Development of an Educational Web Application with an architecture base on the cloud, to facilitate the learning process in programming Logic through pseudocode in Kichwa

Pineda Vega, Jean Pierre y Montalvo Laica, Evelyn Noemí

Departamento de Eléctrica y Electrónica

Carrera de Ingeniería en Software

Artículo académico, previo a la obtención del título de Ingeniera en Software

Ing. Rolando Marcelo Álvarez Veintimilla

03 de septiembre del 2021

Development of an Educational Web Application with an architecture base on the cloud, to facilitate the learning process in programming Logic through pseudocode in Kichwa

Noemí Montalvo L., Jean Pineda V., Marcelo Alvarez V.

Universidad de las Fuerzas Armadas ESPE, Latacunga, Ecuador
{enmontalvo, jppineda, rmalvarez}@espe.edu.ec

Abstract. The purpose of this project is to develop an educational web application that facilitates the learning of programming through the use of pseudocode syntax in Kichwa. Making use of cloud-based services such as cloud functions for servers, CloudBuild for deployment and firebase for hosting and database. This application will link instructions, statements, and operators, which form the basis of programing logic, enabling the generation of a practical and understandable pseudocode that facilitates the teaching of programming.

Throughout the development of this model the follow is defined: the structure and syntax of the pseudocode in Kichwa; the creation of data dictionaries, restricted words and terms; definition of data typology, int, string; definition of the control structure syntaxis, statements: if, then, else loop for, while, definition of the architecture of the code analyzer that determines the process of translation of the pseudocode in Kichwa, the creation of the graphic interface. The web application performs an analysis of the pseudocode written by the user; then a translation is carried out through the pseudocode in Kichwa, for its further compilation and execution.

Keywords: Cloud Computing, Pseudocode, Programming logic, Kichwa.

1 Introduction.

Currently, there are more than 8500 programming languages registered, of which 2,400 were developed in the United States, 600 in the United Kingdom, 160 in Canada and 75 in Australia [1]. Most of this programming languages are developed and structured in English, among the most well-known are Pascal, C, C++, Java, Visual Basic, Php and the most recent ones Python of Holland, Ruby of Japan, and Lua of Brazil. The percentage of written languages it's distributed at the rate of 80% in English, 15% in French and 2% in other languages [2].

There are very few platforms and/or tools in Latin America that allow to program in Spanish, even less in native languages. Ecuador is no exception, with the Kichwa Language [3]. These limitations make the learning process of programming logic complex for Kichwa-speaking communities due to the unavailability of the content in their native language. For this reason the programming tools in native languages are a step towards diminishing the digital gap that has existed in the indigenous communities[4].

Thus, this project of developing an educational web application to facilitate the learning of programming through pseudocode in Kichwa came to be. It allows users to train through the application and to acquire programming skills.

This article is divided in six sections: The first section, the Introduction, focuses on the general details of the project. The second section, Background, shows a study of the existence of programming languages in the Kichwa language. The third section, Problem Formulation, describes the problem to be addressed. The fourth section, Methodology, details the development and architecture of the platform. The fifth section, Analysis of Results, where the experimental results that validate the proposal are shown. The sixth section, Conclusions, where the results obtained are presented to propose future works based on it.

2 Background

Currently, the most widely known and used programming languages according to Stephen Cass are: Phyton, C, Java, C++, C#, R, JavaScript, PHP, Go, Swift; developed and structured in English [5]. Taking in consideration that in the Latin American region the majority of the population are Spanish-speaker, a linguistical barrier for the programming logic is generated.

There are several native languages, such as, Mazateco (Mexico), Shuar (Ecuador), Guaraní (Paraguay), but the case of Quechua refers to a linguistic family distributed in different countries of South America: Colombia, Ecuador, Peru, Bolivia, Chile, and Argentina. In this extensive territory, the quechuas languages are given different names: In most parts of Peru and Bolivia the quechua (qhichwa, qhiswa or qichwa in the indigenous language); in Ecuador el Kichwa (quichua) [6]. In Latin America it is affirmed that the Kichwa it's a language that spread from the north of Argentina to the south of Colombia, covering the current territories of Ecuador, Perú y Bolivia [7], it is estimated that approximately 420 indigenous languages belong to 99 families, which make use of the philology, apart from the Spanish [8].

In Ecuador, two major Kichwa-speaking areas can be distinguished: the Andean zone and the Amazonian zone; the former extends along almost the entire Andean corridor, from the northern provinces of Imbabura and Carchi to the south of Zamora-Chinchipe; the latter covers four of the five Amazonian provinces: Sucumbíos, Orellana, Napo and Pastaza. Additionally, Kichwa speakers have been recorded in Guayas and El Oro, on the coast of the country, as well as in two of the Galapagos Islands (Santa Cruz and San Cristóbal) [9].

Currently, programming logic is considered one of the most demanded skills, therefore, its approach in the educational context [10], the fact that educational material is taught in a language different from the native one in the case of programming, decreases the participation and motivation of people in the teaching and learning process [11]. For this reason, programming platforms in native languages are a step towards decreasing the difficulty in learning programming that has existed for a long time in indigenous communities.

3 Formulation of the problem

The platforms for teaching programming logic are not designed for its use in indigenous communities. These tools are mainly offered in English [12], which hinders the teaching and learning process in other languages. In the case of Ecuador there is no platform that facilitates the introduction of the basics of programming logic for indigenous communities.

In Latin America as a whole, 50% of the indigenous population lives in the countryside. Considering that the Kichwa language is the most widely used indigenous language in Ecuador, even though it is not spoken throughout the country; 7 out of 10 people who identify themselves as indigenous live in rural areas [13] located mainly in the Sierra and eastern [14].

According to Howard, the last INEC census on population and housing was conducted in Ecuador in 2010. It recorded a proportion of 4.08% of the Kichwa-speaking population in absolute terms [15], i.e., indigenous people who do not speak Spanish or another second language. Within this population, 50% have the need to learn to program as part of their high school and higher education studies [16].

The indigenous population still faces difficulties within the education in new technologies, unlike its counterpart in the city.[17] A fact that is aggravated by the reforms to the Organic Law of Intercultural Education (LOEI) which caused the closure of approximately 13,834 community schools that were suspended in 2017. This has a negative impact on the indigenous communities, as evidenced by the desertion of 50% of the students belonging to this sector of the population.[19] This causes stagnation in the development of the towns and creates difficulties in learning subjects related to new technologies such as programming logic [20].

There are different systems that seek to support the education in indigenous communities using the native language, like the web application for teaching the Kichwa language [21]. However, this system is focused on the divulgation of the language, it serves of no use for the new technologies. Given that the documentation of programming languages such as Python is in Spanish, English or other more commonly used languages, we can assume that there is no educational web application that facilitates the teaching-learning process of programming logic through pseudocode in Kichwa. Taking into consideration the above mentioned, the present research work aims to develop a web application that teaches program logic. This allows students to learn programming, developing their skills through a GUI. This is done with the intention to provide a learning opportunity for a globalized world.

4 Methodology

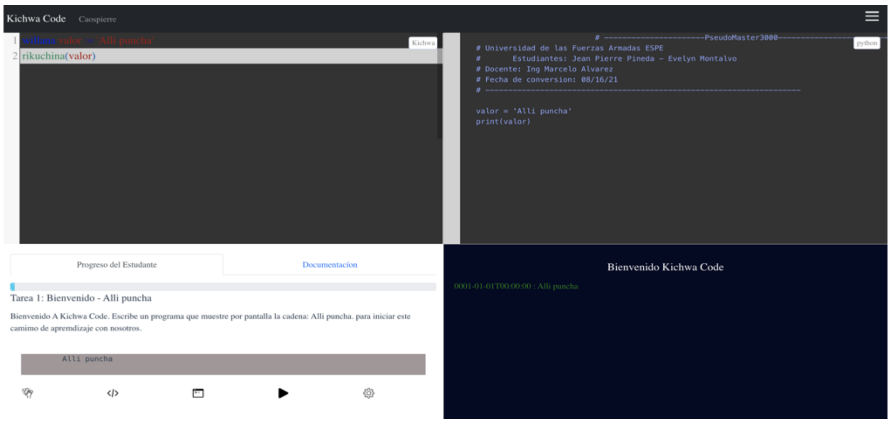
4.1 Kichwa Code

Kichwa Code was created with the purpose of aiding students who start learning programming logic. As a first step, the pseudocode is used to introduce basic principles regarding the handling of basic programming expressions, variables, control structures, etc. Avoiding the existing complexity of a programming language such as Python or Java for the new users. Thus, it is intended to facilitate the writing of algorithms through a pseudocode that uses expressions in Kichwa.

Upon logging into the application the user will be presented with the registration and login option, that provides access to the main page of the web application and its two modules:

Module 1 Code Editor. - This is one of the main components in the web application, consisting of four panels: i) Kichwa pseudocode editor, ii) Activities, iii) Python code visualization, iv) Results visualization. Visualization of Python code, iv) Visualization of results, where the first panel allows to enter code in Kichwa, where the text adopts a different color depending on the type of syntax that handles the pseudocode. The second panel presents the following options: translate, compile and clean screen, that facilitates the interaction with the editor. The third panel allows the user to visualize the Python code generated from the pseudocode. The fourth panel simulates a computer terminal and displays the result of the compilation and the detected errors.

Module 2 Training - It is an implementation of the code editor that consists of two options: 1. Student progress, 2. Documentation. In the first option the user can find tasks proposed for the training course, which consists of exercises designed to generate new knowledge in programming logic using pseudocode in Kichwa. In the second option there are located the definitions and necessary concepts about the use of different terms and syntax used when writing pseudocode in Kichwa. The user's progress on the web application is tracked, as shown in Fig 1.



```
willana valor := 'Alli puncha'
rikuchina(valor)
```

```
                    # ---------------------PseudoMaster3000-----------------
# Universidad de las Fuerzas Armadas ESPE
#       Estudiantes: Jean Pierre Pineda - Evelyn Montalvo
# Docente: Ing Marcelo Alvarez
# Fecha de conversion: 08/16/21
# ------------------------------------------------------------------

valor = 'Alli puncha'
print(valor)
```

Tarea 1: Bienvenido - Alli puncha

Bienvenido A Kichwa Code. Escribe un programa que muestre por pantalla la cadena: Alli puncha. para iniciar este camimo de apremdizaje con nosotros.

```
      Alli puncha
```

Bienvenido Kichwa Code

```
0001-01-01T00:00:00 : Alli puncha
```

Fig 1. Module 2 Training

4.2 System Architecture

The platform has a cloud-based web architecture, consisting of a Frontend, which is implemented through firebase Hosting; and the Backend, designed from a set of cloud functions. Those functions are methods that encapsulate a specific logic, interconnected between components and functionally isolated, which allows a high scalability and maintenance. Each server exposes an API for its consumption, as shown in Fig 2.
Since the project uses google cloud platform services for its execution, the system has the capability of supporting Hight traffic of users, given the deployment executed with CloudBuild [29].

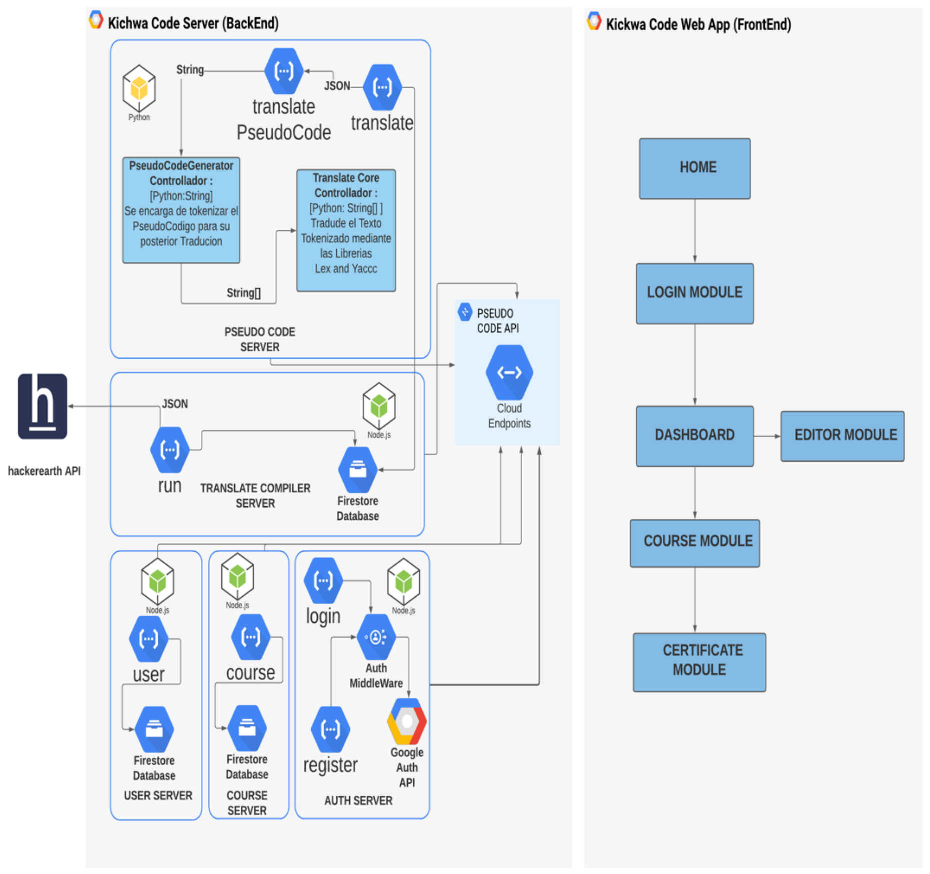


Fig 2. Model structure of the Kichwa Code System

The website consumes the backend API, which has a variety of endpoints for registration, user management, courses, translation and compilation.

In the case of translation, the website sends the pseudocode to the translation and compilation server, which in turn consumes an endpoint of the pseudocode server to obtain the respective Python translation, track errors and return the result for later visualization.

The core of the platform is the pseudocode server, which is responsible for generating Python code from the pseudocode in Kichwa. For the realization of this functionality Lex and Yacc are used.

Lex y Yacc

Lex is a program designed to generate lexical analyzers, which are the initial phase of a compiler that receives as an input a sequence of characters equivalent to the source code of another program, and returns an output consisting of tokens (lexical components) or symbols, which serve as input for a syntactic analyzer [22].

Yacc on the other hand is a program designed for the generation of syntactic parsers, which provides a tool to validate whether a sequence of characters respects the rules of a specific grammar [23].

4.3 Model of the structure and syntax of the pseudocode in Kichwa

This model defines the specification of the syntax of the pseudocode based on structures, which can be a token or a rule. A token being defined as a restricted word within the Python programming language, with its respective equivalence in Kichwa and a rule as a regular expression associated to a specific grammar as shown in Fig 3.

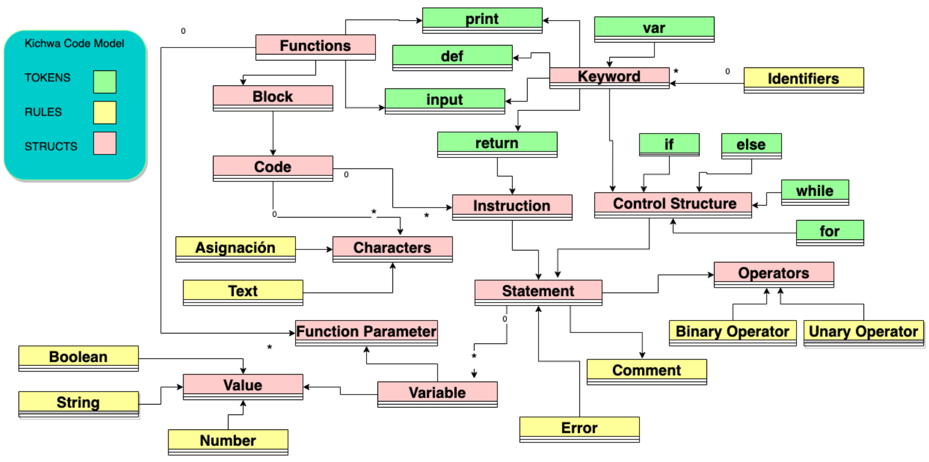


Fig 3. Model specification

Syntax of the Pseudocode

The terms used for the construction of the pseudocode syntax in Kichwa are not intended to be a literal translation of the English terms of the Python language, but rather, a semantic and syntactic equivalence of them, which were obtained from the Kichwa-Spanish dictionary issued by the Ecuadorian Ministry of Education [24], as shown in Table 1.

Table 1. Model tokens

TOKEN	Keyword Python	Keyword Kichwa	Example
WHILE	While	kaman	kaman(x==5)rurana
IF	If	mikuni	if(x=='munay')
ELSE	Else	shina	shina
E		kuturi	kuturi
DEF	def	kallari	kallari main()
RETURN	return	kutiman	kutiman 'munay'
VAR	var or empty text	willana	willana x='munay'
PRINT	print	rikuchina	rikuchina('munay')
ASK	input	tapuna	x=tapuna('munay')
THEN		Rurana	kaman(x==5) rurana

Rules

The existing rules in our algorithm are those mentioned in Table 2, which are associated to a regular expression. This allows to execute a certain logic in the translation of the pseudocode. The regular expression associated to an assignment rule is the following:

```
def t_ASIG(t):
r':=|\+=|-=|\*=|/='
if t.value == ':=':
t.value = '='
return t
```

Table 2. Model rules.

Definition	Kichwa Value	Python Value	Example
Unary Operator	++,--	++,--	x++
Binary Operator	+,-,*./	+,-,*./	x+y
Binary Operator	<,>,==,!=,>=,<=	<,>,==,!=,>=,<=	x>0
Parentheses Open	(	(	x=(y+1)
Closed Parenthesis	)	)	x=(y+1)
Assignment	:=, +=,-=,*=./=	=, +=,-=,*=./=	x:=7
Number			5
String			'Hola'
Boolean	Verdadero, Falso	Verdadero,Falso	
Characters	,	,	,
Comment	$	#	$'Texto'

Algorithm of translation

The translation algorithm is developed in Python through the implementation of the Lex and Yacc libraries [25]: The program receives a sequence of characters and generates a sequence of strings with an assigned and identified meaning, called Tokens. They are the input values for our syntactic analyzer, built using Yacc which validates whether the generated tokens comply with a rule defined in the specification of the pseudocode model as shown in Fig 2, and if so, they execute the respective logic for translation. In this way, Python code is obtained from a pseudocode with Kichwa-based syntax.

4.4 Execution of the algorithm

Let's suppose that the algorithm receives the following input:

willana x:= 'Hello World'

The tokenization process begins, generating a string of characters as the following ones:

"willana", "x", ":", "=", "'", "Todo", "bien", "'"

The following step is performed to validate whether there is a token or a rule associated with each value.

Tabla 3. Grammatical structure of the code associated with a token .

Pseudocode	Token Name	Phyton Equivalent
willana	VAR	var or empty text

Taking as an example the word willana, it is associated with a token, then replaced by its equivalent in Python code as shown in Table 3.

The following characters are associated to a rule, therefore we proceed to execute the respective logic for their translation, as shown in table 4.

Table 4. Equivalente a una regla en código Python.

Pseudocode	Rule Name	Equivalent
x	Identifier	x
:=	Assignment	=
“Hello World”	Text	“Hello World”

Resulting in:

var x= “Hello World”

Compilation

The compilation is done by using the Hackerearth API, which provides endpoints to compile and execute code in several languages. In order to do this, it is necessary to register the web application to acquire an Api Key credential, which allows to make use of all the endpoints that Hackerearth has. Our web application sends the code generated in Python by our algorithm to the translation and compilation server and this in turn consumes the Hackerearth API and returns the response to the website for later display [26].

5 Analysis of Findings:

The tests are conducted with a total of 13 users, 76% are indigenous people who do not know programming logic and the remaining 24% are indigenous people who poses prior knowledge of programming logic. The learning experimentation is carried out in different parts. First the user familiarizes with the operation of the system and the interface. The user must complete the 4 learning scales represented in the application with 8 levels, which are the following ones: 1-3 very easy level, 4-6 easy level, 7 intermediate level, 8 difficult level.

To assess the learning level of the user in basic programming logic prior to using the Kichwa Code web application, the participants were given a survey.

A comparison was made between the knowledge level before and after the completion of the training course. The equation for calculating the normalized conceptual gain, proposed by Hake, is used:

$$g = \frac{\%V - \%Vi}{100\% - \%Vi} \tag{27}$$

Accordingly: g, is the normalized conceptual gain. Vi, is the value of the group average before completing the training course or initial group average. V, is the value of the group average after completing the training course, i.e., the final group average.

After the analysis of the results, a substantial improvement can be observed. Initially the average of the group was 1.9 and in the end the average was 5.9, the conceptual gain being 0.66 which according to Hake, it corresponds to an average gain. A comparative graph of the results obtained is shown in Fig. 4.

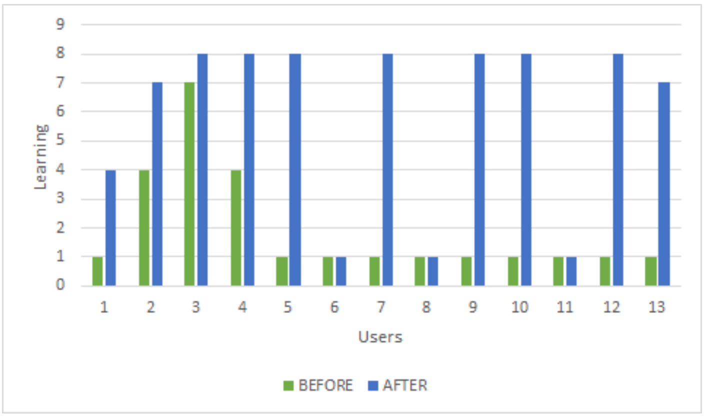


Fig 4. Comparison of the increase in learning before and after the use of the Kichwa Code application.

To evaluate the reduction in the learning time, the information stored in the user's tracking and monitoring database has been used. Additionally, to this information, we have used the data obtained in the previously mentioned survey. For this analysis, a scale measured by hours, consisting of the following levels has been used: 8h, 16h, 120h, 240h.

A comparison was made between the time it takes for the user to complete the programming course in OpenlabEc, with and without the use of the web application. It is used the equation to calculate the percentage reduction according to Salazar, Castillo and Del Castillo:

$$d = Vi - V \quad \text{y} \quad P = \frac{d}{Vi} * 100 \tag{28}$$

Accordingly: d, is the difference between the two means found above. P, is the percentage of time reduction. Vi, is the value of the average time spent completing the programming logic course without using the web application or initial average time. V, is the value of the average time taken to complete the programming logic course using the web application or final average time.

Through the analysis of the results, it is determined that a substantial reduction of time is obtained with the use of the Kichwa Code web application. Initially the average time in hours without using the application was 56 h. and with the use of the application the average time dropped to 21 h. With a difference of 35 h., the percentage of time reduction is 62.5%, which according to Salazar, Castillo and Del Castillo, corresponds to a high percentage of reduction. The Fig. 5 shows a comparative graph of the obtained results.

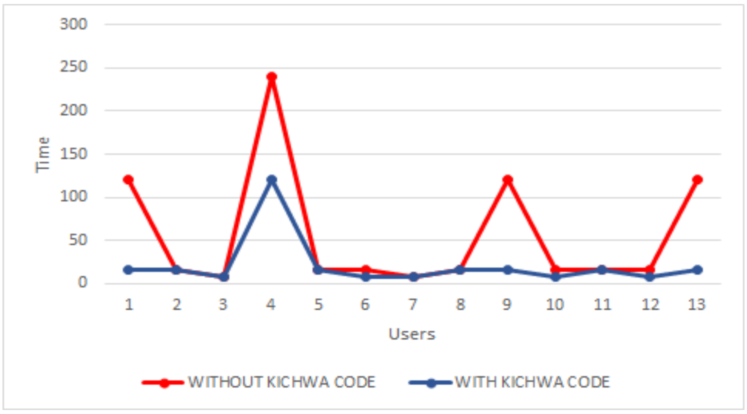


Fig 5. Comparison of the reduction of learning time without and with the use of the Kichwa Code application.

6 Conclusions

In the methodology it is detailed the theoretical framework about the different concepts that exist in the field of programming logic learning, and about the statements and basic instructions in programming.

Based on this theoretical framework, a model of the structure and syntax of the pseudocode in Kichwa was developed as shown in Fig. 3.

Based on this model, an algorithm that acts as a pseudocode interpreter in Kichwa has been developed and a web application has been implemented to facilitate the learning process of programming through the use of pseudocode.

Finally, it is necessary to mention the support of OpenLabEc by acting as an intermediary between this project and the Kichwa-speaking users. The functionality of this project was validated using the web application by the users. The results obtained were organized in comparative graphs to show the increment in learning and used to measure the impact of the application. A normalized conceptual gain indicator was used, giving s 0.66 that according to Hake, corresponds to an average gain. A 62.5% reduction in learning time was achieved, demonstrating that the use of the proposed web application fulfills the function of facilitating the learning of programming through pseudocode in Kichwa.

Although the results were positive, it is important to mention that by working with all the components of the application, the learning process is facilitated in a very broad way. It is recommended that for the use of the web application to be complemented with the assistance of a computer or programming teacher for a better understanding for the student.

7 Future Works

The following project consists of an algorithm of translation of a pseudocode whose syntax of entry can be parametrized for other native languages. The interface of the website its highly scalable. Which allows to aggregate functionalities focused on the entry of values. This represents an opportunity for the user of being able to develop even more complex algorithms.

8 References

1. Gómez, C. M.: Origen y evolución de la mediación: el nacimiento del movimiento ADR. Anuario de derecho civil, 67(3), 931-996. Estados Unidos y su expansión a Europa (2014).
2. Flores Sáez, E.: Reutilización de código fuente entre lenguajes de programación (2012).
3. Dávila, M. R.: Investigación en Progreso: Estudio Comparativo de la Incidencia de los Lenguajes de Programación en la Productividad Informática. Revista Latinoamericana de Ingenieria de Software, 4(6), 255-258 (2016).
4. Pacheco Patiño, M. A.: Elementos de programación Algoritmos, herramientas, programación estructurada. Aplicación a los lenguajes de programación Pascal, el lenguaje de programación "C" y C++ y otros lenguajes actualizados. Introducción al lenguaje de programación Matlab lenguajes de programación orientado a objetos. Aplicaciones a la realidad cotidiana (2019).
5. Cass, S.: The 2017 top programming languages, pp. 1. IEEE Spectrum, 31 (2018).
6. Howard, R.: Las lenguas quechuas en tres países andino-amazónicos: de las cifras a la acción ciudadana. Káñina, 45(1), 7-38 (2021).
7. Cárdenas, J. O.: PROBLEMAS INTERGENERACIONES EN LA CONTINUIDAD DEL QUECHUA. Revista de la Facultad de Derecho y Ciencias Políticas (Cusco), (12), 39-60 (2020).
8. Nuñez, M., Duran, Y., Mojica, Z., & Stewart, M. A.: Descubriendo los Recursos Culturales de Estudiantes Indígenas Latinoamericanos a través de la Literatura. Journal of Latinos and Education, 1-8 (2021).
9. Howard, R.: Las lenguas quechuas en tres países andino-amazónicos: de las cifras a la acción ciudadana, pp. 9-11. Káñina, 45(1), 7-38 (2021).
10. Roig-Vila, R. & Moreno-Isac, V.: El pensamiento computacional en Educación. Análisis bibliométrico y temático. Revista De Educación a Distancia (RED), 20(63) (2020).
11. Aguilera Nicolalde, D. E.: Cierre de instituciones de educación intercultural bilingüe y sus repercusiones en los pueblos indígenas del Ecuador, pp. 120-123. (Bachelor's thesis, Quito: UCE) (2021).
12. Cass, S.: The 2017 top programming languages, pp. 2. IEEE Spectrum, 31 (2018).
13. Garcés, F.: La revitalización de las lenguas Indígenas del Ecuador: una tarea de todos (2020).
14. Paronyan, H., & Díaz, M. C.: CONSIDERACIONES EN TORNO A LA IMPLEMENTACIÓN DE LOS DERECHOS LINGÜÍSTICOS DE LOS PUEBLOS INDÍGENAS DE ECUADOR, pp. 84-85. ISSN 2528-7842 (2019).

15. Howard, R.: Las lenguas quechuas en tres países andino-amazónicos: de las cifras a la acción ciudadana, pp. 8. Káñina, 45(1), 7-38 (2021).
16. Grefa Grefa, L. D., & Ojeda Contreras, J. A.: Monitoreo espacio-temporal y socio ambiental de acceso para jóvenes indígenas de Pastaza a las unidades educativas secundarias y educación superior-UEA (Bachelor's thesis, Universidad Estatal Amazónica) (2019).
17. Paronyan, H., & Díaz, M. C.: CONSIDERACIONES EN TORNO A LA IMPLEMENTACIÓN DE LOS DERECHOS LINGÜÍSTICOS DE LOS PUEBLOS INDÍGENAS DE ECUADOR, pp. 86-88. ISSN 2528-7842 (2019).
18. Aguilera Nicolalde, D. E.: Cierre de instituciones de educación intercultural bilingüe y sus repercusiones en los pueblos indígenas del Ecuador, pp. (Bachelor's thesis, Quito: UCE) (2021).
19. Grefa Grefa, L. D., & Ojeda Contreras, J. A.: Monitoreo espacio-temporal y socio ambiental de acceso para jóvenes indígenas de Pastaza a las unidades educativas secundarias y educación superior-UEA ,pp. 5. (Bachelor's thesis, Universidad Estatal Amazónica) (2019).
20. Vázquez-Maguirre, M.: El desarrollo sostenible a través de empresas sociales en comunidades indígenas de América Latina. Estudios sociales. Revista de alimentación contemporánea y desarrollo regional, 29(53) (2019).
21. Buitrón Cachipuendo, B., Díaz Gispert, L. I., & Cahuasquí Anrango, J. A.: Diseño de un aplicativo web para la enseñanza del idioma kichwa. RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 11(21) (2020).
22. Levine, J. R., Mason, J., Levine, J. R., Mason, T., Brown, D., Levine, J. R., & Levine, P.: Lex & yacc. O'Reilly Media, Inc. (1992).
23. Niemann T.: A Compact Guide to Lex and Yacc.Portland, Oregon (2018).
24. Chango M., Potosí C.: Diccionario Kichwa-Castellano, Ministerio de Educación, Ecuador (2009)
25. D. Beazley. PLY (Python Lex-Yacc). http://www.dabeaz.com/ply/. last accessed 2021/01/08.
26. HackerEarth Docs. https://www.hackerearth.com/docs/wiki/developers/v4/. last accessed 2021/01/16.
27. Hake, R. R.: Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. American journal of Physics, 66(1), 64-74 (1998).
28. Salazar Pinto, C., Castillo Galarza, S. D., & Del Castillo Galarza, S.: Fundamentos básicos de estadística (2017).
29. Documentation Google Cloud. https://cloud.google.com/docs . last accessed 2021/02/07.