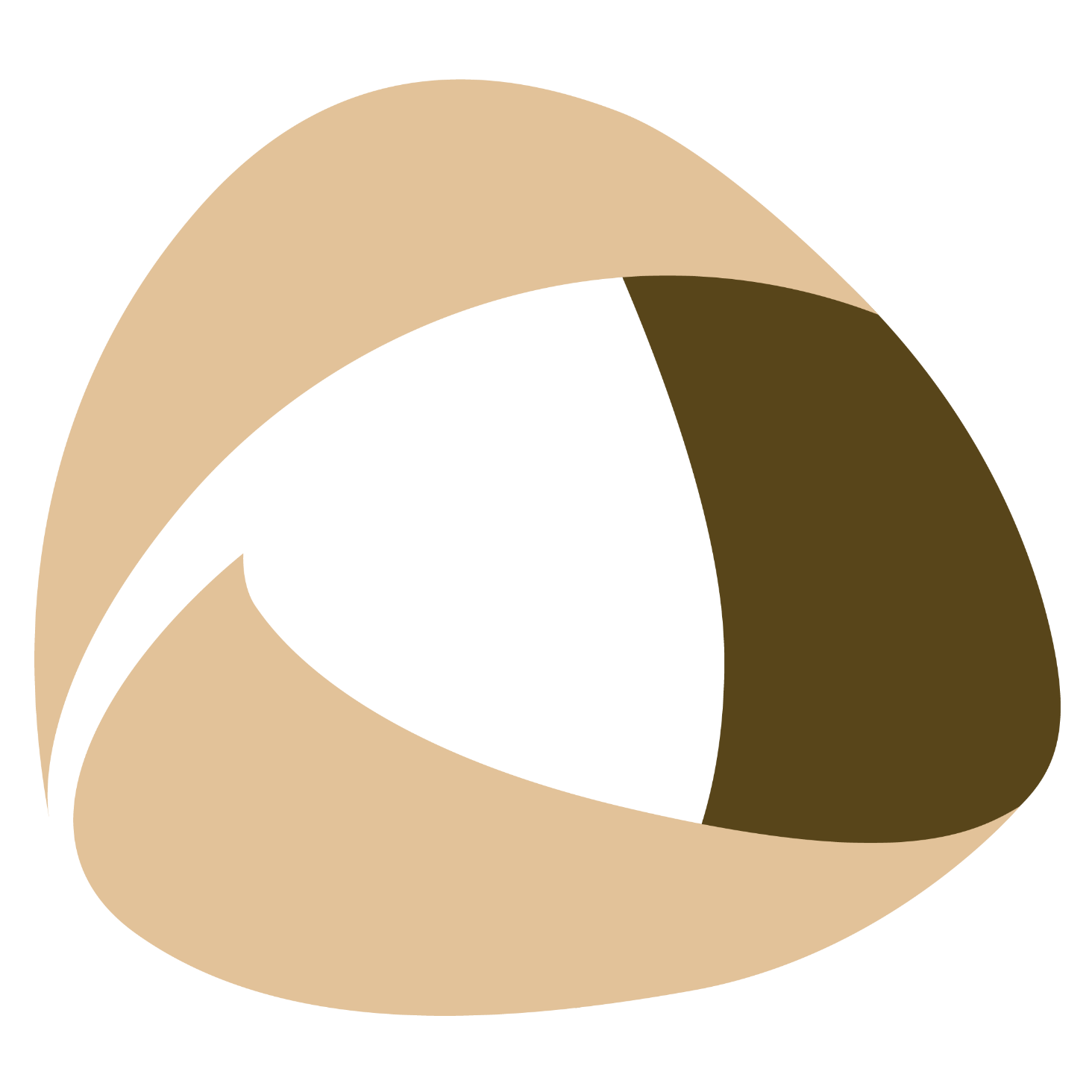
Modul

Membuat Aplikasi Klinik menggunakan Laravel 9

DAC Solution Training Center



Contents

[Getting Started 2](#_Toc119324687)

[Prerequisites 2](#_Toc119324688)

[Instalasi PHP 8 2](#_Toc119324689)

[Environment Variables 3](#_Toc119324690)

[Instalasi Composer 4](#_Toc119324691)

[Instalasi Laravel 9 5](#_Toc119324692)

[Konfigurasi Koneksi Database 7](#_Toc119324693)

[Database Migrations 10](#_Toc119324694)

[Membuat Database Migration 10](#_Toc119324695)

[Database Seeding 12](#_Toc119324696)

[Exercises 14](#_Toc119324697)

[Blade Template 15](#_Toc119324698)

[Membuat View Layout 15](#_Toc119324699)

[Membuat View Index 17](#_Toc119324700)

[Mengirim Data ke View 20](#_Toc119324701)

# Getting Started

## Prerequisites

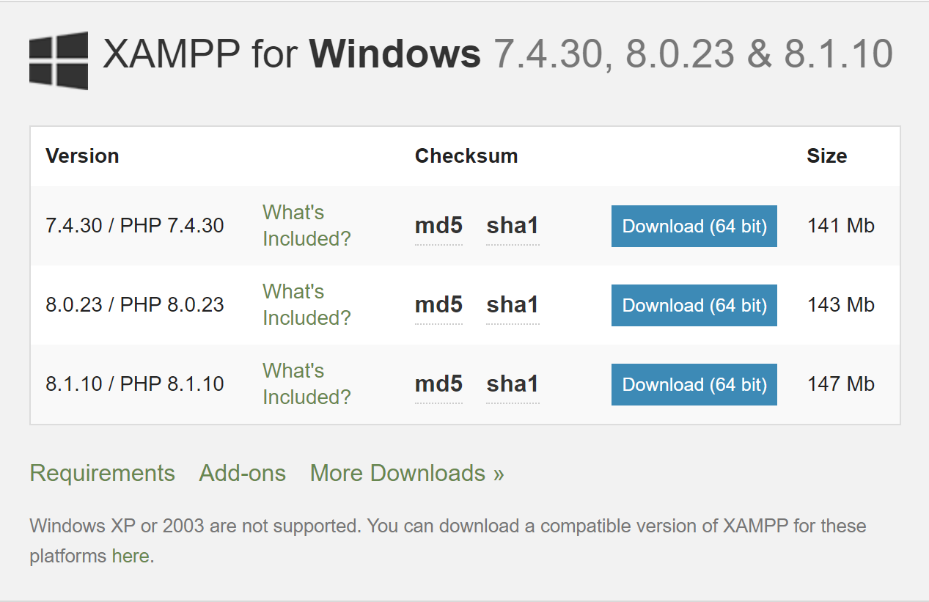
Sebelum membuat project Laravel 9, ada beberapa hal yang diperlukan untuk mendukung proses pembuatan project Laravel 9.

* Code Editor (Visual Studio Code, Sublime Text, dan lain sebagainya)
* PHP 8
* Composer
* Paham konsep pemrograman berorientasi object (OOP)

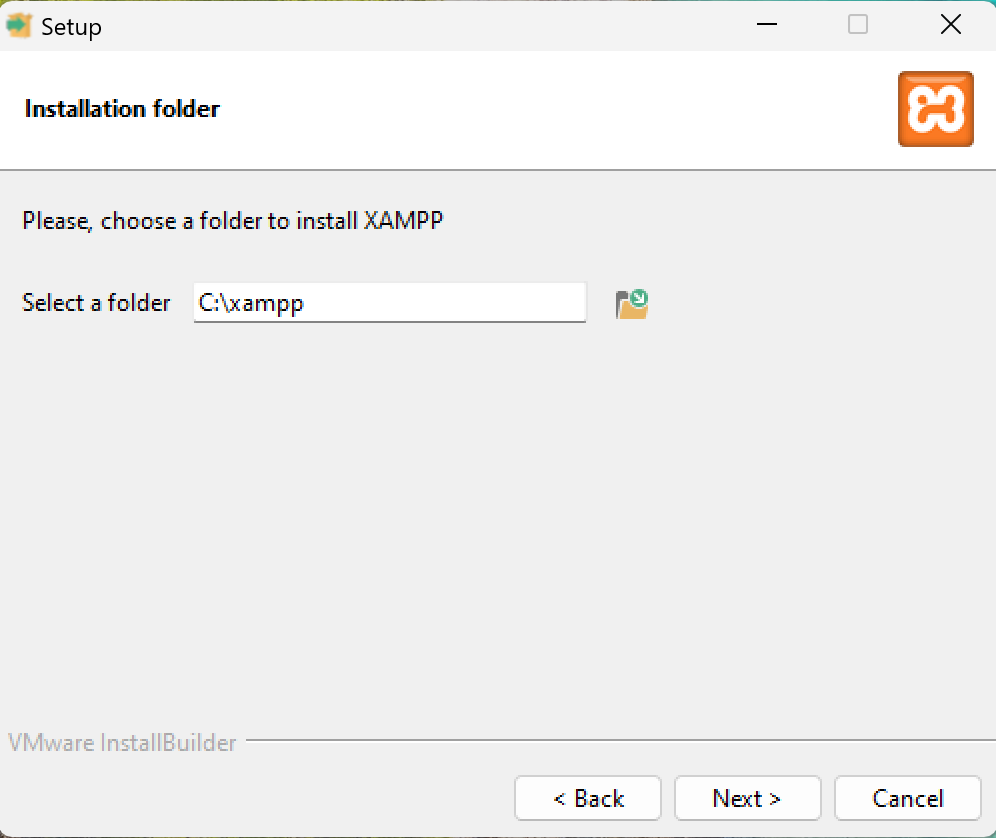
## Instalasi PHP 8

Untuk melakukan instalasi PHP 8, kita bisa mengunduh file installer-nya di situs **www.php.net**. Namun, biasanya jika kita ingin install PHP 8 di komputer kita, kita dapat menggunakan **XAMPP**.

* Buka situs **www.apachefriends.org/download.html**. Kemudian, pilih versi terbaru dan sistem operasi yang sesuai dengan komputer kita, lalu klik tombol **”Download (64 bit)”**.



* Buka file installer dan mulai install XAMPP.
* Klik tombol **Next >** hingga ke pemilihan lokasi instalasi. Secara default, lokasi file instalasi berada di **C:/xampp**.

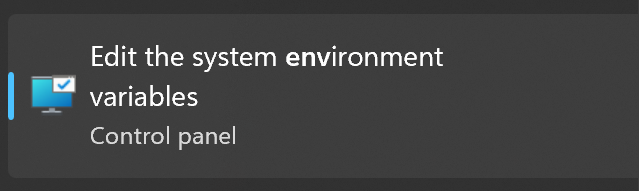


* Klik **Next >** sampai ke proses instalasi XAMPP. Jika sudah selesai, tekan tombol **Finish**.

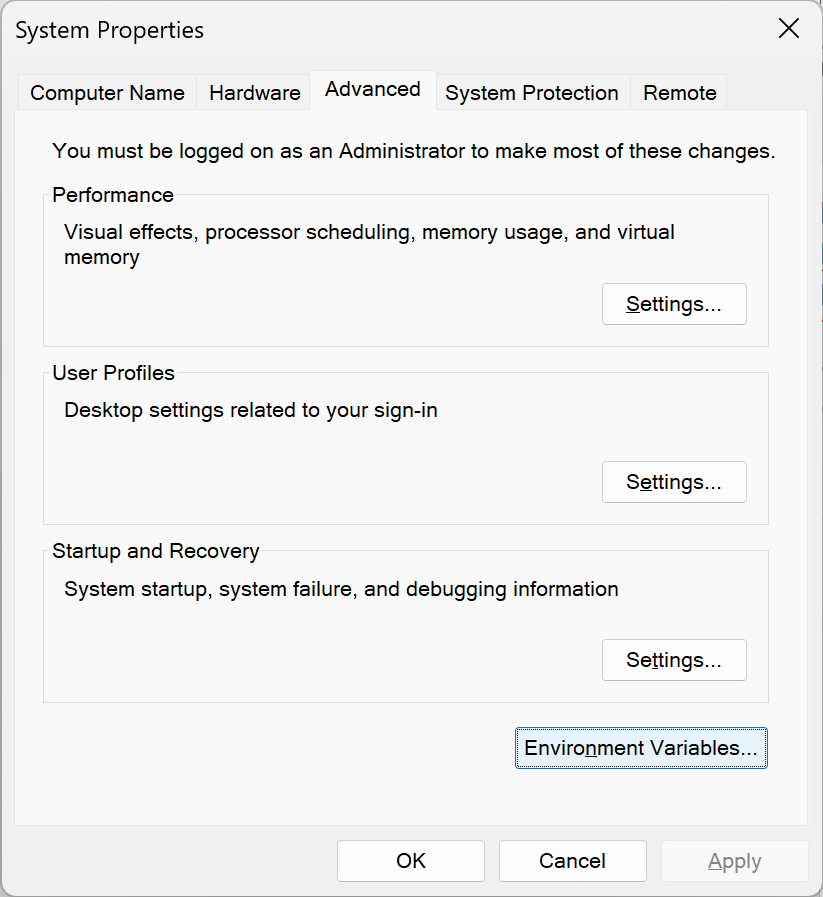
## Environment Variables

Setelah melakukan instalasi PHP 8 melalui XAMPP, komputer kita tetap belum bisa membaca PHP yang sudah terinstall di komputer kita. Untuk membuat komputer kita bisa membaca PHP-nya, kita perlu mengatur **Environment Variables**.

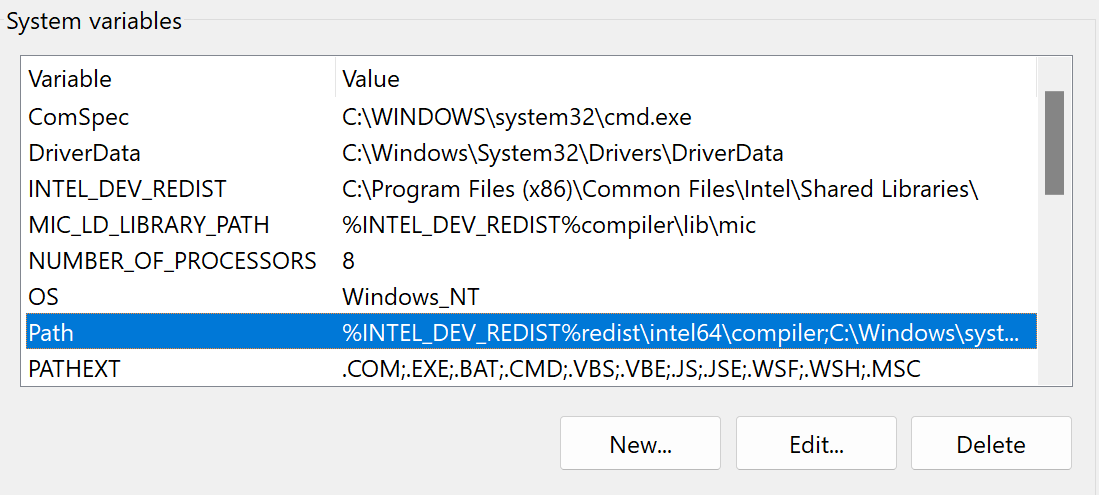
* Buka **Control Panel** pada komputer kita, atau ketikkan **“Edit the system environment variables”.**



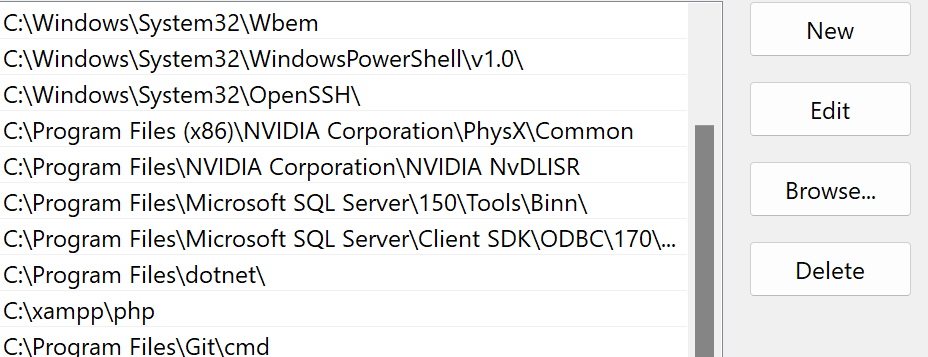
* Pada window System Properties, klik tombol **Environment Variables…** untuk menambahkan Path dari PHP 8.



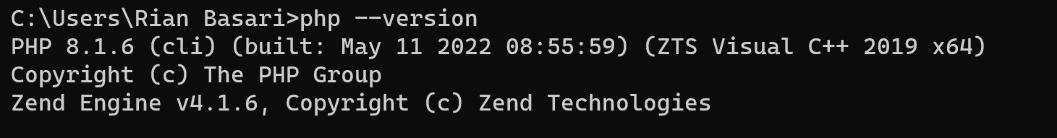
* Pada bagian System Variables, buka variable **Path.**



* Klik tombol **Browse** untuk menambahkan path dari PHP yang sudah kita install. Arahkan ke folder **C:/xampp/php.** Jika sudah, klik tombol **OK**.

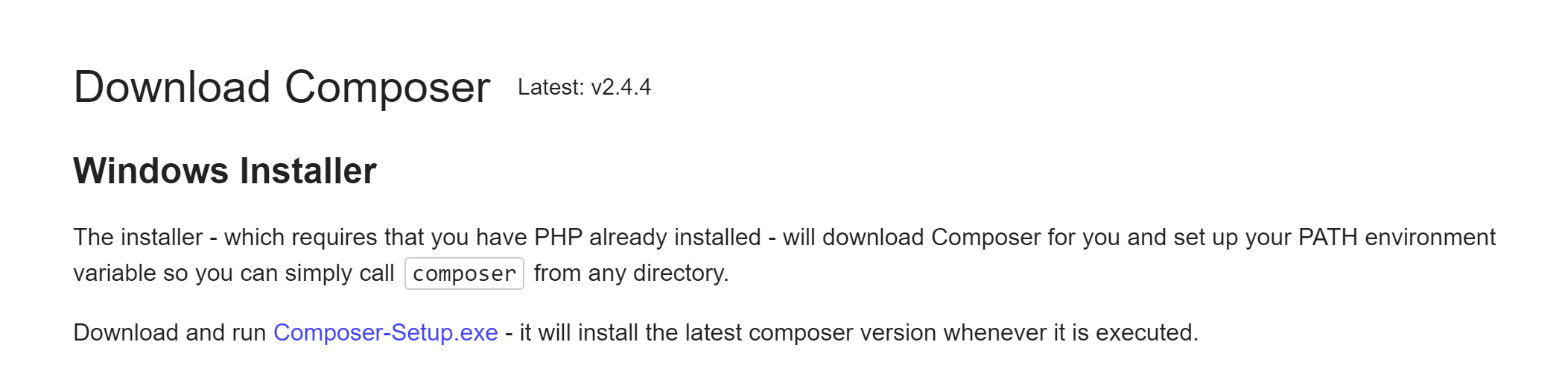


* Cek apakah PHP 8 sudah terinstall di komputer kita melalui **Command Prompt**  dengan command **php --version**. Jika sudah terinstall, akan muncul versi PHP yang sudah terinstall.

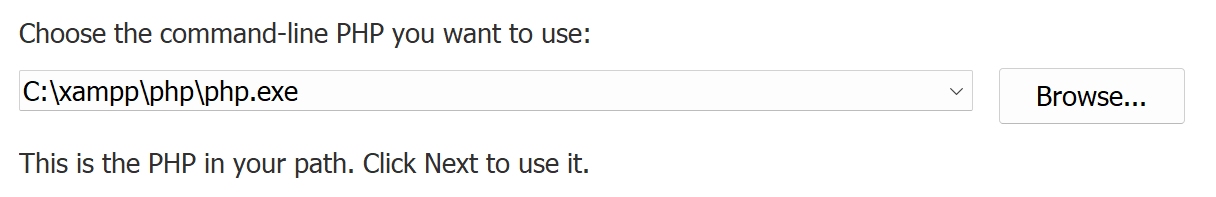


## Instalasi Composer

* Kunjungi situs **getcomposer.org/download** untuk mengunduh file installer **Composer**.
* Pilih instalasi dengan **Windows Installer** agar proses instalasi lebih mudah.
* Klik pada link yang tertulis **Composer-Setup.exe** untuk mulai mengunduh Composer.



* Install file installer seperti biasa dan klik tombol **Next >** sampai ke pemilihan PHP yang akan digunakan oleh Composer.
* Pastikan pilih PHP dari XAMPP yang sudah kita install sebelumnya. **C:\xampp\php\php.exe**

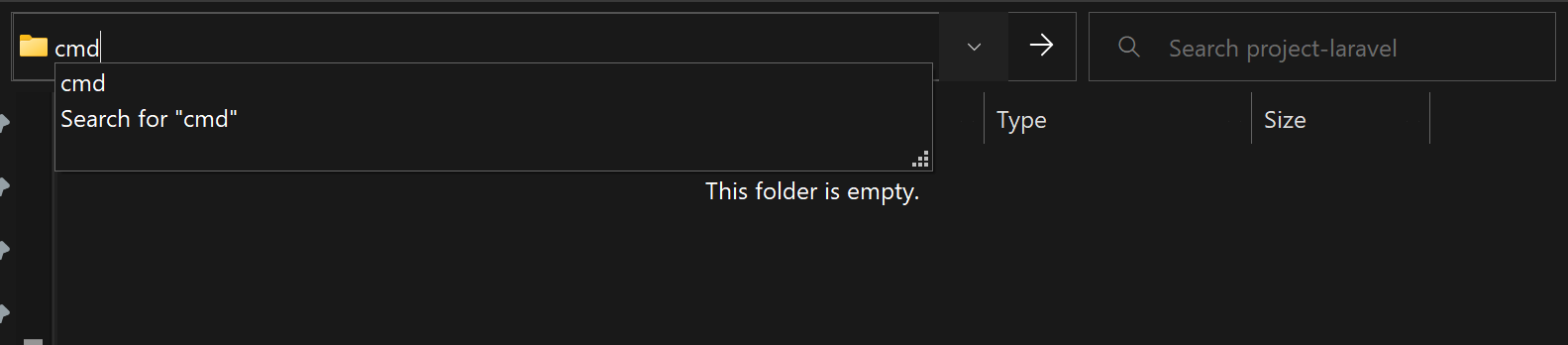


* Klik **Next >** sampai ke proses instalasi. Tunggu proses instalasi selesai, kemudian klik tombol **Finish**.
* Cek Composer menggunakan **Command Prompt** dengan command **composer**.

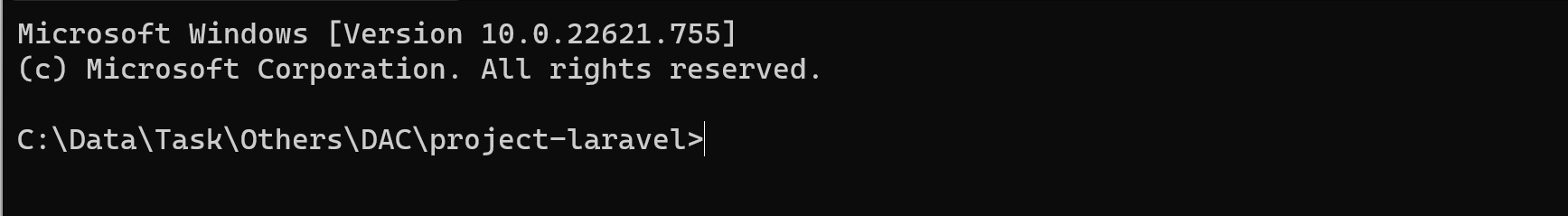
## Instalasi Laravel 9

Untuk membuat project Laravel 9 baru, silahkan tentukan lokasi folder atau drive dimana project Laravel 9 akan di-install.

* Arahkan File Manager ke folder tersebut dan ketik **“cmd”** pada address bar dari folder tersebut. Jika di tekan **Enter**, maka akan terbuka window **Command Prompt.**



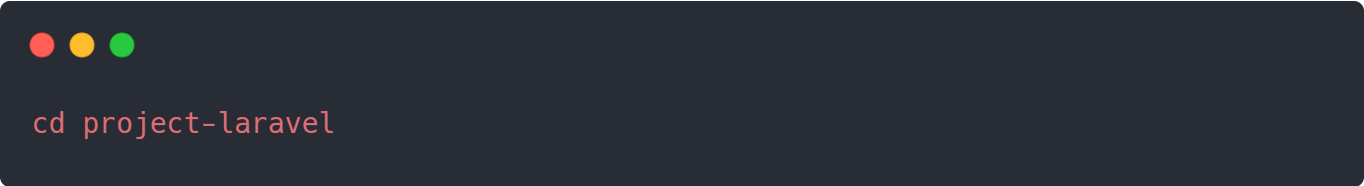
* Pastikan lokasi Command berada di folder yang sudah kita buat.



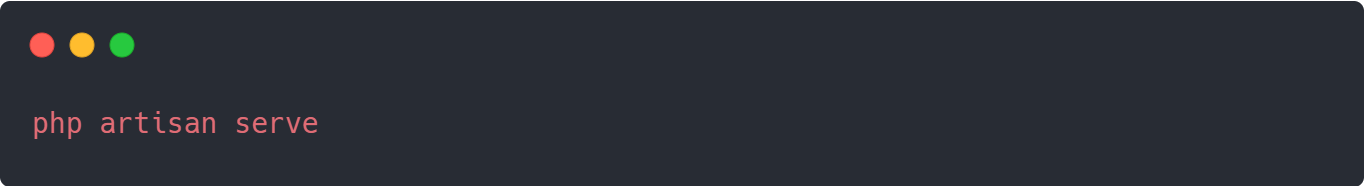
* Install Laravel 9 menggunakan Composer dengan command :

**project-laravel** merupakan nama dari project kita. Pastikan untuk penamaan project tidak mengandung spasi.

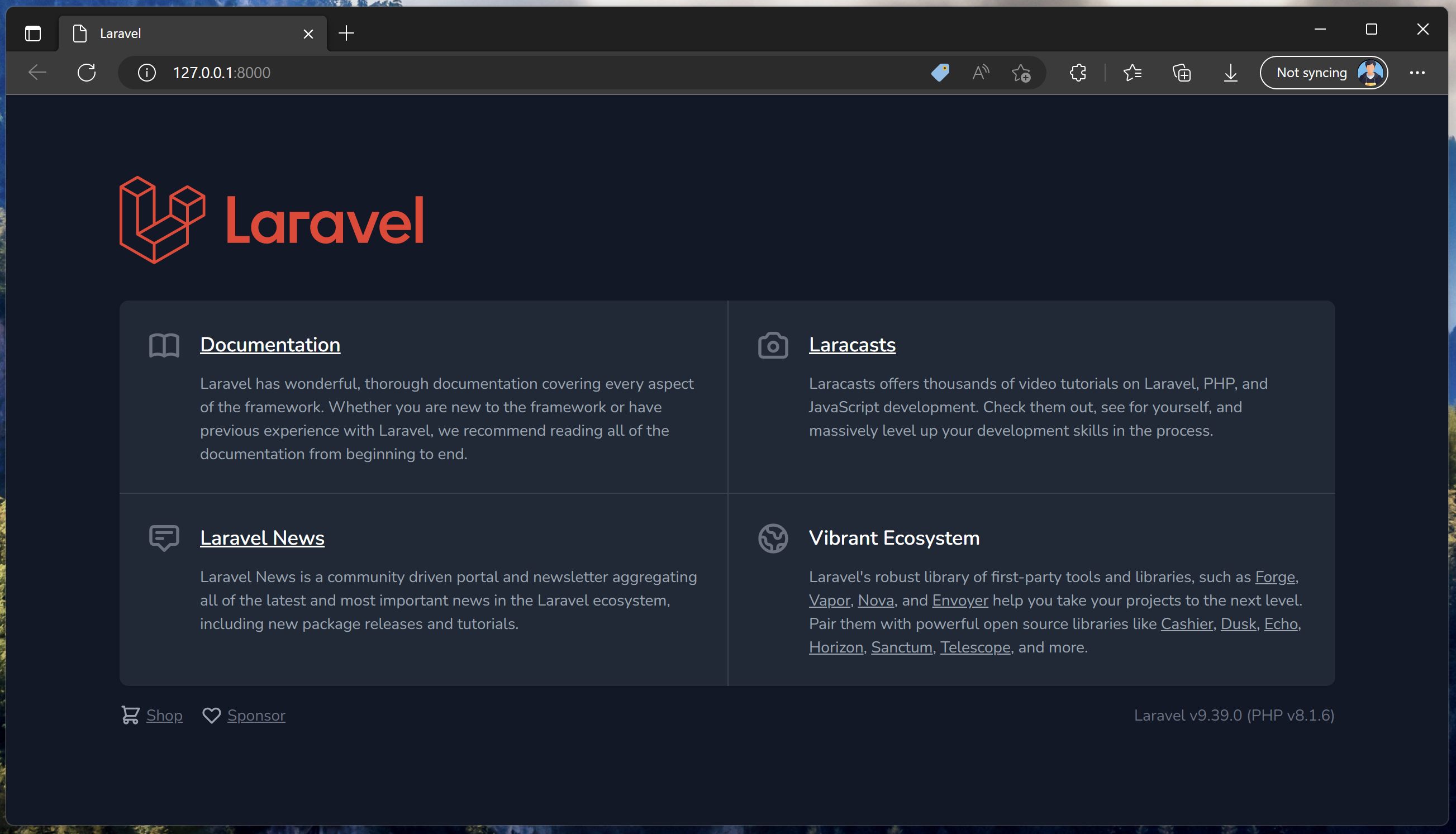
* Masuk ke dalam project yang sudah terinstall Laravel 9 dengan Command :



* Jalankan server lokal Laravel 9 untuk memastikan project Laravel 9 sudah terinstall dan berjalan dengan baik. Untuk menjalankan server Laravel 9, gunakan command :



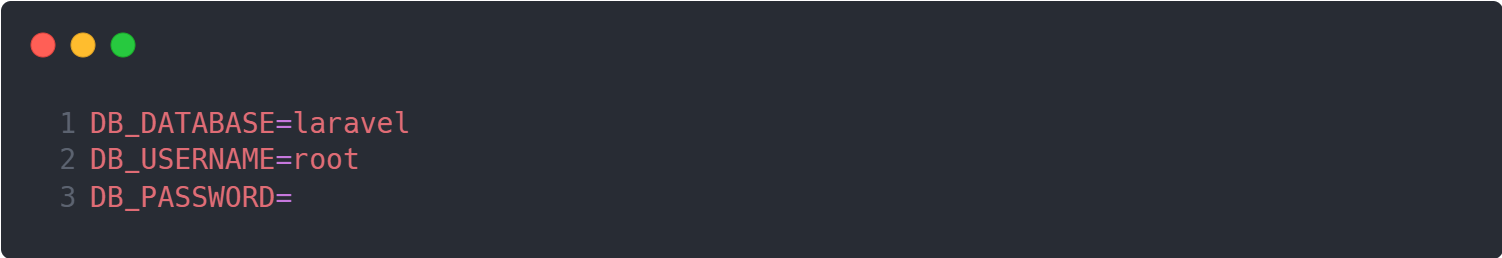
* Buka alamat server Laravel 9 **http://127.0.0.1:8000** akan muncul tampilan awal dari project Laravel 9.



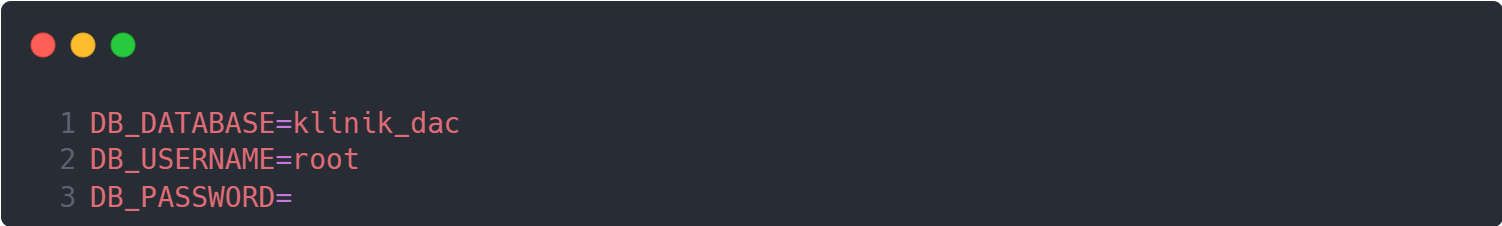
## Konfigurasi Koneksi Database

Karena akan bekerja dengan data, maka kita perlu menyiapkan sebuah database untuk menyimpan data-data tersebut secara dinamis. Untuk mengatur koneksi database apada Laravel 9, pergi ke file **.env**.

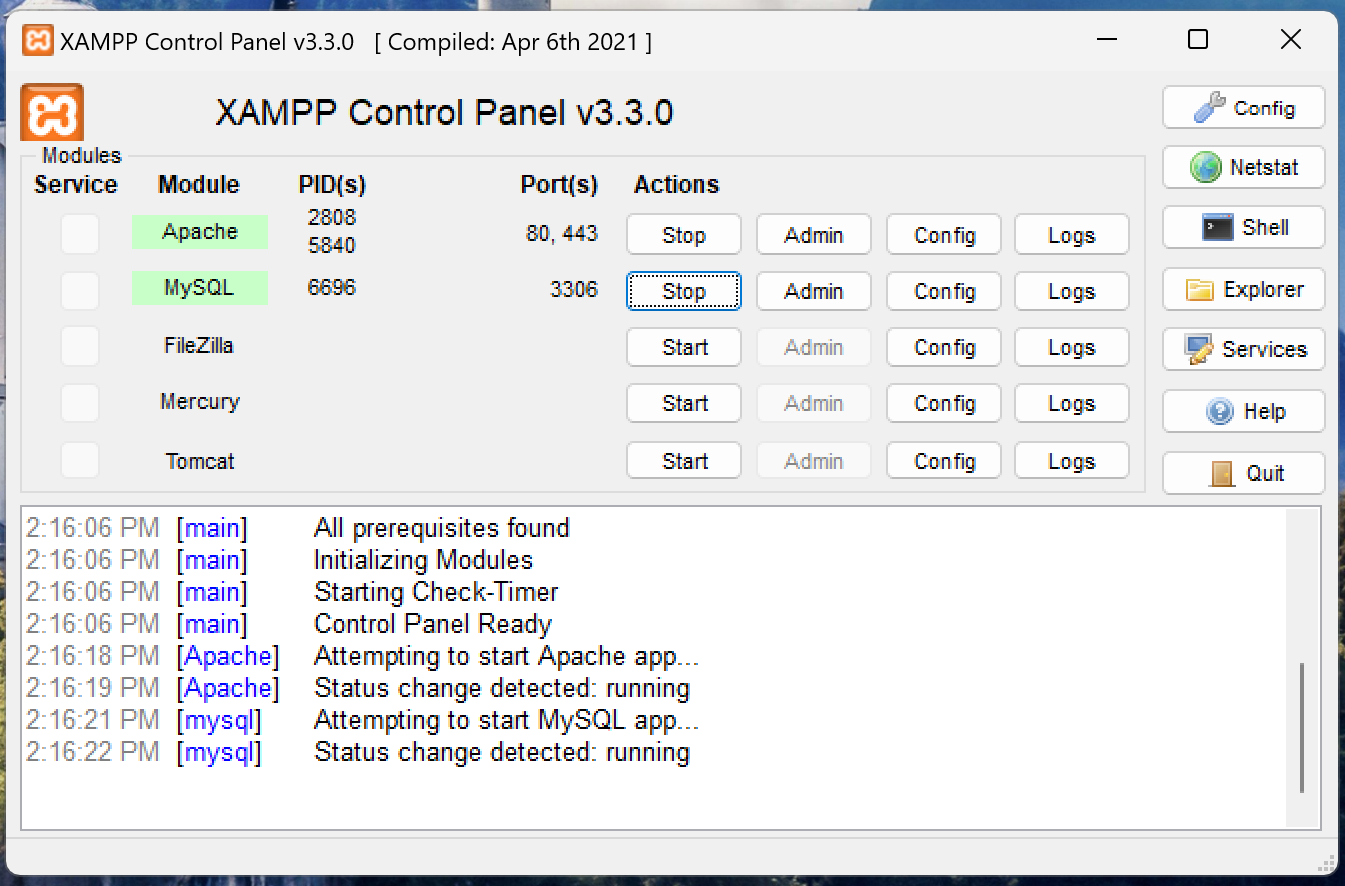
Temukan code berikut :



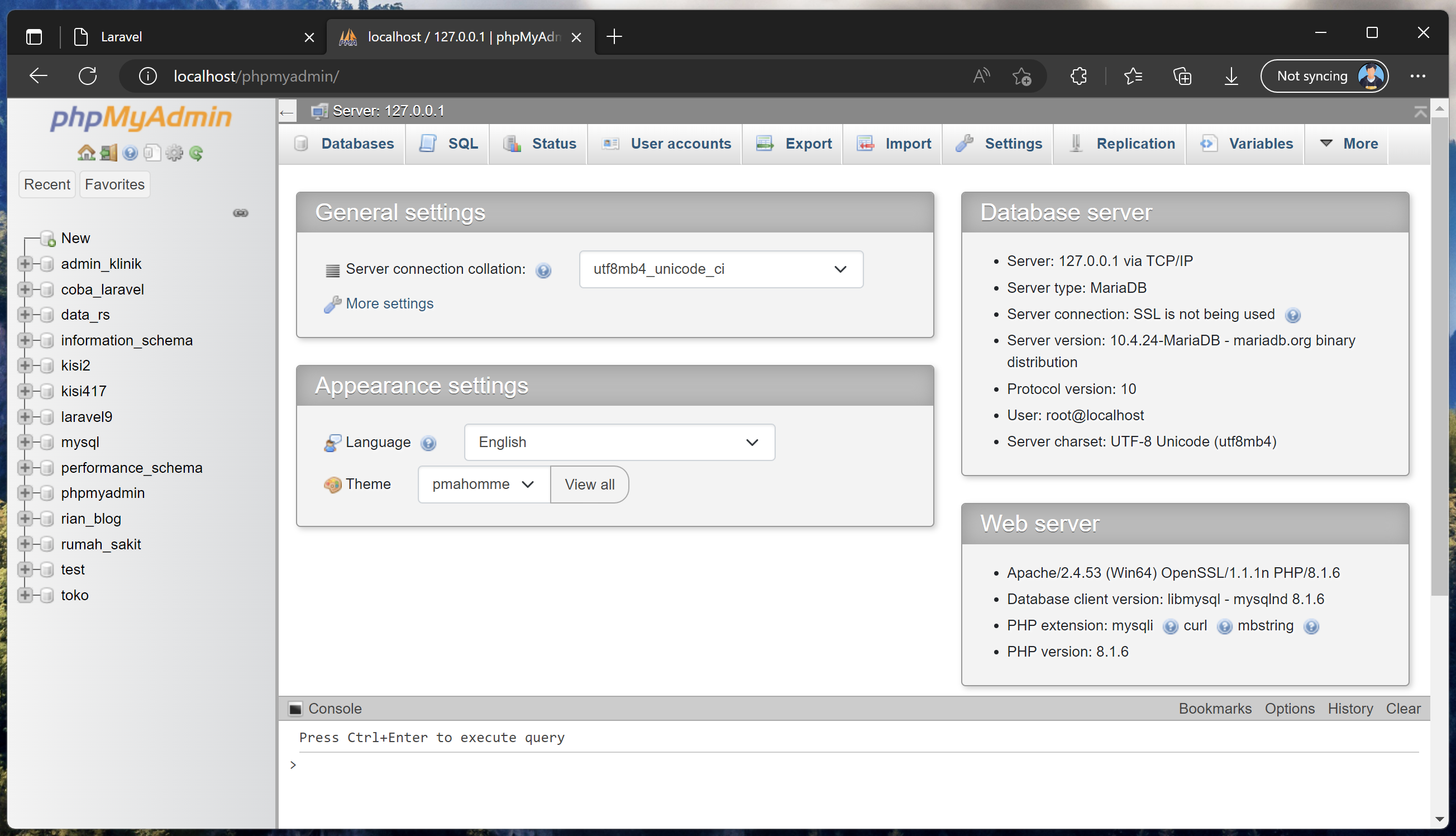
Kemudian ubah sesuai dengan identitas database yang akan dibuat.



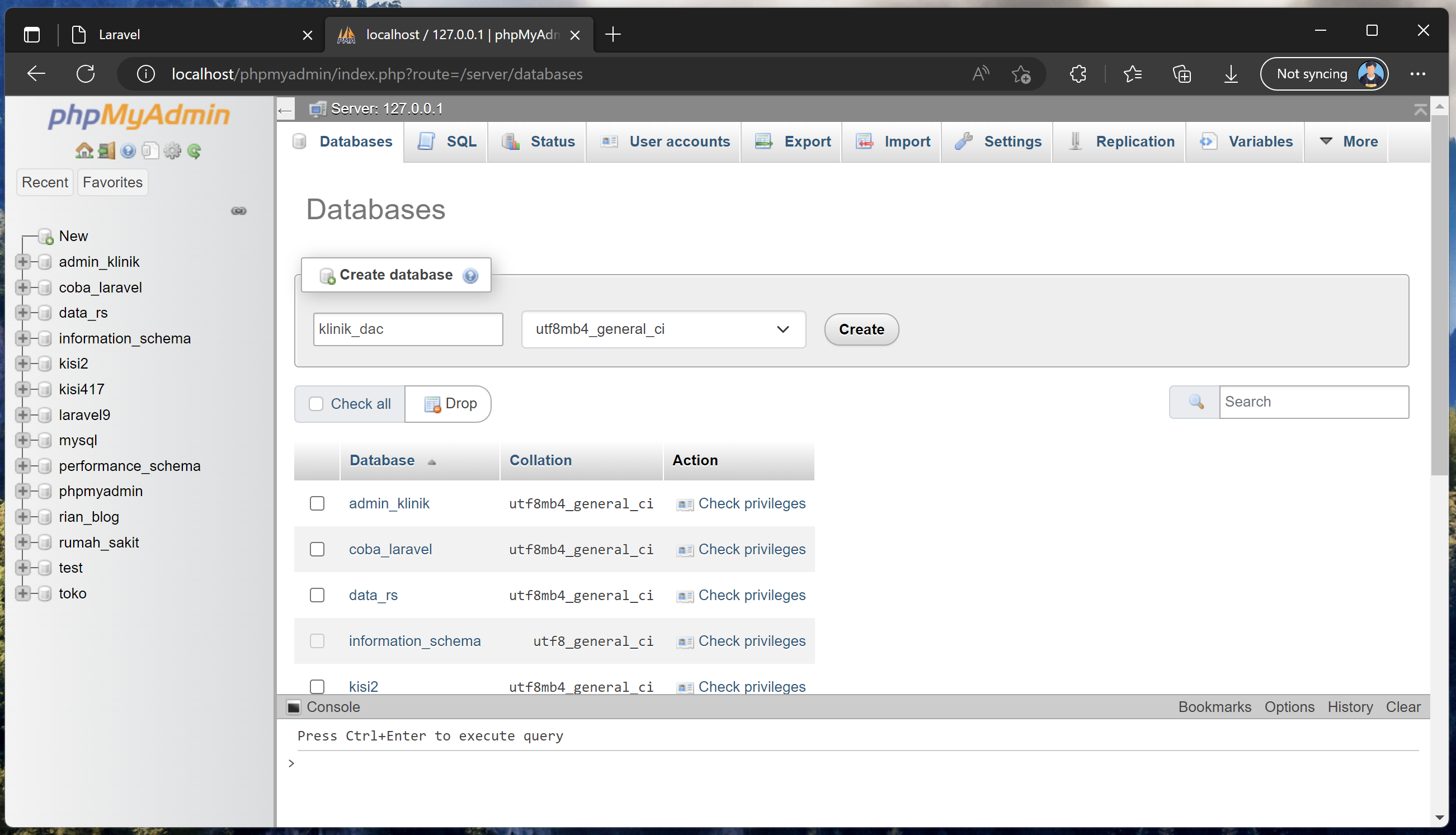
Buat database pada aplikasi **phpmyadmin** yang sudah kita install sebelumnya ketika kita meng-install XAMPP. Sebelum membuka phpmyadmin, buka **XAMPP Control Panel**, kemudian klik tombol **Start** pada menu **Apache** dan **MySQL**.



Jika module Apache dan MySQL sudah berwarna hijau, buka **localhost/phpmyadmin** untuk mengakses aplikasi DBMS dari phpmyadmin.



Buat database baru dengan nama yang sama dengan yang kita ubah di file **.env**.



# Blade Template

## Membuat View Layout

Jika kita menjalankan server Laravel 9 pada saat pertama kali, maka akan muncul halaman awal dari Laravel. Halaman tersebut berada di **resource/view** dengan nama **welcome.blade.php**. Untuk membuat halaman view baru, kita perlu membuat file dengan extensi **.blade.php**.

* Pertama, buat layout dari project kita. Layout ini nantinya akan digunakan untuk banyak view. Buat folder baru di dalam **resource/view** dengan nama **shared**. Kemudian, buat file di dalamnnya dengan nama **layout.blade.php**.
* Buat struktur kode HTML beserta dengan style-nya. Untuk style kita akan menggunakan framework **Bootstrap 5**.

<!doctype *html*>

<html *lang*="en">

    <head>

        <meta *charset*="utf-8">

        <meta *name*="viewport" *content*="width=device-width, initial-scale=1">

        <title>Home - Klinik DAC</title>

        <link *href*="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" *rel*="stylesheet" *integrity*="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" *crossorigin*="anonymous">

    </head>

    <body>

*{{-- Navbar --}}*

*{{-- Main Content --}}*

*{{-- Footer --}}*

    <script *src*="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" *integrity*="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3" *crossorigin*="anonymous"></script>

    </body>

</html>

* Buat navbar di dalam <body>, letakkan pada bagian **{{-- Navbar --}}**

<header>

    <nav *class*="navbar navbar-expand-lg bg-light box shadow px-4">

        <div *class*="container-fluid">

            <a *class*="navbar-brand" *href*="/">Klinik DAC</a>

            <button *class*="navbar-toggler" *type*="button" *data-bs-toggle*="collapse" *data-bs-target*="#navbarNav" *aria-controls*="navbarNav" *aria-expanded*="false" *aria-label*="Toggle navigation">

            <span *class*="navbar-toggler-icon"></span>

            </button>

            <div *class*="collapse navbar-collapse" *id*="navbarNav">

                <ul *class*="navbar-nav">

                <li *class*="nav-item">

                    <a *class*="nav-link active" *href*="/">Home</a>

                </li>

                <li *class*="nav-item">

                    <a *class*="nav-link" *href*="/pasien">Pasien</a>

                </li>

                <li *class*="nav-item">

                    <a *class*="nav-link" *href*="/berobat">Berobat</a>

                </li>

                <li *class*="nav-item">

                    <a *class*="nav-link" *href*="/dokter">Dokter</a>

                </li>

                <li *class*="nav-item">

                    <a *class*="nav-link" *href*="/obat">Obat</a>

                </li>

                <li *class*="nav-item">

                    <a *class*="nav-link" *href*="/rs\_rujuk">RS Rujukan</a>

                </li>

                </ul>

            </div>

            <a *class*="btn btn-outline-dark" *href*="#">Logout</a>

        </div>

    </nav>

</header>

* Buat footer dan letakan di paling bawah dari <body> dan sebelum <script>.

Asd

<footer *class*="border-top footer text-muted">

<div *class*="container my-3">

      &copy; 2022 - Klinik DAC - <a *href*="#">Privacy</a>

      </div>

</footer>

* Buat tempat untuk menampung konten dari setiap view. Buat <div> kosong dan sisipkan di antara navbar dan footer. Untuk mengetahui bahwa di dalam <div> tersebut merupakan tempat untuk konten, maka kita perlu menentukan nama **section.** Gunakan **@yield(‘main-content’)**.

As

<div *class*="container">

<main *class*="pb-3">

      @yield('main-content')

      </main>

</div>

## Membuat View Index

* Buat file baru di dalam **resource/view** dengan nama **index.blade.php**.
* Untuk memanggil file layout yang sudah kita buat, gunakan **@extends(‘shared.layout’)**.

@extends('shared.layout')

* Bungkus isi dari konten file index.blade.php dengan **@section(‘main-content’)** kemudian ditutup dengan **@endsection**.

@section('main-content')

    ...

@endsection

* Berikut contoh untuk halaman view dari index.

@extends('shared.layout')

@section('main-content')

<div *class*="mt-3 py-5 px-5 bg-light rounded-4 shadow">

    <div *class*="my-5">

        <h1>KLINIK DAC</h1>

        <h3>DAC Solution Training Center</h3>

    </div>

</div>

<div *class*="row mt-4">

    <div *class*="col-md-4">

        <h2>Klinik DAC</h2>

        <p>

            Klinik DAC merupakan aplikasi sampel dari DAC Solution Training Center

            yang dibuat untuk pembelajaran tentang bagaimana membuat aplikasi

            dengan Framework Laravel 9.

        </p>

    </div>

    <div *class*="col-md-4">

        <h2>Laravel 9</h2>

        <p>Lihat dokumentasi lengkap mengenai Framework Laravel 9</p>

        <p><a *class*="btn btn-outline-primary" *href*="https://laravel.com/docs/9.x">Lihat Tutorial &raquo;</a></p>

    </div>

    <div *class*="col-md-4">

        <h2>DAC Solution Training Center</h2>

        <p>Lihat lebih lengkap tentang DAC Solution Training Center.</p>

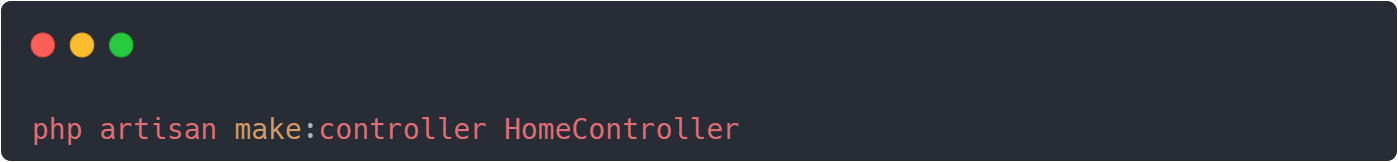
        <p><a *class*="btn btn-outline-primary" *href*="https://www.instagram.com/dac\_trainingcenter/">Lihat Profil &raquo;</a></p>

    </div>

</div>

@endsection

Jika dijalankan, halaman index kita belum bisa tampil. Yang akan tampil tetep view bawaan dari Laravel. Untuk membuatnya agar tampil, kita perlu membuat **Controller**. Untuk membuat Controller gunakan command :



File Controller berada di **app/Http/Controlllers/**. Buka file HomeController yang sudah kita buat tadi, kemudian buat function **index()** di dalamnya.

function index() {

        return view('index');

    }

Function di atas akan menampilkan view dari ‘index’ (index.blade.php).

Jika sudah mengatur Controller, maka kita perlu menentukan rute dari URL yang akan kita masukkan. Kita ingin hanya dengan menuliskan nama host-nya saja **(http://127.0.0.1:8000)**, maka akan muncul file index kita.

Pergi file **web.php** yang berada di folder **routes**, kemudian hapus rute yang sudah ada, dan ganti dengan rute yang baru.

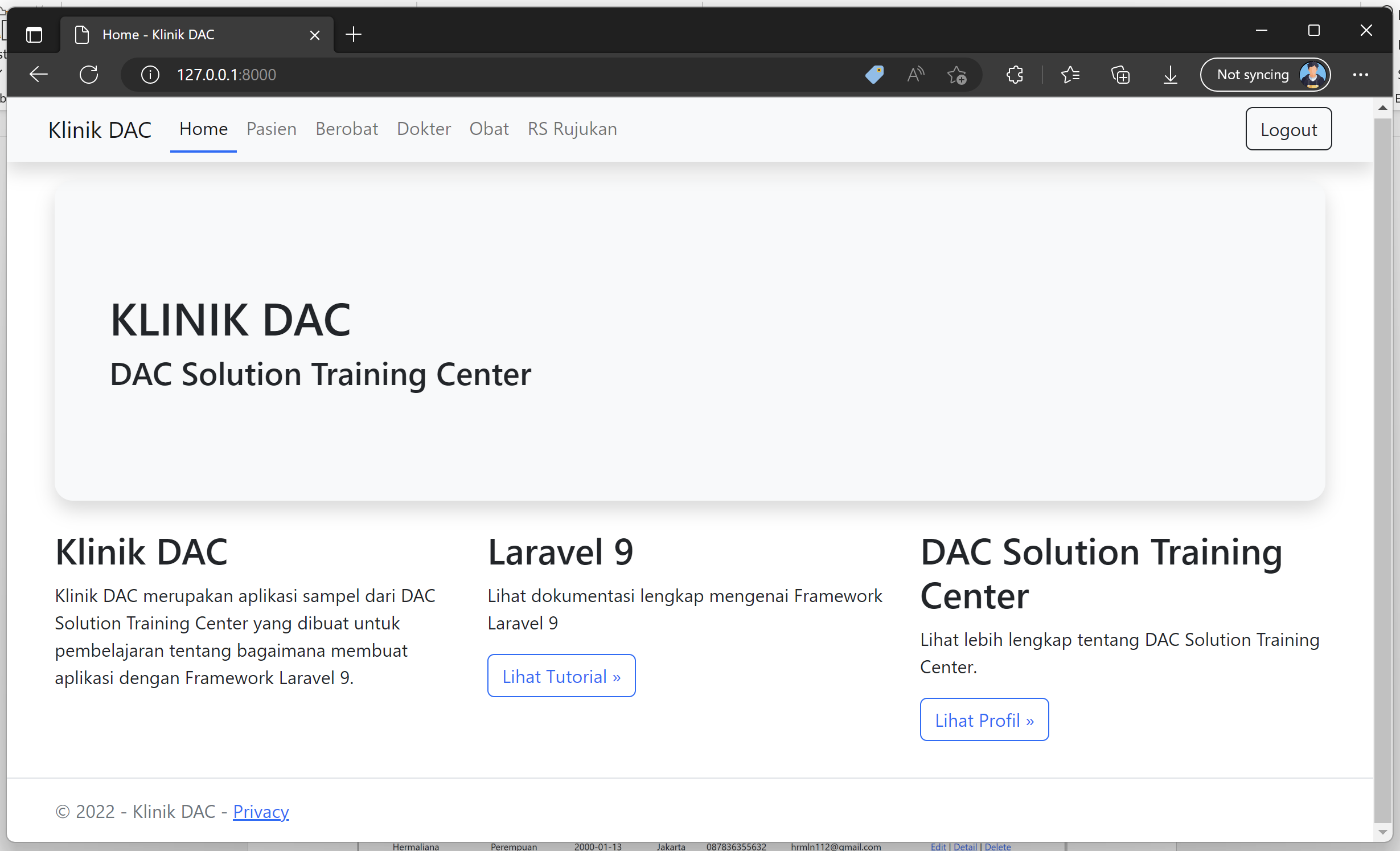
Route::get('/', [HomeController::class, 'index']);

* **‘/’** merupakan URL yang akan memunculkan view **index()**
* **HomeController** merupakan nama Controller yang digunakan
* **‘index’** merupakan function yang digunakan di dalam HomeController

Sebelumnya, pastikan HomeController sudah ter-include pada route kita. Jika belum ada, tambahkan di bagian atas dari file **web.php**

use App\Http\Controllers\HomeController;

Jalankan server Laravel.



## Mengirim Data ke View

Kita sudah berhasil membuat view layout dan view index untuk halaman Home. Namun, ada sedikit masalah. Kita sudah tahu sebelumnya kalau view layout akan kita gunakan sebagai template untuk view lainnya. Tapi, view layout masih bersifat statis. Maksudnya, untuk judul dari setiap view sama. Selain itu, navbar active akan menjadi sama walaupun kita berada di view yang berbeda-beda.

Untuk membuatnya menjadi dinamis, kita bisa mengirimkan data dari Controller ke View.

function index() {

        $title = "Home";

        return view('index')->with('title', $title);

    }

Pada kode di atas, kita mengirimkan data berupa “Home” ke view index yang nantinya akan ditampung ke dalam variable $title pada view index.

Untuk menampilkannya, pergi ke view index, kemudian ubah elemen yang akan dijadikan tulisan dinamis seperti judul halaman dan navbar active. Kita bisa menggunakan “{{ … }}” untuk menyisipkan variable atau syntax blade.

<title>{{ $title }} - Klinik DAC</title>

Pada navbar active, kita perlu menambahkan ternary operator pada bagian class active-nya.

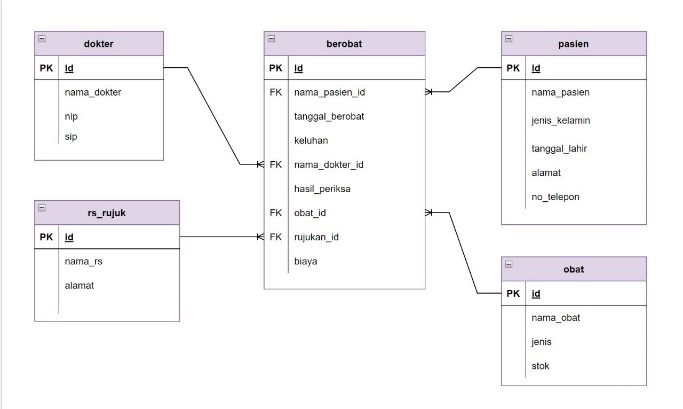
<a *class*="nav-link {{ $title == 'Home' ? 'active border-bottom border-primary border-2' : '' }}" *href*="/">Home</a>

Lakukan ke semua navbar item.

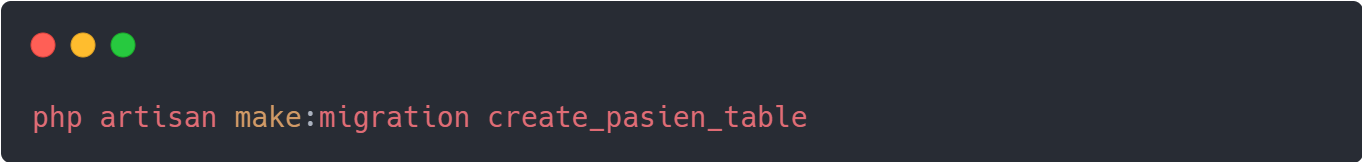
# Database Migrations

## Membuat Database Migration

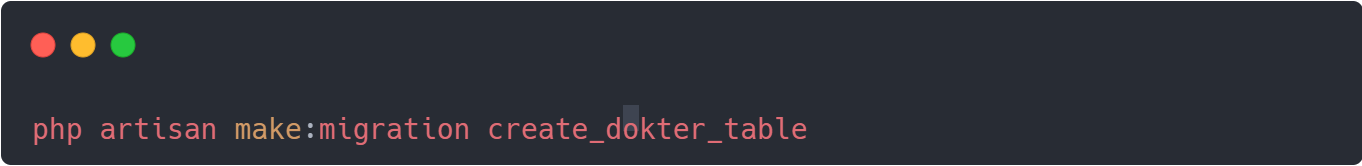
Kita akan membuat database dengan desain tabel sebagai berikut :



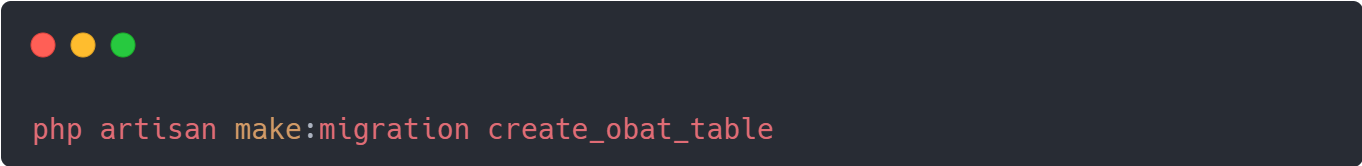
Setelah membuat database pada phpmyadmin, selanjutnya buat file migration untuk tabel Pasien. Buka terminal dan jalankan command :



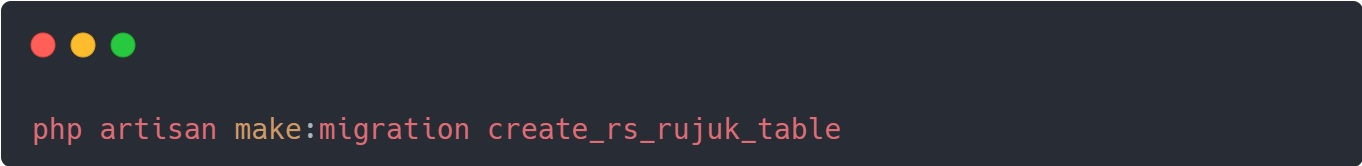
Lakukan hal yang sama untuk membuat file migration tabel Dokter.



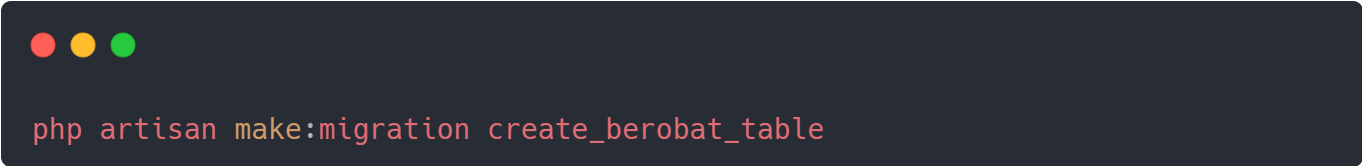
Tabel Obat :



Tabel Rumah Sakit Rujukan :



Dan tabel Berobat.



Hasil dari file migration yang telah kita buat akan diletakkan di **database/migrations**. Buka file migration tadi kemudian isi nama-nama field pada function **up()** sesuai dengan desain database di atas.

File migration pasien :

*public* function up()

    {

        Schema::create('pasien', function (Blueprint $table) {

            $table->id();

            $table->string('nama\_pasien');

            $table->enum('jenis\_kelamin', ['Laki-laki', 'Perempuan']);

            $table->date('tanggal\_lahir');

            $table->text('alamat');

            $table->string('no\_telepon');

            $table->string('email')->nullable();

            $table->timestamps();

        });

    }

File migration dokter :

*public* function up()

    {

        Schema::create('dokter', function (Blueprint $table) {

            $table->id();

            $table->string('nama\_dokter');

            $table->bigInteger('nip');

            $table->bigInteger('sip');

            $table->timestamps();

        });

    }

File migration obat :

*public* function up()

    {

        Schema::create('obat', function (Blueprint $table) {

            $table->id();

            $table->string('nama\_obat');

            $table->string('jenis');

            $table->text('deskripsi');

            $table->date('tanggal\_exp');

            $table->integer('stok');

            $table->timestamps();

        });

    }

File migration rs\_rujuk :

Schema::create('rs\_rujuk', function (Blueprint $table) {

            $table->id();

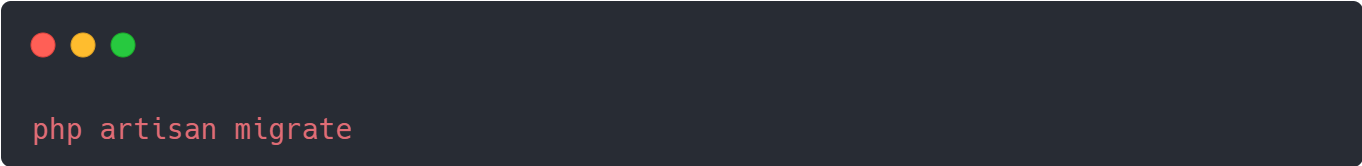
            $table->string('nama\_rs');

            $table->text('alamat');

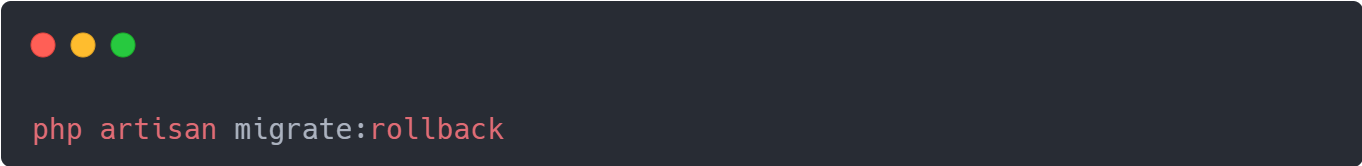
            $table->timestamps();

        });

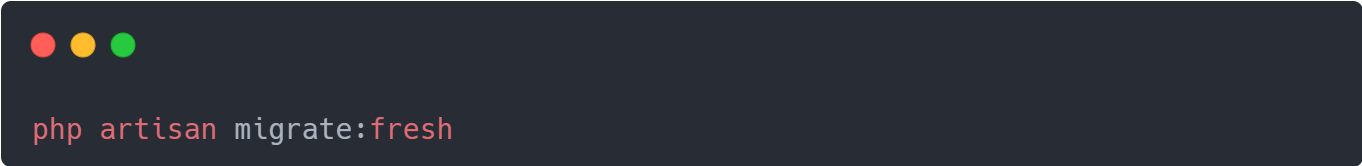
Jika sudah, lakukan **migrate** untuk membuat tabel ke dalam database.



Jika kita ingin mengubah skema dari tabel kita, seperti menambahkan field atau menghapus field, silahkan mengubahnya di file migrasi yang sebelumnya. Namun, jika ingin dilakukan migrate, lakukan perintah **rollback** terlebih dahulu, kemudian jalankan perintah **migrate**.

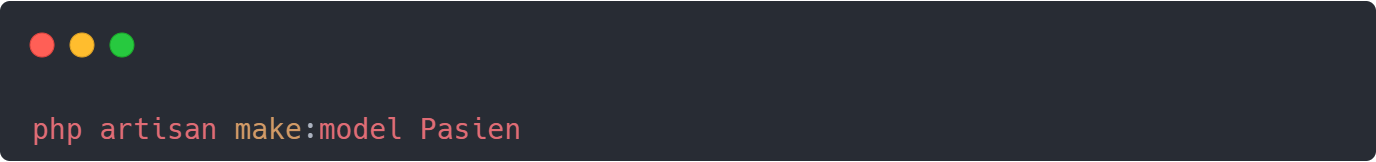


Untuk melakukan rollback dan migrate secara bersamaan, maka gunakan perintah **fresh**.

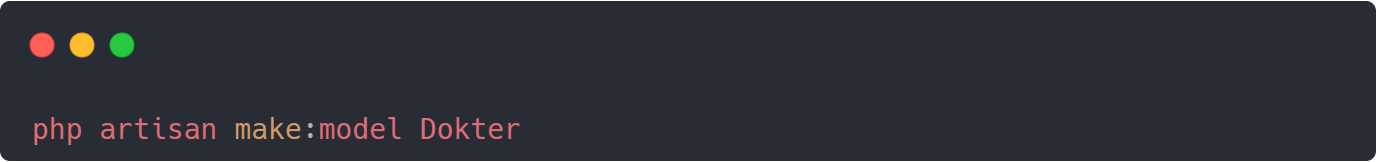


## Membuat Model

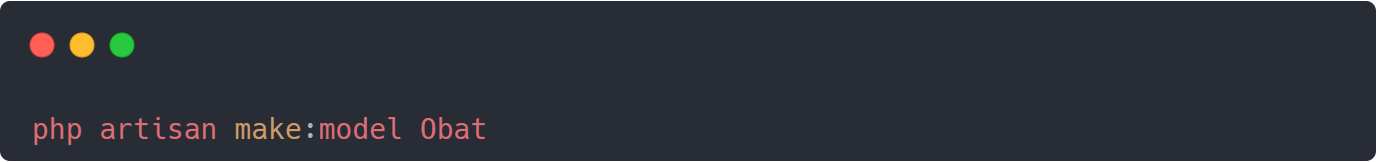
Untuk membuat model, dapat mengetik command :



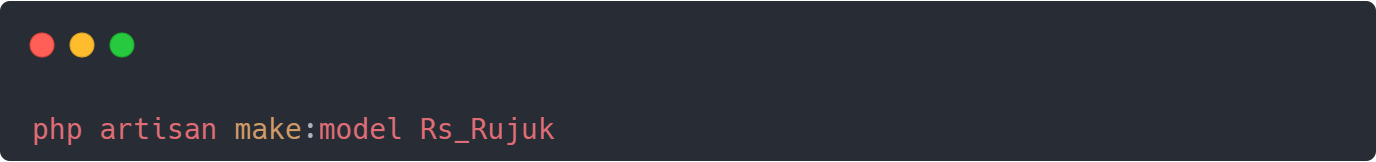
Model Dokter



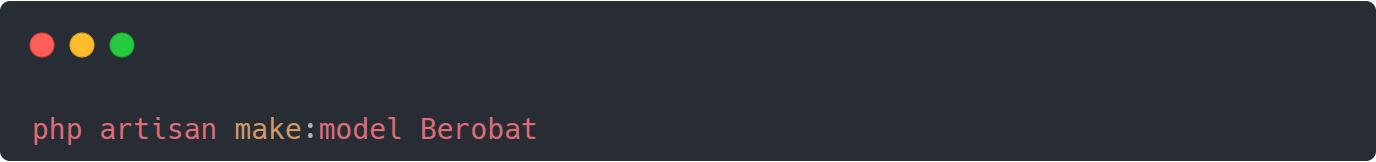
Model Obat



Model Rs\_Rujuk



Model Berobat



Jika dijalankan, akan muncul pesan error bahwa table tidak ditemukan. Ini karena table-nya berubah namanya menjadi ada huruf ‘s’ di belakangnya (Contoh : “pasiens”).

Untuk mengubahnya, buka model yang sudah dibuat, kemudian tambahkan :

*protected* $table = 'pasien';

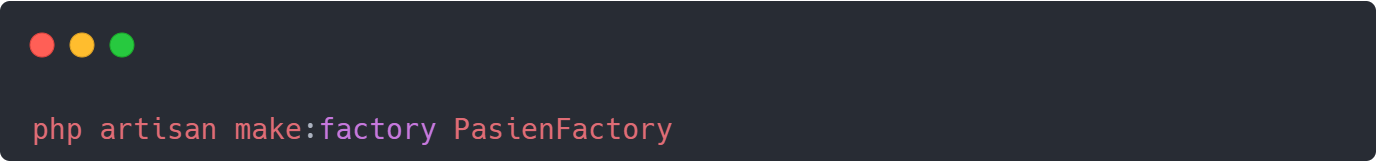
Tambahkan ke semua Model sesuai dengan nama tabelnya.

## Mengisi Data dengan Factory dan Faker

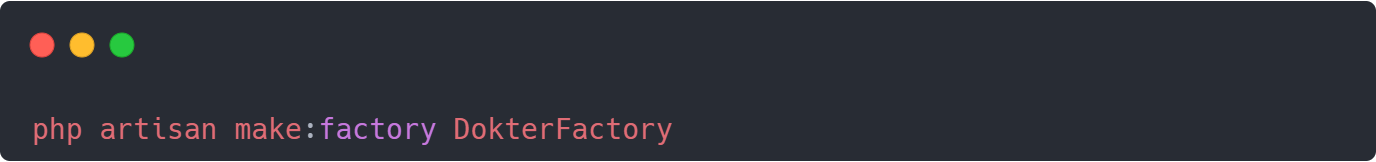
**Faker** merupakan library yang dapat kita gunakan untuk mengisi data dummy ke dalam database kita ketika kita sudah ingin melakukan testing pada aplikasi. Dengan Faker, kita bisa melakukan insert data ke dalam database secara otomatis sesuai jumlah data yang kita mau.

Untuk membuat Faker, kita perlu membuat Factory terlebih dahulu.

PasienFactory:



DokterFactory :



Rs\_RujukFactory :



Walaupun bisa insert data secara otomatis, tidak semua jenis data tidak bisa ditambahkan secara otomatis. Faker memiliki library yang terbatas, dan hanya beberapa jenis data saja yang bisa ditambahkan secara otomatis dan acak. Untuk melihat dokumentasi lengkap mengenai Faker, silahkan kunjungi situs <https://fakerphp.github.io/>.

## Database Seeding

Jika sudah membuat Factory, kita sudah bisa menambahkan data ke dalam masing-masing tabel. Untuk mengisi data, kita bisa pergi ke **DatabaseSeeder.php**. Untuk lokasi DatabaseSeeder berapa di **database/seeders**.

Dalam DatabaseSeeder.php kita hanya perlu menambahkan kode berikut :

*public* function run()

    {

        Pasien::factory()->count(5)->create();

        Dokter::factory()->count(5)->create();

        Rs\_rujuk::factory()->count(5)->create();

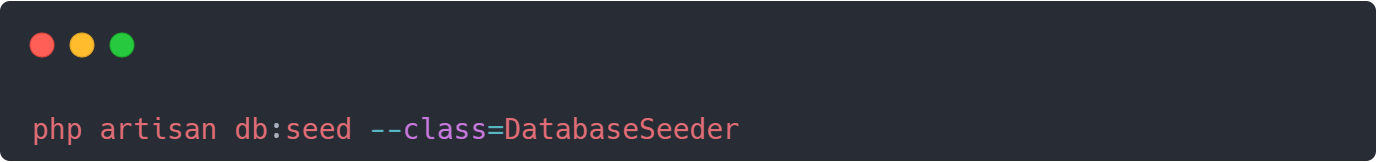
    }

**Pasien** : Model dari Pasien

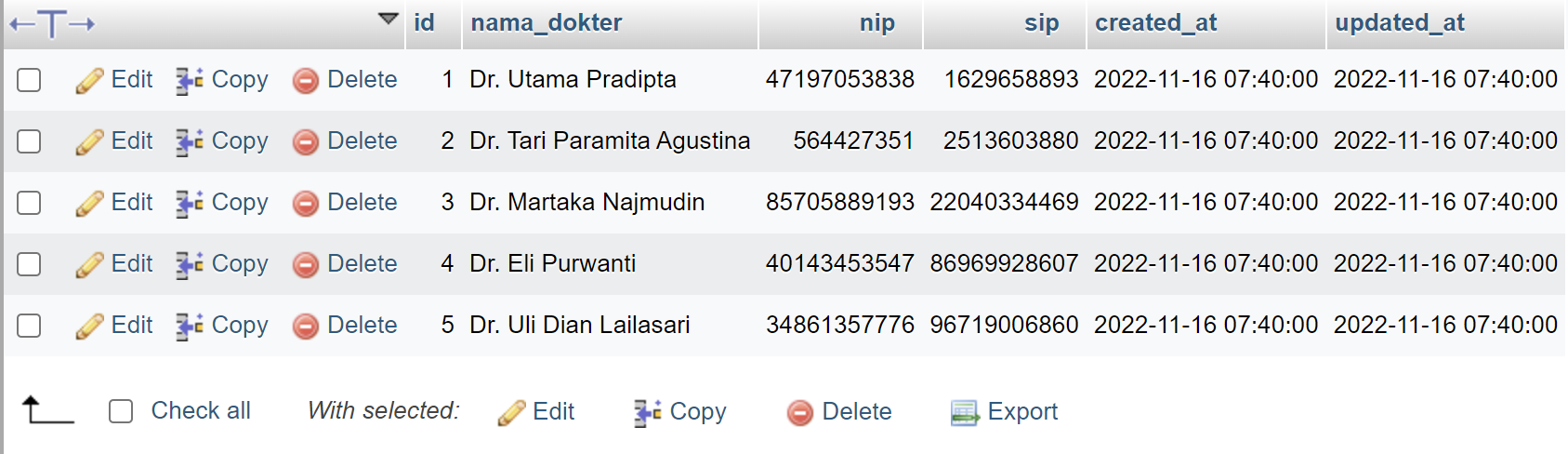
**Count(5)** : Banyaknya record data yang akan di-insert (5 record).

**Create()** : Create data

Jika sudah diisi, jalankan perintah agar seluruh record dapat disimpan ke dalam tabel.



Buka DMBS untuk mengecek datanya.



## Menampilkan Data dengan Eloquent

Untuk menampilkan data dari database, kita harus membuat Controller terlebih dahulu untuk setiap tabel.

