

# **LAPORAN PRAKTEK**

## **Praktek Pemrograman Web Semester 4**



**Nama : Milga Tibalimeten**  
**Nim : C030321043**  
**Kelas : TI-4B**  
**Dosen Pengampu : ARIFIN NOOR ASYIKIN, ST, M.T**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI BANJARMASIN**

**2021/2022**

## PRAKTIKUM 3

### 1.1 Judul

Model View Controller atau MVC

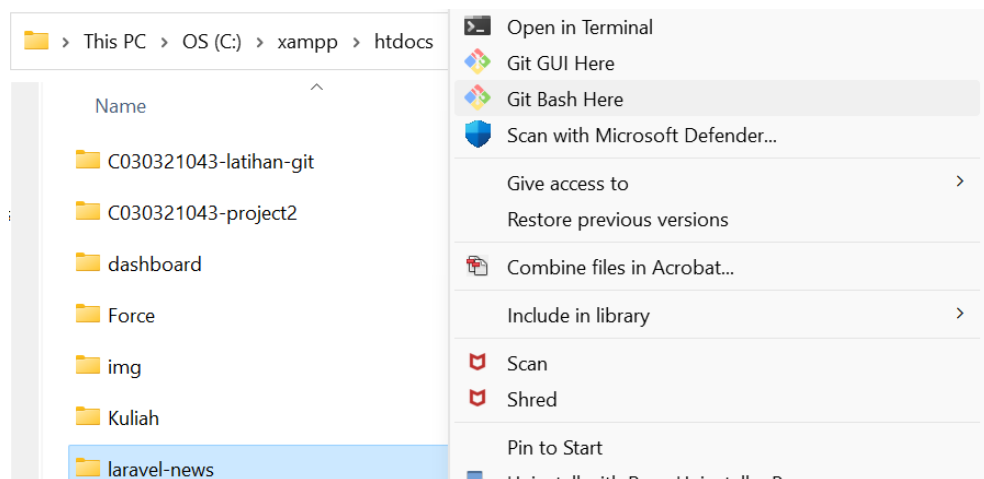
### 1.2 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami teknik Model View Controller (MVC) menggunakan bahasa pemrograman PHP.
- Merancang program menggunakan teknik Model View Controller (MVC).

### 1.3 Install Laravel

installasi pada folder C:\xampp\htdocs. Maka pada direktori ini kami akan memasukan perintah ke dalam command prompt menggunakan bantuan aplikasi git dengan cara seperti dibawah ini.



jika sudah terbuka aplikasi GIT silahkan masukan perintah dibawah ini:

**composer create-project laravel/laravel:^9.0 c030320999-project3**

pada perintah diatas, kita menggunakan composer untuk membuat sebuah project laravel menggunakan versi 9 dan project yang akan dibuat diberi nama dengan laravel-news. pada praktikum sebelumnya kita menggunakan perintah maka akan terinstall adalah laravel 10

**composer create-project laravel/laravel c030320999-project3**

```
MINGW64/c/xampp/htdocs/ X + v
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news
$ composer create-project laravel/laravel:^9.0 C030321043-project3
Creating a "laravel/laravel:^9.0" project at "./C030321043-project3"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v9.5.2)
- Downloading laravel/laravel (v9.5.2)
- Installing laravel/laravel (v9.5.2): Extracting archive
Created project in C:\xampp\htdocs\laravel-news\C030321043-project3
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 108 installs, 0 updates, 0 removals
- Locking brick/math (0.11.0)
- Locking dflydev/dot-access-data (v3.0.2)
- Locking doctrine/inflector (2.0.6)
- Locking doctrine/instantiator (2.0.0)
- Locking doctrine/lexer (3.0.0)
- Locking dragonmantank/cron-expression (v3.3.2)
- Locking egulias/email-validator (4.0.1)
- Locking fakerphp/faker (v1.21.0)
```

```
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
INFO Discovering packages.
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE
82 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
INFO No publishable resources for tag [laravel-assets].
No security vulnerability advisories found
> @php artisan key:generate --ansi
INFO Application key set successfully.
```

## 2. Symbolic Link

Laravel akan menyimpan file-file yang di upload kedalam folder yang bernama storage. akan tetapi laravel secara default hanya bisa membaca file yang berada di dalam folder public. dengan menjalankan perintah artisan `storage:link`, maka kita akan membuatkan sebuah link atau shortcut dari folder storage kedalam folder public.

```
MINGW64/c/xampp/htdocs/ X + v
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news
$ cd C030321043-project3
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$ |
```

Dengan begini maka laravel bisa membaca semua file-file yang ada di dalam folder storage melalui folder public. silahkan masukkan perintah berikut ini di dalam terminal/CMD:

**php artisan storage:link**

jika proses diatas sudah selesai, maka akan ditandai seperti berikut:

```
MINGW64/c:/xampp/htdocs/ x + v
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c:/xampp/htdocs/laravel-news
$ cd C030321043-project3
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c:/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan storage:link
INFO The [C:\xampp\htdocs\laravel-news\C030321043-project3\public\storage] link has been connected to [C:\xampp\htdocs\laravel-news\C030321043-project3\storage\app\public].
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c:/xampp/htdocs/laravel-news/C030321043-project3
$
```

This PC > OS (C:) > xampp > htdocs > laravel-news > C030321043-project3				
Name	Date modified	Type	Size	
app	1/31/2023 11:05 PM	File folder		
bootstrap	1/31/2023 11:05 PM	File folder		
config	1/31/2023 11:05 PM	File folder		
database	1/31/2023 11:05 PM	File folder		
lang	1/31/2023 11:05 PM	File folder		
public	1/31/2023 11:05 PM	File folder		
resources	1/31/2023 11:05 PM	File folder		
routes	1/31/2023 11:05 PM	File folder		
storage	1/31/2023 11:05 PM	File folder		
tests	1/31/2023 11:05 PM	File folder		
vendor	4/18/2023 8:50 PM	File folder		
.editorconfig	1/31/2023 11:05 PM	Editor Config Source ...	1 KB	
.env	4/18/2023 8:50 PM	ENV File	2 KB	
.env.example	1/31/2023 11:05 PM	EXAMPLE File	2 KB	
.gitattributes	1/31/2023 11:05 PM	Git Attributes Source ...	1 KB	
.gitignore	1/31/2023 11:05 PM	Git Ignore Source File	1 KB	

maka bisa melihat di dalam folder public akan otomatis ter-genarte sebuah folder dengan nama storage.

This PC > OS (C:) > xampp > htdocs > laravel-news > C030321043-project3 > public				
Name	Date modified	Type	Size	
storage	4/18/2023 9:15 PM	File folder		
.htaccess	1/31/2023 11:05 PM	HTACCESS File	1 KB	
favicon.ico	1/31/2023 11:05 PM	ICO File	0 KB	
index.php	1/31/2023 11:05 PM	PHP File	2 KB	
robots.txt	1/31/2023 11:05 PM	Text Document	1 KB	

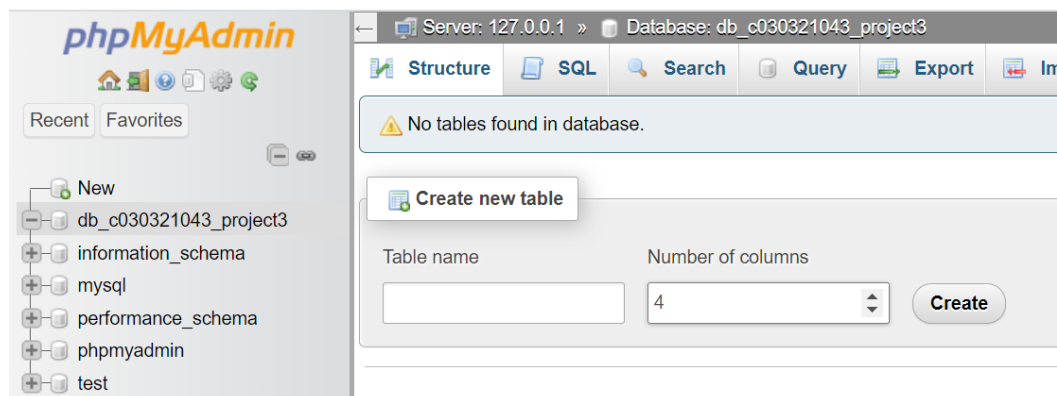
## 1.4 Database

MySQL adalah sebuah DBMS (Database Management System) menggunakan perintah SQL (Structured Query Language) yang banyak digunakan saat ini dalam pembuatan aplikasi berbasis website. MySQL dibagi menjadi dua lisensi, pertama adalah Free Software dimana perangkat lunak dapat diakses oleh siapa saja. Dan kedua adalah Shareware dimana perangkat lunak berpelik memiliki batasan dalam penggunaannya. Diantara manfaat dan keuntungan menggunakan database MySQL adalah. Bersifat Open Source, Mendukung Penggunaan Multi User dan masih banyak lagi.

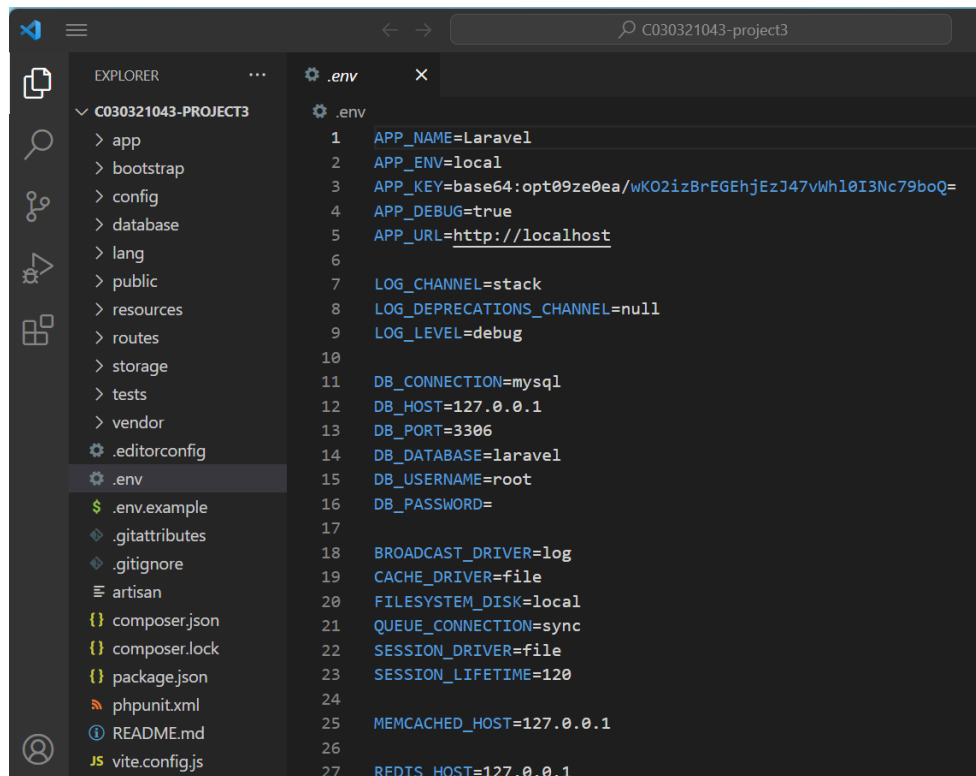
### 1. Koneksi Laravel Ke Mysql

Jika didalam pemrograman php native, tentu saja mengingat bagaimana cara mengkoneksikan sebuah aplikasi dengan MySQL biasanya terletak di dalam file koneksi.php atau connect.php dan sebagainya. Didalamnya kita biasanya menuliskan nama host database, nama user database nama database dan juga password database jika di butuhkan. Seperti halnya dengan php native, laravel juga terdapat koneksi seperti itu dan kita tidak perlu membuat file dengan ekstensi .php, dikarenakan laravel sudah menyediakan fasilitas konfigurasi untuk melakukan koneksi dengan database melalui file .env yang terletak pada folder root project kita. Buka file tersebut dan akan terlihat seperti dibawah ini.

Buatlah database dengan nama db\_nimkalian\_project3



kemudian setting config databasenya

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'C030321043-PROJECT3' with various folders like 'app', 'bootstrap', 'config', 'database', etc., and files like '.env', '.env.example', 'composer.json', etc. The main editor area displays the contents of the '.env' file, which contains configuration settings for a Laravel application, including database credentials, cache settings, and session configuration.

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:opt09ze0ea/wK02izBrEGEHjEzJ47vWhl0I3Nc79boQ=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
```

**MVC** adalah sebuah arsitektur perancangan kode program. Yang tujuannya untuk memecah kode program utama menjadi 3 komponen terpisah dengan tugas yang spesifik. Ketiga komponen tersebut adalah:

- a). Pengaksesan database, disebut sebagai Model.
- b). Tampilan design (user interface), disebut sebagai View.
- c). Alur logika program, disebut sebagai Controller.

Nah, gabungan dari Model View Controller inilah disebut dengan istilah MVC.

Migration merupakan salah satu fitur Laravel yang berfungsi seperti version control untuk database. Melalui fitur ini sebuah team pengembangan web development akan dapat bekerja dalam team untuk mengelola dan modifikasi skema basis data aplikasi. Migration biasanya dipasangkan dengan Schema Builder dari Laravel untuk dengan mudah membangun skema basis data aplikasi Anda. Facade Skema Laravel menyediakan dukungan untuk membuat dan memanipulasi tabel di semua sistem basis data yang didukung Laravel.

### **A. Model dan Migrations**

Berikut adalah perintah untuk membuat Model. Bagian akhir perintah diberi kode migration `-m` dengan tujuan nantinya akan menghasilkan sebuah table di dalam database yang sudah kita buat sebelumnya. Jadi didalam laravel, kita tidak perlu membuat beberapa tabel didalam database dengan manual, dengan menggunakan migration di laravel, proses penanganan table bisa dikerjakan dengan cepat dan tepat.

## B. Struktur Model & Migration

-[1] . Model & Migration Category

-[2]. Model & Migration Post

-[3]. Model & Migration Comment

### Langkah 1 - Membuat Model dan Migration Category

Kita sudah sepakat akan membuat model dan migration sesuai point B, maka kita akan memulai dengan membuat model dengan nama Category. Silahkan masuk direktori project laravel yang sudah dibuat kemudian masukkan perintah seperti dibawah ini :

**php artisan make:model Category -m**

Jika perintah di atas berhasil dijalankan maka akan melakukan generate 2 file baru, yaitu :

```
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan make:model Category -m

INFO Model [C:\xampp\htdocs\laravel-news\C030321043-project3\app\Models\Category.php] created successfully.

INFO Migration [C:\xampp\htdocs\laravel-news\C030321043-project3\database\migrations\2023_04_18_135030_create_categories_table.php] created successfully.

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$
```

#### 1. 1 Model Category.php & Mass Assigment

Silahkan buka file app/Models/Category.php kemudian sesuaikan menjadi seperti berikut ini :

```
Category.php
app > Models > Category.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Category extends Model
9  {
10     use HasFactory;
11     /**
12      * fillable
13      *
14      * @var array
15      */
16     protected $fillable = [
17         'name', 'slug', 'image'
18     ];
19 }
20
```

**Mass Assignment**, mempunyai tujuan agar attribute yang akan kita buat nanti dapat melakukan proses **Create, Read, Update** dan **Delete** ke dalam table database yang berkaitan dengan model tersebut.

Dari perubahan kode di atas, kita menambahkan 1 **properti** baru yaang bernama \$fillable dan di dalamnya terdapat data dalam bentuk array yang merupakan attribute dari table categories yang akan kita buat berikut ini.

### 2023\_04\_18\_135030\_create\_categories\_table.php

Silahkan buka file  
database/migration/2023\_04\_18\_135030\_create\_categories\_table.php

kemudian pada function up kemudian sesuaikan dan secara keseluruhan akan menjadi seperti berikut ini:

```
Category.php • 2023_04_18_135030_create_categories_table.php •
database > migrations > 2023_04_18_135030_create_categories_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('categories', function (Blueprint $table) {
17             $table->id();
18             $table->string('image');
19             $table->string('name');
20             $table->string('slug')->unique();
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('categories');
33     }
34 };
35
```

Pada baris kode di atas kita menambahkan 3 attribute baru untuk table categories, diantaranya adalah :

- image - menggunakan tipe data string
- name - menggunakan tipe data string



- slug - menggunakan tipe data string dan di atur menjadi unique, yang artinya isi dari attribute ini tidak boleh sama.

## Langkah 2 - Membuat Model dan Migration Post

Setelah berhasil membuat Model dan Migration untuk Category, sekarang kita lanjutkan untuk membuat Model dan Migration Post. Sama halnya dalam membuat model dan migration Category, Silahkan jalankan perintah berikut ini di dalam terminal/CMD :

### php artisan make:model Post -m

Jika perintah di atas berhasil dijalankan maka akan melakukan generate 2 file baru, dengan informasi sebagai berikut ini :

```
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c:/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan make:model Post -m

[INFO] Model [C:\xampp\htdocs\laravel-news\C030321043-project3\app\Models\Post.php] created successfully.

[INFO] Migration [C:\xampp\htdocs\laravel-news\C030321043-project3\database\migrations\2023_04_18_142442_create_posts_table.php] created successfully.
```

### 2. 1 Model & Migration Post

Seperti pada langkah sebelumnya, pertama kita akan melakukan konfigurasi di dalam file Model terlebih dahulu untuk menambahkan **Mass Assignment**. Silahkan buka file Post.php yang terletak pada direktori app/Models/Post.php kemudian sesuaikan dan secara keseluruhan akan menjadi seperti berikut ini :

```
app > Models > Post.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Post extends Model
9  {
10     use HasFactory;
11     /**
12      * fillable
13      *
14      * @var array
15      */
16     protected $fillable = [
17         'title', 'slug', 'category_id', 'user_id', 'content', 'image', 'description'
18     ];
19 }
20
```

Setelah berhasil menambahkan **Mass Assignment** di dalam Model Post, sekarang mari kita lanjutkan untuk menambahkan attribute di dalam migration agar nanti saat di migrate akan ter-generate menjadi table baru yang bernama posts beserta attribute-attribute di dalamnya.

Silahkan buka file database/migrations/**2023\_04\_18\_142442\_create\_posts\_table.php** kemudian pada function up kemudian sesuaikan dan secara keseluruhan dan menjadi seperti berikut ini :

```
database > migrations > 2023_04_18_142442_create_posts_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      public function up()
10     {
11         Schema::create('posts', function (Blueprint $table) {
12             $table->id();
13             $table->string('title');
14             $table->string('slug')->unique();
15             $table->unsignedInteger('category_id');
16             $table->unsignedInteger('user_id');
17             $table->text('content');
18             $table->string('image');
19             $table->text('description');
20             $table->timestamps();
21         });
22     }
23
24     public function down()
25     {
26         Schema::dropIfExists('posts');
27     }
28 };
```

Dari perubahan kode di atas, kita menambahkan 8 attribute baru, yaitu :

- title - menggunakan tipe data string.
- slug - menggunakan tipe data string dan bersifat unique, yang artinya isi tidak boleh ada yang sama.
- category\_id - menggunakan unsignedInteger yang mana akan kita gunakan sebagai forenkey yang berelasi dengan attribute id di dalam table categories
- user\_id - menggunakan unsignedInteger yang mana akan kita gunakan sebagai forenkey yang berelasi dengan attribute id di dalam table users.
- content - menggunakan tipe data text.
- image - menggunakan tipe data string.
- description - menggunakan tipe data text.

### Langkah 3 - Membuat Model dan Migration Comment

Pada langkah kali ini kita akan membuat Model dan juga Migration untuk Comment. Silahkan jalankan perintah berikut ini di dalam terminal/CMD :

**php artisan make:model Comment -m**

Jika perintah di atas berhasil dijalankan maka akan melakukan generate 2 file baru, dengan informasi sebagai berikut ini :

```

ASUS@LAPTOP-BMLPSNE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan make:model Comment -m

[INFO] Model [C:\xampp\htdocs\laravel-news\C030321043-project3\app\Models\Comment.php] created successfully.

[INFO] Migration [C:\xampp\htdocs\laravel-news\C030321043-project3\database\Migrations\2023_04_18_145442_create_comments_table.php] created successfully.

ASUS@LAPTOP-BMLPSNE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$

```

### 3. 1 Model & Migration Comment

Pertama kita akan melakukan perubahan di dalam file Model terlebih dahulu untuk menambahkan Mass Assignment. Silahkan buka file Comment.php yang baru saja kita buat melalui perintah diatas, yang terletak pada direktori app/Models/Comment.php. Kemudian sesuaikan dan secara keseluruhan dan menjadi seperti berikut ini :

```

app > Models > Comment.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Comment extends Model
9  {
10     use HasFactory;
11     /**
12      * fillable
13      *
14      * @var array
15      */
16     protected $fillable = [
17         'post_id', 'name', 'email', 'comment'
18     ];
19 }
20

```

Dari perubahan kode di atas, kita menambahkan 1 properti baru dengan tipe array yang bernama \$fillable dan di dalamnya kita set beberapa attribute agar dapat melakukan manipulasi ke dalam database.

Selanjutnya kita akan menambahkan attribute-attribute di dalam file migration comment. Silahkan buka file database/migrations/2023\_04\_18\_145442\_create\_comments\_table.php. Kemudian sesuaikan dan secara keseluruhan dan menjadi seperti berikut ini :

```

database > migrations > 2023_04_18_145442_create_comments_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('comments', function (Blueprint $table) {
17             $table->id();
18             $table->unsignedInteger('post_id');
19             $table->string('name');
20             $table->string('email');
21             $table->text('comment');
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('comments');
34     }
35 };
36

```

Dari penambahan kode di atas, kita menambahkan 4 attribute baru, yaitu :

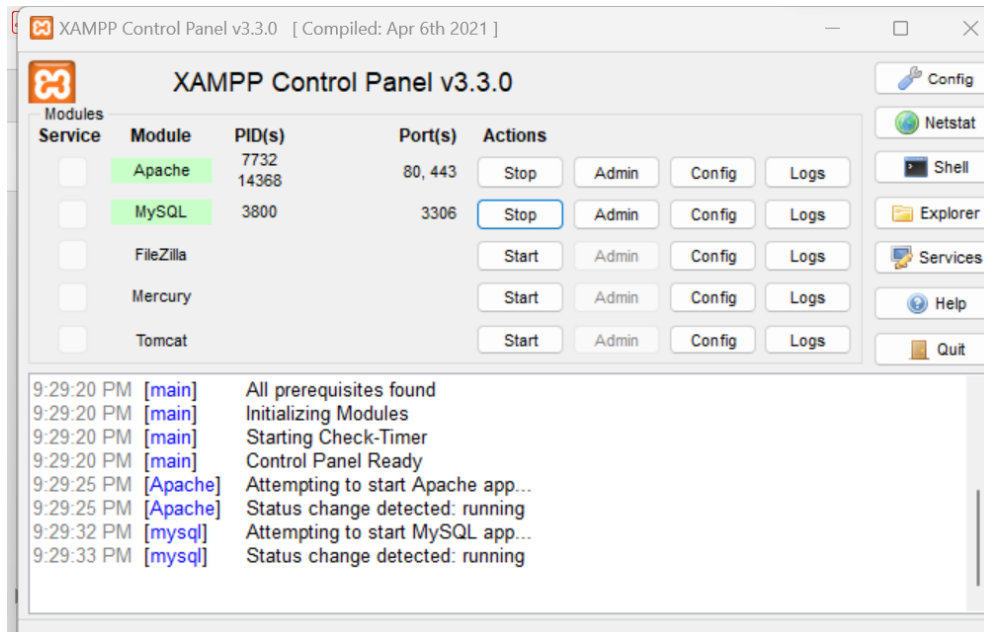
- post\_id - menggunakan unsignedInteger yang akan dijadikan sebagai forenkey dan akan berelasi dengan attribute id yang berada di dalam table posts.
- name - menggunakan tipe data string.
- email - menggunakan tipe data string.
- comment - menggunakan tipe data text.

Selamat, sampai pada langkah kali ini, berhasil membuat model dan migration untuk aplikasi kita, langkah selanjutnya adalah kita akan melakukan proses migration ke dalam database kita, Semangat terus ya :D

Setelah kita berhasil membuat beberapa file models dan juga file beberapa migrations untuk kebutuhan table dari aplikasi yang akan kita bangun, sekarang kita akan menjalankan perintah migrate, yang mana perintah tersebut akan melakukan generate dari file-file migrations kita ke dalam beberapa table beserta attribute yang sudah kita setting sebelumnya.

Jadi proses pembuatan column di dalam database tidak perlu dilakukan secara manual. Inilah salah satu manfaat menggunakan Framework Laravel. Masih didalam direktori laravel-news kemudian silahkan masukkan perintah berikut di dalam terminal/CMD :

Catatan: Pastikan sudah mengaktifkan MySQL di dalam aplikasi Xampp .



## php artisan migrate

Jika perintah di atas berhasil dijalankan dan ditandai informasi seperti gambar dibawah ini, maka sudah dipastikan kita berhasil membuat beberapa tabel beserta column yang sudah kita atur bersama pada tutorial sebelumnya tadi,

```
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c:/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan migrate

[WARN] The database 'laravel' does not exist on the 'mysql' connection.

Would you like to create it? (yes/no) [no]
> yes

[INFO] Preparing database.

Creating migration table ..... 56ms DONE

[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 40ms DONE
2014_10_12_100000_create_password_resets_table ..... 34ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 28ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 37ms DONE
2023_04_18_135030_create_categories_table ..... 10ms DONE
2023_04_18_142442_create_posts_table ..... 8ms DONE
2023_04_18_145442_create_comments_table ..... 10ms DONE

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c:/xampp/htdocs/laravel-news/C030321043-project3
$
```

Atau jika ingin memastikan, silahkan bisa melihat database laravel-news. maka disana akan tampil table sesuai informasi diatas, maka akan seperti berikut ini :

Server: 127.0.0.1 » Database: laravel

Structure SQL Search Query Export Import Operations Privileges Routines Events More

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> categories	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> comments	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> password_resets	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
<input type="checkbox"/> posts	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
8 tables	Sum	7	InnoDB	utf8mb4_general_ci	192.0 KiB	0 E

Jika sudah pernah belajar tentang php dasar, maka pasti sudah pernah mendengar apa itu yang dinamakan fungsi JOIN dengan menggunakan JOIN, kita akan di permudah untuk menampilkan beberapa data yang berasal lebih dari 1 table.

Logika sederhana nya, jika kita akan menampilkan data siswa, yang mencakup nama siswa alamat dan juga kelas, maka kita akan mengambil data dari tabel\_siswa dan juga tabel\_kelas dengan menggunakan masing-masing primary key yang sudah ditetapkan.

Di dalam laravel, istilah ini disebut dengan Eloquent Relationships .

## 1. Apa yang di maksud Eloquent Relationships ?

Seperti pada penjelasan di atas, fitur yang disediakan laravel ini memungkinkan kita lebih mudah untuk menghubungkan antara tabel satu dengan tabel lainnya. Secara teori, laravel mempunyai beberapa fitur Relationships diantaranya adalah :

### 1. One To One

Pada relasi ini, kita bisa melakukan hubungan antar tabel, seperti contoh yang sudah kita kerjakan, kita mempunyai table posts dan juga tabel category, dan didalam table posts terdapat Primary Key yang bernama id, primary key ini akan kita gunakan sebagai kunci untuk menghubungkan antara table post dan table comment dengan menggunakan Foreign Key yang berada di dalam table category dengan nama post\_id

### 2. One To Many

One-to-many relathionship memiliki struktur relasi yang hampir sama dengan one-toone, perbedaanya adalah satu data di dalam table A boleh memiliki banyak data di dalam table B. Contoh sederhananya adalah kita sudah membuat table posts dan juga table comments maka yang di harapkan adalah satu data post bisa memiliki banyak komentar

### 3. Many To Many

Many-to-many relationships merupakan relasi dimana data di dalam table A bisa memiliki banyak data di dalam table B dan juga sebaliknya, data di dalam table B bisa memiliki banyak data di dalam table

A. Disini kita tidak dapat melakukan relasi langsung antara table A ke dalam table B, maka dari itu kita harus memiliki satu table lagi untuk menghubungkan dari kedua table tersebut, biasanya table ini di sebut dengan pivot table.

Pada studi kasus kali ini, kita akan memanfaatkan fungsi dari One To One dan juga fungsi One To Many, sampai jumpa pada langkah selanjutnya

Seperti yang dijelaskan pada dokumentasi laravel, Eloquent adalah sebuah fitur untuk mengelola data yang ada pada database dengan sangat mudah. Eloquent ORM (Object Relation Mapping) adalah sebuah fitur dari Laravel yang di dalamnya terdapat fungsi-fungsi active record (query SQL) untuk mengelola data di database. Dengan menggunakan Eloquent ORM, database bisa kita bungkus ke dalam objek, sehingga operasi CRUD (Create, Read, Update, Delete) pada tabel database dapat dilakukan tanpa melibatkan perintah / query SQL sama sekali, bahkan sampai relasi antar tabelnya juga.

Seperti contoh di dalam php native, jika hendak menampilkan data dari table maka kita akan menggunakan query

**Select \* from nama\_tabel**

akan tetapi di dalam laravel bisa mudah dengan menggunakan eloquent accesor. Seperti contoh, kita sudah mempunyai model dengan nama file User.php kurang lebih seperti berikut jika kita menggunakan accesor

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Casts\Attribute;
use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Get the user's first name.
     *
     * @return \Illuminate\Database\Eloquent\Casts\Attribute
     */
    protected function name(): Attribute
    {
        return Attribute::make(
            get: fn ($value) => strtoupper($value);,
        );
    }
}
```

Pada baris Illuminate\Database\Eloquent\Casts\Attribute kita melakukan import untuk mengaktifkan fitur accessor kemudian sebagai contoh kita akan menampilkan nama yang terdapat pada table user, dengan menggunakan variable get dan menambahkan fungsi strtoupper fungsi ini akan merubah tulisan yang menggunakan huruf kecil menjadi kapital. Misal di dalam table user terdapat satu field dengan nama user Teknik Informatika maka akan ditampilkan menggunakan huruf kapital menjadi TEKNIK INFORMATIKA

Jika pada pembahasan sebelumnya kita semua telah mengetahui apa itu relationship dan juga macam-macam nya, dan juga kita sudah mengerti apa yang di maksud dengan accessor. Kali ini kita akan menerapkan keduanya ke dalam file models yang sudah kita sbuat sebelumnya,

## Langkah 1. Setting relasi & Accessor Model Category

Silahkan buka file Category.php yang terletak di dalam direktori app/Models/Category.php kemudian rubah kode nya menjadi seperti berikut ini :

```
app > Models > Category.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Casts\Attribute;
7  use Illuminate\Database\Eloquent\Model;
8
9  class Category extends Model
10 {
11     use HasFactory;
12
13     /**
14      * fillable
15      *
16      * @var array
17      */
18     protected $fillable = [
19         'name', 'slug', 'image'
20     ];
21
22     /**
23      * posts
24      *
25      * @return void
26
27     */
28     public function posts()
29     {
30         return $this->hasMany(Post::class);
31     }
32
33     /**
34      * image
35      *
36      * @return Attribute
37      */
38     protected function image():Attribute
39     {
40         return Attribute::make(
41             get:fn($value)=>asset('/storage/categories/'.$value),
42         );
43     }
44 }
```

## Langkah 2. Setting relasi & Accessor Model Comment

Silahkan buka file Comment.php yang terletak di dalam direktori app/Models/Comment.php kemudian rubah kode nya menjadi seperti berikut ini:



```

app > Models > Comment.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Casts\Attribute;
7  use Illuminate\Database\Eloquent\Model;
8
9  class Comment extends Model
10 {
11     use HasFactory;
12     /**
13      * fillable
14      *
15      * @var array
16      */
17     protected $fillable = [
18         'post_id', 'name', 'email', 'comment'
19     ];
20     /**
21      * createdAt
22      *
23      * @return Attribute
24      */
25     protected function createdAt():Attribute
26     {
27         return Attribute::make(
28             get:fn($value)=>\Carbon\Carbon::locale('id')->parse($value)
29             ->translatedFormat('l, d F Y'),
30         );
31     }
32
33     /**
34      * updatedAt
35      *
36      * @return Attribute
37      */
38     protected function updatedAt():Attribute
39     {
40         return Attribute::make(
41             get:fn($value)=>\Carbon\Carbon::locale('id')->parse($value)
42             ->translatedFormat('l d, F Y'),
43         );
44     }
45 }
46

```

### Langkah 3. Setting relasi & Accessor Model Post

Silahkan buka file Post.php yang terletak di dalam direktori app/Models/Post.php kemudian rubah kode nya menjadi seperti berikut ini :

app > Models > Post.php

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\casts\Attribute;
7  use Illuminate\Database\Eloquent\Model;
8
9  class Post extends Model
10 {
11     use HasFactory;
12     /**
13      * fillable
14      *
15      * @var array
16      */
17     protected $fillable = [
18         'title', 'slug', 'category_id', 'user_id', 'content', 'image', 'description'
19     ];
20
21     /**
22      * category
23      *
24      * @return void
25      */
26     public function category()
```

```
27     {
28         return $this->belongsTo(Category::class);
29     }
30
31     /**
32      * user
33      *
34      * @return void
35      */
36     public function user()
37     {
38         return $this->belongsTo(User::class);
39     }
40
41     /**
42      * comments
43      *
44      * @return void
45      */
46     public function comments()
47     {
48         return $this->hasMany(Comment::class);
49     }
50
```

```
51     /**
52      * image
53      *
54      * @return Attribute
55      */
56     protected function image():Attribute
57     {
58         return Attribute::make(
59             get:fn($value)=>asset('/storage/posts'.value),
60         );
61     }
62
63     /**
64      * createdAt
65      *
66      * @return Attribute
67      */
68     protected function createdAt():Attribute
69     {
70         return Attribute::make(
```

```

71         get:fn($value)=>\Carbon\Carbon::locale('id')->parse($value)
72         ->translatedFormat('l, d F Y'),
73     );
74 }
75
76 /**
77  * updatedAt
78  *
79  * @return Attribute
80  */
81 protected function updatedAt():Attribute
82 {
83     return Attribute::make(
84         get:fn($value)=>\Carbon\Carbon::locale('id')->parse($value)
85         ->translatedFormat('l, d F Y'),
86     );
87 }
88 }

```

Laravel memiliki fitur yang bernama **Database Seeding**, fitur ini bisa kita manfaatkan untuk membuat dummy data ke dalam database. Dan disini kita akan manfaatkan fitur ini untuk membuat dummy data user, yang mana akan digunakan untuk melakukan proses authentication di dalam halaman admin.

### Langkah 1. Make Seeder User

Untuk membuat sebuah seeder kita bisa menggunakan bantuan **Artisan**, yaitu `make:seeder` dan diikuti dengan nama seeder yang akan dibuat. Sekarang, silahkan jalankan perintah berikut ini di dalam terminal/CMD :

#### php artisan make:seeder UserSeeder

Jika perintah diatas berhasil dijalankan, maka kita akan mendapatkan 1 file baru di dalam folder `database/seeder/UserSeeder.php`. Dan ditandai dengan pesan seperti gambar di bawah ini :

```

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan make:seeder UserSeeder

INFO Seeder [C:\xampp\htdocs\laravel-news\C030321043-project3\database\seeder\UserSeeder.php] created successfully.

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$

```

Silahkan buka file tersebut dan sesuaikan menjadi seperti berikut ini :

```

database > seeders > UserSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
9
10 class UserSeeder extends Seeder
11 {
12     /**
13      * Run the database seeds.
14      *
15      * @return void
16      */
17     public function run()
18     {
19         DB::table('users')->insert([
20             'name' => 'Milga Tibalimeten',
21             'email' => 'milgatibalimeten@gmail.com',
22             'password' => Hash::make('milgatibalimeten01'),
23         ]);
24     }
25 }
26
27

```

## Langkah 2. Dump UserSeeder

Jika kita sudah melakukan setting terhadap UserSeeder di atas, langkah selanjutnya adalah melakukan DUMP data ke dalam database kita menuju table users dengan value yang sudah kita atur pada baris kode :

```

        DB::table('users')->insert([
            'name' => 'Milga Tibalimeten',
            'email' => 'milgatibalimeten@gmail.com',
            'password' => Hash::make('milgatibalimeten01'),
        ]);
    }
}

```

Jika kita melakukan perintah composer dump-autoload yang bertujuan untuk menggenerate file autoload.php

Silahkan teman teman masukkan perintah di bawah ini ke dalam command prompt

**composer dump-autoload**

```

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$ composer dump-autoload
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE

Generated optimized autoload files containing 5261 classes
ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$

```

dan juga perintah

**php artisan db:seed --class=UserSeeder**

```

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$ php artisan db:seed --class=UserSeeder

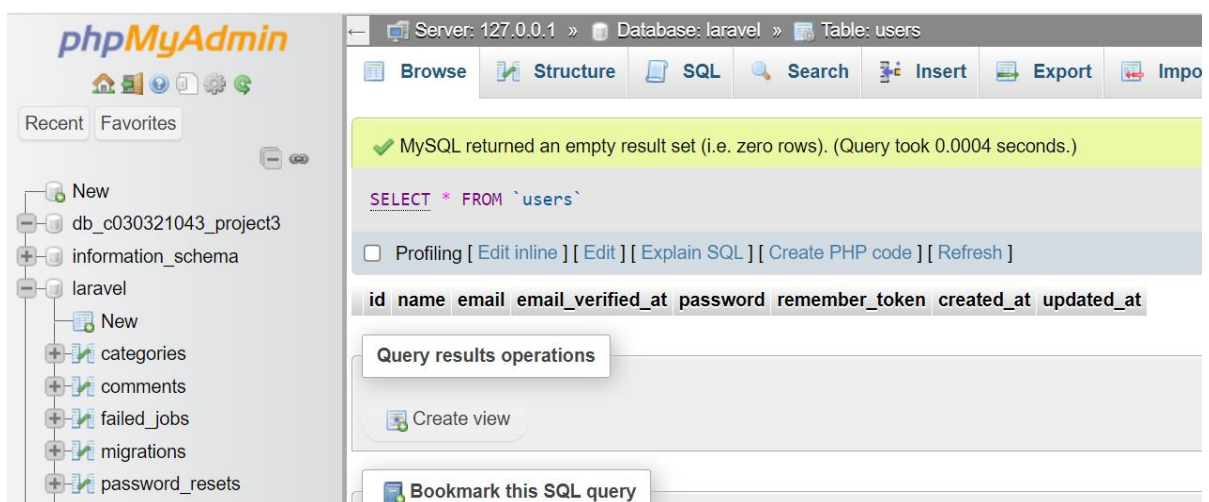
 INFO  Seeding database.

ASUS@LAPTOP-BMLP5NE4 MINGW64 /c/xampp/htdocs/laravel-news/C030321043-project3
$

```

Jika kedua perintah di atas berhasil di lakukan dengan ditandai gambar seperti dibawah ini maka sudah berhasil membuat Data Seeder User.

Maka kita akan mendapati data di dalam table users dengan isi yang sudah kita tetapkan sebelumnya. Maka akan terlihat seperti pada gambar di bawah ini :







***\*NOTE: Langkah selanjutnya terjadi error dan masih berusaha menyelesaikan 😊***