Laporan Modul 6: Model dan Laravel Eloquent

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

Abstrak

Praktikum ini bertujuan untuk memahami dan mengimplementasikan konsep fundamental Model dan Eloquent ORM (Object-Relational Mapping) dalam framework Laravel 12, serta mengintegrasikannya dengan pola desain seperti Data Transfer Object (DTO), Plain Old Class Object (POCO), dan Repository Pattern. Implementasi dilakukan melalui tiga sesi latihan: 1. Mengikat data formulir menggunakan Model gaya POCO tanpa database, 2. Strukturisasi transfer data menggunakan DTO dan Service Layer, dan 3. Membangun aplikasi CRUD (Create, Read, Update, Delete) daftar tugas (Todo List) yang berinteraksi langsung dengan database MySQL menggunakan Laravel Eloquent. Hasil praktikum menunjukkan bahwa Model Eloquent secara efektif mengabstraksi operasi database, sementara penerapan pola DTO dan Repository Pattern berhasil meningkatkan modularitas, pemisahan concern, dan pemeliharaan kode aplikasi.

1. Dasar Teori

Pada praktikum ini dasar teorinya itu berpusat dpada arsitektur MVC (Model-View-Controller) di Laravel, dengan fokus pada komponen Model.

- **Model dan Eloquent ORM**
  - **Model** dalam Laravel merepresentasikan struktur data aplikasi dan umumnya terhubung dengan sebuah tabel di database. Model bertindak sebagai jembatan antara logika bisnis aplikasi dan data yang tersimpan.
  - **Eloquent ORM** adalah implementasi bawaan Laravel yang memudahkan interaksi database. Melalui Eloquent, operasi CRUD dapat dilakukan menggunakan sintaks PHP berorientasi objek yang intuitif, daripada menulis raw SQL.
- **POCO (Plain Old Class Object)**
  - **POCO** adalah kelas PHP sederhana yang digunakan untuk menyimpan data tanpa perilaku atau dependensi framework yang kompleks. Dalam praktikum ini, POCO digunakan untuk meniru perilaku Model guna mengikat dan menampilkan data formulir secara sederhana, memisahkan logika data dari dependensi database.
- **Data Transfer Object (DTO)**
  - **DTO** adalah objek yang digunakan khusus untuk mentransfer data antar lapisan aplikasi (misalnya, dari Controller ke Service Layer). Tujuannya adalah untuk memastikan data masuk dalam format yang terstruktur, memvalidasi dan memisahkan data mentah (request) dari logika bisnis inti, sehingga menghasilkan kode yang lebih rapi dan mudah diuji.

- **Repository Pattern**
  - **Repository Pattern** merupakan pola desain yang mengabstraksi logika akses data. Pola ini memisahkan lapisan logika bisnis dari lapisan persistensi (database). Dengan menggunakan interface dan implementasi repository, aplikasi dapat dengan mudah beralih jenis database tanpa memengaruhi logika bisnis di Controller atau Service Layer.

---

2. Langkah-Langkah Praktikum

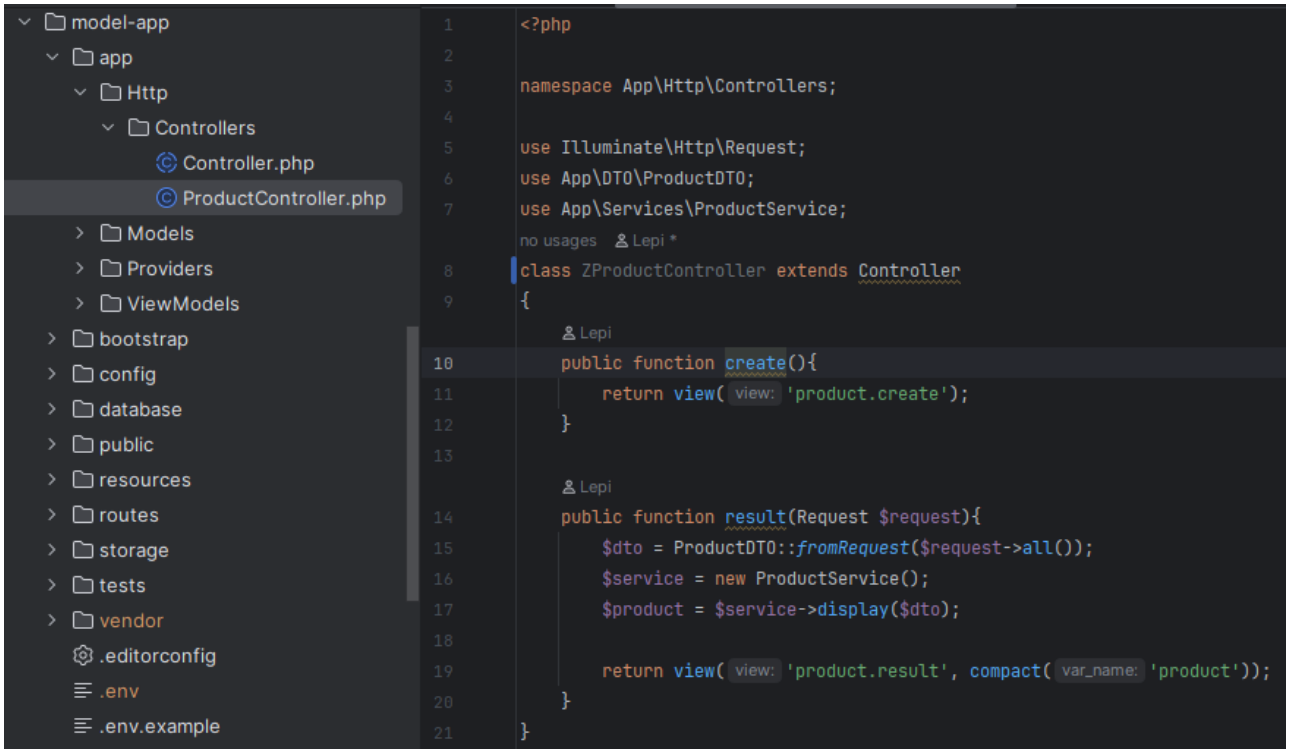
Tuliskan langkah-langkah yang sudah dilakukan, sertakan potongan kode dan screenshot hasil.

2.1 Praktikum 1 – Menggunakan Model untuk Binding Form dan Display

- Membuat Model Data Sederhana (POCO)

- Membuat Controller ProductController

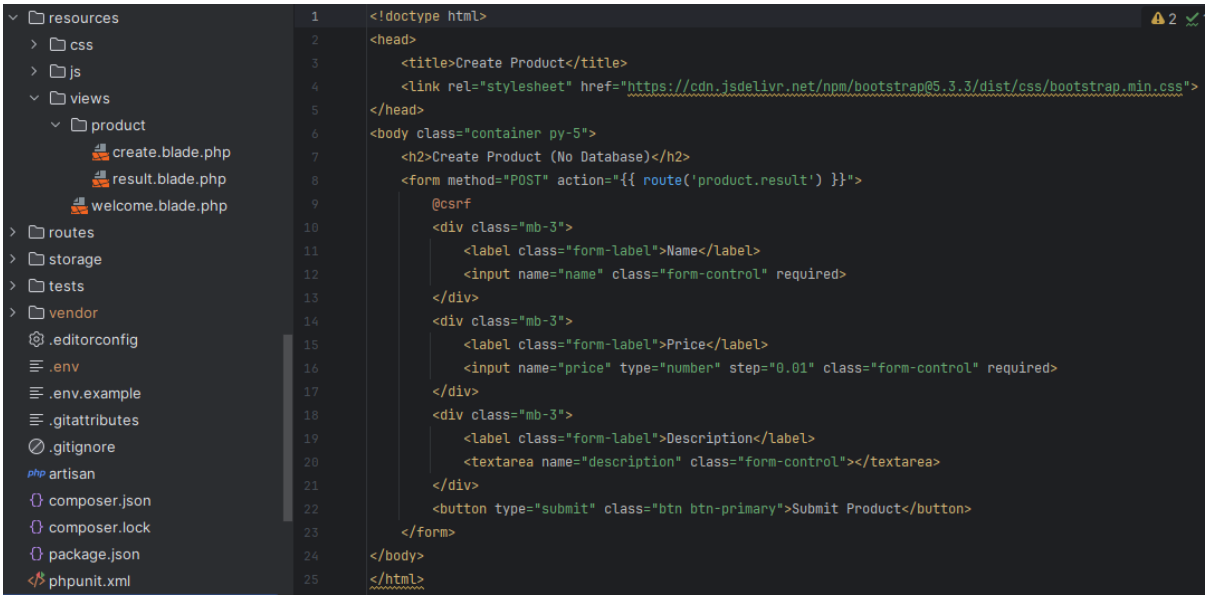
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/model-app (main)
$ php artisan make:controller ProductController
```



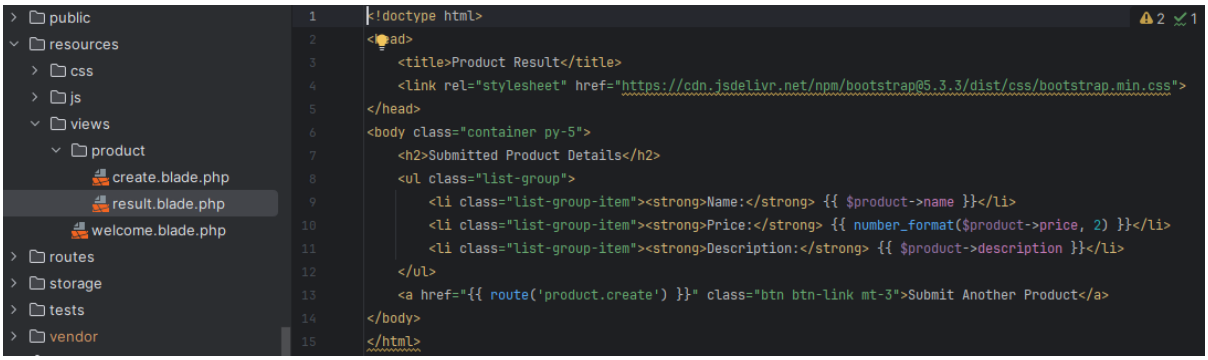
- Mendefinisikan Rute

```php
<?php

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

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

Route::get( uri: '/product/create', [ProductController::class, 'create'])->name( name: 'product.create');
Route::post( uri: '/product/result', [ProductController::class, 'result'])->name( name: 'product.result');
```

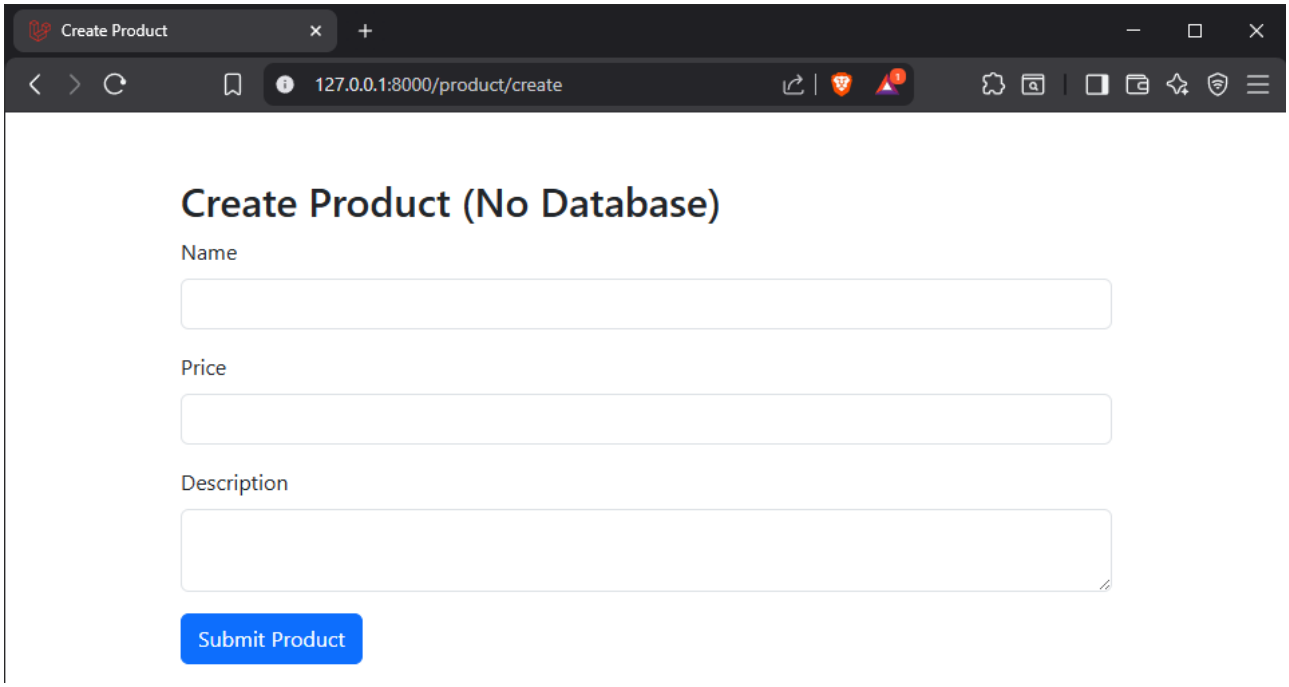
- Membuat Tampilan (Views) dengan Bootstrap.
  - Membuat Views create.blade.php.

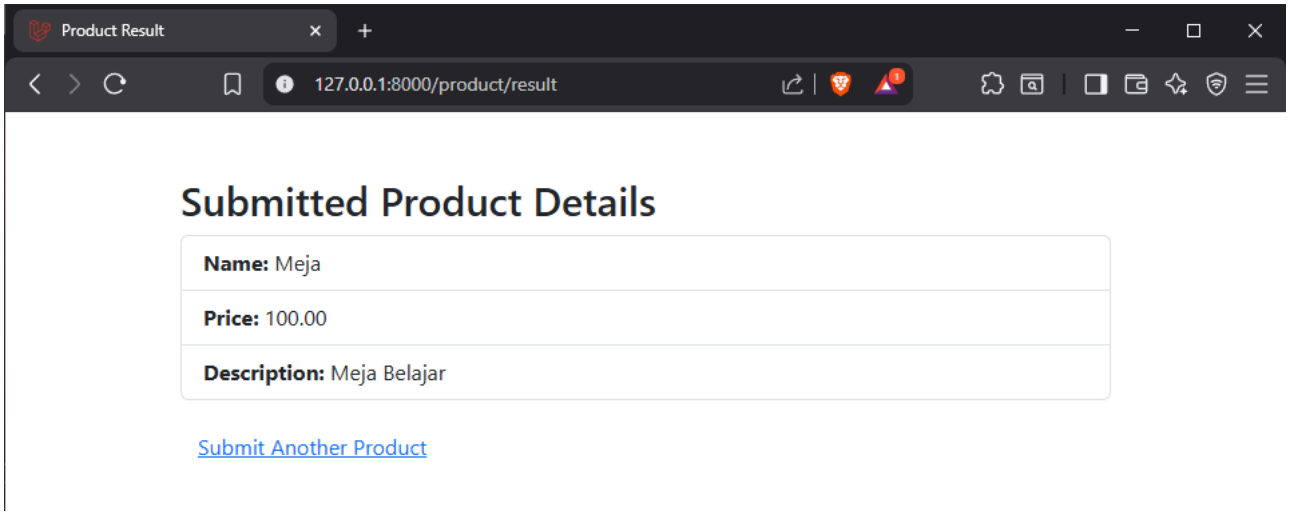


  - Membuat Views result.blade.php.



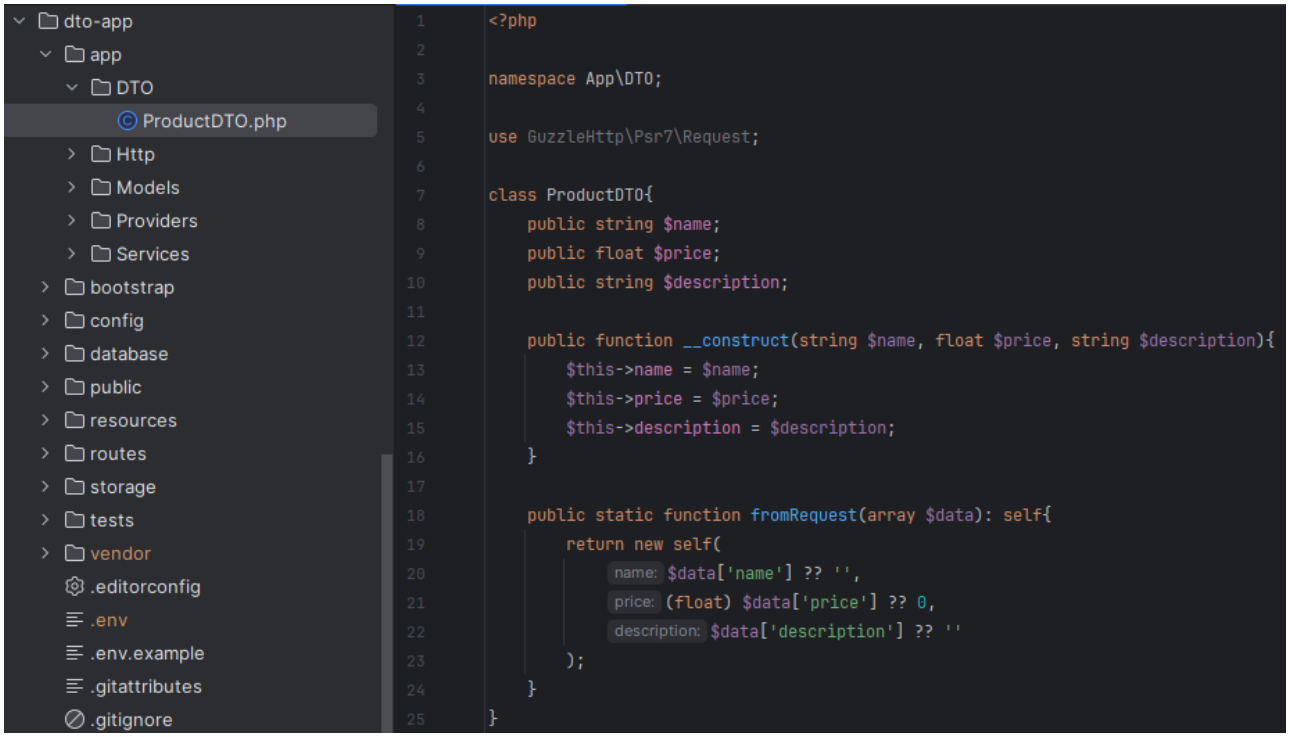
- Menjalankan aplikasi dan Menunjukkan hasil dibrowser.



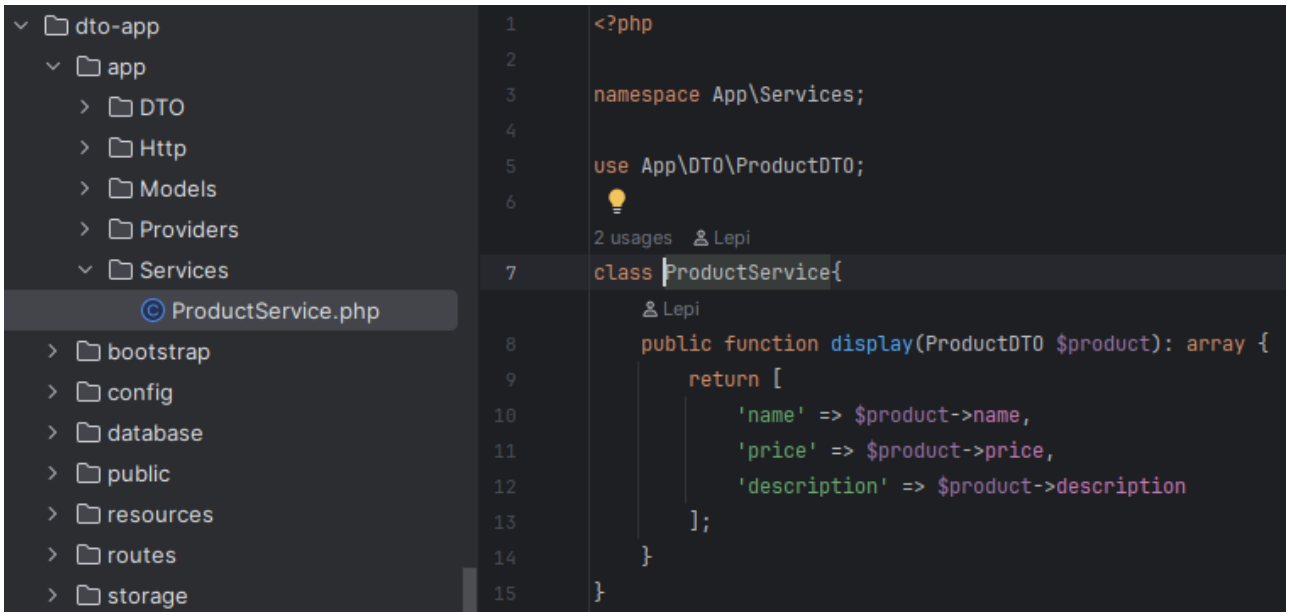


2.2 Praktikum 2 – Menggunakan Group Route

- Membuat Kelas DTO kemudian membuat file ProductDTO.php.

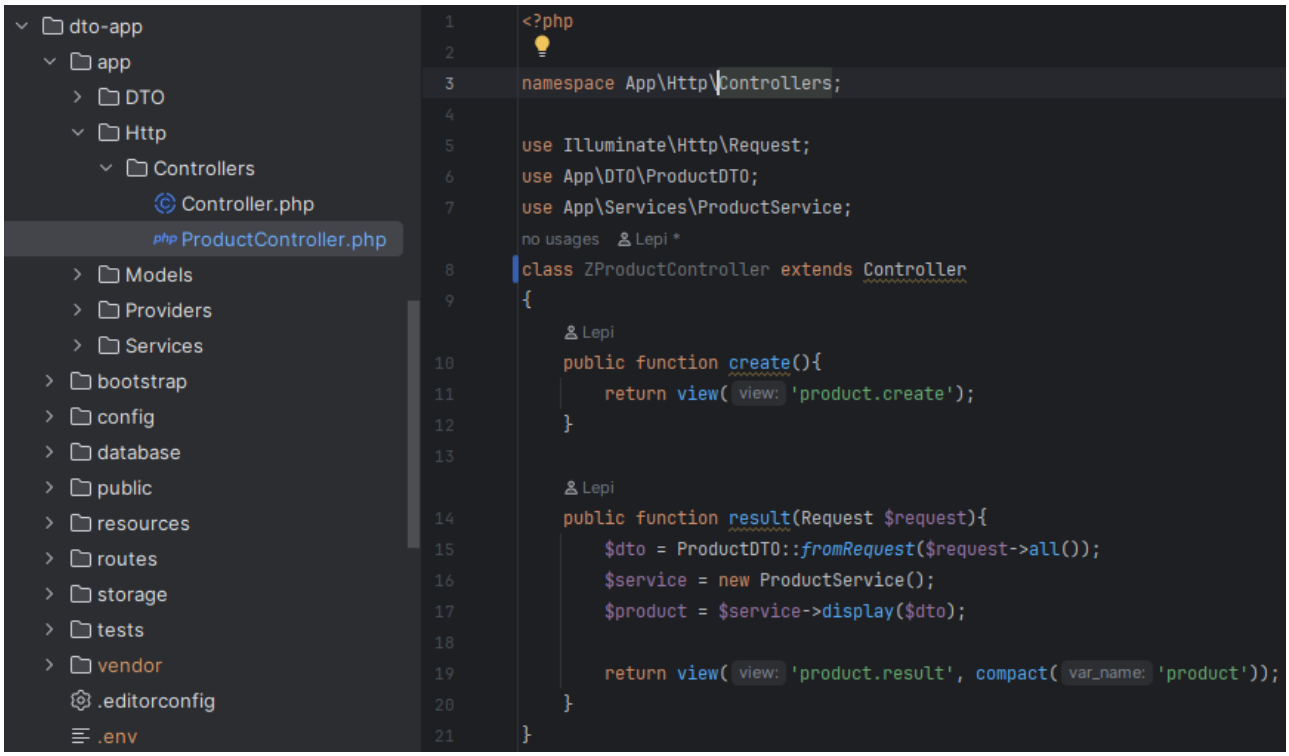


- Membuat Service Layer kemudian membuat file ProductService.php.



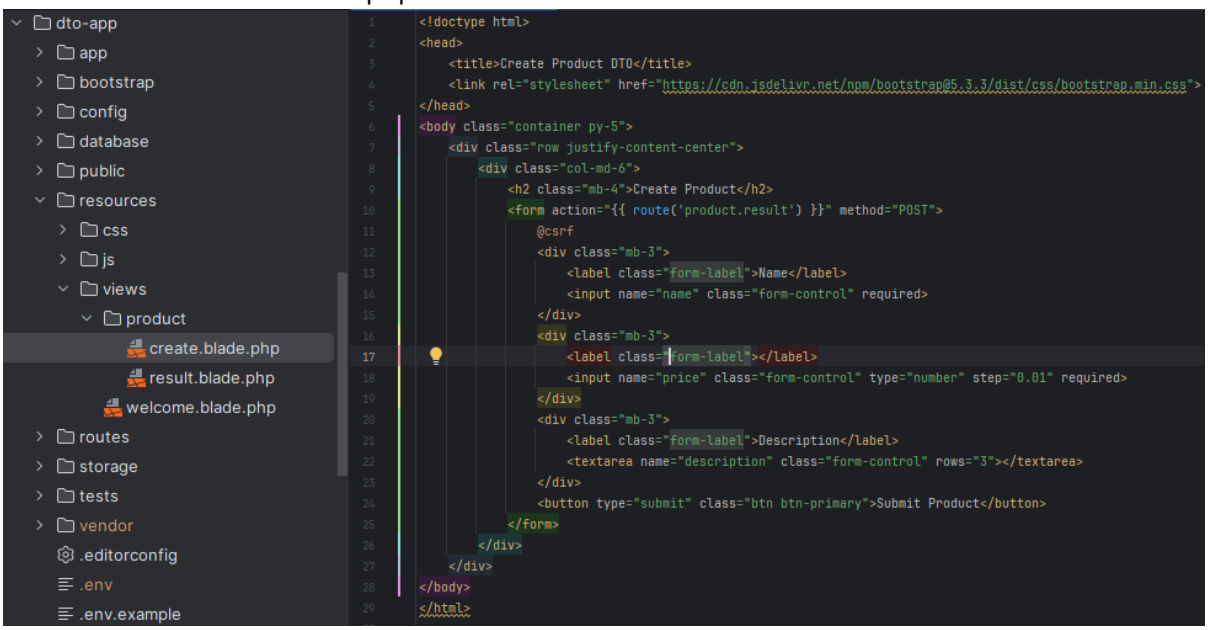
- Membuat Controller ProductController.

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

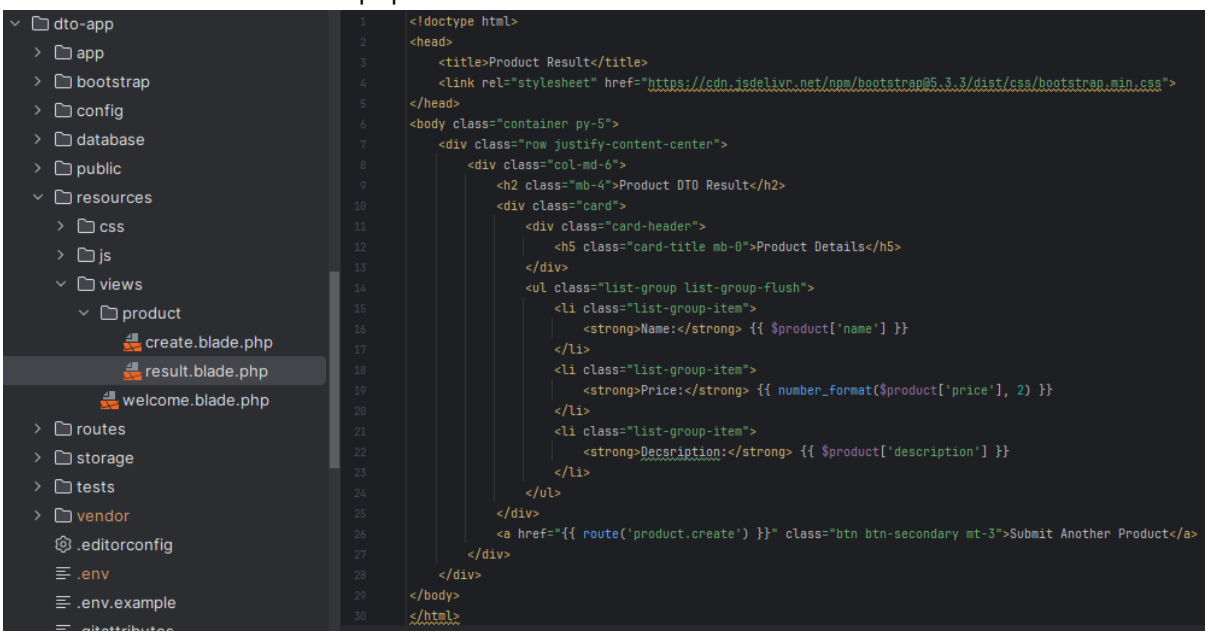


- Mendefinisikan Rute.

- Membuat Tampilan (Views) dengan Bootstrap.
  - Membuat Views create.blade.php.

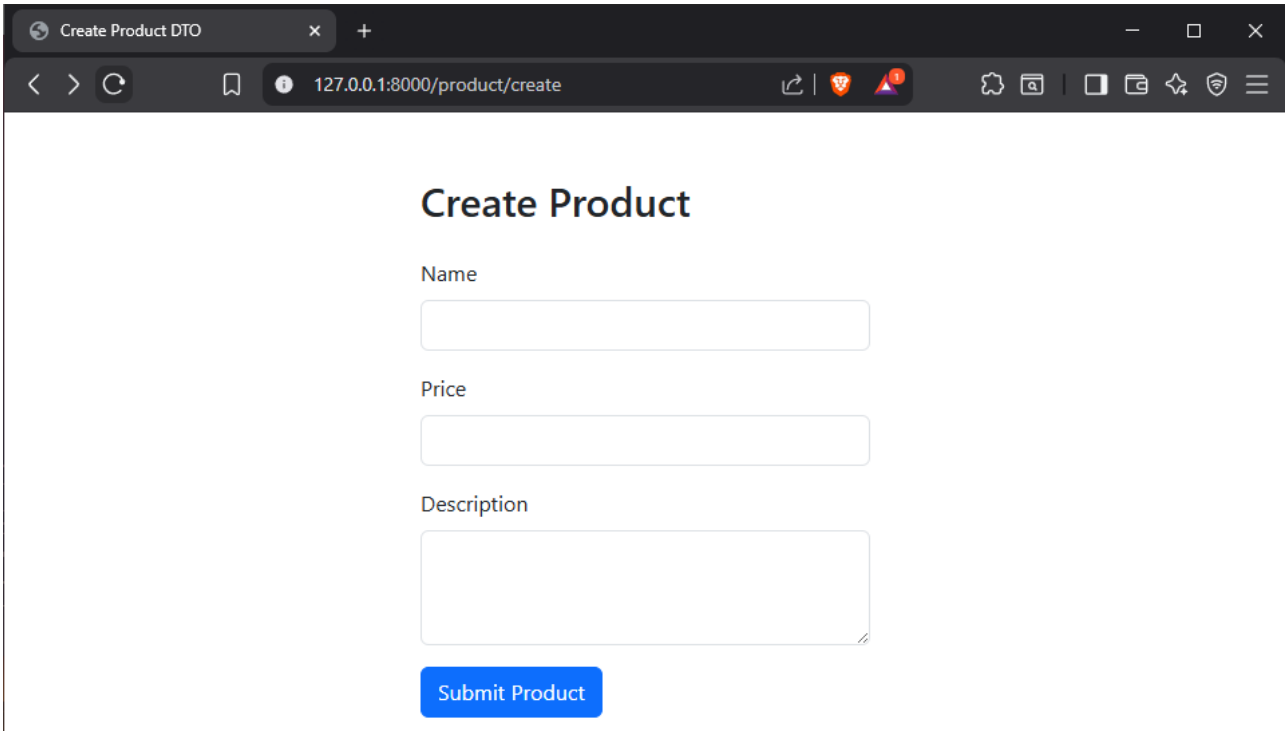


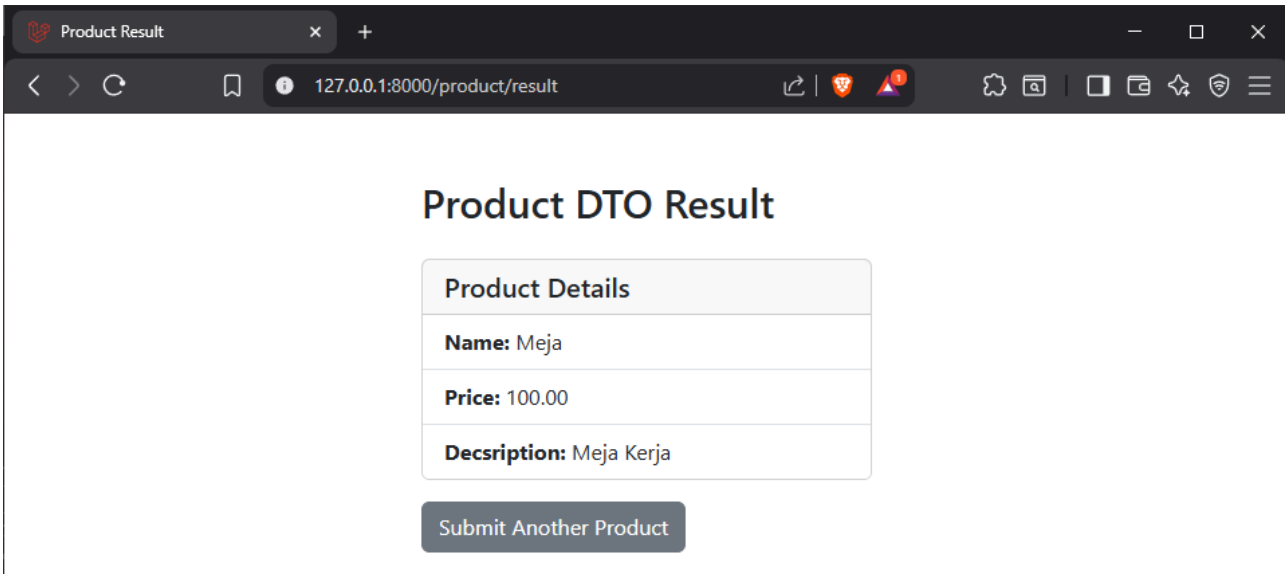
- Membuat Views result.blade.php.




```blade
<!doctype html>
<head>
    <title>Product Result</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">
</head>
<body class="container py-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h2 class="mb-4">Product DTO Result</h2>
            <div class="card">
                <div class="card-header">
                    <h5 class="card-title mb-0">Product Details</h5>
                </div>
                <ul class="list-group list-group-flush">
                    <li class="list-group-item">
                        <strong>Name:</strong> {{ $product['name'] }}
                    </li>
                    <li class="list-group-item">
                        <strong>Price:</strong> {{ number_format($product['price'], 2) }}
                    </li>
                    <li class="list-group-item">
                        <strong>Decsription:</strong> {{ $product['description'] }}
                    </li>
                </ul>
            </div>
            <a href="{{ route('product.create') }}" class="btn btn-secondary mt-3">Submit Another Product</a>
        </div>
    </div>
</body>
</html>
```

- Menjalankan aplikasi dan Menunjukkan hasil dibrowser.





2.3 Praktikum 3 – Pengelompokan Prefix dengan Namespace Rute di Laravel 12

- Menginstall dependency MySQL.

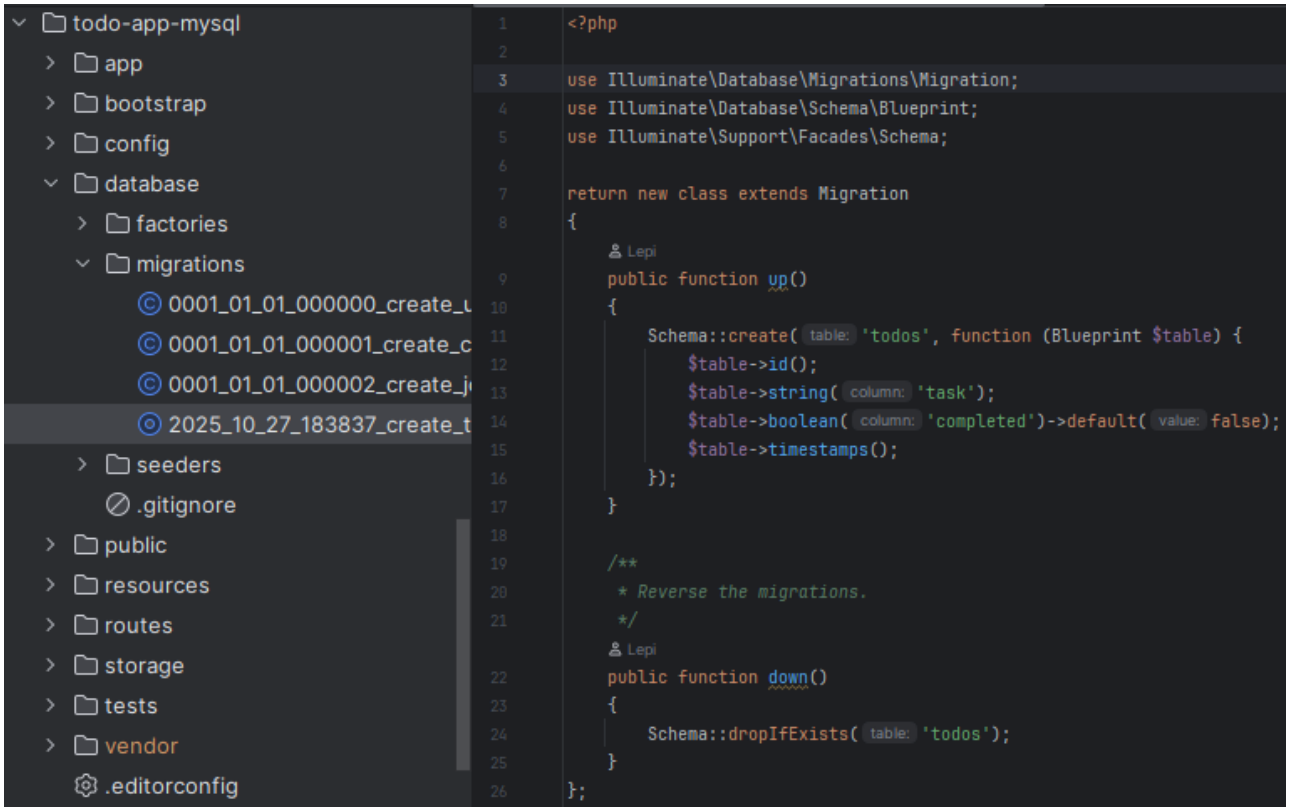
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/todo-app-mysql (main)
$ composer require doctrine/dbal
```

- Membersihkan config cache.

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/todo-app-mysql (main)
$ php artisan config:clear
```

- Membuat Migration untuk Tabel todos lalu jalankan migrasi (php artisan migrate).

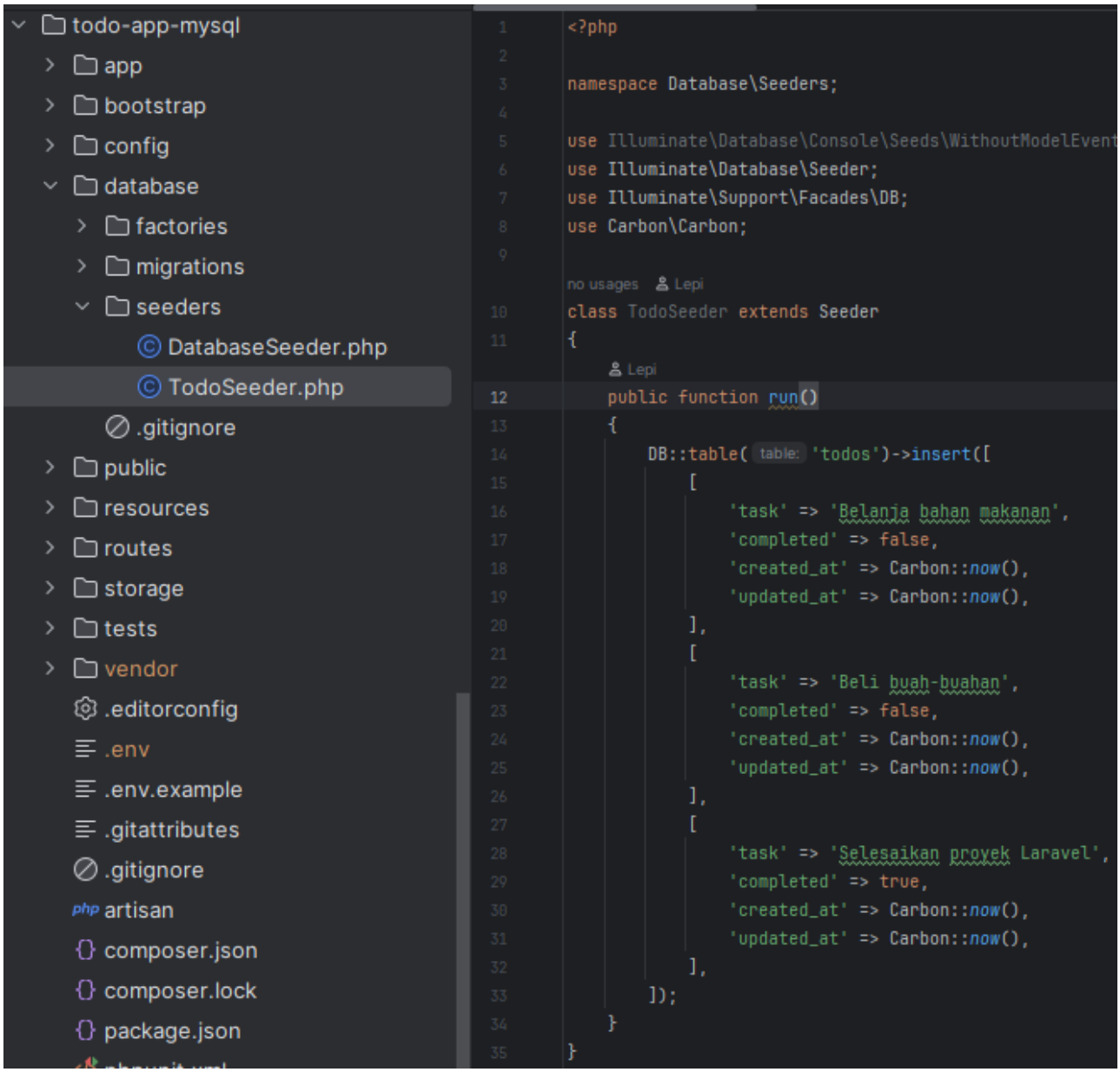
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/todo-app-mysql (main)
$ php artisan make:migration create_todos_table
```




```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/todo-app-mysql (main)
$ php artisan migrate
```

- Membuat Seeder untuk Data Dummy.

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

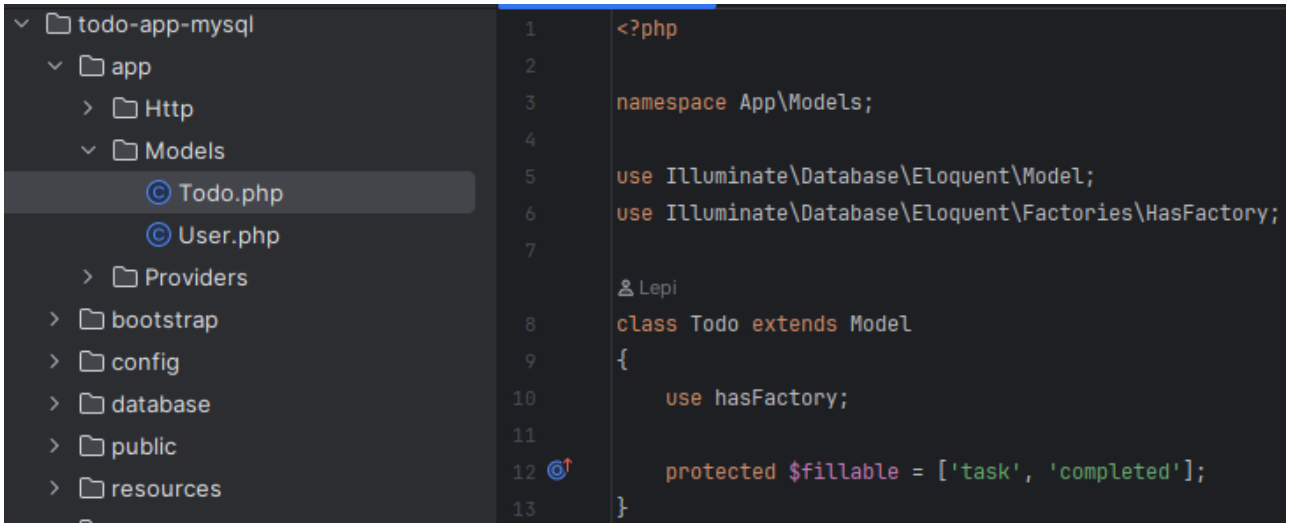


- Menjalankan seeder untuk mnegisi database.

```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/todo-app-mysql (main)
$ php artisan db:seed --class=TodoSeeder
```

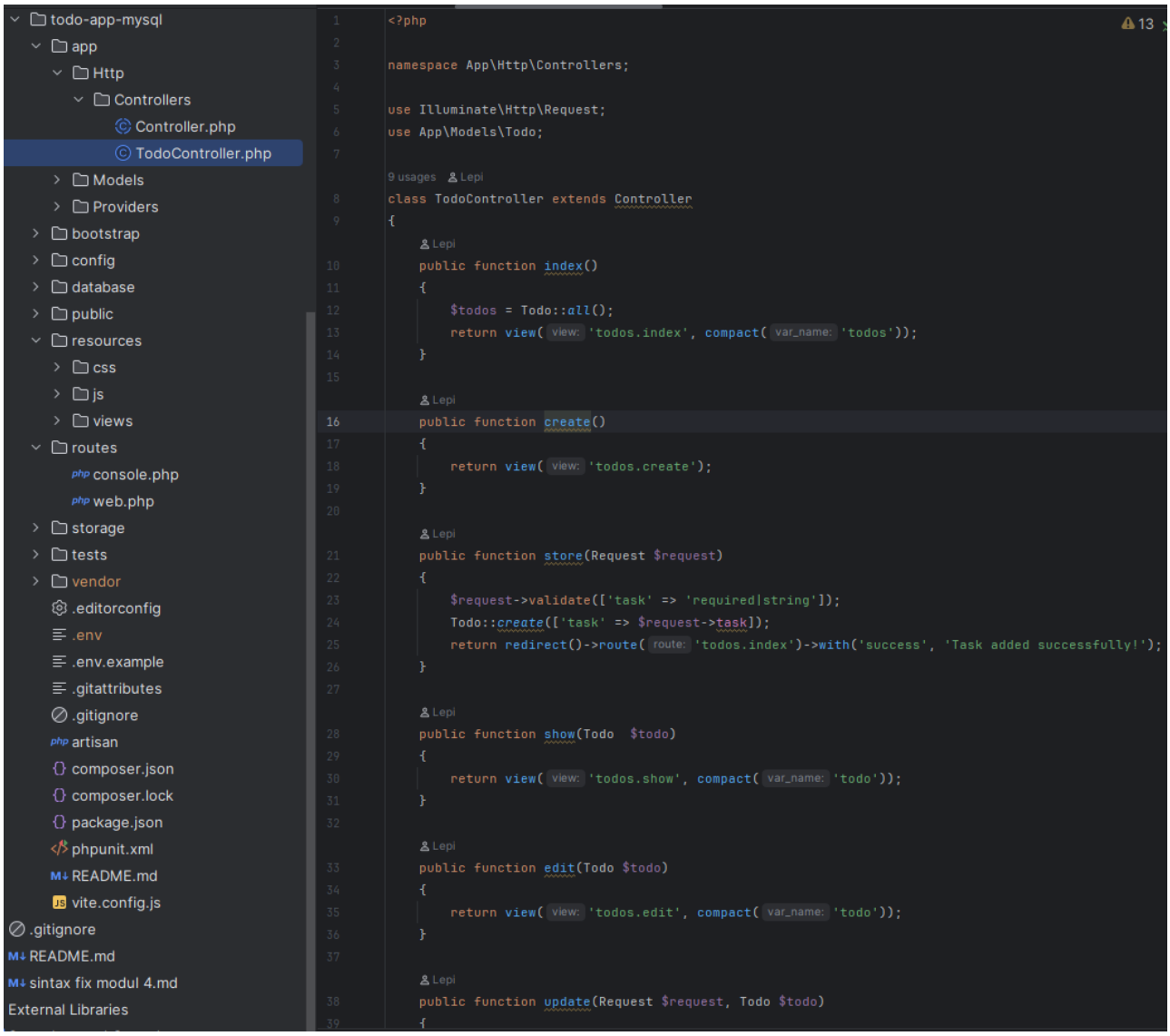
- Membuat Model Todo.

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



- Membuat controller TodoController untuk Operasi CRUD.

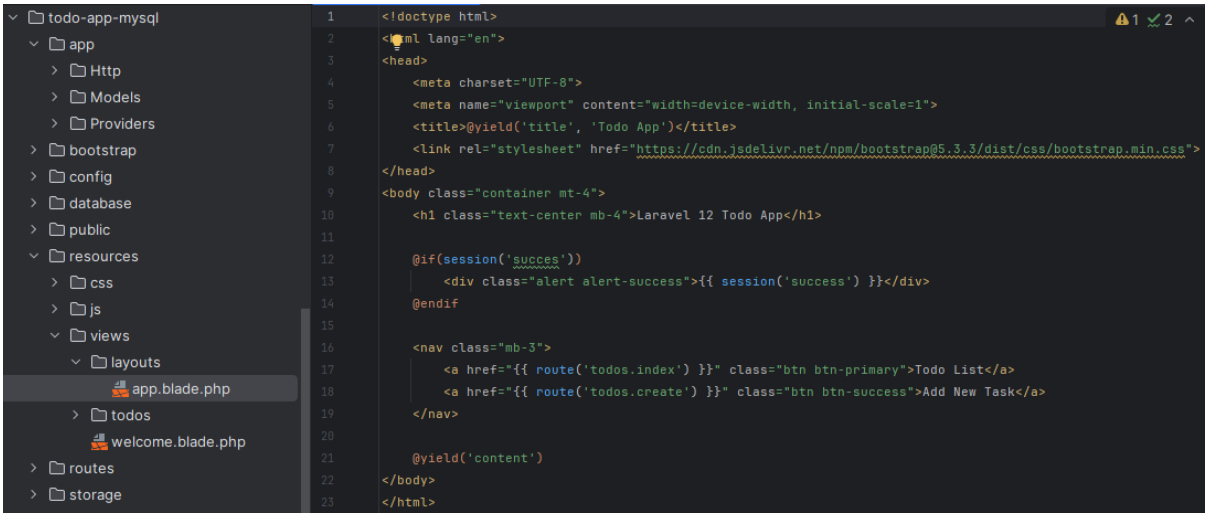
```
Administrator@WIN-C32CTMDNOHK MINGW64 /e/Kampus/Semester 3/Workshop Web Lanjut/web-lanjut-2024573010077/project/todo-app-mysql (main)
$ php artisan make:controller TodoController
```



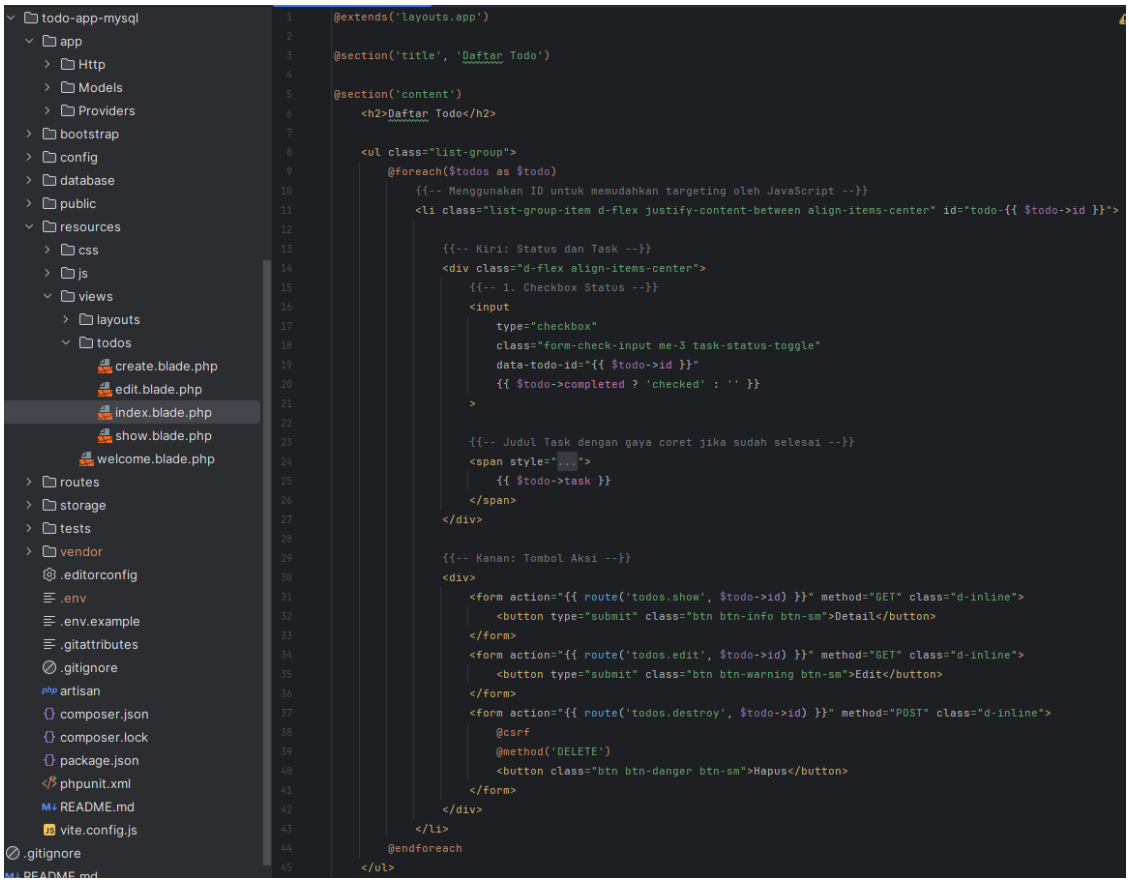
- Mendefinisikan Rute Web.

- Membuat Tampilan Blade dengan Bootstrap.

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



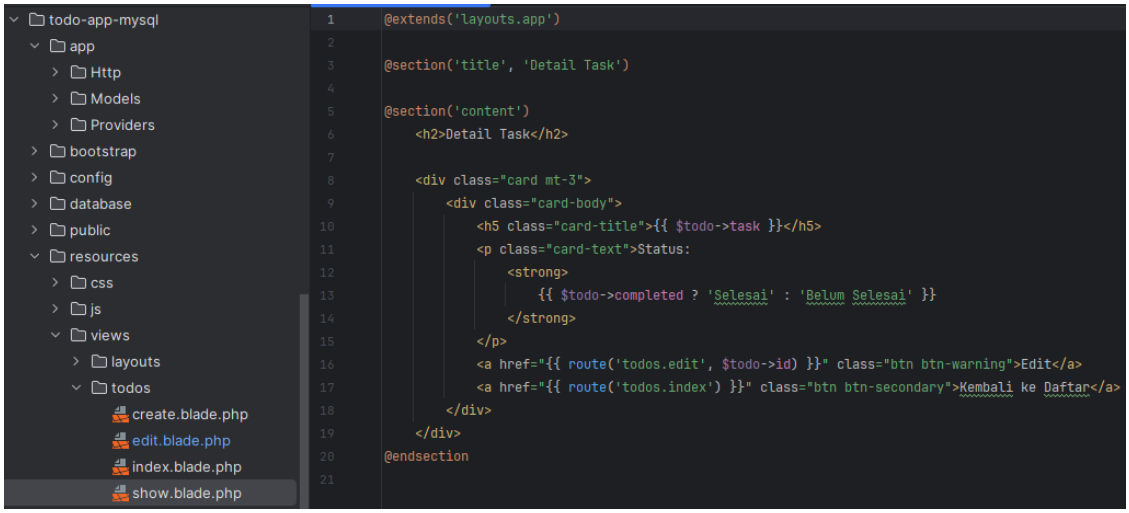
- Membuat folder todos di resources/views.
  - Membuat Views index.blade.php.



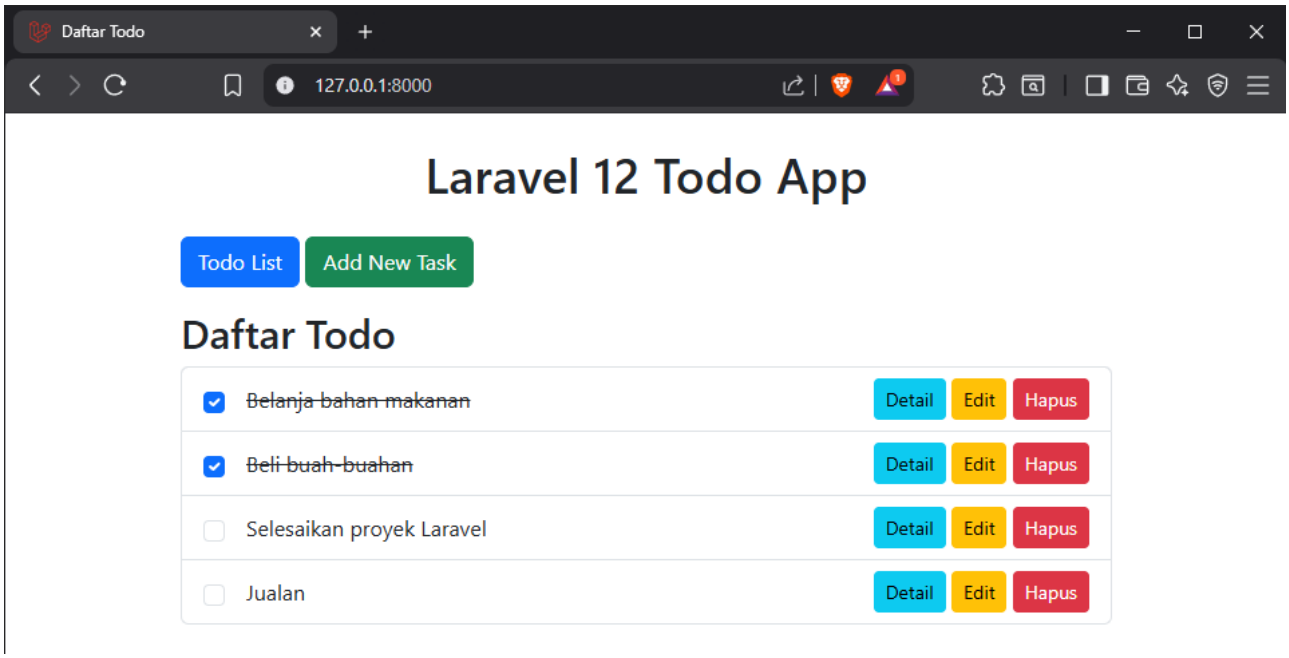
  - Membuat Views create.blade.php.

- Membuat Views edit.blade.php.

- Membuat Views show.blade.php.



- 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 serangkaian aplikasi fungsional yang mendemonstrasikan bagaimana

mengimplementasikan dan mengintegrasikan Model Eloquent ORM dengan pola desain arsitektural di Laravel.

  - **Implementasi Model POCO (Praktikum 1):** Berhasil membuat dan menggunakan kelas ProductViewModel (gaya POCO) untuk binding data yang dikirim melalui formulir tanpa koneksi database. Ini menunjukkan bagaimana Model dapat digunakan untuk strukturisasi data bahkan tanpa Eloquent.
  - **Integrasi DTO dan Service Layer (Praktikum 2):** Berhasil memisahkan data input dari logika pemrosesan dengan mengimplementasikan ProductDTO. Data dari Request diubah menjadi DTO, kemudian DTO diteruskan ke ProductService, menunjukkan pemisahan concern yang bersih antara Controller dan lapisan bisnis.
  - **Aplikasi CRUD Penuh dengan Eloquent (Praktikum 3):** Hasil utama adalah terciptanya aplikasi Todo List lengkap. Aplikasi ini berhasil melakukan operasi Create, Read, Update, dan Delete data pada database MySQL, dengan semua interaksi diatur secara efisien oleh Model Todo dan fitur Migrations serta Seeder Laravel.

- Bagaimana Validasi Input Bekerja di Laravel?
  Berdasarkan Praktikum 3 (Pembuatan Todo List), validasi input di Laravel bekerja sebagai mekanisme middleware yang memastikan integritas dan keamanan data sebelum diproses dan disimpan oleh Model.
  - **Mekanisme Kontrol di Controller:** Validasi diaktifkan pada method store dan update di TodoController menggunakan $request->validate(). Misalnya, untuk tugas baru: $request->validate(['task' => 'required|string']);.
  - **Pengecekan Aturan:** Aturan (required dan string) digunakan untuk memaksa pengguna memberikan data yang harus ada dan sesuai tipe data.
  - **Respons Otomatis Laravel:** Jika validasi gagal, Laravel secara otomatis mengalihkan (redirect) pengguna kembali ke formulir sebelumnya, sambil membawa pesan error dan input lama (old input) melalui session. Ini memungkinkan View menampilkan pesan kesalahan tanpa perlu penanganan manual di Controller.
- Apa peran Masing-Masing Komponen (Route, Controller, View) dalam Program yang Dibuat?
  Ketiga komponen ini bekerja bersama dalam siklus request-response MVC, di mana Model Todo menjadi komponen inti penanganan data.
  - **Route (Gerbang Permintaan):**
    - **Peran:** Bertugas memetakan URL ke action Controller yang spesifik, mendefinisikan endpoint aplikasi CRUD.
    - **Aksi:** Menggunakan Resource Routing (atau rute terpisah) untuk mengarahkan /todos/{todo} ke method seperti show, edit, update, atau destroy di Controller.
  - **Controller (Koordinator Logika):**
    - **Peran:** Menerima permintaan, melakukan validasi, memanggil Model untuk manipulasi data (logika bisnis), dan mengembalikan View yang sesuai.

        - Aksi: Di method index, Controller memanggil Todo::all() dan meneruskan hasilnya ke View. Di method store, Controller memanggil Todo::create($data) setelah validasi. Controller adalah titik interaksi utama dengan Model.

    - **View (Antarmuka Presentasi):**

        - **Peran:** Bertanggung jawab penuh untuk merender struktur HTML dan menampilkan data yang dikirimkan oleh Controller.
        - **Aksi:** Menggunakan template Blade (misalnya todos/index.blade.php) untuk menampilkan daftar tugas ($todos) yang diambil dari Model, termasuk menampilkan formulir, detail, dan pesan sukses/error.

---

4. Kesimpulan

Secara keseluruhan, praktikum mengenai Model dan Laravel Eloquent ini berhasil mengilustrasikan penerapan prinsip-prinsip arsitektural inti dalam pengembangan aplikasi berbasis Laravel. Implementasi Model Eloquent ORM terbukti menjadi solusi yang efisien dan elegan untuk mengabstraksi semua operasi persistence data (CRUD) terhadap database MySQL, yang menghasilkan kode Controller yang lebih bersih dan fokus pada logika bisnis. Lebih lanjut, melalui implementasi POCO, DTO, dan Service Layer, praktikum ini menunjukkan pentingnya separation of concern. DTO memastikan transfer data yang terstruktur antara lapisan aplikasi, sementara POCO memberikan cara yang fleksibel untuk memodelkan data non-database. Kesimpulannya, penguasaan terhadap Model Eloquent dan integrasinya dengan pola desain modern adalah fundamental untuk membangun aplikasi Laravel yang tidak hanya fungsional tetapi juga terstruktur dengan baik, mudah dikelola, dan scalable di masa depan.

---

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/
