Laporan Modul 7: Eloquent Relationship & Pagination

Mata Kuliah: Workshop Web Lanjut
Nama: M. Iqbal Sayuti
NIM: 2024573010057
Kelas: TI-2C

Abstrak

Pada praktikum ini dipelajari bagaimana mengimplementasikan Eloquent Relationship untuk menghubungkan data antar tabel secara efisien serta penggunaan Pagination untuk membagi data ke dalam halaman-halaman yang lebih terstruktur. Melalui penerapan relasi seperti one-to-one, one-to-many, dan many-to-many, proses pengambilan data menjadi lebih mudah dan konsisten, sementara fitur paginasi dengan paginate() dan links() membantu menampilkan data berjumlah besar secara rapi dan meningkatkan pengalaman pengguna.

1. Dasar Teori

Dalam Laravel, Eloquent menyediakan cara yang efisien untuk mengelola hubungan antar tabel serta menampilkan data dalam bentuk halaman (pagination) agar lebih terstruktur dan mudah diakses.

1. Definisi Relasi = Eloquent menyediakan mekanisme untuk mendefinisikan hubungan antar model seperti one-to-one, one-to-many, many-to-many, dan polymorphic.
2. Fungsi Relasi di Model = Setiap relasi didefinisikan melalui fungsi khusus dalam model yang mengembalikan objek relasi, misalnya hasOne(), hasMany(), atau belongsTo().
3. Pengambilan Data Terkait = Dengan relasi, data dapat diambil menggunakan konsep lazy loading ($model->relasi) atau eager loading (Model::with('relasi')->get()).
4. Definisi Pagination = Teknik untuk membagi dataset besar menjadi beberapa halaman agar lebih teratur dan memudahkan navigasi pengguna.
5. Metode Paginate = Laravel menyediakan metode paginate() dan simplePaginate() untuk menghasilkan data yang otomatis dibagi berdasarkan jumlah item per halaman.
6. Navigasi Halaman Otomatis = Blade dapat menampilkan kontrol navigasi dengan fungsi {{ $data->links() }}.
7. Performa Lebih Baik = Pagination mengurangi beban server dengan hanya memuat data yang diperlukan pada setiap halaman.

2. Langkah-Langkah Praktikum

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

- Langkah 1: Buat dan Buka Proyek laravel
  laravel new complex-relationships
  cd complex-relationships
  code .

- Langkah 2 : Buat database `Create database eloquentrelation_db;`
  Kemudian install depedency MySQL dan setelahnya ubah file .env sesuai nama database
  `composer require doctrine/dbal`
  `php artisan config:clear`

- Langkah 3 : Buat migrasi untuk skema table
  `php artisan make:migration create_profiles_table`
  `php artisan make:migration create_posts_table`
  `php artisan make:migration create_tags_table`
  `php artisan make:migration create_post_tag_table`

- Langkah 4 : edit semua file skema yg telah dibuat tadi

database > migrations > 2025_11_04_072435_create_profiles_table.php

```php
<?php

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

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

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

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

database > migrations > 2025_11_04_072448_create_post_table.php

```php
<?php

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

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

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

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

database > migrations > 2025_11_04_072456_create_tags_table.php

```php
<?php

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

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

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

```

database > migrations > 2025_11_04_072539_create_post_tag_table.php

```php
<?php

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

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

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

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

Kemudian jalankan `php artisan migrate`

- Langkah 5 : Mendefinisikan Model Eloquent Jalankan perintah ini untuk membuat model :

```
php artisan make:model Profile
php artisan make:model Post
php artisan make:model Tag
```

- Langkah 6 : Edit file model yg telah dibuat

app > Models > Profile.php

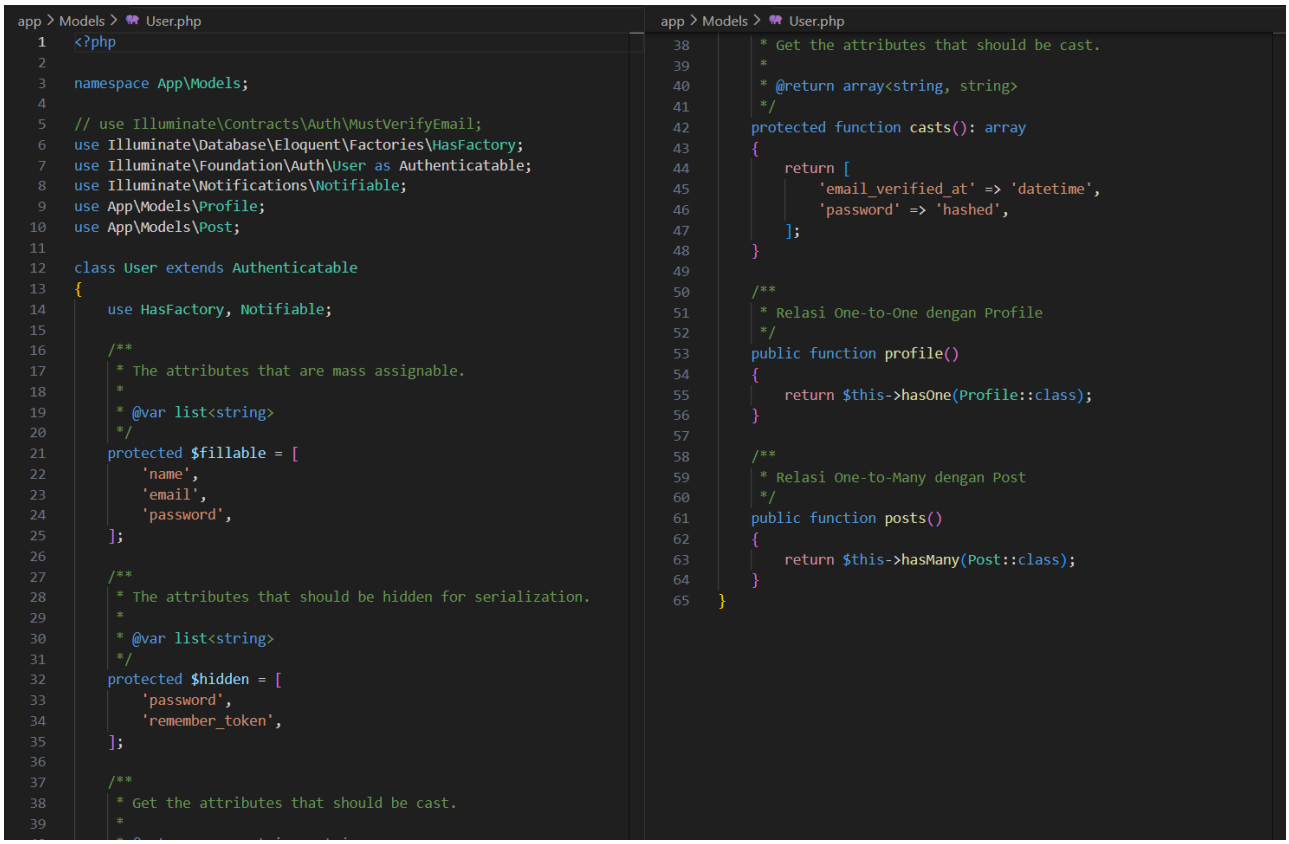
```php
<?php

namespace App\Models;

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

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

app > Models > Post.php

```php
<?php

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

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

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

app > Models > Tag.php

```php
<?php

namespace App\Models;

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

class Tag extends Model
{
    use HasFactory;

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

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



app > Models > User.php

```php
<?php

namespace App\Models;

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

class User extends Authenticatable
{
    use HasFactory, Notifiable;

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

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
```

app > Models > User.php

```php
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }

    /**
     * Relasi One-to-One dengan Profile
     */
    public function profile()
    {
        return $this->hasOne(Profile::class);
    }

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

- Langkah 7 : Membuat seeder `php artisan make:seeder DatabaseSeeder` kemudian ubah file DatabaseSeeder

database > seeders > DatabaseSeeder.php

```php
<?php

namespace Database\Seeders;

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

class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
    // Membuat 10 user menggunakan factory
    User::factory(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);
        }
    }
```

Jalankan `php artisan db:seed`

- Langkah 8 : Membuat Controller

  `php artisan make:controller UserController`

  `php artisan make:controller PostController`

  Kemudian edit kedua file controller yang sudah dibuat

app > Http > Controllers > UserController.php

```php
<?php

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

class UserController extends Controller
{
    /**
     * Menampilkan daftar semua user dengan profile dan posts terkait
     */
    public function index()
    {
        $users = User::with('profile', 'posts')->get();
        return view('users.index', compact('users'));
    }

    /**
     * Menampilkan detail user tertentu
     */
    public function show(User $user)
    {
        // Menggunakan route model binding, Laravel akan otomatis
        // mencari user berdasarkan ID yang diberikan
        return view('users.show', compact('user'));
    }
}
```

app > Http > Controllers > PostController.php

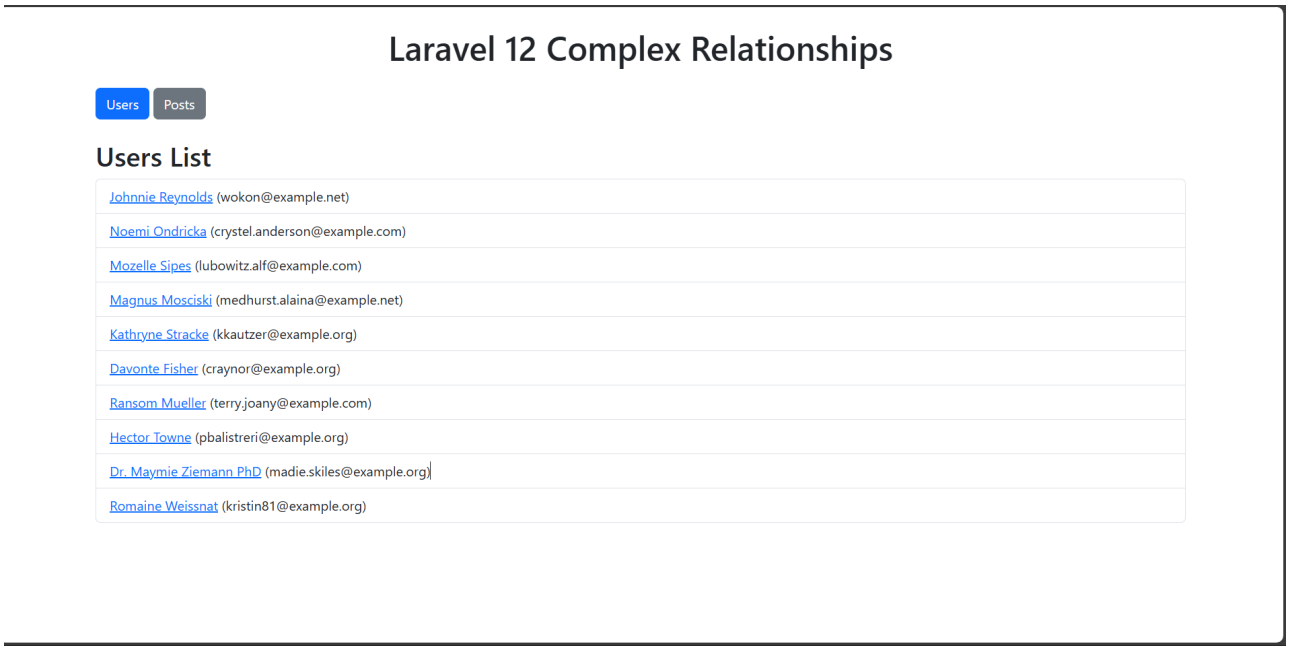
```php
<?php

namespace App\Http\Controllers;

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

class PostController extends Controller
{
    /**
     * Menampilkan daftar semua posts dengan user dan tags terkait
     */
    public function index()
    {
        $posts = Post::with('user', 'tags')->get();
        return view('posts.index', compact('posts'));
    }

    /**
     * Menampilkan detail post tertentu
     */
    public function show(Post $post)
    {
        // Menggunakan route model binding, Laravel akan otomatis
        // mencari post berdasarkan ID yang diberikan
        return view('posts.show', compact('post'));
    }
}
```

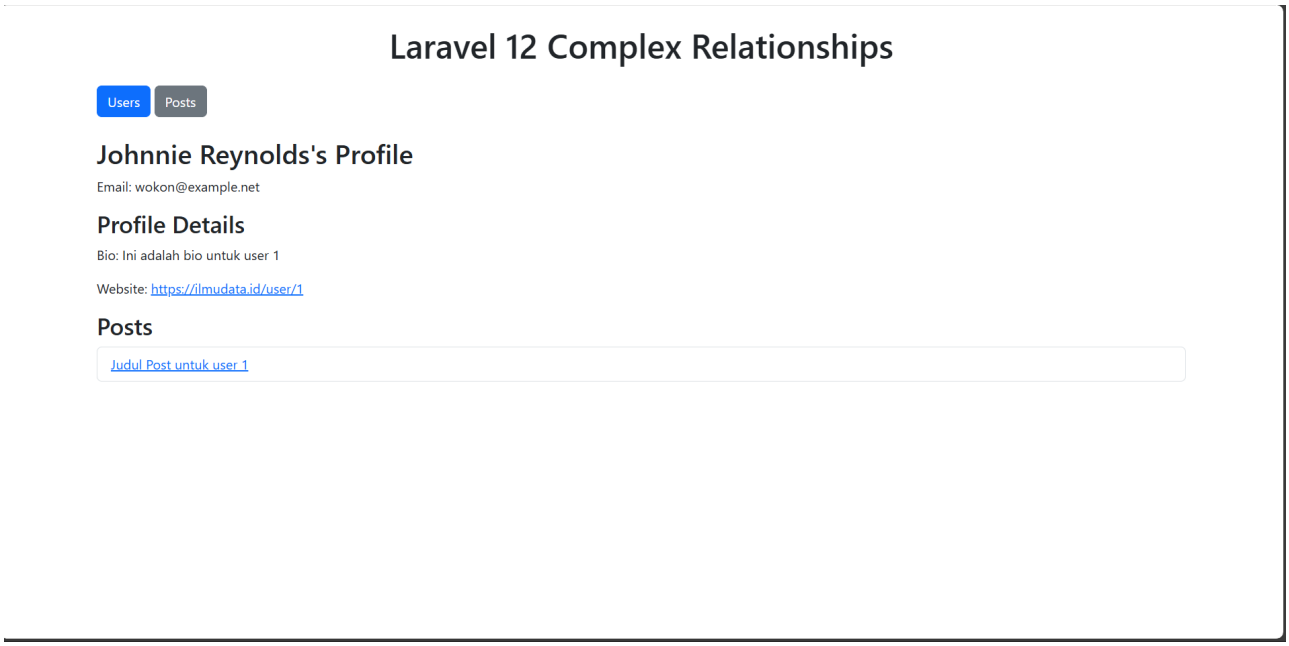
- Langkah 9: Definisikan route
  Buka Routes/Web.php Dan isi dengan code berikut

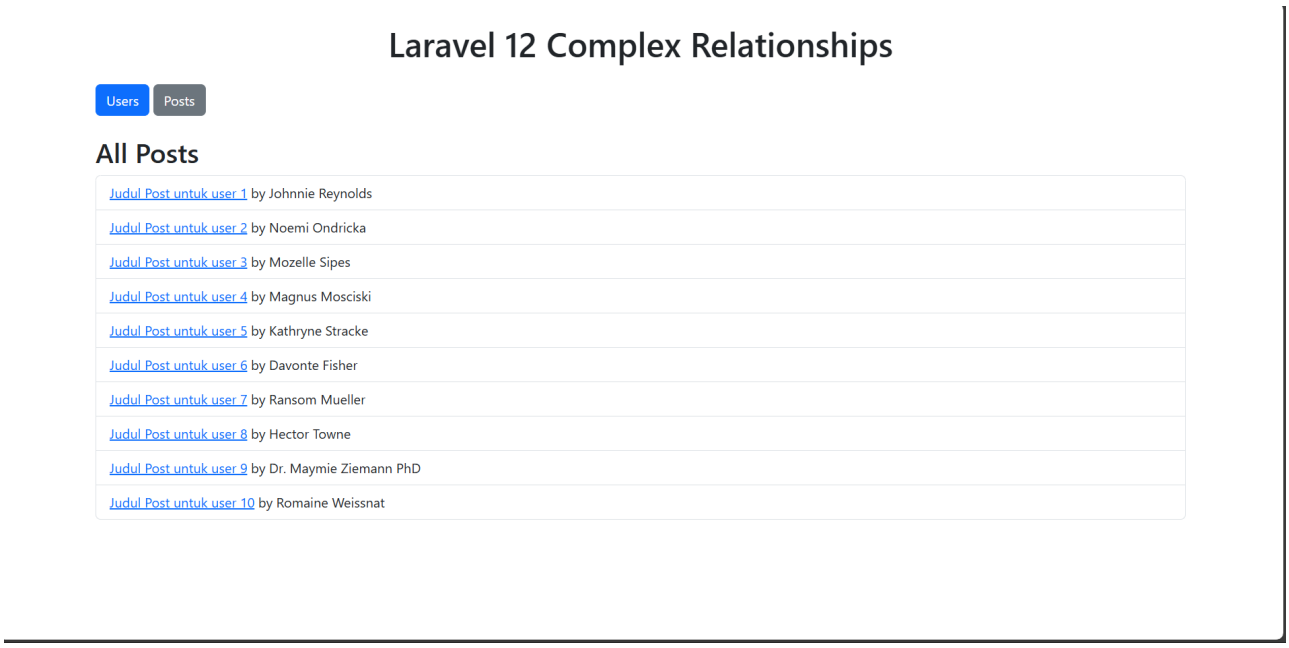
routes > web.php

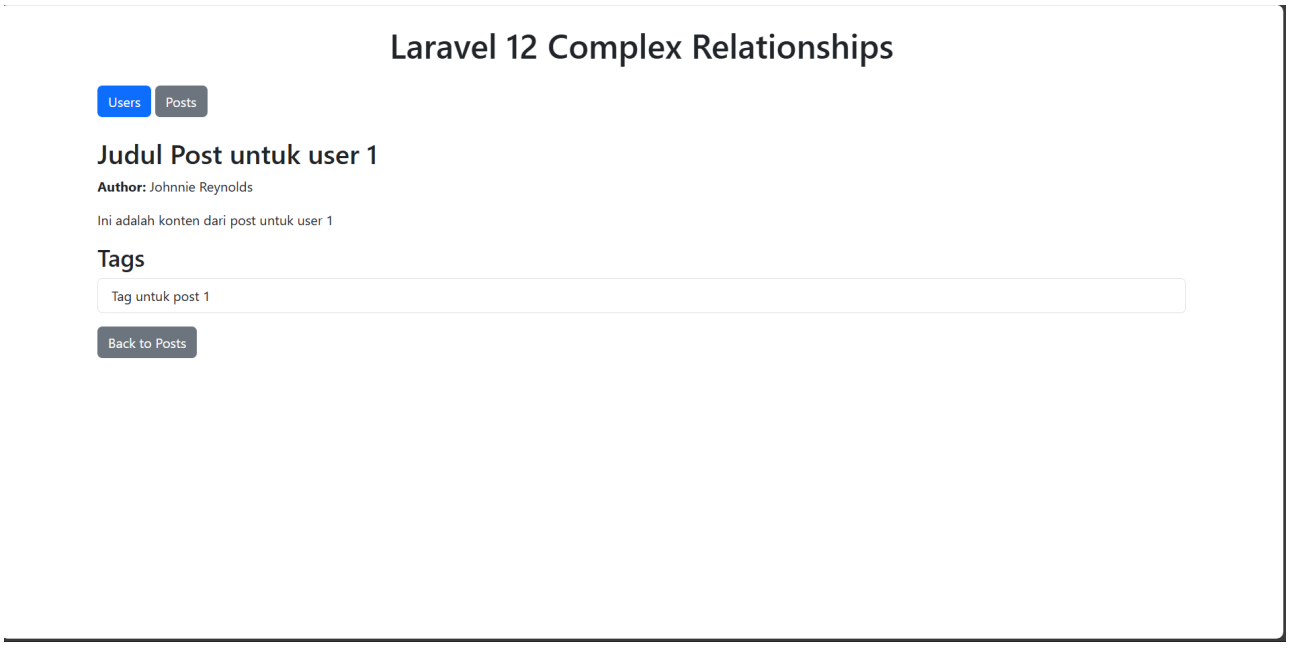
```php
<?php

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

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

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

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

```

- Langkah 10: Uji Rute
  Mulai Php `Artisan serve`
  Buka Browser Dan Kunjungi
  http://127.0.0.1:8000/product/users
  http://127.0.0.1:8000/product/users/1
  http://127.0.0.1:8000/product/posts
  http://127.0.0.1:8000/product/posts/1




Laravel 12 Complex Relationships
Users
Posts
Johnnie Reynolds's Profile
Email: wokon@example.net
Profile Details
Bio: Ini adalah bio untuk user 1
Website: https://ilmudata.id/user/1
Posts
Judul Post untuk user 1


Laravel 12 Complex Relationships
Users
Posts
All Posts
Judul Post untuk user 1 by Johnnie Reynolds
Judul Post untuk user 2 by Noemi Ondricka
Judul Post untuk user 3 by Mozelle Sipes
Judul Post untuk user 4 by Magnus Mosciski
Judul Post untuk user 5 by Kathryne Stracke
Judul Post untuk user 6 by Davonte Fisher
Judul Post untuk user 7 by Ransom Mueller
Judul Post untuk user 8 by Hector Towne
Judul Post untuk user 9 by Dr. Maymie Ziemann PhD
Judul Post untuk user 10 by Romaine Weissnat


Laravel 12 Complex Relationships
Users
Posts
Judul Post untuk user 1
Author: Johnnie Reynolds
Ini adalah konten dari post untuk user 1
Tags
Tag untuk post 1
Back to Posts

- Langkah 11 : Membuat Views Menggunakan Bootstrap Buat layouts : resources/views/layouts/app.blade.php

resources > views > layouts > app.blade.php

```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title')</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container mt-4">

    <h1 class="text-center mb-4">Laravel 12 Complex Relationships</h1>

    @if(session('success'))
        <div class="alert alert-success">{{ session('success') }}</div>
    @endif

    <nav class="mb-4">
        <a href="{{ route('users.index') }}" class="btn btn-primary">Users</a>
        <a href="{{ route('posts.index') }}" class="btn btn-secondary">Posts</a>
    </nav>

    @yield('content')

</body>
</html>
```

buat file resources/views/users/index.blade.php

resources > views > users > index.blade.php

```blade
@extends('layouts.app')

@section('title', 'Users')

@section('content')
    <h2>Users List</h2>
    <ul class="list-group">
        @foreach($users as $user)
            <li class="list-group-item">
                <a href="{{ route('users.show', $user->id) }}">{{ $user->name }}</a> ({{ $user->email }})
            </li>
        @endforeach
    </ul>
@endsection
```

Buat file resources/views/users/show.blade.php

resources > views > users > show.blade.php

```blade
@extends('layouts.app')

@section('title', 'User Profile')

@section('content')
    <h2>{{ $user->name }}'s Profile</h2>
    <p>Email: {{ $user->email }}</p>

    <h3>Profile Details</h3>
    <p>Bio: {{ $user->profile->bio ?? 'No bio available' }}</p>
    <p>Website: <a href="{{ $user->profile->website ?? '#' }}">{{ $user->profile->website ?? 'No website' }}</a></p>

    <h3>Posts</h3>
    <ul class="list-group">
        @foreach($user->posts as $post)
            <li class="list-group-item">
                <a href="{{ route('posts.show', $post->id) }}">{{ $post->title }}</a>
            </li>
        @endforeach
    </ul>
@endsection
```

Buat file resources/views/posts/index.blade.php

resources > views > posts > index.blade.php

```blade
@extends('layouts.app')

@section('title', 'Posts')

@section('content')
    <h2>All Posts</h2>

    <ul class="list-group">
        @foreach($posts as $post)
            <li class="list-group-item">
                <a href="{{ route('posts.show', $post->id) }}">{{ $post->title }}</a> by {{ $post->user->name }}
            </li>
        @endforeach
    </ul>
@endsection
```

Buat file resources/views/posts/show.blade.php

resources > views > posts > show.blade.php

```blade
@extends('layouts.app')

@section('title', 'Post Details')

@section('content')
    <h2>{{ $post->title }}</h2>
    <p><strong>Author:</strong> {{ $post->user->name }}</p>
    <p>{{ $post->content }}</p>

    <h3>Tags</h3>
    <ul class="list-group">
        @foreach($post->tags as $tag)
            <li class="list-group-item">{{ $tag->name }}</li>
        @endforeach
    </ul>

    <a href="{{ route('posts.index') }}" class="btn btn-secondary mt-3">Back to Posts</a>
@endsection
```

Praktikum 2: Paginasi dengan Eloquent ORM

- Langkah 1: Buat dan Buka Proyek laravel
  laravel new productpagination
  cd productpagination
  code .
- Langkah 2: Konfigurasi dan buat database Buat database bernama pagination_db dan isikan ke file .env
  Kemudian install depedency
- Langkah 3: Membuat Model dan Migrasi Product Jalankan `php artisan make:model Product -m`
  Perbarui file migrasi :

database > migrations > 2025_11_04_083621_create_products_table.php

```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.
     */
    public function up(): void
    {
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->decimal('price', 10, 2);
        $table->timestamps();
    });
    }

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

Jalankan `php artisan migrate`

- Langkah 4: Buat Seeder untk data dummy
  Jalankan `php artisan make:seeder ProductSeeder`
  lalu isi kode :

database > seeders > ProductSeeder.php

```php
<?php

namespace Database\Seeders;

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

class ProductSeeder extends Seeder
{
    public function run(): void
    {
        Product::factory()->count(50)->create();
    }
}
```

Kemudian Perbaharui fileProduct.php

app > Models > Product.php

```php
<?php

namespace App\Models;

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

class Product extends Model
{
    use HasFactory;

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

- Langkah 5 : Buat factory untuk model Product jalankan perintah berikut:
  `php artisan make:factory ProductFactory --model=Product`

Kemudian perbarui file yg barusan dibuat

database > factories > ProductFactory.php

```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

class ProductFactory extends Factory
{
    public function definition(): array
    {
        return [
            'name' => fake()->word(),
            'price' => fake()->randomFloat(2, 10, 1000),
        ];
    }
}
```

- Langkah 6 : Modifikasi file
  Buka dan perbarui file database/seeders/DatabaseSeeder.php

database > seeders > DatabaseSeeder.php

```php
<?php

namespace Database\Seeders;

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

class DatabaseSeeder extends Seeder
{
    use WithoutModelEvents;

    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        $this->call([
            ProductSeeder::class,
        ]);
    }

}
```

Dan jalankan `php artisan db:seed`

- Langkah 7 : Membuat Controller untuk Paginasi
  Jalankan `php artisan make:controller ProductController` dan perbarui

app > Http > Controllers > ProductController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Product;

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

- Langkah 8: Tambahkan route baru
  Buka routes/web.php dan tambahkan:

routes > web.php

```php
<?php

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

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

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

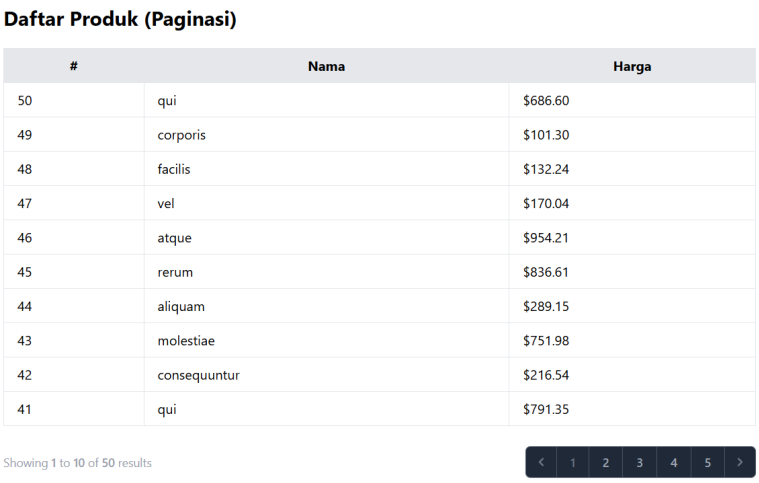
- Langkah 9: Membuat View untuk Daftar Produk dengan Paginasi : Buat File Baru di Direktori resources/views/products/index.blade.php :
  Isi Dengan Code Berikut

esources > views > products > index.blade.php

```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 Produk (Paginasi)</h1>

    <table class="table-auto w-full border-collapse border 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>
```

- Langkah 10: Jalankan Aplikasi
  Jalankan Server Pengembangan
  `php php artisan serve`
  Akses aplikasi di:

http://127.0.0.1:8000/products



Daftar Produk (Paginasi)

#	Nama	Harga
50	qui	$686.60
49	corporis	$101.30
48	facilis	$132.24
47	vel	$170.04
46	atque	$954.21
45	rerum	$836.61
44	aliquam	$289.15
43	molestiae	$751.98
42	consequuntur	$216.54
41	qui	$791.35

Showing 1 to 10 of 50 results

‹ 1 2 3 4 5 ›

---

3. Hasil dan Pembahasan

Pada praktikum 7 ini, Anda telah berhasil:

- Mengimplementasikan berbagai jenis Eloquent Relationship seperti one-to-one, one-to-many, dan many-to-many untuk menghubungkan data antar tabel.
- Mendefinisikan fungsi relasi pada model menggunakan metode seperti hasOne(), hasMany(), belongsTo(), dan belongsToMany().
- Menggunakan eager loading (with()) untuk mengambil data terkait secara efisien dan mengurangi jumlah query ke database.
- Menampilkan data hasil relasi pada tampilan (view) sehingga setiap entitas dapat menampilkan data yang terkait dengannya.
- Menerapkan pagination menggunakan metode paginate() pada query Eloquent untuk membagi dataset menjadi beberapa halaman.
- Menggunakan {{ $data->links() }} di Blade untuk menghasilkan navigasi halaman secara otomatis.
- Menggabungkan pagination dengan relasi, seperti mem-paginate data yang memiliki hubungan dengan model lain.

Hasil praktikum menunjukkan bahwa penggunaan Eloquent Relationship memudahkan pengelolaan data yang saling terhubung tanpa harus membuat query SQL manual yang panjang, sementara fitur Pagination membantu menampilkan data dalam jumlah besar secara lebih terstruktur, efisien, dan ramah pengguna. Secara keseluruhan, kombinasi relasi dan paginasi membuat aplikasi lebih fleksibel, rapi, dan optimal dalam menangani data.

---

4. Kesimpulan

Pada praktikum ini dapat disimpulkan bahwa penerapan Eloquent Relationship dan Pagination memberikan kemudahan yang signifikan dalam pengelolaan serta penampilan data pada aplikasi Laravel. Eloquent Relationship memungkinkan pengembang menghubungkan tabel secara efisien melalui relasi yang

terstruktur, sehingga proses pengambilan dan pengolahan data menjadi lebih sederhana tanpa memerlukan query SQL kompleks. Sementara itu, Pagination membantu membagi data berjumlah besar ke dalam halaman yang lebih teratur dan mudah diakses, sehingga meningkatkan performa aplikasi sekaligus pengalaman pengguna. Dengan memahami kedua konsep ini, pengembangan aplikasi berbasis Laravel menjadi lebih optimal, rapi, dan mudah dikembangkan di tahap berikutnya.

---

5. Referensi

- chatgpt.com
- https://hackmd.io/@mohdrzu/r1RPvWaCxx
