VERSI 1.1

NOVEMBER2025

[PEMROGRAMAN WEB]

MODUL 5 - LARAVEL API DAN DATABASE

DISUSUN OLEH:

AMINUDIN, S.KOM., M.CS.

YUSUF NUR MUHAMMAD

KRISNA BIMANTORO

TIM LABORATORIUM INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PENDAHULUAN

TUJUAN

Memberikan pemahaman kepada mahasiswa mengenai **Laravel API** dan **Database**, sehingga mampu:

1. Menginstal Laravel,
2. Membuat database dan tabel dengan migration,
3. Menggunakan ORM Eloquent,
4. Membuat model, controller, dan routing,
5. Mengimplementasikan endpoint CRUD,
6. Melakukan uji coba API dengan Postman.

TARGET MODUL

Setelah menyelesaikan modul ini mahasiswa diharapkan mampu:

1. Menjelaskan konsep API, URI, endpoint, dan method,
2. Membuat database dan konfigurasi di Laravel,
   Menggunakan migration untuk membuat dan mengubah tabel,
3. Mengembangkan API dengan Laravel (CRUD lengkap),
4. Menguji API menggunakan Postman.

PERSIAPAN

Sebelum mengikuti praktikum, mahasiswa perlu:

1. Menguasai dasar-dasar PHP,
2. Sudah menginstal **Laragon** sebagai web server stack,
3. Menyiapkan **Composer** dan **Laravel Installer**,
4. Menyediakan aplikasi uji API (**Postman**).

KEYWORDS

API, Endpoint, URI, Laravel, MySQL, CRUD, Postman

TABLE OF CONTENTS

MATERI

DATABASE

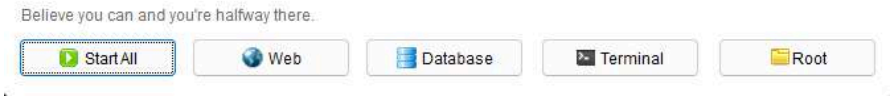
Dari kalian sendiri mungkin sudah tidak asing mendengar kata *database* jadi database sendiri ialah tempat penyimpanan data yang terstruktur agar mudah diakses, dikelola, dan dimanipulasi oleh aplikasi. Pada konteks pemrograman web database digunakan untuk menyimpan informasi seperti data pengguna, data nilai dan lain sebagainya.

Pada pemrograman web kali ini kita akan menggunakan jenis database yang berbasis SQL (*Structured Query Language*) yang sebelumnya pernah dipelajari pada mata kuliah basis data namun menggunakan **database relasional** yang berbeda, yaitu akan menggunakan database MySQL.

Step by step cara menjalankan database di local:

Windows User

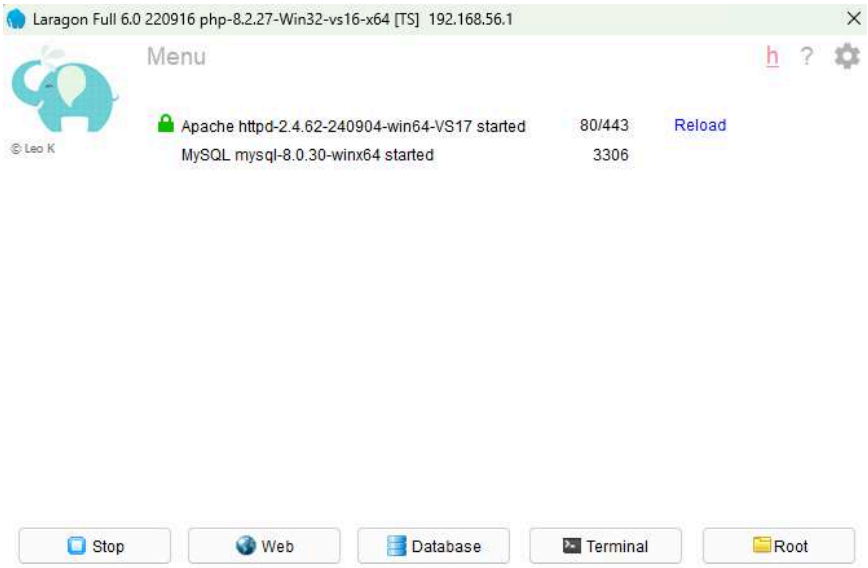
1. Pada kali ini kita akan menggunakan laragon sebagai **Web server stack**



Pada tampilan awal laragon kalian akan menemui tampilan seperti di atas, untuk mengaktifkan database local menggunakan laragon cukup mudah hanya perlu menekan tombol 

yang ada tampilan laragon

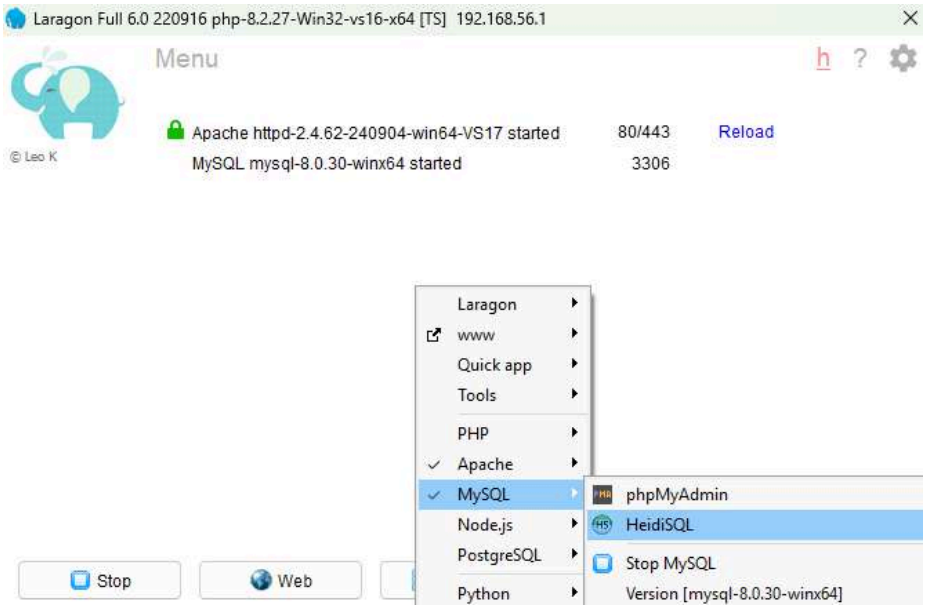
2. Tampilan setelah diaktifkan



Tampilan diatas menunjukan bahwa laragon sudah berhasil menjalankan 2 service, yaitu pertama service apache atau webserver php(akan digunakan untuk run php) dan juga database mysql dengan port default (3306)

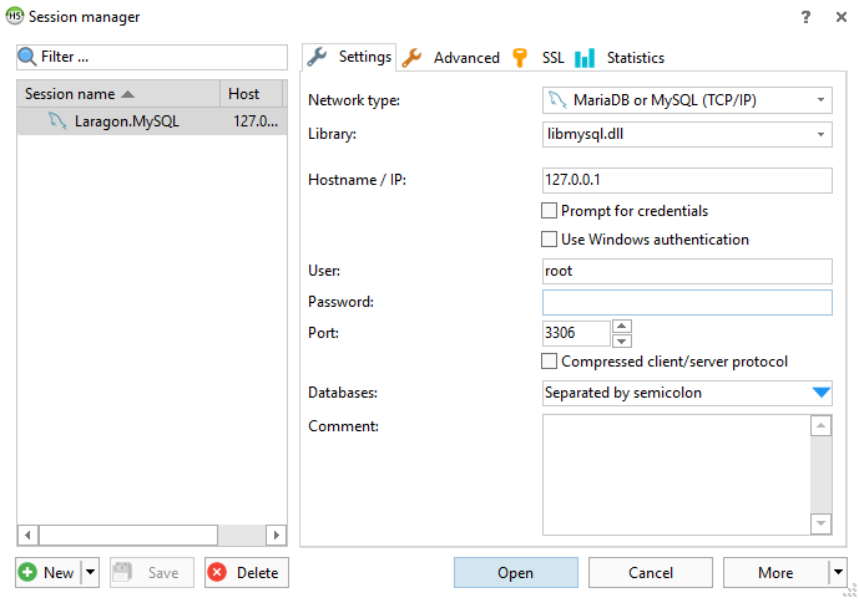
3. Akses database

   Pada laragon sendiri sudah memiliki management database bawaannya sendiri, yaitu HeidiSQL(direkomendasikan), banyak management database yang lain contohnya phpmyadmin atau navicat(direkomendasikan)



Kalian bisa klik kanan pada bagian kosong laragon dan arahkan kursor ke MySQL dan pilih HeidiSQL

4. Tampilan connection HeidiSQL

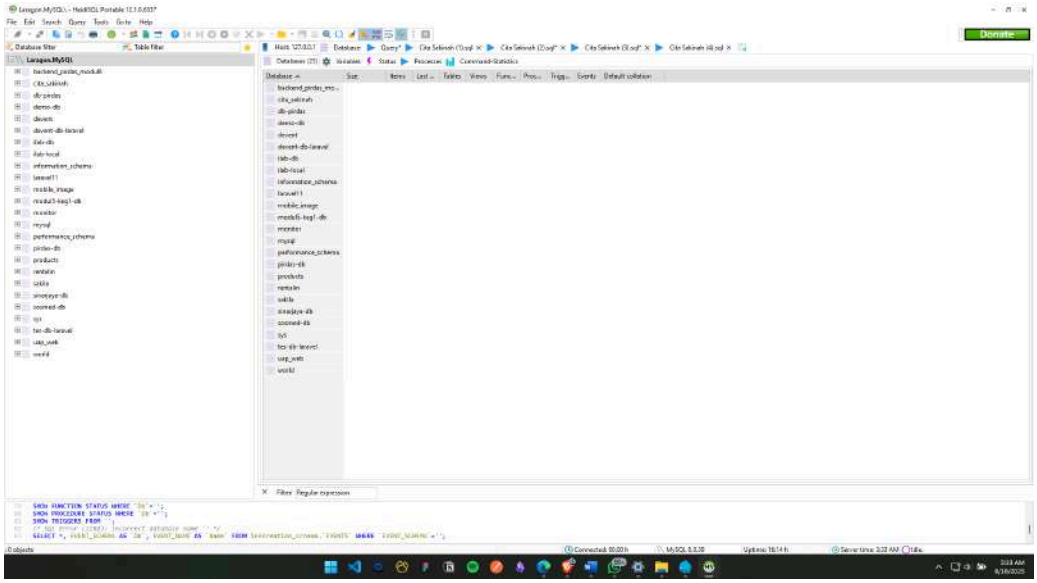


Ada beberapa poin yang perlu diperhatikan disini yaitu:

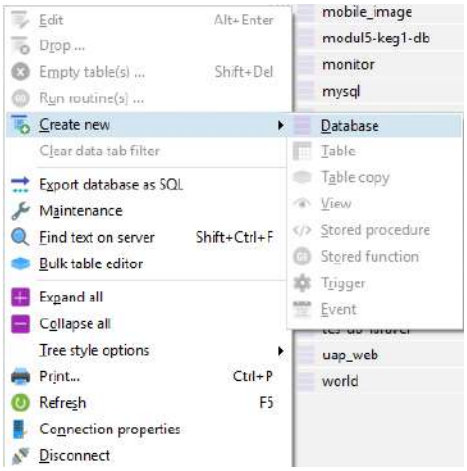
- Network Type: MariaDB or MySQL (TCP/IP) -> biarkan default
- Hostname/IP: 127.0.0.1 -> default localhost
- User: root -> default
- Password: defaultnya null atau kosong, tapi jika kalian membuat password sendiri silahkan disesuaikan
- Port: 3306 -> default

Konfigurasi diatas akan digunakan pada saat ingin konfigurasi database di Laravel

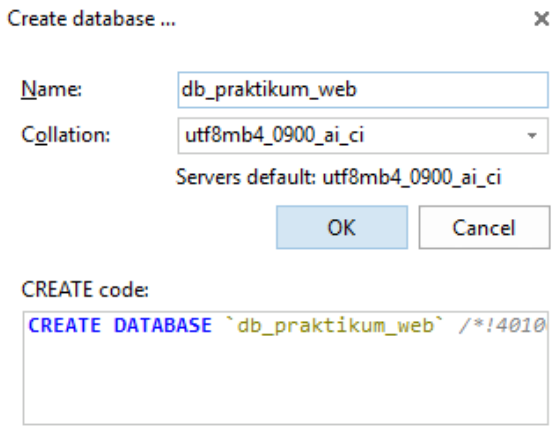
5. Tampilan awal HeidiSQL



6. Membuat database di HeidiSQL

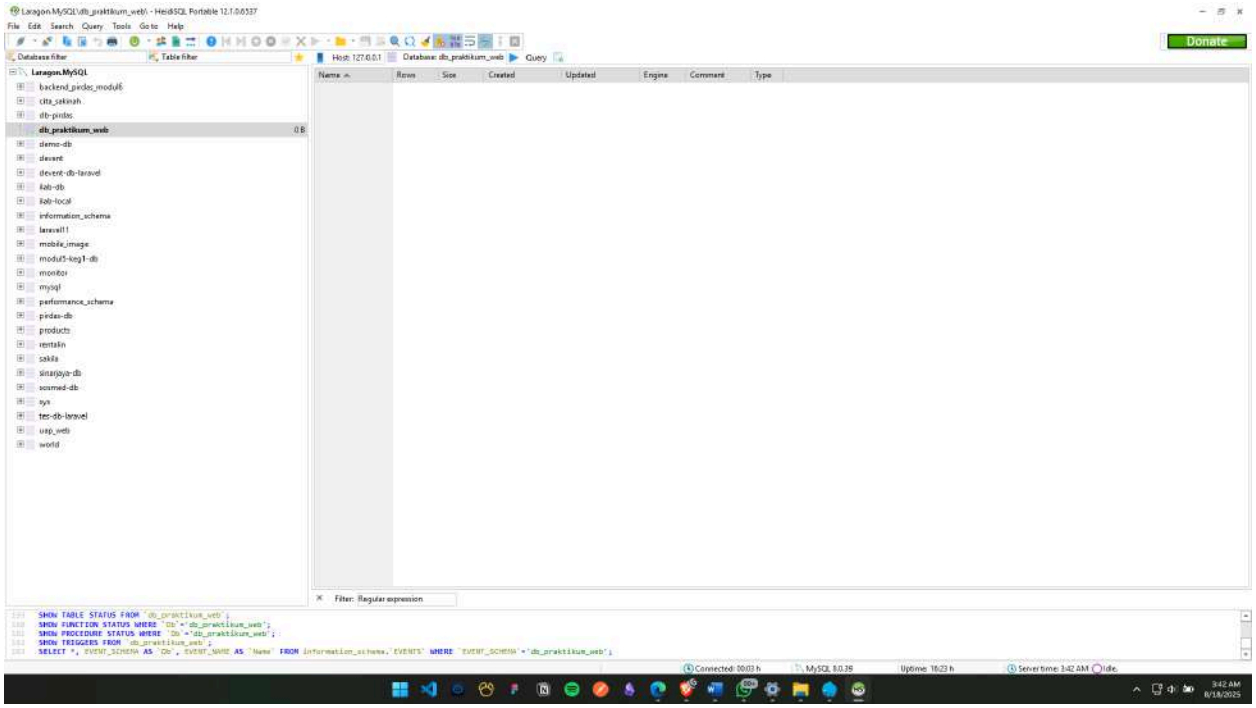


Pada bagian kiri tampilan (list database) klik kanan dan **create new** kemudian pilih **Database**



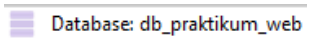
Pada tampilan create database bisa diisikan name sesuai kebutuhan kalian.

7. Akses database



Pada bagian kiri tampilan HeidiSQL merupakan list database yang ada pada local kalian, cara untuk connect ke database kalian adalah klik kiri 2 kali pada database yang dituju 

contoh klik 2 kali pada database ini yang sebelumnya dibuat.

Kemudian pada bagian kanan atas pilih 

nantinya data table akan muncul disini

Mac User

Untuk pengguna mac bisa menjalankan *brew install mysql* dan *brew services start mysql* pada terminal. Untuk connect database pada db manager kalian bisa menggunakan opsi menggunakan *navicat* atau *dbeaver.*

Pada modul ini kalian tidak diwajibkan menggunakan database manager sesuai modul, dibebaskan menggunakan apa saja.

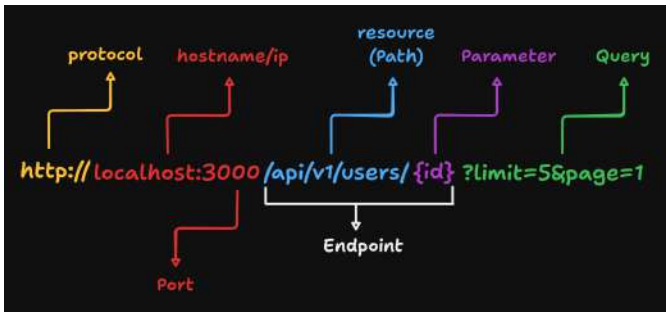
APA ITU API?

API atau kepanjangannya adalah *Application Programming Interface bukan streak*🔥 adalah antarmuka yang berfungsi sebagai penghubung antara sebuah aplikasi dan aplikasi lainnya, atau antara klien dan server, untuk memungkinkan integrasi fitur tanpa harus menambahkan data secara manual.

Contoh pada gambar diatas tampilan antarmuka website secara umum tidak bisa mengakses database secara langsung (client side) jadi API disini berguna untuk *menjembatani* agar website kita dapat digunakan secara dinamis menggunakan database.

URI dan Endpoint

Apa itu URI? URI (Uniform Resource Identifier) adalah alamat spesifik yang diperlukan untuk menemukan dan berinteraksi dengan sumber daya dalam suatu API. Dengan kata lain, URI merupakan alamat unik untuk suatu informasi tertentu yang terdapat di dalam API.



Pada gambar diatas adalah contoh dari struktur URI penggunaan parameter digunakan untuk menampilkan detail dari data users, dan penggunaan query disini berguna untuk memfilter response yang akan ditampilkan, pada URI diatas adalah contoh, jika menggunakan parameter untuk menampilkan detail dari user tidak perlu ada query lagi.

Sedangkan **Endpoint** adalah titik akses API—tempat Anda mengirim request untuk suatu operasi. Secara praktis, endpoint = HTTP method + URI path (sering ditambah versi). Contoh endpoint.

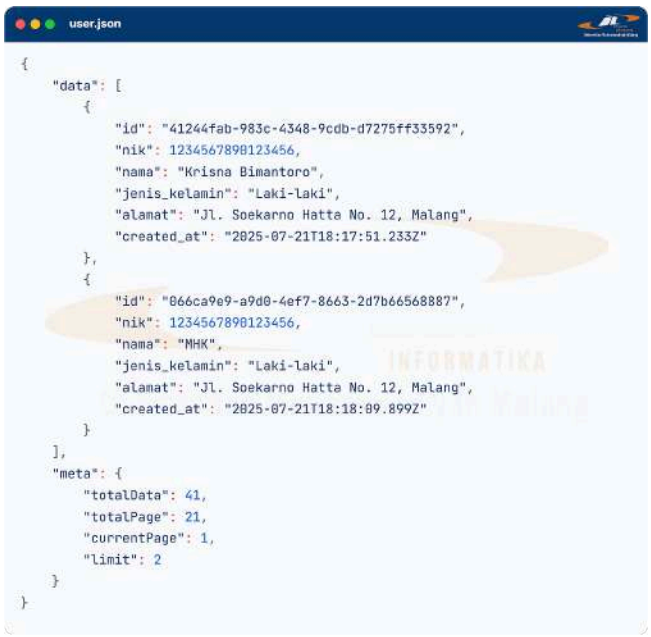
- GET /api/v1/users -> menampilkan list semua user bisa dikombinasikan dengan query untuk melimitasi response atau pagination.
- GET /api/v1/users/:id -> menampilkan detail user berdasarkan parameter id user tersebut.

Method

Pada REST API memiliki beberapa method HTTP yang bisa setiap methodnya mewakili tugasnya masing-masing (CRUD) diantaranya:

- GET

  Digunakan untuk mengambil data (READ) dari database yang responsenya akan berupa json.



  Contoh response ketika GET ke endpoint ke alamat uri http://localhost:3050/api/v1/user?page=1&limit=2

- POST

  Digunakan untuk menambahkan data baru (CREATE), endpoint meminta request pada body bisa berbentuk string ataupun file.

- PUT/PATCH

  Digunakan untuk memperbarui atau update data (UPDATE), PUT dan PATCH memiliki perbedaan, PUT memiliki behavior memperbarui seluruh resource pada satu baris data atau semua field, sedangkan PATCH kita bisa update salah satu field yang ingin diubah contoh nama atau kontak.

- DELETE

  Digunakan untuk menghapus data (DELETE) penerapan delete pada API bisa dua cara, dengan menerapkan *soft delete* (data tidak di drop di database) atau delete secara permanen (drop dari database).

Sebelumnya kita akan lebih kenal apa itu framework php, Framework PHP adalah sebuah platform yang digunakan sebagai kerangka kerja untuk membangun aplikasi web berbasis PHP. Framework PHP berisi kumpulan template kode yang sering digunakan, yang dapat membantu pengembang untuk membuat aplikasi web dengan lebih cepat.

Sedangkan laravel sendiri sebuah framework PHP open source yang berisi banyak modul dasar untuk mengoptimalkan kinerja PHP dalam pengembangan aplikasi web, terutama karena PHP adalah bahasa pemrograman yang dinamis dan Laravel di sini dapat berperan untuk membuat pengembangan web menjadi lebih cepat, lebih aman, dan lebih sederhana.

Laravel sendiri bekerja di sisi back-end atau sisi server. Selain powerfull, Laravel juga mudah dipahami. Dengan mengikuti pola arsitektur model-view-controller (MVC), Laravel dapat mempercepat proses pembuatan aplikasi web. Pada arsitektur MVC, pengembangan dapat dilakukan dengan lebih cepat karena pengembang dapat fokus pada satu bagian saja seperti model (bagian yang mengatur database), view (bagian yang mengatur tampilan ke pengguna), dan controller (bagian yang menghubungkan antara model dan view jika ada permintaan dari pengguna).

KELEBIHAN LARAVEL

Berikut ini adalah penjelasan mengenai kelebihan dari Laravel:

- Mempercepat waktu pengembangan aplikasi karena Laravel menggunakan komponen dari framework lain dan library bawaan dalam mengembangkan aplikasi web.
- Manajemen sumber daya yang lebih mudah karena menggunakan namespace dan interface.
- Performa aplikasi yang lebih baik. Laravel telah lulus uji kualitas dan kecepatan sehingga aplikasi yang dibangun dengan Laravel dapat memiliki performa yang lebih cepat.
- Aplikasi yang dibangun dengan Laravel lebih aman secara default. Aplikasi dapat lebih aman dari CSRF, dan injeksi SQL. Laravel juga dilengkapi dengan beberapa langkah keamanan dengan menerapkan prinsip keamanan OWASP (Open Web Application Security Project).
- Lebih sedikit kode. Dengan menggunakan framework Laravel, dapat menggunakan lebih sedikit kode asli dengan menggunakan fungsi-fungsi bawaan Laravel.
- Dukungan komunitas yang luas. Saat ini ada komunitas Laravel yang besar yang berarti bahwa masalah apa pun yang mungkin dihadapi dapat diselesaikan secara tepat waktu.

INSTALASI LARAVEL

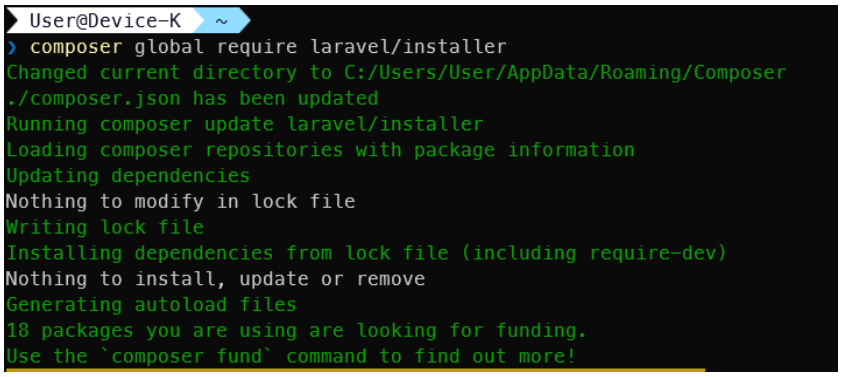
1. **Installasi Composer**

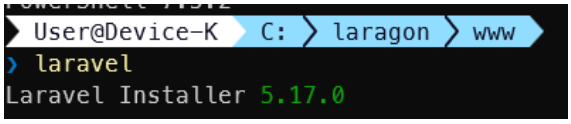
   Documentation Installer Windows: *INSTALL_COMPOSER_WINDOWS*
   Documentation Installer Mac: *INSTALL_COMPOSER_MAC*

   Jika sudah terinstall pada terminal kalian (power shell) ketik composer maka akan muncul seperti pada gambar.

2. **Install Laravel CLI**

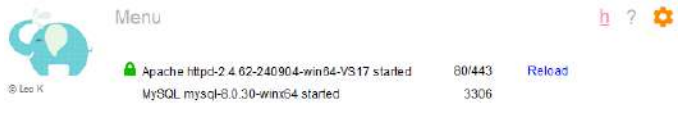
   Pada terminal kalian ketik
   composer global require laravel/installer



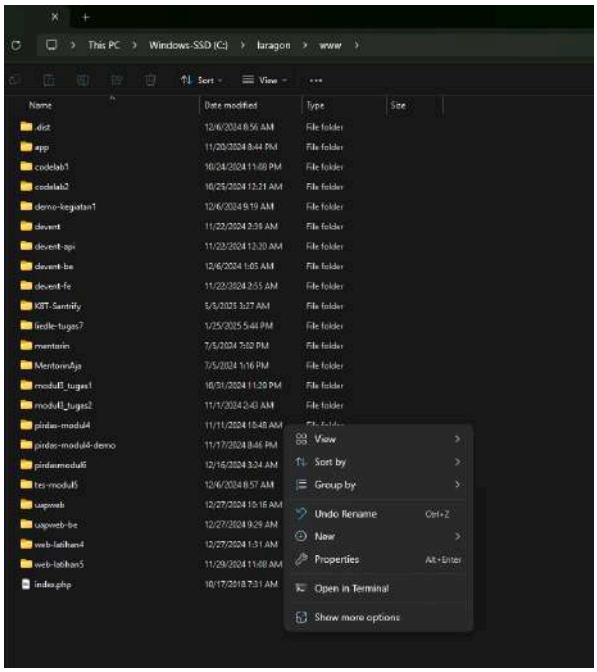


   Pada laptop pembuat modul sudah terinstall laravel/installer maka kemungkinan log akan berbeda.

3. **Membuat Project Laravel**

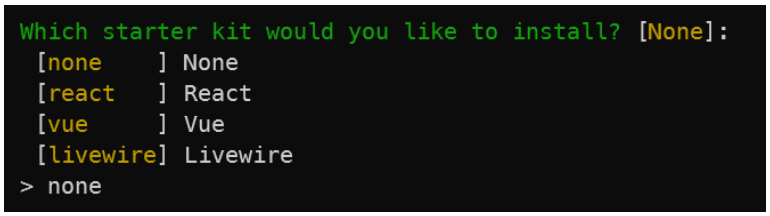


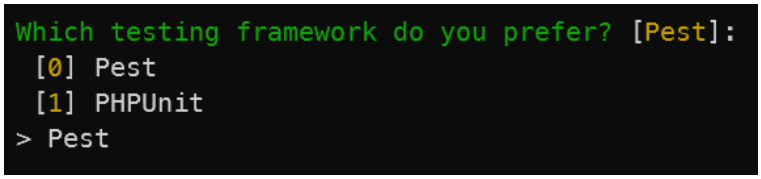
Pada tampilan awal laragon klik tombol **Root,** yang akan membuka folder root dari laragon, hal ini harus dilakukan karena php dijalankan di webserver laragon bukan di install di mesin kalian.



Klik kanan pada bagian yang kosong dan klik **Open in Terminal**
Pada terminal ketikan ***laravel new pemrograman-web*** untuk membuat project

Selanjutnya ikuti perintah seperti gambar di bawah jika ada pilihan:





Pada bagian ini dibebaskan pilih yang mana

```
Which database will your application use? [MySQL]:
  [mysql  ] MySQL
  [mariadb] MariaDB
  [pgsql  ] PostgreSQL
  [sqlite ] SQLite (Missing PDO extension)
  [sqlsrv ] SQL Server (Missing PDO extension)
> mysql
```

```
Default database updated. Would you like to run the default database migrations? (yes/no) [yes]:
> no
```

```
Would you like to run npm install and npm run build? (yes/no) [yes]:
> yes
```

Jika semua sudah selesai dan tanpa error maka log akan menampilkan pesan berikut

```
   INFO  Application ready in [pemrograman-web]. You can start your local development using:

→ cd pemrograman-web
→ composer run dev

  New to Laravel? Check out our documentation. Build something amazing!
```

Selamat sudah selesai install laravel, link dokumentasi laravel
LARAVEL_DOCUMENTATION

4. **Pindah direktori**

```
User@Device-K  C:  laragon  www
❯ cd .\pemrograman-web\
```

Pindah ke direktori saat membuat project laravel

5. **Buka VS Code menggunakan terminal**

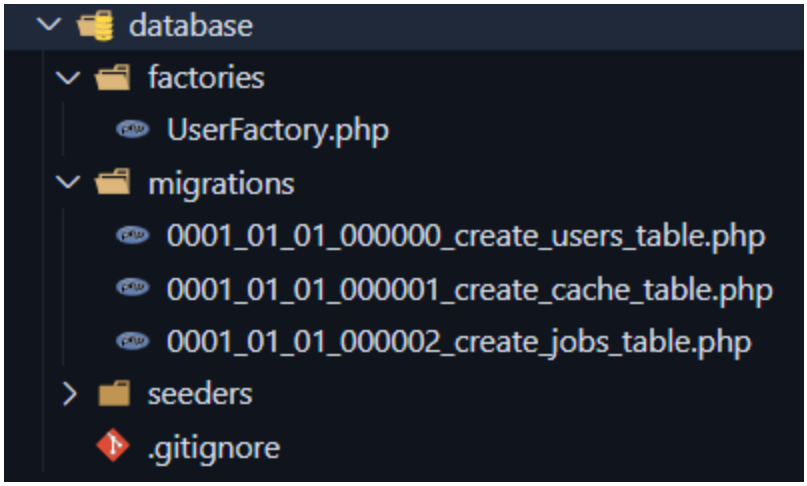
```
User@Device-K  C:  laragon  www  pemrograman-web
❯ code .
```

6. **Konfigurasi Database**

Pada bagian ***database*** sebelumnya kita sudah berhasil membuat database, sekarang kita akan menggunakan database itu di project laravel kita dengan mengkonfigurasi file .env.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_praktikum_web
DB_USERNAME=root
DB_PASSWORD=
```

Pastikan konfigurasi yang ada pada .env sudah benar dengan yang ada di konfigurasi database.



Bisa dilihat pada gambar, di dalam folder database pada project laravel, laravel sudah menyediakan migration bawaan yang bisa kita gunakan, migration ini akan membuat beberapa tabel.

Kegunaan dari migration ini adalah kita tidak perlu repot-repot query sql untuk membuat table baru, karena laravel sudah memiliki ORM bawaan. Apa itu ORM? Akan dijelaskan pada bagian berikutnya

Pada terminal di vs code kalian, ketik ***php artisan migrate*** ini akan menjalankan semua file migration yang ada di folder migrations.

```
User@Device-K  C:  laragon  www  pemrograman-web
> php artisan migrate
```

Jika menampilkan log seperti dibawah maka tandanya berhasil

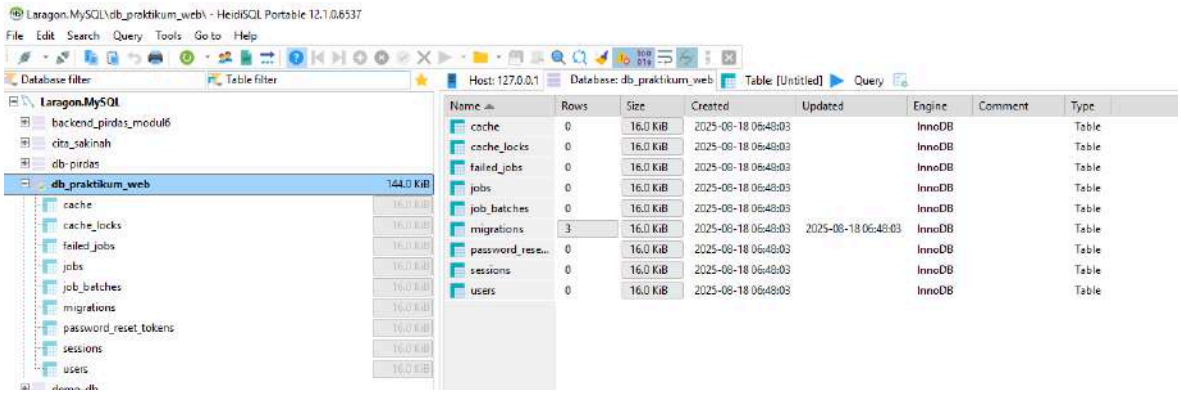
```
INFO  Preparing database.

Creating migration table ..... 38.79ms DONE

INFO  Running migrations.

0001_01_01_000000_create_users_table ..... 104.49ms DONE
0001_01_01_000001_create_cache_table ..... 29.96ms DONE
0001_01_01_000002_create_jobs_table ..... 73.61ms DONE
```

Selanjutnya silahkan buka HeidiSQL dan cek apakah table berhasil dibuat



Bisa dilihat pada gambar tabel sudah berhasil dibuat dari migration yang sudah dijalankan.

7. **Jalankan Laravel**

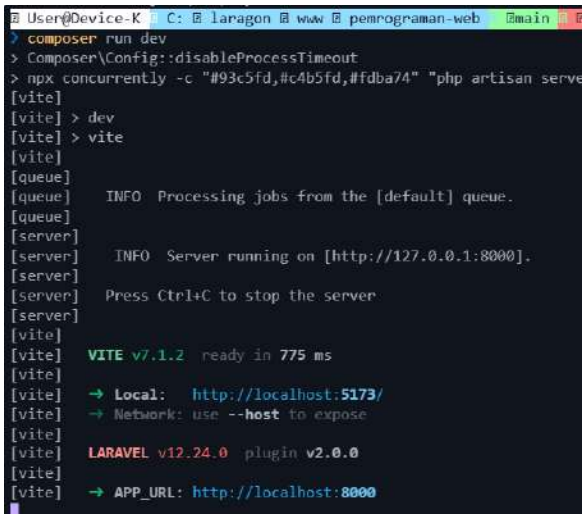
   Terdapat 2 cara untuk menjalankan laravel, pertama dengan mengetikan

   php artisan serve –-port:8020

   Command di atas hanya akan menjalankan laravel saja pada port 8020

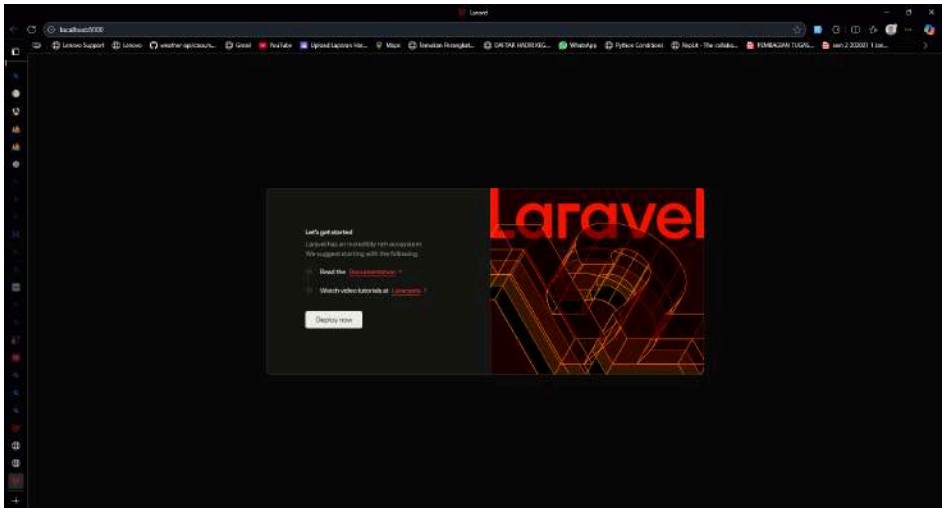
   Cara kedua adalah dengan:

   composer run dev



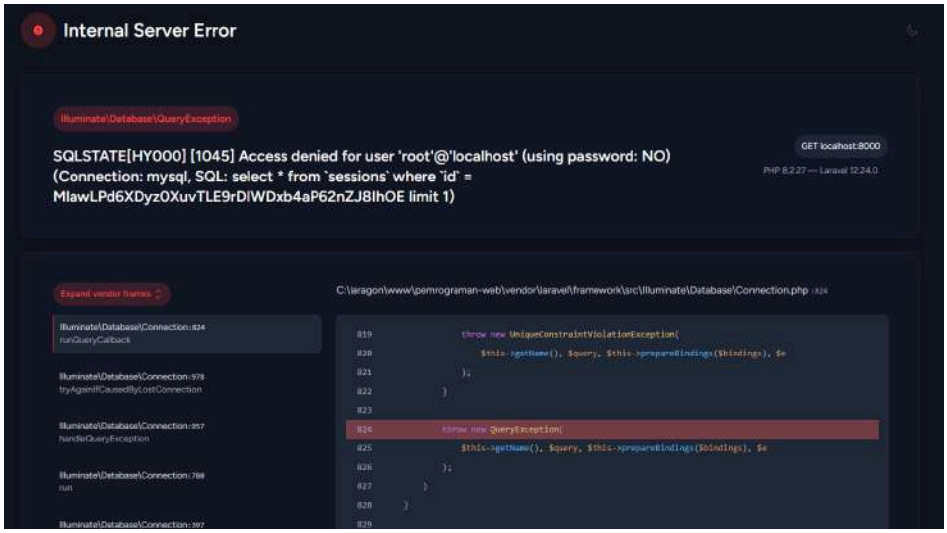
   Command ini akan menjalankan Vite dan juga laravel di default port

8. Akses laravel pada website



Jika tampilannya diatas maka dipastikan laravel berjalan dengan normal, untuk mengakses bisa menggunakan ***localhost:port*** di atau tekan control+click pada link di terminal

9. (Bad Ending) Jika mengalami errror



Seperti gambar diatas, silahkan cek kembali konfigurasi database kalian pastikan konfigurasinya benar.

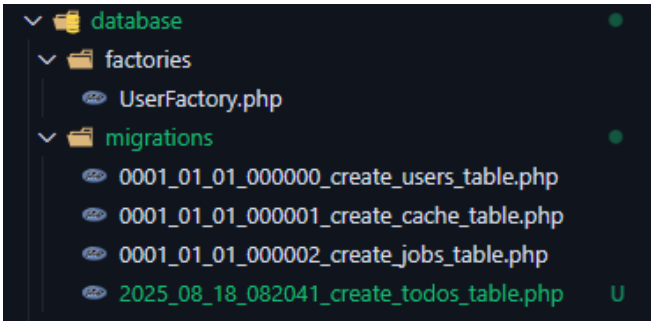
ELOQUENT ORM

Seperti yang sudah dijelaskan sebelumnya, pada laravel sudah memiliki ORM bawaannya sehingga kita tidak perlu membuat query manual atau raw query pada service backend kita. Apa itu ORM? **ORM (Object Relational Mapping)** adalah sebuah teknik/programming tool yang menghubungkan antara **objek dalam kode** (misalnya class, object di PHP/Java/Python) dengan **tabel di database relasional** (MySQL, PostgreSQL, dll).

Membuat Table

Pada terminal masukkan command

php artisan make:migration create_todos_table --create=todos



Otomatis akan membuat file baru di folder database/migrations, yang isinya seperti berikut

createtodostable.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('todos', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('todos');
    }
};
```

Penjelasan

- Pada function up digunakan untuk menjalankan migration
    - Pada function ini akan membuat table bernama todos
    - Membuat id dan juga timestamp
- Pada function down digunakan untuk reverse migration

Modifikasi file migration tersebut menjadi

createtodostable.php

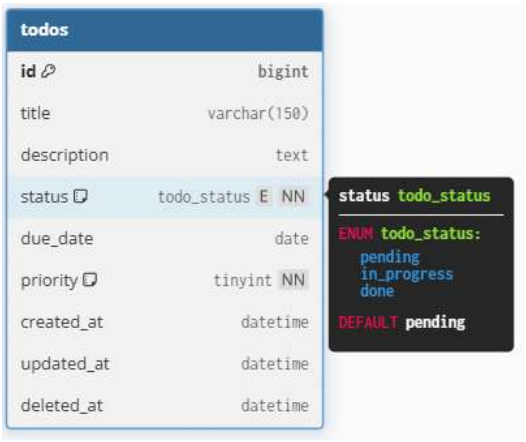
```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('todos', function (Blueprint $table) {
            $table->id();
            $table->string('title', 150);
            $table->text('description')->nullable();
            $table->enum('status', ['pending','in_progress','done'])->default('pending');
            $table->date('due_date')->nullable();
            $table->unsignedTinyInteger('priority')->default(3); // 1=high, 2=med, 3=low
            $table->timestamps();
            $table->softDeletes();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('todos');
    }
};
```

Yang artinya kita akan membuat table seperti berikut



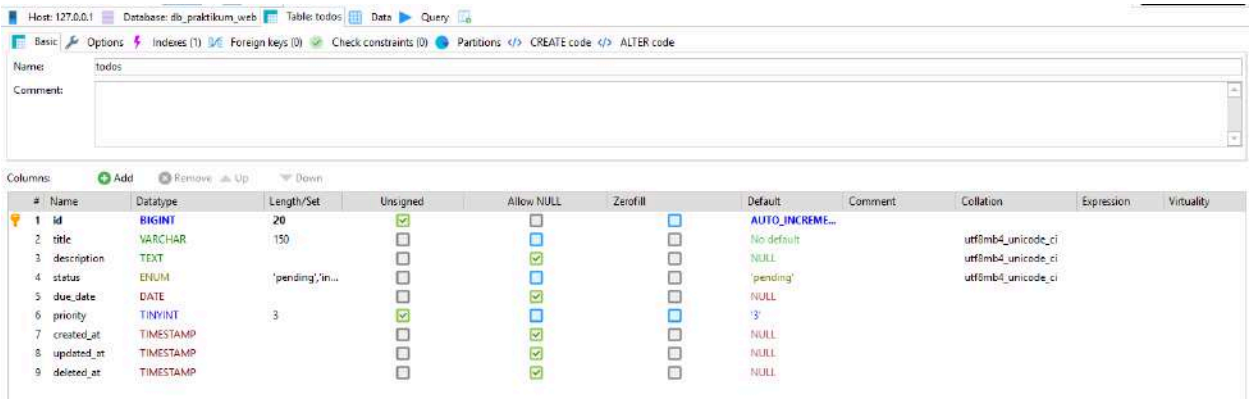
Jalankan migration dengan perintah
php artisan migrate

```
User@Device-K C: laragon www pemrograman-web main
❯ php artisan migrate

  INFO  Running migrations.

  2025_08_18_082041_create_todos_table ..... 380.73ms DONE
```

Jika berhasil log akan menampilkan seperti pada gambar diatas

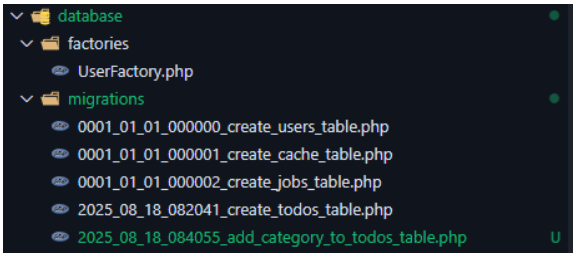


Kemudian buka HeidiSQL untuk mengecek apakah table sudah tertambah dan sudah benar.

Alter Table: Menambahkan Kolom Baru

Untuk alter table caranya sama seperti membuat table baru, namun perbedaannya adalah flag pada pembuatan migration. Contoh ingin menambahkan kolom baru bernama **category,** jalankan perintah
php artisan make:migration add_category_to_todos_table --table=todos

Pada perintah diatas akan membuat file migration baru dengan schema table todos bukan create.



Ubah file migration yang baru menjadi seperti di bawah

altertodostable.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('todos', function (Blueprint $table) {
            $table->enum('category', ['personal','work','study','others'])
            ->default('personal')
            ->after('priority');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('todos', function (Blueprint $table) {
            $table->dropColumn('category');
        });
    }
};
```

Penjelasan:

- Menambahkan kolom baru bernama **category**
- Membuat table category berjenis enum ['personal','work','study','others']
- Default valuenya adalah personal
- Kolom category akan ditambahkan setelah kolom priority

Jalankan migration dengan perintah

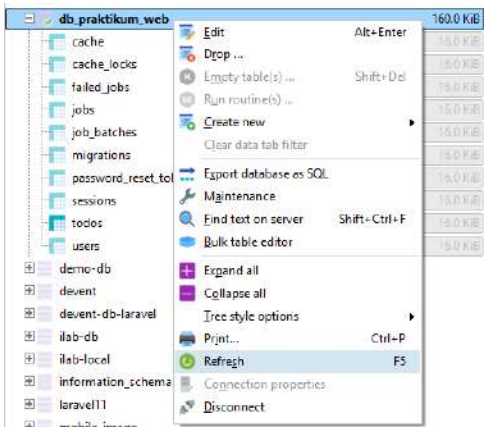
php artisan migrate

```
User@Device-K C: laragon www pemrograman-web main
> php artisan migrate

  INFO  Running migrations.

  2025_08_18_084055_add_category_to_todos_table ..... 36.67ms DONE
```

Jika berhasil log akan menampilkan seperti pada gambar diatas



Refresh database dengan klik kanan database, dan refresh

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow NULL	Zerofill	Default
1	id	BIGINT	20	☑	☐	☐	AUTO_INCREME...
2	title	VARCHAR	150	☐	☐	☐	No default
3	description	TEXT		☐	☑	☐	NULL
4	status	ENUM	'pending','in_progress','done'	☐	☐	☐	'pending'
5	due_date	DATE		☐	☑	☐	NULL
6	priority	TINYINT	3	☑	☐	☐	'3'
7	category	ENUM	'personal','work','study','others'	☐	☐	☐	'personal'
8	created_at	TIMESTAMP		☐	☑	☐	NULL
9	updated_at	TIMESTAMP		☐	☑	☐	NULL
10	deleted_at	TIMESTAMP		☐	☑	☐	NULL

Isi dari table todos akan berubah menambahkan kolom category

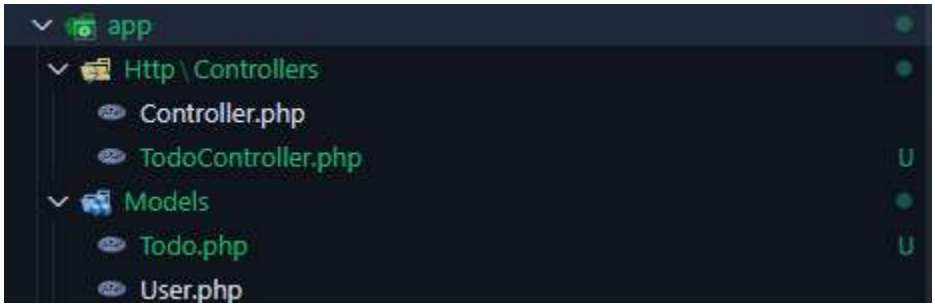
GENERATE MODEL DAN CONTROLLER

Pada laravel kita tidak perlu membuat file manual di project kita, cukup dengan menggunakan cli yang akan meng-generate model dan controller kita. Contoh untuk membuat model dan controller

- Membuat Model dengan command
  php artisan make:model Todo

- Membuat Controller dengan command
  php artisan make:controller TodoController --api --model=Todo
  Membuat controller TodoController dan berisikan function yang dibutuhkan API (CRUD) dengan model Todo yang sebelumnya dibuat

- Struktur Folder

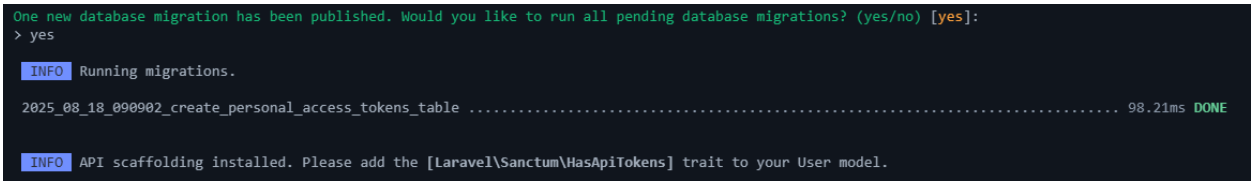


Main folder dari controller dan model kita berada di folder app/ dan TodoController berada di folder Http/Controllers sedangkan model Todo berada di folder Models/. Ketika kalian generate sebelumnya file otomatis akan seperti diatas

API Routing

Agar controller kita bisa diakses dan terdaftar di routing kita perlu menambahkan api route dengan command berikut:
php artisan install:api

Silahkan tunggu dan ketika ada permintaan seperti gambar dibawah ikuti gambar



Jika Log sudah seperti diatas, dipastikan install api sudah berhasil dan pada folder routes akan bertambah satu file baru bernama api.php, semua routing API kita akan kita route di file ini

Modifikasi file api.php seperti berikut

api.php

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\TodoController;

Route::get('/user', function (Request $request) {
    return $request->user();
})->middleware('auth:sanctum');
Route::apiResource('todos', TodoController::class);
```

Penjelasan

- Kita perlu menambahkan **use App\Http\Controllers\TodoController;** untuk memanggil class controller, sudah dijelaskan di modul sebelumnya
- Pada code **Route::apiResource('todos', TodoController::class);** artinya kita mendaftarkan semua function kita ke route **[index, store, show, update, destroy]**

Jika routing sudah berhasil kita bisa memvalidasi dengan menggunakan perintah ***php artisan route:list***

```
> php artisan route:list

GET|HEAD        / ..... 
GET|HEAD        api/todos ..... todos.index › TodoController@index
POST            api/todos ..... todos.store › TodoController@store
GET|HEAD        api/todos/{todo} ..... todos.show › TodoController@show
PUT|PATCH       api/todos/{todo} ..... todos.update › TodoController@update
DELETE          api/todos/{todo} ..... todos.destroy › TodoController@destroy
GET|HEAD        api/user ..... 
GET|HEAD        sanctum/csrf-cookie ..... sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieController@show
GET|HEAD        storage/{path} ..... storage.local
GET|HEAD        up .....
```

Bisa dilihat pada gambar terdapat list endpoint yang dapat kita gunakan, dan controller todo sudah terdaftar di route dengan endpoint **api/todos**

MEMBUAT ENDPOINT (CRUD)

Endpoint API didasari method yang sebelumnya sudah dibahas pada section Method. Hal yang pertama dilakukan adalah memodifikasi model kita mengikuti dengan table migration yang sudah dibuat sebelumnya.

Modifikasi Model Todo.php seperti dibawah

todo.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Todo extends Model
{
    use SoftDeletes;

    protected $fillable = [
        'title', 'description', 'status', 'due_date', 'priority', 'category'
    ];

    protected $casts = [
        'due_date' => 'date',
        'priority' => 'integer',
    ];
}
```

Modifikasi file TodoController.php mengikuti code di bawah untuk masing-masing method.

GET ALL

todocontroller.php

```php
public function index(Request $req)
    {
        $q = Todo::query();

        if ($s = $req->query('search')) {
            $q->where(fn($qq)=>$qq->where('title','like',"%$s%")->orWhere('description','like',"%$s%"));
        }
        if ($status = $req->query('status'))   $q->where('status', $status);
        if ($cat    = $req->query('category')) $q->where('category', $cat);
        if ($prio   = $req->query('priority')) $q->where('priority', $prio);

        $q->latest('created_at');

        $todos = $q->paginate($req->integer('limit', 10));
        return response()->json($todos);
    }
```

GET BY ID

todocontroller.php

```php
public function show(Todo $todo)
    {
        return response()->json($todo);
    }
```

POST

todocontroller.php

```php
public function store(Request $request)
    {
        $data = $request->validate([
            'title'       => 'required|string|max:150',
            'description' => 'nullable|string',
            'status'      => 'nullable|in:pending,in_progress,done',
            'due_date'    => 'nullable|date',
            'priority'    => 'nullable|integer|min:1|max:3',
            'category'    => 'nullable|in:personal,work,study,others',
        ]);

        $todo = Todo::create($data);
        return response()->json($todo, 201);
    }
```

PATCH

todocontroller.php

```php
public function update(Request $request, Todo $todo)
    {
        $data = $request->validate([
            'title'       => 'sometimes|required|string|max:150',
            'description' => 'sometimes|nullable|string',
            'status'      => 'sometimes|in:pending,in_progress,done',
            'due_date'    => 'sometimes|nullable|date',
            'priority'    => 'sometimes|integer|min:1|max:3',
            'category'    => 'sometimes|in:personal,work,study,others',
        ]);

        $todo->update($data);
        return response()->json($todo);
    }
```

DELETE

todocontroller.php

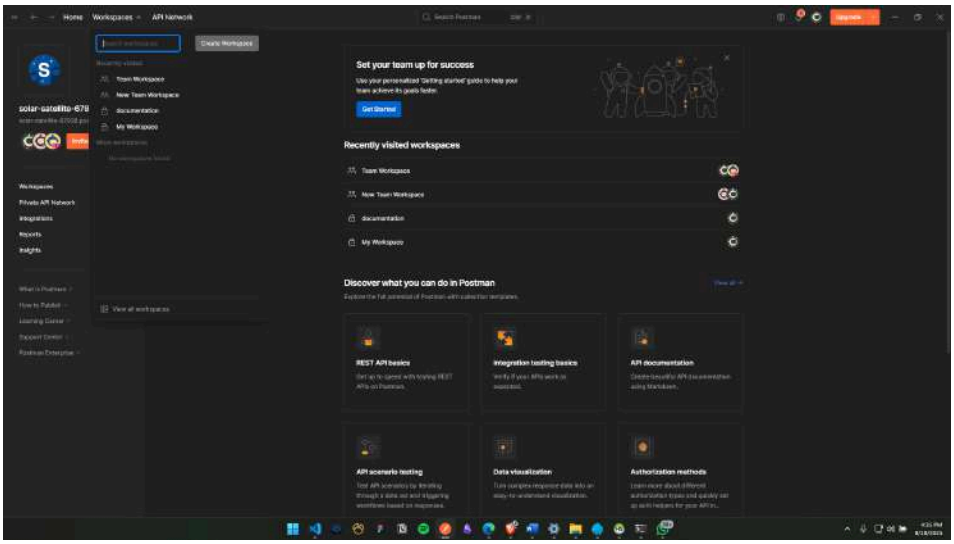
```php
public function destroy(Todo $todo)
    {
        $todo->delete();

        return response()->json([
            'message' => 'Todo berhasil dihapus',
            'id'      => $todo->id
        ], 200);
    }
```

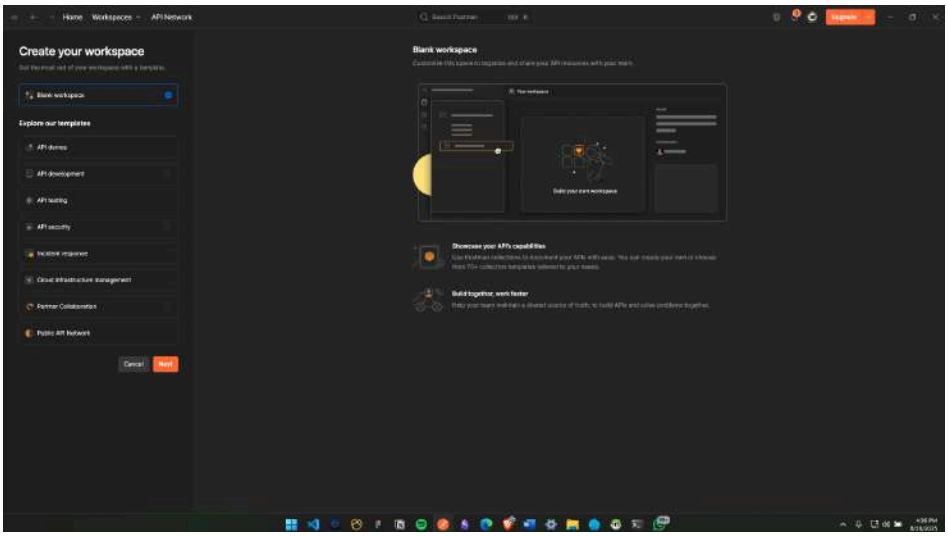
TESTING API

Dari API yang sudah kita buat di laravel tadi, kita akan coba testing menggunakan aplikasi **postman.** Postman tidak hanya berguna untuk testing api tapi dapat juga membuat dokumentasi API dan juga berkolaborasi dengan tim.

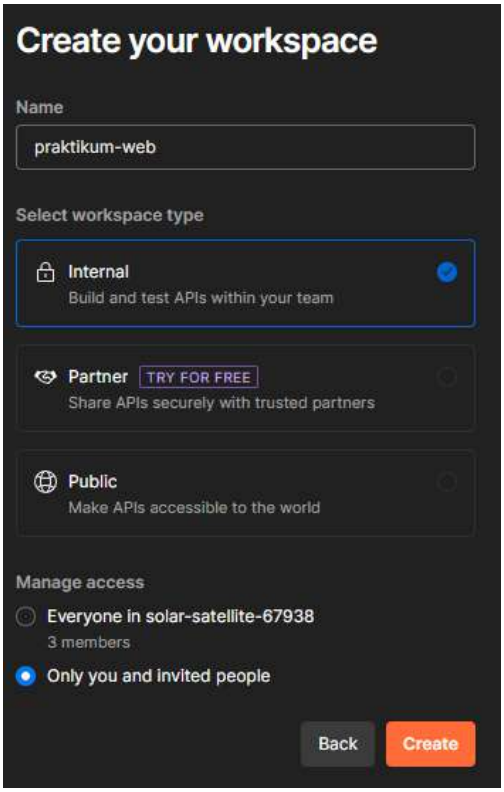
Membuat Workspace



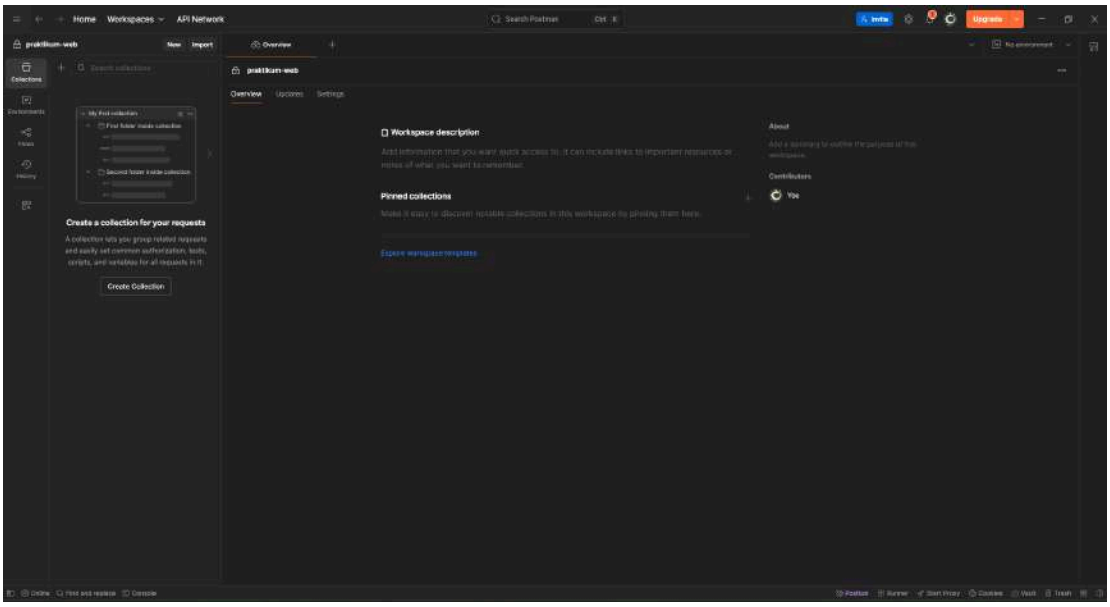
Pada tampilan awal postman di pojok kiri atas terdapat workspace, dan klik create workspace



Next saja, biarkan default *(blank workspace)*



Untuk nama bisa bebas, tapi lowercase semua dan tidak ada spasi untuk akses bisa diikuti pada gambar, dan tekan tombol create

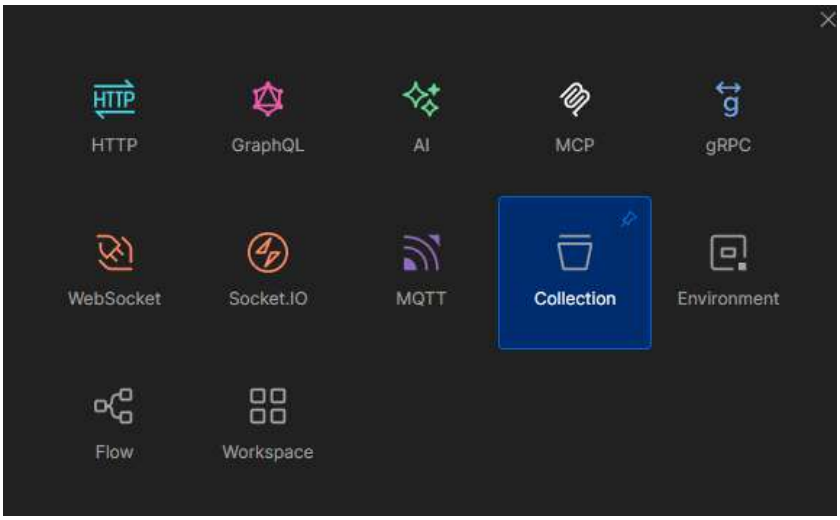


Jika tampilan sudah seperti diatas berarti berhasil membuat workspace

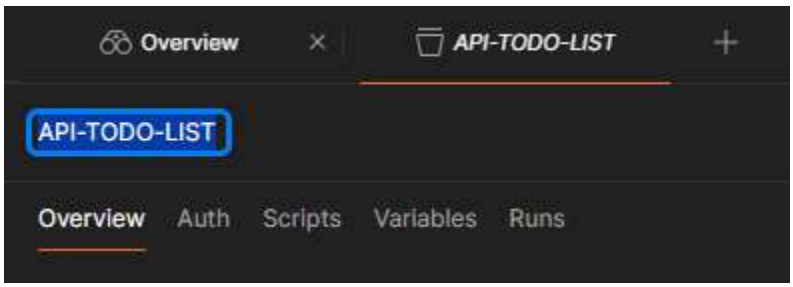
Membuat Collection(folder)

Agar API kita tersusun rapi, kita butuh untuk membuat collection atau seperti folder
Pada kiri atas di nama workspace tekan New

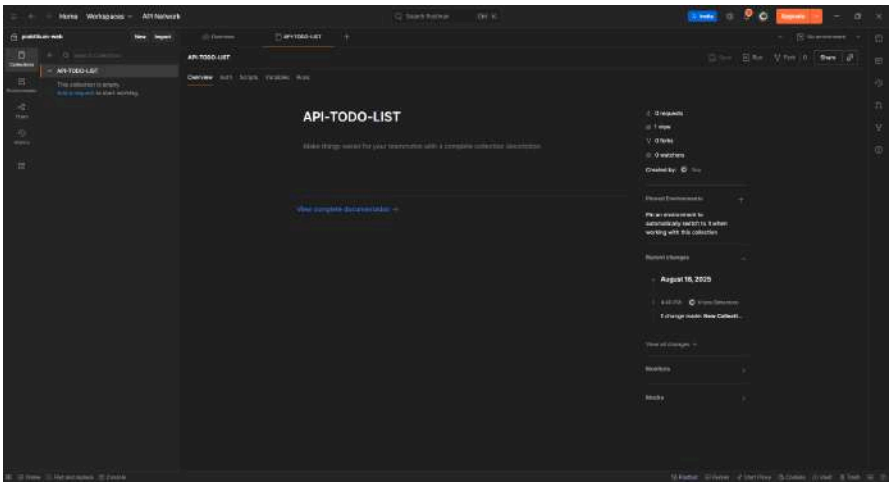
Pilih Collection



Ubah nama collection seperti gambar dibawah

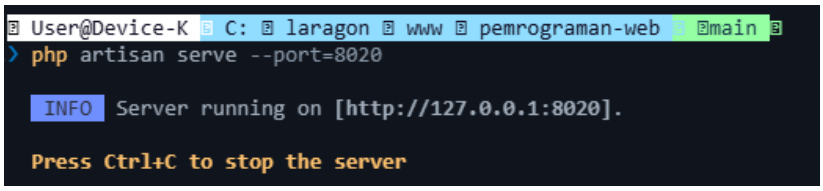


Jika tampilan sudah seperti gambar di bawah sudah berhasil membuat collection

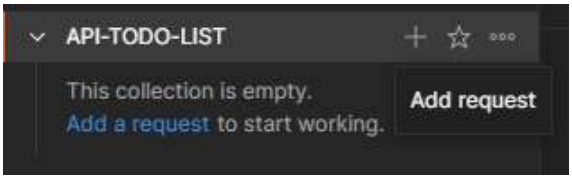


Membuat Method Request

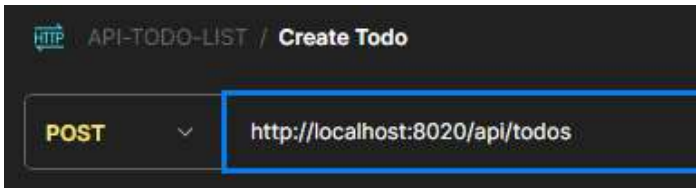
Sebelum melakukan request, laravel kita jalankan dahulu dengan perintah dibawah atau bisa mengikuti pada materi instalasi laravel



Pada Bagian kiri tampilan postman tambahkan request dengan menekan tombol + seperti gambar dibawah

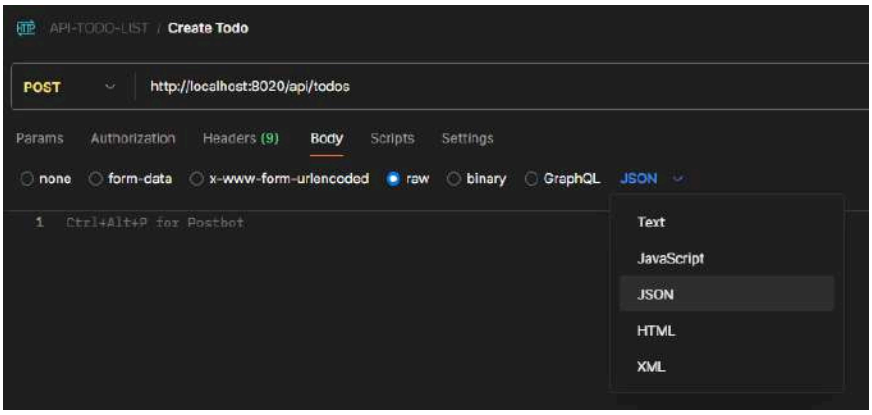


Membuat Request POST

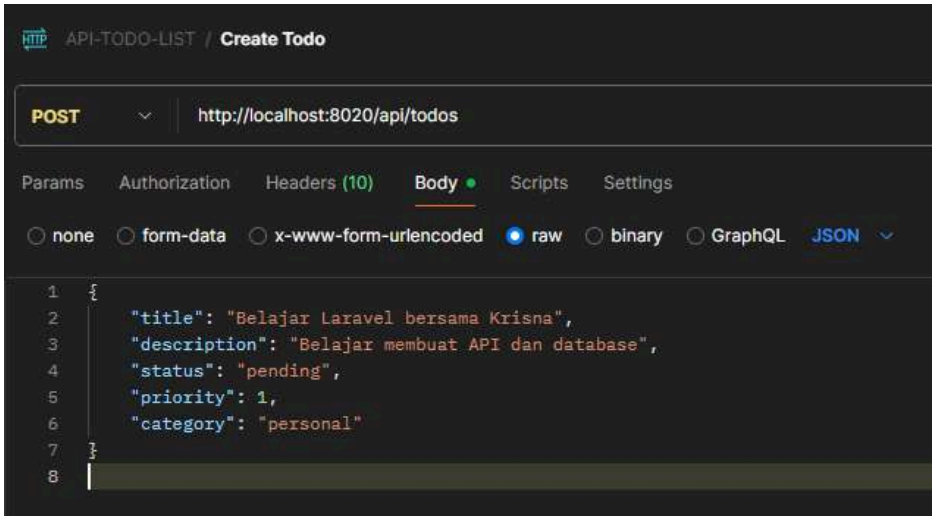


Sesuaikan request seperti di gambar

- Mengubah method menjadi POST
- Mengubah nama request menjadi Create Todo
- Mengubah URL menjadi http://localhost:8020/api/todos mengikuti port saat kalian jalankan laravel sebelumnya



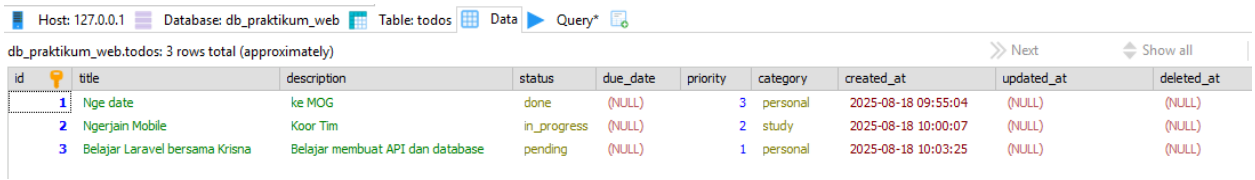
- Pindah tab ke Body, karena Post memerlukan Request Body
- Select ke raw dan pilih tipenya ke JSON



Isi dengan json seperti di atas

Pada bagian kanan URL terdapat tombol Send berwarna Biru bisa diklik untuk melakukan request

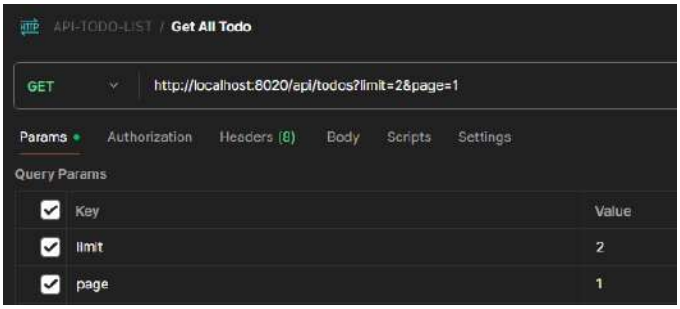
Ketika sukses akan seperti gambar diatas dengan return code 201, mari kita cek pada database



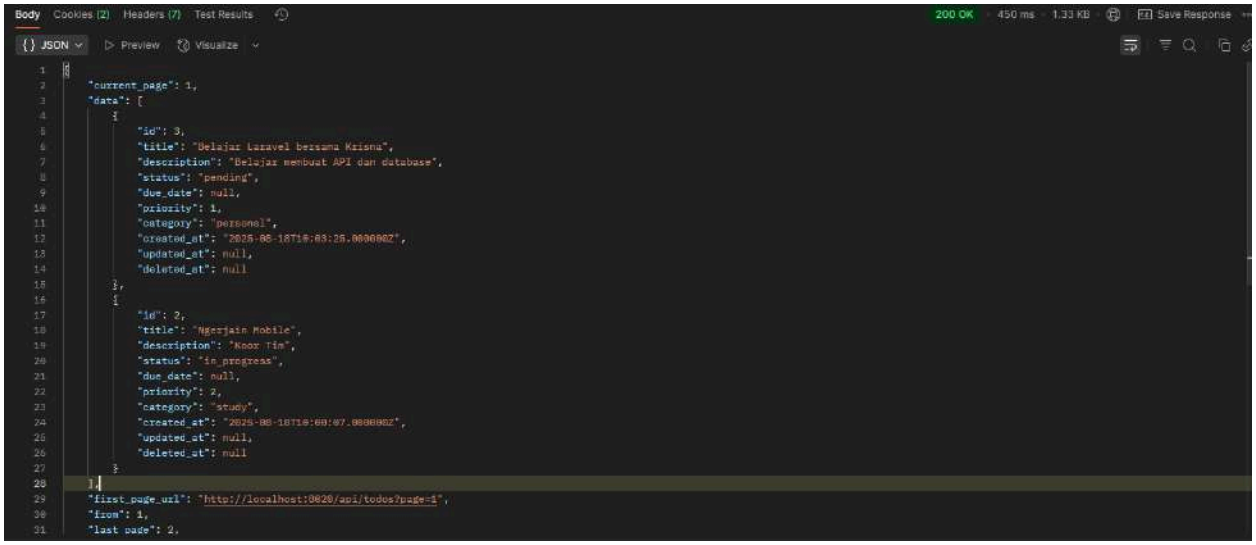
id	title	description	status	due_date	priority	category	created_at	updated_at	deleted_at
1	Nge date	ke MOG	done	(NULL)	3	personal	2025-08-18 09:55:04	(NULL)	(NULL)
2	Ngerjain Mobile	Koor Tim	in_progress	(NULL)	2	study	2025-08-18 10:00:07	(NULL)	(NULL)
3	Belajar Laravel bersama Krisna	Belajar membuat API dan database	pending	(NULL)	1	personal	2025-08-18 10:03:25	(NULL)	(NULL)

Data berhasil ditambahkan pada database, pada gambar terdapat 3 data karena saya melakukan request sebanyak 3 kali, **silahkan lakukan POST sebanyak 3 kali dengan value berbeda.**

Membuat Request GET ALL



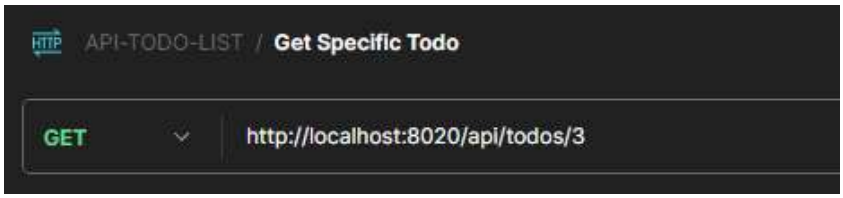
Buat request baru seperti diatas



Response diatas sudah memiliki pagination

Membuat Request GET BY ID

Buat request baru dan sesuaikan seperti dibawah, kita akan menampilkan data dengan **id 3**

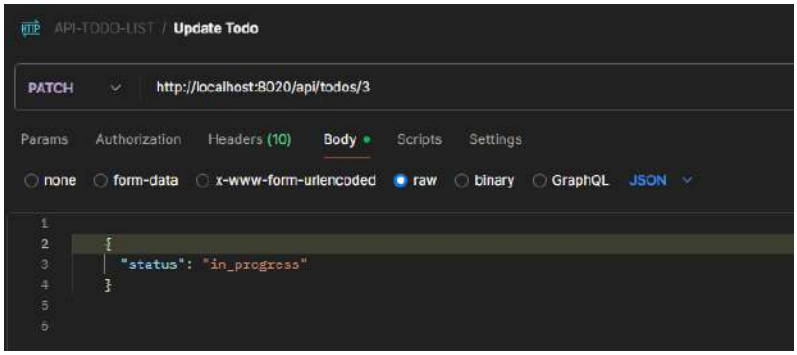


Klik send

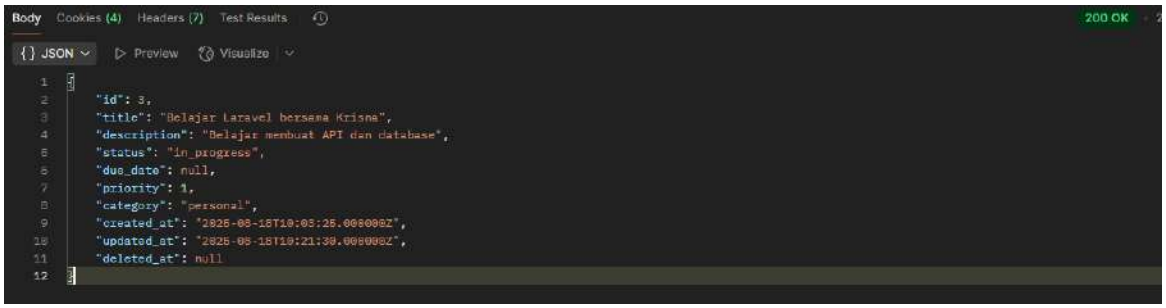
Response seperti diatas

Membuat Request PATCH

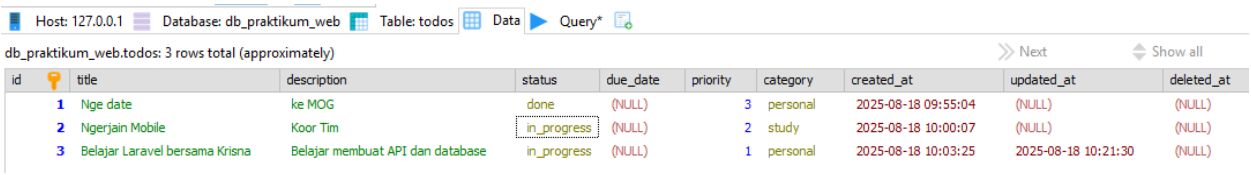
Buat request baru dan sesuaikan seperti dibawah,kita akan mengupdate data dengan **id 3**



Klik send



Response jika sukses seperti diatas

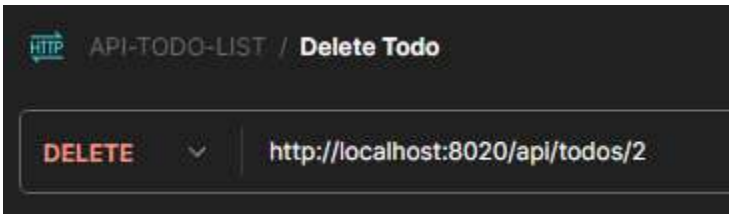


id	title	description	status	due_date	priority	category	created_at	updated_at	deleted_at
1	Nge date	ke MOG	done	(NULL)	3	personal	2025-08-18 09:55:04	(NULL)	(NULL)
2	Ngerjain Mobile	Koor Tim	in_progress	(NULL)	2	study	2025-08-18 10:00:07	(NULL)	(NULL)
3	Belajar Laravel bersama Krisna	Belajar membuat API dan database	in_progress	(NULL)	1	personal	2025-08-18 10:03:25	2025-08-18 10:21:30	(NULL)

Pada database sudah berhasil terupdate status pada id 3 menjadi in_progress

Membuat Request DELETE

Buat request baru dan sesuaikan seperti dibawah, kita akan menghapus data dengan **id 2**



Klik send

Response akan seperti diatas

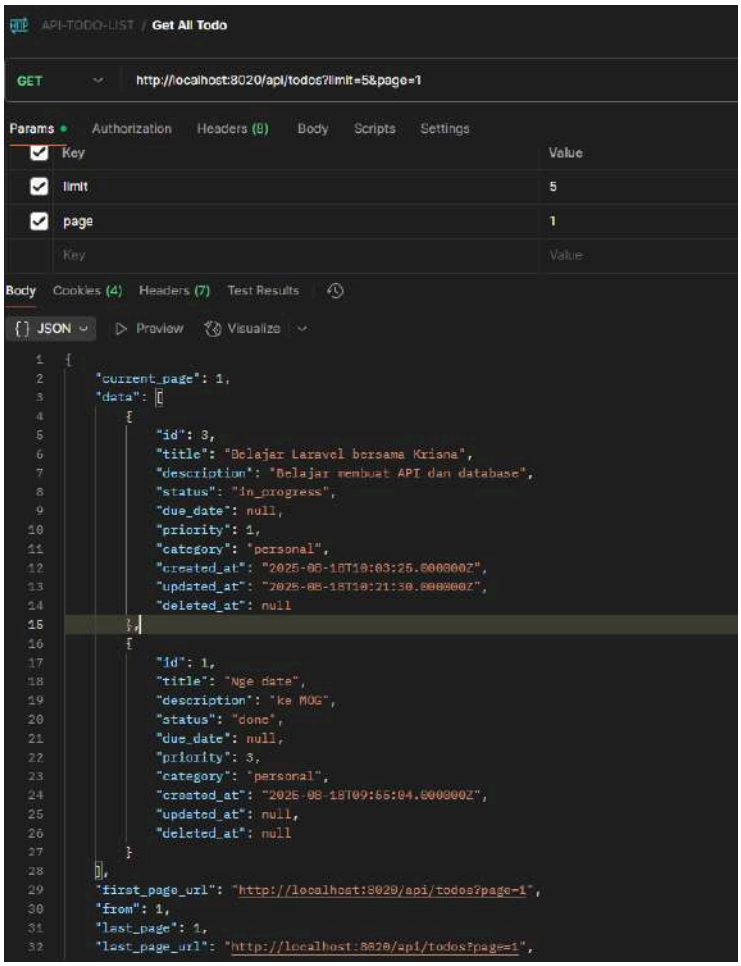
Host: 127.0.0.1 Database: db_praktikum_web Table: todos Data Query*

db_praktikum_web.todos: 3 rows total (approximately)

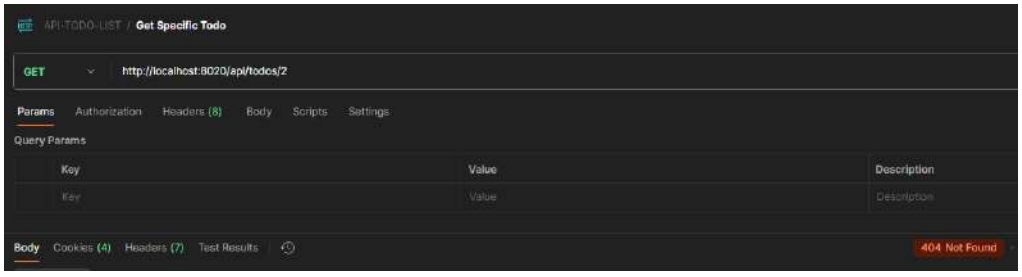
Next Show all Sor

id	title	description	status	due_date	priority	category	created_at	updated_at	deleted_at
1	Nge date	ke MOG	done	(NULL)	3	personal	2025-08-18 09:55:04	(NULL)	(NULL)
2	Ngerjain Mobile	Koor Tim	in_progress	(NULL)	2	study	2025-08-18 10:00:07	2025-08-18 10:26:58	2025-08-18 10:26:58
3	Belajar Laravel bersama Krisna	Belajar membuat API dan database	in_progress	(NULL)	1	personal	2025-08-18 10:03:25	2025-08-18 10:21:30	(NULL)

Bisa dilihat pada database data tidak di drop namun terdapat timestamp deleted_at, ini berguna ketika ingin merestore data yang tidak sengaja terdelete cukup dengan mengubah value deleted_at menjadi null kembali.



Data yang di delete tadi tidak akan tampil di endpoint Get All



Ketika dicoba menggunakan Get by ID data Not Found return 404

CODELAB

1. Lakukan step instalasi composer dan laravel seperti contoh pada modul.
2. Setup database seperti modul
3. Buatlah API dan database todo list yang dicontohkan pada modul (cukup ikuti step by step pada modul)
4. Buatlah collection postman dengan contoh seperti pada modul.
5. Pada pekan materi silahkan tunjukan bahwa laravel dapat berjalan, contoh pada **INSTALASI LARAVEL**
6. Penilaian Codelab:
    a. Jika memungkinkan untuk selesai codelab step 1-4 di lab pada pekan materi, dapat langsung ditunjukan kepada asistennya
    b. Jika tidak memungkinkan tunjukan pada pekan demo

Note: Step 5 codelab wajib ditunjukan kepada asisten pada pekan materi, jika tidak, codelab tidak dapat dinilai. Asisten dapat memberikan note bagi praktikan yang sudah selesai pada step 5 dan dapat memberikan nilai 100 bagi yang menyelesaikan step 6.

TUGAS

Buatlah API dan database sesuai dengan tema yang yang kalian gunakan dan implementasikan dengan ketentuan seperti berikut:

1. Database wajib terdapat table untuk menampung data (tergantung case masing-masing) dan tentukan tipe datanya.
2. Buatlah endpoint untuk 1 fitur utama, terdapat 5 endpoint yang harus dibuat yaitu
    a. Get All dengan pagination query (limit,page,search,orderBy,sortBy)
    b. Get Detail atau Get One dengan param id
    c. Create
    d. Update dengan param id
    e. Delete dengan param id
3. Buatlah Collection Postman dan request http untuk masing-masing endpoint.

KRITERIA & DETAIL PENILAIAN

Kriteria Penilaian	Persentase Penilaian
Codelab	**Total 15%**
Tugas	
Mengimplementasikan Database	10%
Membuat Endpoint API	25%
Membuat Collection Postman	15%
Pemahaman	
Kemampuan menjelaskan tugas dan menjawab pertanyaan	35%
Total	**Total 100%**