Laporan Modul 7: Eloquent Relationship & Pagination

Mata Kuliah: Workshop Web Lanjut
Nama: Ahmad Aulia Fahlevi
NIM: 2024573010077 **Kelas:** TI-2C

Abstrak

Praktikum ini berfokus pada pemanfaatan Eloquent ORM di Laravel untuk mendefinisikan dan berinteraksi dengan berbagai jenis relasi antar model dalam database MySQL. Relasi yang diimplementasikan meliputi One-to-One (User ↔ Profile), One-to-Many (User → Posts), dan Many-to-Many (Posts ↔ Tags). Prosesnya melibatkan pembuatan Migrations untuk skema database yang kompleks, pendefinisian Model dengan method relationship yang sesuai (hasOne, belongsTo, hasMany, belongsToMany), dan Seeder untuk mengisi data dummy. Hasil praktikum adalah aplikasi web fungsional yang mampu mengambil data terhubung secara efisien, menggunakan fitur Eager Loading (with()) untuk mengatasi masalah N+1 Query, yang merupakan demonstrasi kunci dalam pengembangan aplikasi berbasis data yang kompleks dan berkinerja tinggi.

1. Dasar Teori

Dasar teori praktikum ini adalah Eloquent Relationship yang memungkinkan interaksi data antar tabel database menggunakan sintaks PHP berorientasi objek.

- **Jenis-Jenis Eloquent Relationship**
    - **One-to-One (hasOne/belongsTo):** Satu record dari satu model terhubung ke tepat satu record dari model lain (misalnya, User dan Profile). Relasi balik (Inverse) didefinisikan dengan belongsTo.
    - **One-to-Many (hasMany/belongsTo):** Satu record dari model induk terhubung ke banyak record di model anak (misalnya, satu User dapat memiliki banyak Post). Relasi balik selalu belongsTo.
    - **Many-to-Many (belongsToMany):** Banyak record dari satu model dapat terhubung ke banyak record dari model lain (misalnya, Post dan Tag). Hubungan ini memerlukan tabel perantara (Pivot Table), yang dalam kasus ini dinamai post_tag.
- **Eager Loading (with())**
    - **Tujuan:** Untuk mengatasi masalah N+1 Query. Tanpa Eager Loading, jika mengambil 10 Post dan kemudian mengakses User dari setiap Post secara terpisah, akan terjadi 1 query untuk Post ditambah 10 query untuk User (total 11 query). -**Implementasi:** Dengan menggunakan Model::with('relationship_name')->get(), Laravel hanya mengeksekusi 2 query: satu untuk data utama, dan satu untuk semua data relasi terkait. Ini sangat penting untuk performa aplikasi.
- **Komponen Pendukung**

- **Migrations:** Digunakan untuk mendefinisikan skema database secara terprogram, termasuk foreign key dan unique constraints yang diperlukan untuk relasi (misalnya user_id pada tabel posts).
- **Model dan Controller:** Model mendefinisikan hubungan, sementara Controller menggunakan method hubungan tersebut
  untuk mengambil data dan meneruskannya ke View.

---

2. Langkah-Langkah Praktikum

Tuliskan langkah-langkah yang sudah dilakukan, sertakan potongan kode dan screenshot hasil.

2.1 Praktikum 1 – Eloquent ORM Relationships: One-to-One, One-to-Many, Many-to-Many

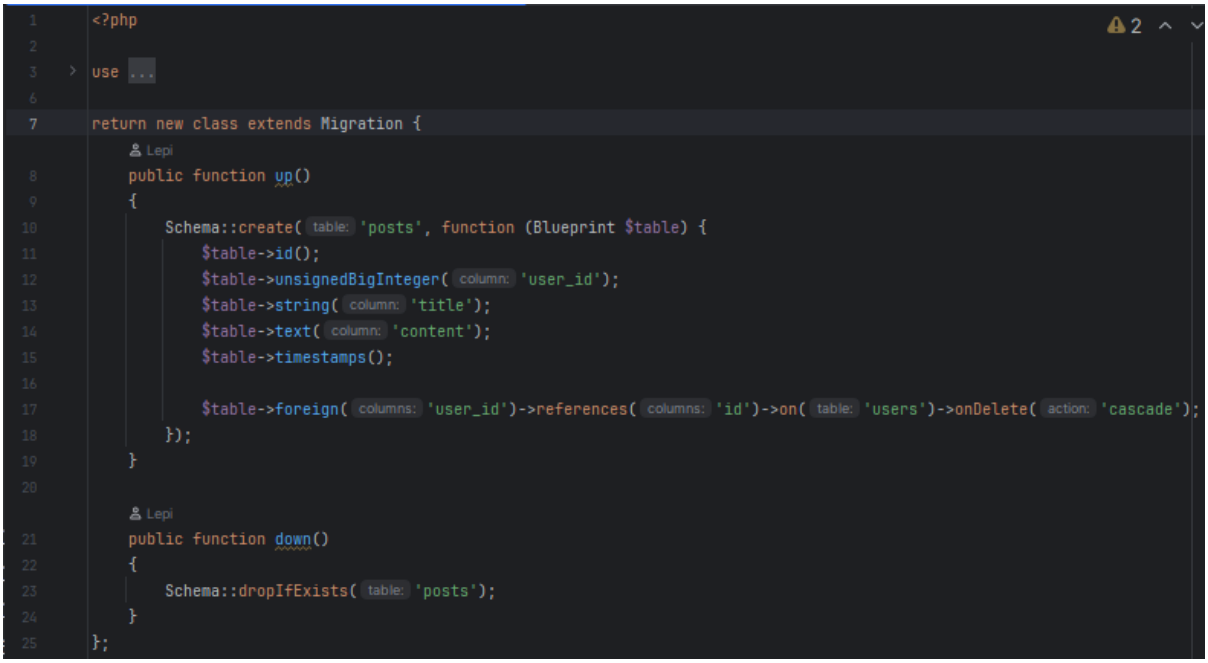
- Membuat Migrasi untuk Skema Database

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:migration create_profile_table


Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:migration create_posts_table


Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:migration create_tags_table


Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:migration create_post_table
```

  - Mengedit Migrasi profile table yang sudah dibuat.

```php
<?php

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

return new class extends Migration {
    Lepi
    public function up()
    {
        Schema::create( table: 'profiles', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger( column: 'user_id')->unique();
            $table->text( column: 'bio')->nullable();
            $table->string( column: 'website')->nullable();
            $table->timestamps();

            $table->foreign( columns: 'user_id')->references( columns: 'id')->on( table: 'users')->onDelete( action: 'cascade');
        });
    }

    Lepi
    public function down()
    {
        Schema::dropIfExists( table: 'profiles');
    }
};
```

- Mengedit Migrasi Posts table yang sudah dibuat.

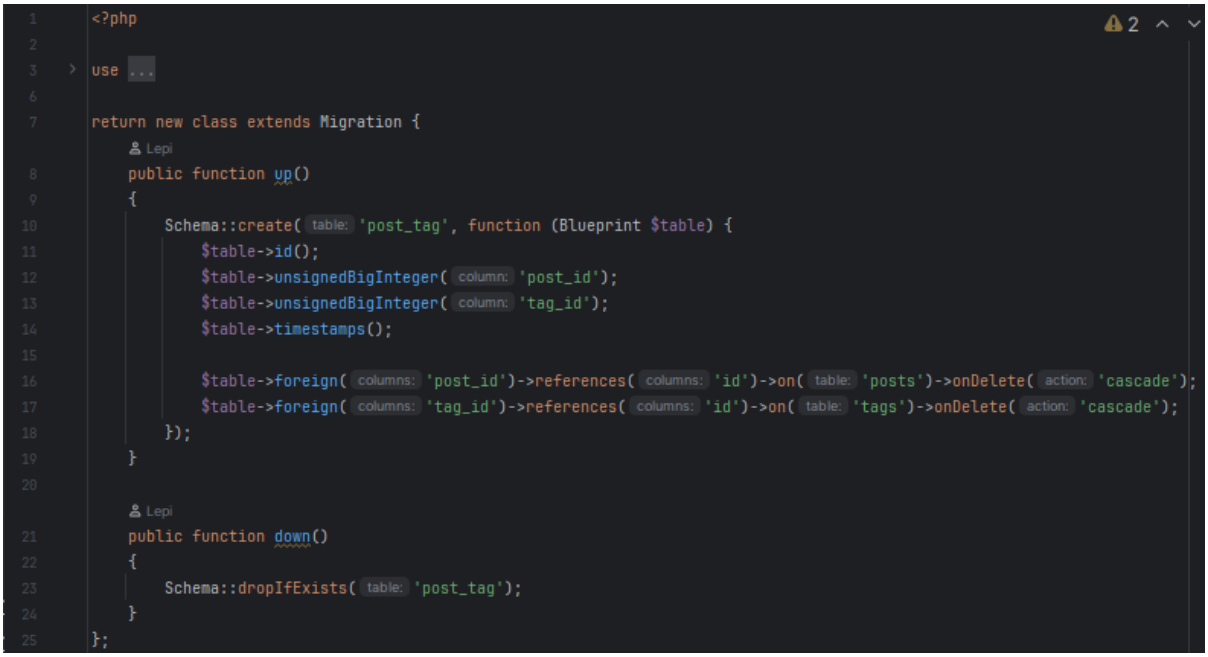


```php
<?php

use ...

return new class extends Migration {
    Lepi
    public function up()
    {
        Schema::create( table: 'posts', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger( column: 'user_id');
            $table->string( column: 'title');
            $table->text( column: 'content');
            $table->timestamps();

            $table->foreign( columns: 'user_id')->references( columns: 'id')->on( table: 'users')->onDelete( action: 'cascade');
        });
    }

    Lepi
    public function down()
    {
        Schema::dropIfExists( table: 'posts');
    }
};
```

- Mengedit Migrasi Tags table yang sudah dibuat.

```php
<?php

use ...

return new class extends Migration {
    Lepi
    public function up()
    {
        Schema::create( table: 'tags', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'name')->unique();
            $table->timestamps();
        });
    }

    Lepi
    public function down()
    {
        Schema::dropIfExists( table: 'tags');
    }
};
```

- Mengedit Migrasi Tag tabel yang sudah dibuat.



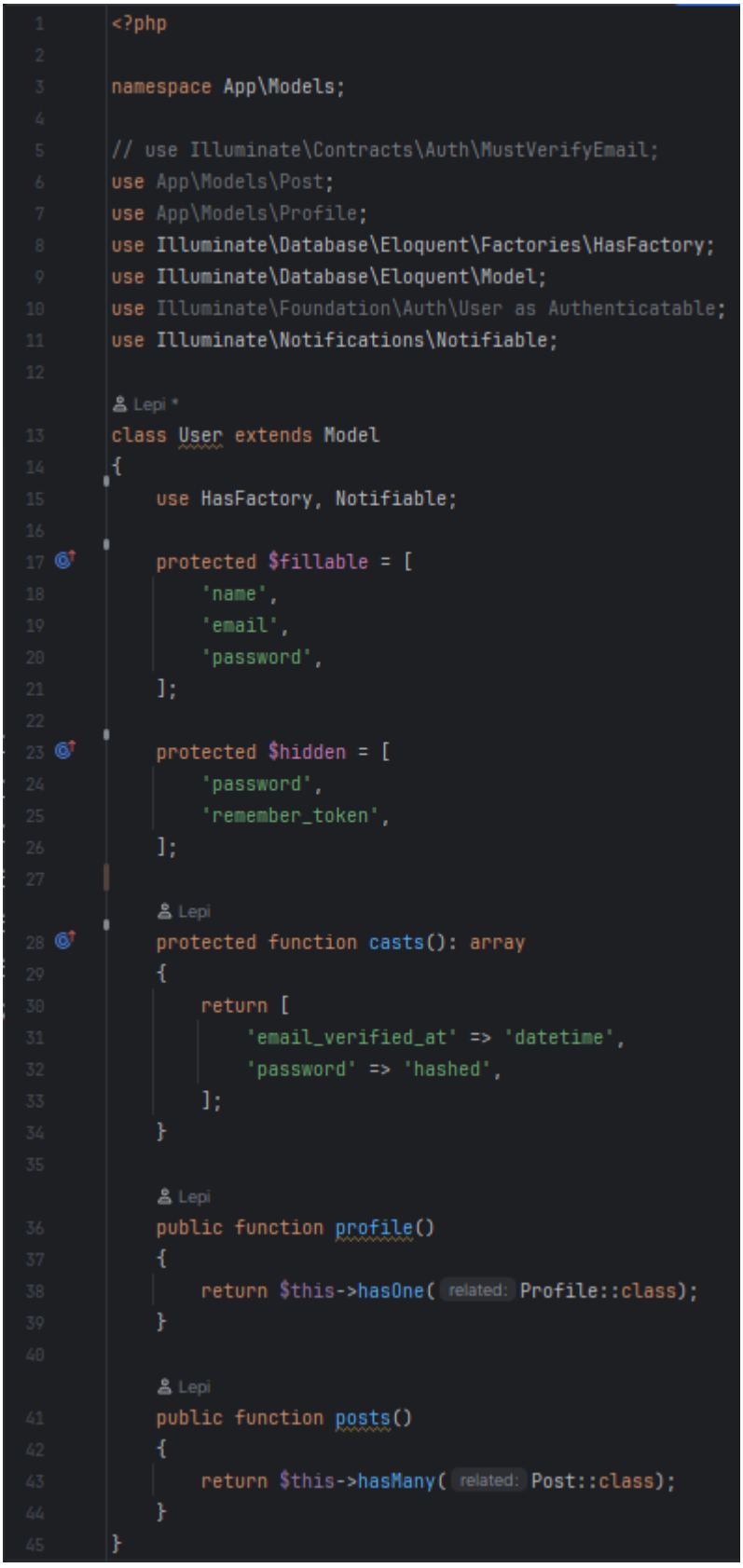
```php
<?php

use ...

return new class extends Migration {
    Lepi
    public function up()
    {
        Schema::create( table: 'post_tag', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger( column: 'post_id');
            $table->unsignedBigInteger( column: 'tag_id');
            $table->timestamps();

            $table->foreign( columns: 'post_id')->references( columns: 'id')->on( table: 'posts')->onDelete( action: 'cascade');
            $table->foreign( columns: 'tag_id')->references( columns: 'id')->on( table: 'tags')->onDelete( action: 'cascade');
        });
    }

    Lepi
    public function down()
    {
        Schema::dropIfExists( table: 'post_tag');
    }
};
```

- Setelah mengedit file migrasi. lalu lakukan migrasi dengan perintah "php artisan migrate"

- Mendefinisikan Model Eloquent

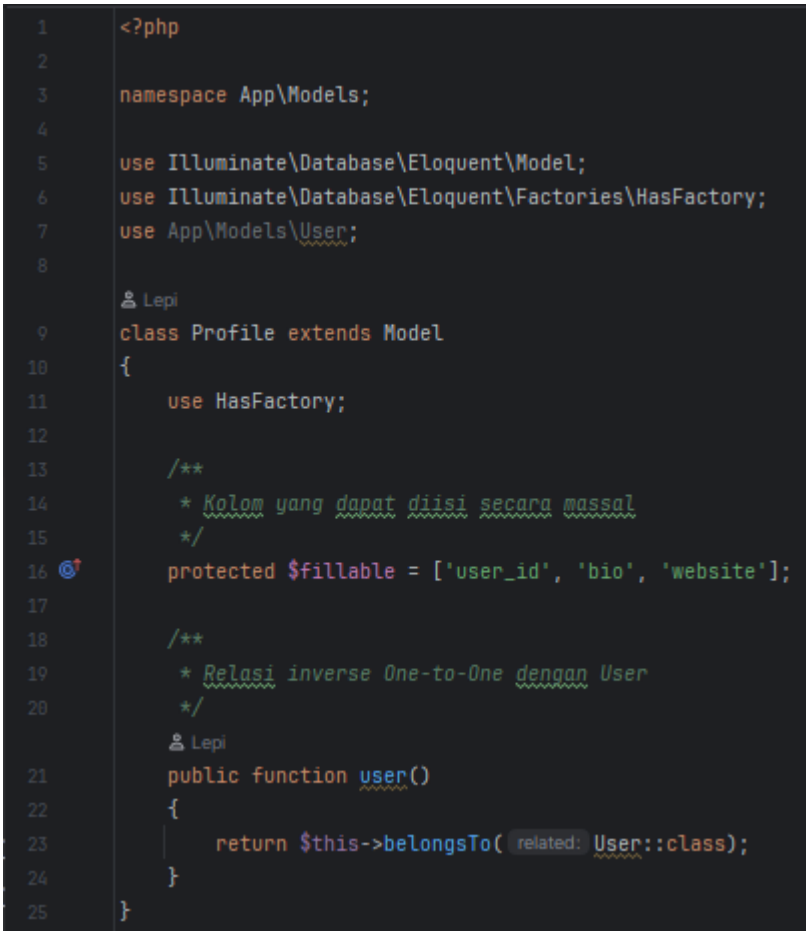
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:model Profile


Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:model Post


Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:model Tag
```

- Mengedit file Model User.php



```php
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use App\Models\Post;
use App\Models\Profile;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

Lepi *
class User extends Model
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    Lepi
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }

    Lepi
    public function profile()
    {
        return $this->hasOne( related: Profile::class);
    }

    Lepi
    public function posts()
    {
        return $this->hasMany( related: Post::class);
    }
}
```

- Mengedit file Model Profile.php

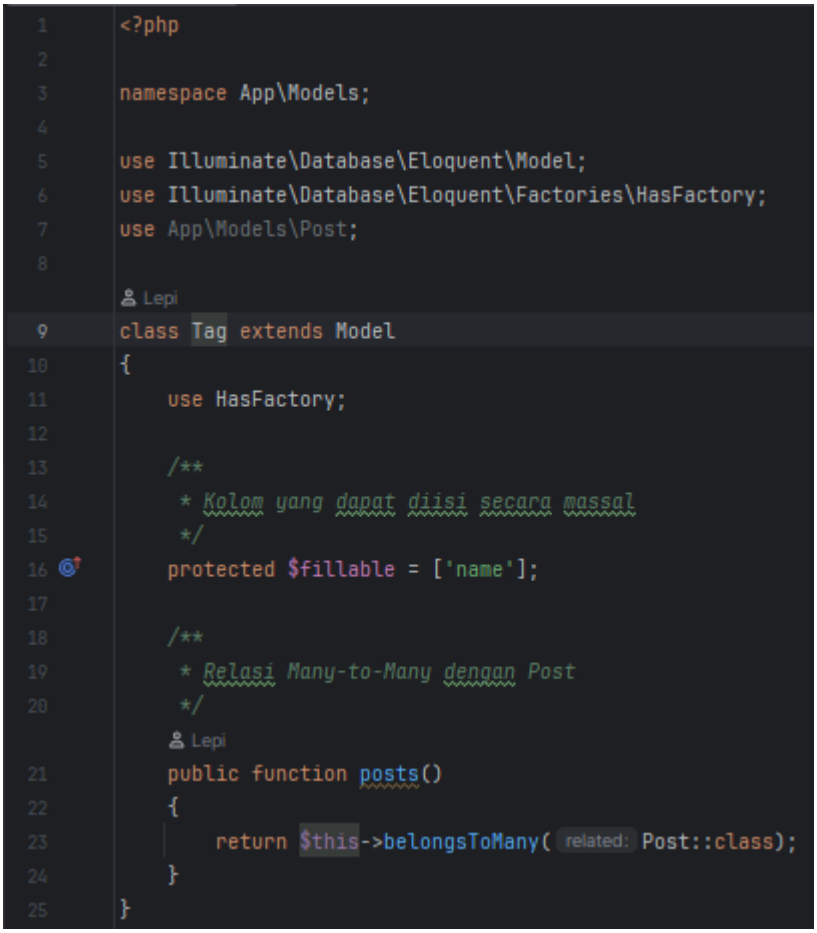


```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use App\Models\User;

Lepi
class Profile extends Model
{
    use HasFactory;

    /**
     * Kolom yang dapat diisi secara massal
     */
    protected $fillable = ['user_id', 'bio', 'website'];

    /**
     * Relasi inverse One-to-One dengan User
     */
    Lepi
    public function user()
    {
        return $this->belongsTo( related: User::class);
    }
}
```

- Mengedit file Model Post

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use App\Models\User;
use App\Models\Tag;

 Lepi
class Post extends Model
{
    use HasFactory;

    /**
     * Kolom yang dapat diisi secara massal
     */
    protected $fillable = ['user_id', 'title', 'content'];

    /**
     * Relasi inverse One-to-Many dengan User
     */
     Lepi
    public function user()
    {
        return $this->belongsTo( related: User::class);
    }

    /**
     * Relasi Many-to-Many dengan Tag
     */
     Lepi
    public function tags()
    {
        return $this->belongsToMany( related: Tag::class);
    }
}
```

- Mengedit file Model Tag



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use App\Models\Post;

Lepi
class Tag extends Model
{
    use HasFactory;

    /**
     * Kolom yang dapat diisi secara massal
     */
    protected $fillable = ['name'];

    /**
     * Relasi Many-to-Many dengan Post
     */
    Lepi
    public function posts()
    {
        return $this->belongsToMany( related: Post::class);
    }
}
```

- Membuat Seeder lalu mengedit setelah itu menjalankannya dengan perintah "php artisan db:seed"

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:seeder DatabaseSeeder
```

```php
<?php

namespace Database\Seeders;

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

Lepi
class DatabaseSeeder extends Seeder
{
    use WithoutModelEvents;

    /**
     * Seed the application's database.
     */
    Lepi
    public function run(): void
    {
        // Membuat 10 user menggunakan factory
        User::factory( count: 10)->create();

        // Membuat profile untuk setiap user
        foreach (User::all() as $user) {
            $user->profile()->create([
                'bio' => 'Ini adalah bio untuk user ' . $user->id,
                'website' => 'https://ilmudata.id/user/' . $user->id,
            ]);
        }

        // Membuat post untuk setiap user
        foreach (User::all() as $user) {
            $user->posts()->create([
                'title' => 'Judul Post untuk user ' . $user->id,
                'content' => 'Ini adalah konten dari post untuk user ' . $user->id,
            ]);
        }

        // Membuat tag dan mengasosiasikannya dengan posts
        foreach (Post::all() as $post) {
            $tag = Tag::create(['name' => 'Tag untuk post ' . $post->id]);
            $post->tags()->attach($tag->id);
        }
    }
}
```

- Membuat Controller
  - Membuat Controller UserController

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:controller UserController
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;

Lepi
class UserController extends Controller
{
    Lepi
    public function index()
    {
        $users = User::with( relations: 'profile', 'posts')->get();
        return view( view: 'users.index', compact( var_name: 'users'));
    }

    Lepi
    public function show(User $user)
    {
        return view( view: 'users.show', compact( var_name: 'user'));
    }
}
```

- Membuat Controller PostController

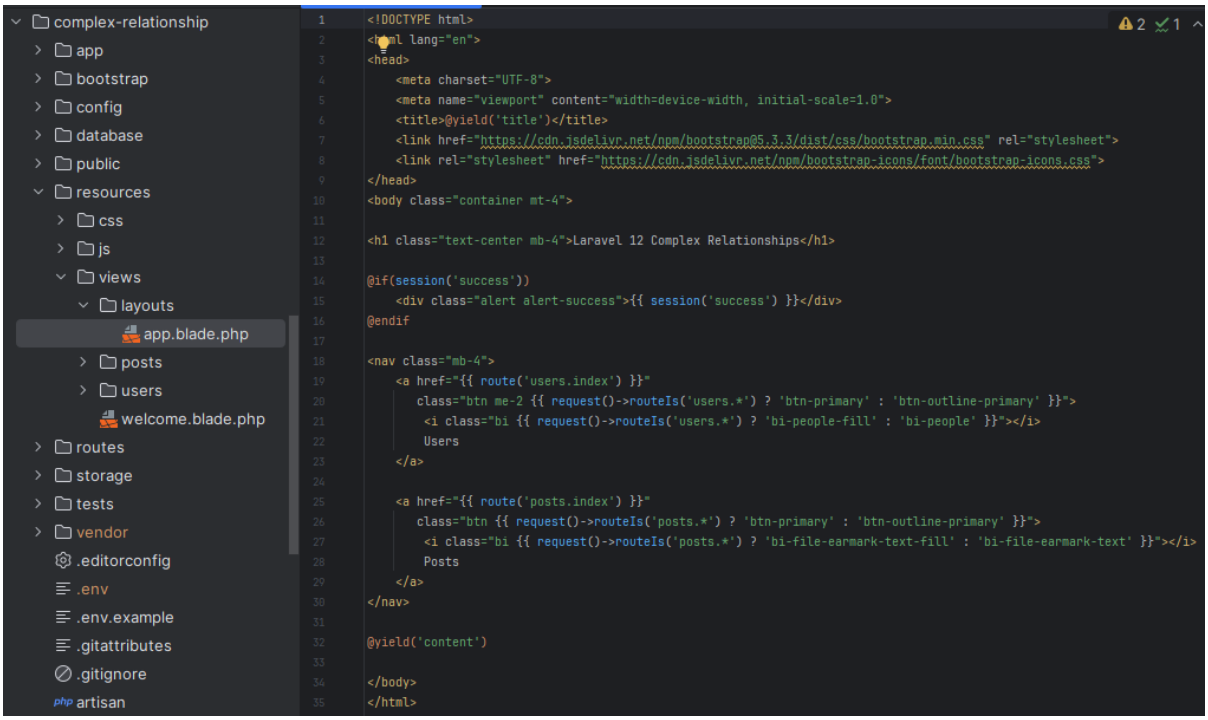
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:controller PostController
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Post;

3 usages  Lepi
class PostController extends Controller
{
    Lepi
    public function index()
    {
        $posts = Post::with( relations: 'user', 'tags')->get();
        return view( view: 'posts.index', compact( var_name: 'posts'));
    }

    Lepi
    public function show(Post $post)
    {
        return view( view: 'posts.show', compact( var_name: 'post'));
    }
}
```

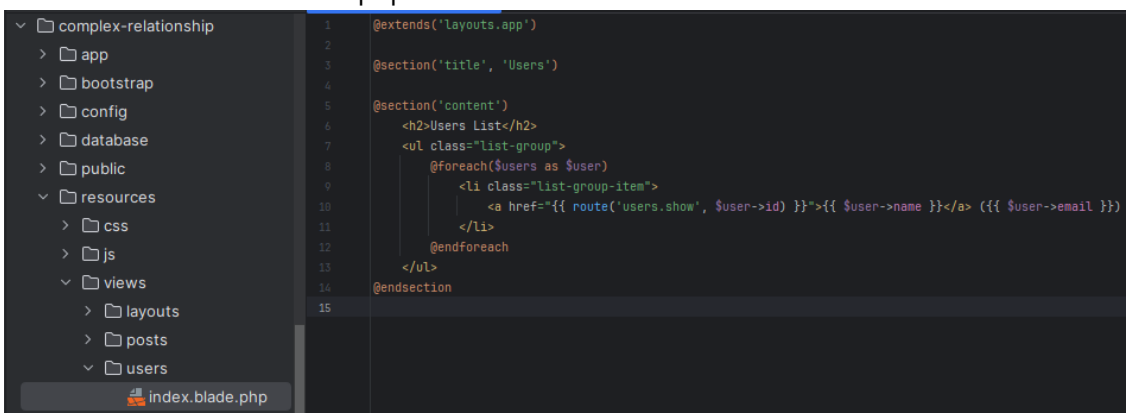
- Mendefinisikan Web Routes.

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UserController;
use App\Http\Controllers\PostController;

Route::get( uri: '/', function () {
    return view( view: 'welcome');
});

// Routes untuk User
Route::get( uri: '/users', [UserController::class, 'index'])->name( name: 'users.index');
Route::get( uri: '/users/{user}', [UserController::class, 'show'])->name( name: 'users.show');

// Routes untuk Post
Route::get( uri: '/posts', [PostController::class, 'index'])->name( name: 'posts.index');
Route::get( uri: '/posts/{post}', [PostController::class, 'show'])->name( name: 'posts.show');
```

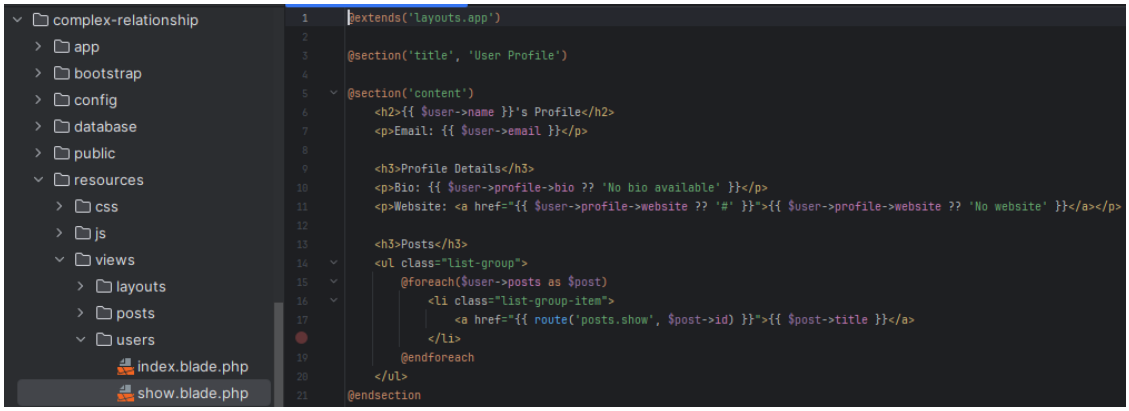
- Membuat Views Menggunakan Bootstrap.
  - Membuat folder layouts dan file app.blade.php di resources/views.



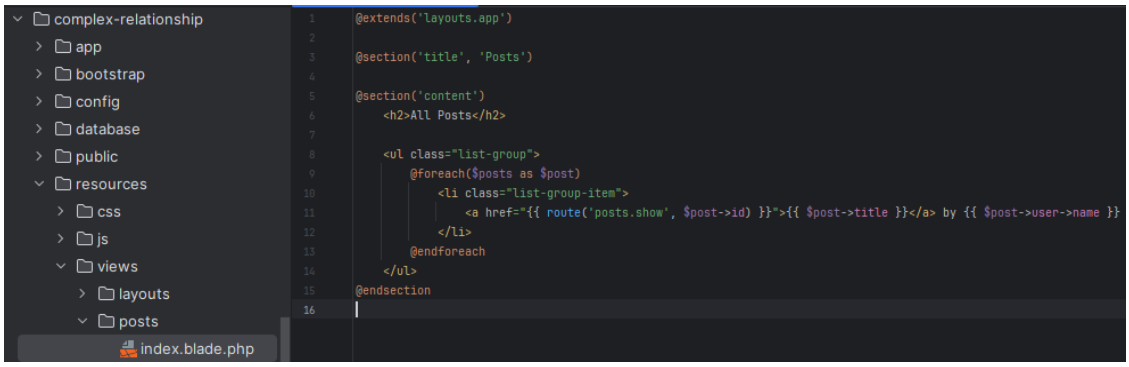
  - Membuat folder users di resources/views.
    - Membuat Views index.blade.php.



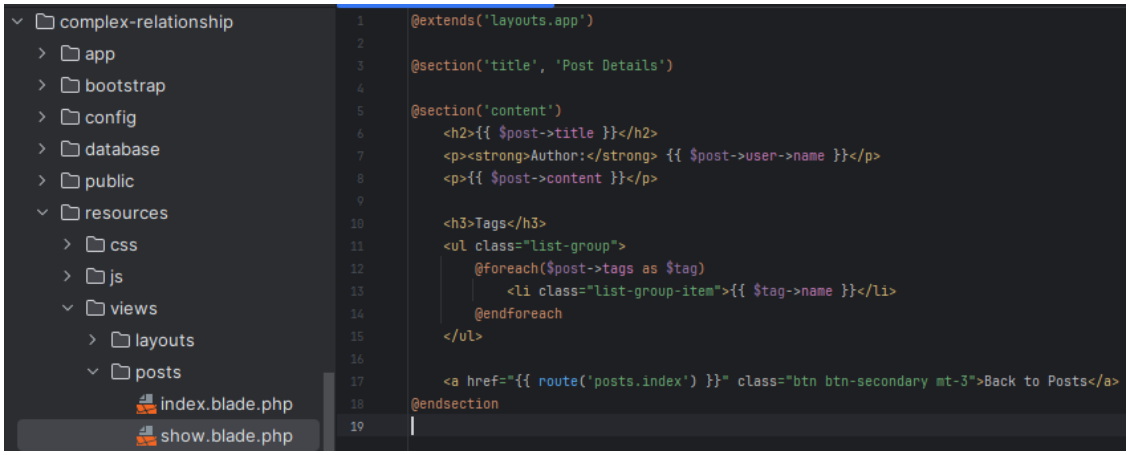
- Membuat Views show.blade.php.



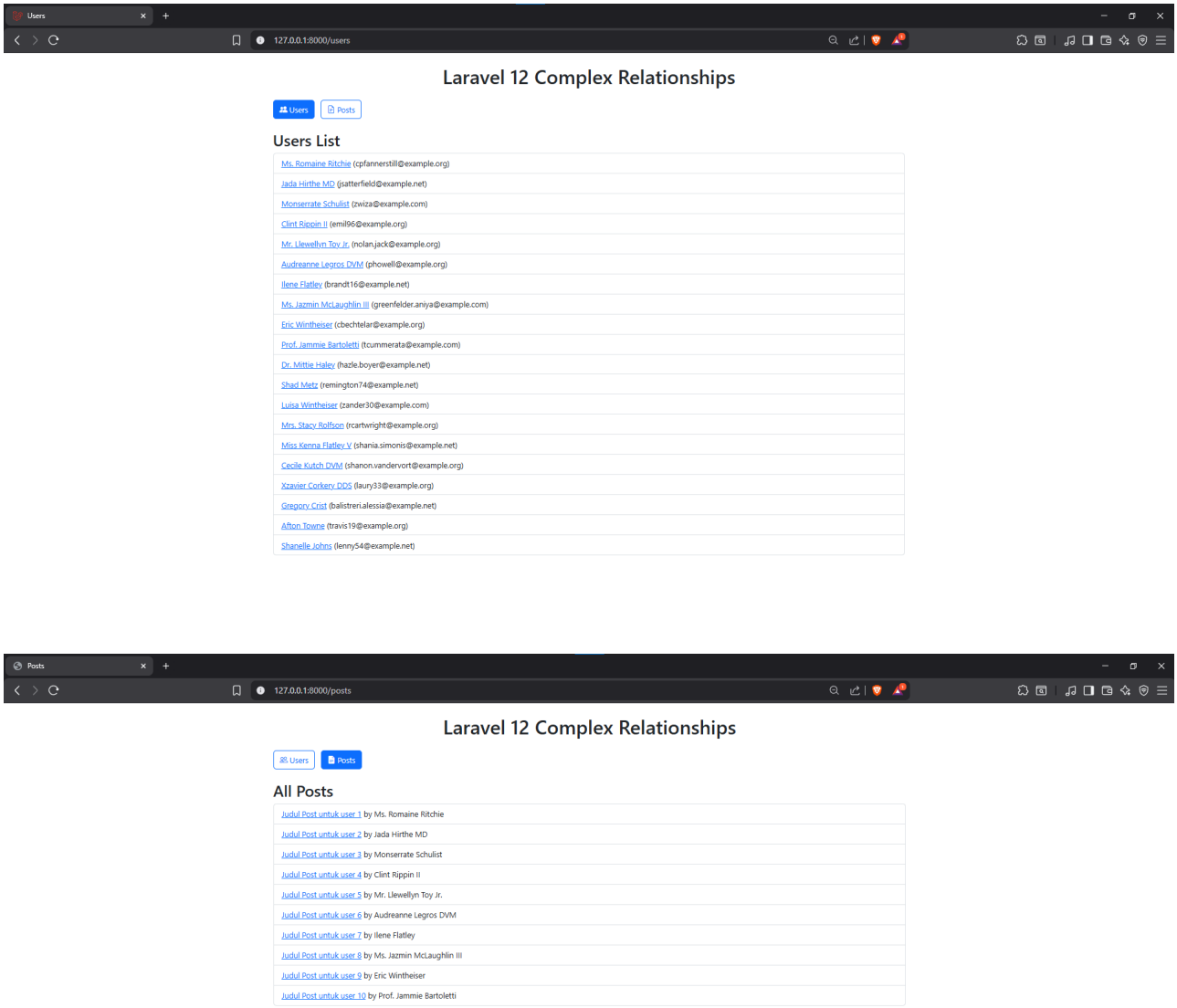
- Membuat folder posts di resources/views.
  - Membuat Views index.blade.php.



  - Membuat Views show.blade.php.



- Menjalankan aplikasi dan Menunjukkan hasil dibrowser.



2.2 Praktikum 2 – Paginasi dengan Eloquent ORM

- Membuat Model dan Migrasi Product, lalu menjalankan migrasi dengan perintah "php artisan migrate"

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/productpagination (main)
$ php artisan make:model Product -m
```

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    Lepi
    public function up(): void
    {
        Schema::create( table: 'products', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'name');
            $table->decimal( column: 'price', total: 10, places: 2);
            $table->timestamps();
        });
    }

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

- Membuat Seeder untuk Data Dummy.

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/productpagination (main)
$ php artisan make:seeder ProductSeeder
```

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Product;

1 usage  Lepi
class ProductSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    Lepi
    public function run(): void
    {
        Product::factory()->count( count: 50)->create();
    }
}
```

- Memperbarui file app/models/product.php

```php
<?php

namespace App\Models;

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

Lepi
class Product extends Model
{
    use HasFactory;

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

- Membuat Factory untuk Model Product.

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/productpagination (main)
$ php artisan make:factory ProductFactory --model=Product
```

```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

no usages  Lepi
class ProductFactory extends Factory
{
    Lepi
    public function definition(): array
    {
        return [
            'name' => fake()->word(),
            'price' => fake()->randomFloat( nbMaxDecimals: 2,  min: 10,  max: 1000),
        ];
    }
}
```

- Memodifikasi file database/seeders/DatabaseSeeder.php, lalu menjalankan perintah "php artisan db:seed"

```php
<?php

namespace Database\Seeders;

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

Lepi *
class DatabaseSeeder extends Seeder
{
    use WithoutModelEvents;

    Lepi
    public function run(): void
    {
        // User::factory(10)->create();

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

        $this->call([
            ProductSeeder::class,
        ]);
    }
}
```

- Membuat Controller ProductController.

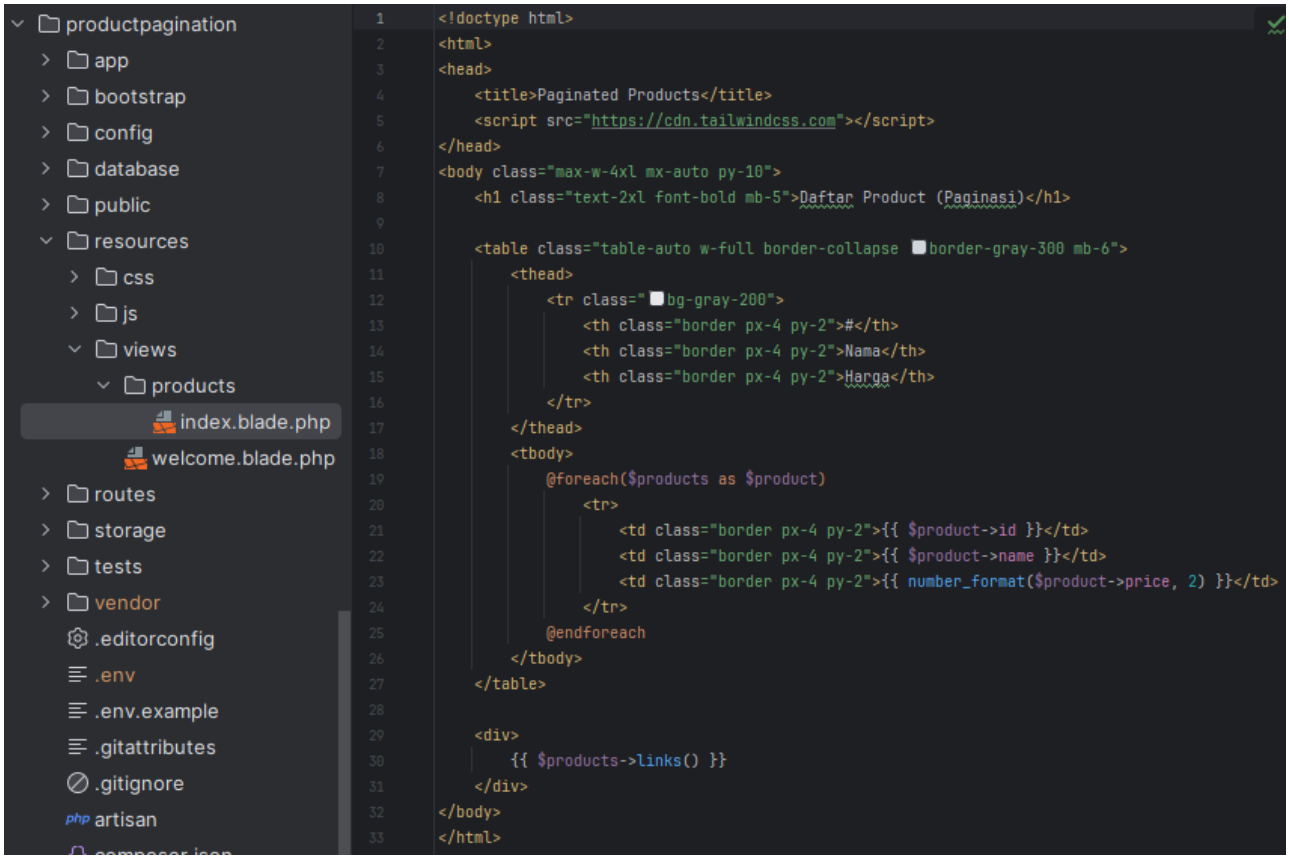
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/productpagination (main)
$ php artisan make:controller ProductController
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Product;

Lepi
class ProductController extends Controller
{
    Lepi
    public function index()
    {
        $products = Product::orderBy('id', 'asc')->paginate(10);
        return view( view: 'products.index', compact( var_name: 'products'));
    }
}
```

- Mendefinisikan Route

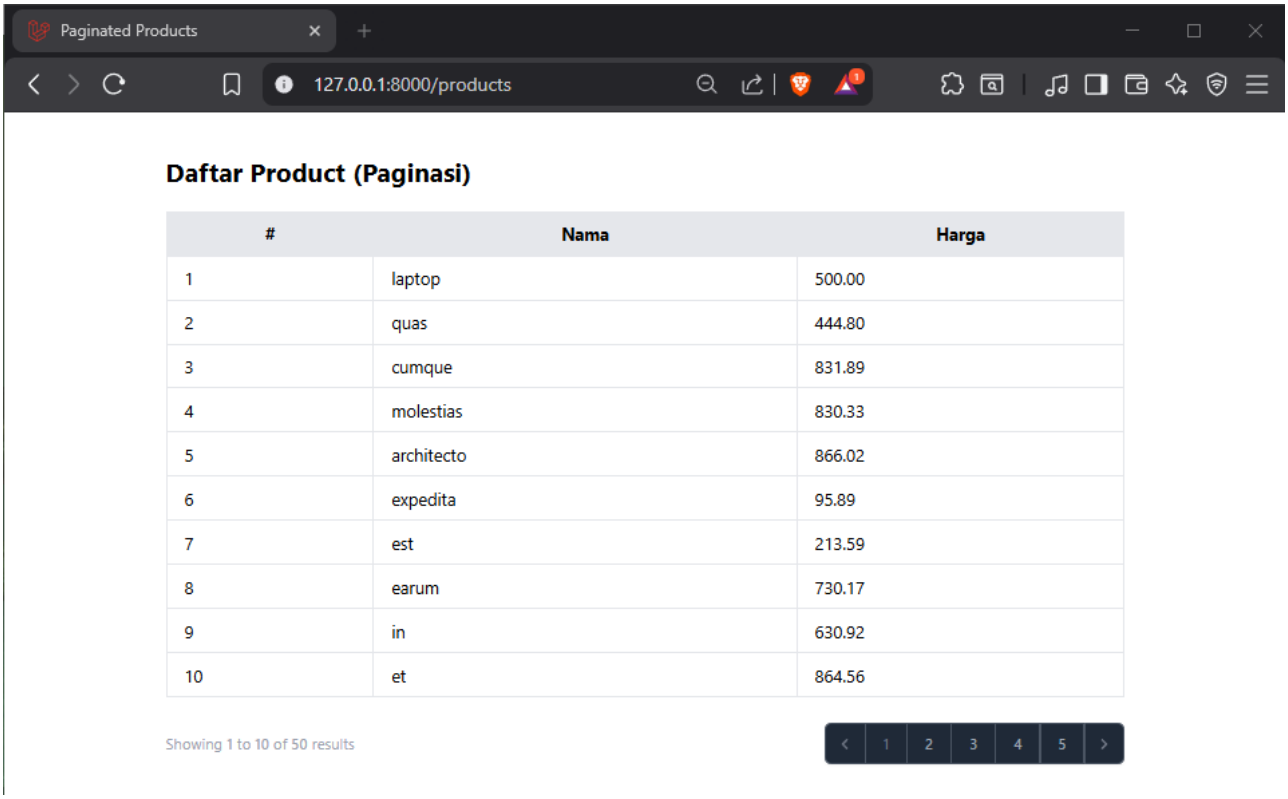
```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProductController;

Route::get( uri: '/', function () {
    return view( view: 'welcome');
});

Route::get( uri: '/products', [ProductController::class, 'index'])->name( name: 'products.index');
```

- Membuat View index.blade.php untuk Daftar Produk dengan Paginasi.




```blade
<!doctype html>
<html>
<head>
    <title>Paginated Products</title>
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="max-w-4xl mx-auto py-10">
    <h1 class="text-2xl font-bold mb-5">Daftar Product (Paginasi)</h1>

    <table class="table-auto w-full border-collapse border-gray-300 mb-6">
        <thead>
            <tr class="bg-gray-200">
                <th class="border px-4 py-2">#</th>
                <th class="border px-4 py-2">Nama</th>
                <th class="border px-4 py-2">Harga</th>
            </tr>
        </thead>
        <tbody>
            @foreach($products as $product)
                <tr>
                    <td class="border px-4 py-2">{{ $product->id }}</td>
                    <td class="border px-4 py-2">{{ $product->name }}</td>
                    <td class="border px-4 py-2">{{ number_format($product->price, 2) }}</td>
                </tr>
            @endforeach
        </tbody>
    </table>

    <div>
        {{ $products->links() }}
    </div>
</body>
</html>
```

- Menjalankan aplikasi dan Menunjukkan hasil dibrowser.



---

3. Hasil dan Pembahasan

Jelaskan apa hasil dari praktikum yang dilakukan.

- Apa Hasil dari Praktikum yang dilakukan?
  Hasil dari praktikum ini adalah aplikasi berbasis data yang berhasil mengelola dan menampilkan data dari skema database kompleks dengan memanfaatkan Eloquent ORM:
  - **Skema Relasi Fungsional:** Semua relasi yang direncanakan berhasil diimplementasikan:
    - **One-to-One** (User ↔ Profile): Pengambilan data profil dari pengguna tertentu berjalan lancar ($user->profile).
    - **One-to-Many** (User → Posts): Berhasil menampilkan semua postingan yang dimiliki oleh satu pengguna tertentu ($user->posts).
    - **Many-to-Many** (Posts ↔ Tags): Berhasil mengaitkan banyak Tag ke satu Post dan sebaliknya, dengan pengambilan data Tag menggunakan $post->tags.
  - **Penyelesaian N+1 Problem:** Penggunaan Eager Loading (Post::with('user', 'tags')->get()) di PostController berhasil mengambil semua Post beserta User (penulis) dan Tag terkait hanya dalam jumlah query yang minimal, memastikan performa aplikasi yang optimal.
  - **Antarmuka Tampilan Data:** Berhasil menampilkan data relasi yang kompleks secara terstruktur di View (menggunakan Bootstrap), memungkinkan navigasi yang jelas antar User dan Post mereka.
- Bagaimana Validasi Input Bekerja di Laravel?
  Validasi input di Laravel bekerja sebagai mekanisme middleware awal di Controller (meskipun tidak

secara eksplisit ditunjukkan di method ini, ini adalah standar Laravel) untuk memastikan integritas data sebelum relasi Eloquent diproses.

- **Penerapan pada Data Relasi:** Sebelum menyimpan data Post baru, validasi akan memastikan bahwa user_id, title, dan content sesuai dengan aturan (misalnya title adalah required). Jika validasi gagal, Model tidak akan pernah dipanggil.
- **Integritas Foreign Key:** Selain validasi input, integritas data relasi (seperti user_id di tabel posts) dijamin oleh foreign key constraints yang didefinisikan dalam Migrations ($table->foreign('user_id')->references('id')->on('users')). Ini memastikan bahwa Post tidak dapat dibuat tanpa User yang valid.

- Apa peran Masing-Masing Komponen (Route, Controller, View) dalam Program yang Dibuat?
  Ketiga komponen ini bekerja secara sinergis untuk memproses dan menyajikan data relasional:

- - **Route (Penentu Jalur Relasi):**
    - **Peran:** Mendefinisikan endpoint untuk melihat data relasi.
    - **Aksi:** Menggunakan Route Model Binding (misalnya Route::get('/users/{user}', [UserController::class, 'show'])). Route Model Binding secara otomatis mengambil instance Model User berdasarkan ID yang diberikan di URL, menyederhanakan kode Controller.
  - **Controller (Koordinator Query):**
    - **Peran:** Menentukan data apa yang harus diambil dan bagaimana data tersebut harus dimuat.
    - **Aksi:** Controller (misalnya PostController) memutuskan untuk menggunakan Eager Loading (with('user', 'tags')) saat mengambil data. Controller tidak menulis HTML tetapi meneruskan objek Model yang sudah dimuat relasinya ke View.
  - **View (Penyaji Data Relasi):**
    - **Peran:** Bertanggung jawab menampilkan data yang terhubung secara logis dan mudah dibaca.
    - **Aksi:** Mengakses data relasi menggunakan notasi panah, seperti {{ $post->user->name }} atau mengulang daftar relasi Tag dengan @foreach($post->tags as $tag). View hanya melakukan presentasi, bergantung sepenuhnya pada data relasi yang sudah dimuat oleh Controller.

---

4. Kesimpulan

Praktikum Eloquent Relationship berhasil mendemonstrasikan bagaimana Laravel secara efektif mengelola dan menyajikan data dari skema database yang kompleks. Keberhasilan ini didukung oleh pendefinisian relasi yang akurat (hasOne, hasMany, belongsToMany) dalam Model. Pencapaian teknis terbesar adalah penerapan Eager Loading (with()), yang secara kritis mengatasi masalah N+1 Query, memastikan bahwa aplikasi dapat mengambil data relasional (seperti penulis Post dan Tags terkait) dengan performa tinggi. Melalui pemanfaatan Route Model Binding dan Eager Loading di Controller, serta penyajian yang jelas di View, praktikum ini membuktikan bahwa Eloquent adalah alat yang sangat kuat untuk mengembangkan aplikasi berbasis data yang kompleks secara efisien, terstruktur, dan performance-minded.

5. Referensi

- Sumber dari :
  - Laraval 12 Training Kit: A Practical Guide to Modern Web Development. Link: https://lnkd.in/gm6ms5cf
  - BELAJAR LARAVEL Tutorial Framework Laravel Untuk Pemula by SANDHIKA GALIH. Link: https://www.youtube.com/@sandhikagalihWPU
  - Website Full Stack Open. Link: https://fullstackopen.com/en/