VIGHUB: a Technology Forecasting Tool based on Mining Software Repositories

VIGHUB: una Herramienta de Pronóstico Tecnológico basada en Minería de Repositorios de Software

DOI: http://doi.org/10.17981/ingecuc.18.1.2022.07

Artículo de Investigación Científica. Fecha de Recepción: 11/02/2022. Fecha de Aceptación:.2402/2022.

Carlos Giovanny Hidalgo-Suárez
Universidad del Valle. Cali (Colombia)
carlos.hidalgo@correounivalle.edu.co

Victor Andres Bucheli-Guerrero
Universidad del Valle. Cali (Colombia)
victor.bucheli@correounivalle.edu.co

Hugo Armando Ordoñez-Eraso
Universidad del Cauca. Popayán (Colombia)
hugoordonez@unicauca.edu.

To cite this paper:
C. Hidalgo-Suarez, V. Bucheli-Guerrero & H. Ordoñez-Eraso, "VIGHUB: una Herramienta de Pronóstico Tecnológico basada en Minería de Repositorios de Software", *INGE CUC*, vol. 18, no. 1, pp. 83–94, 2022. DOI: http://doi.org/10.17981/ingecuc.18.1.2022.07

Abstract

Introduction— Academics, developers, and companies focused on technological development seek to know what exists and what is still missing in this field. One of the ways they use is the review of bibliographic sources (state-of-the art). In this sense, a tool was developed that allows the current state to be identified semi-automatically.

Objective— This article proposes a tool that extracts information from repositories hosted on GitHub. It analyzes the data using computational techniques and presents the results through visualizations that identify the field's technological evolution studied through the most used programming languages, central repositories, and organizations.

Method— A model based on Mining Software Repositories (MSR) is used, which integrates an architecture based on microservices, using different programming languages, which allowed the construction of the VigHub tool. The model focuses on four aspects— Selection of a topic, extraction of the data source, analysis of information using computational techniques, and finally, the results are communicated through visualizations.

Results— The VigHub tool was available online to carry out 3 case studies. The first in the academy, where technologies, programming languages, users, and companies interested in developing VLE's (Virtual Learning Environment) were identified from 2011 to 2021. The second and third were carried out by companies (industrial environment), which stated that using the VigHub tool supports data analysis and valuable results identification.

Conclusions— A tool that allows identifying a part of the current state of technology could be a helpful tool for academics, developers, and companies, saving human resources, time, and possible repeated developments---code reuse. The VigHub tool aims to support the construction of state-of-the-art. Its results are complementary to the traditional method.

Keywords— Mining Software Repositories; Technology Forecasting; State-of-the technique; GitHub; Technological maps

Resumen

Introducción— Académicos, desarrolladores y empresas enfocadas en el desarrollo tecnológico, buscan conocer lo que ya existe y lo que aún falta en este campo. Una de las formas que utilizan, es realizar revisiones sobre fuentes bibliográficas (estado del arte). En este sentido, se desarrolló una herramienta que permite identificar el estado actual de una tecnología de forma semi-automática.

Objetivo— Este artículo propone una herramienta que extrae información de repositorios alojados en GitHub. Analiza los datos utilizando técnicas computacionales y presenta los resultados a través de visualizaciones que identifican la evolución tecnológica del campo estudiado a través de los lenguajes de programación, principales, repositorios y organizaciones.

Metodología— Se utiliza un modelo basado en Repositorios de Software de Minería (MSR), el cual integra una arquitectura basada en microservicios utilizando diferentes lenguajes de programación, lo que permitió la construcción de la herramienta VigHub. El modelo se centra en cuatro aspectos— selección de un tema tecnológico, extracción de la fuente de datos, análisis de la información mediante técnicas computacionales y finalmente, se muestran los resultados a través de visualizaciones.

Resultados— Se dispuso la herramienta VigHub de manera online para realizar 3 casos de estudio. El primero en la academia, donde se identifico desde el año 2011 al 2021, las tecnologías, los lenguajes de programación, los usuarios y empresas interesadas en el desarrollo de VLE's (Virtual Learning Environment). El segundo y tercero fueron ejecutados por empresas (ambiente industrial), que afirmaron que el uso de la herramienta VigHub, apoya tanto en el análisis de datos como en la identificación de resultados útiles.

Conclusiones— Contar con una herramienta que a partir de una sola consulta permite identificar parte del estado actual de una tecnología, podría ser una herramienta útil para académicos, desarrolladores y empresas, que ahorrarían recursos humanos, tiempo y posibles desarrollos repetidos---reutilización de código. La herramienta VigHub pretende apoyar en la construcción de un estado de arte. Sus resultados son complementarios al método tradicional.

Palabras clave— Minería de repositorios de software; vigilancia tecnológica; estado de la técnica; GitHub; mapas tecnológicos

I. Introduction

The projects Software are in constant evolution, with access to information and technological advances. In this context, the creation of starting strategies and the taking of information decisions are crucial for the competition; this implies creating technological strategies to anticipate and take better advantage of the opportunities [1], [2]. The Mining Software Repositories (MSR) focus on analyzing the data available in software repositories, such as version control repositories, mailing list archives, bug tracking systems, and reporting systems, among other contexts [3].

This work propose a software tool which supports the MSR process based on technological forecasting. It is integrated computational techniques to extract data from the collaborative platform GitHub, process, analyze and visualize the information of software technology in a software tool VigHub. This information is useful to create strategies and support the decision-making process in software development.

VigHub was developed according to microservices architecture [4], which optimizes the processes and makes them adaptable with different modules (it was developed in different technologies and different programming languages). According to to different authors [5], [6], VigHub follows a forecasting model compose of the following 4 stages: the first stage, the planning, choose the data source. The second stage (Extraction, Transformation and Loading-ETL), is filtering and debugging information. The third stage is responsible for processing the data using computer techniques (data extraction, natural language processing, and machine learning). And the final stage communicates the information through visualizations.

VigHub has been tested in two software development companies and has contributed to two scientific papers in computer science. The graphics generated (technological maps) by the tool have allowed answering interesting questions such as: what are the programming languages used in the development of technology? What are the successful projects on the subject? What users and organizations are important in the subject? Among other issues that are relevant to find the current state of a specific technology.

The paper is organized in the following sections: a review of related works in Section 2. In Section 3, the VigHub tool is described in detail. In Section 4, the results of 3 case studies using VigHub are presented. a case about virtual judges and two more cases in Colombian. Finally, Section 5 shows the discussion, and Section 6, the conclusions.

II. Related Work

Mining Software Repositories (MSR) allows analyzing the evolution of the software automatically, through the application of data mining techniques in the data of the development history. Studies in this field reveal useful information for the development of a particular project or even find patterns in the evolution of the software that can be generalized to other software systems [7]. The following lines present some projects and works that show the impact that the construction of MSR tools has had in support of software development.

The BOA tool is based on MSR [8]. This tool uses a domain-specific programming language (RUST) to analyze large-scale software repositories. In addition, it uses distributed computing techniques to execute queries on MSR efficiently, so you can extract interesting information in a simple way. In 2017, Boa had almost 1.3 million projects in his knowledge base.

Other tools such as MetricMiner [9] and Mezuro [10] are web tools to support researchers when working with mining software repositories. When a researcher needs to do a comparative-similar study, he needs to install many different tools, libraries, among others, in addition to spending a lot of computational resources. The MetricMiner web application makes use of the power of cloud computing to scale. In this way, researchers do not have to worry about resources. The Mezuro focuses on supporting software developers, using reference values (metadata) to obtain metrics, and pointing out possible problems in the source code.

Depsy [11], a project developed by ImpactStory, is a pioneering platform that seeks to provide significant incentives for scientists and developers, granting the software the necessary recognition so that it can be cited without being linked to a research paper that describes it, in addition to having allied tools that allow Depsy to perform searches in Python and R language packages, and also allows to detail the impact measurements in the scientific software and the papers

that refer to them. Depsy provides several statistics about the packages, such as the number of downloads, the number of citations, and the value of PageRank in the dependency network, this can be very useful for non-computer readers, since it indicates statistics based on percentiles, which allows knowing the impact of each tool software in the scientific field.

PyDriller [12], is a Python framework that helps developers in analyzing Git repositories. With PyDriller you can easily extract information about commits, developers, modified files, diffs, and source code. However, it does not have visualizations, only metadata extraction. likewise, Perceval [13], is a tool that simplifies the collection of project data by covering the most popular tools and platforms related to contributing to Open-Source development, thus enabling the definition of software analytics. offers the results in the flexible JSON format and gives the possibility to connect the results with analysis and / or visualization tools.

The Table 1 features are used to compare the VigHub tool, with tools and platforms that allow to store, extract, and analyze metadata from repositories and generate descriptive graphics. Some of the comparative features are taken from the paper [9]. If you are interested in trying the tool, you can access it through the link: http://eiscapp.univalle.edu.co/vighubjson/ and you can use the user: cgiohidalgo@gmail.com and password: testvighub.

Table 1.
Comparison of VigHub with other tools.

	VigHub	GitHub	Mezuro	Boa	MetricMiner	Depsy	PyDriller
Application web.	x	x	x	x	x		
Interface to query data.	x	x		x	x	x	x
Graphic interface to visualize data.	x						
Git repositories.	x	x	x		x		x
SVN repositories.			x		x		x
CVS repositories.			x		x	x	x
Implementa- tion of NLP.	x	x				x	
Prediction implementation.	x						

Source: Self-made.

III. Methodology

The tool model is based on forecasting processes technology and technological surveillance [14], [15]. This practice allows, through systematic processes, to capture and analyze information that leads to identify decision-making opportunities [16]. The tool model proposes four main phases of technological forecasting: planning, transform, analysis, and communication (Table 2).

Table 2.
Model for VigHub.

Phases of the VigHub tool	Description
Phase l - Planning	Need for information: source of information that provides structured or semi-structured metadata with information on specific topics of software development.
Phase ll - Extract, Transform and Load	Extraction: Extract the metadata from the GitHub platform with the API V3 through specific software development queries. Transform: Characterize the information coming from the queries generated in metadata, according to the level of relevance of the data. Load: Save the purified data in a relational database that allows consulting the information for each data obtained.
Phase lll - Analysis of the information	Analysis strategies: Implementation of computational techniques (libraries, subroutines, processes, methods) that allow processing the data to obtain information relationships that are not found in the data.
Phase lV - Information communication	Data representation: visualize the analyzed information to communicate ideas and explore all dimensions of the data. beneficiaries: Scientific and technological field (software development, academy, and industry).

Source: Self-made.

A. *Technical model*

The VigHub tool, in its functional version, is built up using mostly JavaScript programming language based with angular framework, complemented with Python, PHP, HTML and CSS and with libraries for graphics, Google Chart and D3. It is organized in containers to be scalable and modular (docker). The processes run in a container instance, and the processed information is stored in a PostgreSQL relational database. The user environment is completely decoupled from the server; views are generated in the browser using direct URI requests to the database.

Fig. 1 shows the architecture by components of the tool and indicates the sequential process after the user enters a query in the VigHub search engine. The construction of the tool follows the phases of the model described above.

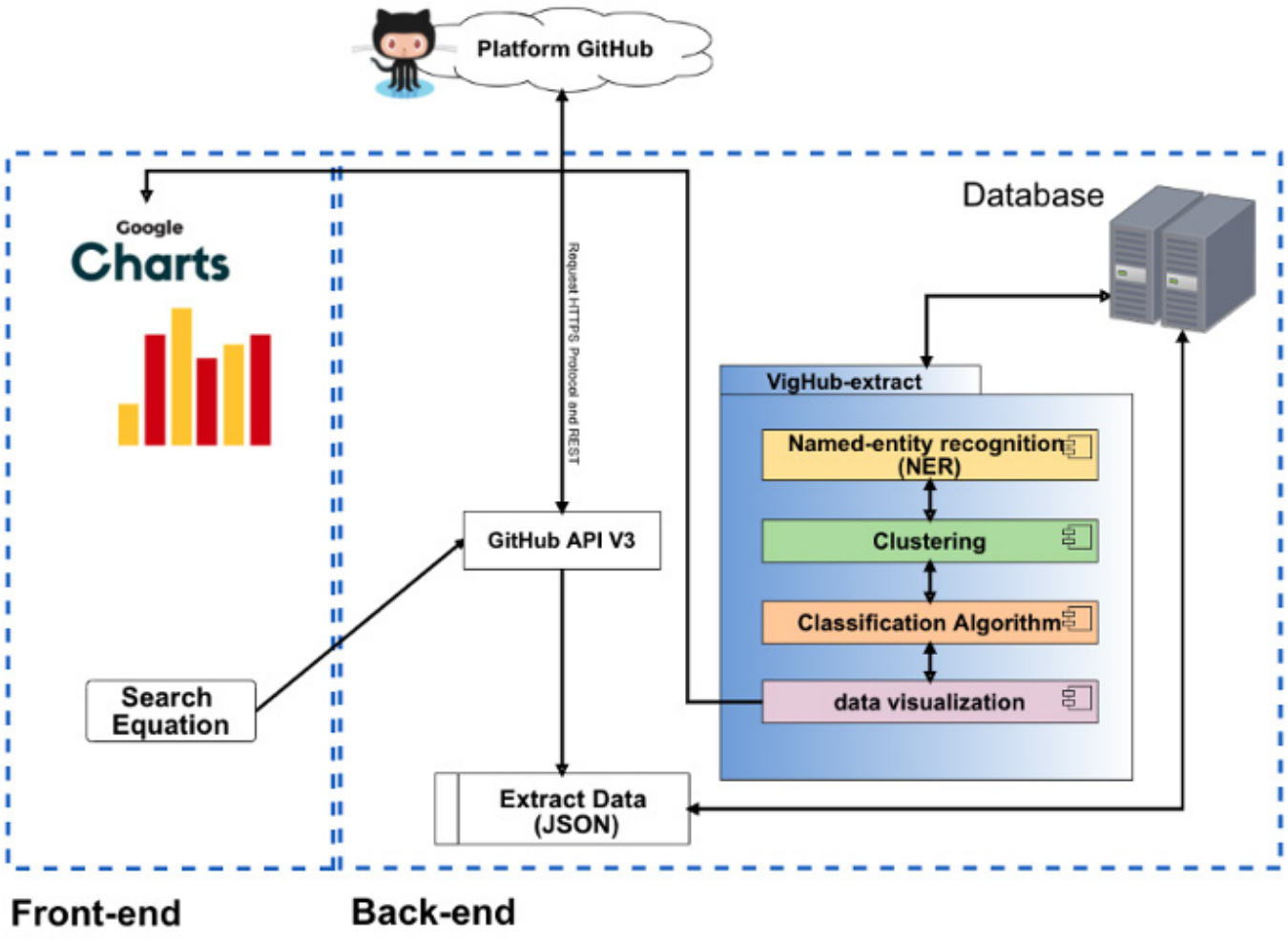

Fig. 1. Model for VigHub.
Source: Self-made.

From a topic of interest or research, in which we want to investigate to know the current state of technologies, keywords (search equation) are entered in the VigHub browser, this connects the API-GitHub.

The Git service sends a request, the Git service sends a request, for example: “api.github.com/search/repositories?q=”. where “q” is a variable that receives the entered query waiting for GitHub to get all the metadata that corresponds to the entered search equation. When git responds, get a JSON (JavaScript Object Notation) file, containing features of each repository like the query that was entered. Among the data are username, programming language, scores, number of forks, number of stars, number of branches, and other data. the VigHub service stores the JSON file in the database, assigning a user ID and a query ID.

The VigHub-extract service, from the database, extracts the data by each repository name and saves it in a new JSON object. Then, check that there are no inconsistencies in the data and URLs: that the repository belongs to a real GitHub user, that the URL is public and can be accessed by HTTPS protocol. If a repository does not have these criteria, it is removed from the query, it is also removed if the repository name appears more than once. Once the data is cleaned, these processes are run:

a) In the metadata of the repositories, there are some characteristics that are in the GitHub API, but they are not found explicitly, for example, in a query, how many users belong to any organization? or how many programming languages are used to develop the same system? Or what are the keywords of a topic? In this sense, the VigHub-extract service uses spaCy NER model [17], to answer the previous questions, where initially the data of a query is taken, followed by a pre-processing by performing a Split of the data (specifically the description of each repository), so hundreds of words are obtained, later, the words are touched, identifying the number of the appearance of each word in the different repositories, for common words (with, and, describe, tool, install, among others) it is applied stopwords method.

   In this step, the most relevant keywords are obtained for a query in VigHub, this is the input to find the type of technology by year, wherefrom a title, summary, or description of a repository it can be obtained that it is implemented. One more step is made to group the type of technology for each year. The process is like obtain a number of repositories and programming languages for each year. A value (language or year) is obtained in a repository, and it is consulted in the other repositories, if they are the same, it is touched. On the other hand, to find if a user who owns a repository belongs to an organization, the value "Factory" is obtained from the GitHub API if it appears true, it means that there is a direct URL to a company's web page. To obtain the name, scraping-dos are used.

   Finally, to make an annotation of the new data, in new queries with similar topics, the model is used, with a training of 80% and a test of 20%. In this step, the data is finally saved in a temporary JSON that is stored locally on the server, then this will be taken for the construction of the visualizations.

b) Identify the groups of programming languages (repository vs language), groups of the best repositories (stars vs clones), and groups of the most important (stars vs forks), a hierarchical clustering algorithm used scikit-learn with a maxclust criterion, which receives as input a matrix formed by features of the repositories (columns) and the number of repositories found (rows). The features to consider are 11: "forks", "forks_count", "stargazers_count", "watchers_count", "watchers", "size", "default_branch", "master", "open_issues", "open_issues_count" and "subscribers_count". As for each query, the results are different in quantity, the elbow method is used to find the appropriate number of clusters.

   In this step, the data is finally saved in a temporary JSON that is stored locally on the server, then this will be taken for the construction of the visualizations.

c) For a researcher, student, developer, the best programming language is very important to develop a software project on a specific topic. For example, for the development of business software, perhaps the best language can be java or Python, or for car sales software, it may be better to use JavaScript or C++. This will depend on the needs of the user; however, you can have help, knowing how many previous developments there are in a programming language. Thus, a classifier is implemented, using the Naive Bayes model. Where the number of the characteristic of stars, number of copies, number of programming languages, year of creation, and year of maintenance is used. The class is binary, having 0 (not viable) and 1 (viable). The scikit-learn GaussianNB model is used, the configuration parameters are arranged to use 80% in train and 20% in test. The classifier compares a language that the user enters by keyboard in the VigHub interface, the model is executed, and shows the probability that a project is viable given a programming language. For the verification, the measurement f1 score and the ROC curve are used, data that is also shown in the VigHub interface. In this step, the data is finally saved in a temporary JSON that is stored locally on the server, this will then be taken for the construction of the visualizations.

Finally, the VigHub-extract service sends the JSON information to a function of the Google Charts [18] library to generate descriptive graphs, among which the treemap, bubble graphs, bars, tables, and fishbone (timeline) can be found.

IV. Results

A. *Case study 1*

To know the correct use of VigHub, a method based on steps is proposed to identify the Software-focus research, keywords, and query strings. To show the method in operation, an example is made to contribute to the development of software and education. The Virtual Learning Environments (VLEs) have been taken as the subject of study. These innovative systems are currently being developed to encourage education on the web. These systems include assessment tools, student tracking, collaboration, and communication. They operate 24 hours per day, seven days per week, allowing institutions to teach not only traditional full-time students but also those who can't attend in person, for example, in distance-learning courses or night classes [19].

1) *Software-focus research*

The different keywords and the different synonymous ways of referring to the Virtual Learning Environment were obtained with the support of experts. The keywords found are Virtual-Learning-Environment, Learning Management System, Content Management Systems, and Learning Content Management System.

2) *Making of query strings*

The search equations are created with the keywords obtained in the previous phase. These are expressions composed by the keywords and by Boolean operators admitted by the GitHub API. The result is: ("Virtual-Learning-Environment" AND "tech"), ("Virtual-Learning-Environment" or "Learning Management System" or "Content Management Systems" or "Learning Content Management System").

3) *Search results*

The search strings entered on the VigHub search engine, and 226 repositories related to VLEs are obtained and the technological maps are generated. The results show speculations informed based on the timeline generated by the VigHub tool, where it can be compared between years if they present different developments per year or changes in programming languages or changes in technologies.

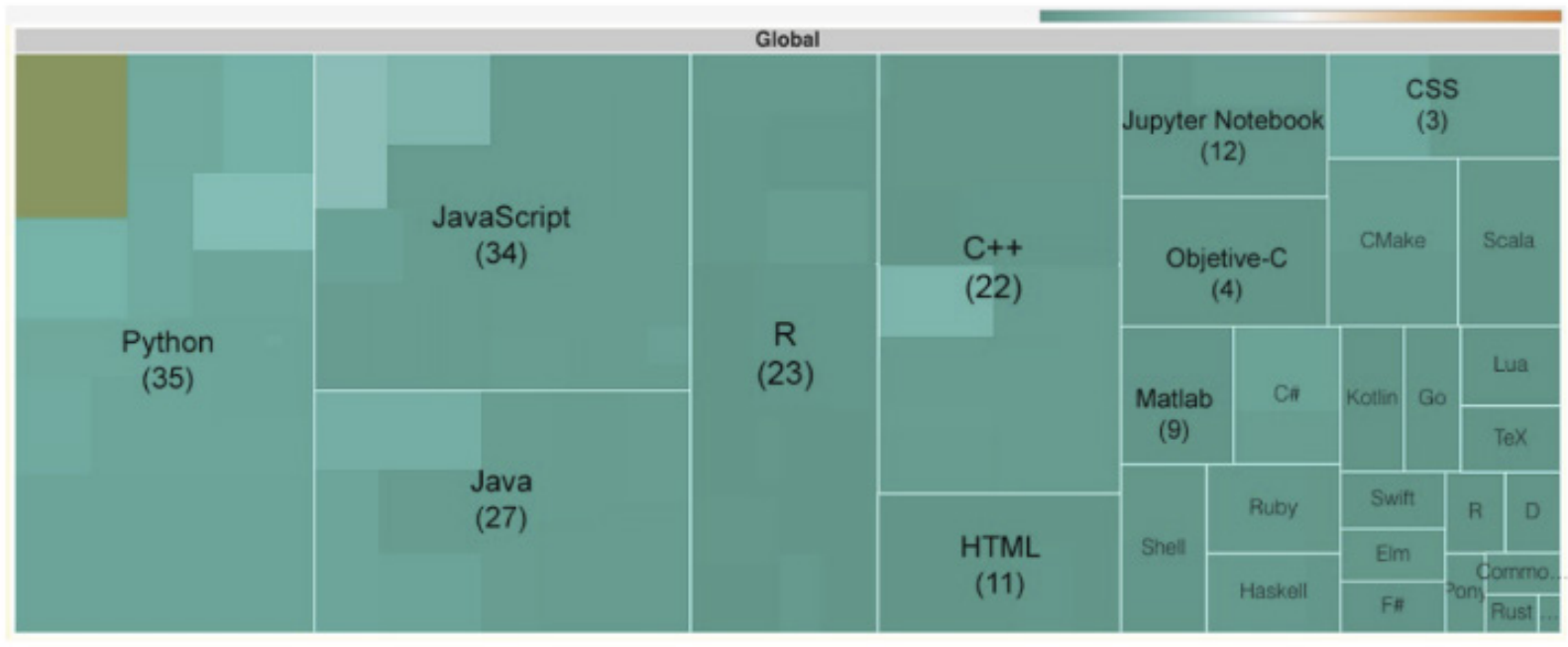

Fig. 2. Language and importance of the repository in the subject; according to its size: the higher is the picture that covers a language, the greater the number of developments. According to its color: the orange color identifies the most relevant languages and developments in GitHub.
Source: Self-made.

Technological map of the programming languages: Knowing which programming languages are used to develop VLEs enables you to determine current and future market trends and technology needs. In addition, this information could be considered for making decisions about a new product focused on Virtual-Learning-Environment. Considering the 226 projects, in Fig. 2, the relevant repositories/projects are shown according to the number of stars, copies, and visits of each one. In this case, the projects with greater relevance according to these characteristics are PHP, Ruby, and C#. In addition, the graph shows that in the technological development of VLEs in GitHub, 20 programming languages have been used. According to the number of repositories, the most used are Python, JavaScript, Java, and R, which have 35, 34, 27, and 23, respectively. Languages such as Matlab, Ruby, Jupyter Notebook, C++ and HTML, have between 22 and 8 repositories. Finally, languages such as C#, Go, Objective C, TypeScript, have between 7 and 1 repositories.

Technological map of the evolution of the topic: Timelines are possibly one of the most powerful ways to visualize in any area, activity, or present certain events. With the help of timelines of a technological investigation, you have the possibility to highlight important tools or development base, indicate how it should be developed in chronological order, explain the progress of your project, or highlight the milestones in the use of programming languages or specific technologies. Based on an exhaustive analysis of the changes concerning technologies and paradigms between 2011 and 2021, it was possible to obtain a specific perspective to identify which were the most important projects, the trending languages, and the types of technology used for each year (Fig. 3). For the construction of this graph, the results of the VigHub tool are used, however, the visualization has been modified manually to obtain a compiled result. On the other hand, the VigHub results show the timeline, but in this paper, the results were complemented since some repositories do not contain information in GitHub, but in bibliographic sources such as Scopus and ScienceDirect. It is important to mention that each year only the most relevant technologies are described, that does not mean that there were no projects using other technology.

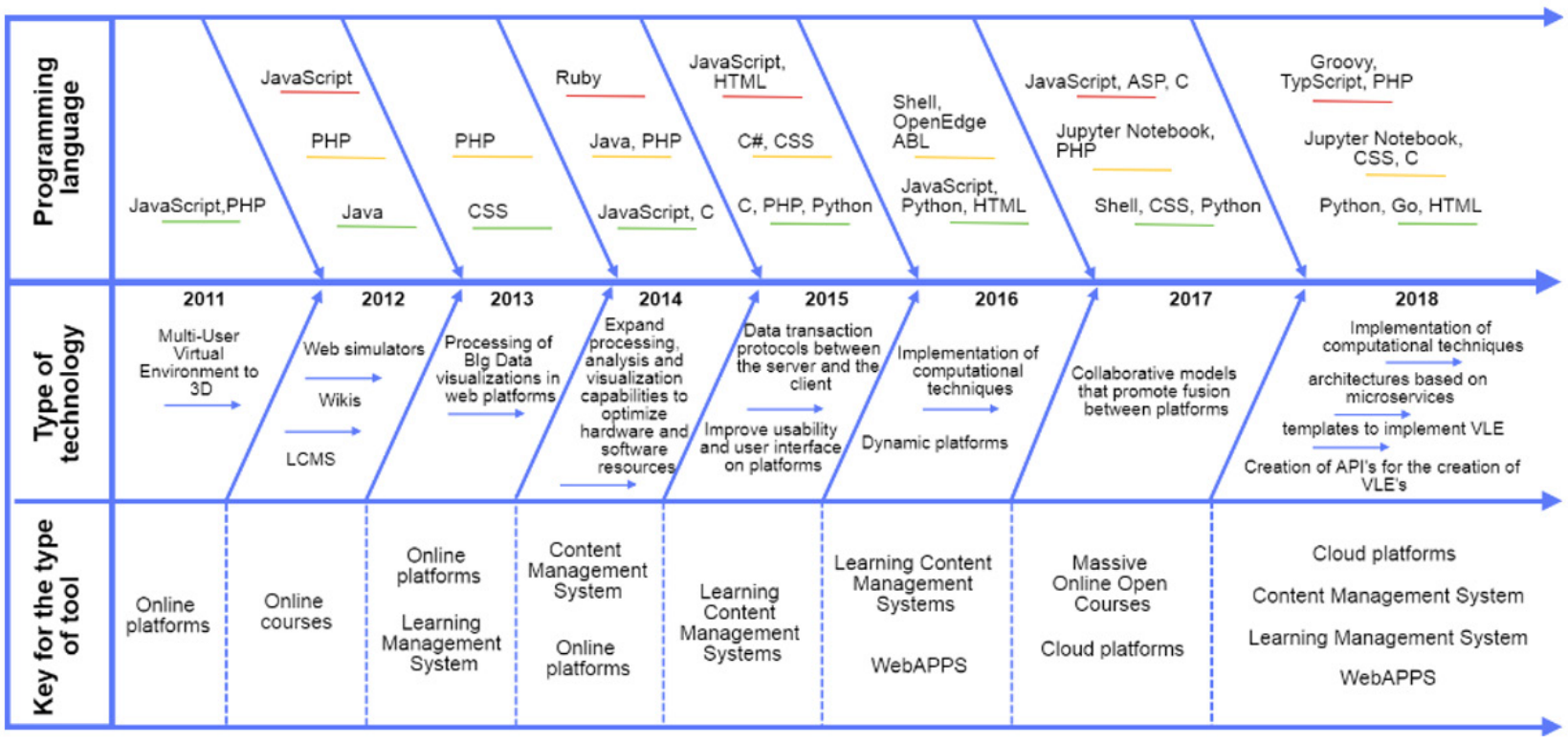

Fig. 3. The most relevant language with green, orange, and red; green indicates the most used language per year. The most important developments for this research are shown at the top of the line that corresponds to the years. Trends and classification by category for each year are shown at the bottom of each line.
Source: Self-made.

2011: There were two developments regarding learning environments; they were developed in JavaScript and PHP. Both are platforms that allow the creation of multi-users in virtual worlds created with Second Life and OpenSim [20], [21].

2012: In this year, repositories developed in Java, such as PHP and JavaScript, can be found. These developments focus on the creation of web simulators as a teaching method [22]. API's that allow the integration between web platforms [23]. The use of wikis is extended to projects like [24], [25]. Finally, the use of Learning Content Management (LCMS) as Moodle and Joomla becomes a trend in higher education institutions [26], [27].

2013: The development of VLEs decreases. Perhaps this is because in the previous year, many projects were developed, and in 2013 only updates were made, but not new inventions. It is important to mention that the few projects are focused on data processing, highlighting the use of Big Data [28], [29]. The views on web platforms are developed by using the PHP language.

2014: The efforts made by the developers to implement processing, analysis, and visualization capabilities to optimize hardware and software resources were more significant than in the previous years. Tools, algorithms, and languages are combined to create VLEs with automatic grading systems [30], quantitative qualifications [31], evaluation with software unit tests [32]. The languages highlighted during a year are JavaScript, Java, and C.

2015: The amount of data becomes a problem for VLEs. Therefore, it can be highlighted that, in this year, the processing of information is the main activity. Data transaction protocols are created between the server and the client. The languages used are C and Python, in a variety of projects [33], [34]. Also, this year, there is a remarkable growth to improve the usability and user interface on the platforms. The implementation of JavaScript, HTML, and CSS, is demonstrated in several projects [30], [35], [36].

2016-2017: The VLEs begin to implement computational techniques that allow to automate data analysis processes, increasing the use of JavaScript and Python, applying NLP and Machine learning to obtain data with added information [37], [38].

2018-2019: The implementation of computational techniques in the VLE's makes the latter dynamic. Also, it prevents them from using many software resources. For this year, collaborative models that promote the intersection of platforms are created. Such platforms can add resources, improve teaching and learning, called Massive Online Open Courses (MOOC). Some projects have achieved good results [39]-[41].

2020-2021: The growth of the development of VLEs is remarkable, about 40% above the previous ones. The most used languages are Python, Go, and Jupyter Notebook. The projects are varied, there are some that implement computational techniques [40], [42], others use microservices-based architectures to optimize processes [43], [44], others create templates for users in order to implement their own VLEs [45], [46], and finally, some focus on the creation of API's for the creation of VLEs as servers in distributed networks [47], [48].

Successful users and organizations: Table 3 and Table 4, shows the top organizations and users who have been referenced the most or made the top 10 on GitHub. These users generally appear as collaborators on projects in VLE. To obtain this data, topics and collaborators are taken in the GitHubAPI, then a filter is made with collaborations by topic function, and we obtain the value of participation in projects for each user or organization. These tables are created manually, although the same results are displayed in the VigHub interface frequency of times a project appears on GitHub.

TABLE 3.
THE ORGANIZATIONS THAT ARE CURRENTLY ACTIVE IN THE TOPIC ARE SHOWN.

Organizations	Participation in projects	Country
hysnsec	3	USA
mikeizbicki	2	(USA)Berkeley, CA
FenixEdu	2	(USA) Santa Fe, NM
sloodle	2	(USA) San Francisco, CA
Blosm	1	USA
gradecraft-development	2	South West England, UK
ATutor	2	(USA) Berkeley, CA

Source: Self-made.

TABLE 4.
THE USERS THAT ARE CURRENTLY ACTIVE IN THE TOPIC ARE SHOWN.

Name/Username	Repository	Country
David Forester (drforester).	34	USA
viniciusvec.	27	USA
JUN ZENG (SOYJUN).	28	USA
Victor I. Afolabi (victor-iyiola).	16	Nigeria
Fernando Cillero (fernando24164).	11	España
David (CoxDavidStCox).	10	Noruega
Rabindra Nath Nandi (robi56).	12	Bangladesh

Source: Self-made.

D. Case *Study 2*

The main advantage of VigHub is that it has a graphical interface to visualize data after having processed them, which allows the user to analyze the data in the same window. In addition, the implementation of computational techniques allows extracting, processing, and analyzing the most relevant data for each generated query. The tool has been launched to the academic and business public, which has made it possible to identify the importance of this in different fields.

In the industry, two Colombian companies have made specific consultations on VigHub. The first one is the company "Soluciones Integrales en Tecnologias de la Información" (SITI), which carries out academic management processes in the southwest of the country. The second company is "Soluciones y Suministros para Ingenierías" (SSI), which focuses on constructing sensors for counting and classifying vehicles.

These two companies had used the GitHub search engine to find out if repositories contained information (metadata) that could be useful for their specific projects, but his results were unfavorable. It is important to mention that this result may be because the study is limited to a few repositories. Table 5 shows the queries entered by each company, the result of the consultation with GitHub, and the result with VigHub.

TABLE 5.
INDUSTRY CASE IN QUERIES WITH GITHUB AND GITHUB.

Company	Query	Result GitHub	Result VigHub
S.I.T.I	Administrative system of educational processes.	2	35
SSI	Counting and classification of vehicles by sensor.	0	28

Source: Self-made.

V. CONCLUSIONS

One of the purposes of VigHub is to support researchers to obtain some of the state-of-the-art of a specific technology. At the Universidad Nacional de Colombia, they are developing an online virtual judge for programming [14].

To learn about virtual programming judges, VigHub was used, resulting in a state-of-the-art [49]. Likewise, at the Universidad del Valle (Colombia), a systematic review was developed to find tools for genomics. VigHub was used to obtain the technological maps of these tools. In both cases, the writing works were accepted in high-impact journals.

VigHub was designed to support developers, researchers, and entrepreneurs to gain a grounding in building a project or software development, both in industry and academia. Due to its modular architecture, the tool integrates new processing algorithms and new data sources. In the future, data from SVN and CVS repositories will be integrated to enhance VigHub's analytics capabilities.

It is essential to mention that this proposal intends to complement obtaining the State-of-the-art techniques. However, it does not intend to replace the traditional form through searches in scientific databases.

Acknowledgements

Thanks to the Universidad del Valle (Colombia) and Universidad del Cauca (Colombia), Faculty of Engineering, Systems, and Information Engineering School and the fundation CEIBA (Centro de Estudios Interdisciplinarios Básicos y Aplicados) (Nariño, Colombia).

References

[1] A. Peralta & F. P. Romero, "Decision making from knowledge obtained after previous behavior analysis. Practical implementation to project management of software development," *Rev Cintex*, vol. 20, no. 2, pp. 97–111, Nov. 2015. https://revistas.pascualbravo.edu.co/index.php/cintex/article/view/26

[2] D. Güemes-Peña, C. López-Nozal, R. Marticorena-Sánchez & J. Maudes-Raedo, "Emerging topics in mining software repositories: Machine learning in software repositories and datasets", *Prog Artif Intell*, vol. 7, no. 3, pp. 237–247, Mar. 2018. https://doi.org/10.1007/s13748-018-0147-7

[3] O. Meqdadi, N. Alhindawi, J. Alsakran, A. Saifan & H. Migdadi, "Mining software repositories for adaptive change commits using machine learning techniques," *Inf Softw Technol*, vol. 109, pp. 80–91, May. 2019. https://doi.org/10.1016/j.infsof.2019.01.008

[4] M. Garriga, "Towards a taxonomy of microservices architectures," presented at *International Conference on Software Engineering and Formal Methods*, SEFM, TLS, FR, 27-29 Jun. 2018. https://doi.org/10.1007/978-3-319-74781-1_15

[5] K. Bakshi, "Microservices-based software architecture and approaches," presented at *Aerospace Conference Proceedings*, IEEE, Big Sky, MT, 4-11 Mar. 2017. https://doi.org/10.1109/AERO.2017.7943959

[6] Y. San Juan & F. Romero, "Management, extraction and storing sources for technological watch and competitive intelligence," presented at *VIII Congreso Internacional de Tecnologías y Contenidos Multimedia*, CITCM, HAB, CU, 19-23 Mar. 2018.

[7] M. A. Saied, A. Ouni, H. Sahraoui, R. G. Kula, K. Inoue & D. Lo, "Improving reusability of software libraries through usage pattern mining," *JSS*, vol. 145, pp. 164–179, Nov. 2018. https://doi.org/10.1016/j.jss.2018.08.032

[8] R. Dyer, H. A. Nguyen, H. Rajan & T. N. Nguyen, "Boa: Ultra-large-scale software repository and source-code mining," *ACM Trans Softw Eng Methodol*, vol. 25, no. 1, pp. 1–34, Dec. 2015. https://doi.org/10.1145/2803171

[9] F. Z. Sokol, M. F. Aniche & M. A. Gerosa, "MetricMiner: Supporting researchers in mining software repositories," presented at 2013 *IEEE 13th International Working Conference on Source Code Analysis and Manipulation*, SCAM, EIN, NL, 22-23 Sept. 013. https://doi.org/10.1109/SCAM.2013.6648195

[10] C. M. Filho, "Kalibro: Uma ferramenta de configuração e interpretação de métricas de código-fonte," *Projeto de conclusão de curso*, USP, SP, BR, 2009. https://www.ime.usp.br/~cef/mac499-09/monografias/carlos-morais/Monografia.pdf

[11] D. S. Chawla, "The unsung heroes of scientific software," *Nature*, vol. 529, no. 7584, pp. 115–116, Jan. 2016. https://doi.org/10.1038/529115a

[12] D. Spadini, M. Aniche & A. Bacchelli, "PyDriller: Python framework for mining software repositories," presented at *26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE, NYC, NY, USA, 4-9 Nov. 2018. https://doi.org/10.1145/3236024.3264598

[13] S. Dueñas, V. Cosentino, G. Robles & J. M. Gonzalez-Barahona, "Perceval: software project data at your will," presented at *40th International Conference on Software Engineering: Companion*, ICSE-Companion, GBG, SE, 27 May.-3 Jun. 018. https://ieeexplore.ieee.org/document/8449430

[14] J. J. Ramírez-Echeverry, F. Restrepo-Calle & F. A. González, "Uncode: interactive system for learning and automatic evaluation of computer programming skills", presented at *10th International Conference on Education and New Learning Technologies*, EDULEARN, PMI, ES, 2-4 Jul. 2018. https://doi.org/10.21125/edulearn.2018.1632

[15] E. Ortíz, "La evaluación del impacto científico en las investigaciones educativas a través de un estudio de caso," *REDIE*, vol. 17, no. 2, pp. 89–100, May. 2015. https://www.scienceopen.com/document?vid=0de24d4c-b9e3-4739-b394-346f7480b4fe

[16] A. Berges-García, J. M. Meneses-Chaus & J. F. Martínez-Ortega, "Methodology for evaluating functions and products for technology watch and competitive intelligence (TW/CI) and their implementation through web," *PEI*, vol. 25, no. 1, pp. 103–113, Jan. 2016. https://doi.org/10.3145/epi.2016.ene.10

[17] SpaCy, "Industrial-Strength Natural Language Processing in Python," Accessed: Oct. 18, 2019. [*Online*]. Available: https://spacy.io/

[18] Google Developers. "Google Charts." Accessed: 2018. [*Online*]. Available: https://developers.google.com/chart

[19] P. T. Goeser, F. G. Hamza-Lup, W. M. Johnson & D. Scharfer, "VIEW: A Virtual Interactive Web-based Learning Environment for Engineering," *AEEE*, vol. 2, no. 3, pp. 1–24, Dec. 2011. https://doi.org/10.48550/arXiv.1811.07463

[20] WISE-Community, "WISE VLE," Feb. 25, 2015. [*Online*]. Available: https://github.com/WISE-Community/WISE-VLE--Deprecated--

[21] F. Supriadi, M. Agreindra Helmiawan, Y. Y. Sofiyan & A. Guntara, "A Model of Virtual Learning Environments Using Micro-Lecture, MOODLE, and SLOODLE," presented at *8th International Conference on Cyber and IT Service Management*, CITSM, PGX, ID, 23-24 Dec. 020. https://doi.org/10.1109/CITSM50537.2020.9268785

[22] Knowm, "Proprioceptron," Oct. 27, 2012. [*Online*]. Available: https://github.com/knowm/Proprioceptron

[23] Yjwong, "com.nuscomputing.ivlelapi," Aug. 14, 2012. [*Online*]. Available: https://github.com/yjwong/com.nuscomputing.ivlelapi

[24] 40thieves, "WikiVLE," Jun. 23, 2012. [*Online*]. Available: https://github.com/40thieves/WikiVLE

[25] Jbittencourt, "massinha," Jul. 5, 2012. [*Online*]. Available: https://github.com/jbittencourt/massinha

[26] Conel, "moodle-1.9," Aug. 20, 2012. [*Online*]. Available: https://github.com/conel/moodle-1.9

[27] Elkuku, "JDevAndLearn," Jul. 28, 2012. [*Online*]. Available: https://github.com/elkuku/JDevAndLearn

[28] Champiewebfolio, "CloudPod," Jan. 6, 2013. [*Online*]. Available: https://github.com/champiewebfolio/CloudPod

[29] RheoDesign, "AAVS-Beijing," Oct. 23, 2013. [*Online*]. Available: https://github.com/RheoDesign/AAVS-Beijing

[30] Roxolan, "vlemean," Aug. 11, 2015. [*Online*]. Available: https://github.com/roxolan/vlemean

[31] luistp001, "LT-Autograder," Sep. 16, 2012. [*Online*]. Available: https://github.com/luistp001/LT-Autograder

[32] StephenBergeron, "RubySoup," Apr. 1, 2014. [*Online*]. Available: https://github.com/StephenBergeron/RubySoup

[33] Deepapanwar, "vle," Jun. 16, 2015. [*Online*]. Available: https://github.com/deepapanwar/vle

[34] Soyjun, "Implement-ODR-protocol," Apr. 10, 2015. [*Online*]. Available: https://github.com/SOYJUN/Implement-ODR-protocol

[35] Brukmoon, "eduqo-vle," Apr. 23, 2015. [*Online*]. Available: https://github.com/Brukmoon/eduqo-vle

[36] Sykonba, "PeerReviewSystem," Nov. 2, 2015. [*Online*]. Available: https://github.com/Sykonba/PeerReviewSystem

[37] DavidStCox, "nlp-vle," Apr. 10, 2017. [*Online*]. Available: https://github.com/DavidStCox/nlp-vle

[38] Lumeng, "univ-washington-machine-learning-python-virtualenv," Dec. 3, 2017. [*Online*]. Available: https://github.com/lumeng/univ-washington-machine-learning-python-virtualenv

[39] Blosm-org, "blosm-core," Sep. 30, 2017. [*Online*]. Available: https://github.com/blosm-org/blosm-core

[40] Cvgokhale, "Course-Completion-Rate-Prediction," Jul. 16, 2017. [*Online*]. Available: https://github.com/cvgokhale/Course-Completion-Rate-Prediction

[41] Victor-iyiola, "navigating-a-virtual-world-using-dynamic-programming," Nov. 26, 2017. [*Online*]. Available: https://github.com/victor-iyiola/navigating-a-virtual-world-using-dynamic-programming

[42] Charvi5, "VirtualLearning-Analysis-Classification," Apr. 4, 2018. [*Online*]. Available: https://github.com/charvi5/VirtualLearning-Analysis-Classification

[43] Viniciusvec, "hackops," Mar. 2, 2018. [*Online*]. Available: https://github.com/viniciusvec/hackops

[44] Fernando24164, "breakfast_docker," Feb. 2, 2018. [*Online*]. Available: https://github.com/fernando24164/breakfast_docker

[45] pupilfirst, "pupilfirst," Aug. 2, 2021. [*Online*]. Available: https://github.com/pupilfirst/pupilfirst

[46] tparisi, "LearningVirtualReality," Mar. 4, 2016. [*Online*]. Available: https://github.com/tparisi/LearningVirtualReality

[47] Aayushi15061997, "Reinforcement_Learning_ThompsonSampling," Jan 29, 2018. [*Online*]. Available: https://github.com/aayushi15061997/Reinforcement_Learning_ThompsonSampling

[48] The-Dank-Network, "TDVLE-API," Mar. 23, 2019. [*Online*]. Available: https://github.com/The-Dank-Network/TDVLE-API

[49] C. G. Hidalgo, V. A. Bucheli, F. Restrepo-Calle & F. A. González, "A strategy based on technological maps for the identification of the state-of-the-art techniques in software development projects: Virtual judge projects as a case study," in *Communications in Computer and Information Science*, C. J. Serrano & J. Martínez-Santos, Cham, CH: Springer, 2018, vol. 885, pp. 338–354. https://doi.org/10.1007/978-3-319-98998-3_27

Carlos Giovanny Hidalgo-Suarez. Systems Engineer. Master in Engineering. PhD student at Universidad del Valle (Colombia). member of the artificial intelligence research group (GUIA) https://orcid.org/0000-0003-2308-0720

Victor Andres Bucheli-Guerrero. Systems Engineer. Master in Engineering and Computing. PhD of Engineering. Associate Professor Universidad del Valle (Colombia). Director of the Artificial Intelligence Research Group (GUIA), Senior Researcher. https://orcid.org/0000-0002-0885-8699

Hugo Armando Ordoñez-Eraso. Systems Engineer. Especialist in Administration Computer, Master in Engineering and Computing, PhD of Engineering. Professor Universidad del Cauca (Colombia). Member of the GTI research group. https://orcid.org/0000-0002-3465-5617