Matrix: Jurnal Manajemen Teknologi dan Informatika
Volume 14 Issue 2 Year 2024 Pages 66-73

DOI: http://dx.doi.org/10.31940/matrix.v14i2.66-73
URL: http://ojs2.pnb.ac.id/index.php/matrix

Laravel framework and native PHP: Comparison in the creation of rest API

Yuri Ariyanto [1]***, Mochamad Farhan Fitrahtur Rachmad*** [2]*** *, Dwi Puspitasari*** [3]

[1,2,3]*Department Information Technology, Politeknik Negeri Malang, Indonesia*

*Corresponding Author: *farhanfitrahtur@gmail.com*

Abstract: As technology evolves, the need for data communication grows. One technology facilitating this communication between applications and databases is the REST API, widely used by companies to provide access to their data and services. REST APIs can be developed using PHP, known as PHP Native. However, PHP Native faces performance issues and a monotonous, repetitive code structure, making maintenance and scalability challenging. Consequently, developers have created PHP frameworks like Laravel. This research compares the Laravel framework with PHP Native in REST API development, evaluating aspects such as processing speed, code efficiency, URL routing structure, and architectural models. The analysis reveals four key points: REST API processing speed, code efficiency, URL routing structure, and project architectural models. In terms of processing speed, PHP Native is slightly superior to Laravel, with speeds of 18.3 ms compared to 344.52 ms, due to additional processes in Laravel like routes, controllers, and models. For code efficiency, Laravel excels due to its ORM feature, unlike PHP Native, which requires manual SQL query writing. In URL routing, Laravel is superior with a structured routing feature, whereas PHP Native requires process file inclusion in route calls. Lastly, Laravel's architectural model, which implements MVC, is superior to PHP Native, which lacks a standardized architectural model, leading developers to create their own.

Keywords: Code Efficiency, Laravel Framework, MVC Architecture, PHP Native, Rest API

History Article: Submitted 8 May 2024| Revised 3 June 2024 | Accepted 30 July 2024

How to Cite: Y. Ariyanti, M. F. F. Rachmad, and D. Puspitasari, "Laravel framework and native PHP: Comparison in the creation of rest API," *Matrix: Jurnal Manajemen Teknologi dan Informatika*, pp. 66-73, 2024, doi.org/10.31940/matrix.v14i2. 66-73.

Introduction

As technology develops, the need for data communication is also necessary. The REST API is one technology that allows data to be shared between databases and apps. Resources and activities that can be performed on them make up a REST API. A web browser or client-side JavaScript code running in a web browser can be used to call the REST API's operations [1]. REST APIs are also used by many companies to allow others to access their data and services [2]. In creating a REST API, you can use the PHP programming language or it can be called PHP Native. PHP Native is a programming language or instructions that are created without the intervention of other developers in the process. However, native PHP has performance problems so it is necessary to create a better and faster REST API [3], then the code structure is still monotonous and repetitive which makes application maintenance and scalability sometimes difficult due to the unorganized structure [4], because of this some developers have started developing PHP frameworks called frameworks. An integrated collection of software elements, including objects, classes, and components, that work together to create a reusable architecture for a group of connected apps is called a framework. More specifically, frameworks improve program extensibility, flexibility, and portability by separating software application-dependent components from software application- and platform-independent parts [5].

Currently, there are many PHP programming language frameworks available, such as CodeIgniter Yii and Laravel. PHP is a programming language based on open source that is free to download. Until now, the latest version of PHP available is version 7.0.8, which can be downloaded from the official PHP website [6]. The majority of modern web servers have a PHP processor, and most operating systems offer a standalone interpreter [7]. Many web programmers such as HTML/CSS choose PHP as the programming language used [8].

The research uses the Laravel framework for analysis. Laravel serves as a PHP-based web platform for developing high-end web applications with meaningful and elegant syntax. Laravel was created by Taylor Otwell in July 2011 and published more than five years after Codeigniter. It includes a robust toolset and an application architecture. It also discusses numerous aspects of technologies including the ASP.NET, CodeIgniter, Ruby on Rails, and several others [9]. The Laravel framework is an open-source software with a copyright and is distributed with licensing terms designed to ensure the source code will always be available [10]. Using Laravel can create information systems more quickly, besides that, the features in Laravel can be easy to use [11].

As previously explained regarding the advantages and disadvantages of each PHP programming language, both the Laravel framework and PHP Native, the research carried out compared the Laravel framework with native PHP. This research aims to test the two PHP programming languages to obtain the advantages and disadvantages of each in REST API development such as REST API processing speed, program code efficiency, URL structure, and project architectural model used in the project.

Methodology

This research methodology explains the flow of comparative analysis stages of the Laravel framework with native PHP in creating a REST API. The research method is shown in Figure 1.

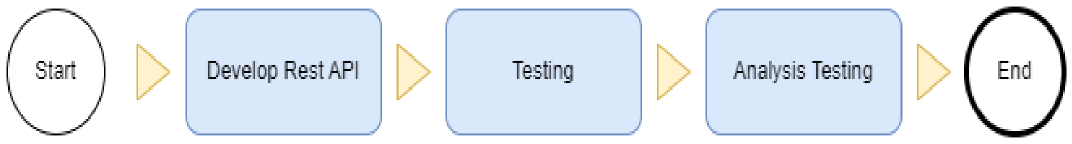


Figure 1. Research Methodology

Stage 1: Develop REST API
During this stage, the REST API is built with the Laravel framework and native PHP. The built REST API includes creating, reading, updating, and deleting operations on employee data. The detailed processes are as follows:
Create: Develop the process to add data to the database.
Read: Develop the process to retrieve data from the database.
Update: The process of updating data in the database is being developed.
Delete: Develop the process to delete data from the database.

Stage 2: Testing
In this stage, testing is performed on the previously developed REST API to ensure there are no bugs or errors in any of the processes. Testing is the procedure of confirming and validating that a software or app program complies with the business and technical requirements that guide its design and development, functions as expected, and identifies critical errors or deficiencies in the application that must be corrected based on severity.

Stage 3: Analysis Testing
In this stage, an analysis is conducted on the REST API created using both the Laravel framework and native PHP. The analysis includes:
Processing Speed of the REST API: Comparing the speed at which each platform executes API requests.
Code Efficiency: Evaluating the efficiency and cleanliness of the code written on each platform.
URL Routing Structure: Comparing the URL routing structures used by Laravel and native PHP.
Project Architectural Model: Assessing the architectural models used on both platforms' projects.

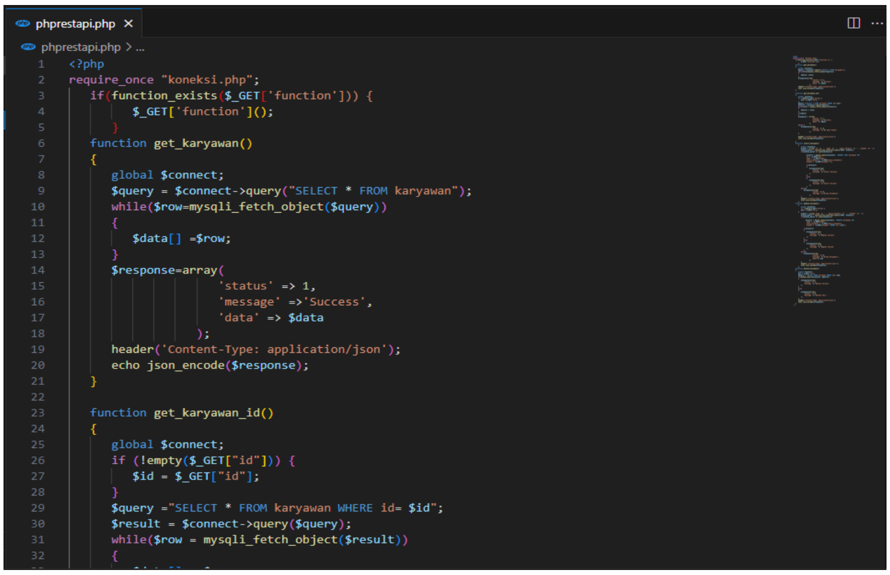
Results and Discussions

Results

The results obtained from several stages of the research method, namely developing REST API, testing, and analysis are comparisons of the Laravel and native PHP frameworks such as REST API processing speed, program code efficiency, URL routing structure, and project architectural model used in the project.

Develop REST API

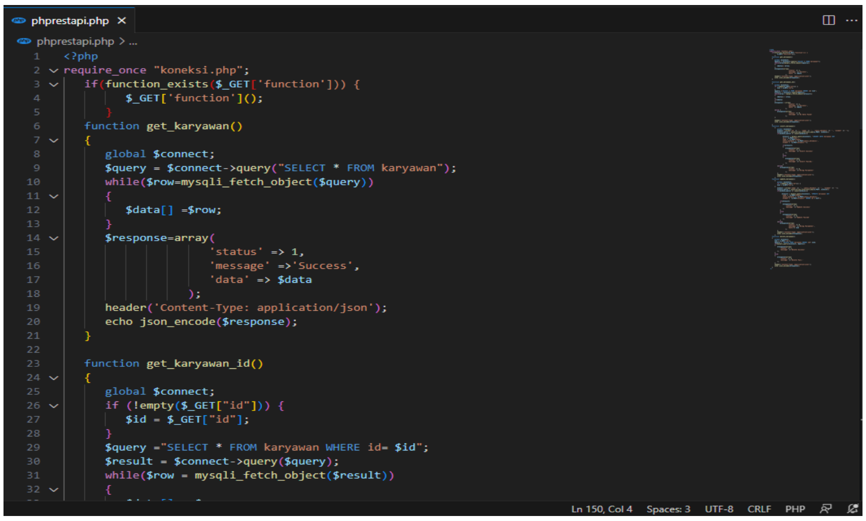
In developing the REST API, the REST API allows you to create, read, update, and delete processes on employee data. For development using the Laravel framework and native PHP, see Figure 2 and Figure 3.



```php
<?php
require_once "koneksi.php";
    if(function_exists($_GET['function'])) {
            $_GET['function']();
        }
    function get_karyawan()
    {
        global $connect;
        $query = $connect->query("SELECT * FROM karyawan");
        while($row=mysqli_fetch_object($query))
        {
            $data[] =$row;
        }
        $response=array(
                            'status' => 1,
                            'message' =>'Success',
                            'data' => $data
                        );
        header('Content-Type: application/json');
        echo json_encode($response);
    }

    function get_karyawan_id()
    {
        global $connect;
        if (!empty($_GET["id"])) {
            $id = $_GET["id"];
        }
        $query ="SELECT * FROM karyawan WHERE id= $id";
        $result = $connect->query($query);
        while($row = mysqli_fetch_object($result))
        {
```

Figure 2. Creating a REST API using PHP Native

Figure 2 shows the steps involved in using PHP Native to create a REST API. This method uses creating, reading, updating, and deleting actions for employee data manually by developing PHP scripts. Setting up a database connection, creating queries, and processing query results are just a few of the manual SQL code authoring chores involved in these CRUD procedures. A slower development process and a higher chance of errors can result from the repeated code structure and the requirement for manual handling at every stage.



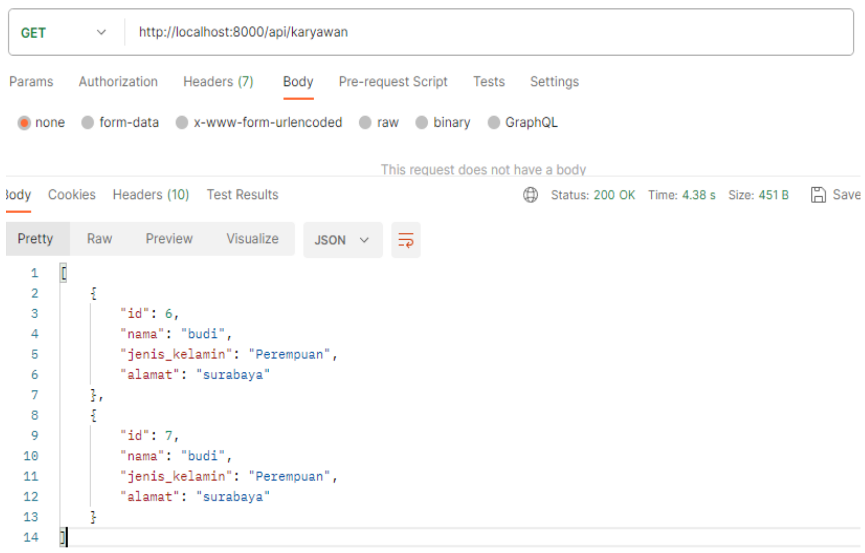
```php
<?php
require_once "koneksi.php";
    if(function_exists($_GET['function'])) {
            $_GET['function']();
        }
    function get_karyawan()
    {
        global $connect;
        $query = $connect->query("SELECT * FROM karyawan");
        while($row=mysqli_fetch_object($query))
        {
            $data[] =$row;
        }
        $response=array(
                            'status' => 1,
                            'message' =>'Success',
                            'data' => $data
                        );
        header('Content-Type: application/json');
        echo json_encode($response);
    }

    function get_karyawan_id()
    {
        global $connect;
        if (!empty($_GET["id"])) {
            $id = $_GET["id"];
        }
        $query ="SELECT * FROM karyawan WHERE id= $id";
        $result = $connect->query($query);
        while($row = mysqli_fetch_object($result))
        {
```

Figure 3. Creating an API that uses REST with the Laravel development platform

Figure 3 shows how to develop a REST API using the framework built on Laravel. This method implements CRUD for employee data through Laravel's built-in features that support REST API operations well. Laravel uses the Eloquent ORM to manage interactions with the database, and manual writing of SQL code is reduced. REST API development becomes more modular, maintainable, and scalable thanks to Laravel's features, such as organized routing, middleware, and MVC (Model View Controller).

Testing

At this stage, testing is carried out on the REST API which has been developed previously, there are no bugs or errors in each process. Testing was carried out with Postman software, which is software described on their website as an API platform that can be used to build APIs and can be used for testing [13]. For REST API testing, refer to Figure 4 and Figure 5.



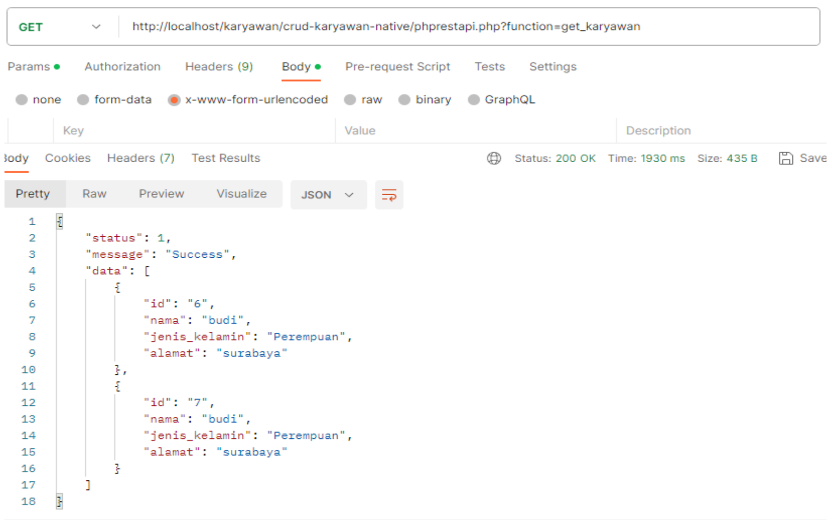
GET http://localhost:8000/api/karyawan

Status: 200 OK Time: 4.38 s Size: 451 B

```
[
    {
        "id": 6,
        "nama": "budi",
        "jenis_kelamin": "Perempuan",
        "alamat": "surabaya"
    },
    {
        "id": 7,
        "nama": "budi",
        "jenis_kelamin": "Perempuan",
        "alamat": "surabaya"
    }
]
```

Figure 4. Testing the REST API in the Laravel framework using postman

Figure 4 shows the process of testing a REST API created with the Laravel framework and Postman software. Postman sends an HTTP request to a created REST API endpoint and verifies that the request was answered correctly. Testing is performed on all CRUD (create, read, modify, and delete) operations on employee data to ensure that each API function functions correctly and that there are no bugs or errors. With Laravel, REST API testing becomes easier because features such as routing and controllers are well organized.



GET http://localhost/karyawan/crud-karyawan-native/phprestapi.php?function=get_karyawan

Status: 200 OK Time: 1930 ms Size: 435 B

```
{
    "status": 1,
    "message": "Success",
    "data": [
        {
            "id": "6",
            "nama": "budi",
            "jenis_kelamin": "Perempuan",
            "alamat": "surabaya"
        },
        {
            "id": "7",
            "nama": "budi",
            "jenis_kelamin": "Perempuan",
            "alamat": "surabaya"
        }
    ]
}
```

Figure 5. Testing REST API on PHP Native using postman

Figure 5 shows the REST API testing process developed using PHP Native using Postman software. As in testing with Laravel, Postman sends HTTP requests to the REST API endpoint and verifies the response. Testing includes all CRUD operations on employee data to ensure that the API functions work correctly and that there are no bugs or errors. In the Native PHP approach, every request and response is handled by a manually written PHP script, so testing helps ensure that the manual implementation works as expected without errors.

Analysis Testing

At this stage, an analysis of the REST API which has been created using the Laravel framework and native PHP is carried out, shown in Table 1.

Table 1. Analysis REST API

No	Analysis	PHP Native	Laravel Framework
1	Rest API Speed	18.3 ms	344.52 ms
2	Code efficiency in creating REST APIs	7 processes	1 process
3	URL structure in REST API	3 paths	2 paths

Discussions

Process Speed Analysis Results for REST API

The study was carried out on the rapidity of the REST API activity, totaling five endpoints for each technology used, namely using the Laravel framework and native PHP, which were produced in Table 2 and Table 3.

Table 2. REST API processing speed using the Laravel framework

Endpoint API	Speed
/api/karyawan (GET)	312 ms
/api/karyawan/create (POST)	499 ms
/api/karyawan/update/{id} (POST)	403 ms
/api/karyawan /{id} (GET)	324 ms
/api/karyawan/delete/{id} (DELETE)	546 ms

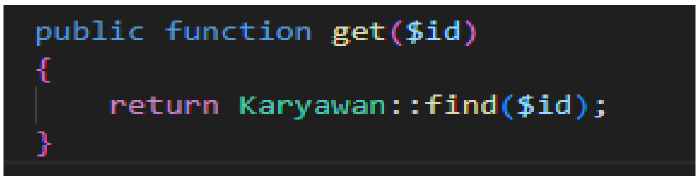
Table 3. REST API processing speed using PHP Native

Endpoint API	Speed
/karyawan/crud-karyawan-native/phprestapi.php?function =get_karyawan (GET)	680 ms
/karyawan/crud-karyawan-native/phprestapi.php?function =insert_karyawan (POST)	143 ms
/karyawan/crud-karyawan-native/phprestapi.php?function =update_karyawan&id={id} (POST)	126 ms
/karyawan/crud-karyawan-native/phprestapi.php?function =get_karyawan_id&id={id} (GET)	12 ms
/karyawan/crud-karyawan-native/phprestapi.php?function =delete_karyawan&id={id} (DELETE)	66 ms

Based on the analysis carried out from Table 2 and Table 3, the results obtained are that Native PHP is superior in speed of processing the REST API than the Laravel framework because the files for processing the REST API required in native PHP are much faster than the Laravel framework. After all, the Laravel framework has several processes that must be passed, such as routes, controllers and models.

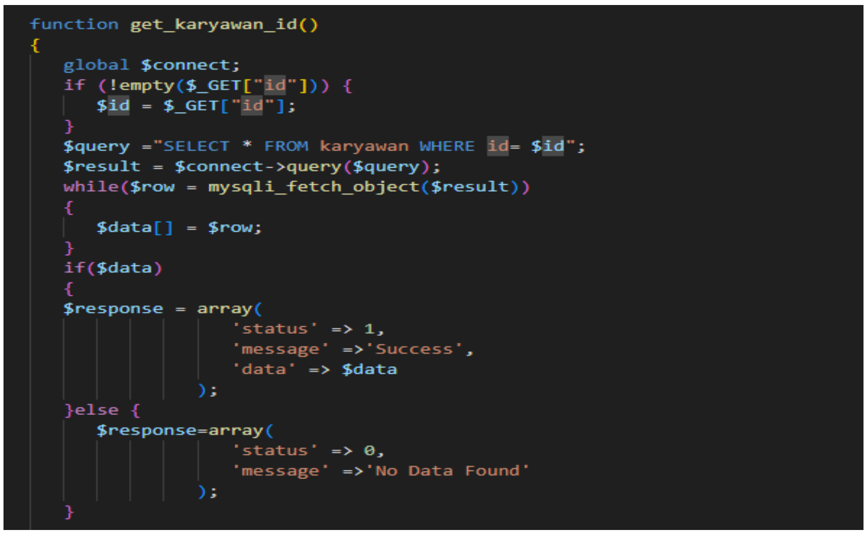
Results of Program Code Efficiency Analysis

The analysis carried out regarding the efficiency of the program code in communicating data with the database using the REST API is shown in Figure 6 and Figure 7.



```
public function get($id)
{
    return Karyawan::find($id);
}
```

Figure 6. Code for developing a REST API with the Laravel framework



```
function get_karyawan_id()
{
    global $connect;
    if (!empty($_GET["id"])) {
        $id = $_GET["id"];
    }
    $query ="SELECT * FROM karyawan WHERE id= $id";
    $result = $connect->query($query);
    while($row = mysqli_fetch_object($result))
    {
        $data[] = $row;
    }
    if($data)
    {
    $response = array(
                    'status' => 1,
                    'message' =>'Success',
                    'data' => $data
                );
    }else {
        $response=array(
                    'status' => 0,
                    'message' =>'No Data Found'
                );
    }
```

Figure 7. Software code for developing a REST API with PHP Native

Based on the analysis carried out in Figure 6 and Figure 7, it was found that the Laravel framework in writing program code is more efficient, or called clean code compared to native PHP. Clean code is code that is readable so that other people can understand the code directly [14]. Simple to follow code makes other software developers know it better, which leads to improved code maintainability [15]. Furthermore, clean code includes set guidelines and procedures that will help programmers in writing program code, such as utilizing meaningful names, indenting, minimizing replication, and a lot more [16]. Eloquent ORM is a Laravel component that maps an object-oriented core model to a database that is relational [17] compared to native PHP which is required to write SQL query syntax first when you want to create a data communication process with the database.

REST API Process Speed Analysis Results

The analysis carried out regarding the URL structure is the global address of documents on the World Wide Web, and it serves as the main means of finding documents on the Internet [18]. URLs in routing commonly called endpoints are resources used to build REST APIs and communicate data between client-server [19], the endpoint structure of the Laravel framework and native PHP is shown in Table 4.

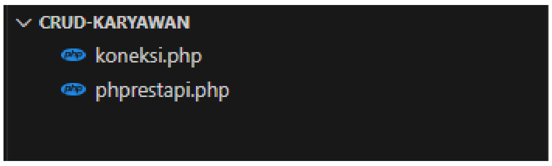
Table 4. REST API URL Structure

Laravel	http://localhost:8000/api/karyawan
PHP Native	http://localhost/karyawan/crud-karyawan-native/phprestapi.php?function=get_karyawan

Based on the analysis carried out in Table 4, it was found that the Laravel framework's endpoint URL structure was superior to the endpoint URL structure of native PHP. The Laravel framework has a routing feature that can manage API endpoints in a structured manner, whereas in native PHP the route call must be included with the process file.

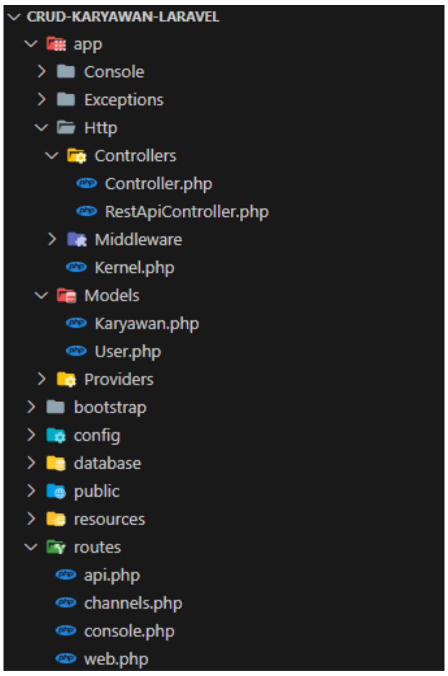
Project Architectural Model Analysis Results

The analysis carried out regarding the project architectural model in the Laravel framework and native PHP is shown in Figure 8 and Figure 9.



```
CRUD-KARYAWAN
    koneksi.php
    phprestapi.php
```

Figure 8. Project architecture model in PHP Native



```
CRUD-KARYAWAN-LARAVEL
  app
    Console
    Exceptions
    Http
      Controllers
        Controller.php
        RestApiController.php
      Middleware
      Kernel.php
    Models
      Karyawan.php
      User.php
    Providers
  bootstrap
  config
  database
  public
  resources
  routes
    api.php
    channels.php
    console.php
    web.php
```

Figure 9. Project architectural model in the Laravel framework

Based on the analysis carried out in Figure 8 and Figure 9, it was found that the Laravel framework in the project architecture model was superior to native PHP. Because the Laravel framework implements a mode architecture, namely MVC (Model View Controller). MVC is a design pattern used to develop web applications, as it helps developers to have a clear understanding of all the modules. The MVC architecture separates the model, view, and controller layers [20]. Meanwhile, native PHP does not apply an architectural model and developers create their own architectural models.

Conclusion

Based on the results of the analysis carried out, this research compares the Laravel framework with PHP Native in the development of REST APIs, focusing on processing speed, code efficiency, URL routing structure, and architectural models. The study identifies four key findings. Firstly, PHP Native demonstrates superior processing speed with an average of 18.3 ms compared to Laravel's 344.52 ms. This advantage is due to the additional processes in Laravel, such as routes, controllers, and models. Secondly, in terms of code efficiency, Laravel excels because of its Object Relational Mapping (ORM) feature, which simplifies database interactions, whereas PHP Native requires manual SQL query writing, leading to a more repetitive and monotonous code structure. Thirdly, Laravel outperforms PHP Native in URL routing structure due to its robust routing features that allow for more organized and intuitive API endpoints. In contrast, PHP Native necessitates including process files in route calls, making the structure less manageable. Lastly, Laravel's adherence to the MVC architectural model provides a clear separation of concerns, enhancing code organization and maintainability. PHP Native, lacking a standardized architectural model, often leads developers to create their own, which can vary in quality. Thus, while PHP Native is faster, Laravel offers superior code efficiency, URL routing, and architectural structure, making it a more robust choice for REST API development.

References

[1] V. Surwase, "REST API Modeling Languages -A Developer ' s Perspective Related papers REST API Modeling Languages - A Developer ' s Perspective," *IJSTE - Int. J. Sci. Technol. Eng.*, vol. 2, no. 10, pp. 634–637, 2016, [Online]. Available: https://www.academia.edu/27064725/REST_API_Modeling_Languages_A_Developers_Perspective?bulkDownload=thisPaper-topRelated-sameAuthor-citingThis-citedByThis-secondOrderCitations&from=cover_page

[2] L. Murphy, T. Alliyu, A. Macvean, M. B. Kery, and B. A. Myers, "Preliminary Analysis of REST API Style Guidelines," *PLATEAU'17 Work. Eval. Usability Program. Lang. Tools*, pp. 1–9, 2017, [Online]. Available: https://codeplanet.io/principles-good-restful-api-design/%0Ahttp://www.cs.cmu.edu/~NatProg/papers/API-Usability-Styleguides-PLATEAU2017.pdf

[3] N. Solanki, D. Shah, and A. Shah, "A Survey on different Framework of PHP," *Int. J. Latest Technol. Eng. Manag. Appl. Sci.*, vol. VI, no. VI, pp. 155–158, 2017, [Online]. Available: https://www.ijltemas.in/DigitalLibrary/Vol.6Issue6/155-158.pdf

[4] M. Doroodchi and S. Dastgheib, "A framework for web application development," *Proc. 2008 Int. Conf. Softw. Eng. Res. Pract. SERP 2008*, no. March, pp. 311–316, 2008, doi: 10.17148/iarjset.2022.9218.

[5] D. C. Schmidt, A. Gokhale, and B. Natarajan, "Frameworks : Why They Are Important and How to Apply Them Effectively 1 Introduction Key Characteristics of Frameworks Key Challenges in Developing and Reusing Frameworks," *Electr. Eng.*

[6] Adam Huda Nugraha, "Making A Web-Based Application For Sales Kitchen Bunda Ghina Using Php And Mysql," *Int. J. Sci. Technol. Manag.*, vol. 2, no. 5, pp. 1787–1792, 2021, doi: 10.46729/ijstm.v2i5.351.

[7] N. J. Sebastian, "Comparative study of the Pros and Cons of Programming languages Java , Scala , C ++ , Haskell , VB . NET , AspectJ , Perl , Ruby & Scheme," *arXiv Prepr. arXiv1008.3431*, 2010.

[8] S. Sotnik, V. Manakov, and V. Lyashenko, "Overview: PHP and MySQL Features for Creating Modern Web Projects," *Int. J. Acad. Inf. Syst. Res.*, vol. 7, no. 1, pp. 11–17, 2023, [Online]. Available: www.ijeais.org/ijaisr

[9] M. I. Kausar Bagwan and P. D. Swati Ghule, "A Modern Review on Laravel-PHP Framework," *IRE Journals*, vol. 2, no. 12, pp. 1–3, 2019.

[10] D. Bretthauer, "Open source software: A history," *Inf. Technol. Libr.*, vol. 21, no. 1, pp. 3–10, 2002.

[11] A. Ramelan, F. Adriyanto, C. H. B. Apribowo, M. H. Ibrahim, M. E. Sulistyo, and K. S. Arief, "Iot lora-based energy management information system with rad method and laravel frameworks," *J. Commun. Softw. Syst.*, vol. 17, no. 4, pp. 366–372, 2021, doi: 10.24138/jcomss-2021-0003.

[12] S. M. . Quadri and S. U. Farooq, "Software Testing – Goals, Principles, and Limitations," *Int. J. Comput. Appl.*, vol. 6, no. 9, pp. 7–10, 2010, doi: 10.5120/1343-1448.

[13] J. Ranta, "Testing AWS hosted Restful APIs with Postman," no. January, 2023.

[14] R. C. Martin, *Clean Code*. 2016. doi: 10.1007/978-1-4842-1212-7_3.

[15] P. Rachow, S. Schroder, and M. Riebisch, "Missing clean code acceptance and support in practice - An empirical study," *Proc. - 25th Australas. Softw. Eng. Conf. ASWEC 2018*, pp. 131–140, 2018, doi: 10.1109/ASWEC.2018.00026.

[16] B. Latte, S. Henning, and M. Wojcieszak, "Clean code: on the use of practices and tools to produce maintainable code for long-living software," *CEUR Workshop Proc.*, vol. 2308, no. February, pp. 96–99, 2019.

[17] H. Halimi and I. Jound, "Comparison of performance between Raw SQL and Eloquent ORM in Laravel," *Fac. Comput. Blekinge Inst. Technol. Sweden*, pp. 1–37, 2016, [Online]. Available: http://www.diva-portal.org/smash/get/diva2:1014983/FULLTEXT02

[18] A. N. S. Charan, Y. H. Chen, and J. L. Chen, "Phishing Websites Detection using Machine Learning with URL Analysis," *Proc. - 2022 IEEE World Conf. Appl. Intell. Comput. AIC 2022*, pp. 808–812, 2022, doi: 10.1109/AIC55036.2022.9848895.

[19] A. Lawi, B. L. E. Panggabean, and T. Yoshida, "Evaluating graphql and rest api services performance in a massive and intensive accessible information system," *Computers*, vol. 10, no. 11, 2021, doi: 10.3390/computers10110138.

[20] S. Singh and J. Iyer, "Comparative study of MVC (model view controller) architecture with respect to struts framework and PHP," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 3, pp. 142–150, 2016.