UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE ENGENHARIA DE COMPUTAÇÃO

FRANCISCO RUBENS FÉLIX MARTINS FILHO

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

QUIXADÁ

2023

FRANCISCO RUBENS FÉLIX MARTINS FILHO

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

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Egenheria de Computação.

Orientador: Prof. Dr.Jefferson de Carvalho Silva.

QUIXADÁ

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M343a Martins Filho, Francisco Rubens Félix.
Análise comparativa de tecnologias javascript focadas no front-end para desenvolvimento web / Francisco Rubens Félix Martins Filho. – 2023.
40 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Computação, Quixadá, 2023.
Orientação: Prof. Dr. Jefferson De Carvalho Silva.

1. JavaScript (Linguagem de programação de computador). 2. Framework (Arquivo de computador). 3. React (JavaScript). 4. Next.js.. 5. Vue.js. I. Título.

CDD 621.39

FRANCISCO RUBENS FÉLIX MARTINS FILHO

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

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Egenheria de Computação.

Aprovada em: 08/12/2023.

BANCA EXAMINADORA

Prof. Dr.Jefferson de Carvalho Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcio Espíndola Freire Maia
Universidade Federal do Ceará (UFC)

Prof. Dr. Victor Aguiar Evangelista de Farias
Universidade Federal do Ceará (UFC)

À minha mãe e pai, tudo são por vocês.

AGRADECIMENTOS

Em primeiro lugar, agradeço à mim mesmo, por nunca ter desistido, até quando eu achava que tudo estava perdido.

À minha mãe, Aline Maria Teixeira Castro, por sempre ter me dado uma boa eduçação e bons ensinamentos, esses que levo para minha vida inteira. E ao meu pai, que sempre esteve comigo em todos os momentos da minha vida.

À minha namorada, Marília, que esteve comigo durante uma boa parte da faculdade, principalmente durante o tempo em que eu estava fazendo o TCC, pela companhia e companheirismo até nos piores momentos.

Ao Prof. Dr. Jefferson de Carvalho Silva, pela excelente orientação.

"Tudo tem suas maravilhas, até a escuridão e o silêncio. E eu aprendi que, seja qual for o estado que eu me encontre, nele estarei contente." (Hellen Keller)

RESUMO

Com o passar do tempo, a tecnologia sobretudo às aplicações web estão sendo cada vez mais exigidas no âmbito empresarial, para atender as constantes demandas, elas estão se tornando cada vez mais complexas, exigindo cada vez mais recursos, então elas precisam ser optimizadas. Os frameworks foram criados exatamente para atender essa crescente demanda. Hoje em dia, já existem várias tecnologias/frameworks que lidam com os constantes desafios na web, onde cada um possui sua particularidade

Este trabalho apresenta uma análise comparativa das tecnologias JavaScript React, Vue e Next no contexto do desenvolvimento web front-end. O projeto consistiu na criação de uma Pokedex utilizando cada uma dessas tecnologias. Além disso, foi conduzida uma pesquisa de campo com desenvolvedores experientes no mercado.

A análise das tecnologias abrangeu diversos aspectos, incluindo a documentação disponível, a configuração do ambiente de desenvolvimento, o suporte oferecido, o tamanho e a atividade das comunidades de desenvolvedores, a curva de aprendizagem e as perspectivas no mercado de trabalho.

Os resultados desta pesquisa fornecem uma visão abrangente das características distintas de React, Vue e Next, permitindo que desenvolvedores e empresas tomem decisões informadas ao escolher a tecnologia mais adequada para seus projetos front-end.

Palavras-chave: javascript (linguagem universal da web); frameworks (arquivo de computador); react (biblioteca javascript); next e vue (tipos de frameworks).

ABSTRACT

Over time, technology, especially web applications, has become increasingly crucial in the business sphere. To meet the ever-growing demands, these applications are becoming more complex and resource-intensive, necessitating optimization.

Frameworks were developed precisely to address this escalating demand. Nowadays, several technologies/frameworks handle the ongoing challenges in web development, each with its own unique characteristics.

This study presents a comparative analysis of JavaScript technologies React, Vue, and Next within the context of front-end web development. The project involved the creation of a Pokedex using each of these technologies. Additionally, field research was conducted with experienced developers in the industry.

The analysis of these technologies encompassed various aspects, including available documentation, development environment configuration, support, the size and activity of developer communities, the learning curve, and job market prospects.

The findings of this research offer a comprehensive understanding of the distinctive features of React, Vue, and Next, enabling developers and businesses to make informed decisions when selecting the most suitable technology for their front-end projects.

Keywords: javascript (universal language of the web); frameworks (computer file); react (javascript library); next and vue (types of frameworks).

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos dias de hoje, com o avanço do desenvolvimento de tecnologias para sistemas WEB, *frameworks* que utilizam o Javascript como linguagem tem se tornado bastante populares, além de serem desenvolvidos para ajudar os desenvolvedores à codificar sempre no mais alto nível e rapidez desejada (GIZAS, 2012). Com base nas palavras desse autor, alguns fatores que levam a maioria esmagadora de desenvolvedores *frontend* à escolherem *frameworks* diversos, são as constantes atualizações e a comunidade ser bem ativa, ou seja, possuem um suporte desejado ao desenvolvedor que está iniciando, por exemplo. Além disso, cada *framework* tem seu grau de importância e particularidade (DUARTE, 2015). A especificação de qual usar em um projeto, vai depender da necessidade dele, bem como prover um código de extrema qualidade (NOVAC *et al.*, 2021).

No entanto, com essa grande variedade de *frameworks*, a escolha tende a se tornar um desafio (ZAKAS, 2010). É importante escolher um *framework* que seja adequado para o tipo de projeto que se pretende desenvolver e que ofereça recursos e funcionalidades que atendam às necessidades das entidades envolvidas no projeto (FARLAND, 2011).

As aplicações atuais estão cada vez mais exigindo poderio de processamento. Portanto, o desempenho do JavaScript do navegador é um dos problemas de usabilidade de desenvolvimento mais importante. Além de escolher um *framework* que seja adequado para o tipo de projeto que se pretende desenvolver e, que ofereça recursos e funcionalidades que atendam às necessidades específicas do desenvolvedor e do cliente. (DINIZ, 2022).

Nesse contexto, esta pesquisa tem como objetivo realizar uma análise comparativa entre diferentes *frameworks* JavaScript focados no *frontend* para o desenvolvimento web. Ao comparar esses *frameworks*, pretendo fornecer uma visão geral sobre suas principais características, vantagens e desvantagens, a fim de auxiliar os desenvolvedores na escolha do *framework* mais adequado para o seu projeto.

Serão analisados as tecnologias mais populares no mercado, como React e Vue.js, além de outro *framework* emergente que está ganhando espaço na comunidade de desenvolvimento, como o Next.js. Serão considerados critérios como facilidade de uso, desempenho, escalabilidade, comunidade de desenvolvedores, documentação, entre outros fatores relevantes para a escolha do *framework* ideal.

2 OBJETIVOS

O objetivo geral deste trabalho é gerar um comparativo de tecnologias Javascript que atualmente são usadas nos mais variados âmbitos do desenvolvimento Frontend WEB, com base em aspectos essenciais que podem ser usados de escolha para cada *framework*.

2.1 Objetivos Específicos

1. Entender qual tecnologia focada no frontend mais utilizada na comunidade
2. Comparar as tecnologias a nível de desempenho de tempo de compilação, tempo de renderização de componentes, tempo de build, tempo de execução de uma página que recebe uma grande quantidade de dados providos de uma API (PokeAPI).
3. Compreender e comparar as tecnologias a nível de testes, gerenciamento de estados e curva de aprendizagem.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão apresentados os principais conceitos necessários para o entendimento e desenvolvimento do projeto proposto neste trabalho.

3.1 *JavaScript*

JavaScript é a linguagem universal de programação da web, que foi criado em 1995 por Brendan Eich em apenas 10 dias enquanto o mesmo trabalhava na Netscape. A maioria, se não, todos os sites modernos usam *JavaScript*, e todos os navegadores, em computadores, celulares, incluem interpretadores *JavaScript* (FARLAND, 2011).

Normalmente, é implementado junto com *HTML* (HyperText Markup Language) e *CSS3*(Cascading Style Sheets) em client-side (lado do cliente) para criar tecnologias, sobretudo aplicações web interativas. Ao logo do tempo, houve uma grande mudança na forma como se desenvolver software, sobretudo com o surgimento do JSF (JavaScript Frameworks) (MARIANO, 2017)

3.1.1 Frameworks

Um *framework* é um conjunto de ferramentas, bibliotecas e componentes que fornecem uma estrutura para o desenvolvimento de software. Ele oferece recursos e abstrações que ajudam os desenvolvedores a criar aplicações de forma eficiente, produtiva e organizada (FERREIRA, 2018).

Ele descreve como uma aplicação deve ser construída, permitindo que o código fique mais organizado, reforçando a escalabilidade e flexibilidade de uma aplicação. Ele oferece a descrição de uma arquitetura, porém, é ainda mais que um template, possui construtores etc. Atualmente, existem *frameworks* tanto para frontend quanto para o backend, onde diversos fatores acabam influenciando a decisão de desenvolvedores no quesito escolha (GIZAS, 2012).

3.1.2 React.js

React é uma biblioteca *JavaScript* para construção de interfaces de usuário (UI), desenvolvida pelo Facebook. Embora muitas vezes seja chamado de "framework", tecnicamente, o React é uma biblioteca, pois não possui uma estrutura completa de aplicação com todas as

funcionalidades necessárias para construir um aplicativo completo (JAVEED, 2019). Em vez disso, ele se concentra principalmente na camada de visualização, permitindo que os desenvolvedores criem componentes reutilizáveis que gerenciam seu próprio estado e são compostos para construir interfaces de usuário complexas (REACT.JS, 2023).

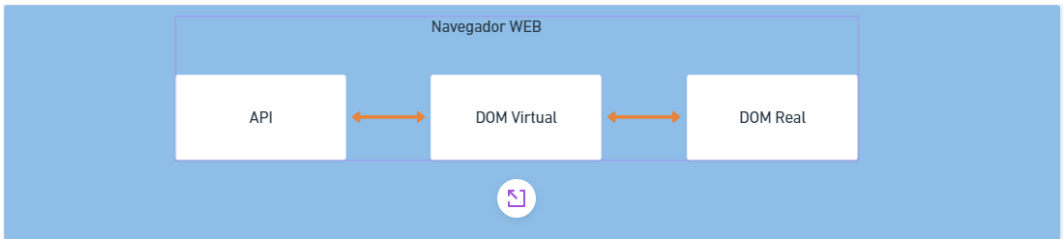
3.1.2.1 Arquitetura

O *React*, diferentemente do restante dos *frameworks* possui características próprias. Ele permite que o desenvolvedor faça alterações em seus dados, mas sem atualizações de páginas subsequentes, definindo assim sua arquitetura principal como VIEW, do MVC(Model - View - Controller) (DINIZ, 2022). Essa biblioteca cria representações abstratas de *Views*, se dividindo em partes da visão dos componentes (REACT.JS, 2023)

Possui por padrão o tipo de renderização *SSG*(Static Site Generation), porém, com a ajuda de alguns *frameworks*, consegue criar sites *SSR*(Server-side rendering). A arquiterura do React possui três camadas: API(Application Programming Interface), módulo de gerenciamento de estado de componente de memória (o famoso DOM ou Object Model) e módulo de renderização de elemento DOM real (virtual) (DINIZ, 2022).

Manipular o DOM é uma operação custosa e precisa ser minimizada. Entretanto, manipular o DOM manualmente gerará uma quantidade de código verboso e muitas vezes repetitivo, o que no desenvolvimento é um grande erro comum. A tecnologia React soluciona isso, dando ao desenvolvedor o poder de desenvolver um DOM virtual ao invés do real DOM. É como se, à grosso modo, o react encontrasse a diferença entre o DOM real e o virtual, e conduz um número mínimo de operaçõs necessárias no DOM para atingir um novo estado do componente (DINIZ, 2022)

Figura 1 – Arquitetura do *React*



Fonte: Elaborado pelo o autor

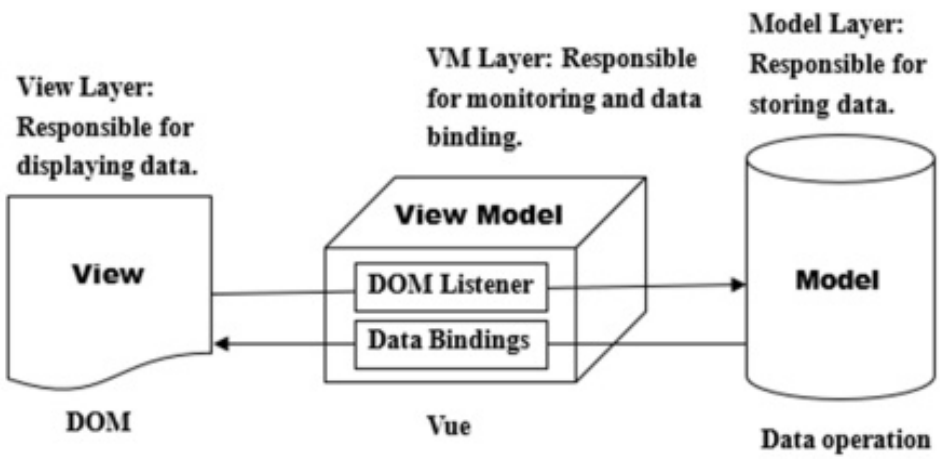
3.1.3 Vue.js

O *Vue* é um *framework* baseado em *JavaScript*, focado em construir aplicações WEB , foi criado por Evan You em 2013; o seu foco é construir aplicações SPA (Single-Page applications), assim como o React. De acordo com (KYRIAKIDIS; MANIATIS, 2016), o Vue possui um grande ecossistema de desenvolvimento, desde ferramentas auxiliadoras para desenvolvimento, quanto para plugins básicos.

3.1.3.1 Arquitetura

Diferentemente do *React*, sua arquitetura ViewModel faz parte do padrão MVVM (Model-View-ViewModel), a conexão entre a camada View e a Model, é mais forte. Com base na figura a baixo, podemos ver a relação de uma melhor forma:

Figura 2 – Arquitetura *Vue*



Fonte: (LI; ZHANG, 2021)

De acordo com a documentação oficial do Vue (VUE.JS, 2023), pode-se perceber que as aplicações são feitas principalmente com componentes que podem ser reutilizáveis, chamadas de Vue Instance. Como particularidade do *framework*, os componentes podem ser registrados como global na aplicação, assim, uma declaração global daquele componente, acaba o tornando disponível para todos os componentes filhos.

Apesar do seu foco ser outro, com o *NextJs*, dá-se para trabalhar com aplicações usando a renderização SSR (Server-side rendering), quando a renderização é feita pelo lado do servidor.

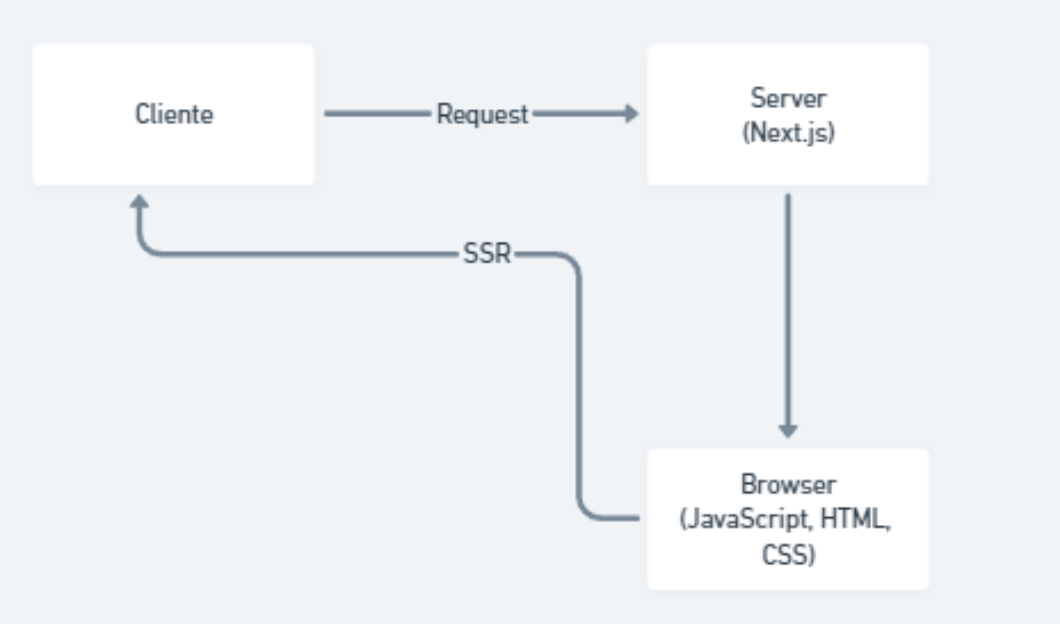
3.1.4 Next.js

Next.js é um framework de código aberto construído em cima do *React*, que permite a criação de aplicativos web modernos e escaláveis. Ele é projetado para ajudar os desenvolvedores a construir aplicativos complexos de forma mais fácil, fornecendo uma variedade de recursos poderosos, como renderização no lado do servidor (SSR), pré-renderização, geração estática de sites (SSG) e muito mais (NEXT.JS, 2023).

3.1.4.1 Arquitetura

Uma das principais diferenças entre o *Next.js* e o *React* é que o *Next.js* vem com um servidor embutido que permite que os desenvolvedores criem aplicativos que são mais *SEO-friendly*, pois podem ser pré-renderizados no lado do servidor, enquanto no React, isso precisa ser feito manualmente. Outra diferença, é que o *Next.js* vem com roteamento automático, o que significa que é fácil para os desenvolvedores criar rotas em seus aplicativos sem precisar se preocupar com a configuração do servidor. Em geral, o *Next.js* é uma ferramenta valiosa para desenvolvedores que desejam criar aplicativos web modernos e eficientes com React.

Figura 3 – Arquitetura *Next.js*



Fonte: Elaborado pelo autor.

De acordo com a documentação do *Next.js*, eis aqui algumas features que o frameworks possui:

– Busca de dados optimizada com suporte async/await em componentes React e a API

fetch() que se alinha com React e a plataforma Web

– Optimizações do componente de imagem aprimorado com carregamento lento do navegador nativo. Novo módulo de fontes com otimização automática de fontes.

4 TRABALHOS RELACIONADOS

Nesta seção, serão apresentados alguns trabalhos relacionados com o projeto proposto neste trabalho.

4.1 *Comparative study of some applications made in the Angular and Vue.js frameworks*

Em (NOVAC *et al.*, 2021) é apresentado um comparativo com análises técnicas entre *frameworks JavaScript*, em específico o *Angular e o Vue.js.* O objetivo principal é repassar o porquê que o *JavaScript* é uma ótima linguagem para desenvolvimento de aplicações *WEB*, possui uma fácil manipulação de variáveis etc, e claro, a comparação feita entre dois *frameworks* famosos que usam essa linguagem como base.

Em um primeiro momento, é apontado os pontos positivos de cada tecnologia, bem como seus pontos negativos. O que se usar em cada caso, e o porquê da escolha. É falado também, que pelo fato do *Vue.js* ser um *framework* mais novo, acaba sendo mais fácil de ser usado, diferentemente do *Angular*, que é mais antigo e mais verboso (NOVAC *et al.*, 2021).

Os resultados foram bem satisfatórios, com pouca diferença entre os dois, no que diz respeito à renderização de componentes e pintura dos mesmos. Porém em relação a nível de sistema,ou seja, o desempenho da máquina em questão para conseguir rodar a aplicação feita em *Angular*, acabou consumindo mais do que a aplicação do Vue.js (NOVAC *et al.*, 2021).

4.2 *Evaluating the perfomance of web rendering technologies based on JavaScript: Angular, React, and Vue*

Neste artigo publicado por (DINIZ, 2022), é apresentado por meio de uma aplicação genérica de brinquedo, possuindo uma metodologia um pouco diferente do primeiro artigo relacionado. Onde neste presente artigo, é explorado mais o conceito do DOM virtual, antes de falar propriamente dos *frameworks* e a biblioteca *React*.

Quando se fala dos *frameworks*, aponta apenas diferenças básicas, mas focando principalmente nas diferenças entre a renderização de componentes e a sua relação componente-DOM virtual. É falado também, sobre a curva de aprendizado de cada tecnologia, e sua relação com a quantidade de pessoas que usam tais tecnologias. Faz comparações tanto com visitas ao site *StackOverflow*, bem como quantidade de visualizações em vídeos de plataformas da internet, como o *Youtube* por exemplo. Foi constatado que o *React* possui um grande nível de

visualizações em comparação as outras duas tecnologias também apresentadas (DINIZ, 2022).

As métricas usadas foram com base no desenvolvimento de aplicações e as perspectivas de uso que melhor se encaixavam para aquela determinada ação. Os resultados mostraram diferenças bem significativas, e minuciosas em relação a cada tecnologia. Como por exemplo, o tempo do *Angular* para construir pacotes é o mais pesado em relação aos outros dois. Além de comparar o tempo de interação dos componentes, de cada página de cada aplicação, foi constatado que o *React* mostrou ser mais rápido que o próprio *JavaScript* nativo (DINIZ, 2022).

4.3 Análise comparativa entre frameworks front-end baseados em JavaScrpit para aplicações web

Em (FERREIRA, 2018), é apresentado a listagem de grandes *frameworks* de *JavaScript*, que atuam no mercado, apontando características únicas de cada um, com o intuito principal de ajudar desenvolvedores de software na escolha daquele que mais se adequa às necessidades pessoais e de projeto.

Ocorre uma comparação de arquitetura, fator de documentação e suporte da comunidade, compatibilidade de cada *framework* à cada navegador, compara também tempo de renderização e por fim, colhe informações mais assertivas para identificar aquele que mais atende aos principais requisitos com base no que se precisa (FERREIRA, 2018).

Os resultados foram satisfatórios, e bem semelhantes de um modo geral, comparado *framework* à *framework*. Por sua vez, assim como em (DINIZ, 2022), o *React* trouxe vantagens diantes dos concorrentes, porém em questão de perfomance, o *Vue.js* mostrou-se ser mais veloz.

4.4 Análise Comparativa

Assim como realizado em (NOVAC *et al.*, 2021), (DINIZ, 2022) e em (FERREIRA, 2018) este trabalho pretende realizar o comparativo de tecnologias *JavaScript* com foco no desenvolvimento WEB. Pretende-se utilizar algumas soluções semelhantes às usadas pelos principais participantes do desafio, além de utilizar técnicas não tão convencionais para buscar a melhor tecnologia com base nas necessidades individuais e gerais, como por exemplo, uma pesquisa com desenvolvedores que já estão no mercado.

Além disso, neste presente trabalho, ocorrerá a comparação de outra tecnologia que não foi abordada em nenhum dos trabalhos relacionados, que é o *Next.js*, uma poderosa

ferramenta de desenvolvimento frontEnd e que é bastante atual. Além também, de fazer uma aplicação genérica que use o *scroll* infinito, com dados que serão puxados de uma API, e ao final, fazer comparações no que diz respeito à desempenho, comunicação com DOM, renderização de componentes.

Na Tabela a seguir, é apresentado um resumo das características encontradas nos trabalhos relacionados descritos anteriormente, e a comparação com o projeto apresentado neste trabalho.

Tabela 1 – Tabela comparativa.

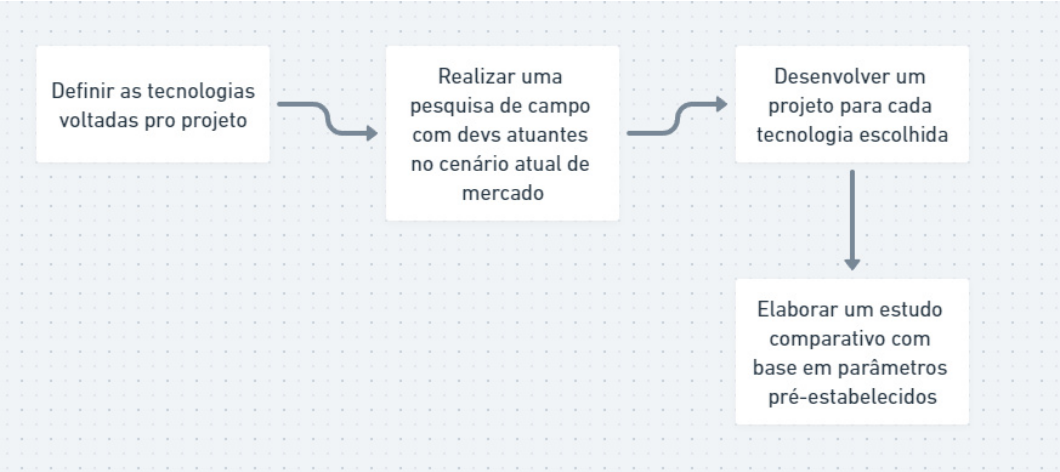
Características	**(NOVAC *et al.*, 2021)**	**(DINIZ, 2022)**	**(FERREIRA, 2018)**	**Presente Trabalho**
Pesquisa com desenvolvedores ou atores	Não	Não	Não	Sim
Testes de desempenho através de um projeto	Sim	Sim	Sim	Sim
Utilização dos *frameworks* mais populares do mercado	Sim	Sim	Sim	Sim
Análise da comunidade com base em alguns critérios (Documentação, mercado, curva de aprendizagem)	Sim	Não	Não	Sim

Fonte: Elaborado pelo autor.

5 PROCEDIMENTOS METODOLÓGICOS

Para atingir as metas estabelecidas nesta tarefa, é essencial um conjunto de fases a serem seguidas, conforme pode ser notado no diagrama de fluxo exibido na Figura a seguir:

Figura 4 – Fluxograma de procedimentos metodológicos



Fonte: Elaborado pelo autor.

5.1 Definir as tecnologias voltadas para o projeto de Pesquisa

A etapa inicial delineada no plano de pesquisa envolve a determinação das tecnologias a serem consideradas para a comparação. Os padrões utilizados para essa seleção envolveram a escolha de três tecnologias predominantes no desenvolvimento de interfaces web, que possuem uma notável popularidade entre os desenvolvedores. Isso é evidenciado pela presença de uma comunidade dinâmica e envolvida, uma curva de aprendizagem altamente favorável e sua ampla utilização no mercado de trabalho.

Em resumo, aqui estão alguns critérios que foram levados em consideração:

1. Popularidade entre a população do mercado
2. Comunidade ativa
3. Curva de Aprendizagem
4. Documentação
5. Seu uso no mercado de trabalho

5.2 Realizar uma pesquisa de campo com desenvolvedores *frontend* atuantes no cenário

Para executar a pesquisa de campo com desenvolvedores *frontend* atuantes no cenário atual, foi estabelecido um conjunto de fases. Primeiramente, foram estabelecidos critérios de seleção para identificar os participantes mais apropriados. Levou-se em conta a experiência mínima de dois anos no desenvolvimento frontend e a participação atual em projetos web. Posteriormente, foi elaborado um questionário estruturado abordando aspectos relevantes das tecnologias *JavaScript* direcionadas ao desenvolvimento web. Com as respostas dos desenvolvedores, isto é, o público-alvo, foi viável consolidar resultados já esperados, como reunir a lista de tecnologias amplamente utilizadas no mercado de trabalho, quais incentivos para a escolha da tecnologia e qual o grau de contentamento com a tecnologia selecionada pelo desenvolvedor.

Além disso, com base nas respostas, houve uma sondagem da lista de ferramentas que mais são utilizadas pelos desenvolvedores. Assim, foram coletadas informações pertinentes, pelas quais os desenvolvedores estão optando por trabalhar num framework/tecnologia. Seja para estudo, trabalho ou projeto pessoais.

5.3 Desenvolver um projeto para cada tecnologia escolhida

Para aprofundar a análise comparativa das tecnologias escolhidas, nessa terceira etapada, foi desenvolvido um projeto específico para todas elas. Essa seção foi dividida em 4 partes, onde será falado como o escopo da aplicação foi desenvolvida, qual a aplicação que será desenvolvida e também, qual os requisitos. E por fim, descrever a lista que será feita dos testes, e realização do benchmark com base no resultado desses testes.

5.3.1 Traçar plano e fazer a implementação da aplicação com as tecnologias escolhidas da pesquisa

Esta etapa envolveu a especificação dos pré-requisitos para a aplicação e sua construção utilizando as três tecnologias selecionados para este estudo atual. O aplicativo era uma *Pokedéx*, que listava um grande número de Pokémons, por meio de uma *API*. Esse consumo de informações se deu por meio dos 3 projetos. O projeto da Pokedéx teve esses requisitos:

1. Consumir dados de uma API
2. Renderização de um grande número de componentes (Cards)
3. Listar os Pokemons e filtrá-los

5.3.2 Realizar e projetar os resultados dos testes de benchmark do projeto

Nesta fase, procedeu-se à execução de testes previamente determinados na aplicação desenvolvida na seção anterior. Uma variedade de ferramentas foi empregada para facilitar os testes de referência, e o autor estabeleceu os seguintes testes.

1. Tempo de renderização de componente
2. Tempo de compilação
3. Tempo de execução de acesso à API
4. Tamanho do projeto

Com a realizaçação desses testes, foi possível fazer a análise dos resultados, e com base nessa análise de benchmark, em cada projeto de cada tecnologia, foi possível identificar características importantes de cada framework.

5.4 Elaborar um estudo comparativo dessas tecnologias

Com base nos projetos desenvolvidos para cada uma das tecnologias selecionadas, foi possível elaborar um estudo comparativo detalhado. Esse estudo levou em consideração diversos aspectos, como desempenho, facilidade de implementação, modularidade, capacidade de escalabilidade, disponibilidade de recursos, suporte da comunidade, pesquisas em sites, repositórios no *Github*. Tal estudo comparou o modo de inicialização de cada projeto, ferramentas que podem ser utilizadas e em quais situações é melhor de usar uma tecnologia em relação à outra, qual renderização é aceita em cada tecnologia. Cada descoberta foi analisada e discutida de forma apronfundada.

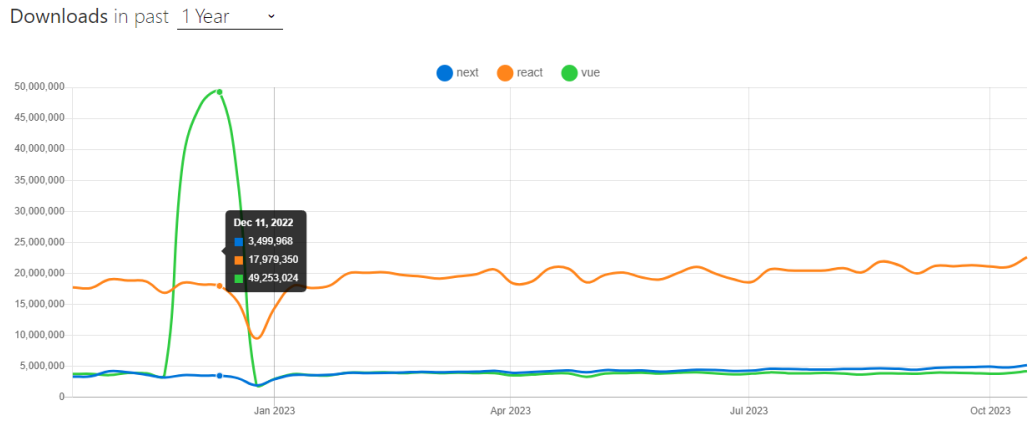
6 RESULTADOS

Nesta seção, serão apresentados os resultados desse presente trabalho.

6.1 Definição doas tecnologias voltadas para o projeto de Pesquisa

Para chegar nesses resultados que irão ser apresentados, da seção 5.0.1, buscou-se em alguns sites na internet, que os seus ecossistemas são basicamente sobre *Javascript*. Tais sites foram: Survey StackOverflow [1], npm trends [2] e StateOfJs [3]. Neles foram vistos os *Frameworks* que mais atendiam às expectativas/requisitos, com isso, foram escolhidos 3 tecnologias, *React*,*Vue* e *Next*. De acordo com a imagem a seguir, conseguimos ter uma ideia sobre a quantidade de desenvolvedores que usam tais tecnologias, nos mais diferentes âmbitos.

Figura 5 – Downloads feitos por desenvolvedores ao longo de 01 ano, de acordo com o site npm trends



Fonte: Site *npm trends*.

6.2 Pesquisa de campo com os desenvolvedores selecionados

Para entender quais são as melhores tecnologias no âmbito do desenvolvimento web, descobrir quais são as mais utilizadas, e avaliar a experiência dos desenvolvedores, com o uso de ferramentas, descobrir quais os motivos motivos de escolha do *Framework*, foi necessário fazer essa pesquisa de campo. Com um total de 15 respostas na pesquisa realização na seção 5.0.2, os resultados foram listados a baixo.

1. No mercado de trabalho, qual nível de entendimento você tem na área do frontend?

[1] https://survey.stackoverflow.co/2023/most-popular-technologies-language-prof
[2] https://npmtrends.com/next-vs-react-vs-vue
[3] https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/

2. Entre esses, quais tecnologias você trabalha/trabalhou?
3. O que te levaria à escolher uma tecnologia para ser trabalhar hoje em dia?
4. Com qual dessas tecnologias você teve o seu primeiro contato?
5. Você já trocou alguma vez de tecnologia durante sua vida profissional, se sim, por qual motivo?
6. Em sua opinião, qual é a importância de uma boa documentação no processo de aprendizado e uso de um novo framework/biblioteca JavaScript?
7. Com base no seu framework/tecnologia escolhida, liste alguma ferramenta que você utiliza no seu dia-a-dia. Ex:IDE, Bibliotecas, ferramentas para testes unitários
8. Hoje em dia, você é satisfeito com a escolha da sua tecnologia?

6.2.1 No mercado de trabalho, qual nível de entendimento você tem na área do frontend?

Esta pergunta serve para avaliar o conhecimento e a experiência dos desenvolvedores em relação ao desenvolvimento frontend. As respostas ajudaram a identificar se os entrevistados possuem uma compreensão abrangente e profunda do desenvolvimento frontend ou se estão mais focados em áreas específicas.

Tabela 2 – No mercado de trabalho, qual nível de entendimento você tem na área do frontend?

Senioridade	Número de Desenvolvedores
Júnior	6
Pleno	6
Sênior	3

Fonte: elaborada pelo autor

6.2.2 Entre esses, quais tecnologias você trabalha/trabalhou?

A resposta a esta pergunta revela quais tecnologias específicas os desenvolvedores estão mais familiarizados e confortáveis em usar. Isso forneceu um contexto valioso para entender a popularidade e a preferência de determinadas tecnologias no mercado. Essa pergunta em si não foi uma de múltipla escolha, cada desenvolvedor poderia colocar mais de uma resposta, por exemplo.

Tabela 3 – Entre esses, quais tecnologias você trabalha/trabalhou?

Tecnologia	Número de Desenvolvedores
React	14
Vue	5
Next	4

Fonte: elaborada pelo autor

6.2.3 O que te levaria à escolher uma tecnologia para ser trabalhar hoje em dia?

Esta pergunta ajudou a identificar os critérios e os fatores determinantes que os desenvolvedores consideram ao selecionar uma tecnologia para trabalhar. As respostas abordaram aspectos como desempenho, facilidade de aprendizado, documentação robusta, suporte da comunidade, entre outros.

1. Curva no aprendizado - 4 respostas
2. Facilidade de uso - 3 respostas
3. Necessidade (é a tecnologia usada na empresa)/Demanda dentro da empresa - 4 respostas
4. Melhor ferramentas para resolução de problemas que o mercado oferece - 2 respostas
5. Bom suporte - 1 respostas
6. Ser bastante escalável/Optimizável - 3 respostas
7. Manutenabilidade - 4 respostas
8. Documentação - 6 respostas

6.2.4 Com qual dessas tecnologias você teve o seu primeiro contato?

Esta pergunta forneceu insights sobre a familiaridade inicial dos desenvolvedores com as tecnologias específicas. Compreender a primeira experiência deles pode influenciar a preferência e a compreensão aprofundada de uma determinada tecnologia.

Tabela 4 – Com qual dessas tecnologias você teve o seu primeiro contato?

Tecnologia	Número de Desenvolvedores
React	13
Vue	1
Next	1

Fonte: elaborada pelo autor

6.2.5 Você já trocou alguma vez de tecnologia durante sua vida profissional? Se sim, por qual motivo?

1. Não - 3 respostas
2. Do web para o mobile - 1 respostas
3. Não se identificou - 1 respostas
4. Por causa da empresa que trabalha - 9 respostas

6.2.6 Em sua opinião, qual é a importância de uma boa documentação no processo de aprendizado e uso de um novo framework/biblioteca JavaScript?

1. Essencial - 5 respostas
2. Manutenção do código - 4 respostas
3. Aprendizagem - 3 respostas
4. Importante para entender novos pontos da tecnologia - 2 respostas

6.2.7 Com base no seu framework/tecnologia escolhida, liste alguma ferramenta que você utiliza no seu dia-a-dia. Ex: IDE, Bibliotecas, ferramentas para testes unitários

As respostas a essa pergunta ofereceram informações valiosas sobre o ecossistema de ferramentas e recursos que acompanham cada tecnologia. Destaca-se a conveniência e a eficácia do conjunto de ferramentas associadas a cada framework e como elas contribuem para o fluxo de trabalho do desenvolvedor.

1. IntelliJ IDEA - 6 respostas
2. VSCode - 9 respostas
3. Jest - 5 respostas
4. Sublime Text - 2 respostas
5. WSL - 1 respostas
6. Data Grip - 1 respostas
7. ChatGPT - 1 respostas
8. Axios - 2 respostas
9. Redux - 3 respostas
10. Material UI - 2 respostas
11. SonarQube - 1 respostas

12. Pupperteer - 1 respostas

6.2.8 Hoje em dia, você é satisfeito com a escolha da sua tecnologia?

Esta pergunta ajuda a avaliar a satisfação geral dos desenvolvedores com a tecnologia que estão usando atualmente. Compreender os pontos fortes e fracos percebidos de cada tecnologia pode oferecer uma visão geral das preferências e dos desafios que os desenvolvedores enfrentam em sua prática diária.

Tabela 5 – Hoje em dia, você é satisfeito com a escolha da sua tecnologia?

Resposta	Número de Desenvolvedores
Sim	15

Fonte: elaborada pelo autor

Um dos pontos que mais foi percebido, é que no âmbito do mercado, conforme necessidades específicas, os grupos de desenvolvimento, tendem a trocar de tecnologia, para seguirem novos fluxos dentro da empresa. O React foi a biblioteca que mais teve respostas dos desevolvedores, isso se destaca principalmente por causa da sua extrema popularidade.

6.2.9 Projeto para cada tecnologia escolhida

Nessa seção, irão ser apresentados os resultados obtidos por meio do projeto de uma página, que representa uma *Pokedéx*, nas 3 diferentes linguagens. Os resultados dos testes seguiram a seção 5.0.3.2.

Tudo que foi feito nessa seção, realizou-se por meio de um computador, com a seguinte configuração:

- **Placa Mãe:** AsRock A320M-HD
- **Processador:** AMD Ryzen 5 1600 Six-Core Processor
- **Memória RAM:** 24 GB 1400 MHz DDR4
- **Placa de Vídeo:** NVIDIA GeForce GTX 1650 SUPER

6.2.10 Execução dos testes

Logo a baixo, estão os testes que foram feitos, seguindo as métricas da seção 5.0.3.2:

- **Tempo de renderização de componente**
- **Tempo de compilação**

- **Tempo de execução de acesso à API**
- **Tamanho do projeto**

Como dito anteriormente, foram usados em cada projeto, as tecnologias React, Next e Vue. Até o momento que foi feito os projetos, e os testes de desempenho, as versões de cada Framework foram essas:

Tabela 6 – Tecnologias e suas versões

Tecnologia	Versão
React	18.2.0
Vue	3.3.7
Next	14.0.1

Fonte: elaborada pelo autor

Tendo em vista o projeto em si, algumas limitações visuais, ou até mesmo complexas, deixaram a desejar um pouco, pois o foco dos projetos era analisar por meio de testes, as individualidades de cada Framework.

Por meio de:

- Listar Pokemons
- Filtrar Pokemon(s) escolhido(s)
- Tempo de Compilação
- Tamanho e Tempo de build

Foram analisados cada comportamento, e em cada caso, serviu para avaliar as tecnologias. O navegador padrão para os testes, foi o Google Chrome, que até o momento, é o mais popular navegador do mundo. Os projetos estão separados em repositórios individuais, no site do Github, mais especificamente no repositório do autor [4]

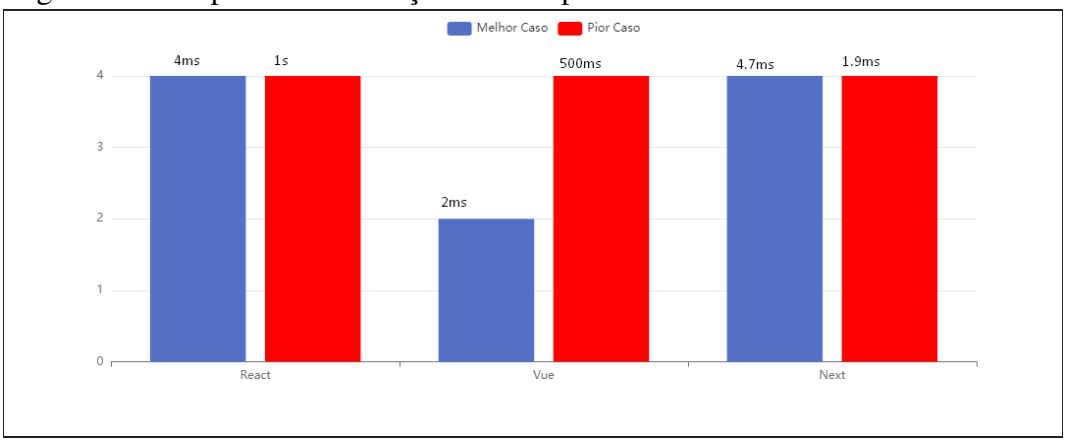
Todos os projetos possuem a mesma arquitetura de componentes. Onde dois componentes principais (MainNavBar e MainCard) são renderizados em outro componente, o componente *Home*. Um grande volume de dados, provenientes da *PokeAPI* [5] foi proposital, pois o foco é testar a performance de cada tecnologia. Para fins de análises, os gráficos que serão mostrados nas próximas figuras, foram limitados para fins visuais, mas como o intuito é apenas medir, não faz tanto sentido se preocupar com limitações de valores.

No primeiro teste, o de verificar o tempo em que todos os Cards eram renderizados, notou-se uma grande semelhança no *React* e o *Next*, porém no *Vue*, todo o componente foi

[4] https://github.com/rubbinhoxd
[5] https://pokeapi.co/

renderizado de uma forma mais rápida. O melhor caso do React (4ms), foi parecido com o melhor caso do Next (4.7), já no *Vue*, o melhor caso foi de 2ms, como podemos ver na Figura 6.

Figura 6 – Tempo de renderização do Componente Home

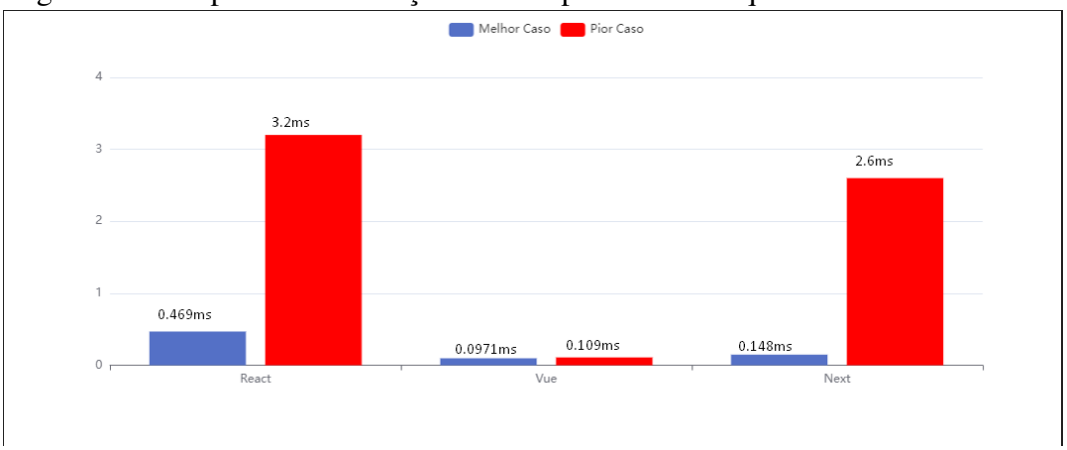


Fonte: Elaborado pelo autor.

O *Next* teve o pior caso (1.9s), entre as tecnologias. Ele não se deu tão bem com uma aplicação onde ocorriam várias renderizações de um componente, dentro de outro. O *Vue* possuiu um pior caso de 500ms e o *React*, possuiu um pior caso de 1s.

No outro teste, da renderização individual de cada componente Card, isto é, quando os Pokemons são filtrados, os resultados foram mostrados na figura 7.

Figura 7 – Tempo de renderização do Componente Card quando filtrado

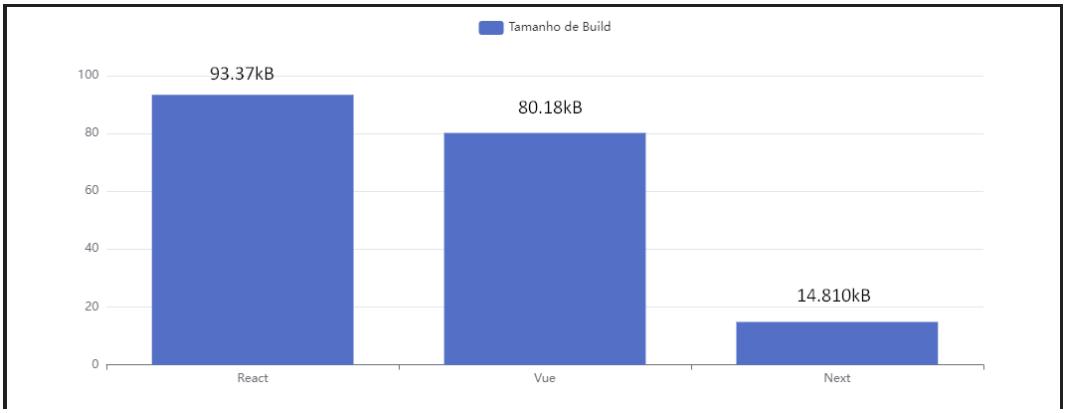


Fonte: Elaborado pelo autor.

O *Vue* teve um melhor caso em relação aos demais (0.0971ms), e o melhor pior caso (0.109ms) do que os demais. Novamente, em tempos de renderização de componentes, o *React* e o *Next* tiveram tempos bastante semelhantes, na casa dos *ms*. O primeiro, possuiu um melhor caso de 0.469ms e um pior caso de 3.2ms, enquanto o segundo (*Next*) possuiu um melhor caso de 0.148ms, e pior caso de 2.6ms .

Já na figura 8, o próximo teste consistiu em comparar o tamanho de *build* de cada aplicação WEB. O menor resultado, das 3, foi o do *Next*, com um tamanho de build de 14.810kB, seguindo logo após, o *Vue*, que teve um tamanho de 80.18kB. E por fim, a aplicação *React* teve o maior tamanho, com 93.37kB. Tal informação é útil para saber qual aplicação consome mais memória, e de certa forma, leva mais tempo para ser renderizada. É mostrado na prática, que o React é um pouco mais pesado em comparação as outras tecnologias.

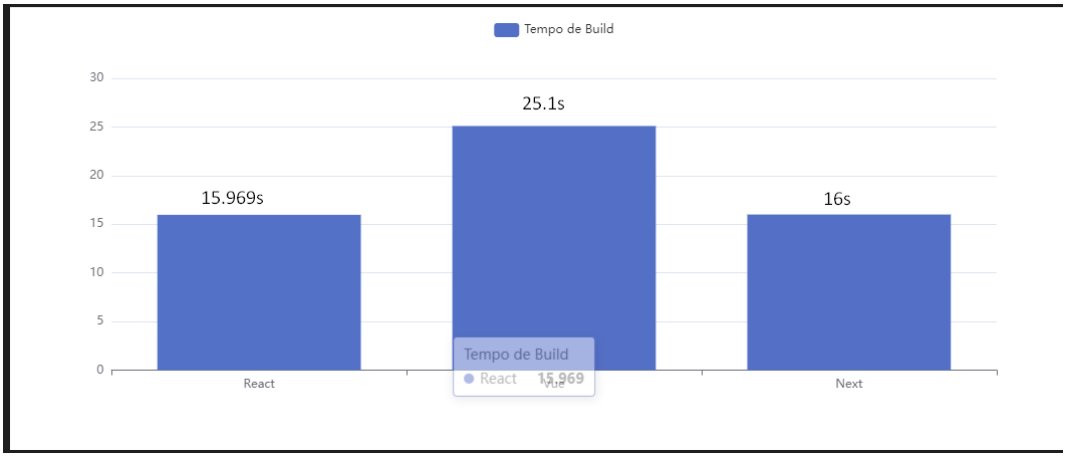
Figura 8 – Tamanho de build de cada aplicação WEB



Fonte: Elaborado pelo autor.

No tempo de build, o *React* e o *Next* tiveram tempos semelhantes, o primeiro teve um tempo de 15.969s e o segundo, 16s. Isso diz muita coisa, pois mesmo o tamanho do build do *React*, ele conseguiu ter um tempo quase igual ao que teve um tamanho de build menor. E o *Vue*, por sua vez, teve um tempo de build de 25.1s. Tudo é mostrado na figura 9

Figura 9 – Tempo de build de cada aplicação WEB

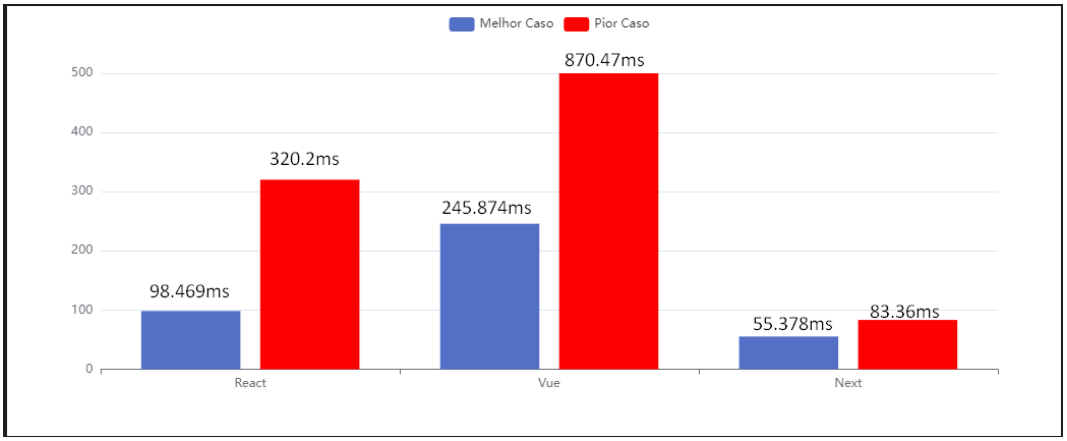


Fonte: Elaborado pelo autor.

Por fim, a figura 10 mostra o tempo de acesso à API de cada aplicação. Ou seja, por meio de uma requisição usando o *Axios*, para a API do *PokeAPI*, os cards foram renderizados

nos seus respectivos *Json's*. O *Next* demonstrou ter uma grande facilidade em comunicação com API's, possuiu um melhor caso, de 55.378ms, e o menor pior caso, com 83.36ms. O *React* por sua vez, possuiu um Pior caso de 320.2ms e um melhor caso de 98.469ms para cada requisição. O *Vue* se manteve estável, e possuiu um melhor caso de 245.874ms e um pior caso de 870.47ms.

Figura 10 – Tempo de acesso à API Pokedex na visão de cada tecnologia



Fonte: Elaborado pelo autor.

6.2.11 Ferramentas

Com o passar do tempo, foi visto que ferramentas para testes, são mais comuns do que se imagina, e são bastanta usáveis em todos os âmbitos do desenvolvimento WEB. Em relação as tecnologias do trabalho, foram encontrados boas semelhanças entre eles. Todas as tecnologias possuem uma diversidade para realização de testes em suas aplicações. Em relação ao *React* - *Next*, há grandes semelhanças. Pois um projeto feito em *React* utiliza *Jest* ou *Testing Library*, da mesma forma o *Next*, que faz utilizações de componentes do *React*.

Para gerenciamento de estados, novamente o *React* e *Next* possuem as mesmas tecnologias, que são o *Redux* e o *Context API*. Já o *Vue*, utiliza o *VueX*.

Para concluir, ferramentas que ajudam no desenvolvimento de aplicações WEB, que algumas mesmo, já foram vistas no tópico da Pesquisa de Campo, são comuns para as mais variadas tecnologias. O *Material UI - MUI Lab* é uma biblioteca que possui componentes totalmente renderizáveis, com estilo próprio, boa palheta de cores, tem um nível de acessibilidade para as pessoas que não conseguem enxergar tão bem, é comum para as 3 tecnologias.

O SonarQube e Axios também são ferramentas bem famosas, e são tecnologias que hoje em dia no mercado de trabalho do desenvolvimento, são essenciais porque uma é responsável pela manutenabilidade do código, e o outro serve para comunição com API's.

Apesar de existir uma gama infinita de ferramentas, uma coisa é certa: cada uma possui sua especificidade e importância no projeto. Ficou nítido na pesquisa, que para os desenvolvedores, as ferramentas são indispensáveis no desenvolvimento e no dia-a-dia.

6.3 Estudo comparativo de cada tecnologia

Nesta última etapa, os resultados obtidos ao longo do trabalho, foram considerados. Com estes resultados, critérios de avaliação foram definidos. Documentação, Comunidade, Curva de aprendizagem, Mercado de trabalho, modo de inicialização do projeto, foram alguns dos critérios usados. Depois dessas avaliações individuais, foi colocado numa imagem o resumo dessa seção, das diferenças obtidas entre o *React* x *Vue* x *Next*.

6.3.1 React

Documentação: Atualmente, possui uma documentação que dá suporte para 17 linguaguens diferentes, e mais 9 estão em progresso. Totalizando ao todo, 26 linguagens.

Comunidade: O *React* possui uma grande comunidade, tanto brasileira, quanto ao redor do mundo. O número de visualizações, posts feitos em blogs na WEB, dúvidas postas em sites famosos na comunidade, como o StackOverflow [6] dizem muito sobre essa afirmação.

Curva de Aprendizagem: Com base na pesquisa feita nesse estudo, percebemos que por meio dos resultados, desenvolvedores atuantes no mercado, há bastante tempo, apontaram a curva de aprendizagem do *React* como um fator crucial para começar os estudos na tecnologia, ou continuar trabalhando nela até os dias atuais.

Mercado de Trabalho: Por meio de vários sites, como o *LinkedIn* [7], por exemplo, podemos ver as mais variadas vagas para Desenvolvedores, nos mais variados níveis (Júnior, Pleno ou Sênior), afirmando que o mercado para esse tecnologia, é muito bom.

Inicialização: De uma forma bem simples, por meio de uma *tag script*, usando *Node.js* e um pacote de instalação (*NPM* ou *Yarn*), facilmente o primeiro projeto consegue ser criado. Com as dependências principais já instaladas. Basta executar o comando: npx create-react-app nome-do-meu-projeto.

[6] https://stackoverflow.com/
[7] https://www.linkedin.com/

6.3.2 Vue

Documentação: Atualmente, possui uma documentação que abrange 8 linguaguens, mas eles frizam que estão em busca de mais contribuidores, para traduzirem sua documentação em mais linguagens.

Comunidade: A comunidade do *Vue*, também é bastante grande. Foi visto na pesquisa, que o número de donwloads dos pacotes do *Vue* foram maiores que o próprio *React*, no primeiro trimestre do ano passado. No seu próprio Github, o número de seguidores (12,9mil) é maior, do que o do *React*, que atualmente possui 3.6mil.

Curva de Aprendizagem: Possui uma boa curva de aprendizagem, que também é bem curta, assim como o *React*. Com pouco conhecimento, já é possível fazer boas aplicações WEB com *Vue*.

Mercado de Trabalho: No *LinkedIn* também é possível ver a grande quantidade de vagas de trabalho, para essa tecnologia. O número de vagas é muito grande, e o mercado está constantemente crescendo, principalmente pelas demandas das Empresas.

Inicialização: Usando o CLI do próprio *Vue*, que é um pacote de instalação e também uma *tag script*.Porém, o desenvolvedor pode optar também, caso queira, a baixar via pacotes tradicionais (*NPM* e *Yarn*). Usando o CLI, basta executar o seguinte comando: vue create nome-do-seu-projeto.

6.3.3 Next

Documentação: Possui suporte apenas ao Inglês, e até o momento, não estão em progresso, nenhuma nova linguagem. Tal ponto deve ser ressaltado, pois quando a documentação possui suporte apenas para uma única linguagem, a acessibilidade diminui, pois quem não tem o conhecimento prévio naquela específica, pode acabar perdendo-se um pouco, para conseguir entender um determinado assunto.

Comunidade: Das três tecnologias, a página que possui o menor número de contribuintes, é a do *Next*, mas ainda sim é um bom número. A comunidade é bastante ativa. Ao longo do ano, os números de donwloads dos pacotes, foi quase o mesmo que o do *Vue*.

Curva de Aprendizagem: A curva de aprendizagem é um pouco mais longa, se comparar com o *React*, por exemplo. Porque o *Next* é um framework derivado do *React*, então além de saber essa tecnologia, deve-se também aprender as particularidades do *Next*.

Mercado de Trabalho: O mercado de trabalho também é grande, muito dinâmico. Na maioria das vagas, o conhecimento em *Next* é um diferencial para as empresas, então quem domina esse *framework* possui uma grande vantagem nos demais concorrentes.

Inicialização: De uma forma bem simples, por meio de uma *tag script*, usando *Node.js* e um pacote de instalação (*NPM* ou *Yarn*), facilmente o primeiro projeto consegue ser criado. Com as dependências principais já instaladas. Basta executar o seguinte comando: npx create-next-app@latest.

A seguir, será mostrado um resultado geral sobre o comparativo:

Figura 11 – Resultados gerais do comparativo

Popularidade	React	Vue	Next
Pesquisas no npm trends, stateOfJs e StackOverflow	+ 450.000 resultados até 20 de Outubro	+200.000 resultados até 20 de Outubro	+150.000 resultados até 20 de Outubro
Repositório no Github	16.096 commits, 216.000 estrelas, 1641 contribuidores, + 18.000.0000 de usuários	3562 commits, 206.000 estrelas, 362 contribuidores	18.569 commits, 115.000 estrelas, 2995 contribuidores, +2.000.000 de usuários
Número de linguagens suportadas em cada documentação & Informação	26	8	Apenas Inglês
Visualizações no Youtube	+15.000.000 de views	+5.000.000 de views	+1.000.000 de views

Fonte: Elaborado pelo autor.

Por meio de pesquisas em sites da internet, como o *Github*, *npm trends, StackOverflow, stateOfJs, Youtube*, foi possível ir mais a fundo, no âmbito da comunidade de cada tecnologia. Os pontos descobertos foram: o *React* mais uma vez demonstra ser o que possui a maior comunidade, principalmente pelo número de views nos vídeos do *Youtube*, logo em seguida vem o *Vue* e por fim, o *Next*. Ambos os repositórios possuem uma grande quantidade de estrelas e contribuintes. Isso é bom em todos os sentidos, pois, com um grande número de colaboradores, constantemente *features* são feitas nas tecnologias, facilitando por sua vez, a experiência da comunidade de usuários. E quanto mais estrelas um determinado repositório possui, a tendência à longo prazo, é que o número aumente, fazendo assim, com que a comunidade fortaleça-se cada vez mais.

Após as comparações, a documentação, como elemento crucial no desenvolvimento

eficiente, demonstrou nuances distintas entre as tecnologias estudadas. A extensão, clareza e atualização da documentação desempenham papéis essenciais na eficácia do processo de aprendizagem e na produtividade a longo prazo.

A comunidade, um dos pilares que fortalecem qualquer tecnologia, revelou-se vital para o suporte contínuo, troca de conhecimentos e resolução de problemas complexos. A riqueza da comunidade pode influenciar diretamente na robustez e na evolução de uma tecnologia.

A curva de aprendizagem, embora inerentemente subjetiva, destacou-se como um fator determinante na seleção da tecnologia. A facilidade de assimilação e o suporte educacional disponível podem acelerar ou retardar significativamente o processo de adoção.

Finalmente, o mercado de trabalho emerge como um indicador poderoso da relevância e da demanda por uma determinada tecnologia. A valorização no mercado, a disponibilidade de oportunidades e a remuneração associada fornecem perspectivas tangíveis para os profissionais que investem tempo e energia em seu aprimoramento.

7 CONCLUSÕES FINAIS

Ao longo desta pesquisa, adentramos o vasto universo das tecnologias front-end, explorando e comparando React, Vue e Next. Cada passo metodológico, desde a definição das tecnologias até a conclusão do estudo comparativo, ofereceu perspectivas valiosas sobre o panorama dinâmico do desenvolvimento web moderno.

Inicialmente, ao definir as tecnologias para o escopo da pesquisa, observamos a crescente popularidade e adoção generalizada de React, a estabilidade e simplicidade do Vue, e a proposta inovadora e integrada do Next. Essa escolha estratégica estabeleceu o cenário para a pesquisa de campo, onde a voz dos desenvolvedores front-end, atuantes na linha de frente da indústria, trouxe nuances práticas e experiências do mundo real para enriquecer nossa compreensão.

A etapa de desenvolvimento de projetos proporcionou insights operacionais, revelando a ergonomia, a flexibilidade e as peculiaridades de cada tecnologia. A criação de aplicativos utilizando React, Vue e Next permitiu uma imersão prática, destacando pontos fortes e desafios específicos de cada framework.

A fase culminante do estudo comparativo mergulhou nas quatro dimensões críticas: documentação, comunidade, curva de aprendizagem e mercado de trabalho. A documentação, como guia essencial, mostrou variações em termos de extensão e clareza, impactando diretamente a eficiência do desenvolvimento. A comunidade, um tecido vivo em torno dessas tecnologias, revelou-se como uma fonte inestimável de suporte e colaboração. A curva de aprendizagem, subjetiva por natureza, apresentou diferentes desafios, refletindo nas preferências individuais dos desenvolvedores. E, por fim, o mercado de trabalho, como um termômetro da relevância, indicou tendências e demandas no cenário profissional.

Ao finalizar este estudo, é evidente que cada uma das tecnologias investigadas traz consigo uma oferta única. React, consolidado e adotado amplamente, Vue, com sua simplicidade e versatilidade, e Next, fornecendo uma experiência integrada para o desenvolvimento web. A escolha entre essas ferramentas não é apenas uma questão de recursos técnicos, mas também uma consideração cuidadosa das preferências individuais, das demandas do projeto e das tendências do mercado.

Em última análise, este trabalho não apenas oferece uma visão comparativa dessas tecnologias, mas também destaca a dinâmica constante do ecossistema front-end. Navegar nesse cenário em evolução requer não apenas habilidades técnicas aguçadas, mas também uma

abordagem adaptável e uma disposição para a aprendizagem contínua.

À medida que encerramos este capítulo, é imperativo reconhecer que o front-end é mais do que apenas códigos e frameworks; é uma jornada incessante de descoberta e inovação. Que este estudo sirva como bússola para aqueles que buscam orientação nas ondas sempre mutáveis do desenvolvimento web front-end.

REFERÊNCIAS

DINIZ. 2022 XVLIII Latin American Computer Conference (CLEI). In: **Evaluating the performance of web rendering technologies based on JavaScript**: Angular, React, and Vue. [*S. l.*: *s. n.*], 2022. p. 1–9. ISSN: 2771-5752. Acesso em: 12 out. 2023.

DUARTE, N. F. B. **Frameworks e Bibliotecas Javascript**. Tese (Doutorado) – Instituto Politecnico do Porto (Portugal), 2015. Acesso em: 04 dez. 2023.

FARLAND. **JavaScript & JQuery:**: The Missing Manual. [*S. l.*]: "O'Reilly Media, Inc.", 2011. Google-Books-ID: IX95AAAAQBAJ. ISBN 9781449320461. Acesso em: 05 de Fev. 2023.

FERREIRA. **. Análise comparativa entre frameworks frontend baseados em javascript para aplicações web.** 2018. Disponível em: https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/502. Acesso em: 12 out. 2023.

GIZAS. Proceedings of the 21st International Conference on World Wide Web. In: **Comparative evaluation of javascript frameworks**. Lyon France: ACM, 2012. p. 513–514. ISBN 9781450312301. Disponível em: https://dl.acm.org/doi/10.1145/2187980.2188103. Acesso em: 02 fev. 2023.

JAVEED, A. 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT). In: **Performance Optimization Techniques for ReactJS**. Coimbatore, India: IEEE, 2019. p. 1–5. ISBN 9781538681589. Disponível em: https://ieeexplore.ieee.org/document/8869134/. Acesso em: 18 abr. 2023.

KYRIAKIDIS, A.; MANIATIS, K. **The Majesty of Vue.js**. [*S. l.*]: Packt Publishing, 2016. Google-Books-ID: Xp7cDgAAQBAJ. ISBN 9781787125209.

LI, N.; ZHANG, B. The Research on Single Page Application Front-end development Based on Vue. **Journal of Physics: Conference Series**, v. 1883, n. 1, p. 012030, abr. 2021. ISSN 1742-6596. Disponível em: https://dx.doi.org/10.1088/1742-6596/1883/1/012030. Acesso em: 17 abr. 2023.

MARIANO, C. Benchmarking JavaScript Frameworks: Dissertations. jan. 2017. Disponível em: https://arrow.tudublin.ie/scschcomdis/94. Acesso em: 07 mar. 2023.

NEXT.JS. **Getting Started | Next.js**. 2023. Disponível em: https://nextjs.org/docs. Acesso em: 03 mai. 2023.

NOVAC, O. C.; MADAR, D. E.; NOVAC, C. M.; BUJDOSó, G.; OPROESCU, M.; GAL, T. 2021 16th international conference on engineering of modern electric systems (emes). In: **Comparative study of some applications made in the Angular and Vue.js frameworks**. [*S. l.*: *s. n.*], 2021. p. 1–4. Acesso em: 12 Out. 2023.

REACT.JS. **Getting Started – React**. 2023. Disponível em: https://legacy.reactjs.org/docs/getting-started.html. Acesso em: 18 abr. 2023.

VUE.JS. **Introduction | Vue.js**. 2023. Disponível em: https://vuejs.org/guide/introduction.html. Acesso em: 03 mai. 2023.

ZAKAS, . **High Performance JavaScript**. [*S. l.*]: "O'Reilly Media, Inc.", 2010. Google-Books-ID: ZOtVCgAAQBAJ. ISBN 9781449389147. Acesso em: 12 out. 2023.