REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

ANÁLISE COMPARATIVA DE BIBLIOTECAS E FRAMEWORKS JAVASCRIPT EMERGENTES PARA O DESENVOLVIMENTO FRONT-END DE APLICAÇÕES WEB

COMPARATIVE ANALYSIS OF EMERGING JAVASCRIPT LIBRARIES AND FRAMEWORKS FOR FRONT-END WEB APPLICATION DEVELOPMENT

ANÁLISIS COMPARATIVO DE BIBLIOTECAS Y FRAMEWORKS JAVASCRIPT EMERGENTES PARA EL DESARROLLO FRONT-END DE APLICACIONES WEB

Luana Copetti
Ciência da computação UTFPR, Brasil
E-mail: luanacopetti@alunos.utfpr.edu.br
André Sandmann
Doutorado: Engenharia Agrícola, UTFPR, BRASIL
E-mail: sandmann@utfpr.edu.br
Ana Carolina Jungbluth
Ciência da computação UTFPR, Brasil
E-mail: anajungbluth@alunos.utfpr.edu.br
Kelyn Schenatto
Doutorado: Engenharia Agrícola **UTFPR, Brasil**
E-mail: kschenatto@utfpr.edu.br
Ricardo Sobjak
Doutorado: Engenharia Agrícola, UTFPR, Brasil
E-mail: ricardosobjak@utfpr.edu.br
Liliane Hellmann
Doutorado em Engenharia Química pela Universidade Estadual do Oeste do Paraná, Brasil
E-mail: lilianehellmann@utfpr.edu.br
Jeconias Rocha Guimarães
Licenciatura Plena em Física pela Universidade Federal do Pará, Brasil

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

E-mail: jrguimaraes@utfpr.edu.br

Renato Hallal

Licenciatura Plena em Matemática (UFSCar), Brasil

E-mail: renatohallal@utfpr.edu.br

Resumo

O desenvolvimento web tem acompanhado a crescente demanda por aplicações interativas e com bom desempenho, tornando frameworks e bibliotecas JavaScript importantes para a construção de aplicações. Nesse contexto, a escolha da tecnologia mais adequada pode ser desafiadora diante da variedade de opções existentes. Este trabalho apresenta uma análise comparativa entre Svelte e Solid, consideradas tecnologias emergentes, e React, uma tecnologia consolidada no desenvolvimento web. O objetivo foi comparar o desempenho e o uso de recursos das três tecnologias, considerando o tempo de execução dos métodos CRUD, o consumo de memória e CPU, o tempo de renderização inicial e o tamanho do build. Os resultados indicaram que o Solid apresentou o menor tempo de renderização inicial e o menor tamanho de build, enquanto o Svelte se destacou pelo menor consumo de memória e CPU e pelos menores tempos de execução na maioria das operações CRUD. O React apresentou os maiores valores em grande parte dos critérios avaliados.
Palavras-chave: desenvolvimento web; comparação de desempenho; eficiência de recursos.

Abstract

Web development has followed the growing demand for interactive applications with good performance, making JavaScript frameworks and libraries important for application development. In this context, choosing the most suitable technology can be challenging due to the variety of available options. This study presents a comparative analysis of Svelte and Solid, considered emerging technologies, and React, a well-established technology in web development. The objective was to compare the performance and resource usage of the three technologies, considering CRUD operation execution time, memory and CPU consumption, initial rendering time, and build size. The results indicated that Solid achieved the shortest initial rendering time and the smallest build size, while Svelte stood out for its lower memory and CPU consumption and shorter execution times in most CRUD operations. React presented the highest values in most of the evaluated criteria.
Keywords: web development; performance comparison; resource efficiency.

Resumen

El desarrollo web ha acompañado la creciente demanda de aplicaciones interactivas y de alto rendimiento, lo que convierte a los *frameworks* y bibliotecas de JavaScript en elementos clave para

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

la creación de aplicaciones. En este contexto, elegir la tecnología más adecuada puede resultar un desafío dada la variedad de opciones existentes. Este trabajo presenta un análisis comparativo entre Svelte y Solid —consideradas tecnologías emergentes— y React, una tecnología consolidada en el desarrollo web. El objetivo fue comparar el rendimiento y el uso de recursos de las tres tecnologías, considerando el tiempo de ejecución de las operaciones CRUD, el consumo de memoria y CPU, el tiempo de renderizado inicial y el tamaño de la compilación (*build*). Los resultados indicaron que Solid presentó el menor tiempo de renderizado inicial y el menor tamaño de compilación, mientras que Svelte destacó por un menor consumo de memoria y CPU, así como por tiempos de ejecución más rápidos en la mayoría de las operaciones CRUD. React registró los valores más elevados en gran parte de los criterios evaluados.
Palabras clave: desarrollo web; comparación de rendimiento; eficiencia de recursos.

1. 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 (LEVLIN, 2020) e se destaca como a linguagem de programação Front-End mais popular do mundo (MARIANO, 2017). Segundo pesquisas do site StackOverflow (2023a), 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 (GIZAS; CHRISTODOULOU; PAPATHEODOROU, 2012). Segundo Camargos et al. (2019), 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,

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

a presença de frameworks e bibliotecas 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 (2023b) e StateOfJS (2022). 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 elementos, 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 frameworks ou bibliotecas emergentes superam o framework ou biblioteca principal em termos de desempenho e uso de recursos.

2. Materiais e métodos

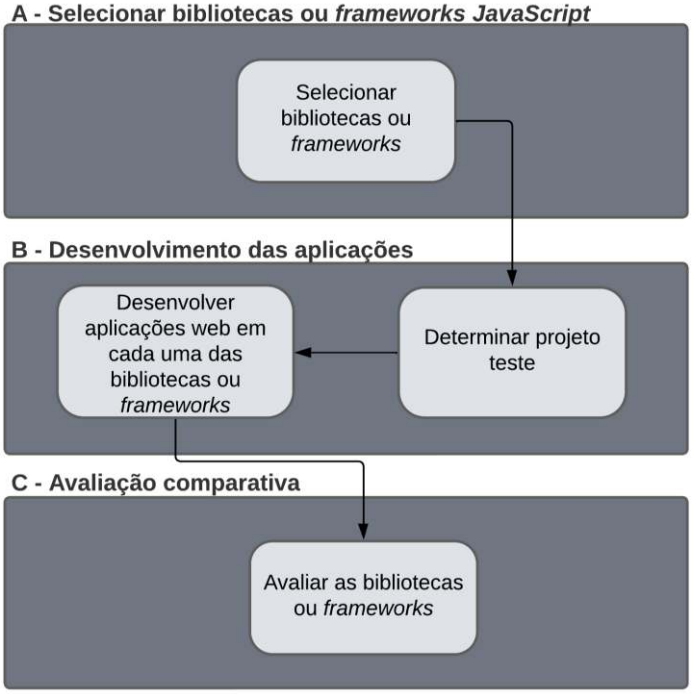
O desenvolvimento do trabalho está dividido em três partes conforme Figura 1. Na primeira parte a seleção de três bibliotecas ou frameworks: dois emergentes e um de referência, ou seja, o principal framework ou biblioteca utilizados pela comunidade de desenvolvedores. A segunda parte se concentra no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

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

Figura 1 – Fluxograma do desenvolvimento do trabalho



Fonte: Autoria própria.

3. 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 1 é apresentado os dados das principais bibliotecas e frameworks.

Tabela 1 - 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\%

Fonte: Adaptado de StateOfJS (2022), Github (2023), StackOverflow (2023b).

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 (2023), 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

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

utilização, o React lidera tanto nas pesquisas do StackOverflow (2023b), com uma taxa de uso de 40,46%, quanto nas do StateOfJS (2022), 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.

Com base em pesquisas, foram selecionados um framework JavaScript e uma biblioteca JavaScript emergentes: Svelte 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 do StateOfJS (2022). A Tabela 2 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 2 - Dados da biblioteca e framework emergente

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

Fonte: Adaptado de StateOfJS (2022), Rambeau (2020, 2021, 2022), StackOverflow (2023b).

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.

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

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

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.

2.2 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 usado 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 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), API REST é uma interface que permite a comunicação entre sistemas, utilizando métodos HTTP para acessar e manipular recursos de forma estruturada. que conectará as páginas da aplicação e permitirá acessar e manipular os dados. Essa API será desenvolvida em Node.js juntamente com o framework Express devido à sua capacidade de fornecer um sistema de rotas completo. Utilizando solicitações HTTP (Get, Post, Put

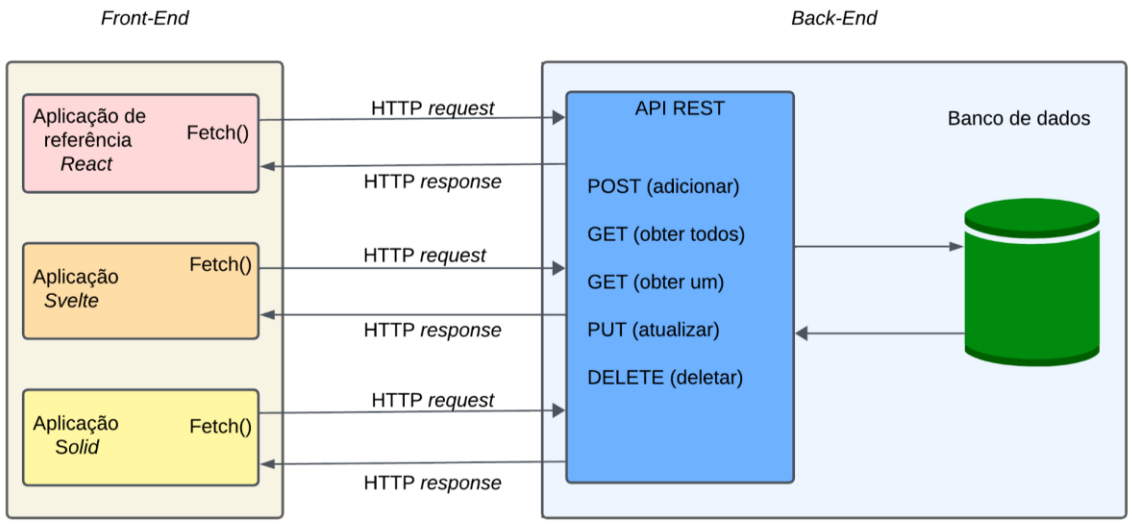
CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

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.

Figura 2 – Representação do funcionamento das aplicações



Fonte: Autoria própria.

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). 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.

2.3 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:

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

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

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 testes 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 serão realizados dez testes e calculado a média. Para medir o consumo de memória vai ser utilizada a ferramenta da Google, o Chrome DevTool 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 serão realizados dez testes e calculado a média. Para medir o consumo de CPU vai ser utilizada a ferramenta da Google, o Chrome DevTools na aba Desempenho.

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 testes e calculado a média.

Tamanho final do pacote: o tamanho do pacote das aplicações influenciam em sua performance e na experiência do usuário. Por isso, será avaliado o tamanho final das três aplicações.

Segundo Pimentel-Gomes (2023), o Coeficiente de Variação (CV) é classificado como baixo (<10%), médio (10–20%), alto (20–30%) ou muito alto (>30%).

3 RESULTADOS E DISCUSSÃO

Nesta seção, são apresentados e discutidos os resultados da análise comparativa entre React, Solid e Svelte, considerando os critérios de avaliação definidos na Seção 2.3. A análise busca identificar os pontos fortes e as limitações de cada tecnologia em aspectos como desempenho e uso de recursos.

CC BY

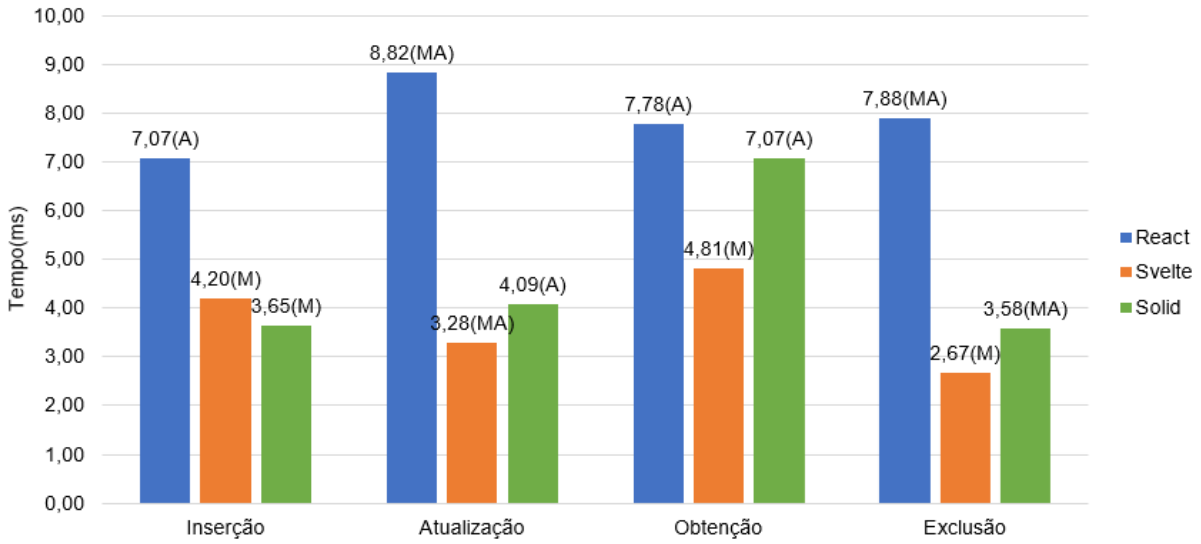
REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

3.1 Tempo de execução dos métodos CRUD

O tempo de execução dos métodos CRUD foi avaliado considerando as operações de inserção, atualização, exclusão e consulta de livros, conforme apresentado na Figura 3. Sendo o CV: B=baixo; M=médio; A=alto; MA=muito alto.

Figura 3 – Tempo médio de execução dos métodos CRUD nas três ferramentas (*React, Svelte e Solid*)



Fonte: Autoria própria.

Nos testes de execução dos métodos CRUD, o React apresentou os maiores tempos médios em todas as operações. O Solid obteve o menor tempo na inserção, enquanto o Svelte apresentou os menores tempos nas operações de atualização, obtenção e exclusão. Assim, o Svelte apresentou o melhor desempenho geral nesse critério, seguido pelo Solid e pelo React.

Quanto ao CV, o Svelte apresentou maior variabilidade na atualização, enquanto o Solid teve maior variação na exclusão. O React apresentou CV alto ou muito alto em todas as operações, indicando menor estabilidade dos tempos de execução em comparação às demais tecnologias.

3.2 Consumo de memória

O consumo de memória foi avaliado para cada operação CRUD, considerando seu impacto no desempenho da aplicação e na experiência do usuário. A Figura 4 apresenta as médias obtidas nos métodos de inserção, atualização,

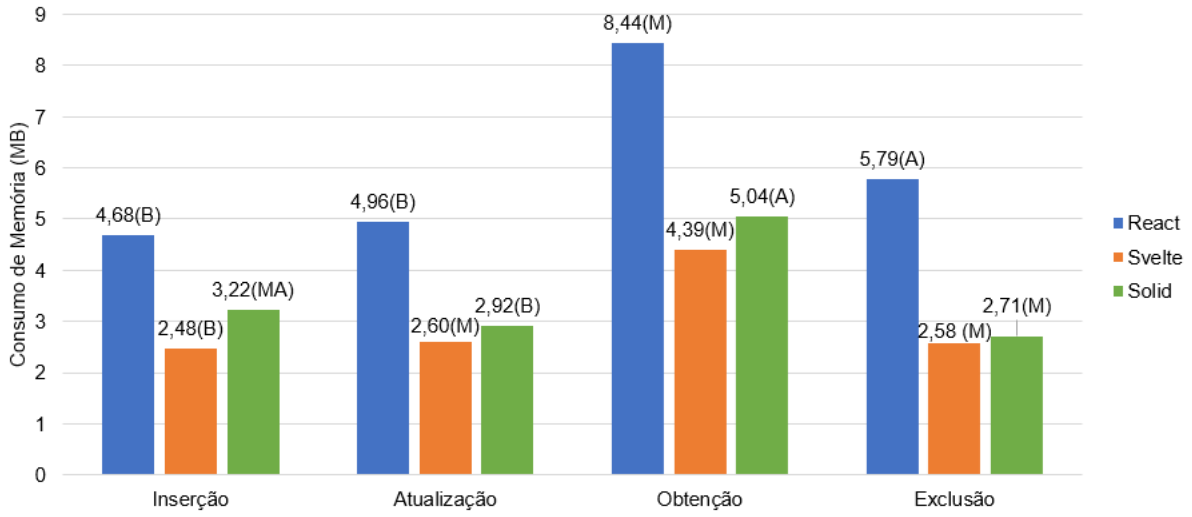
CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

exclusão e consulta de livros. Sendo o CV: B=baixo; M=médio; A=alto; MA=muito alto.

Figura 4 – Consumo de memória dos métodos CRUD nas três ferramentas (*React, Svelte e Solid*)



Fonte: Autoria própria.

Nos testes de consumo de memória, o React apresentou as maiores médias em todas as operações, enquanto o Svelte registrou os menores valores: 2,48 MB na inserção, 2,60 MB na atualização, 4,39 MB na obtenção e 2,58 MB na exclusão. O Solid apresentou consumo intermediário em todos os métodos.

Assim, o Svelte destacou-se pelo menor consumo médio e maior estabilidade na maioria das operações. O Solid apresentou maior variabilidade em alguns cenários, especialmente na inserção e obtenção. Já o React, apesar do maior consumo médio, apresentou baixa ou média variação em grande parte dos testes, com maior variabilidade apenas na operação de obtenção.

3.3 Consumo de CPU

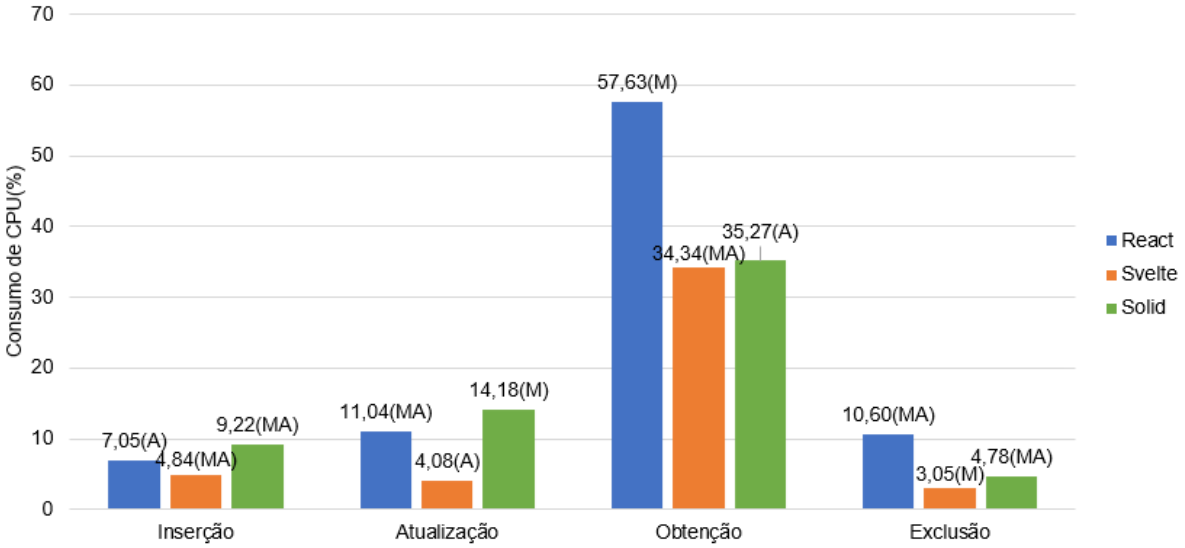
O consumo de CPU também foi avaliado nas operações CRUD devido à sua influência no desempenho da aplicação e na experiência do usuário. A Figura 5 apresenta as médias de consumo nas operações de inserção, atualização, exclusão e consulta de livros. Sendo o CV: B=baixo; M=médio; A=alto; MA=muito alto.

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

Figura 5 – Consumo de CPU dos métodos CRUD nas três ferramentas (*React, Svelte e Solid*)



Fonte: Autoria própria.

Nos testes de consumo de CPU, o Svelte apresentou as menores médias em todas as operações CRUD, destacando-se na obtenção de livros, com 34,34%, seguido pelo Solid, com 35,27%, enquanto o React registrou 57,63%. O Solid apresentou os maiores consumos nas operações de inserção e atualização, enquanto o React obteve os maiores valores nas operações de obtenção e exclusão. Quanto ao CV, o Svelte apresentou maior variabilidade nas operações de inserção, atualização e obtenção. O Solid apresentou CV alto ou muito alto na maioria dos métodos, indicando maior oscilação no consumo. O React também apresentou elevada variabilidade, principalmente nas operações de atualização e exclusão. Dessa forma, o Svelte destacou-se pelo menor consumo médio de CPU, embora tenha apresentado variações consideráveis entre os testes.

3.4 Tempo de renderização inicial

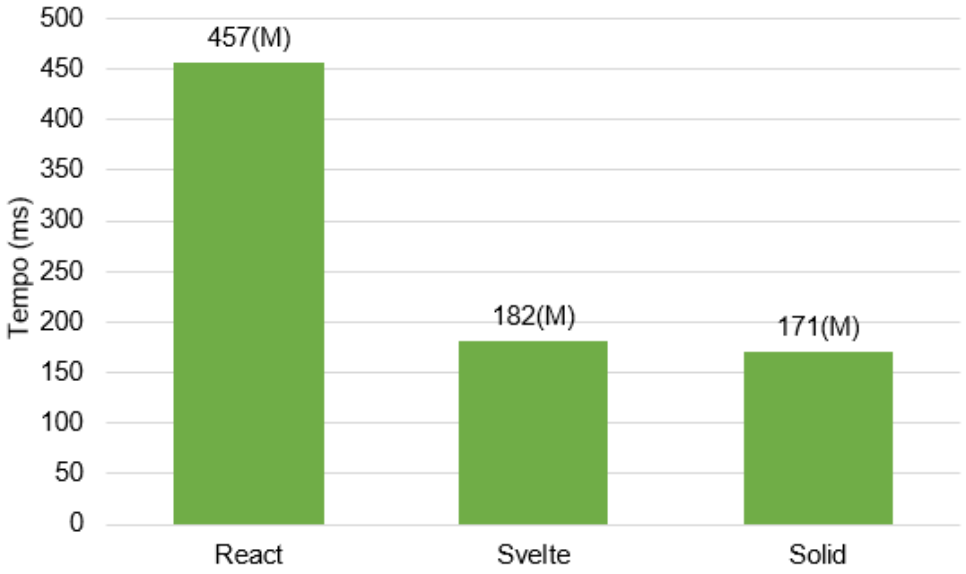
O tempo de renderização inicial foi avaliado devido à sua influência na experiência do usuário. A Figura 6 apresenta os resultados obtidos nos testes realizados com as aplicações desenvolvidas utilizando as três tecnologias. Sendo o CV: B=baixo; M=médio; A=alto; MA=muito alto.

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

Figura 6 – Tempo de renderização inicial das três ferramentas (*React, Svelte e Solid*)



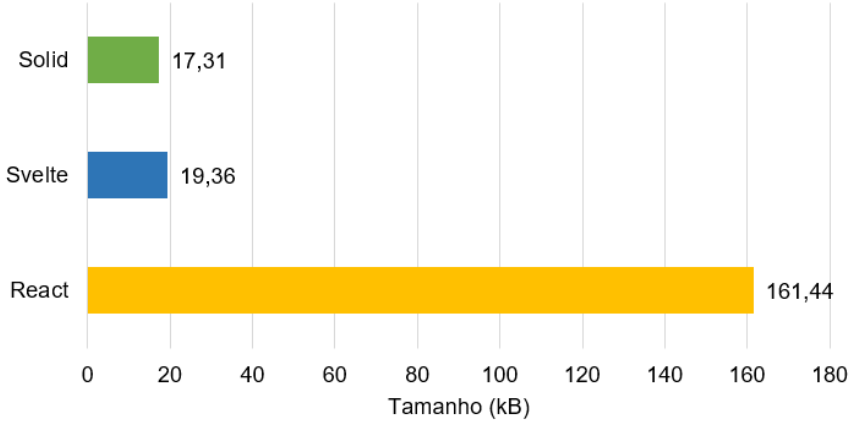
Fonte: Autoria própria.

Na renderização inicial, o Solid apresentou o menor tempo médio, com 171 ms, seguido pelo Svelte, com 182 ms, enquanto o React registrou 457 ms. Assim, o Solid obteve o melhor desempenho neste critério. As três tecnologias apresentaram CV médio, indicando variação moderada nos tempos de renderização.

3.5 Tamanho do build

O tamanho do build foi analisado por sua influência no desempenho das aplicações. A Figura 7 apresenta os tamanhos dos builds gerados pelas aplicações desenvolvidas com React, Solid e Svelte.

Figura 7 – Tamanho do *build* das três ferramentas (*React, Svelte e Solid*)



Fonte: Autoria própria.

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

O React apresentou o maior tamanho de build, com 161,44 KB, enquanto Solid e Svelte registraram valores significativamente menores, de 17,31 KB e 19,36 KB, respectivamente. Dessa forma, Solid e Svelte apresentaram builds mais leves em comparação ao React, com destaque para o Solid, que obteve o menor tamanho.

4 CONSIDERAÇÕES FINAIS

Os resultados indicam diferenças entre React, Solid e Svelte nos critérios avaliados. O Solid apresentou o melhor desempenho na renderização inicial e o menor tamanho de build. O Svelte destacou-se pelo menor consumo médio de memória e CPU, além dos menores tempos de execução na maioria dos métodos CRUD. O React, por outro lado, apresentou os maiores tempos de execução, consumo de memória e tamanho de build. Assim, os resultados sugerem que Solid e Svelte apresentam vantagens em desempenho e uso de recursos, enquanto a escolha da tecnologia deve considerar as necessidades específicas de cada projeto.

Referências

CAMARGOS, J. G. C. de et al. Uma análise comparativa entre os frameworks javascript
angular e react. Computação & Sociedade, v. 1, n. 1, 2019. Disponível em: http: //revista.fumec.br/index.php/computacaoesociedade/article/view/7307. Acesso em: 14 ago. 2023.

GITHUB. Let's build from here. 2023. Disponível em: https://github.com/. Acesso em: 06 out. 2023.

GIZAS, A.; CHRISTODOULOU, S.; PAPATHEODOROU, T. Comparative eva luation of javascript frameworks. In: Proceedings of the 21st International Conference on World Wide Web. [s.n.], 2012. p. 513–514. Disponível em: https://dl.acm.org/doi/abs/10.1145/2187980.2188103?casa_token=nSM2ei9yCGIAAAAA:yALvBgUWIbyB9yuDw3036rIeKtChDWv8hnn6Prg8PMLfWBU0qyt-xPEsGwReNQqdG7dEy8-6I6xjJw. Acesso em: 18 ago. 2023.

LEVLIN, M. DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte. 2020. Dissertação (Mestrado), 2020. Disponível

CC BY

REMUNOM
ISSN 2178-6925

Received: 13/10/2026 - **Accepted:** 13/11/2026
Vol: 13.24
DOI: 10.66104/qyb3ec73
Pages: 1-15

em:
https://www.doria.fi/handle/10024/177433. Acesso em: 24 ago. 2023.
MARIANO, C. L. Benchmarking javascript frameworks. 2017. Dissertação (Mestrado), 2017. Disponível em: https://arrow.tudublin.ie/scschcomdis/94/. Acesso em: 24 ago. 2023.
RAMBEAU, M. 2020 JavaScript Rising Stars. 2020. Disponível em: https://risingstars.js.org/2020/en#section-framework. Acesso em: 20 out. 2023.
RAMBEAU, M. 2021 JavaScript Rising Stars. 2021. Disponível em: https://risingstars.js.org/2021/en#section-framework. Acesso em: 20 out. 2023.
RAMBEAU, M. 2022 JavaScript Rising Stars. 2022. Disponível em: https://risingstars.js.org/2022/en#section-framework. Acesso em: 20 out. 2023.
STACKOVERFLOW. Programming, scripting, and markup languages. 2023a. Disponível em: https://survey.stackoverflow.co/2023/#section-most-popular-technologies-programming-scripting-and-markup-languages. Acesso em: 18 ago. 2023.
STACKOVERFLOW. Webframeworksandtechnologies.2023b. Disponível em: https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies.Acesso em: 18 ago. 2023.
STATEOFJS. Front-end Frameworks. 2022. Disponível em: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/. Acesso em: 14 ago. 2023.

CC BY