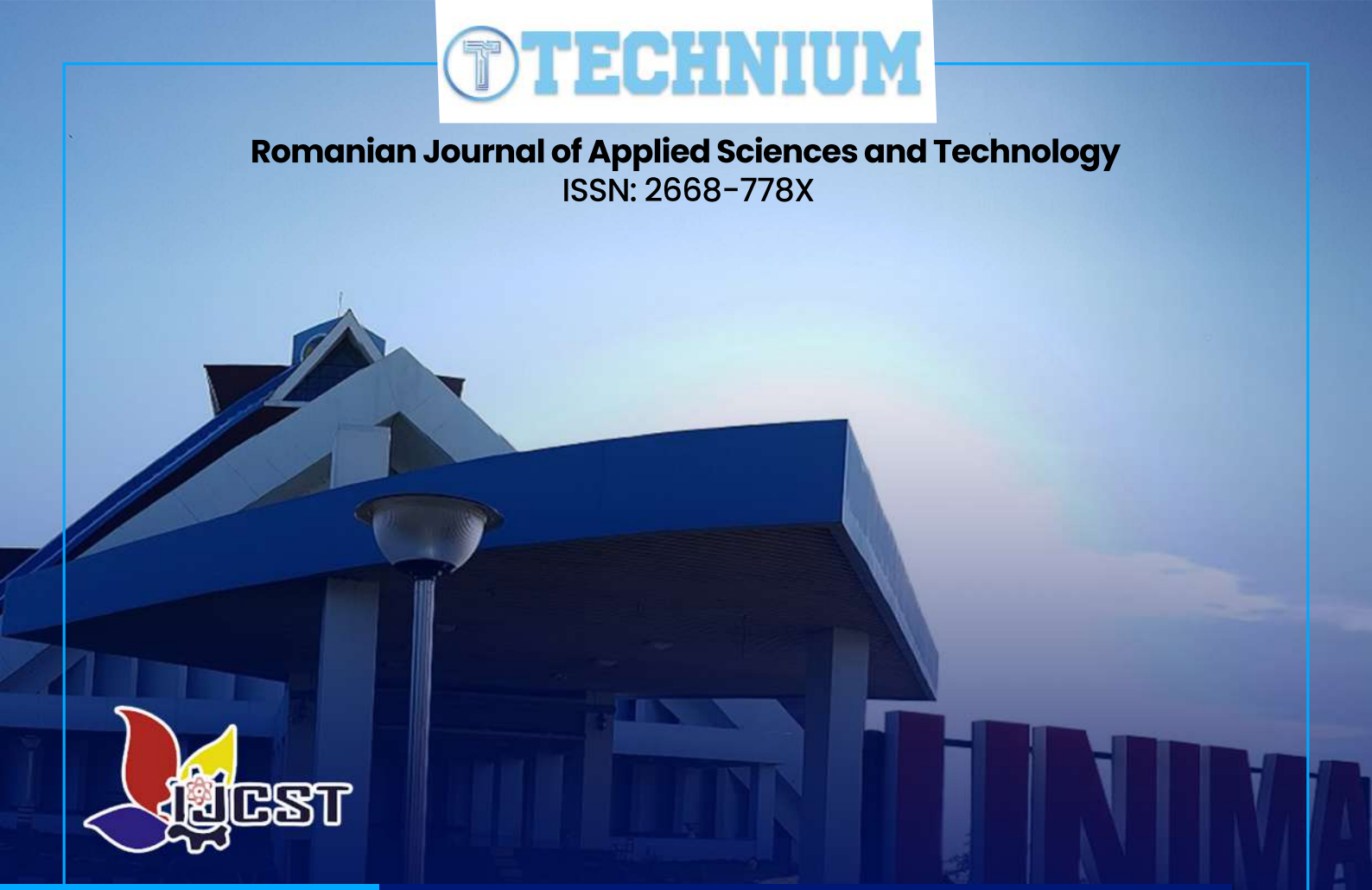
TECHNIUM

Romanian Journal of Applied Sciences and Technology
ISSN: 2668-778X



vol. 16 / 2023

The 7th International Conference on Science Technology

organized by
Faculty of Social Science and
Law Universitas Negeri Manado and
Consortium of International Conference
on Science and Technology

The Innovation Breakthrough in Digital and Disruptive Era

Technium: Romanian Journal of
Applied Sciences and Technology
Vol. 16, 118-123, October, 2023
ISSN: 2668-778X
www.techniumscience.com

An Approach for Automatic Generating RESTFul API Code based on SQL DDL Code

Fawwaz Ali Akbar [1*], *Eka Prakarsa* Mandyartha[2], and *Hendra* Maulana[3]

[1,2] Informatics Department, Faculty of Computer Science, Universitas Pembangunan Nasional Veteran Jawa Timur, Indonesia
[3] Digital Business Department,, Faculty of Computer Science, Universitas Pembangunan Nasional Veteran Jawa Timur, Indonesia

Abstract. RESTFul API is an emerging technology that is used for communication between various applications or systems. RESTFul API is commonly used in the development of systems and applications. Various application, such as mobile, web, or desktop, are use RESTFul API to handle and compute business processes. Currently, Development of the RESTFul API is easier with the code generator tools and some approaches that will generate RESTful API application automatically. With that tools, developers more easier dan efficient when developing their applications. In this research, we try to create a new approach to create a RESTFul API code generator. This approach is based on the SQL DDL code in the SQL dump file as input. We use this approach because, in the initial phase of building a system, developers must provide their database structure before building their applications. With proposed approach, RESTFul API code generator is more effective and suitable with the developer's workflow. Based on testing result, The proposed approach can perform well for automatic generating code for RESTful API based on SQL DDL given. And provide basic functionality in RESTFul API such as create, read, update, and delete data.

* Corresponding author: fawwaz_ali.fik@upnjatim.ac.id

Technium: Romanian Journal of
Applied Sciences and Technology
Vol. 16, 118-123, October, 2023
ISSN: 2668-778X
www.techniumscience.com

1 Introduction

RESTFul API is an emerging technology that is used for communication between various applications. RESTFul API is commonly used in the application development. Various application forms, such as mobile, web, or desktop, use RESTFul API to handle and computing system business processes.

The RESTFul API makes the system architecture more modular. With RESTFul API we can build various application such as Desktop, Mobile, and Web application. Everything is connected by one RESTFul API. Otherwise, a single application can use multiple RESTFul APIs to accommodate its business process needs.

A code generator is a tool used to generate application code automatically. There are many tools and approaches used to generate code. The code generator can generate only the skeleton code or complete code that is ready to run.

The development of the RESTFul API is currently being made easier with the code generator tools and approaches to generate RESTful API application [1][2][3][4][5]. So developers more easier dan efficient when developing their applications. In this research, we try to create a new approach to create a RESTFul API code generator. This approach is based on the SQL DDL code in the SQL dump file. We use this approach because, in the initial phase of building a system, developers must provide their database structure before building their applications. So our proposed approach, RESTFul API code generator approach with SQL DDL files, is more effective and suitable with the developer's workflow.

2 Related Research

Queirós [1] develops a code generator to create a RESTful API. This research was developed on the yeoman platform. The API code developed is based on ExpressJS, MongoDB, Mongoose as a model, and EJS. This code generator implements the MVC concept which divides the controller (ExpressJS) Model (Mongoose/MongoDB) and Views with EJS.

In other studies[2], they develop a web service code generator. The focus is on generating code according to developer needs. First, the user determines the type of API, determines the endpoint of the web service, determines the parameters for the created resource, and finally selects a template for the response.

Gomez et. al [3] do research to develop code generation for the Spring Boot framework in JAVA language. Spring Boot is a part of the Spring framework specifically used for RESTful APIs. This study focuses on code generation for basic CRUD operations. The results of this study have been published as tools in the Eclipse IDE.

3 Proposed Method

In this study, we try to create an approach that adapts to how developers develop their applications.

In general, applications are developed based on the data needed, so first, developers create database designs. After the database design is created, the developer proceeds to the next stage, which is developing an application or system according to the specified needs.

The proposed approach that developed in this study is shown in Figure 1. There are seven steps to develop a code generator for the RESTFul API based on the SQL Dump file. First, parse the data from SQL dump file which contains all the structures and SQL queries within database. The SQL file will be parsed with SQL Parser [6][7]. The function of the SQL parser is to parse each SQL statement into an AST tree.

The second step is choose SQL statements that contain DDL or Data Definition Language. DDL contains the statement structure of a table. We do not use DML statements such as inserting tables and other processes, because what is needed is only the DDL structure. And we only use CREATE statement only.

Third step, the results of the parsing process in the previous step are used as input for the next process, in this case, the SQL DDL code for the table structure. From the table structure, the table name will be taken. The name of the table will be used as a reference for naming the File Model and Naming Resource API.

In the fourth step, the table structure from the output of the second process, it will be parsed to get table's fields from the table that is being processed. The table's fields from DDL SQL will be used as the basis for creating Model Class at the model layer.

The next process is the process of generating Model code and RESTFul API. Both processes are implemented because this approach uses a framework that uses the MVC design pattern which divides the application into several layers. In this study, the target layers are the Model and Controller. At the controller layer, the request-response process from the API is implemented, while at the model layer is used to create an data structure abstraction which is later used by the controller to access data to the database.

The fifth process, generates Model Class code based on the previous process. The output of this process is the file and code on the model layer. The output file will be placed in the '*dist*' folder in the application project.

The sixth process generates the RESTFul API code. The API code uses the output of the previous process such as table name information, table fields, and table primary keys. In addition, Model file that has been previously generated is used in the RESTFul API code Resource file. The output file will be placed in the '*dist*' folder in the application project.

All these processes will be repeated in as many tables as found in the SQL Dump file.

Technium: Romanian Journal of
Applied Sciences and Technology
Vol. 16, 118-123, October, 2023
ISSN: 2668-778X
www.techniumscience.com

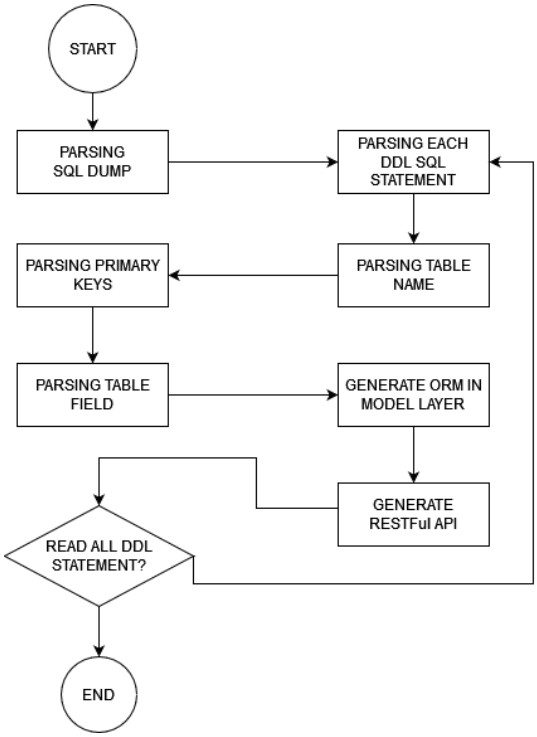


Fig. 1. Proposed Approach for Automatic Generating RESTFul API Code

In this proposed approach, we use CodeIgniter Framework version 4 as target output. The simulation of the research approach is shown as follows. Suppose we have a SQL Dump file:

```
DROP TABLE IF EXISTS `student`;
CREATE TABLE `student` (
  `npm` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  `phone` int(10) unsigned DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`npm`)
)   ENGINE=InnoDB   AUTO_INCREMENT=131242   DEFAULT
CHARSET=utf8;

INSERT   INTO   `student`   VALUES   ('13125',   'Demo',
'8567989', 'email@mail.com');
INSERT  INTO  `student`  VALUES  ('13126',  'sample',
'8567989', 'email@mail.com');
```

The code will be parsed using SQL Parser. The result of the parser is in the form of an AST Tree. After the parser data is obtained, the next step will be to look for the SQL statement containing the table definition that belongs to the DDL category. SQL statements that include into the DML category will be ignored, as will the INSERT statement. So the data obtained after parsing the CREATE TABLE statement is as follows. In this approach we only accomodate CREATE TABLE statement in SQL DDL.

```
CREATE TABLE `student` (
  `npm` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  `phone` int(10) unsigned DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`npm`)
)
```

The next step is the parsing process to get the table name information. This table name will be used for naming files and classes in models and controllers or resources. So it will get information that the name of the table is *'student'*.

After getting the table name information. Then look for the primary key information for each table. From the previous example, the primary key information is obtained from the *PRIMARY KEY ('npm')* syntax. It is parsed so that it only gets the *'npm'* field.

The next process is parsing fields from a table. We will parse the following code:

```
`npm` int(10) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(100) DEFAULT NULL,
`phone` int(10) unsigned DEFAULT NULL,
`email` varchar(100) DEFAULT NULL,
```

The information taken from the code is the field name. Data types are not included because the target language used is loosely typed programming so data type information is not needed. So the output of this process is a list of file names in a table.

```
['npm', 'name', 'phone', 'email']
```

After the table structure information is obtained which includes the table name, primary key, and table fields. The next step is to generate a model. In this approach, we use a simple template to generate model code.

```
//code template for auto generation code

namespace App\Models;

use CodeIgniter\Model;

class {{table name}}Model extends Model
{

    protected $table = {{table name}};
    protected $primaryKey = [ {{pk field}} ];
    protected $allowedFields = [ {{table field}} ];

}
```

After the model is created. The next step is to generate code for the RESTFul API. These codes will handle requests and responses from clients. Same as on the model. We use template code to generate API code. The generated code has the Create, Read, Update, and Delete (CRUD) functionality.

```
//code template for auto generation code

namespace App\Controllers;

use App\Models\{{model name}};
use CodeIgniter\API\ResponseTrait;
use CodeIgniter\RESTful\ResourceController;

class {{table name}}extends ResourceController
{
    use ResponseTrait;

    public function index()
    {
      //code for get all data
```

Technium: Romanian Journal of
Applied Sciences and Technology
Vol. 16, 118-123, October, 2023
ISSN: 2668-778X
www.techniumscience.com

```
    }

    public function show($id = null)
    {
      //code for get specific data
    }

    public function create()
    {
      //code for insert data
    }

    public function update($id = null)
    {
      //code for update data
    }

    public function delete($id = null)
    {
      //code for delete data
    }
}
```

Implementation of the RESTFul API code requires pre-generated Model code. All processes will generate codes that can be used to build RESTFul API applications code.

4 Result and Discussion

The proposed approach is implemented with the PHP programming language and the target framework as the code output is the Codeigniter Framework.

The proposed approach is translated into a code architectural framework for implementation needs. Fig 2. shows the system architecture which is divided into several module.

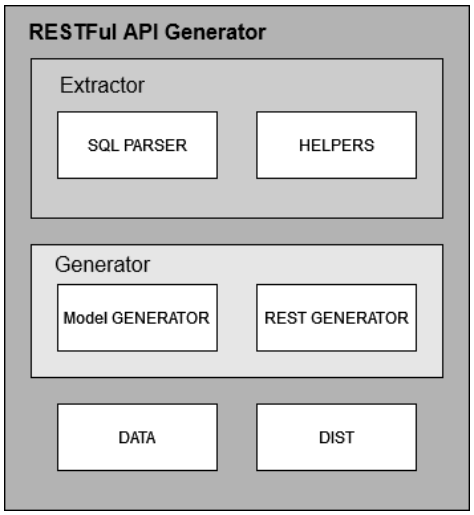


Fig. 2. The Architecture of Proposed Approach

The system architecture is divided into three main module. That are extractor module, generator module, and input-output section. Module Extractor contains SQL Parser code and Helpers Code. The Helpers code contains codes that support system functionality.

The Generator module functions as an automatic code generator to generate the RESTFul API code. The generator module is divided into two main parts: Model Generator and REST Generator. The Model Generator serves as an automatic code generator for the model. While the REST Generator functions as an automatic coder for requests and responses from the client.

In the input-output section, there is 'Data' that serves as storage for SQL Dump files. The SQL Dump file becomes the input of the system. While 'DIST' is a place to store the results of code generation.

The result of this approach is the code files for the Model and the RESTFul API in the form of the Controller code.

The following code is an example of SQL Dump input as the main information in the RESTFul API code generation.

```
/*
MySQL Data Transfer

Source Server         : mariadb
Source Server Version : 50505
Source Host           : localhost:3306
Source Database       : generateapi

Target Server Type    : MYSQL
Target Server Version : 50505
File Encoding         : 65001

Date: 2022-02-16 21:44:54
*/

SET FOREIGN_KEY_CHECKS=0;

-- ----------------------------
-- Table structure for `student`
-- ----------------------------
DROP TABLE IF EXISTS `student`;
CREATE TABLE `student` (
  `npm` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  `phone` int(10) unsigned DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`npm`)
) ENGINE=InnoDB AUTO_INCREMENT=131242 DEFAULT
CHARSET=utf8;


-- ----------------------------
-- Records of student
-- ----------------------------
INSERT INTO `student` VALUES ('13125', 'John', '8567989',
'email@mail.com');
INSERT INTO `student` VALUES ('13126', 'Brad', '8567989',
'email@mail.com');
INSERT INTO `student` VALUES ('13127', 'Kim', '8567989',
'email@mail.com');
```

The result of generating the RESTFul API code from the SQL Dump code will be stored in the 'dist' folder. Shown in Fig 3. The code will be divided into two parts, namely the model and controller folders. The model folder contains the Model file with the class name corresponding to the table name. The controller folder contains a Controller file with a name according to the table name. The number of model and controller files corresponds to the number of DDL Tables in SQL Dump.

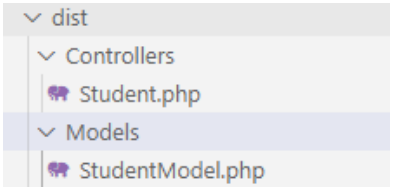


Fig. 3. The Generated Code

Technium: Romanian Journal of
Applied Sciences and Technology
Vol. 16, 118-123, October, 2023
ISSN: 2668-778X
www.techniumscience.com

The following is an example of the result of generating code from the Controller/Student.php file.

```
<?php

/**
 * This code is auto generated
 * student Class for Controller
 */

namespace App\Controllers;

use App\Models\StudentModel;
use CodeIgniter\API\ResponseTrait;
use CodeIgniter\RESTful\ResourceController;

class Student extends ResourceController
{
    use ResponseTrait;

    public function index()
    {
        $model = new StudentModel();
        $data['student'] = $model->findAll();
        return $this->respond($data);
    }

    public function show($id = null)
    {
        $model = new StudentModel();
        $data = $model->where('npm', $id)->first();
        if ($data) {
            return $this->respond($data);
        } else {
            return $this->failNotFound('No data found');
        }
    }

    public function create()
    {
        $model = new StudentModel();
        $data = [
            'npm' => $this->request->getVar('npm'),
            'name' => $this->request->getVar('name'),
            'phone' => $this->request->getVar('phone'),
            'email' => $this->request->getVar('email'),
        ];
        $model->insert($data);
        $response = [
            'status'   => 200,
            'error'    => null,
            'messages' => [
                'success' => 'student created
successfully'
            ]
        ];
        return $this->respondCreated($response);
    }
   ..............
   ..............
```

The following is an example of the result of generating code from the Model/StudentModel.php file.

```
<?php

/**
 * This code is auto generated
 */

namespace App\Models;

use CodeIgniter\Model;

/**
 * student Class for Models
 * @property from Fawwaz Ali Akbar
 */
class StudentModel extends Model
{
    protected $table = 'student';
    protected $primaryKey = ['npm'];
    protected $allowedFields = ['npm', 'name', 'phone',
'email'];
}
```

4.1 Testing

We test the proposed method with two real project. Curriculum Conversion System and School Journal System. We only use SQL DDL from the project. Description of the project shown in Table 1.

Table 1. Data Test

Project	Number of Tables	Type file
Curriculum Conversion System [P1]	5 Table	SQL DDL
School Journal System [P2]	8 Table	SQL DDL

After we run our approach to the data test. We found that proposed approach can automatically generate code RESTful API based on SQL DDL given. The test result shown in Table 2 below.

Table 2. Test Result

Project	Number of Tables	Number of Controller	Number of Model	Functionality
[P1]	5 Table	5 Class	5 Class	CRUD
[P2]	8 Table	8 Class	8 Class	CRUD

Based on test result in Table 2. The proposed approach can perform well for automatic generating code for RESTful API based on SQL DDL given. In P1 have 5 table, and converted in 5 class in Controller and 5 class in Model. Same in P2 have 8 table, and converted in 8 class each in Controller and Model. And every RESTful API code that generated have create, read, update, and delete (CRUD) functionality.

This research is a preliminary study to build a code generator tool for the RESTful API using a framework. There are some of our findings that have not been covered in this study:

- Still accommodates only one framework, that is CodeIgniter version 4.
- The generated code has a one-to-one relationship between the model and the controller. In general, model code can be used across multiple controllers.
- The SQL DDL code structure that is used as a reference is only the 'CREATE TABLE' statement. Need to accommodate other statements like ALTER.
- It is necessary to accommodate the form of SQL code generated by various RDMS.
- Only generate code for Model and Controller.

Based on the findings, research can be developed by accommodating the findings so that the tools developed can have compatibility with many Frameworks and variations of SQL Dump code.

Technium: Romanian Journal of Applied Sciences and Technology
Vol. 16, 118-123, October, 2023
ISSN: 2668-778X
www.techniumscience.com

5 Conclusion

In this study tried to create an approach that adapts to how developers develop their applications to create automatic RESTFul API code generation. This approach uses SQL DDL code on SQL Dump as input. From the SQL DDL information, the RESTFul API code is generated automatically. This research is preliminary research to develop RESTFul API code generator tools based on SQL DDL. Based on testing result, The proposed approach can perform well for automatic generating code for RESTful API based on SQL DDL given. And provide basic functionality in RESTFul API such as create, read, update, and delete data. There are several findings in this study that can be used as the basis for the development of further research.

References

1. Queirós, Ricardo. "Kaang: a RESTful API generator for the modern web." In 7th Symposium on Languages, Applications and Technologies SLATE 2018, pp. 1-1. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018.
2. Queirós, Ricardo. "SeCoGen-A Service Code Generator." In 8th Symposium on Languages, Applications and Technologies (SLATE 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
3. Gómez, Omar S., Raúl H. Rosero, and Karen Cortés-Verdín. "CRUDyLeaf: a DSL for generating spring boot REST APIs from entity CRUD operations." Cybernetics and Information Technologies 20, no. 3 (2020): 3-14.
4. Cao, Hanyang, Jean-Rémy Falleri, and Xavier Blanc. "Automated generation of REST API specification from plain HTML documentation." In Service-Oriented Computing: 15th International Conference, ICSOC 2017, Malaga, Spain, November 13–16, 2017, Proceedings, pp. 453-461. Springer International Publishing, 2017.
5. Xue, Qinghan, Lei Liu, Weipeng Chen, and Mooi Choo Chuah. "Automatic generation and recommendation for API mashups." In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 119-124. IEEE, 2017.
6. Akbar, Fawwaz Ali, Siti Rochimah, and Rizky Januar Akbar. "Investigation of SQL Clone on MVC-based Application." IPTEK Journal of Proceedings Series 1 (2018): 72-77.
7. Swanhart, J., 2022. GitHub - greenlion/PHP-SQL-Parser: A pure PHP SQL (non validating) parser w/ focus on MySQL dialect of SQL. [online] GitHub. Available at: <https://github.com/greenlion/PHP-SQL-Parser> [Accessed 4 September 2022].