Michał Szalej[1], Jakub Włodarczyk[2],
Tymoteusz Wenerski[3], Damian Malczewski[4]
[1] Opole University of Technology, Opole, Poland, s101482@student.po.edu.pl,
[2] Opole University of Technology, Opole, Poland, s101491@student.po.edu.pl,
[3] Opole University of Technology, Opole, Poland, s101477@student.po.edu.pl,
[4] Opole University of Technology, Opole, Poland, s101169@student.po.edu.pl.

Comparison of Selected Web Frameworks in Specific Developer Use Cases

KEYWORDS
different applications, framework performance, financial trading platforms, web-based news websites

ABSTRACT
This article discusses the results of benchmark tests that were conducted to evaluate the performance of different frameworks for different types of applications. The tests found that Just.js is the recommended framework for real-time financial trading platforms and web-based news websites due to its exceptional performance on specific tests and its lightweight and easy-to-use nature. Mayminihttp is recommended for e-commerce platforms because of its good performance on multiple query, JSON serialization and data update tests and scalability. The article also emphasizes that the best framework for a particular application will depend on various factors such as the specific requirements of the application, the development team's expertise, and available resources, so it is recommended to evaluate other factors before making a decision on which framework to use such as support and documentation, compatibility, security, and resource management.

1. INTRODUCTION

With each passing year, the use of web technologies by average users is growing exponentially, leading to a high demand for performance, reliability, and scalability of the systems being created. An essential part of the toolset for creating websites are Frameworks, as they speed up the development process and shorten the time it takes, thus helping to meet coding standards [1].

With the emergence of many programming languages for web applications, it is difficult to realize the advantage of one language in this area over others [2]. We make an effort to realize these issues and compare mainly used languages such as Java, C++, C, C#, PHP, JavaScript, Rust and Python, and their most popular and welldocumented frameworks. For Java, the studied structures are Spring and Helidon, for PHP Laravel and Comet, for Python Django and Japronto, for Rust mayminihttp and Salvo. In C#, ASP.net and BeetleX have been checked, in JavaScript solutions are Node.js, and Just.js. In C language Framework h2o, and in C++ Drogon.

A framework makes web programming easier and better organized in many ways. Firstly, frameworks increase programming productivity because writing a piece of code that would typically take hours and take hundreds of lines, can be done in a matter of minutes with built-in functions. Secondly, a widely used framework has a significant advantage in terms of security because its users become long-term testers. If a user finds a security problem, they can report it to the framework creators' website so that the developer team can fix it. Thirdly, often most popular frameworks are free, and since they help the programmer write code faster, the final cost of application development will be significantly lower. Fourthly, a framework usually comes with a support team, documentation, or a large forum where users can quickly get answers. Choosing a framework is an important step as it will determine the speed and quality of the final product [3].

Many users browse multiple websites at once navigating through various sources of information, so software limitations can easily lead to the loss of viewers [4]. Frameworks are also good for maintaining an application over a long period of time, because they make it easier to understand the code of the application that always within one technology works in a predictable and known way to the developers.

The main goal of this research is to help web developers in choosing the best solution for per-

forming their work, and to compare the capabilities of the standards of their native programming language with available alternatives among other programming languages.

2. MEASUREMENT METHODS AND ENVIRONMENT

2.1. Measurement methods

The measurements were carried out by TechEmpower, which is a project that conducts regular performance and functionality tests on a variety of web application frameworks and platforms. The tests, known as the benchmarks, evaluate the performance of different frameworks and platforms in scenarios such as serving dynamic content, executing database queries, and handling HTTP requests. Tests use automated scripts to simulate real-world workloads and record the performance of the systems being tested. The results of these tests are analysed and published in the form of rankings and charts on the TechEmpower website. The Framework Benchmarks project includes several different types of tests:

- "Plaintext" tests: These tests measure the time it takes for a framework to return a simple "Hello, World!" message. These tests are designed to measure the raw performance of the framework and do not include any additional processing or rendering.
- "JSON serialization" tests: These tests measure the time it takes for a framework to serialize a JSON object and return it to the client. These tests are intended to measure the performance of the framework's JSON serialization library.
- "Single query" tests: These tests measure the time it takes for a framework to execute a single database query and return the results to the client. These tests are intended to measure the performance of the framework's database integration.
- "Multiple query" tests: These tests are similar to the single query tests, but they execute multiple queries in a single request. These tests are intended to measure the framework's performance when executing multiple queries in a short time.
- "Fortunes" tests: These tests measure the time it takes for a framework to execute a database query, generate an HTML page from the results, and return the page to the client. These tests are intended to measure the performance of the framework when rendering dynamic content.
- "Updates" tests: These tests measure the time it takes for a framework to execute a series of database updates and return the results to the client. These tests are intended to measure the performance of the framework when executing multiple updates in a short period of time.
- "Cached queries" tests: These tests are similar to the single query tests, but they use caching to store the results of the query in memory. These tests are intended to measure the performance of the framework's caching implementation.

2.2. Environment

The above-mentioned benchmarks are a set of performance tests designed to evaluate the performance of web application frameworks. These tests are run in a variety of environments, cloud based, and physical. Cloud environment is set on Azure server provider. Physical benchmarks are tested on “Citrine” server which consists of three homogeneous Dell R440 servers each equipped with an Intel Xeon Gold 5120 CPU, 32 GB of memory, an enterprise SSD and dedicated Cisco 10-gigabit Ethernet switch. The goal of the benchmarks is to provide a fair comparison of the performance of different frameworks under a consistent set of conditions, in order to help developers, choose the best tool for their needs. By publishing the results of these benchmark tests, divided not only on language, but also by environment, developers can make informed decisions about which technologies are best suited to their needs. This can help to ensure that they are using the most appropriate tools for their projects and can ultimately lead to better software development practices and more efficient and effective applications.

3. ANALYZE METHOD

In our analysis of the TechEmpower benchmark data, we used Matlab to create graphs and visualizations of the results. Matlab is a powerful and widely used software tool for data analysis, visualization, and mathematical computation, and it allowed us to represent the data in a clear and concise manner. The graphs and visualizations we created with Matlab helped to highlight trends and patterns in the data and provided a more intuitive way of comparing the performance and functionality of different web application frameworks.

Overall, the use of Matlab played a crucial role in our analysis and presentation of the benchmark data, helping to make the results more accessible and easier to understand for a wide audience.
Matlab Script to analyze data is as follows:

```
Framework =
string(height(jsonser));
Percentage = ze-
ros([height(jsonser), 1]);
max = 0;

for i = 1:height(jsonser)
    Framework(i) =
string(jsonser{i, 1});
    if jsonser{i, 3} > max
       max = jsonser{i, 3};
    end
end

for i = 1:height(jsonser)
    Percentage(i) = jsonser{i, 3} /
max * 100;
End

for i = 1:height(jsonser)
    for j = 1:height(jsonser)
        if Percentage(i) < Percent-
age(j)
           tmp = Percentage(i);
           tmp2 = Framework(i);
           Percentage(i) = Percent-
age(j);
           Framework(i) = Frame-
work(j);
           Percentage(j) = tmp;
           Framework(j) = tmp2;
        end
    end
end

figure(1);
b = barh(Percent-
age);
grid on;
yticklabels(Frame-
work);
xlabel("Percentage
[%]");
box off
title("Multiple Query compara-
tion.");
b.FaceColor = "#4f5bd5";
```

4. COMPUTING ALGORITHMS

Each test result, except the sum up score, were responses per second for a specific framework. The final score was a composite result of all tests with individual weights. Weights were determined using TPR scoring algorithm. The algorithm is quite simple:

1. Select 10 frameworks to cover a wide range of test scores and languages.
2. Calculate mean RPS of selected 10 frameworks.
3. *Normalize* the magnitude of each means to align with the most popular test – JSON serialization. For example, if JSON average RPS is equal to 150,000 and Fortunes mean RPS is equal to 10,000 – the Fortunes normalizing will be given score of 15 (150,000/10,000 = 15).
4. Apply semi-fixed test biases listed below:
   a. JSON – 1.0
   b. Single query – 0.75
   c. 20-query – 0.75
   d. Fortunes – 1.5
   e. Updates – 1.25
   f. Plaintext – 0.75

5. RESULTS

In Figure 1, a comparison is made of the process of retrieving a plaintext response from a database. This information can be useful in understanding the various steps and considerations involved in accessing and retrieving data from a database. By comparing the different approaches to getting a plaintext response, you can gain insight into the pros and cons of each method and determine the most appropriate solution for your needs.

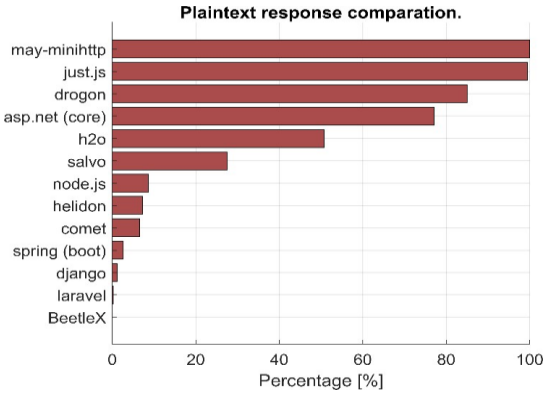


Figure 1. Comparison of getting a plaintext response in relation to best result in the collection

This information can be helpful for developers working with databases and looking to optimize the performance and efficiency of their data retrieval processes. In addition, understanding the

process of getting a plaintext response from a database can be useful for anyone interested in the underlying mechanics of data storage and retrieval.

Figure 2 is showing evaluation of the web framework on its ability to serialize an object into JSON format and return it as a response to the client. The object being serialized consists of a single key-value pair, with the key "message" and the value "Hello, World!". The performance of the web framework is measured based on how efficiently and quickly it can serialize this object and return it as a response. This test is designed to assess the web framework's ability to handle basic JSON serialization tasks and provide fast and reliable responses to clients.

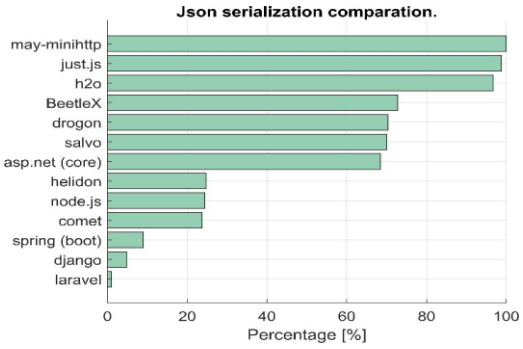


Figure 2. Comparison of JSON Serialization

It is important to note that the simplicity of the object being serialized in this test may not accurately reflect the complexity of real-world JSON serialization tasks that web frameworks are expected to handle. However, this test can still provide valuable information about the web framework's performance and can be used to compare the efficiency of different frameworks in handling basic JSON serialization tasks.

Additionally, the results of this test can be used to identify any potential bottlenecks or inefficiencies in the web framework's JSON serialization process, which can be addressed and optimized in future development.

In Figure 3 the test is making a request to the web framework, which retrieves a single row from a database table and converts it into a JSON format for the response. The performance of the web framework is evaluated based on how efficiently and quickly it can complete this process. This test is designed to measure the web framework's ability to handle simple database queries and serialize the resulting data as JSON.

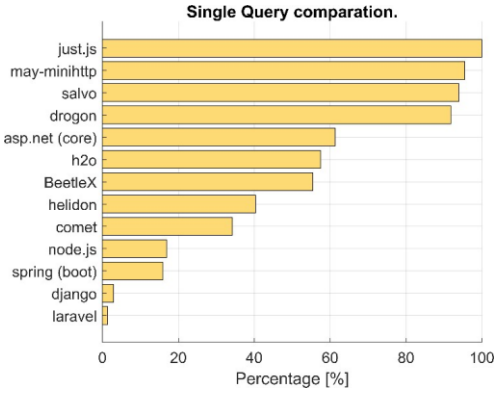


Figure 3. Schematic shows comparison of single query request to one database table

To further elaborate, the purpose of this test is to assess the web framework's performance in a scenario where it needs to retrieve a single row of data from a database and return it to the client in a JSON format. This is a common task that web frameworks are expected to perform, and so it is important to evaluate their efficiency and speed at doing so. The results of this test can be used to compare the performance of different web frameworks and identify any strengths or weaknesses in their handling of simple database queries and JSON serialization.

In Figure 4 the shown comparison presents a test of retrieving multiple rows from database and converting it to JSON format prepared for responses. This process is used to assess the performance of a web framework based on how efficiently and quickly it can execute simple database queries and convert the retrieved data into a JSON format. The aim of this test is to evaluate the web framework's capability to handle this type of task.

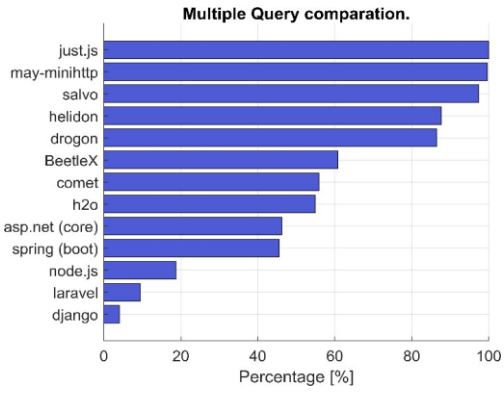


Figure 4. Comparison of fetching multiple rows from single database table and serializing of these rows as a JSON response

In Figure 5, the performance of a framework's Object-Relational Mapping (ORM) is tested in re-

trieving and manipulating data from a database table containing fortune cookie messages. The ORM is used to fetch all rows from the table, which has an unknown number of rows, and an additional fortune cookie message is inserted into the list at runtime. The list is then sorted by the message text, and the resulting list is delivered to the client using a server-side HTML template. The test ensures that the message text is properly escaped and treated as untrusted, and that the UTF-8 fortune messages are rendered correctly. This information can be useful in understanding the capabilities and performance of a framework's ORM in retrieving and manipulating data from a database. It can also be helpful for developers working with databases and looking to optimize their code for data retrieval and manipulation.

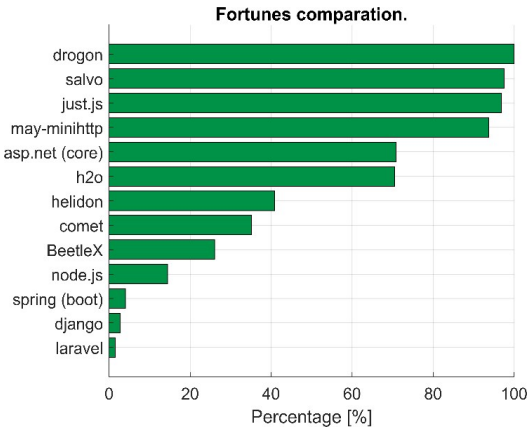


Figure 5. Comparison of fetching a full data table of Unix random fortune cookies

Figure 6 showed comparison of performance on updating databases. In this test, the system is designed to handle incoming requests by performing multiple databases write operations. Each request involves the following steps:

1. Fetching multiple rows from a simple database table
2. Converting the rows to in-memory objects
3. Modifying one attribute of each object in memory
4. Updating each associated row in the database individually
5. Serializing the list of objects as a JSON response

The test is run multiple times, testing the performance of the system when performing 1, 5, 10, 15, and 20 updates per request. It's important to note that the number of statements per request is twice the number of updates, since each update is paired with a query to fetch the object. The test is run at 512 concurrencies, which means that there are 512 requests being processed simultaneously. The purpose of this test is to evaluate the system's ability to handle multiple databases write operations efficiently.

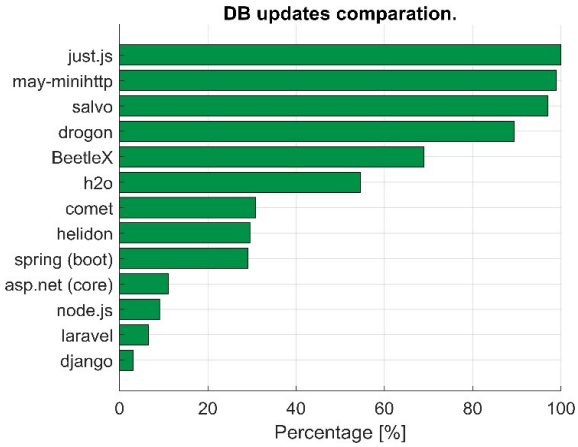


Figure 6. Comparison of performance on updating databases

The scores shown in Figure 7 for the performance of different web frameworks are calculated by taking a weighted average of the results of several tests. The weights assigned to each test are as follows: JSON serialization = 1.0, single query = 1.737, multiple queries = 21.745, fortunes = 4.077, data updates = 68.363, and plaintext = 0.163. These weights reflect the relative importance of each test in evaluating the overall performance of the web frameworks. It is important to understand that the weights assigned to each test in calculating the overall performance scores are subjective and may not necessarily reflect the relative importance of these tests to all users of the web framework. The specific needs and priorities of an individual or organization will determine which aspects of a web framework's performance are most critical, and therefore, which tests and weights should be given more consideration. For example, if an organization primarily focuses on developing web applications that require frequent updates to large amounts of data, then the "data updates" test and its corresponding weight of 68.363 would be more relevant and important in evaluating the overall performance of the web framework. On the other hand, if an organization primarily focuses on developing web applications that primarily serve static content, then the "plaintext" test and its corresponding weight of 0.163 may be more relevant and important.

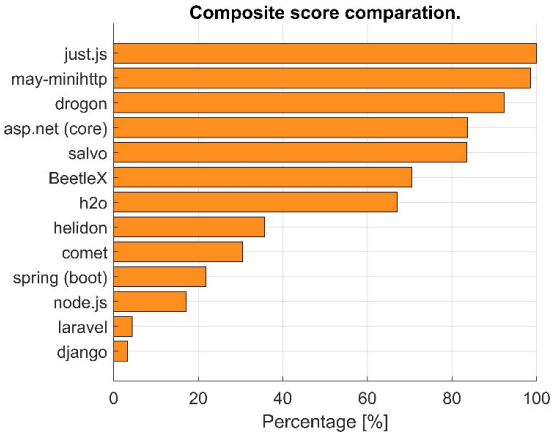


Figure 7. Comparison of composite scores

In Figure 8, the number of watchers for selected repositories on GitHub is presented. This data can be useful in highlighting the level of interest and attention that specific projects are receiving. By displaying the number of watchers, you can provide insight into the level of engagement and support for a particular repository. This information can be helpful for developers looking to contribute to or use a particular project, as well as for those interested in understanding the impact and reach of different open-source projects on GitHub. In addition, the number of watchers can be a good indicator of the potential size and activity of a repository's community.

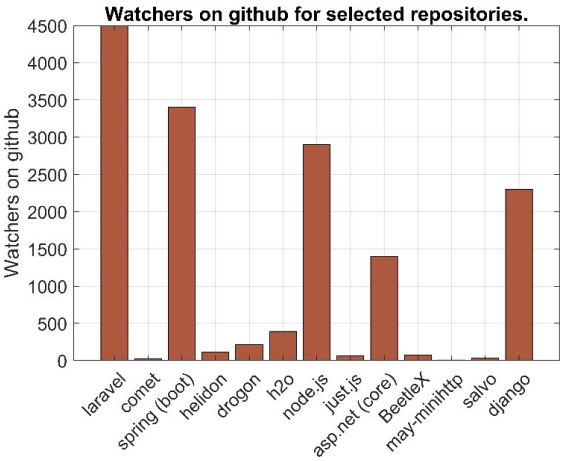


Figure 8. Showcase of number of watchers for selected repositories on GitHub

In Figure 9 the number of forks for selected repositories on GitHub is displayed. This information can be useful in demonstrating the level of community engagement and support for a particular repository. By showing the number of forks, you can provide insight into the popularity and adoption of specific projects. This information can be helpful for developers looking to contribute to or use a particular project, as well as for those interested in understanding the impact and reach of different open-source projects on GitHub.

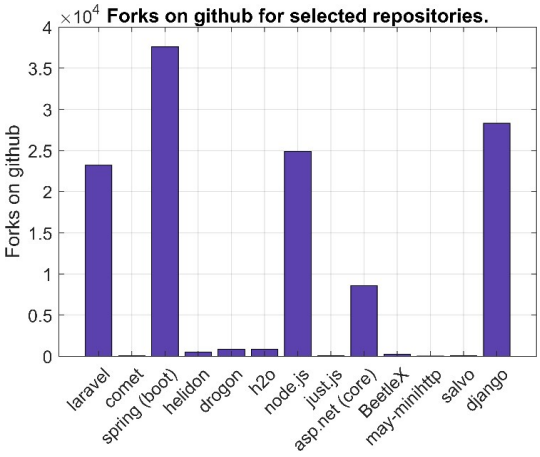


Figure 9. Showcase of number of forks for selected repositories on GitHub

In Figure 10, the number of stars for selected repositories on GitHub is shown. The data demonstrates the popularity and recognition of these specific projects, providing insight into the level of appreciation and support they have received. This information can be helpful for developers looking to contribute to or use these projects, as well as for those interested in understanding the impact and reach of different open-source projects on GitHub. The number of stars is often considered a form of endorsement on GitHub and can be a good indicator of the overall quality and usefulness of a repository.

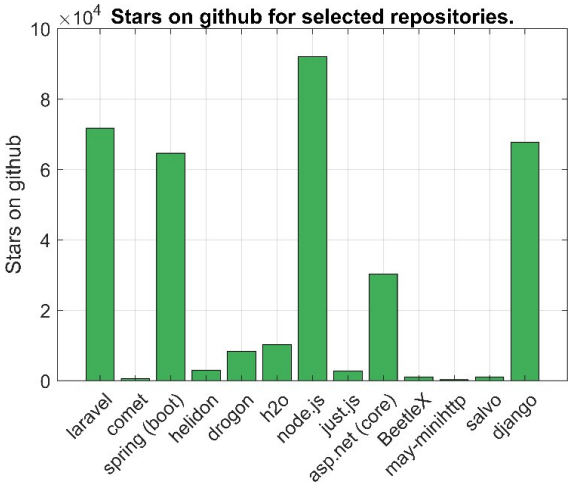


Figure 10. Showcase of number of stars for selected repositories on GitHub

According to insights.stackoverflow.com (Figure 11), the popularity of programming languages can vary over time. Some languages, such as Java

and Python, have consistently high levels of popularity and remain near the top of the chart throughout the plotted time period. Other languages, such as C++ and Rust, have more variable levels of popularity and fluctuate more in their ranking on the chart.

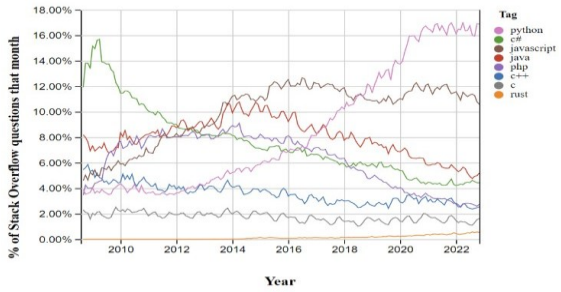


Figure 11. Schematic of percentage popularity of compared languages relative to Stack Overflow questions in their topic

It's important to note that this plot shows the popularity of programming languages based on the number of questions asked about them on Stack Overflow, which may not necessarily reflect their overall popularity or usage in the industry. Other factors, such as the number of job openings or the amount of code written in a language, could also be used to measure popularity.

The popularity of web frameworks can be influenced by the popularity of the programming language they are written in, but it is not directly tied to it.

For example, a web framework written in a popular programming language may have an advantage in terms of adoption and usage, since developers who are familiar with the language may be more likely to use the framework (Table 1). On the other hand, a web framework written in a less popular language may have a harder time gaining traction, even if it is well-designed and functional.

However, the popularity of a web framework is not solely determined by the popularity of the language it is written in. Other factors, such as the ease of use, performance, and features of the framework, can also play a role in its popularity (Table 2, 3 and 4).

In summary, the popularity of web frameworks can be influenced by the popularity of the programming language they are written in, but it is not directly tied to it. Other factors can also play a role in the popularity of a web framework.

Table 1. Schematic of percentage popularity of compared languages relative to Stack Overflow questions in their topic

Framework	Language	Updates	Single Query	Plaintext	Multiple Query	Json Serialization	Fortunes
drogon	c++	21,877	617,680	5,969,800	29,928	1,086,998	556,046
salvo	rust	23,733	631,785	1,928,951	33,700	1,082,630	542,547
just.js	javascript	24,454	673,201	6,982,125	34,620	1,526,714	538,414
may-minihttp	rust	24,192	642,348	7,023,484	34,493	1,546,221	520,976
asp.net (core)	c#	2,696	412,877	5,415,391	16,019	1,058,054	394,144
h2o	c	13,338	386,738	3,566,050	19,026	1,495,931	391,802
helidon	java	7,225	271,734	506,995	30,368	381,536	227,138
comet	php	7,556	230,193	460,879	19,347	366,200	195,551
BeetleX	C#	16,860	373,208	1,082	21,055	1,125,056	144,880
node.js	javascript	2,223	113,707	607,368	6,498	376,679	80,243
spring (boot)	java	7,128	106,783	185,720	15,763	139,448	22,478
django	python	754	19,402	79,188	1,396	73,479	15,038
laravel	php	1,611	9,366	15,012	3,267	14,926	8,437

Table 2. Content presents values calculated based on the weighted multiplicators that financial trading platform developer framework prefers

Framework	Language	Updates	Single Query	Plaintext	Multiple Query	Json Serialization	Fortunes	Score
just.js	javascript	122,270	3,366,005	13,964,250	103,860	3,053,428	538,414	21,148,227
may-minihttp	rust	120,960	3,211,740	14,046,968	103,479	3,092,442	520,976	21,096,565
drogon	c++	109,385	3,088,400	11,939,600	89,784	2,173,996	556,046	17,957,211
asp.net (core)	c#	13,480	2,064,385	10,830,782	48,057	2,116,108	394,144	15,466,956
h2o	c	66,690	1,933,690	7,132,100	57,078	2,991,862	391,802	12,573,222
salvo	rust	118,665	3,158,925	3,857,902	101,100	2,165,260	542,547	9,944,399
BeetleX	C#	84,300	1,866,040	2,164	63,165	2,250,112	144,880	4,410,661
helidon	java	36,125	1,358,670	1,013,990	91,104	763,072	227,138	3,490,099
comet	php	37,780	1,150,965	921,758	58,041	732,400	195,551	3,096,495
node.js	javascript	11,115	568,535	1,214,736	19,494	753,358	80,243	2,647,481
spring (boot)	java	35,640	533,915	371,440	47,289	278,896	22,478	1,289,658
django	python	3,770	97,010	158,376	4,188	146,958	15,038	425,340
laravel	php	8,055	46,830	30,024	9,801	29,852	8,437	132,999

Table 3. Content presents values calculated based on the weighted multiplicators that financial trading platform developer framework prefers

Framework	Language	Updates	Single Query	Plaintext	Multiple Query	Json Serialization	Fortunes	Score
may-minihttp	rust	120,960	1,927,044	14,046,968	172,465	7,731,105	520,976	24,519,518
just.js	javascript	122,270	2,019,603	13,964,250	173,100	7,633,570	538,414	24,451,207
drogon	C++	109,385	1,853,040	11,939,600	149,640	5,434,990	556,046	20,042,701
asp.net (core)	C#	13,480	1,238,631	10,830,782	80,095	5,290,270	394,144	17,847,402
h2o	c	66,690	1,160,214	7,132,100	95,130	7,479,655	391,802	16,325,591
salvo	rust	118,665	1,895,355	3,857,902	168,500	5,413,150	542,547	11,996,119
BeetleX	C#	84,300	1,119,624	2,164	105,275	5,625,280	144,880	7,081,523
helidon	java	36,125	815,202	1,013,990	151,840	1,907,680	227,138	4,151,975
comet	php	37,780	690,579	921,758	96,735	1,831,000	195,551	3,773,403
node.js	javascript	11,115	341,121	1,214,736	32,490	1,883,395	80,243	3,563,100
spring (boot)	java	35,640	320,349	371,440	78,815	697,240	22,478	1,525,962
django	python	3,770	58,206	158,376	6,980	367,395	15,038	609,765
laravel	php	8,055	28,098	30,024	16,335	74,630	8,437	165,579

Table 4. Content presents values calculated based on the weighted multiplicators that news webpage developer framework prefers

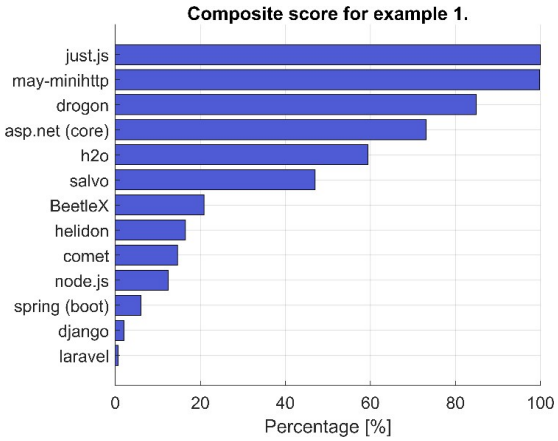
Framework	Language	Updates	Single Query	Plaintext	Multiple Query	Json Serialization	Fortunes	Score
just.js	javascript	48,384	3,211,740	35,117,420	68,986	3,092,442	2,604,880	44,143,852
may-minihttp	rust	48,908	3,366,005	34,910,625	69,240	3,053,428	2,692,070	44,140,276
drogon	c++	43,754	3,088,400	29,849,000	59,856	2,173,996	2,780,230	37,995,236
asp.net (core)	c#	5,392	2,064,385	27,076,955	32,038	2,116,108	1,970,720	33,265,598
h2o	c	26,676	1,933,690	17,830,250	38,052	2,991,862	1,959,010	24,779,540
salvo	rust	47,466	3,158,925	9,644,755	67,400	2,165,260	2,712,735	17,796,541
helidon	java	14,450	1,358,670	2,534,975	60,736	763,072	1,135,690	5,867,593
comet	php	15,112	1,150,965	2,304,395	38,694	732,400	977,755	5,219,321
BeetleX	C#	33,720	1,866,040	5,410	42,110	2,250,112	724,400	4,921,792
node.js	javascript	4,446	568,535	3,036,840	12,996	753,358	401,215	4,777,390
spring (boot)	java	14,256	533,915	928,600	31,526	278,896	112,390	1,899,583
django	python	1,508	97,010	395,940	2,792	146,958	75,190	719,398
laravel	php	3,222	46,830	75,060	6,534	29,852	42,185	203,683

Defined developer types:

1. A developer working on a real-time financial trading platform will likely focus on the Single Query test, as the platform needs to quickly respond to individual queries. This developer will likely use in-memory databases and other performance optimization techniques to minimize response time and improve overall system performance.

Assigned Values:

- DB updates: 5
- Single Query: 5
- Plaintext Response: 2
- Multiple Query: 3
- Json Serialization: 2
- Fortunes: 1

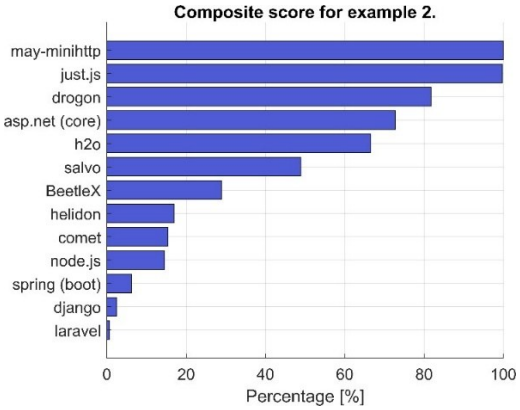


Composite scores calculated based on the weighted multiplicators that financial trading platform developer framework prefers

2. A developer working on an e-commerce platform will likely focus on the Multiple Query test, JSON Serialization test and Data Update test. The platform needs to process and return large amounts of data in JSON format, handle multiple queries at the same time and perform frequent updates to the product catalog. They may use document-based databases, pagination, asynchronous processing and efficient libraries for serialization and data updates.

Assigned Values:

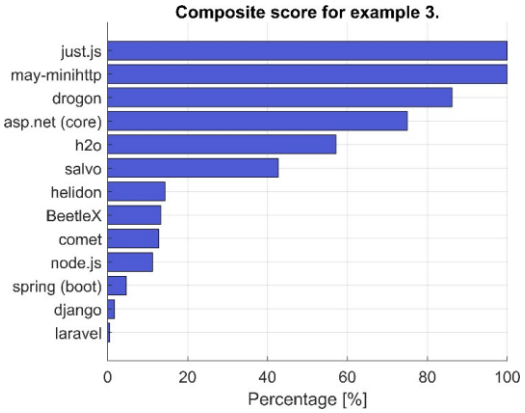
- DB updates: 5
- Single Query: 3
- Plaintext Response: 2
- Multiple Query: 5
- Json Serialization: 5
- Fortunes: 1



Composite scores calculated based on the weighted multiplicators that e-commerce developer framework preferes

3. A developer working on a web-based news website will likely focus on the Single Query test and Fortunes test, as the website needs to quickly respond to requests for cached content and display large amounts of data in a tabular format. This developer will likely use caching mechanisms, Content Delivery Network (CDN) and efficient libraries for displaying data in tables to improve performance and scalability of the website.
Assigned Values:

- DB updates: 2
- Single Query: 5
- Plaintext Response: 5
- Multiple Query: 2
- Json Serialization: 2
- Fortunes: 5



Composite scores calculated based on the weighted multiplicators that news webpage developer framework prefers

6. CONCLUSIONS

According to the benchmark tests that were conducted, different types of developers may find different frameworks to be the best fit for their needs. For real time financial trading platforms, Just.js is the recommended framework. This is because it provides exceptional performance on the Single Query test, which is crucial for this type of application. Additionally, Just.js is lightweight and easy to set up, making it a great option for high-performance, low-latency applications.

On the other hand, e-commerce platforms may benefit from using May-minihttp as their framework of choice. This is because it offers excellent performance on the Multiple Query test, JSON serialization test, and Data update test, which are the most important for this type of application. Furthermore, May-minihttp has good scalability, making it ideal for large-scale ecommerce platforms.

For web-based news websites, Just.js is again the recommended framework. It offers excellent performance on the Fortunes test, which is critical for this type of application. Additionally, Just.js is lightweight and easy to use, making it a great choice for high-performance, low-latency applications.

It is important to note that these results are based on the specific set of tests and configurations that were used. The best framework for a particular application will depend on various factors such as the specific requirements of the application, the development team's expertise, and available resources. Therefore, it is recommended to evaluate other factors before making a decision on which framework to use.

Additionally, it is important to consider the level of support and documentation available for the framework. A framework with a large and active community, as well as comprehensive documentation, can make development and troubleshooting much easier. Furthermore, looking into the framework's compatibility with other technologies and tools that you plan to use in your project can also be a crucial factor. For example, if you plan to use a specific database management system, it is important to ensure that the framework has good support for it.

Another important consideration is the framework's ability to handle security and data protection. It is important to ensure that the framework has built-in security features and that it is up-to-date with the latest security standards. This can

help to minimize the risk of data breaches and other security issues.

Finally, it is also worth evaluating the framework's ability to handle and manage resources, such as memory and CPU usage. A framework that is efficient in managing resources can help to minimize the risk of performance bottlenecks and crashes.

In conclusion, while the benchmark tests results can give a good idea of the performance of different frameworks, it is important to also consider other factors such as support and documentation, compatibility, security, and resource management before making a decision on which framework to use. It is also important to keep in mind that the best framework for a particular application will depend on the specific requirements and constraints of the project and development team.

REFERENCES

[1] Laaziri M., Benmoussa K., Khoulji S., and Kerkeb M.L.: *A Comparative study of PHP frameworks performance*, Procedia Manuf, vol. 32, 2019, pp. 864–871, 2019, doi: 10.1016/J.PROMFG.2019.02.295,

[2] Dwarampudi V., Dhillon S.S., Shah J., Sebastian N.J., and Kanigicharla N.S.: Comparative study of the Pros and Cons of Programming languages Java, Scala, C++, Haskell, VB .NET, AspectJ, Perl, Ruby, PHP & Scheme - a Team 11 COMP6411-S10 Term Report, doi: 10.48550/arXiv.1008.3431,

[3] Prokofyeva N., and V. Boltunova V.: *Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems*, Procedia Comput Sci, vol. 104, 2017, pp. 51–56, doi: 10.1016/J.PROCS.2017.01.059,

[4] Ghaemmaghami S.S.S., Emam S.S., and Miller J.: *Automatically inferring user behavior models in largescale web applications*, Inf Softw Technol, vol. 141, 2022, p. 106704, doi: 10.1016/J.INFSOF.2021.106704,

[5] TechEmpower: *TechEmpower Framework Benchmarks Round 21*, TechEmpower, 19-Jul-2022, [Online], Available: https://www.techempower.com/benchmarks/#section=datar21, [Accessed: 11-Dec-2022],

[6] Johnston A., and Huntley G.: Just-JS/just: *A very small V8 JavaScript runtime for Linux only*, GitHub, 09Oct2020, [Online], Available: https://github.com/just-js/just, [Accessed: 11-Dec-2022],

[7] Beetlex-Io: *Beetlex-IO/BeetleX: High performance dotnet core socket TCP communication components, support TLS, HTTP, HTTPS, WebSocket, RPC, Redis protocols, custom protocols and 1M Connections Problem Solution*, GitHub, 21-Feb-2020, [Online], Available: https://github.com/beetlex-io/BeetleX, [Accessed: 11- Dec2022],

[8] Dotnet: *Dotnet/ASPNETCORE: ASP.NET core is a cross-platform .NET framework for building modern cloudbased web applications on Windows, Mac, or Linux*, GitHub, 08-Jul-2014, [Online], Available: https://github.com/dotnet/aspnetcore, [Accessed: 11-Dec2022],

[9] DrogonFramework: *Drogonframework/Drogon: Drogon: A C++14/17/20 based HTTP Web Application Framework Running on Linux/MacOS/Unix/Windows*, GitHub, 11-Jun-2019, [Online], Available: https://github.com/drogonframework/drogon. [Accessed: 11Dec-2022],

[10] h2o: *H2O/H2O: H2O – the optimized HTTP/1, HTTP/2, HTTP/3 server*, GitHub, 18-Feb-2015, [Online], Available: https://github.com/h2o/h2o, [Accessed: 11-Dec2022],

[11] Helidon-Io: *Helidon-IO/Helidon: Java libraries for writing microservices*, GitHub, 15-Sep-2018, [Online], Available: https://github.com/helidon-io/helidon, [Accessed: 11-Dec-2022],

[12] Laravel: *Laravel/Laravel: Laravel is a web application framework with expressive, elegant syntax. we've already laid the foundation for your next big idea - freeing you to create without sweating the small things*, GitHub, 14-May2017, [Online], Available: https://github.com/laravel/laravel, [Accessed: 11-Dec-2022],

[13] Nodejs: *Nodejs/node: Node.js JavaScript runtime*, GitHub, 05-Sep-2017, [Online], Available: https://github.com/nodejs/node, [Accessed: 11-Dec-2022],

[14] Gotsuliak S., and Anewenah W.: *Gotzmann/Comet: Modern PHP framework for Building Blazing Fast Rest APIs and microservices*, GitHub, 08-May-2020. [Online]. Available: https://github.com/gotzmann/comet. [Accessed: 11-Dec-2022].

[15] Spring-Projects: *Spring-projects/spring-boot: Spring boot*, GitHub, 09-May-2018, [Online], Available: https://github.com/spring-projects/spring-boot, [Accessed: 11-Dec-2022],

[16] Django: *Django/Django: The web framework for perfectionists with deadlines*, GitHub, 03-Nov-2005, [Online], Available: https://github.com/django/django, [Accessed: 11-Dec-2022],

[17] Salvo-Rs: *Salvo-rs/salvo: Salvo is a powerful and simplest web server framework in Rust World*, GitHub, 11Apr2022, [Online], Available: https://github.com/salvors/salvo, [Accessed: 11-Dec-2022],
[18] Huang X.: *Xudong-Huang/MAY_MINIHTTP: Mini HTTP implemented on top of May*, GitHub, 16-Jan-2018. [Online], Available: https://github.com/XudongHuang/may_minihttp, [Accessed: 11-Dec-2022],
[19] Stack Overflow: *Stack Overflow Most Popular Languages*, Stack Overflow Insights, 11-Dec-2022, [Online], Available: https://insights.stackoverflow.com/trends, [Accessed: 11Dec-2022].