INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PICOS
CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MATHEUS LORAN LOIOLA SOARES

ANÁLISE COMPARATIVA DE TECNOLOGIAS JAVASCRIPT FOCADAS NO FRONT-END PARA DESENVOLVIMENTO WEB

PICOS ,PIAUÍ

2026

MATHEUS LORAN LOIOLA SOARES

ANÁLISE COMPARATIVA DE TECNOLOGIAS JAVASCRIPT FOCADAS NO FRONT-END PARA DESENVOLVIMENTO WEB

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e desenvolvimento de sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos .

Orientador: Profª João Paulo Lima do Nascimento

PICOS , PIAUÍ

2026

ANÁLISE COMPARATIVA DE TECNOLOGIAS JAVASCRIPT FOCADAS NO FRONT-END PARA DESENVOLVIMENTO WEB

Matheus Loran Loiola Soares[1]
João Paulo Lima do Nascimento[2]

RESUMO

O desenvolvimento de aplicações *web* modernas têm apresentado grande evolução devido ao crescimento constante de tecnologias baseadas em JavaScript, tornando necessária a análise das ferramentas utilizadas na construção de interfaces digitais. Este trabalho aborda uma análise comparativa entre as tecnologias React, Vue.js e Next.js aplicadas ao desenvolvimento *front-end*, com o objetivo de identificar suas principais características, vantagens e limitações em diferentes cenários de desenvolvimento. A pesquisa foi realizada por meio de uma abordagem bibliográfica, exploratória e comparativa, utilizando documentos oficiais, artigos científicos, materiais técnicos e informações relacionadas ao uso dessas tecnologias. Foram analisados aspectos como arquitetura, desempenho, facilidade de configuração, curva de aprendizado, documentação, ecossistema, popularidade e empregabilidade. Os resultados obtidos permitiram identificar diferenças entre tecnologias analisadas evidenciando aspectos técnicos e mercadológicos que subsidiam sua comparação e discussão ao longo do trabalho. O estudo contribui para a compreensão das características das tecnologias analisadas, fornecendo subsídios para futuras pesquisas e para a avaliação de tecnologias voltadas ao desenvolvimento de aplicações *web*.

Palavras-chave: JavaScript , Desenvolvimento *web* , *Front-end* ,React , Vue.js e Next.js .

[1]Graduando em Análise e desenvolvimento de sistemas. Instituição de ensino. E-mail
[2]Mestre em Engenharia de Produção (UFRN). IFPI. joao.nascimento@ifpi.edu.br.

ABSTRACT

The development of modern web applications has experienced significant growth due to the continuous advancement of JavaScript-based technologies, making it essential to analyze the tools used in the development of digital interfaces. This study presents a comparative analysis of the React, Vue.js, and Next.js technologies applied to front-end development, aiming to identify their main characteristics, advantages, and limitations in different development scenarios. The research was conducted through a bibliographic, exploratory, and comparative approach, using official documentation, scientific articles, technical materials, and information related to the use of these technologies. The analysis considered aspects such as architecture, performance, configuration complexity, learning curve, documentation, ecosystem, popularity, and employability. The results made it possible to identify differences among the analyzed technologies, highlighting technical and market-related aspects that support their comparison and discussion throughout the study. This research contributes to a better understanding of the characteristics of the analyzed technologies, providing support for future studies and for the evaluation of technologies applied to modern web application development.

Keywords: JavaScript ,Web development , Front-end, React , Vue.js and Next.js.

Data de aprovação: 31/07/2026

1 INTRODUÇÃO

O desenvolvimento de aplicações *web* evoluiu significativamente nas últimas décadas, impulsionado pelo avanço das tecnologias digitais e pela crescente demanda por sistemas cada vez mais interativos, responsivos e eficientes (Crispiniano, 2021). Nesse contexto, a linguagem JavaScript tornou-se uma das principais tecnologias para o desenvolvimento *front-end*, permitindo a criação de interfaces dinâmicas e proporcionando melhor experiência aos usuários. Ao mesmo tempo , surgiram diversos *frameworks* e bibliotecas que ampliaram as possibilidades de desenvolvimento, oferecendo recursos voltados à reutilização de componentes,

organização do código e aumento da produtividade das equipes de desenvolvimento (Martins Filho, 2023 ; Alves Pinto; Hoffmann; Uriarte, 2022).

Entre as principais tecnologias utilizadas atualmente destacam-se React, Vue.js e Next.js, que possuem ampla adoção tanto pela comunidade acadêmica quanto pelo mercado de desenvolvimento de *software* (React, 2025; Vue.js, 2025; Next.js, 2025). Embora compartilhem o objetivo de facilitar a construção de aplicações *web* modernas, essas tecnologias apresentam diferenças relacionadas à arquitetura, ao modelo de renderização, ao desempenho, à escalabilidade, ao ecossistema de bibliotecas e à facilidade de desenvolvimento. Essas características influenciam diretamente a escolha da tecnologia mais adequada para cada projeto (Siahaan; Kenidy, 2023; Lorandi, 2022).

A evolução constante do ecossistema JavaScript, associada à crescente demanda por aplicações web de alto desempenho, torna relevante a realização de estudos que permitam compreender as características, vantagens e limitações das diferentes tecnologias utilizadas nesse contexto. Embora existam pesquisas relacionadas ao desenvolvimento *front-end*, parte desses estudos concentra-se em aspectos específicos ou em uma única tecnologia, dificultando uma visão comparativa mais ampla entre diferentes soluções utilizadas no desenvolvimento web (Siahaan; Kenidy, 2023; Lorandi, 2022).

Diante desse contexto, este trabalho faz uma comparação entre React, Vue.js e Next.js, levando em conta aspectos técnicos e mercadológicos que são importantes para o desenvolvimento de aplicações *web* modernas. A partir dessa análise, busca-se identificar as principais características de cada tecnologia e fornecer informações que ajudem desenvolvedores, pesquisadores e organizações a escolher a tecnologia mais adequada às necessidades dos diferentes projetos.

Especificamente, o estudo busca definir critérios de comparação entre as tecnologias analisadas, avaliar a qualidade da documentação oficial, a curva de aprendizado e a configuração do ambiente de desenvolvimento, comparar aspectos relacionados ao desempenho e à otimização para mecanismos de busca *Search Engine Optimization* (SEO) e analisar sua adoção no mercado de trabalho, bem como a popularidade e o suporte oferecido pelas comunidades de desenvolvedores.

Para atingir esses objetivos, a pesquisa foi desenvolvida por meio de levantamento bibliográfico e análise comparativa, utilizando artigos científicos, trabalhos acadêmicos, livros, documentações oficiais e fontes especializadas como

fontes de informação. A partir desses materiais, foram organizados os critérios utilizados na análise das tecnologias, considerando aspectos técnicos e mercadológicos.

2 REFERENCIAL TEÓRICO

Estudos comparativos sobre tecnologias voltadas ao desenvolvimento *front-end*, como o de Ferreira (2018), mostram que a escolha de uma tecnologia deve considerar não apenas o desempenho, mas também aspectos como facilidade de uso, curva de aprendizado e suporte da comunidade. Nesse contexto, a escolha da solução utilizada no desenvolvimento *front-end* pode influenciar aspectos como manutenção, desempenho e escalabilidade das aplicações web. Este capítulo apresenta os fundamentos teóricos das tecnologias analisadas, React, Vue.js e Next.js, com ênfase em suas características, arquiteturas e formas de utilização no desenvolvimento *web*.

2.1 React

O React é uma biblioteca JavaScript voltada ao desenvolvimento de interfaces de usuário, criada pelo Facebook (atual Meta) e lançada em 2013 (React, 2025). Seu principal objetivo é facilitar a construção de Aplicações de Página Única (*Single Page Applications – SPAs*), permitindo uma navegação mais dinâmica ao evitar o carregamento completo das páginas. Para isso, adota uma arquitetura baseada em componentes reutilizáveis e independentes, característica que favorece a organização do código, a manutenção e a escalabilidade das aplicações (React, 2025).

Além da arquitetura baseada em componentes, o React utiliza o *Virtual DOM* para otimizar as atualizações da interface, reduzindo a quantidade de manipulações diretas no *DOM* e contribuindo para um melhor desempenho das aplicações. Sua flexibilidade permite o desenvolvimento de projetos de diferentes níveis de complexidade, embora apresente uma curva de aprendizado moderada em razão de conceitos como gerenciamento de estado (Zeeshan, 2025).

A ampla adoção do React também pode ser observada em aplicações desenvolvidas por grandes empresas de tecnologia, como Meta (Facebook e

Instagram), Netflix, Uber, Airbnb, Spotify, Hulu e Disney. O consumo de bibliotecas por organizações que atendem milhões de usuários demonstra sua capacidade de suportar aplicações de grande porte e diferentes segmentos do mercado, incluindo redes sociais, serviços de streaming, mobilidade e comércio eletrônico (Geeksforgeeks, 2025).

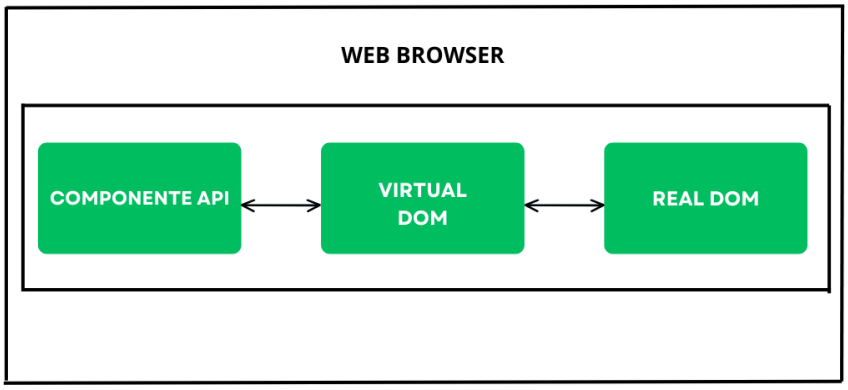
No cenário atual, o React mantém posição de destaque entre as principais tecnologias JavaScript utilizadas no desenvolvimento *front-end*. Dados de 2025 indicam participação de aproximadamente 42,62% entre os sites que utilizam alguma tecnologia ou *framework* JavaScript, correspondendo a cerca de 11,2 milhões de aplicações ativas. Em relação à adoção pela comunidade de desenvolvedores, pesquisas apontam índices entre 40% e 44%, evidenciando a consolidação da tecnologia tanto no mercado quanto entre os profissionais de desenvolvimento (Zeeshan, 2025).

2.1.1 Arquitetura do React

A arquitetura do React é baseada na organização da interface em componentes reutilizáveis e na atualização eficiente da interface por meio do Virtual *DOM*. A Figura 1 apresenta, de forma simplificada, o processo de atualização da interface quando ocorre uma alteração nos dados ou no estado de um componente.

Quando uma alteração é identificada, o React cria uma nova representação virtual da interface e a compara com a versão anterior. A partir dessa comparação, são identificadas apenas as partes que precisam ser atualizadas no *DOM* real, evitando alterações desnecessárias e contribuindo para a eficiência da aplicação (React, 2025).

Figura 1 - Arquitetura do react com virtual DOM



Fonte : Elaboração própria , 2025

A figura permite observar que o Virtual *DOM* funciona como uma representação intermediária entre os componentes da aplicação e o *DOM* real. Dessa forma, o processo de comparação das versões da interface determina quais elementos precisam ser atualizados no navegador, contribuindo para uma atualização mais eficiente da interface.

2.1.2 Ferramentas e ecossistemas do React

Dentre as principais ferramentas que complementam o desenvolvimento com React, destacam-se:

- Bit - Plataforma colaborativa para compartilhamento e reutilização de componentes React.
- Jest – *Framework* de testes unitários em JavaScript.
- Redux – Biblioteca para gerenciamento centralizado de estado da aplicação.
- React Developer Tools – Extensão para inspeção e depuração de componentes React.
- React Bootstrap – Implementação dos componentes do Bootstrap para uso nativo com React.
- Cypress – Ferramenta de testes end-to-end para aplicações *web*.
- WAI-ARIA – Conjunto de diretrizes para acessibilidade em aplicações ricas na *web*.

2.2 Vue.js

O Vue.js é um *framework* JavaScript progressivo e de código aberto, desenvolvido por Evan You e lançado em 2014 após sua experiência com o AngularJS no Google (Vue.js, 2025). Projetado para adoção incremental, o *framework* possibilita sua utilização tanto em pequenas funcionalidades quanto no desenvolvimento de aplicações completas, incluindo Aplicações de Página Única (*Single Page Applications – SPAs*), quando integrado a ferramentas e bibliotecas complementares (Vue.js, 2025).

Uma das principais características do Vue.js é sua simplicidade de utilização,

proporcionada pela curva de aprendizado reduzida, sintaxe intuitiva e documentação abrangente. Essas características facilitam sua adoção por novos desenvolvedores, bem como sua integração gradual em sistemas existentes. Dados recentes indicam elevado índice de aceitação pela comunidade, com aproximadamente 93% dos desenvolvedores demonstrando interesse em continuar utilizando a tecnologia (Vue School, 2025).

O uso do Vue.js também pode ser visto em empresas de vários setores como Netflix, Alibaba, Adobe, Trivago, GitLab, Nintendo, BMW, Xiaomi, Upwork Apple e Google. O uso do *framework* por essas empresas mostra que ele pode funcionar bem em aplicativos grandes ou pequenos. Ele faz isso usando uma estrutura que usa peças que podem ser usadas várias vezes. Isso ajuda a deixar o código mais organizado e melhora a produtividade no desenvolvimento (Trio, 2025).

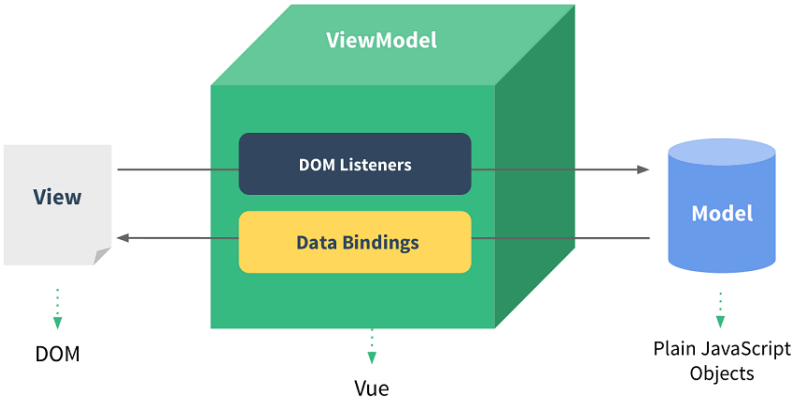
Atualmente o Vue.js está entre os *frameworks* JavaScript mais populares. Em 2025 essa tecnologia teve cerca de 6.4 milhões de downloads por semana no npm mostrando um grande aumento comparado aos anos anteriores e se tornando a segunda tecnologia mais popular depois do React. Além disso, pesquisas mostram que a maioria dos desenvolvedores disseram que escolheriam o Vue.js novamente para projetos futuros mostrando sua alta popularidade entre os desenvolvedores (Vue School,2025).

2.2.1 Arquitetura do Vue.js

O Vue.js utiliza um sistema reativo baseado em componentes, permitindo que alterações nos dados sejam refletidas automaticamente na interface. Sua estrutura também pode ser relacionada ao padrão Model-View-ViewModel (MVVM), no qual a camada responsável pela lógica de apresentação estabelece a comunicação entre os dados e a interface (Vue.js, 2025).

A Figura 2 apresenta essa organização, destacando a relação entre os dados, a camada de visualização e os mecanismos responsáveis pela atualização da interface.

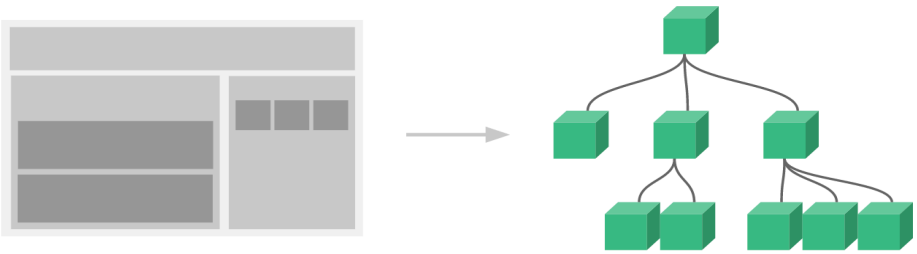
Figura 2 : Arquitetura MVVM do vue.js



Fonte : Vue.js, 2025.

Além dessa organização, as aplicações desenvolvidas com Vue.js são estruturadas a partir de componentes reutilizáveis. A Figura 3 apresenta essa estrutura em forma de árvore, permitindo visualizar a relação hierárquica entre os componentes que formam a aplicação.

Figura 3 : Estrutura em árvore de componentes no vue.js



Fonte : Vue.js , 2025 .

Essa organização favorece a modularidade e a reutilização de código, permitindo que diferentes partes da interface sejam desenvolvidas e mantidas de forma independente.

2.2.2 Ferramentas e ecossistemas do Vue.js

- Vue CLI – Interface de linha de comando oficial para criação e gerenciamento de projetos.
- Vue press – Gerador de sites estáticos baseado em Markdown.
- Vuex – Biblioteca oficial de gerenciamento de estado.

- Nuxt.js – Framework full-stack baseado em Vue.js.

2.3 Next.js

O Next.js é um *framework* de código aberto construído sobre o React, desenvolvido pela Vercel, que amplia os recursos disponíveis para a construção de aplicações *web*. Entre suas principais funcionalidades estão diferentes estratégias de renderização, como *Server-Side Rendering* (SSR) e *Static Site Generation* (SSG), além da geração incremental de páginas, otimização de imagens, *code splitting* e roteamento baseado em arquivos (Next.js, 2025).

Dessa forma, o Next.js utiliza o React como base para a construção das interfaces, mas acrescenta uma estrutura própria para o desenvolvimento de aplicações *web*, reunindo recursos que normalmente precisam ser configurados ou integrados separadamente em projetos React. Essa abordagem pode simplificar o desenvolvimento de aplicações que necessitam de recursos como renderização no servidor, geração estática, roteamento e otimizações de desempenho (Next.js, 2025).

Adicionalmente, empresas verificadas utilizam o Next.js globalmente, com uma presença significativa nos Estados Unidos e em setores como serviços empresariais, manufatura, software e varejo (Andy.G, 2025). Apple , Amazon , Samsung, Walmart e Microsoft adotam o Next.js devido ao seu alto desempenho, SEO eficaz, desenvolvimento simplificado e boa escalabilidade, o que resulta em aplicações mais rápidas e produtivas (Landbase, 2025).

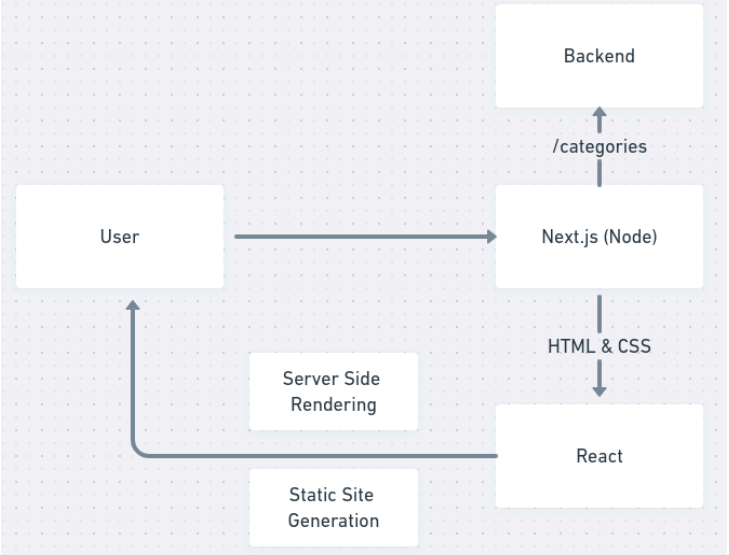
2.3.1 Arquitetura do Next.js

A arquitetura do Next.js amplia a estrutura do React ao oferecer recursos próprios para o desenvolvimento de aplicações *web*. Entre esses recursos estão diferentes estratégias de renderização, que permitem que o conteúdo seja gerado no servidor, no momento da construção da aplicação ou no cliente, conforme a configuração utilizada.

A Figura 4 apresenta uma visão da arquitetura de renderização do Next.js, destacando a relação entre a aplicação, o servidor e o navegador. Diferentemente de uma aplicação baseada exclusivamente em *Client-Side Rendering* (CSR),

determinadas páginas podem ter seu conteúdo gerado previamente e enviado ao navegador como HTML, permitindo que o usuário receba inicialmente uma página já estruturada.

Figura 4 : Arquitetura de renderização do Next.js



Fonte : OLIVEIRA , L . (2022)

Após o carregamento inicial, o React é responsável pela interatividade da interface do cliente. Dessa forma, o Next.js utiliza o React como base e acrescenta mecanismos que permitem diferentes estratégias de renderização e recursos adicionais para aplicações web (Next.js, 2025).

2.3.2 Ferramentas e ecossistemas do Next.js

- Zod – Biblioteca de validação de esquemas no *server-side*.
- Shadcn– Coleção de componentes acessíveis e customizáveis.
- Vercel – Plataforma de *deploy* otimizada para Next.js .
- ESLint – Ferramenta de *linting* integrada por padrão.
- Next.js CLI – Ferramenta oficial para criação e gerenciamento de projetos.
- Prisma – ORM moderno com excelente integração ao Next.js.
- Tailwind CSS – *Framework* CSS utilitário amplamente adotado.

3 METODOLOGIA

Este estudo utilizou uma metodologia de caráter bibliográfico, exploratório e comparativo, com enfoque qualitativo, apropriada para investigações teóricas no campo da Ciência da Computação. A opção por essa abordagem baseia-se nos princípios expostos por Wazlawick (2014), que enfatiza a pesquisa bibliográfica e os estudos comparativos como métodos adequados para analisar e avaliar tecnologias computacionais.

De acordo com Gil (2014), uma pesquisa bibliográfica envolve uma coleta sistemática de conhecimentos já estabelecidos por meio de livros, artigos científicos e publicações especializadas. Essa metodologia é frequentemente utilizada em pesquisas exploratórias que visam entender opiniões a partir de diversas perspectivas teóricas. No âmbito da Computação, estudos de caráter bibliográfico e comparativo são comumente usados para analisar *frameworks*, ferramentas e novas tecnologias.

A metodologia foi organizada em etapas sucessivas, envolvendo a definição das tecnologias analisadas, o levantamento bibliográfico, a seleção e organização das fontes e a sistematização dos critérios utilizados na comparação.

3.1 Definição das tecnologias voltadas para o projeto

A primeira fase da pesquisa envolveu a determinação das tecnologias JavaScript a serem analisadas. As tecnologias React, Vue.js e Next.js foram selecionadas com base em sua ampla utilização no desenvolvimento *front-end* atual, importância técnica e influência no mercado de trabalho.

A seleção dessas tecnologias levou em consideração fatores como maturidade tecnológica, aplicabilidade em projetos concretos, suporte da comunidade, disponibilidade de documentação oficial e presença no cenário nacional e internacional. Santos (2023) enfatiza que a escolha de tecnologias tem um impacto direto na produtividade das equipes, na padronização dos projetos e na qualidade final das aplicações *web*.

3.2 Elaboração do estudo comparativo teórico das tecnologias

O estudo comparativo foi realizado com base em critérios estabelecidos anteriormente, seguindo a metodologia proposta por Wazlawick (2014) para pesquisas em Computação.

A análise foi organizada em dois eixos principais: o eixo técnico e o eixo de mercado. No eixo técnico, são considerados elementos como a estrutura das tecnologias, modelos de renderização, mecanismos de reatividade, simplicidade na configuração do ambiente de desenvolvimento, curva de aprendizado, excelência da documentação oficial e ecossistema de bibliotecas. No eixo de mercado, foram escolhidos aspectos como a popularidade das tecnologias, envolvimento da comunidade, regularidade das atualizações, presença no mercado de trabalho e padrões de adoção.

Essa organização possibilitou uma análise coerente e imparcial, abordando tanto os aspectos técnicos quanto os práticos ligados ao uso de tecnologias no desenvolvimento *front-end*.

3.3 Levantamento bibliográfico e seleção das fontes

O levantamento bibliográfico foi realizado com o objetivo de identificar informações relacionadas às tecnologias analisadas e aos critérios utilizados na comparação. Foram consultados artigos científicos, trabalhos acadêmicos, livros técnicos, documentações oficiais, repositórios e publicações especializadas sobre desenvolvimento web.

Durante o levantamento, as fontes foram analisadas considerando sua relação com o objetivo da pesquisa e sua contribuição para a compreensão e comparação das tecnologias. Foram priorizados materiais que apresentassem informações relacionadas às características técnicas, modelos de renderização, desempenho, adoção, popularidade e aspectos mercadológicos de React, Vue.js e Next.js.

O levantamento também envolveu a consulta de diferentes tipos de fontes para aprofundamento do tema. A distribuição dos materiais consultados é apresentada na Tabela 1.

Tabela 1 – Distribuição das fontes consultadas durante o levantamento

Tipo de fonte	Quantidade	Finalidade
Artigos científicos e trabalhos acadêmicos	10	Fundamentação teórica e resultados de estudos comparativos
Livros	2	Fundamentação metodológica
Documentações, repositórios e fontes técnicas	9	Caracterização e informações técnicas das tecnologias
Fontes especializadas e de mercado	8	Informações sobre popularidade, adoção e mercado
Total	29	

Fonte: Elaborado pelo autor (2026).

Os critérios utilizados para a seleção das fontes são apresentados no Quadro 1.

Quadro 1 – Critérios utilizados no levantamento bibliográfico

Elemento	Critério adotado
Palavras-Chave	“React”, “Vue.js”, “Next.js”, “JavaScript”, “front-end”, “desempenho”, “renderização” e “desenvolvimento web”
Critérios de inclusão	Materiais relacionados às tecnologias estudadas
Critérios de exclusão	Materiais duplicados, sem relação direta com o tema
Seleção	Materiais considerados relevantes para a fundamentação e análise comparativa

Fonte: Elaborado pelo autor (2026).

Nem todas as fontes consultadas foram utilizadas como citações no artigo, sendo algumas empregadas apenas para aprofundamento teórico e compreensão das tecnologias. Assim, foram mantidas nas referências finais apenas as fontes efetivamente utilizadas.

3.3 Coleta e organização de dados secundários

A coleta de dados foi realizada utilizando fontes secundárias, configurando-se como uma pesquisa bibliográfica. Documentos oficiais das tecnologias em análise, artigos científicos, trabalhos acadêmicos, livros técnicos, relatórios de mercado e publicações especializadas no campo de desenvolvimento *web* são empregados. As informações coletadas foram coordenadas de maneira sistemática, em conformidade com os critérios definidos nos eixos técnicos e de mercado, o que permite a categorização e análise dos dados.

É importante destacar que, uma vez que essas tecnologias estão em constante evolução, parte dos dados referentes à popularidade e adoção de mercado (*downloads*, participação em sites, ofertas de emprego) foi coletada de plataformas especializadas e relatórios técnicos do setor, como NPM Trends, TanStack Stats, Stack Overflow Developer Survey e GitHub.Isso ocorre porque o setor de tecnologia é um campo em constante movimento, onde a literatura acadêmica muitas vezes fica para trás em relação ao que está atual (Crispiniano, 2021). Mesmo assim, sempre que foi viável, esses dados de mercado foram confrontados com estudos acadêmicos comparativos (Ferreira, 2018; Alves Pinto; Hoffmann; Uriarte, 2022; Lorandi, 2022; Patel; Sharma, 2023) para validar as tendências e assegurar uma fundamentação científica mais sólida à análise.

A escolha das fontes priorizou materiais recentes, com reconhecimento acadêmico e confiabilidade tanto na comunidade científica quanto no meio profissional. Esta fase fornece o suporte teórico e analítico necessário para executar a comparação sugerida.

Os dados organizados serviram como base para a realização da análise comparativa entre as tecnologias selecionadas, permitindo a discussão crítica de suas características técnicas e mercadológicas à luz dos objetivos propostos neste projeto.

3.4 Sistematização dos Critérios da Análise Comparativa

A sistematização dos critérios de análise comparativa constitui etapa fundamental da metodologia (Wazlawick, 2014), com o objetivo de definir parâmetros claros, objetivos e mensuráveis para a comparação entre React, Vue.js e Next.js.

3.4.1 Eixo técnico

O eixo técnico junta os critérios que têm a ver com as características funcionais e estruturais dos *frameworks* que foram analisados. Com esses critérios, foi possível olhar para aspectos que afetam diretamente o desenvolvimento de aplicações *web*, como desempenho, organização do código, facilidade de manutenção e produtividade do desenvolvedor. A análise técnica ajudou a ver as particularidades de cada *framework* e entender suas vantagens e limitações em diferentes situações de desenvolvimento.

3.4.2 Eixo Mercadológico

O eixo mercadológico traz critérios que têm a ver com a adoção e a importância dos *frameworks* no contexto atual do desenvolvimento de *software*. Nessa visão, foram olhados indicadores que mostram como eles são aceitos pela comunidade de desenvolvedores, sua presença no mercado de trabalho, evolução tecnológica e nível de suporte disponível. Esses aspectos ajudam a análise técnica ao mostrar como cada tecnologia está posicionada no mercado e sua aplicabilidade em projetos reais.

4 RESULTADOS

4.1 Análise Técnica das Tecnologias JavaScript

A análise técnica das tecnologias React, Vue.js e Next.js foi feita com base nos critérios que foram definidos na metodologia. Isso incluiu aspectos relacionados à arquitetura, modelos de renderização, otimização para mecanismos de busca (SEO),

curva de aprendizado, configuração do ambiente de desenvolvimento, escalabilidade e ecossistema de bibliotecas.

Os resultados obtidos ajudaram a identificar diferenças importantes entre as tecnologias analisadas. Isso é especialmente verdadeiro no que diz respeito às estratégias de renderização, facilidade de adoção e capacidade de expansão para aplicações maiores.

Quadro 2 – Comparação técnica entre React, Vue.js e Next.j

Critério	React.js	Vue.js	Next.js
Arquitetura	Baseada em componentes	MVVM + Componentes	Meta-framework
Renderização	CSR	CSR e SSR parcial	SSR e SSG nativos
SEO	Médio	Médio	Alto
Curva de aprendizado	Média	Baixa	Média
Configuração	Moderada	Simples	Simples
Escalabilidade	Alta	Alta	Muito alta
Ecossistema	Muito amplo	Amplo	Muito amplo

Fonte: Elaboração própria com base nas documentações oficiais do React, Vue.js e Next.js (2026).

Os dados do Quadro 2 mostram que o React tem uma arquitetura de componentes e um dos ecossistemas mais amplos do mercado, fatores que ajudam a sua utilização em aplicações complexas e muito escaláveis. A grande quantidade de bibliotecas disponíveis aumenta as possibilidades de desenvolvimento, embora a configuração inicial possa exigir mais conhecimento técnico quando comparada às outras tecnologias analisadas.

O Vue.js se destacou principalmente pela simplicidade de configuração e pela menor curva de aprendizado. A sua combinação entre o padrão MVVM e a arquitetura baseada em componentes ajuda na organização do código e diminui a complexidade de desenvolvimento, fazendo dele uma alternativa atrativa para equipes que buscam produtividade e rapidez na implementação de aplicações *web*.

O Next.js trouxe os resultados mais expressivos em aspectos relacionados à renderização e otimização para mecanismos de busca. O suporte nativo a *Server-Side Rendering (SSR)* e *Static Site Generation (SSG)* é uma vantagem significativa em aplicações que dependem de desempenho, indexação em mecanismos de busca e carregamento eficiente de páginas. Além disso, sua alta escalabilidade o torna uma opção adequada para projetos corporativos e aplicações com grande volume de acessos.

Os resultados demonstram diferenças entre as tecnologias JavaScript analisadas, principalmente nos aspectos arquiteturais, estratégias de renderização e organização do desenvolvimento.

4.1.1 Performance e Renderização

Além das características apresentadas na comparação técnica, a performance das tecnologias JavaScript está diretamente relacionada às estratégias utilizadas para renderizar as aplicações. Esse fator influencia o tempo de carregamento, a atualização da interface e a experiência do usuário, sendo um dos principais critérios considerados na avaliação de tecnologias para desenvolvimento *front-end* (Diniz, 2022; Siahaan; Kenidy, 2023).

Nos testes analisados, o React destacou-se nas métricas relacionadas à interatividade da aplicação. De acordo com Diniz (2022), a tecnologia teve o menor *Time to Interactive (TTI)* ,métrica que representa o tempo necessário para que a

página esteja completamente carregada e apta à interação do usuário, registrando média de 300 ms. No entanto, Hoffmann, Pinto e Uriarte (2022),notaram que o tempo de renderização aumenta à medida que a quantidade de elementos processados cresce, variando de 2,83 ms para 10 elementos até 7.185,03 ms para 100.000 elementos.Esses resultados indicam que, embora o React ofereça rápida interatividade, o tempo de processamento tende a aumentar em aplicações com maior volume de componentes.

Entre as tecnologias comparadas, o Vue.js obteve os melhores resultados de desempenho. Segundo Diniz (2022), o *framework* foi cerca de 758% mais rápido que o React e 595% mais rápido que o Angular na manipulação do *Document Object Model (DOM)*, além de ter o menor tamanho de *bundle* ,correspondente ao conjunto de arquivos JavaScript gerados para a aplicação e enviados ao navegador. Esses resultados também foram observados por Hoffmann, Pinto e Uriarte (2022), cujos testes registraram tempos de renderização entre 1,70 ms e 3.257,3 ms, menores do que os obtidos pelo React em todos os cenários analisados. Além disso, Siahaan e Kenidy (2023) observaram uma média de 0,9 s no *WebPageTest,*ferramenta utilizada para avaliação do desempenho de aplicações web , reforçando a eficiência do *framework* no carregamento inicial.

O Next.js destacou-se principalmente nas métricas de carregamento inicial das páginas. De acordo com Siahaan e Kenidy (2023), o *framework* apresentou uma média de 1,5 s no *WebPageTest*, 3,9 s no *PageSpeed Insights* e 0,8 s no *GTmetrix,* ferramentas de análise de desempenho amplamente utilizadas na avaliação de aplicações *web.* Esses resultados sugerem que as estratégias de renderização do Next.js contribuem para a entrega inicial do conteúdo.

Dessa forma, os resultados demonstram que as tecnologias apresentam elevado desempenho, porém com vantagens distintas em cada métrica analisada. Conforme analisado, o React destacou-se pelo menor *Time to Interactive (TTI)*, o Vue.js apresentou melhor desempenho na renderização e manipulação do *DOM*, enquanto o Next.js obteve melhores resultados nas métricas relacionadas ao carregamento inicial. Assim, a escolha da tecnologia deve considerar os requisitos específicos de cada aplicação (Diniz, 2022; Siahaan; Kenidy, 2023).

4.1.2 Curva de aprendizagem e Configuração

Dessa forma, os resultados indicam que React e Vue.js apresentam desempenho semelhante na renderização de interfaces, enquanto o Next.js se destaca quando há necessidade de otimizar o carregamento inicial das páginas e melhorar a indexação em mecanismos de busca, principalmente devido às estratégias de renderização disponibilizadas pelo *framework* (Diniz, 2022; Siahaan; Kenidy, 2023; Next.js, 2025).

O Vue.js apresentou a menor curva de aprendizagem entre as tecnologias analisadas. Sua arquitetura progressiva permite que o desenvolvedor utilize inicialmente apenas funcionalidades básicas, incorporando novos recursos conforme a complexidade do projeto aumenta (Vue.js, 2025). Segundo Lorandi (2022), essa característica facilita a adoção da tecnologia principalmente por desenvolvedores iniciantes. Além disso, dados apresentados pelo Vue School (2025) mostram que aproximadamente 93% dos desenvolvedores pretendem continuar utilizando o Vue.js, indicando elevado nível de satisfação com a experiência de desenvolvimento.

O React possui uma curva de aprendizagem intermediária. Embora sua arquitetura baseada em componentes favoreça a reutilização do código, o desenvolvimento de aplicações completas normalmente exige conhecimentos adicionais sobre gerenciamento de estado, roteamento e bibliotecas complementares (Martins Filho, 2023). De acordo com a documentação oficial, essa flexibilidade permite maior adaptação a diferentes cenários de desenvolvimento, porém aumenta a quantidade de conceitos que precisam ser aprendidos pelo desenvolvedor (Meta, 2025).

O Next.js apresenta uma curva de aprendizagem superior à do React por incorporar recursos adicionais, roteamento baseado em arquivos e otimizações automáticas (Next.js, 2025). Embora essas funcionalidades aumentem a complexidade inicial, elas reduzem significativamente a necessidade de configurações manuais durante o desenvolvimento de aplicações maiores.

Assim, os resultados demonstram que o Vue.js favorece uma adoção mais rápida, enquanto React e Next.js oferecem maior flexibilidade para projetos que demandam arquiteturas mais robustas e recursos avançados de desenvolvimento (Ferreira, 2018; Martins Filho, 2023; Next.js, 2025).

4.1.3 Ecossistema e Integração

O ecossistema foi analisado considerando a disponibilidade de bibliotecas, ferramentas complementares, documentação oficial e integração com outras tecnologias utilizadas no desenvolvimento *web*. Esses fatores influenciam diretamente a produtividade das equipes e a evolução das aplicações ao longo do tempo (Alves Pinto; Hoffmann; Uriarte, 2022).

O React apresenta o ecossistema mais amplo entre as tecnologias analisadas, resultado da sua elevada adoção pela comunidade e pelo mercado de desenvolvimento de software. A grande disponibilidade de bibliotecas para gerenciamento de estado, roteamento, interfaces e comunicação com APIs proporciona maior flexibilidade para atender diferentes tipos de projetos (Meta, 2025). Essa característica também favorece sua utilização em aplicações corporativas de grande porte (Martins Filho, 2023).

O Vue.js possui um ecossistema mais centralizado, disponibilizando oficialmente ferramentas como Vue Router e Pinia, o que reduz a dependência de bibliotecas externas e facilita a integração entre os diferentes recursos do *framework* (Vue.js, 2025). Segundo Lorandi (2022), essa padronização simplifica o processo de desenvolvimento e manutenção das aplicações.

O Next.js utiliza todo o ecossistema React e adiciona recursos próprios voltados para aplicações modernas, incluindo otimização automática de imagens, divisão automática de código (*code splitting*), componentes de servidor e diferentes estratégias de renderização (Next.js, 2025). Essas funcionalidades permitem maior controle sobre desempenho, organização da aplicação e experiência do usuário, reduzindo a necessidade de configurações adicionais.

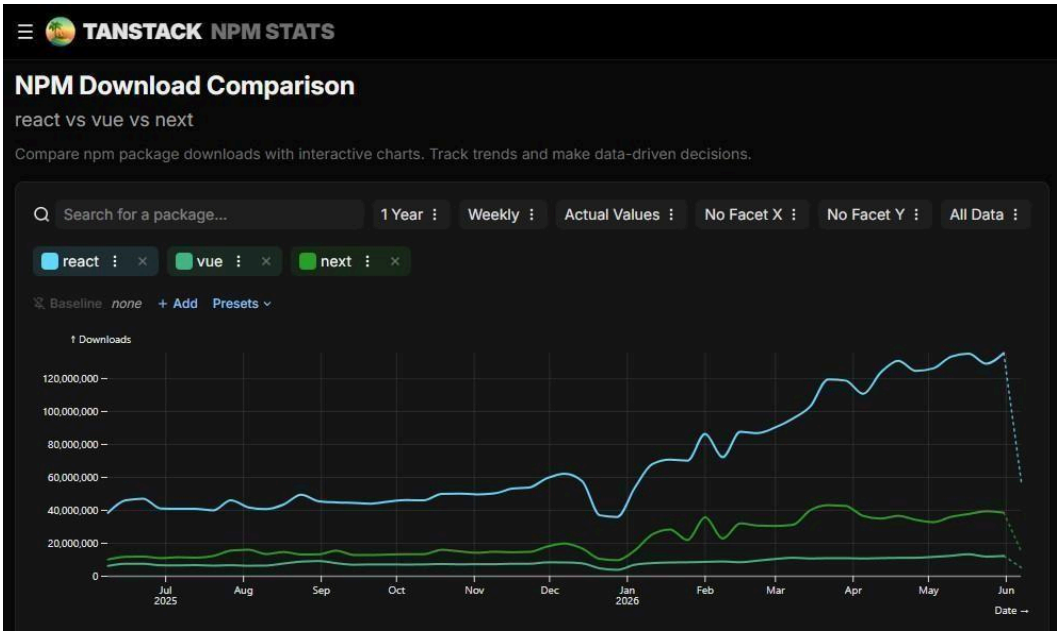
Portanto, os resultados mostram que o React se destaca pela maturidade e diversidade do seu ecossistema; o Vue.js pela simplicidade de integração e organização das ferramentas oficiais; e o Next.js pela combinação do ecossistema React com recursos nativos voltados à otimização e escalabilidade de aplicações modernas (Alves Pinto; Hoffmann; Uriarte, 2022; Next.js, 2025).

4.2 Análise Mercadológica das Tecnologias

Além da análise técnica, foram avaliados indicadores mercadológicos relacionados à popularidade, adoção, atividade da comunidade e presença profissional das tecnologias analisadas. Para isso, foram utilizados dados de plataformas como npm Trends, GitHub, Stack Overflow Developer Survey e da Pesquisa Salarial de Programadores 2026, realizada pelo Código Fonte TV.

A Figura 5 apresenta a evolução dos downloads de React, Vue.js e Next.js. O React apresentou maior volume de downloads durante o período analisado, alcançando aproximadamente 3,6 bilhões de downloads acumulados e cerca de 135 milhões na última semana observada. O Vue.js registrou aproximadamente 443 milhões de downloads acumulados e 12 milhões na última semana, enquanto o Next.js apresentou crescimento expressivo ao longo do período analisado. Esses dados indicam diferentes níveis de adoção e utilização dentro do ecossistema JavaScript.

Figura 5 – Evolução dos *downloads* das tecnologias React, Vue.js e Next.js

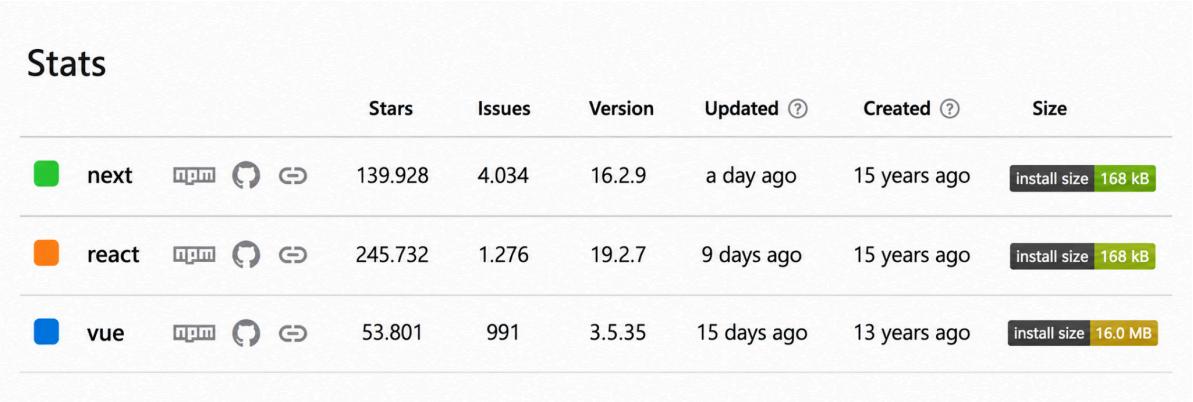


Fonte: TanStack Stats (2026)

Os dados apresentados na Figura 5 demonstram que o React possui maior volume de utilização no período analisado. O Next.js, por sua vez, apresentou crescimento ao longo do período, enquanto o Vue.js manteve uma utilização relevante, embora inferior aos valores observados para React e Next.js. Esses resultados contribuem para identificar o nível de adoção das tecnologias pela comunidade de desenvolvimento.

A Figura 6 apresenta indicadores relacionados aos repositórios oficiais das tecnologias analisadas no GitHub. O React apresentou aproximadamente 245 mil *Stars*, enquanto o Next.js registrou cerca de 140 mil. O indicador *Stars* representa usuários que demonstraram interesse ou consideraram relevante determinado repositório.

Figura 6 – Indicadores dos repositórios oficiais das tecnologias analisadas no GitHub .

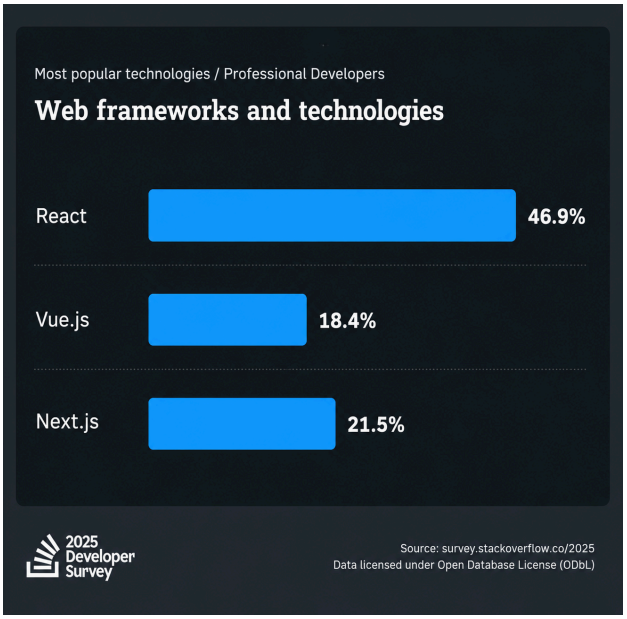


Fonte: GitHub e Npm Trends (2026).

A quantidade de *Stars* reforça a expressiva participação da comunidade no desenvolvimento e acompanhamento dos projetos. Entretanto, esse indicador deve ser considerado de forma complementar, pois não representa, isoladamente, a quantidade de usuários ou o nível de utilização de uma tecnologia.

Outro indicador de adoção profissional é apresentado pela Stack Overflow Developer Survey de 2025, conforme a Figura 7. Esses resultados demonstram a maior presença do React entre os profissionais pesquisados, ao mesmo tempo em que evidenciam a relevância de Vue.js e Next.js no desenvolvimento web.

Figura 7 - Popularidade das tecnologias JavaScript entre desenvolvedores profissionais



Fonte: Stack Overflow Developer Survey (2025).

Os dados da Figura 7 complementam os indicadores de downloads e de participação da comunidade, permitindo observar a adoção das tecnologias sob uma perspectiva profissional. Embora o React apresente maior percentual de utilização, Vue.js e Next.js também apresentam participação significativa entre os desenvolvedores analisados.

Outro parâmetro foi obtido na Pesquisa Salarial de Programadores 2026, realizada pelo Código Fonte TV com 17.046 participantes. Na categoria “Frameworks / Ferramentas”, React ocupou a 3ª posição, com média salarial de R$ 10.479,97, Vue.js a 14ª, com R$ 9.040,18, e Next.js a 11ª, com R$ 10.503,69 (Código Fonte TV, 2026).

Tabela 2 – Indicadores mercadológicos das tecnologias analisadas

Tecnologia	Utilização profissional (2025)	Média salarial (2026)	Participantes
React	46,9%	R$ 10.479,97	1.291
Vue.js	18,4%	R$ 9.040,18	252
Next.js	21,5%	R$ 10.503,69	468

Fonte: Elaborado pelo autor com base Código Fonte TV (2026).

Os resultados apresentados na Tabela 2 mostram maior presença do React entre os profissionais participantes, enquanto Next.js apresentou média salarial ligeiramente superior à do React e Vue.js apresentou o menor valor entre as três tecnologias. Entretanto, os dados salariais devem ser interpretados como indicadores de mercado, pois a remuneração também é influenciada por fatores como experiência, nível profissional e modelo de contratação.

Em conjunto, os indicadores mercadológicos reforçam a relevância das três tecnologias, com o React apresentando maior presença profissional, o Vue.js mantendo participação no mercado e o Next.js demonstrando crescimento e média salarial próxima à do React. Esses resultados complementam a análise técnica e contribuem para uma compreensão mais ampla do posicionamento das tecnologias no desenvolvimento *web*.

4.2 Análise Experimental do Desempenho

Para complementar a análise teórica e mercadológica, foram realizados testes práticos com aplicações equivalentes desenvolvidas em React, Vue.js e Next.js. As aplicações utilizaram o mesmo catálogo de produtos e foram avaliadas em suas versões de produção por meio do Lighthouse, utilizando a configuração para dispositivos móveis**.**

Cada tecnologia foi submetida a três execuções, sendo calculada a média dos resultados. Foram analisadas as métricas *First Contentful Paint (FCP)*, que indica o tempo até o primeiro conteúdo ser exibido na tela; *Largest Contentful Paint (LCP)*, que representa o tempo necessário para carregar o maior elemento de conteúdo visível; *Total Blocking Time (TBT)*, que mede o tempo em que a página permanece bloqueada para responder às interações do usuário; e *Speed Index*, que indica a velocidade com que o conteúdo visual da página é apresentado durante o carregamento.

Tabela 3 – Resultados médios dos testes de desempenho

Tecnologia	FCP	LCP	TBT	Speed Index
React	1,30 s	1,30 s	70 ms	1,30 s
Vue.js	1,10 s	1,10 s	0 ms	1,10 s
Next.js	0,90 s	1,40 s	470 ms	0,90 s

Fonte: Elaborado pelo autor com base nos testes realizados (2026).

Os resultados apresentados na Tabela 3 mostram diferenças entre as tecnologias. O Next.js apresentou os menores valores de FCP e *Speed Index*, ambos de 0,90 s. Por outro lado, o Vue.js apresentou o menor LCP, com 1,10 s, e o menor TBT, com 0 ms. O React apresentou valores intermediários nas métricas analisadas.

Dessa forma, os testes indicam que não houve uma tecnologia superior em todas as métricas.Cada solução apresentou características diferentes de desempenho na aplicação desenvolvida, reforçando que a escolha da tecnologia deve considerar os requisitos específicos de cada projeto.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma análise comparativa entre as tecnologias JavaScript React, Vue.js e Next.js usados no desenvolvimento front-end, levando em conta critérios técnicos e de mercado sobre a adoção dessas tecnologias.

A partir da revisão bibliográfica e da sistematização dos critérios de comparação, foi possível ver que a escolha de uma tecnologia JavaScript depende das características e necessidades de cada projeto, não sendo possível definir uma tecnologia como melhor em todos os casos.

A análise técnica mostrou diferenças importantes entre as tecnologias testadas. O React teve mais flexibilidade e maturidade no ecossistema; o Vue.js trouxe uma abordagem focada na simplicidade e produtividade; enquanto o Next.js trouxe recursos voltados para aplicações modernas que precisam de estratégias avançadas de renderização.

A análise do mercado confirmou que fatores como comunidade, uso e crescimento também afetam a escolha dessas tecnologias. Os dados coletados em

plataformas de desenvolvimento mostraram diferentes níveis de consolidação entre as tecnologias analisadas.

Assim, o estudo ajuda a organizar informações técnicas e mercadológicas que podem ajudar desenvolvedores na escolha de tecnologias *front-end* conforme o contexto do projeto, pensando em requisitos como desempenho, escalabilidade, manutenção e experiência da equipe.

REFERÊNCIAS

ALVES PINTO, L.; HOFFMANN, S.; URIARTE, L. R. **Análise comparativa entre as tecnologias de front-end React, Angular e Vue.** Revista de Extensão e Iniciação Científica da UNISOCIESC, 2022. Disponível em: https://dalfovo.com/ojs/index.php/reis/article/view/398. Acesso em: 9 dez. 2025.

ANDY, G. **Next.js in August 2025**: The React Framework That Definitively Won the Modern Web. Medium, 2025. Disponível em: https://medium.com/@andy.a.g/next-js-in-august-2025-the-react-framework-that-definitively-won-the-modern-web-fc37935e3919. Acesso em: 10 jan. 2026.

CRISPINIANO, A. G. **Estudo comparativo entre frameworks frontend para a criação de um progressive web app (PWA)**. Campina Grande, 2021. Disponível em: https://dspace.sti.ufcg.edu.br/handle/riufcg/24999. Acesso em: 9 dez. 2025.

DINIZ. **Evaluating the performance of web rendering technologies based on JavaScript: Angular, React, and Vue.** In: XVLIII Latin American Computer Conference (CLEI), 2022. [S. l.: s. n.], 2022. p. 1–9. ISSN: 2771-5752. Acesso em: 14 jul. 2026.

FERREIRA, H. K. **Análise comparativa entre frameworks frontend baseados em JavaScript para aplicações web**. Revista Interface Tecnológica, Taquaritinga, v. 15, n. 2, p. 111–123, 2018. DOI: 10.31510/infa.v15i2.502. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/502. Acesso em: 9 dez. 2025.

GEEKSFORGEEKS. **Companies that use React**. 2025. Disponível em: https://www.geeksforgeeks.org/reactjs/companies-that-use-react/. Acesso em: 10 jan. 2026.

GIL, A. C. **Métodos e técnicas de pesquisa social**. São Paulo: Atlas, 2014.

GITHUB. **React repository**. Disponível em: https://github.com/facebook/react. Acesso em: 30 jun. 2026.

GITHUB. **Vue.js repository**. Disponível em: https://github.com/vuejs/core. Acesso em: 30 jun. 2026.

LANDBASE. **Companies using Next.js in 2025**.Disponível em: https://data.landbase.com/technology/next-js/. Acesso em: 10 jan. 2026.

LORANDI, G. B. **Estudo comparativo entre Angular, Vue.js e React.js para desenvolvimento de aplicações web front-end**. Curitiba, 2022. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/37663. Acesso em: 9 dez. 2025.

MARTINS FILHO, F. R. F. **Comparação de frameworks JavaScript para desenvolvimento web front-end: React e Vue.js**. Fortaleza, 2023. Disponível em:

http://repositorio.ufc.br/handle/riufc/75925. Acesso em: 9 dez. 2025.

META. **React Documentation**. 2025. Disponível em: https://react.dev/. Acesso em: 30 jun. 2026.

NEXT.JS. **Next.js Documentation**. 2025. Disponível em: https://nextjs.org/docs. Acesso em: 30 jun. 2026.

NPM. npm Trends: **Compare package downloads**. Disponível em: https://npmtrends.com/. Acesso em: 30 jun. 2026.

PAGEPRO. **Stack Overflow 2025**: React e Next.js crescem em uso, mas perdem admiração. 2025. Disponível em: https://pagepro.co/blog/react-tldr/stack-overflow-2025-react-and-next-js-grow-in-use-drop-in-admiration/. Acesso em: 10 jan. 2026.

PATEL, R.; SHARMA, N. **Comparative Analysis of Angular, React, and Vue.js for SPA Development**. International Journal of Science and Research (IJSR), v. 12, n. 6, 2023. Disponível em: https://www.ijsr.net/archive/v12i6/SR24401230015.pdf. Acesso em: 1 jul. 2026.

SANTOS, G. N. **Estudo comparativo entre ferramentas de inicialização de projetos web**. Campina Grande, 2023. Disponível em: https://dspace.sti.ufcg.edu.br/handle/riufcg/34590. Acesso em: 9 dez. 2025.

SIAHAAN, M.; KENIDY, R. **Comparação do desempenho de renderização de React, Vue, Next e Nuxt**. Jurnal Mantik, v. 7, n. 3, p. 1851–1860, 2023. DOI: 10.35335/mantik.v7i3.4242. Disponível em: https://iocscience.org/ejournal/index.php/mantik/article/view/4242. Acesso em: 2 dez. 2025.

STACK OVERFLOW. **Developer Survey 2025**. Disponível em: https://survey.stackoverflow.co/2025/. Acesso em: 30 jun. 2026.

TANSTACK. **npm Package Statistics**. Disponível em: https://tanstack.com/stats/npm. Acesso em: 30 jun. 2026.

TRIO. **Top 15 real-world websites using Vue.js in 2025**. 2025. Disponível em: https://trio.dev/websites-using-vue/. Acesso em: 10 jan. 2026.

VERCEL. **Next.js GitHub Repository**. Disponível em: https://github.com/vercel/next.js. Acesso em: 30 jun. 2026.

VUE.JS. **Vue.js Documentation**. 2025. Disponível em: https://vuejs.org/. Acesso em: 30 jun. 2026.

VUE SCHOOL. **Vue.js – 2025 in review and a peek into 2026**. 2025. Disponível em: https://vueschool.io/articles/news/vue-js-2025-in-review-and-a-peek-into-2026/. Acesso em: 10 jan. 2026.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. 2. ed. Rio de Janeiro: Elsevier, 2014.

ZEESHAN, A. **Escopo e exposição do mercado ReactJS em 2025**. P2P Clouds, 2025. Disponível em: https://zeeshan.p2pclouds.net/blogs/reactjs-market-scope-and-exposure/. Acesso em: 10 jan. 2026.