PHP CALANGO FRAMEWORK AND DIGITAL TRANSFORMATION: AN INTRODUCTORY GUIDE FOR BEGINNERS

Sergio Silva Ribeiro
Crandall University – Moncton – NB – Canada
sergio.ribeiro@crandallu.ca

Abstract. *Calango is a lightweight PHP framework released in 2012 as an open-source project under the LGPL-3.0 license. It suits program language teaching, research applications, small or mid-size business websites, APIs, microservices, and prototypes. Calango works with the MVC structure, and its simplicity allows the developer greater freedom in their applications. This paper presents a tutorial introduction to help potential framework users get started through a step-by-step guideline.*

Keywords: Calango Framework, PHP, MVC, Web Development, Tutorial.

1. INTRODUCTION

PHP is one of the most popular server-side programming languages for web development. It has an important role in the era of digital transformation. The industry and open-source community use it to develop web applications and frameworks [1]. Examples of open-source projects implemented in PHP are WordPress, Drupal, phpBB, MantisBT, and phpMyAdmin [2]. In industry, PHP is applied to smart cities, e-commerce, CMS development, online corporate web portals, web services, education and agriculture [3] [4] [5] [6] [7].

The main goal behind adopting a framework when creating an application is to reduce the development time and effort when building an extensive, complex application [8]. Extensive frameworks like Laravel, Symfony, and Zend are excellent when working on large projects. They will require the developers to understand the framework, its conventions, and its many abstraction layers. However, working with simple, lightweight frameworks like Calango offers numerous advantages. Usually, they offer an easy-to-use MVC architecture that will require no or minimal setup to start. The curve learning is short, has better performance, less overhead, is straightforward, transparent, and easy to read, customize, and hack [9] [10].

A lightweight framework is ideal for program language teaching, research applications, small or mid-size business websites, APIs, microservices, and prototypes. This paper describes and provides a tutorial introduction to the PHP Calango Framework. It starts with a general framework overview, covers the get-started, and offers a step-by-step guideline for its initial usage.

2. OVERVIEW

The PHP Calango Framework was created by Sergio Ribeiro and released in 2012 as an open-source project under the LGPL-3.0 license [11]. Professor Ribeiro led a research group composed of students from the analysis and development of systems underground program at Guairaca College, now UniGuairaca University Center. The primary motivation for creating the framework was to allow students to explore the MVC concept when learning web development using PHP. Originating from the African word Kalanga, meaning "small lizard," the Brazilian Portuguese variation Calango is primarily used to refer to this reptile in the north region of Brazil [12]. Calango is small, agile and resistant, and the characteristics of this reptile inspired the framework and were used to name it. Calango was created to be simple and easy to implement. It is a framework that works with the MVC structure. Its simplicity allows the developer to have greater freedom in their applications.

The use of the framework will require basic PHP knowledge, an understanding of PHP object-oriented programming by working (Classes, Methods, Properties), ability to work with databases like MySQL, knowledge of the MVC pattern, and familiarity with HTML/CSS.

MVC (Model-View-Controller) is a pattern proposed in 1978 to make the program reusable and simplify the complexity of a program structure to make it more intuitive. The program is organized into distinctive layers in the MVC structure to separate the business logic from the interface (Figure 1). The Model layer encapsulates the data associated with the business logic. The View processes the data to present the model in the UI (User Interface). The Controller manages the interactions between the layers, controlling the flow of the application [13].

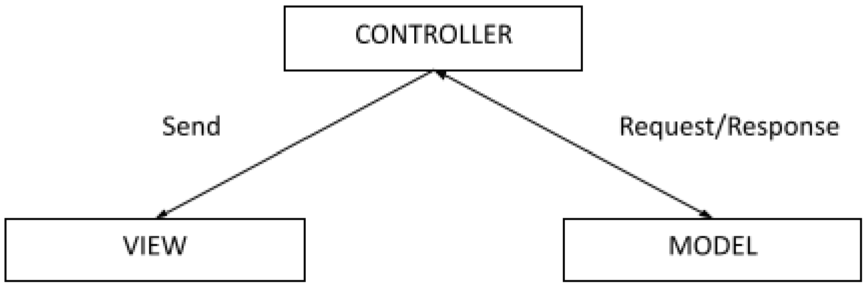


Figure 1. MVC data flow

When Calango is installed, the root directory holds the folders named model, view, and controller, representing the MVC layers. All related PHP classes and their associated files are stored in these folders.

The framework has been employed efficiently in various applications like professional websites, programming language education, academic projects, inventory management systems, and e-commerce solutions [14] [15] [16] [17] [18] [19].

3. INSTALLATION

Before proceeding with Calango installation, ensure that PHP is installed, have a web server environment like Apache with XAMPP or WAMP, and a database management system like MySQL.

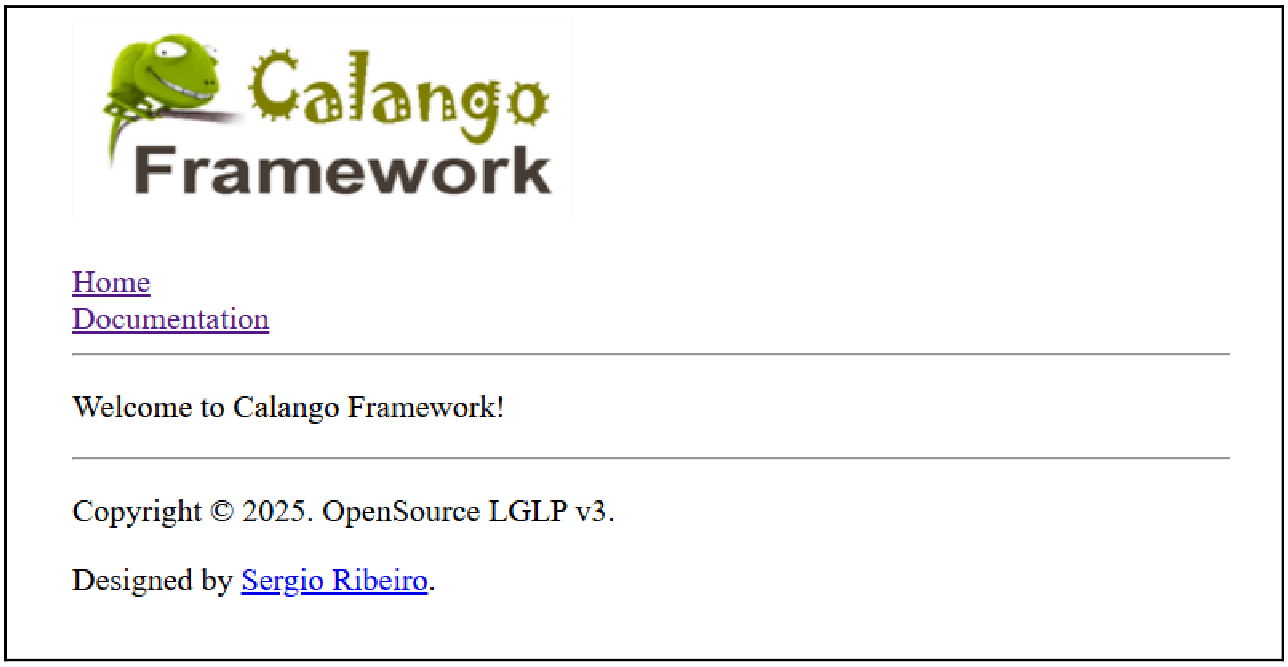
Calango framework can be downloaded directly from https://github.com/profssribeiro/calango or linked to the GitHub repository. The framework must be installed in the web server root directory. The root directory name will vary depending on the web server used. If you are using XAMPP, the root directory will be `htdocs`, and for those using USBWebServer, it will be the `root` folder.

After installing Calango into your web server root directory, you will have the framework main directory structure as shown in Figure 2.

```
/root/
  └──calango/
       ├──controller/
       ├──core/
       ├──model/
       ├──plugin/
       ├──public/
       └──view/
```

Figure 2. Calango Directory

If you are running the web server on your local machine, the framework can be accessed by typing it into your browser: `http://localhost/calango`. If installed successfully, you will see the Framework main page in your browser (Figure 3).



Calango Framework

Home
Documentation

Welcome to Calango Framework!

Copyright © 2025. OpenSource LGLP v3.

Designed by Sergio Ribeiro.

Figure 3. Calango main page

4. FRONT-END

The front-end refers to the application interface composed of text, pictures, tables, buttons, and other visual resources. It is the part of the web system where the users can interact. The interface will be rendered into the browser using HTML in a web solution [20].

Calando is designed to be simple and easy to implement by following the MVC architecture. The View layer manages the presentation, rendering the user interface. The front-end layer of the framework is organized into the folders `/public/css/`, `/public/js/`, `/public/img/`, `/view/`, and `/view/html/` (Figure 4). The public subfolders are intuitive, where the `/css/` directory holds CSS files, the `/js/` directory holds JavaScript files, and the `/img/` directory holds images or picture files. The `/view/` directory holds the PHP classes responsible for rendering the user interface. These classes will load the HTML files stored in the `/view/html/` directory to compose the front-end interface.

```
/config/
  └──config.php
/controller/
  └──Main.class.php
/core/
/model/
/plugin/
/public/
  ├──css/
  ├──img
  └──js/
/view/
  ├──Footer.class.php
  ├──Header.class.php
  ├──Menu.class.php
  └──html/
      ├──footer.hml
      ├──header.html
      ├──home.html
      ├──menu.hml
      └──template.html
.htaccess
Index.php
```

Figure 4. Calango Directory Structure

The **`/view/html/`** directory holds the main HTML file used to generate the HTML output (user interface), the **`template.html`** file (Figure 5). This master template is a base layout for rendering views consistently across the web application.

```
<!DOCTYPE html>
<html>
    <head>
          <title>Calango Framework</title>
    </head>
    <body>
          <header>#HEADER#</header>
          <menu style="list-style:none; display:inline;">#MENU#</menu>
          <hr>
          <main>#CONTENT#</main>
          <hr>
          <footer>#FOOTER#</footer>
    </body>
</html>
```

Figure 5. Template.html file content

The master template contains the typical structure of a web page's general HTML layout composed of the header, menu, content, and footer. Note the dynamic content's placeholders **`#HEADER#`**, **`#MENU#`**, **`#CONTENT#`**, and **`#FOOTER#`**. The framework dynamically detects the HTML files and injects the output into placeholders in **`template.html`**. For the **`#HEADER#`** placeholder, the file **`/view/html/header.hml`** is injected. The same applies to the placeholders **`#MENU#`** and **`#FOOTER#`**. For the placeholder **`#CONTENT#`**, the file **`/view/html/home.html`** is loaded and injected. So, the front-end is modified by changing the HTML files.

5. BACK-END

The back-end refers to all the processes running on the server side in response to any request from the client (front-end) [21]. Calango will run the class controller Main (**`Main.class.php`**) stored in the directory /controller/ by default when typing the URL **`http://localhost/calango/`**. The parse URL syntax is **`Module/action/key`**. In this case, the URL **`http://localhost/calango/`** is equivalent to typing **`http://localhost/calango/Main/show/home`** in the browser. The parameters passed are "Main" (Module), "show" (action), and "home" (key). The module is a controller class to be stored in the **`/controller/`** directory as *Module.class.php*, and inside the file, the PHP class will have the same name (Figure 6).

```
<?php


class Main{

    public function show(){

           return Html::load(App::$key.".html");

    }

}
```

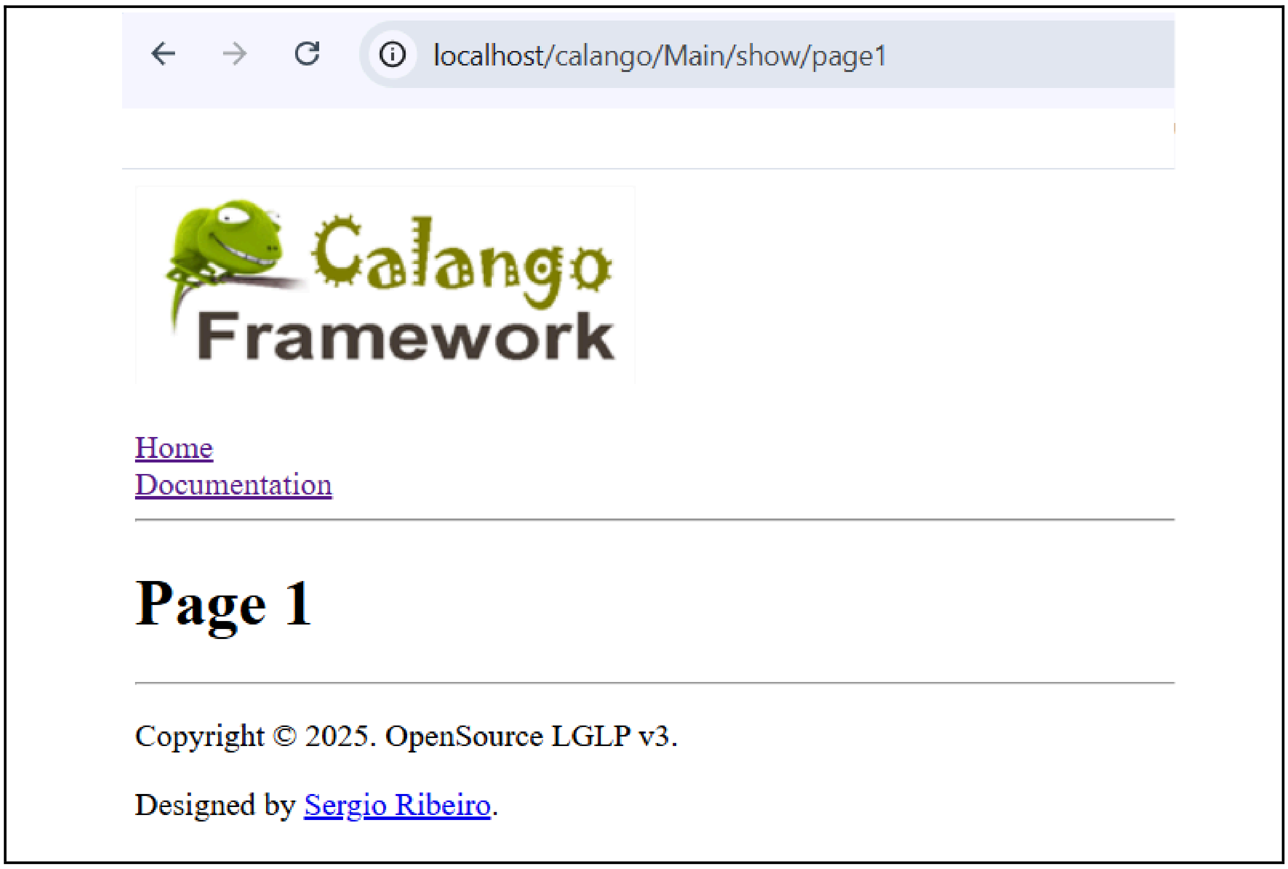
Figure 6. Main.class.php file content

In the code (Figure 6), the class Main loads a specific static HTML file given by the key (**`App::$key`**) via **`Html::load()`**. When not informed, **`App::$key`** receive the value "home" and the concatenation **`App::$key.".html"`** □ **`home.html`** is loaded.

The class Main can load as many HTML files as needed. For example, creating the HTML files **`/view/html/page1.html`** (Figure 7) and **`/view/html/page2.html`** could be displayed in the browser by typing the URLs **`Main/show/page1`** (Figure 8) and **`Main/show/page2`**.

```
<p><h1>Page 1</h1></p>
```

Figure 7. page1.html file content



Home
Documentation

Page 1

Copyright © 2025. OpenSource LGLP v3.

Designed by Sergio Ribeiro.

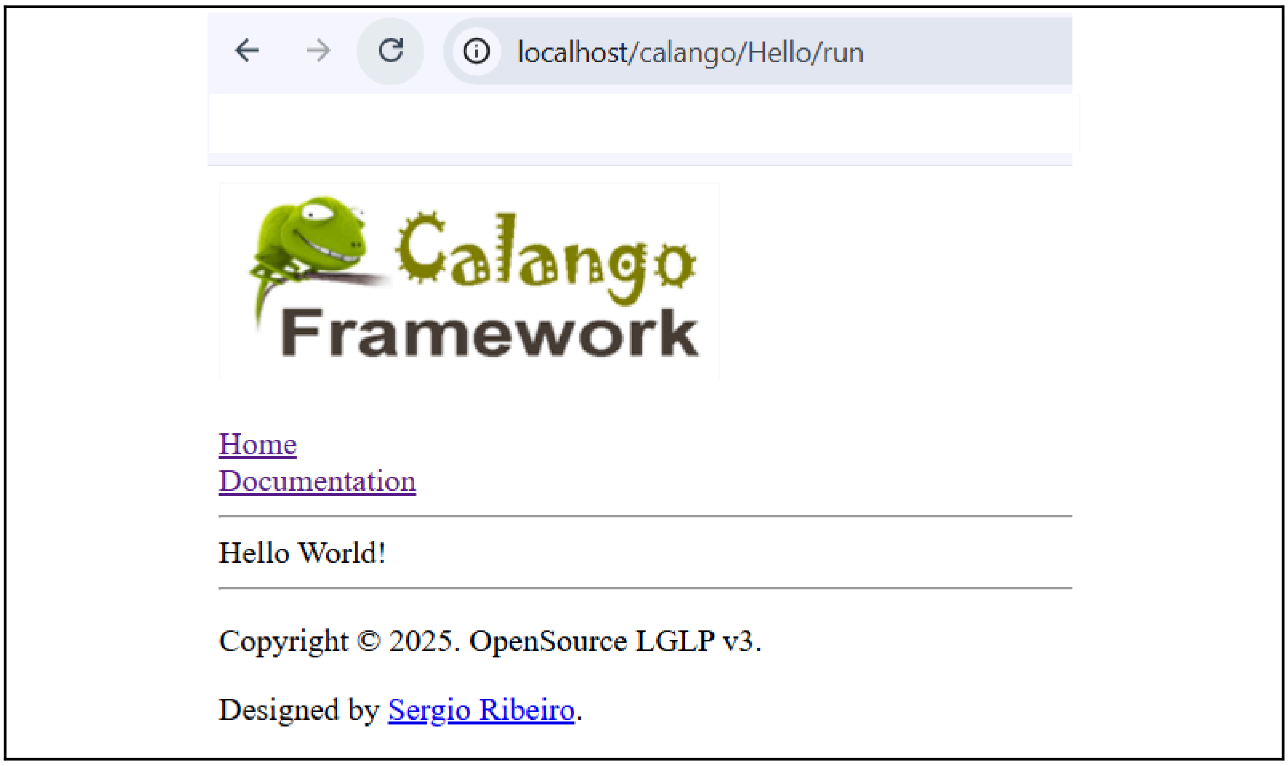
Figure 8. page1.html loaded

You can also create your own class controller like the class ***Hello*** in Figure 9. The class must be saved as `/controller/Hello.class.php`.

```
<?php
class Hello{
    public function run(){
            return "Hello World!";
    }
}
```

Figure 9. Hello.class.php file content

To run the new controller, type the URL `http://localhost/calango/Hello/run` in the browser. Figure 10 shows the output result.


localhost/calango/Hello/run

Home
Documentation

Hello World!

Copyright © 2025. OpenSource LGLP v3.

Designed by Sergio Ribeiro.

Figure 10. Hello class output

The controller layer of the Framework processes the request, interacting with models to retrieve or manipulate data as needed. The examples demonstrate its simplicity and flexibility, allowing the developer to control the application.

6. ENVIRONMENT SETUP

In terms of environment setup, it is important to understand the distinction between development and production environments. There are some setup, security, and configuration considerations.

The development environment usually includes a local setup that allows the developers to write, test, and debug the application before deploying it to a hosted domain server. In this environment, errors and debugging are enabled for testing, and security measures are relaxed to facilitate the development process. A local web server like XAMPP, WAMPP, or USBWebServer will run the project on localhost. The server will include a PHP version and a database management system like MySQL, MariaDB, or SQLite. Files as stored locally into the document root, typically a subdirectory inside a local web server directory like `c:\xampp\htdocs\myproject`, `/var/www/html/myproject`, or `c:\usbwebserver\root\myproject`. There is no domain, and the application is accessed via `http://localhost/myproject` or `http://127.0.0.1/myproject`.

```
//Path and internal folders
define('URL','http://'.$_SERVER['HTTP_HOST'].'/myproject/');
…
define('PATH_IMG',$_SERVER['DOCUMENT_ROOT'].'/myproject/public/img/');
```

Figure 11. Changing the file /config/config.php

Find the file `/config/config.php` and change the word *calango* with *myprojec* in the lines used to define URL and PATH_IMG (Figure 11).

```
…
RewriteBase /myproject
…
```

Figure 12. Changing the file .htaccess

Also, find the file in `.htaccess` and change the word *calango* with *myprojec* in the lines used to set the RewriteBase.

When moving your application from a local development environment to a production one, the application is placed on a hosted web server. Usually, your application will be accessed from a domain name like *https://mywebsite.com*. In this case make sure to setup the `config.php` and `.htaccess` files accordingly. In the config.php file you will want to set the `ini_set('display_errors',true)` command to `false`. Also, set the database parameters to connect the database hosted in your web server domain.

7. CONCLUSION

Calango is a lightweight PHP framework appropriate for program language teaching, research applications, small or mid-size business websites, APIs, microservices, and prototypes. Calango has been employed efficiently in various applications like professional websites, programming language education, academic projects, inventory management systems, and e-commerce solutions.

This paper presented a tutorial introduction to help potential framework users get started by showing them how it starts, a general framework overview, and a step-by-step guideline for its initial usage. Calango offers many more components not covered in this paper, such as additional features for front-end and back-end, integration with additional plugins, and database usage. Calango is an open-source and growing framework. In future works, we plan to explore its functionality in potential applications.

REFERENCES

[1] M. Hills, P. Klint (2014) PHP air: Analyzing PHP systems with rascal. In 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE) (pp. 454-457).

[2] A. Siame, D. Kunda (2017). Evolution of PHP applications: A systematic literature review. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 5(1), 28-39.

[3] X. Liu, A. Heller, P. S. Nielsen (2017). CITIESData: a smart city data management framework. Knowledge and Information Systems, 53, 699-722.

[4] S. Sotnik, V. Manakov, V. Lyashenko (2023). Overview: PHP and MySQL features for creating modern web projects.

[5] D. Sudharsan, J. Adinarayana, S. Ninomiya, M. Hirafuji, T. Kiura (2012). Dynamic real time distributed sensor network based database management system using XML, JAVA and PHP technologies. International journal of database management systems, 4(1), 9.

[6] A. S. Pereira, S. D. Piovesan (2011). Applications using Virtual Reality. Iberoamerican Journal of Applied Computing, 1(1).

[7] F. G. Montoya, J. Gómez, A. Cama, A. Zapata-Sierra, F. Martínez, J. L. De La Cruz, F. Manzano-Agugliaro (2013). A monitoring system for intensive agriculture based on mesh networks and the android system. Computers and Electronics in Agriculture, 99, 14-20.

[8] N. Soundarajan (1997). Understanding frameworks. Object-Oriented Application Frameworks.

[9] P. Hrkút, M. Meško, M. Ďuračík (2024). A Custom Framework for Innovative Approach of Teaching Web Development Courses. In 2024 36th Conference of Open Innovations Association (FRUCT) (pp. 256-261). IEEE.

[10] E. Männistö (2023). Building a simple PHP framework.

[11] S. S. Ribeiro (2012). PHP Calango Framework. GitHub. Retrieved March 24, 2025, from https://github.com/profssribeiro/calango

[12] J. T. Schneider (1985). Sub-saharan cultural extension in Brazil the relevance of lexical data. Studies in African linguistics, 16(2), 223-234.

[13] S. Chen (2018). Understanding of the management information system based on MVC pattern. In AIP Conference Proceedings (Vol. 1955, No. 1). AIP Publishing.

[14] S. S. Ribeiro (2019). Dr. Sergio Ribeiro, Associate Professor, Crandall University. Academic Website. https://profssr.com/

[15] G. F. Galvão, S. S. Ribeiro (2013). Utilização da Orientação a Objetos para desenvolvimento de um sistema web para controle de estoque. Revista Científica Semana Acadêmica, vol. 1(45).

[16] R. M. Teixeira, R. Orlovski, S. S. Ribeiro (2013). Desenvolvimento de Sistema Gerenciador de Aprendizagem para Escola de Informática. Revista Científica Semana Acadêmica, vol. 1(48).

[17] J. A. Aires, S. S. Ribeiro, R. Orlovski (2013). Desenvolvimento de Sistema de Gerenciamento e Controle para Academias. Revista Científica Semana Acadêmica, v 1(48).

[18] M. A. Somer, S. S. Ribeiro, R. Orlovski (2014). Desenvolvimento de Sistema Web para Sorveteria. Revista Científica Semana Acadêmica, v 1(50).

[19] ERP+. (2023). Pagina Institutional. ERP+ Website. https://erpmais.com.br/

[20] I. H. Madurapperuma, M. S. Shafana, M. J. A. Sabani (2022). State-of-art frameworks for front-end and back-end web development. In: 2nd International Conference on Science and Technology, Sri Lanka.

[21] A. G. Rahaman, V. Gayatri, C. S. Kiran, K. S. Pavan, B. Bhumika, V. Sateesh (2023). Development of Web Applications by Integrating Frontend and Backend Tools. International Journal of Innovative Research in Computer and Communication Engineering, 5002-5007.