Patrícia Pieroni Amarante

# ANÁLISE COMPARATIVA ENTRE AS FERRAMENTAS FRONT-END JAVASCRIPT PARA O DESENVOLVIMENTO DE APLICAÇÕES DE PÁGINA ÚNICA (SPA): ANGULAR, REACT E VUE

Formiga - MG

2023

Patrícia Pieroni Amarante

# ANÁLISE COMPARATIVA ENTRE AS FERRAMENTAS FRONT-END JAVASCRIPT PARA O DESENVOLVIMENTO DE APLICAÇÕES DE PÁGINA ÚNICA (SPA): ANGULAR, REACT E VUE

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Roger Santos Ferreira

Formiga - MG

2023

Amarante, Patrícia Pieroni
A485a Análise comparativa entre as principais ferramentas Frontend Javascript: angular, react e vue / Patrícia Pieroni Amarante -- Formiga : IFMG, 2023.
70p. : il.

Orientador: Prof. MSc. Roger Santos Ferreira
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Ferramentas front-end JavaScript. 2. Angular. 3. React. 4. Vue. 5. Análise comparativa. I. Ferreira, Roger Santos. II. Título.

CDD 004

*Ficha catalográfica elaborada pela Bibliotecária Msc. Simoni Júlia da Silveira*

**MINISTÉRIO DA EDUCAÇÃO**
**SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA**
**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS**
**Campus Formiga**
**Diretoria de Ensino**
**Docência Área Acadêmica de Computação**
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

## PATRÍCIA PIERONI AMARANTE

# ANÁLISE COMPARATIVA ENTRE AS PRINCIPAIS FERRAMENTAS FRONT-END JAVASCRIPT: ANGULAR, REACT E VUE

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

**APROVADO** em 30 de junho de 2023.

BANCA EXAMINADORA

Prof. Roger Santos Ferreira (Orientador)

Prof. Manoel Pereira Junior (IFMG)

Prof. Fernando Paim Lima (IFMG)

Formiga, 30 de junho de 2023.

Documento assinado eletronicamente por **Roger Santos Ferreira**, **Professor**, em 30/06/2023, às 18:15, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Manoel Pereira Junior**, **Professor**, em 30/06/2023, às 21:39, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Paim Lima**, **Professor**, em 02/07/2023, às 12:25, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **1598079** e o código CRC **D77CD656**.

*Dedico este trabalho a todos aqueles que contribuíram para o meu crescimento acadêmico e pessoal ao longo desta jornada, especialmente aos meus pais, namorado, irmão, e professores.*

# Agradecimentos

Agradeço aos meus pais, Mara Pieroni Amarante e Luiz Alberto Amarante, ao meu namorado, Túlio Brito Fernandes, ao meu irmão, César Pieroni Amarante, e aos meus professores, em especial ao meu professor orientador Roger Santos Ferreira, por todo o apoio ao longo deste trabalho de conclusão de curso. Todo conhecimento, incentivo e suporte foram fundamentais. Expresso minha sincera gratidão. Muito obrigada!

*"A ciência da computação não é apenas sobre código, é sobre entender como a tecnologia pode melhorar a vida das pessoas." (R. S. Nikhil Murthy)*

# Resumo

Com o crescimento do desenvolvimento web, escolher a tecnologia certa para a criação de aplicações *web* pode ser um desafio. Nesse contexto, as ferramentas *front-end* JavaScript desempenham um papel fundamental na construção de interfaces de usuários interativas e responsivas. Este trabalho oferece um panorama comparativo do Angular, React e Vue, permitindo que os desenvolvedores se baseiem em tabelas e situações comparativas para tomar decisões informadas sobre a escolha da ferramenta mais adequada para suas necessidades e preferências. A análise é realizada com base em fatores como popularidade, maturidade e estabilidade, curva de aprendizado e desempenho. As métricas e critérios abordados neste estudo comparativo foram considerados relevantes com base em diversos artigos e referências especializadas. Ao levar em conta os fatores mencionados, este trabalho segue uma abordagem fundamentada em pesquisas, testes e análises prévias.

**Palavras-chave**: desenvolvimento web, ferramentas front-end JavaScript, Angular, React, Vue, análise comparativa.

# Abstract

With the growth of web development, choosing the right technology for building web applications has become a challenge. In this context, JavaScript front-end tools play a key role in building interactive and responsive user interfaces. This work offers a comprehensive and comparative overview of Angular, React and Vue, allowing developers to rely on comparative tables and situations to make informed decisions about choosing the most suitable tool for their needs and preferences. The analysis is performed based on factors such as popularity, maturity, and stability, learning curve and performance. The metrics and criteria addressed in this comparative study were considered relevant based on several articles and specialized references. By considering the aforementioned factors, this work follows an approach based on previous, tests and analysis.

**Keywords**: web development, front-end tools JavaScript, Angular, React, and comparative analysis of Vue.

# Lista de ilustrações

# Sumário

# 1 Introdução

Nos últimos anos, a demanda por aplicações *web* altamente complexas e sofisticadas vem crescendo demasiadamente, substituindo antigos aplicativos *desktop*, que antes necessitavam da instalação de diversas ferramentas na máquina do usuário e hoje estão disponíveis na *web*. Embora a base do desenvolvimento permaneça a mesma, com HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheet*) e JavaScript (JS), são exigidas do cliente (navegador) cada vez mais responsabilidades, mais um fato que contribui para o aumento da criação de *frameworks* e bibliotecas, já que oferecem códigos e estruturas que facilitam e agilizam o desenvolvimento (PEKARSKY, 2020).

O JavaScript é atualmente uma das principais linguagens para programação do lado do cliente em navegadores *web*, a maioria dos sites utilizam JavaScript de alguma forma para tornarem as páginas mais interativas e apresentarem mais funcionalidades, sua popularidade, ocasionou uma desenfreada velocidade de inovação dentro da comunidade, dificultando aos desenvolvedores escolher as ferramentas apropriadas de tantas, aparentemente, boas opções (AMBLER; CLOUD, 2015).

*Frameworks* são estruturas compostas por um conjunto de códigos genéricos que possibilitam o desenvolvimento de sistemas e aplicações, eles funcionam como uma espécie de *template* ou modelo que, quando utilizados, oferecem certos artifícios e elementos estruturais básicos para a criação de alguma aplicação ou *software*, facilitando e agilizando o seu desenvolvimento (SAUVé, 2007).

Ao observar o gráfico de quantidade de perguntas pesquisadas no Stack Overflow, nota-se que o React foi a ferramenta mais procurada, seguido por Angular e Vue, como ilustrado pela figura 1, o gráfico de análise de popularidade também disponibilizado pelo Stack Overflow em seu relatório (Figura 2), também confirma a relevância e popularidade dessas três ferramentas, em relação às tecnologias envolvendo JavaScript, justificando a adoção destas, neste estudo comparativo.

Comparar Angular, React e Vue é fundamental para entender as diferenças em termos de recursos, desempenho, flexibilidade, comunidade de desenvolvedores e suporte. Essa análise comparativa auxilia a identificar qual dessas ferramentas é a mais adequada para atender às necessidades específicas de um projeto de software dentro de um contexto de aplicação de página única (SPA).

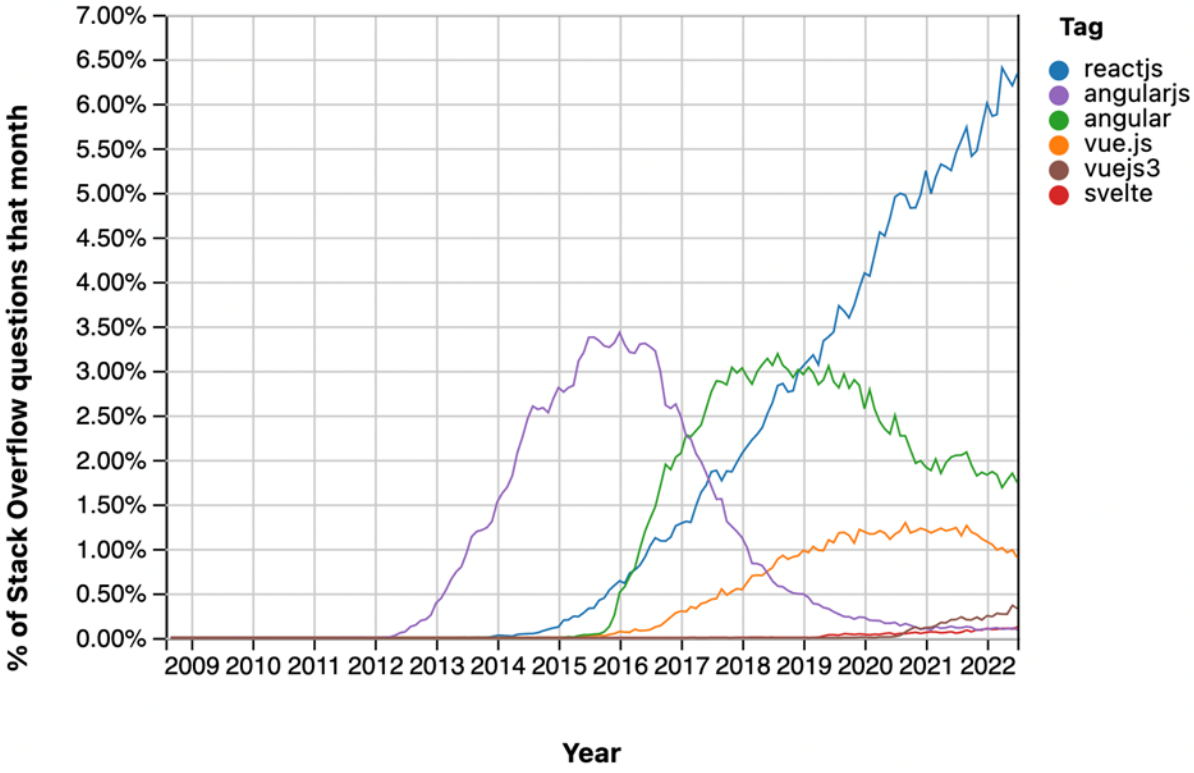



Figura 1 – Gráfico da quantidade de perguntas no Stack Overflow no mês de junho (2009-2022)

Fonte: (STACK..., 2022)

## 1.1 Descrição do problema

A escolha da ferramenta JavaScript mais adequada é um desafio para a comunidade de desenvolvimento devido à ampla variedade, complexidade e constante evolução dessas ferramentas, como evidenciado por várias pesquisas e trabalhos relacionados (consulte a seção 2.4 para mais informações). No entanto, essa etapa de seleção é crítica e decisiva para um projeto de desenvolvimento, pois ele atua como a espinha dorsal do projeto. Uma vez escolhido e iniciado o desenvolvimento, qualquer mudança posterior pode exigir reconstrução e remodelação de toda a estrutura do projeto (GALLAGHER; HATCH; MUNRO, 2008). Essa complexidade e o impacto significativo dessas decisões destacam a importância de uma análise comparativa detalhada entre as principais ferramentas *front-end* JavaScript, como Angular, React e Vue, para auxiliar os desenvolvedores na escolha mais adequada para seus projetos.

## 1.2 Objetivos

Nesta seção, serão apresentados os objetivos gerais e específicos deste trabalho.

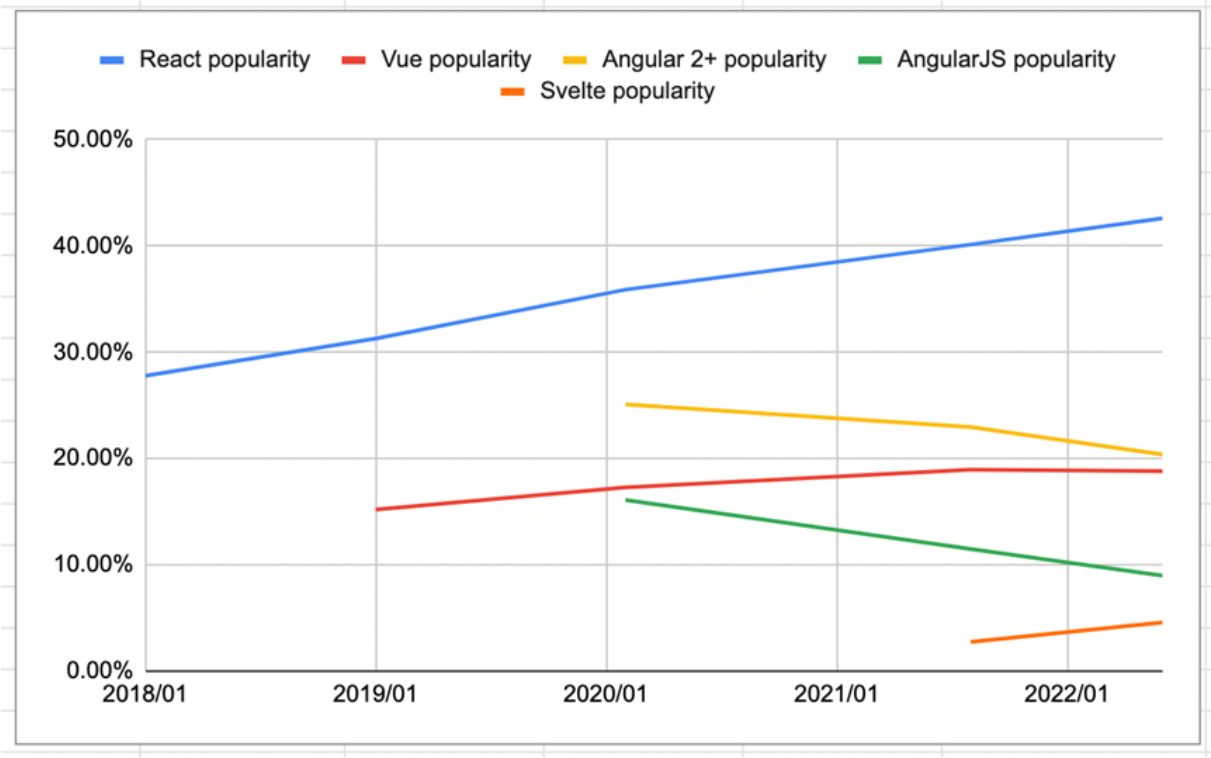



Figura 2 – Gráfico de popularidade das ferramentas front-end JavaScript no relatório do Stack Overflow 2022

Fonte: (STACK..., 2022)

### 1.2.1 Objetivo geral

O objetivo desta monografia é realizar um estudo quantitativo comparativo das principais ferramentas JavaScript - Angular, React e Vue - a fim de auxiliar no processo de seleção dessas ferramentas. O objetivo é fornecer uma análise das características de cada uma, com base em métricas e critérios considerados relevantes para o desenvolvimento de aplicações. O estudo tem como propósito estabelecer uma base comparativa, para auxiliar na escolha de qual pode ser a mais apropriada para o desenvolvimento.

Para atingir esse objetivo, estas foram avaliadas por meio da construção de um simples aplicativo de página única de controle financeiro, com foco apenas no desenvolvimento *front*, analisando o tempo de carregamento com diferentes quantidades de dados e utilizando também o JS *Framework Benchmark* nas comparações de *benchmarks* no contexto de manipulações do DOM.

Aplicativos de múltiplas páginas e *fullstack* (*back* e *front-end*) não foram abordados neste estudo. É importante destacar que este estudo se concentra exclusivamente na versão 2 e posterior do Angular, conhecida como Angular, enquanto as versões anteriores, conhecidas como AngularJS, estão fora do escopo da pesquisa.

### 1.2.2 Objetivos específicos

Cada ferramenta foi avaliada em relação a certos critérios e métricas. Para tal, foi realizado uma sequência de etapas:

1. Definição e testes de critérios de comparação entre as ferramentas;
2. Levantamento de informações sobre cada ferramenta de forma a embasar os resultados quantitativos em análises das comunidades envolvidas;
3. Para cada uma das ferramentas selecionadas, foi desenvolvida uma aplicação modelo de controle financeiro, com as mesmas funcionalidades e de forma simplificada, visando obter a máxima semelhança possível entre as implementações das ferramentas, com o objetivo de mensurar o tempo de carregamento da aplicação para diferentes quantidades de dados. Possibilitando a realização de uma comparação precisa do desempenho de cada ferramenta em relação ao tempo de carregamento.
4. Utilização de recursos específicos do *Benchmark* para JS Frameworks, na realização das comparações de desempenho no contexto de manipulações DOM.
5. Realização da comparação direta entre as ferramentas propostas, avaliando-as de acordo com as implementações, *benchmark* e critérios definidos.

# 2 Fundamentação Teórica

Esta seção apresenta uma visão geral da pesquisa sobre as áreas exploradas e ferramentas utilizadas.

## 2.1 Ferramentas de desenvolvimento front-end

As ferramentas de desenvolvimento desempenham um papel crucial no processo de criação de aplicativos, fornecendo uma estrutura pré-definida na qual os desenvolvedores podem construir seus projetos. Essas estruturas podem ser categorizadas como *frameworks* ou bibliotecas, embora possuam objetivos semelhantes, há diferenças significativas entre elas.

De acordo com E. Johnson (JOHNSON, 1997), um *framework* é um *design* reutilizável representado por um conjunto de classes abstratas, servindo como uma estrutura básica que pode ser customizada por desenvolvedores. Além disso, pode ser considerado uma aplicação "semi completa"(SCHMIDT et al., 2000).

E. Normand (NORMAND, 2022) destaca a importância de utilizar frameworks ou bibliotecas de desenvolvimento front-end para lidar com desafios comuns na construção de aplicações web. Esses desafios incluem a complexidade da web e das linguagens utilizadas, a necessidade de reconstruir repetidamente funcionalidades básicas, as vulnerabilidades de segurança e a tomada de várias decisões ao longo do processo de desenvolvimento. Utilizando ferramentas de desenvolvimento, é possível resolver esses problemas de forma mais eficiente, facilitando a construção e manutenção de aplicações web.

## 2.2 *Frameworks front-end* baseados em MVC

*Frameworks front-end* baseados em MVC simplificam o desenvolvimento, fornecendo regras claras para organizar o código e as camadas. Eles ajudam a dividir as responsabilidades entre as equipes de *front* e *back-end*, facilitando a colaboração e os testes de integração, utilizando a interface Ajax como ponto central de colaboração (POP; ALTAR, 2014).

O padrão MVC (Modelo-Visão-Controle) é amplamente utilizado no desenvolvimento de aplicações para organizar o código de forma estruturada. Ele divide a aplicação em três camadas: o modelo, que processa e acessa os dados da aplicação a visão que fornece a representação visual dos dados e o controle que coordena a interação entre o modelo e a visão (LEFF; RAYFIELD, 2001).

Resumidamente, o objetivo do *framework* MVC é simplificar o desenvolvimento *front-end*, fornecendo regras claras para organizar o código e as camadas, tornando as responsabilidades do código do *front* mais compreensíveis durante o processo de desenvolvimento e teste, contribuindo para a eficiência do trabalho e a qualidade do sistema resultante.

## 2.3 Desenvolvimento web

Aplicações web são programas desenvolvidos utilizando tecnologias como JavaScript, HTML e CSS, que são executados diretamente no navegador do usuário. Essas aplicações podem ser acessadas através de um navegador da web, sem a necessidade de instalação adicional, o que as torna fáceis de usar e acessíveis em diferentes sistemas operacionais. Essa abordagem tem sido amplamente adotada devido à sua praticidade, escalabilidade e facilidade de acesso.

- UX/UI: No desenvolvimento de sites, aplicações *web* e aplicativos móveis, a experiência do usuário (UX) e a *interface* do usuário (UI) desempenham um papel fundamental. Para Santos (SANTOS, 2019), UX é importante porque coloca a perspectiva do usuário como a espinha dorsal de qualquer fluxo de experiência. É preciso entender as necessidades, desejos e expectativas dos usuários para criar produtos que realmente os satisfaçam.

  A UI, por sua vez, abrange a parte visual e a usabilidade dos produtos, sendo uma parte integrante do UX, que engloba todas as interações e emoções, além do aspecto visual. Os usuários esperam facilidade, rapidez e uma experiência agradável ao interagir com as aplicações, enquanto os desenvolvedores buscam manutenibilidade e uma ampla gama de opções durante o processo de desenvolvimento (SANTOS, 2019).

- Aplicação de página-única (*Single Page Application* - SPA): é um conceito amplamente adotado, embora não haja uma definição precisa e única para ele. De acordo com A. Mesbah e A. Van Deursen, uma SPA é composta por componentes individuais que podem ser atualizados ou substituídos independentemente, sem a necessidade de recarregar a página inteira a cada interação do usuário (MESBAH; DEURSEN, 2007). Essa abordagem oferece uma experiência mais fluida e responsiva, permitindo uma interação mais dinâmica e rápida com o aplicativo web.

- Comunicação em SPA: A comunicação segue um fluxo específico. A figura abaixo (Figura 3) ilustra essa comunicação entre o usuário e o servidor *web*.

  1. Requisição inicial: o usuário faz uma solicitação HTTP para um URL específico.

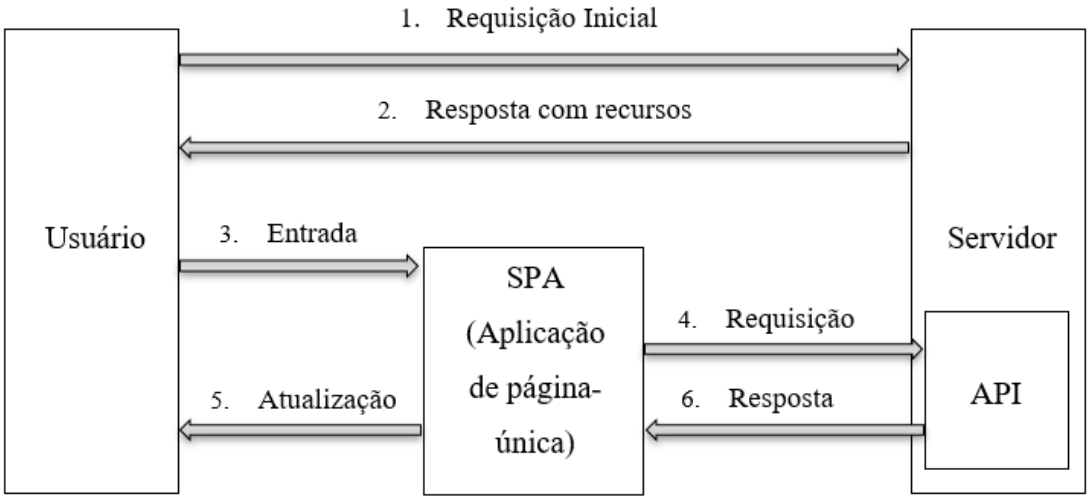



Figura 3 – Representação da comunicação SPA entre usuário e servido *web*. (MOLIN, 2016).

2. Resposta com recursos: O servidor *web* responde enviando as dependências JavaScript e bibliotecas adicionais necessárias para o SPA.
3. Entrada: o usuário interage com o SPA, realizando alterações em seu estado.
4. Requisição/solicitação: O SPA faz solicitações assíncronas à API do servidor para buscar dados ou executar ações.
5. Resposta: o servidor processa a solicitação e envia os resultados de volta ao SPA.
6. Atualização: com os novos dados recebidos, o SPA atualiza os componentes da interface de usuário, proporcionando uma experiência interativa e responsiva.

Esse fluxo ilustra como um SPA permite uma experiência de usuário mais fluida, evitando a necessidade de recarregar a página inteira a cada interação, tornando o aplicativo mais rápido e responsivo (GOCHKE, 2019).

- Vinculação de dados (*Data Binding*): A vinculação de dados, conhecida como *Data binding*, é uma técnica essencial que mantém a sincronia entre fontes de dados e informações em uma aplicação. Ela estabelece uma conexão entre a *interface* de usuário (UI) e a lógica de negócio, permitindo que os modelos e os componentes da visualização sejam automaticamente atualizados em tempo real.

  Em resumo, reflete os valores das variáveis dos componentes na página, sem a necessidade de manipulações diretas no DOM. Existem dois tipos:

  1. unidirecional, como ilustrado na figura 4, comum em aplicativos *web* tradicionais, onde as alterações nos dados não são automaticamente refletidas para o usuário,

para atualizar o modelo, é necessário que o usuário envie a visualização de volta ao servidor, onde as alterações são processadas e uma nova mesclagem é enviada de volta ao usuário (BORZEMSKI et al., 2014);

2. bidirecional, usado em SPAs, onde as alterações feitas pelo usuário são instantaneamente refletidas no modelo e vice-versa (Figura 4). A vinculação bidirecional é semelhante à arquitetura modelo-visão-controle (MVC) (FINK; FLATOW, 2014), explicada na seção 2.2.

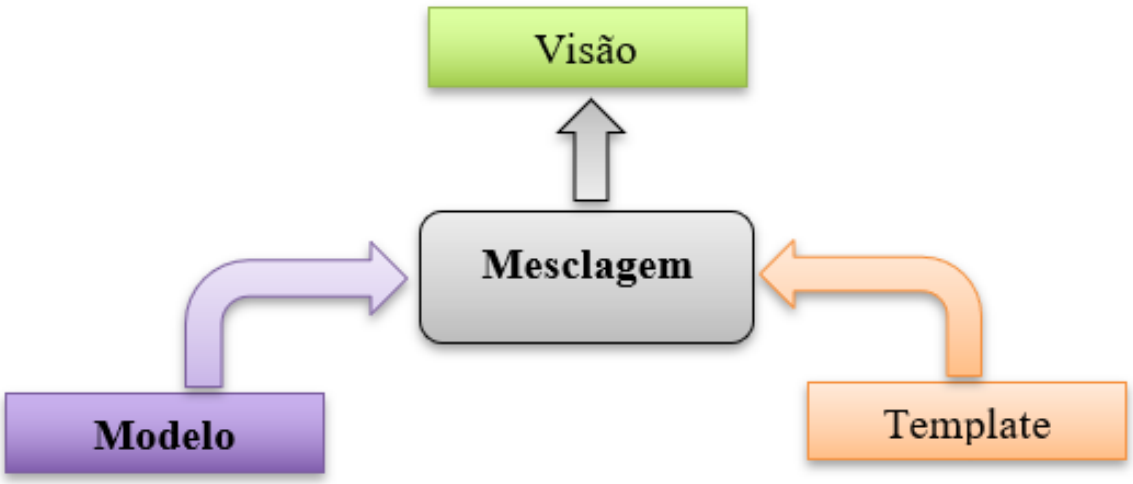



Figura 4 – Vinculação de dados unidirecional (*One-Way Data Binding*).

Fonte: (ALTEXSOFT, 2018).

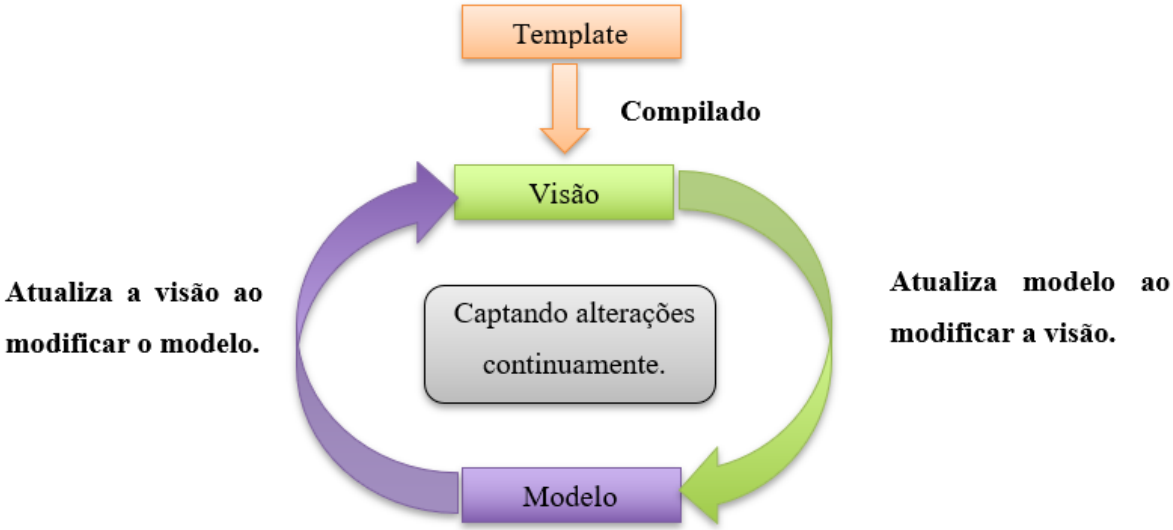



Figura 5 – Vinculação de dados bidirecional (*Two-Way Data Binding*).

Fonte: (ALTEXSOFT, 2018).

## 2.4 Trabalhos relacionados

Nesta seção, é apresentado exemplos relevantes de estudos relacionados ao tema de comparação e avaliação de ferramentas de desenvolvimento *web*. Essas pesquisas oferecem informações significativas que auxiliam na seleção e análise dessas tecnologias, abordando diversos aspectos relevantes.

Malmström (MALMSTRöM, 2014), apresenta um método abrangente de comparação de *frameworks Single Page Application* (SPA) escritos em JavaScript. O autor define sete requisitos-chave, como manutenibilidade, maturidade, desempenho e popularidade, e discute como cada um atende a esses critérios. Além disso, ele realiza um teste de desempenho em dois dos *frameworks* analisados, fornecendo dados adicionais sobre suas capacidades.

Gerdessen (GERDESSEN, 2007), propõe um método específico para comparar dois *frameworks* Java *back-end*. O autor coleta critérios relevantes da literatura e os adapta para se aplicarem a *frameworks* relacionados à *web*. Os critérios incluem escalabilidade, persistência, segurança, disponibilidade, customização, interoperabilidade, desempenho e separação de responsabilidades. Gerdessen realiza uma análise comparativa dos dois *frameworks* com base nesses critérios, fornecendo um referencial teórico sólido para a avaliação.

Salas-Zarate et al. (SALAS-ZÁRATE et al., 2015), apresentam uma lista de melhores práticas e requisitos para o desenvolvimento *web* e aplicam esses critérios na comparação de *frameworks web*. Eles consideram aspectos como desempenho, disponibilidade, usabilidade, segurança, manutenibilidade, portabilidade e confiabilidade. Para exemplificar a aplicação desses requisitos, os autores implementam um protótipo usando o *framework Lift* e o utilizam como base para a análise comparativa.

Ignacio Fernández-Villamor et al. (FERNÁNDEZ-VILLAMOR; DÍAZ-CASILLAS; IGLESIAS, 2008), propõem um modelo de comparação de frameworks web ágeis. Eles definem uma 'arquitetura *blueprint*' que inclui oito critérios importantes, como domínio, descrição e persistência, apresentação, segurança, usabilidade, teste, orientação a serviço, orientação a componentes e adoção. Os autores avaliam esses critérios por meio de um conjunto de perguntas, atribuindo uma pontuação para cada framework. Eles também consideram pesos para cada pergunta e conduzem uma discussão final para destacar os pontos fortes e fracos de cada *framework*.

Esses estudos contribuem para o campo de comparação de ferramentas de desenvolvimento web, fornecendo abordagens e critérios específicos para avaliar os diferentes aspectos destes.

# 3 Materiais e Métodos

Este capítulo visa expor e explicar os fundamentos e ferramentas exploradas neste trabalho, além de demonstrar como cada ponto é relacionado e utilizado na comparação da arquitetura de *software* e das respectivas ferramentas, assim como a explicação dos critérios de avaliações escolhidos e seus impactos e relevâncias.

## 3.1 HTML e CSS

HTML (*HyperText Markup Language*) é a base fundamental da *web*, sendo uma linguagem de marcação interpretada pelo navegador que define a estrutura e o significado do conteúdo de uma página da web. Para descrever a aparência e o estilo da página, outras tecnologias, como CSS (*Cascading Style Sheets*), são geralmente utilizadas (MINNICK, 2021).

CSS é uma linguagem composta por 'camadas' que define a apresentação e o layout de documentos desenvolvidos em linguagens de marcação, como XML, HTML e XHTML, que permite a formatação e estilização dos elementos da página (MINNICK, 2021).

## 3.2 JavaScript

JavaScript é uma linguagem de programação de alto nível, interpretada e orientada a objetos baseada no padrão ECMAScript, a qual, combina conceitos da programação funcional com a programação orientada a objetos e consiste em ECMAScript, DOM (*Document Object Model*) e o BOM (*Browser Object Model*). O ECMAScript define a sintaxe e as funcionalidades da linguagem, o DOM fornece uma interface para manipulação dos elementos de uma página *web*, e o BOM possibilita a interação com o navegador. Essa combinação de componentes torna o JavaScript uma linguagem poderosa, oferecendo simplicidade, eficiência, dinamismo, interatividade, multiplataforma, interpretabilidade e leveza (SAXENA et al., 2010).

O JavaScript manipula o DOM, criando interatividade e respondendo a eventos do usuário. Possui um ecossistema rico em bibliotecas e frameworks populares como Angular, React e Vue. É uma linguagem versátil usada no desenvolvimento web, permitindo a criação de páginas interativas, dinâmicas e responsivas com suporte ao DOM, AJAX e programação orientada a objetos (DevMedia, 2023). Além disso, seu ecossistema é rico em bibliotecas e *frameworks* populares, como Angular, React e Vue, que simplificam o desenvolvimento de *interfaces* de usuário avançadas e escaláveis.

### 3.2.1 Padrão ECMAScript

O ECMAScript é um padrão que define a especificação da linguagem JavaScript. Antes do JavaScript se tornar popular, seus criadores se associaram à ECMA (*European Computer Manufactures Association*) em 1996 para padronizar a linguagem. Como o nome 'JavaScript' já estava patenteado, decidiu-se utilizar o termo ECMAScript, resultando no nome oficial da tecnologia (LEE et al., 2012).

A padronização do ECMAScript é realizada pelo grupo ECMA-262, composto por empresas como Microsoft e Google. O grupo mantém os padrões e normativas da linguagem, que evolui por meio de diferentes versões, introduzindo novos recursos, melhorias na sintaxe e correções de bugs. É importante acompanhar as especificações para aproveitar as funcionalidades mais recentes e garantir a compatibilidade entre ambientes JavaScript, mantendo o código consistente e portátil (LEE et al., 2012).

### 3.2.2 DOM e BOM

DOM (*Document Object Model*) é uma interface de programação padronizada para documentos HTML e XML, permitindo que navegadores e scripts manipulem o conteúdo da web. Ele representa a estrutura lógica dos documentos como uma árvore de objetos. BOM (*Browser Object Model*) é uma hierarquia de objetos que permite ao JavaScript acessar e manipular objetos nativos do navegador, com propriedades e métodos úteis. A raiz da BOM é o objeto window, que representa a janela ou guias abertas e possui vários filhos (VOUTILAINEN; MIKKONEN; SYSTÄ, 2016).

### 3.2.3 TypeScript

Desenvolvido pela Microsoft, o TypeScript é uma extensão do JavaScript que adiciona recursos como tipagem estática e verificação de tipo em tempo de compilação, melhorando a detecção de erros e a experiência de desenvolvimento. Ele também oferece uma estrutura de classes completa, interfaces, módulos e funções de seta similares a funções *lambda*, inclui as últimas versões do ECMAScript e possui recursos exclusivos. Embora seja um *superset* do JavaScript todo o código TypeScript é convertido para JavaScript durante o processo de compilação (HERMAN; RAYCHEV; TOBIN-HOCHSTADT, 2015).

Em relação aos frameworks comparados, React e Vue utilizam JavaScript como sua linguagem principal, sem a necessidade de adicionar o TypeScript. Por outro lado, o Angular faz uso do TypeScript como sua linguagem principal de desenvolvimento.

### 3.2.4 AJAX

AJAX, ou *Asynchronous JavaScript and XML*, refere-se a um conjunto de tecnologias de desenvolvimento *web* no lado do navegador que permite a criação de aplicações *web* interativas e responsivas, ele utiliza a API *XMLHttpRequest* do JavaScript para enviar solicitações assíncronas ao servidor e receber dados em diferentes formatos, como XML e JSON (GARRETT, 2005).

Uma de suas vantagens é a capacidade de atualizar partes específicas de uma página da web, em vez de recarregar a página inteira, reduzindo a quantidade de dados transmitidos entre o navegador e o servidor, resultando em tempos de resposta mais rápidos e um uso mais eficiente da largura de banda (BALARAMAN, 2007).

Com o AJAX, os aplicativos da web podem enviar solicitações ao servidor em segundo plano, permitindo a atualização dinâmica da página com novos dados sem interromper a experiência do usuário. Isso desempenha um papel importante nas aplicações de página única (SPA), onde a maior parte do processamento ocorre no cliente e as interações com o servidor são realizadas por meio de solicitações AJAX. O desenvolvimento de ferramentas front-end, como Angular, React e Vue, facilitou a implementação de funcionalidades AJAX em aplicações web (PAULSON, 2005).

## 3.3 Ferramentas JavaScript

As ferramentas fornecem código padrão (*boilerplate*) e soluções pré-fabricadas para facilitar o desenvolvimento de projetos utilizando JavaScript (EDWIN, 2014).

### 3.3.1 Angular

O Angular surgiu como um projeto interno da Google em 2010, com o objetivo de facilitar o desenvolvimento de aplicações *web* mais complexas. Inicialmente lançado como AngularJS, o *framework* adotou a filosofia de programação declarativa, estendendo o HTML para tornar a criação de interfaces com conteúdo dinâmico mais fácil e eficiente. Com o tempo, o AngularJS ganhou popularidade e a colaboração de uma comunidade ativa de desenvolvedores contribuiu para seu crescimento e aprimoramento (BHASKAR; MANJUNATH, 2020).

No entanto, a evolução tecnológica e as demandas do mercado levaram a uma reescrita completa. Em 2016, foi lançada a versão Angular 2, que trouxe melhorias significativas e uma mudança fundamental: a adoção do TypeScript como linguagem principal. Essa decisão trouxe benefícios como a tipagem estática, a detecção de erros em tempo de compilação e a melhor organização do código (DZIWOKI, 2017).

O Angular utiliza uma sintaxe baseada em TypeScript, uma linguagem de programação que estende o JavaScript com recursos de tipagem estática. Essa abordagem traz benefícios importantes para o desenvolvimento, como a identificação antecipada de erros, facilitando a depuração e a manutenção do código. Além disso, segue uma arquitetura de componentes, onde cada componente é responsável por uma parte específica da interface do usuário. Esses componentes podem ser facilmente reutilizados e combinados para construir interfaces complexas, promovendo uma abordagem modular e escalável para o desenvolvimento (Angular, 2019).

Uma série de recursos o tornam uma escolha poderosa para o desenvolvimento front-end. Um desses recursos é o *two-way data binding*, que permite a sincronização automática de dados entre o modelo e a interface do usuário. Isso significa que qualquer alteração feita nos dados é refletida instantaneamente na interface, proporcionando uma experiência de usuário dinâmica e responsiva (FREEMAN, 2022).

O Angular oferece um sistema de injeção de dependências robusto, facilitando a criação e o gerenciamento eficiente das dependências entre os componentes, promovendo a reutilização e a manutenção de um código organizado e limpo. Além disso, possui suporte a roteamento, permitindo uma navegação suave entre diferentes partes da aplicação, sem a necessidade de recarregar a página. Essa funcionalidade resulta em uma experiência de usuário mais fluida e rápida, ele também possui uma arquitetura testável, permitindo que os desenvolvedores escrevam testes unitários e de integração para garantir a qualidade e a estabilidade da aplicação (FARWELL, 2017).

### 3.3.2 React

O React foi inicialmente desenvolvido pelo Facebook como uma solução interna para enfrentar os desafios do desenvolvimento de interfaces complexas e escaláveis. Em 2013, o Facebook lançou o React como uma biblioteca JavaScript de código aberto, permitindo que desenvolvedores de todo o mundo se beneficiassem dessa poderosa ferramenta. Desde então, tem crescido exponencialmente em popularidade e se tornou a base para muitas aplicações web modernas (FEDOSEJEV, 2016).

Faz uso de uma sintaxe chamada JSX, que combina elementos do JavaScript e do XML para definir a estrutura e o conteúdo da interface do usuário. Essa abordagem permite que os desenvolvedores criem componentes reutilizáveis, que combinam lógica e marcação em um único arquivo. Além disso, adota o conceito de 'componentização', onde a interface é dividida em componentes independentes e autocontidos, facilitando a manutenção e a reutilização de código (BODUCH, 2018).

Um dos principais recursos do React é o Virtual DOM, que possibilita atualizações eficientes na interface do usuário. O Virtual DOM é uma representação virtual da estrutura

da interface e o React o utiliza para calcular as diferenças entre o estado atual e o estado desejado da interface. Com isso, apenas as partes necessárias são atualizadas, resultando em um melhor desempenho e uma experiência de usuário mais fluida (AGGARWAL et al., 2018).

Outro aspecto importante é o conceito de '*props*' e '*state*'. As *props* são propriedades passadas para os componentes, permitindo a passagem de dados entre eles (React Documentation, 2019a). Já o *state* representa o estado interno de um componente e é utilizado para controlar as alterações na interface do usuário com base em eventos e interações do usuário. Além disso, adota o princípio da unidirecionalidade de dados, o que significa que os dados fluem de cima para baixo na hierarquia dos componentes, simplificando o rastreamento e o gerenciamento de dados, tornando o código mais previsível e fácil de manter (React Documentation, 2019b).

O React também possui uma comunidade ativa e uma vasta gama de bibliotecas e ferramentas adicionais, como o *React Router* para gerenciamento de rotas e o *Redux* para gerenciamento de estado global, permitindo que os desenvolvedores personalizem e ampliem o React de acordo com as necessidades de seus projetos (PRESTON; SO SO, 2018).

### 3.3.3 Vue

Desenvolvido por Evan You, um ex-engenheiro do Google, o Vue foi lançado em 2014 com o objetivo de ser uma alternativa mais leve e fácil de aprender em comparação a outros frameworks front-end, que rapidamente ganhou popularidade devido à sua abordagem progressiva, o que significa que é possível adotar gradualmente suas funcionalidades em projetos existentes (SILVA, 2021) .

A sintaxe é baseada em templates declarativos, combinando HTML e JavaScript em um formato chamado *Vue Template Syntax*. Essa sintaxe intuitiva permite que os desenvolvedores criem componentes reativos e interativos com facilidade. Além disso, utiliza o conceito de '*bindings*' (vínculos) para conectar os dados do componente à interface do usuário de forma dinâmica, permitindo que as alterações nos dados sejam refletidas automaticamente na interface (SAKS, 2019).

Um dos aspectos marcantes é sua capacidade de reatividade, que monitora automaticamente as alterações nos dados e atualiza a interface do usuário de forma eficiente, sem a necessidade de intervenção manual. Isso proporciona uma experiência de desenvolvimento suave e uma interface responsiva para os usuários, além da capacidade de criar componentes reutilizáveis. Os componentes são blocos independentes de código que podem ser combinados para formar uma interface complexa. Com o Vue, os desenvolvedores podem facilmente criar, compartilhar e reutilizar componentes em diferentes partes do projeto,

resultando em um código mais limpo e modular (KYRIAKIDIS; MANIATIS; YOU, 2016).

Também oferece suporte ao conceito de '*computed properties*' (propriedades computadas), que permite derivar novos valores com base nos dados existentes. Essas propriedades computadas são atualizadas automaticamente sempre que os dados subjacentes mudam, garantindo que a interface do usuário esteja sempre sincronizada com o estado da aplicação (STEYER, 2022).

Além disso, possui uma documentação abrangente e uma comunidade ativa, o que facilita o aprendizado e o suporte para os desenvolvedores. A vasta gama de plugins e extensões disponíveis também amplia as funcionalidades do Vue, permitindo que ele seja personalizado para atender às necessidades específicas de cada projeto

## 3.4 Critérios de análise

A presente seção apresenta a lista (não exaustiva) de critérios de análise já documentados pela literatura relevante na área.

### 3.4.1 Popularidade

A popularidade de uma ferramenta é um indicador importante, pois demonstra sua aceitação e adoção pela comunidade de desenvolvedores, resultando em uma comunidade engajada que oferece suporte, compartilha conhecimento e contribui para o desenvolvimento de recursos adicionais. Além disso, uma ferramenta popular geralmente tem uma base de usuários maior, influenciando positivamente na disponibilidade de talentos e oportunidades de colaboração (SMITH, 2020).

### 3.4.2 Maturidade e Estabilidade

Ao selecionar uma ferramenta de desenvolvimento *front-end*, a maturidade e estabilidade são aspectos fundamentais a serem considerados. Esses atributos indicam a confiabilidade, segurança e capacidade de lidar com projetos reais e complexos (WENDLER, 2012). A preferência de grandes empresas e organizações renomadas também pode ser um indicador da qualidade e robustez do framework (DUARTE, 2015).

### 3.4.3 Curva de Aprendizado

A curva de aprendizado de um framework é um fator importante a ser considerado ao escolher uma tecnologia para o desenvolvimento, uma curva suave indica que os desenvolvedores podem aprender e se familiarizar rapidamente com o framework, começando a contribuir para projetos mais facilmente (CHEN, 2020).

Para isso, é essencial que o framework tenha uma sintaxe clara e intuitiva, uma arquitetura bem documentada e recursos educacionais disponíveis, como tutoriais e documentação abrangente. Além disso, um bom gerenciamento de dados e a disponibilidade de bibliotecas externas também podem facilitar a aprendizagem **(SMITH, 2020)**. É importante considerar as habilidades e experiências dos desenvolvedores, bem como analisar a documentação técnica do framework, para determinar a suavidade da curva de aprendizado.

Segundo Johnson et al. (JOHNSON; SMITH; DAVIS, 2019), frameworks que possuem uma estrutura bem documentada e abstrações eficientes para o gerenciamento de dados podem facilitar esse processo. Além disso, a existência e a utilização de bibliotecas externas influenciam o aprendizado (GHAZARYAN, 2023). É importante ressaltar que a análise da curva de aprendizado é um critério subjetivo e particular, e os resultados podem variar de acordo com as habilidades e experiências individuais dos desenvolvedores.

### 3.4.4 Desempenho

O desempenho é um fator crítico para a experiência do usuário e a eficiência operacional. Conforme destacado por Lee et al. (LEE et al., 2012), uma ferramenta com bom desempenho oferece tempos de carregamento rápidos, uma interface responsiva e baixo consumo de recursos do sistema, como memória e processamento. Resultando em uma experiência do usuário mais fluida, menor latência e maior eficiência energética. Avaliar o desempenho de uma ferramenta é importante para garantir que ele atenda aos requisitos de desempenho do projeto e ofereça uma experiência satisfatória aos usuários finais, além de determinar sua eficiência e capacidade de lidar com cargas de trabalho exigentes.

# 4 Metodologia

## 4.1 Popularidade

Esta pode ser avaliada por meio de métricas como estrelas e *forks* no GitHub, *downloads* de pacotes NPM, atividade no Stack Overflow, demanda no mercado de trabalho, comunidade *online*, adoção pela indústria e métricas do HackerRank e Google Trends. Essas métricas fornecem dados sobre o engajamento da comunidade, o uso e a aceitação pelos desenvolvedores. Ao analisá-las em conjunto, é possível ter uma visão abrangente da popularidade.

## 4.2 Maturidade e Estabilidade

Embora a maturidade e a estabilidade sejam características subjetivas, algumas métricas quantitativas podem auxiliar nessa avaliação. O número de lançamentos, a atividade de commits, as estrelas e forks no GitHub, a atividade no Stack Overflow, os downloads de pacotes no NPM, os problemas abertos e solicitações de pull, e a adoção por empresas renomadas são exemplos dessas métricas.

No entanto, é importante considerar essas métricas em conjunto com fatores qualitativos, como a qualidade da documentação e o suporte da comunidade. Além disso, a escolha do framework ideal também depende dos requisitos e preferências específicas do projeto e da equipe de desenvolvimento.

## 4.3 Curva de Aprendizado

É importante ressaltar que a análise da curva de aprendizado é um critério subjetivo e particular, e os resultados podem variar de acordo com as habilidades e experiências individuais dos desenvolvedores. A comparação dos requisitos de conhecimento necessários deve ser baseada em uma análise cuidadosa das informações disponíveis na documentação técnica de cada framework (XING; HUANG; LAI, 2019).

A curva de aprendizado foi avaliada comparando a sintaxe do framework, arquitetura, gerenciamento de dados, ciclo de vida e facilidade de uso de bibliotecas de terceiros. As informações para isso foram coletadas a partir dos documentos técnicos de cada ferramenta e também de materiais disponíveis sobre o tema, foram consultados vários artigos, cursos e as documentações oficiais de cada uma dessas tecnologias. Essa abordagem visa fornecer uma visão mais completa e objetiva sobre as necessidades de conhecimento prévio

ao trabalhar com cada uma delas.

## 4.4 Desempenho

Os aspectos mensurados foram: análise de resultados obtidos nos testes realizados na aplicação modelo implementada, análise dos resultados coletados pelo benchmark e pelo aplicativo web Perf Track.

### 4.4.1 Aplicação modelo

Foi implementado uma aplicação simples que envolve o carregamento de dados ao iniciar nas três ferramentas, das quais foram coletadas informações sobre o tempo necessário para carregamento, bem como as métricas de linhas de código e tamanho de pacote.

Para medir o tempo de carregamento de dados, foi utilizado o Lighthouse, uma ferramenta de código aberto desenvolvida pelo Google, amplamente reconhecida como uma referência para a análise de desempenho de aplicativos da web.

Com o Lighthouse, foi realizado uma auditoria de desempenho na aplicação desenvolvida em cada uma das ferramentas: Angular, React e Vue. Essa auditoria fornecerá métricas detalhadas sobre o tempo de carregamento da página, incluindo o tempo necessário para carregar e processar recursos como JavaScript, CSS e imagens.

Além do tempo de carregamento, o Lighthouse também oferece informações sobre outras métricas de desempenho, como o tempo necessário para interatividade da página (*First Contentful Paint*, *Time to Interactive*) e a velocidade de renderização (*Speed Index*). Essas métricas auxiliaram na comparação do desempenho entre as tecnologias.

Ao utilizar o Lighthouse, foi possível obter resultados consistentes e confiáveis, pois a ferramenta realiza testes automatizados em um ambiente controlado. Garantindo medições precisas e comparáveis entre as ferramentas, possibilitando uma avaliação objetiva e padronizada do desempenho das aplicações desenvolvidas com Angular, React e Vue.

Uma métrica comumente utilizada é a contagem de linhas de código do framework. Comparar o tamanho do código fonte entre os frameworks pode fornecer insights sobre sua complexidade e possíveis impactos no desempenho. Menos linhas de código podem indicar uma estrutura mais enxuta e uma maior eficiência na execução do software.

Neste estudo comparativo, foi utilizado a ferramenta de estatísticas de código Cloc para contar as linhas de código. Cloc conta as linhas de código na pasta src de cada repositório sem contar as linhas em branco e linhas de comentários. Linhas de código indicam quão concisa é uma determinada biblioteca/framework/linguagem (ALDANIAL, 2021).

O tamanho dos pacotes gerados pelos frameworks é outro fator importante a ser considerado. Frameworks com pacotes menores tendem a ter um impacto menor no tempo de carregamento da página e na utilização de recursos do dispositivo do usuário. Analisar o tamanho dos arquivos JavaScript e CSS gerados pelos frameworks pode ajudar a identificar diferenças significativas em termos de eficiência e desempenho.

Bundlephobia é uma ferramenta *online* que fornece informações sobre o tamanho e o impacto de pacotes npm. Ao carregar o arquivo package.json de projetos front-end no site, ele calcula automaticamente o tamanho das dependências (BUNDLEPHOBIA. . . , 2023). Essa ferramenta foi utilizada como referência para analisar o tamanho das dependências das ferramentas em estudo.

Essa abordagem permite uma comparação direta em relação ao desempenho, fornecendo informações sobre o tempo de carregamento, concisão de código e o tamanho dos pacotes. Essas métricas contribuíram para a análise comparativa do desempenho das ferramentas Angular, React e Vue.

### 4.4.2 Benchmarks Local DOM

Para complementar a análise de desempenho, foi utilizado o benchmark desenvolvido por Krause (KRAUSE, 2021). Esse benchmark é uma ferramenta poderosa e precisa para a análise comparativa do desempenho, com foco nas operações de manipulação do DOM.

O *benchmark* utiliza uma abordagem baseada na criação de uma tabela com entradas aleatórias, que serve como cenário para medir o tempo necessário para realizar diferentes operações de manipulação do DOM. Uma característica importante é o uso de uma linha do tempo personalizada, que registra dados brutos de desempenho, e fazendo o uso da estrutura Aurelia para aguardar o gatilho do evento antes de atualizar e renderizar o DOM, garantindo uma sincronização adequada e evitando resultados imprecisos, permitindo uma avaliação mais precisa do tempo de resposta e da eficiência das ferramentas em relação às operações de DOM.

Ao utilizar o *benchmark* de Krause, foi possível obter métricas confiáveis e comparáveis dos desempenhos em termos de manipulações do DOM, permitindo uma análise aprofundada das diferenças de desempenho entre Angular, React e Vue, fornecendo informações valiosas sobre a eficiência e a capacidade de resposta de cada um em relação às operações de DOM.

### 4.4.3 Desempenho de Página de Website

O desempenho de uma página de *website* é uma medida da eficiência e velocidade com que a página é carregada e exibida no navegador. Um bom desempenho de página é

fundamental para proporcionar uma experiência positiva ao usuário, reduzir a taxa de rejeição e melhorar os resultados de busca.

Foi utilizado o aplicativo web Perf Track, criado pelo Google Chrome Labs (Google Chrome Labs, 2020), que coleta tamanhos de amostra de URLs de sites com base na linguagem de programação utilizada para construí-los e disponibiliza relatórios com vários dados para análise.

# 5 Resultados

Esta seção, apresenta os resultados e discussões das comparações dos critérios propostos.

## 5.1 Análise Comparativa dos Critérios

A análise comparativa dos critérios permite uma avaliação objetiva e fundamentada das opções disponíveis, facilitando a tomada de decisões informadas.

### 5.1.1 Popularidade / Maturidade e Estabilidade

A popularidade é um dos fatores que indicam a adoção pela comunidade, enquanto a maturidade refere-se ao seu desenvolvimento ao longo do tempo e a estabilidade um dos critérios que apontam a confiabilidade e consistência da tecnologia, como explicado na seção 3.4.1.

#### 5.1.1.1 Google Trends

Ao comparar a popularidade do Angular, React e Vue usando o Google Trends, podemos obter informações sobre o interesse de busca por essas ferramentas. É importante ressaltar que a mensuração é feita pela popularidade das buscas e não indica diretamente o uso ou adoção real, especialmente quando se trata de aplicativos de grande porte e empresariais.

Dito isso, os dados do Google Trends mostram que há um interesse significativo nas três tecnologias. O volume de buscas por Angular, React e Vue indica que os desenvolvedores estão ativamente buscando informações e recursos relacionados.

O gráfico ilustrado na Figura 6 apresenta a popularidade relativa ao longo do tempo das palavras-chave 'React', 'Angular' e 'Vue' nas buscas da web. Ele permite uma comparação do interesse das pessoas por esses termos nos últimos 5 anos. Durante esse período, o React tem mantido uma posição de destaque, com alta popularidade e busca na web. O Angular também demonstra um nível estável de interesse, embora ligeiramente inferior ao React. Por sua vez, o Vue apresenta um crescimento significativo, indicando um aumento de interesse e adoção por parte dos desenvolvedores. A tabela 1, ilustra o resumo dessa comparação, destacando o React com alta popularidade, o Angular com popularidade moderada e o Vue com crescimento significativo.

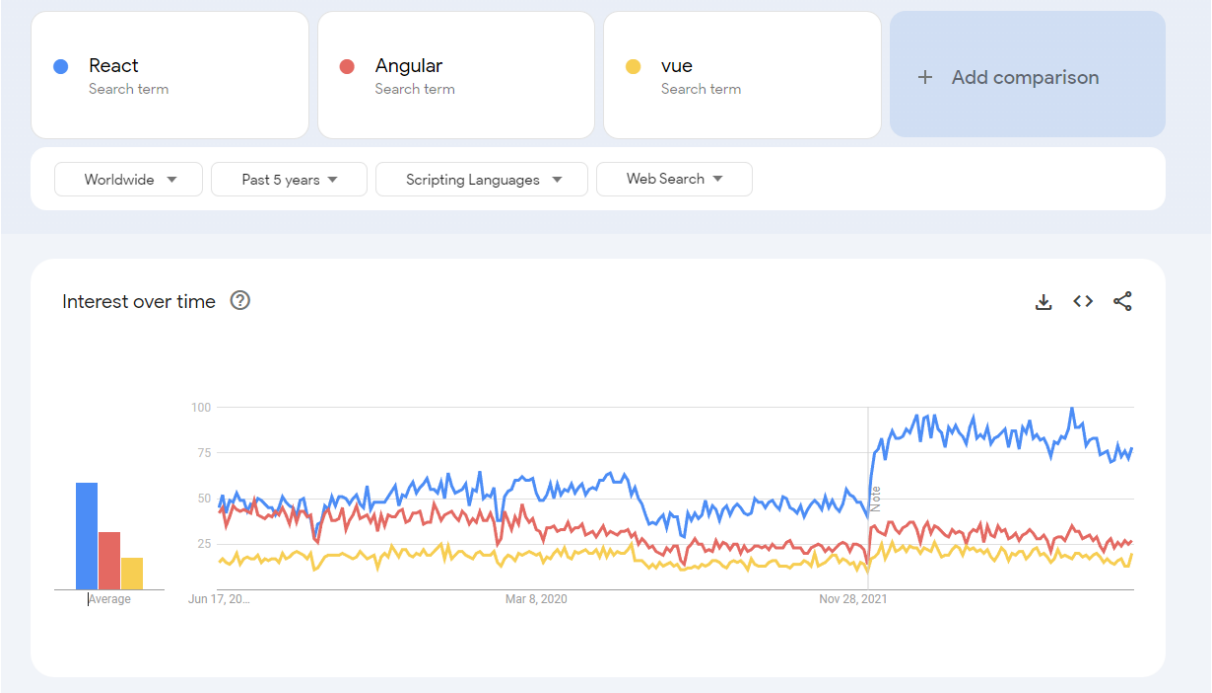



Figura 6 – Gráfico de popularidade relativa ao longo do tempo das palavras-chave 'React', 'Angular' e 'Vue' nas buscas da web.

Fonte: (TRENDS, 2023a)

Tabela 1 – Popularidade das Ferramentas - Google Trends

| Ferramenta | Popularidade | Busca na Web |
|---|---|---|
| Angular | Moderada | Moderada |
| React | Alta | Alta |
| Vue | Crescente | Moderada |

#### 5.1.1.2 NPMTrends

A fonte dos dados exibidos no gráfico ilustrado pela figura 7 é o site NPMTrends, que é uma referência utilizada para analisar a popularidade relativa das ferramentas, mensurada com base no número de downloads semanais dos pacotes npm associados a cada uma. Esses dados são coletados e apresentados visualmente, permitindo uma comparação direta entre as ferramentas em termos de sua adoção e uso pela comunidade de desenvolvedores. Observando o gráfico, podemos fazer algumas observações:

O React mantém uma alta popularidade, com uma tendência de crescimento contínuo ao longo do tempo. O Angular, embora tenha sido popular no passado, mostra uma tendência de estagnação ou ligeira diminuição em comparação com o React. O Vue, inicialmente menos popular, está experimentando um crescimento ao longo do tempo e se aproximando cada vez mais em popularidade.

Essas observações apontam o React como o mais utilizado, seguido pelo Angular e Vue. No entanto, o Vue está ganhando popularidade rapidamente e se tornando uma

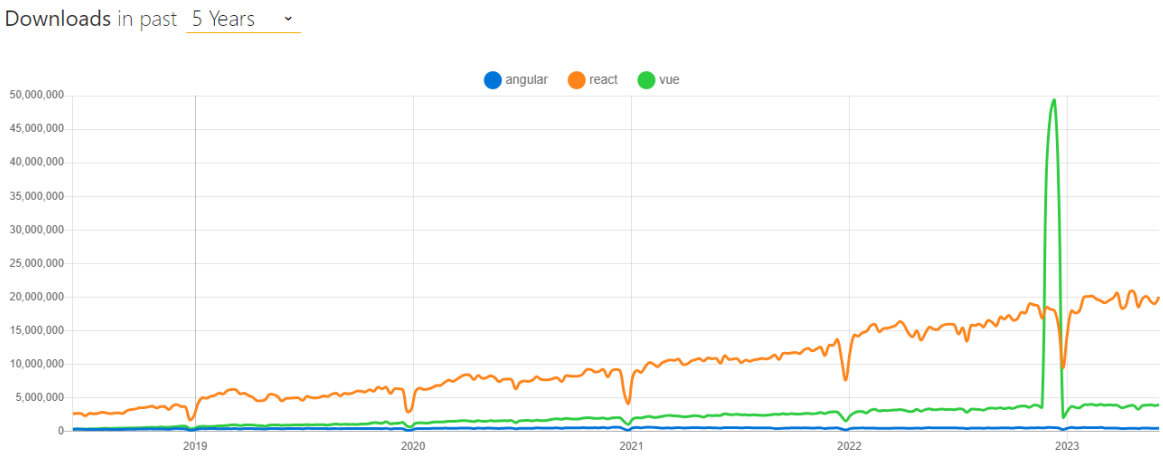



Figura 7 – Comparação da popularidade dos frameworks Angular, React e Vue ao longo do tempo, com base nos downloads semanais dos pacotes npm.

Fonte: (TRENDS, 2023b)

opção competitiva.

O site NPMTrends, além de exibir a popularidade relativa dos frameworks Angular, React e Vue, também fornece estatísticas comparativas sobre o uso desses frameworks no contexto do Git, como ilustrado na figura 8, da qual, podemos fazer algumas análises estatísticas de cada uma.

Stats

| | Stars | Issues | Version | Updated | Created | Size |
|---|---|---|---|---|---|---|
| angular | 59,142 | 464 | 1.8.3 | a year ago | 11 years ago | minzipped size 62.3 KB |
| react | 209,475 | 1,337 | 18.2.0 | a year ago | 12 years ago | minzipped size 2.5 KB |
| vue | 38,394 | 1,062 | 3.3.4 | a month ago | 10 years ago | minzipped size 34.6 KB |

Figura 8 – Tabela de comparação de das ferramentas no contexto do Git, com base nos dados do NPMTrends

Fonte: (TRENDS, 2023b)

Com base nas informações fornecidas, podemos fazer algumas análises sobre as estatísticas das ferramentas Angular, React e Vue:

- Popularidade relativa: React é o mais popular, com 209.897 estrelas no GitHub, seguido pelo Angular com 59.142 estrelas e Vue com 38.394 estrelas. Sugerindo React como o mais adotado pela comunidade de desenvolvedores em comparação com os outras duas ferramentas.
- Atividade e envolvimento da comunidade: React possui o maior número de problemas abertos (1.301), seguido pelo Angular (464) e Vue (1.037). Isso pode indicar um

alto envolvimento da comunidade em relação ao React, com muitas discussões e interações em torno do framework.

- Versões e atualizações: O React possui a versão mais recente 18.2.0, seguido pelo Angular na versão 1.8.3 e o Vue na versão 3.3.4. Esses números sugerem que o React pode ter um ritmo de atualizações mais frequente em comparação aos outros.
- Idade dos projetos: O React e o Vue têm uma história de mais de 10 anos, com React sendo criado há 12 anos e Vue há 10 anos. O Angular é um pouco mais antigo, com 11 anos de existência. Essa longevidade pode indicar a maturidade e a estabilidade desses frameworks.

| Framework | Popularidade Atual |
|---|---|
| Angular | Moderada |
| React | Alta |
| Vue | Crescente |

Tabela 2 – Tabela de comparação sucinta dos frameworks.

Nesta tabela 2, as ferramentas são listadas com base em sua popularidade atual, conforme indicado pelos dados dos gráficos. O React é considerado altamente popular, enquanto o Angular possui uma popularidade moderada e o Vue está em ascensão.

#### 5.1.1.3 Stack Overflow *Trends*

O gráfico do Stack Overflow *Trends* para as tags 'reactjs', 'vue.js' e 'angular' mostra a variação no interesse relativo dessas tecnologias ao longo do tempo, com base no número de perguntas feitas no Stack Overflow (Figura 9).

Podemos observar que o React tem sido a tecnologia mais popular entre as três, com um crescimento constante ao longo dos anos, mantendo uma liderança em termos de interesse e envolvimento da comunidade.

O Angular, por sua vez, também demonstra um nível estável de interesse, embora em um patamar um pouco abaixo do React. Ele ainda mantém uma base sólida de perguntas feitas pelos desenvolvedores.

Já o Vue apresenta um crescimento notável em termos de interesse no Stack Overflow, sendo um indicativo de aumento de interesse e adoção por parte dos desenvolvedores.

#### 5.1.1.4 Ofertas de Trabalho

Com base nos dados obtidos para as plataformas Indeed, LinkedIn e Glassdoor no Brasil no dia 12/06/2023, ilustrado na figura 10, podemos observar que Angular e React

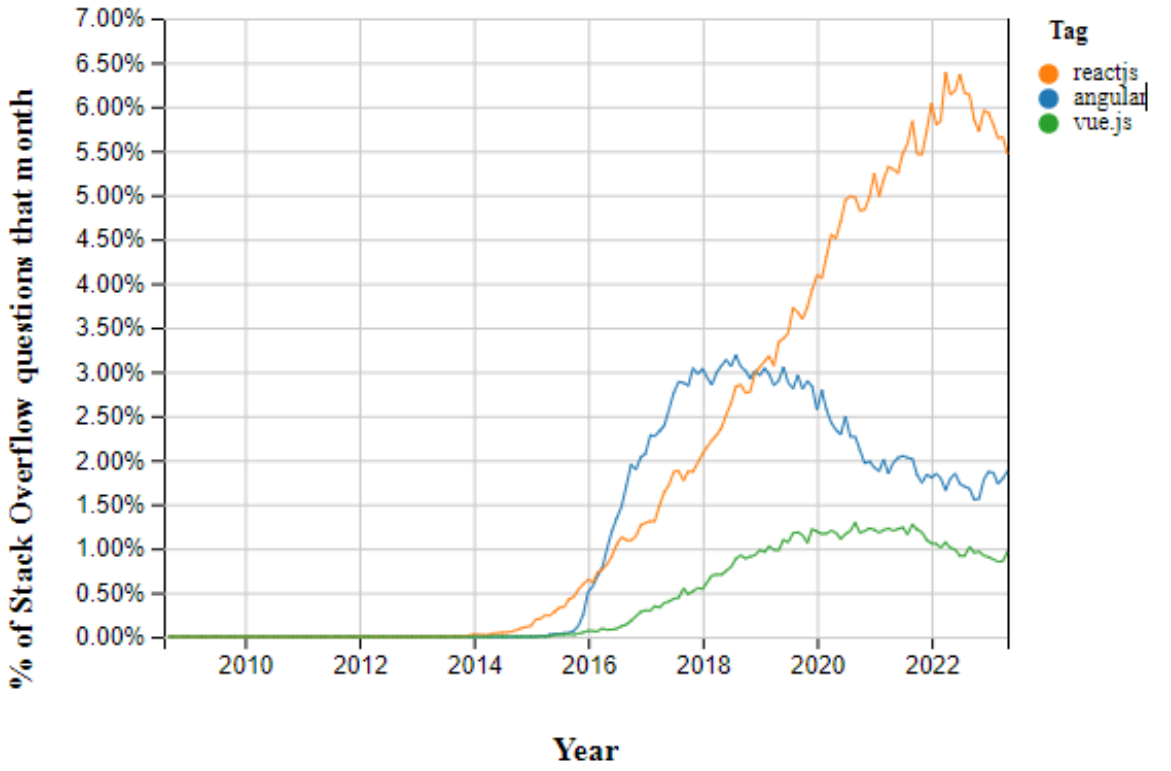



Figura 9 – Legenda: Gráfico de interesse relativo das tecnologias 'React.js', 'Vue.js' e 'Angular' ao longo do tempo, baseado no número de perguntas feitas no Stack Overflow.

Fonte: (STACK..., 2022)

apresentam uma maior demanda no momento. O Vue, embora em menor medida, também mostra uma presença e demanda por profissionais.

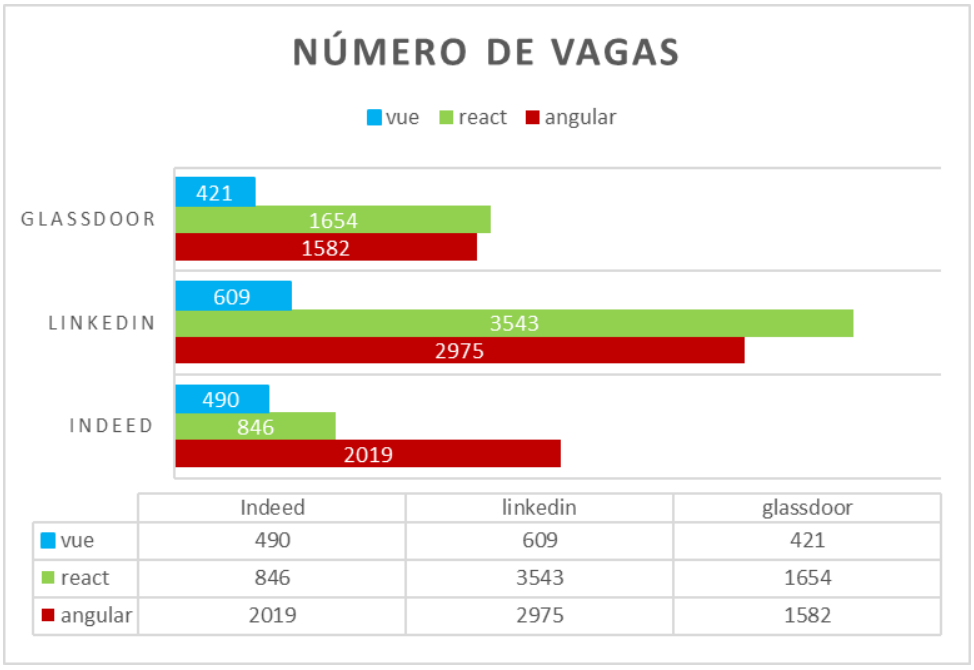
É importante ressaltar que esses dados são específicos do período de 12/06/2023 e se referem ao contexto brasileiro, podendo sofrer variações ao longo do tempo.

#### 5.1.1.5 Github Stats

Para a análise das estatísticas do GitHub, foi utilizado o website https://vesoft-inc.github.io/github-statistics/. Essa plataforma fornece métricas abrangentes sobre repositórios hospedados no GitHub, permitindo uma comparação precisa da popularidade e atividade das tecnologias, os dados em questão, foram consultados dia 12/06/2023. (INC., 2023).

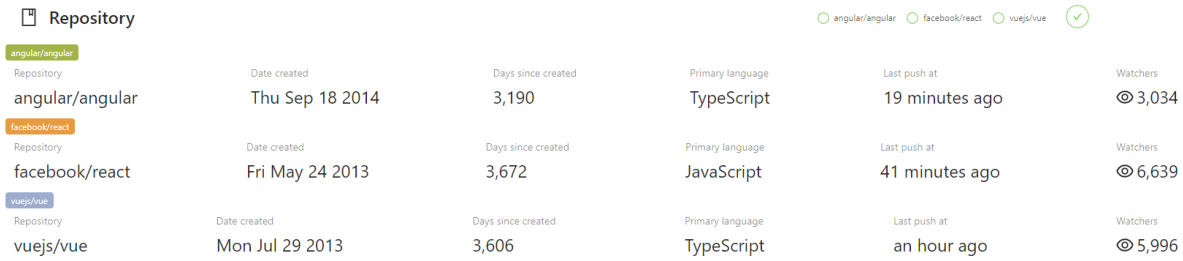
##### 5.1.1.5.1 Repository

A figura 11, apresenta informações relevantes sobre os repositórios do Angular, React e Vue, no GitHub, incluindo a data de criação, o número de dias desde a criação, a linguagem principal utilizada, a data do último *push* e o número de observadores (*watchers*).




| | Indeed | linkedin | glassdoor |
|---|---|---|---|
| vue | 490 | 609 | 421 |
| react | 846 | 3543 | 1654 |
| angular | 2019 | 2975 | 1582 |

Figura 10 – Pesquisa de ofertas de trabalho Angular vs React vs Vue, contexto brasileiro



Repository

| Repository | Date created | Days since created | Primary language | Last push at | Watchers |
|---|---|---|---|---|---|
| angular/angular | Thu Sep 18 2014 | 3,190 | TypeScript | 19 minutes ago | 3,034 |
| facebook/react | Fri May 24 2013 | 3,672 | JavaScript | 41 minutes ago | 6,639 |
| vuejs/vue | Mon Jul 29 2013 | 3,606 | TypeScript | an hour ago | 5,996 |

Figura 11 – Comparação dos dados dos repositórios do Angular, React e Vue no GitHub

Fonte: (INC., 2023)

Esses dados permitem uma análise comparativa dos aspectos temporais, linguagem de programação e engajamento da comunidade em relação a cada ferramenta, de acordo com os dados ilustrados, podemos observar que:

- Idade e tempo de existência do repositório: o React é o mais antigo, tendo seu repositório criado em maio de 2013, seguido pelo Vue em julho de 2013 e pelo Angular em setembro de 2014.
- Linguagem principal: O Angular e o Vue são escritos em TypeScript, enquanto o React é escrito em JavaScript. O TypeScript é uma linguagem que adiciona recursos de tipagem estática ao JavaScript, o que pode oferecer vantagens em termos de segurança, detecção de erros e produtividade para equipes de desenvolvimento maiores. No entanto, o JavaScript continua sendo uma linguagem amplamente adotada e bem estabelecida na comunidade de desenvolvimento.

- Atividade de desenvolvimento: Todos os três frameworks têm um histórico recente de atividade de desenvolvimento, com *commits* (*pushes*) ocorrendo dentro de uma hora ou menos. Isso indica que os projetos estão ativamente sendo mantidos e atualizados pela comunidade de desenvolvedores.
- Observadores (*watchers*): O número de observadores em um repositório pode indicar o nível de interesse e suporte da comunidade. O React possui o maior número de observadores, com 6.639, seguido pelo Vue com 5.996 e o Angular com 3.034. Isso sugere que o React tem uma base de usuários maior e uma comunidade mais ativa, enquanto o Angular tem menos observadores em comparação.

#### 5.1.1.5.2 Commits

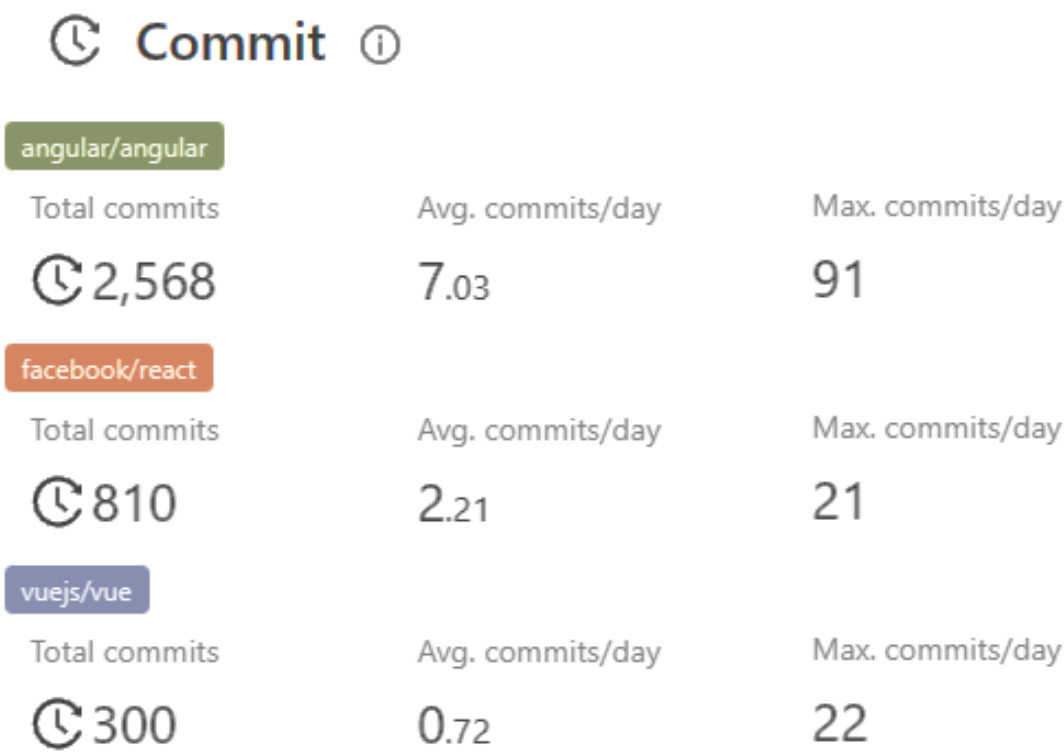



Figura 12 – Comparação dos dados de commits nos repositórios do Angular, React e Vue

Fonte: (INC., 2023)

Ao analisar os dados de *commits* nos repositórios do Angular, React e Vue, ilustrados na figura 12, que apresenta informações sobre o número total, a média por dia e o máximo em um único dia para cada ferramenta (Angular, React e Vue), podemos observar o seguinte:

Angular: Possui um total de 2.568 *commits*, com média de 7,03 por dia e um máximo de 91 em um único dia. Indicando um alto nível de atividade e engajamento no desenvolvimento.

React: Tem um total de 810 *commits*, com média de 2,21 por dia e um máximo de 21 em um único dia. Apresentando uma atividade consistente, porém em menor escala em comparação ao Angular.

Vue: Apresenta 300 *commits* no total, com média de 0,72 por dia e um máximo de 22 em um único dia. Apontando uma atividade mais baixa em comparação as outras duas ferramentas.

Essas informações sugerem diferentes níveis de atividades e engajamentos nos desenvolvimentos, com o Angular liderando em termos de quantidade e frequência de commits, seguido pelo React e, por último, o Vue, como resumido na figura 13.

| | Média Commits por Dia | Máx Commits em um Dia | Total de Commits |
|---|---|---|---|
| Angular | 7.03 | 91 | 2568 |
| React | 2.21 | 21 | 810 |
| Vue | 0.72 | 22 | 300 |

Figura 13 – Comparação entre as ferramentas relacionado a quantidade de commits

#### 5.1.1.5.3 Estrelas

O site 'star-history.com' apresenta o histórico de estrelas dos repositórios no GitHub ao longo do tempo. Ao comparar o histórico de estrelas dessas três ferramentas, é possível ter uma visão geral da sua popularidade relativa ao longo do tempo. Isso pode ajudar na avaliação da preferência da comunidade, na adoção pelos desenvolvedores e no crescimento de cada um dos frameworks.

O gráfico exibido na figura 14, mostra a tendência de aumento ou diminuição das estrelas recebidas por cada um dos repositórios desde 2014 até 2022. As estrelas são uma métrica utilizada para medir a popularidade de um projeto no GitHub, indicando o interesse e o engajamento da comunidade de desenvolvedores.

O React apresenta um crescimento exponencial nas estrelas do repositório, sendo o framework mais popular. O Angular mantém um crescimento estável, indicando uma base de usuários consistente. O Vue mostra um crescimento gradual e constante, ganhando popularidade ao longo do tempo. A análise do histórico de estrelas reflete a preferência da comunidade e o interesse dos desenvolvedores por esses frameworks.

Embora o gráfico de estrelas não forneça informações diretas sobre a maturidade de um framework, a presença de um histórico de estrelas consistente e um desenvolvimento contínuo indicam uma maior maturidade. Tanto o Angular quanto o React mostram um crescimento estável e consistente no número de estrelas, o que sugere que esses frameworks têm sido amplamente utilizados e refinados ao longo do tempo. O Vue, embora tenha

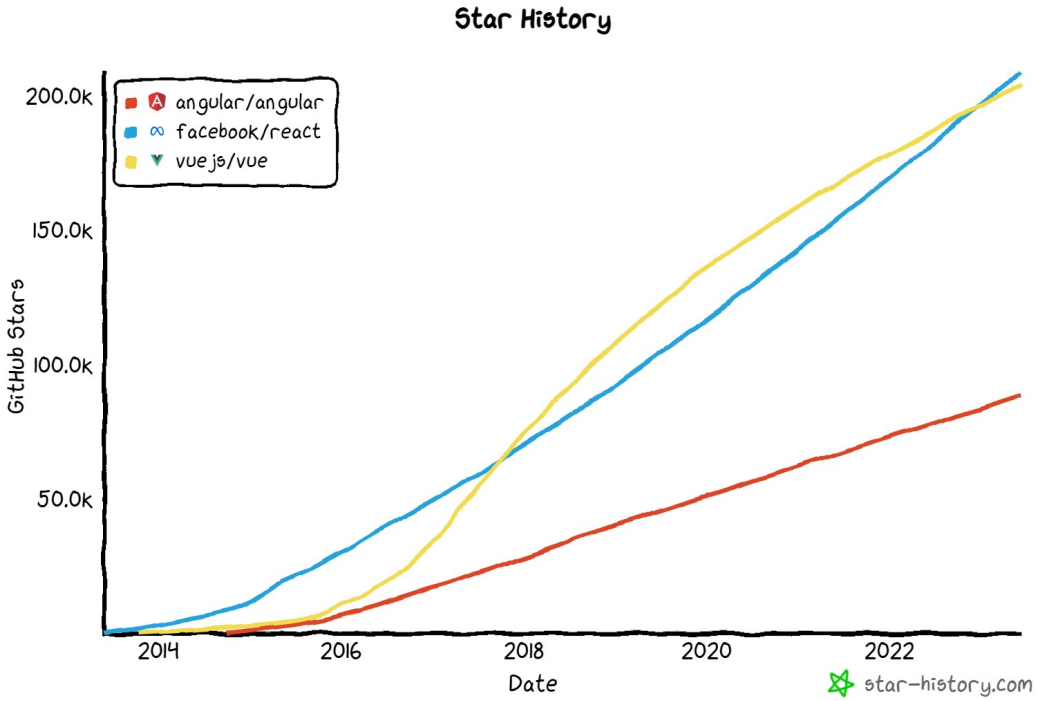



Figura 14 – Gráfico de histórico de estrelas dos repositórios do Angular, React e Vue no GitHub (2014 - 2022)

Fonte:(HISTORY, 2023)

um crescimento mais lento, também demonstra uma maturidade crescente à medida que ganha mais popularidade e adoção.

#### 5.1.1.6 Adoção por grandes empresas

Todas as três ferramentas também são utilizadas por marcas conhecidas, em produção, como apresentado na tabela 3.

Tabela 3 – Marcas notáveis que utilizam as ferramentas

| Angular | React | Vue |
|---|---|---|
| Google | Facebook | Alibaba |
| Wix | Netflix | Baidu |
| Weather.com | Paypal | Expedia |
| Healthcare.gov | AirBnB | Nintendo |
| Forbes | Uber | GitLab |

Em conclusão, as três são adotadas por grandes empresas em seus projetos de produção, sendo um indicativo de confiança na capacidade dessas tecnologias em termos de manutenibilidade, estabilidade e popularidade. A presença dessas ferramentas em empresas renomadas reforça sua posição como opções viáveis e confiáveis para o desenvolvimento de software em diferentes setores da indústria.

### 5.1.2 Curva de Aprendizado

Angular é uma ferramenta de desenvolvimento que segue o padrão MVC (Model, View, Controller), possuindo uma ampla gama de funcionalidades disponíveis e, portanto, pode requerer mais tempo para dominá-la. Utilizando TypeScript, um subconjunto do JavaScript, Angular permite escrever código JavaScript orientado a objetos, além de oferecer recursos de tipo que ajudam a prevenir erros comuns durante o desenvolvimento, como passar argumentos inválidos para funções. Após a compilação, o TypeScript é convertido em JavaScript puro, permitindo o uso de JavaScript no desenvolvimento, embora com menos robustez. Angular possui inúmeras funcionalidades poderosas, tornando-se útil para construir aplicações grandes e complexas. No entanto, o fato de ter funcionalidades únicas, pode aumentar o tempo de aprendizagem.

React, por outro lado, é uma ferramenta de desenvolvimento focada na camada de visualização (View). Ao lidar apenas com a camada de visualização, é necessário importar outras bibliotecas para implementar o padrão MVC completo. O React introduz o conceito de JSX, que combina HTML e JavaScript, sendo necessário o conhecimento sobre.

Por fim, o Vue também é uma ferramenta de desenvolvimento focada na camada de visualização (View), seus conceitos estão intimamente relacionados ao HTML e JavaScript básicos, proporcionando uma sensação de familiaridade ao programador. O início do desenvolvimento com o Vue é bastante simples, exigindo apenas a importação em um arquivo HTML. O Vue é considerado adequado para implementações simples em aplicações menores, mas à medida que a aplicação cresce em escala, é necessário utilizar arquivos específicos e lidar com configurações e padrões mais complexos.

Em resumo, Angular oferece uma ampla gama de funcionalidades, mas requer mais tempo para dominá-lo. React possui uma abordagem especializada na camada de visualização, com o uso de JSX, que pode exigir um tempo adicional de aprendizado. Já o Vue é considerado mais fácil de aprender, com conceitos familiares aos programadores que estão acostumados com HTML e JavaScript básicos. A escolha entre essas ferramentas dependerá das necessidades do projeto, do tempo disponível para aprendizado e das preferências da equipe de desenvolvimento.

Foi proposta uma representação gráfica de uma possível curva de aprendizado no curso 'Angular, React e Vue: Um guia completo' (SCHWARZMüLLER, 2023), conforme ilustrado na Figura 15. Essa representação é frequentemente referenciada em diversos sites para demonstrar e explicar a necessidade de conhecimento em cada ferramenta. Ao analisar o gráfico, podemos observar o seguinte:

- Angular: envolve entender o TypeScript, que é a linguagem de programação utilizada, bem como o fluxo de trabalho específico do Angular. É necessário compreender o conceito de componentes e módulos, além de aprender sobre injeção de dependência.

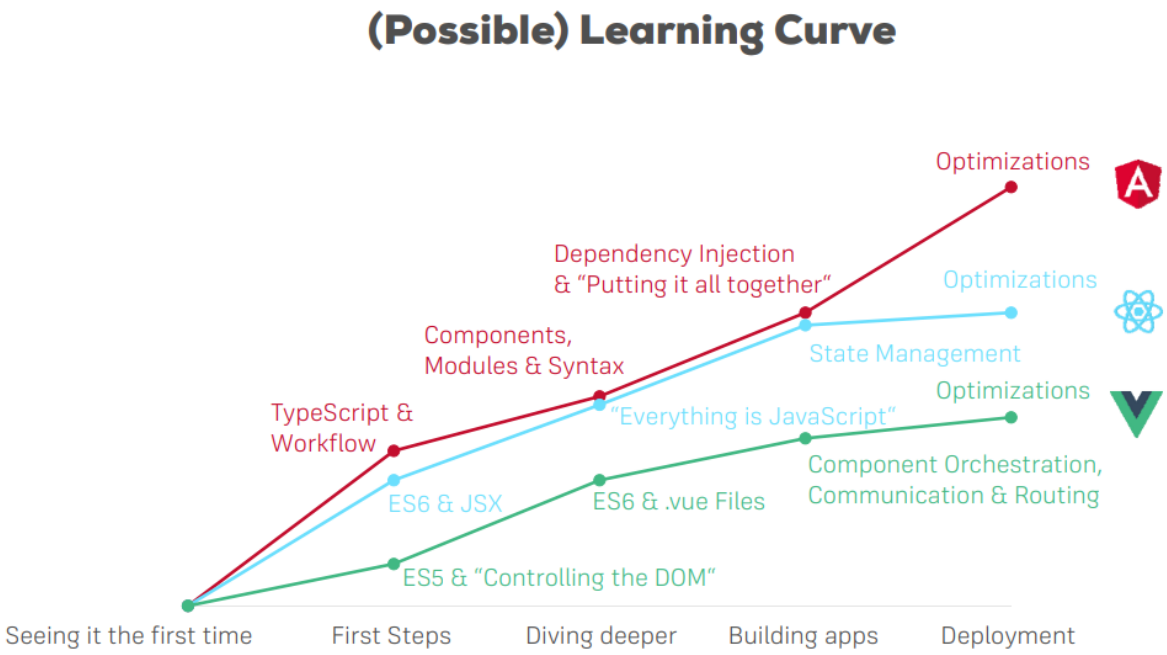



Figura 15 – Possível curva de aprendizado comparativa entre Angular, React e Vue

Fonte: (UDEMY, 2023)

Além disso, para otimizar o desempenho do aplicativo, é importante explorar técnicas como *tree-shaking*, AOT *(Ahead of Time Compilation)*, *lazy loading* e renderização no lado do servidor.

- React: uma das principais tarefas ao aprender React é familiarizar-se com o ES6 (EcmaScript 2015) e o uso de JSX (JavaScript XML). Embora essa transição possa exigir algum esforço inicial, uma vez dominados, esses recursos se tornam poderosas ferramentas para o desenvolvimento. Além disso, uma parte crucial do aprendizado é compreender como gerenciar o estado do aplicativo utilizando bibliotecas como *Redux* e *Flux*. Essas soluções ajudam a manter a consistência dos dados em um aplicativo React de médio a grande porte.

- Vue: sua curva de aprendizado é geralmente considerada mais suave em comparação com Angular e React. É adotado uma abordagem progressiva, o que significa que você pode começar a construir aplicativos Vue de maneira incremental, adicionando recursos à medida que avança. Depois de aprender os conceitos básicos de reatividade, você estará pronto para desenvolver aplicativos Vue, que permite que você escreva todo o seu aplicativo como um único objeto JavaScript grande, tornando a estrutura e a organização do código relativamente simples.

É importante destacar que a curva de aprendizado pode variar de acordo com a

experiência prévia do desenvolvedor, suas habilidades de programação e a familiaridade com conceitos como componentização e gerenciamento de estado. No entanto, em geral, Vue tende a ter uma curva de aprendizado mais suave, enquanto Angular pode ser considerado mais complexo e React considerado intermediário. A figura 16, apresenta um resumo e conclusão das informações coletadas.

| Framework | Principais Conceitos | Curva de Aprendizado |
|---|---|---|
| Angular | TypeScript, componentes, módulos, injeção de dependência | Complexa |
| React | ES6, JSX, gerenciamento de estado com Redux e Flux | Moderada |
| Vue | Abordagem progressiva, reatividade, estrutura simplificada | Suave |

Figura 16 – Curva de aprendizado comparativa entre Angular, React e Vue

## 5.2 Desempenho

Apresentação dos testes e resultados obtidos para a métrica de desempenho.

### 5.2.1 Aplicação modelo

Foi desenvolvido um aplicativo de gerenciamento financeiro nas três ferramentas front-end: Angular, React e Vue. O principal objetivo dessa implementação foi mensurar o tempo de carregamento inicial da aplicação utilizando a ferramenta Lighthouse para avaliação de desempenho.

O código-fonte de cada ferramenta está disponível no GitHub (PATRICIAP, 2023b) para referência e análise. A aplicação modelo foi desenvolvida de maneira mais simples e próxima nas três ferramentas, garantindo uma base de código comum e funcionalidades e interface iguais, ilustrado na figura 17. A aplicação de controle financeiro permite adicionar, editar, excluir transações e calcular o saldo, que são exibidos em uma lista. Os dados iniciais são carregados a partir de um arquivo JSON.

Essa abordagem foi adotada para possibilitar uma comparação precisa do desempenho das ferramentas em relação ao tempo de carregamento da aplicação. O Lighthouse foi utilizado para realizar medições objetivas e padronizadas em diferentes cenários de dados, permitindo identificar possíveis diferenças significativas entre as ferramentas.

É importante ressaltar que o foco principal dessa implementação foi a análise do desempenho das ferramentas front-end em relação ao tempo de carregamento da aplicação,

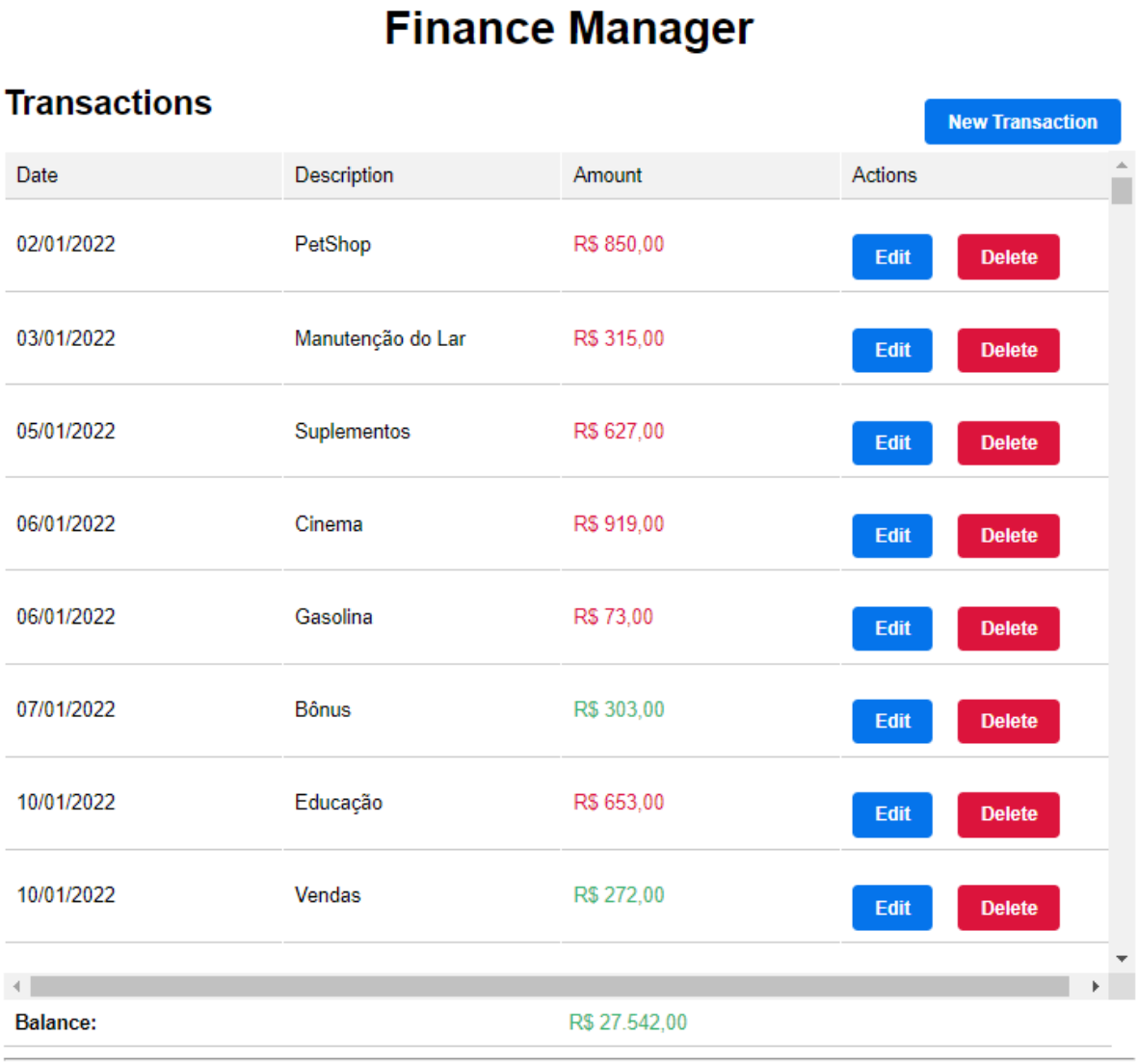



Figura 17 – Interface da aplicação

utilizando o Lighthouse como ferramenta de medição. Outros aspectos relacionados ao back-end ou funcionalidades avançadas não foram considerados nesse estudo específico.

A estrutura de dados utilizada para representar uma transação financeira possui os seguintes campos:

- **id**: Um número inteiro que identifica exclusivamente cada transação.
- **date**: Um objeto do tipo `Date` que representa a data da transação.
- **description**: Uma string que descreve a transação.
- **amount**: Um número que indica o valor monetário da transação.

- **type**: Uma string que indica o tipo da transação, podendo ser `income` (entrada de dinheiro) ou `expense` (saída de dinheiro).

Para realizar a análise e teste da aplicação financeira proposta, foram gerados dados simulados de transações por meio de um algoritmo, também disponível no repositório do github (PATRICIAP, 2023a). Esse algoritmo foi executado para gerar diferentes quantidades de transações, incluindo 10, 100, 1000 e 10000 transações.

Foi utilizado o módulo *random* do Python para gerar valores aleatórios dos atributos de cada transação, divididas em *expense* e *income* e cada uma delas com subcategorias representando diferentes tipos de despesas ou receitas.

As transações geradas contêm os seguintes atributos: um identificador único chamado *id*, uma data aleatória entre 1 de janeiro de 2022 e 31 de dezembro de 2023 denominada *date*, uma descrição selecionada aleatoriamente dentre as subcategorias correspondentes à categoria da transação chamada *description*, um valor monetário aleatório entre 1 e 1000 denominado *amount* e um indicador de entrada ou saída de dinheiro chamado *type* (sendo *income* para entrada e *expense* para saída).

Após a geração das transações, elas foram ordenadas em ordem crescente de data. Em seguida, foram convertidas para o formato JSON para facilitar o armazenamento e manipulação dos dados. Cada conjunto de transações geradas foi salvo em um arquivo separado, cujo nome refletia a quantidade de transações geradas.

Essa abordagem de geração de dados simulados permitiu a criação de conjuntos com 10, 100, 1000 e 10000 transações, possibilitando a avaliação da aplicação financeira em diferentes cenários de uso. Vale ressaltar que os dados gerados são fictícios e utilizados apenas para fins de demonstração e testes no contexto deste trabalho.

#### 5.2.1.1 Linhas de Código

Foi executado a ferramenta cloc nos diretórios especificados, excluindo a pasta 'assets', que contém os arquivos de dados JSON utilizados para popular o aplicativo. Essa análise permite obter algumas métricas sobre o código-fonte presente nos diretórios. A tabela 4 apresenta uma visão resumida da análise das linhas de código nos projetos.

Tabela 4 – Quantidade de Arquivos e Linhas de Código por Ferramenta

| **Ferramenta** | **Quantidade de Arquivos** | **Quantidade de Linhas de Código** |
|---|---|---|
| Angular | 10 | 424 |
| React | 7 | 509 |
| Vue | 3 | 326 |

Com base nos dados fornecidos, podemos concluir que o Vue possui a menor quantidade de arquivos e linhas de código em comparação ao Angular e React. Isso pode

```
Microsoft Windows [versão 10.0.22621.1778]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\patri>cloc --exclude-dir=assets C:\Users\patri\TCC\vue_application\vue-app\src
       3 text files.
       3 unique files.
       1 file ignored.

github.com/AlDanial/cloc v 1.96  T=0.04 s (74.4 files/s, 9423.9 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Vuejs Component                  1             52              0            317
TypeScript                       1              1              0              6
JavaScript                       1              1              0              3
-------------------------------------------------------------------------------
SUM:                             3             54              0            326
-------------------------------------------------------------------------------

C:\Users\patri>cloc --exclude-dir=assets C:\Users\patri\TCC\react_application\react-app\src
       6 text files.
       6 unique files.
       1 file ignored.

github.com/AlDanial/cloc v 1.96  T=0.40 s (14.9 files/s, 1023.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                       4             16              7            200
CSS                              2             31              0            158
-------------------------------------------------------------------------------
SUM:                             6             47              7            358
-------------------------------------------------------------------------------

C:\Users\patri>cloc --exclude-dir=assets C:\Users\patri\TCC\angular_application\angular-app\src
      10 text files.
      10 unique files.
       2 files ignored.

github.com/AlDanial/cloc v 1.96  T=0.05 s (198.9 files/s, 9846.7 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
TypeScript                       6             28              3            155
CSS                              2             30              1            149
HTML                             2              9              0            120
-------------------------------------------------------------------------------
SUM:                            10             67              4            424
-------------------------------------------------------------------------------

C:\Users\patri>
```

Figura 18 – Quantidade de arquivos e linhas de código por ferramenta

indicar uma abordagem mais concisa e simplificada na estrutura do código. No entanto, é importante lembrar que a quantidade de código não é o único fator determinante para o desempenho de uma aplicação.

#### 5.2.1.2 Análise de Tamanho de Pacotes com Bundlephobia

Foi realizado uma análise do tamanho dos pacotes da aplicação modelo utilizando a ferramenta Bundlephobia, que analisa o tamanho dos pacotes JavaScript. Ela fornece informações detalhadas sobre o tamanho mínimo do pacote, o tamanho com compressão gzip e estimativas de tempo de carregamento em diferentes tipos de conexões de internet,

o resultado é apresentado na figura 19 referente a análise do package.json da aplicação em Angular, figura 20 da análise do package.json da aplicação em React e a figura 21 da análise do package.json da aplicação em Vue.

A tabela resume os dados das análises do Bundlephobia para as três ferramentas: Angular, React e Vue. Analisando cada métrica:

- Min: indica o tamanho mínimo do bundle em *megabytes* (MB). Quanto menor o valor, melhor, pois indica um *bundle* mais compacto. O Angular tem um *bundle* mínimo de 1MB, o React tem 360.5KB e o Vue tem 250.4KB. Nesse aspecto, o Vue é a opção mais leve em termos de tamanho de bundle.

- Min + Gzip: é o tamanho mínimo do *bundle* após ser comprimido com o algoritmo Gzip. Novamente, quanto menor o valor, melhor, pois indica um bundle mais compacto após a compressão. O React possui um bundle de 89KB após ser comprimido com Gzip, seguido pelo Vue com 80KB e o Angular com 275.6KB. Nesse aspecto, o React apresenta o bundle mais leve após a compressão.

- 2G Edge (s): indica o tempo de carregamento do *bundle* em segundos (s) em uma conexão 2G Edge. Quanto menor o valor, melhor, pois indica um carregamento mais rápido do *bundle*. O Vue tem o menor tempo de carregamento em uma conexão 2G Edge, com 2.67s, seguido pelo React com 2.97s e o Angular com 9.19s. Nesse aspecto, o Vue tem um desempenho melhor em termos de tempo de carregamento.

- *Slow* 3G (s): é o tempo de carregamento do *bundle* em segundos (s) em uma conexão 3G lenta. Assim como no caso anterior, quanto menor o valor, melhor, indicando um carregamento mais rápido. O Vue novamente apresenta o melhor desempenho, com 1.6s de tempo de carregamento em uma conexão 3G lenta, seguido pelo React com 1.78s e o Angular com 5.51s.

Ao comparar Angular, React e Vue em termos de tamanho do *bundle*, compressão com Gzip e tempos de carregamento em conexões 2G Edge e 3G lenta, podemos resumir da seguinte forma, apresentado na figura 22:

Com base nessas métricas, o Vue se destaca como a opção com o menor tamanho de *bundle*, melhor compressão e tempos de carregamento mais rápidos tanto em conexões 2G Edge quanto em 3G lenta. O React também apresenta bons resultados em todas as métricas, enquanto o Angular tem um tamanho de *bundle* maior e tempos de carregamento mais lentos.

**Results**

SORT BY: Name: A → Z Size: High → Low

| | | | | | |
|---|---|---|---|---|---|
| 1. | @angular/animations v15.2.0 | 5 kB MIN | 1.3 kB MIN + GZIP | 26 ms SLOW 3G | 1 ms EMERGING 4G |
| 2. | @angular/common v15.2.0 | 93.4 kB MIN | 25.4 kB MIN + GZIP | 0.51 s SLOW 3G | 29 ms EMERGING 4G |
| 3. | @angular/compiler v15.2.0 | 334.2 kB MIN | 92.9 kB MIN + GZIP | 1.86 s SLOW 3G | 106 ms EMERGING 4G |
| 4. | @angular/core v15.2.0 | 237.5 kB MIN | 74.1 kB MIN + GZIP | 1.48 s SLOW 3G | 85 ms EMERGING 4G |
| 5. | @angular/forms v15.2.0 | 85.1 kB MIN | 15.9 kB MIN + GZIP | 318 ms SLOW 3G | 18 ms EMERGING 4G |
| 6. | @angular/platform-browser v15.2.0 | 31.2 kB MIN | 8.4 kB MIN + GZIP | 167 ms SLOW 3G | 10 ms EMERGING 4G |
| 7. | @angular/platform-browser-dynamic v15.2.0 | 3.4 kB MIN | 1.4 kB MIN + GZIP | 28 ms SLOW 3G | 2 ms EMERGING 4G |
| 8. | @angular/router v15.2.0 | 100.6 kB MIN | 24.6 kB MIN + GZIP | 493 ms SLOW 3G | 28 ms EMERGING 4G |
| 9. | rxjs v7.8.0 | 69.3 kB MIN | 17.7 kB MIN + GZIP | 353 ms SLOW 3G | 20 ms EMERGING 4G |
| 10. | tslib v2.3.0 | 7.6 kB MIN | 2.6 kB MIN + GZIP | 52 ms SLOW 3G | 3 ms EMERGING 4G |
| 11. | zone.js v0.12.0 | 33.3 kB MIN | 11.3 kB MIN + GZIP | 226 ms SLOW 3G | 13 ms EMERGING 4G |
| | TOTAL | 1 MB MIN | 275.6 kB MIN + GZIP | 9.19 s 2G EDGE | 5.51 s SLOW 3G |

Figura 19 – Resultado da análise Bundlephobia - Angular

Fonte: (BUNDLEPHOBIA, 2023)

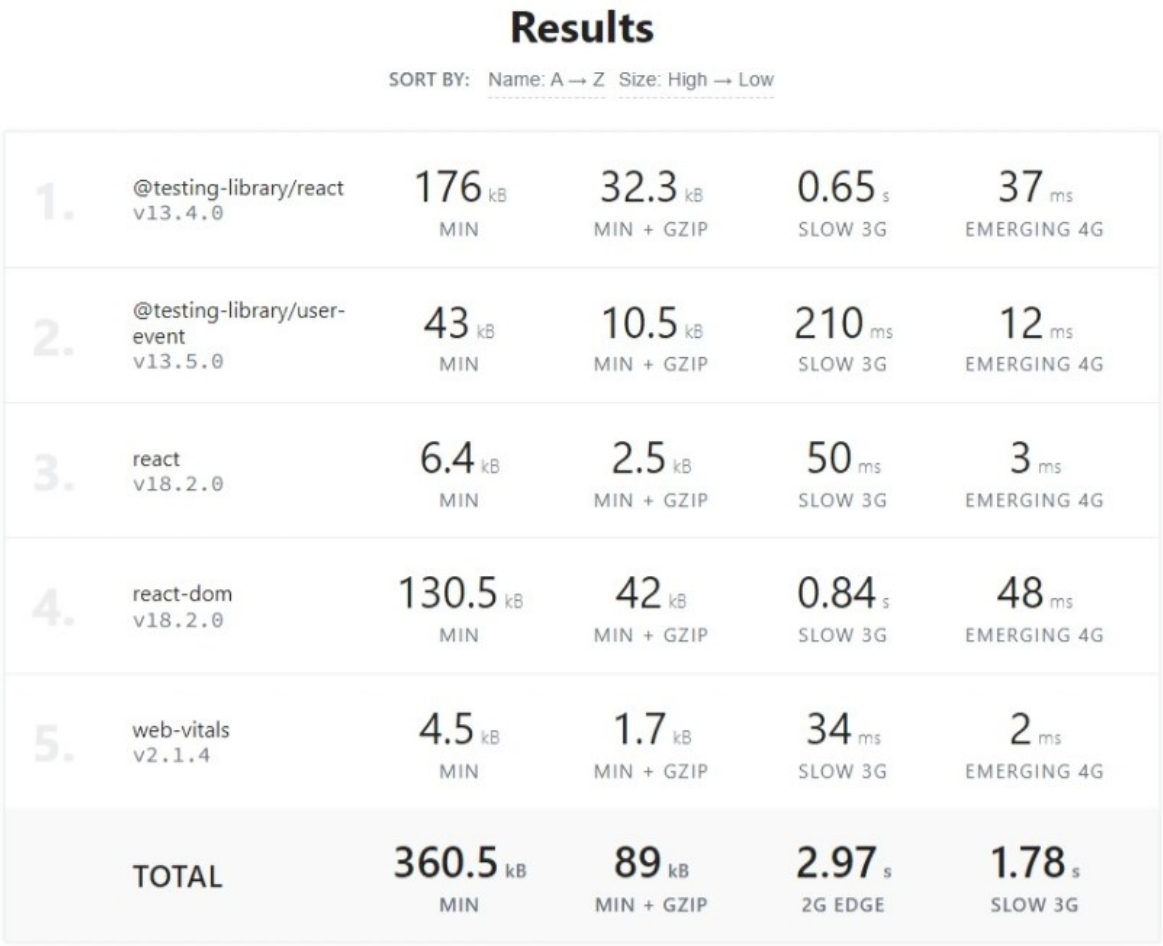



Figura 20 – Resultado da análise Bundlephobia - React

Fonte: (BUNDLEPHOBIA, 2023)

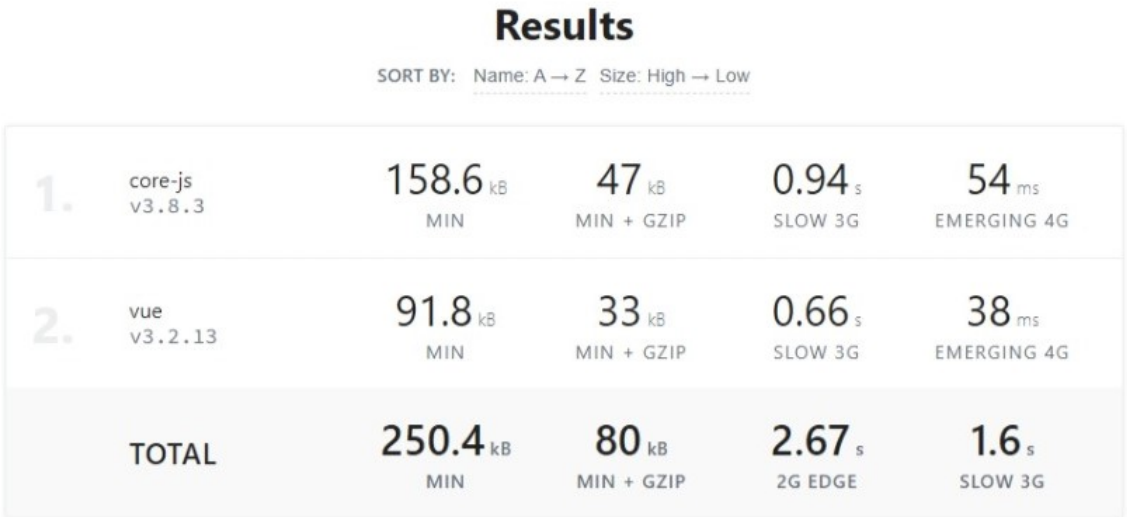



Figura 21 – Resultado da análise Bundlephobia - Vue

Fonte: (BUNDLEPHOBIA, 2023)

| | Min (MB) | Min + Gzip(KB) | 2G Edge (s) | Slow 3G (s) |
|---|---|---|---|---|
| Angular | 1 | 275.6 | 9.19 | 5.51 |
| React | 0.352 | 89 | 2.97 | 1.78 |
| Vue | 0.244 | 80 | 2.67 | 1.6 |

Figura 22 – Tabela de comparação da análise Bundlephobia para as ferramentas

#### 5.2.1.3 Desempenho carregamento inicial

Para comparar o tempo de carregamento inicial das aplicações, foi utilizado a ferramenta Lighthouse, que oferece uma análise abrangente do desempenho de uma página da *web*, ele fornece várias métricas relevantes, como *First Contentful Paint* (FCP), *Speed Index* (SI), *Largest Contentful Paint* (LCP), *Time to Interactive* (TTI) e *Speed Index* (SI) e outros.

O teste de carregamento inicial para cada, foi simulado em ambiente *desktop* e *mobile*, com diferentes quantidades de transações: 10, 100, 1000 e 10000. Para obter dados mais precisos, foi executado cada teste 10 vezes e calculado a média dos resultados. Além disso, para avaliar a confiabilidade dos resultados, foi utilizado um intervalo de confiança de 90%.

Além de ter sido realizado a limpeza de *cache* antes de cada execução do teste de carregamento inicial. Essa abordagem garante que cada execução seja independente e não seja influenciada pelo armazenamento de testes anteriores. Ao realizar essa limpeza, elimina-se qualquer influência sobre o tempo de carregamento inicial. Dessa forma, os resultados obtidos refletem de maneira mais precisa o desempenho real da aplicação, considerando que nenhum recurso está sendo carregado a partir da *cache* do navegador.

*First Contentful Paint* (FCP): mensura o tempo decorrido desde o início da navegação até o momento em que o primeiro elemento visual é exibido na tela. Essa métrica indica a rapidez com que o conteúdo inicial da página é carregado e pode ser usado como um indicador da percepção inicial de velocidade pelos usuários.

*Largest Contentful Paint* (LCP): mede o tempo decorrido desde o início da navegação até o momento em que o maior elemento visível é exibido na tela. Esse elemento geralmente é o que chama mais atenção do usuário e pode ser um texto, figura ou outro elemento de conteúdo. O LCP é uma métrica importante para avaliar a velocidade de carregamento perceptível da página.

*Cumulative Layout Shift* (CLS): mede a estabilidade visual da página durante o carregamento, ele avalia a quantidade de deslocamento inesperado de elementos visuais na tela, que pode ocorrer quando elementos são carregados de forma assíncrona ou quando o espaço reservado para esses elementos não é definido corretamente. Um baixo valor de

CLS indica que a página é mais estável visualmente.

*Speed Index* (SI): é uma métrica que avalia a velocidade percebida do carregamento da página. Ele mede o tempo necessário para que o conteúdo acima da dobra seja exibido na tela de forma progressiva. Valores menores indicam um carregamento mais rápido da página e uma percepção mais ágil por parte dos usuários.

Cada uma dessas métricas fornece informações importantes sobre o desempenho de carregamento da página. O FCP e o LCP indicam o tempo de exibição do conteúdo visual, o CLS avalia a estabilidade visual da página e o SI mede a velocidade percebida do carregamento.

A figura 23 apresenta uma análise comparativa do desempenho das aplicações em termos das métricas FCP (*First Contentful Paint*), LCP (*Largest Contentful Paint*), CLS (*Cumulative Layout Shift*) e SI (*Speed Index*), explicadas anteriormente, simulando o carregamento inicial em *desktop* e *mobile*, para diferentes quantidades de transações (10, 100, 1000 e 10000).

As linhas representam o desempenho de cada ferramenta em relação às métricas. O eixo horizontal do gráfico indica a quantidade de transações (10, 100, 1000, 10000), enquanto o eixo vertical representa o tempo em segundos para cada métrica. A comparação entre as linhas ajuda a visualizar as diferenças de desempenho entre as ferramentas em diferentes cenários de uso, em ambiente desktop e *mobile*.

Dessa forma, a tabela ilustrada na figura 24 proporciona uma visualização comparativa dos valores de desempenho para cada métrica e permite identificar quais ferramentas apresentam um melhor desempenho em relação ao tempo de carregamento inicial e estabilidade visual.

Cada célula da tabela ilustrada na figura 24, representa uma métrica específica do desempenho, e os valores são preenchidos de acordo com os resultados obtidos nos testes. A tabela é dividida em colunas que representam as métricas de desempenho, como First Contentful Paint (FCP), Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) e Speed Index (SI). As linhas da tabela representam as diferentes quantidades de transações.

Para cada célula da tabela, são inseridos os valores correspondentes às métricas de desempenho e quantidade de transações. Esses valores são obtidos a partir das medições realizadas durante os testes de desempenho, levando em consideração a média dos resultados para cada cenário.

As células pintadas em verde, amarelo e vermelho na tabela são usadas para indicar a qualidade do desempenho de cada ferramenta e métrica de acordo com a pontuação do lighthouse, demonstrada na figura 25. As células verdes representam um desempenho bom, as células amarelas indicam que há espaço para melhoria e as células vermelhas indicam

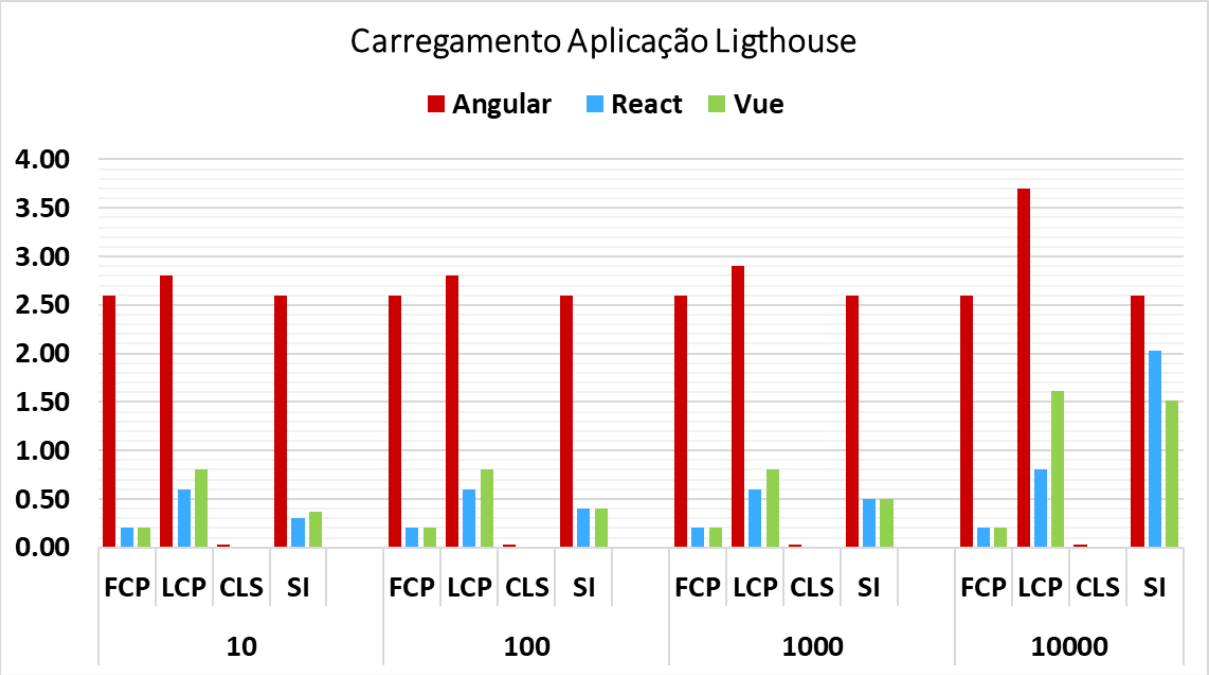



Figura 23 – Comparação de valores das métricas de desempenho (FCP, LCP, CLS e SI) obtidas através do Lighthouse para as ferramentas Angular, React e Vue, em desktop e mobile

um desempenho ruim.

Ao analisar os resultados, podemos observar o resultado para cada métrica:

- FCP: tempo de renderização do conteúdo inicial varia entre as aplicações, mas em geral é rápido para todas as quantidades de transações e plataformas.
- LCP: tempo de renderização do maior elemento da página varia significativamente entre as aplicações, sendo mais rápido para React e Vue em todas as quantidades de transações e plataformas.
- CLS: a estabilidade visual das aplicações é boa, com valores muito baixos para todas as quantidades de transações e plataformas.
- SI: o índice de velocidade varia entre as aplicações, com React e Vue apresentando valores mais baixos, indicando um carregamento mais rápido da página.

Em geral, podemos concluir que React e Vue tendem a ter um desempenho melhor em termos de tempo de carregamento (LCP) e velocidade (SI) em comparação com Angular que pode ser mais lento no tempo de carregamento devido a fatores como tamanho do pacote e complexidade do ciclo de vida. No entanto, otimizações podem melhorar seu desempenho.

| Carregamento Aplicação Ligthouse (90% de intervalo de confiança) | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | Desktop | | | Mobile | | |
| TRANSAÇÕES | MÉTRICAS | Angular | React | Vue | Angular | React | Vue |
| 10 | FCP | 2.6 ± 0 | 0.2 ± 0 | 0.2 ± 0 | 15.5 ± 0 | 0.6 ± 0 | 0.6 ± 0 |
| | LCP | 2.8 ± 0 | 0.6 ± 0 | 0.8 ± 0 | 16.64 ± 0.07 | 2.8 ± 0 | 3.53 ± 0.03 |
| | CLS | 0.025 ± 0 | 0 ± 0 | 0 ± 0 | 0.07 ± 0 | 0 ± 0 | 0 ± 0 |
| | SI | 2.6 ± 0 | 0.3 ± 0 | 0.37 ± 0.03 | 15.5 ± 0 | 1.07 ± 0.09 | 0.91 ± 0.02 |
| | | | | | | | |
| 100 | FCP | 2.6 ± 0 | 0.2 ± 0 | 0.2 ± 0 | 15.5 ± 0 | 0.6 ± 0 | 0.6 ± 0 |
| | LCP | 2.8 ± 0 | 0.6 ± 0 | 0.8 ± 0 | 16.66 ± 0.04 | 2.82 ± 0.02 | 3.6 ± 0 |
| | CLS | 0.025 ± 0 | 0 ± 0 | 0 ± 0 | 0.07 ± 0 | 0 ± 0 | 0 ± 0 |
| | SI | 2.6 ± 0 | 0.4 ± 0 | 0.4 ± 0 | 15.5 ± 0 | 1.01 ± 0.04 | 0.9 ± 0 |
| | | | | | | | |
| 1000 | FCP | 2.6 ± 0 | 0.2 ± 0 | 0.2 ± 0 | 15.5 ± 0 | 0.6 ± 0 | 0.6 ± 0 |
| | LCP | 2.9 ± 0 | 0.6 ± 0 | 0.8 ± 0 | 17.3 ± 0 | 2.84 | 3.91 ± 0.02 |
| | CLS | 0.025 ± 0 | 0 ± 0 | 0 ± 0 | 0.063 ± 0.01 | 0 ± 0 | 0 ± 0 |
| | SI | 2.6 ± 0 | 0.5 ± 0 | 0.5 ± 0 | 15.5 ± 0 | 1.46 ± 0.05 | 1.37 ± 0.06 |
| | | | | | | | |
| 10000 | FCP | 2.6 ± 0 | 0.2 ± 0 | 0.2 ± 0 | 15.5 ± 0 | 0.6 ± 0 | 0.6 ± 0 |
| | LCP | 3.7 ± 0.16 | 0.8 ± 0 | 1.61 ± 0.02 | 21.05 ± 1.63 | 3.36 ± 0.1 | 7.13 ± 0.03 |
| | CLS | 0.025 ± 0 | 0 ± 0 | 0 ± 0 | 0 ± 0 | 0 ± 0 | 0 ± 0 |
| | SI | 2.6 ± 0 | 2.03 ± 0.05 | 1.52 ± 0.02 | 15.5 ± 0 | 6.07 ± 0.46 | 4.51 ± 0.05 |

Figura 24 – A tabela resume as métricas de desempenho (FCP, LCP, CLS e SI) obtidas através do Lighthouse para as ferramentas Angular, React e Vue em desktop e mobile.

| Métrica | Bom (Ideal) | Precisa de Melhoria | Ruim |
|---|---|---|---|
| First Contentful Paint (FCP) | Menos de 1.8s | 1.8s a 3s | Mais de 3s |
| Largest Contentful Paint (LCP) | Menos de 2.5s | 2.5s a 4s | Mais de 4s |
| Cumulative Layout Shift (CLS) | Menos de 0.1 | 0.1 a 0.25 | Mais de 0.25 |
| Speed Index (SI) | Menos de 3.8s | 3.8s a 6s | Mais de 6s |

Figura 25 – Classificação das métricas da avaliação de desempenho do Lighthouse

O Lighthouse determina o desempenho de uma aplicação *web* medindo o tempo de carregamento, avaliando a estabilidade visual e verificando a conformidade com boas práticas. Ele calcula uma pontuação geral de desempenho com base nas métricas discutidas

acima. O desempenho para cada ferramenta está ilustrado graficamente pela figura 26.

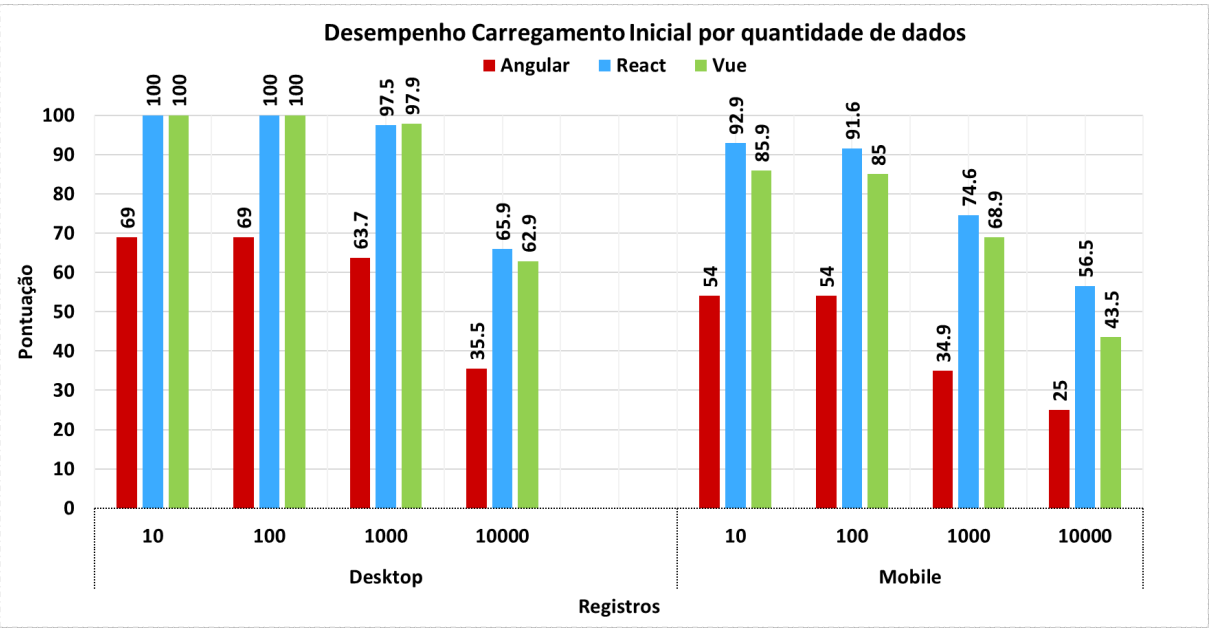



Figura 26 – Gráfico comparativo do desempenho de carregamento inicial entre Angular, React e Vue para desktop e mobile, com diferentes quantidades de dados.

| Desempenho Carregamento Inicial por quantidade de dados com 90% de intevalo de confiança. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Desktop | | | | Mobile | | | |
| | 10 | 100 | 1000 | 10000 | 10 | 100 | 1000 | 10000 |
| Angular | 69 ± 0 | 69 ± 0 | 63.7 ± 0.35 | 35.5 ± 0.82 | 54 ± 0 | 54 ± 0 | 34.9 ± 0.16 | 25 ± 0 |
| React | 100 ± 0 | 100 ± 0 | 97.5 ± 0.27 | 65.9 ± 0.68 | 92.9 ± 3.45 | 91.6 ± 0.50 | 74.6 ± 0.27 | 56.5 ± 1.21 |
| Vue | 100 ± 0 | 100 ± 0 | 97.9 ± 0.16 | 62.9 ± 0.16 | 85.9 ± 0.38 | 85 ± 0 | 68.9 ± 0.38 | 43.5 ± 0.27 |

Figura 27 – Gráfico comparativo do desempenho de carregamento inicial entre Angular, React e Vue para desktop e mobile, com diferentes quantidades de dados, apresentando um intervalo de confiança de 90%.

A tabela ilustrada pela figura 27 apresenta o desempenho de carregamento inicial dos frameworks Angular, React e Vue em diferentes quantidades de dados (10, 100, 1000 e 10000) e em ambientes *desktop* e *mobile*. Os valores são representados em porcentagem, onde 100% é o desempenho ideal.

Analisando os valores, observamos que o Angular tem um desempenho inferior em comparação com o React e o Vue em todas as quantidades de dados, o que significa que o Angular leva mais tempo para carregar e exibir o conteúdo inicial da página.

Tanto no *desktop* quanto no *mobile*, o React apresenta um desempenho melhor em relação ao carregamento inicial em todas as quantidades de dados, com valores próximos a 100%, indicando uma carga rápida e eficiente do conteúdo inicial.

O Vue também tem um bom desempenho, ficando próximo ao React em algumas quantidades de dados, mas mostrando um desempenho ligeiramente inferior em cargas mais pesadas.

Em resumo, a tabela mostra que o React tem o melhor desempenho no carregamento inicial, seguido pelo Vue, enquanto o Angular fica em desvantagem nesse aspecto. Esses resultados são representados pela porcentagem de desempenho em relação à referência de 100% na tabela.

#### 5.2.1.4 Análise Benchmarks Local DOM

O benchmark 'js-framework-benchmark' (KRAUSE, 2021) é uma ferramenta utilizada para comparar o desempenho de diferentes frameworks JavaScript ao executar uma série de tarefas comuns em aplicações web. Ele mensura a velocidade de renderização, manipulação de eventos, criação de elementos DOM, entre outras operações e disponibiliza o resultado em uma tabela que compara o desempenho, executando o benchmark no navegador Chrome versão 104 no sistema operacional Windows. A tabela mostra o tempo médio em milissegundos que cada framework leva para executar cada tarefa do *benchmark*.

As tabelas ilustradas pelas imagens 29, 29 e 30 apresentam os resultados do *benchmark* comparando o desempenho entre: vue-v3.2.37, angular-v13.0.0 e react-v17.0.2. A análise é dividida em três seções principais: duração em milissegundos de tarefas comuns em aplicações, ilustrada na figura 28; métricas de inicialização (Lighthouse com simulação móvel) representada pela figura 29 e alocação de memória em MBs, ilustrada na figura 30.

Na seção de duração em milissegundos, são fornecidos os tempos de execução para várias tarefas, como criar linhas, atualizar linhas, destacar uma linha selecionada, trocar linhas, remover linhas, criar muitas linhas, anexar linhas a uma tabela grande e limpar uma tabela. Os tempos são apresentados para cada em diferentes cenários (quantidade de linhas, *slowdown* do CPU etc.), indicando a duração média em milissegundos e o valor de *slowdown* em relação ao framework mais rápido. Os valores mais baixos indicam um desempenho mais rápido e eficiente.

Na seção de métricas de inicialização, são apresentadas métricas relacionadas ao tempo de interatividade consistente e ao peso total em *kilobytes* dos recursos carregados na página. As métricas são apresentadas para cada ferramenta e indicam o valor médio e o valor comparativo em relação ao Vue (que tem o valor 1.00). Novamente, os valores mais baixos indicam um melhor desempenho.

Na seção de alocação de memória em MBs, são fornecidos dados sobre a quantidade de memória usada pelos frameworks em diferentes estágios, como após o carregamento da página, após adicionar linhas, após atualizar linhas e após criar e limpar linhas.

Analisando os resultados, podemos observar que o framework React (versão 17.0.2) teve o melhor desempenho em várias métricas e tarefas do *benchmark*. Em geral, obteve tempos de execução mais baixos em comparação com Vue e Angular em diversas tarefas, como criar linhas, atualizar linhas, trocar linhas e remover linhas.

| Duração em milissegundos ± intervalo de confiança de 95% (Desaceleração = Duração / Mais Rápido) | | | |
|---|---|---|---|
| Nome<br>Duração para... | vue-v3.2.37 | angular-v13.0.0 | react-v17.0.2 |
| **criar linhas**<br>criando 1.000 linhas (5 execuções de aquecimento). | 119.3 ± 1.2<br>(1.01) | 118.5 ± 1.4<br>(1.00) | 126.2 ± 1.4<br>(1.06) |
| **substituir todas as linhas**<br>atualizando todas as 1.000 linhas (5 execuções de aquecimento). | 113.2 ± 1.5<br>(1.00) | 130.7 ± 1.0<br>(1.15) | 126.7 ± 0.9<br>(1.12) |
| **atualização parcial**<br>atualizando a cada 10ª linha para 1.000 linhas (3 execuções de aquecimento). Desaceleração de CPU de 16x. | 359.0 ± 11.2<br>(1.05) | 343.0 ± 6.7<br>(1.00) | 399.5 ± 5.4<br>(1.16) |
| **selecionar linha**<br>destacando uma linha selecionada. (5 execuções de aquecimento). Desaceleração de CPU | 53.7 ± 1.2<br>(1.23) | 43.6 ± 0.8<br>(1.00) | 105.2 ± 3.7<br>(2.41) |
| **trocar linhas**<br>trocar 2 linhas para tabela com 1.000 linhas. (5 execuções de aquecimento). Desaceleração de CPU de 4x. | 73.6 ± 4.5<br>(1.00) | 512.8 ± 3.4<br>(6.97) | 494.1 ± 7.7<br>(6.71) |
| **remover linha**<br>removendo uma linha. (5 execuções de aquecimento). | 28.0 ± 0.6<br>(1.09) | 25.7 ± 0.4<br>(1.00) | 27.8 ± 0.5<br>(1.08) |
| **criar muitas linhas**<br>criando 10.000 linhas. (5 execuções de aquecimento com 1.000 linhas). | 1,149.2 ± 5.1<br>(1.00) | 1,215.1 ± 11.5<br>(1.06) | 1,488.1 ± 10.2<br>(1.29) |
| **adicionar linhas a tabela grande**<br>adicionando 1.000 a uma tabela com 10.000 linhas. Desaceleração de CPU de 2x. | 268.8 ± 7.3<br>(1.00) | 303.2 ± 2.8<br>(1.13) | 322.5 ± 4.0<br>(1.20) |
| **limpar linhas**<br>limpando uma tabela com 1.000 linhas. Desaceleração de CPU de 8x. (5 execuções de aquecimento). | 90.0 ± 2.9<br>(1.00) | 231.0 ± 6.8<br>(2.57) | 101.0 ± 2.9<br>(1.12) |
| média geométrica de todos os fatores na tabela | 1.04 | 1.43 | 1.52 |

Figura 28 – O gráfico compara o desempenho de carregamento inicial do Angular, React e Vue em diferentes quantidades de dados para desktop e mobile.

Fonte: (KRAUSE, 2021)

Além disso, o React também apresentou uma melhor performance em termos de métricas de inicialização, como tempo de interatividade consistente e peso total dos

recursos carregados na página, apresentando tempos mais rápidos em comparação com as outras ferramentas nessas métricas.

Quanto à alocação de memória, o React também teve resultados favoráveis, com valores menores em várias etapas de alocação de memória, como após o carregamento da página, adição e atualização de linhas, e criação e limpeza de linhas.

**Métricas de Inicialização (lighthouse simulação modo celular)**

| Name | vue-v3.2.37 | angular-v13.0.0 | react-v17.0.2 |
|---|---|---|---|
| **Interativo de forma consistente** Um TTI pessimista - quando a CPU e a rede estão definitivamente ociosas | 2,104.8 ± 49.6 (1.00) | 2,783.1 ± 0.4 (1.32) | 2,554.8 ± 0.2 (1.21) |
| **Peso total em kilobytes** Custo de transferência de rede (pós-compressão) de todos os recursos carregados na página | 196.4 ± 0.0 (1.00) | 291.2 ± 0.0 (1.48) | 274.5 ± 0.0 (1.40) |
| média geométrica de todos os fatores na tabela | 1 | 1.4 | 1.3 |

Figura 29 – Tabela comparativa das métricas de inicialização (Lighthouse simulação móvel).

Fonte: (KRAUSE, 2021)

**Alocação de memória em MBs ± intervalo de confiança de 95%**

| Name | vue-v3.2.37 | angular-v13.0.0 | react-v17.0.2 |
|---|---|---|---|
| **Memória pronta** Uso de memória após o carregamento da página | 1.3 (1.00) | 2.0 (1.57) | 1.4 (1.14) |
| **Memória em execução** Uso de memória após adicionar 1.000 linhas | 4.4 (1.00) | 5.3 (1.22) | 5.6 (1.28) |
| **Atualizar cada 10ª linha para 1.000 linhas (5 ciclos)** Uso de memória após clicar em atualizar cada 10ª linha 5 vezes | 4.4 (1.00) | 5.4 (1.22) | 6.1 (1.39) |
| **Criar/limpar 1.000 linhas (5 ciclos)** Uso de memória após criar e limpar 1.000 linhas 5 vezes | 1.6 (1.00) | 2.7 (1.71) | 2.2 (1.39) |
| **Memória em execução 10k** Uso de memória após adicionar 10.000 linhas | 30.5 (1.00) | 32.6 (1.07) | 39.5 (1.30) |
| média geométrica de todos os fatores na tabela | 1 | 1.34 | 1.29 |

Figura 30 – Tabela comparativa da alocação de memória em MBs

Fonte: (KRAUSE, 2021)

Portanto, com base nos resultados apresentados nesse *benchmark*, o React (versão

17.0.2) demonstrou ter o melhor desempenho entre os frameworks Vue, Angular e React nas diferentes tarefas e métricas avaliadas. É importante ressaltar que o desempenho pode variar dependendo das necessidades e requisitos específicos de um projeto, então é sempre importante considerar o contexto e os objetivos individuais ao escolher um framework.

Devido ao uso do Virtual DOM, o React e o Vue possibilitam um desempenho mais rápido para aplicativos que necessitam de atualizações muito frequentes. No entanto, com muitos recursos avançados, o Angular também é uma boa escolha se o seu aplicativo precisa de atualizações ocasionais.

A seleção e a troca de linhas não apresentarão resultados muito perceptíveis. No entanto, a seleção é mais comum do que a troca, e nesse aspecto o React e o Angular se destacam em relação ao Vue.

### 5.2.2 Desempenho de Página de Website

Os dados coletados no aplicativo web Perf Track, ilustrados na figura 31, mostram como o Angular, React e Vue se saem no carregamento da página, *bytes* utilizados, *inputs* e outras métricas, conforme de novembro de 2020.

Essa métrica não é definitiva para tirar conclusões, pois existem muitos fatores que afetam o desempenho. Analisando esses números, as páginas em React e Vue têm um desempenho melhor do que as páginas em Angular. Além disso, entre as três opções, o React é o mais lento para renderizar uma página, com apenas 47% dos websites apresentando um desempenho aceitável nessa área. No entanto, todas as três linguagens têm um bom desempenho, com cerca de 85% de eficiência no tempo de resposta antes que os usuários interajam com a página.

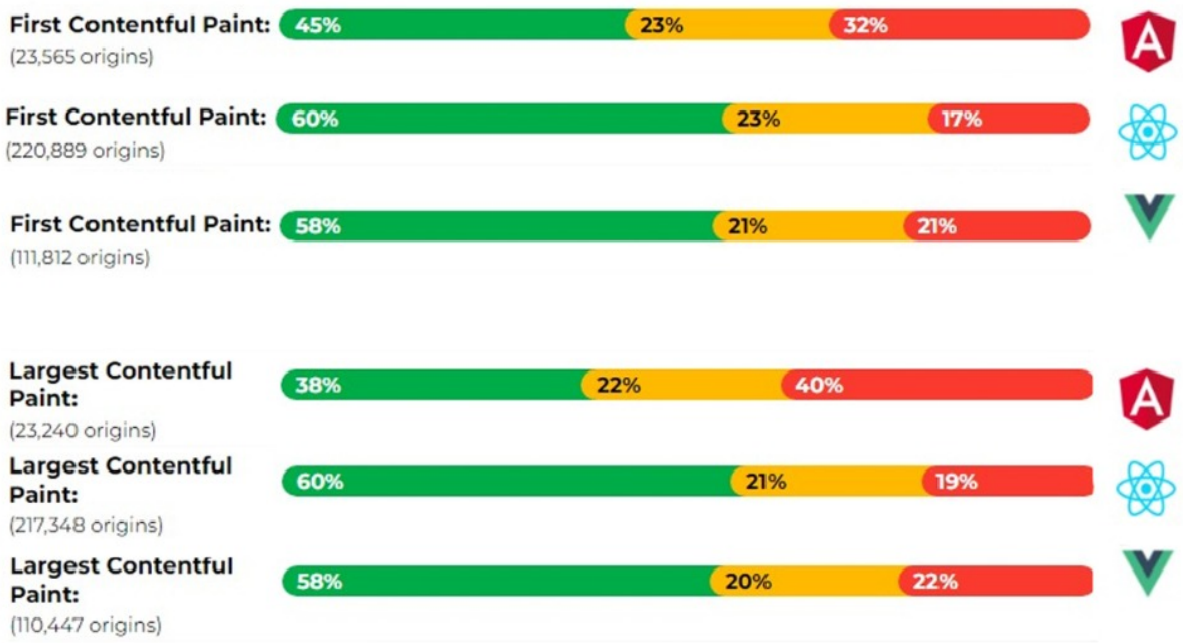



Figura 31 – Captura de tela do Perf Tracker mostrando o desempenho de Angular, React e Vue em várias métricas de carregamento de páginas em novembro de 2020.

Fonte: (GoogleChromeLabs, 2023)

### 5.2.3 Tabelas comparativas

Nesta seção, é apresentado as tabelas comparativas dos resultados obtidos nas métricas e testes realizados. Essas tabelas foram elaboradas para facilitar a visualização e compreensão das diferenças entre os valores obtidos, oferecendo uma representação visual clara e organizada dos dados, permitindo uma comparação direta entre os diferentes aspectos avaliados.

A Figura 32 exibe a tabela comparativa dos resultados das métricas de Popularidade/Maturidade coletadas. Em seguida, na Figura 33, a tabela comparativa dos resultados obtidos na métrica de desempenho em relação aos testes executados na aplicação modelo, que proporciona uma visão detalhada do desempenho das diferentes soluções em relação aos testes específicos realizados na aplicação modelo em questão. Por fim, a Figura 34 apresenta a tabela comparativa dos resultados obtidos na métrica de desempenho em relação aos testes do benchmark utilizado.

| | Angular | React | Vue |
|---|---|---|---|
| Desenvolvido | Google | Facebook | Evan You |
| Popularidade / Maturidade | Moderada | Alta | Crescente |
| Google Trends | Moderada | Alta | Crescente |
| Linguagem Principal | TypeScript | JavaScript | TypeScript |
| Idade Repositório | 11 anos | 12 anos | 10 anos |
| Versão Mais Recente | 1.8.3 | 18.2.0 | 3.3.4 |
| Npm Trends | 59,152 estrelas | 208,897 estrelas | 38,188 estrelas |
| Observadores | 3,034 | 6,639 | 5,996 |
| Commits | 2,568 | 810 | 300 |
| Estrelas | Estável | Crescimento Exponencial | Crescimento Gradual |
| Curva de Aprendizado | Complexa | Moderada | Suave |
| Adoção por grandes empresas | Google, Wix,Weather.com, healthcare.gov ,Forbes | Facebook,Airbnb, Uber, Netflix, Twitter, Pinterest, Reddit, Udemy,Paypal | Alibaba, Baidu, Expedia, Nintendo, GitLab |
| Número de Vagas (Indeed+Linkedin +Glassdoor) | 6576 | 6043 | 1521 |

Figura 32 – Tabela comparativa dos resultados obtidos das métricas de Popularidade/Maturidade



| Aplicação Modelo | Angular | React | Vue |
|---|---|---|---|
| Tamanho do Pacote (Min) | 1 MB | 0.352 MB | 0.244 MB |
| Tamanho do Pacote (Min + Gzip) | 275.6 KB | 89 KB | 80 KB |
| Tamanho do Pacote (2G Edge) | 9.19s | 2.97s | 2.67s |
| Tamanho do Pacote (Slow 3G) | 5.51s | 1.78s | 1.6s |
| Desempenho Carregamento Inicial (Desktop) | Moderado | Bom | Bom |
| Desempenho Carregamento Inicial (Mobile) | Moderado | Bom | Moderado |
| Carregamento Lighthouse (Desktop) | Moderado | Bom | Bom |
| Carregamento Lighthouse (Mobile) | Moderado | Bom | Bom |

Figura 33 – Tabela comparativa dos resultados obtidos da métrica de desempenho em relação aos testes executados na aplicação modelo

| Benchmark | Angular | React | Vue |
|---|---|---|---|
| Criar Linhas | 118.5 ± 1.4 (1.00) | 126.2 ± 1.4 (1.06) | 119.3 ± 1.2 (1.01) |
| Substituir Linhas | 130.7 ± 1.0 (1.15) | 126.7 ± 0.9 (1.12) | 113.2 ± 1.5 (1.00) |
| Atualização Parcial | 343.0 ± 6.7 (1.00) | 399.5 ± 5.4 (1.16) | 359.0 ± 11.2 (1.05) |
| Selecionar Linha | 43.6 ± 0.8 (1.00) | 105.2 ± 3.7 (2.41) | 53.7 ± 1.2 (1.23) |
| Trocar Linhas | 512.8 ± 3.4 (6.97) | 494.1 ± 7.7 (6.71) | 73.6 ± 4.5 (1.00) |
| Remover Linha | 25.7 ± 0.4 (1.00) | 27.8 ± 0.5 (1.08) | 28.0 ± 0.6 (1.09) |
| Criar Muitas Linhas | 1,215.1 ± 11.5 (1.06) | 1,488.1 ± 10.2 (1.29) | 1,149.2 ± 5.1 (1.00) |
| Adicionar Linhas | 303.2 ± 2.8 (1.13) | 322.5 ± 4.0 (1.20) | 268.8 ± 7.3 (1.00) |
| Limpar Linhas | 231.0 ± 6.8 (2.57) | 101.0 ± 2.9 (1.12) | 90.0 ± 2.9 (1.00) |
| Métricas Inicialização (Interativo) | 2,554.8 ± 0.2 (1.21) | 2,783.1 ± 0.4 (1.32) | 2,104.8 ± 49.6 (1.00) |
| Métricas Inicialização (Peso Total) | 274.5 ± 0.0 (1.40) | 291.2 ± 0.0 (1.48) | 196.4 ± 0.0 (1.00) |
| Alocação de Memória (Memória Pronta) | 2.0 (1.57) | 1.4 (1.14) | 1.3 (1.00) |
| Alocação de Memória (Memória em Execução) | 5.3 (1.22) | 5.6 (1.28) | 4.4 (1.00) |
| Alocação de Memória (Atualizar) | 5.4 (1.22) | 6.1 (1.39) | 4.4 (1.00) |
| Alocação de Memória (Criar/Limpar) | 2.7 (1.71) | 2.2 (1.39) | 1.6 (1.00) |
| Alocação de Memória (Memória em Execução 10k) | 32.6 (1.07) | 39.5 (1.30) | 30.5 (1.00) |

Figura 34 – Tabela comparativa dos resultados obtidos da métrica de desempenho em relação aos testes do benchmark utilizado

# 6 Conclusão

## 6.1 Conclusões Finais

Neste trabalho, foram realizadas análises comparativas entre as ferramentas front-end Angular, React e Vue, considerando critérios como popularidade, maturidade, estabilidade, curva de aprendizado, disponibilidade de recursos humanos e desempenho no contexto SPA.

Cada uma dessas ferramentas possui suas próprias vantagens e desvantagens, e a escolha entre elas depende das necessidades e requisitos específicos de cada projeto. O Angular se destaca por sua estrutura robusta e escalabilidade, sendo uma opção atraente para projetos de grande porte que exigem um alto nível de controle e organização. O React, por sua vez, possui uma comunidade ativa e um ecossistema maduro, tornando-o uma escolha popular para projetos que valorizam a reatividade e a modularidade. Já o Vue, com sua curva de aprendizado suave e abordagem simples, pode ser uma opção atraente para desenvolvedores iniciantes ou projetos menores.

Com base na comparação dos dados coletados em termos de Popularidade/Maturidade, tanto React quanto Vue apresentam uma classificação alta, enquanto Angular tem uma classificação moderada.

No que diz respeito aos resultados obtidos nos testes de desempenho de carregamento inicial da aplicação modelo implementada e métricas de benchmark no contexto SPA, Vue obteve um desempenho geralmente superior em relação ao Angular e React em várias áreas. No entanto, é importante observar que esses resultados podem variar dependendo do tamanho e complexidade do aplicativo, bem como do cenário de uso específico.

Dentro do escopo proposto por este trabalho e pelas análises dos dados coletados sobre cada métrica, Vue se destacou principalmente por seu desempenho.

## 6.2 Contribuições do Trabalho

Este trabalho contribui para o entendimento das principais características e diferenças entre as ferramentas front-end Angular, React e Vue, fornecendo uma análise comparativa baseada em critérios relevantes. As conclusões e insights obtidos podem auxiliar desenvolvedores e equipes de projeto na tomada de decisões informadas ao escolher uma ferramenta front-end para seus projetos.

Além disso, este estudo também apresentou uma abordagem metodológica para a

avaliação de ferramentas front-end, destacando métricas e critérios importantes a serem considerados. Essa estrutura pode ser utilizada como referência para futuras pesquisas e análises comparativas de outras ferramentas front-end.

Por fim, é importante ressaltar que este trabalho não esgota todas as possibilidades de análise e comparação entre as ferramentas. Existem outras dimensões e critérios que podem ser explorados, bem como a evolução constante das próprias ferramentas, o que sugere que estudos futuros podem complementar e expandir as descobertas aqui apresentadas.

## 6.3 Trabalhos futuros

Além das comparações realizadas neste estudo entre as ferramentas front-end Angular, React e Vue, existem diversas oportunidades para futuras pesquisas e aprofundamentos nesse campo. Algumas áreas de estudo sugeridas incluem:

Análise de desempenho avançada: Embora este estudo tenha explorado métricas básicas de desempenho, como linhas de código, tamanho de pacotes e consumo de memória, uma investigação mais aprofundada poderia abordar o desempenho em cenários mais complexos. Isso envolveria a avaliação do desempenho das ferramentas em aplicações com maior volume de dados e operações mais intensivas.

Escalabilidade e desempenho em ambientes de produção: Uma extensão natural deste trabalho seria a realização de testes de escalabilidade e desempenho em ambientes reais de produção. Isso permitiria examinar como as ferramentas se comportam sob tráfego significativo e em situações de alta demanda.

Usabilidade e experiência do usuário: Além das métricas técnicas, seria relevante investigar a usabilidade e a experiência do usuário proporcionadas por cada uma das ferramentas. Isso poderia incluir testes de usabilidade, coleta de feedback dos usuários e comparação das melhores práticas de design e interação.

Integração com o Back-End: Embora este estudo tenha se concentrado exclusivamente no desenvolvimento front-end, seria interessante realizar uma análise comparativa das ferramentas em conjunto com diferentes tecnologias e frameworks de back-end. Isso permitiria avaliar como as ferramentas se integram a diferentes arquiteturas de back-end, APIs e tecnologias de persistência de dados.

Essas são apenas algumas sugestões para trabalhos futuros, e cada uma delas pode ser explorada de maneira mais aprofundada e detalhada, contribuindo para um maior entendimento das diferenças e características de cada ferramenta front-end no contexto de desenvolvimento de aplicações.

# Referências


AGGARWAL, S. et al. Modern web-development using reactjs. *International Journal of Recent Research Aspects*, v. 5, n. 1, p. 133–137, 2018. Citado na página 26.

ALDANIAL. *AlDanial/cloc*. 2021. Disponível em: <https://github.com/AlDanial/cloc>. Citado na página 30.

ALTEXSOFT. *React vs Angular Compared: Which One Suits Your Project Better?* 2018. <https://www.altexsoft.com/blog/engineering/react-vs-angular-compared-which-one-suits-your-project-better/>. Citado na página 20.

AMBLER, T.; CLOUD, N. *JavaScript Frameworks for Modern Web Dev*. New York: Apress, 2015. Citado na página 13.

Angular. *Angular - Template Syntax*. 2019. [Online]. Disponível em: <https://angular.io/guide/template-syntax#binding-syntax-an-overview>. Citado na página 25.

BALARAMAN, S. Review of "pro jsf and ajax: Building rich internet components by jonas jacobi and john fallows, "apress, 2006, isbn: 1590595807. *ACM Queue*, v. 5, p. 44, 04 2007. Citado na página 24.

BHASKAR, A.; MANJUNATH, A. An interpretation and anatomization of angular: A google web framework. *International Research Journal of Engineering and Technology (IRJET)*, v. 7, n. 05, 2020. Citado na página 24.

BODUCH, A. *React and React Native*. Birmingham: Packt Publishing, 2018. Citado na página 25.

BORZEMSKI, L. et al. Information technology architecture for optimal reporting. *Issues In Information Systems*, p. 235–247, 2014. Disponível em: <https://doi.org/10.48009/1_iis_2014_224-234>. Citado na página 20.

BUNDLEPHOBIA. *BundlePhobia*. 2023. <https://bundlephobia.com/scan>. Citado 2 vezes nas páginas 49 e 50.

BUNDLEPHOBIA | SIZE OF NPM DEPENDENCIES. 2023. <https://bundlephobia.com/>. Citado na página 31.

CHEN, J. e. a. The importance of learning curve in evaluating front-end frameworks. *Journal of Web Development*, 2020. Citado na página 27.

DevMedia. *Trabalhando com DOM em JavaScript*. 2023. <https://www.devmedia.com.br/trabalhando-com-dom-em-javascript/29039>. Citado na página 22.

DUARTE, N. F. B. *Frameworks e Bibliotecas Javascript*. Dissertação (Mestrado) — Instituto Superior de Engenharia do Porto, Porto, 2015. Citado na página 27.

DZIWOKI, M. *AngularJS vs Angular*. 2017. <https://gorrion.io/blog/angularjs-vs-angular>. 2023. Citado na página 24.

EDWIN, N. M. Software frameworks, architectural and design patterns. *Journal of Software Engineering and Applications*, v. 07, n. 08, p. 670–678, 2014. Citado na página 24.

FARWELL, M. Yakov fain on angular. *IEEE Software*, IEEE Computer Society, v. 34, n. 06, p. 109–112, 2017. Citado na página 25.

FEDOSEJEV, A. React. js essentials. Packt Publishing, 2016. Citado na página 25.

FERNÁNDEZ-VILLAMOR, J. I.; DÍAZ-CASILLAS, L.; IGLESIAS, C. Á. A comparison model for agile web frameworks. v. 14, 2008. Citado na página 21.

FINK, G.; FLATOW, I. *Pro Single Page Application Development Using Backbone.js and ASP.NET*. Berkeley, CA: Apress, 2014. Citado na página 20.

FREEMAN, A. Using data bindings. In: *Pro Angular: Build Powerful and Dynamic Web Apps*. [S.l.]: Springer, 2022. p. 249–273. Citado na página 25.

GALLAGHER, K.; HATCH, A.; MUNRO, M. Software architecture visualization: An evaluation framework and its application. *IEEE Transactions on Software Engineering*, IEEE, v. 34, n. 2, p. 260–270, Mar 2008. Citado na página 14.

GARRETT, J. J. Ajax: A new approach to web applications. p. 1–5, 2005. Citado na página 24.

GERDESSEN, A. *Framework comparison method*. Dissertação (Mestrado) — University of Amsterdam, Amsterdam, 2007. Citado na página 21.

GHAZARYAN, A. *Choosing the right framework for your project*. DZone, 2023. Disponível em: <https://dzone.com/articles/choosing-the-right-framework-for-your-project>. Citado na página 28.

GOCHKE, A. Single page web application technologies. *International Journal for Research in Applied Science and Engineering Technology*, v. 7, p. 1692–1697, 05 2019. Citado na página 19.

Google Chrome Labs. *Perf Track*. 2020. <https://github.com/GoogleChromeLabs/perf-track>. Citado na página 32.

GoogleChromeLabs. *Perf Track - GitHub Repository*. 2023. <https://github.com/GoogleChromeLabs/perf-track>. Citado na página 60.

HERMAN, D.; RAYCHEV, V.; TOBIN-HOCHSTADT, S. Safe & efficient gradual typing for TypeScript. In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. [s.n.], 2015. Disponível em: <https://dl.acm.org/doi/abs/10.1145/2676726.2676971>. Citado na página 23.

HISTORY, S. *Star History - Angular, React, Vue*. 2023. <https://star-history.com/angular/angularfacebook/reactvuejs/vueDate>. Citado na página 41.

INC., V. *GitHub Statistics*. 2023. <https://vesoft-inc.github.io/github-statistics/>. Citado 3 vezes nas páginas 37, 38 e 39.

JOHNSON, A.; SMITH, B.; DAVIS, C. The importance of framework maturity and stability. *Software Engineering Journal*, v. 25, n. 2, p. 45–58, 2019. Citado na página 28.

JOHNSON, R. E. Components, frameworks, patterns. Association for Computing Machinery, New York, NY, USA, p. 10–17, 1997. Disponível em: <https://doi.org/10.1145/258366.258378>. Citado na página 17.

KRAUSE, S. *js-framework-benchmark*. 2021. <https://github.com/krausest/js-framework-benchmark>. Citado 4 vezes nas páginas 31, 56, 57 e 58.

KYRIAKIDIS, A.; MANIATIS, K.; YOU, E. *The majesty of Vue. js*. Leanpub: Packt Publishing, 2016. 14–30 p. Citado na página 27.

LEE, H. et al. Safe: Formal specification and implementation of a scalable analysis framework for ecmascript. Citeseer, p. 96, 2012. Citado 2 vezes nas páginas 23 e 28.

LEFF, A.; RAYFIELD, J. Web-application development using the model/view/controller design pattern. In: *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*. Seattle, WA, USA: IEEE, 2001. p. 118–127. Citado na página 17.

MALMSTRöM, T. J. *Structuring Modern Web Applications*. Dissertação (Mestrado) — KTH School of Computer Science and Communication, February 2014. Citado na página 21.

MESBAH, A.; DEURSEN, A. van. Migrating multi-page web applications to single-page ajax interfaces. 2007. Citado na página 18.

MINNICK, J. L. *Responsive Web Design with HTML5 & CSS*. Boston, MA: Cengage, 2021. Citado na página 22.

MOLIN, E. *Comparison of Single-Page Application Frameworks*. Dissertação (Mestrado) — Luleå University of Technology, 2016. Disponível em: <https://www.diva-portal.org/smash/get/diva2:1037481/FULLTEXT01.pdf>. Citado na página 19.

NORMAND, E. *Why do we use Web Frameworks?* 2022. Disponível em: <https://ericnormand.me/article/why-web-frameworks>. Citado na página 17.

PATRICIAP. *generate_transactions: Algorithm for Generating Simulated Financial Transactions*. 2023. <https://github.com/PatriciaP/generate_transactions/>. Citado na página 46.

PATRICIAP. *GitHub Repository: PatriciaP*. 2023. <https://github.com/PatriciaP>. Citado na página 44.

PAULSON, L. D. Building rich web applications with ajax. *Computer*, v. 38, n. 10, p. 14–17, Oct. 2005. Citado na página 24.

PEKARSKY, M. Does your web app needs a front-end framework? *Stack Overflow*, 2020. Disponível em: <https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/>. Citado na página 13.

POP, D.-P.; ALTAR, A. Designing an mvc model for rapid web application development. *Procedia Engineering*, v. 69, p. 1172–1179, jan. 2014. Citado na página 17.

PRESTON; SO SO, P. React. *Decoupled Drupal in Practice: Architect and Implement Decoupled Drupal Architectures Across the Stack*, Springer, p. 313–334, 2018. Citado na página 26.

React Documentation. *Components and Props*. 2019. URL: <https://reactjs.org/docs/components-and-props.html>. Citado na página 26.

React Documentation. *State and Lifecycle*. 2019. URL: <https://reactjs.org/docs/state-and-lifecycle.html>. Citado na página 26.

SAKS, E. Javascript frameworks: Angular vs react vs vue. 2019. Citado na página 26.

SALAS-ZÁRATE, M. d. P. et al. A list of requirements and best practices for web development applied to web frameworks. *IEEE Latin America Transactions*, IEEE, v. 13, n. 3, p. 1010–1015, 2015. Citado na página 21.

SANTOS, K. Importância do ux: A perspectiva do usuário como a espinha dorsal de qualquer fluxo de experiência. *Revista de Experiência do Usuário*, v. 4, n. 2, p. 25–30, 2019. Citado na página 18.

SAUVé, P. *O que é um framework?* 2007. <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Accessed on: 15 May 2022. Citado na página 13.

SAXENA, P. et al. A symbolic execution framework for javascript. IEEE Xplore, may 2010. Citado na página 22.

SCHMIDT, D. et al. Patterns, frameworks, and middleware: Their synergistic relationships. 2000. Citado na página 17.

SCHWARZMüLLER, M. *Angular, ReactJS & Vue.js - Quickstart & Comparison*. 2023. Udemy. Disponível em: <https://qat.udemy.com/course/angular-reactjs-vuejs-quickstart-comparison/>. Citado na página 42.

SILVA, S. B. C. d. Análise do uso de boas práticas no front-end de um sistema seb: um estudo de caso com vue. js. 2021. Citado na página 26.

SMITH, J. The impact of popular frameworks in web development. *Software Development Review*, v. 15, n. 4, p. 78–92, 2020. Citado 2 vezes nas páginas 27 e 28.

STACK Overflow Developer Survey 2022. 2022. <https://survey.stackoverflow.com/2022/>. Citado 3 vezes nas páginas 14, 15 e 37.

STEYER, R. *Vue.js in Depth: The Vue Instance, Vue Templates, and Data Binding*. Wiesbaden: Springer Fachmedien Wiesbaden, 2022. 43–69 p. ISBN 978-3-658-37596-6. Disponível em: <https://doi.org/10.1007/978-3-658-37596-6_4>. Citado na página 27.

TRENDS, G. *Google Trends - Explore - React, Angular, Vue*. 2023. <https://trends.google.com/trends/explore?cat=31&date=today%205-y&q=React,Angular,vue>. Citado na página 34.

TRENDS npm. *npm Trends - Angular vs React vs Vue*. 2023. <https://npmtrends.com/angular-vs-react-vs-vue>. Citado na página 35.

UDEMY. *Angular, ReactJS & Vue.js - Quickstart & Comparison*. 2023. Online Course. Disponível em: <https://www.udemy.com/course/angular-reactjs-vuejs-quickstart-comparison/?couponCode=D_0623>. Citado na página 43.

VOUTILAINEN, J.-P.; MIKKONEN, T.; SYSTÄ, K. Synchronizing application state using virtual dom trees. v. 9881, p. 142–154, 2016. Citado na página 23.

WENDLER, R. The maturity of maturity model research: A systematic mapping study. *Information and Software Technology*, v. 54, n. 12, p. 1317–1339, 2012. ISSN 0950-5849. Special Section on Software Reliability and Security. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584912001334>. Citado na página 27.

XING, Y.; HUANG, J.; LAI, Y. Research and analysis of the front-end frameworks and libraries in e-business development. In: *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*. New York, NY, USA: Association for Computing Machinery, 2019. (ICCAE 2019), p. 68–72. ISBN 9781450362870. Disponível em: <https://doi.org/10.1145/3313991.3314021>. Citado na página 29.