WordPress Architecture Modernization Projects

Daniela Ferreira[1], Teresa Pereira[2(✉)], Isabel Mendes[3], and António Amaral[4]

[1] Information Systems Department, University of Minho, Campus de Azurém, 4804-533 Guimarães, Portugal

[2] Information Systems Department and Algoritmi Center, Universidade do Minho, Campus de Azurém, 4804-533 Guimarães, Portugal
tpereira@dsi.uminho.pt

[3] School of Technology and Management (ESTGA-UA), Universidade de Aveiro, and Algoritmi Center, Rua Comandante Pinho e Freitas, nº 28, 3750-127 Águeda, Portugal

[4] School of Engineering (ISEP), Polytechnic Institute of Porto, R. Dr. António Bernardino de Almeida 431, 4249-015 Porto, Portugal

Abstract. Technological evolution is very present in today's world. The Internet of Everything (IoE) is one of the next steps in this evolution. WordPress is a website-building tool that is used today to its minimum capability. The main problem is how to make WordPress customizable and reliable to support device networking and communication. This can be done by implementing a new architecture that supports WordPress as a powerful tool, allowing for scalability and maintenance. The adoption of a recent trend known as DevOps is an important step in creating and developing a strong website and ensuring its integration with multiple devices. It emphasizes continuous testing, delivery, and integration. This article is based on a literature review to justify using micro-services architecture and the DevOps approach to build a reliable and robust working tool in WordPress that can be part of the IoE.

Keywords: WordPress · Architecture · Device networking · IoE

1 Introduction

In the world of software development, everything is changing at a very fast pace, along with the latest trends and the demands of clients. For that reason, companies need to be prepared for these changes in projects. To do this, companies should keep up with the market.

The Internet of Everything is currently the next big thing in technology due to the possibility of connecting devices, data, and people. It is a very important aspect of everyday life because it can be the bridge between the human world and machines. However, to be used, it must be built with tools and technologies that allow communication and networking to happen. These tools must be reliable and robust enough to support the demands and requirements.

T. Pereira et al. (Eds.): IoECon 2024, LNICST 607, pp. 121–131, 2025.
https://doi.org/10.1007/978-3-031-84426-3_10

WordPress is the most widely used CMS (content management system) in the world, with between 43% and 65% market share [1]. However, it is mostly used for smaller projects, and its architecture has not kept up with modern web development standards, which can be a problem if one wants to implement an Internet of Everything approach using WordPress. This can be avoided by changing the way we, as users, use WordPress. That is, by recognizing WordPress as a powerful work tool capable of developing and running large and complex websites and by applying the latest trends in architecture and DevOps.

Therefore, the main goals of this article are to automate a WordPress instance with modern standards of web development and improve the automation mechanisms for new WordPress projects and instances, and the research question is: "How to build a WordPress instance capable of maximizing the benefits and features of the tool?"

This paper will be an analysis of the state-of-the-art of WordPress and other technologies, as well as a new architecture and the DevOps approach to work. These will be able to give WordPress the scalability and flexibility it lacks and demonstrate the importance of implementing continuous integration and delivery in projects that will implement the Internet of Everything. A new architecture will also be proposed in this article based on the literature review.

2 Methodology

This paper essentially focuses on a literature review to justify the theoretical architecture presented below. To proceed with the state-of-the-art, some scientific repositories have been defined where important articles and information sources relevant to this paper can be found. These include Scopus, Web of Science, and Google Scholar. Within these repositories, selected studies with great proximity to and impact on the scope of this work were then analyzed. To select such scientific papers, they had to be Portuguese or English-written from 2016 and above, except for articles relevant to the area, and an examination of the abstract, introduction, and conclusion was made. When it came to the technologies to be used in the architecture presented, the main source of information was their official websites and documentation.

2.1 Research Results

Based on the methodology outlined above, articles and other information were selected and further used to write the literature review for this document. The Table 1 below shows the number of articles used for each state-of-the-art topic and examples of those authors.

Of the ten articles selected for the microservices theme, four described the advantages of this architecture and another four the disadvantages. However, two of them focus on the advantages as well as the disadvantages.

The topic of DevOps is divided into three sub-themes: continuous delivery, continuous integration, and continuous development. The same three articles were used for all these sub-themes. To explain the concept of DevOps, one article was used that was based on a study conducted through interviews and surveys of companies.

Table 1. Topics and number of articles.

Topic	*Number of articles*	*Authors (examples)*
Microservices	10	Ponce et al. (2019)
DevOps	5	Lwakatare et al. (2019)
WordPress	9	Bartlett (2022)
PHP-FPM	3	PHP-FPM Organization
NGINX	3	Kholodkov (2015)
Docker	3	Morabito et al. (2018)
Reverse proxy	1	NGINX Organization
Node.js	1	Node.js Organization

WordPress is one of the most important topics of this literature review, and so it used nine articles and other sources of information. Out of the nine sources of information, two described a new architectural perspective of WordPress, and another two identified how to use the technology as a dependency.

When it comes to studying the technologies to be used in the proposed architecture, three sources of information were used to describe the technical side of these technologies.

Most of the articles selected are exploratory and based on literature reviews.

3 State-of-the-Art

3.1 Microservices

Microservices are a form of architecture that is used by large companies such as Netflix, Amazon, Spotify, and Twitter [1]. It is based on developing software systems into separate and autonomous services, each with a specific focus, its own repository, automated development pipelines, its own database, and being independently scalable. As stated by [2], this architecture emerged from market demand for new features in applications, and microservices can address concerns such as balancing high scalability with faster implementation because services can be built and deployed by independent teams that focus on improving the service itself and its business value. It also allows for increased agility in software projects by turning pieces of software into independent units, both in terms of development, releases, deployment, and scalability [1].

According to [3, 4], the services in this architecture are the unit of modularity, and it is modularity that allows for service isolation, meaning that if one service is compromised, the rest will not be. Because they are built like independent units, DevOps and continuous delivery practices are much easier to adopt and more acceptable within the organization.

Advantages of Microservices

According to some authors [2–5], the microservices architecture prefers lightweight

middleware and REST protocols to perform service communication and provides benefits such as agility, autonomy, heterogeneity, resilience, scalability, more efficient teams, code reuse, and ease of system complexity. As a result of these benefits, this architecture is a good choice for organizations that are using the DevOps approach [4]. The service teams are free to use any technology and programming language they choose as long as it does not compromise the overall functionality of the system.

Disadvantages of Microservices
As disadvantages, as concluded by [3, 4, 6, 7], the microservices architecture forces developers to deal with distributed systems with higher complexity (decisions about communication mechanisms between services), fault tolerance, monitoring, testing, and database transactions. Things get a little more complex when it comes to constraining services, finding the right set of services, and determining the size of the services. Microservices require careful coordination, computational overhead, and IDEs that are not designed for distributed systems.

Microservices and Containers
Containers are superior to virtual machines for microservices because of their lightweight nature, ease of management, and fast development [8, 9]. They are the standard for microservices development [1] due to their independent nature and development, image management, container management, and resource management. Together, containers and microservice architecture can efficiently and easily meet customers' needs.

Microservices and DevOps
DevOps is a frequently used strategy to mitigate the impact of microservices [10]. That is because both support the idea of breaking down complex problems into smaller components, which can be done with small teams, and because microservices in containers can run independently, as DevOps promotes continuous integration and deployment.

The combination of microservices and DevOps is advantageous because it enables frequent software releases, ensures system reliability and scalability, provides resilience in the event of failures, enables the management of distributed teams, and pro- vides agility and operational efficiency [10].

3.2 DevOps

In a study of five companies [11], most respondents described DevOps as the responsibility of the development team to design, implement, test, deploy, and maintain web applications and services in production. They interact with the operations team to help the development team automate pipelines, address security, scalability, and performance issues, and resolve incidents together.

It can be defined as a collaboration between developers and infrastructure engineers for faster application development and deployment [12]. It automates the software development process from development to production using methods such as continuous integration and delivery [10]. DevOps is a collaborative and multidisciplinary organizational effort to help automate the continuous delivery of software and complements Agile techniques by encouraging iterations in short cycles to ensure the stability of the

software product [13]. However, [11] states that the concept of DevOps is not universally agreed upon in the literature and that the lack of a clear and solid definition may be purposeful to allow each team to tailor it to their needs.

A study identified the practical implications of adopting and implementing DevOps [11]. These included a steep learning curve, manual approval and testing of updates or new features, and value-added activities to align responsibilities, ensure trust in the development team, and make investments in automated testing.

DevOps is strongly associated with three principles: continuous integration, continuous deployment, and continuous delivery. These three are described below:

Continuous Integration
Continuous Integration (CI) is a software development practice in which team members integrate code frequently [14]. CI allows companies to release products in shorter, more frequent cycles and improve software quality. Regular code merging reduces software integration difficulties and costs [12].

Continuous Delivery
Continuous Delivery (CD) is a software engineering practice in which teams build, test, and release software in short cycles and support automated builds and testing [12]. It aims to ensure that the application is always ready for production by subjecting it to automated code quality and defect testing. It also helps reduce risk, lower costs, and get user feedback faster [14].

Continuous Deployment
Continuous deployment is the practice of continuously and automatically deploying the application to production environments as soon as changes are ready. It differs from continuous delivery in that it focuses on deploying changes directly into the production environment, while continuous deployment is strictly automatic. Continuous Deployment is a continuation of Continuous Delivery and can be adopted by all organizations, unlike Continuous Deployment [14].

DevOps can support architecture change by enabling faster and more effective testing, deployment, and monitoring of changes; boosting team collaboration and communication; and improving the scalability and durability of the architecture. Collaboration and communication enable the team to understand the changes and why they are happening, while CI/CD automates testing, deployment, and monitoring, making risks more visible and detecting failures earlier in the process. In this sense, DevOps is a method that can be used to smooth and accelerate the process of architecture reengineering.

When it comes to the Internet of Everything, DevOps is also an important approach because it enables the continuous testing and integration that the IoE needs to succeed. Collaboration and communication help teams integrate devices more efficiently, and continuous integration and continuous delivery help automate testing and reduce the risk of failures. By increasing scalability, DevOps helps handle the large amount of data processed by the devices. With DevOps, developers can create platforms with a good user experience.

3.3 WordPress

WordPress is an open-source content management system created in 2003 by Mike Little and Matt Mullenweg. It uses technologies such as PHP and MySQL. Is licensed under the GPL v2 and is used by 43% of websites worldwide [15]. With this tool, it is possible to build a website and manage its aspects, such as content and design features. In the past, WordPress was only used as a tool to create blogs. Today, it is used to create any kind of website, mainly eCommerce [16].

WordPress can be edited by anyone and uses PHP and MySQL as its programming languages and databases. It is easy to install, manage, and learn. It is flexible enough to adapt to all kinds of sizes and devices and supports all types of media files. The platform offers a large community of software developers to help troubleshoot issues, frequent updates, and user-friendly Search Engine Optimization (SEO) [16, 17]. Web development professionals use WordPress to develop websites for their customers because it reduces development time, is advantageous, and provides a good CMS that clients can easily use [17].

Headless WordPress

WordPress is an application with a traditional architecture, based on two subsystems: content storage and management in the backend and content delivery to the frontend. This architecture ensures that the CMS offer its users everything they need to manage their website, both in terms of assets and more aesthetically, in terms of design and presentation. However, by using a classic WordPress approach, the developer may lose control over the structure of the website and may not be able to customize it to best suit the needs of his project [18]. Classic WordPress is monolithic (it bundles the backend and the frontend), slow because it forces the site to load more than is needed at the moment, and vulnerable because it uses plugins to customize the website [19].

Headless WordPress is an approach that separates the backend (body) from the frontend (head) by removing the latter. The interface for creating, editing, and updating content remains, but the part that groups the themes and plugins is gone [19]. Headless WordPress was chosen because it makes the site more secure, provides conditions for better performance, can be integrated with any platform, can be made available on any device, and can be easily published to social networks. The developer gets full control over his website and the teams can develop both, the backend and the frontend, as they wish [19].

WordPress as a Dependency

An alternative approach to web development in WordPress is to treat the tool as a dependency. Since WordPress is based on the PHP programming language, a PHP dependency manager, Composer, is used. WordPress can be considered a project dependency because customers want to get a product in website format without having to know how WordPress is used, whether it is installed normally or as a dependency.

In a WordPress context, dependencies are the WordPress core, plugins and themes, and other dependencies that meet the needs of the project. By default, Composer installs the dependencies in a folder called Vendor, but WordPress itself is not in this folder and is automatically copied to a subdirectory of the project's root directory.

There are two commonly used viewpoints for using WordPress as a dependency: Roots, with its Bedrock configuration, and John Bloch Composer.

According to [20], when using the Bedrock technique, the WPackagist is already in use, as opposed to the John Bloch alternative, which requires specifying which repository to use. WPackagist is a repository of plugins and themes that have been modified and are suitable for use and management in Composer. John Bloch's technique, on the other hand, is more straightforward.

Since not using Composer requires committing all code related to the project (theme, plugin, or WordPress core), having WordPress as a dependency allows the development team to have a cleaner and smaller code repository [21].

3.4 Technologies

PHP-FPM

PHP-FPM is a fastCGI Process Manager and is used for high-traffic websites because it is an efficient way to minimize memory and other resource consumption while keeping platform performance high [22, 23]. PHP-FPM can manage processes, restart the whole service in case of problems, detect function too time-consuming and is compatible with NGINX.

Since WordPress is built with PHP, PHP-FPM is the best alternative because it ensures that all site visitors get same access to the server and caches already executed scripts to limit resource consumption [24].

NGINX

According to [25], NGINX has emerged as a reliable and scalable web server and is the preferred option of many developers and engineers. It is regarded as a tool that can be combined with PHP and has a simple but scalable architecture, easy configuration, and minimal memory requirements. It is open source and supported by a large community that provides good support to developers with their concerns.

NGINX is event-driven and uses an asynchronous method. Each request can be executed simultaneously by the worker without interfering with other requests [26]. It routes requests to the best server in the cluster [27].

NGINX effectively performs operations as an intermediate to improve website speed.

Proxy

Due to its popularity, NGINX has grown to include reverse proxying and load balancing to control traffic and distribute it to slower upstream servers, ranging from databases to microservices [28].

A reverse proxy is used to coordinate requests to the most appropriate server and responses to the correct client. This type of proxy is advantageous because it can contribute to efficient load balancing, accepts multiple communication protocols, acts as a security barrier due to its role as an intermediary in the client-server connection, and can cache the most frequent requests to reduce communication time. These benefits enable the reverse proxy to improve the performance, security, and scalability of Web application.

Docker
Docker is an open-source platform created in 2013 that allows users to easily build, run, manage, and remove applications [29]. A Docker container is an executable instance of a Docker image, which can be created by editing a specification file or using images from repositories. Another alternative is to use images from repositories; to do so, the image needs to be installed locally if it is not already cached [30]. The containers run the images and create isolated and controlled environments for their applications. It uses a layered file system and version control to enable continuous delivery and integration [31]. It integrates well with DevOps and the microservices architecture due to its benefits of isolation, portability, and scalability.

Node.js
This technology is an open-source platform that runs JavaScript. It allows developers to build applications quickly and efficiently. Node.js has a large library of modules and packages, making it very easy for developers to use it in projects of different scopes [32]. These are available in the NPM registry.

Node.js can be integrated with many tools and frameworks and is fast, scalable, versatile, and efficient.

4 Design of the Microservices Architecture

According to the microservices architecture, services should be small and have only one business need. From a Docker perspective, each service will be a container.

When it comes to WordPress itself, it will be handled as a dependency on the project, meaning that the service itself will not be WordPress but PHP-FPM. Composer should then be installed inside that container.

Even though Composer is a powerful tool, it does not have the ability to manage PHP scripts at runtime. On the other hand, Composer is only used to install a dependency (WordPress) and does not run anything else. This gives reason to use PHP-FPM as the only service because PHP-FPM will run if the application runs, fulfilling the business need to keep WordPress up and running for as long as necessary, unlike Composer.

The next service is the database, which is used to maintain and update the data on the WordPress website. To do this, it should be linked to the PHP-FPM service, which can be done through ports.

Node.js is an important tool for managing the assets of the project. This is related to the fact that WordPress is developed in a headless form. The content of the website, which includes images, scripts, and styles of themes and plugins, should be loaded by node. This is an important service because it oversees the front end of the website.

The other two services required to complete this architecture are NGINX and NGINX Reverse Proxy. The reverse proxy handles the requests and responses back to the rightful servers and clients, while NGINX works as a web server.

The proposed architecture can be observed in the next image (Fig. 1):

The implementation of this architecture should be based on continuous integration and delivery to ensure smoother integration of services, faster application development and deployment, and less time spent on errors.

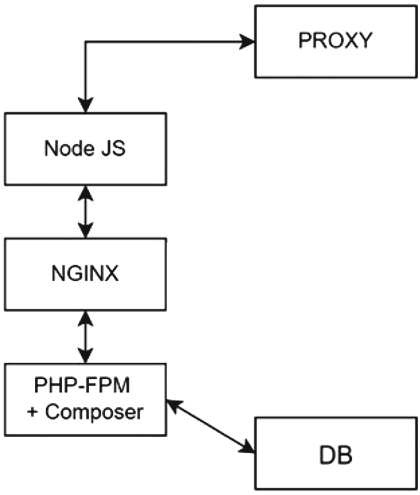

Fig. 1. Proposed Architecture

5 Conclusion

WordPress can be a powerful tool when used to its maximum potential, taking advantage of new and emerging trends and WordPress' own characteristics. By breaking down the monolithic architecture of the tool into a headless and reliable architecture, WordPress can be improved in scalability, reliability, and maintainability, and further, it can meet the needs of the IoE by integrating all devices in the home or office and processing that data into readable and understandable dashboards.

By adopting the DevOps approach, developers can combine the best of both worlds and improve both the quality of work and productivity. Continuous integration and continuous delivery are important to the development cycle of websites that integrate the IoE. That is achieved by continuously testing and developing solutions to transform and display data from other devices.

It is important to make WordPress easy to customize because APIs should be used to communicate with other machines. This can only be done by creating a reliable architecture and integrating WordPress with other technologies, as discussed above.

A new architecture like microservices and the implementation of technologies that make WordPress stronger and more flexible are key to a powerful IoE project using the CMS.

6 Future Work

In the future, it is expected that a security layer such as SSL or TLS will be added to protect the overall system. This protocol should be added to the NGINX Reverse Proxy because it is the first service to communicate with the client. This change will allow the application to only respond to requests with additional security, making communication more secure for everyone.

Since this paper only reflects the theory of the project, it is indeed required in the future to be put into practice in a real-life scenario. Each service should be developed and tested individually, integrated with the rest of the services by establishing the necessary protocols, and automated in the process of installing WordPress as a dependency. After checking that each container works well by itself and with the others, the application should also be tested. The tests can evaluate the performance and identify possible risks and failures. The work should also be monitored and optimized with tools and methods such as DevOps and its CI/CD.

References

1. Sultan, S., Ahmad, I., Dimitriou, T.: Container security: issues, challenges, and the road ahead. IEEE Access. **7**, 52976–52996 (2019). https://doi.org/10.1109/ACCESS.2019.2911732
2. Ponce, F., Marquez, G., Astudillo, H.: Migrating from monolithic architecture to microservices: a rapid review. In: 2019 38th International Conference of the Chilean Computer Science Society (SCCC), pp. 1–7. IEEE, Concepcion, Chile (2019). https://doi.org/10.1109/SCCC49216.2019.8966423
3. Richardson, C.: Microservices Patterns: with Examples in Java. Manning Publications, Shelter Island, New York (2019)
4. Vale, G., et al.: Designing microservice systems using patterns: an empirical study on quality trade-offs (2022). https://doi.org/10.48550/ARXIV.2201.03598
5. Mazlami, G., Cito, J., Leitner, P.: Extraction of microservices from monolithic software architectures. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 524–531. IEEE, Honolulu, HI, USA (2017). https://doi.org/10.1109/ICWS.2017.61
6. Baresi, L., Garriga, M., De Renzis, A.: Microservices identification through interface analysis. In: De Paoli, F., Schulte, S., Broch Johnsen, E. (eds.) Service-Oriented and Cloud Computing, pp. 19–33. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67262-5_2
7. Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., Steinder, M.: Performance evaluation of microservices architectures using containers. In: 2015 IEEE 14th International Symposium on Network Computing and Applications, pp. 27–34. IEEE, Cambridge, MA, USA (2015). https://doi.org/10.1109/NCA.2015.49
8. Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H., Li, Z.: Microservices: architecture, container, and challenges. In: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 629–635. IEEE, Macau, China (2020). https://doi.org/10.1109/QRS-C51114.2020.00107
9. Salah, T., Zemerly, M.J., Yeun, C.Y., Al-Qutayri, M., Al-Hammadi, Y.: Performance comparison between container-based and VM-based services. In: 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), pp. 185–190. IEEE, Paris (2017). https://doi.org/10.1109/ICIN.2017.7899408
10. Waseem, M., Liang, P., Shahin, M.: A systematic mapping study on microservices architecture in DevOps. J. Syst. Softw. **170**, 110798 (2020). https://doi.org/10.1016/j.jss.2020.110798
11. Lwakatare, L.E., et al.: DevOps in practice: a multiple case study of five companies. Inf. Softw. Technol. **114**, 217–230 (2019). https://doi.org/10.1016/j.infsof.2019.06.010
12. Donca, I.-C., Stan, O.P., Misaros, M., Gota, D., Miclea, L.: Method for continuous integration and deployment using a pipeline generator for agile software projects. Sensors **22**, 4637 (2022). https://doi.org/10.3390/s22124637
13. Leite, L., Rocha, C., Kon, F., Milojicic, D., Meirelles, P.: A survey of DevOps concepts and challenges. ACM Comput. Surv. **52**, 1–35 (2020). https://doi.org/10.1145/3359981

14. Shahin, M., Ali Babar, M., Zhu, L.: Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE Access. **5**, 3909–3943 (2017). https://doi.org/10.1109/ACCESS.2017.2685629
15. Democratize Publishing. https://pt.wordpress.org/about/. Accessed 14 Nov 2022
16. What Is WordPress? Explained for Beginners. https://kinsta.com/knowledgebase/what-is-wordpress/. Accessed 17 Nov 2022
17. Bartlett, D.: WordPress. In: Easy Steps, Leamington Spa (2022)
18. Juliver, J.: What is a headless CMS? What content marketers should know. https://blog.hubspot.com/website/headless-cms. Accessed 6 Dec 2022
19. Gienow, M.: What is headless WordPress? https://www.gatsbyjs.com/blog/what-is-headless-wordpress/. Accessed 12 Feb 2023
20. McHale, C.: Using composer with WordPress. https://www.nexcess.net/blog/using-composer-with-wordpress/. Accessed 18 Dec 2022
21. Why you should manage WordPress with Composer. https://docs.platform.sh. Accessed 7 Dec 2022
22. Briones, J.-P.: PHP-FPM: the future of PHP handling. https://www.inmotionhosting.com/support/server/php-fpm/php-fpm-the-future-of-php-handling/. Accessed 12 Deb 2023
23. Why Do You Need PHP FastCGI Process Manager? https://www.plesk.com/blog/various/why-do-you-need-php-fpm/. Accessed 12 Feb 2023
24. PHP-FPM for WordPress. https://glowfroghosting.com/php-fpm-for-wordpress/. Accessed 8 Dec 2022
25. Kholodkov, V.: Nginx Essentials: Excel in Nginx Quickly by Learning to Use Its Most Essential Features in Real-Life Applications. Packt Publishing, Birmingham, Mumbaif (2015)
26. What is NGINX? https://www.nginx.com/resources/glossary/nginx/. Accessed 7 Dec 2022
27. Ma, C., Chi, Y.: Evaluation test and improvement of load balancing algorithms of Nginx. IEEE Access **10**, 14311–14324 (2022). https://doi.org/10.1109/ACCESS.2022.3146422
28. NGINX Reverse Proxy | NGINX Plus. https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/. Accessed 29 Apr 2023
29. Morabito, R., Cozzolino, V., Ding, A.Y., Beijar, N., Ott, J.: Consolidate IoT edge computing with lightweight virtualization. IEEE Netw. **32**, 102–111 (2018). https://doi.org/10.1109/MNET.2018.1700175
30. Ahmed, A., Pierre, G.: Docker container deployment in fog computing infrastructures. In: 2018 IEEE International Conference on Edge Computing (EDGE), pp. 1–8. IEEE, San Francisco, CA (2018). https://doi.org/10.1109/EDGE.2018.00008
31. Sharma, P., Chaufournier, L., Shenoy, P., Tay, Y.C.: Containers and virtual machines at scale: a comparative study. In: Proceedings of the 17th International Middleware Conference, pp. 1–13. ACM, Trento, Italy (2016). https://doi.org/10.1145/2988336.2988337
32. Introdução à Node.js. https://nodejs.dev/pt/learn/introduction-to-nodejs/. Accessed 29 Apr 2023