REVISTA OWL (*OWL Journal*)
www.revistaowl.com.br – ISSN: 2965-2634

DESENVOLVIMENTO DE UMA APLICAÇÃO WEB/MOBILE PARA CONTROLE DE CARREGAMENTO DE VEÍCULOS ELÉTRICOS

DEVELOPMENT OF A WEB/MOBILE APPLICATION FOR CONTROLLING THE CHARGING OF ELECTRIC VEHICLES

DOI: 10.5281/zenodo.18763877

Valnyr Vasconcelos Lira[1]
Alison Andrade Alves[2]
Denis Willian Paulo da Silva[3]
Pedro Pereira da Silva[4]

RESUMO

Com o crescimento do uso de veículos elétricos surge a necessidade da ampliação da infraestrutura de carregamento, acompanhada da necessidade de sistemas inteligentes capazes de monitorar, controlar e otimizar o uso dos carregadores. Nesse sentido, este trabalho apresenta o projeto e a implementação de uma solução tecnológica voltada à gestão eficiente do processo de recarga de veículos elétricos. A aplicação proposta foi desenvolvida com uma arquitetura moderna, escalável e multiplataforma, integrando tecnologias amplamente utilizadas no desenvolvimento de sistemas distribuídos. No frontend web, utilizou-se o React para a criação de interfaces dinâmicas, responsivas e de fácil usabilidade. Para o ambiente mobile, o React Native permitiu o desenvolvimento de uma aplicação única compatível com diferentes sistemas operacionais, garantindo padronização visual e funcionalidade consistente entre plataformas. No backend, foi adotado o Spring Boot, responsável pela implementação das regras de negócio, comunicação com o banco de dados e disponibilização de APIs REST. Essa camada gerencia informações como usuários, veículos, sessões de recarga e históricos de

1 Docente do IFPB - Campus Esperança, Brasil. E-mail: valnyr@ifpb.edu.br.
2 Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, IFPB - Campus Esperança, Brasil. E-mail: alison.alves@academico.ifpb.edu.br.
3 Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, IFPB - Campus Esperança, Brasil. E-mail: denis.paulo@academico.ifpb.edu.br.
4 Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, IFPB - Campus Esperança, Brasil. E-mail: pedro.silva.15@academico.ifpb.edu.com.

consumo do carregador. O banco de dados PostgreSQL foi utilizado para o armazenamento estruturado e seguro das informações, oferecendo confiabilidade, integridade dos dados e bom desempenho em consultas e transações. A utilização do Docker possibilitou a conteinerização dos serviços, facilitando a implantação, manutenção e escalabilidade da aplicação em diferentes ambientes. O trabalho evidencia a viabilidade do uso de tecnologias web e mobile integradas como uma solução eficiente para o gerenciamento de carregamento de veículos elétricos, abrindo espaço para futuras expansões e integrações com sistemas inteligentes de energia.

Palavras-chave: carregamento; veículo elétrico, gerenciamento.

ABSTRACT

With the growth in the use of electric vehicles comes the need to expand charging infrastructure, accompanied by the need for intelligent systems capable of monitoring, controlling, and optimizing the use of chargers. In this sense, this work presents the design and implementation of a technological solution aimed at the efficient management of the electric vehicle charging process. The proposed application was developed with a modern, scalable, and multiplatform architecture, integrating technologies widely used in the development of distributed systems. On the web frontend, React was used to create dynamic, responsive, and user-friendly interfaces. For the mobile environment, React Native allowed the development of a single application compatible with different operating systems, ensuring visual standardization and consistent functionality across platforms. On the backend, Spring Boot was adopted, responsible for implementing business rules, communicating with the database, and providing REST APIs. This layer manages information such as users, vehicles, charging sessions, and charger consumption history. The PostgreSQL database was used for structured and secure data storage, offering reliability, data integrity, and good performance in queries and transactions. The use of Docker enabled the containerization of services, facilitating the deployment, maintenance, and scalability of the application in different environments. The work highlights the feasibility of using integrated web and mobile technologies as an efficient solution for managing electric vehicle charging, opening up space for future expansions and integrations with intelligent energy systems.

Keywords: charging; electric vehicle; management.

1. INTRODUÇÃO

Desde tempos imemoriais, a humanidade busca aprimorar suas tecnologias para otimizar processos e garantir um futuro mais sustentável, e a história dos veículos é um testemunho dessa busca. Curiosamente, a ideia de propulsão elétrica antecede a de motores a combustão interna. Há registros de que os veículos elétricos (VEs) já existiam pelo menos 27 anos antes de seus equivalentes movidos a combustíveis fósseis (Souza, 2022). O

desenvolvimento dos VEs foi impulsionado pelas descobertas fundamentais da eletricidade e do magnetismo. Michael Faraday, com seu trabalho sobre Rotação Eletromagnética em 1821, lançou as bases para o motor elétrico (Souza, 2022). Pouco depois, em 1832, o cientista britânico William Sturgeon inventou o primeiro motor elétrico de corrente contínua com comutador capaz de acionar máquinas, um marco crucial na história da tecnologia elétrica (Larminie; Lowry, 2012).

Em um cenário global cada vez mais consciente dos desafios ambientais e da necessidade de reduzir a dependência de combustíveis fósseis, os veículos elétricos ressurgem com força, posicionando-se como uma solução promissora para o transporte. A transição de motores a combustão interna para propulsores elétricos representa não apenas uma mudança tecnológica, mas uma verdadeira revolução na forma como nos locomovemos e interagimos com o meio ambiente. Essa mudança é evidenciada pelo crescimento exponencial do mercado de veículos eletrificados. Conforme dados da Empresa de Pesquisa Energética (EPE, 2024), a participação dos VEs no licenciamento total no Brasil tem crescido aceleradamente. Em 2023, as vendas atingiram 93 mil unidades, representando 4,3% do licenciamento total e um aumento impressionante de 90,7% em relação ao ano anterior. Até outubro de 2024, esse número já havia saltado para 138 mil unidades, correspondendo a cerca de 7% do licenciamento de veículos novos no período (EPE, 2024).

Esse crescimento notável demonstra que os veículos elétricos estão conquistando cada vez mais espaço em um mercado historicamente dominado pelos veículos a combustão. Montadoras como a Renault têm investido em projetos com ótimo custo-benefício, como o Kwid E-Tech, o mais acessível no Brasil, com preços a partir de R$ 146.990,00 em julho de 2022, visando expandir suas frotas elétricas (Renault, 2022). Este cenário de adoção crescente traz consigo a necessidade premente de desenvolver soluções inovadoras para o suporte e a manutenção desses veículos. Um dos pilares para a massificação dos VEs é a disponibilidade de uma rede de carregamento eficiente, acessível e inteligente. Atualmente, o Brasil, assim como outros países em desenvolvimento, enfrenta o desafio de expandir sua infraestrutura de carregamento para acompanhar o ritmo de crescimento da frota de veículos elétricos. Este

contexto abre um vasto campo para a pesquisa e o desenvolvimento de estações de carregamento que não só atendam às demandas energéticas, mas que também se integrem a ecossistemas inteligentes e sustentáveis.

No entanto, este avanço rápido revela um gargalo crítico que ameaça a massificação dos VEs: a insuficiência da infraestrutura de carregamento. A disponibilidade de uma rede de recarga eficiente, acessível e inteligente é um pré-requisito fundamental para garantir a viabilidade e a confiança do consumidor na mobilidade elétrica. O Brasil, como outros países em desenvolvimento, enfrenta o desafio premente de expandir sua infraestrutura para acompanhar o ritmo de crescimento da frota de veículos elétricos.

Neste cenário, o presente projeto se justifica pela necessidade urgente de desenvolver soluções inovadoras e de baixo custo que contribuam para a expansão dessa infraestrutura. A proposta é o desenvolvimento de uma aplicação web/mobile, que deverá servir como interface de suporte tecnológico para o controle e gerenciamento de uma estação de carregamento simplificada a ser instalada no IFPB - Campus Esperança, com ênfase no aproveitamento da geração fotovoltaica já existente no campus.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Tecnologias utilizadas

Este projeto foi concebido para operar de forma integrada tanto no ambiente web quanto em dispositivos móveis (mobile). Para alcançar este objetivo, a interface com o usuário foi centralizada no ecossistema *JavaScript*, utilizando *React* para a construção da aplicação web e *React Native* para a versão móvel. Para sustentar essa arquitetura e gerenciar a lógica de negócio, foi desenvolvido um robusto *backend* com o *framework Spring Boot*, que expõe uma API segura e performática. A persistência dos dados é garantida pelo PostgreSQL, escolhido por sua confiabilidade e capacidade de gerenciar dados relacionais complexos. Finalmente, para assegurar a consistência entre os ambientes e simplificar a implantação, toda a aplicação é encapsulada e orquestrada em contêineres utilizando *Docker*. Juntas, essas

tecnologias formam uma *stack* completa e moderna, capaz de entregar uma solução escalável e de fácil manutenção.

2.1.1 React

React JS é uma biblioteca *JavaScript* amplamente utilizada para a criação de interfaces de usuário (UI - *user interface*). Criada em 2011 pelo time do Facebook, o *React* foi desenvolvido com o objetivo de aperfeiçoar a atualização e a sincronização de atividades simultâneas no feed de notícias das redes sociais, entre eles chat, status, listagem de contatos e outros. O *React JS* é uma biblioteca *frontend* baseada na linguagem *JavaScript*, seu principal objetivo é permitir o desenvolvimento de interfaces baseadas em componentes para aplicações web. Constitui uma base de conhecimento necessária para sua utilização, os conceitos de componentização, estado, propriedades, sintaxe *JavaScript Sintaxe Extension* (JSX), entre outras (Lins, 2019, p.11).

Segundo a nova documentação do *React* (2025), componentes de função em *JavaScript* retornam uma marcação. O *React* pode criar componentes que dividem a interface em partes independentes e reutilizáveis, o que permite considerar cada elemento isoladamente. Um exemplo é a criação de componentes como cabeçalhos e rodapés que podem ser usados em várias páginas de um aplicativo. Esta abordagem modular e extensível tem vantagens de desenvolvimento porque cada componente é desenvolvido de forma independente e não depende diretamente de outros elementos.

2.1.2 React Native

O *React Native* é um *framework* de código aberto criado pelo Facebook em 2015, que estende os princípios do *React* para o desenvolvimento de aplicações móveis. Sua finalidade é permitir que desenvolvedores construam aplicações para as plataformas Android e iOS a partir de um único código-base, escrito em *JavaScript*. Diferente de outras abordagens

híbridas que renderizam componentes dentro de uma *WebView*, o *React Native* utiliza uma "ponte" (*bridge*) para se comunicar com as APIs de renderização nativas e, como destaca Braga (2019, p. 18), "gera o código nativo da(s) plataforma(s)", resultando em uma interface com performance e aparência genuínas.

2.1.3 Spring Boot

Para a construção do *backend* da aplicação - a camada responsável por processar as regras de negócio, gerenciar o acesso aos dados e se comunicar com o banco de dados - foi escolhido o *Spring Boot*. O *Spring Boot* é uma ferramenta baseada no *Spring Framework* que facilita e agiliza o desenvolvimento de aplicações, pois gerencia de forma simplificada toda a parte de dependências e configuração do ambiente (Moraes, 2024).

Sua eficiência se baseia em componentes chave, como os *Spring Boot Starters*, que agregam várias dependências relacionadas em um único pacote. A dependência *spring-boot-starter-web*, por exemplo, agrupa todas as configurações necessárias para um servidor, banco de dados e aplicações web, simplificando drasticamente o gerenciamento do projeto. Outro componente essencial é o *AutoConfiguration*, que gerencia as configurações padrão da aplicação e permite customizações através de anotações, reduzindo a necessidade de configurações manuais e permitindo que o desenvolvedor se concentre na lógica de negócio (Moraes, 2024).

No contexto deste projeto, o *Spring Boot* será utilizado para desenvolver uma API RESTful. Esta API servirá como a ponte de comunicação entre o *frontend* (aplicações web e móvel desenvolvidas em *React* e *React Native*) e o banco de dados, expondo *endpoints* seguros para que os clientes possam solicitar dados, enviar informações e interagir com o sistema de forma estruturada e eficiente.

2.1.4 PostgreSQL

A camada de persistência de dados é um componente crítico de qualquer sistema, responsável por armazenar, gerenciar e garantir a integridade das informações. Para este projeto, o sistema de gerenciamento de banco de dados escolhido foi o PostgreSQL, um sistema de base relacional *open source*.

Segundo Silva (2023, p. 22), o PostgreSQL destaca-se por sua "arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade". Sua estrutura é fundamentada no modelo relacional, que organiza os dados em tabelas compostas por linhas e colunas, utilizando chaves estrangeiras para definir os relacionamentos entre elas, o que é ideal para cenários que exigem um esquema bem definido e alta consistência (Silva, 2023, p. 14).

Um dos pilares que garantem a robustez do PostgreSQL é sua total compatibilidade com as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Conforme detalhado por Silva (2023, p. 16), essa conformidade assegura a confiabilidade das transações, garantindo que as operações sejam executadas por completo ou totalmente revertidas em caso de falha, mantendo o banco de dados em um estado válido e consistente.

A escolha pelo PostgreSQL para este projeto foi, portanto, estratégica. Sua forte aderência ao padrão SQL e a capacidade de gerenciar relacionamentos complexos com integridade referencial são essenciais para a lógica de negócio da aplicação. No contexto deste sistema, o PostgreSQL é responsável por armazenar todas as informações cruciais, desde as credenciais de usuários até os registros de atividades.

2.1.5 Docker

Para garantir a padronização e a portabilidade do ambiente de desenvolvimento e produção, foi adotado o *Docker*. O *Docker* é uma plataforma de código aberto que automatiza a implantação de aplicações dentro de contêineres de software isolados. Essa tecnologia

resolve um dos problemas mais comuns no ciclo de desenvolvimento: a inconsistência de ambientes entre as máquinas dos desenvolvedores e os servidores de produção.

Conforme descrito por Barbosa (2023, p. 36), o *Docker* permite "empacotar e executar aplicativos em contêineres isolados", que encapsulam não apenas o código da aplicação, mas também todas as suas dependências, como bibliotecas, *frameworks* e até mesmo o sistema operacional. Essa abordagem "elimina problemas comuns relacionados a diferenças de configuração e dependências entre os desenvolvedores, facilitando a colaboração e reduzindo os conflitos" (Barbosa, 2023, p. 36). Ao criar um ambiente controlado e reproduzível, garante-se que a aplicação funcione de maneira idêntica, independentemente de onde seja executada.

No contexto deste projeto, o *Docker*, em conjunto com o *Docker Composse*, é utilizado para orquestrar os múltiplos serviços que compõem a aplicação: o *backend* desenvolvido com *Spring Boot*, o banco de dados PostgreSQL e as aplicações *frontend* em *React* e *React Native*. Através de um arquivo de configuração declarativo, o *Docker Compose* define e gerencia os contêineres de cada serviço, suas redes e volumes de dados, simplificando a configuração e a execução de um ambiente complexo com um único comando (Barbosa, 2023, p. 37). A adoção dessas ferramentas é, portanto, fundamental para assegurar a consistência, a portabilidade e a escalabilidade da solução, proporcionando uma base sólida e eficiente para todo o ciclo de vida do software.

3. METODOLOGIA

Esta pesquisa possui natureza aplicada com caráter experimental, com o objetivo principal de realizar o desenvolvimento de uma aplicação web e mobile para controle de um sistema de carregamento veicular.

Para seu desenvolvimento foram seguidas as seguintes etapas metodológicas:

- análise e escolha das tecnologias possíveis de serem aplicadas;
- modelagem do banco de dados que armazenará os dados coletados;
- desenvolvimento da aplicação web: prototipagem e desenvolvimento;

- desenvolvimento da aplicação mobile: prototipagem e desenvolvimento.

4. RESULTADOS E DISCUSSÕES

4.1 Modelagem do Banco de Dados

Para definir a estrutura de armazenamento dos dados do sistema, foi realizado um processo de modelagem em duas etapas. Inicialmente, foi elaborado um modelo conceitual para identificar as principais entidades do sistema, como **Usuario**, **Veiculo** e **Sessao_Carga**, e suas relações de alto nível.

Posteriormente, este modelo foi refinado em um modelo lógico, resultando no Diagrama Entidade-Relacionamento (DER) apresentado na Figura 1. Este diagrama detalha as entidades como tabelas, seus atributos (colunas), chaves primárias e os relacionamentos com cardinalidade, servindo como a planta baixa para a construção do banco de dados.

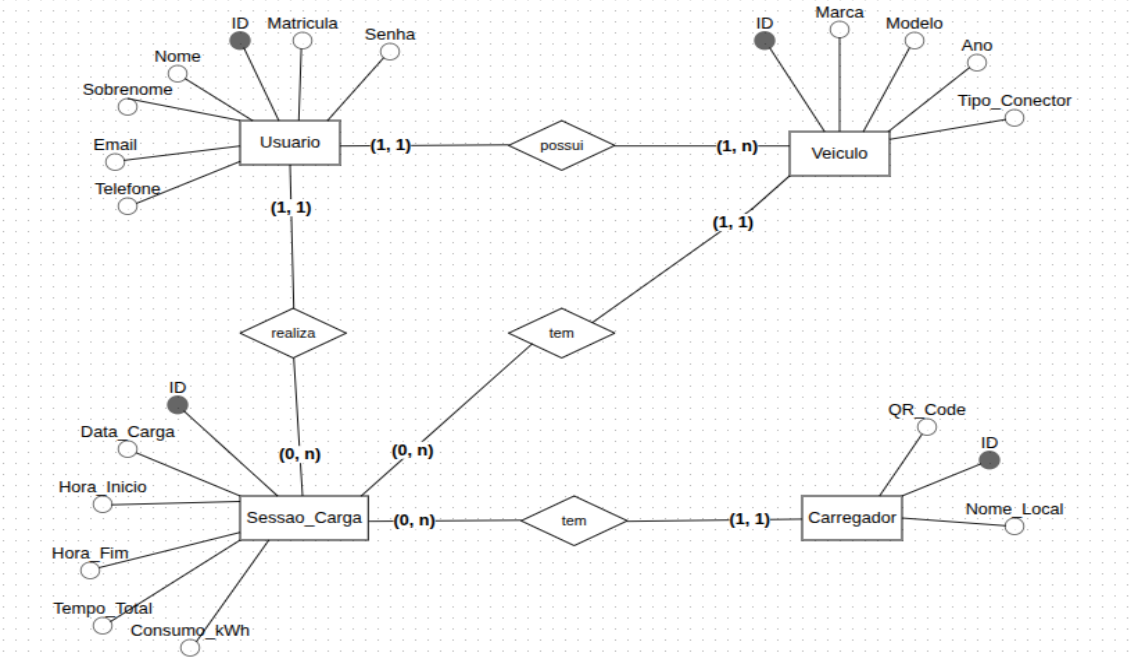


Figura 1: Diagrama Entidade-Relacionamento (DER) do banco de dados.
Fonte: Própria (2026).

A Figura 2 apresenta o modelo lógico do banco de dados, que detalha a estrutura das tabelas do sistema. Derivado do Diagrama Entidade-Relacionamento, este modelo define as tabelas **Usuario**, **Veiculo**, **Sessao_Carga** e **Carregador** com seus respectivos atributos, especificando as chaves primárias (PK) e as chaves estrangeiras (FK) que implementam os relacionamentos.

As relações estabelecem as regras de negócio do sistema: um **Usuario** pode possuir múltiplos (0,n) **Veiculos**, mas cada veículo pertence a um único (1,1) **Usuário**. De forma semelhante, um **Usuario**, um **Veiculo** e um **Carregador** podem estar associados a múltiplas (0,n) sessões de carga ao longo do tempo. Crucialmente, cada **Sessao_Carga** é unicamente vinculada a um (1,1) **Usuario**, um (1,1) **Veiculo** e um (1,1) **Carregador**, garantindo a integridade e o rastreamento completo de cada operação.

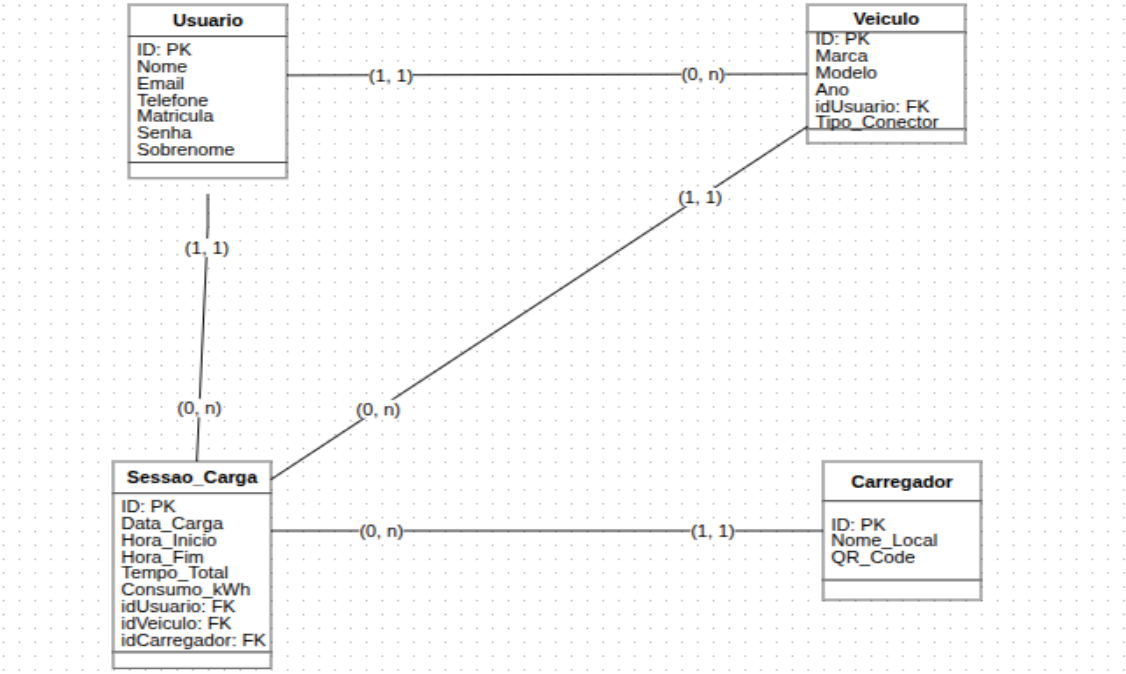
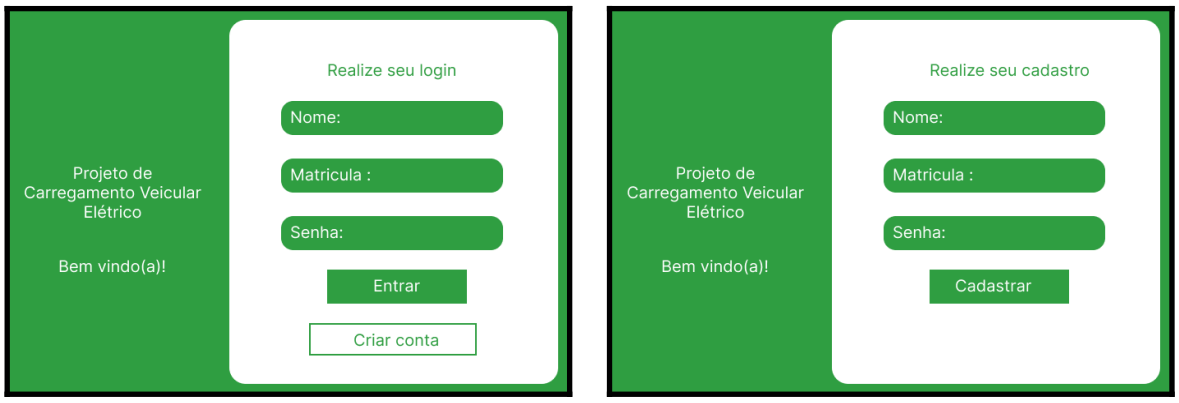


Figura 2: Modelo Lógico do banco de dados.
Fonte: Própria (2026).

4.2 Aplicação Web

A Figura 4a ilustra o protótipo da tela de *login* para o sistema web do projeto, na visão de um usuário. A interface adota um layout de duas colunas, com um painel lateral esquerdo que apresenta o título 'Projeto de Carregamento Veicular Elétrico' e uma mensagem de boas-vindas. A seção principal, à direita, contém o formulário de autenticação, onde o usuário deve inserir seu 'Nome', 'Matrícula' e 'Senha'. Abaixo dos campos, são disponibilizadas as opções para 'Entrar' no sistema e para 'Criar conta', caso o usuário ainda não possua cadastro.

Já a Figura 4b exibe o a tela de cadastro de novos usuários para o sistema web. Mantendo a identidade visual e o layout de duas colunas da tela de login, a interface apresenta um formulário simplificado para a criação de conta, solicitando ao usuário as informações de 'Nome', 'Matrícula' e 'Senha'. A submissão dos dados para registro é realizada através do botão 'Cadastrar'.

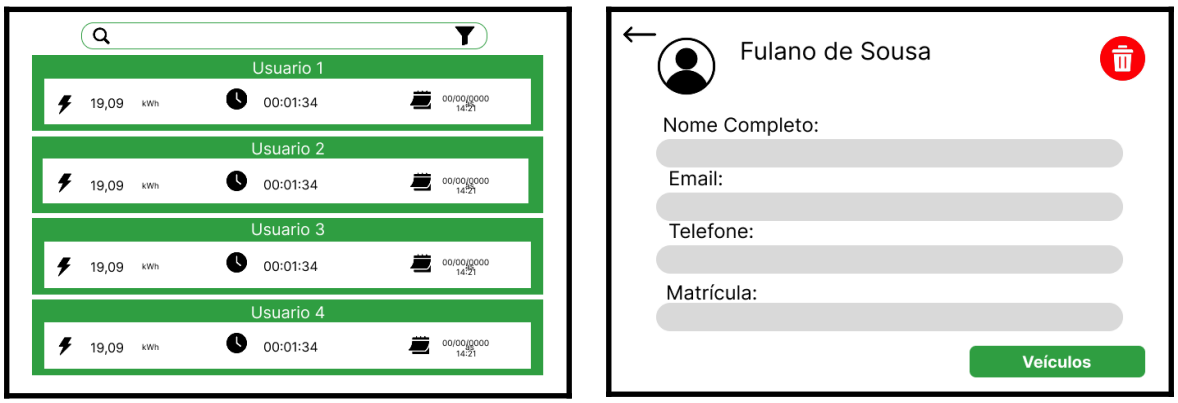


a) Login b) Cadastro

Figura 4: Protótipo das telas de login e de cadastro do sistema web.
Fonte: Própria (2026)

A Figura 5a apresenta o protótipo do painel de monitoramento do sistema web, que serve como a tela principal para a gestão dos carregamentos. No topo, uma barra de busca acompanhada de um ícone de filtro permite a consulta e filtragem dos registros. A área principal exibe uma lista de sessões de carregamento, onde cada item representa a atividade

de um usuário específico ('Usuario 1', 'Usuario 2', etc.). Para cada sessão, são exibidos os dados essenciais: a energia total consumida em kWh, a duração do carregamento e a data/hora do evento, permitindo um acompanhamento detalhado das operações.



a) Painel de monitoramento b) Detalhes do usuário

Figura 5: Protótipo do painel de monitoramento e da tela de detalhes do usuário do sistema web.
Fonte: Própria (2026)

Já a Figura 5b ilustra o protótipo da tela de detalhes do usuário, que seria acessada, por exemplo, ao selecionar um registro no painel de monitoramento do sistema web. A interface exibe as informações cadastrais de um usuário específico - 'Nome Completo', 'Email', 'Telefone' e 'Matrícula' - em formato de visualização. A tela oferece duas ações principais: um ícone de exclusão no canto superior direito para remover o usuário do sistema, e um botão 'Veículos' na parte inferior, que navega para a lista de veículos associados a esta conta.

A Figura 6a detalha o protótipo da tela de histórico de carregamentos por veículo, acessada ao clicar no botão 'Veículos' na tela de detalhes do usuário. No topo, a interface mantém a barra de busca e o filtro, permitindo a consulta de registros específicos. A área principal lista o histórico de carregamentos, organizado por cada veículo do usuário ('Veículo 1', 'Veículo 2'). Para cada registro, são apresentadas as informações detalhadas da sessão, como a energia consumida (kWh), a duração e a data/hora do evento.

a) Histórico de carregamentos por veículo

b) Detalhes do veículo

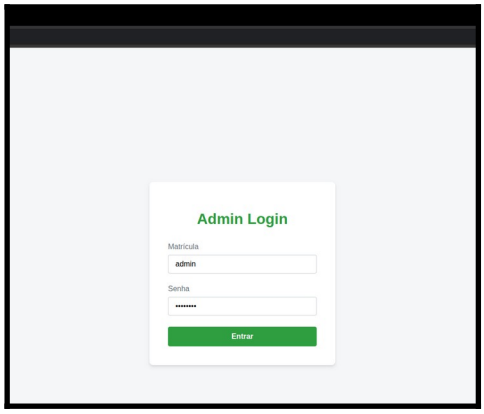
c) Histórico detalhado por veículo

Figura 6: Protótipo do painel de histórico de carregamentos por veículo, da tela de detalhes do veículo e do histórico detalhado por veículo do sistema web.
Fonte: Própria (2026)

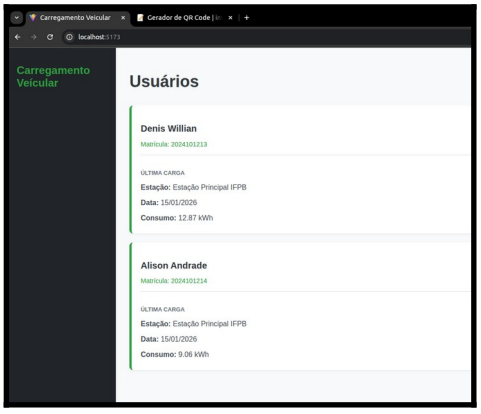
A Figura 6b apresenta o protótipo da tela de detalhes de um veículo específico dentro do painel administrativo do sistema web. A interface exibe as informações cadastrais do veículo, como 'Nome', 'Marca', 'Modelo' e 'Ano', em um formato de visualização. A principal funcionalidade desta tela é o botão 'Carregamentos', que, ao ser acionado, direciona o administrador para o histórico de carregamentos exclusivo daquele veículo, permitindo uma análise focada. Já a Figura 6c exibe o protótipo da tela de histórico de um veículo específico. Esta interface é apresentada após o administrador selecionar a opção 'Carregamentos' na tela anterior. O layout é limpo e focado na apresentação dos dados, com um título que identifica o

veículo ('Histórico - Veículo 1') e um botão de retorno. O corpo da tela é composto por uma lista de todas as sessões de carregamento associadas àquele veículo, onde cada item detalha a energia consumida (kWh), a duração da sessão e a data/hora do evento.

A porta de entrada para a área de gerenciamento é uma tela de login dedicada, construída com *React*, *TypeScript* e *Vite*, apresentada na Figura 7a, correspondendo a visão de um administrador do sistema. O formulário foi projetado para ser limpo e objetivo, solicitando as credenciais de administrador. A lógica de autenticação é robusta, consumindo o *endpoint* seguro POST /api/auth/login do *backend*. Em caso de sucesso, a aplicação armazena o *token* JWT retornado no *localStorage* do navegador e redireciona o administrador para o *dashboard* principal, estabelecendo uma sessão segura.



a) Autenticação do Painel de Administração



b) Listagem e resumo de atividades dos usuários

Figura 7: Telas de administração web.
Fonte: Própria (2026)

O *dashboard* apresentado na Figura 7b: é a tela central do painel de administração, projetado para fornecer uma visão geral e rápida do sistema. Utilizando uma chamada autenticada ao *endpoint* GET /api/admin/usuarios, a interface busca e renderiza uma lista de todos os usuários cadastrados na plataforma. Para cada usuário, é apresentado um *card* de resumo com informações essenciais, como nome, matrícula, e os detalhes da sua última sessão de carregamento. O layout com menu lateral fixo e área de conteúdo principal foi

estruturado para ser intuitivo e escalável.

A Figura 8 apresenta a tela do centro de gerenciamento para um usuário individual. O administrador tem acesso a todos os dados cadastrais, como matrícula, e-mail e telefone. A interface também consome o *endpoint* GET /api/usuarios/{id}/veiculos para exibir uma lista de todos os veículos cadastrados por aquele usuário, com suas respectivas especificações. Funcionalidades críticas de administração, como o botão "Excluir Usuário", estão disponíveis nesta tela. A ação é protegida e só pode ser executada por um administrador autenticado, que ao ser acionada, envia uma requisição DELETE para o *endpoint* seguro da API, garantindo a integridade e segurança do sistema.

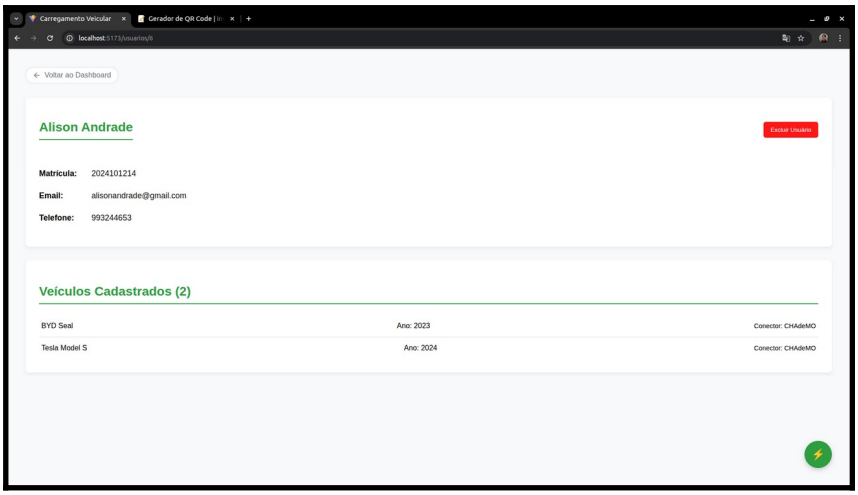


Figura 8: Tela de gerenciamento de dados e veículos de um usuário específico.

Fonte: Própria (2026)

A plataforma de administração oferece uma visão aprofundada da atividade de cada usuário. A partir da tela de detalhes, apresentada na Figura 9, o administrador pode acionar um componente de modal que exibe o histórico completo de todas as sessões de carregamento do usuário selecionado. Este modal consome o *endpoint* GET /api/usuarios/{id}/sessoes, apresentando os dados de forma clara e organizada, com informações sobre o veículo utilizado, a estação, datas e o consumo de energia de cada sessão. O uso do modal

proporciona uma experiência de usuário fluida, permitindo a consulta de dados detalhados sem a necessidade de sair da página principal de gerenciamento do usuário.

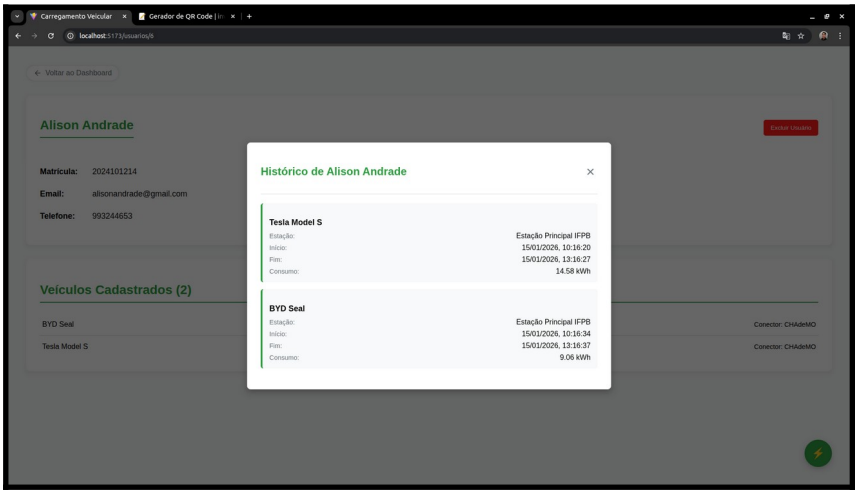


Figura 9: Modal de histórico de carregamento completo do usuário.

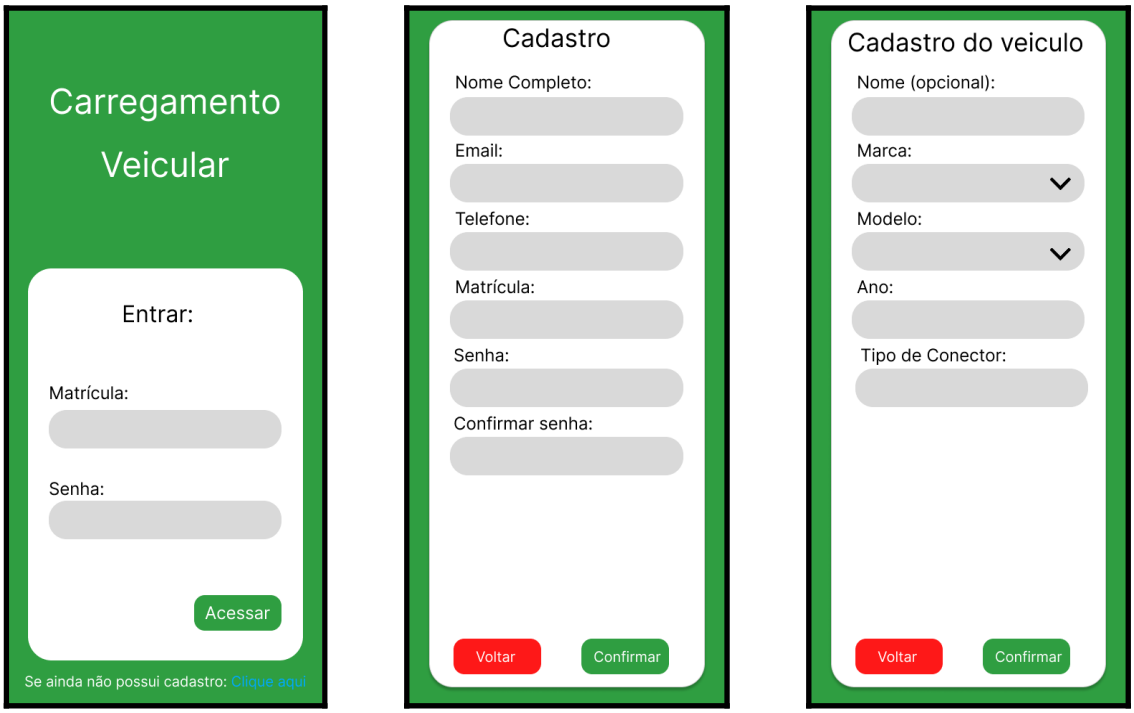
Fonte: Própria (2026)

4.3 Aplicação Mobile

Para a aplicação mobile, o processo de prototipação foi guiado pelas funcionalidades chave definidas para o sistema, com foco na usabilidade em dispositivos móveis. Utilizando a plataforma Figma, foi criado um protótipo visual que simula a interação do usuário e serve como um guia para o desenvolvimento. As telas a seguir detalham o fluxo de navegação e as principais funcionalidades do aplicativo.

A Figura 10a exibe o protótipo da tela de login, contendo os campos para 'Matrícula' e 'Senha', o botão 'Acessar' e um link para o cadastro de novos usuários. A Figura 10b apresenta o protótipo da tela de cadastro do usuário. O formulário solicita um conjunto de informações para a criação de uma nova conta, incluindo 'Nome Completo', 'Email', 'Telefone', 'Matrícula', e a definição de uma 'Senha' com seu campo de confirmação. Na parte inferior, os botões 'Confirmar' e 'Voltar' permitem, respectivamente, submeter os dados para registro ou cancelar

a operação e retornar à tela anterior. A Figura 10c ilustra o protótipo da tela para o cadastro de veículos. Neste formulário, o usuário deve fornecer informações essenciais do seu veículo, como 'Marca', 'Modelo', 'Ano' e Tipo de conector. Os campos 'Marca' e 'Modelo' são apresentados como menus de seleção (*dropdowns*) para garantir a padronização dos dados, e há um campo opcional para um nome personalizado do veículo. A parte inferior da tela contém os botões de ação 'Confirmar', para salvar o registro do veículo, e 'Voltar', para cancelar a operação.



a) Login b) Cadastro de usuário c) Cadastro de veículo

Figura 10: Protótipos de telas de login, cadastro de usuário e de veículos do sistema mobile.

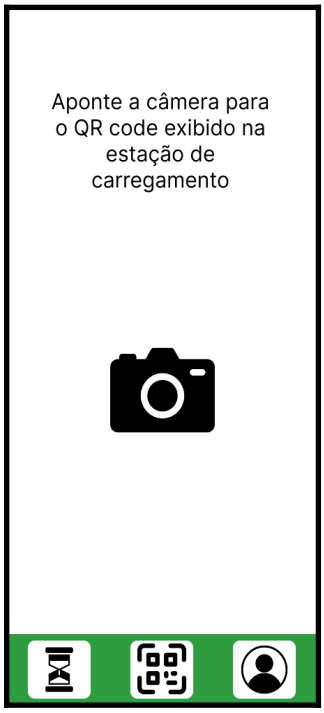
Fonte: Própria (2026)

A Figura 11a exibe o protótipo da tela principal do aplicativo, que apresenta o histórico de carregamentos do usuário. Cada item da lista representa um carregamento individual, exibindo informações cruciais como a quantidade de energia consumida em kWh, a duração do carregamento e a data/hora em que ocorreu. Na parte inferior da tela, uma barra

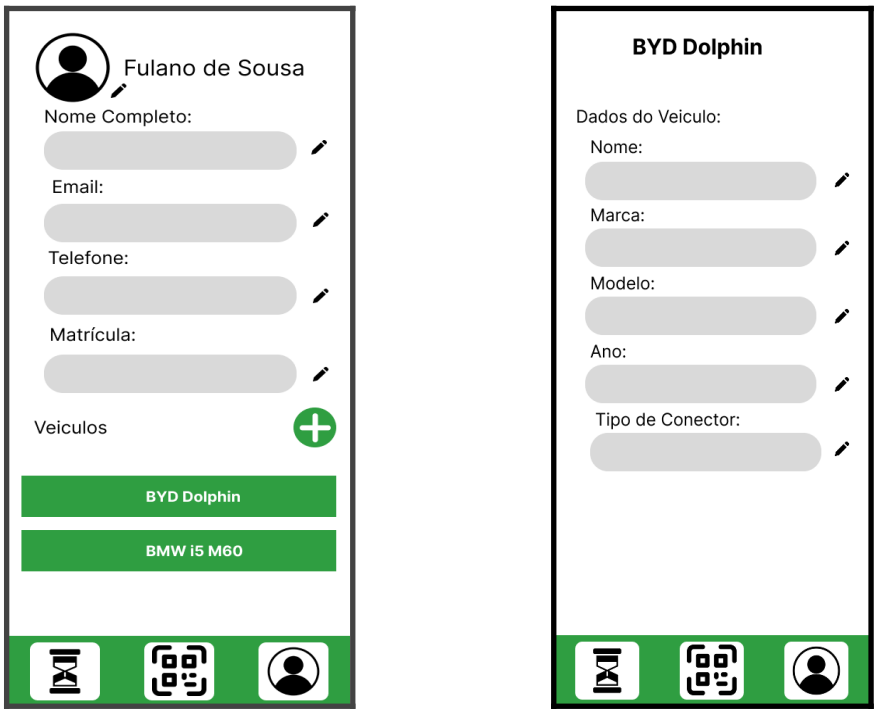
de navegação (*navbar*) oferece acesso rápido a outras seções importantes do aplicativo: o histórico (tela atual), a leitura de QR Code para iniciar um novo carregamento, e o perfil do usuário. A Figura 11b demonstra o protótipo da tela para iniciar um novo carregamento.

Acessada pelo ícone central da barra de navegação, esta interface é projetada para ser simples e funcional. Ela instrui o usuário a apontar a câmera do dispositivo para o QR Code presente na estação de carregamento, utilizando um texto claro no topo e um ícone de câmera centralizado para guiar a ação. A barra de navegação permanece visível na parte inferior, permitindo que o usuário alterne facilmente para outras seções do aplicativo. A Figura 11c apresenta a versão atualizada do protótipo da tela de perfil do usuário. Além de centralizar os dados cadastrais pessoais, como 'Nome Completo', 'Email' e 'Matrícula', todos editáveis, a interface incorpora uma seção dedicada ao gerenciamento de 'Veículos'. Nesta área, o usuário pode visualizar uma lista de seus veículos cadastrados e adicionar novos veículos ao seu perfil através do botão de adição (+), tornando a gestão mais completa e centralizada.

a) Histórico



b) QR Code



c) Perfil atualizado d) Detalhes do veículo

Figura 11: Protótipo das telas de histórico, QR Code, perfil e detalhes veículo do sistema mobile.

Fonte: Própria (2026)

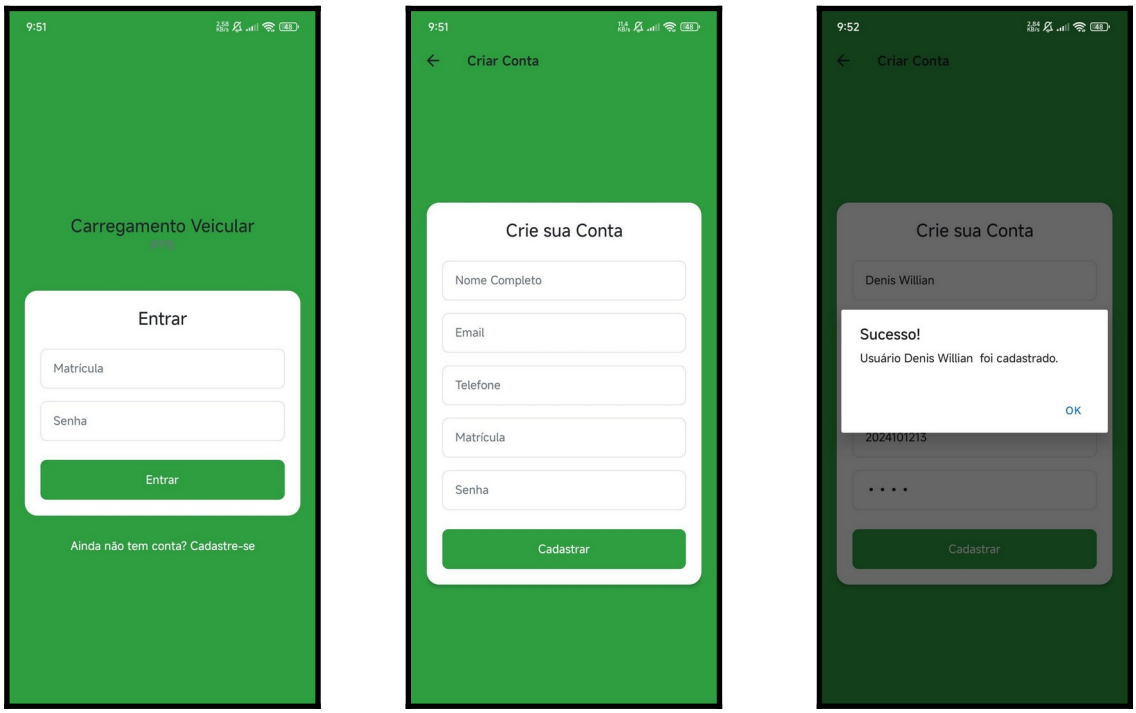
Complementando a tela de perfil, a Figura 11d detalha a interface para a visualização e edição de um veículo específico. Esta tela é acessada quando o usuário seleciona um dos veículos da sua lista. Ela exibe os 'Dados do Veículo', como 'Nome', 'Marca', 'Modelo', 'Ano' e Tipo de conector em um formato de formulário. Cada campo é acompanhado por um ícone de edição, indicando que o usuário pode atualizar as informações do veículo diretamente nesta interface.

Tendo como base os protótipos de tela apresentados nas Figuras 10 e 11, a implementação das telas do aplicativo mobile, foi realizada em *React Native* com *TypeScript*, com o uso de *hooks* modernos do *React* para criar uma interface reativa e eficiente. O *hook useFocusEffect*, do *React Navigation*, é utilizado para disparar a busca de dados (carregarVeiculos) toda vez que a tela entra em foco, garantindo que as informações exibidas,

como a lista de veículos do usuário, estejam sempre atualizadas. A lógica de chamada à API é encapsulada na camada de serviço (veiculoService), mantendo o componente da tela limpo e focado na renderização da UI.

O *AuthContext* é o responsável por gerenciar o estado de autenticação em toda a aplicação mobile. Utilizando o *Context* API do *React*, ele provê o estado do usuário e o *token* JWT para todos os componentes. O *hook useEffect* é utilizado para carregar o *token* salvo no *AsyncStorage* durante a inicialização do app, garantindo a persistência da sessão. A função login demonstra como, após a autenticação bem-sucedida, o *token* é salvo tanto no estado do *React* quanto no armazenamento persistente do dispositivo, além de configurar a instância do axios para enviar o *token* automaticamente em requisições futuras.

A tela inicial para autenticação do usuário é apresentada na Figura 12a. O formulário solicita matrícula e senha, que são enviadas de forma segura para o *endpoint* de login da API. O design utiliza um card centralizado sobre o fundo verde, alinhado com a identidade visual do projeto. A tela para registro de novos usuários na plataforma é apresentada na Figura 12b. Os campos de texto (TextInput) são customizados para seguir o design do aplicativo e utilizam tipos de teclado específicos para cada entrada de dados (e-mail, telefone), melhorando a usabilidade. Após o envio dos dados e a confirmação de sucesso do *backend* (*status 201 Created*), o aplicativo exibe um alerta nativo para informar ao usuário que seu cadastro foi concluído com sucesso, conforme apresentado na Figura 12c. Ao confirmar, o usuário será redirecionado para a tela de login.



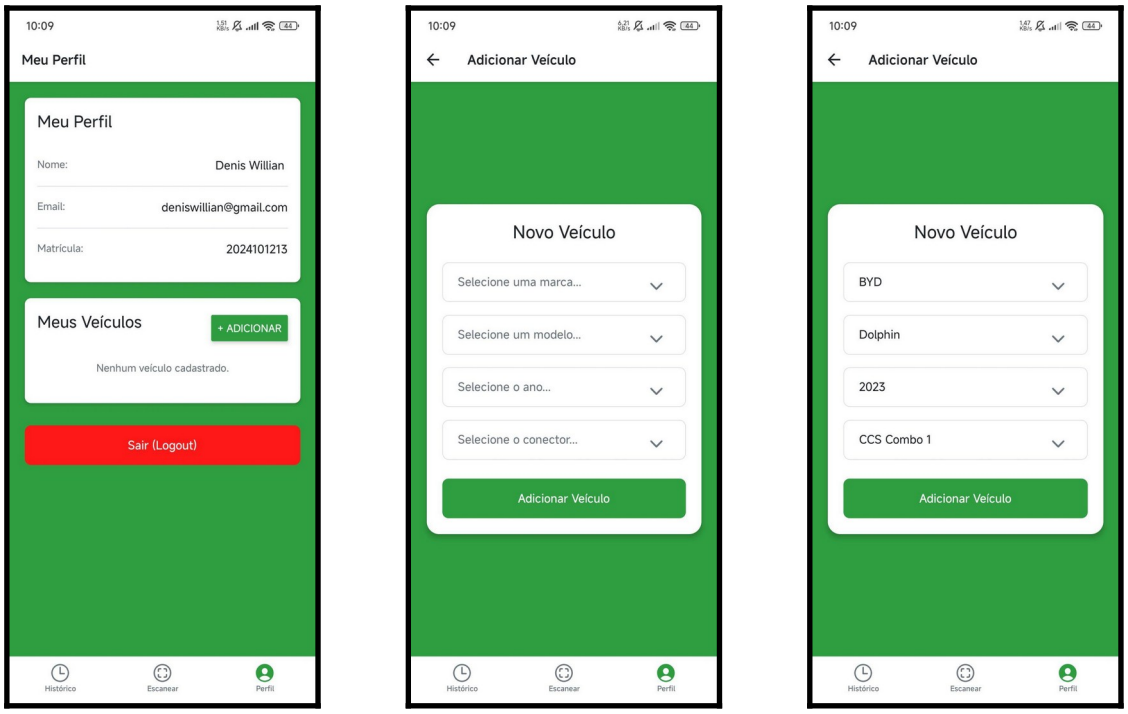
a) Login b) Novo usuário c) Gerenciamento de veículo

Figura 12: Telas de login, novo usuário e sucesso no cadastro do aplicativo mobile.

Fonte: Própria (2026)

A tela de Perfil, apresentada na Figura 13a exibe os dados do usuário logado e a seção "Meus Veículos". Nesta imagem, o usuário ainda não possui veículos cadastrados, e a aplicação exibe uma mensagem indicando que a lista está vazia. O formulário para cadastro de veículos, acessado a partir da tela de Perfil, é apresentado na Figura 13b. Utiliza componentes de seleção (*Picker*) para os campos Marca, Modelo, Ano e Conector, guiando o usuário e garantindo a consistência dos dados. Após o usuário selecionar a marca "BYD", o campo "Modelo" é automaticamente populado com os modelos correspondentes àquela marca, conforme apresentado na Figura 13c. Essa lógica de dependência foi implementada no *frontend* para criar uma experiência de usuário mais inteligente e reduzir erros.

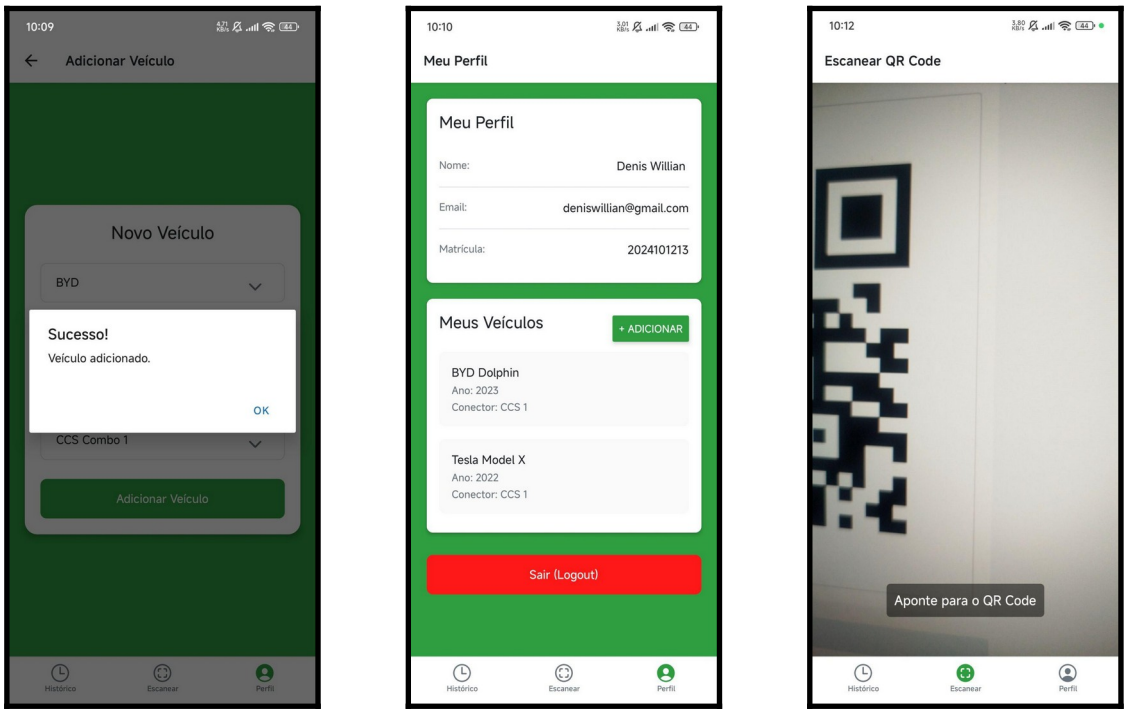
Ao submeter o formulário, a aplicação envia os dados para o *endpoint* POST /api/usuarios/{id}/veiculos. Com a resposta de sucesso da API, um alerta confirma a adição do veículo, como mostrado na Figura 14a e, em seguida, redireciona o usuário de volta à tela de Perfil. A tela de Perfil consome o *endpoint* GET /api/usuarios/{id}/veiculos e renderiza a lista de veículos do usuário. A funcionalidade é reativa: a lista é atualizada automaticamente sempre que o usuário retorna à esta tela, como mostrado na Figura 14b.



a) Perfil do usuário b) Novo veículo c) Cadastro de veículo

Figura 13: Telas de perfil do usuário, novo veículo e cadastro de veículo do aplicativo mobile.

Fonte: Própria (2026)



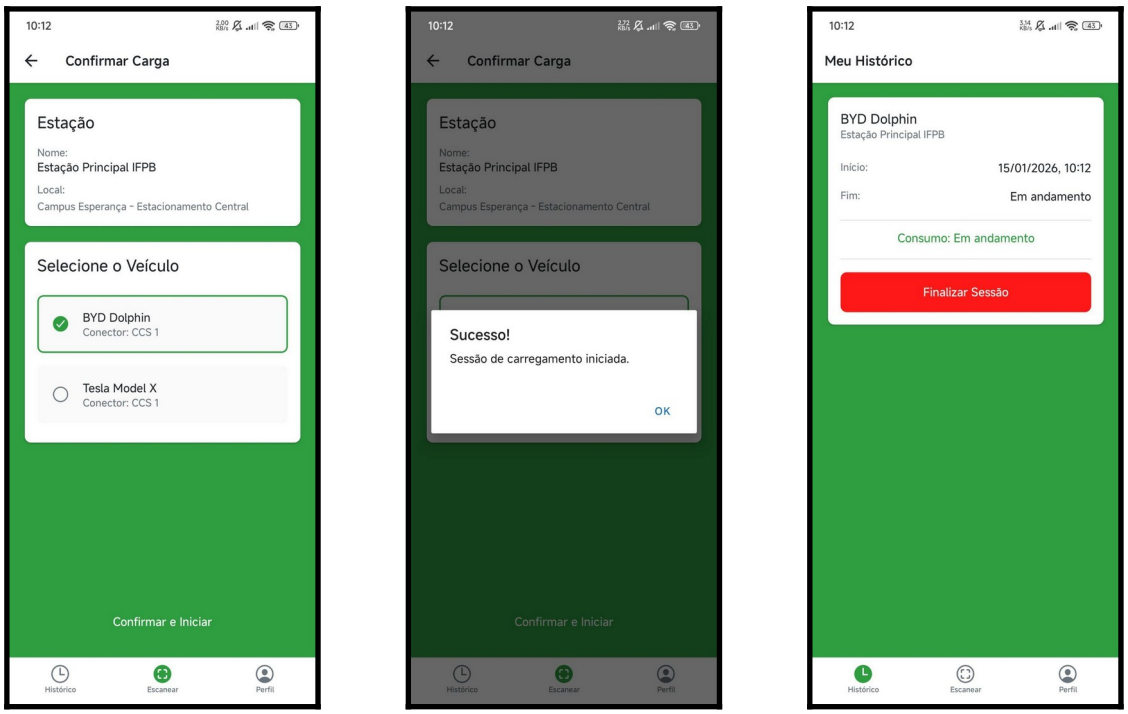
a) Veículo cadastrado b) Atualização dos dados c) Escanear QR Code

Figura 14: Telas de veículo cadastrado, atualização e escaneamento de QR Code do aplicativo mobile. Fonte: Própria (2026)

A tela "Escanear", apresentada na Figura 14c, utiliza a biblioteca *expo-camera* para acessar a câmera do dispositivo. A aplicação fica em estado de escuta, pronta para detectar um QR code e extrair a informação de identificação da estação de carregamento.

Após escanear o QR code, o aplicativo busca os dados da estação na API e exibe esta tela de confirmação, conforme apresentado na Figura 15a. O usuário pode visualizar as informações da estação e, crucialmente, selecionar qual de seus veículos cadastrados deseja utilizar para a recarga. Ao clicar em "Confirmar e Iniciar", o aplicativo envia os IDs do usuário, veículo e estação para o *endpoint* POST /api/sessoes/iniciar. Um alerta de sucesso confirma que a sessão foi iniciada no *backend*. Na tela "Meu Histórico", apresentada na Figura 15b, uma sessão que acabou de ser iniciada é exibida com os status "Em andamento". Crucialmente, um botão "Finalizar Sessão" (na cor de alerta) é renderizado condicionalmente,

dando ao usuário o controle para encerrar a recarga, conforme apresentado na Figura 15c.

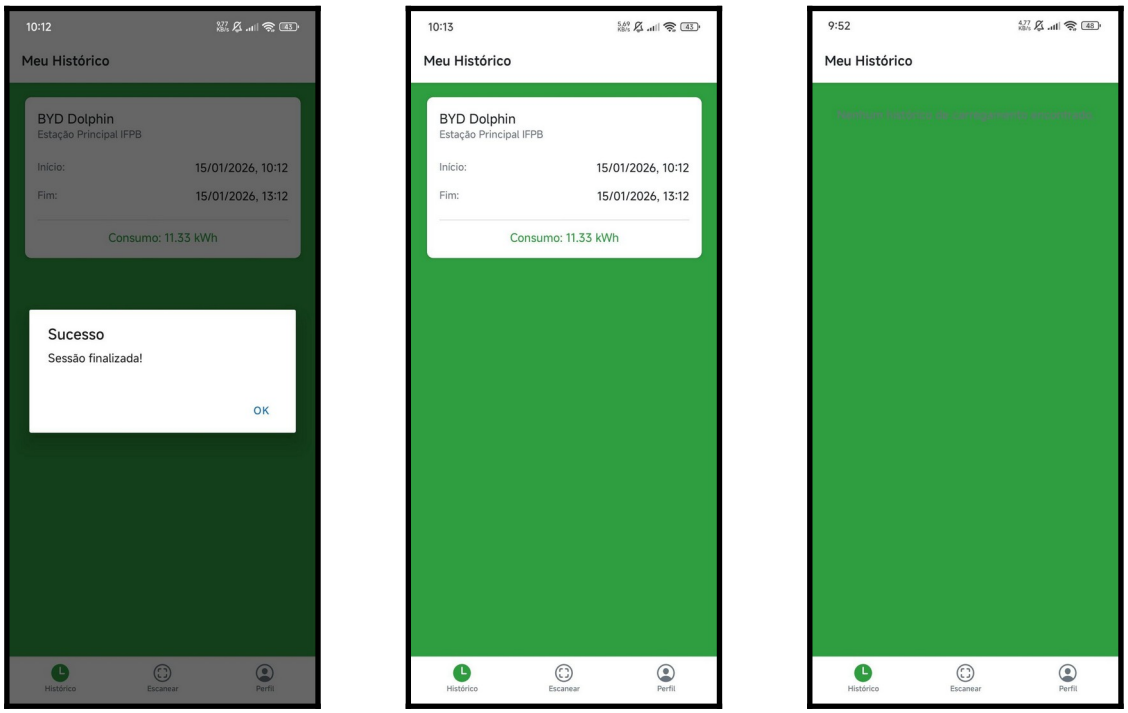


a) Seleção de veículo b) Início da sessão c) Veículo carregando

Figura 15: Tela de seleção de veículo, início da sessão e veículo carregando, do aplicativo mobile.

Fonte: Própria (2026)

Ao pressionar o botão "Finalizar Sessão", o aplicativo envia a requisição correspondente ao *backend*. Um alerta de sucesso informa ao usuário que a operação foi concluída, apresentada na Figura 16a. Após a finalização, a tela de Histórico é atualizada. O mesmo item agora exibe os dados finais da sessão, como a hora de término e o total de energia consumida, como apresentado na Figura 17b. O botão "Finalizar Sessão" desaparece, e o card se torna um registro permanente. A aplicação trata o caso em que um usuário ainda não possui sessões de carregamento, exibindo uma mensagem informativa em vez de uma lista vazia, o que melhora a clareza da interface, como apresentado na Figura 16c.



a) Sessão finalizada b) Histórico atualizado c) Histórico vazio

Figura 16: Tela de finalização de sessão e histórico do aplicativo mobile.

Fonte: Própria (2026)

4.4 Backend

O *backend* pode ser visto como responsável pelo processamento interno, regras do negócio e gerenciamento de dados do sistema. É uma parte da aplicação que atua nos bastidores, ficando invisível ao usuário final do sistema. Para exemplificar o desenvolvimento do *backend* para o sistema de controle de carregamento veicular, são apresentados na sequência dois trechos de código: o primeiro relativo à regra do negócio e o segundo relativo à arquitetura de segurança da API.

O trecho de código apresentado na Figura 17, extraído da classe SessaoCargaService, demonstra a implementação da principal regra de negócio do sistema. O método iniciarSessao é anotado com @Transactional, garantindo a atomicidade das operações no banco de dados. A

lógica orquestra múltiplos repositórios para validar a existência do usuário, do veículo e da estação. Além disso, aplica uma regra de negócio crucial ao verificar se a estação está com o status DISPONIVEL antes de alterar seu estado para OCUPADO e, finalmente, criar e persistir o novo registro de SessaoCarga.

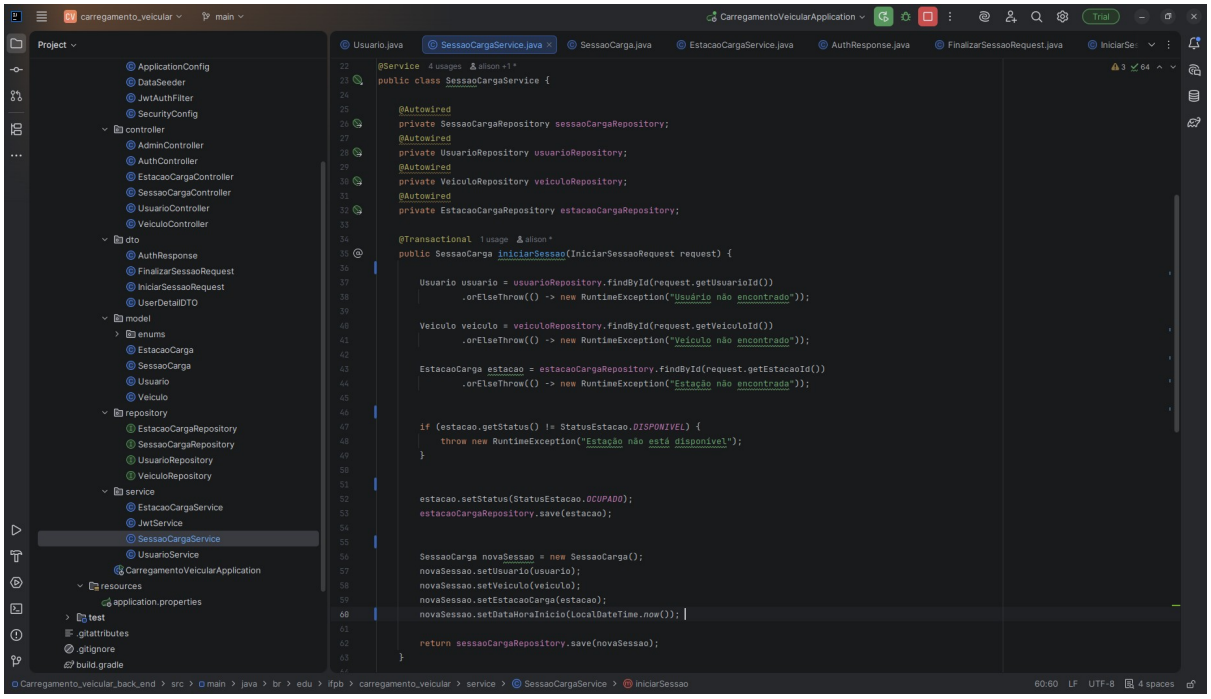


Figura 17: Implementação da lógica de negócio para início de uma sessão de carregamento.

Fonte: Própria (2026)

Já o trecho da classe SecurityConfig, apresentado na Figura 18, é o centro da arquitetura de segurança da API. Ele define a cadeia de filtros de segurança (SecurityFilterChain) e estabelece as regras de autorização. Notavelmente, a configuração desabilita o CSRF e define a política de gerenciamento de sessão como STATELESS, essencial para uma API baseada em tokens. As regras de authorizeHttpRequests são configuradas para permitir acesso público a rotas de autenticação (/api/auth/**) e, crucialmente, restringir o acesso a rotas administrativas (/api/admin/**) apenas para usuários

com o cargo ADMIN, demonstrando a implementação de autorização baseada em papéis (Role-Based Access Control).

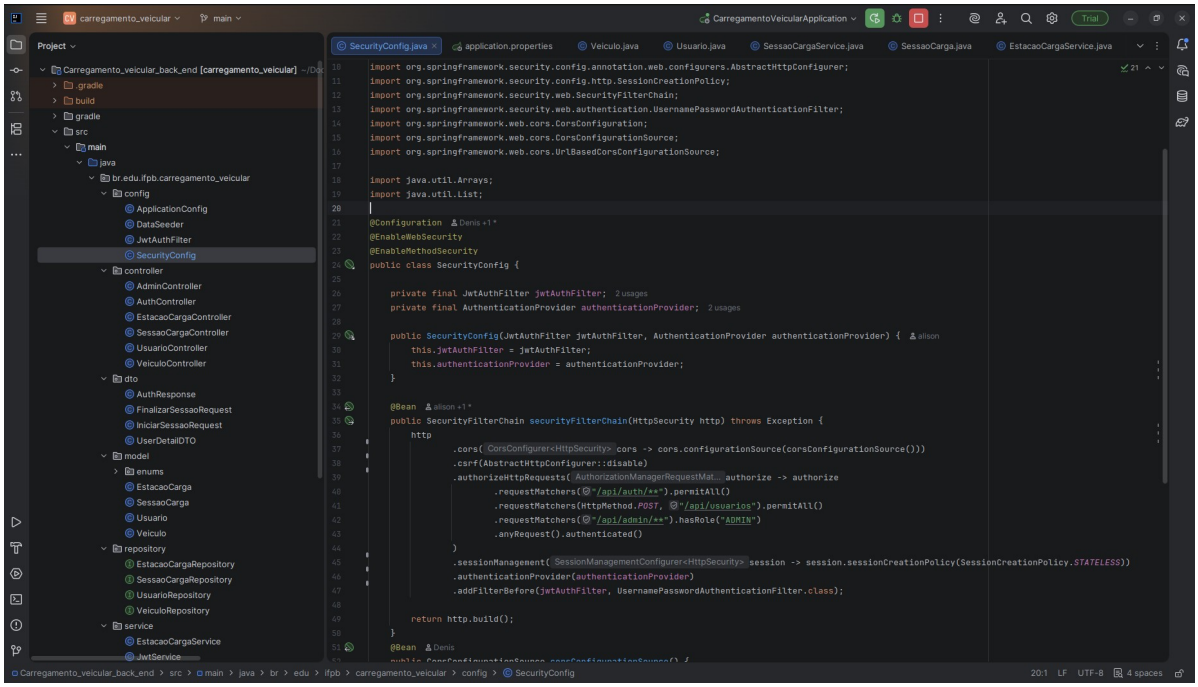


Figura 18: Configuração da cadeia de filtros de segurança e autorização por cargos.

Fonte: Própria (2026)

5. CONCLUSÃO

Neste artigo foi apresentado o desenvolvimento de uma aplicação web/mobile para o gerenciamento de carregamento de veículos elétricos. O desenvolvimento da aplicação web/mobile demonstrou a viabilidade do uso de tecnologias modernas e integradas para a gestão eficiente de um sistema de carregamento veicular: para o *frontend* foi utilizado *React* para a construção da aplicação web e *React Native* para a versão mobile, *Spring Boot* para o *backend*, PostgreSQL para o banco de dados, e por fim, o *Docker* para empacotamento da aplicação em contêineres permitindo a portabilidade e a escalabilidade da solução.

A solução proposta atende aos requisitos de controle, monitoramento e organização do processo de recarga, permitindo, além disso, a escalabilidade, manutenção facilitada e possibilidade de evolução do sistema, como a integração com redes inteligentes, novos dispositivos e funcionalidades avançadas de análise de dados. Dessa forma, o projeto contribui de forma relevante para o avanço de soluções tecnológicas voltadas à sustentabilidade e ao crescimento do uso de veículos elétricos.

Financiamento

Este trabalho foi desenvolvido com apoio financeiro do Instituto Federal da Paraíba - Campus Esperança, por meio da Chamada Interconecta IFPB - Edital Nº 01/2025 - Apoio a projetos de pesquisa, inovação, desenvolvimento tecnológico e social, contemplado com ajuda de custo de taxa de bancada e bolsa discente.

Conflito de interesses

Os autores declaram não haver conflito de interesses.

REFERÊNCIAS

BARBOSA, Fabio Junior Barros. **Ampliação e modernização do projeto Meu-TCC: uma abordagem para aprimorar a eficiência e a escalabilidade do backend**. 2023. 72 f. Trabalho de Conclusão de Curso (Graduação em Tecnologias da Informação e Comunicação), Universidade Federal de Santa Catarina, Araranguá, 2023. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/248928/TCC.pdf?sequence=1&isAllowed=y. Acesso: 28 de agosto de 2025.

BRAGA, Rondinelly Castelo. **Desenvolvimento nativo vs React Native: um estudo de portabilidade do jogo híbrido Coup Acessível**. 2019. 39 f. Trabalho de Conclusão de Curso (Graduação em Sistemas e Mídias Digitais), Universidade Federal do Ceará, Instituto UFC Virtual, Fortaleza, 2019. Disponível em: https://repositorio.ufc.br/handle/riufc/55977.

Acesso: 28 de agosto de 2025.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). Demanda de Energia dos Veículos Leves: 2025-2034. Nota Técnica EPE/DPG/SDB/2024/10. Rio de Janeiro: EPE, dez. 2024.

LARMINIE, J.; LOWRY, J. Electric vehicle techonoly explained. 2ª. ed. Chennai: John Wiley & Sons Ltd, 2012.

LINS, Gabriel de Souza. **Utilizando ReactJS para o desenvolvimento de um sistema de alocação e reserva de salas no campus da UFC em Quixadá.** 2019. 36 f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2019. Disponível em: https://repositorio.ufc.br/handle/riufc/49762 . Acesso: 28 de agosto de 2025.

MORAES, Tássio Ricardo Silva de. **Desenvolvimento de API REST com Spring Boot para integração de dados de produção do Hospital das Clínicas da Universidade Federal de Pernambuco**. 2024. 47 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Instituto Federal de Ciência e Tecnologia de Pernambuco, Recife, 2024. Disponível em: https://repositorio.ifpe.edu.br/xmlui/handle/123456789/1489. Acesso: 28 de agosto de 2025.

RENAULT. NOVO RENAULT KWID E-TECH 100% ELÉTRICO. Disponível em: https://www.renault.com.br/veiculos-eletricos/kwid-e-tech.html?CAMPAIGN=br-pt-r-t-def-model-kwid-etech-go-classic-shop-institucional&ORIGIN=sea_defensive&gclsrc=aw.ds&&gclid=CjwKCAjwt7SWBhAnEiwAx8ZLavCmUQD2iLgmpuWt3wM20YdRwLD65sFBfNvsGN9p86ySi9cXGA9-fBoC7v0QAvD_BwE. Acesso em: 31 de agosto de 2025.

SILVA, João Victor Fernandes de Souza. **Comparação de desempenho entre os bancos de dados PostgreSQL e Neo4j para acesso a dados complexos**. 2023. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação), Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2023. Disponível em: https://repositorio.ufu.br/handle/123456789/38755. Acesso: 28 de agosto de 2025.

SOUZA, L. A. D. Michael Faraday, 2022. Disponível em: https://brasilescola.uol.com.br/quimica/michael-faraday.htm. Acesso em: 31 de agosto de 2025.

Recebido em: 23/01/2026
Aprovado em: 06/02/2026
Publicado em: 24/02/2026