

VERSION 4.0  
SEPTEMBER 12, 2023



# PEMROGRAMAN WEB

## FRAMEWORK LARAVEL DASAR (BACKEND) – MODUL 6

DISUSUN OLEH:  
M. SYAUQI AMIQ AMRULLAH  
ALIF FATWA RAMADHANI

DIAUDIT OLEH:  
AMINUDIN, S.KOM., M.CS.

LAB. INFORMATIKA  
UNIVERSITAS MUHAMMADIYAH MALANG

## PEMROGRAMAN WEB

---

### PERSIAPAN MATERI

- [https://www.w3schools.com/php/php\\_oop\\_what\\_is.asp](https://www.w3schools.com/php/php_oop_what_is.asp)
  - <https://www.w3schools.com/php/default.asp>
- 

### TUJUAN

Mahasiswa mampu menggunakan Framework Laravel dan melakukan komunikasi database dengan Framework Laravel, Modul ini telah disesuaikan dengan beberapa poin penilaian Materi Uji Kompetensi.

### PERSIAPAN SOFTWARE/APLIKASI

#### Hardware dan Infrastruktur

- Laptop/PC
- Koneksi Internet

#### Software

- Text Editor: Visual Studio Code (Recomended)
- Extension VSCode: PHP Intelephense (Code Formatter agar Rapih)
- PHP Versi 8.1 atau lebih
- Chrome Extension: JSON Viewer
- MySQL 8.0+

#### Settings:

- Default Formatter pada VSCode di setting dengan extension PHP Intelephense

---

### NOTES

**Modul ini hanya menjelaskan dasar – dasar dari framework laravel, kunjungi dokumentasi resmi dari Laravel untuk penjelasan lengkapnya.**

<https://laravel.com/docs/10.x/installation>

**Lalu persiapkan juga Composer pada device kalian yang berfungsi sebagai package manager Laravel nantinya. Silahkan explore dan cari di google untuk penginstalan Composer.**

---

## MATERI

### FRAMEWORK LARAVEL

Mengutip dari dokumentasi resminya Laravel adalah kerangka kerja aplikasi web dengan sintaksis yang ekspresif dan elegan. Sebuah kerangka kerja web menyediakan struktur dan titik awal untuk membuat aplikasi Anda, memungkinkan Anda fokus untuk membuat sesuatu yang luar biasa sementara kami mengurus detail-detailnya.

Laravel berusaha untuk memberikan pengalaman pengembang yang luar biasa sambil menyediakan fitur-fitur kuat seperti injeksi dependensi yang mendalam, lapisan abstraksi database yang ekspresif, antrian dan pekerjaan terjadwal, pengujian unit dan integrasi, dan banyak lagi.

Baik Anda baru mengenal kerangka kerja web PHP atau memiliki pengalaman bertahun-tahun, Laravel adalah kerangka kerja yang bisa tumbuh bersama Anda. Kami akan membantu Anda untuk mengambil langkah-langkah pertama sebagai pengembang web atau memberikan dorongan saat Anda meningkatkan keahlian Anda ke level berikutnya. Kami tak sabar untuk melihat apa yang Anda bangun.

---

## CODELAB

### A. Installation

Sebelum melakukan instalasi Laravel pastikan pada device kalian sudah terinstall PHP versi 8.1+ dan Composer. Untuk memastikan PHP dan Composer kalian sudah terinstall coba jalankan command berikut pada terminal kalian:

**“php -v”**

Maka akan muncul versi PHP kalian, dan untuk mengecek Composer silahkan jalankan:

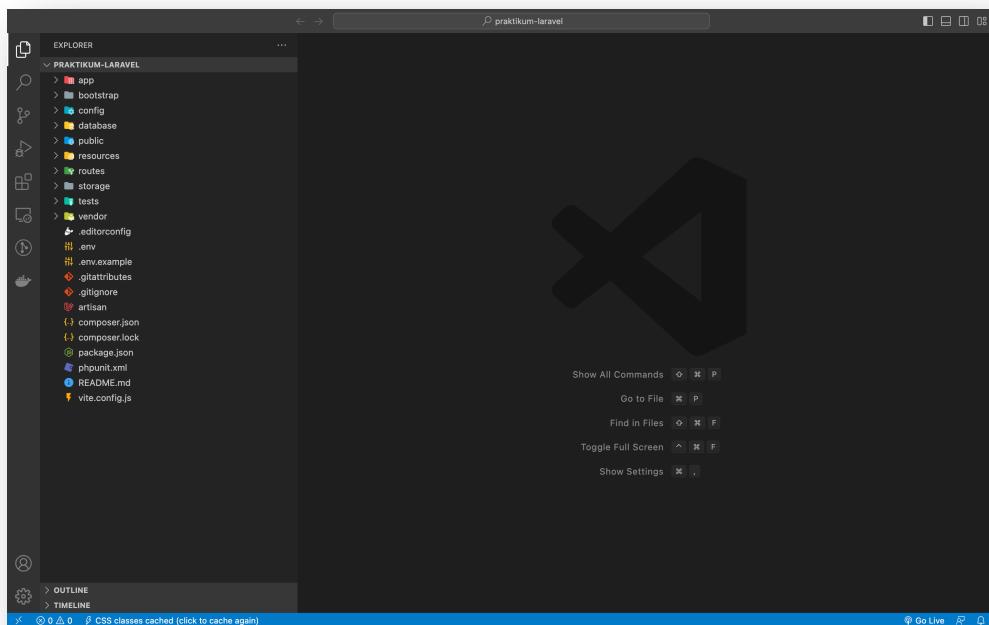
**“composer”**

Maka akan muncul versi Composer kalian.

Selanjutnya buka terminal pada directory tempat kalian akan menyimpan project Laravel nya. Jalankan command berikut:

**“composer create-project laravel/laravel praktikum-laravel”**

Maka akan terbuat folder project Laravel baru bernama “praktikum-laravel”, selanjutnya buka folder tersebut dengan visual studio code kalian seperti pada gambar berikut:



Selamat kalian sudah membuat project laravel dengan nama “praktikum-laravel”, selanjutnya kita akan setting environtmentnya.

## B. Environtment

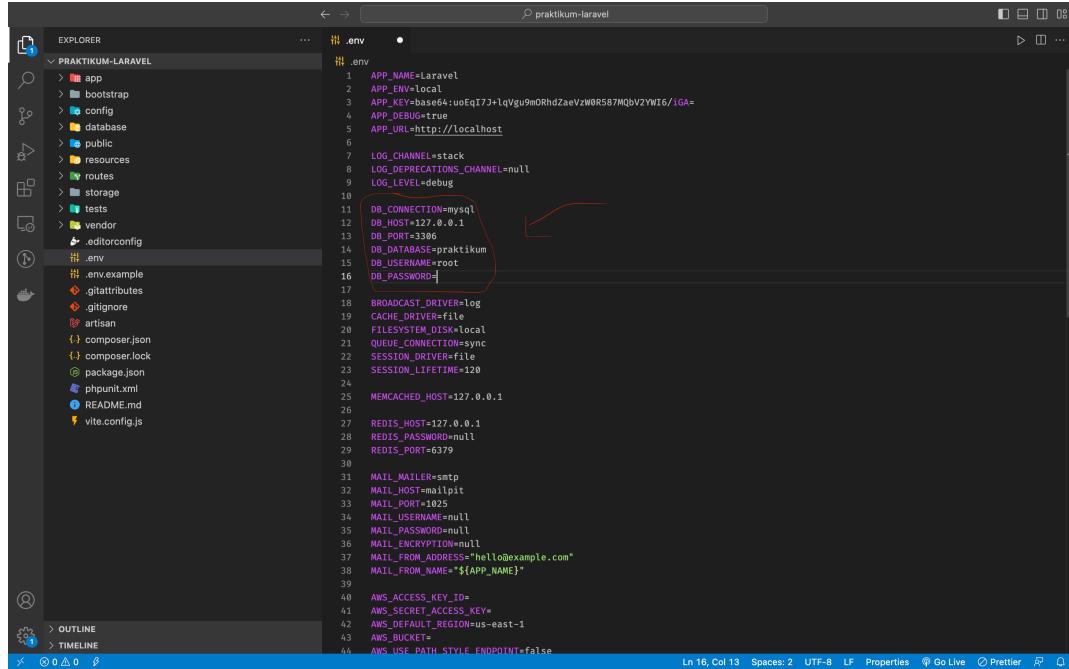
Environtment adalah lingkungan kerja kita dimana environtment ini berfungsi untuk meletakkan variable yang terkait dengan koneksi ke database, nama app, base url, dan variable – variable lainnya. Pada framework Laravel, environtment ditandai dengan file bernama “.env”. Silahkan buka file “.env” kalian seperti pada gambar berikut:

```

.env
1 APP_NAME=laravel
2 APP_ENV=local
3 APP_KEY=base64:uoEqI7J+lqVgu9mOrhdZaeVzW0R587MQbV2YWI6/1Gw
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DRIVER=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDISS_HOST=127.0.0.1
28 REDISS_PASSWORD=null
29 REDISS_PORT=6379
30
31 MAIL_DRIVER=smtp
32 MAIL_HOST=mailtrap
33 MAIL_PORT=1025
34 MAIL_USERNAME=null
35 MAIL_PASSWORD=null
36 MAIL_ENCRYPTION=null
37 MAIL_FROM_ADDRESS="Hello@example.com"
38 MAIL_FROM_NAME="${APP_NAME}"
39
40 AWS_ACCESS_KEY_ID=
41 AWS_SECRET_ACCESS_KEY=
42 AWS_DEFAULT_REGION=us-east-1
43 AWS_BUCKET=
44 AWS_USE_PATH_STYLE_ENDPOINT=false

```

Pada bagian variable yang diawali dengan “DB\_” adalah variable yang berfungsi untuk melakukan setting koneksi database kita. Sebelumnya pastikan kalian sudah menjalankan MySQL seperti pada modul sebelumnya. Lakukan konfigurasi pada variable tersebut yang sesuai dengan konfigurasi database pada device kalian seperti pada gambar berikut:



```

.env
1 APP_NAME=laravel
2 APP_ENV=local
3 APP_KEY=base64:uoEqI7J+lqVgu9mORhdZaeVzW0R587MqbV2YWI6/1Gw=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=praktikum
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtplib
32 MAIL_HOST=mailpit
33 MAIL_PORT=1025
34 MAIL_USERNAME=null
35 MAIL_PASSWORD=null
36 MAIL_ENCRYPTION=null
37 MAIL_FROM_ADDRESS="hello@example.com"
38 MAIL_FROM_NAME="${APP_NAME}"
39
40 AWS_ACCESS_KEY_ID=
41 AWS_SECRET_ACCESS_KEY=
42 AWS_DEFAULT_REGION=us-east-1
43 AWS_BUCKET=
44 AWS_USE_PATH_STYLE_ENDPOINT=false

```

Sesuaikan dengan database yang kalian gunakan, jika MySQL pada device kalian tidak menggunakan password, cukup kosongi pada bagian “DB\_PASSWORD”. Pada bagian “DB\_HOST” tercatat secara default yaitu 127.0.0.1 yang bermaksud IP tersebut adalah “localhost”. Kalian bisa melakukan konfigurasi pada variable lainnya sesuai dengan kebutuhan kalian dan bahkan kalian bisa membuat variable baru.

### C. Migration

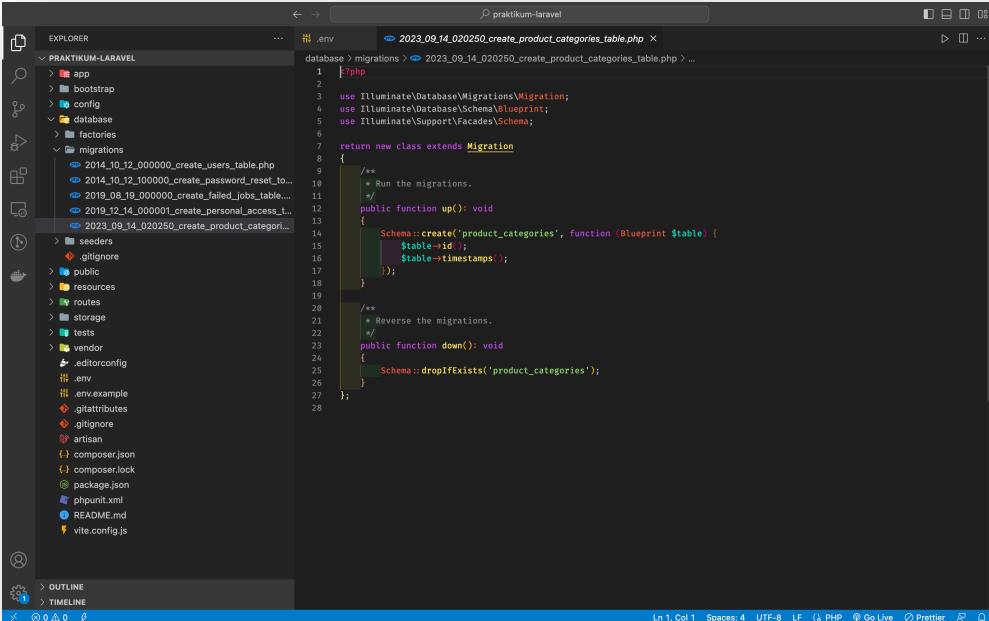
Laravel migration adalah fitur untuk membuat suatu skema tabel dengan berbentuk sebuah code. Dalam membuat tabel menggunakan migration pastikan dibuat dengan aturan konvensi yang berlaku. Untuk membuat sebuah tabel pastikan dibuat konsep “plural” seperti contohnya “product\_categories”. Untuk membuat tabel “product\_categories” menggunakan Laravel migration cukup ketikkan command berikut pada terminal di directory project Laravel kita:

“php artisan make:migration create\_product\_categories\_table”

Maka akan muncul seperti gambar berikut:

```
sh-3.2$ php artisan make:migration create_product_categories_table
[INFO] Migration [database/migrations/2023_09_14_020250_create_product_categories_table.php] created successfully.
sh-3.2$
```

Lalu buka folder “database” dan masuk ke dalam folder “migrations” dan file migration yang kalian buat akan berada di directory ini. Silahkan buka file yang baru saja kalian buat dan akan terlihat seperti pada gambar berikut:



The screenshot shows a code editor with a dark theme. On the left is the file tree (EXPLORER) for a Laravel project named "PRAKTIKUM-LARAVEL". The "migrations" folder contains several files, and "2023\_09\_14\_020250\_create\_product\_categories\_table.php" is selected. The right pane shows the contents of this migration file:

```

1  php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10     * Run the migrations.
11     */
12    public function up(): void
13    {
14        Schema::create('product_categories', function (Blueprint $table) {
15            $table-id();
16            $table->timestamps();
17        });
18    }
19
20    /**
21     * Reverse the migrations.
22     */
23    public function down(): void
24    {
25        Schema::dropIfExists('product_categories');
26    }
27}

```

At the bottom of the editor, there are status indicators: Line 1, Col 1, Spaces: 4, UTF-8, LF, Go Live, Prettier, and a refresh icon.

Kalian bisa membuat field pada tabel “product\_categories” dengan cara mengetikan syntax migration seperti berikut:

```
● ● ●  
1 Schema::create('product_categories', function (Blueprint $table) {  
2     $table->id();  
3     $table->string("name");  
4     $table->string("description")->nullable();  
5     $table->timestamps();  
6 });
```

Untuk syntax selengkapnya kalian bisa explore sendiri pada dokumentasi resmi Laravel dan pada sumber lainnya. Setelah kalian membuat skema tabel untuk merealisasikan tabel tersebut silahkan ketik command berikut pada terminal:

“php artisan migrate”

Maka akan muncul seperti berikut:

```
sh-3.2$ php artisan migrate  
[INFO] Preparing database.  
Creating migration table ..... 87ms DONE  
[INFO] Running migrations.  
2014_10_12_000000_create_users_table ..... 86ms DONE  
2014_10_12_100000_create_password_reset_tokens_table ..... 73ms DONE  
2019_08_19_000000_create_failed_jobs_table ..... 50ms DONE  
2019_12_14_000001_create_personal_access_tokens_table ..... 79ms DONE  
2023_09_14_020250_create_product_categories_table ..... 21ms DONE  
sh-3.2$
```

Setelah berhasil silahkan cek pada database kalian melalui phpMyAdmin atau aplikasi DBMS lainnya. Maka akan terbuat tabel “product\_categories” dengan attribute sesuai dengan kode yang kalian buat:

Name	Rows	Data Length	Engine	Created Date
failed_jobs	0	16.00 KB	InnoDB	2023-09-14 02:23:09
migrations	0	16.00 KB	InnoDB	2023-09-14 02:23:09
password_reset_tokens	0	16.00 KB	InnoDB	2023-09-14 02:23:09
personal_access_tokens	0	16.00 KB	InnoDB	2023-09-14 02:23:09
product_categories	0	16.00 KB	InnoDB	2023-09-14 02:23:09
products	3	16.00 KB	InnoDB	2023-08-25 03:23:09
users	0	16.00 KB	InnoDB	2023-09-14 02:23:09

```

CREATE TABLE `product_categories` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
)

```

Tentunya banyak syntax command untuk Laravel migration tidak hanya “php artisan migrate” kalian bisa explore syntax lainnya pada dokumentasi resmi Laravel.

#### D. Model

Pada framework Laravel terdapat fitur ORM (Object Relational Mapping) bernama Laravel Eloquent, dimana berfungsi untuk membuat tabel pada database kita menjadi sebuah Class dan kita bisa melakukan query dengan method yang sudah disediakan pada Class tersebut. Untuk membuat Model yang terasosiasi pada tabel kita cukup buat Model dengan konsep “singular” dan sesuaikan dengan nama tabel kita, cara nya cukup ketikan command berikut:

“php artisan make:model ProductCategory”

Maka akan muncul berikut:

```

sh-3.2$ php artisan make:model ProductCategory
[INFO] Model [app/Models/ProductCategory.php] created successfully.
sh-3.2$ 

```

Masuk kedalam folder “app” lalu masuk kedalam folder “Model” dan file Model yang baru kalian buat akan tersimpan di directory tersebut. Selanjutnya buka file tersebut dan isikan beberapa variable seperti pada gambar berikut:



```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class ProductCategory extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         "name",
14         "description"
15     ];
16 }
17

```

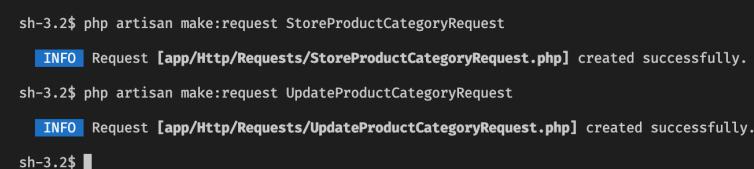
Pada variable \$fillable menandakan field tabel mana saja yang bisa kita gunakan untuk melakukan insert menggunakan Laravel Model ini. Kalian bisa membuat variable, method, dan menambahkan penggunaan Trait sesuai kebutuhan kalian dan kalian bisa melihat petunjuk lengkapnya pada dokumentasi resmi Laravel.

## E. Request

Laravel Request adalah fitur Laravel yang berfungsi untuk membuat validasi terhadap Request Body yang nantinya kita gunakan untuk melakukan CRUD pada API yang akan kita buat nantinya. Cara membuatnya cukup ketikkan command berikut:

“php artisan make:request {nama request kalian}”

Dalam kasus kali ini akan dicontohkan membuat 2 request yaitu untuk melakukan Create dan Update data, cara nya seperti pada gambar berikut:



```

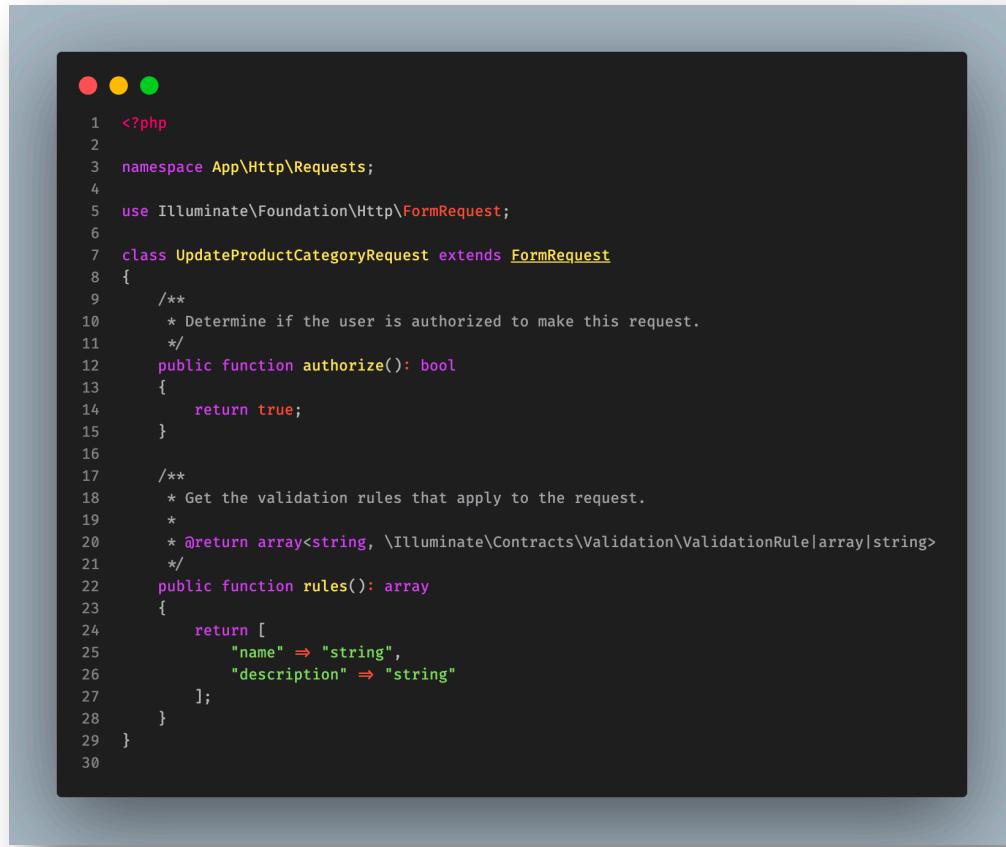
sh-3.2$ php artisan make:request StoreProductCategoryRequest
[INFO] Request [app/Http/Requests/StoreProductCategoryRequest.php] created successfully.
sh-3.2$ php artisan make:request UpdateProductCategoryRequest
[INFO] Request [app/Http/Requests/UpdateProductCategoryRequest.php] created successfully.
sh-3.2$ 

```

Pada file **app/Http/Request/StoreProductCategoryRequest.php**

```
1 <?php
2
3 namespace App\Http\Requests;
4
5 use Illuminate\Foundation\Http\FormRequest;
6
7 class StoreProductCategoryRequest extends FormRequest
8 {
9     /**
10      * Determine if the user is authorized to make this request.
11      */
12     public function authorize(): bool
13     {
14         return true;
15     }
16
17     /**
18      * Get the validation rules that apply to the request.
19      *
20      * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>
21      */
22     public function rules(): array
23     {
24         return [
25             "name" => "string|required",
26             "description" => "string|required"
27         ];
28     }
29 }
30
```

Pada file **app/Http/Request/UpdateProductCategoryRequest.php**



```

1 <?php
2
3 namespace App\Http\Requests;
4
5 use Illuminate\Foundation\Http\FormRequest;
6
7 class UpdateProductCategoryRequest extends FormRequest
8 {
9     /**
10      * Determine if the user is authorized to make this request.
11      */
12     public function authorize(): bool
13     {
14         return true;
15     }
16
17     /**
18      * Get the validation rules that apply to the request.
19      *
20      * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>
21      */
22     public function rules(): array
23     {
24         return [
25             "name" => "string",
26             "description" => "string"
27         ];
28     }
29 }
30

```

Pada bagian method authorize() pastikan return nya true dan pada bagian method rules() berisi aturan validasi yang kita buat pada Request tersebut.

#### F. Resource

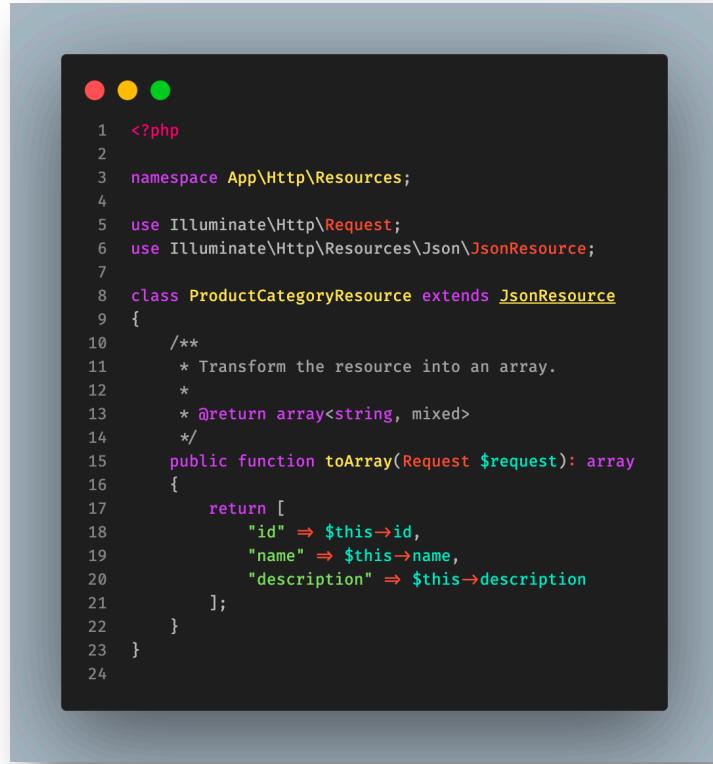
Laravel Resource merupakan fitur untuk melakukan formatting terhadap response dari API yang akan kita buat dalam hal ini kita akan menggunakan format JSON sebagai response dari API kita nantinya. Resource sendiri terdapat 2 tipe yaitu collection dan single resource, dimana collection adalah array of object dan single resource adalah object itu sendiri. Cara membuat single resource cukup ketikan command pada terminal seperti berikut:

“php artisan make:resource ProductCategoryResource”

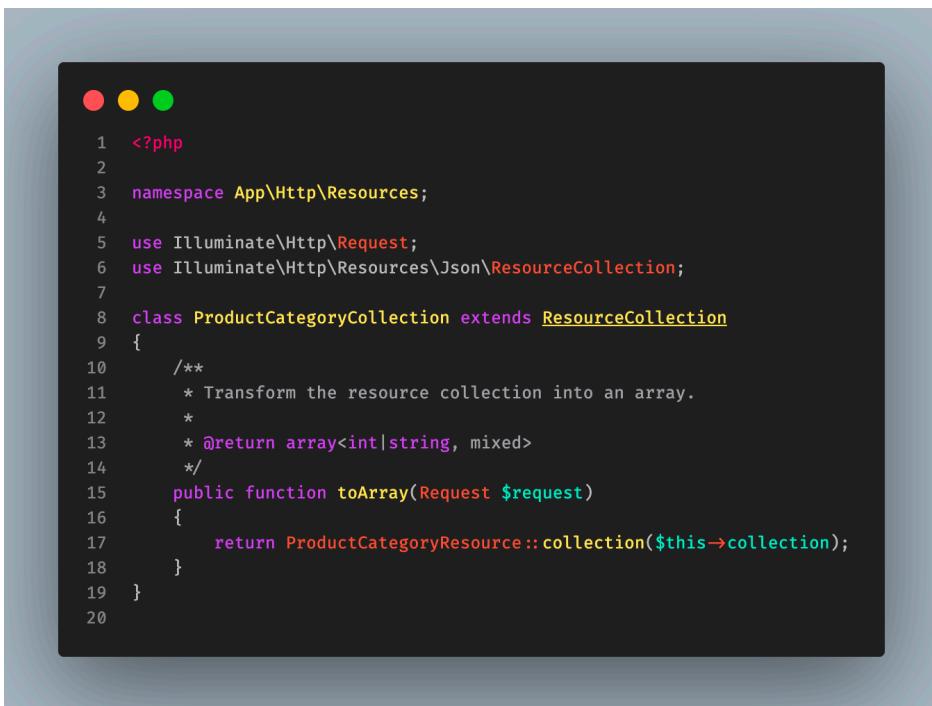
Sedangkan untuk collection resource cukup ketikkan command berikut:

“php artisan make:resource ProductCategoryCollection”

Setelah itu modifikasi kedua file resource yang kita buat seperti pada gambar berikut:

**Pada file app/Http/Resources/ProductCategoryResource.php**

```
1 <?php
2
3 namespace App\Http\Resources;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Http\Resources\Json\JsonResource;
7
8 class ProductCategoryResource extends JsonResource
9 {
10     /**
11      * Transform the resource into an array.
12      *
13      * @return array<string, mixed>
14      */
15     public function toArray(Request $request): array
16     {
17         return [
18             "id" => $this->id,
19             "name" => $this->name,
20             "description" => $this->description
21         ];
22     }
23 }
24
```

**Pada file app/Http/Resources/ProductCategoryCollection.php**

```
1 <?php
2
3 namespace App\Http\Resources;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Http\Resources\Json\ResourceCollection;
7
8 class ProductCategoryCollection extends ResourceCollection
9 {
10     /**
11      * Transform the resource collection into an array.
12      *
13      * @return array<int|string, mixed>
14      */
15     public function toArray(Request $request)
16     {
17         return ProductCategoryResource::collection($this->collection);
18     }
19 }
20
```

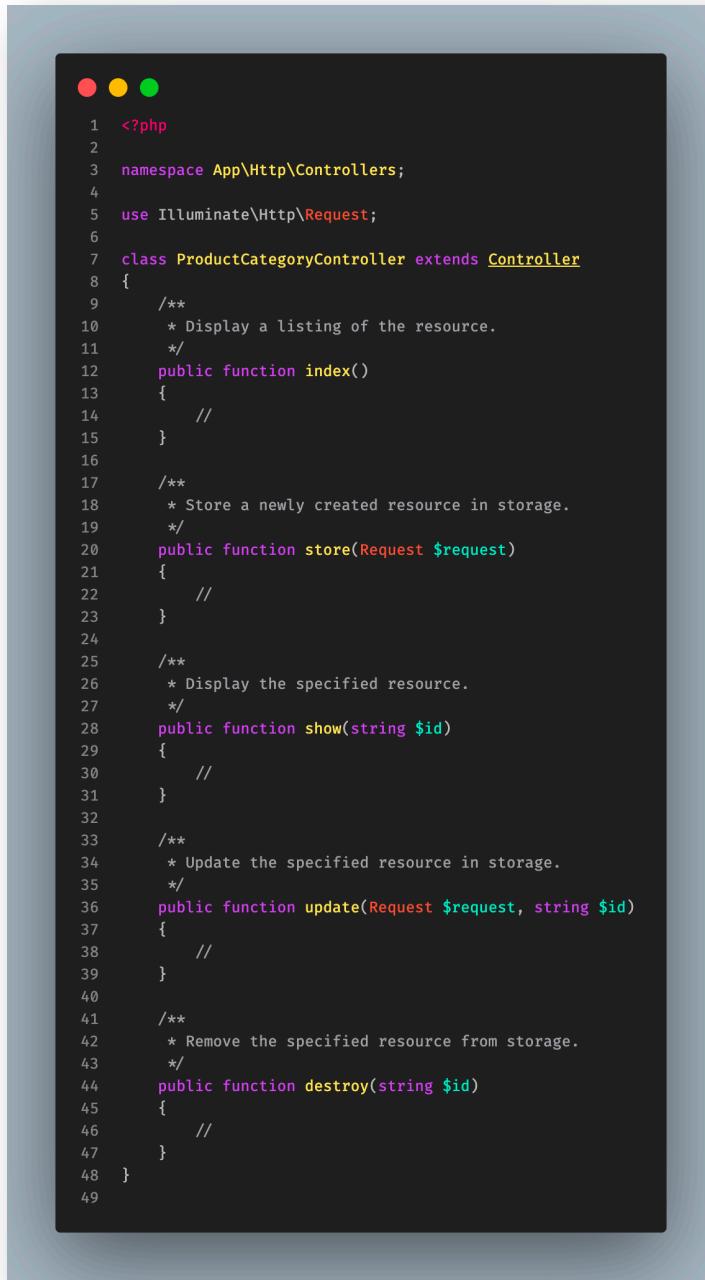
Kedua file resource ini nantinya kita gunakan untuk melakukan formatting JSON response API kita, dalam contoh kali ini kita akan menampilkan hanya field id, name dan description saja tanpa field created\_at dan updated\_at.

#### G. Controller

Pada modul sebelumnya kita sudah pernah membuat sebuah Controller dan pada framework Laravel juga terdapat Controller selayaknya arsitektur MVC pada umumnya. Pada kali ini kita akan membuat Laravel Controller khusus untuk pembuatan API dengan cara ketikkan command berikut pada terminal:

**“php artisan make:controller ProductCategoryController --api”**

Maka akan terbuat file pada **app/Http/Controller/ProductCategoryController.php** seperti pada gambar berikut:



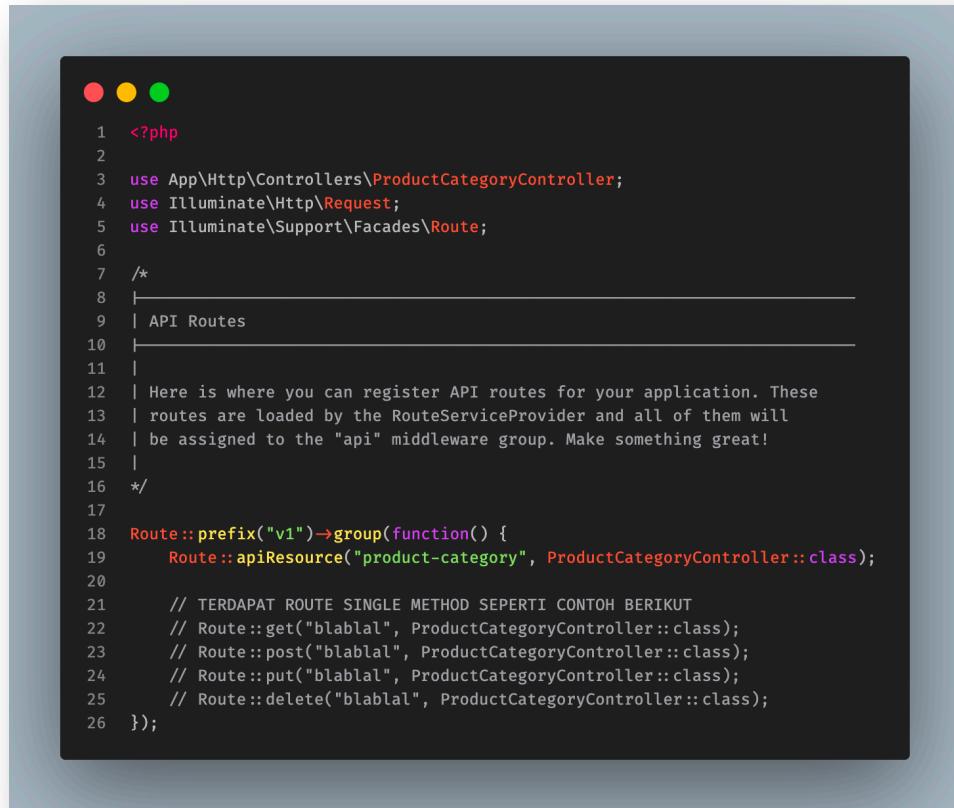
```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductCategoryController extends Controller
8 {
9     /**
10      * Display a listing of the resource.
11      */
12     public function index()
13     {
14         //
15     }
16
17     /**
18      * Store a newly created resource in storage.
19      */
20     public function store(Request $request)
21     {
22         //
23     }
24
25     /**
26      * Display the specified resource.
27      */
28     public function show(string $id)
29     {
30         //
31     }
32
33     /**
34      * Update the specified resource in storage.
35      */
36     public function update(Request $request, string $id)
37     {
38         //
39     }
40
41     /**
42      * Remove the specified resource from storage.
43      */
44     public function destroy(string $id)
45     {
46         //
47     }
48 }
49
```

Pada file tersebut sudah terdapat template Class beserta Method yang akan kita gunakan untuk membuat handler atau controller REST API yang akan kita buat.

## H. Routes

Laravel juga memiliki fitur routing untuk menentukan route api yang akan kita buat, route sendiri berguna untuk menentukan API dengan method tertentu seperti GET,

POST, PUT, PATCH, DELETE. Untuk routing khusus API terdapat pada file **routes/api.php**, selanjutnya silahkan masuk ke file tersebut dan modifikasi seperti pada gambar berikut:



```
1 <?php
2
3 use App\Http\Controllers\ProductCategoryController;
4 use Illuminate\Http\Request;
5 use Illuminate\Support\Facades\Route;
6
7 /*
8 | API Routes
9 |
10 */
11
12 | Here is where you can register API routes for your application. These
13 | routes are loaded by the RouteServiceProvider and all of them will
14 | be assigned to the "api" middleware group. Make something great!
15 |
16 */
17
18 Route::prefix("v1")->group(function() {
19     Route::apiResource("product-category", ProductCategoryController::class);
20
21     // TERDAPAT ROUTE SINGLE METHOD SEPERTI CONTOH BERIKUT
22     // Route::get("blablab", ProductCategoryController::class);
23     // Route::post("blablab", ProductCategoryController::class);
24     // Route::put("blablab", ProductCategoryController::class);
25     // Route::delete("blablab", ProductCategoryController::class);
26 });

```

Dalam file tersebut terdapat prefix “v1” untuk versioning dan didalamnya terdapat Route::apiResource() yang berarti route tersebut sudah include semua method GET, POST, PUT , PATCH, DELETE tanpa harus mendefinisikan single route satu persatu. Untuk mengecek route kalian sudah terdaftar atau tidak cukup ketikkan command berikut pada terminal:

“php artisan route:list”

Maka akan muncul seperti berikut:

```
sh-3.2$ php artisan route:list
GET|HEAD / ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
POST _ignition/execute-solution ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET|HEAD _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD api/v1/product-category ..... product-category.index > ProductCategoryController@index
POST api/v1/product-category ..... product-category.store > ProductCategoryController@store
PUT|PATCH api/v1/product-category/{product_category} ..... product-category.update > ProductCategoryController@update
DELETE api/v1/product-category/{product_category} ..... product-category.destroy > ProductCategoryController@destroy
GET|HEAD sanctum/csrf-cookie ..... sanctum.csrf-cookie > Laravel\Sanctum > CsrfCookieController@show

Showing [10] routes
sh-3.2$
```

Terlihat dalam gambar route untuk “product-category” kita sudah terdaftar dengan semua method GET, POST, PUT, PATCH, DELETE.

### I. Laravel Server

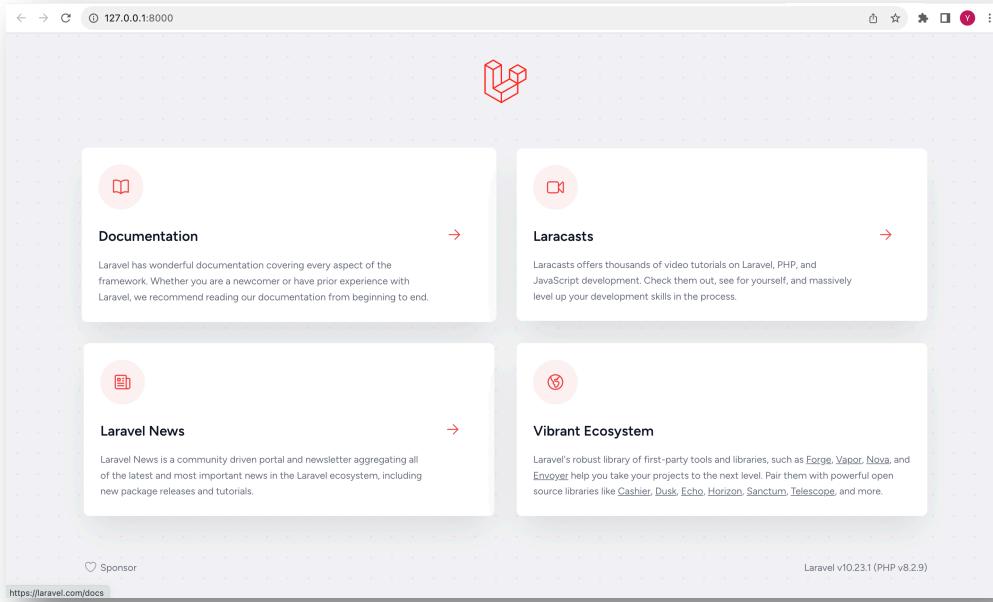
Pada modul sebelumnya kita menjalankan PHP Server menggunakan command “php -S localhost:8000 index.php”, maka dalam Laravel kita bisa menggunakan artisan untuk menjalankan server kita dengan cara mengetikkan command berikut pada terminal:

“php artisan serve”

Maka akan muncul seperti berikut:

```
sh-3.2$ php artisan serve
[INFO] Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
|
```

Jika kalian buka <http://127.0.0.1:8000> pada browser kalian maka akan muncul halaman berikut:



Selamat anda berhasil menjalankan Laravel Server pada project Laravel kalian.

## LATIHAN

Setelah melakukan kegiatan konfigurasi diatas, mari kita membuat REST API dengan konfigurasi yang telah kita buat.

**Rincian endpoint yang kita buat adalah sebagai berikut:**

<b>GET</b>	http://127.0.0.1:8000/api/product-category
<b>GET</b>	http://127.0.0.1:8000/api/product-category/{id}
<b>POST</b>	http://127.0.0.1:8000/api/product-category
<b>PUT</b>	http://127.0.0.1:8000/api/product-category
<b>DELETE</b>	http://127.0.0.1:8000/api/product-category

Buka file ProductCategoryController.php yang telah kita buat sebelumnya dan ikuti contoh berikut:

Pada file **app/Http/Controller/ProductCategoryController.php**



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Http\Requests\StoreProductCategoryRequest;
6 use App\Http\Requests\UpdateProductCategoryRequest;
7 use App\Http\Resources\ProductCategoryCollection;
8 use App\Http\Resources\ProductCategoryResource;
9 use App\Models\ProductCategory;
10 use Exception;
11 use Illuminate\Http\Request;
12
13 class ProductCategoryController extends Controller
14 {
15     /**
16      * Display a listing of the resource.
17      */
18     public function index()
19     {
20         try {
21
22             $queryData = ProductCategory::all();
23             $formattedDatas = new ProductCategoryCollection($queryData);
24             return response()->json([
25                 "message" => "success",
26                 "data" =>$formattedDatas
27             ], 200);
28         } catch (Exception $e) {
29             return response()->json($e->getMessage(), 400);
30         }
31     }
32 }
```

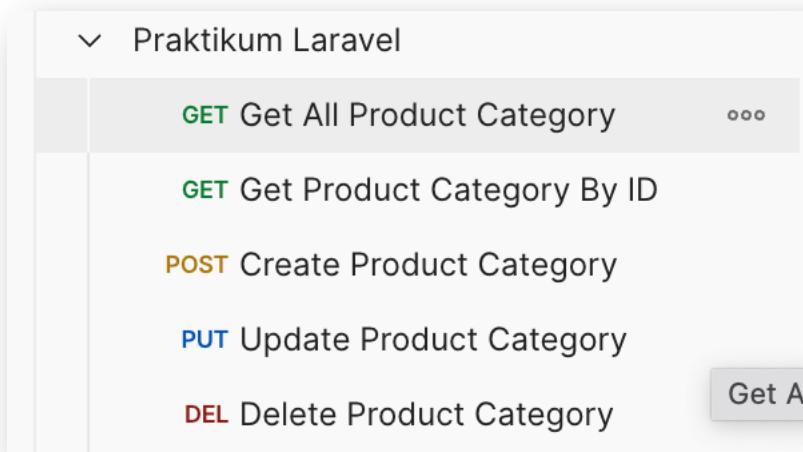
```
1  /**
2   * Store a newly created resource in storage.
3   */
4  public function store(StoreProductCategoryRequest $request)
5  {
6      $validatedRequest = $request→validated();
7      try {
8          $queryData = ProductCategory::create($validatedRequest);
9          $formattedDatas = new ProductCategoryResource($queryData);
10         return response()→json([
11             "message" ⇒ "success",
12             "data" ⇒ $formattedDatas
13         ], 200);
14     } catch (Exception $e) {
15         return response()→json($e→getMessage(), 400);
16     }
17 }
18
19 /**
20  * Display the specified resource.
21 */
22 public function show(string $id)
23 {
24     try {
25         $queryData = ProductCategory::findOrFail($id);
26         $formattedDatas = new ProductCategoryResource($queryData);
27         return response()→json([
28             "message" ⇒ "success",
29             "data" ⇒ $formattedDatas
30         ], 200);
31     } catch (Exception $e) {
32         return response()→json($e→getMessage(), 400);
33     }
34 }
35
```

```
1      /**
2       * Update the specified resource in storage.
3       */
4      public function update(UpdateProductCategoryRequest $request, string $id)
5      {
6          $validatedRequest = $request->validated();
7          try {
8              $queryData = ProductCategory::findOrFail($id);
9              $queryData->update($validatedRequest);
10             $queryData->save();
11             $formattedDatas = new ProductCategoryResource($queryData);
12             return response()->json([
13                 "message" => "success",
14                 "data" =>$formattedDatas
15             ], 200);
16         } catch (Exception $e) {
17             return response()->json($e->getMessage(), 400);
18         }
19     }
20 }
21 /**
22  * Remove the specified resource from storage.
23  */
24 public function destroy(string $id)
25 {
26     try {
27         $queryData = ProductCategory::findOrFail($id);
28         $queryData->delete();
29         $formattedDatas = new ProductCategoryResource($queryData);
30         return response()->json([
31             "message" => "success",
32             "data" =>$formattedDatas
33         ], 200);
34     } catch (Exception $e) {
35         return response()->json($e->getMessage(), 400);
36     }
37 }
38 }
39 }
40 }
```

Setelah itu jalankan Laravel Servernya dengan command pada terminal:

**“php artisan serve”**

Setelah berhasil buka aplikasi Postman dan lakukan testing pada REST API yang kita buat seperti yang kita lakukan pada modul 5 sebelumnya, dan pastikan sesuai dengan rincian endpoint yang kita tentukan:



---

## TUGAS

### TUGAS 1

Buatlah REST API CRUD dengan tema bebas sesuai kreatifitas kalian, REST API tersebut harus memuat method GET, POST, PUT, DELETE dan dilarang menggunakan kode yang sama seperti Codelab!

### TUGAS 2 (OPTIONAL)

Buatlah REST API Authentication Login dengan method POST yang menerima inputan email dan password. Return dari Endpoint tersebut berupa Access Token. Kalian bisa membuat menggunakan package seperti JWT ataupun yang lainnya, namun direkomendasikan menggunakan fitur Laravel Sanctum yang bisa kalian lihat tutorialnya pada dokumentasi resmi Laravel:

<https://laravel.com/docs/10.x/sanctum>

Contoh Response dari API jika menggunakan Laravel Sanctum:

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8000/api/v1/login
- Body:** JSON (containing email and password)
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```

1 {
2   "message": "success",
3   "data": {
4     "access_token": "1|Dh2QH5yhjRGQ2CF1MTNhVmfvFLpAwL2x1Kgasz5ncd7c2b05"
5   }
6 }
```

#### NOTES:

Asisten akan mengecek pemahaman anda mengenai materi pada modul dan konsep dari REST API serta penerapannya menggunakan framework Laravel.

#### KRITERIA & DETAIL PENILAIAN

Kriteria	Presentase Penilaian
Dapat membuat REST API dengan Framework Laravel	60%
Dapat membuat Login Authentication API	20%
Dapat menguji REST API yang telah dibuat dengan Postman	20%