INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS.
CAMPUS COLINAS DO TOCANTINS - TO
LICENCIATURA EM COMPUTAÇÃO

WENDEL ALVES SOUSA

DESENVOLVIMENTO DE UM SISTEMA WEB PARA CADASTRO E GERENCIAMENTO DE VAGAS DE EMPREGO

COLINAS DO TOCANTINS - TO
2026

WENDEL ALVES SOUSA

DESENVOLVIMENTO DE UM SISTEMA WEB PARA CADASTRO E GERENCIAMENTO DE VAGAS DE EMPREGO

Utilizando JavaScript, Node.js, React e SQL Server

Trabalho de Conclusão de Curso apresentado ao curso de Licenciatura em Computação do INSTITUTO FEDERAL DE EDUCAÇÃO. CIÊNCIA E TECNOLOGIA DO TOCANTINS, como requisito parcial para obtenção do título de Licenciatura em Computação.

Orientador: MATEUS NUNES DOS SANTOS

COLINAS DO TOCANTINS - TO

2026

Dados Internacionais de Catalogação na Publicação (CIP)
Bibliotecas do Instituto Federal do Tocantins

S725d Sousa, Wendel Alves
DESENVOLVIMENTO DE UM SISTEMA WEB PARA CADASTRO E GERENCIAMENTO DE VAGAS DE EMPREGO : Utilizando JavaScript, Node.js, React e SQL Server / Wendel Alves Sousa. – Colinas do Tocantins, TO, 2026.
79 p.

Trabalho de Conclusão de Curso (Licenciatura em Computação) – Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, Campus Colinas do Tocantins, Colinas do Tocantins, TO, 2026.

Orientador: Esp. Mateus Nunes Dos Santos

1. Sistemas web. 2. Vagas de emprego. 3. Desenvolvimento de software. I. Nunes Dos Santos, Mateus. II. Título.

CDD 004

Elaborado pelo sistema de geração automatica de ficha catalográfica do IFTO com os dados fornecidos pelo(a) autor(a).

WENDEL ALVES SOUSA

DESENVOLVIMENTO DE UM SISTEMA WEB PARA CADASTRO E GERENCIAMENTO DE VAGAS DE EMPREGO

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Licenciatura em Computação do Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – Campus Colinas do Tocantins, como requisito parcial para a obtenção do grau de Licenciado em Computação.

Aprovado Em 08/06/2026.

Banca Examinadora:

Documento assinado digitalmente
gov.br
MATEUS NUNES DOS SANTOS
Data: 24/06/2026 16:28:37-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Esp Mateus Nunes Dos Santos.

Orientador

Instituto Federal Do Tocantins – Campus Colinas

Documento assinado digitalmente
gov.br
FERNANDO TURIBIO DE MOURA
Data: 25/06/2026 11:47:45-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Me. Fernando Turibio de Moura.

Examinador 1

Instituto Federal Do Tocantins – Campus Colinas

Documento assinado digitalmente
gov.br
LUIS ALBERTO LIBANIO LIMA
Data: 25/06/2026 12:02:56-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Me. Luis Alberto Libanio Lima.

Examinador 2

Instituto Federal Do Tocantins – Campus Colinas

Dedico este trabalho a Deus, por me conceder força, sabedoria e perseverança ao longo desta jornada.

À minha mãe, Maria Raimunda Alves de Sousa Ferreira, e ao meu padrasto, Devaldo Ferreira, pelo apoio, incentivo e por sempre acreditarem em mim.

À minha esposa, Kesia Layanne, pela compreensão, parceria e apoio nos momentos mais desafiadores.

À minha falecida avó, Olinda Alves de Sousa Luz, que, mesmo com pouco estudo, foi uma grande inspiração em minha vida, ensinando-me o valor da persistência, da dedicação e da superação.

A todos vocês, que estiveram presentes nos momentos em que pensei em desistir, deixo aqui minha profunda gratidão.

AGRADECIMENTOS

Agradeço a Deus, por sua infinita misericórdia, pois sem Ele não teria chegado a esta etapa final. Em meio às dificuldades, aos desafios emocionais e espirituais, foi Ele quem me manteve de pé, forte e perseverante. Durante o período da pandemia, momento em que muitos colegas desistiram, encontrei em Deus a força necessária para continuar, com saúde, coragem e determinação. Muito obrigado, meu Deus.

Aos meus professores, expresso minha sincera gratidão por todo o conhecimento compartilhado ao longo desta trajetória. Cada aula representou uma nova oportunidade de aprendizado e crescimento. Mesmo diante das dificuldades iniciais de compreensão, a dedicação, paciência e compromisso de vocês com o ensino foram fundamentais para que eu chegasse até aqui.

Confesso que, no início, não imaginava cursar esta formação, tampouco conhecia profundamente a instituição e o curso de Licenciatura em Computação do IFTO. No entanto, ao longo dessa jornada, fui surpreendido positivamente, não apenas pelo conteúdo acadêmico, mas também pelo desenvolvimento profissional e social proporcionado.

Vocês tiveram um papel essencial na construção da minha trajetória, despertando em mim não apenas o interesse pelo conhecimento, mas também a motivação para compartilhar essa experiência com outras pessoas. Levo comigo aprendizados que vão além da sala de aula.

A todos vocês, minha profunda gratidão. Que Deus os abençoe grandemente.

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de um sistema web para cadastro e gerenciamento de vagas de emprego, visando facilitar a intermediação entre empresas e candidatos. Diante do avanço das tecnologias digitais e da crescente demanda por soluções eficientes no processo de recrutamento e seleção, observa-se a necessidade de plataformas mais simples, acessíveis e organizadas.

A metodologia adotada baseia-se no desenvolvimento de uma aplicação web utilizando tecnologias modernas, como JavaScript, Node.js e no back-end, React no front-end e SQL Server como sistema gerenciador de banco de dados. O sistema proposto busca oferecer uma interface intuitiva, com foco na usabilidade e na experiência do usuário, além de garantir eficiência na organização e recuperação das informações.

Como resultado esperado, pretende-se disponibilizar uma plataforma capaz de otimizar o processo de divulgação e busca por vagas de emprego, contribuindo para maior agilidade, organização e acessibilidade no mercado de trabalho. Além disso, o projeto proporciona a aplicação prática de conceitos relacionados ao desenvolvimento web moderno.

Palavras-chave: Sistema web. Vagas de emprego. Desenvolvimento web. Banco de dados.

ABSTRACT

This work aims to develop a web system for job vacancy registration and management, with the purpose of facilitating the interaction between companies and candidates. Given the advancement of digital technologies and the increasing demand for efficient solutions in recruitment and selection processes, there is a need for platforms that are more accessible, organized, and user-friendly.

The methodology is based on the development of a web application using modern technologies, such as Node.js for the back-end, React for the front-end, and SQL Server as the database management system. The proposed system focuses on usability and user experience, ensuring efficiency in data organization and retrieval.

As an expected result, the system aims to provide a platform capable of optimizing the process of job posting and searching, contributing to greater agility, organization, and accessibility in the job market. Additionally, the project enables the practical application of modern web development concepts.

Keywords: Web system. Job vacancies. Web development. Database.

Lista de figuras

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

CSS – Cascading Style Sheets

DER – Diagrama Entidade-Relacionamento

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IBGE – Instituto Brasileiro de Geografia e Estatística

IFTO – Instituto Federal de Educação, Ciência e Tecnologia do Tocantins

JSON – JavaScript Object Notation

JWT – JSON Web Token

MVC – Model-View-Controller

NPM – Node Package Manager

ORM – Object-Relational Mapping

REST – Representational State Transfer

SPA – Single Page Application

SQL – Structured Query Language

TCC – Trabalho de Conclusão de Curso

UI – User Interface

UX – User Experience

SGBD – Sistema Gerenciador de Banco de Dados

SUMÁRIO

1 INTRODUÇÃO

O mercado de trabalho contemporâneo tem sido fortemente impactado pelos avanços das tecnologias digitais, promovendo transformações significativas na forma como empresas e profissionais se conectam. Nesse contexto, os sistemas web têm se consolidado como ferramentas essenciais para a intermediação de vagas de emprego, proporcionando maior agilidade, alcance e eficiência nos processos de recrutamento e seleção.

Apesar da ampla disponibilidade dessas plataformas, muitas soluções existentes ainda apresentam limitações relevantes, tais como interfaces pouco intuitivas, excesso de informações, presença de anúncios e custos elevados para utilização de funcionalidades essenciais. Esses fatores podem comprometer a experiência do usuário e dificultar o acesso, especialmente para pequenas e médias empresas, bem como para candidatos que buscam oportunidades de forma mais objetiva.

> O software tornou-se um elemento essencial para empresas, governos e sociedade, estando presente em praticamente todas as atividades modernas. (Pressman e Maxim, 2021, p. 5).

Diante desse cenário, este trabalho propõe o desenvolvimento de um sistema web voltado ao cadastro e gerenciamento de vagas de emprego, com ênfase na usabilidade, organização das informações e acessibilidade. A proposta tem como objetivo oferecer uma solução eficiente e de fácil utilização, que favoreça a interação entre empregadores e candidatos de maneira clara e direta.

Para o desenvolvimento do sistema, serão utilizadas tecnologias modernas amplamente adotadas no mercado, como Node.js, React e SQL Server, visando garantir desempenho, escalabilidade e uma interface responsiva e amigável ao usuário. Dessa forma, espera-se contribuir tanto para a melhoria dos processos de intermediação de emprego quanto para a aplicação prática de conceitos atuais no desenvolvimento de sistemas web.

1.1 OBJETIVO GERAL

Desenvolver um sistema web para cadastro e gerenciamento de vagas de emprego, permitindo a interação entre empresas e candidatos de forma eficiente.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver uma interface web utilizando React;
- Criar uma API utilizando JavaScript e Node.js;
- Implementar banco de dados com SQL Server;
- Permitir cadastro de usuários (empresas e candidatos);
- Permitir publicação e gerenciamento de vagas;
- Implementar sistema de autenticação;
- Garantir responsividade e usabilidade do sistema;

1.3 JUSTIFICATIVA

A escolha do tema justifica-se diante do cenário atual do mercado de trabalho, marcado pela constante busca por agilidade e eficiência nos processos de recrutamento e seleção. Empresas necessitam divulgar vagas de maneira organizada e acessível, enquanto candidatos procuram oportunidades compatíveis com seus perfis profissionais de forma prática e objetiva. Nesse contexto, os sistemas web tornaram-se ferramentas fundamentais para aproximar empresas e candidatos, facilitando a comunicação e o gerenciamento das informações.

Com o avanço das tecnologias da informação, diversas plataformas digitais passaram a atuar na intermediação de vagas de emprego. Entretanto, muitas dessas plataformas apresentam excesso de informações, presença de anúncios e interfaces complexas, fatores que podem comprometer a experiência do usuário e dificultar a navegação. Dessa forma, torna-se relevante o desenvolvimento de um sistema com foco em simplicidade, organização e usabilidade.

> Os sistemas de informação transformam dados em informações úteis para a tomada de decisões, coordenação e controle das organizações (Laudon, 2016, p. 12).

Diante disso, o desenvolvimento de um sistema web para cadastro e gerenciamento de vagas de emprego apresenta-se como uma solução capaz de contribuir para a melhoria dos processos de divulgação e busca por oportunidades profissionais. A proposta busca oferecer uma plataforma organizada, responsiva e intuitiva, permitindo maior praticidade tanto para empresas quanto para candidatos.

Além disso, o projeto possui relevância acadêmica e tecnológica, pois possibilita a aplicação prática de conhecimentos relacionados ao desenvolvimento web moderno, utilizando tecnologias amplamente empregadas no mercado, como JavaScript, Node.js, React e SQL Server. Dessa forma, o trabalho contribui tanto para a formação profissional do autor quanto para o desenvolvimento de uma solução voltada às necessidades atuais do mercado de trabalho.

2 REFERENCIAL TEÓRICO

2.1 TECNOLOGIAS E LINGUAGENS DE PROGRAMAÇÃO

O desenvolvimento de sistemas computacionais depende diretamente do uso de tecnologias capazes de estabelecer comunicação entre o ser humano e o computador. Nesse contexto, as linguagens de programação possuem papel fundamental, pois permitem transformar ideias, regras e comandos em instruções compreensíveis pelas máquinas. A utilização dessas linguagens tornou-se indispensável na construção de softwares, aplicações web, sistemas corporativos e diferentes recursos tecnológicos presentes no cotidiano, sendo responsáveis por orientar como os dados serão processados, armazenados e executados ao longo do funcionamento de um sistema, o que, segundo Gotardo (2015), ocorre por meio de regras específicas de escrita e interpretação.

Uma linguagem de programação pode ser compreendida como um conjunto padronizado de instruções utilizado para desenvolver programas computacionais. Ainda de acordo com Gotardo (2015), essas linguagens seguem regras sintáticas e semânticas que determinam tanto a estrutura quanto o significado dos comandos escritos pelo programador. Dessa maneira, não basta apenas inserir palavras ou símbolos aleatórios em um código, pois existe uma organização lógica necessária para que o computador consiga interpretar corretamente cada instrução recebida.

Além da organização estrutural, as linguagens de programação também são responsáveis por definir como os dados serão manipulados dentro de um sistema. Isso envolve processos relacionados ao armazenamento de informações, transmissão de dados, execução de ações específicas e tomada de decisões em determinadas condições. Nesse sentido, a programação torna-se um elemento essencial para o funcionamento das aplicações modernas, permitindo desde operações simples até atividades complexas envolvendo automação, integração de sistemas e gerenciamento de informações.

Ao utilizar uma linguagem de programação, o desenvolvedor produz aquilo que é conhecido como código-fonte. Esse código representa o conjunto de comandos escritos de acordo com as regras da linguagem adotada, permitindo que o sistema execute funções previamente planejadas. Conforme destaca Gotardo (2015), o

código-fonte é elaborado seguindo padrões sintáticos e semânticos específicos, garantindo que o computador interprete corretamente cada instrução inserida pelo programador. Dessa forma, a qualidade e a organização do código influenciam diretamente no desempenho, manutenção e compreensão do sistema desenvolvido.

Outro aspecto importante relacionado às linguagens de programação está associado à evolução tecnológica e à necessidade constante de adaptação às demandas atuais do mercado. Com o avanço da internet e das aplicações digitais, surgiram linguagens e ferramentas voltadas para diferentes finalidades, como desenvolvimento web, aplicações móveis, sistemas empresariais e bancos de dados. Esse cenário contribuiu para a criação de tecnologias mais modernas, flexíveis e eficientes, capazes de atender tanto pequenas aplicações quanto sistemas de grande porte.

As linguagens de programação também possuem relevância significativa na área do desenvolvimento web, principalmente devido à crescente necessidade de aplicações acessíveis, dinâmicas e interativas. Atualmente, muitas soluções computacionais utilizam diferentes tecnologias integradas entre si, permitindo que front-end, back-end e banco de dados trabalhem de maneira conjunta para oferecer melhor desempenho e experiência ao usuário. Nesse contexto, compreender o funcionamento das linguagens e ferramentas utilizadas durante o desenvolvimento torna-se fundamental para garantir aplicações mais organizadas, seguras e funcionais (Gotardo, 2015).

Diante disso, observa-se que as linguagens de programação representam a base estrutural para a construção de sistemas computacionais modernos. Sua utilização possibilita o desenvolvimento de soluções tecnológicas capazes de automatizar processos, organizar informações e atender diferentes necessidades presentes na sociedade contemporânea. Além disso, a escolha adequada das tecnologias influencia diretamente na eficiência, manutenção e escalabilidade das aplicações desenvolvidas, reforçando a importância desses recursos no cenário atual da computação.

2.1.1 JAVASCRIPT

O JavaScript é uma das linguagens de programação mais utilizadas no desenvolvimento web contemporâneo, sendo amplamente empregado na criação de aplicações dinâmicas, interativas e funcionais. Sua popularidade está relacionada principalmente à capacidade de atuar em diferentes camadas do desenvolvimento, permitindo que programadores utilizem a mesma linguagem tanto na construção das interfaces visuais quanto na lógica executada no servidor. Essa característica contribui para maior praticidade durante o desenvolvimento de sistemas modernos, especialmente em aplicações web que necessitam de integração constante entre front-end e back-end.

Inicialmente criado com o objetivo de adicionar interatividade às páginas web, o JavaScript passou por diversas evoluções ao longo do tempo. A linguagem deixou de ser utilizada apenas para pequenas animações ou validações simples dentro do navegador e passou a ocupar papel central no desenvolvimento de aplicações completas. Atualmente, segundo Mororó (2024), o JavaScript é considerado uma linguagem de alto nível orientada a objetos, sendo amplamente utilizada no desenvolvimento de aplicações web modernas devido à sua flexibilidade e capacidade de adaptação a diferentes cenários computacionais.

Uma das principais características do JavaScript está relacionada à sua versatilidade. No front-end, a linguagem é responsável por proporcionar dinamismo às páginas, permitindo a manipulação de elementos da interface em tempo real, atualização de conteúdos sem recarregamento da página e respostas imediatas às ações realizadas pelos usuários. Isso torna a experiência de navegação mais fluida e interativa, fator extremamente importante em aplicações web atuais.

Além da atuação no lado do cliente, o JavaScript também passou a ser utilizado no desenvolvimento back-end, ampliando significativamente suas possibilidades de uso. Com o surgimento de tecnologias voltadas para execução da linguagem fora do navegador, tornou-se possível desenvolver servidores, APIs e sistemas completos utilizando JavaScript. Essa expansão permitiu que desenvolvedores trabalhassem utilizando uma única linguagem em diferentes etapas do projeto, favorecendo maior integração entre os componentes da aplicação e redução da complexidade durante o desenvolvimento.

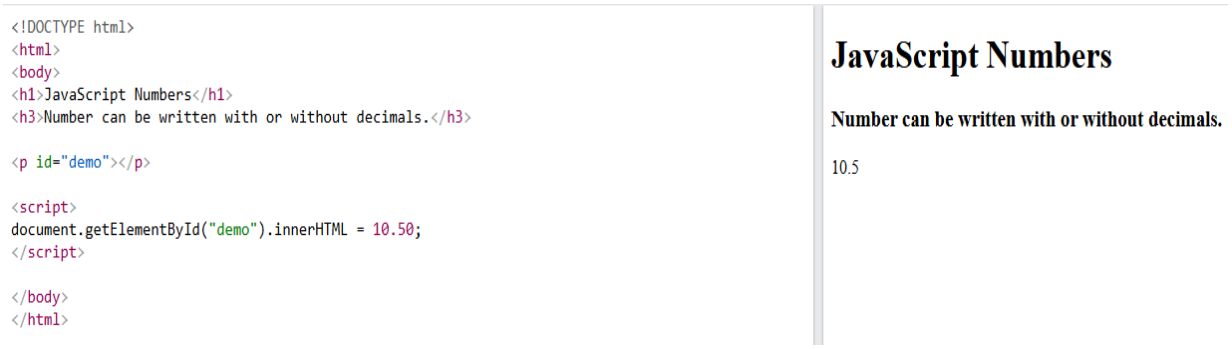
Outro ponto relevante envolve a forma como o JavaScript executa seus códigos. A linguagem é interpretada em tempo de execução, dispensando processos tradicionais de compilação antes da execução do programa. De acordo com Mororó (2024), os navegadores modernos possuem mecanismos específicos responsáveis por interpretar e executar códigos JavaScript, permitindo que aplicações web respondam rapidamente às interações realizadas pelos usuários. Isso contribui diretamente para a agilidade e eficiência das aplicações desenvolvidas com a linguagem.

O JavaScript também se destaca por possuir tipagem dinâmica, característica que oferece maior flexibilidade durante a programação. Dessa maneira, uma mesma variável pode armazenar diferentes tipos de dados ao longo da execução do sistema, sem necessidade de declarações rígidas de tipo. Além disso, a linguagem fornece suporte à programação assíncrona, permitindo que determinadas operações sejam executadas sem interromper o funcionamento principal da aplicação. Recursos como callbacks, Promises e async/await tornaram-se fundamentais no desenvolvimento moderno, principalmente em aplicações que realizam comunicação constante com servidores e bancos de dados.

A evolução contínua do JavaScript contribuiu para que a linguagem se consolidasse como uma das principais tecnologias utilizadas no cenário da computação atual. Sua capacidade de adaptação, aliada à ampla compatibilidade com navegadores, frameworks e bibliotecas, possibilitou a criação de aplicações cada vez mais robustas e eficientes. Nesse contexto, observa-se que o JavaScript não apenas revolucionou o desenvolvimento web, mas também se tornou uma ferramenta indispensável para projetos modernos que exigem desempenho, interatividade e integração entre diferentes ambientes computacionais (Mororó, 2024).

No contexto do projeto, optou-se pelo JavaScript pela facilidade de integração entre front-end e back-end, além das características citadas acima como: velocidade, inúmeras bibliotecas e frameworks

Figura 1 - exemplo de um código javascript.



Fonte: <https://www.w3schools.com/js/tryit.asp?filename=tryjs_syntax_numbers>. Acesso em: 21/05/2026

A linguagem foi escolhida pelo autor devido o código ser executado diretamente no navegador e por suportar programação orientada a objetos trazendo uma praticidade maior devido o sistema ser web.

2.1.2 NODE.JS

O crescimento das aplicações web modernas trouxe a necessidade de tecnologias mais rápidas, escaláveis e capazes de lidar com múltiplas requisições simultaneamente. Nesse cenário, o Node.js passou a ocupar posição de destaque no desenvolvimento back-end por oferecer um ambiente de execução eficiente para aplicações construídas com JavaScript. Sua proposta principal está relacionada à execução da linguagem fora do navegador, permitindo que o JavaScript também seja utilizado no lado do servidor, ampliando significativamente as possibilidades de desenvolvimento de sistemas web.

O Node.js surgiu em 2009, desenvolvido por Ryan Dahl juntamente com outros colaboradores, com o objetivo de criar uma tecnologia baseada em arquitetura não bloqueante. Segundo Barsotti e Gibertoni (2020), a proposta surgiu como alternativa aos modelos tradicionais utilizados por diversas tecnologias da época, que interrompiam determinadas operações do sistema enquanto aguardavam respostas relacionadas à entrada e saída de dados no servidor. Esse modelo tradicional acabava limitando o desempenho das aplicações, especialmente em ambientes com grande volume de acessos simultâneos.

A arquitetura não bloqueante representa uma das principais características do Node.js. Diferentemente de modelos convencionais, nos quais uma tarefa precisa ser concluída para que outra seja iniciada, o Node.js permite que várias operações sejam executadas de maneira paralela. Isso favorece melhor aproveitamento dos recursos computacionais e contribui para aplicações mais rápidas e responsivas. Dessa maneira, sistemas que dependem de múltiplas conexões simultâneas conseguem apresentar melhor desempenho, principalmente em aplicações web modernas que trabalham com troca constante de informações em tempo real.

Outro aspecto relevante está relacionado ao fato de o Node.js utilizar o mecanismo V8, desenvolvido pelo Google para o navegador Google Chrome. Esse mecanismo é responsável por interpretar e executar códigos JavaScript de maneira otimizada, proporcionando maior velocidade durante a execução das aplicações. Conforme destacam Barsotti e Gibertoni (2020), a utilização desse recurso permite que o JavaScript seja executado no servidor com alto desempenho, tornando o Node.js uma solução eficiente para construção de APIs, aplicações web e sistemas escaláveis.

Além disso, o Node.js é orientado a eventos, característica que influencia diretamente na forma como as aplicações processam ações e requisições. Nesse modelo, eventos específicos acionam funções determinadas conforme as interações realizadas no sistema. Isso contribui para maior fluidez na comunicação entre servidor e aplicação, especialmente em sistemas que necessitam de atualização constante de dados ou comunicação em tempo real, como chats, plataformas online e serviços de streaming.

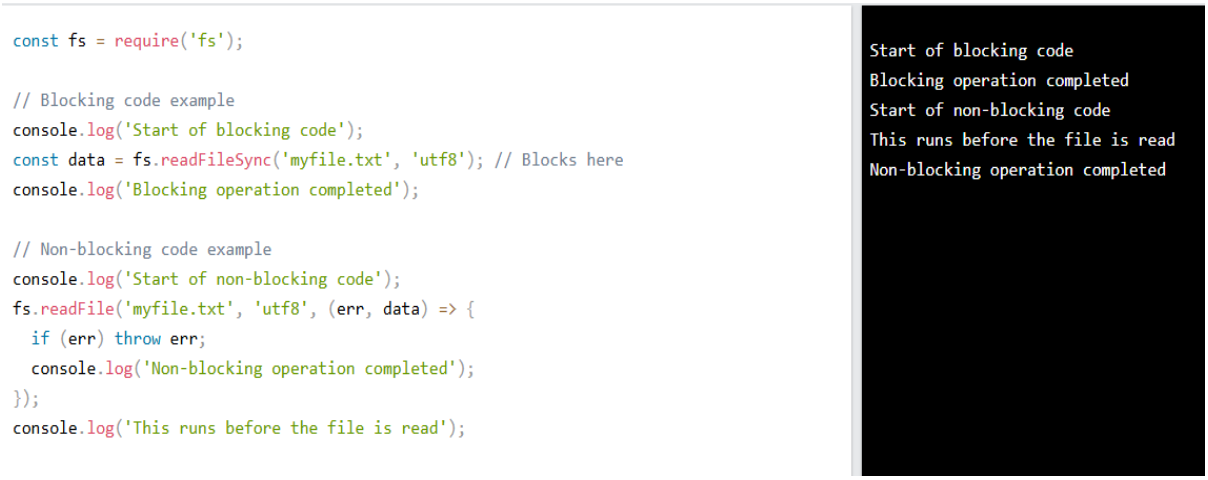
A leveza estrutural do Node.js também representa um fator importante para sua ampla adoção no mercado tecnológico. Sua arquitetura baseada em operações de entrada e saída eficientes permite melhor gerenciamento de conexões simultâneas sem comprometer significativamente o desempenho do servidor. Segundo Barsotti e Gibertoni (2020), essas características fazem com que o Node.js apresente resultados positivos em aplicações com intenso tráfego de rede, tornando-se uma tecnologia adequada para sistemas modernos que exigem rapidez e estabilidade.

Outro benefício relevante envolve a utilização da mesma linguagem tanto no front-end quanto no back-end da aplicação. Essa possibilidade facilita o desenvolvimento dos projetos, reduz a complexidade da manutenção e favorece maior integração entre as equipes responsáveis pelas diferentes etapas do sistema. Além

disso, o uso do JavaScript em ambos os lados da aplicação contribui para maior produtividade durante o desenvolvimento, uma vez que o programador não precisa alternar constantemente entre linguagens distintas.

Diante disso, percebe-se que o Node.js se consolidou como uma das principais tecnologias utilizadas no desenvolvimento back-end contemporâneo. Sua arquitetura não bloqueante, aliada à eficiência no processamento de múltiplas requisições e à integração com JavaScript, tornou a ferramenta amplamente utilizada na construção de aplicações modernas, escaláveis e voltadas para ambientes web de alto desempenho (Barsoti; Gibertoni, 2020).

Figura 2 - exemplo de um código sendo executado no node.js



Fonte: <https://www.w3schools.com/nodejs/nodejs_architecture.asp>. Acesso em: 21/05/2026

O Node.js complementa o desenvolvimento da aplicação ao permitir a execução do JavaScript no lado do servidor, proporcionando maior desempenho, escalabilidade e eficiência na construção de APIs, dashboards, integrações e aplicações em tempo real.

2.1.3 REACT.JS

O desenvolvimento de interfaces modernas para aplicações web exige ferramentas capazes de proporcionar dinamismo, organização e melhor experiência para os usuários. Nesse contexto, o React.js destaca-se como uma das bibliotecas JavaScript mais utilizadas atualmente na construção de interfaces gráficas para aplicações web. Sua popularidade está relacionada principalmente à capacidade de

criar interfaces reutilizáveis, organizadas e eficientes, facilitando o desenvolvimento de sistemas mais modernos e interativos.

Conhecido também apenas como React, o React.js foi inicialmente desenvolvido pelo Facebook com o objetivo de melhorar a construção de interfaces dinâmicas para aplicações web. Posteriormente, a tecnologia tornou-se open source, permitindo que desenvolvedores do mundo inteiro contribuíssem para sua evolução e aprimoramento. Segundo Lobo (2025), o principal propósito do React está relacionado à criação de aplicações web mais dinâmicas, utilizando conceitos que favorecem maior organização e reutilização de código durante o desenvolvimento.

Uma das características mais importantes do React.js está associada à virtualização do DOM[1]. Esse recurso permite que alterações realizadas na interface sejam processadas de maneira mais eficiente, utilizando apenas os elementos necessários sem recarregar completamente a página. Dessa forma, a aplicação consegue apresentar melhor desempenho e maior fluidez durante a interação do usuário, fator extremamente relevante em sistemas modernos que demandam respostas rápidas e constantes atualizações visuais.

O React.js também utiliza uma arquitetura baseada em componentes, permitindo que diferentes partes da interface sejam divididas em estruturas menores e reutilizáveis. Essa abordagem contribui para maior organização do projeto, além de facilitar a manutenção e reutilização de funcionalidades em diferentes partes da aplicação. Conforme destaca Lobo (2025), cada componente pode receber propriedades específicas, conhecidas como props, responsáveis por transmitir informações entre os elementos da interface.

Além das propriedades, outro elemento fundamental dentro do React é o estado dos componentes. O estado permite armazenar informações que podem ser alteradas durante a execução da aplicação, tornando possível a exibição de conteúdos dinâmicos na interface. Quando ocorre alguma mudança nesse estado, o React realiza uma nova renderização do componente correspondente, garantindo que as informações apresentadas ao usuário permaneçam atualizadas de maneira eficiente.

[1] DOM - (Document Object Model) é uma interface de programação que representa documentos HTML, XML ou SVG como uma árvore de objetos. Ele permite que linguagens como JavaScript acessem e alterem dinamicamente a estrutura, o conteúdo e o estilo das páginas web, sem precisar recarregar o navegador

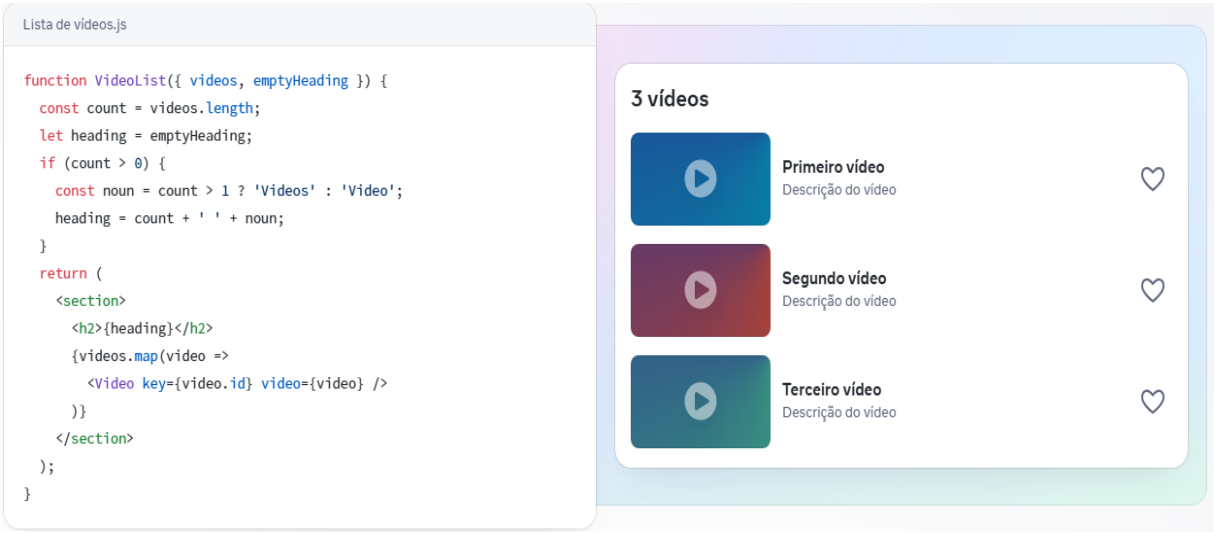
Durante a evolução da biblioteca, duas formas principais de criação de componentes passaram a ser utilizadas: componentes de classe e componentes funcionais. Os componentes de classe foram amplamente empregados nas versões iniciais do React, utilizando métodos específicos para gerenciamento de estado e renderização da interface. Entretanto, com o avanço da biblioteca, os componentes funcionais ganharam maior destaque devido à simplicidade e praticidade oferecidas durante o desenvolvimento.

Atualmente, os componentes funcionais representam a abordagem mais utilizada dentro do React.js. Segundo Lobo (2025), isso ocorreu principalmente após a introdução dos hooks, recursos que permitem utilizar funcionalidades como gerenciamento de estado e controle do ciclo de vida dos componentes sem necessidade de utilizar classes. Ferramentas como useState e useEffect facilitam significativamente o desenvolvimento das aplicações, tornando os códigos mais organizados, legíveis e reutilizáveis.

Outro fator que contribui para a ampla utilização do React está relacionado à integração com outras tecnologias do ecossistema JavaScript. A biblioteca pode ser utilizada juntamente com ferramentas voltadas para gerenciamento de rotas, consumo de APIs e estilização de interfaces, permitindo a construção de aplicações completas e escaláveis. Além disso, sua estrutura modular favorece o desenvolvimento colaborativo entre equipes, tornando o processo mais produtivo e organizado.

Diante disso, observa-se que o React.js tornou-se uma das principais bibliotecas utilizadas no desenvolvimento front-end contemporâneo. Sua arquitetura baseada em componentes, aliada ao desempenho proporcionado pela virtualização do DOM e à flexibilidade dos componentes funcionais, contribui para a construção de interfaces modernas, dinâmicas e eficientes, atendendo às necessidades das aplicações web atuais (Lobo, 2025).

Figura 3 - Exemplo da estrutura de componente React e sua renderização visual.



Fonte: < https://react.dev/ >. Acesso em: 21/05/2026

A motivação para a utilização do React neste projeto está relacionada à sua flexibilidade e à possibilidade de integração futura com o React Native. Essa característica permite o reaproveitamento de parte da lógica e da estrutura desenvolvida na aplicação web para o desenvolvimento de aplicações mobile, reduzindo tempo e custos durante futuras expansões do sistema. Dessa forma, a utilização do React contribui não apenas para o desenvolvimento de uma interface moderna e dinâmica no ambiente web, mas também para uma possível migração ou adaptação da plataforma para dispositivos móveis, proporcionando maior acessibilidade aos usuários por meio de smartphones.

2.1.4 CSS3

O desenvolvimento de interfaces web modernas não depende apenas da estrutura e da funcionalidade das aplicações, mas também da forma como os elementos visuais são apresentados ao usuário. Nesse contexto, o CSS3 desempenha papel fundamental na estilização e organização visual das páginas web, sendo responsável por definir aspectos relacionados a cores, fontes, espaçamentos, animações, posicionamentos e diferentes elementos de layout utilizados nas interfaces.

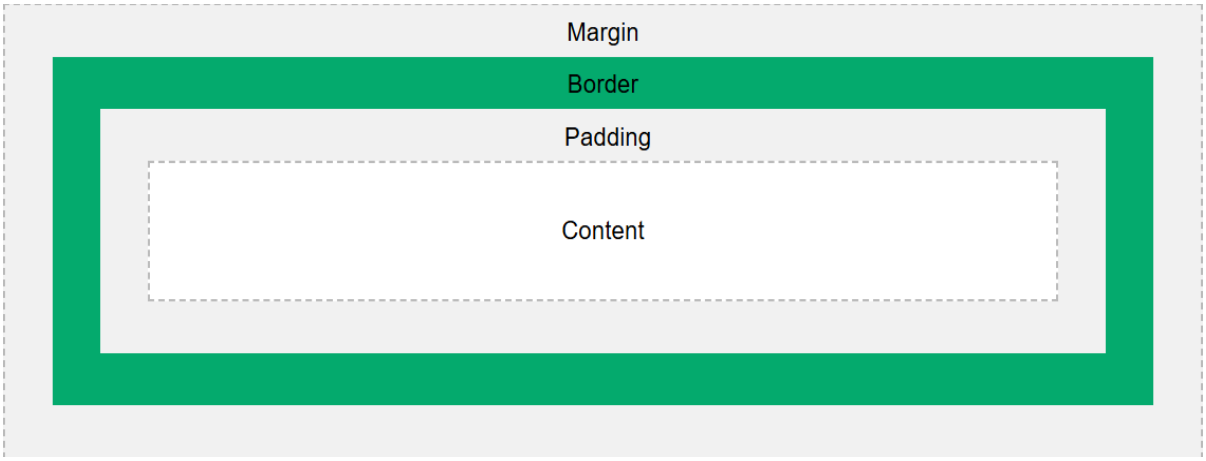
O CSS3 representa uma evolução das versões anteriores da linguagem Cascading Style Sheets, trazendo recursos mais avançados para a construção de páginas web visualmente mais modernas e responsivas. Sua utilização permite separar o conteúdo estrutural da apresentação visual da aplicação, favorecendo maior organização durante o desenvolvimento e facilitando a manutenção dos sistemas. Segundo Serra (2011), o CSS3 possibilitou avanços significativos na construção de interfaces interativas, oferecendo maior flexibilidade para personalização visual das aplicações web.

Entre os principais recursos introduzidos pelo CSS3, destacam-se funcionalidades relacionadas à responsividade e adaptação das interfaces para diferentes dispositivos. Com isso, páginas web podem ajustar automaticamente seus elementos visuais conforme o tamanho da tela utilizada pelo usuário, melhorando significativamente a experiência de navegação em computadores, tablets e smartphones. Além disso, o CSS3 oferece suporte a animações, transições e efeitos visuais que contribuem para interfaces mais dinâmicas e atrativas.

Outro aspecto importante está relacionado à organização dos layouts das aplicações. O CSS3 fornece mecanismos que permitem posicionar elementos de maneira mais eficiente e estruturada, facilitando a construção de interfaces modernas e intuitivas. Dessa forma, o uso adequado dessa tecnologia contribui não apenas para a estética das aplicações, mas também para a usabilidade e acessibilidade dos sistemas desenvolvidos.

Diante disso, percebe-se que o CSS3 se tornou uma tecnologia indispensável no desenvolvimento web contemporâneo, sendo amplamente utilizado na criação de interfaces organizadas, responsivas e visualmente mais sofisticadas. Sua integração com HTML e JavaScript possibilita a construção de aplicações completas, proporcionando melhor experiência visual e maior qualidade no desenvolvimento das interfaces web (Serra, 2011).

Figura 4 – Estrutura do Box Model no CSS3



Fonte: < https://www.w3schools.com/css/css_boxmodel.asp >. Acesso em: 21/05/2026

A utilização do CSS3 neste projeto justifica-se pela necessidade de desenvolver uma interface visual organizada, responsiva e agradável ao usuário. Por meio do CSS3, foi possível aplicar estilos, definir layouts e adaptar a aplicação para diferentes tamanhos de tela, proporcionando melhor experiência de navegação tanto em computadores quanto em dispositivos móveis. Além disso, a ferramenta contribui para a separação entre estrutura e apresentação da aplicação, facilitando a manutenção do código e permitindo maior flexibilidade no desenvolvimento das interfaces do sistema.

2.1.5 SQL

A linguagem SQL, sigla para Structured Query Language, representa uma das principais tecnologias utilizadas no gerenciamento de bancos de dados relacionais. Sua função está relacionada à comunicação entre aplicações e bancos de dados, permitindo realizar operações como armazenamento, consulta, alteração e remoção de informações de maneira estruturada e eficiente. Atualmente, a SQL é amplamente utilizada em diferentes sistemas gerenciadores de banco de dados, tornando-se uma linguagem essencial no desenvolvimento de aplicações computacionais.

A SQL foi desenvolvida com o objetivo de facilitar o gerenciamento das informações armazenadas em bancos de dados relacionais. Segundo Simbine (2021),

essa linguagem permite que desenvolvedores e administradores realizem consultas e manipulem dados utilizando comandos específicos e organizados de forma padronizada. Dessa maneira, diferentes sistemas de banco de dados conseguem utilizar estruturas semelhantes para gerenciamento das informações, favorecendo maior compatibilidade entre tecnologias distintas.

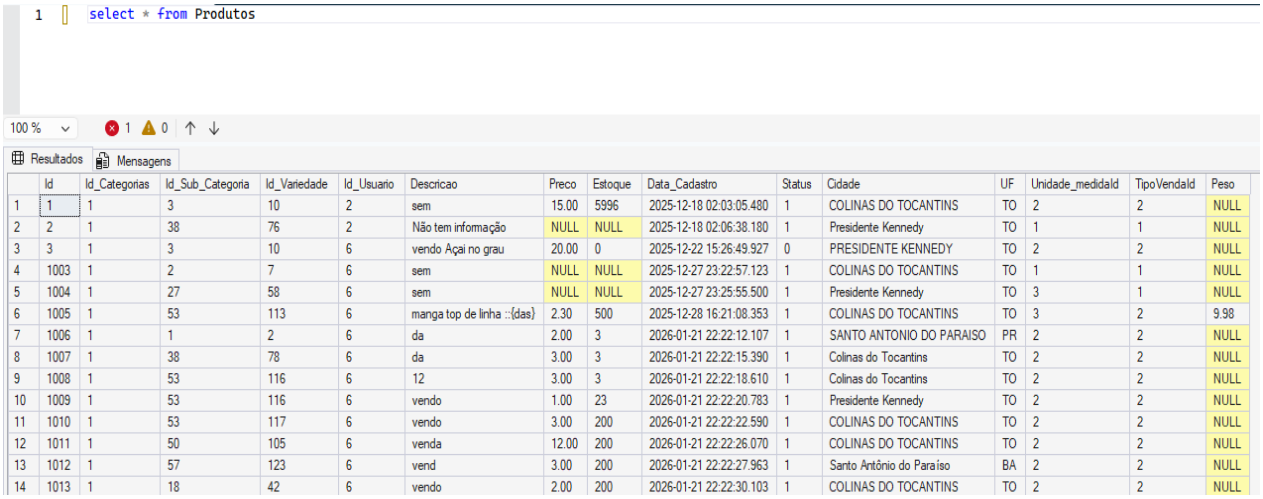
Diversos sistemas de gerenciamento de banco de dados utilizam a linguagem SQL como base para execução de operações relacionadas aos dados. Entre eles, destacam-se ferramentas amplamente utilizadas no mercado, como Oracle, PostgreSQL, MySQL e Microsoft SQL Server. Isso demonstra a importância da SQL no cenário da computação atual, principalmente em aplicações que necessitam de armazenamento estruturado e recuperação eficiente das informações.

Outro aspecto relevante está relacionado à simplicidade proporcionada pela linguagem durante a manipulação dos dados. Por meio de comandos específicos, é possível realizar consultas, inserir registros, atualizar informações e excluir dados armazenados no banco de maneira organizada e rápida. Essa comunicação eficiente entre aplicação e banco de dados contribui diretamente para o funcionamento adequado dos sistemas computacionais modernos.

A linguagem SQL também possui subdivisões responsáveis por diferentes tipos de operações dentro do banco de dados. Conforme destaca Simbine (2021), comandos como DDL, DML e DQL representam subconjuntos da SQL utilizados para criação de estruturas, manipulação de registros e realização de consultas. Essas categorias permitem organizar melhor as operações executadas no banco de dados, favorecendo maior controle e gerenciamento das informações armazenadas.

Diante disso, percebe-se que a linguagem SQL possui grande relevância no desenvolvimento de sistemas que utilizam bancos de dados relacionais. Sua capacidade de organizar, consultar e manipular informações de maneira eficiente faz com que a tecnologia seja indispensável em aplicações modernas, contribuindo para maior segurança, organização e desempenho no gerenciamento dos dados (Simbine, 2021).

Figura 5 - exemplo de uma seleção de informação no SQL.



```
1  select * from Produtos
```

100 % 1 0

Resultados Mensagens

	Id	Id_Categorias	Id_Sub_Categoria	Id_Variedade	Id_Usuario	Descricao	Preco	Estoque	Data_Cadastro	Status	Cidade	UF	Unidade_medidaId	TipoVendaId	Peso
1	1	1	3	10	2	sem	15.00	5996	2025-12-18 02:03:05.480	1	COLINAS DO TOCANTINS	TO	2	2	NULL
2	2	1	38	76	2	Não tem informação	NULL	NULL	2025-12-18 02:06:38.180	1	Presidente Kennedy	TO	1	1	NULL
3	3	1	3	10	6	vendo Açai no grau	20.00	0	2025-12-22 15:26:49.927	0	PRESIDENTE KENNEDY	TO	2	2	NULL
4	1003	1	2	7	6	sem	NULL	NULL	2025-12-27 23:22:57.123	1	COLINAS DO TOCANTINS	TO	1	1	NULL
5	1004	1	27	58	6	sem	NULL	NULL	2025-12-27 23:25:55.500	1	Presidente Kennedy	TO	3	1	NULL
6	1005	1	53	113	6	manga top de linha ::{das}	2.30	500	2025-12-28 16:21:08.353	1	COLINAS DO TOCANTINS	TO	3	2	9.98
7	1006	1	1	2	6	da	2.00	3	2026-01-21 22:22:12.107	1	SANTO ANTONIO DO PARAISO	PR	2	2	NULL
8	1007	1	38	78	6	da	3.00	3	2026-01-21 22:22:15.390	1	Colinas do Tocantins	TO	2	2	NULL
9	1008	1	53	116	6	12	3.00	3	2026-01-21 22:22:18.610	1	Colinas do Tocantins	TO	2	2	NULL
10	1009	1	53	116	6	vendo	1.00	23	2026-01-21 22:22:20.783	1	Presidente Kennedy	TO	2	2	NULL
11	1010	1	53	117	6	vendo	3.00	200	2026-01-21 22:22:22.590	1	COLINAS DO TOCANTINS	TO	2	2	NULL
12	1011	1	50	105	6	venda	12.00	200	2026-01-21 22:22:26.070	1	COLINAS DO TOCANTINS	TO	2	2	NULL
13	1012	1	57	123	6	vend	3.00	200	2026-01-21 22:22:27.963	1	Santo Antônio do Paraíso	BA	2	2	NULL
14	1013	1	18	42	6	vendo	2.00	200	2026-01-21 22:22:30.103	1	COLINAS DO TOCANTINS	TO	2	2	NULL

Fonte: o autor (2026).

A utilização da linguagem SQL no projeto justifica-se pela necessidade de realizar o gerenciamento eficiente e estruturado dos dados armazenados no banco de dados da aplicação. A SQL (Structured Query Language) é uma linguagem padrão amplamente utilizada para criação, manipulação e consulta de bancos de dados relacionais, permitindo maior organização, integridade e segurança das informações.

2.1.6 EXTREME PROGRAMMING (XP)

Segundo Valente (2020), o Extreme Programming (XP) é uma metodologia ágil voltada para projetos que exigem rapidez no desenvolvimento e facilidade de adaptação às mudanças, enfatizando qualidade de código, feedback rápido e evolução contínua do software.

2.1.7 EXPRESS.JS

O Express.js é um framework web minimalista e flexível para Node.js, lançado em 2010 e atualmente um dos mais utilizados no ecossistema JavaScript para construção de servidores e APIs REST. Sua principal função é abstrair e simplificar o módulo HTTP nativo do Node.js, fornecendo uma interface de programação mais expressiva para o tratamento de rotas, middlewares e requisições HTTP. Por meio do sistema de middlewares do Express, é possível encadear funções intermediárias que

processam a requisição antes de chegar ao controlador final, possibilitando a implementação de autenticação, validação de dados, tratamento de erros e compressão de respostas de forma modular e reutilizável, Sua leveza e a ausência de opiniões rígidas sobre a estrutura do projeto conferem ao desenvolvedor total liberdade arquitetural, tornando-o adequado tanto para APIs simples quanto para sistemas de maior complexidade.

2.1.8 JSON WEB TOKEN (JWT).

O JSON Web Token (JWT) é uma tecnologia utilizada para autenticação e troca segura de informações entre sistemas. O token é composto por três partes: cabeçalho, informações do usuário e assinatura digital, garantindo a integridade e a segurança dos dados transmitidos (JONES; BRADLEY; SAKIMURA, 2015). No desenvolvimento com Node.js, a biblioteca jsonwebtoken é amplamente utilizada para criar e validar tokens JWT, permitindo autenticar usuários e proteger rotas da aplicação. Como o JWT não depende do armazenamento de sessões no servidor, ele se torna uma solução eficiente para APIs REST, onde cada requisição carrega suas próprias informações de autenticação.

No desenvolvimento com Node.js, a biblioteca jsonwebtoken é amplamente utilizada para criar e validar tokens JWT, permitindo autenticar usuários e proteger rotas da aplicação. Como o JWT não depende do armazenamento de sessões no servidor, ele se torna uma solução eficiente para APIs REST, onde cada requisição carrega suas próprias informações de autenticação.

2.2 AMBIENTE E FERRAMENTAS DE DESENVOLVIMENTO

Além das linguagens e tecnologias utilizadas na construção das aplicações, o desenvolvimento de sistemas também depende diretamente de ferramentas capazes de auxiliar na escrita, organização, testes e gerenciamento do projeto. Nesse contexto, os ambientes de desenvolvimento desempenham papel importante ao fornecer

recursos que facilitam a produtividade, manutenção do código e integração entre diferentes tecnologias utilizadas durante o desenvolvimento da aplicação.

As ferramentas adotadas ao longo do projeto contribuem não apenas para maior agilidade no processo de desenvolvimento, mas também para melhor controle das funcionalidades implementadas, identificação de erros e realização de testes. Dessa forma, a utilização de ambientes e aplicações adequadas torna-se essencial para garantir maior eficiência, organização e qualidade durante a construção dos sistemas computacionais.

2.2.1 VISUAL STUDIO CODE (VS CODE)

O Visual Studio Code, conhecido popularmente como VS Code, é um editor de código-fonte amplamente utilizado no desenvolvimento de aplicações web, desktop e sistemas em diferentes linguagens de programação. Sua popularidade está relacionada principalmente à combinação entre leveza, desempenho e grande variedade de funcionalidades disponíveis para os desenvolvedores, tornando-se uma das principais ferramentas utilizadas atualmente no ambiente de programação.

O VS Code está disponível para diferentes sistemas operacionais, como Windows, Linux e macOS, permitindo que desenvolvedores trabalhem em diferentes plataformas sem grandes dificuldades. Além disso, o editor oferece suporte nativo para tecnologias amplamente utilizadas no desenvolvimento moderno, como JavaScript, TypeScript e Node.js. Segundo Oliveira (2026), o ambiente também possui um amplo ecossistema de extensões, possibilitando suporte para diversas outras linguagens e ferramentas utilizadas no desenvolvimento de software.

Outro aspecto importante do Visual Studio Code está relacionado à sua capacidade de personalização. O editor permite adicionar extensões responsáveis por ampliar suas funcionalidades, oferecendo recursos como destaque de sintaxe, formatação automática de código, refatoração, depuração de aplicações, integração com controle de versões e execução de projetos diretamente no ambiente de desenvolvimento. Essas funcionalidades contribuem para maior produtividade e organização durante a implementação das aplicações.

Além da flexibilidade, o VS Code também se destaca pela interface intuitiva e pela facilidade de integração com diferentes tecnologias utilizadas no desenvolvimento web moderno. Isso favorece a utilização da ferramenta tanto em

projetos acadêmicos quanto em ambientes profissionais, permitindo que programadores trabalhem de maneira mais eficiente na construção e manutenção dos sistemas.

Diante disso, percebe-se que o Visual Studio Code tornou-se uma ferramenta fundamental no cenário atual do desenvolvimento de software. Sua estrutura leve, associada à ampla capacidade de personalização e compatibilidade com diversas tecnologias, contribui diretamente para maior praticidade, produtividade e qualidade no processo de desenvolvimento das aplicações computacionais (Oliveira, 2026).

Figura 6 - tela inicial do Visual Studio Code.

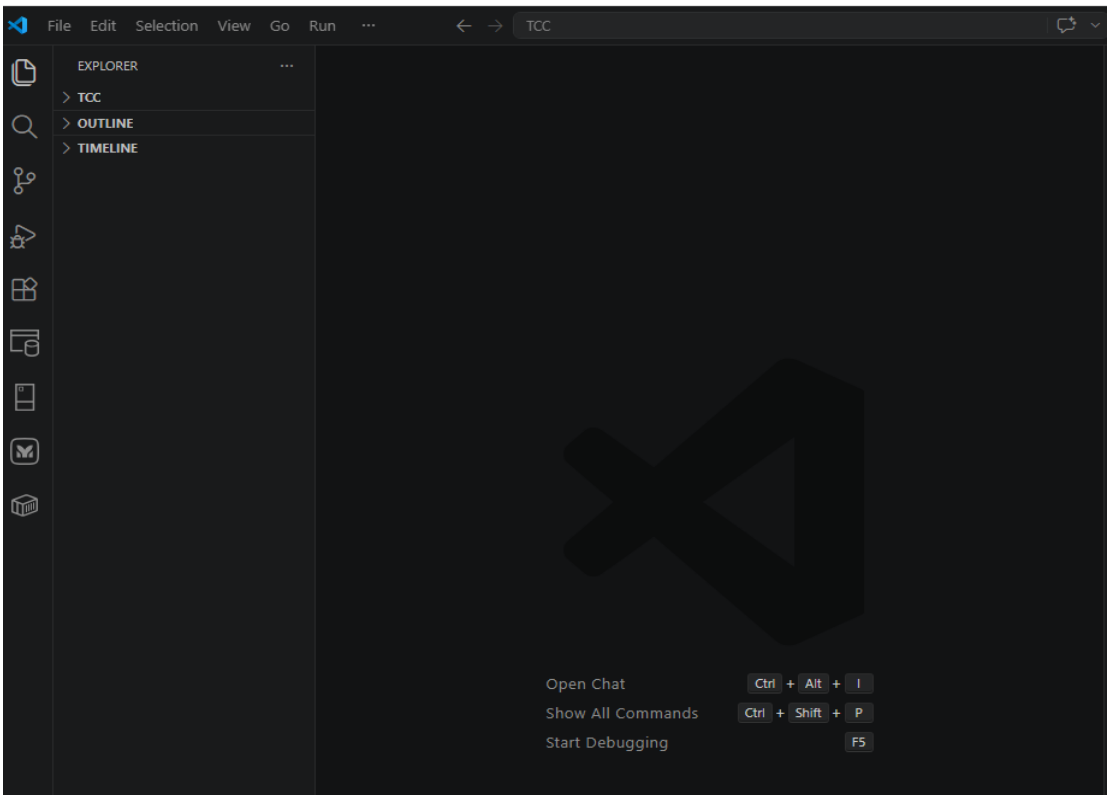


Figura: o autor (2026).

A utilização do Visual Studio Code neste projeto trás a eficiência e sua praticidade, desempenho e ampla compatibilidade com as tecnologias utilizadas no desenvolvimento da aplicação. O editor oferece suporte eficiente para linguagens como JavaScript, além de integração com ferramentas como Node.js, React e SQL Server. Outro fator importante está relacionado à disponibilidade de extensões que auxiliam na organização, depuração e produtividade durante o desenvolvimento do sistema. Dessa forma, o Visual Studio Code contribuiu para maior agilidade na escrita, manutenção e gerenciamento do código-fonte da aplicação.

2.2.2 SQL SERVER EXPRESS 2019

O SQL Server Express 2019 corresponde a uma das versões gratuitas disponibilizadas pela Microsoft para utilização do Microsoft SQL Server. Essa edição foi desenvolvida principalmente para estudos, testes e aplicações de pequeno e médio porte, oferecendo recursos importantes para armazenamento e gerenciamento de dados sem necessidade de custos com licenciamento. Dessa forma, tornou-se uma alternativa bastante utilizada em ambientes acadêmicos, projetos de desenvolvimento e aplicações que não exigem estruturas mais robustas de banco de dados.

Mesmo sendo uma versão gratuita, o SQL Server Express 2019 apresenta recursos considerados avançados para gerenciamento de informações. Segundo Gurgel e Oliveira (2022), essa edição fornece um ambiente confiável para armazenamento de dados em aplicações web e sistemas desktop, permitindo que os dados sejam organizados, consultados e manipulados de maneira eficiente. Isso contribui para maior segurança e estabilidade durante o funcionamento das aplicações.

Outro aspecto importante está relacionado à facilidade de utilização e integração com diferentes tecnologias de desenvolvimento. O SQL Server Express 2019 possui compatibilidade com aplicações desenvolvidas em diversas linguagens de programação e frameworks modernos, permitindo sua utilização em diferentes tipos de projetos computacionais. Além disso, a ferramenta oferece suporte à linguagem SQL, amplamente utilizada para criação de tabelas, consultas, inserção de dados e gerenciamento das informações armazenadas no banco de dados.

A versão Express também se destaca pela simplicidade de instalação e configuração, fator que favorece sua adoção em ambientes de aprendizagem e desenvolvimento acadêmico. Sua estrutura permite que estudantes e desenvolvedores realizem testes e implementações de sistemas sem necessidade de investimentos em versões comerciais mais complexas, mantendo recursos suficientes para construção de aplicações funcionais e organizadas.

Diante disso, percebe-se que o SQL Server Express 2019 representa uma solução eficiente para projetos que necessitam de um sistema de gerenciamento de banco de dados confiável, gratuito e de fácil utilização. Sua ampla utilização em ambientes acadêmicos e aplicações de menor porte demonstra sua relevância como

ferramenta de apoio ao desenvolvimento de sistemas modernos e gerenciamento estruturado de informações (Gurgel; Oliveira, 2022).

Figura 7 - Processos a serem seguidos para a instalação do SQL Server 2019.

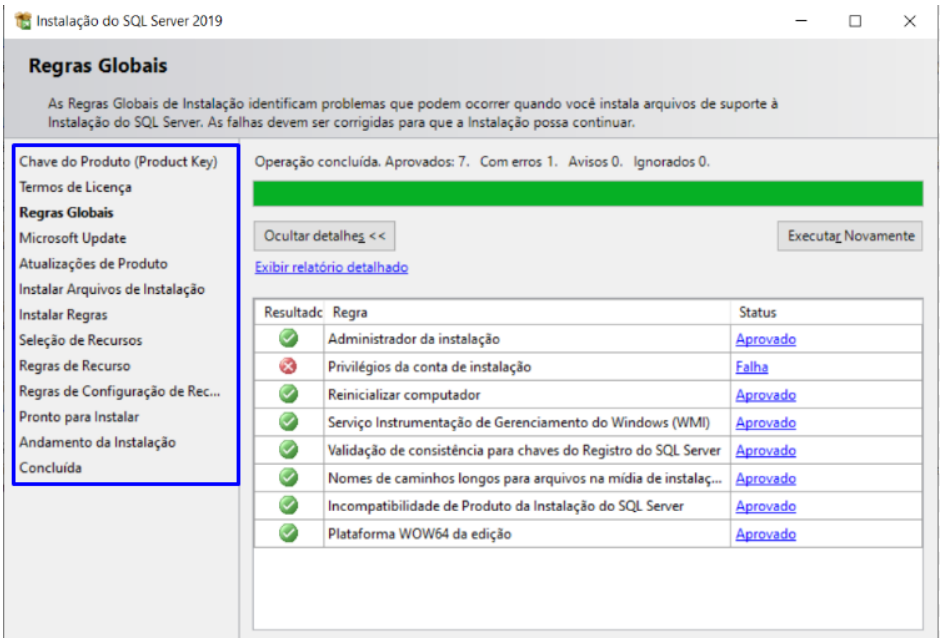


Figura: o autor (2026)

Para o armazenamento de dados foi escolhido o SQL Server Express 2019 que é um sistema gerenciador de banco de dados confiável, gratuito e eficiente para aplicações de pequeno e médio porte. A ferramenta oferece recursos importantes para armazenamento, organização e gerenciamento das informações da aplicação, permitindo maior segurança e integridade dos dados. Além disso, o SQL Server Express 2019 possui fácil integração com o Node.js e suporte à linguagem SQL, facilitando o desenvolvimento das operações relacionadas ao cadastro de usuários, vagas de emprego e candidaturas presentes no sistema.

2.2.3 SQL SERVER MANAGEMENT STUDIO (SSMS)

O SQL Server Management Studio, conhecido pela sigla SSMS, é uma ferramenta desenvolvida pela Microsoft voltada para administração, gerenciamento e manipulação de bancos de dados SQL Server. Sua utilização é bastante comum em ambientes acadêmicos e profissionais, principalmente por oferecer uma interface

gráfica que facilita a execução de consultas, criação de tabelas, gerenciamento de usuários e monitoramento das informações armazenadas no banco de dados.

O SSMS funciona como um ambiente integrado de gerenciamento, permitindo que administradores e desenvolvedores realizem diferentes operações relacionadas ao Microsoft SQL Server de maneira mais prática e organizada. Por meio da ferramenta, é possível executar comandos SQL, visualizar estruturas de banco de dados, gerenciar permissões de acesso e acompanhar o funcionamento do servidor em tempo real. Segundo Mendes (2023), o Microsoft SQL Server apresenta recursos voltados para segurança, gerenciamento e controle de dados, características que são administradas e configuradas diretamente através de ferramentas como o SQL Server Management Studio.

Outro aspecto importante envolve a facilidade proporcionada pela interface gráfica do SSMS. Diferentemente da utilização exclusiva de comandos em linha de texto, a ferramenta oferece menus, painéis e recursos visuais que auxiliam no gerenciamento das informações. Isso contribui para maior produtividade durante o desenvolvimento e administração dos sistemas, especialmente em atividades relacionadas à manutenção e organização do banco de dados.

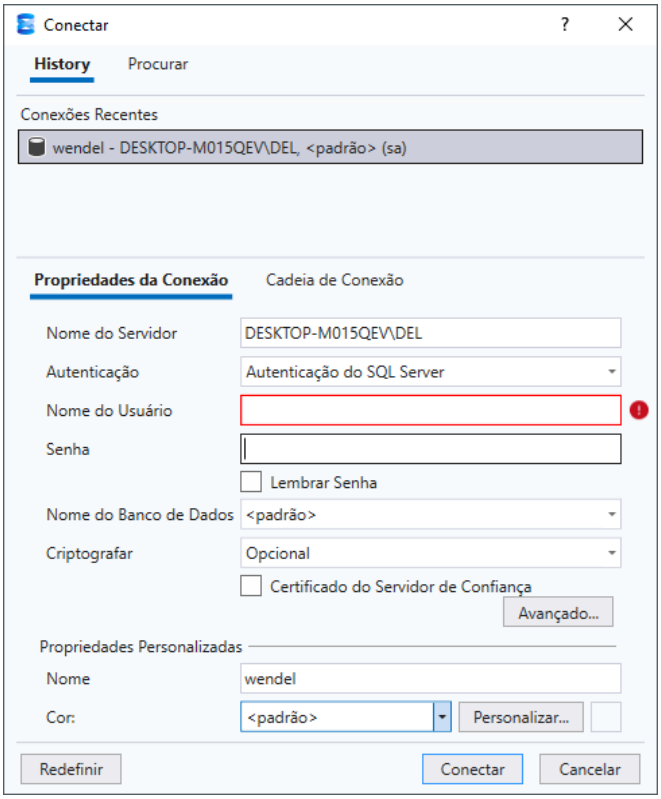
Além disso, o SQL Server Management Studio disponibiliza funcionalidades voltadas para criação e execução de consultas SQL, backup de bancos de dados, monitoramento de desempenho e gerenciamento de segurança. Essas ferramentas auxiliam no controle das informações armazenadas e favorecem maior confiabilidade durante o funcionamento das aplicações. O ambiente também permite identificar erros, analisar consultas e acompanhar processos executados no servidor de banco de dados.

Outro fator relevante está relacionado à integração entre o SSMS e diferentes versões do Microsoft SQL Server, incluindo edições gratuitas como o SQL Server Express. Isso possibilita que estudantes, desenvolvedores e empresas utilizem a ferramenta em projetos acadêmicos e aplicações reais sem necessidade de estruturas extremamente complexas, favorecendo maior acessibilidade durante o desenvolvimento dos sistemas.

Diante disso, percebe-se que o SQL Server Management Studio representa uma ferramenta essencial para administração e gerenciamento de bancos de dados SQL Server. Sua interface intuitiva, aliada aos recursos de controle, monitoramento e execução de consultas, contribui diretamente para maior organização, segurança e

eficiência no gerenciamento das informações armazenadas pelas aplicações computacionais (Mendes, 2023).

Figura 8- tela inicial do SSMS 22.



Fonte: o autor (2026).

A utilização do SQL Server Management Studio (SSMS) neste projeto justifica-se pela necessidade de uma ferramenta prática para administração e gerenciamento do banco de dados SQL Server. O SSMS possibilita a criação e manipulação de tabelas, execução de consultas SQL, gerenciamento de relacionamentos e monitoramento das informações armazenadas no sistema. Além disso, sua interface gráfica facilita o desenvolvimento e a manutenção do banco de dados, contribuindo para maior organização, produtividade e controle das operações realizadas durante o desenvolvimento da aplicação.

2.2.4 POSTMAN

O desenvolvimento de aplicações modernas frequentemente envolve a utilização de APIs para comunicação entre diferentes sistemas, serviços e bancos de dados. Nesse contexto, ferramentas voltadas para testes e validação dessas

integrações tornaram-se essenciais durante o processo de desenvolvimento. Entre as soluções mais utilizadas atualmente, destaca-se o Postman, uma plataforma amplamente empregada para criação, envio, organização e automação de requisições HTTP em aplicações web.

O Postman é utilizado principalmente para testes de APIs, permitindo que desenvolvedores realizem requisições aos serviços da aplicação de maneira prática e organizada. A ferramenta possibilita testar métodos como GET, POST, PUT e DELETE, além de permitir análise das respostas fornecidas pelo servidor, validação de dados e identificação de possíveis falhas durante a comunicação entre sistemas. Dessa forma, o Postman contribui diretamente para maior confiabilidade e eficiência no desenvolvimento das aplicações.

Uma das características que mais contribuem para a popularidade do Postman está relacionada à sua documentação e organização interna. Segundo Santos (2024), a plataforma apresenta uma estrutura intuitiva e de fácil navegação, permitindo que usuários encontrem funcionalidades e recursos de maneira mais simples. Isso favorece tanto desenvolvedores experientes quanto iniciantes que estão tendo os primeiros contatos com APIs e testes de integração.

Outro aspecto relevante envolve a abrangência dos recursos disponibilizados pela ferramenta. O Postman permite desde a realização de requisições simples até processos mais avançados de automação de testes. A plataforma oferece suporte para autenticação, gerenciamento de variáveis, organização de coleções de requisições e execução automatizada de cenários de teste, tornando-se uma solução bastante completa para validação de APIs em diferentes contextos de desenvolvimento.

Além disso, os exemplos práticos disponibilizados pela ferramenta representam um fator importante para o processo de aprendizagem. Conforme destaca Santos (2024), os exemplos permitem que desenvolvedores compreendam melhor a aplicação das funcionalidades em situações reais, facilitando a adaptação dos recursos conforme as necessidades específicas de cada projeto. Isso contribui para maior produtividade e reduz dificuldades relacionadas à implementação de testes em aplicações web.

Outro ponto importante está relacionado aos recursos complementares disponibilizados pela plataforma. O Postman oferece tutoriais, documentações detalhadas, vídeos explicativos e uma comunidade ativa de usuários que

compartilham soluções e experiências relacionadas à ferramenta. Esses recursos auxiliam significativamente no aprendizado e na resolução de problemas encontrados durante o desenvolvimento das aplicações.

A possibilidade de automatizar testes também representa uma das funcionalidades mais relevantes do Postman. A ferramenta permite criar scripts capazes de validar automaticamente respostas recebidas das APIs, verificando informações como códigos de status, tempo de resposta e integridade dos dados retornados pelo servidor. Isso contribui para maior eficiência no processo de testes, reduzindo falhas manuais e favorecendo maior estabilidade das aplicações desenvolvidas.

Além da automação, o Postman possibilita organizar diferentes requisições em coleções, permitindo melhor gerenciamento dos testes realizados no projeto. Essa funcionalidade facilita o trabalho em equipe, já que coleções podem ser compartilhadas entre os membros do desenvolvimento, promovendo maior padronização e organização dos processos relacionados às APIs utilizadas pela aplicação.

Apesar das diversas vantagens apresentadas pela ferramenta, existem algumas limitações apontadas em sua utilização. Segundo Santos (2024), determinados recursos mais avançados relacionados à automação de testes podem apresentar documentações consideradas superficiais, exigindo que usuários busquem conteúdos complementares em outras fontes. Isso pode representar uma dificuldade principalmente para desenvolvedores iniciantes que ainda possuem pouco contato com testes automatizados.

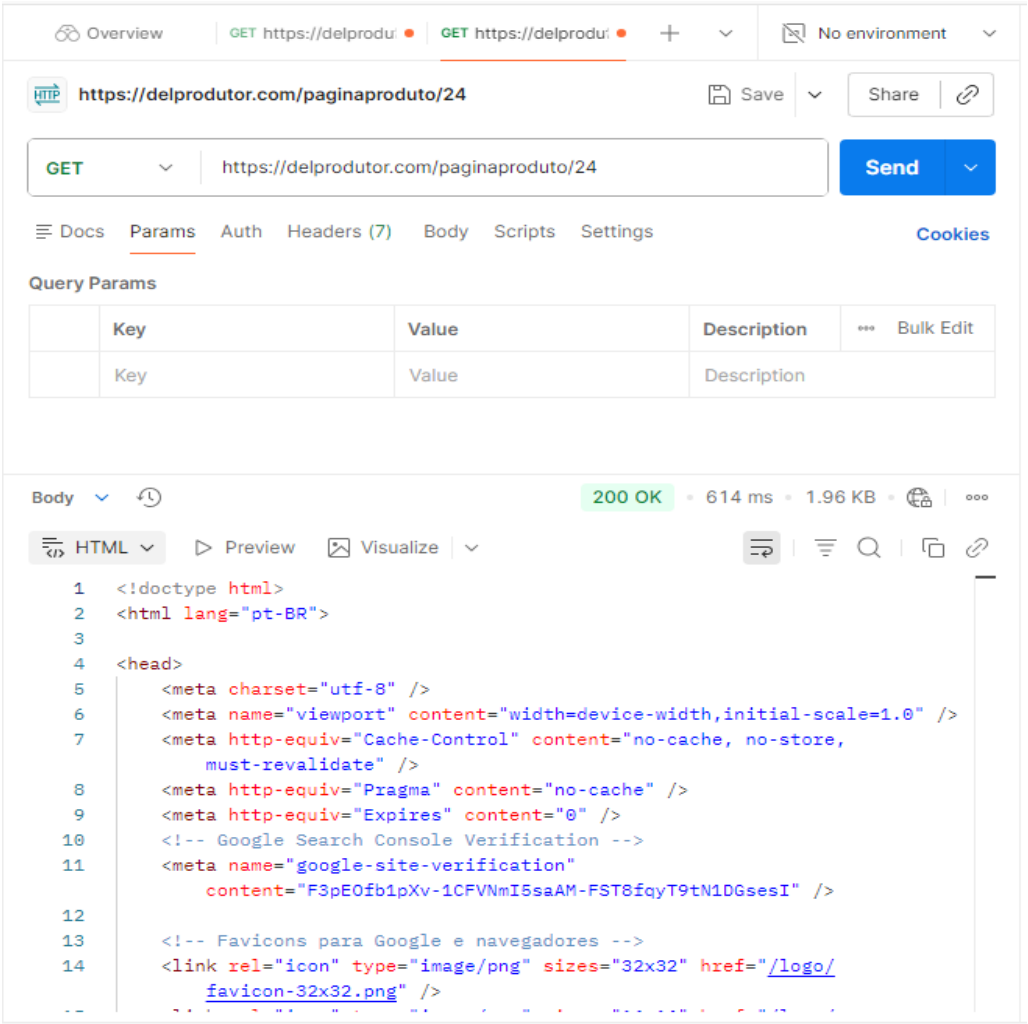
Outro aspecto mencionado envolve a inconsistência em algumas atualizações da documentação oficial. Em determinadas situações, algumas informações disponíveis podem não acompanhar imediatamente as versões mais recentes do software, causando pequenas dificuldades na adaptação de funcionalidades atualizadas. Além disso, em alguns conteúdos específicos, a ausência de exemplos mais próximos de cenários reais pode dificultar a aplicação prática de determinados recursos disponibilizados pela ferramenta.

Mesmo diante dessas limitações, o Postman permanece como uma das principais ferramentas utilizadas no desenvolvimento e teste de APIs. Sua interface intuitiva, associada à variedade de funcionalidades disponíveis, contribui significativamente para maior eficiência durante a validação das aplicações. A

possibilidade de automatizar testes, organizar requisições e monitorar respostas faz com que a ferramenta seja amplamente adotada tanto em ambientes acadêmicos quanto profissionais.

Diante disso, percebe-se que o Postman possui grande relevância no desenvolvimento moderno de aplicações web, principalmente em projetos que dependem de integração entre serviços e comunicação via APIs. Seus recursos voltados para testes, automação e organização contribuem diretamente para melhoria da qualidade, estabilidade e desempenho das aplicações desenvolvidas, tornando a ferramenta essencial em diferentes cenários de desenvolvimento de software (Santos, 2024).

Figura 9- exemplo de uma chamada de rota.



Fonte: o autor (2026).

A necessidade de realizar testes e validações das requisições da API desenvolvida no back-end. A ferramenta Postman permitiu testar endpoints, verificar

respostas do servidor, validar dados enviados e identificar possíveis erros durante a comunicação entre front-end e back-end. Além disso, contribuiu para a organização e monitoramento das rotas da aplicação, auxiliando no desenvolvimento, depuração e garantia do correto funcionamento das funcionalidades implementadas no sistema.

3 METODOLOGIA

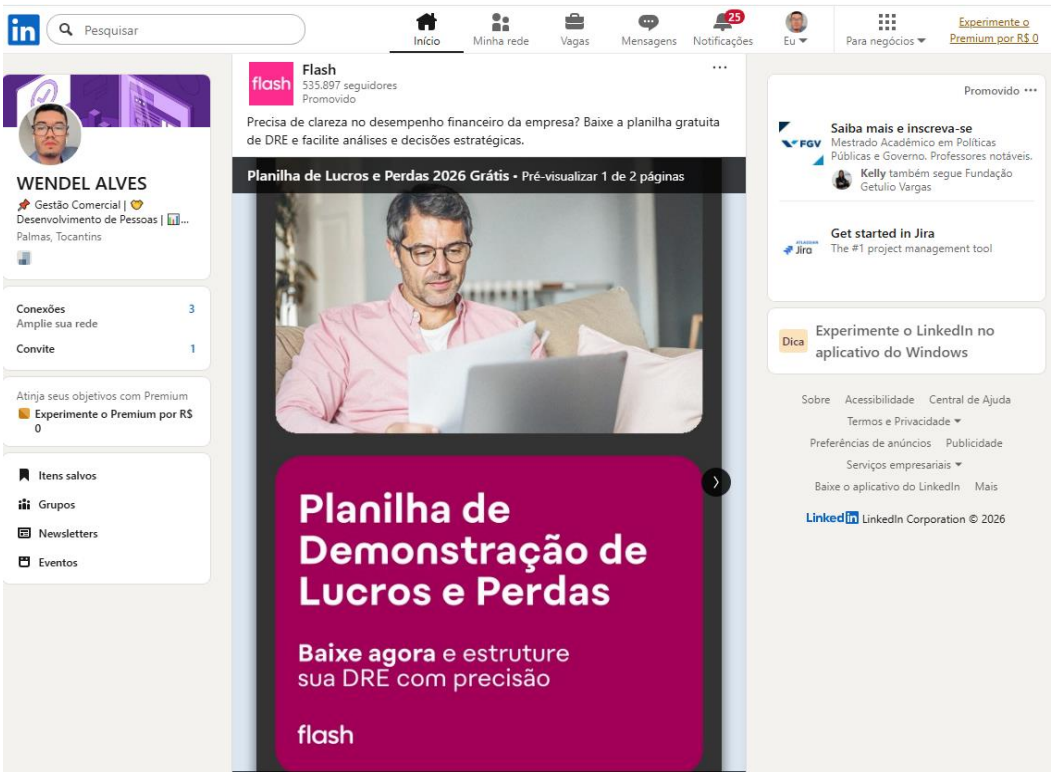
Neste capítulo serão apresentados os materiais, métodos e tecnologias utilizados no desenvolvimento da aplicação, bem como suas respectivas versões.

3.1 IDENTIFICAÇÃO E DELIMITAÇÃO DO PROBLEMA

O projeto teve início a partir da observação de plataformas digitais voltadas ao recrutamento e seleção de candidatos, desenvolvidas com o objetivo de facilitar a conexão entre empresas e profissionais em busca de oportunidades de emprego.

Durante a utilização dessas plataformas, foi possível identificar que algumas apresentam características semelhantes às redes sociais, permitindo publicações de conteúdos diversos na linha do tempo dos usuários, como ocorre no LinkedIn. Embora esses recursos ampliem a interação entre os usuários, em determinados casos podem desviar o foco principal da plataforma, que é a busca por vagas de emprego e processos de recrutamento.

Figura 10 - exemplo de plataforma de recrutamento



Fonte: o Autor (2026)

Além disso, plataformas como Indeed e Solides apresentam uma quantidade elevada de funcionalidades e informações, o que pode gerar dificuldades para usuários com menor familiaridade com a tecnologia. Essa complexidade pode comprometer a experiência de navegação, dificultando a busca por vagas e o acesso às funcionalidades essenciais do sistema.

Diante desse cenário, observa-se que muitos usuários acabam abandonando o uso dessas plataformas durante o processo de procura por emprego, retornando a métodos tradicionais, como entrega presencial de currículos e indicações pessoais.

Com base nessa problemática, este trabalho delimita-se ao desenvolvimento de um sistema web para cadastro e gerenciamento de vagas de emprego, priorizando simplicidade, organização das informações, usabilidade e acessibilidade, buscando oferecer uma experiência mais intuitiva tanto para candidatos quanto para empresas.

3.1.1 METODOLOGIA DE DESENVOLVIMENTO

O desenvolvimento deste projeto foi baseado nos princípios das metodologias ágeis, utilizando conceitos do Manifesto Ágil e da metodologia Extreme Programming (XP). A adoção dessa abordagem teve como objetivo proporcionar maior flexibilidade durante o desenvolvimento, permitindo adaptações contínuas conforme a evolução dos requisitos e das funcionalidades do sistema.

3.2 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos foi realizado com base na análise de sistemas similares existentes no mercado e na identificação das necessidades dos potenciais usuários da plataforma. Os requisitos foram divididos em:

Requisitos Funcionais (RF):

RF01 – O sistema deve permitir o cadastro de usuários com os perfis: Candidato, Empresa e Administrador.

RF02 – O sistema deve permitir o login e o logout de usuários autenticados.

RF03 – O sistema deve permitir que empresas cadastrem, editem e excluam vagas de emprego.

RF04 – O sistema deve permitir que candidatos visualizem as vagas disponíveis e se candidatem.

RF05 – O sistema deve permitir que candidatos gerenciem seu perfil e currículo.

RF06 – O sistema deve permitir que empresas visualizem os candidatos inscritos em suas vagas.

RF07 – O sistema deve disponibilizar um painel administrativo para gerenciamento de usuários e vagas.

RF08 – O sistema deve permitir a busca e filtragem de vagas por cargo, localização, área e tipo de contrato.

RF09 – O sistema deve exibir uma confirmação de candidatura após o envio da inscrição pelo candidato.

RF10 – O sistema deve exibir o status das candidaturas realizadas pelo candidato.

Requisitos Não Funcionais (RNF):

RNF01 – O sistema deve responder às requisições em no máximo 3 segundos em condições normais de uso.

RNF02 – O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela.

RNF03 – As senhas dos usuários devem ser armazenadas com hash bcrypt.

RNF04 – O sistema deve utilizar autenticação baseada em tokens JWT.

RNF05 – O sistema deve funcionar nos navegadores Chrome, Firefox, Edge e Safari em suas versões mais recentes.

RNF06 – O código deve ser organizado de forma modular e documentado para facilitar a manutenção.

O levantamento de requisitos foi realizado com base na análise de plataformas similares existentes no mercado, bem como na identificação das necessidades dos potenciais usuários do sistema. Os requisitos foram classificados em requisitos funcionais e não funcionais.

3.3 PLANEJAMENTO DO SISTEMA

O planejamento do sistema foi estruturado em etapas sequenciais e iterativas:

1. Definição do escopo e dos requisitos;
2. Modelagem do banco de dados (DER e esquema relacional);
3. Definição da arquitetura da aplicação;
4. Criação dos diagramas UML (caso de uso e classes);
5. Desenvolvimento do back-end utilizando JavaScript, Node.js e Express;
6. Desenvolvimento do front-end utilizando JavaScript e React;
7. Integração entre front-end, back-end e banco de dados;
8. Realização de testes funcionais;
9. Documentação e escrita do TCC.

3.4 FERRAMENTAS UTILIZADAS

As ferramentas e tecnologias utilizadas no desenvolvimento deste projeto estão listadas na Tabela 1.

Tabela 1 – Ferramentas e tecnologias utilizadas

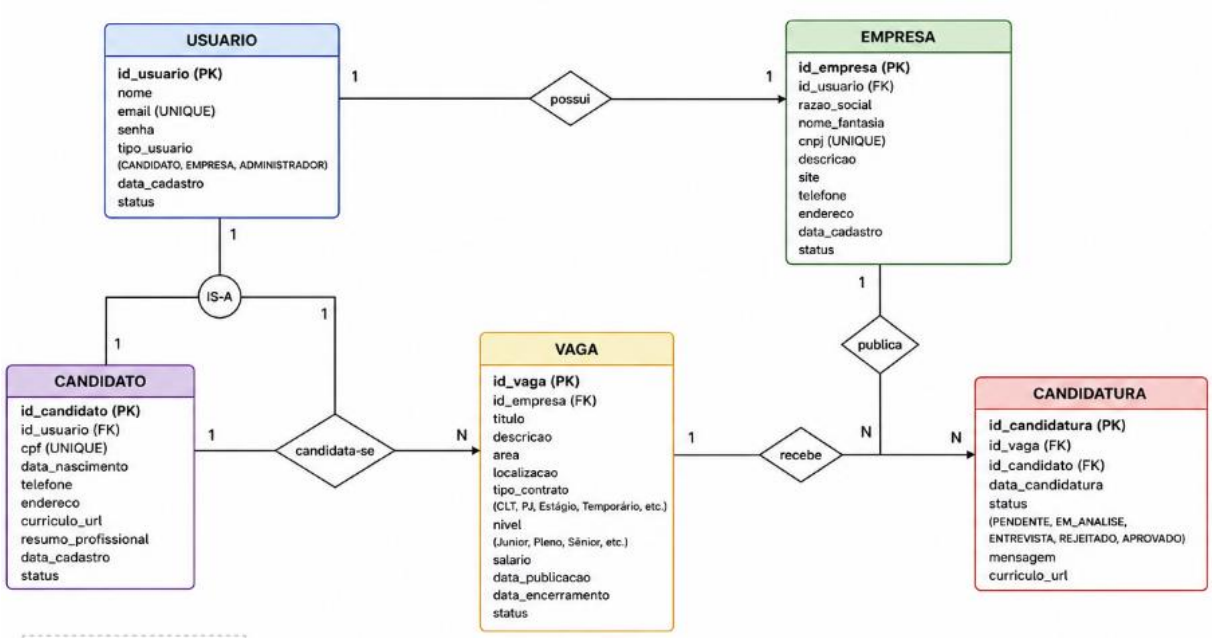
Ferramenta/Tecnologia	Versão	Finalidade
javaScript	Atual	Linguagem API back-end
Node.js	20.x LTS	Ambiente de execução back-end
Express.js	4.x	Framework para API REST
React	18.x	Biblioteca front-end
CSS	Atual	Linguagem Front-end
React Router DOM	6.x	Roteamento no front-end
Axios	1.x	Requisições HTTP no front-end

SQL Server Express	2022	Banco de dados relacional
mssql	10.x	Driver Node.js para SQL Server
bcryptjs	2.x	Hash de senhas
jsonwebtoken	9.x	Geração e validação de JWT
VS Code	Atual	Editor de código
Postman	Atual	Teste de endpoints da API
Git / GitHub	Atual	Controle de versão

Fonte: o autor (2026).

3.5 MODELAGEM DO BANCO DE DADOS

Figura 11 - Diagrama de entidade (DER).



Fonte: Gerada por IA (ChatGPT, 2026).

A modelagem do banco de dados iniciou-se com a identificação das principais entidades do sistema, sendo elas: Usuário, Candidato, Empresa, Vaga e Candidatura. Com base nessas entidades e em seus respectivos atributos, foi elaborado o Diagrama Entidade-Relacionamento (DER), responsável por representar a estrutura lógica do banco de dados da aplicação.

Posteriormente, realizou-se o processo de normalização do banco de dados até a Terceira Forma Normal (3FN), com o objetivo de reduzir redundâncias e garantir maior integridade e organização das informações armazenadas. Após essa etapa, o modelo foi implementado no SQL Server Express por meio de scripts SQL utilizados na criação das tabelas, relacionamentos e restrições de integridade.

3.6 DESENVOLVIMENTO DA API

O desenvolvimento da API REST seguiu os princípios da arquitetura RESTful, organizando os endpoints de acordo com os recursos do sistema, como usuários, vagas e candidaturas. Cada endpoint foi implementado em controllers específicos, enquanto a lógica de acesso ao banco de dados foi separada em funções de serviço (services), promovendo maior organização do código, separação de responsabilidades e facilidade na manutenção da aplicação.

A validação dos dados de entrada foi realizada por meio de verificações nas rotas antes do processamento das requisições pelos controllers, garantindo que apenas dados válidos fossem encaminhados à camada de serviço. O tratamento de erros foi implementado utilizando middlewares centralizados no Express, permitindo maior controle das exceções e padronização das respostas da API.

3.7 DESENVOLVIMENTO FRONT-END

O front-end da aplicação foi desenvolvido utilizando a biblioteca React. A estrutura do sistema foi organizada em componentes reutilizáveis, separados por funcionalidades em diretórios específicos, visando maior organização, manutenção e reaproveitamento de código.

O roteamento entre as páginas foi implementado por meio do React Router DOM v6, permitindo a navegação dinâmica entre as interfaces da aplicação. As requisições à API foram realizadas utilizando a biblioteca Axios, configurada com interceptors para inclusão automática do token JWT nos cabeçalhos das requisições autenticadas.

O gerenciamento de estado da aplicação foi realizado com os hooks nativos do React, como useState, useContext e useEffect, sem a utilização de bibliotecas

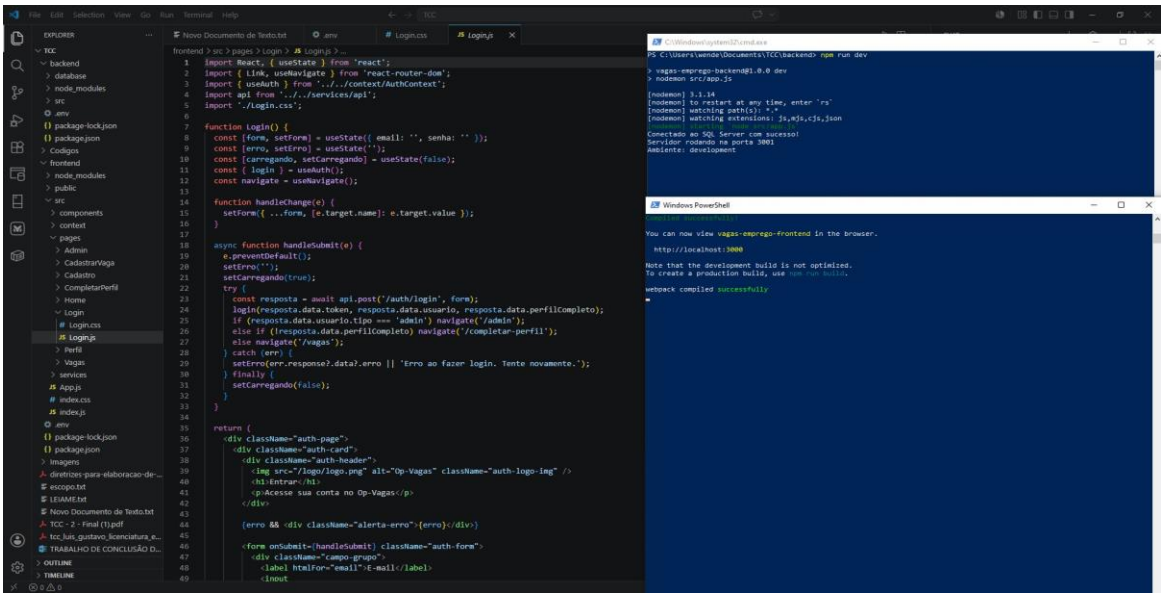
externas para gerenciamento global de estado. Além disso, a Context API foi utilizada para disponibilizar as informações do usuário autenticado entre os diferentes componentes da aplicação.

3.8 TESTES DO SISTEMA

Os testes realizados neste trabalho foram de natureza funcional e manual, com o objetivo de verificar se as funcionalidades desenvolvidas atendem aos requisitos especificados do sistema. Para os testes da API, utilizou-se a ferramenta Postman, realizando testes nos endpoints em diferentes cenários, como requisições válidas, envio de dados inválidos, tentativas de acesso não autorizado e simulações de erros no servidor.

No front-end, os testes foram executados manualmente nos navegadores Google Chrome e Mozilla Firefox, analisando o comportamento das interfaces, a responsividade em diferentes resoluções de tela e a correta integração entre o front-end e o back-end da aplicação.

Figura 12 testes do sistema



Fonte: O autor (2026).

4 DESENVOLVIMENTO DO SISTEMA

Neste capítulo serão apresentados os processos relacionados ao desenvolvimento do sistema, contemplando desde o planejamento da aplicação até sua implementação final. Também serão descritos a arquitetura do sistema, os diagramas de modelagem, a estrutura do banco de dados, a organização das pastas e códigos do back-end e front-end, bem como as principais funcionalidades desenvolvidas na plataforma.

4.1 ARQUITETURA DA APLICAÇÃO

A aplicação foi desenvolvida seguindo o modelo de arquitetura em três camadas (*three-tier architecture*), dividido em:

4.1.1 CAMADA DE APRESENTAÇÃO (FRONT-END)

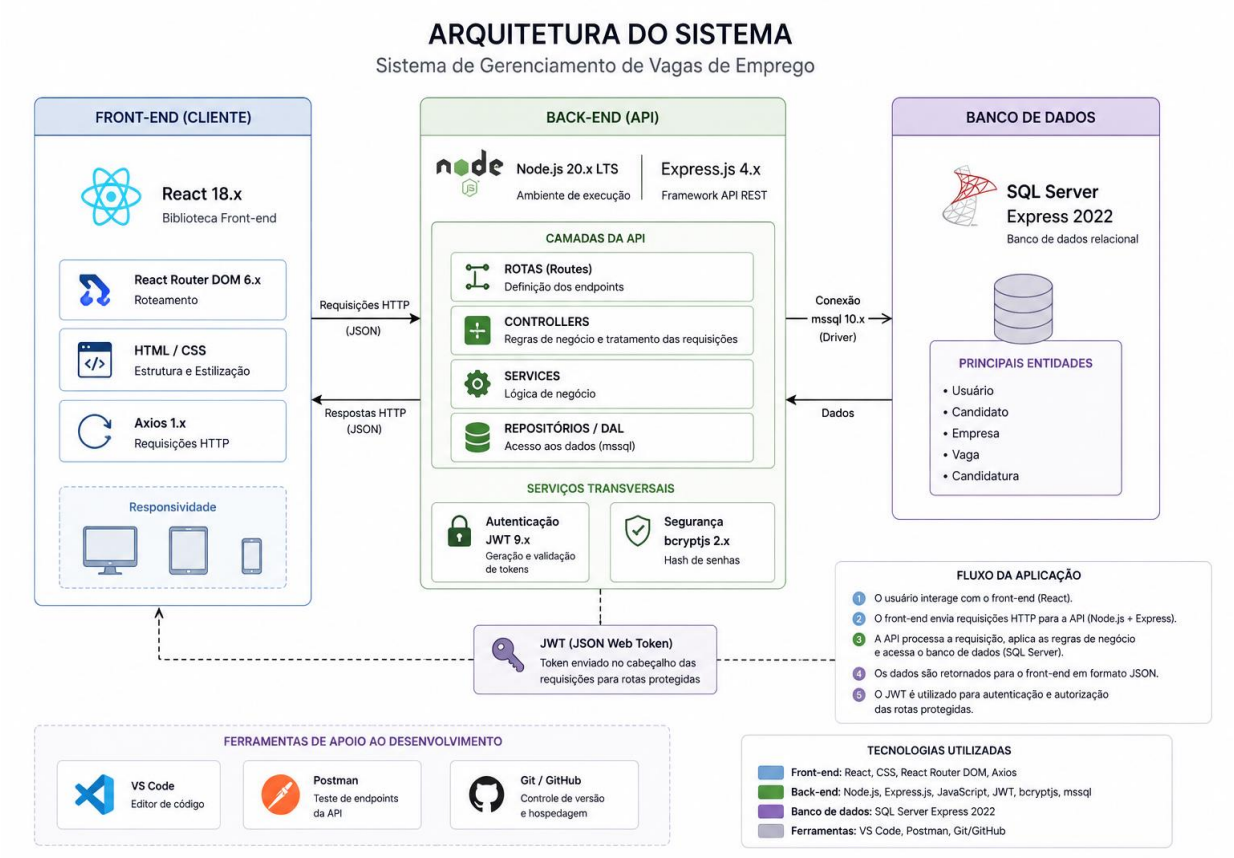
Responsável pela interface gráfica e pela interação com o usuário. Foi desenvolvida utilizando React, executado diretamente no navegador, permitindo navegação dinâmica e comunicação com a API por meio de requisições HTTP.

4.1.2. CAMADA DE LÓGICA DE NEGÓCIO (BACK-END)

Responsável pelo processamento das requisições, aplicação das regras de negócio e comunicação com o banco de dados. O back-end foi desenvolvido utilizando JavaScript e Node.js, disponibilizando uma API REST para integração com o front-end.

4.1.3. CAMADA DE DADOS

Figura 11 – Arquitetura geral da aplicação



Fonte: Gerada por IA (ChatGPT, 2026).

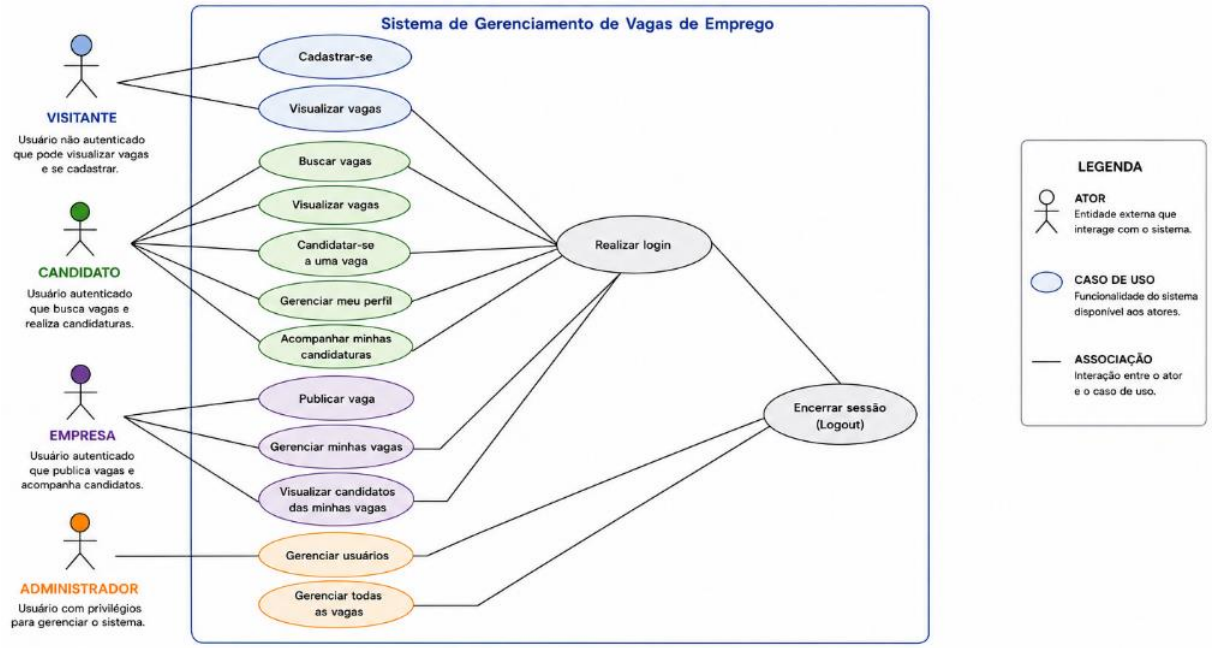
Responsável pelo armazenamento persistente das informações da aplicação. Para isso, foi utilizado o SQL Server Express 2019, contendo as entidades relacionadas ao gerenciamento de usuários, empresas, vagas e candidaturas.

A comunicação entre o front-end e o back-end ocorre por meio de requisições HTTP utilizando dados no formato JSON, seguindo os princípios da arquitetura REST. Além disso, os endpoints protegidos da API utilizam autenticação baseada em JWT (*JSON Web Token*), garantindo maior segurança no acesso aos recursos do sistema.

4.2 DIAGRAMA DE CASO DE USO

Os principais casos de uso identificados foram:

Figura 14 – Diagrama Caso de Uso



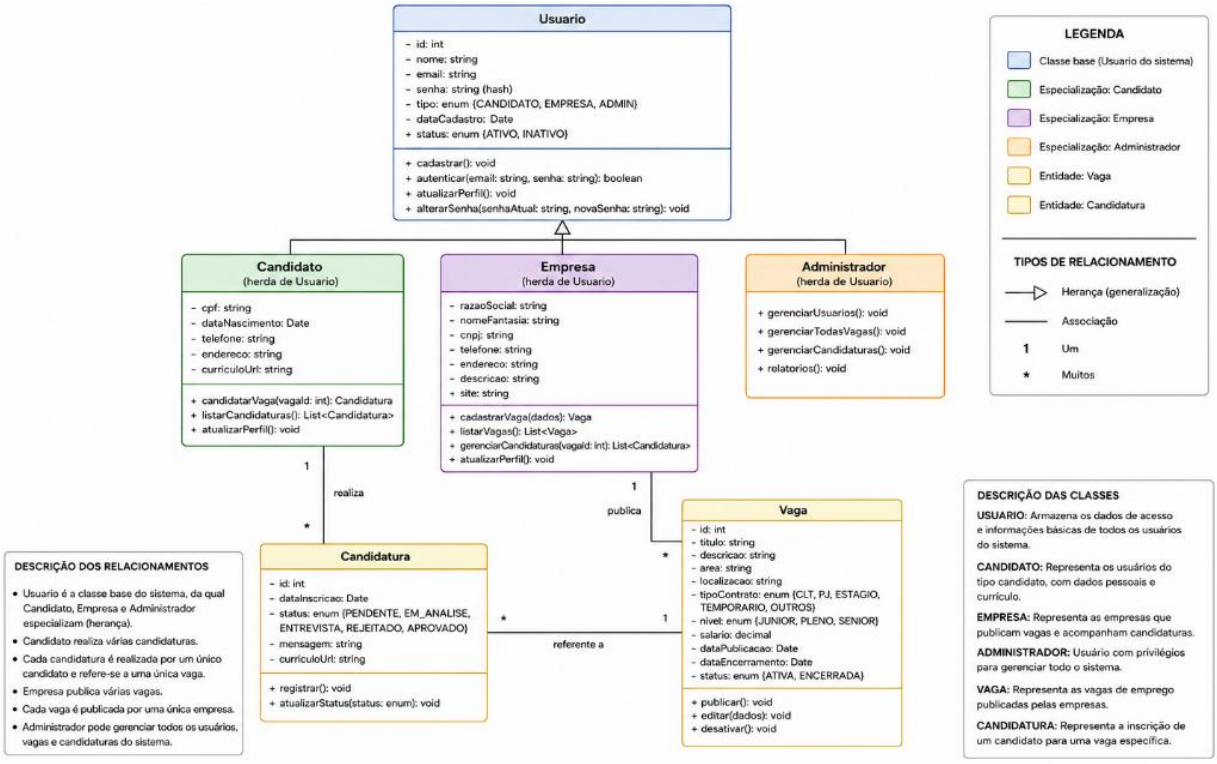
Fonte: Gerada por IA (ChatGPT, 2026).

O Diagrama de Caso de Uso representa as interações entre os atores do sistema e suas respectivas funcionalidades. Os atores identificados no sistema são:

- **Visitante:** usuário não autenticado que pode visualizar vagas disponíveis e realizar cadastro na plataforma;
- **Candidato:** usuário autenticado com perfil de candidato, responsável por realizar candidaturas em vagas e gerenciar seu perfil;
- **Empresa:** usuário autenticado com perfil de empresa, responsável pelo cadastro e gerenciamento de vagas de emprego;
- **Administrador:** usuário com acesso total ao sistema, responsável pelo gerenciamento de usuários e vagas.

4.3 Diagrama de Classes

Figura 15 – Diagrama de Classes



Fonte: Gerada por IA (ChatGPT, 2026)

O Diagrama de Classes representa a estrutura estática do sistema, descrevendo as classes, seus atributos, métodos e relacionamentos. As principais classes do sistema.

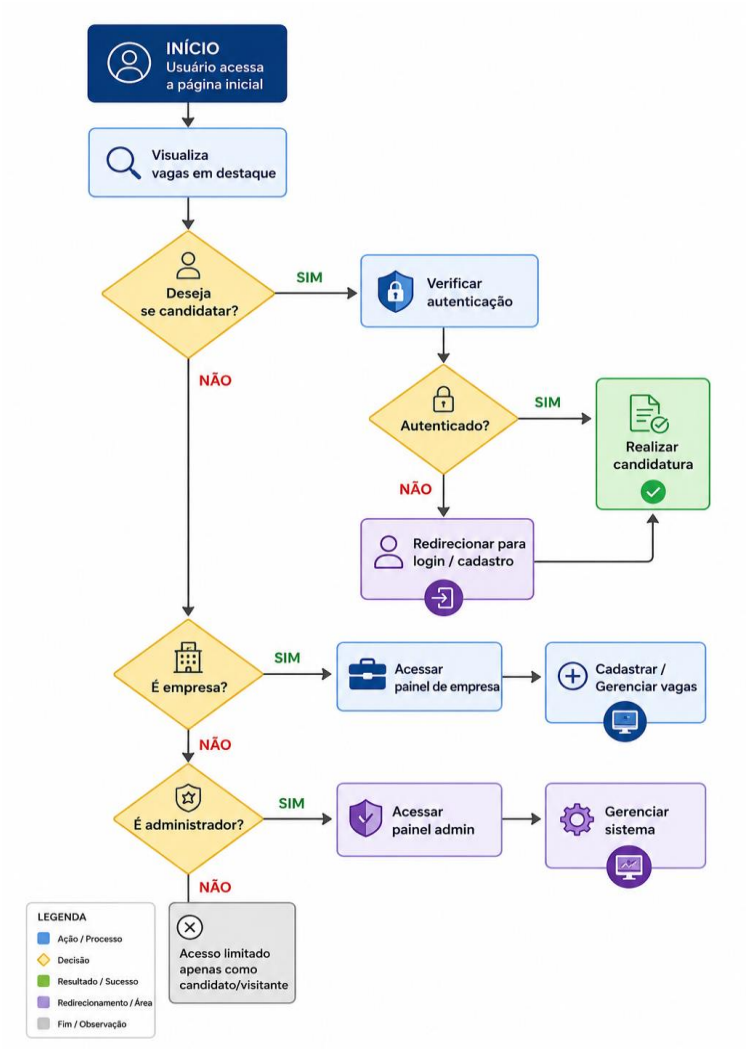
Objetivo de armazenar e gerenciar as informações do sistema de forma organizada, segura e eficiente. A modelagem foi baseada no Diagrama Entidade-Relacionamento (DER), permitindo o relacionamento entre usuários, empresas, vagas e candidaturas.

- **USUÁRIO**: responsável pelo armazenamento das informações de autenticação e identificação dos usuários do sistema;
- **CANDIDATO**: armazena os dados específicos dos candidatos;
- **EMPRESA**: contém os dados das empresas cadastradas na plataforma;
- **VAGA**: registra as vagas de emprego publicadas pelas empresas;
- **CANDIDATURA**: realiza o vínculo entre candidatos e vagas.

Cada entidade possui campos específicos para armazenar dados como nome, senha, endereço e demais informações necessárias. Dessa forma, o banco de dados torna possível o acesso organizado às informações por meio das requisições realizadas pelo backend, garantindo integração, segurança e eficiência no gerenciamento dos dados.

4.4 FLUXOGRAMA DO SISTEMA

Figura 16 – Fluxograma do sistema



Fonte: Gerada por IA (ChatGPT, 2026)

O fluxograma do sistema representa o fluxo principal de navegação e as decisões do sistema:

1. Usuário acessa a página inicial → visualiza vagas em destaque

2. Deseja se candidatar? → SIM: verificar autenticação

- Autenticado? → SIM: realizar candidatura

- Autenticado? → NÃO: redirecionar para login/cadastro

3. É empresa? → SIM: acessar painel de empresa → cadastrar/gerenciar vagas

4. É administrador? → SIM: acessar painel admin → gerenciar sistema

4.5 INTERFACE DO SISTEMA

Figura 17 - interface do sistema

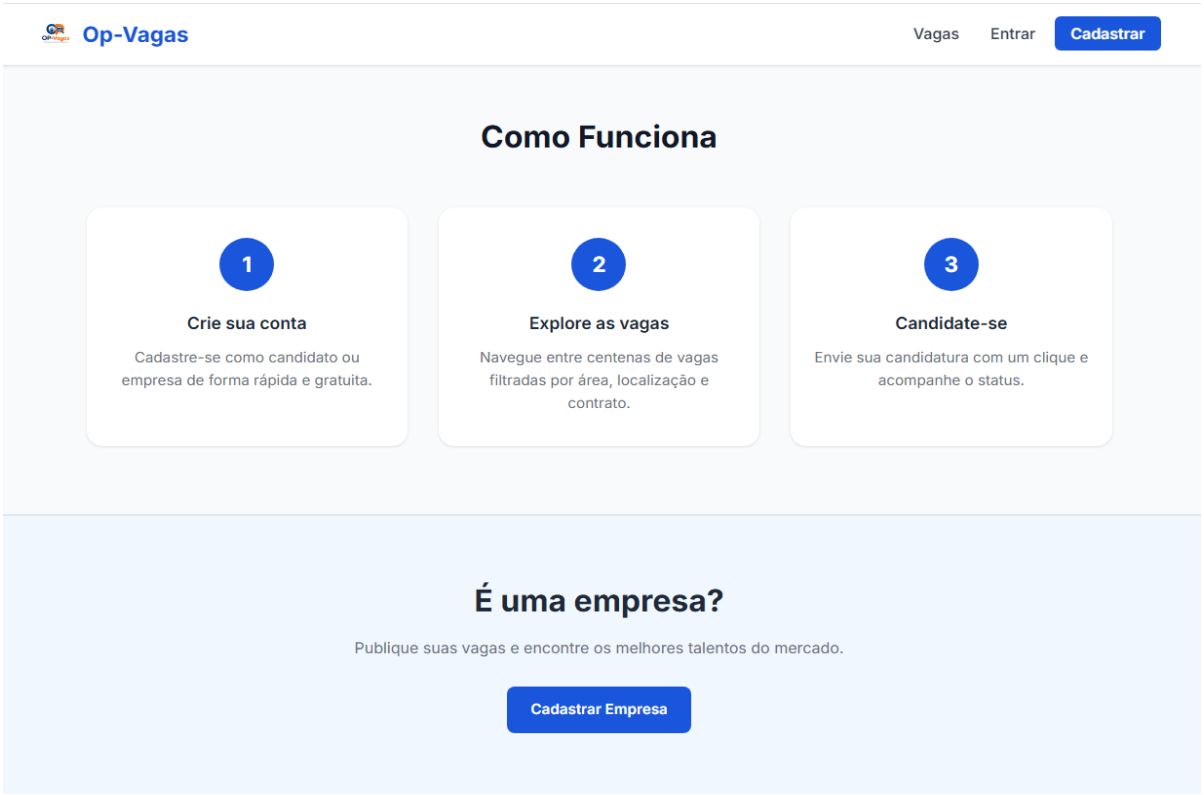
Fonte: o autor(2026).

Na parte superior do sistema está localizada a barra de navegação, conhecida tecnicamente como "Navbar". Essa barra permite que o usuário navegue entre as telas do sistema de forma simples e organizada, sem excesso de informações ou anúncios.

O sistema foi desenvolvido visando a usabilidade e a facilidade de navegação, principalmente para usuários que estão utilizando a plataforma pela primeira vez. Dessa forma, buscou-se manter uma interface objetiva e intuitiva, facilitando a tomada de decisões e tornando a experiência do usuário mais prática durante a utilização do sistema.

4.6 TELA INICIAL

Figura 18 tela Inicial do sistema



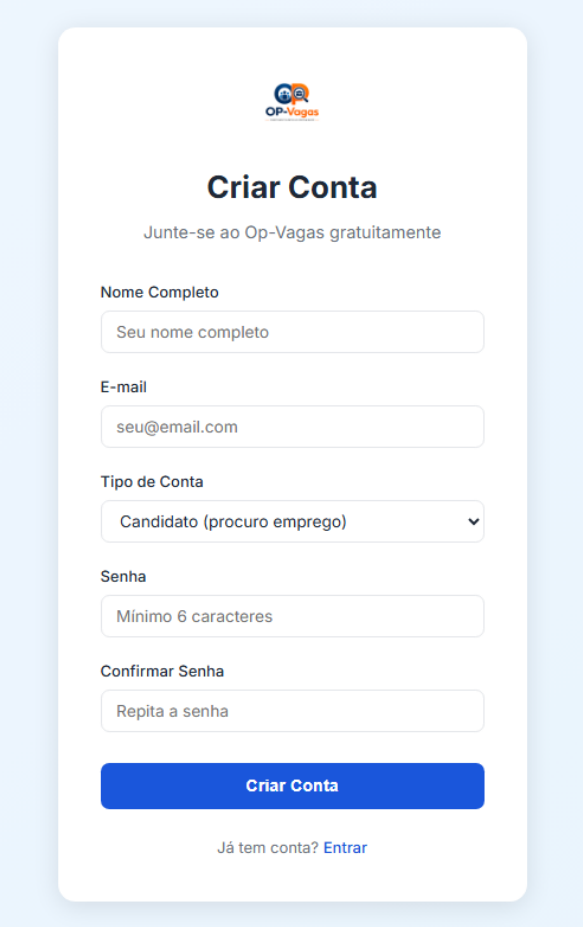
Fonte: o autor (2026).

Na tela inicial, manteve-se um padrão de informações básicas e objetivas. Nela estão disponíveis opções como cadastro simplificado e gratuito, exploração de vagas e candidatura para vagas de emprego disponíveis no sistema.

O sistema não possui limitação de candidaturas, permitindo que o usuário possa se candidatar a diferentes vagas de forma ilimitada. Além disso, para acessar essa tela, não é necessário possuir cadastro ou conta ativa no sistema, pois as informações também podem ser visualizadas por visitantes.

4.6.1 CADASTRO DE USUÁRIO

Figura 19 tela de cadastro

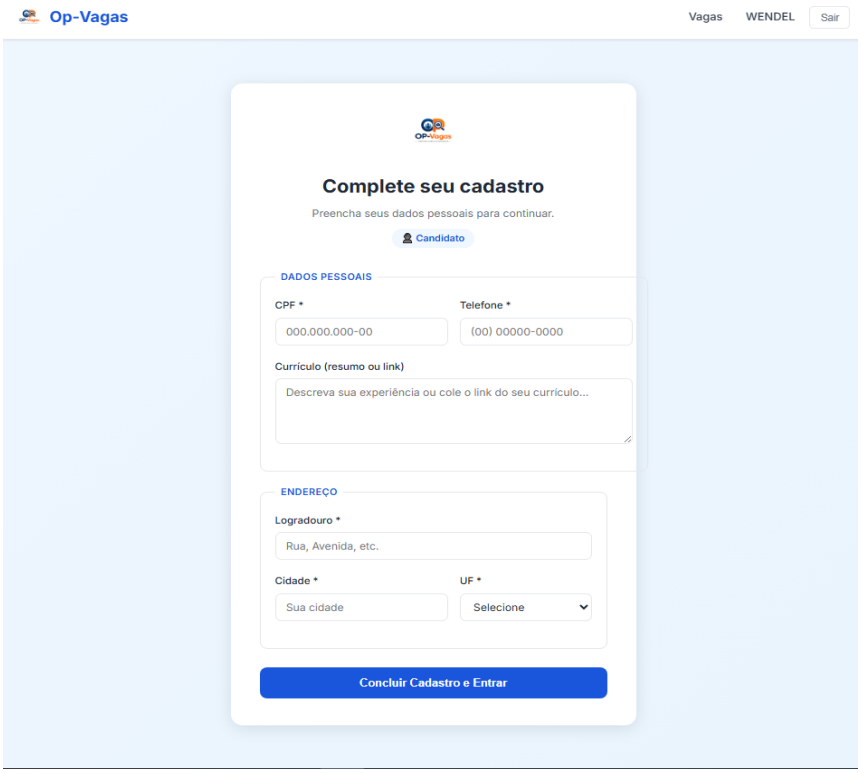


Fonte: o autor (2026).

A tela de cadastro para é exigido umas informações prévia como (Nome Completo, E-mail, Tipo de Conta Candidato ou Empresa, Senha, Confirmar Senha), nessa etapa o usuário vai informar seu dados aqui que vai definir seu perfil de usuário se ele vai procurar a vaga ou se ele vai anunciar para a procura de emprego ele precisará escolher a opção “CANDIDATO”, essa opção é vinculado ao perfil de usuários que procuram empregos após a conclusão dessas informações o usuário está apto para fazer login dentro da plataforma, porém não é só essas informações.

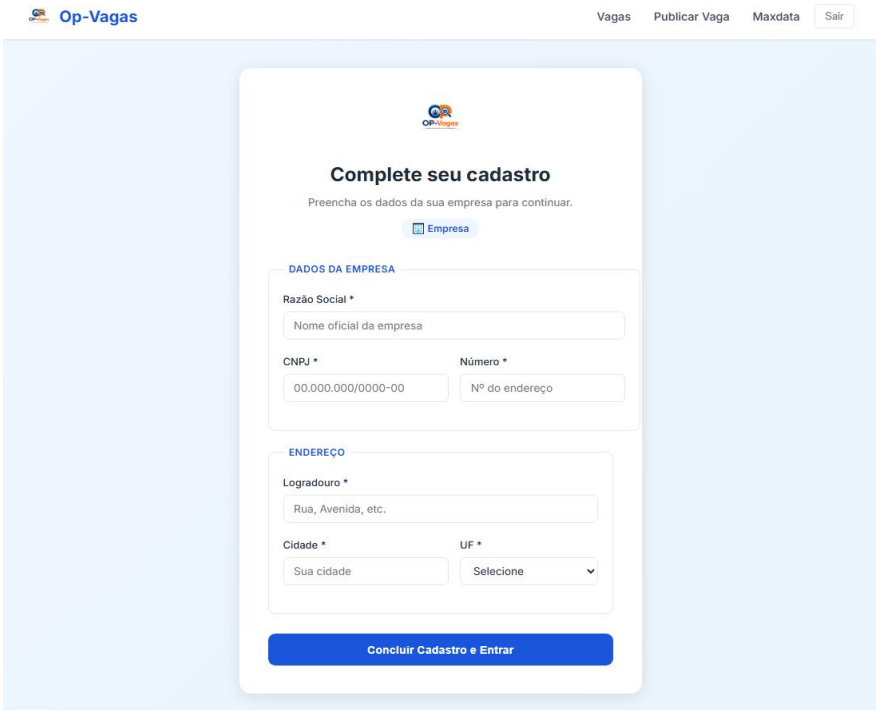
Ao inserir o e-mail e senha a plataforma ela irá fazer uma varredura na tabela “dados_usuario”, dentro do banco de dados sql se já existir dados do usuário, ele redirecionado a tela inicial do sistema, caso seja o primeiro acesso aparecerá um formulário para o usuário preencher com suas informações pessoais.

Figura 20 informação cadastro candidato



Fonte: o autor (2026).

Figura 21 informação cadastro empresa

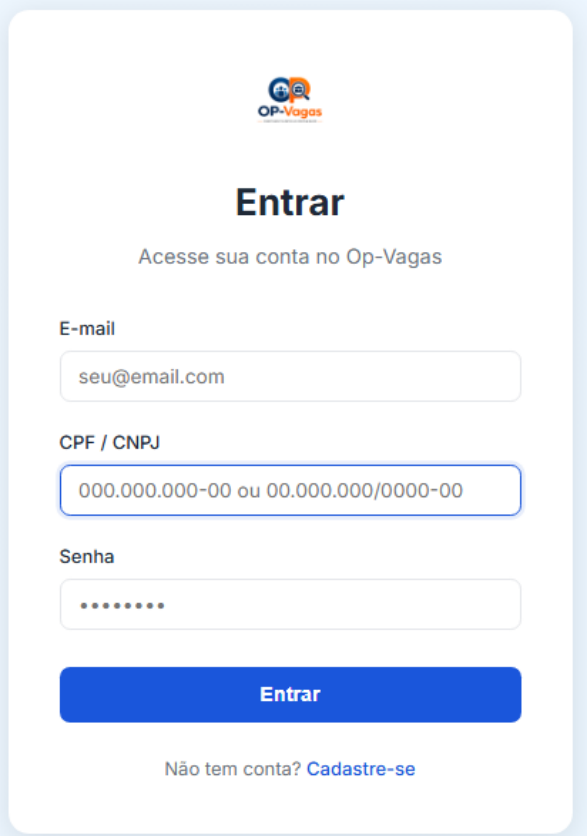


Fonte: o autor (2026).

Nessa etapa os dois perfis precisarão finalizar com suas informações pessoais, esse modelo de validação é conclusão é usado em vários aplicativos hoje em dia até mesmo para rede de streaming dentre outros, dessa forma a validação fica assertiva e objetiva.

4.6.2 LOGIN

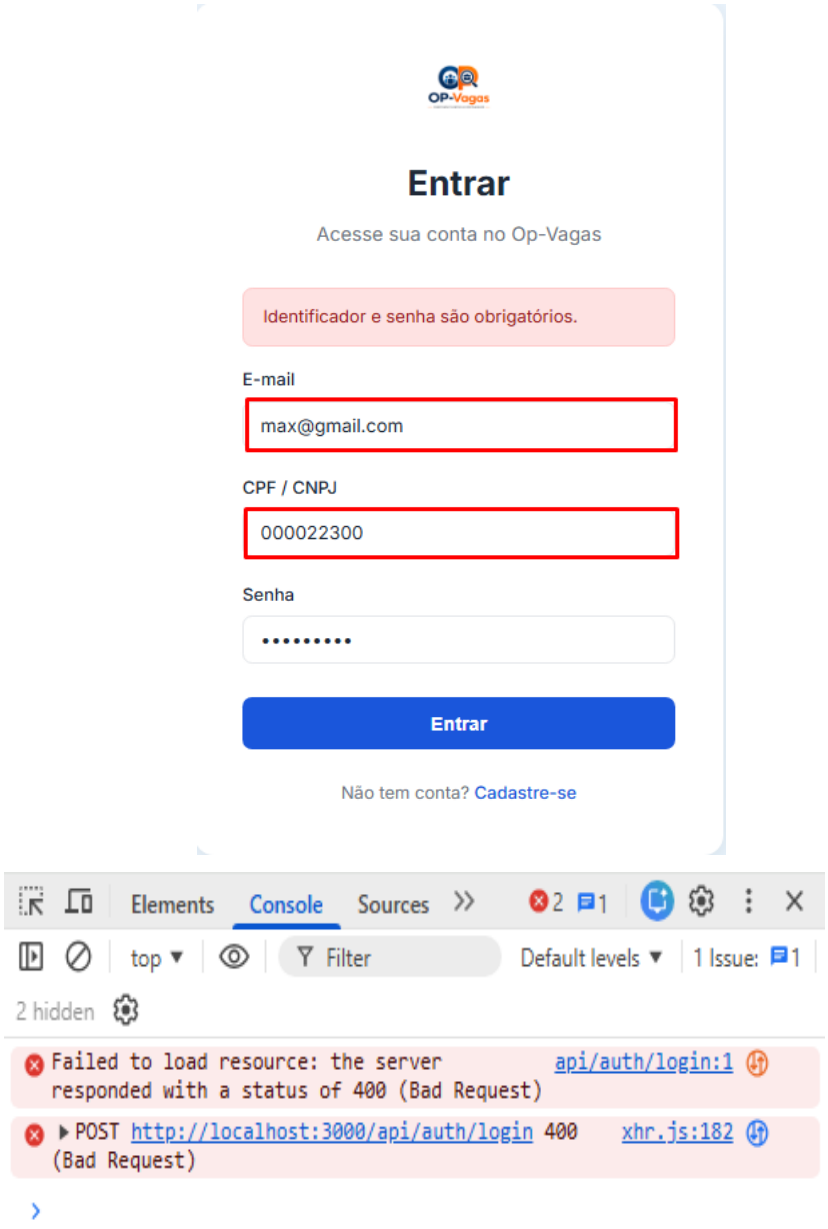
Figura 22 tela de login



Fonte: O Autor (2026).

A tela de login é onde o usuário vai digitar suas credenciais que foram informada lá na etapa do cadastro, com esses dados o sistema fará a busca no banco de dados para ver se os dados inseridos já estão cadastrados se já estiver cadastrado ele prosseguirá com login, se não for achado as credenciais inserido o usuário será redirecionado para a tela de cadastro.

Figura 23 teste na tela de login

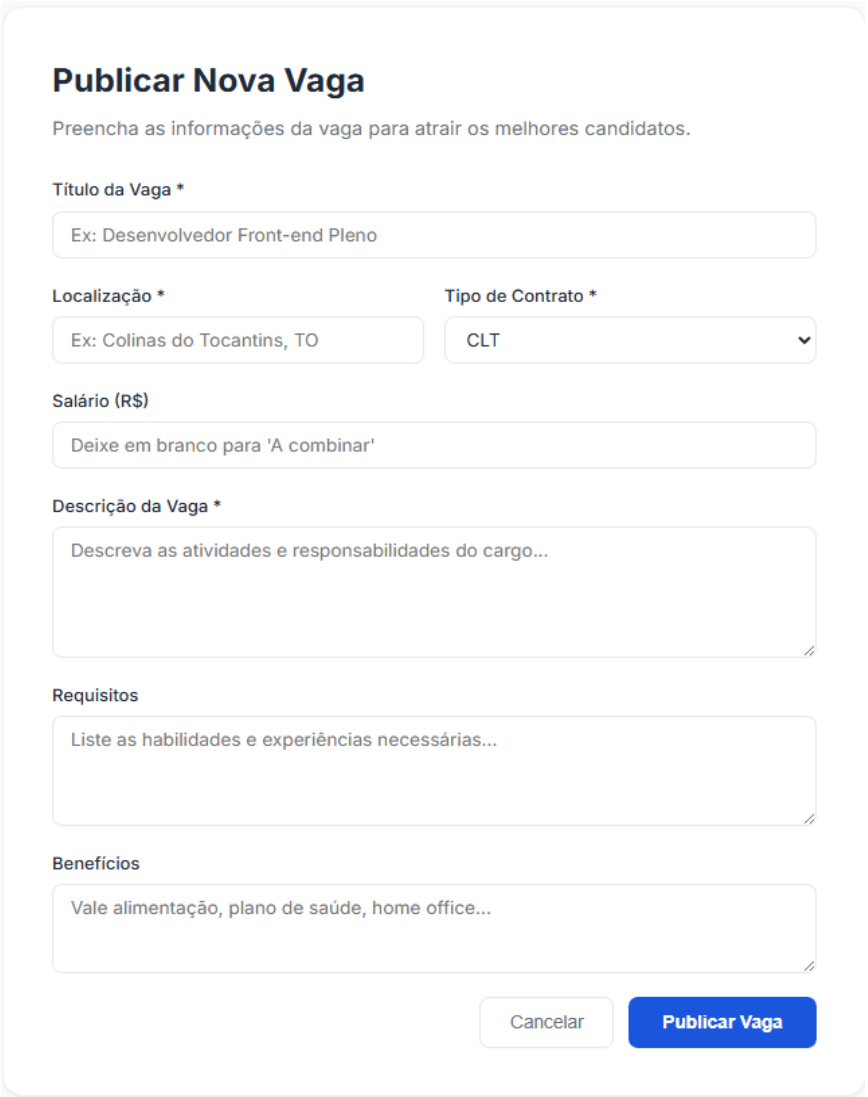


Fonte: O autor(2026)

Esse foi um teste para fazer a validação se a plataforma estaria realmente solicitando as informações que está se pedindo, desta forma vimos que aconteceu um erro, esse erro ocorreu devido não teve os dados de CFP/CNPJ e o e-mail não está correto essas duas informações não estão inseridas no banco de dados.

4.6.3 CADASTRO DE VAGAS

Figura 24 tela cadastro de vaga

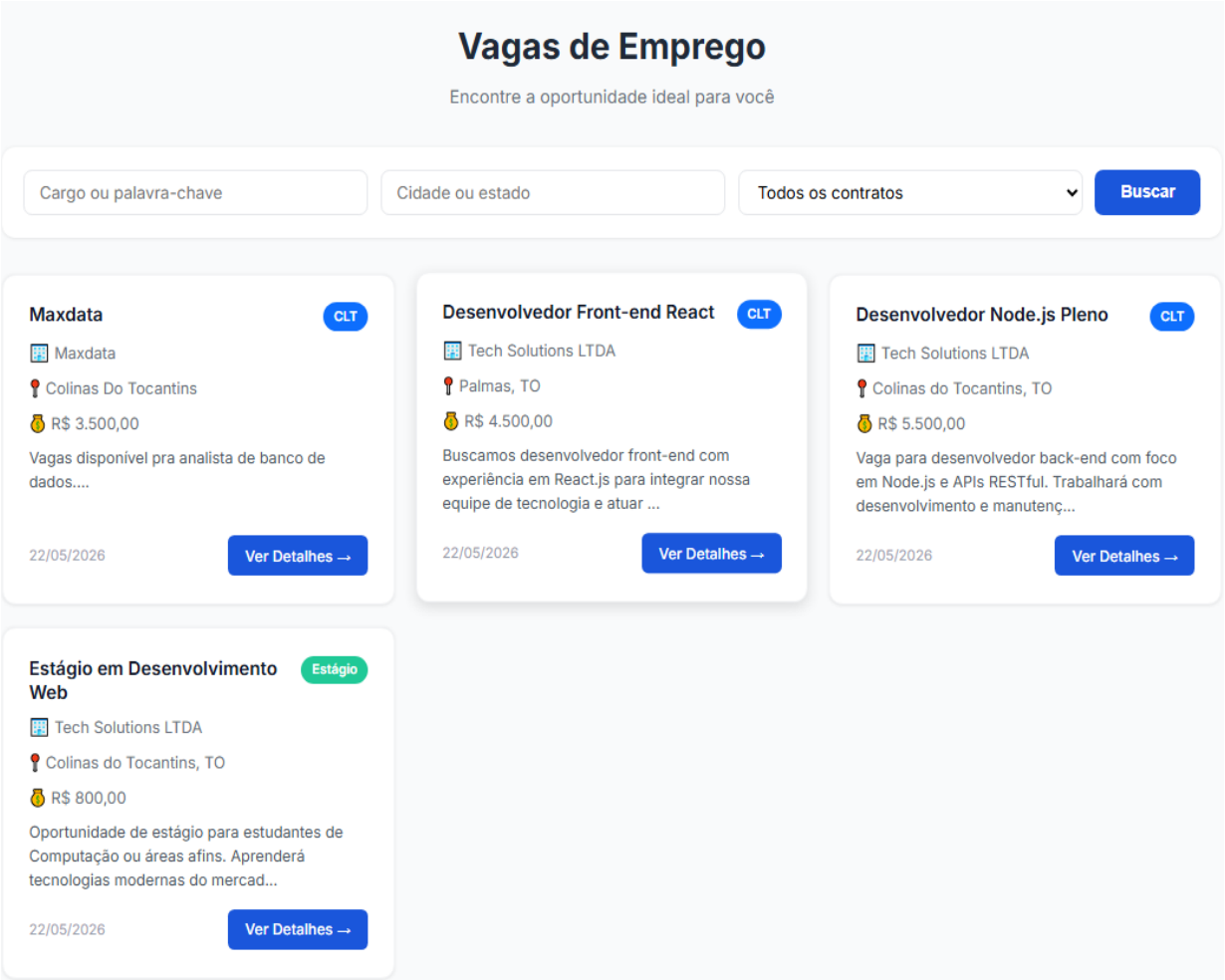


Fonte: O Autor(2026).

O cadastro de vagas pede informações referente às vagas, como Título da vaga, localização, tipo de contrato, salário, descrição da vaga, requisitos e benefícios esses tipo de informação são informações que podem levar o usuário candidato a ter ou não interesse pela vaga divulgada, o sistema visa a simplicidade e agilidade para que pessoas com baixo nível de escolaridade tenha o mesma oportunidade na hora de buscar pelos seus empregos.

4.6.4 LISTAGEM DE VAGAS

Figura 25 vagas publicadas



Fonte: O Autor (2026).

Esta tela foi desenvolvida para que o usuário consiga se identificar, de forma simples e objetiva, as vagas de seu interesse. O layout foi elaborado com foco na usabilidade e na melhor experiência durante a navegação.

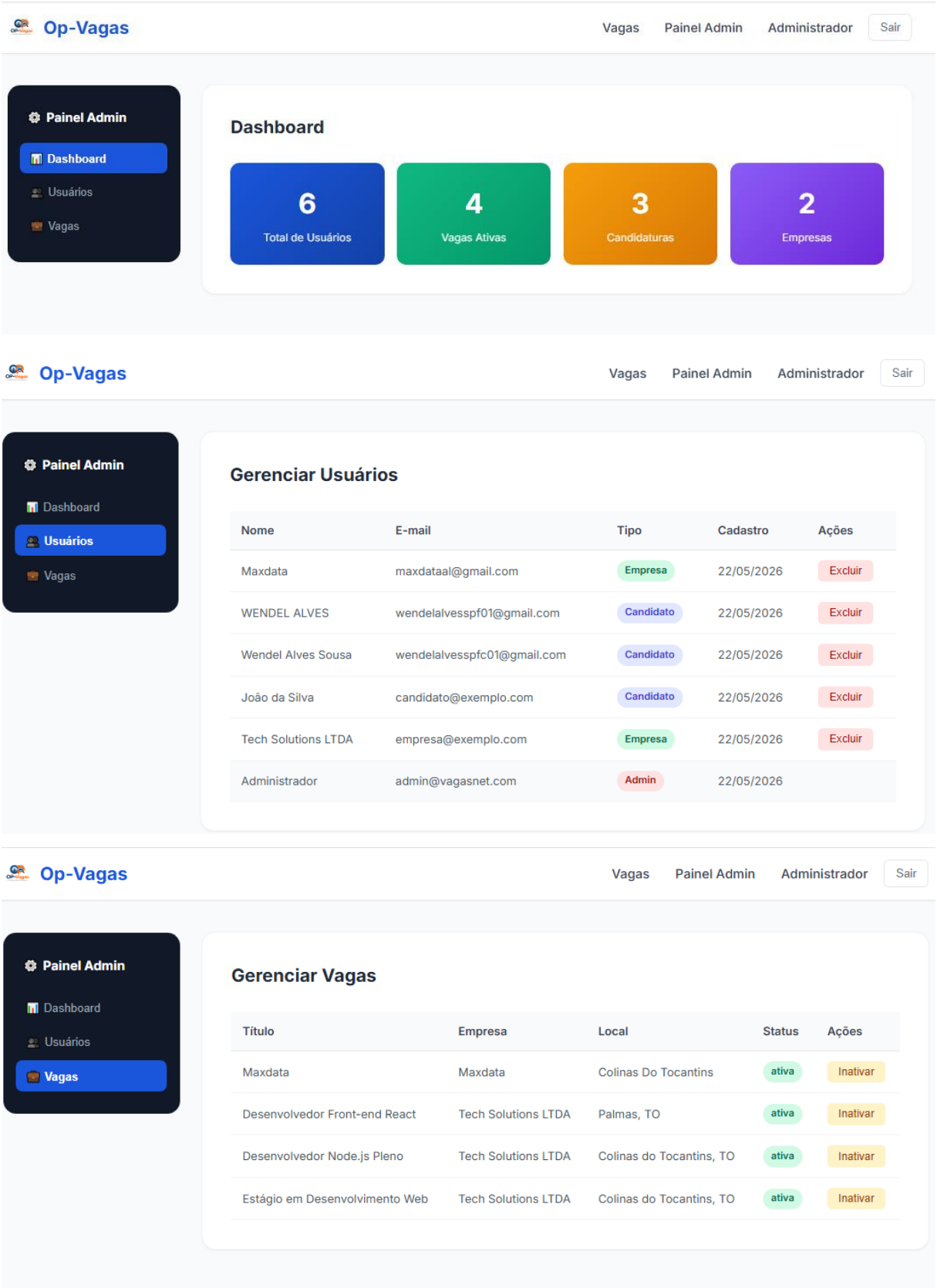
Nos cards das vagas são exibidas informações importantes, como o nome da empresa responsável pela divulgação, localização da vaga, faixa salarial e modalidade de contratação, podendo ser CLT, PJ ou Estagiário. Além disso, o usuário possui acesso à opção de visualizar mais detalhes da vaga e, posteriormente, realizar sua candidatura de forma prática e rápida.

Figura 26 detalhes vagas publicadas

Fonte: O Autor (2026)

4.6.5 PAINEL ADMINISTRATIVO

Figura 27 painel administrativo



Fonte: O Autor (2026).

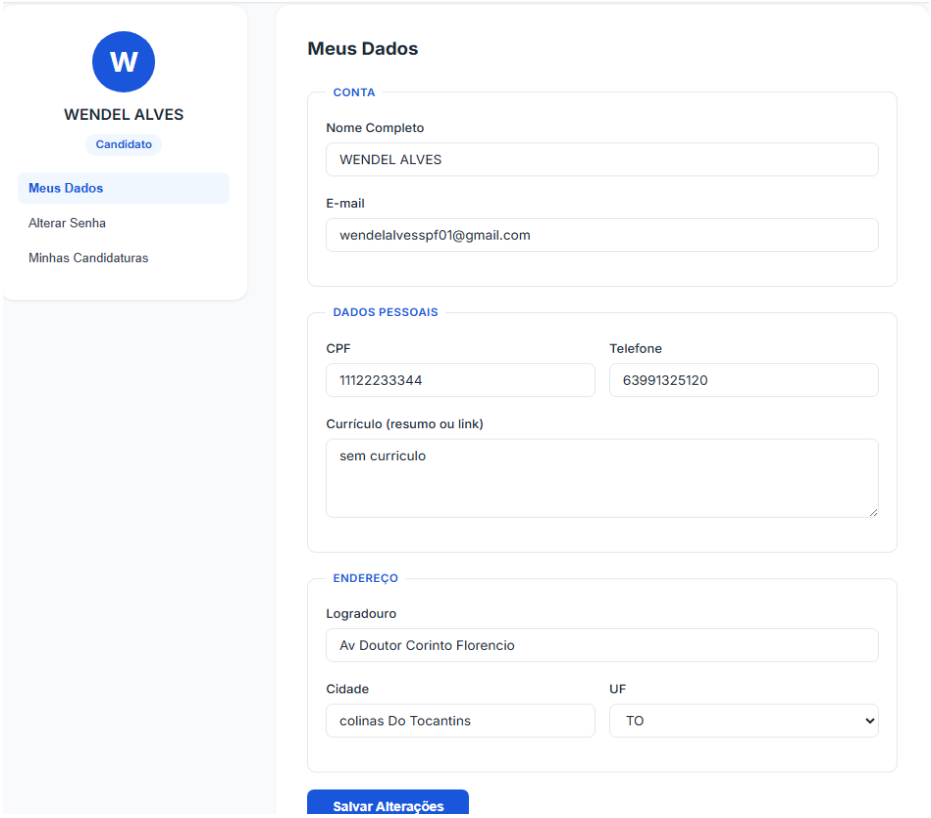
O painel administrativo foi desenvolvido com foco no acompanhamento e monitoramento da plataforma, sem interferir diretamente no fluxo principal do sistema. Por meio do dashboard, o administrador possui acesso a informações quantitativas, como número de empresas cadastradas, total de usuários, vagas ativas e quantidade de candidaturas realizadas.

Além disso, o painel permite acompanhar os usuários que utilizam a plataforma e visualizar, em tempo real, novos cadastros e movimentações relacionadas às vagas divulgadas.

O objetivo do painel administrativo não é realizar alterações frequentes em dados de usuários ou correções de vagas publicadas, pois essas tarefas poderiam comprometer o tempo destinado ao acompanhamento do crescimento da plataforma. Dessa forma, questões relacionadas a suporte e correções específicas ficam direcionadas ao setor responsável, permitindo que a administração concentre-se na análise dos resultados e no impacto da plataforma para seus usuários.

4.6.6 PERFIL DO USUÁRIO

Figura 28 perfil do usuário

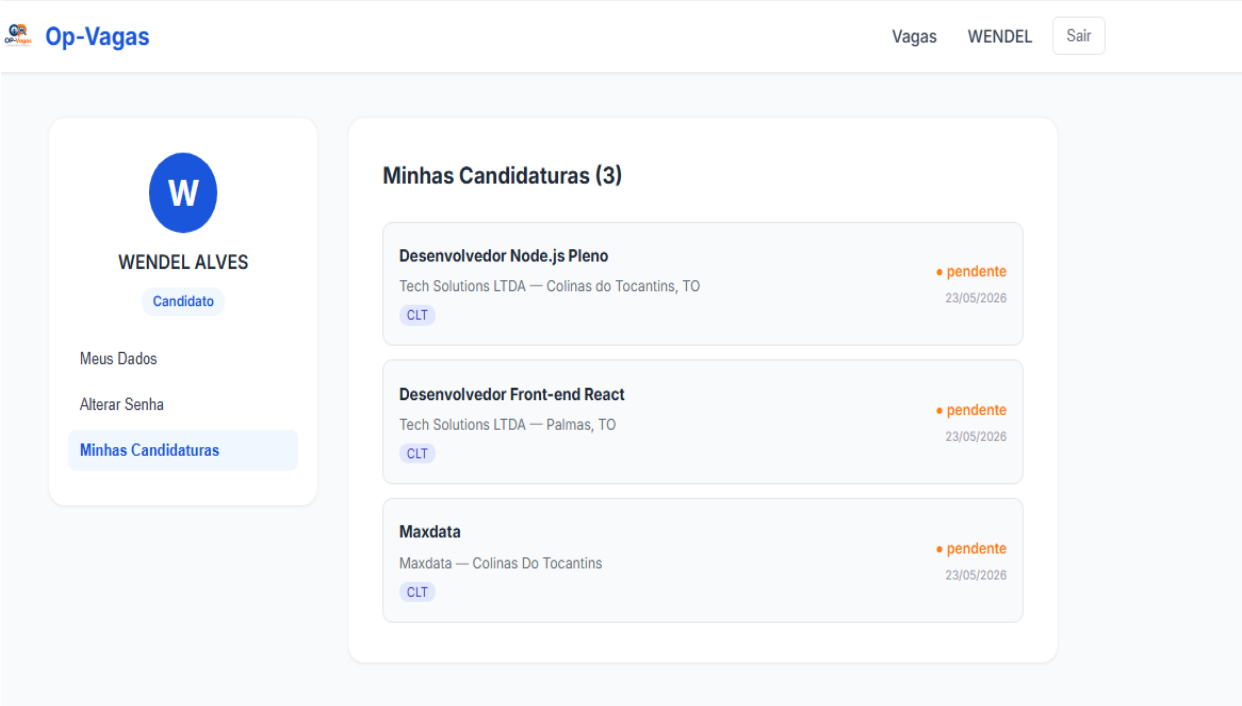


Fonte: O Autor (2026).

Na tela de perfil, o usuário poderá visualizar todos os dados cadastrados na plataforma, com a possibilidade de atualizá-los a qualquer momento. Também será disponibilizada uma opção para alteração de senha, garantindo maior controle e segurança da conta.

Além disso, o perfil contará com uma área destinada às candidaturas realizadas, onde o usuário poderá acompanhar o status de cada inscrição, facilitando o acompanhamento dos processos seletivos e futuras oportunidades de contratação.

Figura 29 demonstração candidaturas



Fonte: O Autor (2026).

4.7 API E ROTAS

A API REST foi estruturada com os seguintes endpoints principais:

Tabela 2 – Endpoints da API REST

Método	Endpoint	Descrição	Auth
POST	/api/auth/registrar	Cadastro de usuário	Não

Método	Endpoint	Descrição	Auth
POST	/api/auth/login	Login (retorna JWT)	Não
GET	/api/vagas	Listar vagas ativas	Não
GET	/api/vagas/:id	Detalhes de uma vaga	Não
POST	/api/vagas	Criar nova vaga	Empresa
PUT	/api/vagas/:id	Editar vaga	Empresa
DELETE	/api/vagas/:id	Excluir vaga	Empresa
GET	/api/vagas/empresa/minhas	Vagas da empresa	Empresa
POST	/api/vagas/:id/candidatar	Candidatar-se à vaga	Candidato
GET	/api/vagas/candidaturas/minhas	Minhas candidaturas	Candidato
GET	/api/vagas/:id/candidaturas	Candidatos da vaga	Empresa
GET	/api/usuarios/perfil	Ver perfil próprio	Autenticado
PUT	/api/usuarios/perfil	Editar perfil	Autenticado
PUT	/api/usuarios/senha	Alterar senha	Autenticado
GET	/api/dados-usuario	Dados complementares	Autenticado
POST	/api/dados-usuario	Salvar dados completos	Autenticado
GET	/api/admin/dashboard	Estatísticas gerais	Admin
GET	/api/admin/usuarios	Listar usuários	Admin
DELETE	/api/admin/usuarios/:id	Excluir usuário	Admin
GET	/api/admin/vagas	Listar todas as vagas	Admin
PUT	/api/admin/vagas/:id/status	Ativar/desativar vaga	Admin

Fonte: elaborado pelo autor (2026).

Exemplo de implementação de rota no Express (rotas de vagas):

Figura 30 demonstração de rotas

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');

const authRoutes = require('./routes/auth');
const vagasRoutes = require('./routes/vagas');
const usuariosRoutes = require('./routes/usuarios');
const adminRoutes = require('./routes/admin');
const dadosUsuarioRoutes = require('./routes/dadosUsuario');
const { conectar } = require('./config/database');

const app = express();

// Middlewares globais
app.use(cors({
  origin: process.env.NODE_ENV === 'production' ? 'https://seu-dominio.com' : 'http://localhost:3000',
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization'],
}));
app.use(express.json());

// Rotas
app.use('/api/auth', authRoutes);
app.use('/api/vagas', vagasRoutes);
app.use('/api/usuarios', usuariosRoutes);
app.use('/api/admin', adminRoutes);
app.use('/api/dados-usuario', dadosUsuarioRoutes);

// Rota de saúde
app.get('/api/health', (req, res) => {
  res.json({ status: 'online', sistema: 'Vagas de Emprego API', versao: '1.0.0' });
});

// Rota não encontrada
app.use((req, res) => {
  res.status(404).json({ erro: 'Rota não encontrada.' });
});
```

Fonte: O Autor (2026).

A imagem acima mostra o ponto de entrada do servidor back-end da aplicação. Nele são carregadas as configurações de ambiente, registrados os middlewares globais, que autoriza a comunicação com o front-end, e o parser JSON, que interpreta o corpo das requisições que são mapeadas, todas as rotas da API para seus respectivos módulos. Ao ser inicializado, o servidor estabelece a conexão com o banco de dados e, somente após confirmá-la, começa a aceitar requisições na porta definida, garantindo que nenhuma chamada seja processada sem que o banco esteja disponível.

4.8 SEGURANÇA DA APLICAÇÃO

A segurança foi tratada em múltiplas camadas do sistema:

- Autenticação JWT: todas as rotas protegidas verificam a validade do token JWT antes de processar a requisição. O token possui prazo de expiração de 7 dias, configurado via variável de ambiente, reduzindo o risco em caso de interceptação.
- Hash de senhas: as senhas são armazenadas com hash bcrypt (fator de custo 12), garantindo que mesmo em caso de vazamento do banco de dados, as senhas originais não sejam recuperáveis.
- Prevenção de SQL Injection: todas as consultas ao banco de dados utilizam parâmetros parametrizados (prepared statements) via a biblioteca mssql, eliminando o risco de injeção de SQL.
- Controle de acesso por perfil: middlewares verificam o tipo de usuário (candidato, empresa, administrador) antes de permitir o acesso a rotas específicas, implementando o princípio do menor privilégio.
- Variáveis de ambiente: dados sensíveis como a chave secreta do JWT, a string de conexão ao banco de dados e outras configurações são armazenados em variáveis de ambiente via arquivo .env, não sendo expostos no código-fonte.
- Validação de entrada: todos os dados enviados pelo cliente são validados no back-end antes do processamento, prevenindo a inserção de dados inválidos ou maliciosos.

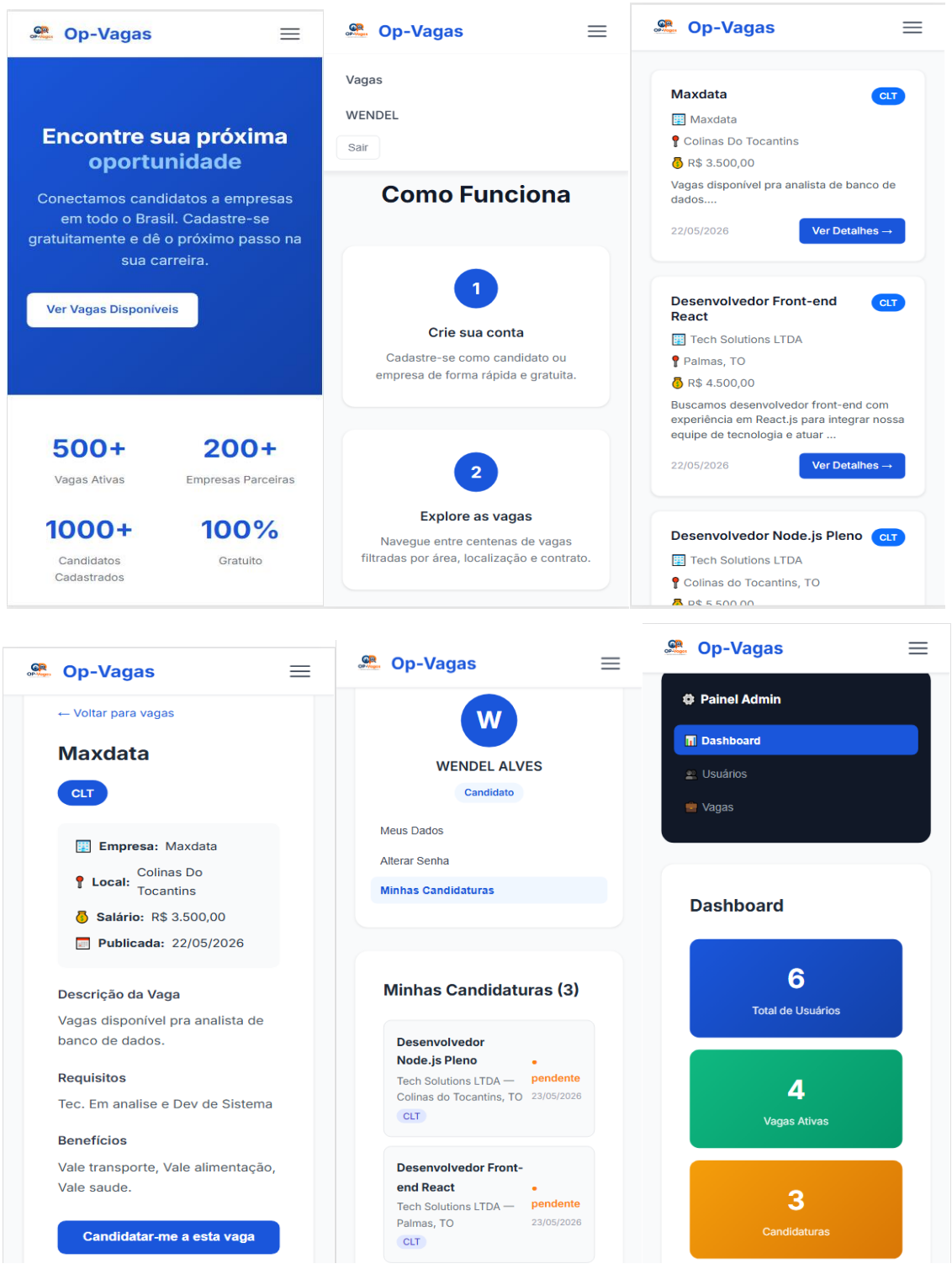
4.9 RESPONSIVIDADE

A responsividade do sistema foi implementada por meio de CSS com media queries, garantindo que a interface se adapte adequadamente a três faixas de tela:

- Desktop: a partir de 1024px — layout completo com sidebar e múltiplas colunas;
- Tablet: entre 768px e 1023px — layout adaptado com menu colapsável;
- Mobile: abaixo de 768px — layout em coluna única com navegação por menu hambúrguer.

A abordagem Mobile foi adotada no desenvolvimento do CSS, definindo primeiro os estilos para telas menores e sobrescrevendo-os progressivamente para telas maiores. Essa abordagem garante uma experiência satisfatória também para usuários que acessam o sistema via smartphone.

Figura 31 demonstração da telas responsivas



Fonte: O Autor (2026).

As imagens das telas apresentadas acima demonstram o funcionamento do sistema em dispositivos móveis, utilizando recursos de responsividade e compatibilidade no modelo de smartphone iPhone 12 Pro. Dessa forma, a aplicação garante boa usabilidade tanto em computadores desktop quanto em dispositivos mobile, proporcionando uma navegação adaptada a diferentes tamanhos de tela.

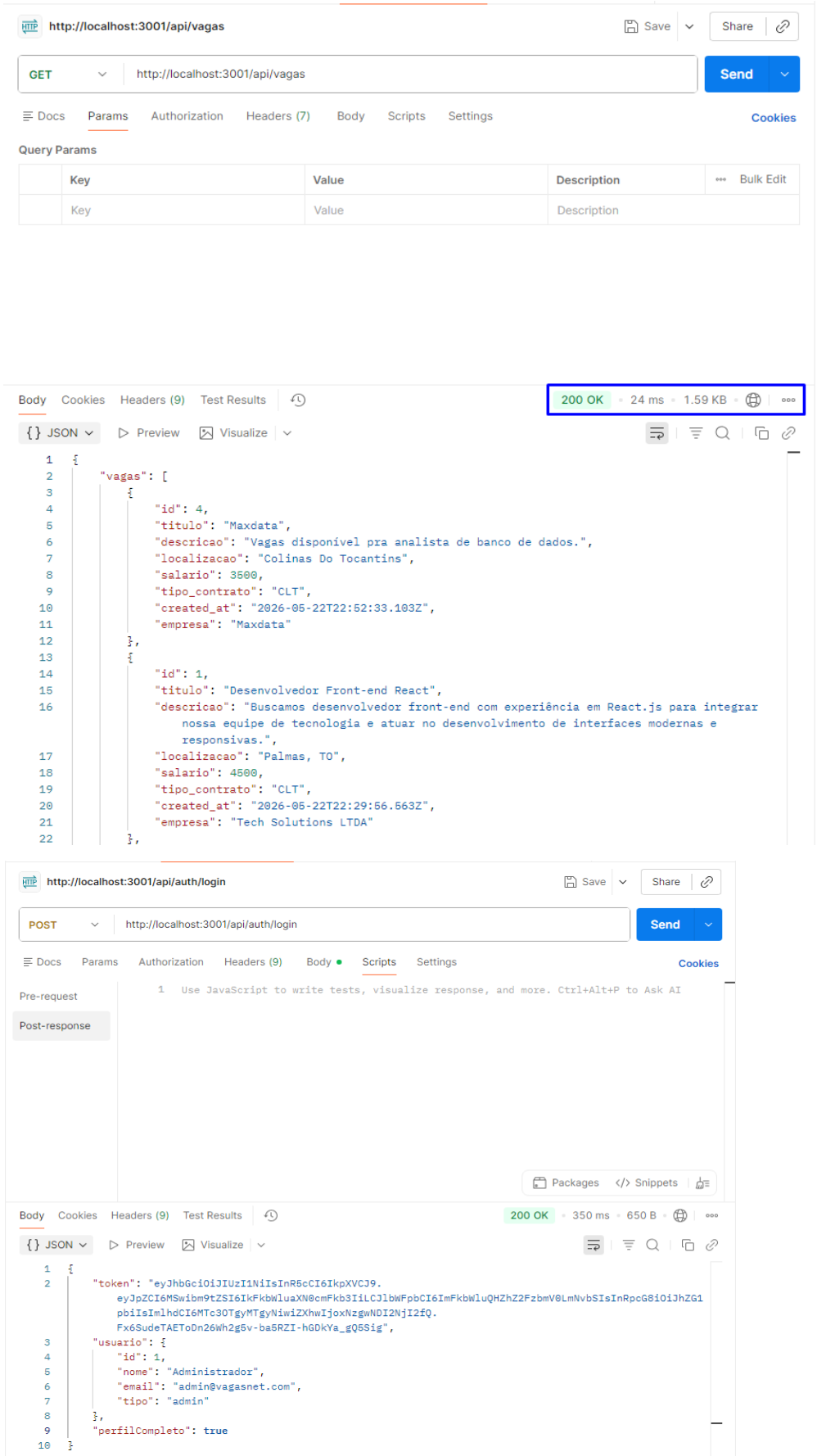
5 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos a partir do desenvolvimento e dos testes funcionais realizados no sistema Op-Vagas, discutindo seu desempenho, usabilidade, vantagens, limitações e possíveis melhorias. A análise é realizada com base nos requisitos funcionais (RF) e não funcionais (RNF) levantados na Seção 3.2 e nas funcionalidades implementadas ao longo do Capítulo 4.

5.1 DESEMPENHO

O desempenho do sistema foi avaliado por meio de testes manuais realizados com o Postman, ferramenta utilizada para enviar requisições à API e medir o tempo de resposta de cada funcionalidade. Os testes foram realizados em ambiente local, com o servidor rodando na mesma máquina utilizada para o desenvolvimento. A listagem de vagas retornou resposta em aproximadamente 24 milissegundos, enquanto o processo de login levou cerca de 350 milissegundos — tempo maior devido à verificação segura da senha, que envolve um processamento intencional mais demorado para dificultar tentativas de invasão. Em ambos os casos, os tempos registrados ficaram bem abaixo do limite de 3 segundos definido pelo requisito RNF01, confirmando que o sistema responde de forma satisfatória às ações do usuário.

Figura 32 demonstração de desempenho



Fonte: O Autor(2026).

O Node.js, tecnologia utilizada no servidor do sistema, é capaz de atender várias requisições ao mesmo tempo sem travar, o que contribui para manter o desempenho estável mesmo com múltiplos acessos simultâneos. Além disso, a listagem de vagas foi configurada para exibir apenas 10 resultados por vez, carregando mais conforme o usuário avança nas páginas. Isso evita que o sistema precise buscar e enviar muitos dados de uma só vez, ajudando a manter os tempos de resposta baixos mesmo quando há muitas vagas cadastradas.

Cabe ressaltar que os testes foram realizados em ambiente de desenvolvimento local, sem a sobrecarga de uma rede pública ou de múltiplos usuários simultâneos. Em um cenário de produção com acesso pela internet, fatores como latência de rede, carga simultânea e configurações do servidor podem impactar os tempos de resposta. Não obstante, a arquitetura adotada — separação entre front-end, back-end e banco de dados — oferece flexibilidade para escalabilidade horizontal quando necessário.

5.2 USABILIDADE

A interface do sistema foi desenvolvida priorizando a simplicidade, a consistência visual e o feedback imediato das ações realizadas. A navegação é organizada de forma hierárquica, com a barra de navegação sempre visível, permitindo ao usuário identificar rapidamente sua localização dentro da aplicação.

Os formulários implementados contam com validação em dupla camada: no front-end, mensagens de erro são exibidas em tempo real antes do envio; no back-end, os dados são novamente validados antes do processamento, prevenindo inconsistências mesmo em caso de requisições diretas à API.

O fluxo principal de candidatura a uma vaga foi projetado para ser concluído em, no máximo, quatro etapas: (1) busca ou visualização da vaga na listagem; (2) acesso à página de detalhes; (3) clique no botão "Candidatar-se"; (4) confirmação visual da candidatura. Esse fluxo direto contribui para a eficiência de uso, reduzindo a carga cognitiva do candidato e atendendo ao requisito RF04.

Para usuários do tipo empresa, o painel de gerenciamento de vagas permite criar, editar, desativar e visualizar os candidatos inscritos de forma centralizada,

atendendo aos requisitos RF03 e RF06. Já o painel administrativo oferece uma visão global do sistema, com estatísticas de uso e controle completo sobre usuários e vagas, atendendo ao RF07.

5.3 VANTAGENS

O sistema desenvolvido apresenta vantagens concretas em relação às práticas informais de divulgação de vagas ainda predominantes em municípios de pequeno e médio porte, como Colinas do Tocantins:

a) Centralização: todas as vagas disponíveis na região podem ser consultadas em um único ambiente digital, eliminando a dispersão de informações em grupos de aplicativos de mensagens, painéis físicos e indicações informais;

b) Formalização e rastreabilidade: o processo de candidatura é registrado no banco de dados com data, status e vínculo entre candidato e vaga, possibilitando que ambas as partes acompanhem o andamento do processo seletivo de forma organizada;

c) Acessibilidade multiplataforma: por se tratar de uma aplicação web responsiva, o sistema pode ser acessado a partir de qualquer dispositivo com navegador e conexão à internet — computadores, tablets e smartphones —, sem necessidade de instalação;

d) Segurança e privacidade: as senhas são armazenadas com hash bcrypt (fator 12), a comunicação é autenticada via JWT e os dados sensíveis são mantidos em variáveis de ambiente, em conformidade com os requisitos RNF03 e RNF04;

e) Escalabilidade arquitetural: a separação entre as camadas de apresentação, lógica de negócio e persistência de dados permite que cada componente seja escalado ou substituído de forma independente, sem impacto nas demais camadas;

f) Viabilidade econômica: por utilizar exclusivamente tecnologias de código aberto (Node.js, React, Express) e a versão gratuita do SQL Server Express, o custo de desenvolvimento e operação é significativamente menor em comparação a soluções comerciais equivalentes.

5.4 LIMITAÇÕES

A os resultados positivos alcançados, o sistema apresenta limitações que devem ser reconhecidas, pois refletem um Trabalho de Conclusão de Curso desenvolvido por um único desenvolvedor:

a) Ausência de notificações em tempo real: o sistema não implementa WebSockets nem mecanismos de push notification. As atualizações de status das candidaturas, por exemplo, só são percebidas pelo candidato ao recarregar manualmente a página de perfil, o que compromete a experiência em cenários dinâmicos;

b) Busca por correspondência textual simples: o mecanismo de filtragem de vagas é baseado em correspondência parcial de texto (operador LIKE do SQL), sem suporte a busca por relevância, sinônimos ou erros de digitação. Isso pode resultar em buscas menos precisas quando comparado a motores de busca dedicados;

c) Ausência de comunicação por e-mail: o sistema não envia notificações automáticas por e-mail para confirmação de cadastro, candidatura ou atualização de status, o que limita o engajamento do usuário fora da plataforma. O requisito RF09 foi parcialmente atendido apenas com feedback visual dentro da aplicação;

d) Ambiente de execução local: o sistema foi desenvolvido e testado exclusivamente em ambiente local (localhost), não tendo passado por processo de implantação em servidor público. Para disponibilização em produção, seria necessária a configuração de infraestrutura de hospedagem, domínio e certificado SSL;

e) Ausência de testes automatizados: os testes realizados foram de natureza funcional e manual. A ausência de uma suíte de testes automatizados (unitários e de integração) aumenta o risco de regressões em manutenções futuras.

5.5 POSSÍVEIS MELHORIAS

Com base nas limitações identificadas e nas tendências da Engenharia de Software, propõem-se as seguintes melhorias para versões futuras do sistema:

a) Notificações em tempo real: implementação de comunicação bidirecional com Socket.IO, permitindo que candidatos sejam notificados instantaneamente sobre atualizações no status de suas candidaturas, sem necessidade de recarregar a página;

b) Envio de e-mails transacionais: integração com a biblioteca Nodemailer ou com serviços como SendGrid para envio automático de e-mails de confirmação de cadastro, candidatura e atualização de status, atendendo integralmente ao requisito RF09;

c) Autenticação social: adição de login via OAuth 2.0 (Google, LinkedIn), simplificando o processo de acesso e reduzindo a barreira de entrada para novos usuários;

d) Busca avançada e filtros inteligentes: substituição do operador LIKE por mecanismos de busca full-text nativos do SQL Server (CONTAINS/FREETEXT) ou integração com Elasticsearch, melhorando significativamente a relevância dos resultados;

e) Testes automatizados: implementação de uma suíte de testes com Jest e Supertest, cobrindo os principais endpoints da API e os componentes React críticos, garantindo maior confiabilidade em ciclos de manutenção e evolução do sistema;

f) Implantação em nuvem: configuração de deploy automatizado em plataformas como AWS, Railway ou Render, com uso de HTTPS e variáveis de ambiente gerenciadas por serviço de secrets, viabilizando o uso público da aplicação;

g) Aplicativo móvel: desenvolvimento de uma versão nativa para dispositivos móveis utilizando React Native, reutilizando a API REST existente e ampliando o alcance da plataforma para usuários com acesso predominante via smartphone.

6 CONSIDERAÇÕES FINAIS

Este Trabalho de Conclusão de Curso teve como objetivo o desenvolvimento de um sistema web para cadastro e gerenciamento de vagas de emprego, denominado Op-Vagas, utilizando as tecnologias React no front-end, Node.js com Express no back-end e SQL Server como sistema gerenciador de banco de dados. O trabalho partiu da identificação de uma lacuna concreta: a ausência de uma plataforma digital regional que intermediasse de forma organizada e segura a relação entre candidatos e empresas na região de Colinas do Tocantins - TO.

Ao longo do desenvolvimento, os objetivos específicos propostos foram atendidos de forma sistemática. O levantamento de requisitos produziu dez requisitos funcionais e seis não funcionais, que orientaram todas as decisões de projeto. A modelagem do sistema resultou em diagramas de caso de uso, classes e entidade-relacionamento, além de um fluxograma do sistema, que documentaram a estrutura e o comportamento esperados antes da implementação. O back-end foi construído com uma API REST organizada, protegida por autenticação JWT e com controle de acesso por perfil de usuário. O front-end foi desenvolvido com React, adotando padrões como Context API, hooks e roteamento com React Router. A integração entre as camadas foi validada por meio de testes funcionais manuais, que confirmaram o comportamento correto das principais funcionalidades do sistema.

Do ponto de vista técnico, o trabalho demonstrou que o ecossistema JavaScript full Stack combinando React, Node.js e SQL Server é capaz de sustentar o desenvolvimento de uma aplicação web completa, funcional e segura, sem a necessidade de múltiplas linguagens de programação ou ferramentas proprietárias de alto custo. A adoção de boas práticas de segurança, como hash de senhas com bcrypt, tokens JWT com prazo de expiração configurável, consultas parametrizadas para prevenção de SQL Injection e separação de configurações sensíveis em variáveis de ambiente, demonstrou a aplicabilidade das recomendações em um projeto acadêmico.

Do ponto de vista social, o Op-Vagas representa uma contribuição tangível para a comunidade local. A disponibilização de uma plataforma digital voltada ao mercado de trabalho regional tem potencial para reduzir a informalidade nos processos

seletivos, ampliar a visibilidade das vagas disponíveis e facilitar o acesso de candidatos e empresas a um ambiente de recrutamento mais transparente e eficiente.

Do ponto de vista acadêmico, este trabalho integrou de forma prática conteúdos de múltiplas disciplinas do Curso de Licenciatura em Computação do IFTO: Programação Web, Banco de Dados, Engenharia de Software, Redes de Computadores e Segurança da Informação. A experiência de conduzir todas as etapas do ciclo de desenvolvimento da análise de requisitos à implementação e aos testes consolidou competências que extrapolam o domínio técnico, abrangendo também o planejamento, a documentação e a tomada de decisões em contextos com restrições reais de tempo e recursos.

As limitações identificadas no Capítulo 5 como a ausência de notificações em tempo real, a busca textual simples e o ambiente exclusivamente local não comprometem a validade dos resultados alcançados, mas apontam direcionamentos naturais para a evolução do sistema em trabalhos futuros. A arquitetura modular adotada favorece a incorporação gradual dessas melhorias sem necessidade de reestruturação do projeto.

Conclui-se que o desenvolvimento do Op-Vagas atingiu plenamente seus objetivos, gerando um produto funcional e relevante tanto para o contexto regional quanto para a formação do autor. O domínio das tecnologias JavaScript full stack, aqui aplicado de forma integrada e orientada a um problema real, representa uma competência de alto valor tanto para a atuação profissional na área de desenvolvimento de software quanto para o exercício da docência em Computação na Educação Básica finalidade central da Licenciatura em Computação do IFTO.

REFERÊNCIAS

BARSOTTI, Nathan; GIBERTONI, Daniela. **IMPACTO QUE O SEQUELIZE TRAZ PARA O DESENVOLVIMENTO DE UMA API CONSTRUÍDA EM NODE.JS COM EXPRESS.JS. Revista Interface Tecnológica**, Taquaritinga, SP, v. 17, n. 2, p. 231–243, 2020. DOI: https://doi.org/10.31510/infa.v17i2.964.

GOTARDO, Reginaldo Aparecido. **Linguagem de programação I**. 1. ed. Rio de Janeiro: SESES, 2015. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/45886476/LIVRO_PROPRIETARIO_-_Linguagem_de_Programa-libre.pdf. Acesso em: 15 maio 2025.

GURGEL, Sanderson; OLIVEIRA, Keverson Soares de. **TagBiometrick: um protótipo para controle no horário de saída escolar dos estudantes no ensino infantil**. 2022. Trabalho de Conclusão de Curso (Pós-Graduação Lato Sensu em Conectividade e Tecnologias da Informação) – Instituto Federal do Espírito Santo, Campus Colatina, Colatina, 2022. Disponível em: https://repositorio.ifes.edu.br/server/api/core/bitstreams/9d94dda1-cf51-45e3-b5aa-2614287f8266/content. Acesso em: 15 maio 2026.

JONES, M.; BRADLEY, J.; SAKIMURA, N. RFC 7519: **JSON Web Token (JWT). Fremont**: IETF, 2015. Disponível em: https://www.rfc-editor.org/rfc/rfc7519.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de Informação Gerenciais**. 14. ed. São Paulo: Pearson, 2016.

LOBO, Antônio Hugo Ribeiro Pereira. **Refatoração de code smells em aplicações React.js**. 2025. 84 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software)- Campus de Quixadá, Universidade Federal do Ceará, Quixadá, 2025. Disponível em: https://repositorio.ufc.br/handle/riufc/80586. Acesso em: 15 maio 2025.

MENDES, José Manuel Vicente. **Análise do controle de acesso no sistema de gestão de base de dados Microsoft SQL Server 2022**. 2023. Dissertação (Mestrado em Engenharia de Segurança Informática) – Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Beja, Beja, 2023. Disponível em: https://repositorio.ipbeja.pt/server/api/core/bitstreams/08e41b3c-3694-4725-9b25-2c3a70d9dd4c/content. Acesso em: 15 maio 2026.

MORORÓ, Jadson Faustino. **Um estudo comparativo entre JavaScript e TypeScript**. 2024. 154 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Campus de Sobral, Universidade Federal do Ceará, Sobral, 2024. Disponível em: https://repositorio.ufc.br/handle/riufc/78817. Acesso em: 15 maio 2025.

OLIVEIRA, Alex Santos de. **Desenvolvimento de uma extensão para o Visual Studio Code com suporte ao método FrameWeb**. 2026. Monografia (Graduação em Ciência da Computação) – Centro Tecnológico, Universidade Federal do Espírito Santo, Vitória, 2026. Disponível em: https://www.inf.ufes.br/~vitorsouza/wp-content/papercite-data/pdf/oliveira-pg26.pdf. Acesso em: 15 maio 2026.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software: uma abordagem profissional**. 9. ed. Porto Alegre: AMGH, 2021.

SANTOS, Gabriel Belloni Benevenuto dos. **Análise comparativa entre Cypress e Postman para automação de testes de API**. 2024. Trabalho de Conclusão de Curso (Graduação) – Universidade de Passo Fundo, Passo Fundo, 2024. Disponível em: https://repositorio.upf.br/server/api/core/bitstreams/e7888d60-0bbf-4738-96bd-948e117b9d0c/content. Acesso em: 15 maio 2026.

SERRA, Ricardo Jorge Maia e. **Interfaces tácteis baseadas em HTML5/CSS3/JavaScript**. 2011. Dissertação (Mestrado Integrado em Engenharia Informática e Computação) – Faculdade de Engenharia, Universidade do Porto, Porto, 2011. Disponível em: https://repositorio-aberto.up.pt/bitstream/10216/63293/1/000149242.pdf. Acesso em: 15 maio 2026.

SIMBINE, Jeremias Filipe. **Trabalho de pesquisa sobre base de dados e linguagem SQL**. 2021. Trabalho de Pesquisa (Disciplina de Práticas de Base de Dados) – Departamento de Informática, Faculdade de Engenharia e Tecnologia, Universidade Pedagógica de Maputo, Maputo, 2021. Disponível em: https://www.kufunda.net/publicdocs/Trabalho%20PBD%20Jeremias.pdf. Acesso em: 15 maio 2026.

VALENTE, Marco Túlio. **Engenharia de Software Moderna: princípios e práticas para desenvolvimento de software com produtividade**. Belo Horizonte:

Independente, 2020. Disponível em: https://engsoftmoderna.info. Acesso em: 24 maio 2020.