

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/belongsToMany):** 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.

```
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      & Lepi
9      public function up()
10     {
11         Schema::create('profiles', function (Blueprint $table) {
12             $table->id();
13             $table->unsignedBigInteger('user_id')->unique();
14             $table->text('bio')->nullable();
15             $table->string('website')->nullable();
16             $table->timestamps();
17
18             $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
19         });
20
21         & Lepi
22         public function down()
23         {
24             Schema::dropIfExists('profiles');
25         }
26     };
27 }
```

- Mengedit Migrasi Posts table yang sudah dibuat.

```
1  <?php
2
3  > use ...;
4
5
6
7  return new class extends Migration {
8      /**
9      * Lepi
10     * public function up()
11     {
12         Schema::create('posts', function (Blueprint $table) {
13             $table->id();
14             $table->unsignedBigInteger('user_id');
15             $table->string('title');
16             $table->text('content');
17             $table->timestamps();
18
19             $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
20         });
21     }
22
23     /**
24     * Lepi
25     * public function down()
26     {
27         Schema::dropIfExists('posts');
28     }
29 };
30 
```

- Mengedit Migrasi Tags table yang sudah dibuat.

```
1  <?php
2
3  > use ...;
4
5
6
7  return new class extends Migration {
8      /**
9      * Lepi
10     * public function up()
11     {
12         Schema::create('tags', function (Blueprint $table) {
13             $table->id();
14             $table->string('name')->unique();
15             $table->timestamps();
16         });
17     }
18
19     /**
20     * Lepi
21     * public function down()
22     {
23         Schema::dropIfExists('tags');
24     }
25 };
26 
```

- Mengedit Migrasi Tag tabel yang sudah dibuat.

```
1  <?php
2
3  > use ...
4
5
6
7  return new class extends Migration {
8      & Lepi
9      public function up()
10     {
11         Schema::create('post_tag', function (Blueprint $table) {
12             $table->id();
13             $table->unsignedBigInteger('post_id');
14             $table->unsignedBigInteger('tag_id');
15             $table->timestamps();
16
17             $table->foreign('post_id')->references('id')->on('posts')->onDelete('cascade');
18             $table->foreign('tag_id')->references('id')->on('tags')->onDelete('cascade');
19         });
20
21     & Lepi
22     public function down()
23     {
24         Schema::dropIfExists('post_tag');
25     }
26 }
```

- 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

```
1 <?php
2
3 namespace App\Models;
4
5 // use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use App\Models\Post;
7 use App\Models\Profile;
8 use Illuminate\Database\Eloquent\Factories\HasFactory;
9 use Illuminate\Database\Eloquent\Model;
10 use Illuminate\Foundation\Auth\User as Authenticatable;
11 use Illuminate\Notifications\Notifiable;
12
13 /**
14  * @Lepi *
15  */
16 class User extends Model
17 {
18     use HasFactory, Notifiable;
19
20     protected $fillable = [
21         'name',
22         'email',
23         'password',
24     ];
25
26     protected $hidden = [
27         'password',
28         'remember_token',
29     ];
30
31     /**
32      * @Lepi
33      */
34     protected function casts(): array
35     {
36         return [
37             'email_verified_at' => 'datetime',
38             'password' => 'hashed',
39         ];
40     }
41
42     /**
43      * @Lepi
44      */
45     public function profile()
46     {
47         return $this->hasOne(related: Profile::class);
48     }
49
50     /**
51      * @Lepi
52      */
53     public function posts()
54     {
55         return $this->hasMany(related: Post::class);
56     }
57 }
```

- Mengedit file Model Profile.php

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use App\Models\User;
8
9 ↳ Lepi
10 class Profile extends Model
11 {
12     use HasFactory;
13
14     /**
15      * Kolom yang dapat diisi secara massal
16      */
17     protected $fillable = ['user_id', 'bio', 'website'];
18
19     /**
20      * Relasi inverse One-to-One dengan User
21      */
22     ↳ Lepi
23     public function user()
24     {
25         return $this->belongsTo( related: User::class);
26     }
27 }
```

- Mengedit file Model Post

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use App\Models\User;
8  use App\Models\Tag;
9
10 Lepi
11 class Post extends Model
12 {
13     use HasFactory;
14
15     /**
16      * Kolom yang dapat diisi secara massal
17      */
18     protected $fillable = ['user_id', 'title', 'content'];
19
20     /**
21      * Relasi inverse One-to-Many dengan User
22      */
23     Lepi
24     public function user()
25     {
26         return $this->belongsTo(related: User::class);
27     }
28
29     /**
30      * Relasi Many-to-Many dengan Tag
31      */
32     Lepi
33     public function tags()
34     {
35         return $this->belongsToMany(related: Tag::class);
36     }
37 }
```

- Mengedit file Model Tag

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use App\Models\Post;
8
9 ↳ Lepi
10 class Tag extends Model
11 {
12     use HasFactory;
13
14     /**
15      * Kolom yang dapat diisi secara massal
16      */
17     protected $fillable = ['name'];
18
19     /**
20      * Relasi Many-to-Many dengan Post
21      */
22     ↳ Lepi
23     public function posts()
24     {
25         return $this->belongsToMany(related: Post::class);
26     }
27 }
```

- 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
```

```

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Seeder;
6 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7 use App\Models\User;
8 use App\Models\Profile;
9 use App\Models\Post;
10 use App\Models\Tag;
11
12 /**
13 * Lepi
14 */
15 class DatabaseSeeder extends Seeder
16 {
17     /**
18      * Seed the application's database.
19      */
20     /**
21      * Lepi
22     */
23     public function run(): void
24     {
25         /**
26          * Membuat 10 user menggunakan factory
27          */
28         User::factory(count: 10)->create();
29
30         /**
31          * Membuat profile untuk setiap user
32          */
33         foreach (User::all() as $user) {
34             /**
35              * $user->profile()->create([
36                  'bio' => 'Ini adalah bio untuk user ' . $user->id,
37                  'website' => 'https://ilmudata.id/user/' . $user->id,
38              ]);
39         }
40
41         /**
42          * Membuat post untuk setiap user
43          */
44         foreach (User::all() as $user) {
45             /**
46               * $user->posts()->create([
47                   'title' => 'Judul Post untuk user ' . $user->id,
48                   'content' => 'Ini adalah konten dari post untuk user ' . $user->id,
49               ]);
50         }
51
52         /**
53          * Membuat tag dan mengasosiasikannya dengan posts
54          */
55         foreach (Post::all() as $post) {
56             /**
57               * $tag = Tag::create(['name' => 'Tag untuk post ' . $post->id]);
58               * $post->tags()->attach($tag->id);
59             */
60         }
61     }
62 }

```

- Membuat Controller

- Membuat Controller UserController

```

Administrator@WIN-C2ZCTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/complex-relationship (main)
$ php artisan make:controller UserController

```

```
1 <?php
2
3     namespace App\Http\Controllers;
4
5     use Illuminate\Http\Request;
6     use App\Models\User;
7
8     class UserController extends Controller
9     {
10         public function index()
11         {
12             $users = User::with( relations: 'profile', 'posts')->get();
13             return view( view: 'users.index', compact( var_name: 'users'));
14         }
15
16         public function show(User $user)
17         {
18             return view( view: 'users.show', compact( var_name: 'user'));
19         }
20     }
```

- 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
```

```
1 <?php
2
3     namespace App\Http\Controllers;
4
5     use Illuminate\Http\Request;
6     use App\Models\Post;
7
8     class PostController extends Controller
9     {
10         public function index()
11         {
12             $posts = Post::with( relations: 'user', 'tags')->get();
13             return view( view: 'posts.index', compact( var_name: 'posts'));
14         }
15
16         public function show(Post $post)
17         {
18             return view( view: 'posts.show', compact( var_name: 'post'));
19         }
20     }
```

- Mendefinisikan Web Routes.

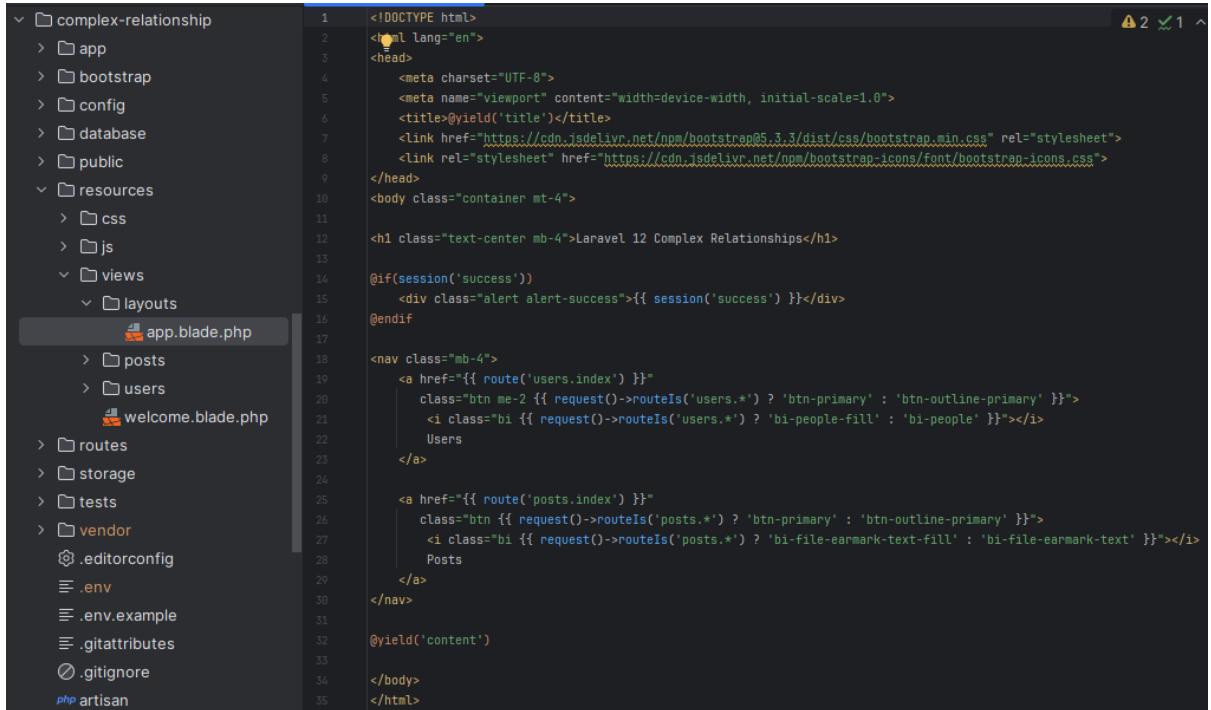
```

1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\UserController;
5  use App\Http\Controllers\PostController;
6
7  Route::get('/', function () {
8      return view('welcome');
9  });
10
11 // Routes untuk User
12 Route::get('users', [UserController::class, 'index'])->name('users.index');
13 Route::get('users/{user}', [UserController::class, 'show'])->name('users.show');
14
15 // Routes untuk Post
16 Route::get('posts', [PostController::class, 'index'])->name('posts.index');
17 Route::get('posts/{post}', [PostController::class, 'show'])->name('posts.show');

```

- Membuat Views Menggunakan Bootstrap.

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



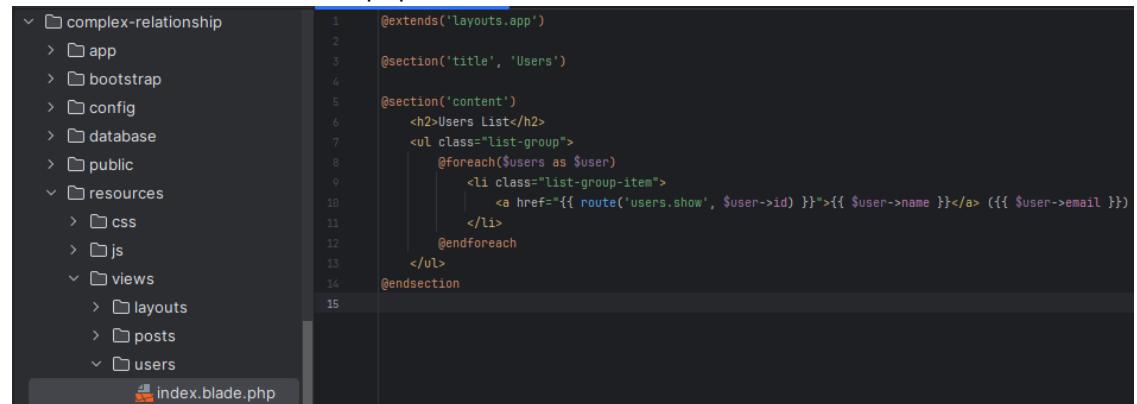
```

<!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@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css">
</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 me-2 {{ request()>routeIs('users.*') ? 'btn-primary' : 'btn-outline-primary' }}"
            <i class="bi {{ request()>routeIs('users.*') ? 'bi-people-fill' : 'bi-people' }}></i>
            Users
        </a>
        <a href="{{ route('posts.index') }}"
            class="btn {{ request()>routeIs('posts.*') ? 'btn-primary' : 'btn-outline-primary' }}"
            <i class="bi {{ request()>routeIs('posts.*') ? 'bi-file-earmark-text-fill' : 'bi-file-earmark-text' }}></i>
            Posts
        </a>
    </nav>
    @yield('content')
</body>
</html>

```

- Membuat folder users di resources/views.

- Membuat Views index.blade.php.



```

@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

```

■ Membuat Views show.blade.php.

```

1  @extends('layouts.app')
2
3  @section('title', 'User Profile')
4
5  @section('content')
6      <h2>{{ $user->name }}'s Profile</h2>
7      <p>Email: {{ $user->email }}</p>
8
9      <h3>Profile Details</h3>
10     <p>Bio: {{ $user->profile->bio ?? 'No bio available' }}</p>
11     <p>Website: <a href="{{ $user->profile->website ?? '#' }}>{{ $user->profile->website ?? 'No website' }}</a></p>
12
13     <h3>Posts</h3>
14     <ul class="list-group">
15         @foreach($user->posts as $post)
16             <li class="list-group-item">
17                 <a href="{{ route('posts.show', $post->id) }}>{{ $post->title }}</a>
18             </li>
19         @endforeach
20     </ul>
21     @endsection

```

- Membuat folder posts di resources/views.

■ Membuat Views index.blade.php.

```

1  @extends('layouts.app')
2
3  @section('title', 'Posts')
4
5  @section('content')
6      <h2>All Posts</h2>
7
8      <ul class="list-group">
9          @foreach($posts as $post)
10             <li class="list-group-item">
11                 <a href="{{ route('posts.show', $post->id) }}>{{ $post->title }} by {{ $post->user->name }}</a>
12             </li>
13         @endforeach
14     </ul>
15     @endsection
16

```

■ Membuat Views show.blade.php.

```

1  @extends('layouts.app')
2
3  @section('title', 'Post Details')
4
5  @section('content')
6      <h2>{{ $post->title }}</h2>
7      <p><strong>Author:</strong> {{ $post->user->name }}</p>
8      <p>{{ $post->content }}</p>
9
10     <h3>Tags</h3>
11     <ul class="list-group">
12         @foreach($post->tags as $tag)
13             <li class="list-group-item">{{ $tag->name }}</li>
14         @endforeach
15     </ul>
16
17     <a href="{{ route('posts.index') }}" class="btn btn-secondary mt-3">Back to Posts</a>
18     @endsection
19

```

- Menjalankan aplikasi dan Menunjukkan hasil dibrowser.

Link	User Name	Email
Ms. Romaine Ritchie	(cpfannerstill@example.org)	
Jada Hirthe MD	(jatterfield@example.net)	
Monserrate Schulist	(zwiza@example.com)	
Clint Rippin II	(emilie6@example.org)	
Mr. Llewellyn Toy Jr.	(nolan.jack@example.org)	
Audreanne Legros DVM	(phowell@example.org)	
Irene Flatley	(brandt16@example.net)	
Ms. Jazmin McLaughlin III	(greenfelder.anya@example.com)	
Eric Wintheiser	(cbechtelar@example.org)	
Prof. Jammie Bartoletti	(tcommerata@example.com)	
Dr. Mitte Haley	(hazle.boyer@example.net)	
Shad Metz	(remington74@example.net)	
Luisa Wintheiser	(zander30@example.com)	
Mrs. Stacy Rollson	(rcartwright@example.org)	
Miss Keena Flatley V	(shana.simons@example.net)	
Cecile Kutch DVM	(shanon.vandervort@example.org)	
Xzavier Corkery DDS	(isury33@example.org)	
Gregory Crist	(balistreri.alexia@example.net)	
Afton Towne	(travis19@example.org)	
Shanelle Johns	(lenny54@example.net)	

Link	Title	Author
Judul Post untuk user 1	by Ms. Romaine Ritchie	
Judul Post untuk user 2	by Jada Hirthe MD	
Judul Post untuk user 3	by Monserrate Schulist	
Judul Post untuk user 4	by Clint Rippin II	
Judul Post untuk user 5	by Mr. Llewellyn Toy Jr.	
Judul Post untuk user 6	by Audreanne Legros DVM	
Judul Post untuk user 7	by Irene Flatley	
Judul Post untuk user 8	by Ms. Jazmin McLaughlin III	
Judul Post untuk user 9	by Eric Wintheiser	
Judul Post untuk user 10	by Prof. Jammie Bartoletti	

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

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     public function up(): void
13     {
14         Schema::create('products', function (Blueprint $table) {
15             $table->id();
16             $table->string('name');
17             $table->decimal('price', total: 10, places: 2);
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists('products');
28     }
29 };


```

- 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

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use App\Models\Product;
8
9 class ProductSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         Product::factory()->count(count: 50)->create();
17     }
18 }


```

- Memperbarui file app/models/product.php

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7
8  class Product extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = ['name', 'price'];
13 }
```

- 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
```

```
1  <?php
2
3  namespace Database\Factories;
4
5  use Illuminate\Database\Eloquent\Factories\Factory;
6
7  no usages  Lepi
8  class ProductFactory extends Factory
9  {
10     Lepi
11     public function definition(): array
12     {
13         return [
14             'name' => fake()->word(),
15             'price' => fake()->randomFloat( nbMaxDecimals: 2, min: 10, max: 1000),
16         ];
17     }
18 }
```

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

```

1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Models\User;
6  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7  use Illuminate\Database\Seeder;
8
9  /**
10  * @Lepi *
11  */
12  class DatabaseSeeder extends Seeder
13  {
14      /**
15      * @Lepi
16      */
17      public function run(): void
18      {
19          /**
20          * @Lepi
21          */
22          User::factory()->create([
23              'name' => 'Test User',
24              'email' => 'test@example.com',
25          ]);
26      }
}

```

- 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

```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\Product;
7
8  /**
9  * @Lepi
10 */
11 class ProductController extends Controller
12 {
13     /**
14     * @Lepi
15     */
16     public function index()
17     {
18         $products = Product::orderBy('id', 'asc')->paginate(10);
19         return view('products.index', compact('var_name: products'));
20     }
}

```

- Mendefinisikan Route

```

1  <?php
2
3      use Illuminate\Support\Facades\Route;
4      use App\Http\Controllers\ProductController;
5
6      Route::get('uri: /', function () {
7          return view('welcome');
8      });
9
10     Route::get('uri: /products', [ProductController::class, 'index'])->name('name: products.index');

```

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

The screenshot shows a code editor with two panes. The left pane displays the project structure:

```

productpagination
  app
  bootstrap
  config
  database
  public
  resources
    css
    js
    views
      products
        index.blade.php (highlighted)
        welcome.blade.php
  routes
  storage
  tests
  vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  artisan
  composer.json

```

The right pane shows the content of the `index.blade.php` file:

```

1  <!doctype html>
2  <html>
3      <head>
4          <title>Paginated Products</title>
5          <script src="https://cdn.tailwindcss.com"></script>
6      </head>
7      <body class="max-w-4xl mx-auto py-10">
8          <h1 class="text-2xl font-bold mb-5">Daftar Product (Paginasi)</h1>
9
10         <table class="table-auto w-full border-collapse border-gray-300 mb-6">
11             <thead>
12                 <tr class="bg-gray-200">
13                     <th class="border px-4 py-2">#</th>
14                     <th class="border px-4 py-2">Nama</th>
15                     <th class="border px-4 py-2">Harga</th>
16                 </tr>
17             </thead>
18             <tbody>
19                 @foreach($products as $product)
20                     <tr>
21                         <td class="border px-4 py-2">{{ $product->id }}</td>
22                         <td class="border px-4 py-2">{{ $product->name }}</td>
23                         <td class="border px-4 py-2">{{ number_format($product->price, 2) }}</td>
24                     </tr>
25                 @endforeach
26             </tbody>
27         </table>
28
29         <div>
30             {{ $products->links() }}
31         </div>
32     </body>
33 </html>

```

- Menjalankan aplikasi dan Menunjukkan hasil dibrowser.

The screenshot shows a web browser window with the title 'Paginated Products'. The address bar indicates the URL is 127.0.0.1:8000/products. The main content is a table titled 'Daftar Product (Paginasi)' with columns '#', 'Nama', and 'Harga'. The table contains 10 rows of data. Below the table, a message says 'Showing 1 to 10 of 50 results' and there is a navigation bar with links from 1 to 5 and a next arrow.

#	Nama	Harga
1	laptop	500.00
2	quas	444.80
3	cumque	831.89
4	molestias	830.33
5	architecto	866.02
6	expedita	95.89
7	est	213.59
8	earum	730.17
9	in	630.92
10	et	864.56

Showing 1 to 10 of 50 results

< 1 2 3 4 5 >

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 pendefinisan 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 :
 - Laravel 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/>
-