Future CMS for e-Business: Will Microservices and Containerization Change the Game?

Carina Landerer and Philipp Brune
Neu-Ulm University of Applied Sciences, Wileystraße 1, Neu-Ulm, Germany

Keywords: Web Services, Microservices, e-Commerce, CMS, Node.js, Docker.

Abstract: Content Management Systems (CMS) are widely adopted commodity applications today. Well-established implementations exist since many years, both commercial and Open Source, e.g. WordPress or Drupal. In consequence, research on CMS has strongly declined since its peak more than a decade ago. However, in recent years new trends for building and running large-scale web applications emerged, such as Node.js, microservice architectures and containerization, while most established CMS still use a traditional monolithic architecture. This rises the question how these emerging new technologies will challenge those established CMS implementations. Therefore, in this paper a microservice architecture for an examplified e-business CMS is proposed, and a Proof-of-Concept implementation is described. The approach is evaluated with respect to its feasibility using a qualitative empirical study. Results indicate that the approach is well-suited for building more state-of-the-art CMS in the near future, which are likely to challenge the position of the traditional monolithic implementations.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

According to Gartner, it took 20 years for Content Management System (CMS) market to become the “over-night-success” it now is (Gartner, 2017). But from the beginning in the late 1990s, it was expected to be very promising. Therefore, a lot of research concerning (Web) CMS (WCMS) was published during that time (McKeever, 2003; Vidgen et al., 2001). E.g., core functionalities such as storing content separately from its formatting and presentation attracted a lot of interest (van de Weerd et al., 2006; Fernandez et al., 1998).

However, since (W)CMS became a commodity and well-established implementations such as WordPress[1] or Drupal[2] exist for more than a decade now, research activities declined and new publications on CMS-related topics became scarce, with one of the few exceptions being studies comparing well-established CMS against each other (Martinez-Caro et al., 2018; Mirdha et al., 2014).

On the other hand, in recent years new concepts and technologies for building and running large-scale web applications emerged, such as server-side JavaScript using Node.js[3], microservice architectures and containerization, e.g. using Docker[4]. Despite this, most established CMS such as Wordpress or Drupal still use a traditional monolithic architecture, mostly extendable by plugins. The impact of these new technologies on the further evolution of CMS has not been addressed in the literature so far.

Therefore, in this paper a microservice architecture for an example e-business CMS is proposed, and a respective Proof-of-Concept implementation is described. The approach is evaluated with respect to its feasibility using a qualitative empirical study.

The rest of this paper is organized as follows: In section 2 the related work is reviewed in detail and the respective research gap is derived. Section 3 illustrates the proposed software architecture and section 4 its Proof-of-Concept implementation. The qualitative empirical evaluation of its feasibility and the obtained results are discussed in sections 5 and 6, respectively. We conclude with a summary of our findings.

[1]https://wordpress.org
[2]https://www.drupal.org
[3]https://nodejs.org/en/
[4]https://www.docker.com

Landerer, C. and Brune, P.
Future CMS for e-Business: Will Microservices and Containerization Change the Game?.
DOI: 10.5220/0008065002100217
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 210-217
ISBN: 978-989-758-386-5
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

2 RELATED WORK

There are few publications about the general impact of e-commerce applications on enterprises. Most of them are from the time when e-commerce was still in its childhood (Wu et al., 2003; Sanders, 2007), but few more recent studies exist on the impact of e-commerce applications in more specific domains (Benitez et al., 2017a; Benitez et al., 2017b). Privacy, trust and e-business adoption have also widely been studied (McKnight and Chervany, 2001; Nilashi et al., 2015), as they play a crucial role for e-commerce success. The same holds true for security of e-commerce applications (Kraft and Kakar, 2009; Niranjanamurthy and Chahar, 2013).

However, most research on e-commerce applications deals with the possibilities of value-creation in such systems, starting already at the beginning of the e-business era in the very early years of this century (Amit and Zott, 2001) until today (Shea et al., 2017).

Regarding web service and web application development in general, Service-oriented Architectures (SOA) have been discussed as a major archtectural paradigm for some time (Erl, 2005; Papazoglou, 2003; Al-Rawahi and Baghdadi, 2005). More recently, it evolved into the state-of-the-art approach of "Microservice Architectures", which are currenly frequently discussed in the literature (He and Yang, 2017; Messina et al., 2016; Khazaei et al., 2016), including their relation to SOA (Pautasso et al., 2017; Xiao et al., 2016). Specific topics such as performance and scalability of microservices compared to monolthic architectures (Villamizar et al., 2015) or the mapping of architectural challenges to microservice architectures (Alshuqayran et al., 2016; Pahl and Jamshidi, 2016) have been investigated as well.

Containerization is another architecture design pattern that is closely related to deploying microservices in the cloud. Containerization technologies like Docker[5] offer a high level of agility in "developing and running applications in cloud environment" (Kang et al., 2016), which has been discussed by various authors (Jaramillo et al., 2016; Kratzke, 2017).

With respect to the languages and platforms for implementing CMS and e-commerce applications, PHP[6] is still the dominating server-side programming language, with a total share of 82% in 2017[7]. Therefore, since its release in 1995, the development of web applications using PHP has been comprehensively studied (Cholakov, 2008; Cui et al., 2009). Since there are many more server-side programming languages, multiple studies exist comparing web development technologies. As PHP is still the predominating language in this area, most of these studies include PHP as one benchmark language (Suzumura et al., 2008; Trent et al., 2008; Purer, 2009).

For the development of the web frontends of CMS and other applications, Angular and React.js are the two most popular technologies at the moment (Shahzad, 2017; Aggarwal, 2018).

Angular is the strongly redesigned successor to the well-known JavaScript framework Angular.js. There is plenty of literature about Angular.js, the precursor of Angular (Jain et al., 2015; Jadhav et al., 2015). However, while Angular is rather different from its predecissor, studies about the use of the new Angular framework since version 2+ are scarce.

On the other hand, in recent years also server-side development using JavaScript with Node.js is becoming increasingly popular (Tilkov and Vinoski, 2010; Bermudez-Ortega et al., 2015; Madsen et al., 2015; Chaniotis et al., 2015; Lei et al., 2014). Comparing Node.js to PHP, according to some studies Node.js provides better performance and scalability due to its event-driven architecture and Javascript's asynchronous request-handling (McCune, 2011).

In contrast, a big advantage of PHP is its broad support by powerful frameworks and its use by popular CMS. Most of the currently used and predominating CMS like Wordpress or Drupal as well as most of their plugins are written in PHP. Therefore, there are some studies about correlations between the language and the CMS functionalities or issues (Eshkevari et al., 2014; Koskinen et al., 2012). Overall, a large body of literature about PHP and CMS exists, while literature about CMS in combination with Node.js is rather scarce.

In addition, the software architecture of the traditional PHP-based CMS systems like WordPress or Drupal is rather monolithic. Nearly no research exists so far on using microservice architectures for CMS development. Only one paper addresses the implementation of a collaborative rich text editor, which is most closely to the functions provided by a CMS (Gadea et al., 2016a).

On the other hand, there is a close connection between Node.js and microservice architectures. Most applications based on microservice architectures use at least one component that is running on Node.js on the server-side (Gadea et al., 2016b).

In the last years, CMS more and more became a commodity and not much has been added to the body of knowledge on CMS anymore, neither in research nor in practice. However, the described emerg-

[5]https://www.docker.com/
[6]http://php.net
[7]https://w3techs.com/technologies/history_overview/programming_language

ing technologies for building and running large-scale web applications, such as server-side JavaScript using Node.js, microservices and containerization, now suggest to review and re-think the implementation of CMS as well.

The dominating CMS implementations such as WordPress or Drupal currently still exhibit a monolithic software architecture, mostly extendable by plugins. This bears the permanent risk of version conflicts during updates due to compatibility problems, in particular if custom plugins are used. Using a microservice architecture for CMS presumable could help to overcome these problems, but has never been studied so far in the literature.

A special case of a CMS is one that is used for building e-business applications, which not only needs to serve static content or interactive components, but also has to provide the secure handling of the online shopping process.

Therefore, in this paper the question is addressed, how a prototypical microservice architecture for an e-business CMS could be designed and implemented using different modern web technologies, particularly for those functionalities where they might offer a specific advantage.

3 DESIGN OF THE SOFTWARE ARCHITECTURE

The functionalities of a prototypical e-business CMS could be grouped according to the three typical user groups: the administrator, the registered user and the website visitor.

The administrator should be able to change content on the system and furthermore manage the access control of other user groups. The website visitor should only see the actual website of the online shop where he can look around and search for products. Website visitors should have the possibility to register on its own and thereby becoming a registered user. Registered users can place orders and save personal data like delivery addresses after they are logged in.

Since the administrator should be able to change content on the website, there is the need of a separated area containing a dashboard, where only users with administrator rights can access and change data. In context of CMS, this area including the dashboard is called the backend of the CMS. Those main administration functions are: Manage offered products, manage users, manage additional static content sites and manage basic site settings.

Since the CMS is basically developed to create an e-business application, it should contain the typical important shop components by default. Thus, after articles are created by the administrator, customer(s) should instantly be able to order products in the online shop. This includes the process of searching, choosing and finally ordering a product. For this processes, the following components are obligatorily required: a product listing page, an article details page and a basket component.

In an online shop, there are always problems that can arise during the ordering process. Therefore, a customer support is needed to provide customers the possibility of directly asking questions to responsible staff members. In the application proposed in this paper, the customer support is implemented by a live chat in which registered users and website visitors can always request help. Therefore, a separated component in the admin area is needed, where staff members that have administrator rights can handle those requests.

To address these requirements, a microservice-based software architecture for the e-business CMS was designed, illustrated in figure 1.

On top of the application, a central Angular-based web frontend integrates the different microservices. All requests are made towards the Nginx reverse proxy server that distributes the request to the corresponding microservices.

Each microservice has its own, independent data store, which could be a NoSQL database, relational database, REST endpoint, SOAP endpoint, a websocket server or a combination of any or all of these data sources. All services provide a single endpoint API which handles all external communication from the client side as well as from other microservices.

This encapsulation allows to choose different technologies like databases or program languages for each microservice. To verify this, different technologies were used for the services of the prototype. Microservices or their components each run in individual Docker containers, e.g. allowing for an easy scale-out in production environments.

4 PROOF-OF-CONCEPT IMPLEMENTATION

To evaluate the proposed architecture, a prototype CMS application was implemented using the components described in the following.

4.1 Angular Frontend

Based on the analysis of different frontend technologies, Angular was selected as it seemed best suited

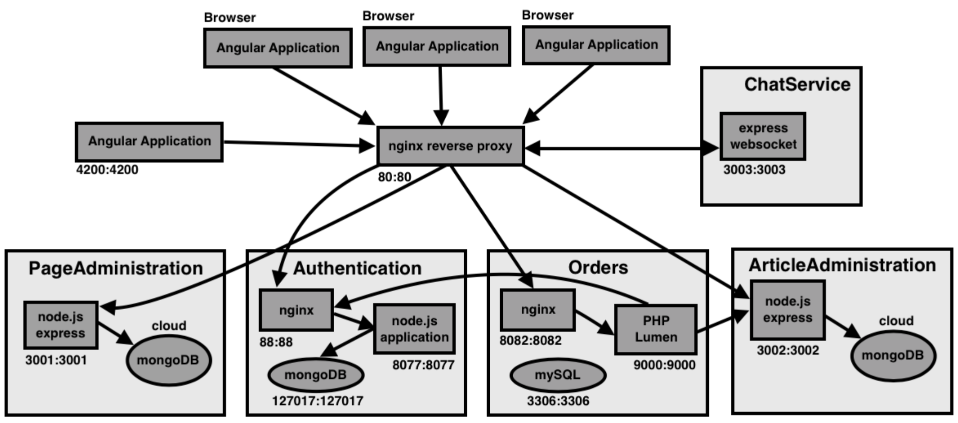

Figure 1: Proposed prototypical microservice architecture for an e-business CMS. Its different functionalities (page administration, article administration, authentication, etc.) are implemented as microservices made up of multiple Docker containers. The respective port mappings for Docker are shown for each container.

for the special case, as CMS require a high level of modularity. On the other hand, the user interface (UI) does not have to offer especially complex user interactions as customers are used to simple handlings in online shops and decline high complexity. Moreover, for special cases of online shops with large level of individual user interactions, typically a custom application is implemented instead of a CMS.

For routing between the pages of the application, the Angular Router is used[8]. As in CMS, it is highly important to clearly restrict accessibility, there is a need of inhibiting access of unauthorized users. This is realized through the possibility of Angular Router to create a guard class that can check if a route should be activated depending on a specific condition.

For this authorization, a token is stored in the cookies of the browser and every time authorization is needed, the token is sent to the authentication service to approve the authority of the user. If the admin authorization is valid, the dashboard components are loaded and the user can enter the admin area for changing content and adding data.

4.2 Article Microservice

The first microservice implemented was the article microservice. According to the previous analysis presented in section 2, Node.js was chosen for the server-side scripting language as passing article objects to the angular frontend leads to a low amount of parsing problems if both client and server-side are written in the same language. This is especially important as passed data can vary a lot depending on user's settings for one article type.

[8]https://angular.io/tutorial/toh-pt5

In this service, the database is cloud-hosted by a platform-as-a-service provider which fully manages MongoDB instances in the cloud. Cloud hosting was chosen for reasons of scalability as different web-shops can use a completely different number of products that are offered.

4.3 Order Microservice

For the order microservice, another software stack was used with technologies close to the well-established LAMP stack. Here, only the Apache webserver was replaced by the lightweight nginx. Nginx however has turned out to be the faster and more lightweight choice while the setup for an nginx server is a little harder, if there are any adaptions to be made. In the provided example, the image could be used unmodified directly from a Docker image.

The main reason for choosing a relational database was the decision to use a PHP framework which is aligned to SQL databases. The selected framework, Lumen, is a fast and lightweight micro-framework that is built on components of the popular PHP framework Laravel.

For creating an order, the user must be registered and logged in. The token sent to the user after login is passed to the order controller for checking the identity of the user again on the server side.

4.4 Authentication Microservice

Authentication is a main requirement in the context of a CMS, as different user roles contain different access rights.

In an environment of microservices, it is implicitly necessary to provide an authentication method that guarantees user authentication in every independent service as it is not possible to store the user data on the backend side and request it while needed.

For secure requesting and passing data belonging to a specific user over an API, typically token-based authentication is used. This kind of authentication method allows third-party applications and other services to access user-specific data with minimum risk of security breaches. Therefore, an Authentication Service was needed for generating, returning and validating those tokens after user login.

For the authentication service, a solution from DockerHub was selected, which provided a ready-to-run microservice that fully implemented token-based authentication. It consists of three different docker containers that constitute a stack consisting of a nginx server, a Node.js backend and a Mongo database.

4.5 Additional Services vs. Plugins

In a common CMS architecture, additional functionalities are accomplished by adding plugins to the monolith. They run quite independent from the core system as communication also works over predefined APIs, but they are still closer connected to the core system then microservices. Plugins can not be run or deployed independently, there is a need of a main CMS instance handling the processes.

In contrast, for adding features in a microservice architecture environment, microservices would be added instead of plugins, which run independently and can be changed or substituted at any time. This reduces compatibility problems as it decreases the risk of version intolerance if a the system is updated.

5 EVALUATION

To analyze the feasibility of the proposed approach, a qualitative empirical study was performed. The purpose of this study was twofold: First to evaluate the CMS functionality of the prototype from a user's perspective, and second to evaluate the underlying microservice-based software architecture. A convenience sample of seven persons was selected for the study, as listed in table 1. According to their respective skills, interviewees 1 to 5 were selected to evaluate the actual CMS functionality, and interviewees 6 and 7 as experts to evaluate the software architecture.All interviews were audio-recorded for subsequent analysis.

To evaluate the CMS functionality, interviewees 1 to 5 were asked to first perform some given tasks within the prototype and afterwards answer some questions about their experiences.

Table 1: Interviewees for the qualitative evaluation.

Interviewee 1	no precognitions with CMS, Student
Interviewee 2	experienced with CMS, Student
Interviewee 3	experienced with CMS, working in a marketing department
Interviewee 4	some precognitions in CMS, Student
Interviewee 5	some precognitions in CMS, Student
Interviewee 6	Head of software development in a smaller company
Interviewee 7	Software developer for some years

A test case was handed out to the participants, in which they were asked to create a fully-implemented online shop using the prototype CMS and afterwards using it from the customers' point of view. The purpose of this test was not to evaluate the user experience design, but the mere feasibility to perform the task using the given systems with its microservice architecture. The participants were observed and asked to "think aloud" during performing the tasks, which was audio-recorded for analysis.

For evaluating the software architecture of the provided prototype, two qualitative interviews were conducted with two experts with long-term experience in the field of software development and architecture (interviewees 6 and 7).

6 RESULTS

6.1 Evaluation of the CMS Functionality

First, the test persons were asked to enter the administrative area with the provided admin login credentials and customize the site towards the provided companies' details, including a modification of the design and the site name. No obvious problems occurred during this task, and every user directly navigated to the settings section, where they could change both.

The second step was to create a static content page that provides additional contact details of the company. Users seemed to be positively surprised from the possibility of supervising the entered details directly and in real-time in a section next to the creation area. In the subsequent interview, some of the test persons mentioned this feature as especially useful (Interviewee 1: "It was really useful, that there was a preview of my content site directly while entering the data", Interviewee 4: "It is quite simple to create an additional content page if you have a direct preview").

The next task was to create a new article for the online shop employing article details that were provided in the task sheet. There were no problems that occurred during this process , the workflow was even mentioned as a positive feature of the CMS (Interviewee 4: "Creating articles was very intuitive and easy", Interviewee 2: "Creating articles was easy and self-explanatory.", Interviewee 1: "There was no problem creating articles, except the fact that it should be explained that there is no need of adding units to the price").

There was a moment of irritation during every test. Participants did not realize if the article was already saved as they did not receive a confirmation message (Interviewee 5: "It was not clear if the article is already saved", Interviewee 1: "It was not possible to see if the content of the article is saved", Interviewee 4: "Sometimes I didn't recognize if the content was already saved"). This issue is also easily avoidable and even quite easy to implement within the prototype.

The last task was to register as a new customer and login to check the online shop's functionality from the customer's point-of view by ordering an article. No issues occured during this process.

6.2 Evaluation of the Software Architecture

First, the architecture of the prototype was presented and the experts were asked to describe the differences from a common CMS like Wordpress. One expert claimed, that microservice architecture is already self-explanatory. Hence, "the modularity is much higher than in common CMS like Wordpress. The code is more independent and therefore it is structured clearer than in a common system because many components are stuck together to one system" (Interviewee 6).

The first important positive characteristic mentioned is modularity and scalability. There is the big advantage of easily "substituting or replacing parts of the system" (Interviewee 6). Also, the opportunity is mentioned that it is possible to complementary deploy a second service, which adds further functionalities to the first one "without the need of rewriting a broad part of the code" (Interviewee 7).

Another advantage noted is the opportunity of better load distribution. In an architecture that is based on a monolith, the whole application must be on one single physical server while it is easily possible to spread the services of a microservice to more physical servers and therefore reach a better load balance (Interviewee 7). The possibility "of putting the services to the cloud" (Interviewee 6) is also highly rated, despite issues of security mentioned at this point as well as possible latency problems.

If the microservice architecture affects the security of an application, depends in general on the setup of the services and the application. "If they are still running on one physical server and all microservices and data are called internally, there is no need of further encryptions than in common systems" (Interviewee 6).

If using different technologies and platforms in different microservices is useful, will depend on the companies and applications size and hard to evaluate. The advantage of programming all code in the same language is staff flexibility as team members can "easily switch between the single services or teams and directly get along with the structure and the code" (Interviewee 7).

On the other hand, switching between technologies sometimes can be "a requirement because technologies are too slow or make things extremely complicated at one part of the code" (Interviewee 6). In this case, the big advantage is that microservices provide the possibility of switching to a new technology without changing all the rest of the code. In this case, using different technologies highly affects performance in a positive way (Interviewee 6).

The question if microservices would affect the daily workflow in a department was clearly confirmed. "It is possible to build teams for every microservice and independently chose technologies within this unit." (Interviewee 7).

It was also mentioned, that in a big project code "is getting complex very fast" (Interviewee 6). In this context, the advantage for new staff members is seen of getting along faster in those "smaller code parts than in a big monolith" (Interviewee 6).

In context of CMS, scalability is a very important topic as extending existing functions adds customized additional functionalities to the system. This depends on the size and the kind of application that is embedded. Using microservices brings a "little more overhead in implementation in the beginning as it needs to be decided about the setup of the API and error handling" (Interviewee 6).

On the other hand, one expert claims that this overhead in contrast is "making the code clearer and cleaner" (Interviewee 7). Programming towards an

API is much cleaner than "getting into additional dependencies with plugins that are using the same databases or services" (Interviewee 7).

All test persons agreed on an overall positive experience of creating an online shop with the provided CMS.

7 CONCLUSION

In conclusion, in this paper a microservice architecture for a CMS for e-business applications was proposed and evaluated. Starting from typical requirements for e-business-applications, the architecture was designed and implemented by Proof-of-Concept implementation using state-of-the-art technologies like Node.js, Angular and Docker. This prototype then was evaluated by means of a qualitative empirical study.

The results indicate that a microservice-based architecture could be beneficial for e-business CMS as well, allowing to build better maintainable applications without exhibiting any drawbacks with respect to functionality.

Further research is needed to extend the prototype to a full-fledged CMS application and evaluate it in more comprehensive lab and field tests.

REFERENCES

Aggarwal, S. (2018). Modern web-development using reactjs.

Al-Rawahi, N. and Baghdadi, Y. (2005). Approaches to identify and develop web services as instance of soa architecture. In *Services Systems and Services Management, 2005. Proceedings of ICSSSM'05. 2005 International Conference on*, volume 1, pages 579–584. IEEE.

Alshuqayran, N., Ali, N., and Evans, R. (2016). A systematic mapping study in microservice architecture. In *Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on*, pages 44–51. IEEE.

Amit, R. and Zott, C. (2001). Value creation in e-business. *Strategic management journal*, 22(6-7):493–520.

Benitez, J., Chen, Y., Teo, T. S., and Ajamieh, A. (2017a). Evolution of the impact of e-business technology on operational competence and firm profitability: A panel data investigation. *Information & Management*.

Benitez, J., Chen, Y., Teo, T. S., and Ajamieh, A. (2017b). Impact of e-business technology on operational competence and firm profitability over time.

Bermudez-Ortega, J., Besada-Portas, E., López-Orozco, J., Bonache-Seco, J., and De la Cruz, J. (2015). Remote web-based control laboratory for mobile devices based on ejss, raspberry pi and node. js. *IFAC-PapersOnLine*, 48(29):158–163.

Chaniotis, I. K., Kyriakou, K.-I. D., and Tselikas, N. D. (2015). Is node. js a viable option for building modern web applications? a performance evaluation study. *Computing*, 97(10):1023–1044.

Cholakov, N. (2008). On some drawbacks of the php platform. In *Proceedings of the 9th international conference on computer systems and technologies and workshop for phd students in computing*, page 12. ACM.

Cui, W., Huang, L., Liang, L., and Li, J. (2009). The research of php development framework based on mvc pattern. In *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pages 947–949. IEEE.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, And Design*. Pearson Education India.

Eshkevari, L., Antoniol, G., Cordy, J. R., and Di Penta, M. (2014). Identifying and locating interference issues in php applications: the case of wordpress. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 157–167. ACM.

Fernandez, M., Florescu, D., Kang, J., Levy, A., and Suciu, D. (1998). Catching the boat with strudel: Experiences with a web-site management system. In *ACM SIGMOD Record*, volume 27, pages 414–425. ACM.

Gadea, C., Trifan, M., Ionescu, D., Cordea, M., and Ionescu, B. (2016a). A microservices architecture for collaborative document editing enhanced with face recognition. In *Applied Computational Intelligence and Informatics (SACI), 2016 IEEE 11th International Symposium on*, pages 441–446. IEEE.

Gadea, C., Trifan, M., Ionescu, D., and Ionescu, B. (2016b). A reference architecture for real-time microservice api consumption. In *Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms*, page 2. ACM.

Gartner (2017). Magic quadrant for web content management. Report, Gartner.

He, X. and Yang, X. (2017). Authentication and authorization of end user in microservice architecture. In *Journal of Physics: Conference Series*, volume 910, page 012060. IOP Publishing.

Jadhav, M. A., Sawant, B. R., and Deshmukh, A. (2015). Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, 6(3):2876–2879.

Jain, N., Bhansali, A., and Mehta, D. (2015). Angularjs: A modern mvc framework in javascript. *Journal of Global Research in Computer Science*, 5(12):17–23.

Jaramillo, D., Nguyen, D. V., and Smart, R. (2016). Leveraging microservices architecture by using docker technology. In *SoutheastCon, 2016*, pages 1–5. IEEE.

Kang, H., Le, M., and Tao, S. (2016). Container and microservice driven design for cloud infrastructure devops. In *Cloud Engineering (IC2E), 2016 IEEE International Conference on*, pages 202–211. IEEE.

Khazaei, H., Barna, C., Beigi-Mohammadi, N., and Litoiu, M. (2016). Efficiency analysis of provisioning microservices. In *2016 IEEE International Conference* *on Cloud Computing Technology and Science (CloudCom)*, pages 261–268. IEEE.

Koskinen, T., Ihantola, P., and Karavirta, V. (2012). Quality of wordpress plug-ins: an overview of security and user ratings. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 834–837. IEEE.

Kraft, T. A. and Kakar, R. (2009). E-commerce security. In *Proceedings of the Conference on Information Systems Applied Research, Washington DC, USA*.

Kratzke, N. (2017). About microservices, containers and their underestimated impact on network performance. *arXiv preprint arXiv:1710.04049*.

Lei, K., Ma, Y., and Tan, Z. (2014). Performance comparison and evaluation of web development technologies in php, python, and node. js. In *2014 IEEE 17th International Conference on Computational Science and Engineering (CSE)*, pages 661–668. IEEE.

Madsen, M., Tip, F., and Lhoták, O. (2015). Static analysis of event-driven node. js javascript applications. In *ACM SIGPLAN Notices*, volume 50, pages 505–519. ACM.

Martinez-Caro, J.-M., Aledo-Hernandez, A.-J., Guillen-Perez, A., Sanchez-Iborra, R., and Cano, M.-D. (2018). A comparative study of web content management systems. *Information*, 9(2):27.

McCune, R. R. (2011). Node. js paradigms and benchmarks. *STRIEGEL, GRAD OS F*, 11:86.

McKeever, S. (2003). Understanding web content management systems: evolution, lifecycle and market. *Industrial management & data systems*, 103(9):686–692.

McKnight, D. H. and Chervany, N. L. (2001). What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology. *International journal of electronic commerce*, 6(2):35–59.

Messina, A., Rizzo, R., Storniolo, P., Tripiciano, M., and Urso, A. (2016). The database-is-the-service pattern for microservice architectures. In *International Conference on Information Technology in Bio-and Medical Informatics*, pages 223–233. Springer.

Mirdha, A., Jain, A., and Shah, K. (2014). Comparative analysis of open source content management systems. In *Computational Intelligence and Computing Research (ICCIC), 2014 IEEE International Conference on*, pages 1–4. IEEE.

Nilashi, M., Ibrahim, O., Mirabi, V. R., Ebrahimi, L., and Zare, M. (2015). The role of security, design and content factors on customer trust in mobile commerce. *Journal of Retailing and Consumer Services*, 26:57–69.

Niranjanamurthy, M. and Chahar, D. D. (2013). The study of e-commerce security issues and solutions. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(7).

Pahl, C. and Jamshidi, P. (2016). Microservices: A systematic mapping study. In *CLOSER (1)*, pages 137–146.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 3–12. IEEE.

Pautasso, C., Zimmermann, O., Amundsen, M., Lewis, J., and Josuttis, N. M. (2017). Microservices in practice, part 1: Reality check and service design. *IEEE Software*, 34(1):91–98.

Purer, K. (2009). Php vs. python vs. ruby–the web scripting language shootout. *Vienna University of Technology*.

Sanders, N. R. (2007). An empirical study of the impact of e-business technologies on organizational collaboration and performance. *Journal of Operations Management*, 25(6):1332–1347.

Shahzad, F. (2017). Modern and responsive mobile-enabled web applications. *Procedia Computer Science*, 110:410–415.

Shea, V. J., Dow, K. E., Chong, A. Y.-L., and Ngai, E. W. (2017). An examination of the long-term business value of investments in information technology. *Information Systems Frontiers*, pages 1–15.

Suzumura, T., Trent, S., Tatsubori, M., Tozawa, A., and Onodera, T. (2008). Performance comparison of web service engines in php, java and c. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 385–392. IEEE.

Tilkov, S. and Vinoski, S. (2010). Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83.

Trent, S., Tatsubori, M., Suzumura, T., Tozawa, A., and Onodera, T. (2008). Performance comparison of php and jsp as server-side scripting languages. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 164–182. Springer-Verlag New York, Inc.

van de Weerd, I., Brinkkemper, S., Souer, J., and Versendaal, J. (2006). A situational implementation method for web-based content management system-applications: method engineering and validation in practice. *Software process: improvement and practice*, 11(5):521–538.

Vidgen, R., Goodwin, S., and Barnes, S. (2001). Web content management. In *Proceedings of the 14th International Electronic Commerce Conference*, pages 465–480.

Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., and Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *Computing Colombian Conference (10CCC), 2015 10th*, pages 583–590. IEEE.

Wu, F., Mahajan, V., and Balasubramanian, S. (2003). An analysis of e-business adoption and its impact on business performance. *Journal of the Academy of Marketing science*, 31(4):425–447.

Xiao, Z., Wijegunaratne, I., and Qiang, X. (2016). Reflections on soa and microservices. In *2016 4th International Conference on Enterprise Systems (ES)*, pages 60–67. IEEE.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS