electronics

MDPI

Article

Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks

Jakub Swacha * and Artur Kulpa

Department of Information Technology in Management, University of Szczecin, 71-004 Szczecin, Poland; artur.kulpa@usz.edu.pl
* Correspondence: jakub.swacha@usz.edu.pl

Abstract: Since the emergence of the first web development frameworks at the turn of the 21st century, many of them have made a name for themselves and rose to wide popularity only to be later ousted by newer frameworks and sometimes even fall to obscurity. In our paper, we would like to depict the last fifteen years of the changing popularity of web development frameworks by observing the number of newly created repositories in open-source projects hosted at GitHub that were based on them as well as the number of questions posted on Stack Overflow regarding the respective frameworks. We analyze the correspondence between these two indicators and put them in the context of the popularity of the programming languages that the respective frameworks are based on, as measured by the TIOBE Programming Community index, in an effort to check whether one source could be used to forecast another.

Keywords: web development; web application frameworks; web frameworks; web programming; web technology evolution

Citation: Swacha, J.; Kulpa, A. Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks. *Electronics* **2023**, *12*, 3563. https://doi.org/10.3390/electronics12173563

Academic Editors: Manuel Palomo-Duarte, Marko Horvat, Igor Mekterović and Juan Antonio Caballero-Hernandez

Received: 26 July 2023
Revised: 18 August 2023
Accepted: 21 August 2023
Published: 23 August 2023

1. Introduction

With the current proliferation of electronic commerce [1], electronic services [2], and electronic government [3], more and more web applications are being developed. For a number of reasons, including (1) the fact that many web applications provide similar or even the same functions applied to different usage domains, which justifies the use of a common skeleton for all such applications; (2) the fact that the lingua franca of the programs to be run on the client side, which is now JavaScript, lacks the rich native standard library as provided by, e.g., .NET languages; and (3) the specific nature of a notable part of the operations that the web applications perform (such as manipulating an HTML document by using the Document Object Model, DOM), which tends to require a lengthy form by using pure JavaScript (or any classic programming language for that matter), already at the turn of the 21st century, the first web development frameworks began to emerge [4]. These software development tools, also known as web application frameworks or web frameworks, allow developers to build web applications in a faster and easier way by providing a basic model, as well as a set of relevant APIs, libraries, and extensions [5] (p. 865).

The benefits of adopting web frameworks are multiple and include:

1. A reduced development time and increased productivity: web frameworks provide ready-made components and code templates [6] that let developers avoid implementing and testing repetitive code, addressing one of the principles of good programming—DRY (do not repeat yourself) [7].
2. Standardization: Web frameworks come with design patterns imposing certain standards in application design and their way of operation [8], which allows developers to focus on developing functionality rather than solving architectural problems, also leading to a better understanding and consistency in code. This moreover facilitates

teamwork, allowing different developers to find their way around the project more easily. New team members can join the project faster. In addition, it supports code reuse [9].
3. Easier portability: by introducing abstraction and separating the logical layer from the technical layer, an application based on the framework can be ported to other platforms or environments with relative ease [10].
4. A responsive design: many frameworks provide built-in tools and components for developing responsive user interfaces, adapting to different screen sizes and devices [11].
5. Improved security: Many frameworks provide built-in security mechanisms, such as protection against XSS (Cross-Site Scripting) or CSRF (Cross-Site Request Forgery) attacks [12]. This is especially important for inexperienced developers.
6. Scalability: As a rule, frameworks are built and developed with scalability in mind [6]. The design patterns they are based on make it easy to expand the application as its complexity increases.
7. Improved sustainability: Frameworks that are popular are regularly updated [13] to respond to new fads, technologies to be supported, or recently identified security vulnerabilities. This ensures that the costs incurred for ongoing software maintenance are minimized.
8. Community support: popular frameworks have active user communities, which are not only sources of specialized knowledge, tutorials, and ready-to-use solutions but are also individuals that can be asked to help solve programming problems, and this can be asked via, e.g., Q&A websites such as Stack Overflow [14].

Over time, web frameworks have become an integral part of the development of web applications and services [9] (p. 329). In fact, a plethora of frameworks have been developed differing in features, advantages, disadvantages, and supported programming languages, as well as the application domains for which they are suitable [9] (p. 330).

This state of things has not gone unnoticed by the research community, and as its obvious consequence, a number of scientific papers dealing with this problem area have emerged. By querying Google Scholar, Web of Science, and Scopus for "Web Development Frameworks", we were able to identify the main vein of relevant research, which is dedicated to comparing the respective frameworks with regard to various aspects. The published comparisons differ in the basis for the selection of considered frameworks and in the considered comparison criteria. The latter may include objective traits of the respective frameworks (such as their scope of use, applied architectural patterns, or supported functionality), empirical performance measurements, or subjective evaluations of their usability for certain purposes.

For instance, Kaluža et al. compare three front-end (Angular, Vue.js, and React.js) [13] and four back-end web development frameworks (Laravel, Ruby on Rails, Django, and Spring) [15] in terms of various usability aspects, indicating the frameworks which are better suited for the considered application areas. Heitkötter et al. compare only mobile web frameworks (jQuery Mobile, The M Project, Sencha Touch, and Google Web Toolkit + mgwt) in terms of their usability for mobile web development and performance. Mishra and Srivastava compare ten frameworks (Ruby on Rails, JSF, Struts, Django, Angular, Laravel, Spring, React, Flask, and Symfony) in terms of their support for best practices in web development and seven frameworks (Rails, Django, Node.js, Laravel, Spring, Flask, and Symfony) in terms of their performance in four experiments covering simple-use cases to find significant differences in the latter aspect [16]. The support for the best practices of web development is also the focus of the comparison by Salas-Zárate et al., who consider 24 such practices in their comparison of eight frameworks (JSF, Struts, CakePHP, Grails, Ruby on Rails, Django, Catalyst, and Lift), indicating one particular framework as the best [6]. Aborujilah et al. investigate only the aspect of the support for web security, comparing the relevant features of five frameworks (Laravel, Spring Boot, Django, Ruby on rails, and ASP.NET Core) [12]. Ollila et al. compare five web development frameworks: Angular, React, Vue, Svelte, and Blazor in terms of their rendering speed to find significant

differences depending on the application scale [17]. In comparison, Laaziri et al. limit the scope of their analysis to just three PHP frameworks (Laravel, Symfony, and CodeIgniter), yet they not only consider several performance variables (such as the number of served requests per second and the response time, for both of which significant differences among frameworks are visible), but they also compare the non-performance-related aspects by using the well-established QSOS method [5]. Kopyl et al. go one step further and compare only two frameworks (ASP.NET Core and Spring Boot) in terms of their usability and performance, exposing notable differences between them [18].

The prior work on web frameworks also comprises other types of papers, such as those proposing a new framework addressing the deficiencies of the existing ones (e.g., [19]) or indicating optimizations which improve the performance of the popular frameworks (e.g., [20]). In contrast, Muhammed et al. acknowledge that organizations operate in different domains and may have different priorities and constraints, so there could be no one-size-fits-all approach with regard to web frameworks; therefore, they propose an automatic tool for selecting a web framework by using a set of criteria and developer preferences based on the Analytic Hierarchy Process (AHP) [9].

The basis for inclusion in the comparison that is usually given in the sources discussed above is the most widely used (see, e.g., [5,13,15–17]) or the best known (see, e.g., [6]); however, there is no agreed way of understanding how these should be measured. Moreover, despite the fact that it is well known that web development is evolving rapidly, even considering the generally quick pace of Information Technology development [21] (p. xxi), so far, no one has investigated how the popularity of web frameworks among the web developers using them changed over time. Note that this is in contrast to the well-established interest in the much slower evolution of the popularity of programming languages (see, e.g., [22–25]).

This calls attention primarily for practical reasons, because by being aware of how the web framework popularity evolves, one can distinguish the frameworks that used to be popular (but are no more), those that are rising in popularity (which is still relatively low), and those that are currently popular among the frameworks described as popular. This knowledge becomes valuable when a new web development project is started and a decision is made in which web technology, time, and resources will be invested. The choice of a popular framework comes with a bonus as the greater the framework's popularity, usually the greater the practical advantages of using it, such as the assurance of reliability and security, more frequent updates and functionality extensions, or better community support. It is also important for skill and talent management, helping with directing efforts on talent acquisition and retention, as well as learning new technological skills from developers. On the research side, there is an open problem of measuring the popularity effectively; in particular, whether and which of the available and easily-accessible indicators can be used to predict the changes in the web framework popularity. This is also a necessary base for further research on the reasons why these changes happen.

Addressing this research gap is the motivation of the research aims set for this work: (1) to provide an overview of how the popularity of the currently most popular web development frameworks varied in time by using two different sources of data; (2) to find to what extent the two considered sources of data provide similar results and (3) whether one can be used to forecast the other; (4) to find to what extent the web framework popularity corresponds to the popularity of the programming languages they are based on; and to avoid leaving the reader unaware of the characteristics of the frameworks whose popularity is analyzed, the last aim is (5) to compare them in a number of aspects, at a general level, and by using objective criteria.

In pursuit of the research aims stated above, the following research questions are stated:

RQ1. Do web frameworks undergo strong and rapid changes in their popularity?
RQ2. Are the two considered indicators of web framework popularity consistent with each other?

RQ3. Can one of the two considered indicators of web framework popularity be effectively used to forecast the other?
RQ4. Are the considered indicators of web framework popularity in correspondence with the popularity of the programming language they are based on?
RQ5. What are the key similarities and differences in the characteristics of the currently popular web frameworks?

The structure of this paper is as follows: In the following section, we present the materials and methods used to perform our study. Then, the obtained results are presented with regard to the comparative analysis of the considered frameworks (addressing research aim 5; Section 3.1), the popularity of the considered frameworks according to the two considered sources (addressing research aim 1; Sections 3.2 and 3.3), the analysis of correspondence between the two considered sources (addressing research aims 2 and 3; Section 3.4), and the analysis of correspondence between the popularity of web frameworks and programming languages (addressing research aim 4; Section 3.5). The final section concludes the main findings.

2. Materials and Methods

2.1. Data Sources

Studies, such as the one reported here, can only be performed by assuming some model of popularity. The model we developed for this purpose is depicted in Figure 1 (note that the purpose of the chart is purely to indicate the overlapping of the respective shapes, so their size depicted in the chart is not based on any data). By using widely used web frameworks, we understand the frameworks indicated by shape D. The problem is it would be impossible to analyze the evolution of the popularity of respective frameworks because there were no surveys performed regularly to gather such data; we only identified a single source reporting the results of such a survey from a single point in time (2022) [26]. If we pursued the analysis of the frameworks attracting the most interest (shape A), a good source to consider would be a source such as Google Trends [27], a Google service that allows one to compare the popularity of search queries in the Google Search engine, indicated by shape B. But, this cannot be treated as a fair enough estimate of shape D, as it includes many queries coming from people who are not framework users but merely want to learn something about it, such as users of other frameworks, computer science students, or people who just read the word and had no idea what it meant.

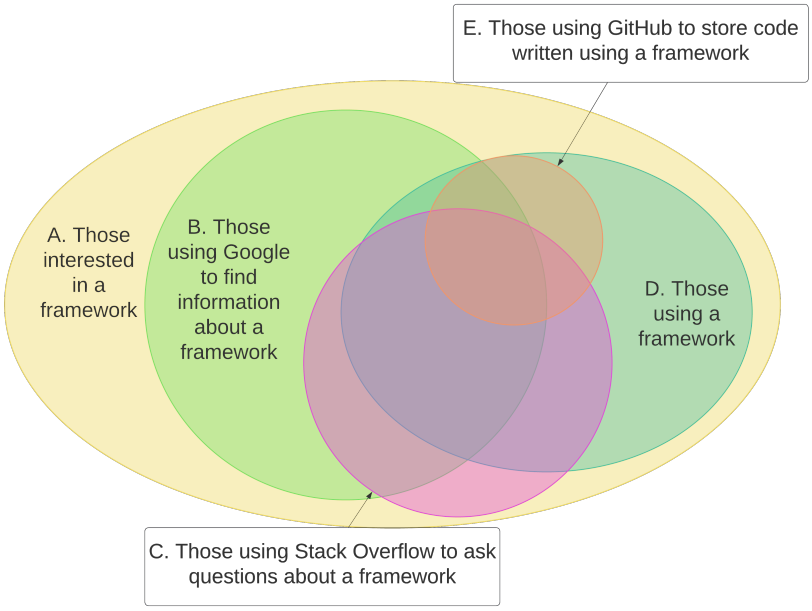

Figure 1. Different scopes of web framework popularity.

For these reasons, we based our study on two other sources. The first one is GitHub [28], a web hosting service for storing and distributing software projects by using the Git version control system. In Figure 1, the data coming from this source are depicted by shape E. The key advantage of this source is that it is free of false positives: it contains only proven users of a given framework (the proof being the code they committed). On the downside, it includes only some of them—those who committed their work to GitHub. This not only omits professional programmers who use the framework to develop closed-source software only but also amateur programmers and students who have no reason to publish their code online. Although we are interested in the relative popularity of frameworks, considering only a specific part of a user base, defined in the same way for all considered frameworks, seems fair; the disadvantage of this source is that the popularity of GitHub among developers using different frameworks is not guaranteed to be the same or constant.

The second source of data we used is Stack Overflow [14], a social networking site dedicated to software developers and programmers in particular. Its main purpose is to exchange knowledge between programmers in a question-and-answer (Q&A) format: people who have a problem with some issue post a question and then receive answers from other users. The data provided by Stack Overflow give an estimate of the population delimited by shape C. It represents the people who made an effort to ask a question on a Q&A forum about a framework. Posting a question indicates a serious interest in a given framework, and it is therefore reasonable to assume that the users constitute a much larger part of the population shown as shape C than shape B. On the other hand, not everyone having a serious interest in a given framework will post a question about it on a Q&A forum, as there are other possible sources of needed information (such as books, other websites, and even the responses to the questions that have already been posted on the very same forum).

The difference in scope of these two sources is not the only reason for including both of them. While the assumption implied by the presented model is that the data from both are to some extent supplementary, there is a rationale to consider them as complementary to some extent as well as they seem to reflect populations of somewhat different traits:

- Beginner programmers, learning a given framework or writing their first software using it, are more likely to post questions on a Q&A website than experienced programmers;
- Conversely, experienced programmers are more likely to commit code to an open-source repository than beginner programmers;
- A framework that is difficult to master and is poorly documented is more likely to cause individuals to post questions on a Q&A website than one which is easy to use and well documented;
- Conversely, such a difficult-to-master and poorly documented framework is less likely to be used to develop open-source software.

With regard to the fourth research aim, one more source of data had to be used as an indicator of the popularity of programming languages. For this purpose, we chose the TIOBE Programming Community index [22], which has been maintained since 2001 and is updated once a month. Its ratings are based on a synthetic aggregate comprising, e.g., the number of skilled engineers worldwide, courses, and third-party vendors, and it uses popular search engines including Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube, and Baidu as its input data sources.

As the fifth research aim is of a different character, the relevant part of our study was based on the existing literature and technical specifications rather than on raw data.

2.2. Scope of the Analysis

Before the data could be collected, the scope of the analysis had to be defined, both in terms of the analyzed time span and the set of considered frameworks.

Regarding the time limits, we decided to cover the last 15 years (2008–2022), assuming there is little sense in covering the very early days of web framework development when most of the key players of today's scene were still to appear (cf. [4]). Regarding the choice

of the frameworks included in the comparison, as we were more interested in the changes in the popularity of the frameworks known today rather than those known back in 2008, we chose the ten frameworks reported as the most popular among developers worldwide in 2022 [26]. Note that in the final selected top ten list, we omitted frameworks that were alternative versions or additions to other frameworks (e.g., Express as node.js was included, ASP.NET Core as ASP.NET was included, Next.js as React.js was included, and Angular.js as Angular was included). Similarly, when matching frameworks to programming languages, we did not differentiate the JavaScript and TypeScript frameworks (all of them were assigned to JavaScript).

Table 1 lists the names of the considered frameworks along with the associated GitHub topics, Stack Overflow tags (see the next subsection for an explanation of these two columns), and names of the programming languages which these frameworks support and/or are written in.

Table 1. Web development frameworks covered in the survey.

Framework	GitHub Topic	Stack Overflow Tag	Programming Language
Node.js	nodejs	node.js	JavaScript
React.js	react	reactjs	JavaScript
jQuery	jquery	jquery	JavaScript
Angular	angular	angular	JavaScript
Vue.js	vue	vue.js	JavaScript
ASP.NET	aspnet	asp.net	C#
Django	django	django	Python
Flask	flask	flask	Python
Laravel	laravel	laravel	PHP
Ruby on Rails	rails	ruby-on-rails	Ruby

2.3. Data Acquisition

The data required to perform the evolution analysis were acquired in an automated way from the three sources described in Section 2.1. With regard to GitHub [28], the projects published therein are categorized by using *topics* assigned by the project owners and, as a rule, the software developed by using or for a particular web development framework is mentioned among the topics. This makes it easy to find and count projects and, as a consequence, newly created repositories that are relevant to a particular web development framework. We chose the number of newly created repositories as the popularity indicator based on GitHub, as in our belief, it gives a highly reliable estimate of whether a specific framework is alive or dead.

GitHub provides an API for downloading detailed data about projects as well as aggregate data. In order to retrieve data on the new repositories in GitHub, the following API query was used:

```
https://api.github.com/search/repositories?
    q=topic:{topic}
    +created:{date_from}..{date_to}
```

where

- `{topic}` is the GitHub topic relevant to the respective web framework (see Table 1),
- `{date_from}..{date_to}` is the period of time for which new repositories were created.

In response to the above query, GitHub returns JSON-serialized data containing (in the `total_count` field) the number of newly created repositories in the period specified in the query.

Having retrieved the data, we noticed an unrealistically low number of repositories reported for the ASP.NET framework (104 for the whole period). After scrutiny, it was found out that ASP.NET is listed by GitHub as one of the programming languages (although

it is not one) and, as a consequence, most of the relevant GitHub repositories are marked as written in the ASP.NET language instead of being tagged with the "aspnet" topic, as is the practice with the repositories developed by using other web frameworks. To overcome this problem, a distinct API query was used to retrieve data on the ASP.NET framework repositories (25,430 of them were found by using this approach for the whole analyzed period):

```
https://api.github.com/search/repositories?
    q=language:asp.net
    +created:{date_from}..{date_to}
```

where

- `{date_from}..{date_to}` is the period of time for which new repositories were created.

With regard to Stack Overflow [14], our query targeted the tags that the question authors use to denote the context of their questions (and thus to increase the probability of obtaining a relevant answer). By referring to the tag denoting a particular web development framework, we were able to find and count the questions related to it. We chose the number of questions as the popularity indicator based on Stack Overflow, as in our belief, it gives a more reliable estimate of whether a specific framework is popular compared to, e.g., the number of posts (i.e., counting answers as well), in which case one question spanning a long discussion would matter more than another one which received a short and sharp answer that quickly closed the discussion.

Stack Overflow belongs to the Stack Exchange network of Q&A websites on topics in diverse fields which, like GitHub, provide an API allowing one to retrieve both detailed and aggregate data about questions and answers. In order to retrieve data on the Stack Overflow questions, the following API query was used:

```
https://api.stackexchange.com/2.3/questions?
    fromdate={date_from}
    &todate={date_to}
    &tagged={tag}
    &site=stackoverflow
    &filter=!Nfq4bHcltk388
```

where

- `{tag}` is the Stack Overflow tag relevant to the respective web framework (see Table 1),
- `{date_from}..{date_to}` is the period of time for which the questions were counted.

In response to the above query, Stack Overflow returns JSON-serialized data containing (in the `total` field) the number of questions made in the period specified in the query, which were assigned the tag specified in the query.

As for the third data source used, the TIOBE Programming Community index, the data were scraped directly from the history chart published there, presenting the popularity measurements since June 2001 [22].

The data retrieval from all three sources listed above was performed with scripts written in Python and using its *requests* library. In the case of the TIOBE Programming Community index, additionally, Python's *Beautiful Soup* library was used for the web scraping.

The retrieved data were not cleared in any way. We are aware of some awkward values; for example, there was 1 newly created repository in 2011 for projects assigned to the Vue.js topic (whereas Vue.js was released only in 2014). We did not investigate the potential cause (which in this particular case could be, e.g., a misplaced topic, a project that migrated to a given framework, or a project developed earlier by using another technology), and we ignored such cases as an unavoidable measurement error because their impact on the analysis was negligible (for instance, the total number of newly created repositories for Vue.js projects from 2014 to 2022 was 47,459).

2.4. Data Analysis

The methods of data analysis were chosen according to the character of the respective research aims. The primary method used to attain research aims 1–4 was a visual analysis. Although this is one of the oldest forms of data analysis, it has not fallen to obsolescence—rather, the opposite is true as the availability of new data visualization technology (see, e.g., [29] (chapter 9)) enabled the fast and easy generation of even complex charts, whereas its key advantage of allowing one to quickly yield conclusions and hypotheses remains valid [30] (pp. 15–16).

As the first research aim is exploratory in character, we decided that a visual analysis was a sufficient means to address it. With regard to attaining research aims 2–4, we conducted a visual analysis by using statistical methods; in particular, a cross-correlation was used to visualize the correspondence of data coming from different sources with respect to different time lag [31], whereas the Granger causality test [32] was used to determine whether the data coming from one of the sources can be useful in forecasting the other.

All the charts presented in this paper were generated by using the Python *matplotlib* library [33], with all necessary data processing performed with the *Pandas* and *statsmodels* Python libraries [29] (chapters 5 and 12). A cross-correlation was calculated by using the *statsmodels*' *ccf* function (with the parameters *adjusted = False, fft = True*), whereas the Granger causality test was performed by using the *statsmodels*' *grangercausalitytests* function (with the test set to *ssr_chi2test* and the maximum lag set to 3).

In the pursuit of the fifth research aim, a comparative analysis of the considered web frameworks was performed, and its results were then visualized in a tabular form.

3. Results and Their Discussion

3.1. Multiaspectual Comparison of the Widely Used Web Frameworks

In order to address the fifth research aim, first, the comparison criteria had to be chosen for the analysis. After considering the criteria used in prior work [5,6,12,16,18], we devised our own set of traits which spanned a number of aspects, including those that provide the most general yet apt description of a given framework (related to the application domain and software architecture) and those that illustrate how well a given framework addresses the common issues of modern web development (related to internationalization and security). The eventually obtained list includes the following:

- The Key Area—the web development area which the given framework supports (either solely or most typically),
- The Software Layer—the side of the client–server model at which the framework is used,
- The Primary Architectural Pattern—the primary architectural design pattern used in developing software with a given framework,
- Primary Data Storage—the place typically used for permanent data storage when using a given framework,
- Internationalization—the way the support for developing localized and multilanguage versions of software is provided in a given framework,
- Security—the way the support for protecting against the most common hacking attacks is provided.

In appreciation of the role of security in most usage domains of web applications, including electronic commerce, electronic services, and electronic government, the most detailed analysis addressed the security aspect, with the following types of attacks having been considered:

- XSS (Cross-Site Scripting)—the web application spreads malicious code injected into it by using a web form or a modified URL [34],
- Clickjacking—an invisible iframe is placed on a page with another visible object above it to capture the click event [35],

- CSRF (Cross-Site Request Forgery)—end users are forced to execute unwanted actions in a web application in which they are authenticated [36],
- DDoS (Distributed Denial of Service)—an attacker aims to block access to a server or service by flooding it with a large number of bogus requests [37],
- Remote Code Execution—an attacker remotely executes arbitrary code in the programming language in which the web application was written and/or a command of the operating system hosting the application [38].

The result of the comparative analysis applying all the criteria listed above is presented in Tables 2 and 3. Note that next to each included framework and programming language is the year of its first release.

Table 2. Comparison of web frameworks based on JavaScript.

		JavaScript (1995)				
		Node.js (2009)	**React.js (2013)**	**jQuery (2006)**	**Angular (2016)**	**Vue.js (2014)**
Key Area		Runtime Environment	GUI	JS booster GUI	GUI	GUI
Software Layer		Back end (Front-end lib)	Front end	Front end	Front end	Front end
Primary Architectural Pattern		Event driven	Flux	Event driven	Component based	MVVM
Primary Data Storage		SQL/NOSQL	Browser local storage	Browser local storage	Browser local storage	Browser local storage
Internationalization		Built in	Add. library	Add. library	Built in	Built in
Security	XSS	Add. library	Built in	No	Built in	Built in
	Clickjacking	No	No	No	No	No
	CSRF	Add. library	No	Add. library	Built in	No
	DDoS	No	Not applicable	Not applicable	Not applicable	Not applicable
	Remote Code Execution	No	No	No	No	Built in

Source: own work based on the web framework documentation as follows: Node.js [39], React.js [40], jQuery [41], Angular [42], and Vue.js [43].

Table 3. Comparison of web frameworks based on other languages (C#, Python, PHP, and Ruby).

		C# (2000)/.NET	**Python (1991)**		**PHP (1995)**	**Ruby (1995)**
		ASP.NET (2002)	**Django (2005)**	**Flask (2010)**	**Laravel (2011)**	**Ruby on Rails (2004)**
Key Area		Full-stack framework	Full-stack framework	Microframework	Full-stack framework	Full-stack framework
Software Layer		Back end	Back end	Back end	Back end	Back end
Primary Architectural Pattern		Page centric/MVC	MVC (MTV)	MVC	MVC	MVC
Primary Data Storage		SQL/ORM	SQL/ORM	SQL/ORM	SQL/ORM	SQL/ORM
Internationalization		Built in	Built in	Add. library	Built in	Built in
Security	XSS	Built in	Built in	Add. library	Built in	Built in
	Clickjacking	Built in	Built in	Add. library	Built in	Built in
	CSRF	Built in	Built in	Add. library	Built in	Built in
	DDoS	No	Built in	Add. library	Add. library	No
	Remote Code Execution	Built in	Built in	No	Built in	Built in

Source: own work based on the web framework documentation as follows: ASP.NET [44], Django [45], Flask [46], Laravel [47], and Ruby on Rails [48].

Looking at the comparison presented in Tables 2 and 3, one can observe the following:

- Although half of the analyzed most popular web frameworks are JavaScript frameworks (which is understandable considering the unique role of this particular language in web application front-end development as it is the only programming language natively supported by almost all contemporary web browsers), the remainder are based on other programming languages; note that although ASP.NET can employ any .NET language, we link it to C#, as recently most ASP.NET applications are developed in this programming language;
- Among the analyzed most popular web frameworks, half are dedicated for full-stack web development, four are specialized for web GUI development (including jQuery, which was conceived as a JavaScript-boosting framework), and one (node.js) provides a full-fledged runtime environment;
- Among the analyzed frameworks, five are back-end frameworks, four are front-end frameworks, and one is a back-end framework which can also be used for front-end development;
- The analyzed frameworks use different primary architectural design patterns, with MVC (Model View Controller [49]) being the most popular (4 of 10 frameworks, including its slight variation known as Model Template View [50]), followed by the event-driven architecture [51] (2 frameworks), the other being Flux [52] and MVVM (Model–View–ViewModel) [53], the component-based architecture [54]; the last case is ASP.NET: originally based on page-centric architecture, it now can also be used to implement the MVC-based applications [55];
- The Primary Data Storage used in the analyzed frameworks is a clear consequence of their front-end or back-end point of application;
- All the analyzed frameworks provide support for developing localized and multilanguage versions of software; however, seven of them have it out of the box, and the three remaining require a dedicated additional library for this purpose;
- The support for addressing security concerns greatly varies among the analyzed frameworks, being much higher for the back-end frameworks than for the front-end frameworks (note that "Not applicable" indicates that a front-end framework cannot help with certain types of attacks for which only a server-side defense mechanism would be adequate).

3.2. Evolution of Web Framework Popularity According to GitHub

Figure 2 presents the number of repositories newly created on GitHub whose topic indicated respective web development frameworks in each year of the analyzed period (2008–2022).

As we can observe, there is a vast difference in the yearly number of newly created repositories for projects developed in their respective frameworks: the number of newly created repositories for Ruby on Rails barely surpassed 11 thousand for the entire analyzed period, whereas the number of newly created repositories for both React.js and Node.js surpassed this amount yearly for six consecutive years (from 2017 on) and Django achieved it in one year (2022). The highest yearly records were attained by these three frameworks in 2022, with React.js peaking at almost 85 thousand, Node.js at almost 44 thousand, and Django at almost 12 thousand.

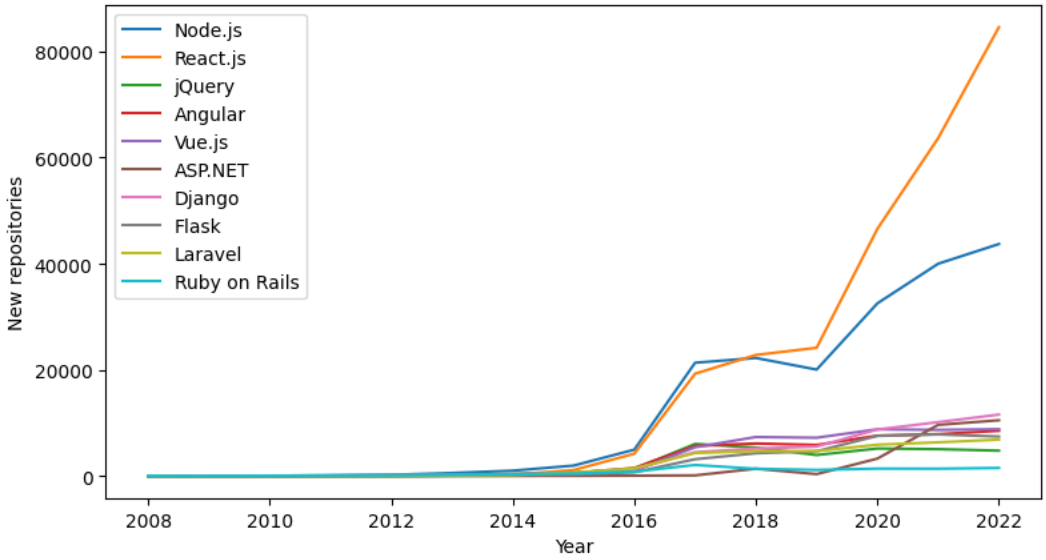

Figure 2. Frameworks by the number of newly created repositories on GitHub.

During the analyzed period, there were notable changes in the popularity of the frameworks measured with the number of newly created GitHub repositories. Looking at the chart in Figure 2, we can identify the following categories of web development frameworks:

- Rising stars that quickly gained a lot of popularity in recent years (React.js and Node.js);
- Steady climbers whose popularity kept growing for a number of years albeit at a far slower pace than that of the rising stars (Django, Angular, and Laravel);
- One which passed through ups and downs but recently attained its peak level (ASP.NET);
- Plateau occupiers whose rise to popularity happened a few years ago and stayed high since then (Vue.js, Flask, and jQuery);
- One whose popularity remained much below the top frameworks throughout the whole analyzed period and is clearly past its peak time now yet is still popular enough to be qualified in the top ten (Ruby on Rails).

It must be noted that the analysis can hardly cover the period before 2014 as GitHub reports a very low number of new repositories for those years, even for frameworks that existed already then. It is not obvious if the cause of this is the low popularity of GitHub among web developers of that time or missing metadata for the repositories created in that period.

3.3. Evolution of Web Framework Popularity According to Stack Overflow

Figure 3 shows the frameworks' popularity measured with the number of the Stack Overflow questions whose tags indicated respective web development frameworks in each year of the analyzed period (2008–2022).

The highest number of questions in one year was registered for jQuery: 145,817 in 2013, and it surpassed 100 thousand in each year from 2012 to 2016. No other framework has been that popular on Stack Overflow; the closest was React.js (94 thousand in 2022, and over 80 thousand each year from 2020 to 2022). There were four frameworks whose number of newly created repositories passed 40 thousand in at least one year: Node.js (2016–2022), Angular (2017–2020), ASP.NET (2011–2014), and Ruby on Rails (2013–2015). The only of the considered frameworks that did not pass 10,000 questions in any year of the analyzed period was Flask.

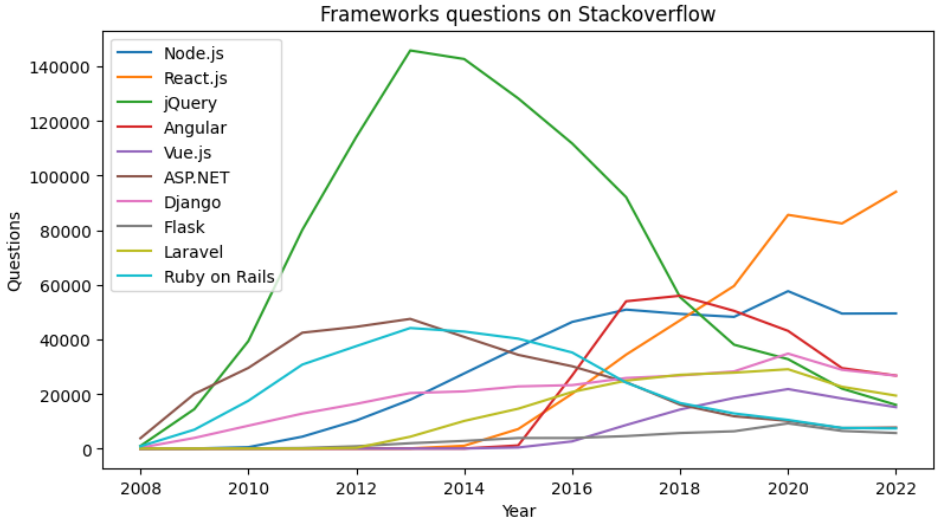

Figure 3. Frameworks by the number of questions posted on Stack Overflow.

Looking at the chart in Figure 3, we can observe that the only web development framework whose popularity consistently kept increasing since its introduction was React.js. All the remaining frameworks are below their top-year levels, though the pace of the decrease is various:

- Three frameworks attained very high or high popularity but retained only a small fraction of it: jQuery (whose number of questions in 2022 was merely 11% of its top level of 2013), ASP.NET (16% of its 2013 level), and Ruby on Rails (17% of its 2020 level);
- Two frameworks attained high popularity and retained a good part of it: Node.js (86% of its 2020 level) and Django (77% of its 2020 level);
- The remaining frameworks are between the two above categories as their drop in the number of questions is not drastic, though still easily noticeable: Vue.js (70% of its 2020 level), Laravel (67% of its 2020 level), Flask (62% of its 2020 level), and Angular (48% of its 2018 level).

The observation that a decreasing number of questions was recorded for most frameworks indicates the possibility of a systematic bias of this indicator because the more questions that were already posted on a Q&A site, the more can be learned from the answers given to them, and therefore the lesser the need to post a new question. Nonetheless, despite over one million questions asked about jQuery prior to 2022, there were still 15,996 new ones asked in 2022, whereas Flask, which accumulated only about 45 thousand questions prior to 2022, received only 5683 in that year. This means that the number of posted questions can still be used as an alternative popularity indicator, because in practice, a thriving community using a given web framework can still generate new questions despite a large amount of them having been answered in the past.

3.4. Comparing Web Framework Popularity between GitHub and Stack Overflow

The chart in Figure 3 is strikingly different from the chart in Figure 2. To make the differences more visible, we combined the data from both sources in Figures 4 and 5 (two charts were used instead of one to ensure the data remain readable). The solid lines represent the number of newly created repositories in GitHub (left scale), whereas the dashed lines represent the number of Stack Overflow questions (right scale).

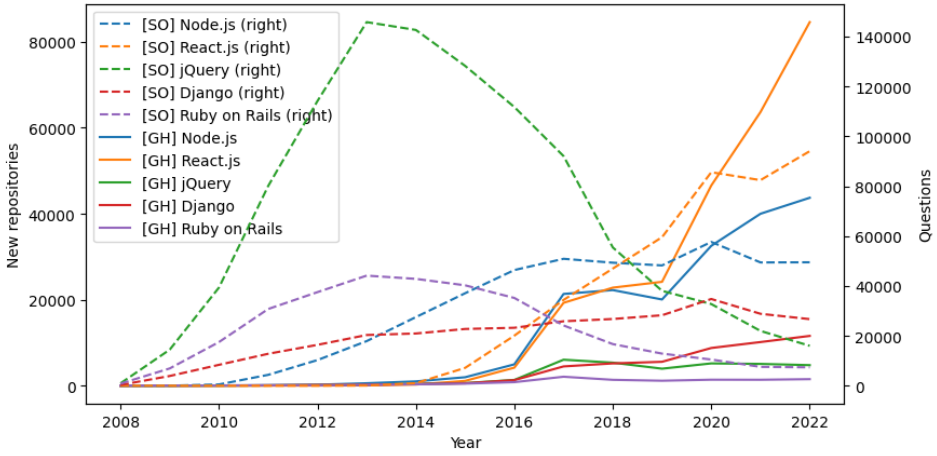

Figure 4. Comparison of GitHub and Stack Overflow popularity (part 1).

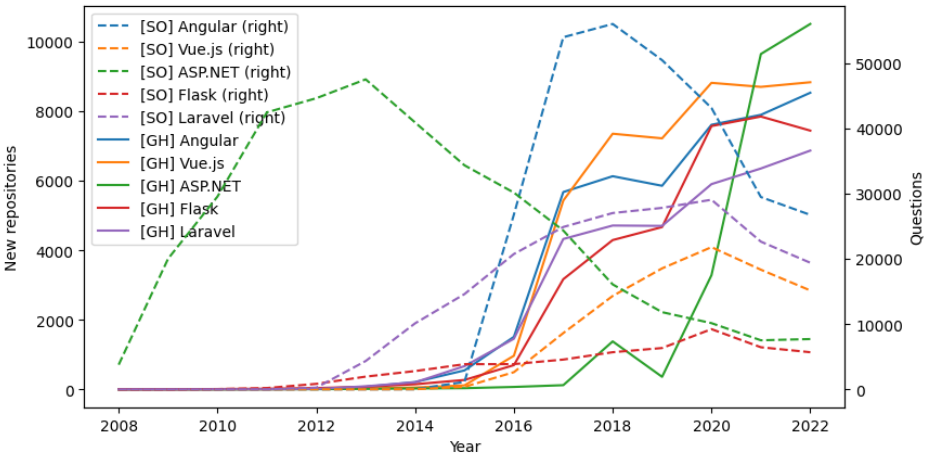

Figure 5. Comparison of GitHub and Stack Overflow popularity (part 2).

The first and most striking observation from the comparison of charts based on the two sources is that the general long-term rising trend clearly visible in the GitHub data corresponds to a short-term falling trend visible in the Stack Overflow data. We discussed the possible cause of the latter at the end of Section 3.3.

The second general observation is the very high popularity of certain frameworks (e.g., jQuery, ASP.NET, and Ruby on Rails) peaking in years prior to 2014, for which GitHub reports very low numbers (they are actually in the range of tens and hundreds and not zeroes as they may look in the chart, but this is still a small fraction of the numbers reported from Stack Overflow). We discussed this issue with the data obtained from GitHub at the end of Section 3.2.

Looking at the data reported by the two sources regarding individual frameworks, various similarities and differences could be observed. If we look at the most recent seven years, both sources indicate the very high (and still growing or staying close to its peak level) popularity of Node.js, React.js, and Django. Both sources indicate the relatively low current popularity of jQuery and Ruby on Rails, which is also past its peak, though the peak levels of these frameworks reported by the two sources are incomparable. For one framework, the two sources agree on the trend but vastly disagree on the current level of popularity (Flask; Vue.js could be added to this group if the data for the last year had been omitted), and for the two others, they disagree on the trend but roughly agree on the level of popularity in recent years (Angular and Laravel), and the lines drawn by the two data sources for ASP.NET bear very little resemblance.

In order to address research aim 3, we investigated whether data from one source can be used to forecast the other. If that were the case, the former could be considered an early indicator of web framework popularity, helping with the choice of technology to base a new project on or a new skill to learn.

Knowing already the discrepancies in the results based on the two sources described above, we decided to analyze the case of each framework individually. We also decided to check both the GitHub-reported popularity preceding the Stack Overflow-reported popularity and vice versa as both situations are plausible:

- An increased number of people learning and trying a framework (visible in Stack Overflow) will turn into an increased number of users, part of which will contribute to open-source projects (visible in GitHub),
- An increased number of open-source projects developed by using a framework (visible in GitHub) will spark an increased number of developers trying to adopt and/or adapt them in their own projects who will encounter problems and ask the community about them (visible in Stack Overflow).

Figure 6 shows the cross-correlation between the number of new repositories in GitHub and the number of questions posted on Stack Overflow, whereas Figure 7 shows the cross-correlation between the number of questions posted on Stack Overflow and the number of new repositories in GitHub.

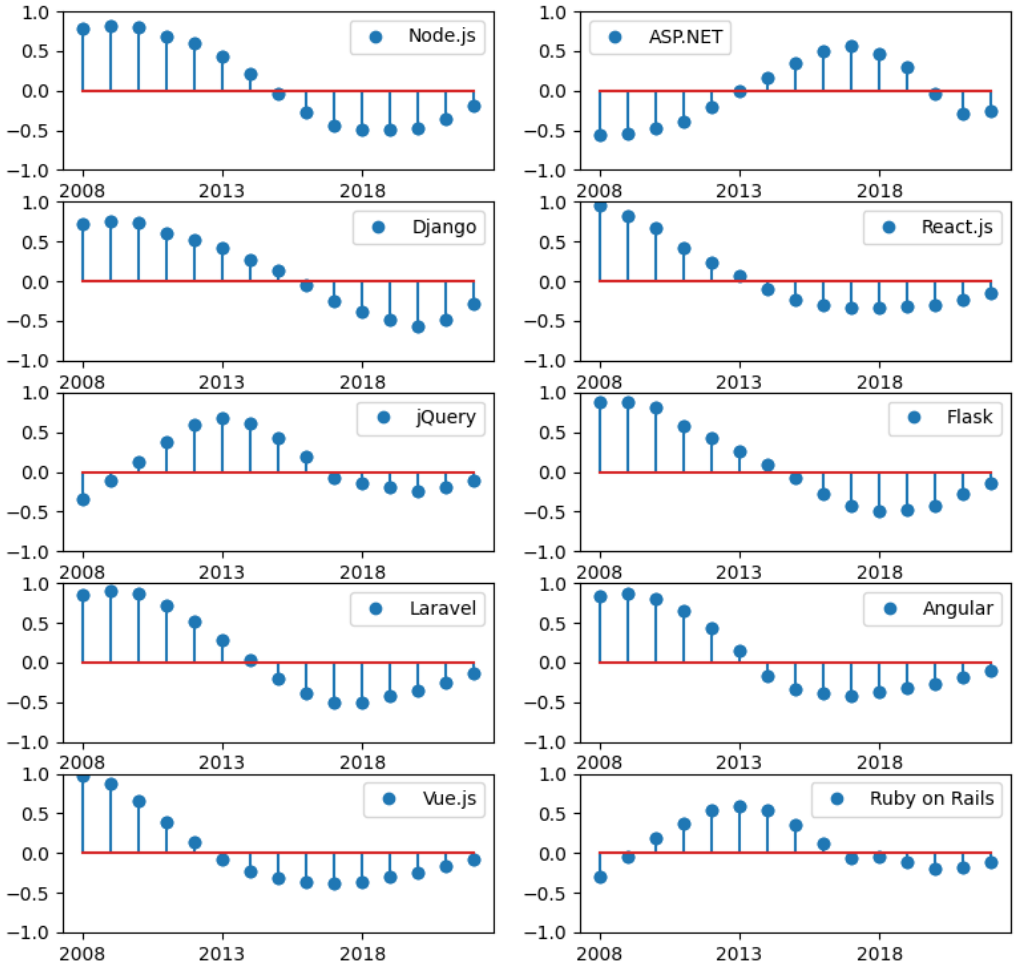

Figure 6. Cross-correlation between data retrieved from GitHub and Stack Overflow.

Looking at Figure 6, for 7 out of the 10 analyzed frameworks, one can observe somewhat similar pictures showing quite high cross-correlations of various signs, which changes around 2014 (i.e., when the level of numbers obtained from GitHub starts to rise). The three

remaining ones (ASP.NET, jQuery, and Ruby on Rails) paint a bit more complicated charts with an additional peak (or bottom) positioned in recent years.

The results pictured in Figure 7 are a bit different, which is especially noticeable in the level of cross-correlation in most recent years which seems to be smaller and less variable than depicted in Figure 6. Nonetheless, as the charts do not give a simple answer to the stated question, the Granger causality test was applied for this purpose (with the maximum lag parameter set to three). The *p*-values obtained in the performed tests are listed in Table 4.

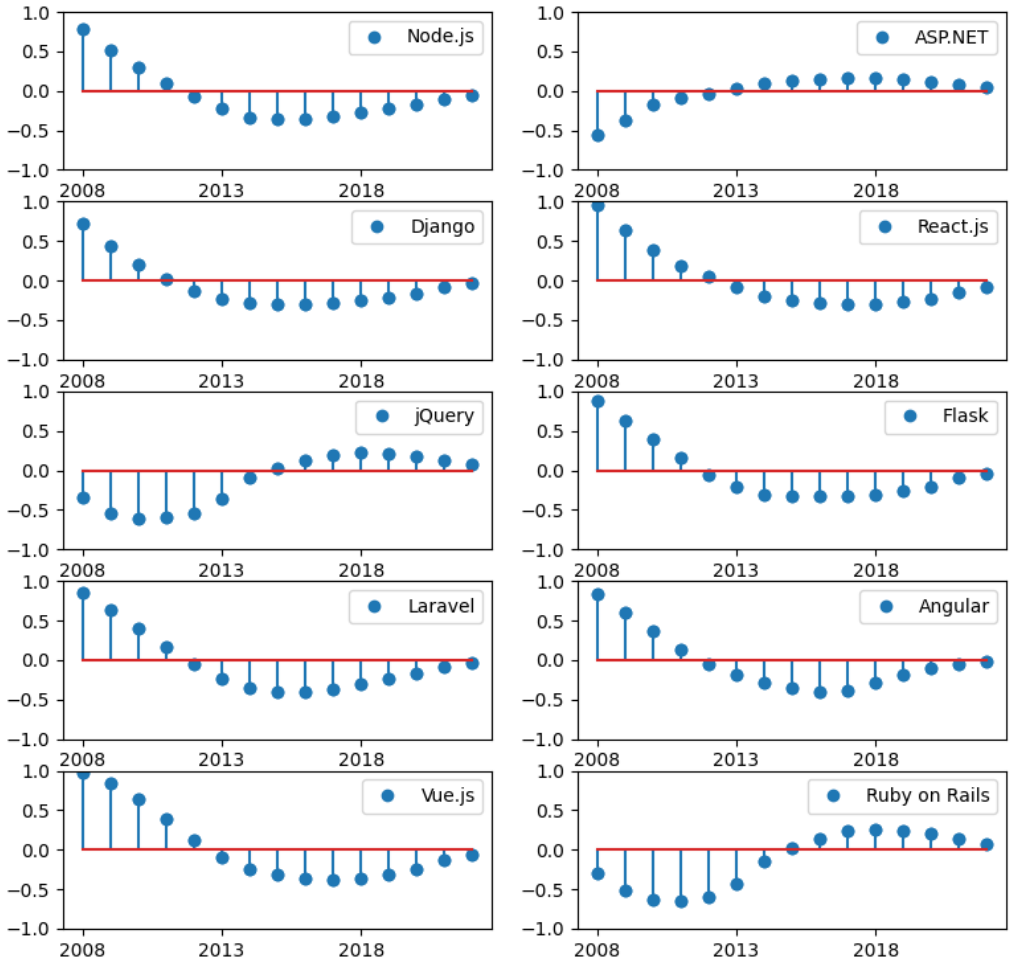

Figure 7. Cross-correlation between data retrieved from Stack Overflow and GitHub.

Table 4. Results of Granger causality test between the two considered data sources.

Framework	GitHub → Stack Overflow	Stack Overflow → GitHub
Node.js	0.0000	0.0010
React.js	0.0000	0.0000
jQuery	0.0000	0.0062
Angular	0.0187	0.0000
Vue.js	0.0000	0.0000
ASP.NET	0.2847	0.1288
Django	0.1499	0.0020
Flask	0.0267	0.0000
Laravel	0.0000	0.0000
Ruby on Rails	0.0000	0.0837

Assuming a significance level of 0.05:

- The number of newly created repositories in GitHub could be used to forecast the number of questions posted in Stack Overflow for all frameworks except ASP.NET and Django;
- The number of questions posted in Stack Overflow could be used to forecast the number of newly created repositories in GitHub for all frameworks except ASP.NET and Ruby on Rails.

We can thus conclude that the correspondence between the data retrieved from the two sources was confirmed (with the exception of ASP.NET), but the obtained results are insufficient to clearly indicate one source as indicating changes ahead of the other (with the exception of Django and Ruby on Rails, but the results obtained in their respective cases are diverging).

3.5. Web Framework Popularity in the Context of Programming Language Popularity

Web frameworks are developed in a given programming language usually to support the web development in that programming language. Addressing the fourth research aim, in this subsection, we investigate whether and to what extent the web framework popularity corresponds to the popularity of the programming languages they are based on. One could expect that a widely used language will automatically increase the popularity of its frameworks, and/or, as web programming is nowadays a primary vein of programming, that a widely used web development framework will automatically increase the popularity of its programming language.

The popularity of the programming languages relevant to the frameworks considered in the analysis, measured with the TIOBE Programming Community index, is shown in Figure 8.

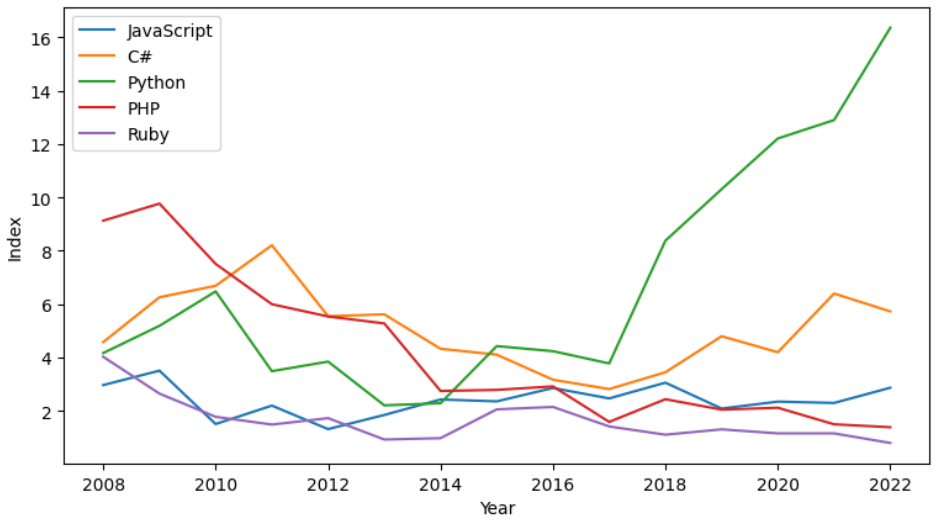

Figure 8. TIOBE Programming Community index of relevant languages.

In Figure 8, we can clearly see that the primary language of web development (JavaScript) is trailing well behind the other two considered programming languages, having wider areas of application. To investigate this matter with a finer grain of detail, in the following figures, the popularity of the frameworks developed for a given programming language (according to GitHub) is compared with the popularity of that programming language (according to TIOBE).

The first of them, shown in Figure 9, combines the number of newly created GitHub repositories (left scale) having assigned one of the JavaScript frameworks in their topics with the JavaScript TIOBE Programming Community index (right scale) throughout the analyzed period.

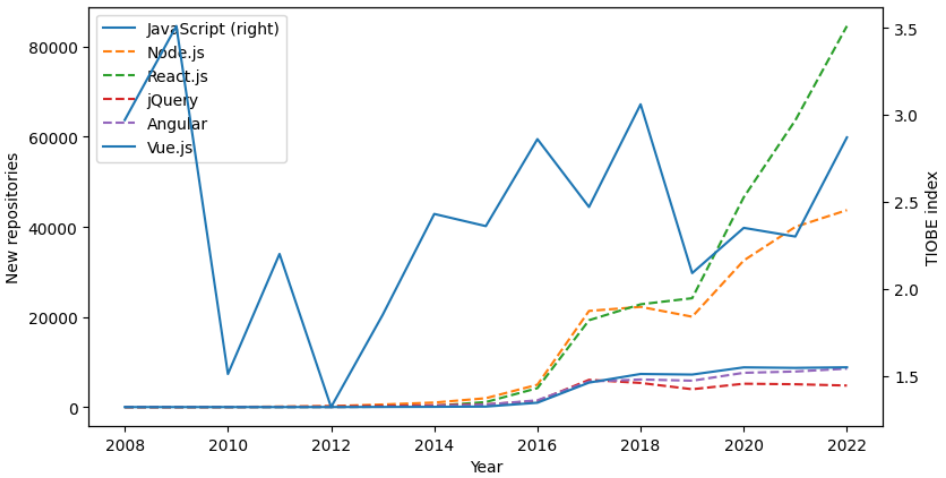

Figure 9. JavaScript TIOBE index vs. new GitHub repositories using frameworks.

Looking at Figure 9, we can observe that the JavaScript index follows quite a complicated path, whereas the frameworks based on it seem to join a rising trend of various strength at some point in time with individual periods of weak slowdown (which seem to coincide with the drop in JavaScript's popularity).

Figure 10 allows us to compare the number of newly created repositories in GitHub assigned to the ASP.NET topic with the C# TIOBE Programming Community index throughout the analyzed period.

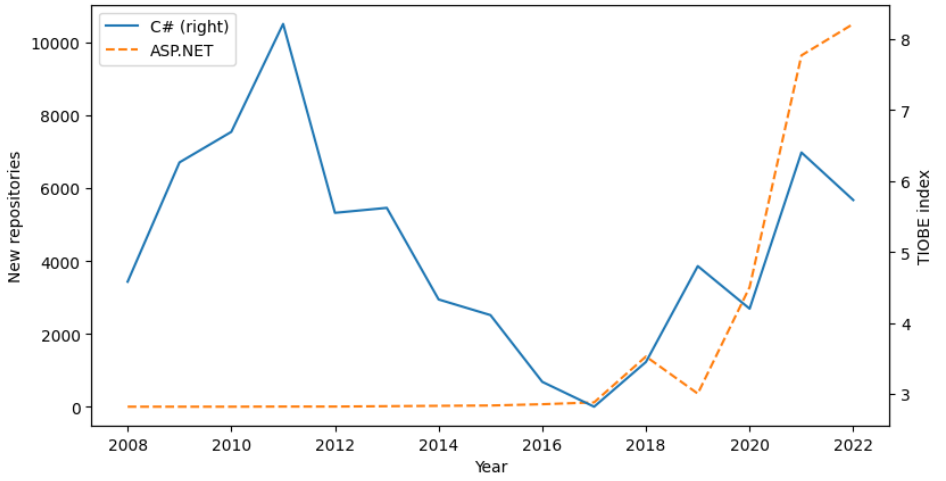

Figure 10. C# TIOBE index vs. new ASP.NET repositories in GitHub.

The chart in Figure 10 shows a visible correspondence between the two indicators yet only in the period 2017–2021. It should be reminded here that C# is the most popular but not the only programming language used in ASP.NET projects as well as the fact that, unlike JavaScript, web development is not the only area in which C# is used.

Figure 11 shows the Python TIOBE index compared with the number of newly created repositories in GitHub having assigned one of the Python frameworks in their topics.

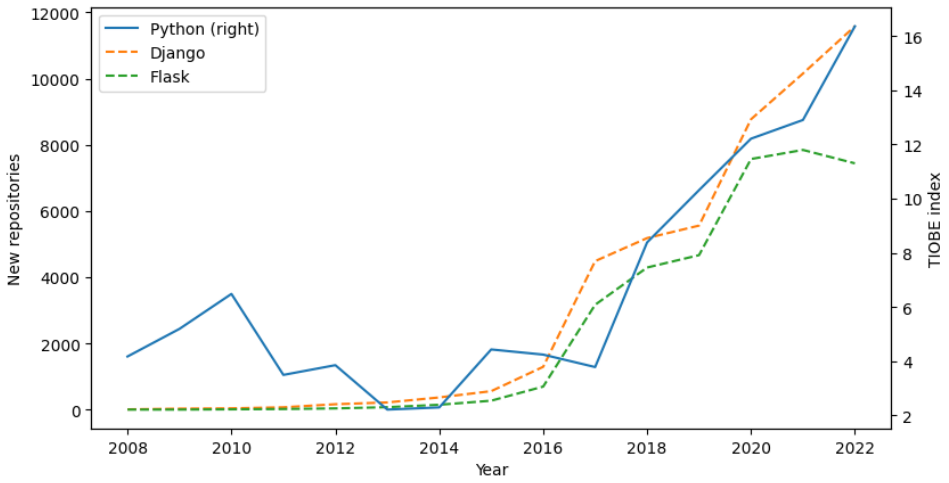

Figure 11. Python TIOBE index vs. new GitHub repositories using Python frameworks.

Quite unexpectedly, Figure 11 reveals a close correspondence between the popularity of Python and the popularity of the web frameworks based on it from the year 2013 on. We can conclude that web development was a factor noticeably contributing to Python's popularity at least since 2013.

Figure 12 combines the TIOBE Programming Community index for PHP with the number of newly created repositories for GitHub projects having assigned Laravel as one of their topics throughout the analyzed period.

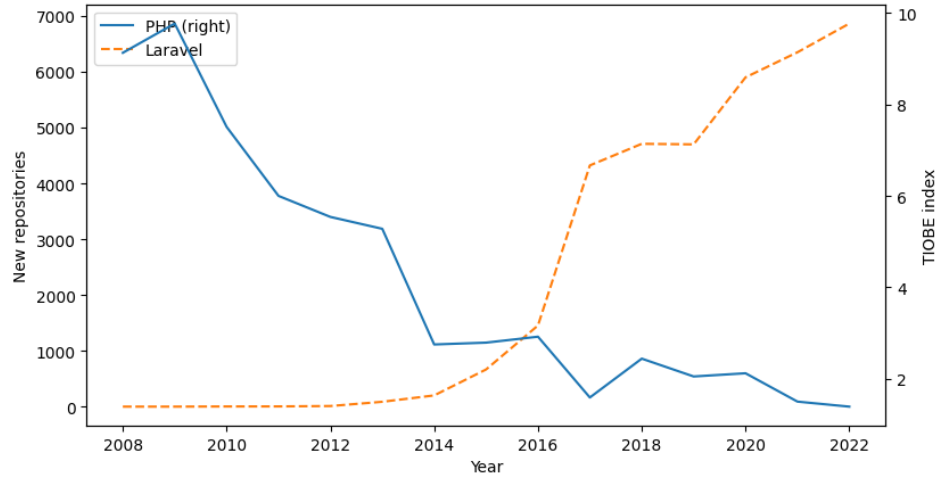

Figure 12. TIOBE index of PHP vs. new Laravel repositories in GitHub.

As seen in Figure 12, both indicators undergo notable changes in their level; however, they are headed in opposite directions. PHP is a language strictly confined to web development, yet this observation can be easily explained by the fact that Laravel, although the only one that made it to the top 10 frameworks in 2022 [26], is merely one of the tens of PHP frameworks available in the market [56] which, combined, must have a much stronger impact on the PHP popularity than Laravel does.

The figure in the series, Figure 13, allows one to compare the number of newly created GitHub repositories assigned to the Ruby on Rails topic with the Ruby TIOBE Programming Community index throughout the analyzed period.

Although Ruby on Rails has been called "unquestionably Ruby's killer app" [57], the chart in Figure 13 does not seem to confirm that as the correspondence between the two indicators is not obvious, as was the case with the Python frameworks.

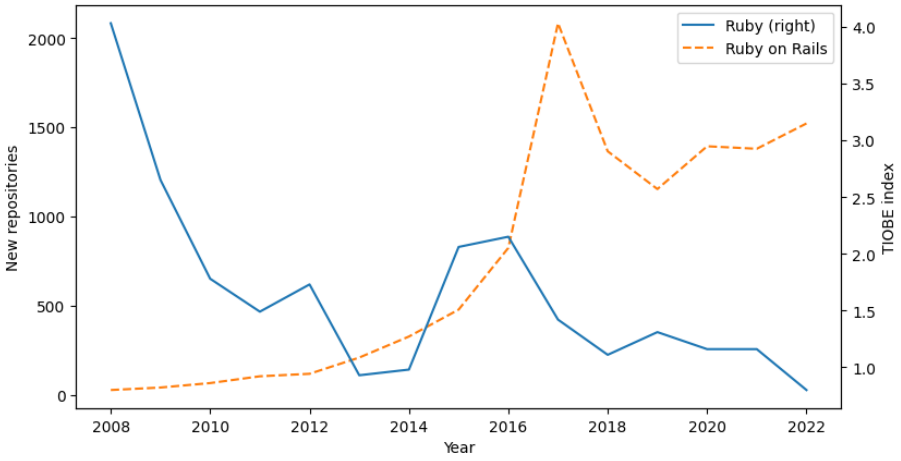

Figure 13. Ruby TIOBE index vs. new Ruby on Rails repositories in GitHub.

In order to verify whether the web framework popularity according to GitHub can be used to forecast the TIOBE Index of the underlying programming language or vice versa, the Granger causality test was applied (with the maximum lag parameter set to three). The *p*-values obtained in the performed tests are listed in Table 5.

Table 5. Results of Granger causality test between GitHub and TIOBE data.

Framework	GitHub → TIOBE	TIOBE → GitHub
Node.js	0.0374	0.0158
React.js	0.0687	0.0945
jQuery	0.0297	0.0079
Angular	0.0298	0.0051
Vue.js	0.0776	0.0272
ASP.NET	0.0885	0.4078
Django	0.0000	0.0021
Flask	0.0000	0.0001
Laravel	0.5419	0.0007
Ruby on Rails	0.2038	0.0000

Assuming a significance level of 0.05:

- The number of newly created repositories in GitHub could be used to forecast the TIOBE Index of the language that a given framework is addressing for six frameworks: Node.js, jQuery, Angular, Django, Flask, and Ruby on Rails;
- The TIOBE Index of the language that a given framework is addressing could be used to forecast the number of newly created repositories in GitHub by using that framework for all frameworks except ASP.NET and React.js.

We can thus conclude that the correspondence between the data retrieved from the two sources was confirmed (with the exception of ASP.NET and React.js). The popularity of a programming language allows one to forecast the popularity of a web framework more often than vice versa.

In the last stage of our study, we again apply the Granger causality test, yet this time to check whether the TIOBE Index of the underlying programming language can be used to forecast the web framework popularity according to Stack Overflow (or vice versa). The *p*-values obtained in the performed tests are listed in Table 6.

Assuming a significance level of 0.05:

- The number of questions posted on Stack Overflow could be used to forecast the TIOBE Index of the language that a given framework is addressing for six frameworks: Node.js, jQuery, ASP.NET, Django, Flask, and Ruby on Rails;

- The TIOBE Index of the language that a given framework is addressing could be used to forecast the number of questions posted on Stack Overflow regarding that framework also for six frameworks: jQuery, Angular, Vue.js, ASP.NET, Laravel, and Ruby on Rails.

Table 6. Results of Granger causality test between Stack Overflow and TIOBE data.

Framework	Stack Overflow → TIOBE	TIOBE → Stack Overflow
Node.js	0.0000	0.0921
React.js	0.2612	0.0946
jQuery	0.0000	0.0377
Angular	0.1465	0.0026
Vue.js	0.0741	0.0056
ASP.NET	0.0000	0.0022
Django	0.0001	0.5549
Flask	0.0000	0.0715
Laravel	0.1394	0.0000
Ruby on Rails	0.0125	0.0000

We can thus once again conclude that the correspondence between the data retrieved from the two sources was confirmed (with the exception of React.js). In the case of Node.js, Django, and Flask, the popularity of the web framework on Stack Overflow allowed us to forecast the popularity of the programming language that it is based on, but not vice versa, whereas in the case of Angular, Vue.js, and Laravel, the popularity of the programming language allowed us to forecast the popularity of the web framework based on it, but not vice versa.

3.6. Research Outlooks

As the presented study opens a new vein in the research on web development frameworks, in Table 7, we strive to match its key contributions with the directions for further research that they indicate.

Table 7. Main contributions and indicated future work directions.

RQ	Contribution	Future Work Directions
1	Confirming that web frameworks undergo strong and rapid changes in popularity	Investigating causes of these changes
2	Showing that the two considered indicators of web framework popularity are only partly consistent with each other (aligning only for specific frameworks and/or specific periods)	Developing an aggregate popularity indicator that would combine these and possibly other factors
3	Showing that neither of the two considered indicators of web framework popularity can be treated as always preceding the other	Search for other indicators that would be capable of reliably predicting the changes in web framework popularity
4	Showing that both the considered indicators of web framework popularity are in correspondence with the popularity of the programming language they are based on, but there are exceptions to this rule	Investigating causes of such exceptions
5	Showing that there are no clear patterns in the characteristics of the currently popular web frameworks	Search for factors that could be effective in forecasting framework popularity

3.7. Limitations

Like all empirical studies, our research has its limitations. As shown in Figure 1, none of the indicators we used as estimators of popularity were actually measuring the software development activity of the users of the respective frameworks.

We also rely on the quality of the data provided by the considered sources, although we know well that they contain human-induced errors in the form of misplaced and missing topics or tags. We assume their ratio to be acceptable, taking into consideration that it is in the best interest of those assigning a topic to GitHub projects or tagging a Stack Overflow question to perform it properly—otherwise, their projects and questions would not be listed in search results for framework-specific queries, and in consequence, they would receive little publicity and few answers, respectively.

4. Conclusions

In this paper, we investigated how the ten currently most popular web development frameworks (according to [26]) fared in the last 15 years. We attained all the defined research aims. With regard to research aim 1, we showed how the popularity of the respective web frameworks evolved across the years according to data obtained from two independent sources. Regarding research aim 2, we identified similarities and differences between the results based on the data obtained from the two considered sources. With regard to research aim 3, we verified whether one can be used to forecast the other. Regarding research aim 4, we confirmed correspondence between the web framework popularity and the popularity of the programming languages that they are based on. With regard to research aim 5, we performed a comparative analysis of the considered frameworks based on a set of versatile and objective criteria.

Monitoring the evolving popularity of web development frameworks and striving to predict it is important for the software industry, individual software companies, and individual developers, as it is instrumental in adapting effectively to technological trends as well as effectively allocating resources and educational effort. To the best of our knowledge, this is the first published study that tackled this topic. As it led to results that are partly ambiguous, thus giving more new questions than answers, we hope that it will spark scientific discussion on the causes and the effective ways of predicting the direction of the evolution of web development frameworks.

Author Contributions: Conceptualization, J.S.; methodology, A.K. and J.S.; software, A.K.; validation, A.K.; formal analysis, J.S.; investigation, J.S.; resources, A.K.; data curation, A.K.; writing—original draft preparation, J.S.; writing—review and editing, A.K. and J.S.; visualization, A.K.; supervision, J.S.; project administration, J.S.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data were obtained from GitHub, Stack Overflow, and TIOBE and are available from the authors with the permission of their respective providers.

Acknowledgments: The authors would like to thank the reviewers of the initial submission of this paper for their insightful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kumar, S.; Lim, W.M.; Pandey, N.; Christopher Westland, J. 20 years of Electronic Commerce Research. *Electron. Commer. Res.* **2021**, *21*, 1–40. [CrossRef]
2. Ronchi, A.M. *e-Services*; Springer International Publishing: Cham, Switzerland, 2019. [CrossRef]

3. Charalabidis, Y.; Loukis, E.; Alexopoulos, C.; Lachana, Z. The Three Generations of Electronic Government: From Service Provision to Open Data and to Policy Analytics. In *Electronic Government*; Lindgren, I., Janssen, M., Lee, H., Polini, A., Rodríguez Bolívar, M.P., Scholl, H.J., Tambouris, E., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–17.
4. Raible, M.; Darwin, I.; Dewanto, L. The History of Web Frameworks Timeline. 2019. Available online: https://github.com/mraible/history-of-web-frameworks-timeline (accessed on 18 April 2023).
5. Laaziri, M.; Benmoussa, K.; Khoulji, S.; Kerkeb, M.L. A Comparative study of PHP frameworks performance. *Procedia Manuf.* **2019**, *32*, 864–871. [CrossRef]
6. del Pilar Salas-Zárate, M.; Alor-Hernández, G.; Valencia-García, R.; Rodríguez-Mazahua, L.; Rodríguez-González, A.; López Cuadrado, J.L. Analyzing best practices on Web development frameworks: The lift approach. *Sci. Comput. Program.* **2015**, *102*, 1–19. [CrossRef]
7. Sarcar, V. Use the DRY Principle. In *Simple and Efficient Programming with C#: Skills to Build Applications with Visual Studio and .NET*; Apress: Berkeley, CA, USA, 2023; pp. 109–130. [CrossRef]
8. Mehrab, Z.; Yousuf, R.B.; Tahmid, I.A.; Shahriyar, R. Mining Developer Questions about Major Web Frameworks. In Proceedings of the 14th International Conference on Web Information Systems and Technologies, Seville, Spain, 18–20 September 2018; pp. 191–198. [CrossRef]
9. Muhammed, T.; Mehmood, R.; Abozinadah, E.; Sharaf, S. SelecWeb: A Software Tool for Automatic Selection of Web Frameworks. In *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*; Mehmood, R., See, S., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 329–346. [CrossRef]
10. Stella, L.F.F.; Jarzabek, S.; Wadhwa, B. A comparative study of maintainability of web applications on J2EE, .NET and Ruby on Rails. In Proceedings of the 2008 10th International Symposium on Web Site Evolution, Beijing, China, 3–4 October 2008; pp. 93–99. [CrossRef]
11. Glassman, N.R.; Shen, P. One Site Fits All: Responsive Web Design. *J. Electron. Resour. Med. Libr.* **2014**, *11*, 78–90. [CrossRef]
12. Aborujilah, A.; Adamu, J.; Shariff, S.M.; Awang Long, Z. Descriptive Analysis of Built-in Security Features in Web Development Frameworks. In Proceedings of the 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Republic of Korea, 3–5 January 2022; pp. 1–8. [CrossRef]
13. Kaluža, M.; Troskot, K.; Vukelić, B. Comparison of front-end frameworks for web applications development. *J. Polytech. Rij.* **2018**, *6*, 261–282. [CrossRef]
14. Stack Overflow. Stack Overflow–Where Developers Learn, Share, & Build Careers. 2023. Available online: https://stackoverflow.com/ (accessed on 19 April 2023).
15. Kaluža, M.; Kalanj, M.; Vukelić, B. A comparison of back-end frameworks for web application development. *J. Polytech. Rij.* **2019**, *7*, 317–332. [CrossRef]
16. Mishra, S.; Srivastava, S. Web development frameworks and its performance analysis—A review. In *Smart Computing*, 1st ed.; Khan, M.A., Gairola, S., Jha, B., Praveen, P., Eds.; CRC Press: London, UK, 2021; pp. 337–343. [CrossRef]
17. Ollila, R.; Mäkitalo, N.; Mikkonen, T. Modern Web Frameworks: A Comparison of Rendering Performance. *J. Web Eng.* **2022**, *21*, 789–814. [CrossRef]
18. Kopyl, P.; Rozaliuk, T.; Smołka, J. Comparison of ASP.NET Core and Spring Boot ecosystems. *J. Comput. Sci. Inst.* **2022**, *22*, 40–45. [CrossRef]
19. Singleton, J.L.; Leavens, G.T. Verily: A Web Framework for Creating More Reasonable Web Applications. In Proceedings of the ICSE '14: 36th International Conference on Software Engineering, New York, NY, USA, 31 May–7 June 2014; pp. 560–563. [CrossRef]
20. Klochkov, D.; Mulawka, J. Improving Ruby on Rails-Based Web Application Performance. *Information* **2021**, *12*, 319. [CrossRef]
21. bin Uzayr, S.; Cloud, N.; Ambler, T. *JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now*; Apress: Berkeley, CA, USA, 2019. [CrossRef]
22. TIOBE. TIOBE Index for April 2023. 2023. Available online: https://www.tiobe.com/tiobe-index/ (accessed on 19 April 2023).
23. OSS Insight. Trends and Insights from GitHub 2022. 2023. Available online: https://ossinsight.io/blog/trends-and-insights-from-github-2022/#top-languages-in-the-open-source-world-over-the-past-four-years (accessed on 25 July 2023).
24. Sobral, S.R. 30 years of CS1: Programming languages evolution. In Proceedings of the ICERI2019 Proceedings, Seville, Spain, 11–13 November 2019; pp. 9197–9205. [CrossRef]
25. Swacha, J. Programming Languages in Education: 50 Years of Evolution as Evidenced by Literature. In Proceedings of the SIGCSE 2023: 54th ACM Technical Symposium on Computer Science Education V. 2, New York, NY, USA, 15–18 March 2023; p. 1385. [CrossRef]
26. Vailshery, L.S. Most Popular Web Frameworks among Developers Worldwide 2022. 2022. Available online: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (accessed on 18 April 2023).
27. Google. Google Trends. 2023. Available online: https://trends.google.com/ (accessed on 25 July 2023).
28. GitHub. GitHub: Let's Build from Here. 2023. Available online: https://github.com/ (accessed on 25 July 2023).
29. McKinney, W. *Python for Data Analysis*, 3rd ed.; O'Reilly: Sebastopol, CA, USA, 2022.
30. Parsonson, B.S.; Baer, D.M.; Kratochwill, T.R.; Levin, J.R. The visual analysis of data, and current research into the stimuli controlling it. In *Single-Case Research Design and Analysis: New Directions for Psychology and Education*; Routledge: London, UK, 2015; pp. 15–40.

31. Gaydecki, P. *Foundations of Digital Signal Processing: Theory, Algorithms and Hardware Design*; Institution of Electrical Engineers: London, UK, 2004.
32. Granger, C.W.J. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica* **1969**, *37*, 424. [CrossRef]
33. McGreggor, D.M. *Mastering Matplotlib*, 1st ed.; Packt Publishing: Birmingham, UK, 2015.
34. Weamie, S.J. Cross-Site Scripting Attacks and Defensive Techniques: A Comprehensive Survey. *Int. J. Commun. Netw. Syst. Sci.* **2022**, *15*, 126–148. [CrossRef]
35. Sahani, R.; Randhawa, S. Clickjacking: Beware of clicking. *Wirel. Pers. Commun.* **2021**, *121*, 2845–2855. [CrossRef]
36. Coram, M. *Cross-Site Request Forgery Challenges and Solutions*; Technical Report; Sandia National Lab. (SNL-NM): Albuquerque, NM, USA, 2019.
37. Singh, A.; Gupta, B.B. Distributed denial-of-service (DDoS) attacks and defense mechanisms in various web-enabled computing platforms: Issues, challenges, and future research directions. *Int. J. Semant. Web Inf. Syst. (IJSWIS)* **2022**, *18*, 1–43. [CrossRef]
38. Biswas, S.; Sohel, M.; Sajal, M.M.; Afrin, T.; Bhuiyan, T.; Hassan, M.M. A study on remote code execution vulnerability in web applications. In Proceedings of the International Conference on Cyber Security and Computer Science (ICONCS 2018), Karabuk, Turkey, 18–20 October 2018; pp. 50–57.
39. OpenJS Foundation. Documentation Node.js. Available online: https://nodejs.org/en/docs (accessed on 25 June 2023).
40. MetaOpenSource. Built-in React Hooks–React. Available online: https://react.dev/reference/react (accessed on 25 June 2023).
41. OpenJS Foundation. jQuery UI. Available online: https://jqueryui.com/ (accessed on 25 June 2023).
42. Google. Introduction to the Angular Docs. Available online: https://angular.io/docs (accessed on 25 June 2023).
43. You, E. Glossary Vue.js. Available online: https://vuejs.org/glossary/ (accessed on 25 June 2023).
44. Microsoft. ASP.NET Documentation. Available online: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0&WT.mc_id=dotnet-35129-website (accessed on 25 June 2023).
45. Django Software Foundation. Django Documentation. Available online: https://docs.djangoproject.com/en/4.2/ (accessed on 25 June 2023).
46. Pallets. Flask Documentation 2.3.x. Available online: https://flask.palletsprojects.com/en/2.3.x/ (accessed on 25 June 2023).
47. Laravel LLC. Laravel Documentation. Available online: https://laravel.com/docs/10.x (accessed on 25 June 2023).
48. Foundation, T.R. The Rails Foundation. Available online: https://guides.rubyonrails.org/ (accessed on 25 June 2023).
49. Leff, A.; Rayfield, J. Web-application development using the Model/View/Controller design pattern. In Proceedings of the Fifth IEEE International Enterprise Distributed Object Computing Conference, Seattle, WA, USA, 4–7 September 2001; pp. 118–127. [CrossRef]
50. Shadhin, F. The MVT Design Pattern of Django. 2021. Available online: https://python.plainenglish.io/the-mvt-design-pattern-of-django-8fd47c61f582 (accessed on 25 July 2023).
51. Michelson, B.M. Event-driven architecture overview. *Patricia Seybold Group Res. Serv.* **2006**, *2*, 10–1571.
52. Boduch, A. *Flux Architecture*; Packt Publishing Ltd.: Birmingham, UK, 2016.
53. Li, N.; Zhang, B. The research on single page application front-end development based on Vue. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; Volume 1883, p. 012030.
54. Yigitbas, E.; Josifovska, K.; Jovanovikj, I.; Kalinci, F.; Anjorin, A.; Engels, G. Component-based development of adaptive user interfaces. In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Valencia, Spain, 18–21 June 2019; pp. 1–7.
55. Freeman, A. *Pro ASP.NET Core MVC*; Apress: Berkeley, CA, USA, 2016. [CrossRef]
56. Jamsheer, K. Best PHP Frameworks for Web Development in 2022. 2022. Available online: https://acodez.in/best-php-frameworks/ (accessed on 18 April 2023).
57. Carlson, L.; Richardson, L. *Ruby Cookbook*, 1st ed.; O'Reilly: Beijing, China; Sebastopol, CA, USA, 2006.