

LARAVEL

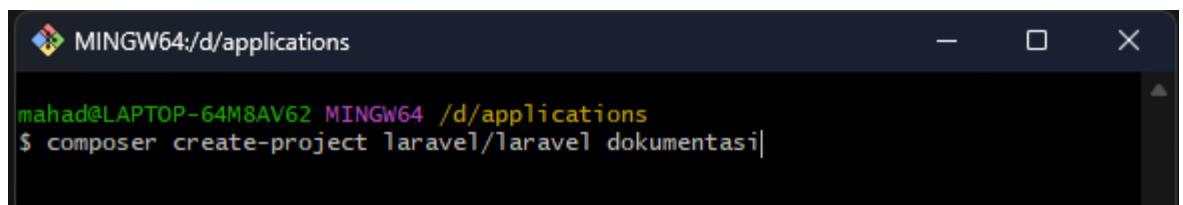
1. Requirment

- PHP 7.4 / 8, MySQL, & phpMyAdmin (Ketiga hal ini sudah ada apabila kita menggunakan XAMPP)
- Composer 2
- Terminal / CMD / PowerShell / GitBash
- Code Editor, disini saya menggunakan Visual Studio Code
- Adapun beberapa extension VsCode yang bisa kita gunakan:
 1. PHP Intelephense
 2. Laravel Artisan
 3. Laravel Snippets
 4. Laravel Blade Snippets
 5. Laravel Blade Spacer
 6. Laravel goto view
 7. Laravel Blade formatter

2. Instalasi

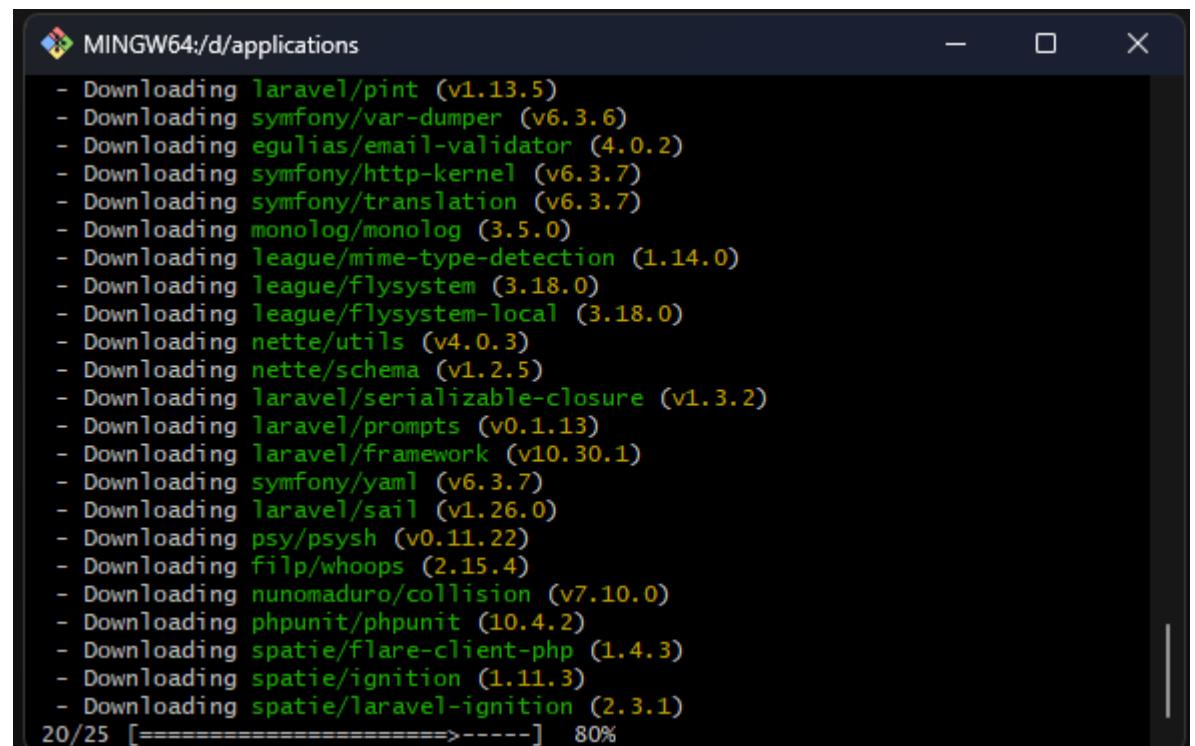
Jika semua yang ada di Requirment sudah terdapat semua, bisa lanjut ketahap ini.

1. Buat Folder Baru yang bernamakan “applications”
2. Buat gitbash atau command prompt lainnya dan pastikan directorynya mengarah ke file “applications”
3. Jalankan perintah “composer create-project laravel/laravel dokumentasi” untuk “dokumentasi” itu adalah nama file/project yang akan dibuat kemudian tekan enter



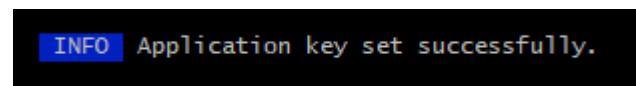
```
MINGW64:/d/applications
mahad@LAPTOP-64M8AV62 MINGW64 /d/applications
$ composer create-project laravel/laravel dokumentasi
```

- Setelah itu jika tampilannya sudah seperti dibawah, tunggu hingga selesai

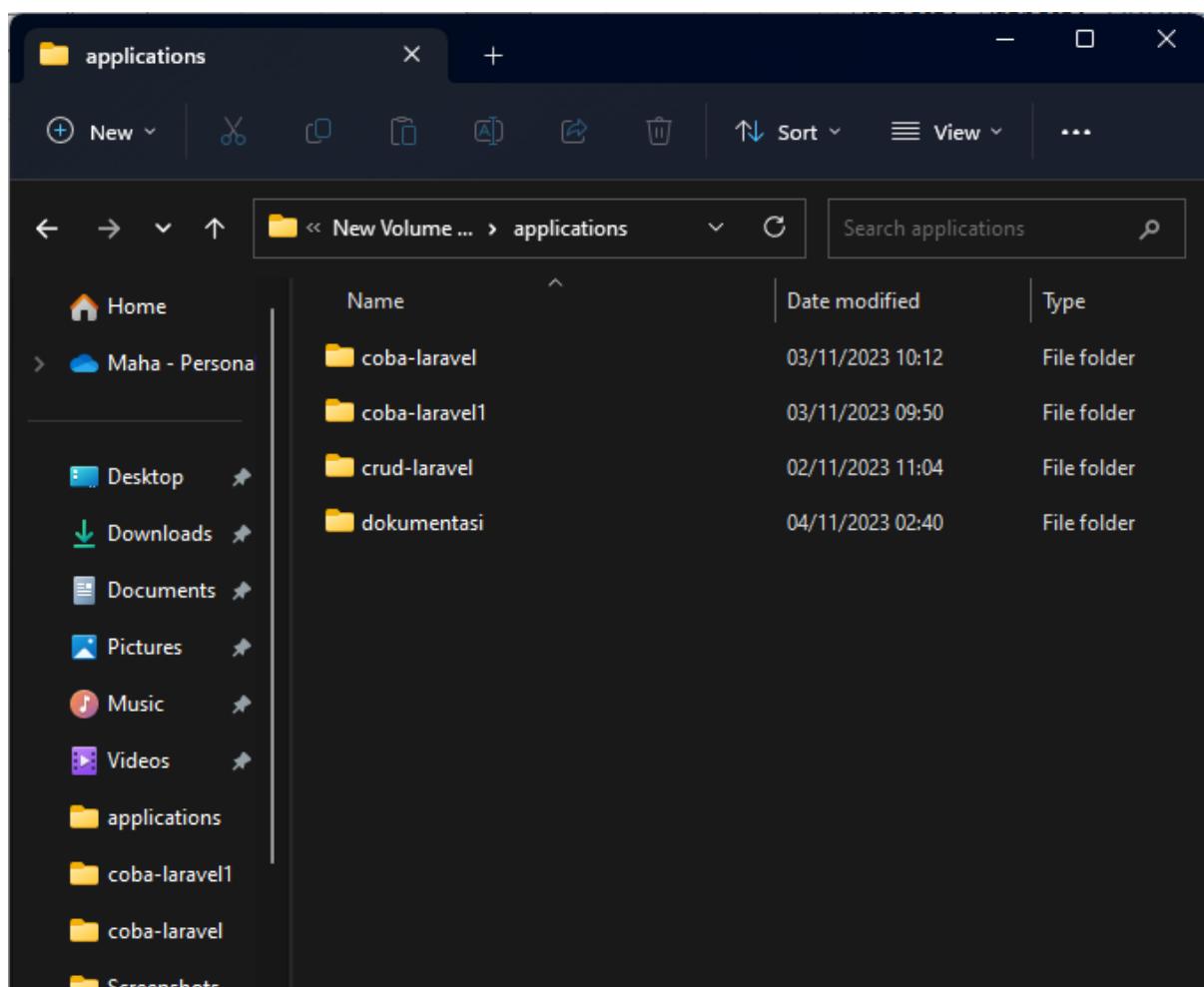


A terminal window titled "MINGW64:/d/applications" showing the output of a composer command. It lists numerous packages being downloaded, including Laravel/pint, symfony/var-dumper, egulias/email-validator, symfony/http-kernel, symfony/translation, monolog/monolog, League/mime-type-detection, League/flysystem, League/flysystem-local, nette/utils, nette/schema, Laravel/serializable-closure, Laravel/prompts, Laravel/framework, symfony/yaml, Laravel/sail, psy/psr7, filp/whoops, nunomaduro/collision, phpunit/phpunit, spatie/flare-client-php, spatie/ignition, and spatie/laravel-ignition. A progress bar at the bottom indicates 80% completion.

- Tampilan apabila sudah selesai akan seperti ini dan juga di folder applications akan terbuat 1 folder bernama “dokumentasi” sesuai dengan yang tadi kita buat



INFO Application key set successfully.



3. Pembuatan Layout

Pertama-tama buat folder layouts didalam folder resources\views kemudian buat file yang bernama main.blade.php kemudian isi dari file tersebut adalah struktur html seperti dibawah ini.

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Dokumentasi Blog | {{ $title }}</title>
    <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
        rel="stylesheet"
        integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
        crossorigin="anonymous">

        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">

        <link rel="stylesheet" href="/css/style.css">
</head>

<body>
    @include('partials.navbar')

    <div class="container mt-4">
        @yield('container')
    </div>

    <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-C6RzsynM9kWDrMNeT87bh950GNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
        crossorigin="anonymous">
    </script>
</body>

</html>
```

Pertama di bagian title terdapat {{ \$title }} ini digunakan untuk menyesuaikan dengan halaman apa yang akan dibuka, value dari variable ini biasanya diatur di controller atau bisa langsung di routes\web.php

Kedua terdapat @include('partials.navbar') ini untuk memanggil bagian navbar yang saya buat terpisah

Ketiga @yield('container') ini digunakan untuk menjadi tempat nanti yang akan diisi di file lain dengan cara

```
@extends('layouts.main')

@section('container')
    <h1>Hello Dunia</h1>
@endsection
```

Extends terlebih dahulu main layoutsnya kemudian tambahkan @section('container') dan @endsection diantar kode yang kita buat.

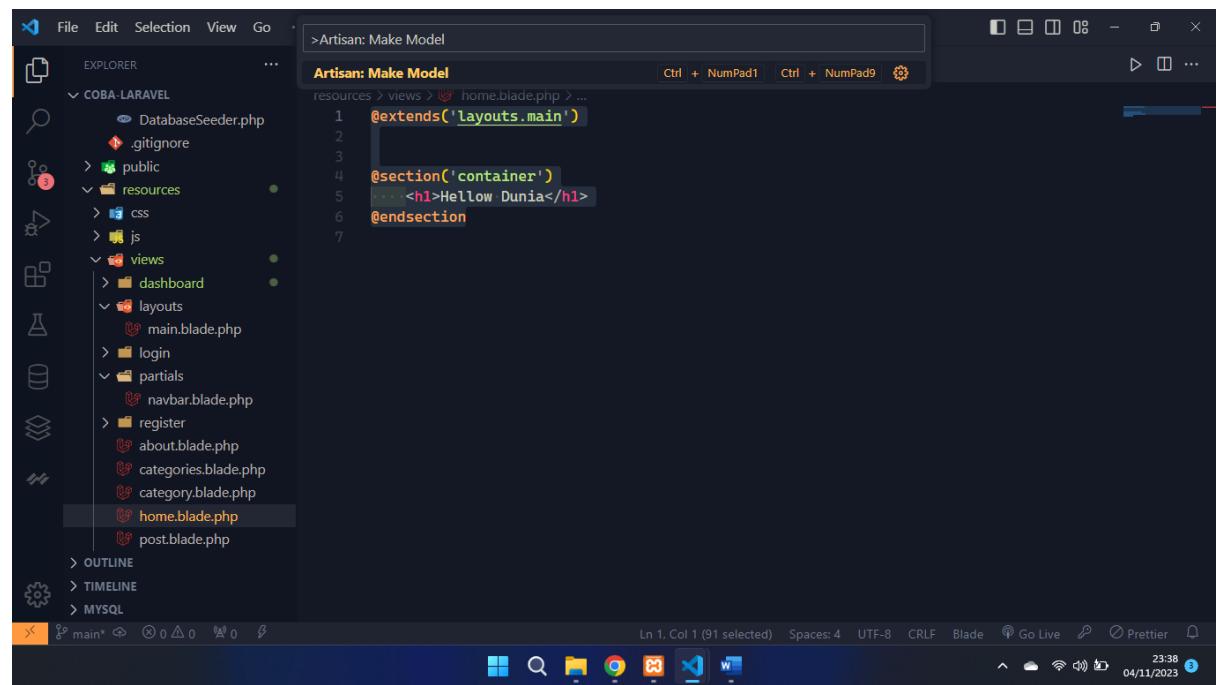
4. Pembuatan model & controller

Cara membuat model

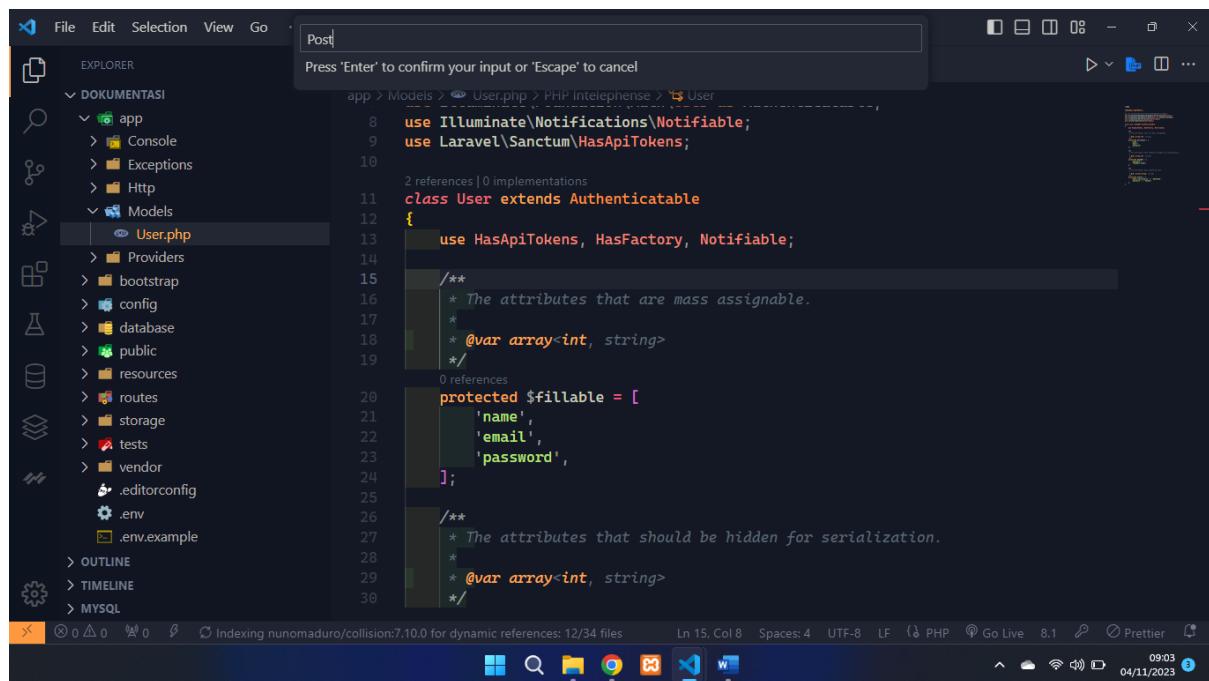
Pastikan sudah menginstal Laravel artisan agar bisa mengetikkan perintah yang di ketik di terminal, lewat command palette

Buka command palette dengan menekan Ctrl + shift + p ketik Artisan: Make Model

Klik



Ketik nama model yang akan dibuat (Post)



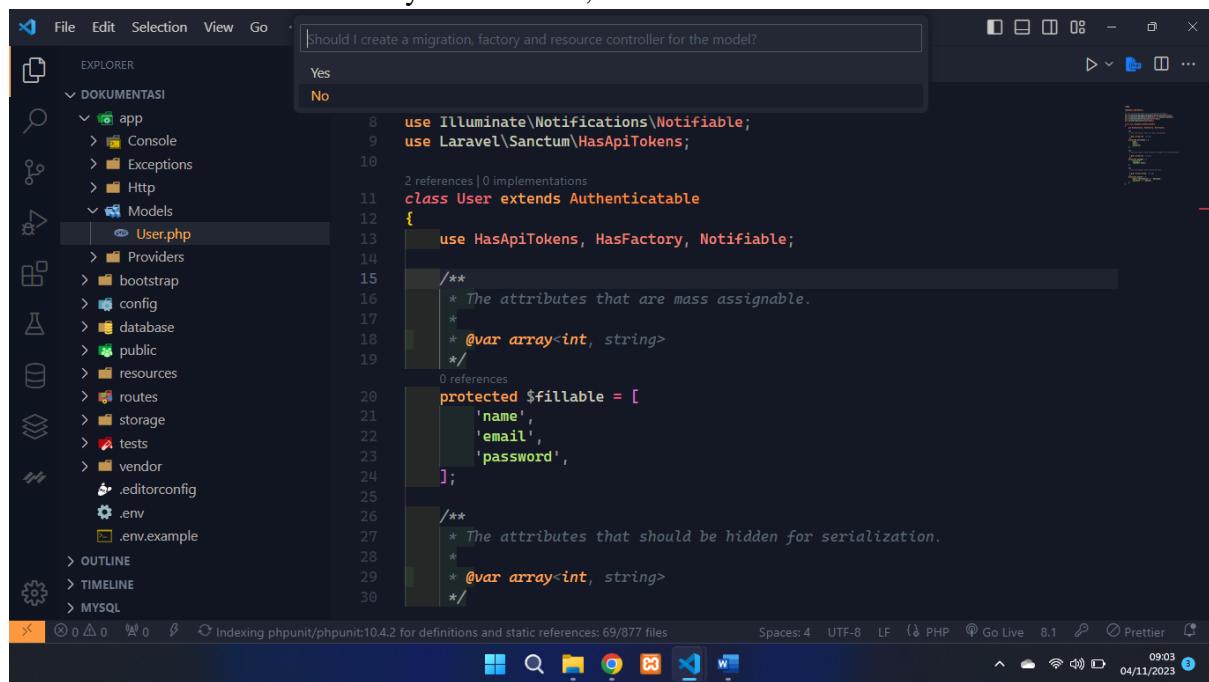
```
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
}
```

Disini akan ditanya apakah ingin sekalian membuatkan migration, factory dan resource controller untuk Post karena saya tidak maka, klik no



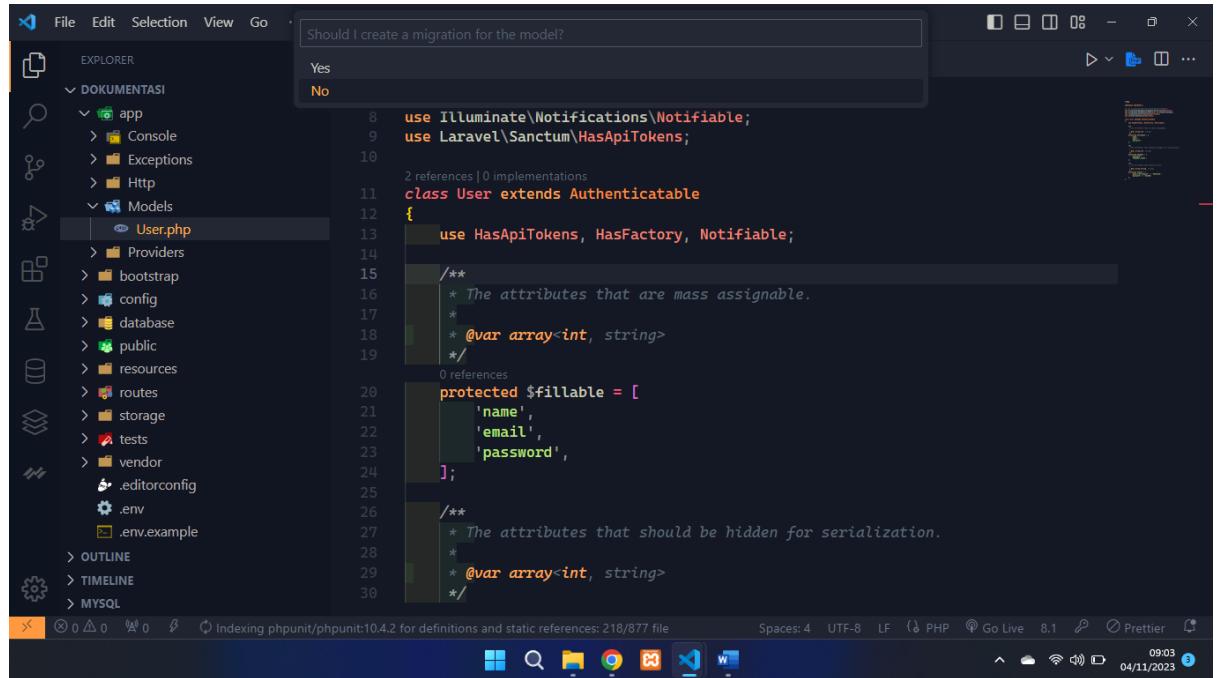
```
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
}
```

Kemudian akan ditanya kembali apakah akan membuat migration untuk model post, karena tidak klik no



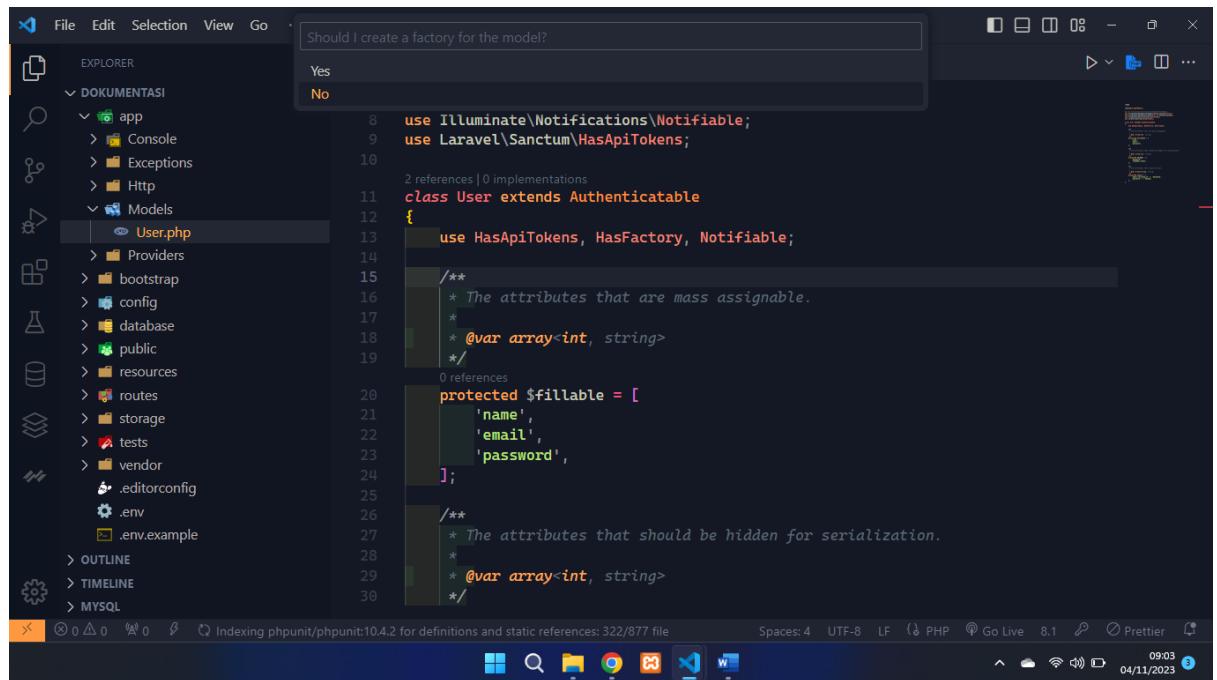
The screenshot shows the Visual Studio Code interface. In the center, a modal window is open with the question "Should I create a migration for the model?". Below the modal, the code editor displays the `User.php` file from a Laravel application. The file contains the following code:use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
 use HasApiTokens, HasFactory, Notifiable;

 /**
 * The attributes that are mass assignable.
 *
 * @var array<int, string>
 */
 protected \$fillable = [
 'name',
 'email',
 'password',
];

 /**
 * The attributes that should be hidden for serialization.
 *
 * @var array<int, string>
 */
}At the bottom of the screen, the status bar shows "Indexing phunit/phunit:10.4.2 for definitions and static references: 218/877 file".

Selanjutnya akan ditanya kembali apakah ingin membuat factory untuk model ini, karena tidak klik no



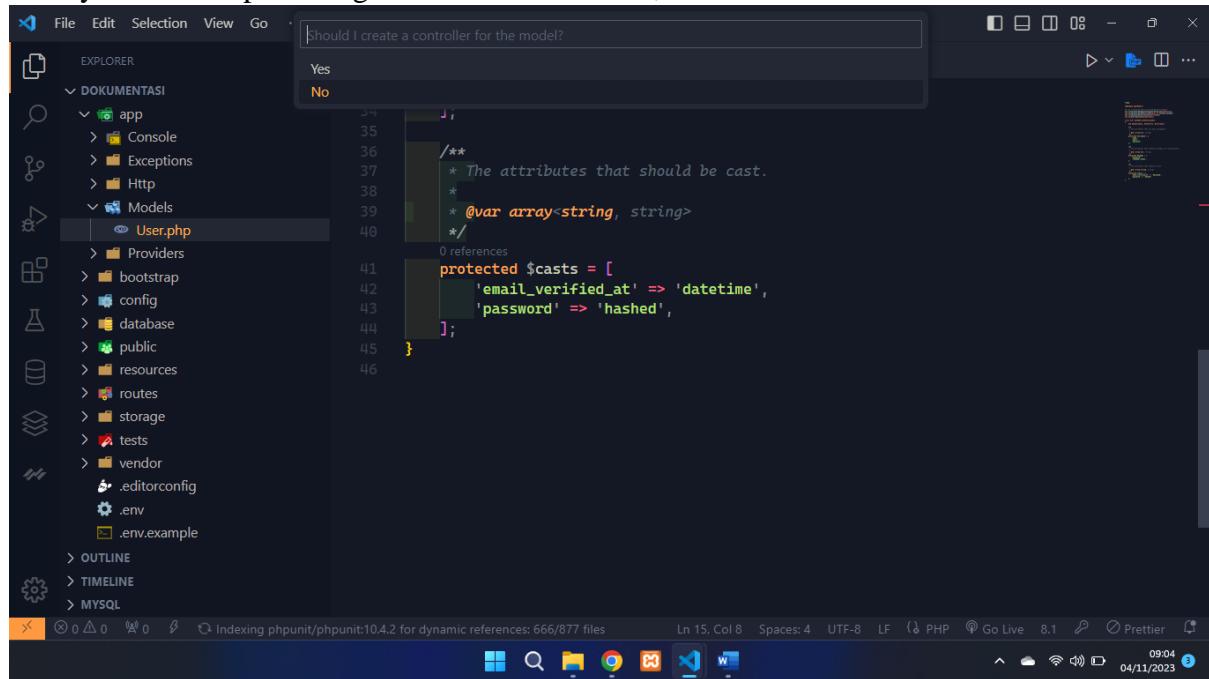
The screenshot shows the Visual Studio Code interface again. A modal window is open with the question "Should I create a factory for the model?". Below the modal, the code editor shows the same `User.php` file. The status bar at the bottom indicates "Indexing phunit/phunit:10.4.2 for definitions and static references: 322/877 file".use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
 use HasApiTokens, HasFactory, Notifiable;

 /**
 * The attributes that are mass assignable.
 *
 * @var array<int, string>
 */
 protected \$fillable = [
 'name',
 'email',
 'password',
];

 /**
 * The attributes that should be hidden for serialization.
 *
 * @var array<int, string>
 */
}

Ditanya kembali apakah ingin membuat controller, karena tidak klik no



```
File Edit Selection View Go ... Should I create a controller for the model? Yes No

EXPLORER DOKUMENTASI app > Console > Exceptions > Http Models > User.php > Providers > bootstrap > config > database > public > resources > routes > storage > tests > vendor .editorconfig .env .env.example > OUTLINE > TIMELINE > MYSQL

User.php



```

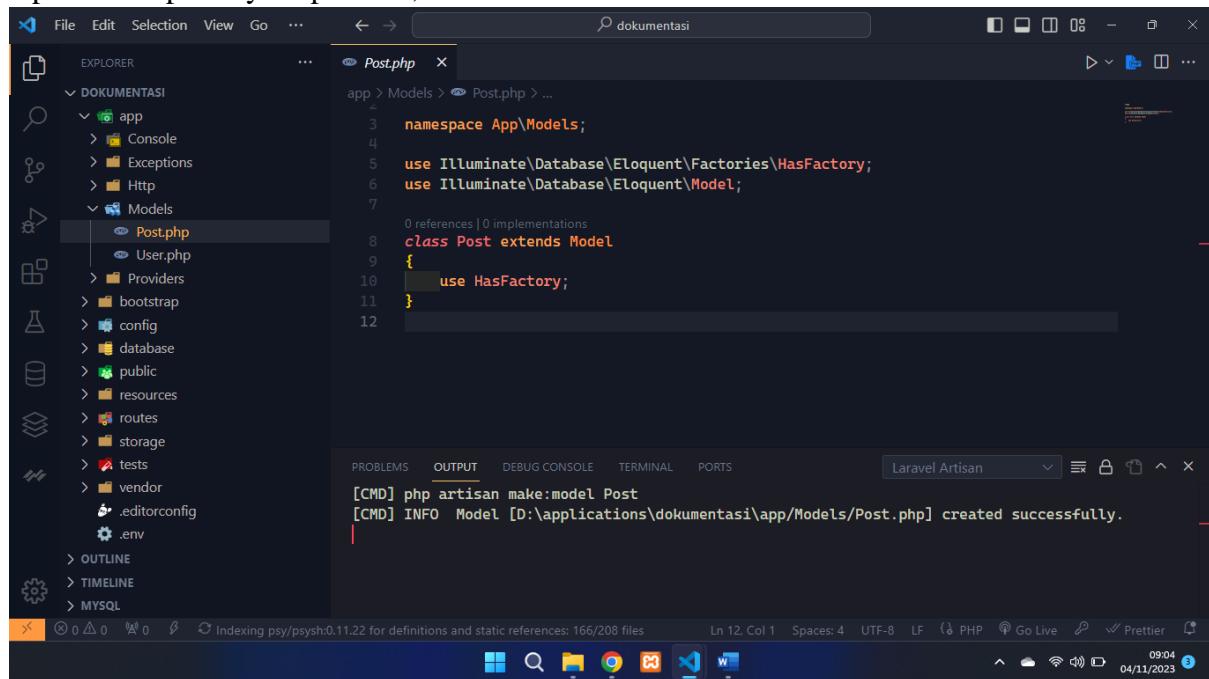
**
* The attributes that should be cast.
*
* @var array<string> string
*
protected $casts = [
 'email_verified_at' => 'datetime',
 'password' => 'hashed',
];

```



Ln 15, Col 8 Spaces: 4 UTF-8 LF ⚡ PHP Go Live 8.1 ⚡ Prettier 09:04 04/11/2023
```

Apabila tampilannya seperti ini, maka model Post telah terbuat.



```
File Edit Selection View Go ... Post.php x

DOKUMENTASI app > Models > Post.php > ...
Post.php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

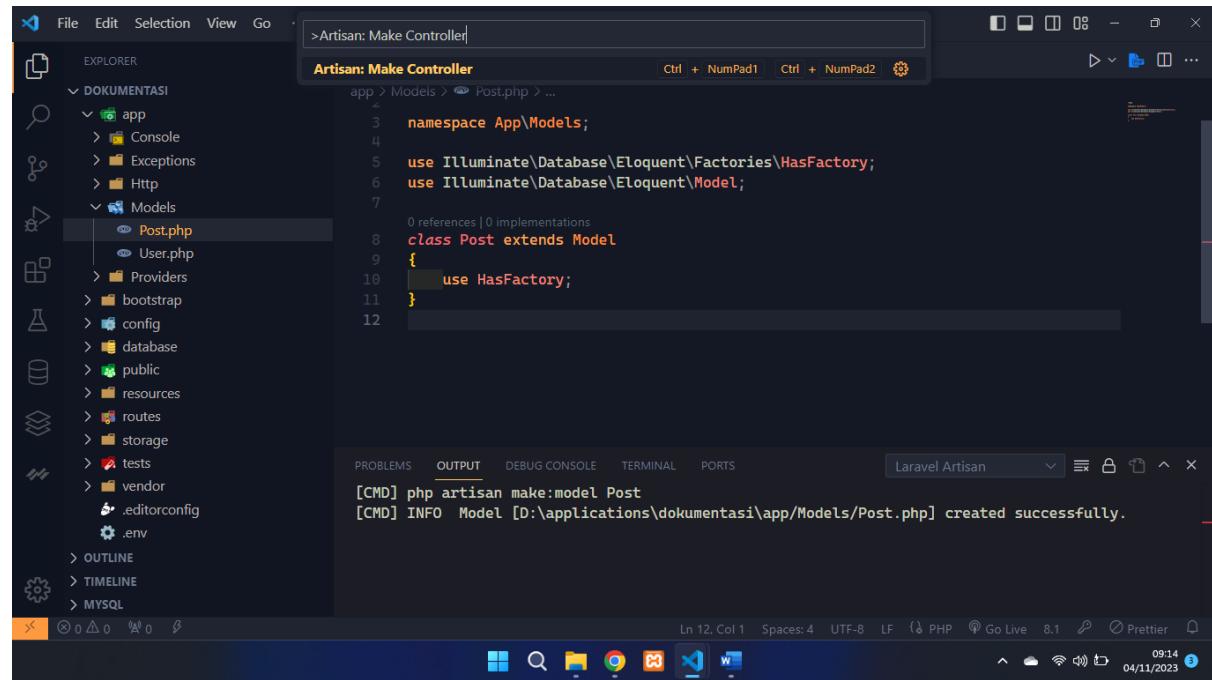
class Post extends Model
{
    use HasFactory;
}

Ln 12, Col 1 Spaces: 4 UTF-8 LF ⚡ PHP Go Live 8.1 ⚡ Prettier 09:04 04/11/2023
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Laravel Artisan [CMD] php artisan make:model Post [CMD] INFO Model [D:\applications\dokumentasi\app\Models\Post.php] created successfully.

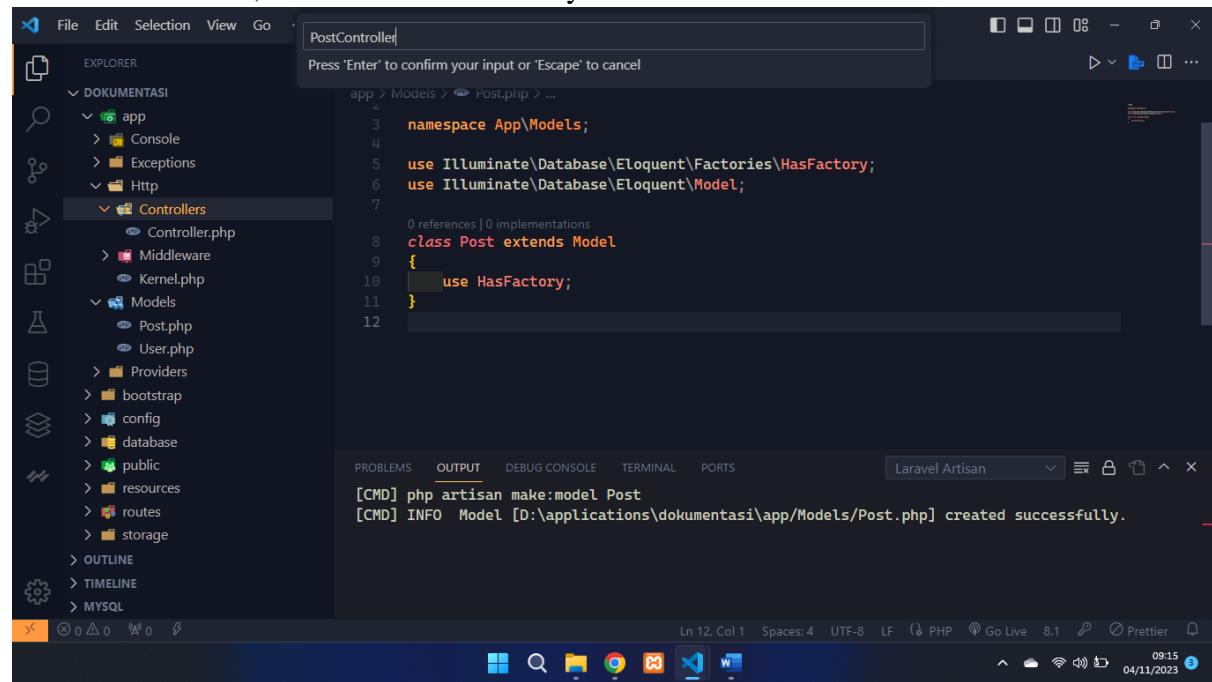
Cara membuat Controller

Buka command palette dengan menekan Ctrl + shift + p Ketik Artisan: Make Controller lalu klik



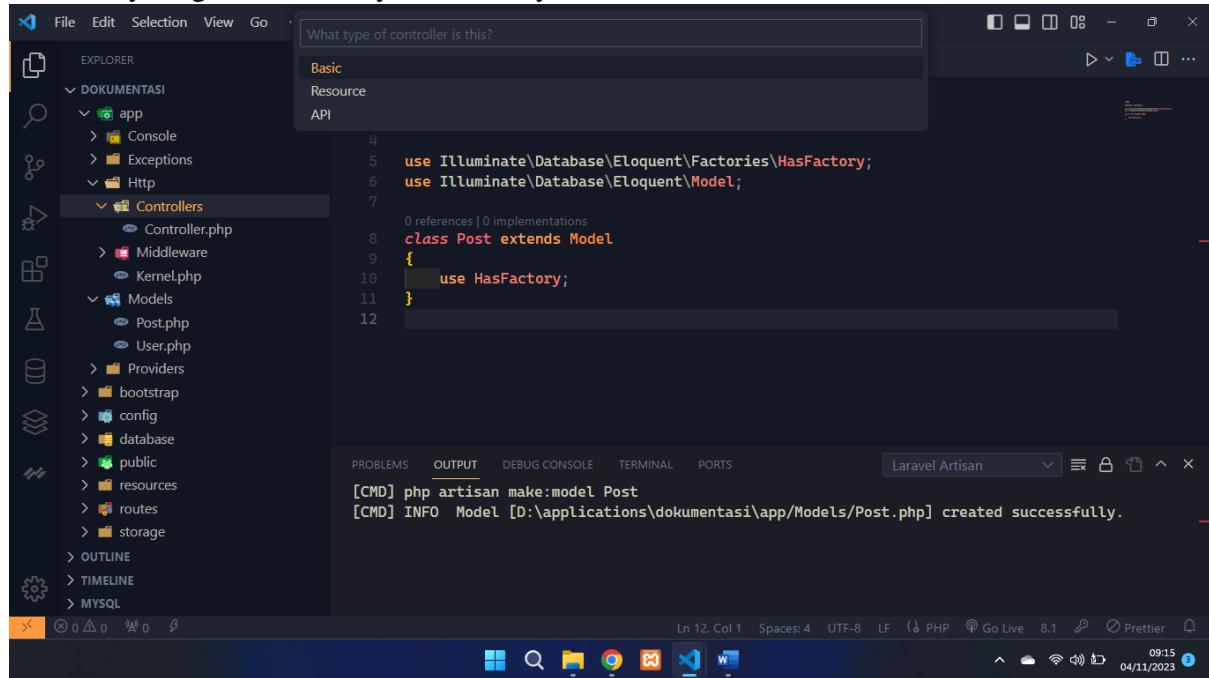
```
>Artisan: Make Controller|  
Artisan: Make Controller  
app > Models > Post.php > ...  
2  
3 namespace App\Models;  
4  
5 use Illuminate\Database\Eloquent\Factories\HasFactory;  
6 use Illuminate\Database\Eloquent\Model;  
7  
8 0 references | 0 implementations  
9 class Post extends Model  
10 {  
11     use HasFactory;  
12 }  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
[CMD] php artisan make:model Post  
[CMD] INFO Model [D:\applications\dokumentasi\app\Models\Post.php] created successfully.  
Ln 12, Col 1 Spaces: 4 UTF-8 LF ⚡ PHP ⚡ Go Live 8.1 ⚡ Prettier 09:14 04/11/2023
```

Isi nama controller, disini nama controllernya PostController



```
PostController|  
Press 'Enter' to confirm your input or 'Escape' to cancel  
app > Models > Post.php > ...  
2  
3 namespace App\Models;  
4  
5 use Illuminate\Database\Eloquent\Factories\HasFactory;  
6 use Illuminate\Database\Eloquent\Model;  
7  
8 0 references | 0 implementations  
9 class Post extends Model  
10 {  
11     use HasFactory;  
12 }  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
[CMD] php artisan make:model Post  
[CMD] INFO Model [D:\applications\dokumentasi\app\Models\Post.php] created successfully.  
Ln 12, Col 1 Spaces: 4 UTF-8 LF ⚡ PHP ⚡ Go Live 8.1 ⚡ Prettier 09:15 04/11/2023
```

Disini jika anda ingin otomatis dibuatkan isi dari controller, bisa pilih resource/api namun karena saya ingin membuatnya sendiri saya memilih basic



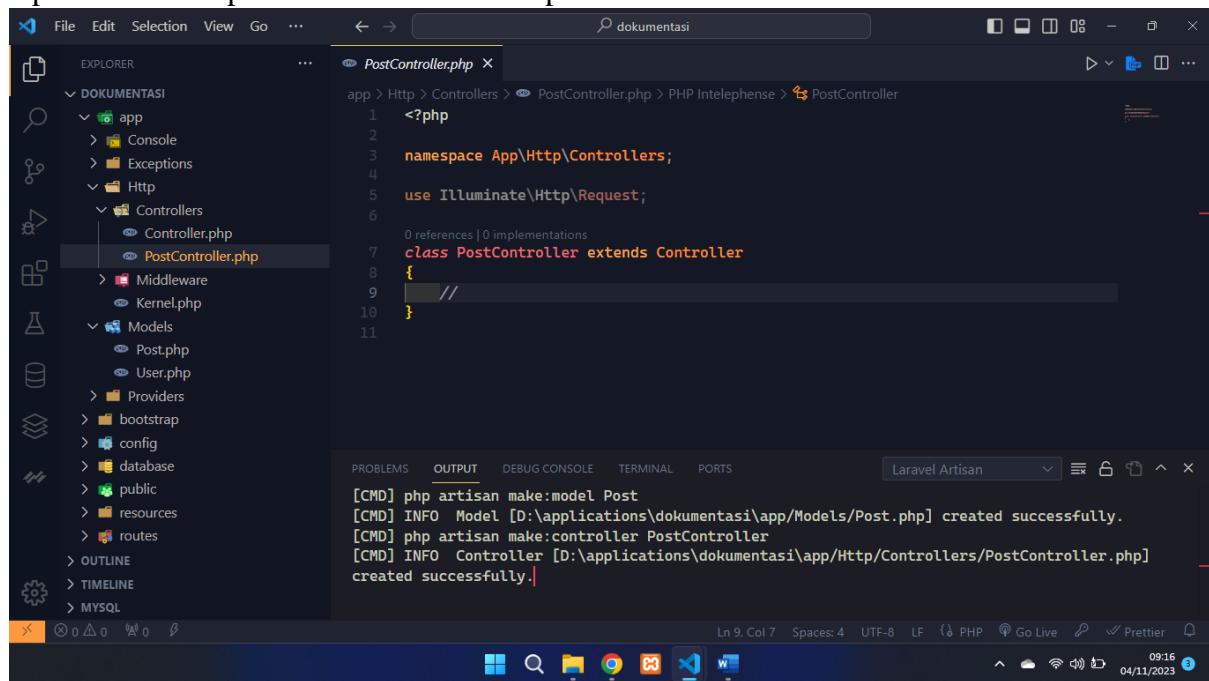
The screenshot shows the Visual Studio Code interface. In the center, a search bar asks "What type of controller is this?" with options: Basic, Resource, and API. Below it, a code editor displays the following PHP code:

```
4 use Illuminate\Database\Eloquent\Factories\HasFactory;
5 use Illuminate\Database\Eloquent\Model;
6
7 0 references | 0 implementations
8 class Post extends Model
9 {
10     use HasFactory;
11 }
```

The left sidebar shows a project structure under "DOKUMENTASI" with "app", "Http", and "Controllers" folders. The "Controllers" folder contains "Controller.php" and "PostController.php". The "Http" folder contains "Middleware", "Kernel.php", and "Controllers" (which contains "Controller.php"). The "Models" folder contains "Post.php" and "User.php". Other sections like "Providers", "bootstrap", "config", "database", "public", "resources", "routes", and "storage" are also listed.

At the bottom, the terminal shows the command: [CMD] php artisan make:model Post and its output: [CMD] INFO Model [D:\applications\dokumentasi\app\Models\Post.php] created successfully.

Apabila sudah seperti ini maka controller pun sudah terbuat.



The screenshot shows the Visual Studio Code interface. The code editor now displays the "PostController.php" file:

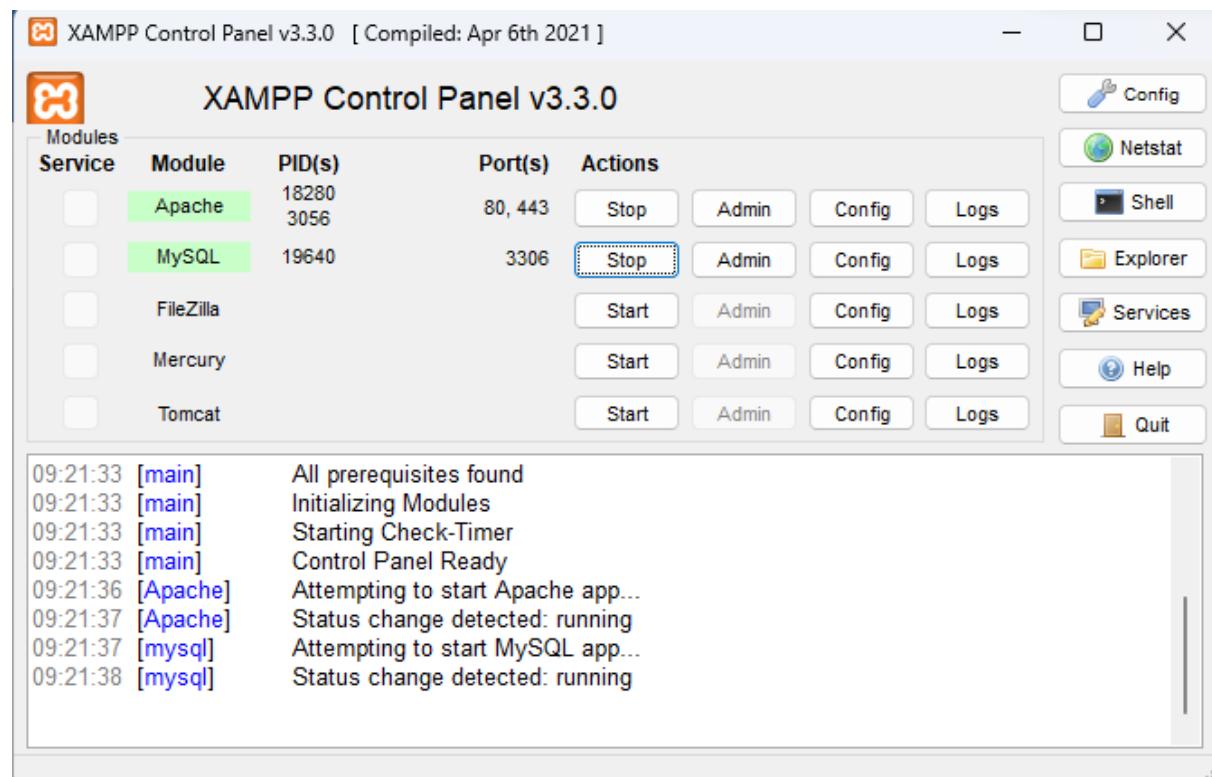
```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 0 references | 0 implementations
8 class PostController extends Controller
9 {
10     /**
11 }
```

The left sidebar shows the same project structure as the previous screenshot. The terminal at the bottom shows the command: [CMD] php artisan make:controller PostController and its output: [CMD] INFO Controller [D:\applications\dokumentasi\app\Http\Controllers\PostController.php] created successfully.

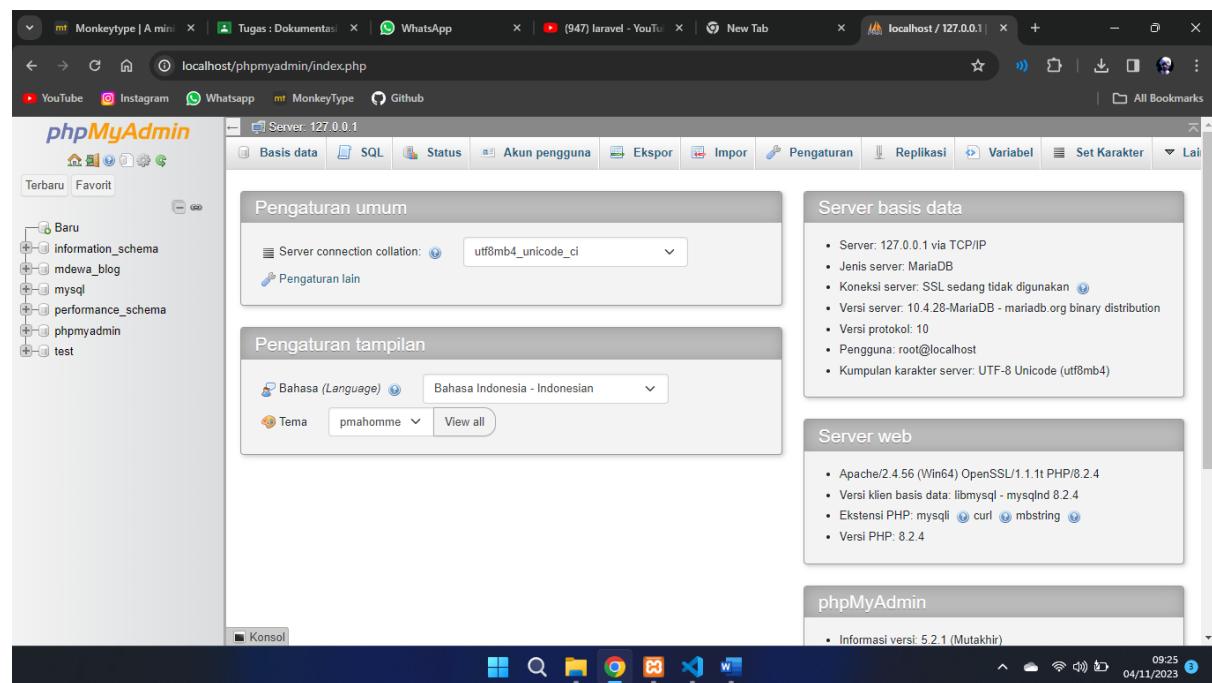
5. Pembuatan Database & Seed

Cara membuat database

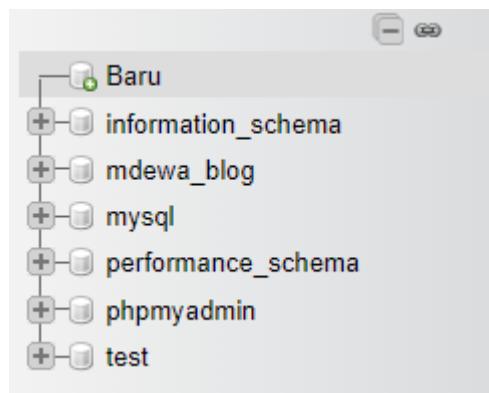
Pastikan service database anda sudah berjalan disini saya menggunakan MySQL yang sudah ada di XAMPP seperti dibawah kemudian klik “Admin” ini akan membuka phpMyAdmin di browser default.



Tampilan awal yang akan terlihat seperti dibawah



Disidebar sebelah kiri klik “Baru”



Kemudian akan terbuka halaman untuk membuat database baru, isi nama database sesuai dengan yang kalian inginkan disini saya mengisi dengan dokumentasi_blog kemudian klik “Buat”

Kemudian jika sudah terbuka halaman seperti ini maka artinya database kita sudah terbuat

Kembali ke VsCode lalu buka file .env kemudian ganti value dari DB_DATABASE yang semulanya Laravel menjadi nama sesuai dengan yang tadi kita buat yaitu dokumentasi_blog

The screenshot shows a Windows 10 desktop with the Visual Studio Code application open. The title bar reads "dokumentasi". The left sidebar (Explorer) lists project files: .env, .env.example, .gitattributes, .gitignore, artisan, composer.json, composer.lock, package.json, phpunit.xml, README.md, and vite.config.js. The bottom left shows icons for Outline, Timeline, and MySQL. The main editor area displays the contents of ".env" file:

```
LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dokumentasi_blog
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

The bottom right shows the terminal output for running Laravel Artisan commands:

```
[CMD] php artisan make:model Post
[CMD] INFO Model [D:\applications\dokumentasi\app\Models\Post.php] created successfully.
[CMD] php artisan make:controller PostController
[CMD] INFO Controller [D:\applications\dokumentasi\app\Http\Controllers\PostController.php] created successfully.
```

The status bar at the bottom indicates: Line 14, Col 29, Spaces: 4, UTF-8, LF, Properties, Go Live, Prettier, and the date/time 04/11/2023 09:30.

Buka terminal ketik php artisan migrate, ini akan membuat/menambahkan table default yang dimiliki Laravel

The screenshot shows the Visual Studio Code interface with a Laravel project open. The left sidebar displays the file structure:

- EXPLORER
- DOKUMENTASI
 - Kernel.php
 - Models
 - Post.php
 - User.php
 - Providers
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - 2014_10_12_0000...
 - 2014_10_12_1000...
 - 2019_08_19_0000...
 - 2019_12_14_0000...
 - seeders
 - .gitignore
- public
- resources
- routes
- OUTLINE
- TIMELINE
- MYSQL

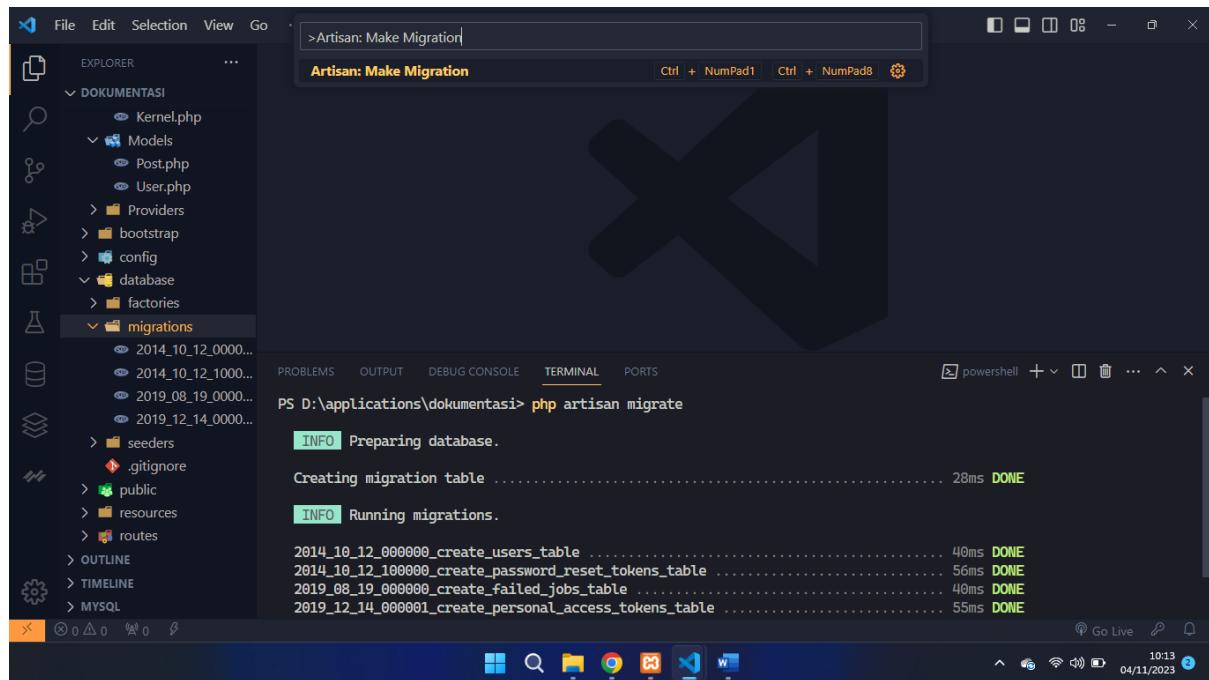
The terminal tab shows the command `php artisan migrate` being run, and the output indicates the successful creation of migration tables and their execution.

```
PS D:\applications\dokumentasi> php artisan migrate
INFO Preparing database.

Creating migration table ..... 28ms DONE
INFO Running migrations.

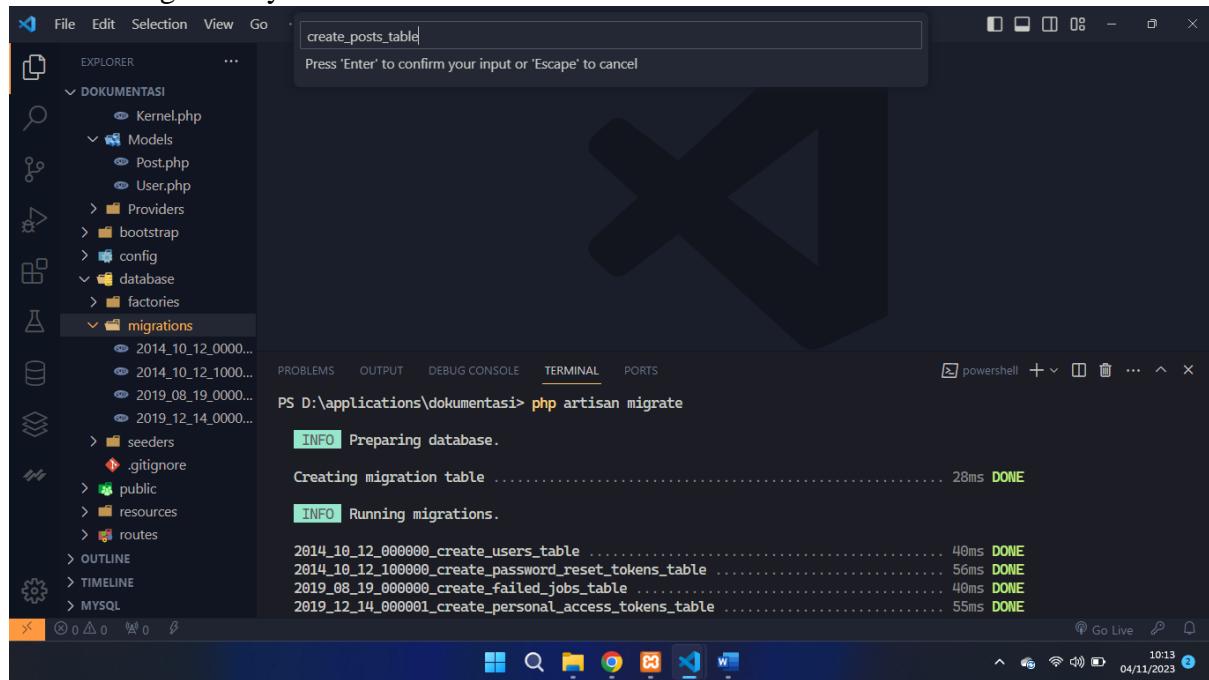
2014_10_12_000001_create_users_table ..... 40ms DONE
2014_10_12_100001_create_password_reset_tokens_table ..... 56ms DONE
2019_08_19_000001_create_failed_jobs_table ..... 40ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 55ms DONE
```

Kemudian untuk membuat table kita sendiri buka command pallete dengan tekan Ctrl+Shift+P ketik Artisan: Make Migration lalu klik



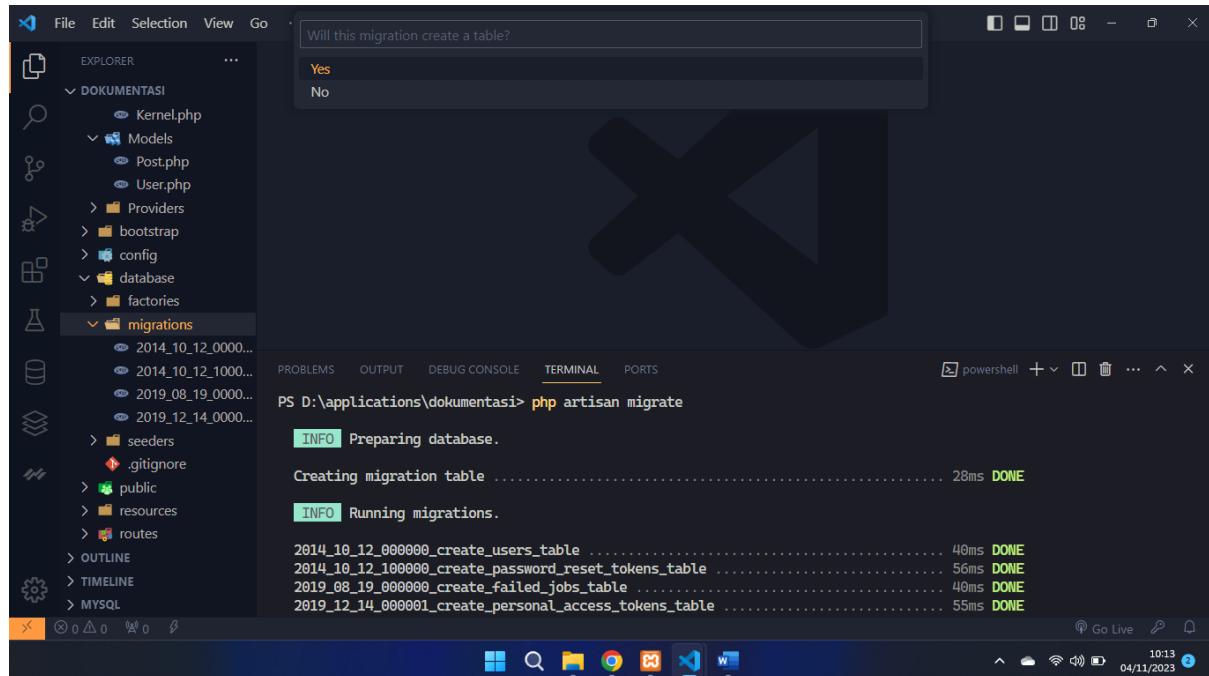
```
File Edit Selection View Go >Artisan: Make Migration| Artisan: Make Migration [Ctrl + NumPad1] [Ctrl + NumPad8] ...
EXPLORER DOKUMENTASI Kernel.php Models Post.php User.php Providers bootstrap config database factories migrations 2014_10_12_0000... 2014_10_12_1000... 2019_08_19_0000... 2019_12_14_0000... seeder .gitignore public resources routes OUTLINE TIMELINE MYSQL
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\applications\dokumentasi> php artisan migrate
INFO Preparing database.
Creating migration table ..... 28ms DONE
INFO Running migrations.
2014_10_12_000000_create_users_table ..... 40ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 56ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 40ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 55ms DONE
10:13 04/11/2023
```

Isi nama migrationnya

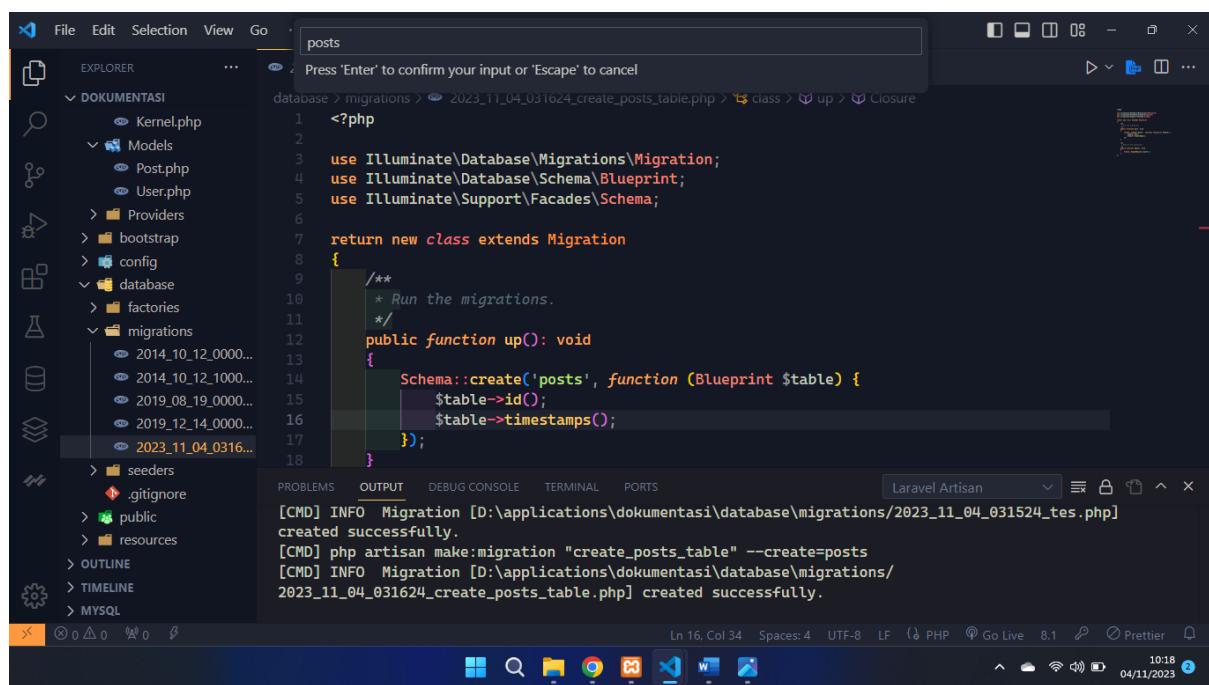


```
File Edit Selection View Go >create_posts_table| Press 'Enter' to confirm your input or 'Escape' to cancel
EXPLORER DOKUMENTASI Kernel.php Models Post.php User.php Providers bootstrap config database factories migrations 2014_10_12_0000... 2014_10_12_1000... 2019_08_19_0000... 2019_12_14_0000... seeder .gitignore public resources routes OUTLINE TIMELINE MYSQL
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\applications\dokumentasi> php artisan migrate
INFO Preparing database.
Creating migration table ..... 28ms DONE
INFO Running migrations.
2014_10_12_000000_create_users_table ..... 40ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 56ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 40ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 55ms DONE
10:13 04/11/2023
```

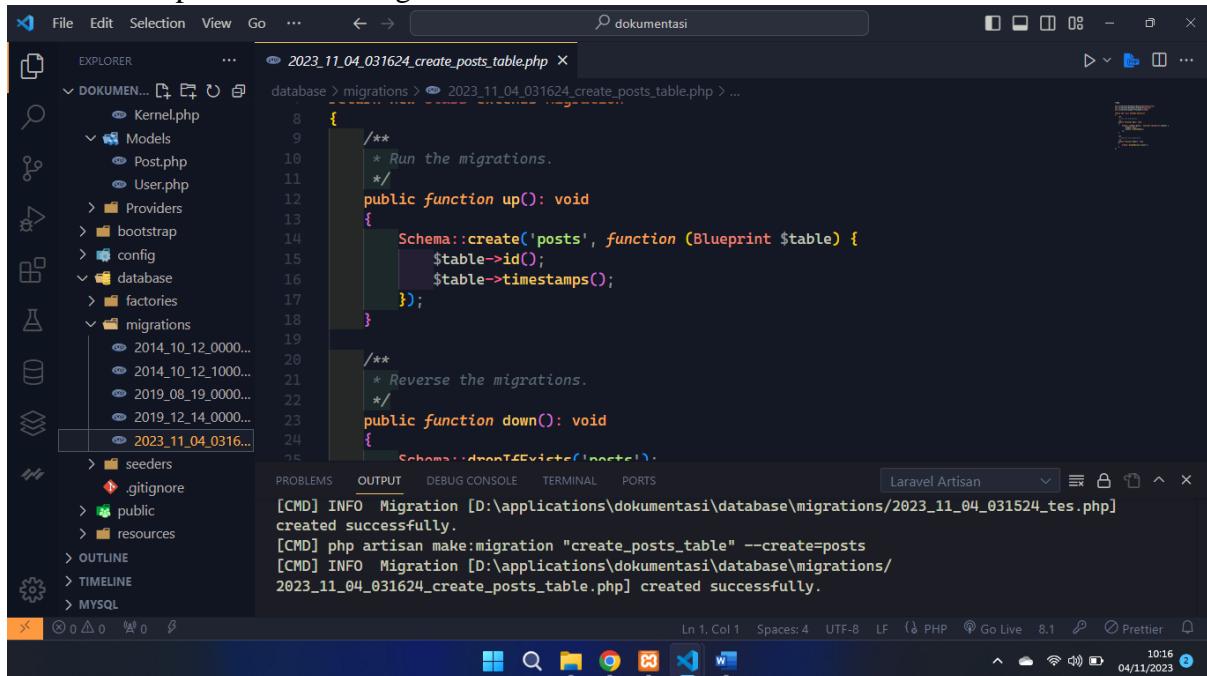
Pilih



Isi nama table untuk di database



Jika sudah seperti ini maka migration kita sudah terbuat



```
2023_11_04_031624_create_posts_table.php

{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

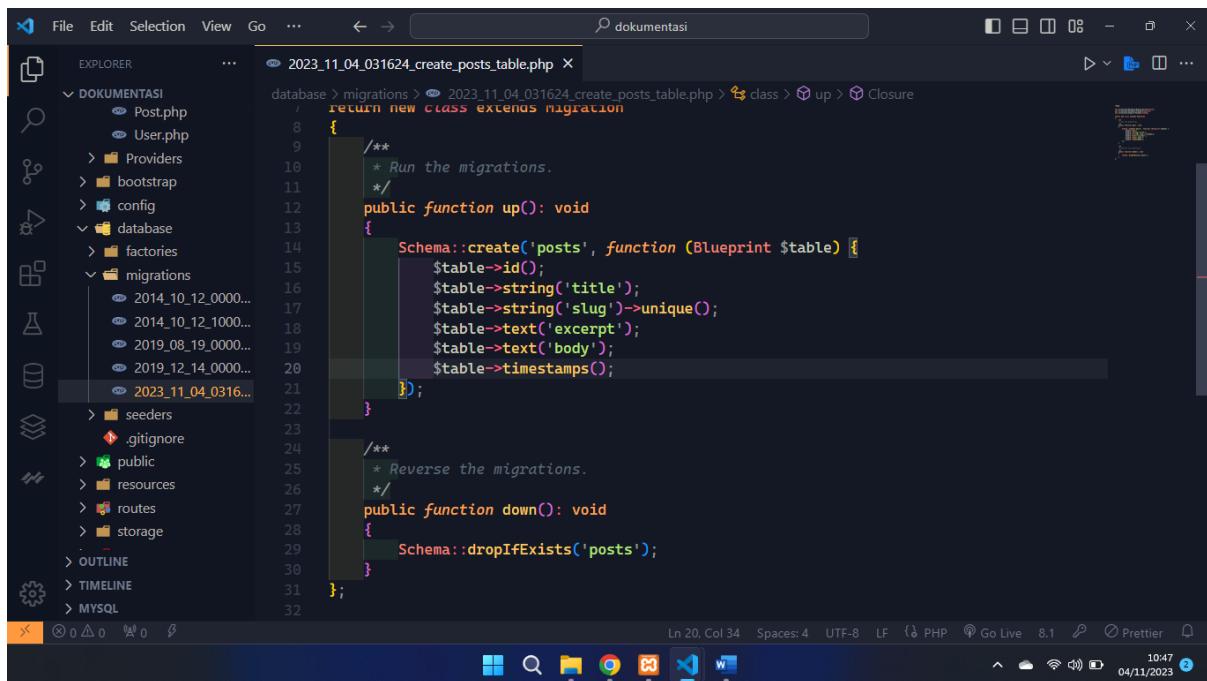
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('posts');
    }
}
```

Laravel Artisan

```
[CMD] INFO Migration [D:\applications\dokumentasi\database\migrations\2023_11_04_031524_tes.php]
created successfully.
[CMD] php artisan make:migration "create_posts_table" --create=posts
[CMD] INFO Migration [D:\applications\dokumentasi\database\migrations\2023_11_04_031624_create_posts_table.php] created successfully.
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF ⚡ PHP Go Live 8.1 ⚡ Prettier 10:16 04/11/2023

Untuk menambahkan kolom bisa ditambahkan di method up, untuk codenya yaitu \$table->tipeData->('namakolom'); sebagai contoh, saya menambahkan kolom title, slug, excerpt, dan body,



```
2023_11_04_031624_create_posts_table.php

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->string('slug')->unique();
            $table->text('excerpt');
            $table->text('body');
            $table->timestamps();
        });
    }

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

Ln 20, Col 34 Spaces: 4 UTF-8 LF ⚡ PHP Go Live 8.1 ⚡ Prettier 10:47 04/11/2023

Setelah itu buka Kembali terminal lalu ketik php artisan migrate:fresh, ini akan menjalankan proses yaitu mendrop semua table lalu membuatnya Kembali.

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows the project structure with files like Post.php, User.php, Providers, bootstrap, config, database, factories, migrations (containing 2014_10_12_00000..., 2014_10_12_10000..., 2019_08_19_00000..., 2019_12_14_00000..., and 2023_11_04_031624_create_posts_table.php), seeders, .gitignore, public, resources, routes, storage, tests, vendor, editorconfig, env, .env.example, .gitattributes, Timeline, and MySQL.
- Terminal:** Displays the command "PS D:\applications\dokumentasi> php artisan migrate:fresh" and its execution log:
 - Dropping all tables 160ms DONE
 - INFO Preparing database.
 - Creating migration table 20ms DONE
 - INFO Running migrations.
 - 2014_10_12_000000_create_users_table 33ms DONE
 - 2014_10_12_100000_create_password_reset_tokens_table 47ms DONE
 - 2019_08_19_000000_create_failed_jobs_table 33ms DONE
 - 2019_12_14_000001_create_personal_access_tokens_table 47ms DONE
 - 2023_11_04_031624_create_posts_table 50ms DONE
- Bottom Status Bar:** Shows the date (04/11/2023), time (10:49), and system icons.

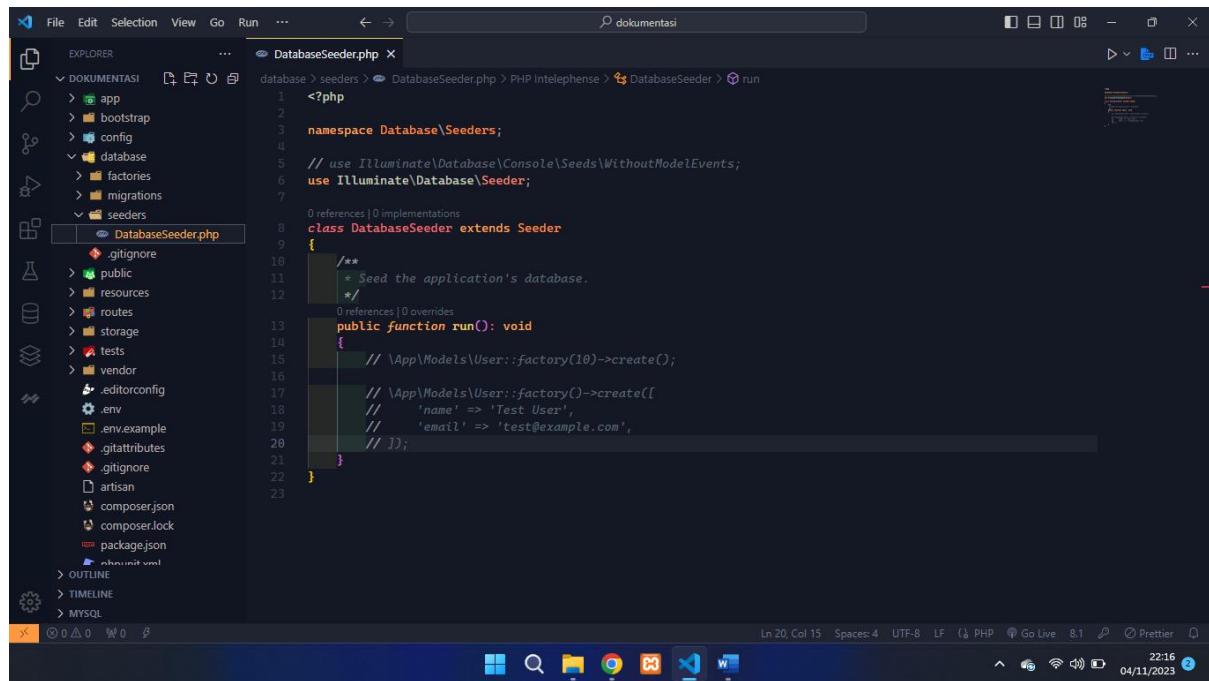
Dan jika dilihat di phpMyAdmin Kembali maka sudah terdapat table posts yang memiliki kolom sesuai dengan yang kita buat di migration tadi.

The screenshot shows the phpMyAdmin interface with the following details:

- Left Sidebar:** Shows the database structure with databases like Baru, dokumentasi_blog, failed_jobs, migrations, password_reset_tokens, personal_access_tokens, posts, users, information_schema, mdewa_blog, mysql, performance_schema, phpmyadmin, and test.
- Current Database:** dokumentasi_blog
- Table:** posts
- Table Structure:** Shows columns: id, title, slug, excerpt, body, created_at, and updated_at.
- Query Editor:** Displays the SQL query: "SELECT * FROM `posts`".
- Buttons:** Profil, Edit dikotak, Ubah, Jelaskan SQL, Buat kode PHP, Segarkan, Tambahkan, Eksport, Import, Hak Akses, Operasi, Pelacakan, Trigger.
- Operasi hasil kueri:** Buttons for Markahi kueri SQL ini, Buat tampilan, and Izinkan semua pengguna untuk mengakses markah ini.
- Konsol:** Bottom panel for running commands.

Pembuatan Seeder

Buka file DatabaseSeeder.php di dalam folder database\seeders

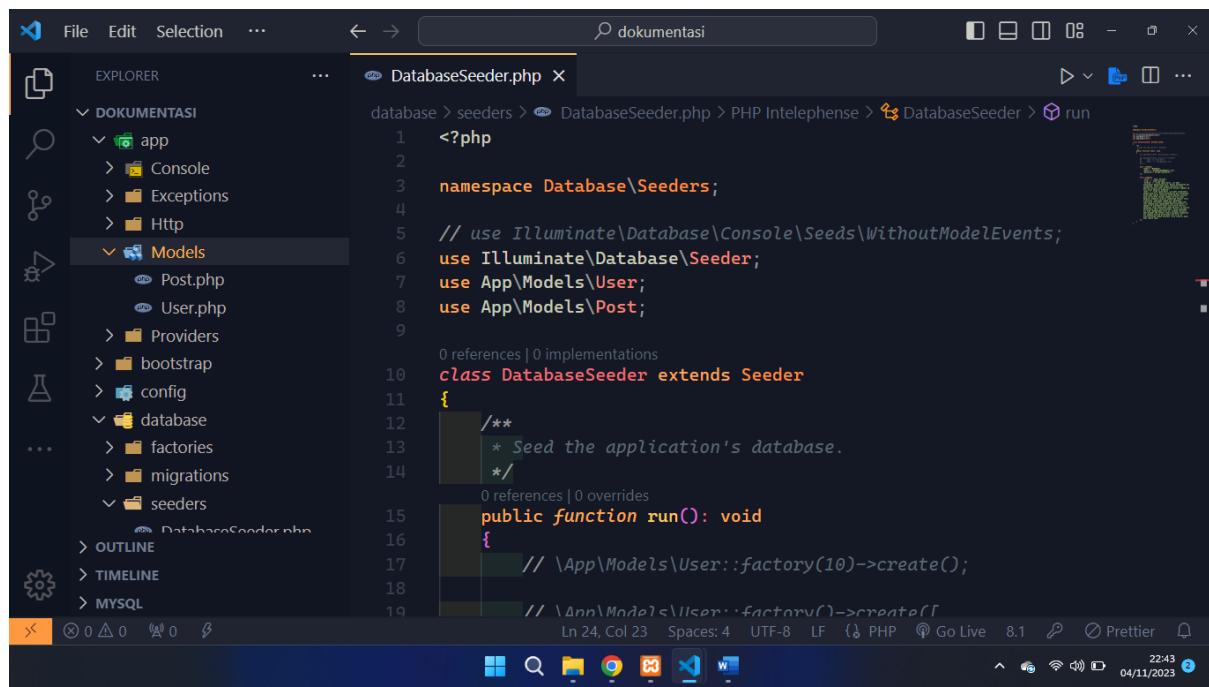


```
<?php
namespace Database\Seeders;

// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        // \App\Models\User::factory(10)->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
    }
}
```

Dalam kasus ini saya membuat data user dan post. Pertama-tama masukkan direktori dari model yang akan kita gunakan, disini yaitu model user dan model post

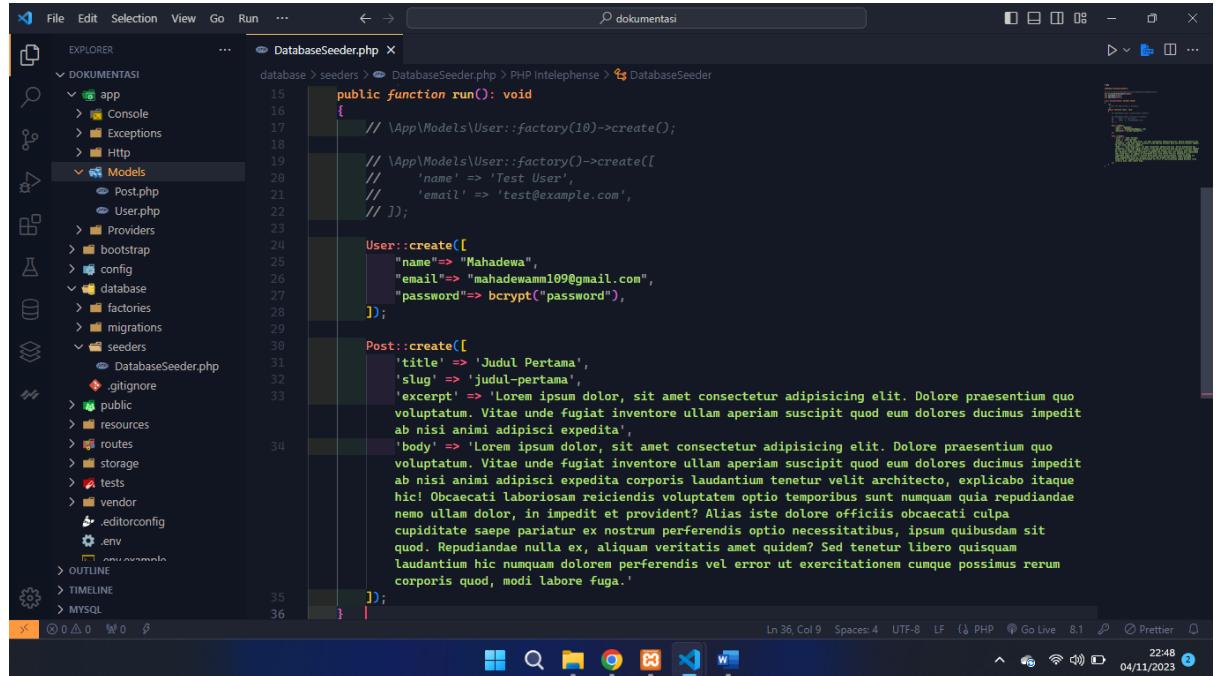


```
<?php
namespace Database\Seeders;

// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\User;
use App\Models\Post;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        // \App\Models\User::factory(10)->create();
        // \App\Models\User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
    }
}
```

Jika sudah, baru kita bisa membuat seeder untuk user dan post seperti dibawah. Disini sesuaikan isi dari seed dengan kolom di database yang akan kita buat.



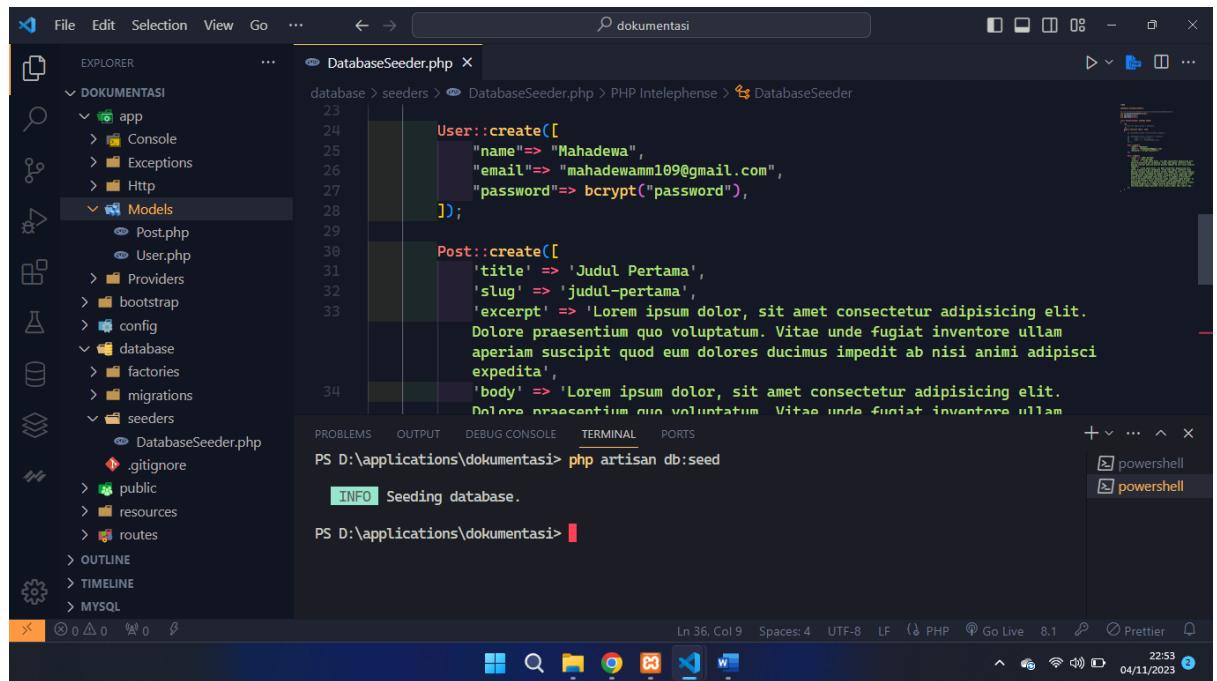
```
public function run(): void
{
    // \App\Models\User::factory(10)->create();

    // \App\Models\User::factory()->create([
    //     'name' => 'Test User',
    //     'email' => 'test@example.com',
    // ]);

    User::create([
        "name"=> "Mahadewa",
        "email"=> "mahadewam109@gmail.com",
        "password"=> bcrypt("password"),
    ]);

    Post::create([
        'title' => 'Judul Pertama',
        'slug' => 'judul-pertama',
        'excerpt' => 'Lorem ipsum dolor, sit amet consectetur adipisicing elit. Dolore praesentium quo voluptatum. Vitae unde fugiat inventore ullam aperiam suscipit quod eum dolores ducimus impedit ab nisi animi adipisci expedita',
        'body' => 'Lorem ipsum dolor, sit amet consectetur adipisicing elit. Dolore praesentium quo voluptatum. Vitae unde fugiat inventore ullam aperiam suscipit quod eum dolores ducimus impedit ab nisi animi adipisci expedita corporis laudantium tenetur velit architecto, explicabo itaque hic! Obcaecati laboriosam reiciendis voluptatem optio temporibus sunt numquam quia repudiandae nemo ullam dolor, in impedit et provident? Alias iste dolore officiis obcaecati culpa cupiditate saepe pariatur ex nostrum perferendis optio necessitatibus, ipsum quibusdam sit quod. Repudiandae nulla ex, aliquam veritatis amet quidem? Sed tenetur libero quisquam laudantium hic numquam dolorem perferendis vel error ut exercitationem cumque possimus rerum corporis quod, modi labore fuga.'
    ]);
}
```

Jika sudah buka terminal lalu ketik php artisan db:seed

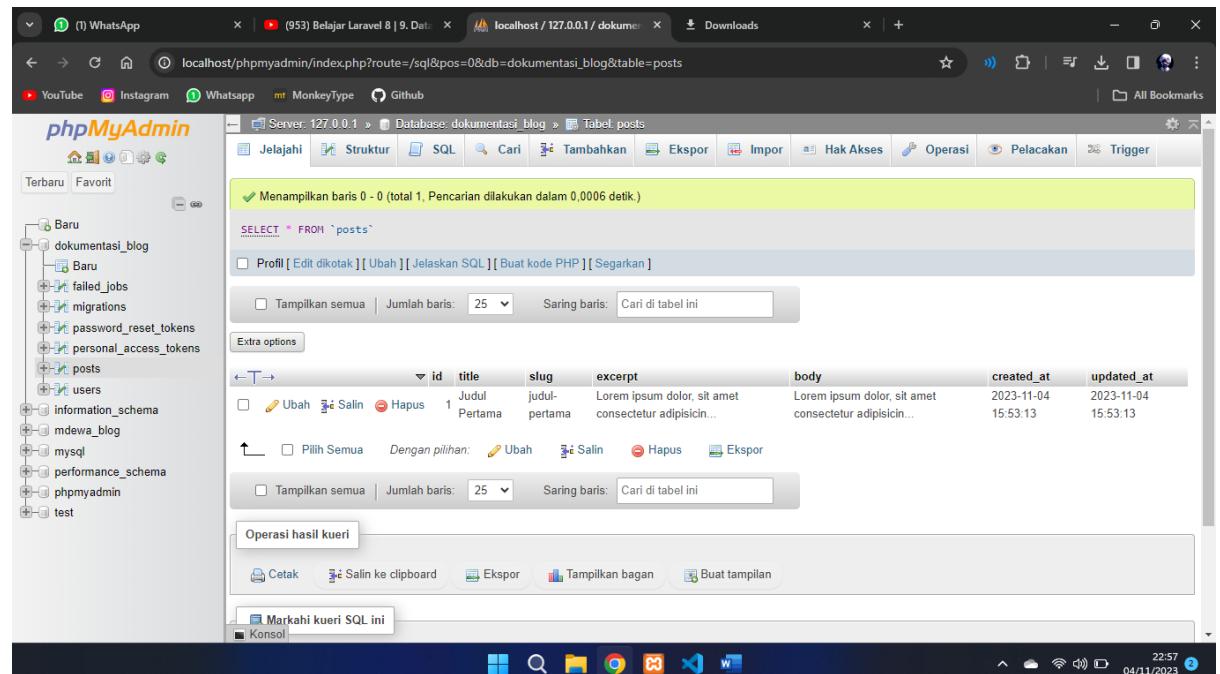


```
User::create([
    "name"=> "Mahadewa",
    "email"=> "mahadewam109@gmail.com",
    "password"=> bcrypt("password"),
]);

Post::create([
    'title' => 'Judul Pertama',
    'slug' => 'judul-pertama',
    'excerpt' => 'Lorem ipsum dolor, sit amet consectetur adipisicing elit. Dolore praesentium quo voluptatum. Vitae unde fugiat inventore ullam aperiam suscipit quod eum dolores ducimus impedit ab nisi animi adipisci expedita',
    'body' => 'Lorem ipsum dolor, sit amet consectetur adipisicing elit. Dolore praesentium quo voluptatum. Vitae unde fugiat inventore ullam aperiam suscipit quod eum dolores ducimus impedit ab nisi animi adipisci expedita corporis laudantium tenetur velit architecto, explicabo itaque hic! Obcaecati laboriosam reiciendis voluptatem optio temporibus sunt numquam quia repudiandae nemo ullam dolor, in impedit et provident? Alias iste dolore officiis obcaecati culpa cupiditate saepe pariatur ex nostrum perferendis optio necessitatibus, ipsum quibusdam sit quod. Repudiandae nulla ex, aliquam veritatis amet quidem? Sed tenetur libero quisquam laudantium hic numquam dolorem perferendis vel error ut exercitationem cumque possimus rerum corporis quod, modi labore fuga.'
]);

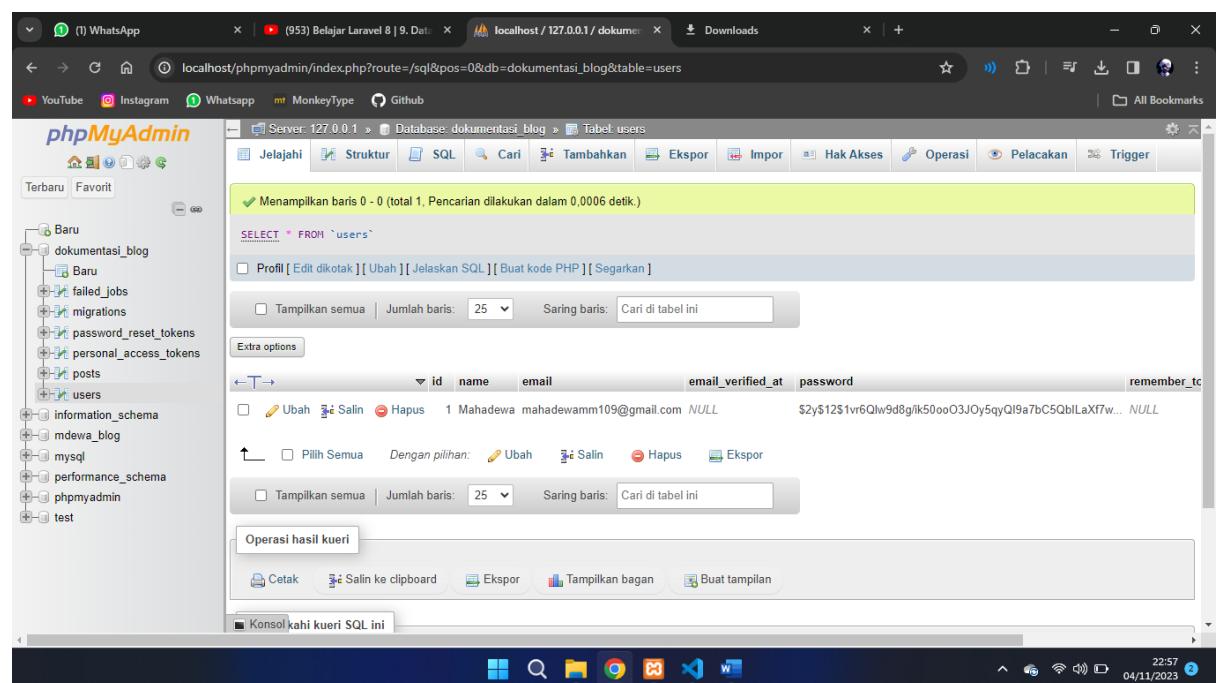
PS D:\applications\dokumentasi> php artisan db:seed
INFO  Seeding database.
PS D:\applications\dokumentasi>
```

Apabila pesan yang muncul adalah Seeding database, maka artinya seed sudah berhasil masuk ke database kita dan jika kita lihat di table user dan post sudah terdapat data yang kita buat di seeder.



The screenshot shows the phpMyAdmin interface for the 'posts' table in the 'dokumentasi_blog' database. The table has columns: id, title, slug, excerpt, body, created_at, and updated_at. One row is displayed:

| | id | title | slug | excerpt | body | created_at | updated_at |
|--|----|---------------|---------------|---|---|---------------------|---------------------|
| | 1 | Judul Pertama | judul-pertama | Lorem ipsum dolor, sit amet, consectetur adipisicing... | Lorem ipsum dolor, sit amet, consectetur adipisicing... | 2023-11-04 15:53:13 | 2023-11-04 15:53:13 |



The screenshot shows the phpMyAdmin interface for the 'users' table in the 'dokumentasi_blog' database. The table has columns: id, name, email, email_verified_at, password, and remember_token. One row is displayed:

| | id | name | email | email_verified_at | password | remember_token |
|--|----|----------|-------------------------|-------------------|--|----------------|
| | 1 | Mahadewa | mahadewamm109@gmail.com | NULL | \$2y\$12\$1vr6Qlw9d8glik50ooO3JOy5qyQI9a7bC5QbILaXf7w... | NULL |