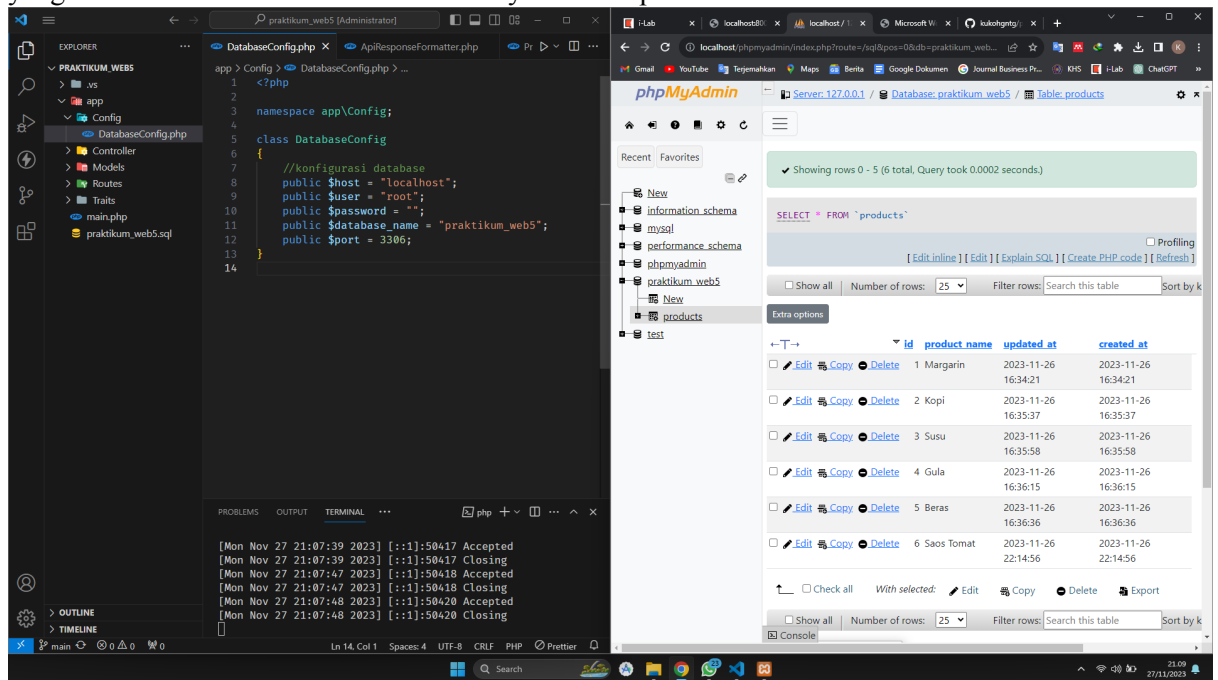


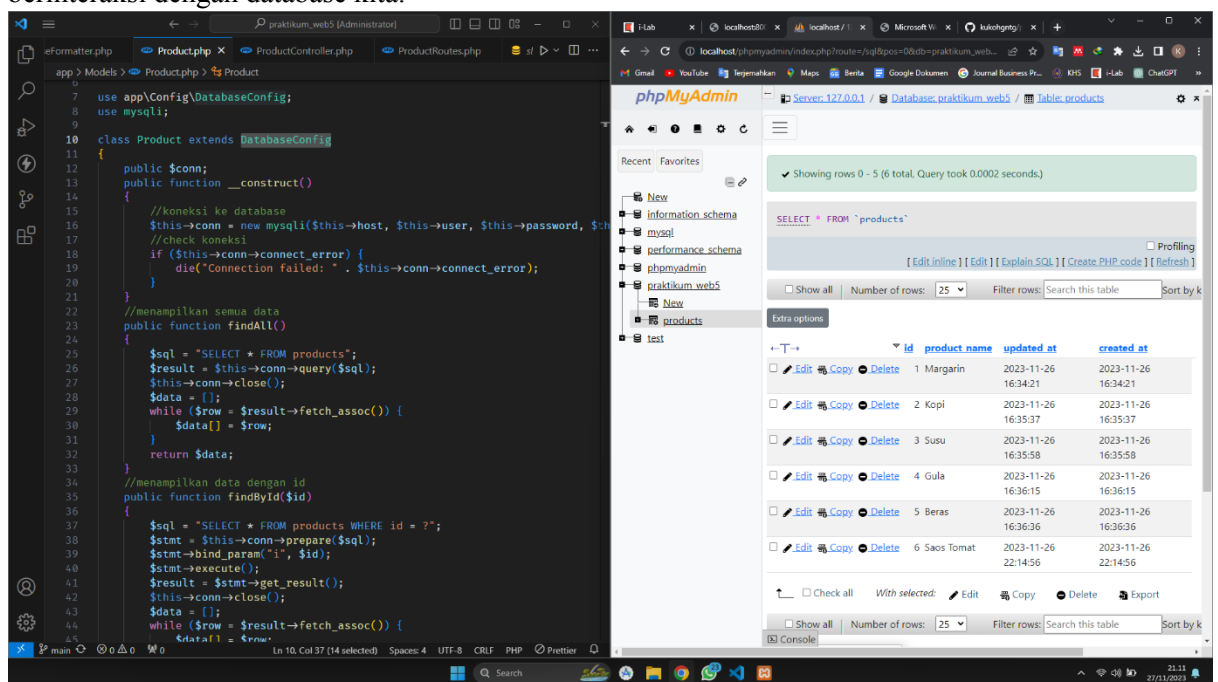
Nama : Hafizh Rafi Ihatra
NIM : 202110370311009
Kelas : Pemrograman Web B

Tugas Codelab

1. Dari gambar berikut Database Config digunakan untuk menyambungkan kita ke database yang telah di buat di sini nama database saya adalah “praktikum web 5”.



2. Dari gambar berikut Product bisa dikatakan adalah sebuah Model yang nantinya kita akan berinteraksi dengan database kita.



3. Dari gambar berikut Product Controller berfungsi untuk memproses permintaan yang dilakukan melalui API kita.

The screenshot displays two windows. The left window shows the `ProductController.php` file in a code editor. The code defines three methods: `index()`, `getById($id)`, and `insert()`. The `index()` method uses `ApiResponseFormatter` to format the response. The `getById($id)` method finds a product by ID and returns it. The `insert()` method takes JSON input, validates it, and inserts a new product into the database.

```
//menggunakan trait
use ApiResponseFormatter;

public function index()
{
    //object model product yang sudah di buat
    $productModel = new Product();
    //memanggil fungsi get all product
    $response = $productModel->findAll();
    //mengembalikan respon dengan memformat terlebih dahulu
    return $this->apiResponse(200, "success", $response);
}

public function getById($id)
{
    $productModel = new Product();
    $response = $productModel->findById($id);
    return $this->apiResponse(200, "success", $response);
}

public function insert()
{
    //mendapatkan input json
    $jsonData = file_get_contents('php://input');
    $inputData = json_decode($jsonData, true);
    //validasi input
    if (json_last_error()) {
        return $this->apiResponse(400, "error invalid input", null);
    }
    //jika input benar
```

The right window shows the phpMyAdmin interface for the `praktikum_web5` database. The `products` table is selected, showing 6 rows. The table structure is as follows:

id	product name	updated at	created at
1	Margarin	2023-11-26 16:34:21	2023-11-26 16:34:21
2	Kopi	2023-11-26 16:35:37	2023-11-26 16:35:37
3	Susu	2023-11-26 16:35:58	2023-11-26 16:35:58
4	Gula	2023-11-26 16:36:15	2023-11-26 16:36:15
5	Beras	2023-11-26 16:36:36	2023-11-26 16:36:36
6	Saos Tomat	2023-11-26 22:14:56	2023-11-26 22:14:56

4. Dari gambar berikut ProductRoutes digunakan untuk mendefiniskan alamat route API.

The screenshot displays two windows. The left window shows the `ProductRoutes.php` file in a code editor. The code defines a `handle($method, $path)` method that routes requests to the `ProductController` based on the HTTP method and path. It handles GET requests for the root path and specific product IDs, POST requests for inserting new products, PUT requests for updating products, and DELETE requests for deleting products.

```
include "app/Controller/ProductController.php";
use app\Controller\ProductController;

class ProductRoutes
{
    public function handle($method, $path)
    {
        //jika request method get dan path sama dengan '/api/product/'
        if ($method === 'GET' && $path === '/api/product/') {
            $controller = new ProductController();
            echo $controller->index();
        }

        //jika request method get dan path mengandung '/api/product/'
        if ($method === 'GET' && strpos($path, '/api/product/') === 0) {
            //extract id dari path
            $pathParts = explode('/', $path);
            $id = $pathParts[count($pathParts) - 1];

            $controller = new ProductController();
            echo $controller->getById($id);
        }

        //jika request method post dan path sama dengan '/api/product/'
        if ($method === 'POST' && $path === '/api/product/') {
            $controller = new ProductController();
            echo $controller->insert();
        }

        //jika request method put dan path mengandung '/api/product/'
        if ($method === 'PUT' && strpos($path, '/api/product/') === 0) {
            //extract id dari path
            $pathParts = explode('/', $path);
            $id = $pathParts[count($pathParts) - 1];

            $controller = new ProductController();
            echo $controller->update($id);
        }

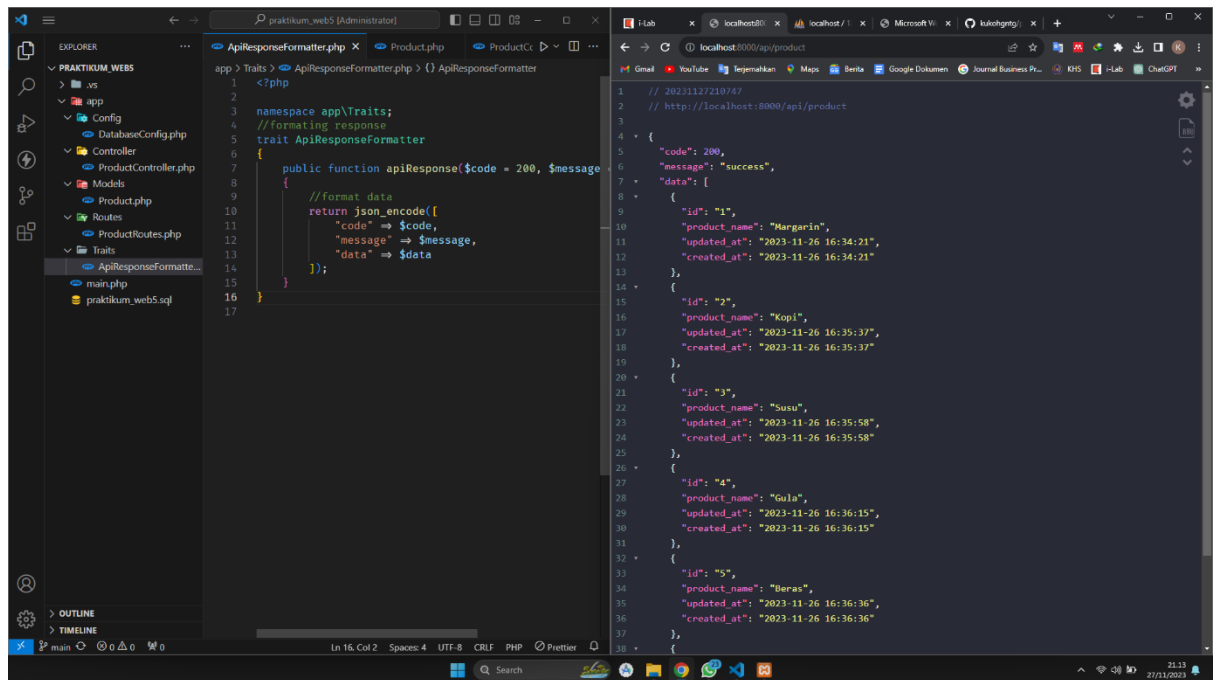
        //jika request method delete dan path mengandung '/api/product/'
        if ($method === 'DELETE' && strpos($path, '/api/product/') === 0) {

```

The right window shows the phpMyAdmin interface for the `praktikum_web5` database. The `products` table is selected, showing 6 rows. The table structure is as follows:

id	product name	updated at	created at
1	Margarin	2023-11-26 16:34:21	2023-11-26 16:34:21
2	Kopi	2023-11-26 16:35:37	2023-11-26 16:35:37
3	Susu	2023-11-26 16:35:58	2023-11-26 16:35:58
4	Gula	2023-11-26 16:36:15	2023-11-26 16:36:15
5	Beras	2023-11-26 16:36:36	2023-11-26 16:36:36
6	Saos Tomat	2023-11-26 22:14:56	2023-11-26 22:14:56

5. Dari gambar berikut ApiResponseFormatter digunakan untuk menformat data kita menjadi JSON.



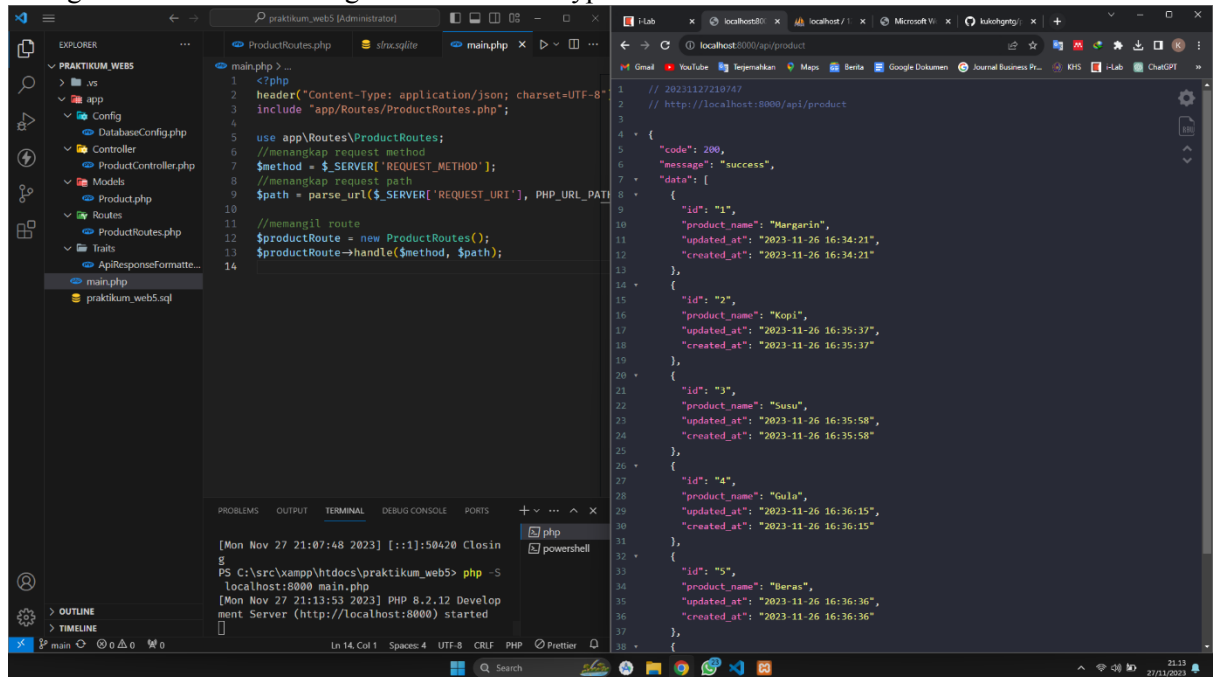
The screenshot shows the Visual Studio Code editor with the file `ApiResponseFormatter.php` open. The code defines a trait `ApiResponseFormatter` with a `ApiResponse` method that formats data into a JSON response. The response structure includes a `code`, a `message`, and a `data` array. The `data` array contains five product objects, each with `id`, `product_name`, `updated_at`, and `created_at` fields.

```
1 <?php
2
3 namespace app\Traits;
4 //formatting response
5 trait ApiResponseFormatter
6 {
7     public function ApiResponse($code = 200, $message
8     {
9         //format data
10        return json_encode([
11            "code" => $code,
12            "message" => $message,
13            "data" => $data
14        ]);
15    }
16 }
17
```

The browser view on the right shows the API response at `localhost:8000/api/product`. The response is a JSON array of five product objects:

```
1 // 20231127210747
2 // http://localhost:8000/api/product
3
4 {
5   "code": 200,
6   "message": "success",
7   "data": [
8     {
9       "id": "1",
10      "product_name": "Margarin",
11      "updated_at": "2023-11-26 16:34:21",
12      "created_at": "2023-11-26 16:34:21"
13    },
14    {
15      "id": "2",
16      "product_name": "Kopi",
17      "updated_at": "2023-11-26 16:35:37",
18      "created_at": "2023-11-26 16:35:37"
19    },
20    {
21      "id": "3",
22      "product_name": "Susu",
23      "updated_at": "2023-11-26 16:35:58",
24      "created_at": "2023-11-26 16:35:58"
25    },
26    {
27      "id": "4",
28      "product_name": "Gula",
29      "updated_at": "2023-11-26 16:36:15",
30      "created_at": "2023-11-26 16:36:15"
31    },
32    {
33      "id": "5",
34      "product_name": "Beras",
35      "updated_at": "2023-11-26 16:36:36",
36      "created_at": "2023-11-26 16:36:36"
37    }
38  ]
39 }
```

6. Dari gambar berikut Main digunakan untuk entypoint.



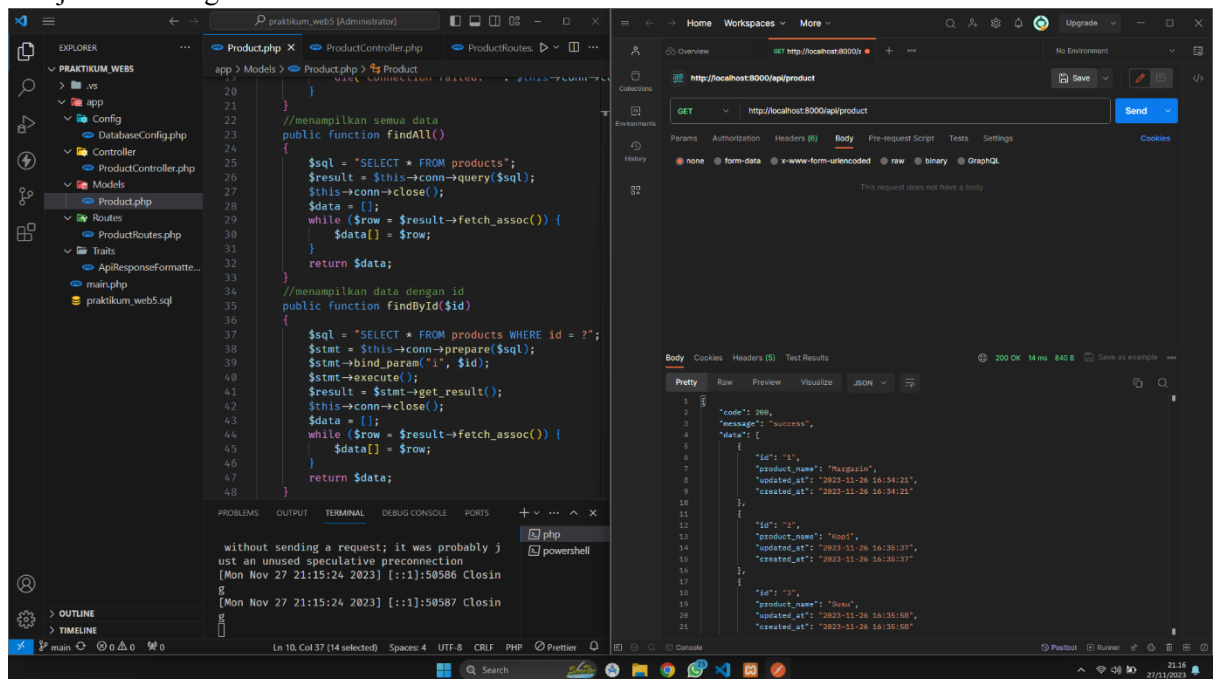
The screenshot shows the Visual Studio Code editor with the file `main.php` open. The code sets up the application header, includes the routes, and defines a `main` function that handles the request. The `main` function uses the `ApiResponseFormatter` trait to format the response.

```
1 <?php
2 header("Content-Type: application/json; charset=UTF-8");
3 include "app/Routes/ProductRoutes.php";
4
5 use app\Routes\ProductRoutes;
6 //menangkap request method
7 $method = $_SERVER['REQUEST_METHOD'];
8 //menangkap request path
9 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
10
11 //memanggil route
12 $productRoute = new ProductRoutes();
13 $productRoute->handle($method, $path);
14
```

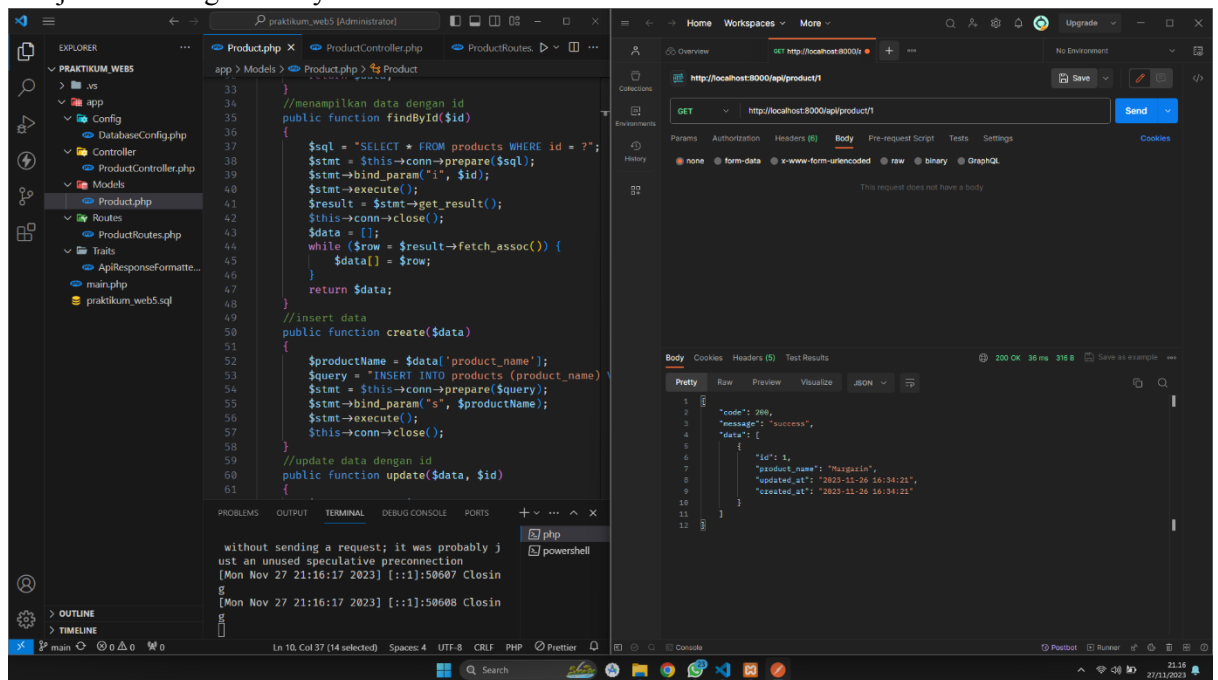
The terminal window at the bottom shows the command to run the application:

```
[Mon Nov 27 21:07:48 2023] [::1]:50420 Closes
PS C:\src\xampp\htdocs\praktikum_web5> php -S localhost:8000 main.php
[Mon Nov 27 21:13:53 2023] PHP 8.2.12 Development Server (http://localhost:8000) started
```

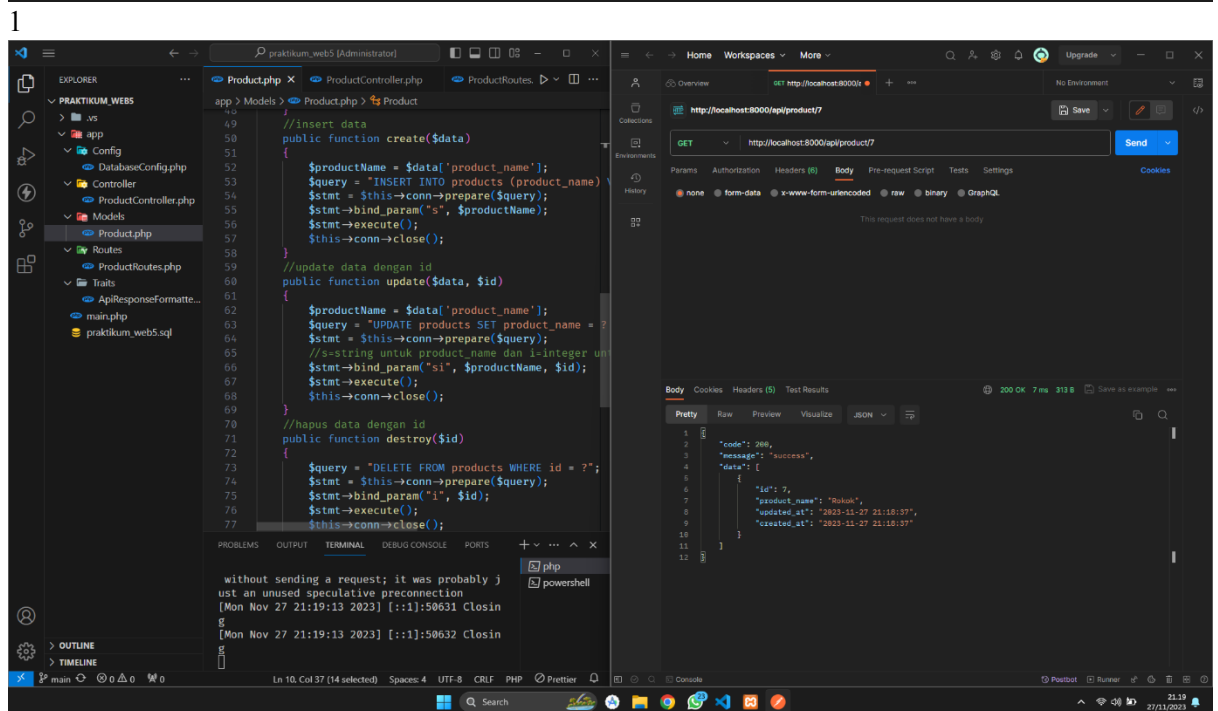
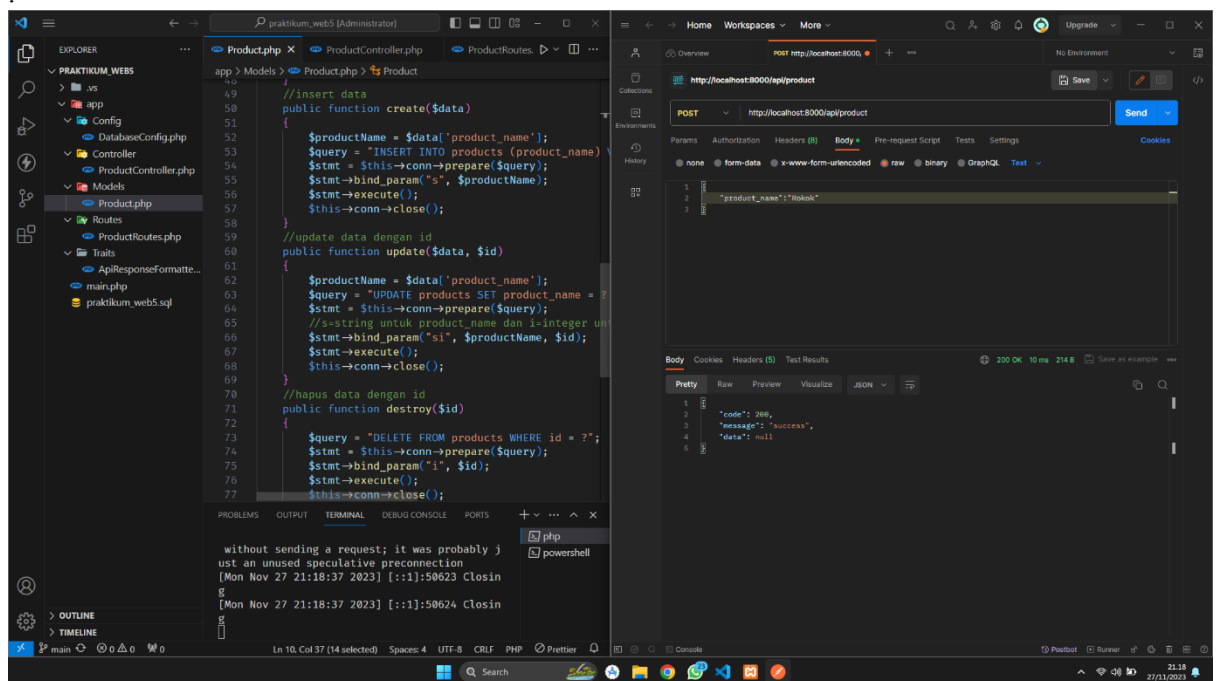
7. Dari gambar berikut GET Digunakan untuk mengakses data atau membaca data yang ada pada resource, di Postman mengakses url <http://localhost:8000/api/product> yang artinya menjalankan fungsi findAll.



8. Dari gambar berikut GET Digunakan untuk mengakses data atau membaca data yang ada pada resource, di Postman mengakses url <http://localhost:8000/api/product/1> yang artinya menjalankan fungsi findById.

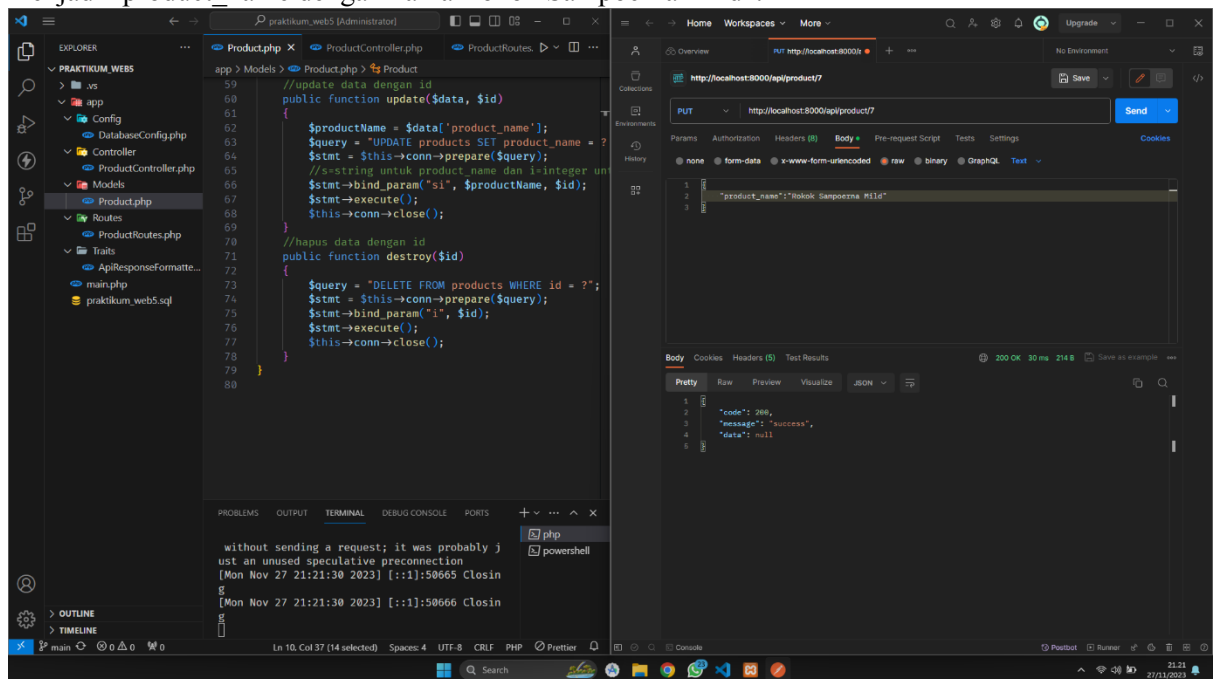


9. Dari kedua gambar berikut POST Digunakan untuk men-create atau membuat sebuah resource baru, di Postman mengakses url <http://localhost:8000/api/product> yang artinya menjalankan fungsi create untuk membuat data baru di “product_name dengan nama Rokok”

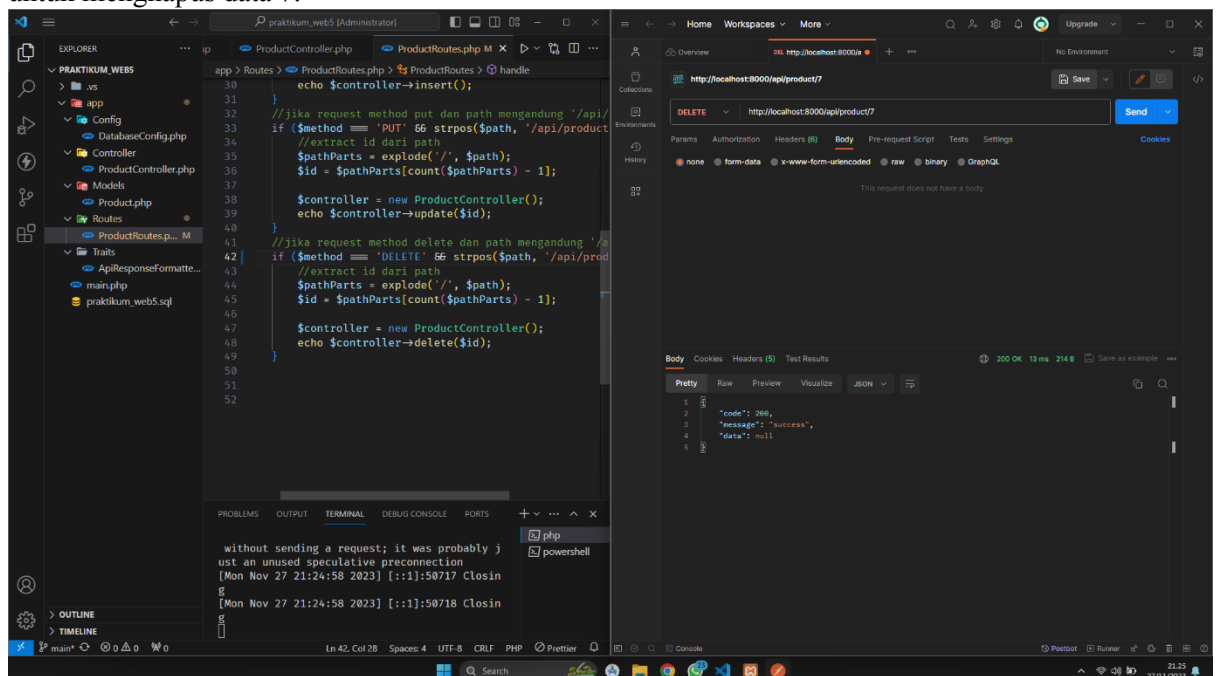


2

10. Dari gambar berikut PUT Digunakan untuk memperbaharui sebuah resource, atau menambahnya, di Postman mengakses url <http://localhost:8000/api/product/7> yang artinya menjalankan fungsi update untuk mengubah data 7 dari “product_name dengan nama Rokok” menjadi “product_name dengan nama Rokok Sampoerna Mild”.



11. Dari gambar berikut DELETE Digunakan untuk menghapus resource, di Postman mengakses url <http://localhost:8000/api/product/7> yang artinya menjalankan fungsi destroy untuk menghapus data 7.



12. Dari gambar berikut GET Digunakan untuk mengakses data atau membaca data yang ada pada resource, di Postman mengakses url <http://localhost:8000/api/product> yang artinya menjalankan fungsi findAll untuk mengetahui data 7 sudah terhapus.

