http://doi.org/10.35784/iapgos.4279 *received: 20.07.2023 | revised: 29.08.2023 | accepted: 27.11.2023 | available online: 20.12.2023*

THE EFFICIENCY AND RELIABILITY OF BACKEND TECHNOLOGIES: EXPRESS, DJANGO, AND SPRING BOOT

Dominik Choma, Kinga Chwaleba, Mariusz Dzieńkowski
Lublin University of Technology, Department of Computer Science, Lublin, Poland

Abstract. *Increasing popularity of web applications has led to the development of many technologies that enable their production, both on the client and server side. This article attempts to compare three most popular server-side frameworks – Django, Spring Boot and Express. Each of the selected technologies is based on a different programming language. These frameworks were compared in terms of request processing time and reliability. Within the conducted research three backend applications handling HTTP requests were created, all of them using the same database consisting of employees' data. Afterwards, a series of load tests was performed to determine levels of efficiency and reliability of created applications for various numbers of virtual users sending requests to the server at the same time. Five test cases with the following number of requests: 1000, 2000, 4000, 8000, and 16000 were planned and performed for each type of HTTP requests handled by the server simultaneously. Based on the obtained results, it was concluded that the Spring Boot framework was the best in terms of request processing time and high reliability. However, it was noted that for many test cases under extreme load, it had a significantly higher percentage of incorrectly processed requests compared to the Express application, even though the application was noticeably slower. The worst results were observed for Django because the test application created for this framework revealed the longest requests processing time and the highest error rate during processing requests out of the three tested applications. The performed studies helped to determine the efficiency and reliability of the tested technologies at various levels of load. Furthermore, the studies were crucial in obtaining knowledge about the evaluated frameworks as well as their properties and formulating conclusions that will be able to help the developers choose technologies before the implementation of their programming projects.*

Keywords: efficiency, reliability, request processing time, Spring Boot, Express, Django

WYDAJNOŚĆ I NIEZAWODNOŚĆ TECHNOLOGII WYTWARZANIA APLIKACJI INTERNETOWYCH STRONY SERWERA: EXPRESS, DJANGO ORAZ SPRING BOOT

Streszczenie. *Wzrastająca popularność aplikacji internetowych doprowadziła do powstania wielu technologii umożliwiających ich wytwarzanie, zarówno po stronie klienta jak i serwera. W niniejszym artykule podjęto się dokonania porównania trzech najbardziej popularnych szkieletów programistycznych strony serwera – Django, Spring Boot, Express. Każda z wybranych technologii opiera się na innym języku programowania. Szkielety zostały porównane pod względem czasu obsługi żądań i niezawodności. W ramach przeprowadzonych badań utworzono trzy serwerowe aplikacje testowe realizujące obsługę żądań HTTP i wykorzystujące tę samą bazę danych, zawierającą dane pracowników. Następnie wykonano serię testów obciążeniowych pozwalających określić wydajność i niezawodność napisanych aplikacji dla różnych liczb wirtualnych użytkowników wysyłających żądania do aplikacji w tym samym momencie. Zaplanowano scenariusze testowe zakładające następujące liczby żądań: 1000, 2000, 4000, 8000 oraz 16000, wykonanych dla każdego z obsługiwanych przez aplikacje testowe typów żądań HTTP. Na podstawie otrzymanych wyników wywnioskowano, że szkielet programistyczny Spring Boot cechuje się najwyższą prędkością wykonywania żądań oraz wysoką niezawodnością. Jednak zauważono także, że dla wielu przypadków testowych przy ekstremalnym obciążeniu miał on wyraźnie wyższy odsetek błędnie obsłużonych żądań w porównaniu z aplikacją utworzoną na bazie szkieletu Express, pomimo że ta była znacznie wolniejsza. Najsłabsze wyniki zaobserwowano dla Django, ponieważ aplikacja testowa opracowana na podstawie tego szkieletu uzyskała zarówno najdłuższe czasy, jak i najwyższy odsetek błędów podczas obsługi żądań spośród wszystkich trzech testowanych aplikacji. Wykonane badania pozwoliły określić wydajność oraz niezawodność przebadanych technologii przy różnych poziomach obciążenia, pozwoliły poznać działanie i właściwości testowanych szkieletów oraz sformułować wnioski, które mogą pomóc deweloperom w doborze technologii przed realizacją ich projektów programistycznych.*

Słowa kluczowe: wydajność, niezawodność, czas obsługi żądań, Spring Boot, Express, Django

Introduction

The widespread availability of the Internet has resulted in the popularity of Internet applications, which facilitate the use of a variety of services directly through a web browser. This eliminates the need to install additional software or consume device hardware resources. Internet applications are software programs hosted on remote servers, and users can access them through a graphical user interface displayed in a browser window. Communication between the browser and web services is facilitated by programming interfaces, commonly referred to as Application Programming Interfaces (APIs). Web applications are dependent on sending requests and receiving responses using various protocols, including the Hypertext Transfer Protocol (HTTP). Over time, numerous programming languages have been developed to facilitate the creation of Internet applications. However, using a pure language would necessitate building all of the necessary functionalities from the ground up. This results in numerous issues and takes up a significant amount of time. Therefore, developers typically employ pre-existing solutions that provide tried-and-tested features that can improve both performance and security.

To improve the efficiency of software development, developers use frameworks that enhance performance of web applications and provide them with greater ease of use. A framework consists of a set of components offering various functions and capabilities to developers. These functions include database management, performing operations on databases, as well as authentication and authorization mechanisms that enhance the security of applications.

A common approach is to divide web applications into two parts: the client-side (frontend) and the server-side (backend). The client-side is responsible for sending requests to the server, receiving and processing responses, and presenting data to the user in an appropriate format. On the other hand, the server-side manages requests, communicates with the database, processes data, and generates responses. This separation allows for the parallel development of both parts of the application, which can reduce the overall software development time. When selecting a framework, it is important to consider the specific requirements of a project, as each of them provides slightly different solutions that can significantly impact parameters such as performance, reliability, maintainability, and portability. These factors contribute to the overall quality of the final product.

The authors of this paper researched the performance and reliability of the most widely used server-side frameworks for JavaScript, Python, and Java. These tests aimed to determine which of the tested frameworks would be the most suitable choice under specific test conditions.

1. Literature overview

Currently web applications have attained a high degree of prevalence, which is associated with the multitude of available server-side tools utilized for their development. The selection of the most suitable technology that satisfies the requirements

of software is one of the primary decisions that must be taken to create a fully functional web application. This process is closely connected with the choice of a suitable programming language and framework. There are numerous scholarly articles addressing the topic of choosing optimal tools for the development of web applications with particular levels of complexity and scope of operations.

A crucial part of the web application development process is the design and implementation of its server-side logic. This subject was considered in the article [5]. The author describes this process in many aspects, including an examination of web applications within the contexts of both static and dynamic websites, the utilization of programming frameworks, and their helpfulness in accessing databases. Then, a more detailed explanation is provided regarding programming frameworks, which includes the characteristics of Express, Spring Boot, and Django and the comparison of their speed in executing a single statement to the database. What is more, the significance of the correct database design is mentioned in the conclusions.

In a subsequent publication, referenced as [2], the authors conducted a comprehensive comparison of four frameworks (Laravel, Ruby On Rails, Django, Spring) based on a three-point scale established by them. The assessment covered various aspects of each framework, including code generators, popularity, business trends, integration with additional software, and plug-in support. According to the authors, Spring received the highest score among the evaluated frameworks, primarily due to its business trends and popularity among programmers. The authors also rate criteria such as scalability, entry threshold for novice programmers, and popularity on specific internet platforms (Haker News, Google Trend, Reddit, GitHub, StackOverflow). In these criteria, Spring ranked third, behind Laravel and Django. The authors concluded that Django was the best framework among those analyzed, citing its ease of use for novice programmers and its adaptability for large web applications.

Articles [1, 4, 6] compare currently popular web application development technologies to identify their advantages and disadvantages. The analysis was carried out based on prepared test applications implementing CRUD (Create, Read, Update, Delete) functionalities. Almost all applications [1, 4] were connected to the database. Only in [6] the database was not used for fear of a possible slowdown in the response and interference in the measurements. The main examined parameter of the performed operations was efficiency. In the case of applications using a database, GET, POST, PUT, and DELETE requests were considered. On the other hand, in the article [5], the bandwidth and its impact on Internet applications and computer resources used were also examined. Article [4] additionally presented the assessment of authentication and authorization, where individual aspects were assessed using a point system. Articles [1, 6] presented the use of JMeter software to simulate virtual users utilizing the created applications.

The comparison of the performance of popular frameworks has been also the subject of scholarly articles [3, 7]. In both publications, two test applications were implemented for the surveyed frameworks, enabling a connection with the database. Furthermore, the Apache JMeter tool was utilized for conducting the tests, as previously mentioned. The first article described the evaluation of the efficiency of the GET, POST, PUT, and DELETE statements for the REST application. In the second article, the efficiency was surveyed using GraphQL as well. The CPU and memory usage was established, and in addition, the first article's tests were conducted for different loads defined by the number of users (1, 8, 64, 128, 256, 512, 1,024). To obtain reliable results, each test was carried out 10 times. The second article presented the parallel execution of requests (100, 250, 500) with varying numbers of rows (1, 50, 100). The tests executed in both articles allowed for the efficiency comparison of frameworks applied for developing the server-side of web applications.

Choosing a technology sufficiently advanced for this application can help in the performance not only for a beginner but also for an experienced programmer. In addition, it is essential to consider the number of elements that are required to combine various programming frameworks and to calculate them for multiple applications. In this study, the authors decided to compile a list of the most popular technologies currently used for web applications on the server side. This was achieved by using test applications on these frameworks and then compiling the results. It is worth noting that this research stands out due to the selection of other tested technologies, their respective versions, and test cases.

2. Aim, hypotheses, scope of work

The aim of this work is to perform a comparative analysis of backend frameworks: Express, Spring Boot, and Django, in their latest stable versions at the time of conducting the research. The efficiency of three test applications, which have been created based on selected backend frameworks, will be compared according to the applied load, depending on the number of requests sent.

The following research hypotheses have been formulated:

1) Express, due to working in the Node ecosystem, which was designed in order to optimize efficiency and scalability of the web applications, is distinguished by better efficiency compared to Spring Boot and Django in the case of a significant number of requests sent to the server.
2) Spring Boot, by utilizing configuration based on annotations, is characterized by the best efficiency compared to Express and Django in the case of a limited number of requests.
3) Django, due to the significant usage of network bandwidth, is marked by the worst efficiency compared to Express and Spring Boot regardless of the number of requests sent.

3. Used technologies and tools

To compare the chosen programming frameworks, identical applications have been developed using technologies such as Spring, Express, and Django based on a REST architectural style. The established test applications consist of the same functionalities. A tool Apache JMeter has been utilized to simulate requests sent from the client-side to the server applications and to measure response time to these requests.

3.1. REST

REST [13] (REpresentational State Transfer) is an architectural style that defines the way web applications are created to be smoothly usable and user-friendly. REST is an implementation of this architecture that uses HTTP protocol to perform operations on resources and returns data in formats such as JSON or XML, enabling their seamless utilization by client-side applications.

3.2. Apache JMeter

JMeter [8] is an open source software specifically designed for testing the efficiency of an application. It enables conducting tests that simulate user traffic and measure the efficiency of a tested system under load. Test cases and parameters can be adjusted based on the user's preferences. What is more, JMeter provides various tools for analyzing the obtained test results such as generated charts or reports. It is stated to be a popular choice for testing multiple protocols, including HTTP. Moreover, it could simulate virtual users who use the tested system. Weaknesses of the system can be identified, which is a crucial step in taking measures to improve the application’s performance.

3.3. Compared frameworks

After analyzing the popularity of the available frameworks on the market it can be observed that Express, Spring Boot, and Django belong to the top-tier technologies used in backend application development [11]. The comparison was conducted by

examining stars given to the official repositories of the surveyed frameworks by GitHub users [10], and votes received in Stack Overflow's annual summary of the programming market for 2022 [12]. This comparison is presented in table 1. It can be noticed that the discussed frameworks are similarly popular. Django is the most popular framework based on data obtained from GitHub, while Express is the most recognized according to the Stack Overflow survey.

Table 1. A comparison of the popularity of the surveyed frameworks based on data obtained from GitHub and Stack Overflow

Framework/Service	GitHub (stars)	Stack Overflow (received votes)
Express	59,600	22.99 %
Spring Boot	65,100	16.13%
Django	68,200	14.65 %

4. Research methodology

Created applications built on the REST architecture were deployed in a designated testing environment. They implement the basic CRUD methods using the same database. The implemented functionalities were tested with varying numbers of user requests sent within a one-second interval, in order to examine the correlation between the number of requests made and response time.

4.1. Test environment

The research was conducted using a computer with Windows 10 operating system installed. Table 2 presents the parameters that are crucial from the perspective of the performed research. On the other hand, Table 3 provides information about the versions of the frameworks used in the research, which were the latest stable versions available at the time, along with their corresponding programming languages.

Table 2. Parameters of the computer utilized in the research

Parameters	Device
Processor	Intel Core i5-10210U
RAM memory	16 GB
Operating system	Windows 10

Table 3. A characteristic of the tested frameworks

Framework	Version	Language	Version
Express	4.18.2	JavaScript	ES6
Spring Boot	3.0.2	Java	17
Django	4.1.6	Python	3.11

4.2. Test cases

A comparative analysis of the backend application development technologies - Express, Spring Boot, and Django – was conducted using the Apache JMeter tool. The test cases were planned based on common HTTP requests sent by the client to the server. A comparison was performed for the following test cases:

1) the measurement of the execution time of a GET request,
2) the measurement of the execution time of a POST request,
3) the measurement of the execution time of a PUT request,
4) the measurement of the execution time of a DELETE request.

The goal of that research is to examine how the application works in variable testing conditions - using various loads. According to the analysis of the literature, it was decided to undertake test cases for the following cases:

1) sending 1 request by 1,000 users at the same time,
2) sending 1 request by 2,000 users at the same time,
3) sending 1 request by 4,000 users at the same time,
4) sending 1 request by 8,000 users at the same time,
5) sending 1 request by 16,000 users at the same time.

Apart from efficiency, the reliability of an application is known to be a crucial characteristic. It was also the subject of the research in this study.

4.3. Test applications

The created test applications were based on a fragment of an open-source MySQL database called Employees [9], which consists of approximately 4 million records. The diagram representing a part of the database containing information about employees and their salaries is presented in figure 1. The employees table contains employee data, and the salary table is related to it through a one-to-many relation which comprises information about employee salary.

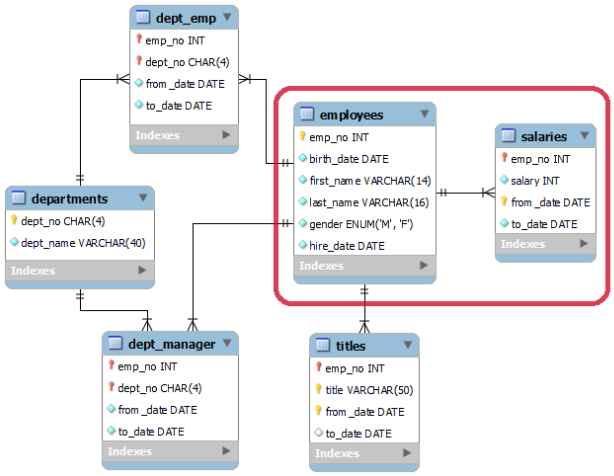


Fig. 1. The diagram presenting a database with a selected fragment, which the application utilizes

The GET request implemented in the study retrieves employee data using the identification number specified in the path. The request returns all details pertaining to the selected employee, along with a list comprising information on their remuneration. Conversely, the POST request is responsible for adding a new employee, achieved by sending an appropriate JSON object containing the user's data in the request's body. The PUT request, on the other hand, modifies the employee data associated with the employee ID specified in the path, replacing it with the data provided in the sent JSON object. Finally, the DELETE request deletes the employee's data with the indicated ID, along with all information concerning their earnings.

During the application testing, it was observed that certain development frameworks implement mechanisms that boost application performance by default, with Spring Boot featuring the largest number of such solutions. Conversely, Express lacks pre-implemented mechanisms of this nature, and their use requires additional programming and configuration efforts. To enhance the efficiency of the test application based on Express, a clustering mechanism was developed, allowing for the launch of multiple application instances within a single process. This mechanism improved overall performance while simultaneously reducing resource consumption.

5. Results analysis

The conducted load tests of the three implemented applications allowed for an analysis of the surveyed frameworks in terms of both the speed of request execution and determining their reliability.

5.1. GET request

The results for the GET request are pictured in figure 2, with a horizontal axis representing the test load and a vertical axis representing the average request time.

In this case for each of the conducted test cases, the application written using Spring Boot was characterized by the best request execution time. The Express framework application was distinguished by its worse efficiency, while the request execution times for the subsequent test cases are

significantly longer compared to those of the applications developed using Spring Boot. The least favorable results in terms of efficiency were obtained from the Django application. The operation execution times are approximately twice the times achieved for the application implemented in Express. Despite the number of requests, the average request execution time ranged between 2,800 and 4,100 ms.

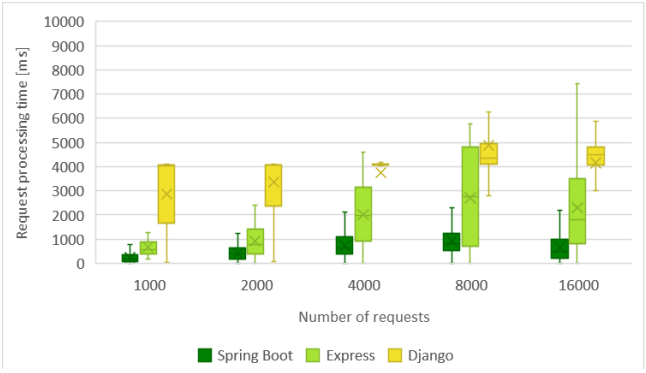


Fig. 2. The GET request processing time depending on the number of requests for each of the tested frameworks

Figure 3 depicts the reliability tested for the GET request, with the vertical axis representing the percentage of wrong requests.

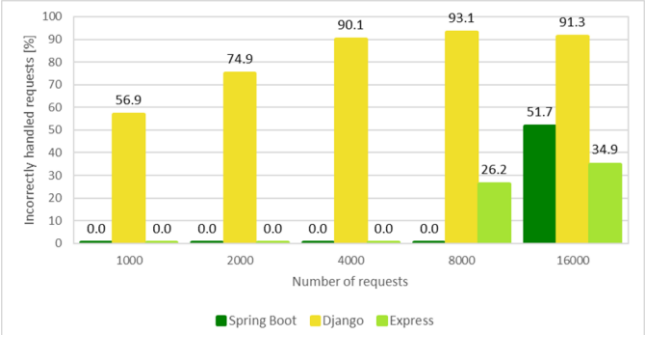


Fig. 3. The percentage of incorrectly handled GET requests by the application based on a given framework

The test application developed using the Spring Boot framework exhibited a high level of reliability, as errors only appeared in the last test case (Fig. 3). For the Express frameworks, errors were noticed for 8,000 users sending requests simultaneously, with approximately 26.23% of requests failing. However, for the server loaded with requests from 16,000 users, the number of unsuccessful requests was lower for this framework than for the Spring Boot framework (34.86% for Express and 51.65% for Spring Boot). On the other hand, the application created with Django was identified by the error occurring at the stage where fewer virtual users were simulated compared to the previous two applications.

5.2. POST request

The results obtained for the GET and POST request were similar. For the evaluated handling of the POST request, the outputs were presented in Figure 4, where the horizontal axis represents the test load, and the vertical axis shows the average time of one request.

It has been acknowledged that the test application based on the Spring Boot framework exhibited the best request execution time. On the other hand, the application utilizing Express demonstrated slightly worse reliability. Although the outcomes marginally varied for an insignificant number of requests sent simultaneously (1,000-2,000), the discrepancy considerably increased for a greater number of virtual users. Among the three test applications, the one based on the Django framework achieved the least favorable result. Execution times for all test cases were markedly higher and substantially deviated from the outputs acquired for the remaining two applications – the obtained times ranged between 3,000-5,000 ms.

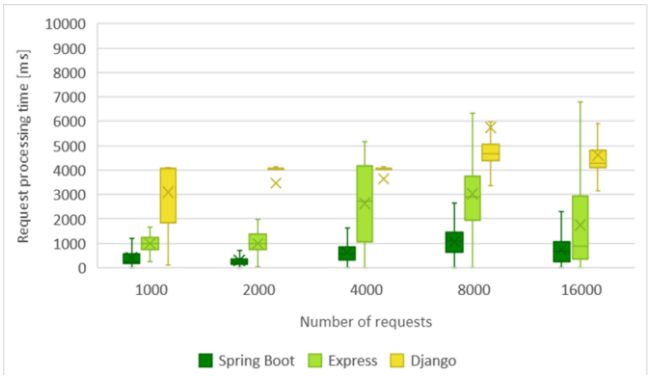


Fig. 4. The POST request processing time depending on the number of requests for each of the tested frameworks

The tested reliability of the POST request was demonstrated in figure 5, with the vertical axis containing information on the percentage of the incorrect requests.

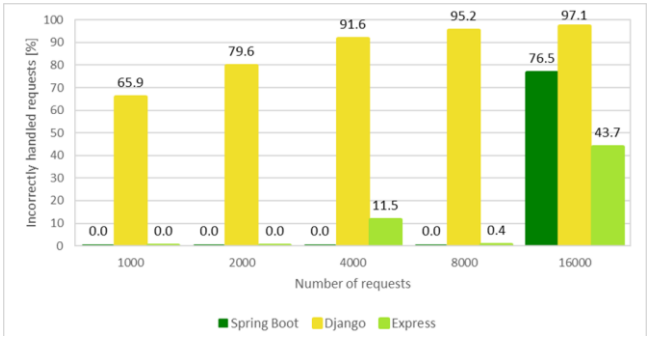


Fig. 5. The percentage of incorrectly handled POST requests by the application based on a given framework

In the conducted tests, the test application utilizing the Django framework exhibited the highest number of failed requests. Specifically, the initial test resulted in a high error rate of 65.9%. Moreover, subsequent tests showed a further increase in the error rate, reaching 97% with 16,000 virtual users. In contrast, the Spring Boot framework exhibited no errors when the server was loaded with 1,000, 2,000, 4,000, and 8,000 virtual users sending requests. Correspondingly, the Express framework demonstrated a zero-error rate for 1,000 and 2,000 requests sent concurrently. An abrupt increase in the number of failed requests was observed during the load simulation involving 16,000 users, as 76% of requests failed for the Spring Boot framework test case. Regarding reliability, Express maintained a zero percent error rate, with a small deviation (11.45% of unsuccessful requests) detected for the test case handling 4,000 requests. Nevertheless, this value decreased to 0.38% for 8,000 requests. For the highest load, 43.69% of requests, sent by 16,000 users at once, were found to be ineffective. Notably, this error rate was lower than that noticed for the application based on the Spring Boot framework under the same load.

5.3. PUT request

Figure 6 shows the average request handling time depending on the number of PUT requests sent. The horizontal axis represents the load levels subjected to experimentation, while the vertical axis corresponds to the average time duration of a single request.

In this instance, it is evident that the Spring Boot framework delivered the most favorable performance outcomes. Nevertheless, it is worth mentioning that when subjected to a server currently handling requests from a thousand virtual users, the Express framework exhibited a noticeably superior processing speed. For other load levels, the average duration of request execution remained stable within the range of 800 to 2,000 ms. Conversely,

Django demonstrated considerably longer request handling times (with the exception of a scenario involving simultaneous requests from 8,000 users), spanning from 3,500 to 4,500 ms.

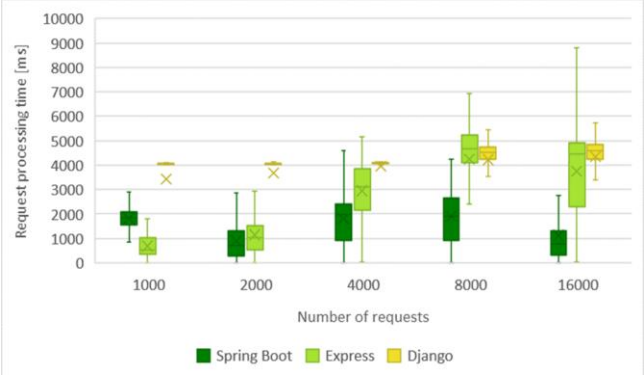


Fig. 6. The PUT request processing time depending on the number of requests for each of the tested frameworks

Figure 6 depicts the reliability data acquired for the PUT request, with the vertical axis displaying information about the percentage of requests that failed. The horizontal axis denotes the number of virtual users whose requests were handled by the server.

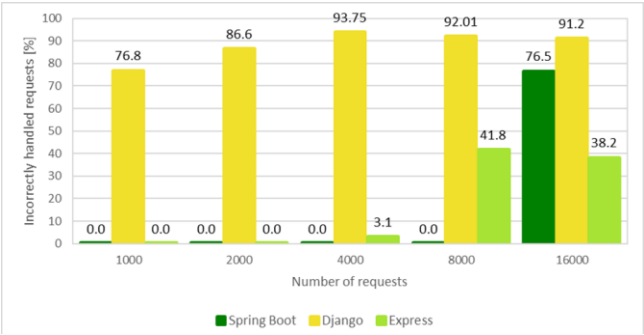


Fig. 7. The percentage of incorrectly handled PUT requests by the application based on a given framework

During the testing, the Spring Boot framework recorded zero failed requests. However, the error rate increased significantly when the server was loaded with 16,000 simultaneous requesting users, with a rate of over 75%, which was considerably higher than that of the Express framework. The Express framework consistently demonstrated average performance with respect to average error rate scores. The only exception was in a case involving 1,000 users requesting a server, where the Express framework outperformed the others. On the other hand, when the server was loaded with 8,000 users, the Express framework exhibited the highest error rate, which was slightly different from that obtained with the Django framework. In the initial three studies, the Express framework exhibited a minimal number of failed requests. However, when the server application was loaded with requests sent by 8,000 and 1,600 users simultaneously, the rate increased to 40%. Conversely, the Django framework fared the poorest of the three evaluated technologies, with a range of failed requests varying between 75% and 90%.

5.4. DELETE request

As a part of the tests for DELETE method requests, the functionality of cascade deletion for specific employees and their corresponding salaries was implemented. The results of this request are illustrated in Figure 8. The vertical axis represents the mean duration of a single request, while the horizontal axis denotes the applied load.

In contrast to prior studies, the results of this experiment did not reveal any distinct disparities in the mean request duration, relative to the used framework and the number of users. Among the three frameworks tested, Express demonstrated the fastest average request execution time of approximately 1,000 ms, for 1,000 virtual users. However, Django and Spring Boot were comparatively slower, with Django exhibiting a difference of less than 500 ms and Spring Boot exhibiting a variance of 1,000 ms. It is noteworthy that Express exhibited the most pronounced decrease in performance among the development frameworks tested, as the number of users increased, ultimately yielding the least favorable results under heavy loads. In contrast, the outcomes obtained with Spring Boot and Django were comparable, with execution time differences for individual tests falling within the range of 1,500 ms. Nonetheless, when tested with 16,000 virtual users, both frameworks exhibited an almost identical mean request execution time, of approximately 3,700 ms.

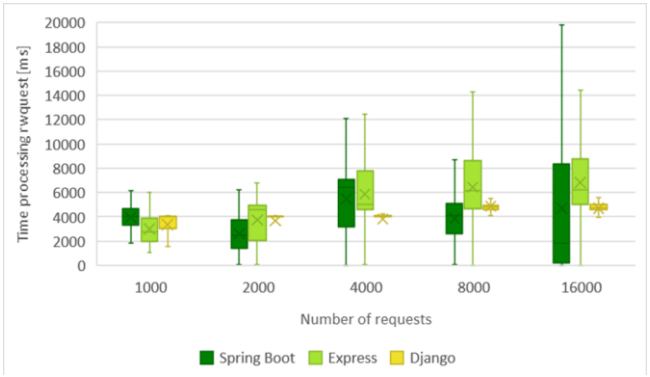


Fig. 8. The DELETE request processing time depending on the number of requests for each of the tested frameworks

Figure 9 displays the data pertaining to the reliability of a PUT request. The vertical axis denotes the proportion of requests that failed, while the horizontal axis represents the applied load under examination.

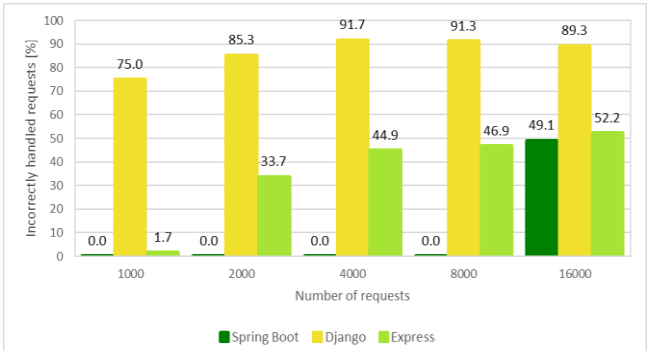


Fig. 9. The percentage of incorrectly handled DELETE requests by the application based on a given framework

With regard to the reliability of the DELETE request, Spring Boot (as depicted in Figure 9) produced the most favorable results, with no errors recorded across request loads ranging from 1,000 to 8,000 virtual users. However, when the server was subjected to a load of 16,000 users, the number of requests that were incorrectly handled amounted to approximately 50%. Notably, Django yielded considerably high error percentages, spanning from the lowest to the highest load tested. Meanwhile, Express exhibited an increasing trend of errors, rising from 0% for the 1,000-user server load to over 50% for a load of 16,000 users.

6. Conclusions

As a part of the experiment, three identical test applications were developed, each implementing a connection with the database and handling HTTP requests. Prior to implementing the test applications, a comprehensive literature review was conducted which helped define the type and parameters of these applications. The chosen database features a compact structure containing six tables, which have been populated with a significant amount of data. The selection of technology was predicated on the identification of the most prevalent server-

side programming languages, followed by choosing the top programming frameworks available for them. The conducted surveys made it possible to determine the efficiency and the reliability of tested technologies at various levels of load, obtain knowledge of the operation and properties of the evaluated frameworks, and draw conclusions that can help developers in selecting appropriate technologies for their programming projects.

Based on the obtained results, the following conclusions were formulated:

1) When comparing test cases under extreme load (with the server handling requests from 16,000 virtual users simultaneously), it was observed that Express had a significantly lower rate of failed requests than Spring Boot, although the application developed with the Express framework was noticeably slower.
2) Out of the surveyed frameworks, Spring Boot is identified by the highest request processing speed and high reliability for a server loaded with requests sent by 1,000-8,000 users.
3) The application utilizing the Django framework demonstrated the longest response time and the highest rate of errors during request handling.

Based on the above conclusions, it can be inferred that the research hypotheses have been verified. The superior results obtained with Spring Boot arise from the implementation of performance-enhancing mechanisms from the Spring framework. Unlike the Express framework, these mechanisms are an integrated part of the Spring and are configured and utilized by default. Express, on the other hand, according to its creator's arrangements, is a minimalist framework, which means that by default there are no these types of mechanisms. They are feasible to use thanks to the many libraries available for those purposes installed through the package manager. Nevertheless, that requires a programmer's knowledge of these mechanisms, their selection, and their configuration. Django exhibited the poorest performance compared to the other two technologies. This may be caused by its threaded architecture, which assigns a distinct thread to handle each request. In situations involving substantial loads, this approach results in augmented memory consumption and required processing time.

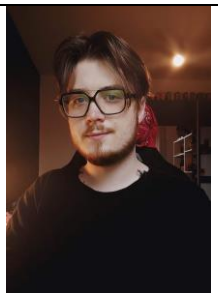
References

[1] Dhalla H. K.: A Performance Comparison of RESTful Applications Implemented in Spring Boot Java and MS.NET Core. Journal of Physics: Conference Series 1933, 2020.
[2] Kaluža M., Kalanj M., Vukelić B.: A comparison of Back-End Frameworks for Web Application development. Zbornik Veleučilišta u Rijeci 7, 2019, 317–332.
[3] Karlsson P.: A performance comparison Between ASP.NET Core and Express.js for creating Web APIs. Jönköping University 2021.
[4] Kopyl P., Rozaliuk T., Smołka J.: Comparison of ASP.NET Core and Spring Boot ecosystems. Journal of Computer Sciences Institute 22, 2022, 40–45.
[5] Muittari J.: Modern Web Back-End. What happens in the back end of the application? Oulu University of Applied Sciences 2022.
[6] Qvarnström E., Jonsson M.: A performance comparison on REST-APIs in Express.js, Flask and ASP.NET Core. Mälardalen University, 2022.
[7] Söderlund S.: Performance of REST applications: Performance of REST applications in four different frameworks. Linnaeus University 2017.
[8] Apache JMeter [https://jmeter.apache.org/] (available: 2023.03.04).
[9] Employees Sample Database [https://dev.mysql.com/doc/employee/en/] (available: 2023-04-18).
[10] GitHub Framework [https://github.com/topics/framework] (available: 2023.01.18).
[11] Most Popular Backend Frameworks – 2012/2022 [https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2022/] (available: 2022.11.22).
[12] Stack Overflow 2022 Developer Survey [https://survey.stackoverflow.co/2022/] (available: 2023.01.18).
[13] What is REST [https://restfulapi.net/] (available: 2023.01.18).

M.Sc. Dominik Choma
e-mail: dominik.choma@pollub.edu.pl

Dominik Choma received his master's degree in computer science in the area of web application at the Faculty of Electrical Engineering and Computer Science at the Lublin University of Technology. The author's research interests include graphic design, frontend, and backend technologies.

http://orcid.org/0009-0004-6302-5683



M.Sc. Kinga Chwaleba
e-mail: kinga.chwaleba@pollub.edu.pl

Kinga Chwaleba received a master's degree with distinction in Computer Science with specialization in Web Applications at the Faculty of Electrical Engineering and Computer Science at the Lublin University of Technology. The author's research interests include Java, Spring Boot, and backend technologies.

http://orcid.org/0009-0007-3458-5464

Ph.D. Mariusz Dzieńkowski
e-mail: m.dzienkowski@pollub.pl

Assistant professor in the Computer Science Department at the Faculty of Electrical Engineering and Computer Science at Lublin University of Technology. His scientific interests include human-computer interaction, eye tracking applications and web application development. He is a member of the Polish Information Processing Society.

http://orcid.org/0000-0002-1932-297X