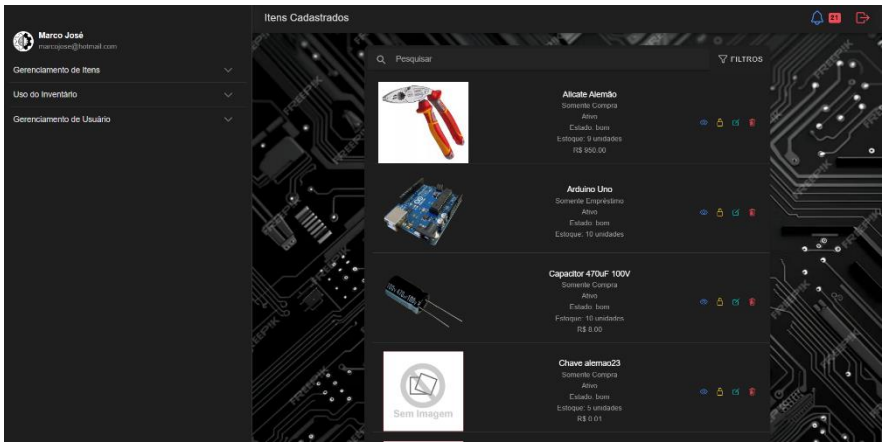
Fonte: Autoria Própria

Figura 67 - Tela Todos os Itens (mobile)



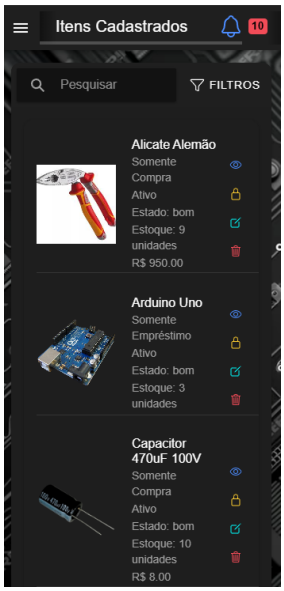
Fonte: Autoria Própria

Figura 68 - Tela Itens Cadastrado (desktop)



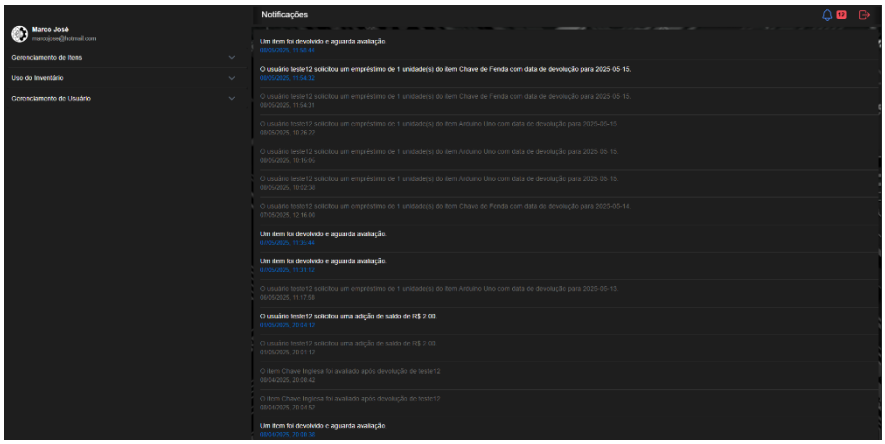
Fonte: Autoria Própria

Figura 69 - Tela Itens Cadastrados (mobile)



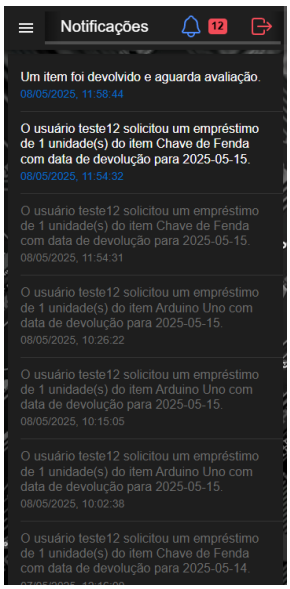
Fonte: Autoria Própria

Figura 70 - Tela de Notificação (desktop)



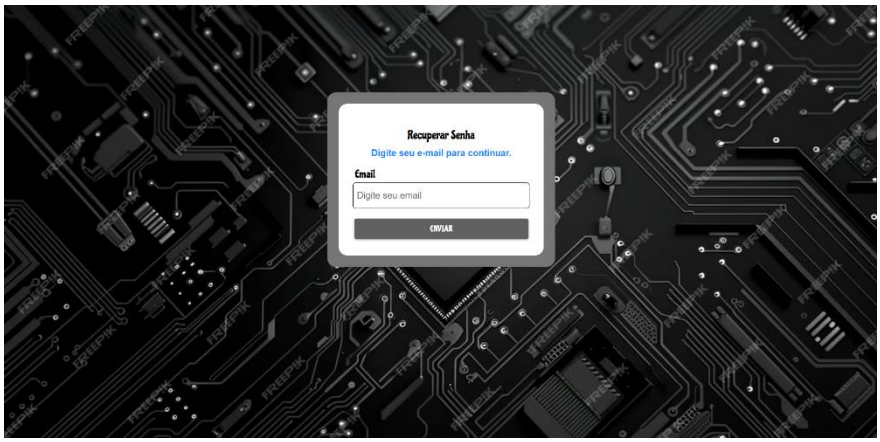
Fonte: Autoria Própria

Figura 71 - Tela de Notificações (desktop)



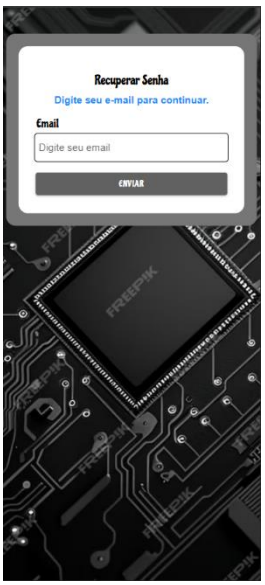
Fonte: Autoria Própria

Figura 72 - Tela Recuperar Senha (desktop)



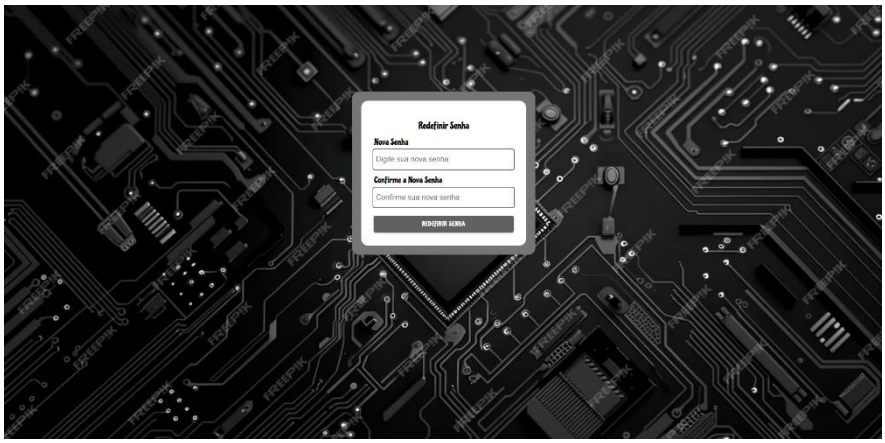
Fonte: Autoria Própria

Figura 73 - Tela Recuperar Senha (mobile)



Fonte: Autoria Própria

Figura 74 - Tela Redefinir Senha (desktop)



Fonte: Autoria Própria

Figura 75 - Tela Redefinir Senha (mobile)

Fonte: Autoria Própria

APÊNDICE D – REPOSITÓRIO GITHUB

https://github.com/edoustyle/LabhardwareWebapp