VERSION 1.2

10 November 2024

[PEMROGRAMAN WEBSITE]

MODUL 4 - PHP REST API DAN MYSQL DATABASE

DISUSUN OLEH:

OGYA ADYATMA PUTRA

GERALDI NATHAN TOMMY SAPUTRA

DIAUDIT OLEH:

AMINUDIN, S.KOM., M.CS.

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[PEMROGRAMAN WEBSITE]

PERSIAPAN MATERI

- https://www.w3schools.com/php/php_oop_what_is.asp
- https://www.w3schools.com/php/default.asp

TUJUAN

Mahasiswa mampu menggunakan MySQL dan melakukan komunikasi database dengan PHP

TARGET MODUL

Mahasiswa mampu memahami serta menerapkan penggunaan MySQL dan melakukan komunikasi database dengan PHP

PERSIAPAN SOFTWARE/APLIKASI

Hardware dan Infrastruktur

- Laptop/PC
- Koneksi Internet

Software

- Text Editor Visual Studio Code (Recommended)

Extension (VS Code)

- PHP Intelephense
- Prettier - Code Formatter | Prettier

Programming Language

- PHP

MATERI

MySQL

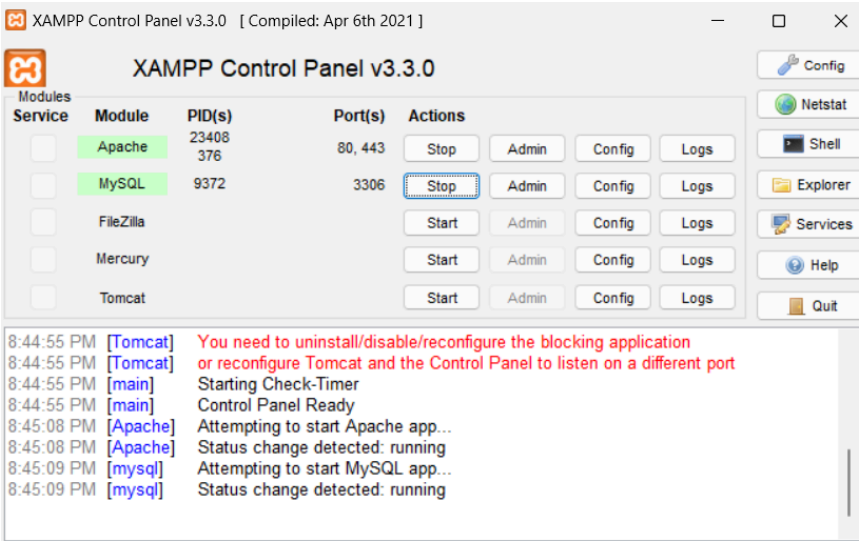
MySQL merupakan sistem manajemen database yang bersifat open source dengan menggunakan perintah dasar atau bahasa pemrograman yang berupa structured query language (SQL) yang cukup populer di dunia teknologi. Dibangun dengan model klien-server, MySQL memungkinkan pengguna untuk menyimpan, mengelola dan mengambil data secara efisien dalam struktur yang terorganisir. Beberapa operasi dasar MySQL yang biasa digunakan dan pernah dipelajari pada mata kuliah basis data.

Operasi Dasar MySQL

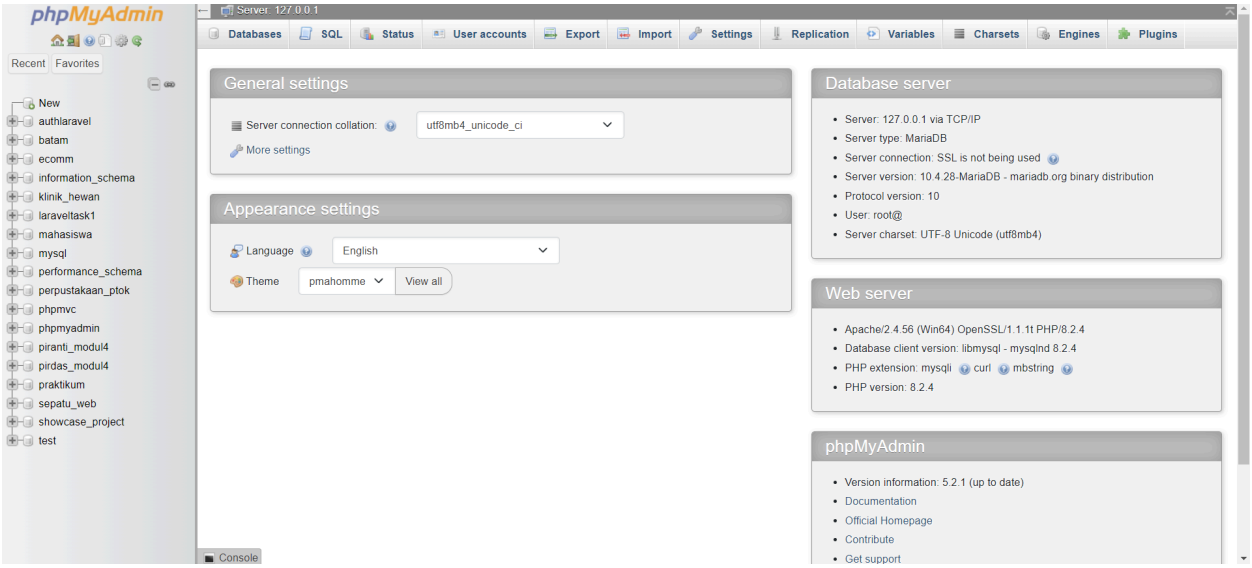
Data Definition Language (DDL)	Data Manipulation Language (DML)	Data Control Language (DCL)
CREATE : membuat database atau tabel baru **ALTER** : Mengubah struktur tabel **DROP** : Menghapus database atau tabel	**INSERT** : Menambahkan data baru ke tabel **SELECT** : Mengambil data dari tabel **UPDATE** : Memperbarui data yang ada pada tabel **DELETE** : Menghapus data dari tabel	**GRANT** : Memberikan izin akses ke pengguna **REVOKE** : Mencabut izin akses ke pengguna

A. Membuat Database dan Tabel dengan phpMyAdmin

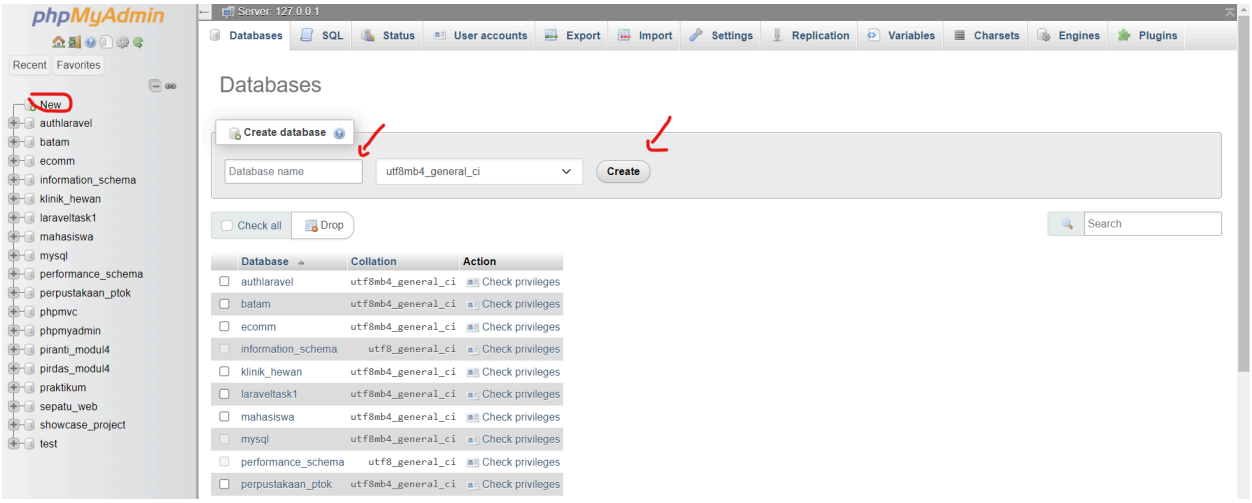
Praktikan harus menginstall MySQL pada masing-masing device dan jalankan MySQL, pada contoh kali ini akan menggunakan XAMPP tetapi jika praktikan memakai tools Mysql yang lain juga diperbolehkan.



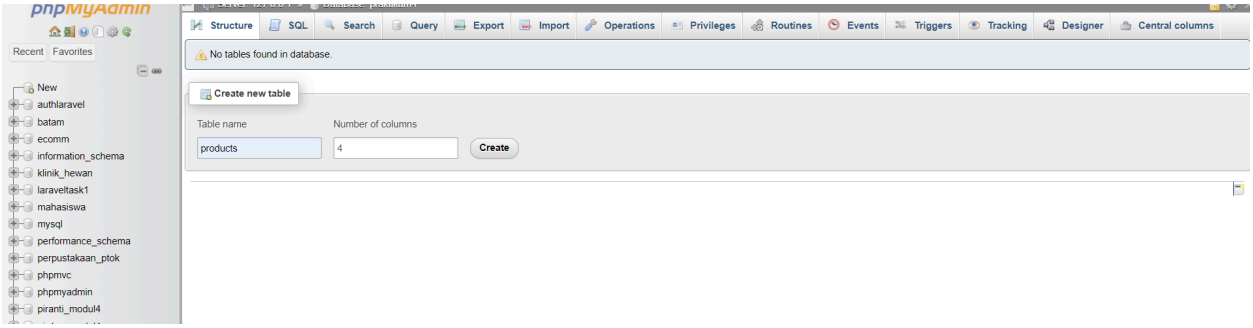
Klik tombol “admin” untuk menuju Web PHP MyAdmin seperti berikut:



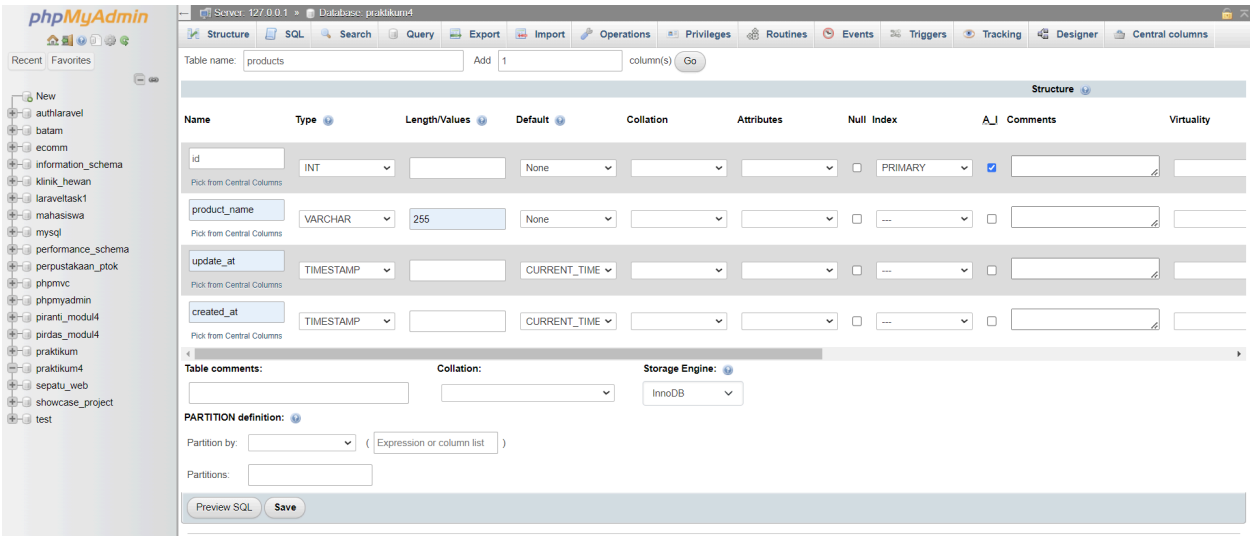
Selanjutnya silahkan membuat database baru secara manual mengikuti arahan gambar dibawah ini



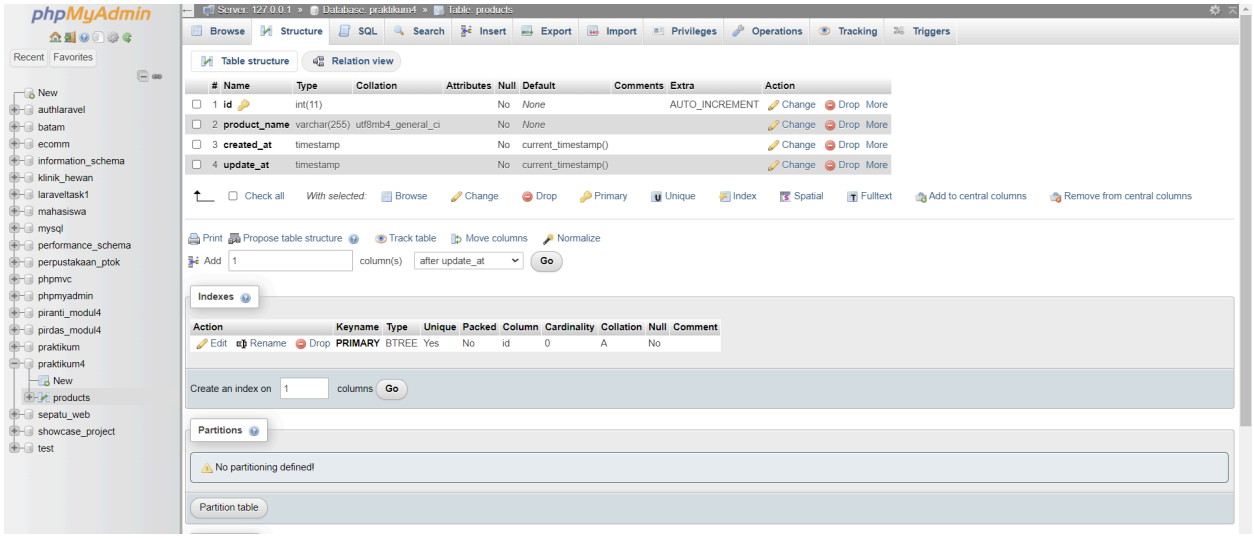
Mengacu pada gambar diatas silahkan klik “New” pada sidebar kiri lalu pada halaman database, silahkan menginputkan nama database kemudian klik tombol “Create” sesuai dengan arahan. Setelah membuat database, langkah selanjutnya yaitu membuat sebuah tabel seperti gambar dibawah.



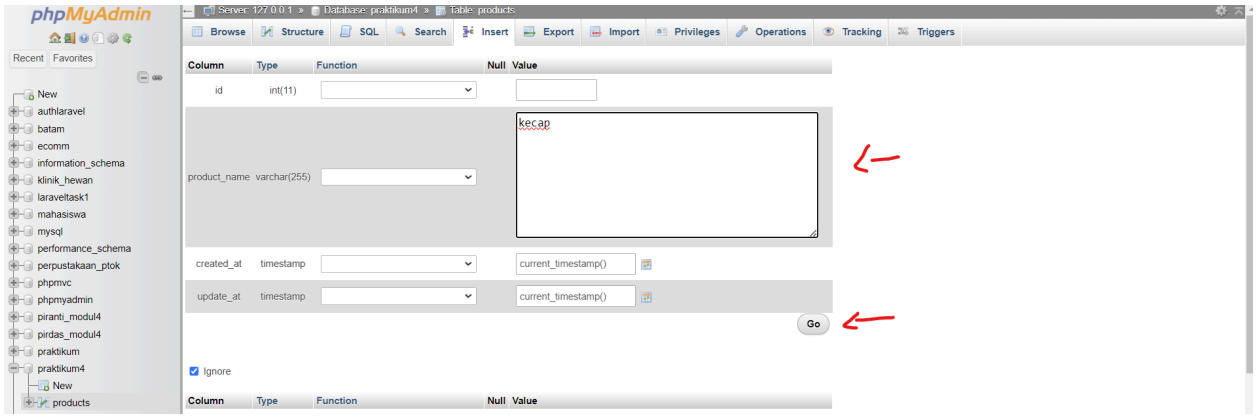
Setelah membuat tabel maka akan muncul tabel database yang baru silahkan buat pada sidebar kiri, lalu pada menu “Structure” kemudian silahkan membuat table dengan nama “products” dengan jumlah column 4 lalu klik “Create” maka akan muncul menu berikut



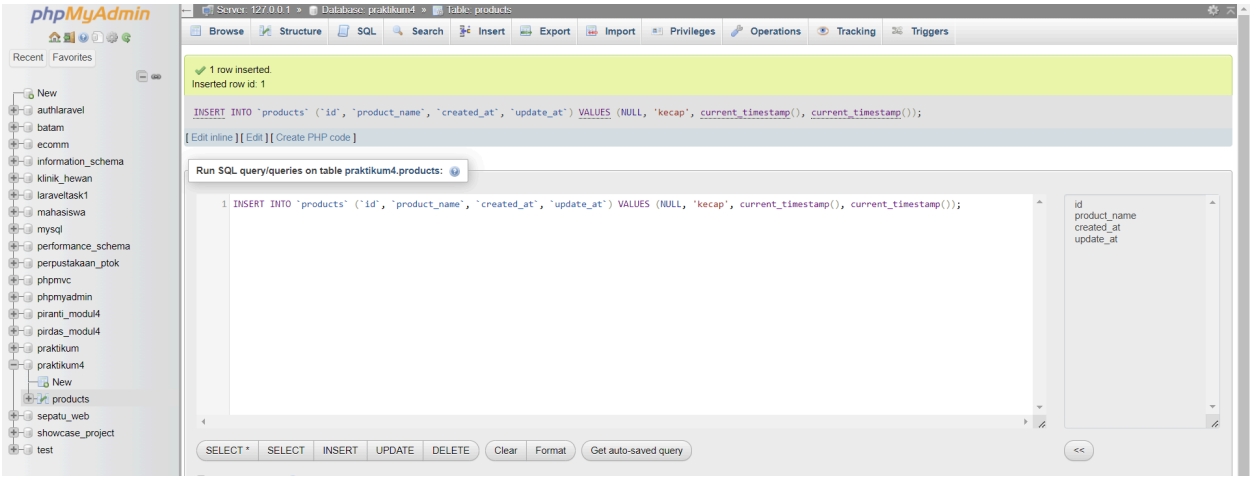
Seperti yang bisa dilihat bahwa terdapat 4 column dengan field “id”, “product_name”, “update_at”, dan “created_at”. Untuk colum “id” pastikan untuk mencentang pada bagian A_I yang menandakan field tersebut Auto Increment, lalu setting Type untuk tipe datanya dan default value untuk field “update_at” dan “created_at”, setelah itu silahkan klik save.



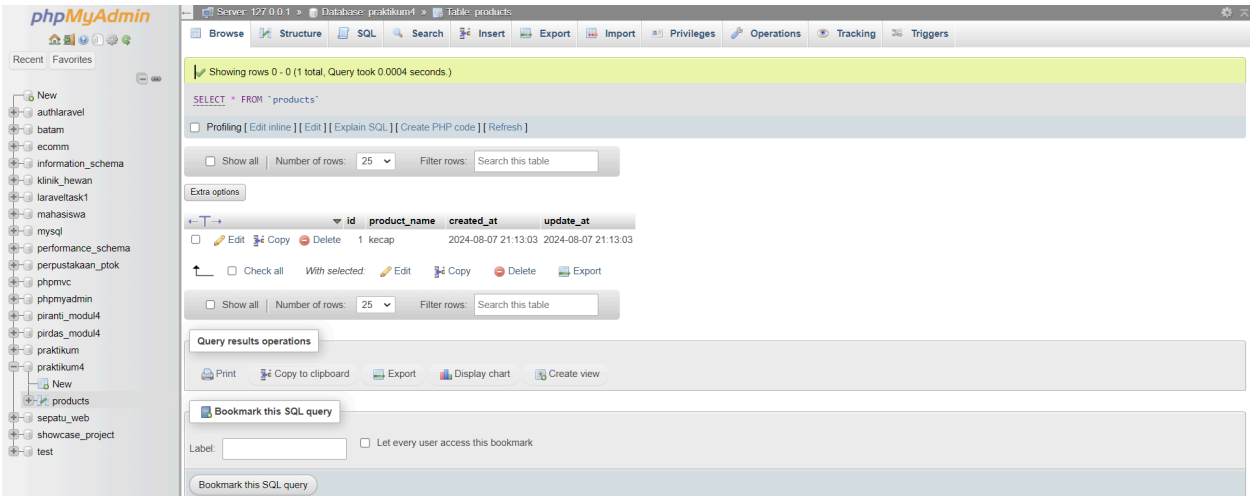
Setelah itu untuk memastikan tabel dan field telah terbuat, dapat diklik tabel products pada sidebar kiri kemudian masuk ke menu "Structure" seperti gambar dibawah ini.



Kemudian silahkan melakukan insert data manual dengan mengisikan field "product_name saja", kemudian di save setelah itu akan muncul pesan seperti berikut



Setelah itu klik menu “Browse” yang akan menampilkan seluruh data yang sudah dibuat seperti gambar berikut:



Ulangi langkah tersebut sehingga dapat menambahkan data minimal 5 kali. Dari mengikuti step sebelumnya sudah berhasil membuat database yang siap untuk diintegrasikan melalui PHP, selanjutnya kita akan membuat Backend Web Service yang nantinya akan menghasilkan sebuah API yang akan menyajikan data dari database kita dengan format JSON, namun sebelumnya kita akan mempelajari konsep RESTful API terlebih dahulu dan menginstall Postman sebagai tools untuk testing RESTful API kita.

HTTP DAN POSTMAN

A. REST API

API merupakan singkatan dari *application programming interfaces*, dimana API digunakan oleh developer untuk mengizinkan dan mengintegrasikan aplikasi yang berada di platform yang berbeda atau perangkat berbeda untuk bisa saling terkoneksi antara satu dengan yang lainnya.

Tujuan utama kenapa API digunakan oleh developer dalam pembuatan sebuah perangkat lunak adalah untuk saling berbagi data antara aplikasi yang sama dari platform yang sama maupun berbeda. Penggunaan API juga bisa digunakan untuk menyediakan function sehingga para developer tidak perlu membuat fungsi lagi dalam codingnya dan hanya perlu memanggilnya saja

Sedangkan REST merupakan kependekan dari Representational State Transfer. REST merupakan sebuah web service yang berjalan di client dan server dimana yang memiliki sifat stateless. Yang mempunyai arti bahwa setiap request yang dikirim oleh aplikasi maka harus menyertakan semua parameter dan datanya dengan lengkap.

Client REST (aplikasinya) akan menjalankan request ke server REST, ketika request terkirim ke server selanjutnya REST server akan memberikan respon. Kemudian client REST akan menampilkan responnya atau jika tidak, client akan melakukan pemrosesan yang lain.

Respon dari REST Server ke REST client dapat diberikan dalam berbagai format, yaitu HTML, XML JSON dan beberapa format yang lain. Namun format yang paling yaitu JSON karena format JSON lebih mudah untuk digunakan dan dipelajari.

Untuk bisa menggunakan RES ini, REST memiliki sebuah standarisasi yang harus dipakai dalam implementasinya, standarisasi REST ini menggunakan URL dan juga HTTP verb. Dengan menggunakan standarisasi yang menggunakan URL maka developer dapat melakukan beberapa operasi berdasarkan HTTP verbs. Secara teknis HTTP verbs ini berupa $_SERVER['REQUEST_METHOD'].

Terdapat 5 method request HTTP yang sangat umum digunakan pada arsitektur REST antara lain:

GET	Digunakan untuk membaca atau

	mengakses data yang ada.
POST	Digunakan untuk membuat sebuah resource baru.
PUT	Digunakan untuk memperbarui atau menambah sebuah resource yang sudah ada.
PATCH	Digunakan untuk mengupdate kumpulan data yang ada di dalam resource secara partial.
DELETE	Digunakan untuk menghapus resource.

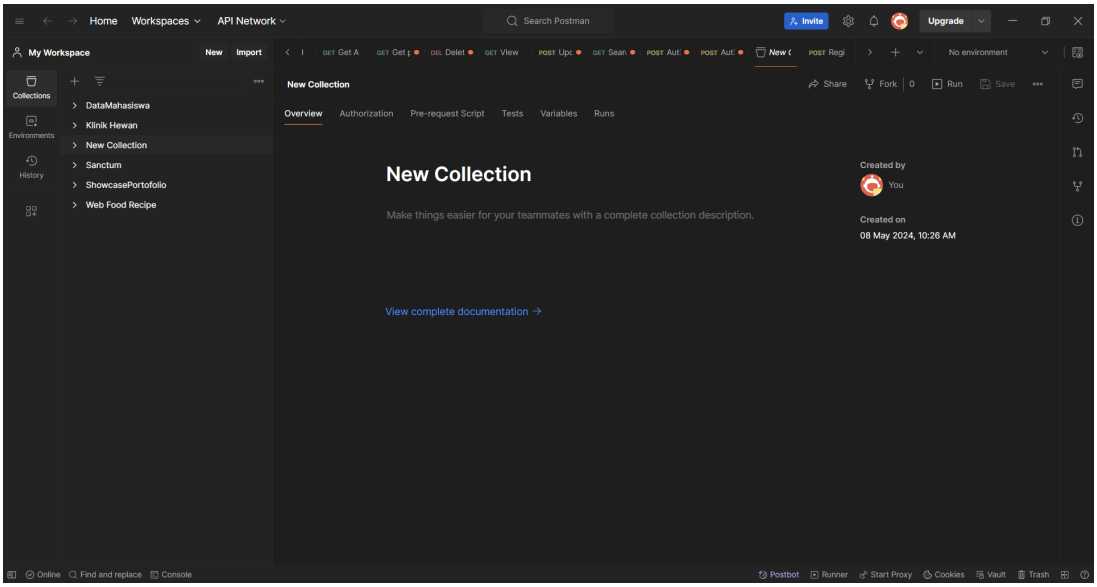
Selain itu juga terdapat HTTP status code, yang merupakan jawaban server atas permintaan data oleh client. Jawab ini berbentuk tiga digit kode yang memiliki arti dari setiap masing-masing status code yang dikembalikan.

1xx - Information Responses	HTTP Request berfungsi untuk membaca dan menjelaskan permintaan pesan yang dikirimkan oleh client. Permintaan atau request yang dikirimkan melalui aplikasi web browser seperti Mozilla Firefox, Google Chrome dan lainnya.
2xx - Successful Responses	Kode status ini menunjukan tindakan yang diminta oleh klien telah diterima, dipahami dan diproses dengan sukses.
3xx - Redirection Messages	Kode status ini menunjukan bahwa tindakan lebih lanjut perlu dilakukan oleh agen pengguna untuk memenuhi permintaan.
4xx - Client Error Responses	Kode status ini menunjukan bahwa server mengalami

	galat/error internal saat mencoba untuk memproses permintaan tersebut. Kesalahan ini cenderung dari server sendiri dan tidak berkaitan dengan permintaan.
5xx - Server Error Responses	Kode status ini menunjukan bahwa server mengalami error internal saat mencoba untuk memproses permintaan tersebut. Kesalahan ini cenderung dari server sendiri dan tidak berkaitan dengan permintaan.

B. POSTMAN

Postman adalah developing tools yang membantu pengguna untuk membangun, menguji dan memodifikasi API. Ia menawarkan para developer berbagai fitur dan fungsi yang penting untuk tahap pengembangan. Instalasi postman bisa didapatkan pada link berikut ini https://www.postman.com/downloads/



LATIHAN

LATIHAN 1

Praktikan dapat menuliskan tema dan tema ini akan dipakai hingga modul 6.

Note : Tidak diperbolehkan mengganti github dan tidak boleh ada tema yang sama dalam satu kelas!

List Pengumpulan Url Github Praktikum Web

LATIHAN 2

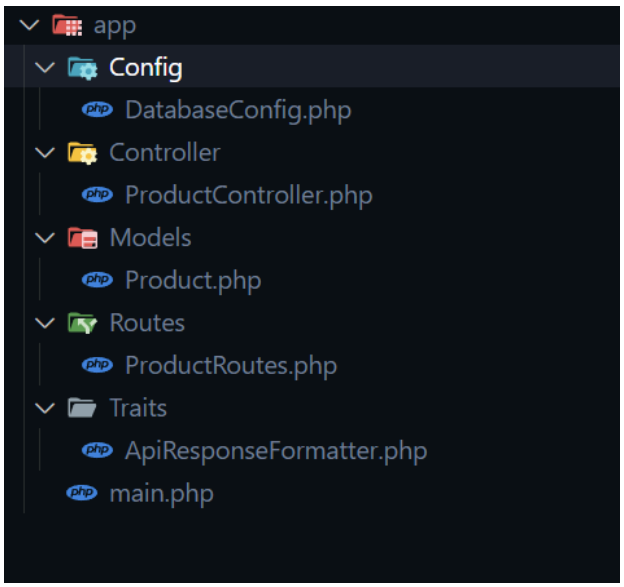
Sekarang kita akan mencoba untuk melakukan interaksi dengan MySQL Database menggunakan PHP dan kita akan membuat 4 Endpoint REST API yang dapat melakukan CRUD.

Berikut rincian endpoint yang akan dibuat

GET	http://localhost:8000/api/product
GET	http://localhost:8000/api/product/{id}
POST	http://localhost:8000/api/product
PUT	http://localhost:8000/api/product/{id}
DELETE	http://localhost:8000/api/product/{id}

Sebelum membuat kode pastikan kita sudah mengaktifkan MySQL dan membuat database sesuai dengan yang sudah ada pada materi diatas.

Struktur Folder:



FIle DatabaseConfig.php

```php
<?php

namespace app\Config;

class DatabaseConfig
{
    // Setting database
    public $host = "localhost";
    public $user = "root";
    public $password = "";
    public $database_name = "praktikum4";
    public $port = 3306;
}

```

Kode di atas mendefinisikan kelas DatabaseConfig yang berfungsi untuk menyimpan konfigurasi koneksi ke database, termasuk nama server ($host), nama pengguna ($user), kata sandi ($password), nama database

($database_name), dan nomor port ($port). Konfigurasi ini diatur di namespace app\Config untuk menghindari konflik nama dengan kode lain. Nantinya, kelas ini akan digunakan untuk menginisialisasi koneksi ke database dalam aplikasi, memastikan semua informasi yang dibutuhkan tersedia untuk mengakses data dengan benar.

File ApiResponseFormatter.php

```php
<?php

namespace app\Traits;

// Untuk formatting response
trait ApiResponseFormatter
{
    public function apiResponse($code = 200, $message = "success", $data = [])
    {
        // Dari parameter akan diformat menjadi format dibawah ini
        return json_encode([
            "code" => $code,
            "message" => $message,
            "data" => $data
        ]);
    }
}

```

Kode di atas mendefinisikan trait bernama ApiResponseFormatter dalam namespace app\Traits, yang berfungsi untuk memformat respons API dalam bentuk JSON. Fungsi apiResponse menerima parameter code, message, dan data, lalu mengembalikannya dalam format JSON dengan struktur standar, misalnya {"code": 200, "message": "success", "data": [...]}, memudahkan konsistensi respons di seluruh aplikasi.

File Product.php

```php
<?php

namespace app\Models;

include "app/Config/DatabaseConfiguration.php";


use app\Config\DatabaseConfig;
use mysqli;

class Product extends DatabaseConfig
{
    public $conn;

    public function __construct()
    {
        // connect ke database mysql
        $this->conn = new mysqli($this->host, $this->user, $this->password, $this->database_name, $this->port);
        // check koneksi
        if ($this->conn->connect_error) {
            die("Connection Failed: " . $this->conn->connect_error);
        }
    }

    // Function menampikan sebuah data
    public function findAll()
    {
        $sql = "SELECT * FROM products";
        $result = $this->conn->query($sql);
        $this->conn->close();
        $data = [];
        while ($row = $result->fetch_assoc()) {
            $data[] = $row;
        }
        return $data;
    }

    // Function menampilkan data dengan id
    public function findById($id)
    {
        $sql = "SELECT * FROM products where id = ?";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("i", $id);
        $stmt->execute();
        $result = $stmt->get_result();
        $this->conn->close();
        $data = [];
        while ($row = $result->fetch_assoc()) {
            $data[] = $row;
        }
        return $data;
    }
```

```php
    // Function untuk membuat sebuah data
      public function create($data)
      {
          $productName = $data['product_name'];
          $query = "INSERT INTO products (product_name) VALUES (?)";
          $stmt = $this->conn->prepare($query);
          $stmt->bind_param("s", $productName);
          $stmt->execute();
          $this->conn->close();
      }

      // Function untuk update data
      public function update($data, $id)
      {
          $productName = $data["product_name"];
          $query = "UPDATE products SET product_name = ? WHERE id = ?";
          $stmt = $this->conn->prepare($query);
          $stmt->bind_param("si", $productName, $id);
          $stmt->execute();
          $this->conn->close();
      }

      // Function delete data dengan id
      public function delete($id)
      {
          $query = "DELETE FROM products WHERE id = ?";
          $stmt = $this->conn->prepare($query);
          $stmt->bind_param("i", $id);
          $stmt->execute();
          $this->conn->close();
      }
  }

```

Kode di atas digunakan untuk membuat model Product dalam aplikasi PHP yang berfungsi mengelola operasi database terkait tabel products. Kelas ini menghubungkan aplikasi ke database menggunakan konfigurasi dari DatabaseConfig dan menyediakan fungsi CRUD (Create, Read, Update, Delete) untuk produk. Fungsi findAll mengambil semua data produk, findById mengambil data produk berdasarkan ID, create menambahkan produk baru, update memperbarui produk berdasarkan ID, dan delete menghapus produk

berdasarkan ID. Kelas ini menggunakan ekstensi mysqli untuk menjalankan query SQL secara aman dan efisien.

File ProductContoller.php

```php
<?php

namespace app\Controller;

include "app/Traits/ApiResponseFormatter.php";
include "app/Models/Product.php";

use app\Models\Product;
use app\Traits\ApiResponseFormatter;

class ProductController
{
    use ApiResponseFormatter;

    public function index()
    {
        // DEFINISI OBJEK MODEL PRODUCT YANG SUDAH DIBUAT
        $productModel = new Product();
        $response = $productModel->findAll();
        // RETURN $response DENGAN MELAKUKAN FORMATTING TERLEBIH DAHULU MENGGUNAKAN TRAIT YANG SUDAH DIPANGGIL
        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()
    {
        // TANGKAP INPUT JSON
        $jsonInput = file_get_contents('php://input');
        $inputData = json_decode($jsonInput, true);
        // VALIDASI INPUT VALID ATAU TIDAK
        if (json_last_error()) {
            return $this->apiResponse(400, "Error invalid input", null);
        }

        $productModel = new Product();
        $response = $productModel->create([
            "product_name" => $inputData['product_name']
        ]);

        return $this->apiResponse(200, "success", $response);
    }

    public function update($id)
    {
        $jsonInput = file_get_contents('php://input');
        $inputData = json_decode($jsonInput, true);
        if (json_last_error()) {
            return $this->apiResponse(400, "Error invalid input", null);
        }

        $productModel = new Product();
        $response = $productModel->update([
            "product_name" => $inputData['product_name']
        ], $id);

        return $this->apiResponse(200, "success", $response);
    }

    public function delete($id)
    {
        $productModel = new Product();
        $response = $productModel->delete($id);

        return $this->apiResponse(200, "success", $response);
    }
}

```

Kode di atas mendefinisikan ProductController dalam namespace app\Controller, yang berfungsi untuk mengelola data produk melalui beberapa metode, yaitu index, getById, insert, update, dan delete. ProductController menggunakan ApiResponseFormatter untuk mengirim respons API yang terstruktur dengan format JSON, termasuk status kode, pesan, dan data.

Metode index mengambil semua produk, getById mengambil produk berdasarkan ID, insert menambahkan produk baru, update memperbarui produk yang ada, dan delete menghapus produk berdasarkan ID.

File ProductRoutes.php

```php
<?php

namespace app\Routes;

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 SAMA DENGAN '/api/product'
        if ($method == "GET" && strpos($path, "/api/product/") == 0) {
            $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 SAMA DENGAN '/api/product'
        if ($method == "PUT" && strpos($path, "/api/product/") == 0) {
            $pathParts = explode("/", $path);
            $id = $pathParts[count($pathParts) - 1];

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

        // JIKA REQUEST METHOD DELETE DAN PATH SAMA DENGAN '/api/product'
        if ($method == "DELETE" && strpos($path, "/api/product/") == 0) {
            $pathParts = explode("/", $path);
            $id = $pathParts[count($pathParts) - 1];

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

```

Kode di atas mendefinisikan ProductRoutes dalam namespace app\Routes, yang menangani routing HTTP untuk berbagai operasi pada produk dalam sebuah API. Class ProductRoutes berisi metode handle yang menerima dua parameter, $method dan $path, untuk menentukan tindakan yang sesuai berdasarkan metode HTTP (GET, POST, PUT, DELETE) dan URI yang diberikan. Misalnya, jika

metode permintaan adalah GET pada path /api/product, maka method index pada ProductController akan dijalankan untuk mengambil semua produk. Untuk path yang mengandung ID produk, ID tersebut diekstrak dan digunakan untuk menjalankan tindakan spesifik seperti getById, update, atau delete.

File main.php

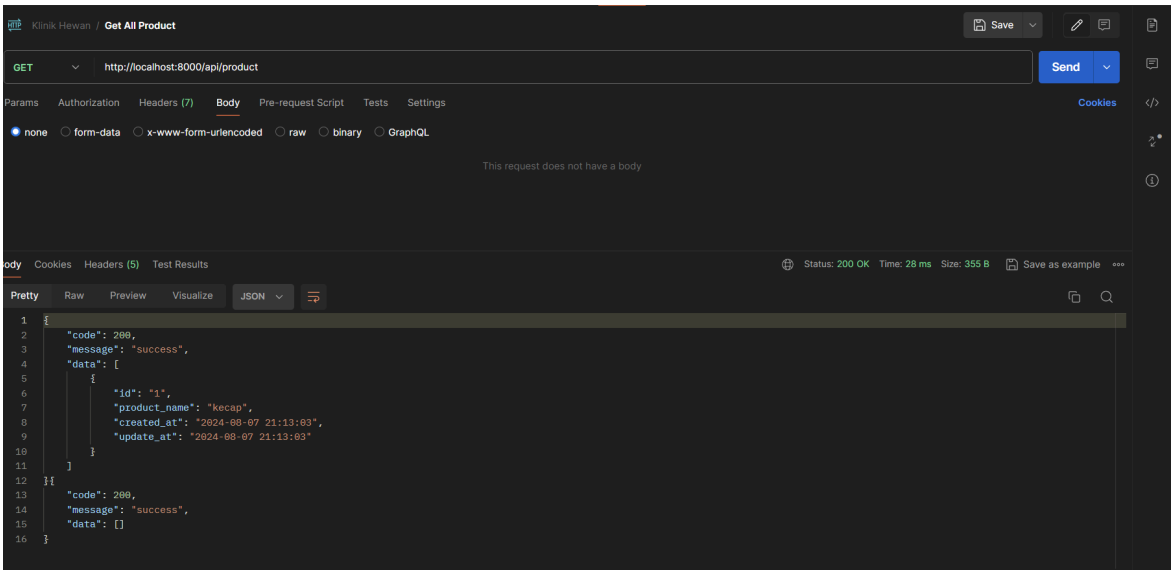
```php
<?php

header("Content-Type: application/json; charset=UTF-8");

include "app/Routes/ProductRoutes.php";

use app\Routes\ProductRoutes;

// MENANGKAP REQUEST METHOD
$method = $_SERVER['REQUEST_METHOD'];
// MENANGKAP REQUEST PATH
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

// PANGGIL ROUTES
$productRoutes = new ProductRoutes();
$productRoutes->handle($method, $path);

```

Setelah mengikuti langkah-langkah diatas, bukalah terminal dan jalankan perintah berikut

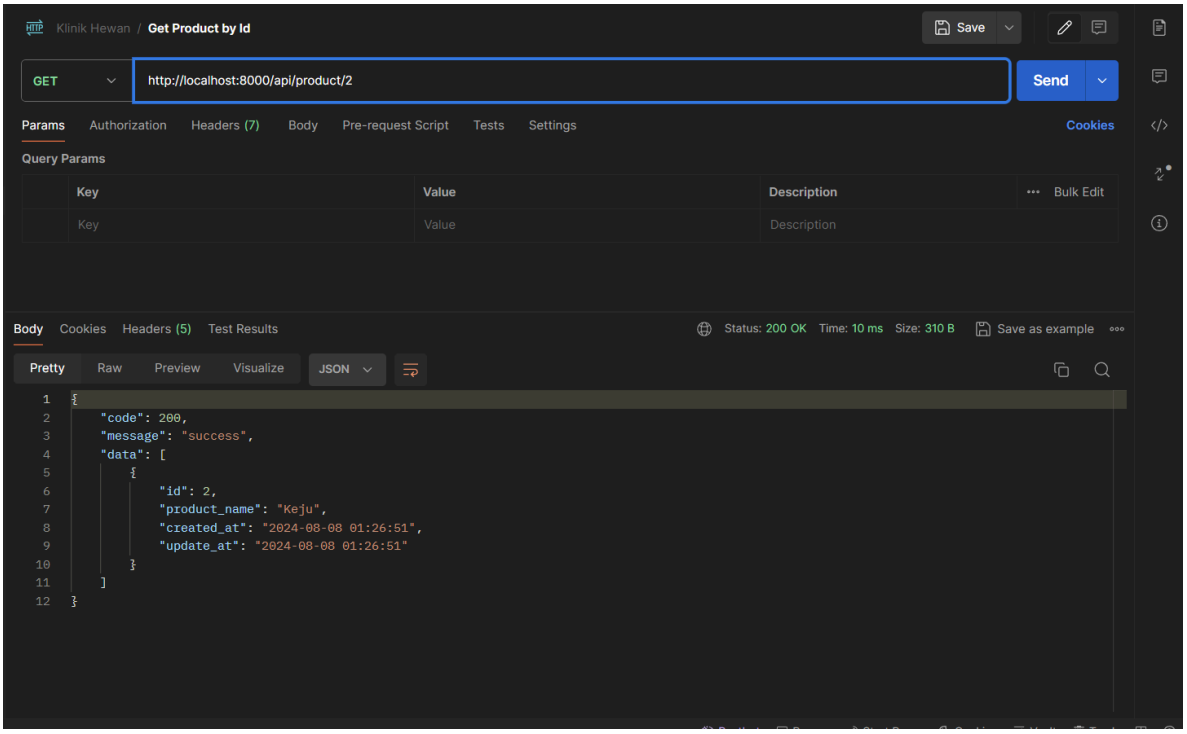
“php -S localhost:8000”

Kemudian bukalah aplikasi postman dan cobalah seperti tabel dibawah ini.

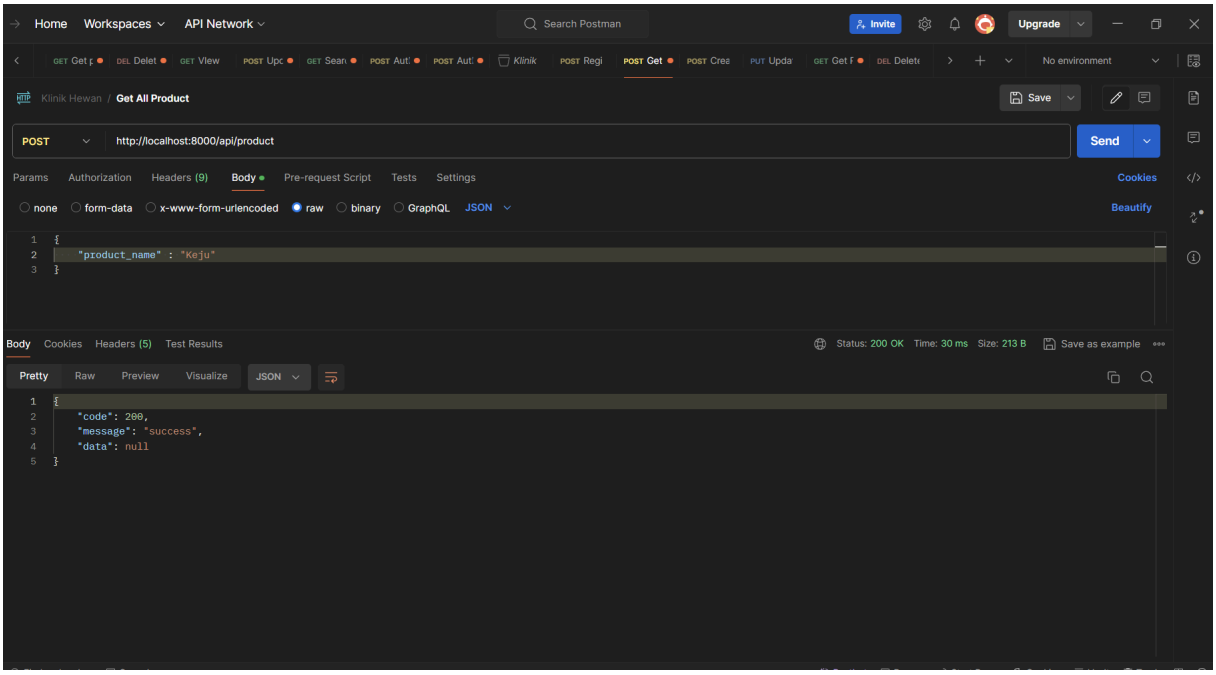
(GET ALL PRODUCT) [GET] : http://localhost:8000/api/product



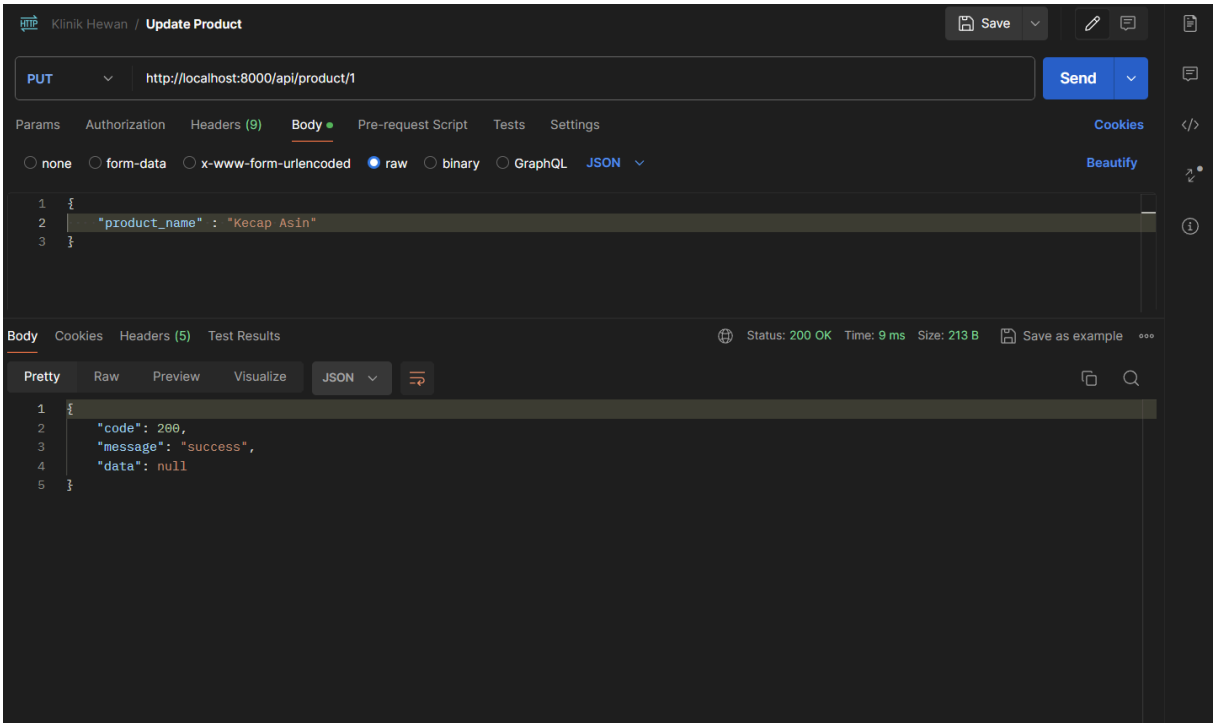
(GET PRODUCT BY ID) [GET] : http://localhost:8000/api/product/{id}



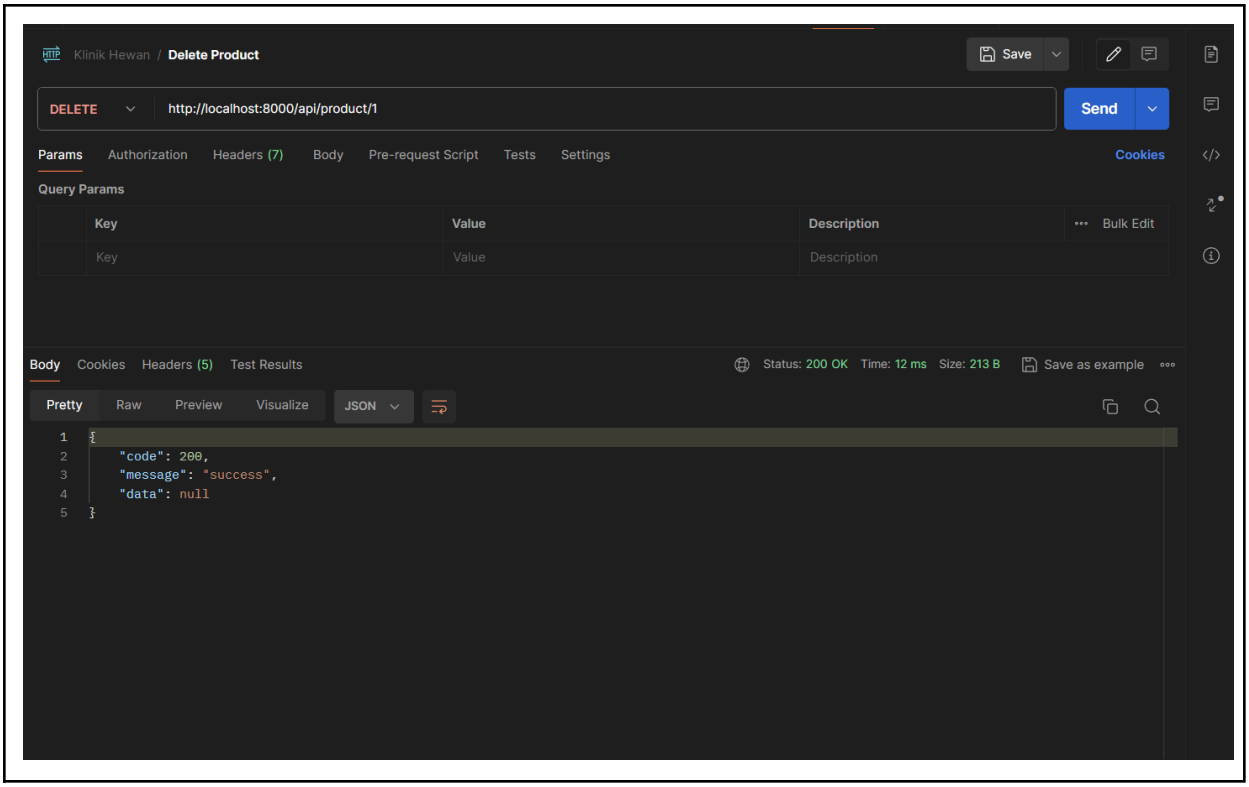
(CREATE PRODUCT) [POST] : http://localhost:8000/api/product



UPDATE PRODUCT [PUT] : http://localhost:8000/api/product/{id}



DELETE PRODUCT [DELETE] : http://localhost:8000/api/product/{id}



TUGAS

TUGAS 1 (FONTEND)

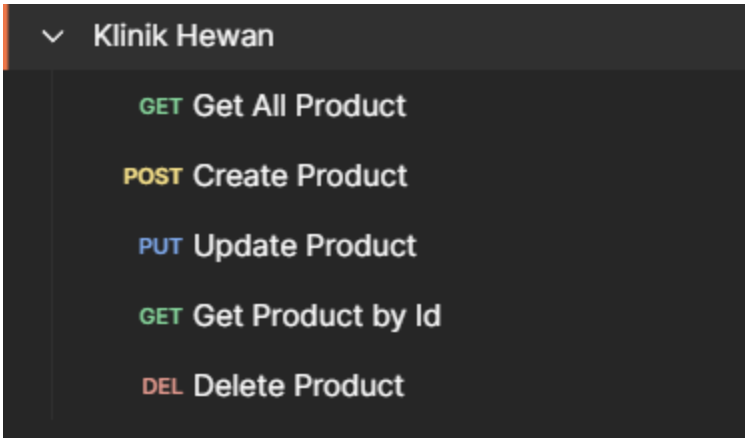
Buatlah sebuah Front End untuk website menggunakan HTML, CSS dan JS sesuai dengan tema yang sudah di isi pada codelab minggu lalu. Pastikan pada website terdapat komponen seperti navbar, main content dan footer.

(semakin banyak konten semakin baik)

NOTE: Tidak diperbolehkan memakai framework css atau javascript.

TUGAS 2 (BACKEND)

Buatlah sebuah Create, Read, Update, Delete (CRUD) Endpoint API backend seperti pada contoh materi diatas dan buatlah collection pada Postman yang berisi Endpoint yang buat masing - masing sesuai dengan HTTP Method (GET, POST, PUT, DELETE) seperti gambar berikut



Pastikan Endpoint API sudah sesuai dengan Tugas 1.

TUGAS 3

Dari Tugas 1 dan Tugas 2 lakukan integrasi antara Frontend dan Backend sehingga website dapat menampilkan data sesuai dengan yang tersimpan dalam database.

NOTE: Dilarang menggunakan kode yang berbeda dengan Tugas 1 dan Tugas 2. Jika kode yang digunakan untuk Tugas 3 berbeda dengan yang sudah diberikan pada tugas sebelumnya, maka nilai maksimal untuk Tugas 3 adalah C.

KRITERIA & DETAIL PENILAIAN

KRITERIA	PERSENTASE PENILAIAN
Dapat menerapkan HTML, CSS dan JS Front End Website	15%
Dapat menerapkan Restful API CRUD	35%
Dapat melakukan integrasi Front End dan Back End	20%
Pemahaman	30%