Abdul Kevin Alexis

Comparative analysis between Single frameworks Page Application (SPA)

Belo Horizonte

2023

Abdul Kevin Alexis

Comparative analysis between Single Page frameworks Application (SPA)

Course Completion Work presented to the Computer Engineering Course at the Federal Center for Technological Education of Minas Gerais, as a partial requirement for obtaining the Bachelor's degree in Computer Engineering.

Federal Center for Technological Education of Minas Gerais – CEFET-MG

Department of Computing

Computer Engineering Course

Advisor: Prof. Dr. Edson Marchetti da Silva

Belo Horizonte

2023

Federal Center for Technological Education of Minas Gerais

Computer Engineering Course

Evaluation of the Course Completion Work

Student: Abdul Kevin Alexis

Job title: Comparative analysis between Single Page Application (SPA) frameworks

Defense date: 12/06/2023

Time: 08:00

Defense location:

https://conferenciaweb.rnp.br/conference/rooms/edson-marchetti-da-silva-2/invite?showsuggestion=fa lse

This Course Completion Work was evaluated by the following panel:

Professor Edson Marchetti da Silva – Advisor
Department of Computing
Federal Center for Technological Education of Minas Gerais

Professor Kecia Aline Marques Ferreira – Member of the evaluation panel
Department of Computing
Federal Center for Technological Education of Minas Gerais

Professor Eduardo Cunha Campos – Member of the evaluation panel
Department of Computing
Federal Center for Technological Education of Minas Gerais

I dedicate this work to everyone in the area of Computer Engineering or Software.

Thanks

Firstly, I would like to thank my supervisor and the people who contributed to my training.

Next, I thank the Brazilian People, this country that welcomed and supported me throughout my stay here. I am very grateful for the opportunity and growth during my study.

I would also like to thank those who allowed me to travel to Brazil. Without your precious help, I would not have been able to make my dream come true.

Special thanks to:

- Esther Fleurijuste, my dear, for your unconditional support;
- Samuel Michel, for his help in obtaining the scholarship;
- Marie Alice Sirène, my godmother, for always believing in me;
- Nicolas Dorvilus, who rested in peace, for his unconditional support and support;
- Beat Lockslina Edmond, for her generosity and solidarity;
- Guerlovecia Pierre, for her help with the correction;
- Father Benh Donais (CJMS), for his encouragement and guidance;
- Socolavim (Fils-Aimé) and DGS Communication (Gary Dalencourt), for financial support ceiro;
- Carline Alexis, Dor Erlens, Rose Darline Camille, Maxime Termitus, @BUC (Laf, Mahatma, AMI, 2tay, Sly. . .), I am so grateful to all of you for being part of this journey. Your support was essential for me to achieve my goals.

Thanks!

“Belief + Perseverance = Achievement or Success”

(Own authorship)

Summary

This work presents the results of an experimental study carried out to compare three *front-end* development *frameworks :* Angular, React and Vue. Single Page Application (SPA) frameworks are one of the main trends in current web *development .* They allow you to create web applications that load just one page, which provides a more fluid and responsive user experience. With the proliferation of SPA *frameworks* available, choosing the ideal one for a given project can be a challenging task. The objective of this work is to carry out a comparative analysis between the Angular, React and Vue *frameworks ,* considering package size, loading time, rendering time, memory usage and execution time. As a method, an application based on microservice architecture was created using each of the alternatives, then these applications were compared to evaluate which one presents better performance. The comparative analysis revealed that React has the best overall performance, followed by Vue and Angular. In terms of bundle size generated to put into production, Vue has the smallest *bundle,* React has the second smallest and Angular has the largest of the three. In terms of loading time, React and Vue are the fastest, followed by Angular. In terms of rendering time, Angular and React, which have similar performance, render faster, followed by Vue, which is slower in rendering. In terms of memory usage, Vue uses less memory, React comes second followed by Angular which uses the most memory. In terms of execution time, React has the best performance in all CRUD operations, followed by Vue and Angular. Based on the results obtained, React or Vue was recommended as an option for applications that need to load quickly. For applications that need to use less memory, Vue is the best option.
For applications that need to perform well in CRUD operations, React is the best option. The results of this study contrast with those of Kaluža, Troskot and Vukeliÿ (2018), who concluded that there is no ideal JavaScript framework for MPA and SPA. Ziani (2021) also carried out a comparative analysis of JavaScript frameworks, but his study was qualitative, while the present study is quantitative. Therefore, the results of this study corroborate the statement by Diniz-Junior et al. (2022), that Angular has better performance in terms of rendering. The results of this study are relevant for beginning software engineers as they provide valuable information for choosing the best SPA technology for architecture-based software microservice.

Keywords: *frameworks.* SPA. Web Development.Angular. React. Vue

Abstract

This work presents the results of an experimental study carried out to compare three front-end development frameworks: Angular, React and Vue. Single Page Application (SPA) frameworks are one of the main trends in current web development. They allow you to create web applications that load just one page, which provides a more fluid and responsive user experience. With the proliferation of SPA frameworks available, choosing the ideal one for a given project can be a challenging task. The objective of this work is to carry out a comparative analysis between the Angular, React and Vue frameworks, considering package size, loading time, rendering time, memory usage and execution As a method, an application based on microservice architecture was created using each of the alternatives, then these applications were compared to evaluate which one presents better performance. The comparative analysis revealed that React has the best overall performance, followed by Vue and Angular. In terms of bundle size generated to put into production, Vue has the smallest tbundle, React has the second smallest and Angular has the largest of the three. In terms of loading time, React and Vue are the fastest, followed by Angular. In terms of rendering time, Angular and React, which have similar performance, render faster, followed by Vue, which is slower in rendering. In terms of memory usage, Vue uses less memory, React comes second followed by Angular which uses the most memory. In terms of execution time, React has the best performance in all CRUD operations, followed by Vue and Angular. Based on the results obtained, React or Vue was recommended as an option for applications that need to load quickly. For applications that need to use less memory, Vue is the best option. For applications that need to perform well in CRUD operations, React is the best option. The results of this study contrast with those of Kaluža, Troskot and Vukeliÿ (2018), who concluded that there is no ideal JavaScript framework for MPA and SPA. Ziani (2021) also carried out a comparative analysis of JavaScript frameworks, but his study was qualitative, while the present study is quantitative. Therefore, the results of this study corroborate the statement by Diniz-Junior et al. (2022), that Angular presents better performance in terms of rendering. The results of this study are relevant for beginning software engineers, as they provide valuable information for choosing the best SPA t on microservice architecture.

Keywords: frameworks. SPA. Web Development.Angular. React. Vue

List of illustrations

List of tables

List of abbreviations and acronyms

AHP	Analytic Hierarchy Process
API	Application Programming Interface
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
SUN	Document Object Model
HTML	HyperText Markup Language
IDE	Integrated Development Environment
MPA	Multi-page application
REST	REpresentational State Transfer
IF THE	Search Engine Optimization
DBMS	Database management system
SPA	Single Page Application

summary

1. Introduction

Currently, the world is witnessing a rapid process of evolution in the need for information facilities in terms of quantity, quality and access. People are more connected around the world and can work with communities outside their usual networks (NEWMAN et al., 2016).

Given these facts, developers need to choose, among the different existing architectures and technologies, which best suits your demands in each case. According to Kornienko, Mishina, and Melnikov (2021), web solutions are preferred over mobile or *desktop* platforms when creating new services. This choice is justified by the fact that any online resources do not impose strict hardware requirements on the user (storage memory, random access memory, power consumption of the central processor unit), do not tie the interaction to a specific type of platform (personal computer, *smartphone,* specialized device). In other words, web applications can be accessed using just a web browser.

According to its *design,* a web application can be developed as a *Multi Page Application* (MPA) or *Single Page Application* (SPA) (KALUŽA; TROSKOT; VUKELIÿ, 2018). MPA is the conventional model for building *front-end software,* in which the browser renders an entire page for each route or tab in the application to present data. Generally, data is rendered on the server side and only one HTML document is sent to the client side with each request. However, SPA is a combination of solutions, dividing a web application into two parts *(front-end and back-end)* and organizing the interaction between them. The *backend* part handles the logic, processes requests, works with the database, while the *frontend* part forms the external view based on the data received from the *backend* and displays the result in the browser.
The exchange of information between the client and server parts of the application usually occurs through a specially developed application programming interface. Modern web applications , which are not developed in a traditional way, are based on the SPA concept.

According to W3Techs (2022), JavaScript is used as a client-side programming language by 97.9% of all websites. Because there are many JavaScript *frameworks* available, choosing one to carry out a project can be a very challenging task. In this context, according to StackOverflow (2021), the most used development alternatives are React.js, JQuery, Express, Angular, Vue.js. Therefore, what criteria should be adopted to choose a *framework* among the most used?

According to the conclusions presented in the work of Ziani (2021), there is no

reliable and unanimously accepted categorical answer on the best JavaScript *framework* to use when developing a *front-end web application.* Similarly, Ferreira and Zuchi (2018) concluded that each *framework* can bring a certain advantage to the project, and it is up to the developer to define which characteristics are most important for its application.

There are several different SPA alternatives available, each with its own strengths and weaknesses, meaning each *framework* has its own unique set of features and capabilities. Angular, React, and Vue are three among the most popular JavaScript *frameworks* for single-page application development. They are all component-based, which makes code modularization and maintenance easier. Furthermore, they all offer a wide range of features and libraries, which makes them suitable for a variety of applications.

The choice of these three frameworks for a comparative analysis is justified by the following reasons:

- relevance: These *frameworks* are the most used on the market, which ensures that the analysis results are relevant to a wide range of professionals;
- diversity: The *frameworks* present different development approaches, which which allows for a more comprehensive analysis;
- potential contribution: Comparative analysis can provide valuable *insights* for software engineers who are evaluating which *framework* to use for their projects.

Therefore, making the best choice for a given project will depend on the specific requirements.

1.1 Objective

The purpose of this work is to perform a comparative analysis between the SPA Angular platform, the React library and the Vue *framework.*[1] The comparative analysis of tools for SPA development can be carried out using a variety of methods. A common method is to create a table that compares the different alternatives tested based on various criteria, such as: ease of use, performance and features. Another method is to create a demo application for each of the alternatives tested, then compare the applications to see which performs best. For

[1] In the context of this work, it is worth highlighting the complexity inherent in the search for a consensual term that encompasses the three SPA technologies: the Angular platform, the React library and the Vue framework. In order to simplify communication and avoid ambiguities, we will adopt the term *"framework" in a* generic way to reference these technologies throughout the document.

This was why an application was implemented in microservices architecture using each of these alternatives in developing the *front-end.* From these developed applications, tests were carried out to measure the size of the generated code, the loading time, in addition to measuring for different load scenarios, the rendering time, memory usage and execution time in each case.

1.2 Justification

Carrying out this work is justified because a comparative analysis of *SPA frameworks can* be a valuable tool for developers who are considering using a SPA tool for a new project. By comparing different alternatives, developers can identify which one best meets the needs of their project.

Below are some of the benefits in which this comparative analysis sought to do:

- help developers identify *front-end* development alternative SPA that best meets the needs of your project;
- help developers save time and money, avoiding the need to try different alternatives;
- help developers choose a *framework* supported by a large community of users;
- help recent graduates or beginners to choose the best option, among the various existing alternatives, to study and develop.

Overall, a comparative analysis of SPA *frameworks* is a valuable tool for developers who are considering using a SPA *framework* for a new project. By carrying out this analysis, developers can be sure that they are making the best choice for their needs and that they are well informed about the different tools and how they work.

1.3 Work structure

The remainder of this work is organized as follows. Chapter 2 presents the theoretical framework. Chapter 3 deals with related work. Chapter 4 discusses the proposed methodology. Chapter 5 presents the results, while chapter 6 presents the conclusions.

2 Theoretical Framework

This chapter presents the theoretical framework that underpins this research, providing the conceptual and theoretical bases necessary for understanding the topic under study. The main concepts, theories and studies that support the analysis will be covered.

2.1 The beginning of the Web

It is very common to hear the names Google, Facebook, Twitter, Instagram. . . , etc. before hearing the expression “social network”. After all, these technology giants boosted the search for information and interactions between people. Due to the need for continuous communication in order to stay connected to everything that happens, it is difficult to live without these technologies in today's world. They are indispensable tools for carrying out personal and work interactions, being part of our daily lives. And this all started with a technology called the Web in the 1980s.

According to Berners-Lee et al. (1994), the web represents a series of things, which must be distinguished. These things include:

– the idea of a world of limitless information in which all items have a reference by which they can be retrieved;

– the address system (URI) that the project implemented to make this world possible despite many different protocols;

– a network protocol (HTTP) used by native web servers offering pledge and resources not otherwise available;

– a markup language (HTML) that every W3 client must understand, and is used for transmitting basic things like text, menus, and simple online help information over the network;

– the dataset available on the Internet using all or some of the items listed above.

Berners-Lee is considered the father of the Web. The main foundations he invented gave rise to the web as we know it today. However, they were not enough to make the web sufficiently interactive on the *browser side.* According to the Mozilla website (2023), whenever a web page presents features beyond displaying static information, such as displaying periodically updated content, maps

interactive or animated graphics in two or three dimensions, it is highly likely that JavaScript is involved in this process. Therefore, Section 2.2 provides a brief introduction to this topic.

2.2 JavaScript

JavaScript, sometimes abbreviated to JS, is a lightweight, interpreted language based on objects with *first-class function,* [1] best known as the *scripting* language. for Web pages. It is also used in several other non- *browser environments,* such as: node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm and dynamic language, supporting object-oriented, imperative and declarative styles such as, for example, functional programming (JAVASCRIPT, 2021).

JavaScript is a high-level, dynamic, interpreted programming language widely used on the web to create interactivity and dynamic effects on web pages. It was developed by Brendan Eich in 1995 while working at Netscape Communications Corporation. It is based on C and Java, and has a similar syntax, but different from both in terms of objectives and purpose. It is also widely supported by all major web browsers and can be used on both the client and server sides. With the popularization of libraries and *frameworks* such as jQuery, AngularJS, React and Vue, JavaScript has become even more popular, which is a feature valued by software developers. One of the main advantages of JavaScript is the ability to create both *Multi Page Application* (MPA) and *Single Page Application* (SPA) applications. These two concepts will be discussed below.

According to Kaluža, Troskot and Vukeliÿ (2018), SPAs are a type of web application that loads all its content on a single HTML page. This means users don't have to wait for the page to reload when interacting with it, which can provide a more seamless and responsive user experience. SPAs are becoming increasingly popular as they can be used to create web applications that are more interactive and engaging than traditional web pages. On the other hand, MPAs are the traditional web pages, each content is loaded on a different page, which is reloaded with each user interaction.

A SPA stands out, in relation to an MPA, due to its better performance, lower

[1] A programming language is considered to have *first-class* function characteristics when its functions are treated equivalently to other variables within the system. In other words, in this type of language, it is possible to use functions as arguments to other functions, return them as the result of a function and assign them to variables, in a similar way to the treatment given to other data entities.

resource consumption, greater interactivity and better user experience. However, it has greater development complexity, difficulty in *Search Engine Optimization* (SEO) and security. On the other hand, an MPA has simplicity of development, ease of SEO and security. However, it has worse performance, greater resource consumption, less interactivity and worse user experience (SHARMA et al., 2019).

The choice between SPA and MPA depends on the requirements and objectives of the web application. If the application requires high performance, interactivity and usability, a SPA may be more suitable. If the application requires simplicity, SEO and security, an MPA may be a better choice.

2.3 Angular

Angular is an open source platform for developing web applications maintained by Google. It is based on TypeScript and was first released in 2010. Angular is known for its component architecture, which allows developers to create web applications in a modular and scalable way. It provides a series of tools for creating web applications, including state management, form validation, routing, services, and more. Angular is used by companies of all sizes, including small *startups* and large corporations, to create high-performance web applications. It is one of the most popular and widely used JavaScript tools around the world (ANGULAR, 2023).

2.4 React

React is an open-source JavaScript library developed by Facebook and maintained by a community of developers. It was launched in 2013 and is used to create user interfaces in web and mobile applications. React focuses on creating reusable components, allowing developers to build web applications in an efficient and scalable way. It uses the virtual DOM (VDOM) which is a programming concept where an ideal, or “virtual”, representation of the user interface is kept in memory and synchronized with the “real” DOM by a library such as ReactDOM.
This process is called reconciliation. Which allows React to quickly render and update the UI without having to refresh the entire page. Additionally, React provides features for state management, routing, form validation, and other important functionality for building web applications. Due to its efficiency, scalability and ease of use, React is widely used by companies of all sizes, including *startups* and large corporations, to develop high-quality web applications (REACT, 2023).

2.5 Vue

Vue.js is an open-source JavaScript *framework* for building user interfaces . It was created by Evan You in 2014 and is maintained by an active community of developers. Vue stands out for its componentization approach, which allows developers to create web applications in a modular and scalable way. Additionally , Vue has a reactive data structure, which allows developers to create dynamic interfaces that automatically update as data changes. Vue also includes features for routing, form validation, animations, and more, making it a popular choice for building web applications. Vue is known for its smooth learning curve and clear documentation, making it a popular option for both beginners and experienced developers. It is widely used by companies of all sizes to create high-quality web applications (VUE, 2023).

2.6 Microservices Architecture

Microservices architecture is a software architectural style that focuses on building applications as a set of independent, collaborative services. Each service is built around a single business functionality and is deployed and managed independently of the other services. These services generally communicate with each other through a well-defined Application Programming Interface (API) and can be developed in different programming languages and using different technologies (MICROSOFT, 2023).

According to Lewis and Fowler (2014), microservices architecture is based on *design* principles such as separation of responsibilities, high cohesion and loose coupling. These principles ensure that each service is highly specialized in a single task, which facilitates the maintenance and evolution of services. Microservices architecture also facilitates horizontal scalability of services, allowing each service to scale independently as needed.

Along the same lines, according to Newman (2021), microservices architecture is an approach to building software systems that emphasizes modularity, scalability, maintenance and independence of services. Each microservice is responsible for a single function and is built around a specific business context. These services are highly independent and can be deployed and managed separately, which makes application development and maintenance more agil

In summary, the concept of microservice architecture was used in the work to provide a context for the comparative analysis of the Angular, React and Vue *frameworks* . Microservice architecture is a software design approach that divides a

system into smaller units, called microservices. Each microservice is responsible for a specific system function. This facilitates the maintenance, evolution and scalability of services.

2.7 REST Architecture

The REpresentational State Transfer (REST) architecture, according to Fielding (2000), is a style of software architecture that is based on Web communication principles. It is widely used to build APIs for web and mobile applications. The REST architecture defines a series of rules and recommendations for structuring APIs, including the use of HTTP resources such as GET, POST, PUT and DELETE to represent actions on application resources. Additionally, the REST architecture uses URLs to identify resources and uses standardized HTTP responses to indicate the result of a request. The REST architecture is scalable and allows the creation of distributed applications, as different applications can access and share resources through a REST API. This makes REST architecture a popular choice for creating applications that need to integrate with other systems or share data between different platforms. Therefore, the REST architecture was used in the work to define how microservices communicate with each other. REST architecture is a software architecture pattern that defines a set of constraints for communication between systems. These constraints ensure that microservices are interoperable and easy to integrate.

3 Related Works

This chapter presents studies related to comparative analysis between *frameworks* SPA.

Kaluža, Troskot and Vukeliÿ (2018) carried out a comparative analysis between *front-end fra-meworks* for web development. The objective of the article was to demonstrate the suitability of available *frameworks* for creating SPA *vs* MPA web applications, define the main functionalities that allow for more customized development of both types of applications, in addition to identifying whether there is a *front-end framework* optimized for create MPA and SPA applications. The authors chose the three most popular *front-end frameworks* in order to analyze: whether just *js import* is enough to get started; whether the structure includes a lot of overhead; whether the workflow is necessary; whether there is side rendering; if there is a need to write and maintain a large amount of complex code; whether the developer should rely on third-party packages; what is the possibility of compressing the application, minimizing the *bundle1* file ; in addition to whether it is possible to manage the state of the data. To meet the objective, they chose the most popular *frameworks* on Github that were still growing at that time, discarding the others that were stagnant. According to these authors, React had 86,000 stars and Vue.js 81,000. On the other hand, Angular was being pushed by giants Google and Microsoft. Eight questions were formulated and answered based on the objective. The answers were used as metrics, expressed in text, the possible values of the answers were: No, Partially and Yes. In addition, each answer was assigned a weight on a scale from 0 to 2. These values serve to reinforce how strong each of the textual alternatives used is given. They came to the conclusion that Angular is the best *framework* for SPA, but on the other hand, it proved to be the most unsuitable for MPA development. Vue is best for MPA. React was in an intermediate situation in both cases.

In the work of Ziani (2021), a comparative and qualitative analysis was carried out JavaScript *frameworks .* The work sought to answer three questions:

– There is an effective method to compare and select a JavaScript *frameworks* like part of developing a web application?

– What are the criteria used to compare JavaScript *framework ?*

[1] The *bundle* is the web-optimized production build created from each framework's build tools. Its toolsets result in a web-compressed HTML file and minified JavaScript file, ready for production deployment. For native JavaScript, a static HTML file is used that encapsulates the equivalent *script ,* containing the functions written in the *framework.*

– Whether all JavaScript *frameworks* are taken into account in the analyses?

According to Ziani, the answers to the first two questions were obtained through literature searches. In other words, for the first question, it seems that there is no single, standardized method for comparing *frameworks* recognized by the scientific community . Regarding the second question, Ziani identified four groups of criteria:

- measurable criteria;
- desired/expected characteristics;
- the result of running external software;
- criteria that are static or measurable but unreliable because they change.

And finally, Ziani did not find a timeless solution, as new *frameworks* are created every year, existing *frameworks* are improved and sometimes refactored without offering backwards compatibility (e.g. Angular V1 *vs* Angular V2) and finally it can happen that *frameworks* are no longer supported. Therefore, Ziani proposed to use the *Analytic Hierarchy Process* (AHP) method using decision tree to compare different JavaScript *frameworks .* This method considers learning effort as a selection criterion ; the development effort; stability, attractiveness and performance as criteria. In conclusion, Ziani pointed out Svelte as the best *framework* among those that were tested: Angular, Preact, React, Svelte and Vue.

Diniz-Junior et al. (2022) evaluated the performance of JavaScript-based web rendering technologies: Angular, React, and Vue. The aim of the work was to evaluate the performance of JavaScript-based web rendering technologies, considering interaction time, DOM manipulation time and bundle size . The authors searched the literature for the main evaluation metrics and analyzed virtual and incremental DOM, creating a standardized application. They tested DOM manipulations using three frameworks that represent these two rendering techniques: Angular for incremental and React and Vue for virtual. As results, the authors sought to contribute:

– a methodology for comparing incremental and incremental web rendering *frameworks* JavaScript-based virtual;

– a comparative case study with the Angular, Vue and React *frameworks ;*

– a generic web application capable of creating, editing and deleting a certain number of elements in the browser window, following the life cycle (framework function that controls the states of an application) proposed by each *framework* and library.

Diniz-Junior et al. (2022) concluded that React had difficulty creating DOM elements. Angular and Vue took 270-290 ms on average respectively, while React took over 5000 ms. The first two are virtual DOM based and produced different results. Angular, in turn, with its incremental DOM implementation, produced results similar to those seen when creating elements for Vue structures, with an average difference of around 20 ms considering 50,000 elements and the best performance when editing DOM scenarios. As a result, the main advantage over VDOM or VNodes is that they can be updated without needing an intermediary to combine the elements in memory. Finally, the authors report that the solution proposed by the virtual DOM implemented within the Vue *framework* had a small overall advantage, performing well in all functions and maintaining peaks below the mark of 1s at the largest sample size.

The work of Ziani (2021) is different from this work due to the method used to assign weight to the criteria, which is subjective. In other words, the weights assigned to each variable proposed by Ziani (2021) follow a logic defined by the author that could be different if proposed by someone else. The work of Kaluža, Troskot and Vukeliÿ (2018) differs from this work by taking into account the concept of MPA. The work of Diniz-Junior et al. (2022) focused a lot on the issue of rendering. However, all these works are similar because they take into account the three *frameworks* Angular, React and Vue. Therefore, this work can contribute to scientific knowledge about JavaScript *frameworks* for developing SPAs based on microservices. For example, the analysis can identify patterns and trends that may be useful for future studies. Furthermore, microservice architecture is an increasingly popular software design approach, and Angular, React, and Vue *frameworks* are the most widely used for single-page application development. A comparative analysis of these *frameworks* in the context of microservice architecture is relevant for a wide range of professionals. Next , in chapter 4, the methodology that was adopted in this work is presented to compare the SPA *frameworks .*

4 Methodology

This chapter presents the proposed methodology to achieve the objectives outlined in this work. The methodology for comparing SPA *frameworks* by developing an application with microservices architecture to test performance consisted of a series of steps described below:

4.1 Bibliographic review

A literature review was carried out to obtain information about the *frameworks,* their main characteristics, advantages and disadvantages, and how to compare them in relation to criteria such as package size, loading time, rendering time, memory usage and execution time. . Three recent studies analyzed SPA *frameworks* for web development. Kaluža, Troskot and Vukeliÿ (2018) compared *frameworks* for SPA and MPA, identifying the main functionalities and suitability for each type of application. Ziani (2021) sought an effective method to compare and select JavaScript *frameworks ,* identifying the most used criteria and *frameworks .* Diniz-Junior et al. (2022) evaluated the performance of JavaScript *frameworks ,* considering interaction time, DOM manipulation and *bundle* size . Studies indicate that there is no ideal *front-end* framework for all cases. The choice of framework must be made considering the type of application, user needs and relevant evaluation criteria. The literature review also provided a general understanding of the concept of microservices architecture and how it is implemented in applications.

4.2 Defining the application architecture and development

Based on the literature review, the evaluation criteria, metrics (bundle size , *loading* time, rendering time, memory usage and execution time) and the *frameworks* to be comparatively evaluated were defined.
In this way, the requirements to be implemented by the application to be built using the three proposed alternatives were defined. The microservices architecture has been defined to ensure that the application is scalable and can handle excess traffic.
This is necessary to define the scope of work. The objective was to develop the same application with the three *frameworks* and compare them using the defined metrics;

4.2.1 Infrastructure

To meet the proposed methodological requirements, it was essential to employ a computing device with adequate storage and processing capacity for the development and, if necessary, simulation of the application. The following technologies were used to carry out these tests:

- Vaio FE15 notebook with 12GB of memory, Intel® Core™ i5-8250U processor 3.40 GHz, running Windows 10 operating system;
- Microsoft Edge 119.0.2151.58 to run the *front-end;*
- Docker1 for containerization [two] services;
- the IDE VS Code for implementing the codes;
- the MYSQL 5.7 DBMS for data storage;
- PM23 5.3.0 for process management;
- Node.js 16.16.0;
- Angular 16.2.0, latest version of long-term support at the time of performing the experiment;
- React 18.2.0, last version of long-term support at the time of performing the experiment;
- Vue 3.3.4, latest long-term support version at the time of performing the experiment.

4.2.2 Application description

The application used for testing can be considered a part of an *e-commerce system.* The developed application focuses on managing the products available for sale in the online store, including their creation, reading, updating and removal (CRUD), as well as storing the images associated with these products.

In this way, an *e-commerce* can benefit from the described system to manage products efficiently, making them easily accessible to customers through a user interface developed with Vue, React and Angular. The system can also

---

1 Docker is a set of platform-as-a-service products that use operating system-level virtualization to deliver software in packages called containers. Containers are isolated from each other and bundle their own software, libraries, and configuration files.

[two] Containerization is the packaging of software code with all necessary components, such as libraries, frameworks and other dependencies, so that they are isolated in their own container.

3 PM2 is a process manager for the Node.js JavaScript *runtime*

help ensure the integrity and consistency of product data, avoiding problems such as incorrect information or lost images.

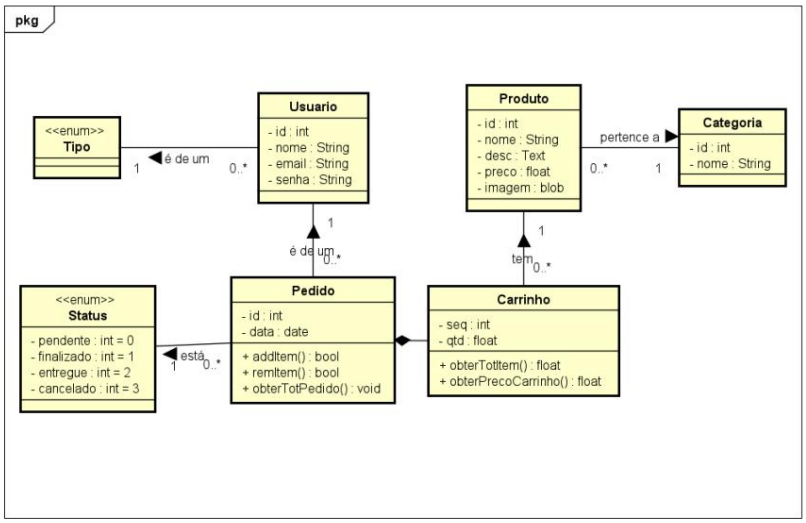
Therefore, the application is made up of two layers:

- *back-end :* an API service that communicates with the MySQL DBMS;
- *front-end:* the application interface implemented with the three *frameworks* (Angular, React, Vue.js).

Figure 1 shows a class diagram that represents the application's classes. This class diagram corresponds to the system requirements which are:

- allow authorized users to create, read, update and delete products;
- make it possible to associate images with each product;
- allow users to search for products by name and category;
- allow users to add products to the cart and complete the purchase.

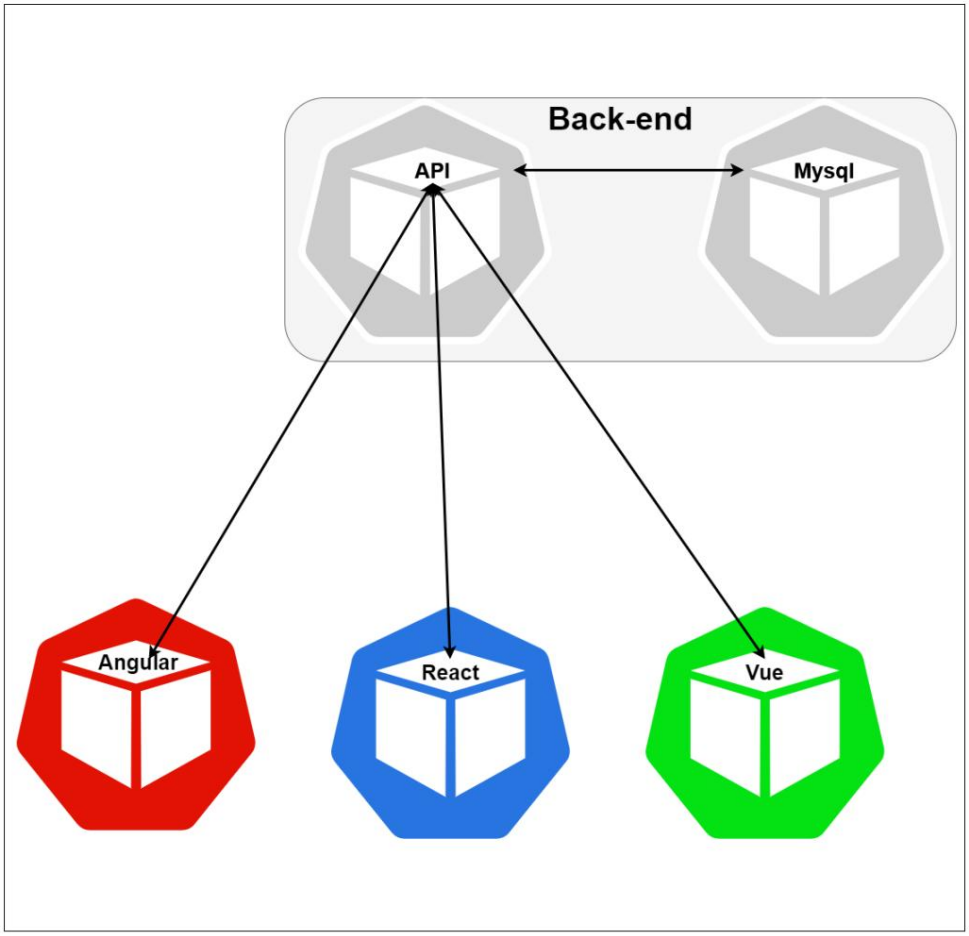
Figure 1 – Class diagram



Source: from the Author (2023).

To simulate a real microservice architecture environment, the concept of containerization was used to implement each service. Figure 2 shows a diagram that represents the application architecture. Each cube represents a container in which a service that is a component of the application has been implemented. In this case, the *front-end* was implemented using SPA Angular, React and Vue technologies to be able to perform the comparative analysis.

Figure 2 – Application architecture



Source: from the Author (2023).

4.3 Performance tests

After the application was developed, performance testing was carried out to compare them. Performance testing aims to measure package size, loading time, rendering time, memory usage, and execution time under different load scenarios. However, to make testing possible, without considering the complexity of interaction in an *e-commerce,* operations in the interface were limited to:

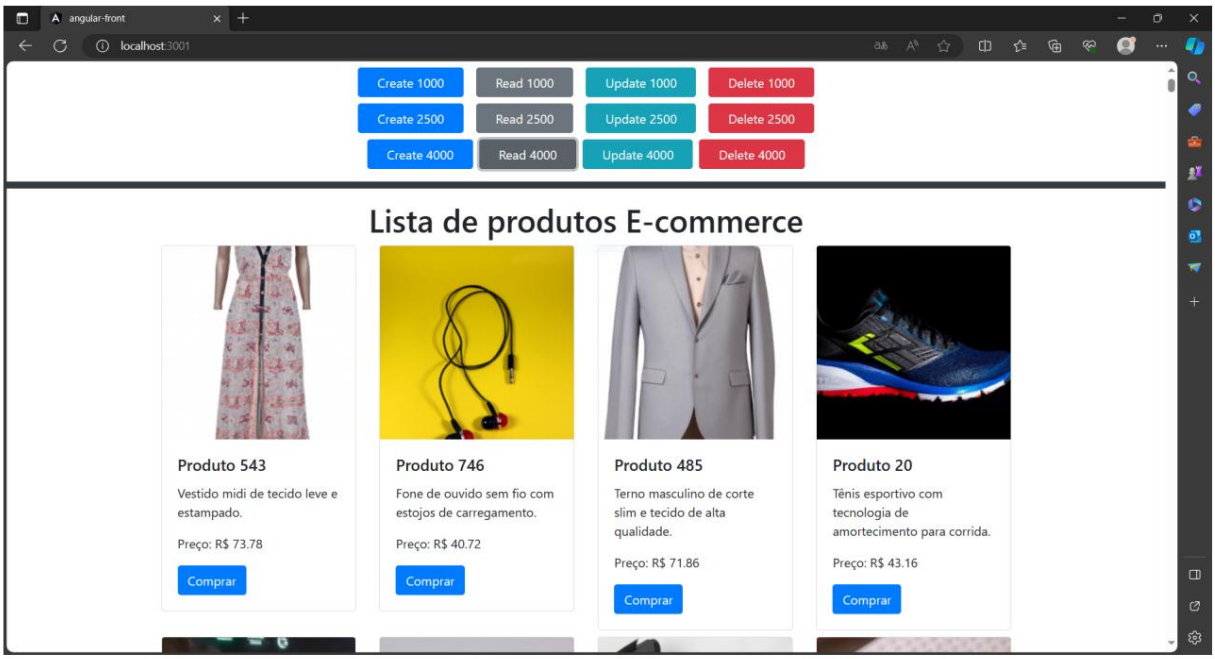
- creation of 1000, 2500 and 4000 products;
- reading of 1000, 2500 and 4000 products;
- edition of 1000, 2500 and 4000 products;
- removal of 1000, 2500 and 4000 products.

Initially, the load was defined using a load of 1000, 5000 and 10000 products as an example. This represents a low-load scenario, a medium-load scenario, and

one with high load. However, the machine used cannot perform the processing because image processing requires many resources that the machine did not have available. Therefore, the load was reduced until reaching these values 1000, 2500 and 4000, respectively representing the small, medium and high load scenario that the machine was able to process. Testing the execution time of CRUD operations was carried out based on user interactions using a button that executes each of the operations with the indicated load. The performance of each *framework* was evaluated by capturing the execution time within the code, using the JavaScript performance API. The collection of other metrics was obtained from the Microsoft Edge Devtools developer tool.

Figure 3 shows the application interface where the different tests were carried out. This interface is the same as the one generated using each of the *frameworks.* In order for the products to reflect reality, the [4] to generate one hundred and fifty descriptions of ChatGPT product was used as shown in Figure 4 and then the AI API of OpenAI's DALL.E2 [5] image generator was used to generate product images based on the descriptions provided by ChatGPT. The *script* to generate the images is shown in Figure 5. In total, 1,500 images were generated, ten for each product description. These images were used to perform the test.

Figure 3 – Test interface

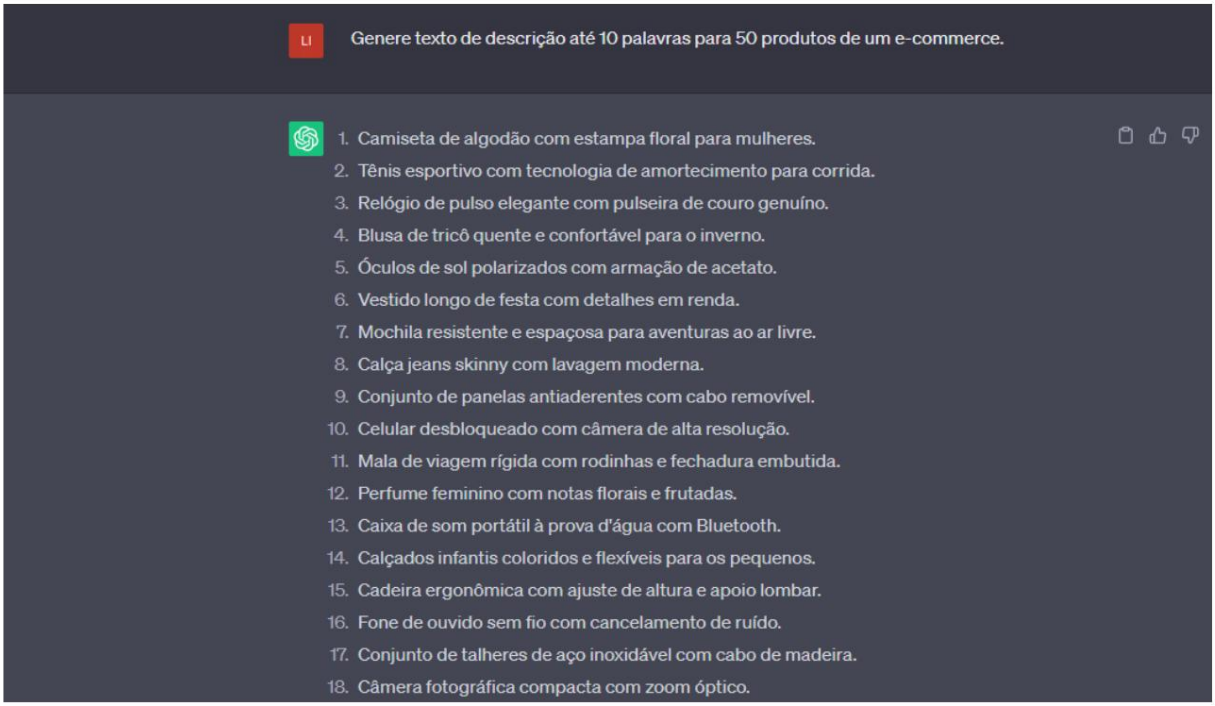


Source: from the Author (2023).

[4] ChatGPT is an *online* artificial intelligence chatbot developed by OpenAI, launched in November 2022. The name "ChatGPT" combines "Chat", referring to its *chatbot functionality,* and "GPT", which stands for *Generative Pre-trained Transformer,* a type of large language model and a prominent framework for generative artificial intelligence.

[5] DALL.E2 is an AI system that can create realistic images and art from a natural language description.

Figure 4 – Product description generation prompt using ChatGPT



Source: ChatGPT, by the Author (2023).

Figure 5 – Script that generates product images using description provided by ChatGPT from OpenAI API call

```
// Função para gerar 10 produtos
async function gerarProdutos() {
  const produtos = [];
let c1=0,c2=0;
  for (let i = 0; i < descriptions.length; i++) {
    if(i!=0 && i%5==0){
      console.log('Esperando 2 minutos');
      await new Promise((resolve) => {
        setTimeout(() => {
          resolve();
        }, 2*60000); // 60000 milissegundos = 1 minuto
      });
    }
    // console.log(`'description ${i+1}:>> '`, descriptions[i]);
    const response = await axios.post('https://api.openai.com/v1/images/generations',
    {
      'prompt':  descriptions[i],
      'n': 10,
      'size': '256x256'
    },
    {
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ' + process.env.OPENAI_API_KEY
      }
    }
  );

```

Source: from the Author (2023).

4.4 Comparative evaluation of frameworks

Based on the results of the performance tests, a comparative evaluation of the implemented applications was carried out. The evaluation showed which of the alternatives is the best option in terms of package size, loading time, rendering time, memory usage and execution time.

Next, in chapter 5, the expected results of this work are presented, expressed through tables and graphs. Then, an analysis of the results obtained is carried out.

5 Results

This chapter presents the results of the study by evaluated metrics.

5.1 Bundle size

Table 1 shows the *bundle* of applications from the three *frameworks,* measured in KB. Vue generated the smallest package, React generated the second smallest, and Angular generated the largest. In other words, the Vue *bundle* is 404 KB, the React bundle is 488 KB and the Angular bundle is 500 KB. Therefore, having the smallest packet generated is an advantage, as not every device has memory capacity to store an application.

Table 1 – Data obtained in the experiments

	Angular	React	Vue
Bundle (KB)	500	488	404
Charging (ms)	7	6	6
Rendering (ms)	4	4	6
JS Heap (MB)	2.4-6.2	2.8-3.8	2.5-3.0

Source: from the Author (2023).

Table 1 shows that Vue is the *framework* with the smallest *bundle ,* React with the second smallest and Angular with the largest *bundle.* The size of the *bundle* generated by Angular is 2.46% larger than React and 23.76% larger than Vue. This means that the application implemented by Vue requires less bandwidth, React's comes in second place and Angular is the one that needs the most bandwidth.

These differences can be explained by a number of factors, including characteristics of the *frameworks,* the libraries they include and the configuration options that they offer. Angular is a complete *framework ,* which includes a wide range of features such as dependency injection, routing, and *templating.* This may contribute to the its larger size. React is a *framework* more focused on rendering components, while Vue is a more modular *framework ,* which allows developers to choose which libraries and resources to include. This may explain the smaller size of these two *frameworks.*

It is important to highlight that the *bundle* is not the only factor to be considered when choosing of a *framework.* Other important factors include ease of use, flexibility and the support community. However, the *bundle* can be an important factor in

applications that need to load quickly or that have bandwidth constraints .

5.2 Charging time

Table 1 shows the loading time of applications for the three *frameworks,* measured in milliseconds. React and Vue app have the shortest loading time, followed by Angular. In other words, React and Vue have a loading time of 6 milliseconds, while Angular has a loading time of 7 milliseconds. This 1 millisecond difference represents a 16.67% increase over React and Vue's load time.

This difference is small, but can be significant in applications where charging time is critical. For example, in mobile applications, where bandwidth is limited, a 1 millisecond increase can make a noticeable difference in the user experience.

There are several reasons why Angular has a slightly longer load time than React and Vue. One of the reasons is that Angular is a more complex *framework ,* with more code and libraries. Another reason is that Angular uses a compiler to convert TypeScript code to JavaScript, which adds an additional step to the loading process.

Overall, the load time difference between React, Vue, and Angular is small. However, this difference can be significant in applications where charging time is critical.

5.3 Rendering time

Table 1 shows the rendering time of applications from the three *frameworks,* measured in milliseconds. Angular and React have the shortest rendering time, followed by Vue with the longest. The rendering time differences between the three *frameworks* are small but significant. Angular and React have the same rendering time, 4 milliseconds. Vue's, in turn, has a 50% longer rendering time of 6 milliseconds.

This 2 millisecond difference may seem small, but it can be significant in applications that require high rendering performance. For example, in applications that require screen content to update quickly, such as games or video *streaming* applications , a difference of 2 milliseconds may be noticeable to the user.

Furthermore, the 50% difference between Vue and the other two *frameworks* can be significant in applications that require frequent rendering. For example, in applications that require screen content to be updated based on external data, such as weather or weather, a 50% difference may result in greater resource consumption.

In summary, the rendering time differences between the three *frameworks* are small but significant. Vue's rendering time is 50% longer than Angular's and React's. This difference can be significant in applications that require high rendering performance or frequent renders.

Here are some examples of how rendering time differences can affect the user experience:

In a game, for example, a difference of 2 milliseconds can result in a noticeable delay in the game's response to user commands. In a video *streaming* application , a difference of 2 milliseconds can result in a noticeable delay in updating the image on the screen. In an application that displays the current time, a 50% difference could result in a time update that is 25% slower.
Therefore, when choosing a *framework* for your application, it is important to consider rendering time , especially if your application requires high performance or frequent rendering.

5.4 Memory Usage

Table 1 shows the memory usage of the three applications implemented using *the frameworks,* measured in megabytes. Figure 6 shows that the application implemented using Vue is the one with the lowest memory usage, with consumption ranging from 2.5 to 3.0 megabytes. This difference can be attributed to a component model similar to React, but with some optimizations that reduce memory consumption.

The React application has the second highest memory usage, with consumption ranging from 2.8 to 3.8 megabytes. This difference can be attributed to a simpler component model, which requires less memory to store information about the components and their states.

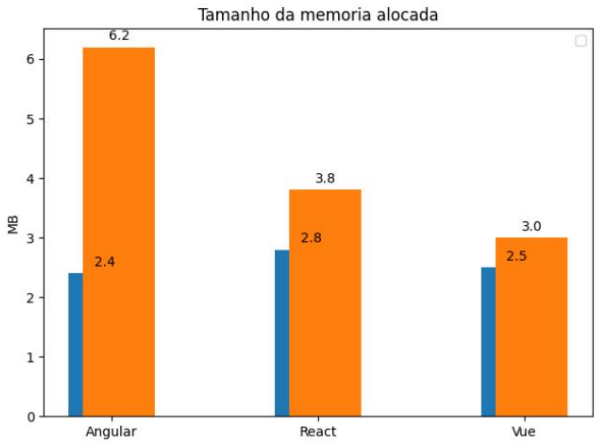
The Angular application is the one with the highest memory usage, with consumption ranging from 2.4 to 6.2 megabytes. This difference can be attributed to a number of factors, including:

- the use of a more complex component model, which requires more memory to store information about components and their states;
- the use of a more complete dependency injection system, which also requires

more memory;

- the use of a broader library of components and services, which can also increase memory consumption.

Figure 6 – Figure of the space (MB) allocated in memory initially by each *framework* , where blue represents the smallest allocated space and orange, the largest allocated space.



Source: from the Author (2023).

In percentage terms, Angular uses about 63.15% more memory than than React and 106.67% more memory than Vue.

This difference in memory usage must be considered when choosing a *framework* for developing a web application. Choosing the ideal *framework* will depend on a number of factors, including the size and complexity of the application, the resources required and the expected performance.

In small, simple applications, the difference in memory usage may not be significant. However, in large, complex applications, the difference can become important, especially if the application runs on devices capable of limited.

5.5 Execution time

Table 2 shows the average execution time for each task for each technology. It can be seen that, in general, Angular performed the worst, followed by React and Vue.

Table 2 – Average execution time x Process

Execution		Average Time (ms)		
		Angular	React	Vue
Create time	1000	17836	17603	16705
	2500	46553	44097	43906
	4000	77313	78034	73317
Read time	1000	43	41	44
	2500	67	57	68
	4000	102	94	97
Update time	1000	38234	36347	34932
	2500	101166	90668	89543
	4000	169862	153900	169808
Delete time	1000	9283	15682	16598
	2500	86612	23845	67686
	4000	178000	38799	151553

Source: from the Author (2023).

Table 3 shows the standard deviation of execution of each task for each technology. The standard deviation is a measure of the variability of results. Standard deviation values Low values indicate that the results are more consistent, while standard deviation values higher values indicate that the results are more dispersed.

Table 3 – Standard Deviation of execution x Process

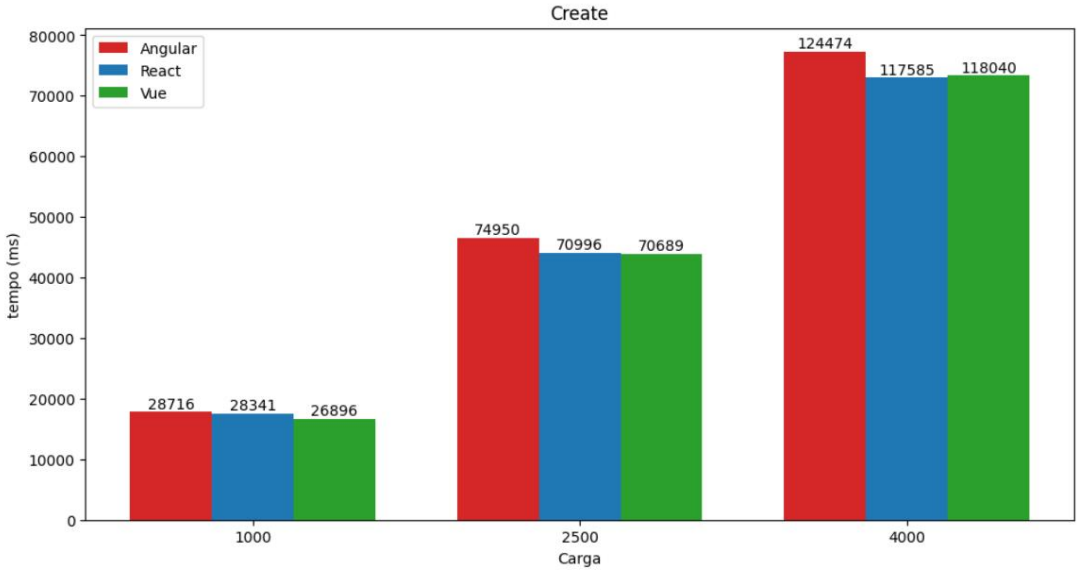
Execution		Standard Deviation (ms)		
		Angular	React	Vue
Create time	1000	43	1504	21
	2500	474	383	1098
	4000	1596		601
Read time	1000		4	456
	2500	4	14	1
	4000	7	9	12
Update time	1000	661	1734	179
	2500	1711	898	1149
	4000	18031	5760	19270
Delete time	1000	242	131	265
	2500	702	615	2048
	4000	2141	710	14769

Source: from the Author (2023).

Figure 7 shows that the average time spent on the product creation operation for different loads. The React app and Vue app perform best, with the trend that shows React can be better than Vue if the load continues

increasing. It is possible to verify that the Angular application has the worst performance in this regard.

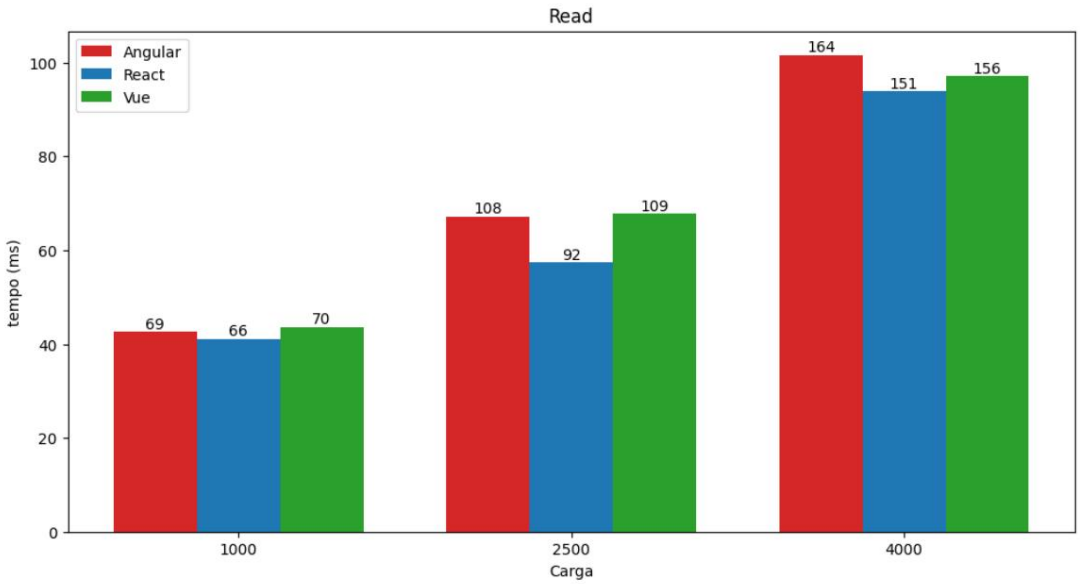
Figure 7 – Execution time (ms) of the product creation operation



Source: from the Author (2023).

Figure 8 shows that the average time spent on the product reading operation for It is under different possible to verify that the React application has the best performance loads. in this regard. It can be said that Vue has the second best performance, although Angular has better performance considering the load of 2500 products, but the difference of 1 ms may be insignificant when considering a margin of error in the measurement. Therefore, Angular had the worst performance in this regard.

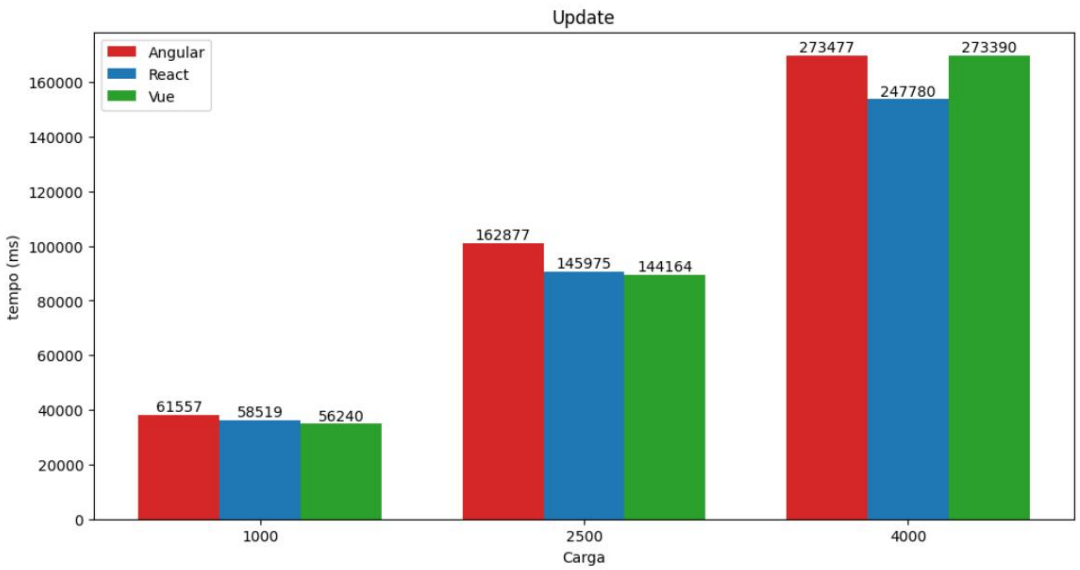
Figure 8 – Execution time (ms) of the product reading operation



Source: from the Author (2023).

Figure 9 shows that the average time spent on the product update operation for the different loads. It is possible to verify that the React application is the best performance for loading 4000 products. However, Vue has the best performance respectively in loads of 1000 and 2500 products. If you consider the small difference between Vue and React, it can be said that React has the best performance and Vue has the second best performance. Therefore, the Angular application has the worst performance.

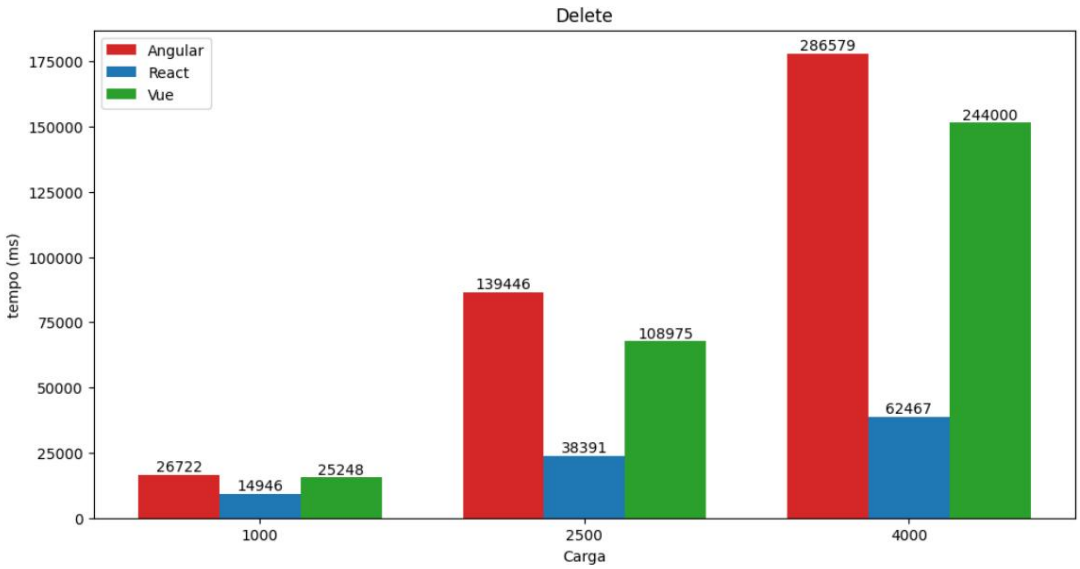
Figure 9 – Execution time (ms) of the product update operation



Source: from the Author (2023).

Figure 10 shows that the average time taken for the remove operation. You can the different payloads. better verify that the React application has the product for performance in this regard. Vue's has an intermediate performance and Angular's has the worst performance. Therefore, for product removal, React's performs best by a considerable margin.

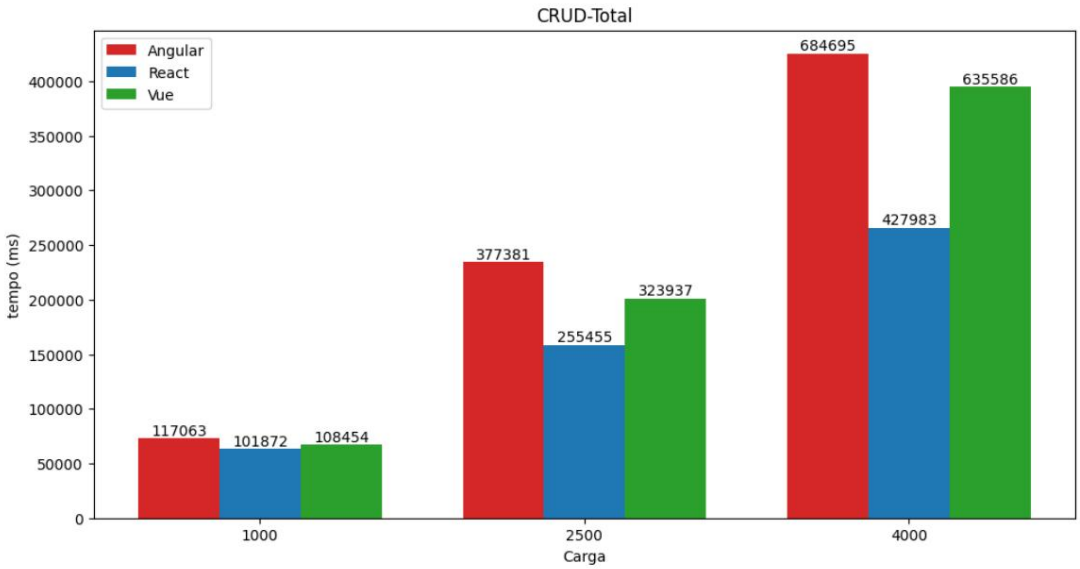
Figure 10 – Execution time (ms) of the product removal operation



Source: from the Author (2023).

Figure 11 shows that the average total time spent for the CRUD operation It is product for the different loads. possible to clearly verify that the application of the React has the best performance. Vue's has an intermediate performance and Angular's has the worst performance. If you consider the CRUD operation as unique, React stands out by a considerable margin, especially when the load is large.

Figure 11 – Total execution time (ms) of the product CRUD operation



Source: from the Author (2023).

The findings of this research differ from those obtained by Kaluža, Troskot and Vukeliÿ (2018), who concluded that there is no ideal JavaScript framework for both SPA and MPA. Although Ziani (2021) conducted a comparative analysis of JavaScript frameworks, it is worth noting that his study was qualitative, while the present work adopts a quantitative approach. Thus, the results presented here corroborate the statement by Diniz-Junior et al. (2022), indicating that Angular demonstrates superior performance in terms of rendering.

It is worth noting that the results obtained in this analysis are based on a specific set of data and conditions. Results may be different for other data sets or conditions. Therefore, this study presents some limitations that must be considered when interpreting the results.

First, the study was carried out in a controlled environment, with a specific set of data. Considering that this work adopted the microservice architecture as an approach, it is possible that the results will be different with different architecture, or in different environments such as Linux or MacOS, or with datasets
many different.

Second, the study focused on a limited set of performance measures. It is possible that other qualitative factors, such as ease of use or maintainability, are more important for some applications than the factors considered in this study.

Thirdly, the study did not consider the facilities and resources available for *frameworks* that can help with development.

6 Conclusion

From the results obtained in this comparative analysis between the Angular, React and Vue *frameworks ,* it can be concluded that the application implemented using React presents the best overall performance, followed by Vue and Angular in last position.

In terms of the size of the bundle generated to put into production, Vue generated the smallest *bundle,* React generated the second smallest and Angular generated the largest of the three. This means that the Angular application requires more bandwidth to load, which can affect the application's performance on devices with less processing power and slower internet access.

In terms of loading time, the React and Vue applications are the fastest, followed by Angular. This means that applications developed with these *frameworks* will load faster, which can improve the experience
of user.

In terms of rendering time, the Angular application and the React application perform the same. They render faster than the Vue app. This means that applications developed with these *frameworks* will have similar performance in terms of content rendering.

In terms of memory usage, the Vue application uses the least memory, followed by React and Angular in last position. This means that applications developed with Angular may consume more memory in other tested technologies, which may affect application performance on devices with lower processing capacity.

In terms of execution time, the React application has the best performance in all CRUD operations, followed by Vue and Angular in last place.
This means that applications developed with React will perform better in CRUD operations.

Based on the results obtained, the following recommendations can be made:

- for applications that need to load quickly, React or Vue are the way to go
  better options;
- for applications that need to use less memory, Vue is the best option;
- for applications that need to perform well in CRUD operations, the
  React is the best option.

Some future work that could be carried out to expand the results from this study include:

- carry out the study in a cloud environment, with different types of applications and data;
- consider other factors when making a decision, such as ease of use or maintainability;
- compare SPA *frameworks* on *mobile* hardware such as Android or IOS.

References

ANGULAR. Google, 2023. Available at: <https://angular.io/guide/what-is-angular>.

BERNERS-LEE, T. et al. The world-wide web. *Communications of the ACM,* ACM New York, NY, USA, v. 37, no. 8, p. 76–82, 1994.

DINIZ-JUNIOR, RN et al. Evaluating the performance of web rendering technologies based on javascript: Angular, react, and vue. P. 1–9, 2022.

FERREIRA, HK; ZUCHI, JD Comparative analysis between javascript-based frontend frameworks for web applications. *Technological Interface Magazine,* v. 15, no. 2, p. 111–123, Dec. 2018. Available at: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/502 >.

FIELDING, RT *Representational State Transfer (REST).* University of California, 2000. Available at: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.

JAVASCRIPT. Mozilla Corporation's, 2021. Available at: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>.

KALUŽA, M.; TROSKOT, K.; VUKELIÿ, B. Comparison of front-end frameworks for web applications development. *Zbornik Veleuÿilišta u Rijeci,* Veleuÿilište u Rijeci, vol. 6, no. 1, p. 261–282, 2018.

KORNIENKO, D.; MISHINA, S.; MELNIKOV, M. The single page application architecture when developing secure web services. In: IOP PUBLISHING. *Journal of Physics: Conference Series.* [Sl], 2021. v. 2091, no. 1, p. 012065.

LEWIS, J.; FOWLER, M. *Microservices: a definition of this new architectural term.* Martin Fowler, 2014. Available at: <https://martinfowler.com/articles/microservices.html>.

MICROSOFT. Microsoft, 2023. Available at: <https://learn.microsoft.com/pt-br/azure/architecture/guide/architecture-styles/microservices >.

MOZILLA. Mozilla Corporation's, 2023. Available at: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript>.

NEWMAN, R. et al. Web 2.0—the past and the future. *International Journal of Information Management,* Elsevier, v. 36, no. 4, p. 591–598, 2016.

NEWMAN, S. *Building microservices.* 2nd ed. Sevastopol: O'Reilly Media, Inc., 2021.

REACT. Meta Platforms, Inc., 2023. Available at: <https://pt-br.reactjs.org/>.

SHARMA, D. et al. A brief review on search engine optimization. In: *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence).* [Sl: sn], 2019. p. 687–692.

STACKOVERFLOW. *Stack overflow developer survey 2021.* 2021. Available at: <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks >.

VUE. MEvan You, 2023. Available at: <https://vuejs.org/>.

W3TECHS. *Usage statistics of JavaScript as client-side programming language on websites.* 2022. Available at: <https://w3techs.com/technologies/details/cp-javascript>.

ZIANI, A. *Quantitative and qualitative comparison of web development frameworks.* Dissertation (Master's) — Université de Namur, 2021.

Appendices

APPENDIX A – Code

<https://github.com/lildiop2/tcc>

Source: from the Author (2023).