

LAPORAN PRAKTEK

Praktek Pemrograman Web Semester 4



Nama : Ervina Yulia Sari SN
Nim : C030321038
Kelas : Teknik Informatika-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

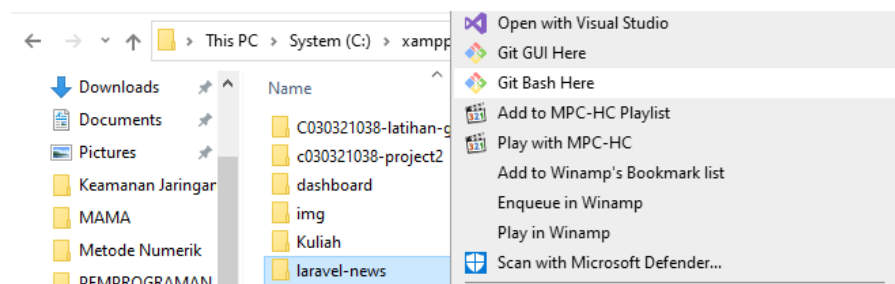
1.2 Tujuan

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:

```
USER@DESKTOP-JOUNESG MINGW64 /c/xampp/htdocs/laravel-news
$ composer create-project laravel/laravel c030321038-project3
Creating a "laravel/laravel" project at "./c030321038-project3"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v10.1.0)
- Downloading laravel/laravel (v10.1.0)
- Installing laravel/laravel (v10.1.0): Extracting archive
Created project in C:\xampp\htdocs\laravel-news\c030321038-project3
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
```

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

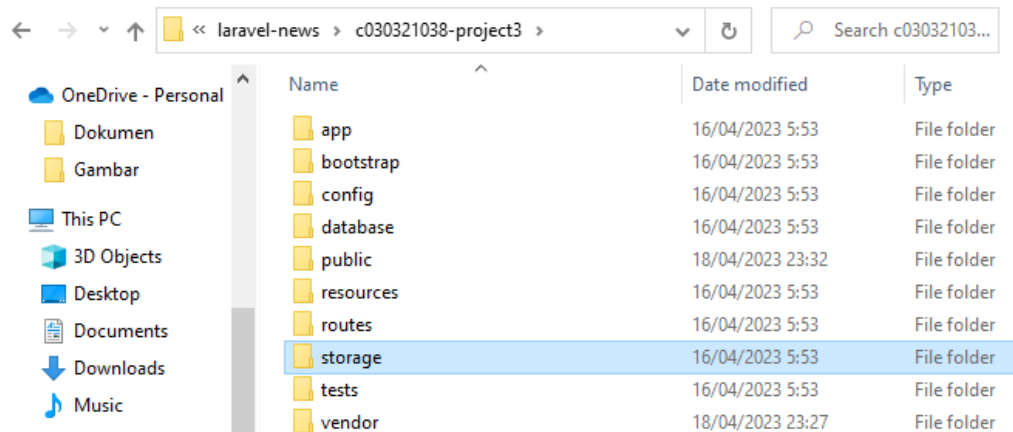
storage:link, maka kita akan membuat sebuah link atau shortcut dari folder storage kedalam folder public.

```
USER@DESKTOP-JOUNESG MINGW64 /c/xampp/htdocs/laravel-news
$ cd c030321038-project3
```

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

```
USER@DESKTOP-JOUNESG MINGW64 /c/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan storage:link

INFO The [C:\xampp\htdocs\laravel-news\c030321038-project3\public\storage] link has been connected to [C:\xampp\htdocs\laravel-news\c030321038-project3\storage\app/public].
```



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

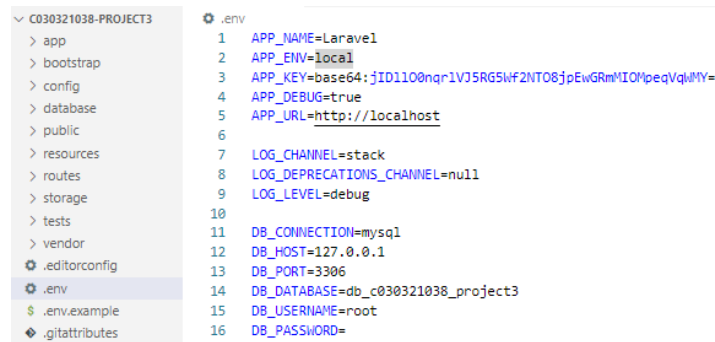
Database

Koneksi Laravel Ke MySql

Buatlah database, kemudian setting config databasenya

Databases





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:

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

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

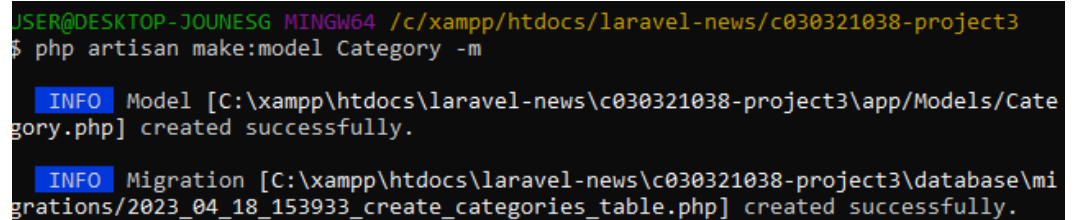
[3]. Model & Migration Comment

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 :



```
USER@DESKTOP-JOUNESG MINGW64 /c:/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan make:model Category -m

[INFO] Model [C:\xampp\htdocs\laravel-news\c030321038-project3\app\Models\Category.php] created successfully.

[INFO] Migration [C:\xampp\htdocs\laravel-news\c030321038-project3\database\Migrations\2023_04_18_153933_create_categories_table.php] created successfully.
```

Buka file `app/Models/Category.php` kemudian ketik coding berikut

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    use HasFactory;

    protected $fillable = [
        'name', 'slug', 'image'
    ];
}
```

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 yang bernama `$fillable` dan di dalamnya terdapat data dalam bentuk array yang merupakan attribute dari table categories yang akan kita buat berikut ini.

`2023_04_10_043113_create_categories_table.php`

Buka file

`database/migrations/2023_04_10_043113_create_categories_table.php`

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

```
<?php

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

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

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

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

- a. image - menggunakan tipe data string
- b. name - menggunakan tipe data string
- c. 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 :

```
USER@DESKTOP-JOUNESG MINGW64 /c:/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan make:model Post -m

[INFO] Model [C:\xampp\htdocs\laravel-news\c030321038-project3\app\Models\Post.php] created successfully.

[INFO] Migration [C:\xampp\htdocs\laravel-news\c030321038-project3\database\migrations\2023_04_18_154712_create_posts_table.php] created successfully.
```

Silahkan buka file Post.php yang terletak pada direktori app/Models/Post.php kemudian sesuaikan dan secara keseluruhan akan menjadi seperti berikut ini :

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;

    protected $fillable = [
        'title', 'slug', 'category_id', 'user_id', 'content', 'image', 'description'
    ];
}
```

Silahkan buka file

database/migrations/2023_04_10_043446_create_posts_table.php kemudian pada function up kemudian sesuaikan dan secara keseluruhan dan menjadi seperti berikut ini :

```

<?php

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

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

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

```

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

- a. title - menggunakan tipe data string.
- b. slug - menggunakan tipe data string dan bersifat unique, yang artinya isi tidak boleh ada yang sama.
- c. category_id - menggunakan unsignedInteger yang mana akan kita gunakan sebagai forenkey yang berelasi dengan attribute id di dalam table categories.
- d. user_id - menggunakan unsigbedInteger yang mana akan kita gunakan sebagai forenkey yang berelasi dengan attribute id di dalam table users.
- e. content - menggunakan tipe data text.
- f. image - menggunakan tipe data string.
- g. 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 :

```
USER@DESKTOP-JOUNESG MINGW64 /c:/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan make:model Comment -m

INFO Model [C:\xampp\htdocs\laravel-news\c030321038-project3\app\Models\Comment.php] created successfully.

INFO Migration [C:\xampp\htdocs\laravel-news\c030321038-project3\database\migrations\2023_04_21_142027_create_comments_table.php] created successfully.
```

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 :

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Comment extends Model
{
    use HasFactory;

    protected $fillable = [
        'post_id', 'name', 'email', 'comment'
    ];
}
```

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_10_045113_create_comments_table.php
Kemudian sesuaikan dan secara keseluruhan dan menjadi seperti berikut ini :

```

<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('comments', function (Blueprint $table) {
            $table->id();
            $table->unsignedInteger('post_id');
            $table->string('name');
            $table->string('email');
            $table->text('comment');
            $table->timestamps();
        });
    }

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

```

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

- a. post_id - menggunakan unsignedInteger yang akan dijadikan sebagai forenkey dan akan berelasi dengan attribute id yang berada di dalam table posts.
- b. name - menggunakan tipe data string.
- c. email - menggunakan tipe data string.
- d. 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

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 direktor laravel-news kemudian silahkan masukkan perintah berikut di dalam terminal/CMD :

Note: 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,

```
USER@DESKTOP-JOUNESG MINGW64 /c/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan migrate

INFO Nothing to migrate.
```

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

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> categories	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 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_reset_tokens	★ 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	32.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-

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 accessor. Seperti contoh, kita sudah mempunyai model dengan nama file User.php kurang lebih seperti berikut jika kita menggunakan acceso

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 :

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class category extends Model
{
    use HasFactory;
    protected $fillable = [
        'name', 'slug', 'image'
    ];

    /**
     * posts
     *
     * @return void
     */
    public function posts()
    {
        return $this->hasMany(Post::class);
    }

    /**
     * image
     *
     * @return Attribute
     */
    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn ($value) => asset('/storage/categories/' . $value),
        );
    }
}
```

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 :

```

<?php

namespace App\Models;

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

class Comment extends Model
{
    use HasFactory;
    protected $fillable = ['pos_id', 'name', 'email', 'comment'];

    protected function createdAt(): Attribute
    {
        return Attribute::make(
            get: fn ($value) => \Carbon\Carbon::locale('id')->parse($value)->translatedFormat('L d F Y'),
        );
    }

    protected function updatedAt(): Attribute
    {
        return Attribute::make(
            get: fn ($value) => \Carbon\Carbon::locale('id')->parse($value)->translatedFormat('L d F Y'),
        );
    }
}

```

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 :

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;
    protected $fillable = [
        'title', 'slug', 'category_id', 'user_id', 'content', 'image',
        'description'
    ];

    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function comments()
    {
        return $this->hasMany(Comment::class);
    }

    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn ($value) => asset('/storage/posts/' . $value),
        );
    }
}

```

```

protected function createdAt(): Attribute
{
    return Attribute::make(
        get: fn ($value) => \Carbon\Carbon::locale('id')->parse($value)->translatedFormat('l, d F Y'),
    );
}

protected function updatedAt(): Attribute
{
    return Attribute::make(
        get: fn ($value) => \Carbon\Carbon::locale('id')->parse($value)->translatedFormat('l, d F Y'),
    );
}

```

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.

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

```

USER@DESKTOP-JOUNES6 MINGW64 /c:/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan make:seeder UserSeeder

INFO Seeder [C:\xampp\htdocs\laravel-news\c030321038-project3\database\seeders\UserSeeder.php] created successfully.

```

Silahkan buka file tersebut dan sesuaikan menjadi seperti berikut ini :

```

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;

class UserSeeder extends Seeder
{
    public function run()
    {
        DB::table('users')->insert([
            'name' => 'Ervina Yulia Sari SN',
            'email' => 'ervinayulia7@email.com',
            'password' => Hash::make('passwordyangmaudibuat'),
        ]);
    }
}

```

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 :

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

```

USER@DESKTOP-JOUNESG MINGW64 /c:/xampp/htdocs/laravel-news/c030321038-project3
$ composer dump-autoload
Generating optimized autoload files
Class App\Models\Category located in C:/xampp/htdocs/laravel-news/c030321038-project3/app/Models/Category.php does not comply with psr-4 autoloading standard. Skipping.

> 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 5779 classes

```

php artisan db:seed --class=UserSeeder

```
USER@DESKTOP-JOUNESG MINGW64 /c:/xampp/htdocs/laravel-news/c030321038-project3
$ php artisan db:seed --class=UserSeeder

INFO Seeding database.
```

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

	id	name	email	email_verified_at	password
		Ervinia			
<input type="checkbox"/>	1	Yulia	ervinayulia7@email.com	NULL	\$2y\$10\$T7hj2GqocVulf2bB715petZAtEMacsdTJFw8xwVZ5...
		Sari SN			

↑ ☐ Check all With selected: ☐ Edit ☐ Copy ☐ Delete ☐ Export

☐ Show all | Number of rows: 25 Filter rows: Search this table