SKANIKA: Sistem Komputer dan Teknik Informatika
Volume 5, Nomor 2, Juli 2022, Halaman 136-144 E-ISSN: 2721-4788

Desain Sistem Backend Berbasis REST API Menggunakan Framework Laravel 7

Hendrik Fery Herdiyatmoko
Fakultas Sains dan Teknologi, Universitas Katolik Musi Charitas, Palembang, Indonesia
E-mail: hendrik@ukmc.ac.id
(* : corresponding author)

Abstrak

Bagian *backend* adalah ruang logis dengan fungsionalitas dan pengoperasian aplikasi perangkat lunak atau sistem informasi. Salah satu implementasinya adalah sistem backend berbasis RESTful. RESTful adalah arsitektur pertukaran data menggunakan protokol HTTP. RESTful atau REST Server menyediakan data untuk diakses oleh REST Client menggunakan pertukaran data dalam format JSON. Masalah dalam penelitian ini bagaimana membangun rest server yang didukung perangkat yang aman, mendukung otomatisasi/artisan, dukungan *package* yang mudah, dan dukungan MVC. Tidak semua framework PHP mendukung kebutuhan tersebut kecuali Framework Laravel. Framework Laravel menyediakan mekanisme untuk membangun REST Server melalui *library* Rest Server yang mendukung implementasi RESTful server secara penuh. Pada penelitian ini dibangun REST Server yang dapat menjalankan fungsi dasar CRUD. Hasil dari *endpoint* yaitu mendapatkan respon dari server berupa JSON. Hasil dari penelitian menghasilkan restful API atau REST Server dengan menggunakan empat http *request* yaitu GET, POST, PUT dan DELETE yang merupakan metode dasar dari REST Server dan telah diuji responnya menggunakan Postman.

Kata kunci: Web Service, REST API, JSON

Abstract

The backend is the logical space with the functionality and operation of a software application or information system. One of the implementations is a RESTful based backend system. RESTful is a data exchange architecture using the HTTP protocol. RESTful or REST Server provides data to be accessed by REST Client using data exchange in JSON format. The problem in this research is how to build a rest server that is supported by secure devices, supports automation/artisan, easy package support, and MVC support. Not all PHP frameworks support this need except the Laravel Framework. The Laravel framework provides a mechanism for building REST Servers via the Rest Server library that supports a full RESTful server implementation. In this research, a REST Server is built that can perform basic CRUD functions. The result of the endpoint is getting a response from the server in the form of JSON. The results of the research produce a restful API or REST Server using four http requests, namely GET, POST, PUT and DELETE which are the basic methods of REST Server and the response has been tested using Postman.

Keywords: Web service, REST API, JSON

1. PENDAHULUAN

Web Service dan semantik web adalah elemen kunci dari Web 3.0 [1]. Web 3.0 mendukung layanan pencarian, media sosial berdasarkan fitur atau layanan tunggal pada arsitektur *Web Service*, termasuk REST API, khususnya menggunakan protokol HTTP, yang menggunakan JSON sebagai format pertukaran data [2][3].

Perkembangan teknologi web cenderung terbagi menjadi 3 konsentrasi utama, salah satunya adalah *backend*. Bagian *back-end* mengacu pada program dan script yang bekerja pada server di belakang layar. Jadi, *backend* bisa diartikan sebagai wadah dari logika fungsional inti dan pengoperasian aplikasi perangkat lunak atau sistem informasi. Sistem *back-end* [4] memastikan bahwa data atau layanan yang diminta dan dikirim oleh sistem *frontend* atau aplikasi disampaikan melalui metode terprogram. *Back-end* terdiri dari logika aplikasi inti, database, data integrasi dan aplikasi, API dan proses *back-end* lainnya [4]. Untuk meningkatkan keamanan *Web Service*, beberapa penelitian sebelumya menggunakan Jason Web Token (JWT) sebagai autentikasi data [5][6]. JWT dapat juga diterapkan di framework Laravel sebagi otentikasi login [7].

Web Service yang didalamnya mengandung REST API dapat diintergrasikan di perangkat mobile untuk mendukung interoperabilitas [8] yang dapat diakses di perangkat mobile android ataupun iOS. *Web Service* dapat digunakan juga untuk penggunaan teknologi Change Data Capture (CDC) yang merangkum sumber data dengan satu set *Web Service*[9]

Berbeda dari penelitian – penelitian sebelumnya, dalam penelitian ini, sistem *back-end* akan dibuat dalam bentuk server *backend* berbasis REST API[10][11]. Di dalam sistem REST, dapat melakukan empat metode HTTP *request* yaitu GET, POST, PUT dan DELETE. Keempat metode HTTP ini melakukan proses CRUD dari database yaitu Create, Read, Update, Delete dengan menggunakan format pertukaran data berbentuk JSON.

Framework Laravel adalah framework web PHP *open-source* gratis, dibuat oleh Taylor Otwell dan ditujukan untuk pengembangan aplikasi web mengikuti pola arsitektur model-view-controller (MVC). Beberapa fitur Laravel adalah sistem pengemasan modular dengan *dedicated dependency manager*. Selain framework Laravel, ada satu framework PHP yang umum digunakan oleh Developer yaitu Codeigniter yang memiliki pola arsitektur MVC [12][13].

JavaScript Object Notation (JSON) [14] adalah standar format berbasis teks untuk mewakili data terstruktur berdasarkan sintaks objek JavaScript. JSON biasanya digunakan untuk mentransmisikan data dalam aplikasi web (misalnya, mengirim beberapa data dari server ke klien, sehingga dapat ditampilkan di halaman web, atau sebaliknya). Format JSON secara sintaksis identik dengan kode untuk membuat objek di JavaScript.

2. METODE PENELITIAN

2.1. *Software Specification*

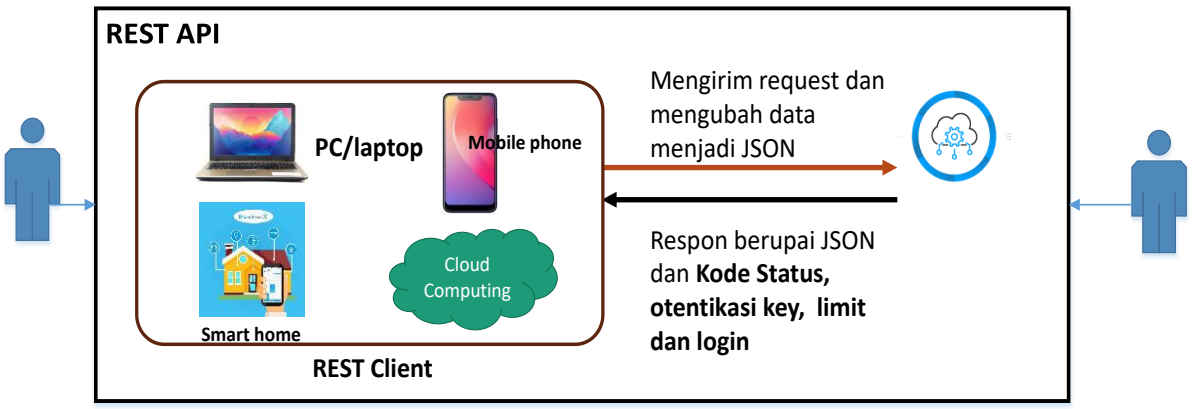
Software yang digunakan dalam menjalankan aplikasi ini adalah sebagai Berikut:

a) Laravel versi 7.x PHP Framework
b) XAMPP web server
c) Aplikasi Postman App untuk respon dari Rest Server

2.2. Tahap Desain

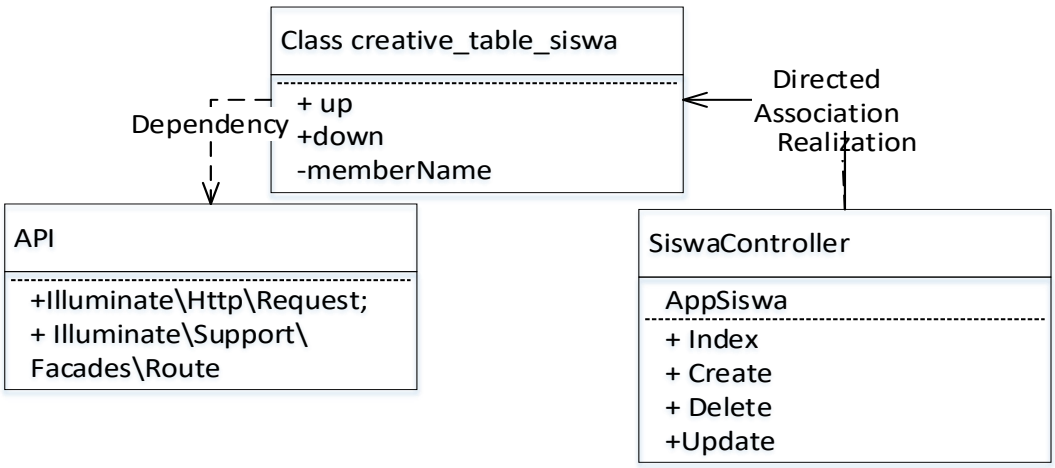
a) Diagram UML (*Unified Modeling Language*) [15]

Diagram *use case* adalah penggambaran grafis dari kemungkinan interaksi pengguna dengan sistem. Diagram use case menunjukkan berbagai use case dan jenis pengguna yang berbeda yang dimiliki. Pada penelitian ini REST Server memiliki satu pemakai yaitu admin. Pada Gambar 1 diperlihatkan diagram *use case* dari penelitian ini.



Gambar 1. Diagram *Use Case Rest Server*

REST Server yang dibangun dibagi menjadi tiga *class* utama yang digunakan yaitu creative_table_siswa, API dan siswaController. *Class-class* tersebut memiliki hubungan asosiatif, yaitu class yang satu mempengaruhi class yang lain. Pada Gambar 2 memperlihatkan class diagram REST Server.



Gambar 2. *Class Diagram*

Pada Gambar 3 Class SiswaController memiliki empat method yaitu method index adalah method default dari Class SiswaController, method create digunakan untuk menjalakan fungsi create atau insert data ke tabel, method delete digunakan untuk menghapus record dari tabel siswa dan method update untuk mengubah *record* tabel siswa.

3. HASIL DAN PEMBAHASAN

Dalam penelitian ini digunakan empat *Endpoint* API [7] yang merepresentasikan CRUD data siswa. *Endpoint* tersebut adalah:

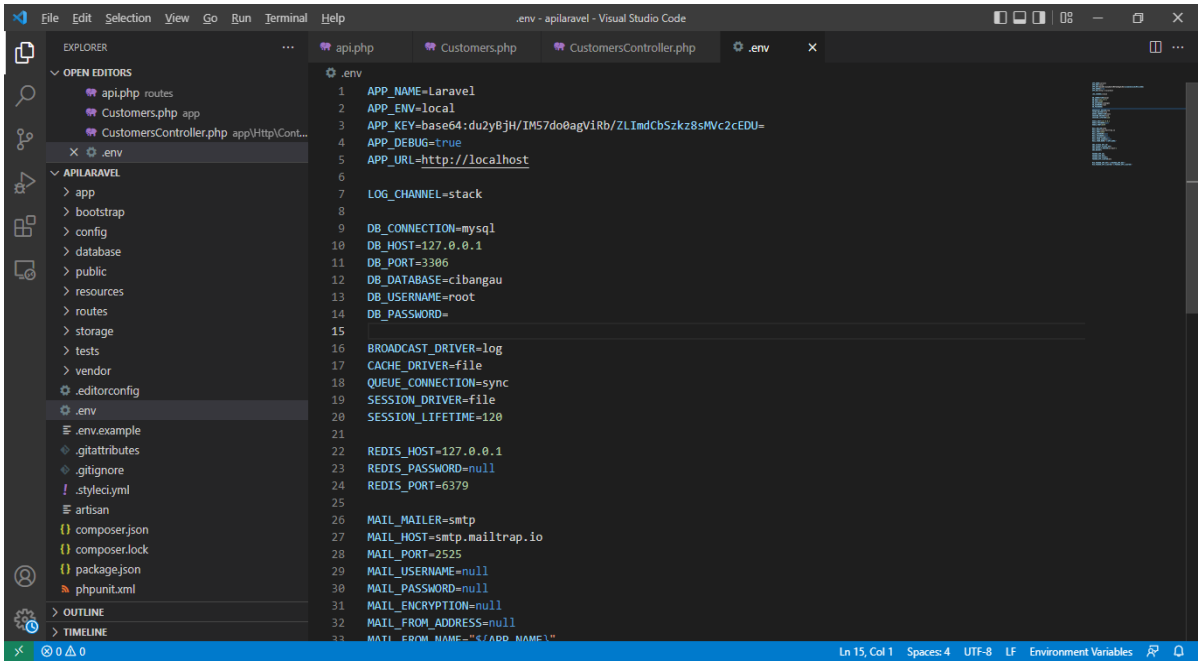
a. GET: http://127.0.0.1:8000/api/siswa, akan mengembalikan semua data siswa dengan menerima GET *request*.
b. POST: http://127.0.0.1:8000/api/siswa/{id}, akan membuat record siswa baru dan akan menerima POST *request*.
c. PUT: http://127.0.0.1:8000/api/siswa/{id}, akan mengubah data siswa yang ada dengan mengacu pada id siswa dan akan menerima PUT *request*.
d. DELETE http://127.0.0.1:8000/api/siswa/{id}, akan menghapus data siswa dengan merujuk pada id siswa dan menerima DELETE *request*

Dalam membangun REST Server ini, dilakukan beberapa tahapan agar rest server dapat memberi respon.

a. Tahap Pertama Adalah Membuat Model Dan Controller
b. Tahap Kedua, Membuat Database Dengan Menggunakan *Migration*
c. Tahap Ketiga Konfigurasi Controller Yang Telah Dibuat
d. Tahap Keempat Adalah Konfigurasi Router Dan Terakhir Adalah Uji Coba REST CLIENT REST API Dengan Menggunakan Ekstensi Visual Studio Thunder Client.

3.1. Konfigurasi *Database*

Tahap konfigurasi *database* dilakukan sebelum membuat REST Server, Langkah ini berguna untuk mempersiapkan database yang akan digunakan dalam project. Gambar 3 merupakan konfigurasi database yang ada di file. env.



Gambar 3. Konfigurasi Database Laravel

3.2. Membuat Model dan Controller

Untuk mebuat model dan controller dipergunakan perintah berikut pada terminal: php artisan make:model Siswa -c –m.

Perintah tersebut artinya membuat model dengan bana Siswa pada laravel sekaligus membuat controller

3.3. Membuat Database Migration

Fitur migration sudah lama diperkenalkan di Laravel. Migration adalah sebuat fitur yang ada pada Laravel, yang merupakan *conrol version system* untuk database. Dengan menggunakan migration Laravel, memungkinkan developer aplikasi untuk mengelola database dengan lebih mudah. Dengan migration database tidak perlu dibuat menggunakan phpMyAdmin namun dapa dilakukan dengan me-*generate* file – file *migration* sebagai *control system.*

Untuk membuat *migration* digunakan perintah pada pseudocode sebagai berikut:

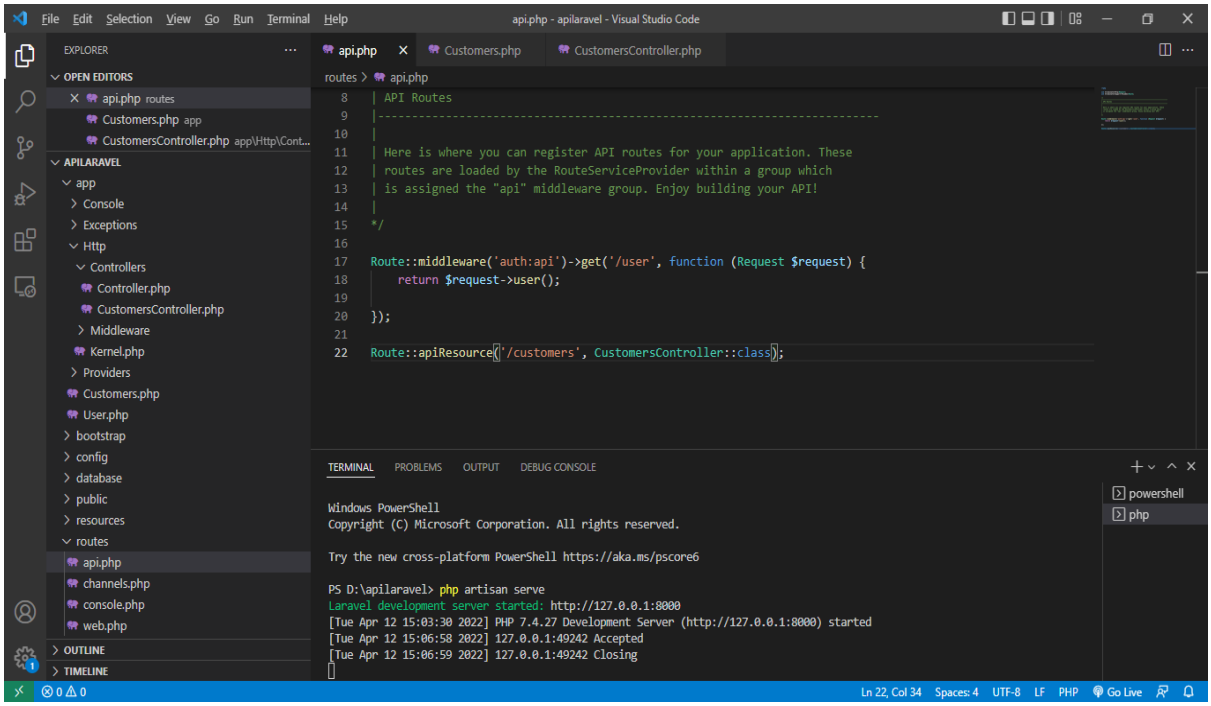
Dengan *migartion* ini akan dibuat tabel siswa yang memiliki 4 field yaitu id, nama, alamat, create_at, update_at.

```
class CreateSiswasTable extends Migration
{
/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('siswas', function (Blueprint $table) {
$table->increments('id');
$table->string('nama');
$table->string('alamat');
$table->timestamps();
});
```

Gambar 4. Membuat Migration

3.4. Routes Laravel

Routes berfungsi untuk mengatur lalu lintas file berdasarkan request dari pengguna. Routes terletak di dalam folder /routes. Routes utama Laravel terletak dalam file web.php. pada Gambar 4 dijelaskan konfigurasi route pada project ini.



Gambar 5. Konfigurasi Route

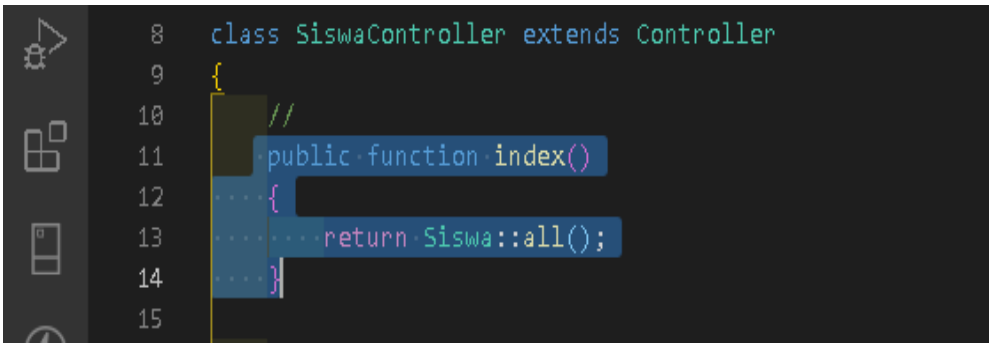
3.5. Konfigurasi Controller

Langkah awal membuat Controller SiswaController adalah dengan memanggil model yang telah dibuat sebelumnya. Padas gambar 4 ditunjukkan potongan program untuk menggunakan model.

```
<? php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Siswa;
```

Gambar 6. Pseudocode Model Siswa

Tahap berikutnya dalam controller adalah membuat empat method yang memiliki fungsi CRUD. Method pertama adalah index untuk menampilkan semua data siswa dengan menerima GET *request.*



Gambar 7. Method index

Untuk *method create*, *update* dan *delete* berturut – turut sebagai berikut:

```php
    public function update(request $request, $id)
    {
        // $siswa = new Siswa;
        $nama = $request->nama;
        $alamat = $request->alamat;

        $siswa = Siswa::find($id);
        $siswa->nama = $nama;
        $siswa->alamat = $alamat;
        $siswa->save();

        return 'Data siswa berhasil diupdate';
    }

    public function delete(Request $request)
    {
        $siswa = new Siswa;
        $siswa->nama = $request->nama;
        $siswa->alamat = $request->alamat;
        $siswa->save();

        return 'Data siswa berhasil dihapus';
    }
}
```

Gambar 8. *Pseudocode* create, *update* dan *delete*

3.6. Konfigurasi Route

Pada file api.php diatur route untuk semua proses CRUD yang ada. Gambar 9 adalah *pseudocode* dari route fungsi route api.php digunakan untuk memetakan controller dan model mana yang akan digunakan.

routes > api.php

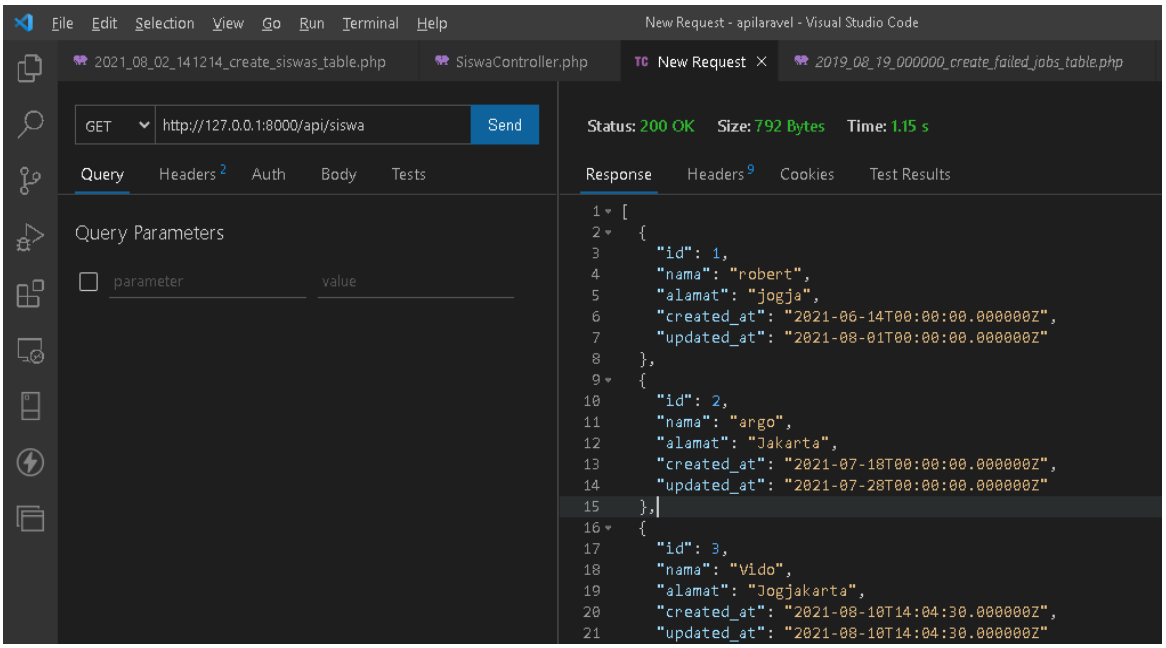
```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::get('siswa', 'SiswaController@index');
Route::post('siswa', 'SiswaController@create');
Route::put('/siswa/{id}', 'SiswaController@update');
Route::delete('/siswa/{id}', 'SiswaController@delete');
```

Gambar 9. *Pseudocode Route*

3.7. Pengujian

a) Pengujian endpoint GET http://127.0.0.1:8000/api/siswa

Pengujian http *request* GET dilakukan dengan menggunakan aplikasi Postman. Gambar 10 berikut merupakan hasil dari *endpoint* GET http://127.0.0.1:8000/api/siswa.

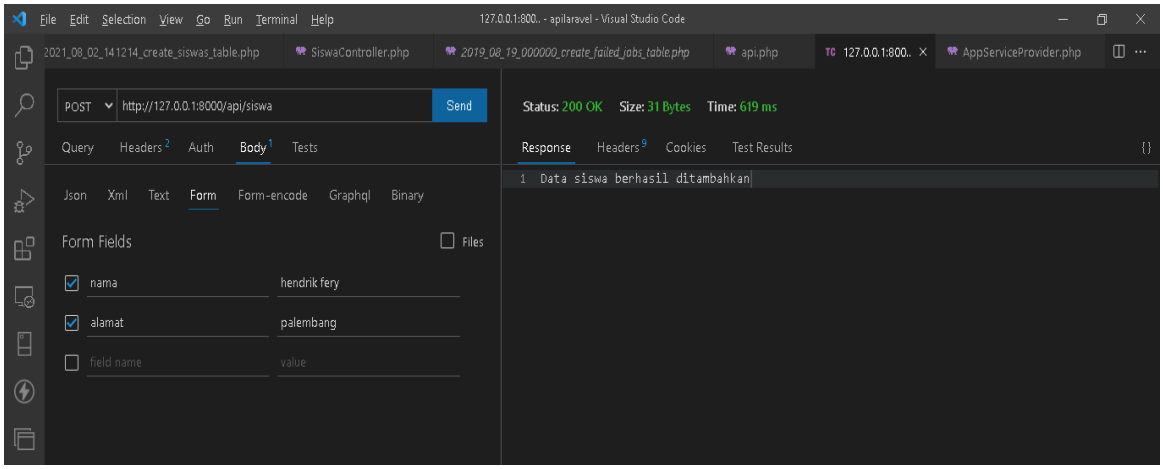


Gambar 10. Respon dari Endpoint GET

Hasil yang dicapai adalah respon berupa json dari semua data siswa, dalam hal ini dari tabel siswa, dan mendapatkan http status adalah 200.

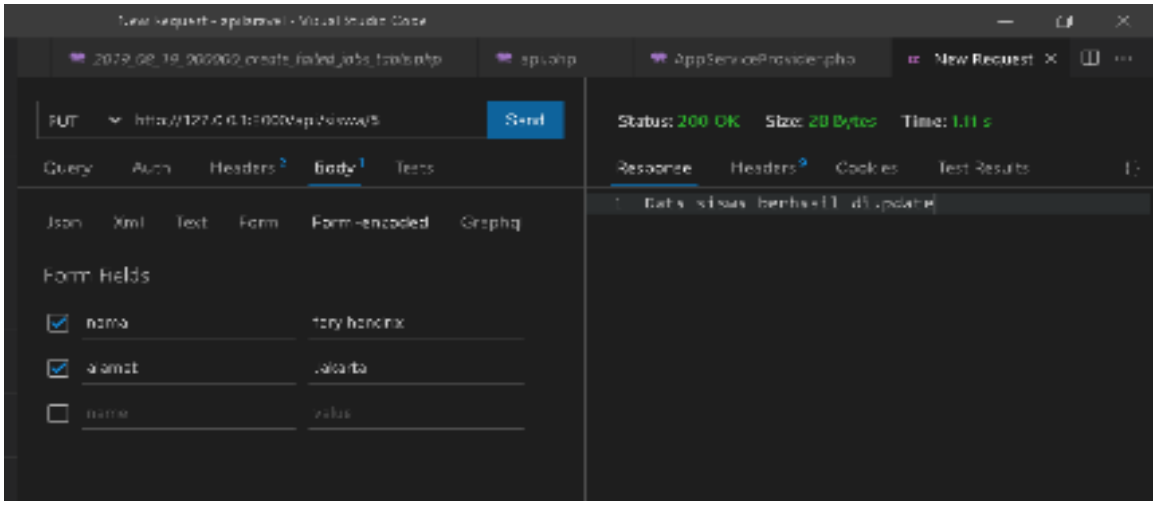
b) Pengujian *endpoint* POST: http://127.0.0.1:8000/api/siswa

Endpoint POST: http://127.0.0.1:8000/api/siswa digunakan untuk menambah data ke dalam tabel siswa. pengujian dilakukan dengan aplikasi Postman dengan *endpoint* http://127.0.0.1:8000/api/siswa, pada tab body-form data, isikan pada kolom *key* nama field sesuai dengan kolom tabel siswa dan *value* adalah nilai yang akan ditambahkan pada tabel siswa. konfigurasi benar maka akan mendapatkan pesan “data berhasil ditambahkan”.



Gambar 11. Respon Endpoint Method POST

c) Pengujian endpoint PUT: http://127.0.0.1:8000/api/siswa/{id}

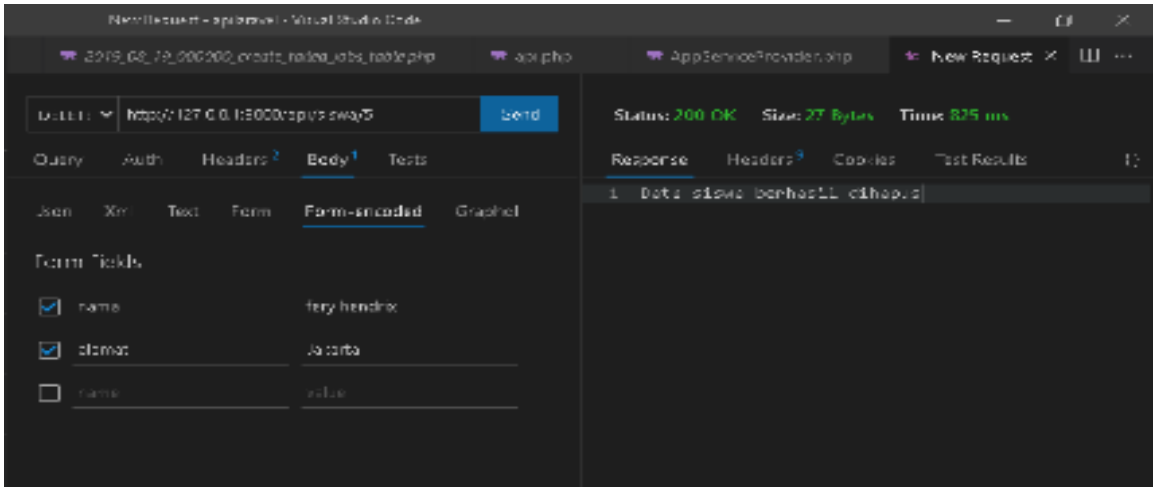


Gambar 12. Respon Endpoint Method PUT

Endpoint ini berfungsi untuk melakukan perintah ubah pada kolom tabel siswa. Jika fungsi update REST API sukses maka respon yang diperoleh adalah pesan "data berhasil diubah" dan kode status 200. Kode 200 adalah kode status HTTP yang artinya successful. Jika gagal melakukan *request* ke REST Server maka kode status adalah 400 atau gagal koneksi.

d) Pengujian endpoint DELETE, http://127.0.0.1:8000/api/siswa/{id}

Endpoint terakhir ini berfungsi untuk menghapus satu *field* dari tabel siswa. Request method Delete sama dengan fungsi SQL yaitu untuk menghapus satu baris data atau satu record. Pada *request method* Delete perlu disertakan ID dari record yang akan diharpus. Cara ini mutlak dilakukan untuk memastikan *record* yang akan dihapus sesuai dengan ID tabel siswa. Gambar 13 adalah hasil dari *endpoint* tersebut.



Gambar 13. Respon Endpoint Method PUT

4. KESIMPULAN

Hasil dari *endpoint* mendapatkan respon dari server berupa json. respon json ini yang dipakai sebagai format petukaran data pada teknologi REST API. Hasil dari penelitian ini diperoleh hasil respon melalui pengujian *request method* GET diperoleh respon dari server dan data diberikan dalam format json data siswa dengan status code 202. Demikian juga untuk *request method* POST, PUT dan DELETE memberikan hasil dengan status code 202 atau *accepted.* Di dalam Framework Laravel

Terdapat beberapa *library* lain yang dapat digunakan untuk membuat REST API seperti dengan Fraktal dan Laravel Transport.

DAFTAR PUSTAKA

[1] J. Hendler, "Web 3.0 Emerging," *Computer*, vol. 42, no. 1, pp. 111-113, 2009.
[2] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Theses for PhD*, pp. 162, 2000.
[3] D. Serrano, E. Stroulia, D. Lau, and T. Ng, "Linked REST APIs: A Middleware for Semantic REST API Integration," *Proc. - 2017 IEEE 24th Int. Conf. Web Serv. ICWS 2017*, pp. 138–145, 2018.
[4] K. Boonchuay, Y. Intasorn, and K. Rattanaopas, "Design and implementation a REST API for association rule mining," *2017 14th International Conference on Electrical Engineering Computer Telecommunications and Information Technology (ECTI-CON)*, pp. 668–671, 2017.
[5] M. M. Hidayat, R. Dimas Adityo, and A. Siswanto, "Design of Restaurant Billing System (E Bill Resto) by Applying Synchronization of Data Billing in Branch Companies to Main Companies Based on Rest API," *Proceeding - ICoSTA 2020 2020 International Conference Smart Technology Appl. Empower. Ind. IoT by Implement. Green Technol. Sustain. Dev.*, 2020.
[6] M. Hosono, H. Washizaki, Y. Fukazawa, and K. Honda, "An Empirical Study on the Reliability of the Web API Document," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2018-December, pp. 715–716, 2018.
[7] X. Chen, Z. Ji, Y. Fan, and Y. Zhan, "Restful API Architecture Based on Laravel Framework," *J. Phys. Conf. Ser.*, vol. 910, no. 1, 2019.
[8] F. Andry, L. Wan, and D. Nicholson, "A mobile application accessing patients' health records through a rest API: How REST-style architecture can help speed up the development of mobile health care applications," *Heal. 2011 - Proc. Int. Conf. Heal. Informatics*, pp. 27–32, 2019.
[9] M. J. Eccles, D. J. Evans, and A. J. Beaumont, "True real-time change data capture with web service database encapsulation," *Proc. - 2010 6th World Congr. Serv. Serv. 2010*, pp. 128–131, 2020.
[10] T. Katayama, M. Nakao, and T. Takagi, "TogoWS: Integrated SOAP and REST APIs for interoperable bioinformatics web services," *Nucleic Acids Res.*, vol. 38, no. 2, pp. 706–711, 2010.
[11] T. Haselmann, G. Thies, and G. Vossen, "Looking into a REST-based API for database-as-a-service systems," *Proc. - 12th IEEE International Conference on Commerce Enterprise Computing CEC 2010*, pp. 17–24, 2020.
[12] A. A. Kayode and A. O. Alabi, "Design and Implementation of a Simplified CodeIgniter Framework for Commercial Vehicles Ticket Reservation System," *Asian J. Res. Computer Science*, vol. 7, no. 2, pp. 1–12, 2021.
[13] D. E. K. Mahardika and M. U. Siregar, "Design and Development of Web Based Employee Payroll Information System Using Codeigniter Framework and Extreme Programming Method," *IJID (International Journal Informatics Dev.*, vol. 7, no. 2, p. 1, 2019.
[14] K. Togias and A. Kameas, "An ontology-based representation of the twitter REST API," *Proc. - International Conference on Tools with Artificial Intelligence. ICTAI*, vol. 1, April 2015, pp. 998–1003, 2018.
[15] "Learn UML Simply Easy Learning." [Online]. Available: https://www.tutorialspoint.com/uml/index.htm. [Accessed: 15-Mar-2019].