Performance Evaluation of Back-end Web Application Programming Languages

Ivan Jelikj and Sasho Gramatikov
Faculty of Computer Science and Engineering
Ss Cyril and Methodius University
Skopje, North Macedonia

ivan.jelikj@students.finki.ukim.mk
sasho.gramatikov@finki.ukim.mk

***Abstract*—There are many programming languages that can be used for back-end web development. Numerous aspects of the project could be affected by the choice of a language. Because of that, the question about the most suitable programming language for a given web application arises.**

One of the main aims of this paper is to quantitatively compare 4 programming languages that could be used for back-end software development: Java, Kotlin, PHP and Python. Execution time, RAM usage, and CPU usage were selected as evaluation criteria. In order to be able to compare them, in all 4 given languages an application with the same functionality was created. The measurements were performed in an isolated environment for a different number of requests and different realistic scenarios.

From the results it could be concluded that Java and Kotlin have in general better execution time in comparison with PHP and Python, especially with a larger number of requests; Python has the smallest usage of CPU while the other 3 languages have similar usage; the usage of RAM in Python and PHP is significantly smaller than Kotlin and Java.

***Index Terms*—performance evaluation, back-end, Java, Python, Kotlin, PHP**

I. Introduction

There is a rapid increase in the number of applications developed in the world today. Many of those applications are web applications. Web applications usually consist of two parts: front-end and back-end. The front-end is usually done with HTML, CSS and JavaScript (or a language that transcompiles to JavaScript, such as TypeScript). For the back-end, there are many more languages that could be used.

The choice of a programming language has an impact from many aspects on the development of software: the complexity of the code, the execution time, the hardware requirements needed for the software to execute, the difficulty of testing the code, the maintenance of the code, etc. All these aspects, in turn, have an impact on the project itself: the delivery time, the cost of the project and ultimately the success of the project.

This paper evaluates the performance of 4 different programming languages: Java, Kotlin, PHP and Python. All of these languages are widely used for professional software development [1]. There are currently 2896188 public repositories on GitHub in Java, 252426 in Kotlin, 928190 in PHP and 3116184 in Python [2]. All 4 languages can be regarded as reasonable candidates when developing a real world back-end web application. This makes the analysis of this study relevant and applicable.

The rest of the paper is organized in the following manner: Section 2 discusses related work, Section 3 discusses the experimental environment and the functionality of the application that was tested, in Section 4 the results and the analysis are provided, and Section 5 is the conclusion of our paper with the summary of the study.

II. Related work

In [3], a comparison is made between Java and Kotlin when developing a web application with the frameworks Spring, Micronaut and Ktor. The execution time, the usage of CPU and the usage of RAM are measured. In this paper, Kotlin shows worse results, but the difference is usually very small (for example, the difference in the usage of the processor is 2%).

In [4], C, C++, Java, Perl, Python, Rexx and Tcl are compared. From the languages that are compared in our paper, Java and Python have similar execution time, while Python is 2 times more efficient in the usage of memory. The time needed to write the software was also measured, and it was concluded that the developers needed twice as much time to write the code in Java, with twice as much code as in Python. This paper is one of the most influential paper in the field of evaluation and comparison of programming languages.

In [5], 10 open source project in Java and Python are compared. The conclusion is that the number of lines of code is similar in both languages.

In [6], Node.js, Python and PHP are compared. It is concluded that Node.js shows better results when there is a high concurrency. This was true no matter the scenario. PHP handles small requests well, but has difficulty with large requests. Both Python and PHP show difficulty when computing a mathematical operation (Fibonacci sequence).

In [7], PHP and Java are compared. It can be concluded that Java shows significantly better results when doing more complex mathematical computations, as well as significantly better scalability.

In [8] the sorting algorithm Quicksort and the game tic-tac-toe are compared for Java and Python. Java has better execution time than Python, especially when many elements are sorted. Java also shows better execution time for the game tic-tac-toe, although the difference is relatively small. The number of lines of code is also compared and it was concluded that Java is more verbose than Python.

Comparing and analysing programming languages, in different contexts and in different scenarios, in which the evaluation is done for various different criteria, is generally quite common topic in the field of software engineering. Despite this, we were unable to find a study regarding the evaluation of the 4 languages in this paper: Java, Kotlin, PHP and Python.

III. Experimental environment

When doing performance evaluation, the server and the client are often separate machines and are connected through a network. The network itself could have an impact on the end results, and that's why that approach was avoided in this paper.

The same machine was used for creating the request as well as serving the request, i.e., it could be said that the client and the server were run on the same machine. It was possible to do so because the environment in which the applications ran was containerized through Docker. Docker has a very lightweight use of memory and very fast time for creating containers [9] [10]. For those reasons, Docker is very suitable for performing performance evaluation, especially because in performance evaluation each measurement is usually performed many times. In our experiment, each measurement was conducted 10 times, and after each measurement the application and the database environment were recreated.

The machine on which the application was run was Intel i7-920 processor, DDR2 RAM with capacity of 17,8 GB, HDD with capacity of 500 GB and operating system Ubuntu 16.04.

The technologies used in the experiment were: Apache JMeter 5.4.1 for creating the requests and measuring them; Docker 18.06 was used for containerization; Python 3.8 with Flask 2.1 and SQLAlchemy 1.4; PHP 7.4; Java JDK 11.0.5 and Kotlin 1.6 with JRE Java 11.0.13, together with Spring Boot 2.6.1 and Hibernate Commons Annotations 5.1.2, and PostgreSQL 14.1. For each language a suitable web server was chosen: Java and Kotlin with Tomcat 9.0.55, Python with Gunicorn 20.1.0, and PHP with Apache HTTP Server 2.4.38. Each server was configured using the best practices for the given number of requests.

The application had 7 different scenarios: creating an entry, reading entries, updating an entry, deleting an entry, CRUD operation in which all the previous operations are executed one after the other, a mathematical operation (sum of numbers) and input operation (finding the word frequency in the sonnets of Shakespeare by reading a text file and then processing it). For the scenarios of updating an entry, deleting an entry, reading an entry and the CRUD operation, authentication was required. The concepts of encryption and authorization, as well as comparing the security aspects of the languages, were outside the scope of this paper. Each scenario is implemented in the given language using the same logic. 1000, 2500, 5000 and 10000 simultaneous requests were made.

IV. Results

A. Execution time

Fig. 1 shows that the average execution time of Kotlin and Java is similar (30,6s vs 31,1s). The average execution time of PHP and Python is 64,1s and 62,7s.

The scalability of Python and PHP is worse than Java and Kotlin. Fig. 2a shows that for a smaller number of simultaneous requests the ratio difference between the average execution time is smaller (Kotlin, as the fastest language for 1000 requests has an execution time of 11,27s, while Python, which has the slowest average execution time of 17,21s, which means that Kotlin is 1,5 times faster). That difference becomes bigger for 2500, 5000 and 10000 requests. For 2500 requests, for example, Kotlin has an average execution time of 19,6s and Python of 43,49s, which means that Kotlin is 2,2 times faster. It can also be concluded that Kotlin and Java have similar average execution time regardless of the number of simultaneous requests.

Fig. 2b shows the biggest difference from all the seven scenarios in the average execution time, the mathematical operation sum. For 1000 requests, Java as the fastest language has an average time of 7,62s and Python as the slowest of 37,05s, which is a difference of 4,9 times. As in the other scenarios, the difference for higher number of simultaneous request is bigger. For 10000 users, Java has a time of 20,65s and PHP of 234,75s, which shows that PHP is in this case 11,4 times slower.

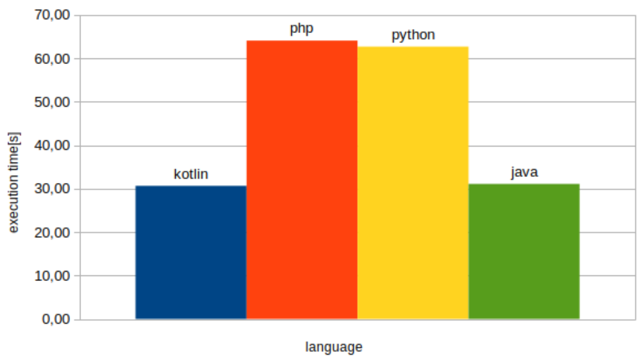
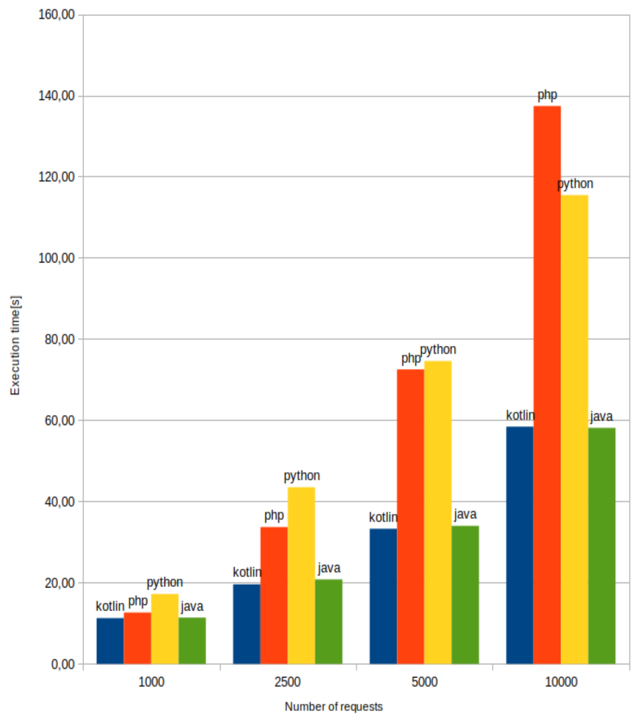
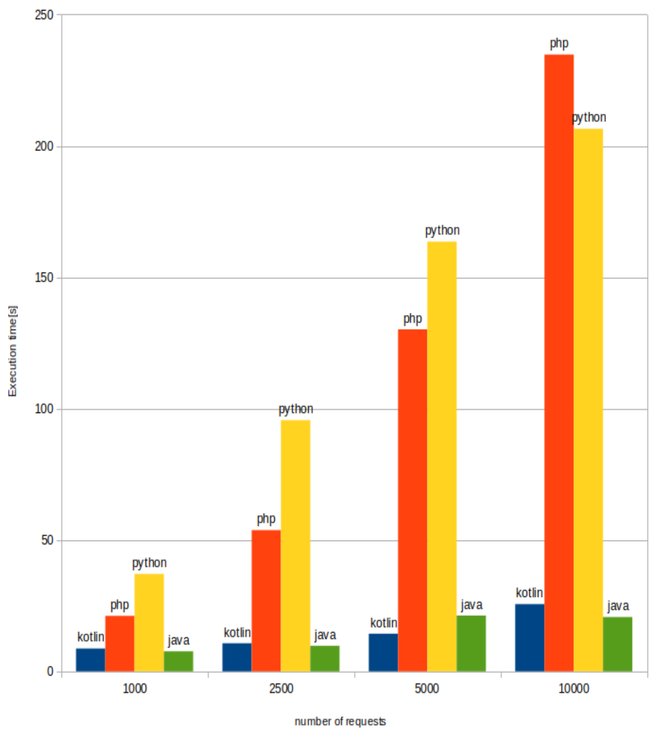


Fig. 1: Average execution time.

Table I shows the share of each language for the different scenarios in the execution time. The share is calculated by initially calculating the share of the given programming language for a given number of requests in the given scenario (for example, the share of Java in the total execution time when the scenario is creating an entry for 1000 simultaneous requests). After that calculation, the same is done for other requests (for example, for 2500, 5000, 10000) and the average is taken. It can be seen that for the operation read, Python has



(a) For all operations.



(b) For the operation sum.

Fig. 2: Average execution time.

the smallest share of 21,18%, for the operation sum Java has the smallest share(6,66%) and for all other scenarios Kotlin has the smallest share in the execution time. Python has the largest share for 5 scenarios and PHP for 2.

Fig. 3 shows the share of a given language in the execution time for all scenarios combined. It can be seen that Kotlin has the smallest share in the execution time and Python has the largest.

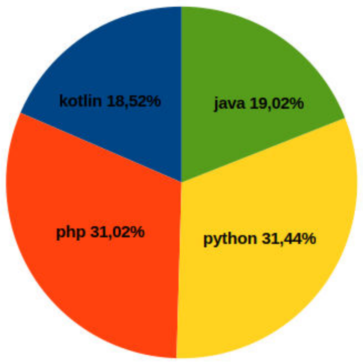


Fig. 3: Share in the execution time.

B. RAM usage

Fig. 5 shows that Kotlin and Java have similar RAM usage and significantly higher usage than PHP and Python (PHP,

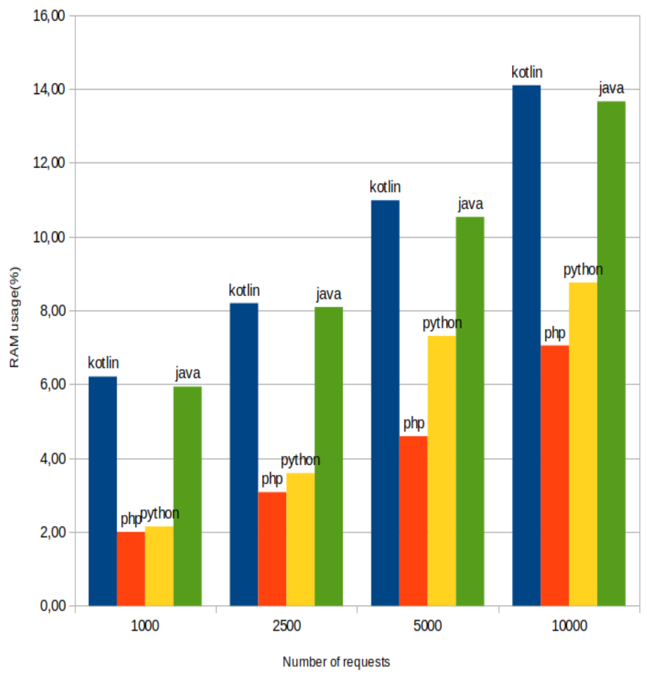
TABLE I: Share in the execution time.

	Kotlin	PHP	Python	Java
create	21,93%	32,92%	22,68%	22,48%
read	22,66%	32,76%	21,18%	23,41%
update	19,73%	28,86%	30,82%	20,59%
delete	19,41%	29,61%	30,63%	20,35%
CRUD	19,08%	29,07%	32,42%	19,43%
Shakespeare	19,96%	26,97%	32,86%	20,22%
sum	6,89%	36,93%	49,52%	6,66%

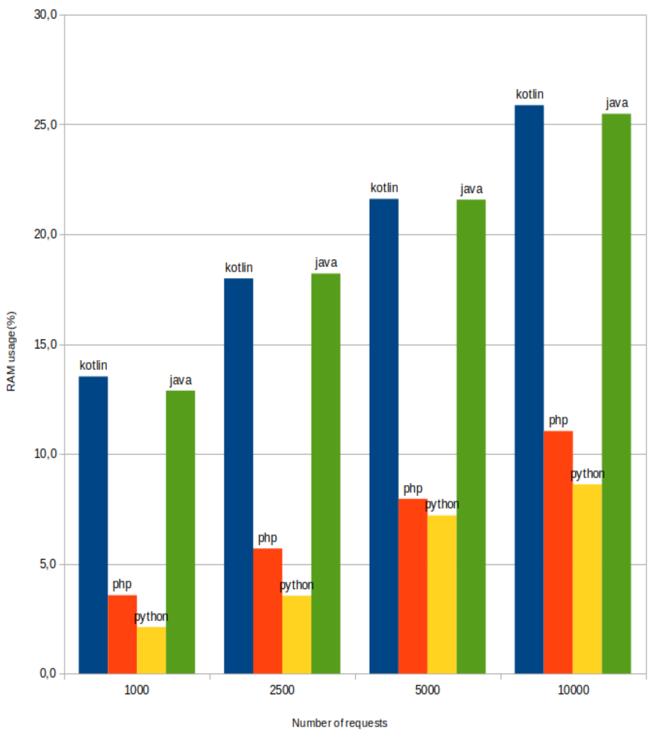
which has the lowest RAM usage, has an average usage of 4,18% and Kotlin, which has the highest, of 9,87%). As shown in Fig. 4a the ratio of difference in RAM usage for Kotlin and Java on the one hand, and PHP and Python on the other, is biggest for a smaller number of simultaneous requests. For 1000 requests, Kotlin has a usage of 6,21% and PHP of 2,00%(3,1 more usage). For 10000 requests Kotlin, which has again the highest usage, has a usage of 14,1% and PHP, which has lowest, of 7,05% (2 times difference).

The biggest difference, shown in Fig. 4b in RAM usage, is present in the scenario of reading files (the Shakespeare scenario). When doing 1000 simultaneous request, Python, as a language with the lowest RAM usage of 2,1% and Kotlin, as a language with the highest RAM usage of 13,5%(a difference of 6,4 times). For 10000 users, Python has a usage of 8,6% and Kotlin of 25,9%(a difference of around 3 times).

Table II shows the share of each language and for each



(a) For all operations.



(b) For the operation Shakespeare.

Fig. 4: Average RAM usage.

scenario in the RAM usage. When doing an update operation and when doing deletion, Java has the largest share in the RAM usage (30,75% and 30,55%), marginally larger share than Kotlin. In all other scenarios, Kotlin has the largest share in the RAM usage. Python has the smallest share for the operation CRUD (13,61%) and Shakespeare(9,72%). For all other operations, PHP has the smallest share in the RAM usage. Fig. 6 shows the share of each language in the total RAM usage. PHP has the smallest share and Kotlin has the largest.

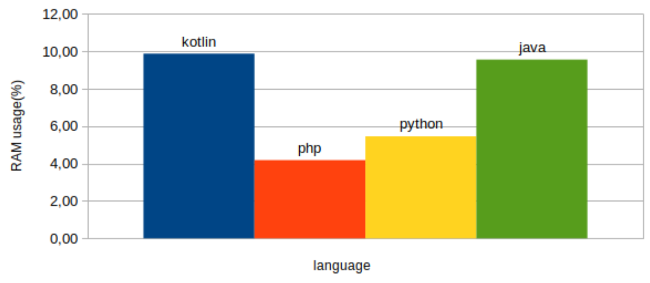


Fig. 5: Average RAM usage.

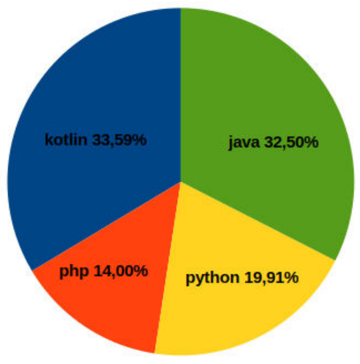
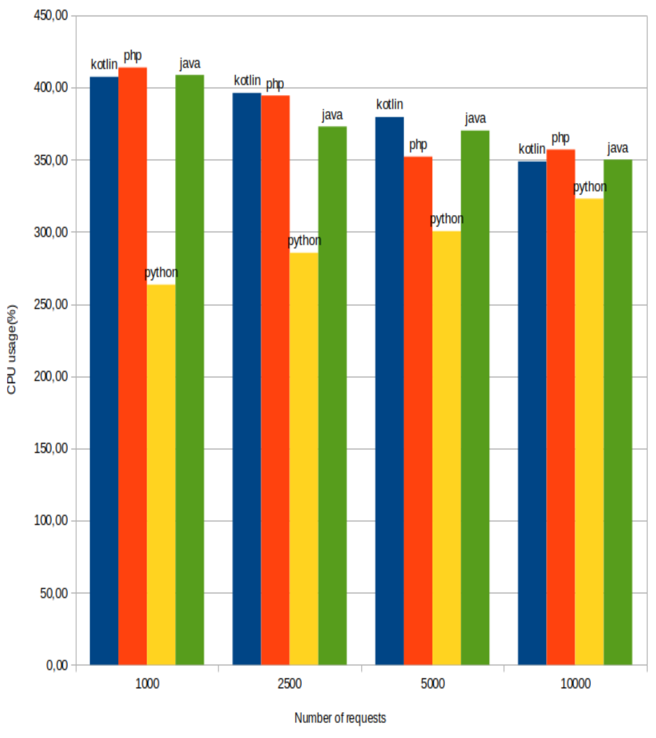


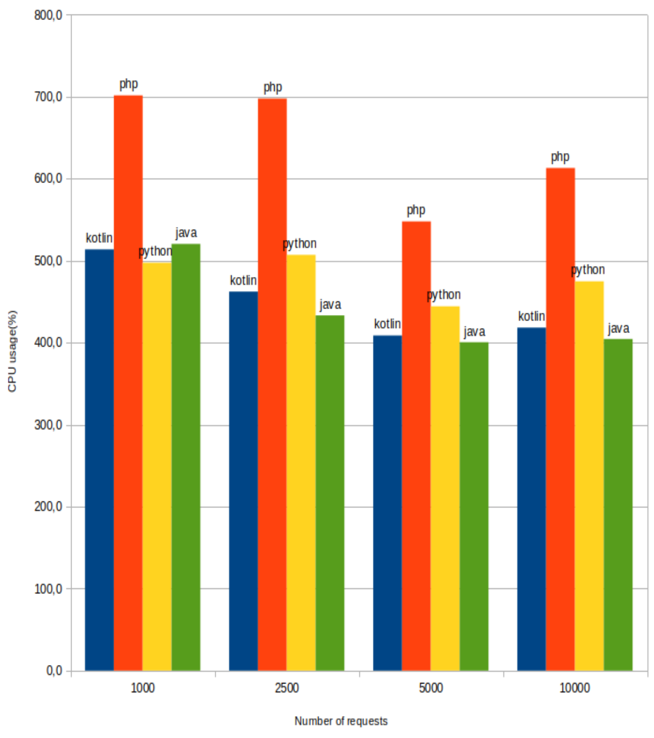
Fig. 6: Share in the RAM usage.

TABLE II: Share in the RAM usage.

	Kotlin	PHP	Python	Java
create	35,69%	11,67%	19,50%	33,14%
read	32,99%	12,89%	22,75%	31,38%
update	30,62%	13,71%	24,93%	30,75%
delete	29,76%	14,17%	25,51%	30,55%
CRUD	36,48%	15,24%	13,61%	34,67%
Shakespeare	38,80%	13,21%	9,72%	38,27%
sum	30,76%	17,09%	23,36%	28,78%



(a) For all operations.



(b) For the operation sum.

Fig. 7: Average CPU usage.

C. *CPU usage*

In Fig. 8 it could be noticed that use of CPU is smaller in Python and that the other three languages have similar CPU usage. Fig. 7a shows that the difference between Python and the other languages gets bigger for fewer numbers of simultaneous requests (1000), and is getting smaller for more requests (10000). For 1000 requests, Python has a CPU usage of 263,43% and PHP, which has the biggest usage, has a usage of 413,87% (PHP uses 1.57 times more CPU when executing the application). For 10000 users Python has a usage of 322,98% and PHP, which has the highest CPU usage for this category of 357%(1.10 times more usage in PHP).

In Fig. 7b it can be seen that the biggest difference in the CPU usage was, as it was the case with the average execution time, with the mathematical operation sum. For 2500 requests, Java, which had the smallest usage, was at 433,2%, while PHP, which had the highest, at 697.7%.

Table III shows the share of each language and for each scenario in the CPU usage. The differences between languages in CPU usage are smaller than in RAM usage and execution time. There are 4 scenarios for which PHP has the largest share, 1 in which Java has it and 2 in which Kotlin has it. For the smallest share, there are 4 in which Python has the smallest usage, 2 in which PHP has it and 1 in which Java has the smallest usage. Fig. 9 shows the share of each language in the total CPU usage.

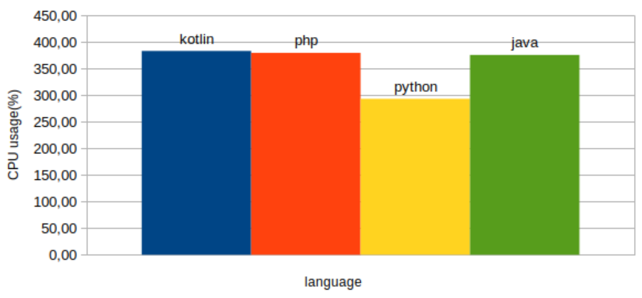


Fig. 8: Average CPU usage.

TABLE III: Share in the CPU usage.

	Kotlin	**PHP**	**Python**	**Java**
create	28,23%	29,39%	14,57%	27,81%
read	27,00%	20,33%	25,38%	27,29%
update	28,41%	27,93%	16,78%	26,88%
delete	27,71%	28,33%	16,72%	27,24%
CRUD	28,18%	28,61%	16,21%	27,01%
Shakespeare	27,84%	21,66%	22,99%	27,52%
sum	22,40%	31,78%	23,98%	21,83%

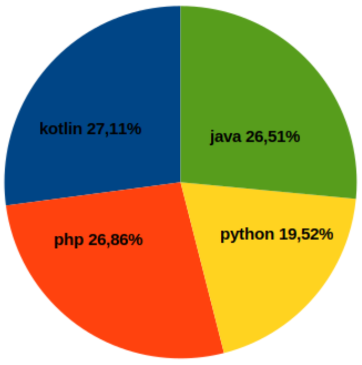


Fig. 9: Share in the CPU usage.

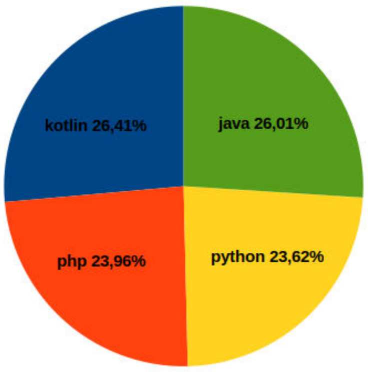


Fig. 11: Share of each language for all criteria.

D. General results

As shown in Fig. 11 when we take all three criteria into account: the execution time, the RAM usage and the CPU usage, the smallest share, that is, the best results are shown by Python with 23,62% share. On second place it is PHP with 23,96%, on third place Java with 26,01% and on the last place Kotlin with 26,41%.

E. Lines of code

As shown in Fig. 10 Python has 564 lines of code, PHP has 630, Kotlin has 863, and Java 964 lines of code. One of the main design aims of Kotlin, to be more concise than Java [11], has been achieved.

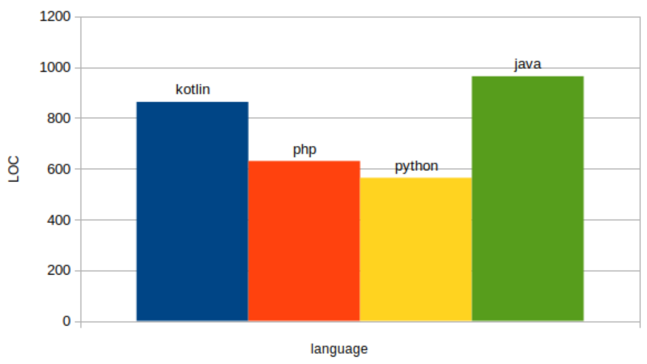


Fig. 10: Lines of code.

V. Conclusion

This paper is a measurement study of four programming languages (Java, Python, PHP and Kotlin) in the context of back-end web development. To the best of our knowledge, this is the first study that does this. Seven different realistic scenarios, with different number of simultaneous requests were used to evaluate the language, which makes the results more practical and applicable.

We can conclude that Java and Kotlin have much better scalability than Python and PHP. Python and PHP show good execution time for a small number of requests (1000), but have difficulty with larger ones. Regardless of the number of requests, the most significant difference between the languages in the execution time is for mathematical computation. For a large number of requests, PHP and Python show more than 11 times slower execution time for a mathematical operation.

Java and Kotlin use significantly more memory than PHP and Python.

With regard to CPU usage, Python shows less usage than the other three languages, which have similar usage.

When the share of each language is taken into account for all the three criteria together (execution time, RAM usage and CPU usage) Python shows the best results, followed by PHP, Java and Kotlin.

With regard to verbosity, Python is the least verbose and Java is the most verbose language.

This paper can facilitate the process of selecting a suitable programming language when starting a new project. The manner in which the measurements are performed, by minimizing the outside influence on the application, could also be used in context of other programming languages. It is also possible to do further research for the same 4 languages, but the scope or the architecture of the application to be different(for instance, using microservices), or to compare different functionalities of the same languages which were not evaluated in this paper.

References

[1] https://www.tiobe.com/tiobe-index/ [Accessed 10 March 2023].
[2] www.github.com [Accessed 10 March 2023].
[3] G. Bujnowski and J. Smołka, ”Java and Kotlin code performance in selected web frameworks,” Journal of Computer Sciences Institute, vol. 16, pp. 219-226, 2020.
[4] L. Prechelt, ”An empirical comparison of seven programming languages,” Computer, vol. 33, no. 10, pp. 23–29, 2000. doi:10.1109/2.876288
[5] G. Destefanis, M. Ortu, S. Porru, S. Swift, and M. Marchesi, ”A statistical comparison of Java and Python software metric properties,” Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics - WETSoM ’16, doi:10.1145/2897695.2897697
[6] K. Lei, Y. Ma, and Z. Tan, ”Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js,” 2014 IEEE 17th International Conference on Computational Science and Engineering, doi:10.1109/cse.2014.142
[7] P. Neves, N. Paiva, and J. Durães, ”A comparison between JAVA and PHP,” Proceedings of the International C* Conference on Computer Science and Software Engineering - C3S2E ’13, doi:10.1145/2494444.2494465

[8] S. Khoirom, M. Sonia, B. Laikhuram, J. Laishram, and T. Davidson, ”Comparative Analysis of Python and Java for Beginners,” International Research Journal of Engineering and Technology (IRJET)

[9] B. B. Rad, J. B. Harrison, and M. Ahmadi, ”An introduction to docker and analysis of its performance,” International Journal of Computer Science and Network Security (IJCSNS) 17.3 (2017): 228.

[10] D. Merkel, ”Docker: lightweight linux containers for consistent development and deployment,” Linux j, vol. 239, no. 2.

[11] D. Jemerov and S. Isakova, ”Kotlin in action,” Manning Publications Co.