Análise comparativa de bibliotecas e *frameworks JavaScript* emergentes para o desenvolvimento *Front-End* de aplicações web

Ana Carolina Jungbluth
Universidade Tecnológica Federal do Paraná (UTFPR)
Medianeira, Brasil
anajungbluth@alunos.utfpr.edu.br

Ricardo Sobjak
Universidade Tecnológica Federal do Paraná (UTFPR)
Medianeira, Brasil
ricardosobjak@utfpr.edu.br

Alessandra Bortoletto Garbelotti Hoffmann
Universidade Tecnológica Federal do Paraná (UTFPR)
Medianeira, Brasil
hoffmann@utfpr.edu.br

***Resumo*—Currently, with the vast range of JavaScript frameworks and libraries available for web development, choosing the most suitable one for a project can be challenging. This article presents a comparative analysis of two emerging JavaScript frameworks or libraries, Svelte and Solid, using as a reference one of the leading JavaScript frameworks or libraries: React, all selected after research. The study evaluates the tools based on criteria such as performance, compatibility, the use of resources, identifying the strengths and limitations of each. The aim is to provide a clear view of the advantages and disadvantages of each framework or library.**

***Keywords*—Web Development; React; Performance Comparison.**

***Resumo*—Atualmente, com a vasta gama de *frameworks* e bibliotecas *JavaScript* disponíveis para o desenvolvimento web, escolher o mais adequado para um projeto pode ser desafiador. Este artigo realiza uma análise comparativa entre dois *frameworks* ou bibliotecas *JavaScript* emergentes, *Svelte* e *Solid*, utilizando como referência um dos principais *framework* ou biblioteca *JavaScript*: o *React*, todos escolhidos após pesquisas. O estudo avalia as ferramentas com base em critérios de desempenho, compatibilidade e uso de recursos, identificando os pontos fortes e as limitações de cada uma. O objetivo é oferecer uma visão clara das vantagens e desvantagens de cada *framework* ou biblioteca.**

***Palavras-chave*—Desenvolvimento Web; *React*; Comparação de Desempenho.**

I. Introdução

Com o avanço das tecnologias, principalmente a internet, as necessidades do mercado estão cada vez maiores. Nesse cenário em constante evolução a busca por aplicações web dinâmicas tem se destacado de forma proeminente. Essa crescente demanda é impulsionada pela necessidade de melhorar a experiência dos usuários finais. Para o desenvolvimento de aplicações web que possuam maior interatividade, níveis avançados e facilidade de manutenção, a linguagem de programação *JavaScript* é amplamente utilizada [1] e se destaca como a linguagem de programação *Front-End* mais popular do mundo [2]. Segundo pesquisas do StackOverflow [3], JavaScript tem a preferência de 63,6% dos entrevistados.

Entretanto, somente a utilização da linguagem *JavaScript* não consegue suprir as demandas do mercado atual. Isso ocorre devido à extensa gama de aplicações que *JavaScript* abrange, bem como à diversidade de requisitos de programação. Como resultado, foram desenvolvidos os chamados *frameworks* e bibliotecas *JavaScript* [4].

Segundo [5] as bibliotecas e os *frameworks*, em sua essência, são conjuntos de ferramentas e funcionalidades pré-definidas que estabelecem a base fundamental de um projeto. Sua principal finalidade reside na simplificação e agilização do processo de desenvolvimento, tornando-o mais eficiente. Além disso, a presença de *frameworks* e bilbiotecas é essencial para atrair uma comunidade ativa para uma linguagem de programação, contribuindo significativamente para sua adoção.

Existe uma ampla gama de *frameworks* e bibliotecas *JavaScript* disponíveis, sendo o *React, Vue.js* e o *Angular* as principais bibliotecas ou *frameworks JavaScript* do lado do cliente, conforme indicado por pesquisas realizadas pelo StackOverflow [6] e StateOfJs [7]. O foco deste trabalho é realizar uma análise comparativa entre duas bibliotecas ou *frameworks JavaScript* emergentes, que tem ganhado destaque nos últimos anos, *versus* a biblioteca ou *framework*, tido como principal referência no desenvolvimento web atual. Serão considerados diversos elementos, como suporte ao desenvolvimento, a qualidade e eficiência das ferramentas de componentes visuais, a compatibilidade dos *frameworks* ou bibliotecas com diferentes navegadores, o reuso de conhecimentos, bem como o consumo de recursos de memória e Central Processing Unit (CPU),

tempo de inicialização das aplicações e o tamanho final dos pacotes. Além disso, serão avaliados os tempos de execução dos métodos de Create, Read, Update e Delete (CRUD). O propósito desta avaliação é fornecer à comunidade de desenvolvedores uma visão ampla, determinando se os dois *fameworks* ou bibliotecas emergentes superam o *framework* ou biblioteca principal em termos de desempenho e recursos oferecidos.

II. Materiais e Métodos

O desenvolvimento do trabalho está dividido em três partes conforme Figura 1. Na primeira parte a seleção de três bibliotecas ou *frameworks*: dois emergentes e um de referência, ou seja, o principal *framework* ou biblioteca utilizado pela comunidade de desenvolvedores. A segunda parte se concentra no desenvolvimento de uma aplicação em cada um dos *frameworks* ou bibliotecas selecionados. A terceira parte consiste em definir os critérios para avaliar as bibliotecas ou *frameworks* selecionados.

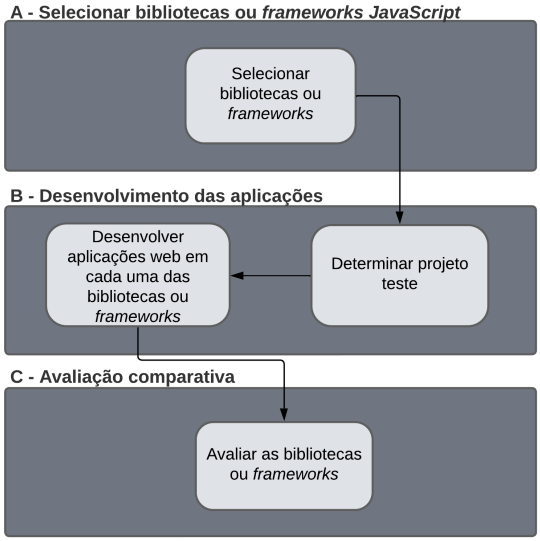

Figura 1. Autoria Própria - Fluxograma do desenvolvimento do trabalho

A. Selecionar os frameworks ou bibliotecas para a realização do estudo

Como referência para a comparação das bibliotecas ou *frameworks JavaScript*, será escolhido a biblioteca ou *framework* de referência. Baseado em pesquisas os principais são o *React, Angular* e o *Vue.js*. Na Tabela I é apresentado os dados das principais bibliotecas e *framework*.

No quesito interesse dos desenvolvedores, observa-se que o *Vue.js* lidera com 51%, seguido de perto pelo *React* com 47%, enquanto o *Angular* apresenta um baixo nível de interesse, de apenas 20%. Em termos de popularidade no GitHub [8], o *React* ocupa a primeira posição, com 214.000 estrelas, seguido de perto pelo *Vue.js*, que tem 205.000 estrelas no seu repositório. Enquanto o *Angular*, apesar de ser o mais antigo, tem apenas 90.600 estrelas em seu repositório. Quanto à utilização, o *React* lidera tanto nas pesquisas do StackOverflow [6], com uma taxa de uso de 40,46%, quanto nas do *StateOfJS* [7], com 82% de uso. O *Angular* mantém consistentemente a segunda posição nas duas pesquisas, enquanto o *Vue.js* se posiciona em terceiro lugar. Com base nesses dados, o *React* é a principal biblioteca de desenvolvimento *Front-End*, liderando em termos de popularidade no GitHub e utilização, sendo esta escolhida como referência para o estudo.

Tabela I
Dados das principais bibliotecas e framework

Framework	Interesse	Estrelas Github	Uso StackOverflow	Uso *StateOfJS*
React	47%	214.000	40,58%	82%
Angular	20%	90.600	17,46%	49%
Vue.js	51%	205.000	16,38%	46%

Adaptado de [7], [8], [6]

Com base em pesquisas, foram selecionados um *framework JavaScript* e uma biblioteca *JavaScript* emergentes: *Svelt* e *Solid*. A escolha de comparar esse *framework* e biblioteca emergentes com a principal biblioteca *React*, foi devido ambos *framework* e biblioteca terem apresentado um crescente interesse da comunidade de desenvolvedores, como indicado pelas pesquisas de [7]. A Tabela II apresenta os dados do *framework* e biblioteca emergentes selecionados. Onde os dados são o nível de interesse deles segundo o *StateOfJS*, o total de ganho de estrelas nos últimos três anos em seus repositórios no Github e sua porcentagem de utilização segundo pesquisas do StackOverflow.

Tabela II
Dados da biblioteca e framework emergente

Framework	Interesse	Ganho de Estrelas	Uso
Svelte	70%	35.600	6,62%
Solid	66%	22.900	1,36%

Adaptado de [7], [9], [10], [11], [6]

O *framework Svelte* e a biblioteca *Solid* despertam significativo interesse na comunidade de desenvolvedores, com taxas de interesse de 70% e 66%, respectivamente, além de terem conquistado um grande número de estrelas nos últimos anos, o que indica sua crescente popularidade. Além disso, *Svelte* é utilizado em 6,62% dos projetos, demonstrando sua presença significativa na comunidade de desenvolvimento. Em contraste, o uso do *Solid* é mais limitado, abrangendo apenas 1,36% dos

projetos.

B. Desenvolvimento da aplicação web para o estudo

As aplicações web a serem desenvolvidas são um sistema de cadastro de livros de uma biblioteca, permitindo adicionar, listar, atualizar e deletar livros, garantindo assim um registro organizado da coleção da biblioteca.

O equipamento que será usada para desenvolver as aplicações é um notebook *Acer* da série *Aspire*, com o modelo A315-23. Este notebook é equipado com um processador *AMD Ryzen 3 3250U*, que possui dois núcleos de processamento operando a uma velocidade de *clock* de 2.6 GHz. Além disso, o notebook possui 8GB de memória *Random Access Memory* (RAM) e *Solid State Drive* (SSD) com capacidade de 240GB. O sistema operacional do notebook utilizado para o desenvolvimento é *Microsoft Windows 11 Home Single Language*.

Para o desenvolvimento das aplicações web será utilizada a *Integrated Development Environment* (IDE) *Visual Studio Code*[1] da *Microsoft*. Além disso serão utilizados o *framework JavaScript Svelte* e as bibliotecas *JavaScript React* e *Solid* para o desenvolvimento do *Front-End*. Por primeiro no desenvolvimento *Front-End*, serão criados os componentes visuais e organizados em páginas funcionais, além de configurar as navegações entre as páginas do CRUD. Essa etapa permitirá avaliar a facilidade de criação e reutilização dos componentes, bem como a disponibilidade de bibliotecas de componentes para cada uma das ferramentas escolhidas.

Para a parte do *Back-End* das aplicações, será desenvolvida uma *Application Programming Interface* (API) *Representational State Transfer* (REST)[2] que conectará as páginas da aplicação e permitirá acessar e manipular os dados. Essa API será desenvolvida em *Node.js*[3] juntamente com o *framework Express* devido à sua capacidade de fornecer um sistema de rotas completo. Utilizando solicitações HTTP (Get, Post, Put e Delete) para realizar operações CRUD, como obtenção de registros, criação de novos registros, atualização e exclusão (Figura 2). Além disso, a API estará conectada a um banco de dados para armazenamento e recuperação eficientes de dados. O banco de dados que será utilizado é o *MongoDB*, pois ele tem uma integração fácil com *Node.js*.

Na Figura 2 é ilustrado o funcionamento das aplicações. Essas aplicações utilizam o método *fetch()* para realizar as HTTP *request* à API. Essas *request* podem ser de diferentes tipos, como POST (para adicionar um livro), GET (para obter todos os livros) ou GET (para obter um livro específico), PUT (para atualizar um livro) e DELETE (para deletar um livro).

[1]https://code.visualstudio.com/

[2]API REST é uma interface que permite a comunicação entre sistemas, utilizando métodos HTTP para acessar e manipular recursos de forma estruturada.

[3]https://nodejs.org/pt

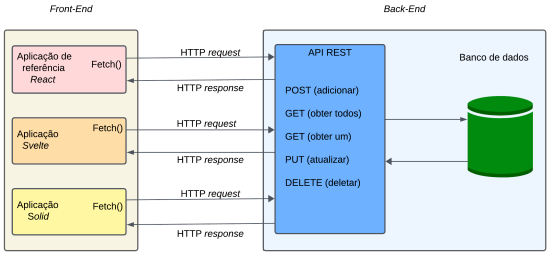

Figura 2. Autoria Própria - Representação do funcionamento das aplicações

A API REST vai atuar como uma camada intermediária entre as aplicações e o banco de dados. O navegador utilizado para rodar as aplicações é o FireFox.

C. Avaliação comparativa

Nessa seção será descrito os critérios para avaliar as bibliotecas e o *framework JavaScript* selecionados. Assim avaliando seu suporte, desempenho e compatibilidade. Os critérios estabelecidos são:

- **Suporte ao desenvolvimento:** Para se escolher um *framework* ou qualquer ferramenta, um fator importante considerado é a disponibilidade de suporte, especialmente em situações de problemas ou dúvidas durante o desenvolvimento [12]. Assim será verificado se o *framework* e as bibliotecas contam com uma documentação detalhada, que oferece uma variedade de exemplos, e se é mantida atualizada. Além disso, será examinada a extensão do suporte fornecido pela comunidade de desenvolvedores, bem como se as ferramentas tem disponível um canal de comunicação para sanar dúvidas.
- **Qualidade e eficiência das ferramentas de componentes visuais:** Verificar a qualidade das ferramentas de componentes visuais, que otimiza o desenvolvimento das interfaces [13]. Para isso será avaliado a facilidade da criação de componentes visuais, a disponibilidade e diversidade de bibliotecas de componentes e a reutilização de componentes em diferentes partes do aplicativo.
- **Tempo de execução dos métodos CRUD:** O tempo de espera da execução de um método pode interferir na experiência do usuário. Assim será avaliado os tempos de execução dos métodos CRUD. Para calcular o tempo de execução será utilizada a função *console.time()* e *console.timeEnd()* do *JavaScript*. Essas funções medirão apenas o tempo que o código leva para ser executado no ambiente local, excluindo o tempo de espera por operações assíncronas, como operações de rede. Para uma melhor análise será realizado dez teste e calculada a média.

- **Consumo de memória:** O consumo de memória é importante para um bom desempenho da aplicação. Assim será avaliado o consumo de memória na execução dos métodos CRUD, onde será realizado dez teste e calculado a média. Para medir o consumo de memória vai ser utilizada a ferramenta da Google, o *Chrome DevTools*[4] na aba Desempenho.
- **Consumo de CPU:** Além do consumo de memória o de CPU também é importante para o desempenho. Por isso, será avaliado o consumo de CPU na execução dos métodos CRUD, onde será realizado dez teste e calculado a média. Para medir o consumo de CPU vai ser utilizada a ferramenta da Google, o *Chrome DevTools* na aba Desempenho.
- **Compatibilidade com diferentes navegadores:** Testar a compatibilidade das aplicações desenvolvidas com diferentes navegadores, ou seja, verificar se as aplicações suportam diferentes navegadores. Para isso foi escolhido os navegadores Google Chrome na versão 118.0.5993.88, *Microsoft Edge* na versão 118.0.2088.46 e o *FireFox* na versão 118.0.2.
- **Tempo de renderização inicial:** O tempo para inicializar uma página pode afetar a experiência do usuário. Por isso, será avaliado o tempo de renderização inicial das aplicações. Para medir o tempo de renderização será utilizado o *Chrome DevTools* ferramenta da Google, na aba *Network*. Para uma melhor análise será feito dez teste e calculado a média.
- **Reuso de conhecimentos:** Melhorar a curva de aprendizado incorporando conhecimentos e abordagens de outras tecnologias, enquanto se mantém alinhado com as práticas aceitas pela comunidade de desenvolvedores. [13]. Assim será avaliado a familiaridade da sintaxe, a facilidade de adoção e se o *framework* e as bibliotecas seguem padrões de programação amplamente usados pelo mercado.
- **Tamanho final do pacote:** o tamanho do pacote das aplicações influenciam em sua performance e na experiencia do usuário. Por isso, será avaliado o tamanho final das três aplicações.

Para os critérios de **suporte ao desenvolvimento, qualidade e eficiência das ferramentas de componentes visuais** e **reuso de conhecimentos**, será utilizada a escala de *Likert* [14] para avaliar cada categoria referente a cada critério.

III. Conclusão

O presente estudo foi conduzido com o objetivo de desenvolver uma análise comparativa entre dois *frameworks* ou bibliotecas emergentes, *Svelte* e *Solid*, com o principal *framework* ou biblioteca de referência: o *React*, escolhido entre os três principais: *React, Angular* e *Vue.js*. A metodologia apresentada, baseada em critérios bem definidos como desempenho, compatibilidade com navegadores e uso de recursos, mostra-se adequada para conduzir uma análise detalhada do comportamento e eficiência dessas ferramentas no desenvolvimento de aplicações web. Embora os resultados ainda não tenham sido discutidos, a expectativa é de que este estudo forneça *insights* valiosos sobre as vantagens que cada ferramenta apresenta em relação aos critérios estabelecidos, auxiliando os desenvolvedores na escolha do *framework* ou biblioteca mais adequado para suas necessidades específicas.

[4]https://developer.chrome.com/docs/devtools?hl=pt-br

Agradecimentos

Agradeço à UTFPR e aos professores pelo apoio e orientação essenciais no desenvolvimento desta pesquisa.

Referências

[1] M. Levlin, "Dom benchmark comparison of the front-end javascript frameworks react, angular, vue, and svelte," Master's thesis, 2020. [Online]. Available: https://www.doria.fi/handle/10024/177433
[2] C. L. Mariano, "Benchmarking javascript frameworks," Master's thesis, 2017. [Online]. Available: https://arrow.tudublin.ie/scschcomdis/94/
[3] StackOverflow, "Programming, scripting, and markup languages," 2023a. [Online]. Available: https://survey.stackoverflow.co/2023/
[4] A. Gizas, S. Christodoulou, and T. Papatheodorou, "Comparative evaluation of javascript frameworks," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 513–514. [Online]. Available: https://www.researchgate.net/publication/254008963_Comparative_evaluation_of_JavaScript_frameworks
[5] J. G. C. de Camargos, J. F. Coelho, H. F. Villela, and J. P. Aramuni, "Uma análise comparativa entre os frameworks javascript angular e react," *Computação & Sociedade*, vol. 1, no. 1, 2019. [Online]. Available: http://revista.fumec.br/index.php/computacaoesociedade/article/view/7307
[6] StackOverflow, "Web frameworks and technologies," 2023b. [Online]. Available: https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies
[7] StateOfJS, "Front-end frameworks," 2022a. [Online]. Available: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/
[8] Github, "Let's build from here," 2023. [Online]. Available: https://github.com/
[9] M. Rambeau, "2020 javascript rising stars," 2020. [Online]. Available: https://risingstars.js.org/2020/en#section-framework
[10] ——, "2021 javascript rising stars," 2021. [Online]. Available: https://risingstars.js.org/2021/en#section-framework
[11] ——, "2022 javascript rising stars," 2022. [Online]. Available: https://risingstars.js.org/2022/en#section-framework
[12] H. K. Ferreira and J. D. Zuchi, "Análise comparativa entre frameworks frontend baseados em javascript para aplicações web," *Revista Interface Tecnológica*, vol. 15, no. 2, pp. 111–123, 2018. [Online]. Available: https://revista.fatectq.edu.br/interfacetecnologica/article/view/502
[13] A. Hudli, S. Hudli, and R. Hudli, "An evaluation framework for selection of mobile app development platform," in *Proceedings of the 3rd International Workshop on Mobile Development Lifecycle*, 2015, pp. 13–16. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2846661.2846678
[14] A. Oliveira, "O que é escala likert e como aplicá-la na pesquisa?" *MindMiners*, 2023. [Online]. Available: https://mindminers.com/blog/entenda-o-que-e-escala-likert/