Laporan Modul 6: Model dan Laravel Eloquent

Mata Kuliah: Workshop Web Lanjut
Nama: Hayzar Muhaiyar **NIM:** 2024573010100 **Kelas:** TI-2C

Abstrak

Pada praktikum ini mahasiswa mempelajari konsep Model dalam framework Laravel serta penerapan Laravel Eloquent ORM untuk mengelola data secara efisien di dalam aplikasi web. Praktikum mencakup tiga bagian utama, yaitu penanganan request dan response pada view menggunakan model sederhana, validasi kustom dengan pesan error, serta implementasi multi-step form dengan penyimpanan data menggunakan session. Melalui kegiatan ini, mahasiswa memahami alur data dari form ke controller, model, hingga view dalam arsitektur MVC Laravel.

---

1. Dasar Teori

Laravel menggunakan arsitektur MVC (Model-View-Controller) untuk memisahkan logika bisnis, tampilan, dan pengelolaan data. Beberapa konsep penting dalam modul ini antara lain:

- Model Model adalah representasi data yang digunakan untuk berinteraksi dengan basis data atau untuk mengelola data sementara dari form. Dalam Laravel, model dapat berupa:
- POCO (Plain Old Class Object): Model sederhana tanpa database.
- Eloquent Model: Model yang terhubung dengan tabel di database.
- Eloquent ORM (Object Relational Mapping) Eloquent menyediakan cara mudah untuk berinteraksi dengan database menggunakan sintaks PHP berorientasi objek.

Contohnya: $users = User::all(); $todo = Todo::find(1); $todo->delete();

- Request dan Response Laravel menyediakan Request untuk menangkap input dari user dan Response untuk mengirimkan data atau tampilan kembali ke browser.
- Validation Validasi berfungsi untuk memastikan data yang dikirimkan pengguna sesuai dengan aturan tertentu sebelum diproses atau disimpan.
- Session Session digunakan untuk menyimpan data sementara antar halaman, seperti pada multi-step form.

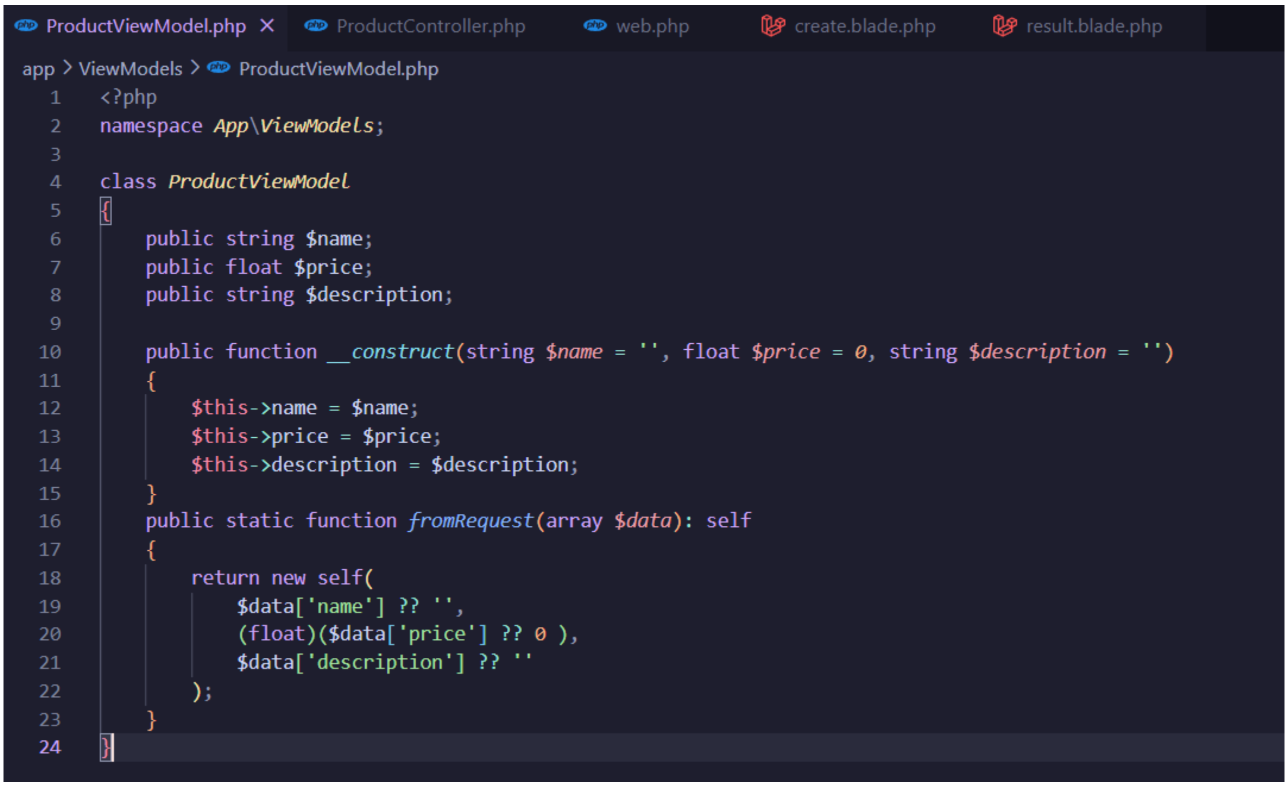
---

2. Langkah-Langkah Praktikum

##Praktikum 1: Menggunakan Model Untuk Binding Form Dan Displayz

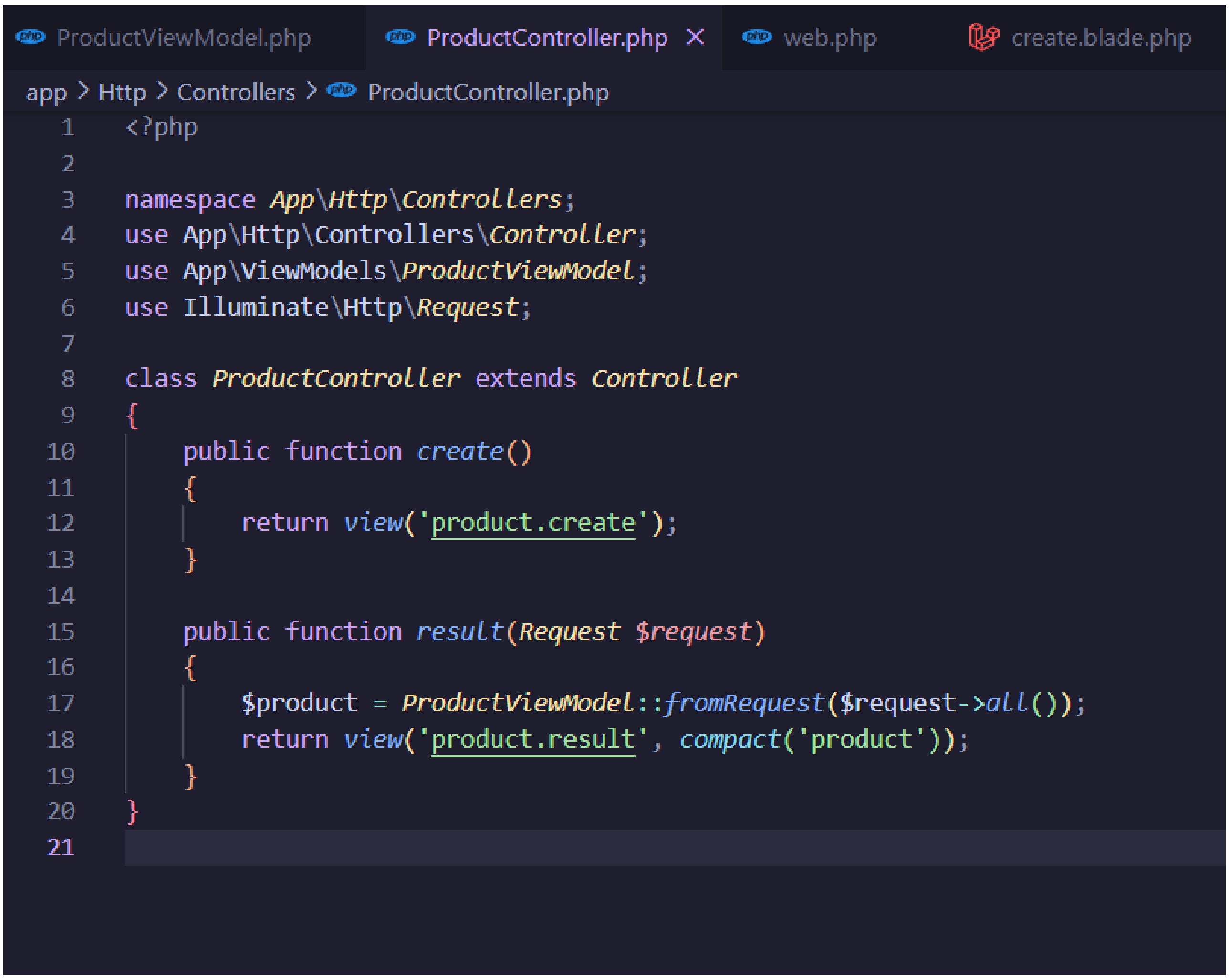
- Langkah 1: Buat dan Buka Proyek laravel laravel new model-app cd model-app code .

- Langkah 2: Membuat Model Data Sederhana (POCO) Buat folder ViewModels di dalam direktori app untuk menyimpan kelas model kit: mkdir app/ViewModels Selanjutnya Buat ProductViewModel.php Di dalam Direktori app/viewModels. kemudian isi dengan code berikut



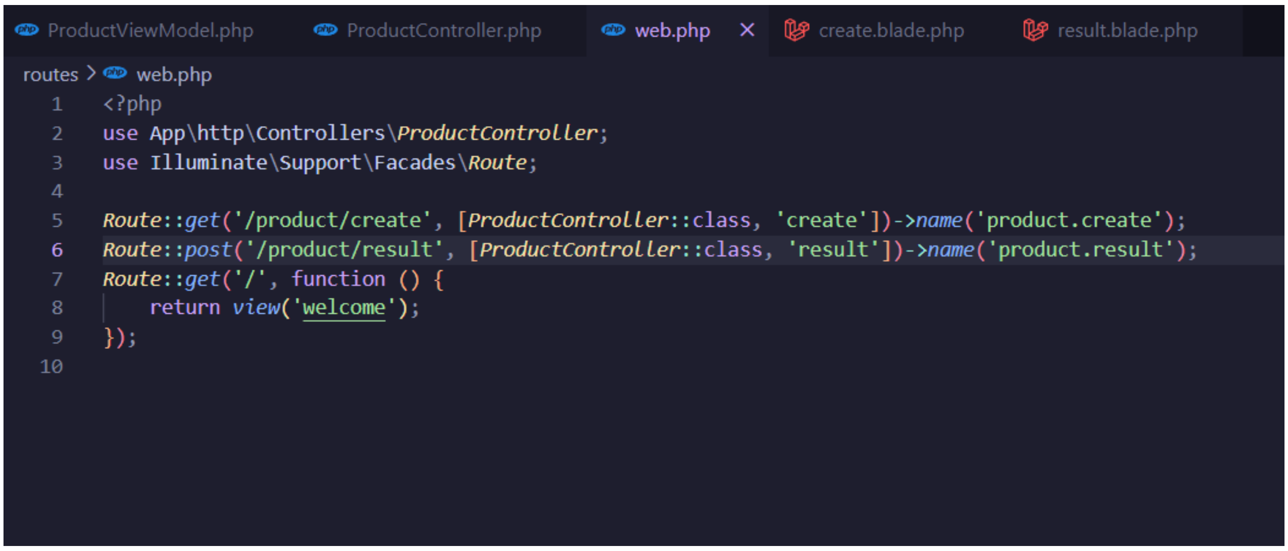
```php
<?php
namespace App\ViewModels;

class ProductViewModel
{
    public string $name;
    public float $price;
    public string $description;

    public function __construct(string $name = '', float $price = 0, string $description = '')
    {
        $this->name = $name;
        $this->price = $price;
        $this->description = $description;
    }
    public static function fromRequest(array $data): self
    {
        return new self(
            $data['name'] ?? '',
            (float)($data['price'] ?? 0 ),
            $data['description'] ?? ''
        );
    }
}
```

- Langkah 3: Buat Controller php artisan make:controller ProductController Kemudian Edit Controller Tersebut:



```php
<?php

namespace App\Http\Controllers;
use App\Http\Controllers\Controller;
use App\ViewModels\ProductViewModel;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function create()
    {
        return view('product.create');
    }

    public function result(Request $request)
    {
        $product = ProductViewModel::fromRequest($request->all());
        return view('product.result', compact('product'));
    }
}

```

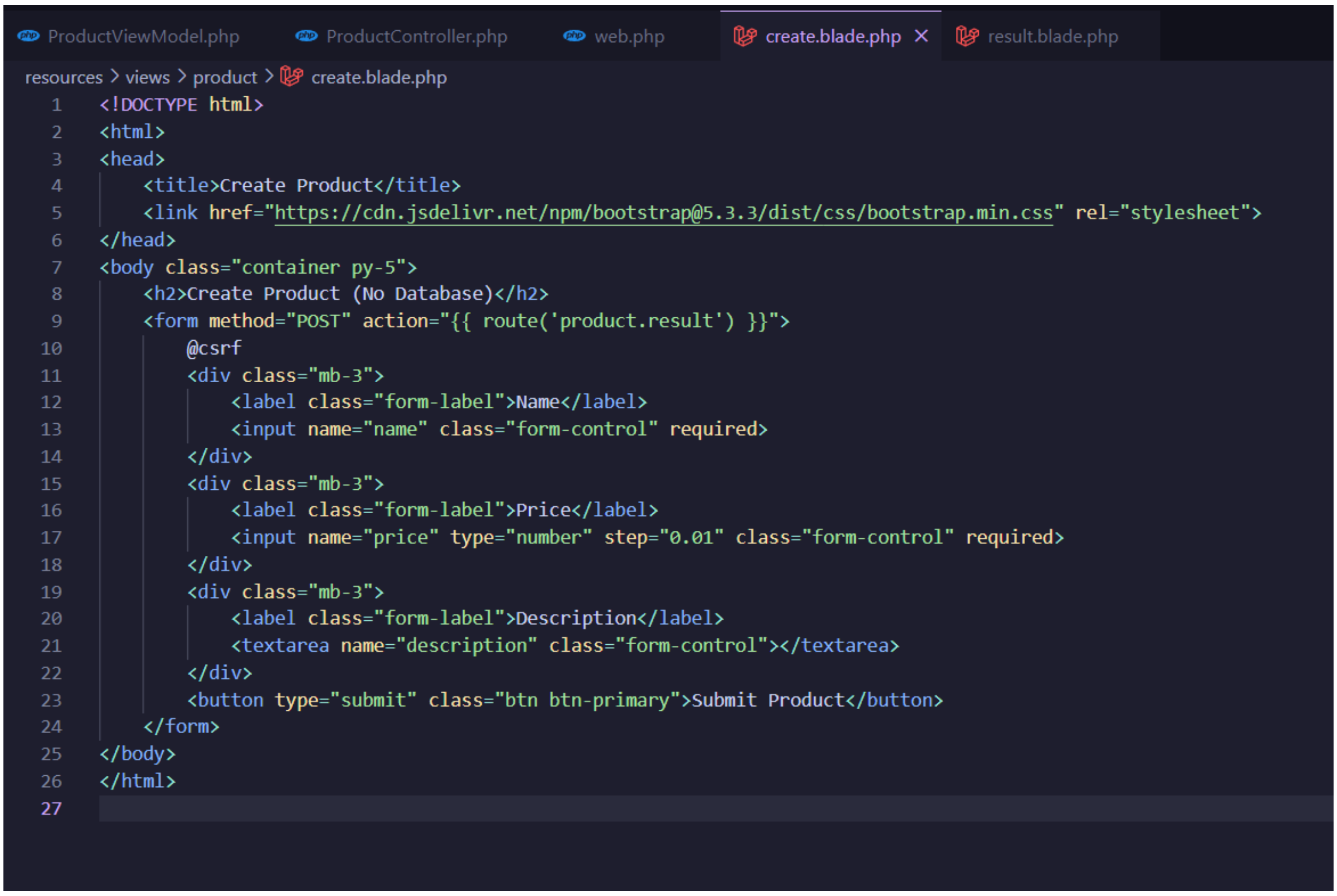
- Langkah 4: Definisikan Rute Edit Routes/Web.php Isi Dengan Code Berikut



```php
<?php
use App\http\Controllers\ProductController;
use Illuminate\Support\Facades\Route;

Route::get('/product/create', [ProductController::class, 'create'])->name('product.create');
Route::post('/product/result', [ProductController::class, 'result'])->name('product.result');
Route::get('/', function () {
    return view('welcome');
});

```

- Langkah 5: Buat Tampilan (Views) Dengan Boostrap Buat direktori product di dalam resources/views: mkdir resources/views/product Kemudian buat dua file: create.blade.php dan result.blade.php.

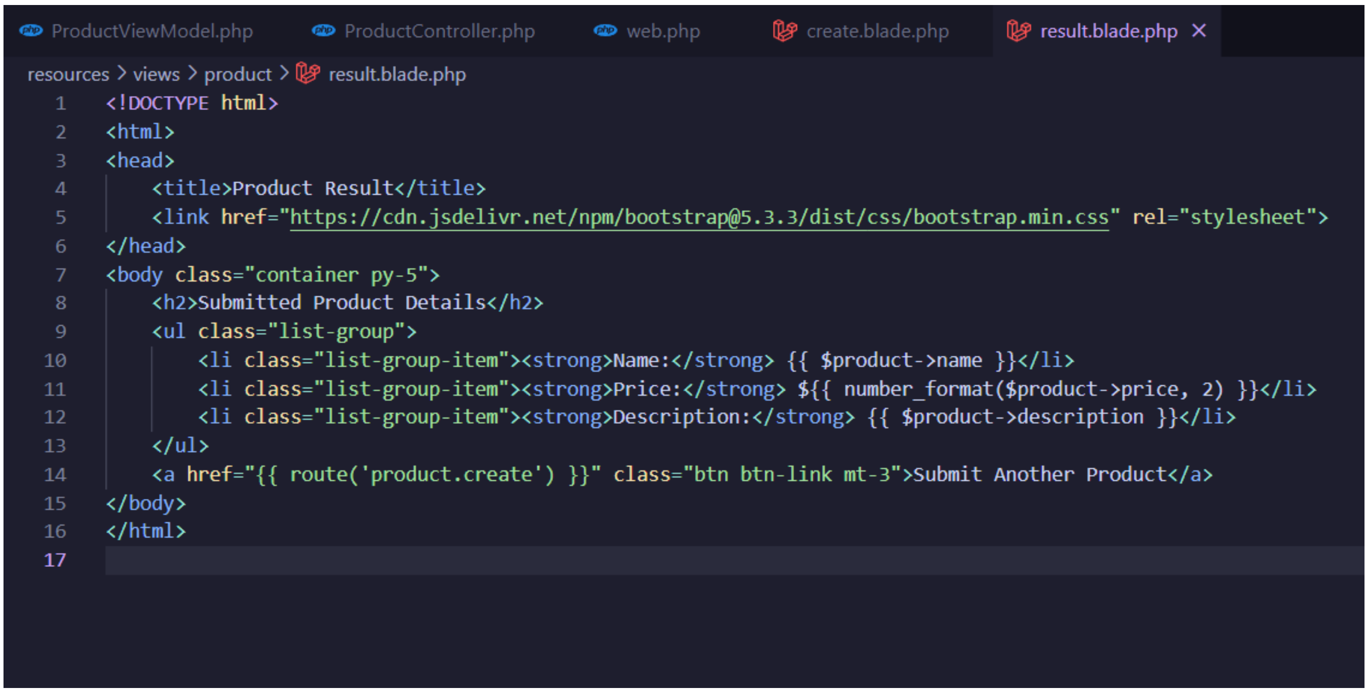
Berikut adalah konten dari masing-masing file:



ProductViewModel.php ProductController.php web.php create.blade.php result.blade.php

resources > views > product > create.blade.php

```html
<!DOCTYPE html>
<html>
<head>
    <title>Create Product</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container py-5">
    <h2>Create Product (No Database)</h2>
    <form method="POST" action="{{ route('product.result') }}">
        @csrf
        <div class="mb-3">
            <label class="form-label">Name</label>
            <input name="name" class="form-control" required>
        </div>
        <div class="mb-3">
            <label class="form-label">Price</label>
            <input name="price" type="number" step="0.01" class="form-control" required>
        </div>
        <div class="mb-3">
            <label class="form-label">Description</label>
            <textarea name="description" class="form-control"></textarea>
        </div>
        <button type="submit" class="btn btn-primary">Submit Product</button>
    </form>
</body>
</html>
```

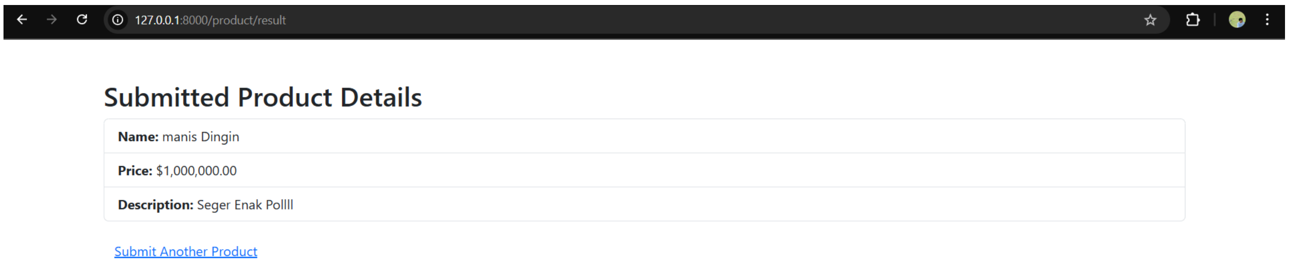


ProductViewModel.php ProductController.php web.php create.blade.php result.blade.php

resources > views > product > result.blade.php

```html
<!DOCTYPE html>
<html>
<head>
    <title>Product Result</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container py-5">
    <h2>Submitted Product Details</h2>
    <ul class="list-group">
        <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>Description:</strong> {{ $product->description }}</li>
    </ul>
    <a href="{{ route('product.create') }}" class="btn btn-link mt-3">Submit Another Product</a>
</body>
</html>
```

- Jalankan Aplikasi Dengan cara : Jalankan php artisan serve di terminal, Kunjungi http://localhost:8000/product/create Isi formulir dan kirim. Anda akan melihat hasilnya ditampilkan di halaman baru tanpa menyimpan ke database.

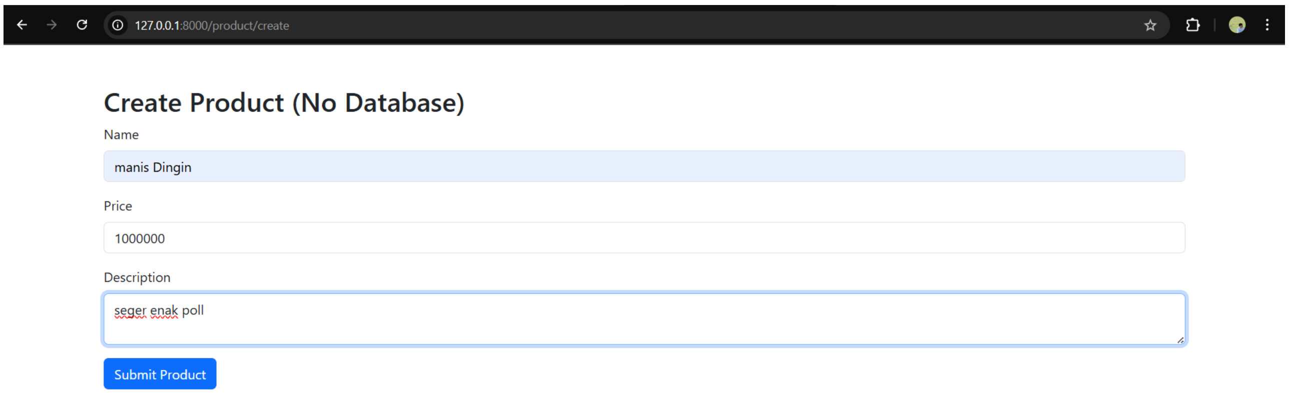


Submitted Product Details

Name: manis Dingin

Price: $1,000,000.00

Description: Seger Enak Pollll



Create Product (No Database)

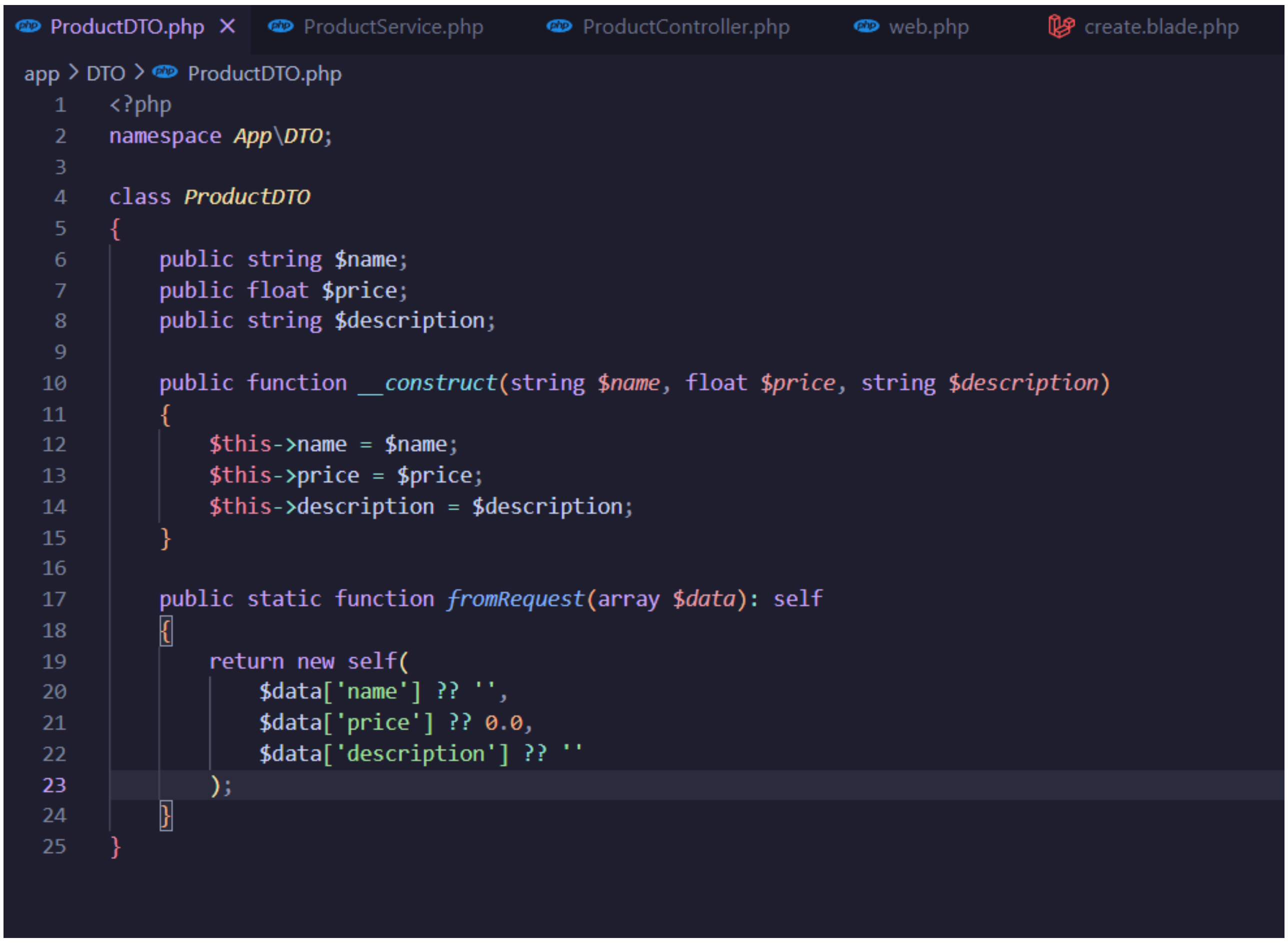
Name: manis Dingin

Price: 1000000

Description: seger enak poll

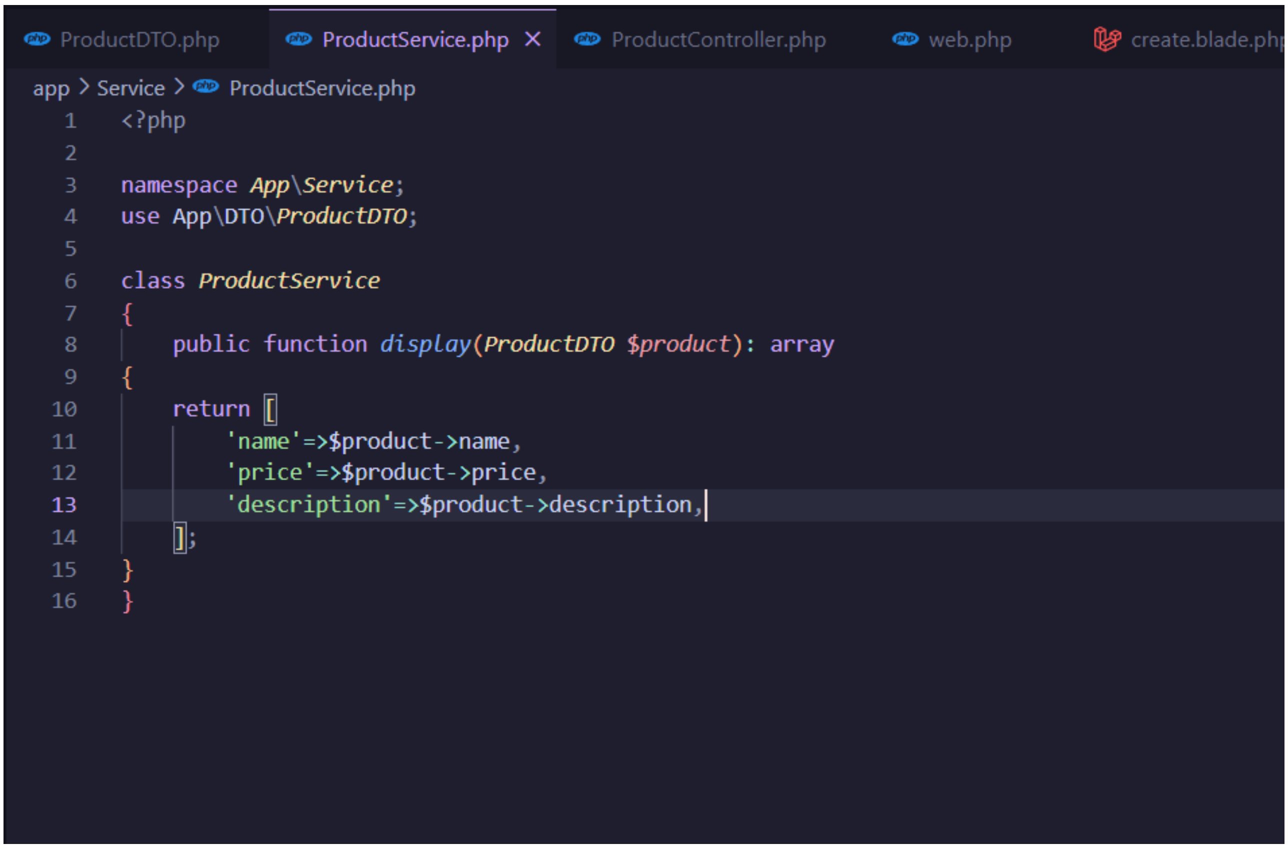
Praktikum 2: Menggunakan DTO (Data Transfer Object)

- Langkah 1: Buat Dan Buka Proyek Laravel Ketik Ini Di terminal laravel new dto-app cd dto-app code .
- Langkah 2: Buat Kelas DTO Buat folder DTO di dalam app: mkdir app/DTO Kemudian Buat File app/DTO/ProductDTO.php: Dan isi dengan Code Berikut



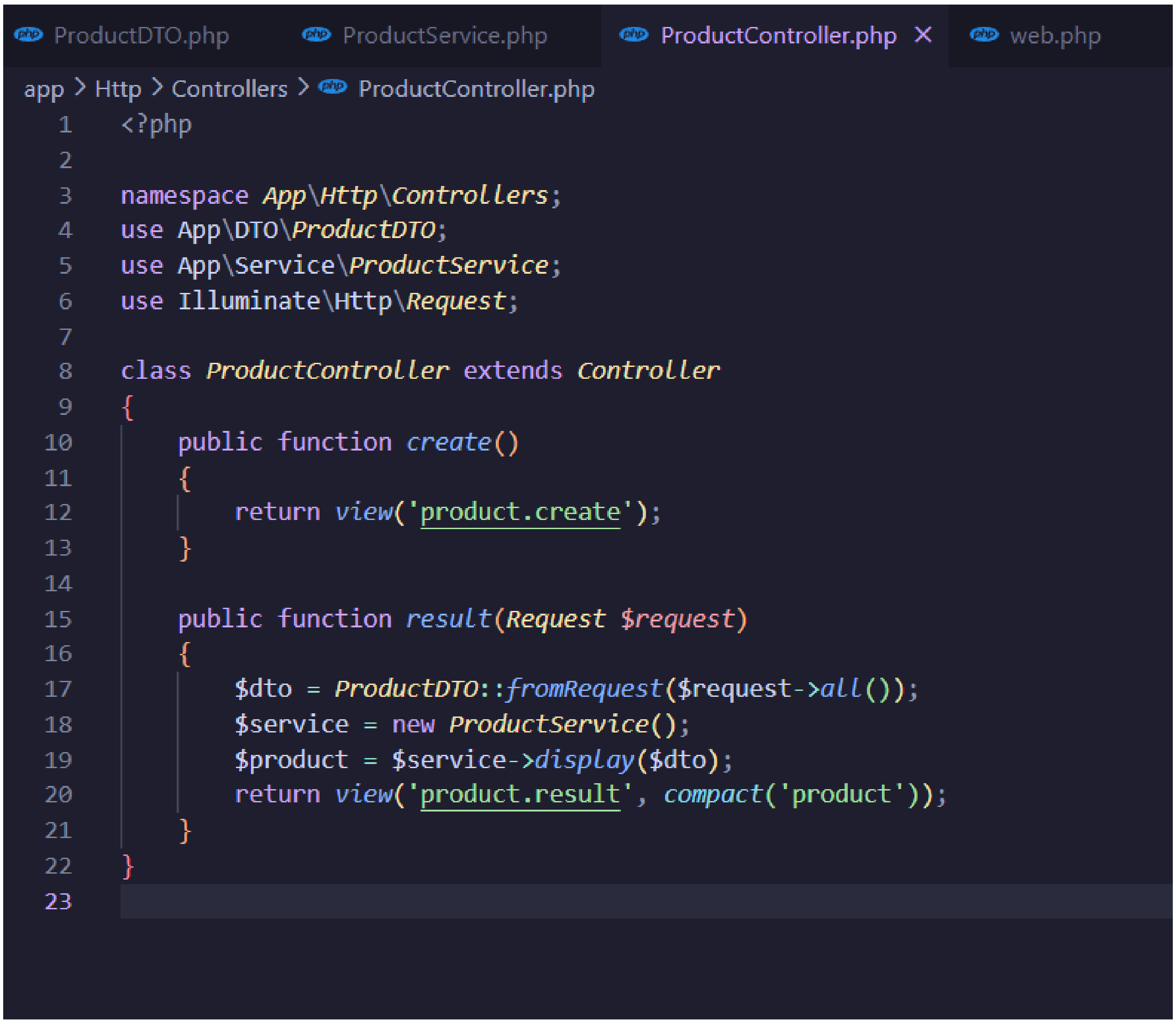
```php
<?php
namespace App\DTO;

class ProductDTO
{
    public string $name;
    public float $price;
    public string $description;

    public function __construct(string $name, float $price, string $description)
    {
        $this->name = $name;
        $this->price = $price;
        $this->description = $description;
    }

    public static function fromRequest(array $data): self
    {
        return new self(
            $data['name'] ?? '',
            $data['price'] ?? 0.0,
            $data['description'] ?? ''
        );
    }
}
```

- Langkah 3: Buat Service Layer Buat folder Services di dalam app: mkdir app/Services Buat file app/Services/ProductService.php: Kemudian isi dengan Code Berikut:



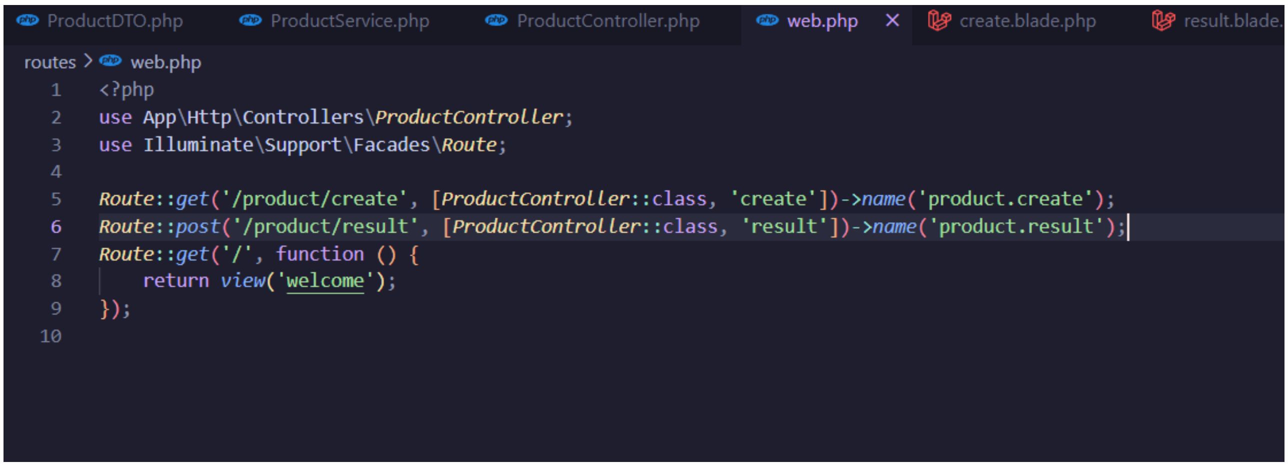
```php
<?php

namespace App\Service;
use App\DTO\ProductDTO;

class ProductService
{
    public function display(ProductDTO $product): array
{
    return [
        'name'=>$product->name,
        'price'=>$product->price,
        'description'=>$product->description,
    ];
}
}
```

- Langkah 4: Buat Controller Buat controller dengan perintah berikut: php artisan make:controller ProductController Kemudian Edit:



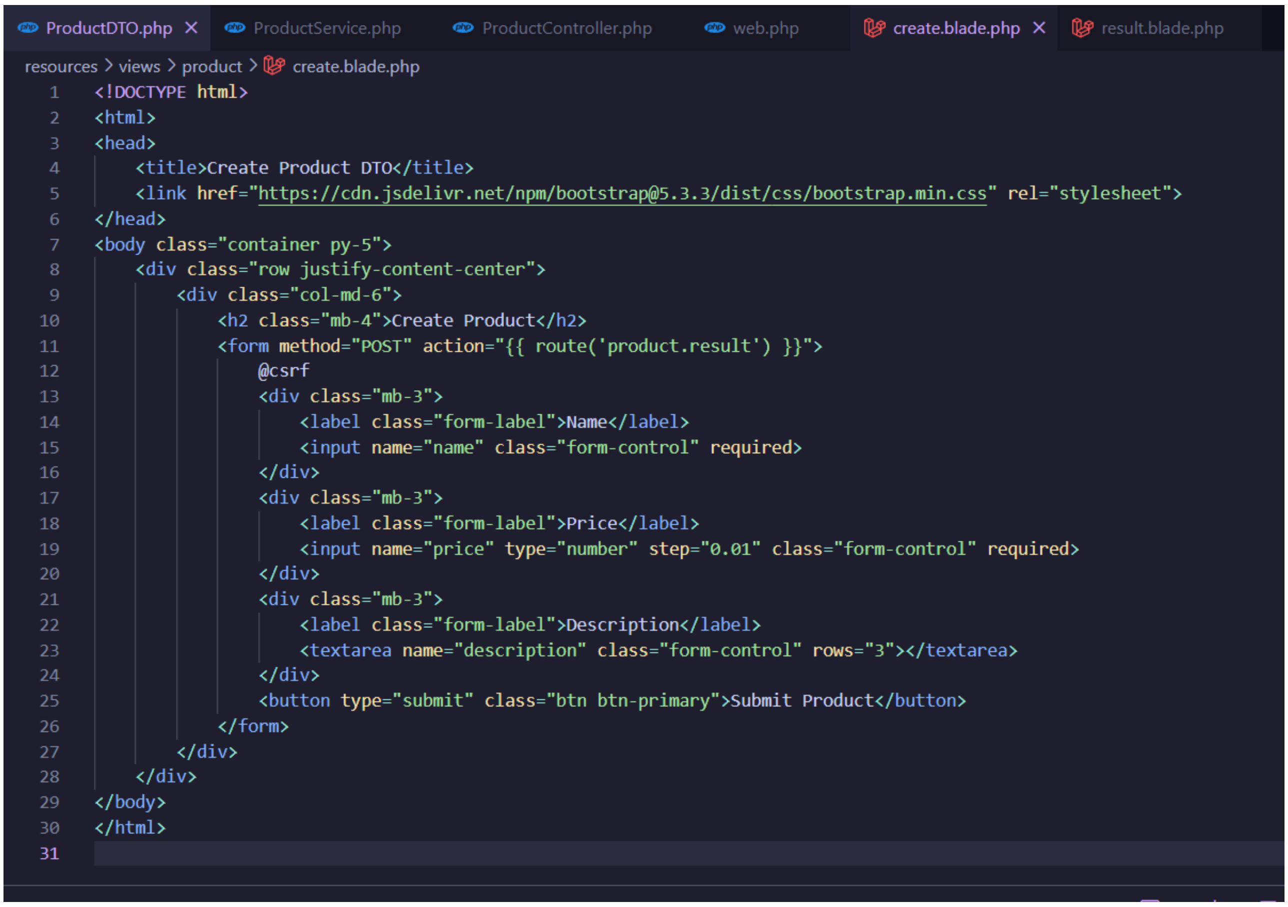
```php
<?php

namespace App\Http\Controllers;
use App\DTO\ProductDTO;
use App\Service\ProductService;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function create()
    {
        return view('product.create');
    }

    public function result(Request $request)
    {
        $dto = ProductDTO::fromRequest($request->all());
        $service = new ProductService();
        $product = $service->display($dto);
        return view('product.result', compact('product'));
    }
}

```

- Langkah 5: Definisikan Rute Isi Dengan Code Berikut ini:

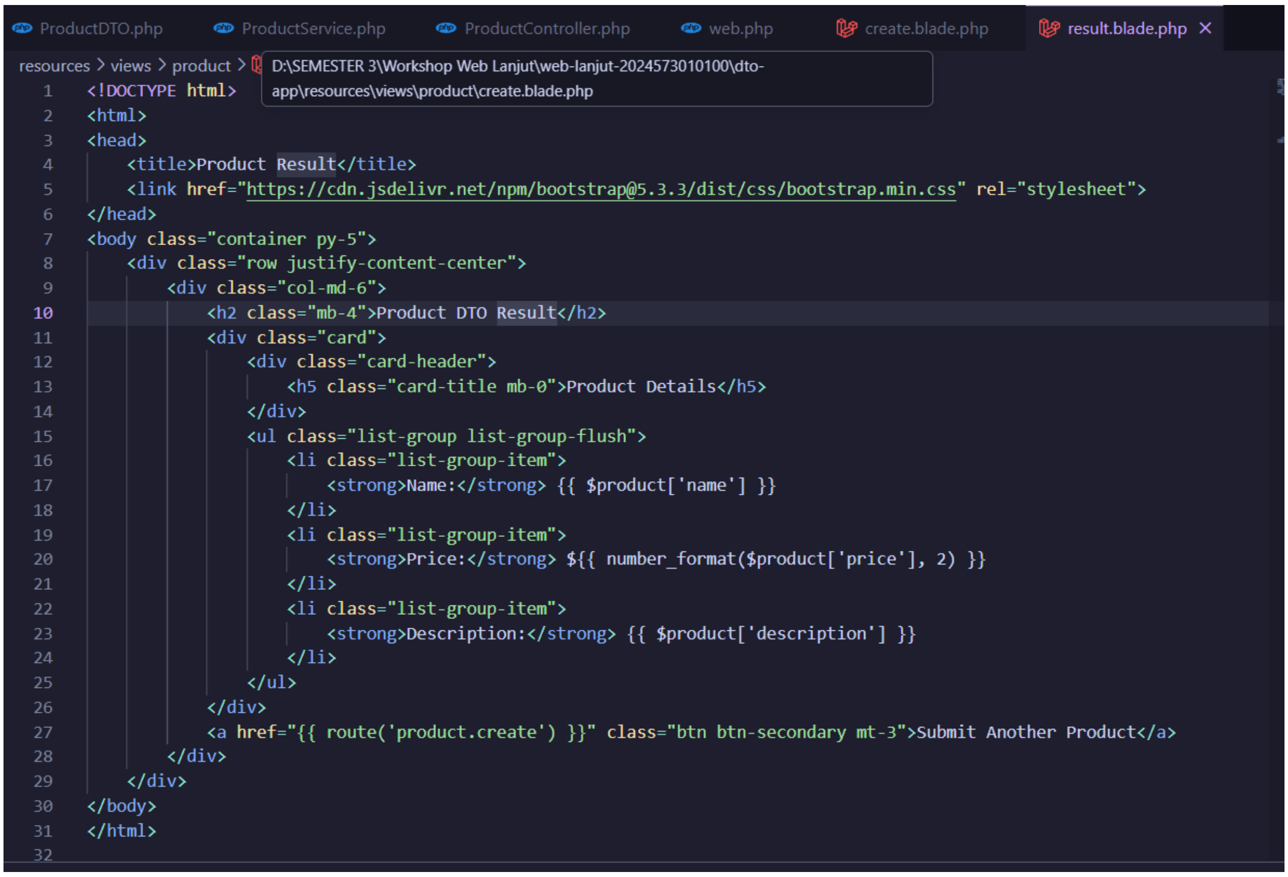


```php
<?php
use App\Http\Controllers\ProductController;
use Illuminate\Support\Facades\Route;

Route::get('/product/create', [ProductController::class, 'create'])->name('product.create');
Route::post('/product/result', [ProductController::class, 'result'])->name('product.result');
Route::get('/', function () {
    return view('welcome');
});

```

- Langkah 6: Buat Tampilan (Views) Dengan Boostrap Buat Direktori Product di dalam resources/views: mkdir resources/views/product Setelah membuat direktori, buat dua file: create.blade.php dan result.blade.php. Kemudian Isi Dengan Code Berikut



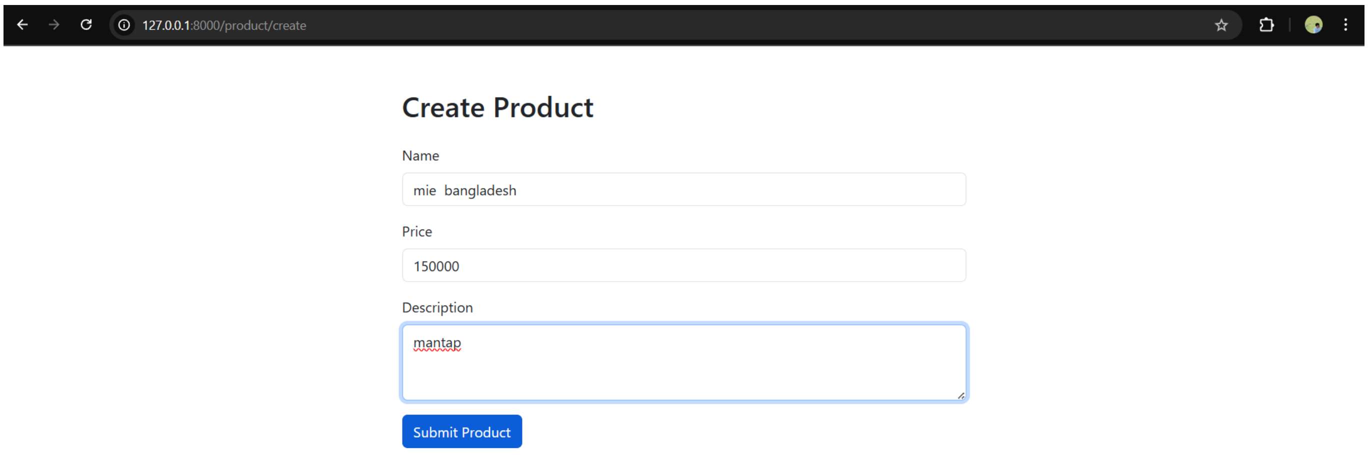
```
<!DOCTYPE html>
<html>
<head>
    <title>Create Product DTO</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container py-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h2 class="mb-4">Create Product</h2>
            <form method="POST" action="{{ route('product.result') }}">
                @csrf
                <div class="mb-3">
                    <label class="form-label">Name</label>
                    <input name="name" class="form-control" required>
                </div>
                <div class="mb-3">
                    <label class="form-label">Price</label>
                    <input name="price" type="number" step="0.01" class="form-control" required>
                </div>
                <div class="mb-3">
                    <label class="form-label">Description</label>
                    <textarea name="description" class="form-control" rows="3"></textarea>
                </div>
                <button type="submit" class="btn btn-primary">Submit Product</button>
            </form>
        </div>
    </div>
</body>
</html>
```


ProductDTO.php
ProductService.php
ProductController.php
web.php
create.blade.php
result.blade.php
resources > views > product >
D:\SEMESTER 3\Workshop Web Lanjut\web-lanjut-2024573010100\dto-app\resources\views\product\create.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Product Result</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</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>Description:</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>
```

- Jalankan Dan uji Aplikasi Setelah menyelesaikan langkah-langkah di atas, jalankan aplikasi dengan perintah berikut: php artisan serve

Kunjungi: http://localhost:8000/product/create Isi formulir → Submit → Lihat hasil yang diteruskan melalui DTO dan service

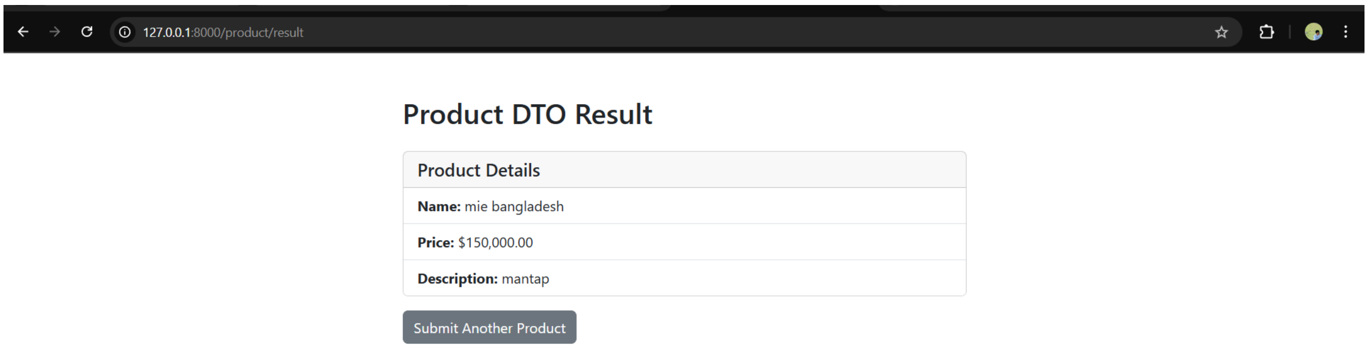

127.0.0.1:8000/product/create

Create Product

Name: mie bangladesh

Price: 150000

Description: mantap


127.0.0.1:8000/product/result

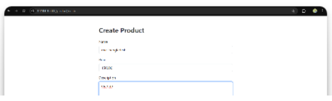
Product DTO Result

Product Details

Name: mie bangladesh

Price: $150,000.00

Description: mantap



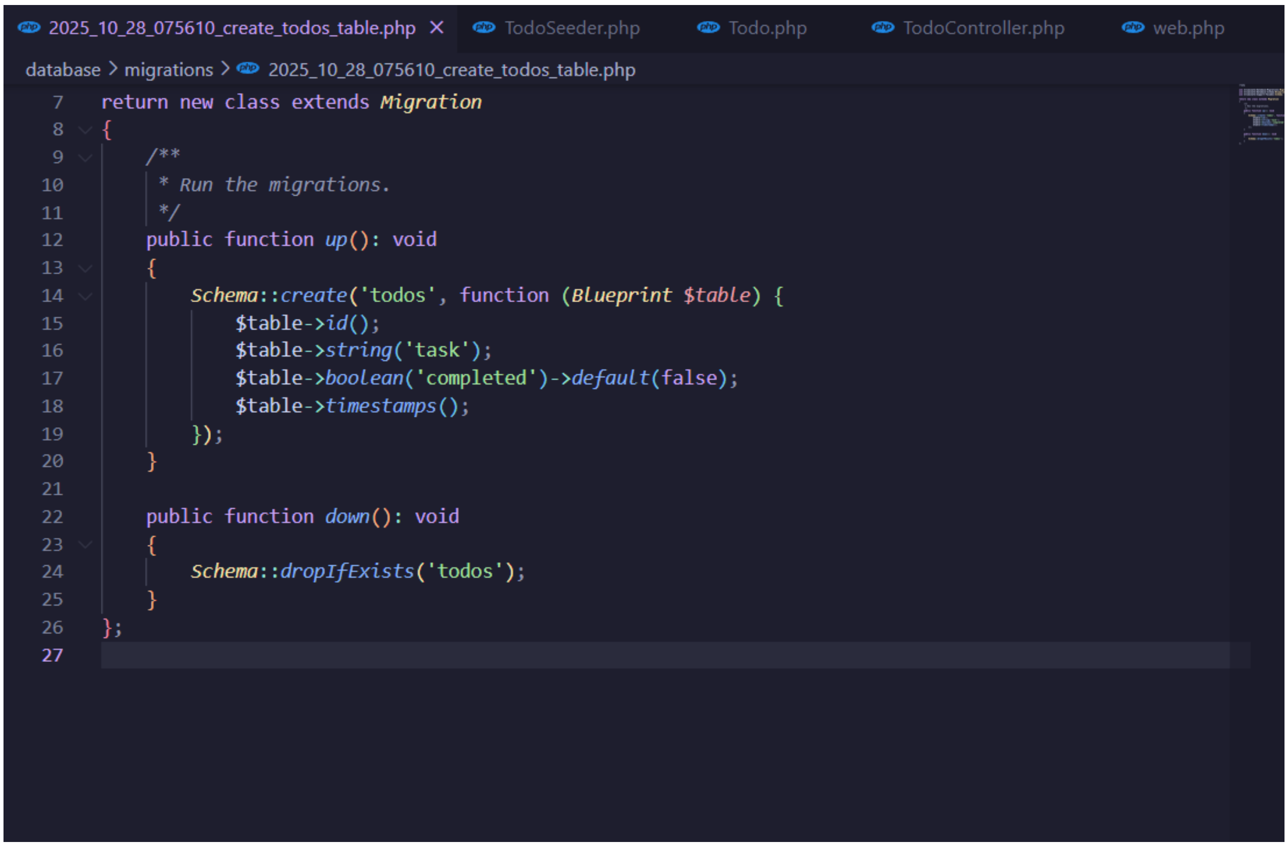
##Praktikum 3: Membangun Aplikasi Web Todo Sederhana dengan Laravel 12, Eloquent ORM, dan MySQL

- Langkah 1: Buat Project Laravel baru laravel new todo-app-mysql cd todo-app-mysql code .
- Pastikan MySQL Berjalan Dan Buat Database:
  CREATE DATABASE tododb;
- Install dependency MySQL: composer require doctrine/dbal
- Konfigurasi MySQL: Edit file .env:

DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=tododb DB_USERNAME= DB_PASSWORD=

Bersihkan Config:clear

- Langkah 2: Buat Migration Untuk Tabel Todos Jalankan Perintah Migrasi: php artisan make:migration create_todos_table Buka file yang dihasilkan di database/migrations/YYYY_MM_DD_create_todos_table.php dan perbarui:

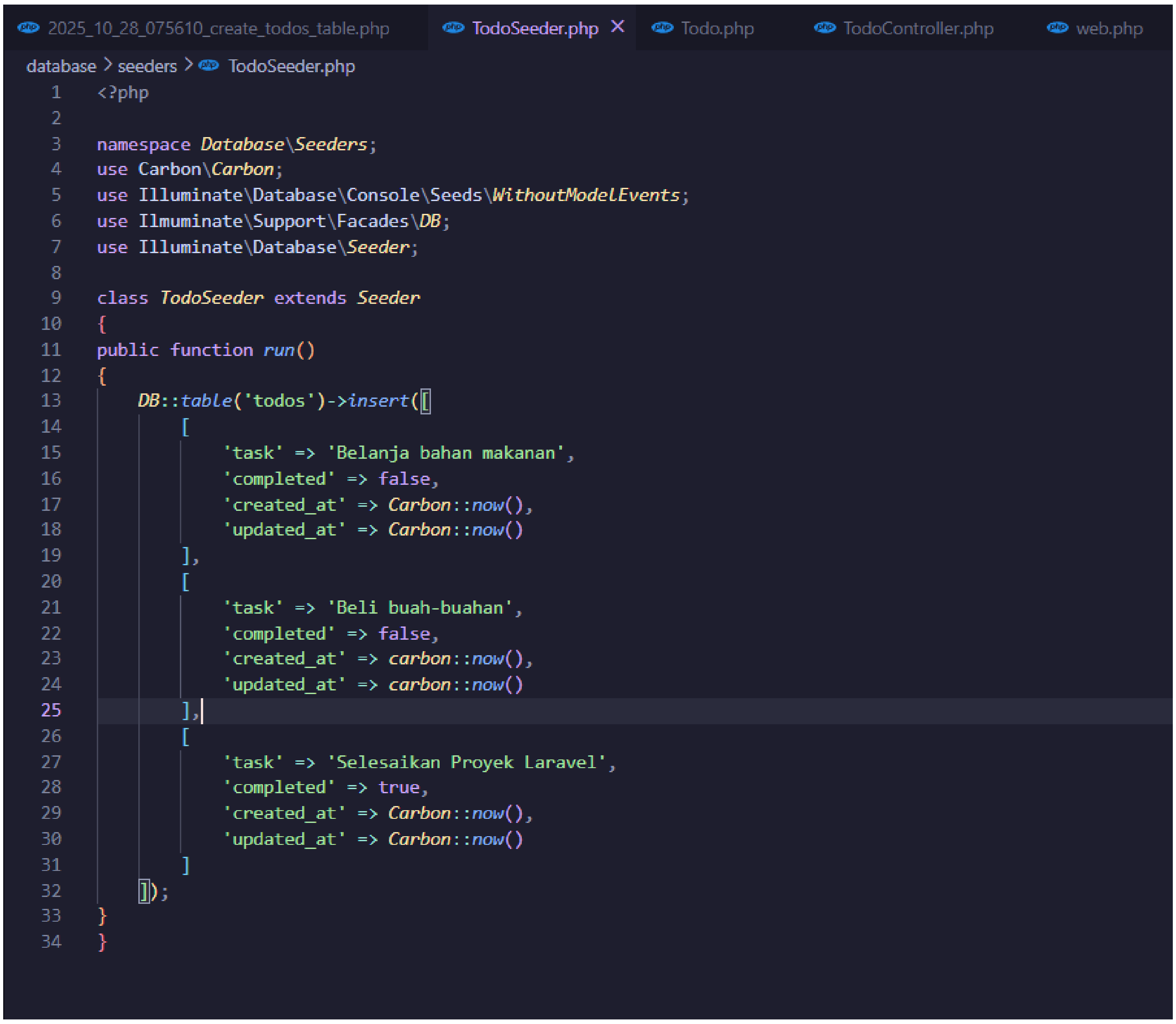

2025_10_28_075610_create_todos_table.php
TodoSeeder.php
Todo.php
TodoController.php
web.php
database > migrations > 2025_10_28_075610_create_todos_table.php

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('todos', function (Blueprint $table) {
            $table->id();
            $table->string('task');
            $table->boolean('completed')->default(false);
            $table->timestamps();
        });
    }

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

Jalankan artisan migrate

- Langkah 3:Buat Seeder Untuk Data Dummy Jalankan Perintah Ini Untuk membuat seeder: php artisan make:seeder TodoSeeder

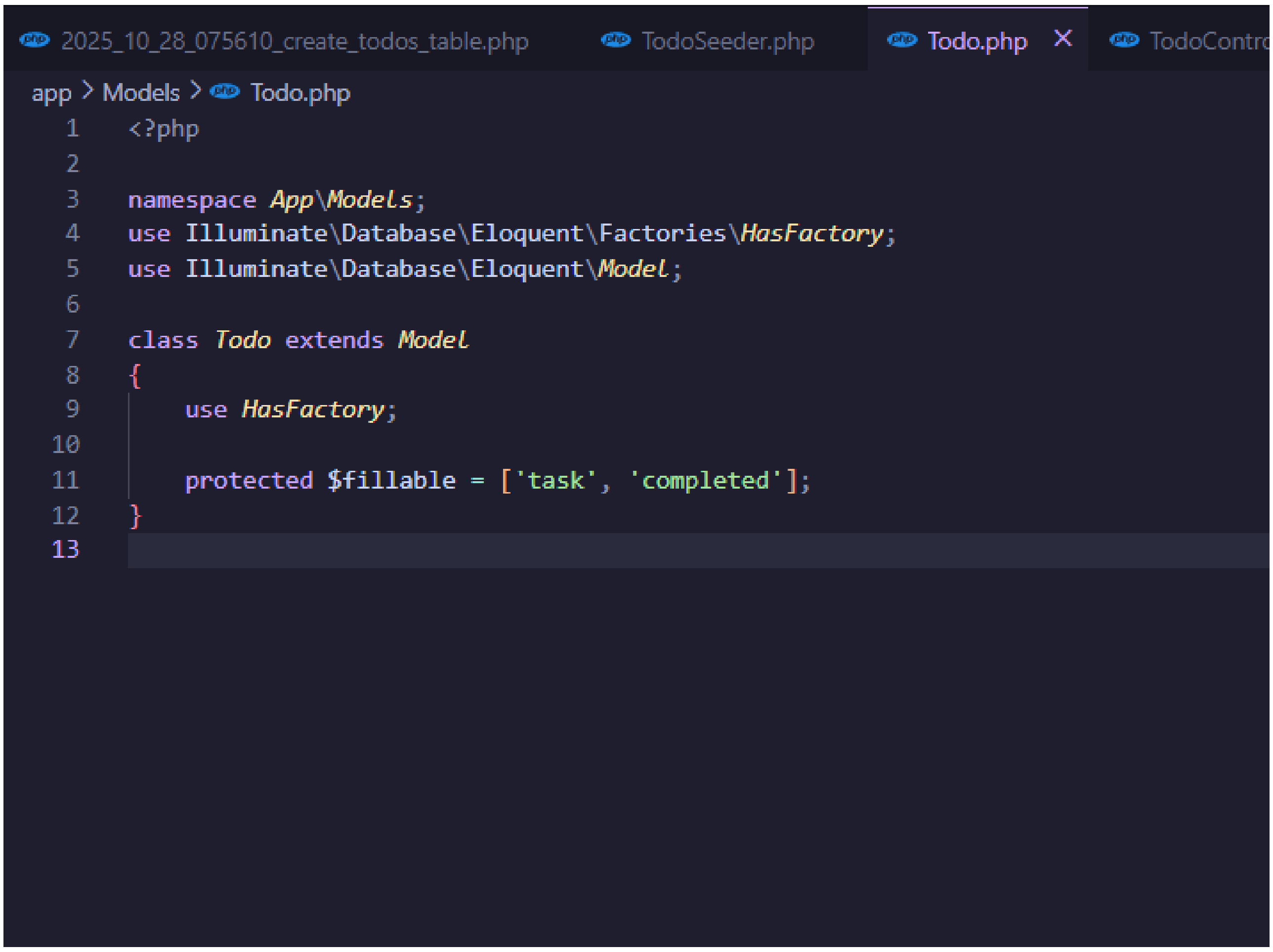
Isikan Kode Berikut


2025_10_28_075610_create_todos_table.php
TodoSeeder.php
Todo.php
TodoController.php
web.php
database > seeders > TodoSeeder.php

```php
<?php

namespace Database\Seeders;
use Carbon\Carbon;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Ilmuminate\Support\Facades\DB;
use Illuminate\Database\Seeder;

class TodoSeeder extends Seeder
{
public function run()
{
    DB::table('todos')->insert([
        [
            'task' => 'Belanja bahan makanan',
            'completed' => false,
            'created_at' => Carbon::now(),
            'updated_at' => Carbon::now()
        ],
        [
            'task' => 'Beli buah-buahan',
            'completed' => false,
            'created_at' => carbon::now(),
            'updated_at' => carbon::now()
        ],
        [
            'task' => 'Selesaikan Proyek Laravel',
            'completed' => true,
            'created_at' => Carbon::now(),
            'updated_at' => Carbon::now()
        ]
    ]);
}
}
```

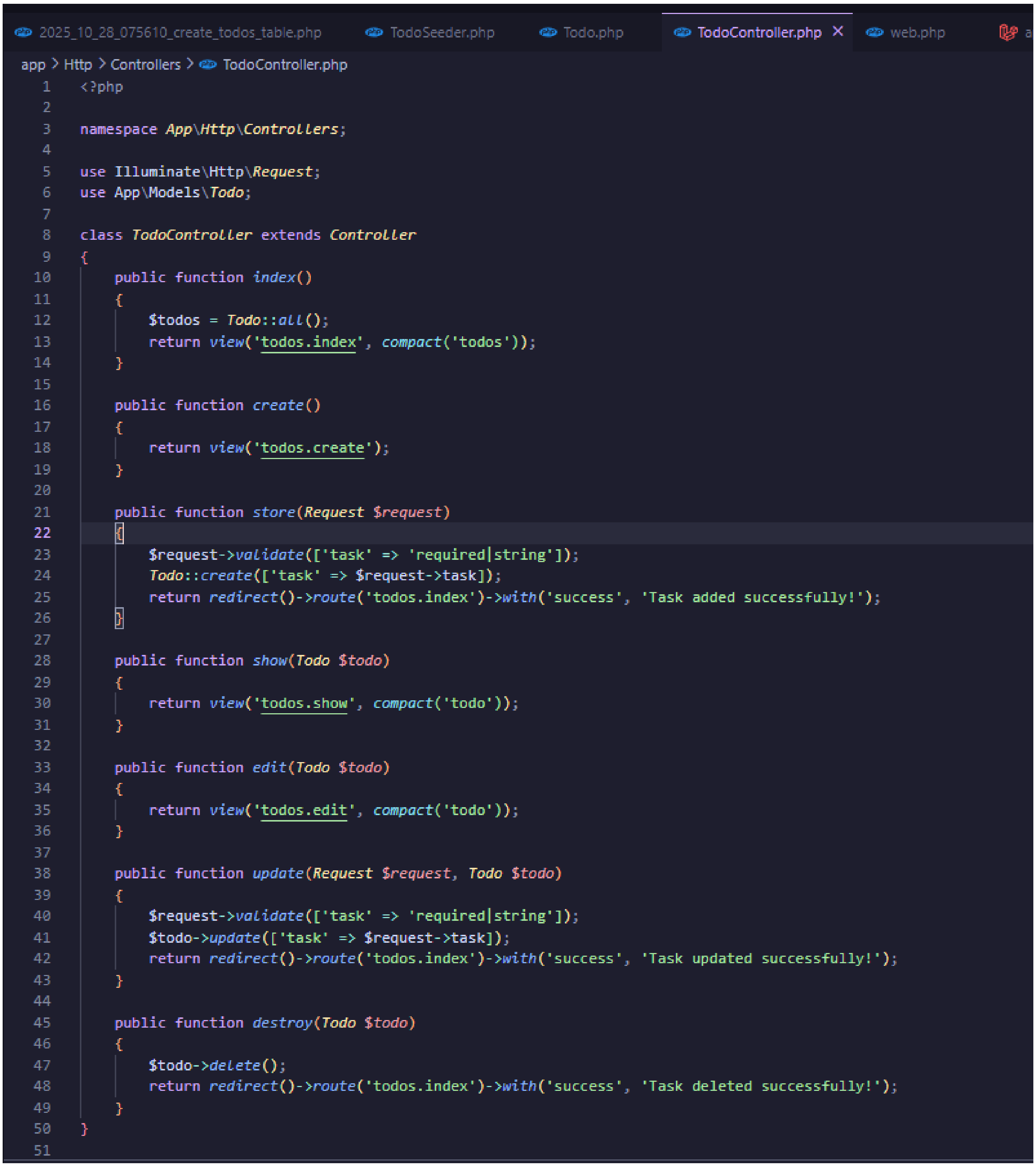
Jalankan Seeder Untuk Mengisi Database: php artisan db:seed --class=TodoSeeder

- Langkah 4: Buat Model Todo Jalankan : php artisan make:model Todo Buka file yang dihasilkan di app/Models/Todo.php dan perbarui:



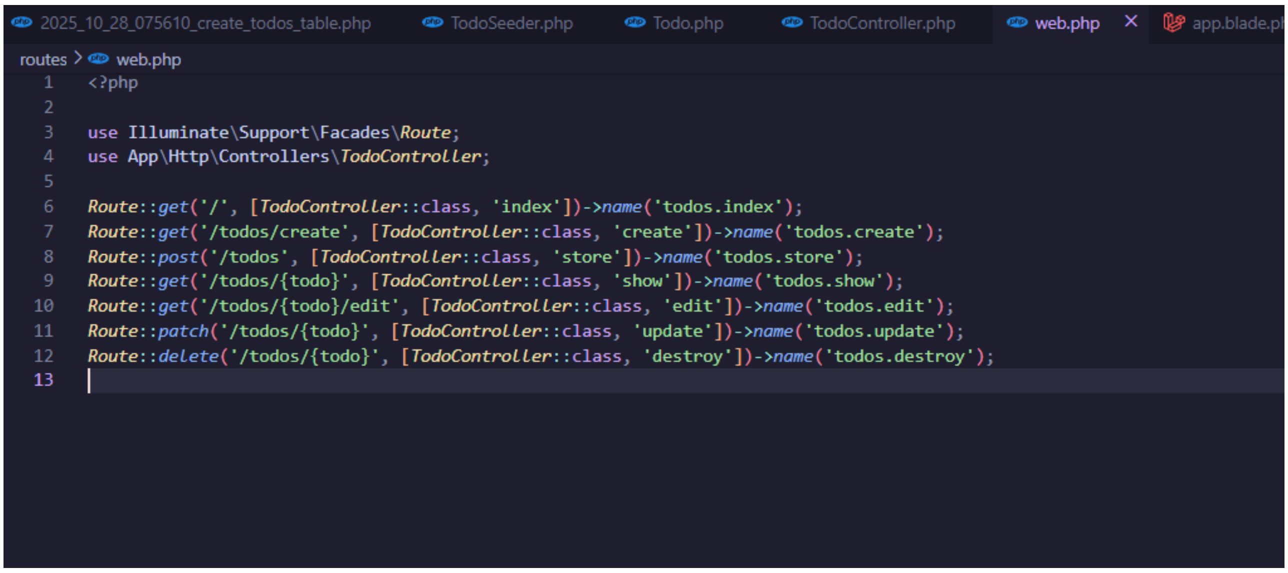
```php
<?php

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

class Todo extends Model
{
    use HasFactory;

    protected $fillable = ['task', 'completed'];
}

```

Jalankan : php artisan make:controller TodoController buka app/Http/Controllers/TodoController.php dan perbarui:


2025_10_28_075610_create_todos_table.php
TodoSeeder.php
Todo.php
TodoController.php
web.php
app > Http > Controllers > TodoController.php

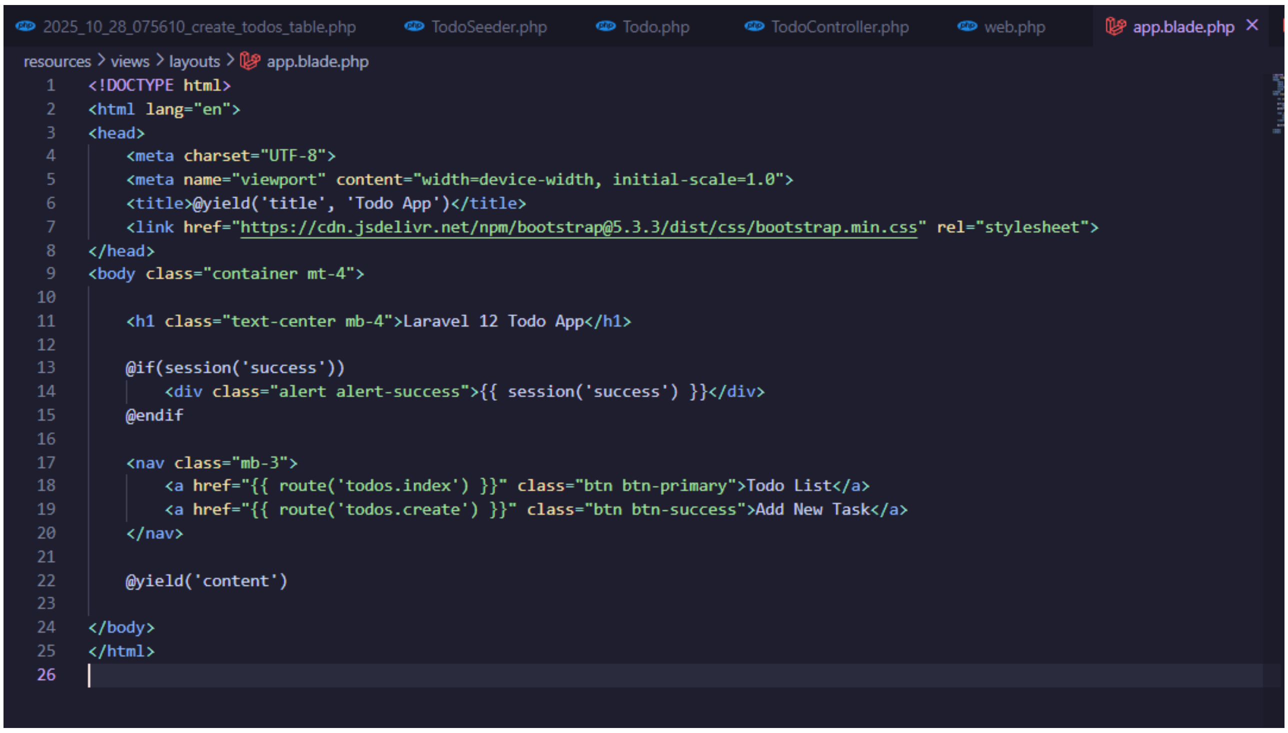
```php
<?php

namespace App\Http\Controllers;

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

class TodoController extends Controller
{
    public function index()
    {
        $todos = Todo::all();
        return view('todos.index', compact('todos'));
    }

    public function create()
    {
        return view('todos.create');
    }

    public function store(Request $request)
    {
        $request->validate(['task' => 'required|string']);
        Todo::create(['task' => $request->task]);
        return redirect()->route('todos.index')->with('success', 'Task added successfully!');
    }

    public function show(Todo $todo)
    {
        return view('todos.show', compact('todo'));
    }

    public function edit(Todo $todo)
    {
        return view('todos.edit', compact('todo'));
    }

    public function update(Request $request, Todo $todo)
    {
        $request->validate(['task' => 'required|string']);
        $todo->update(['task' => $request->task]);
        return redirect()->route('todos.index')->with('success', 'Task updated successfully!');
    }

    public function destroy(Todo $todo)
    {
        $todo->delete();
        return redirect()->route('todos.index')->with('success', 'Task deleted successfully!');
    }
}

```

- Langkah 5: Definisikan Rute Web Edit routes/web.php



```php
<?php

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

Route::get('/', [TodoController::class, 'index'])->name('todos.index');
Route::get('/todos/create', [TodoController::class, 'create'])->name('todos.create');
Route::post('/todos', [TodoController::class, 'store'])->name('todos.store');
Route::get('/todos/{todo}', [TodoController::class, 'show'])->name('todos.show');
Route::get('/todos/{todo}/edit', [TodoController::class, 'edit'])->name('todos.edit');
Route::patch('/todos/{todo}', [TodoController::class, 'update'])->name('todos.update');
Route::delete('/todos/{todo}', [TodoController::class, 'destroy'])->name('todos.destroy');
```

- Langkah 6: Buat Folder Layouts Buat file di resources/view dan buat file resources/views/layouts/app.blade.php

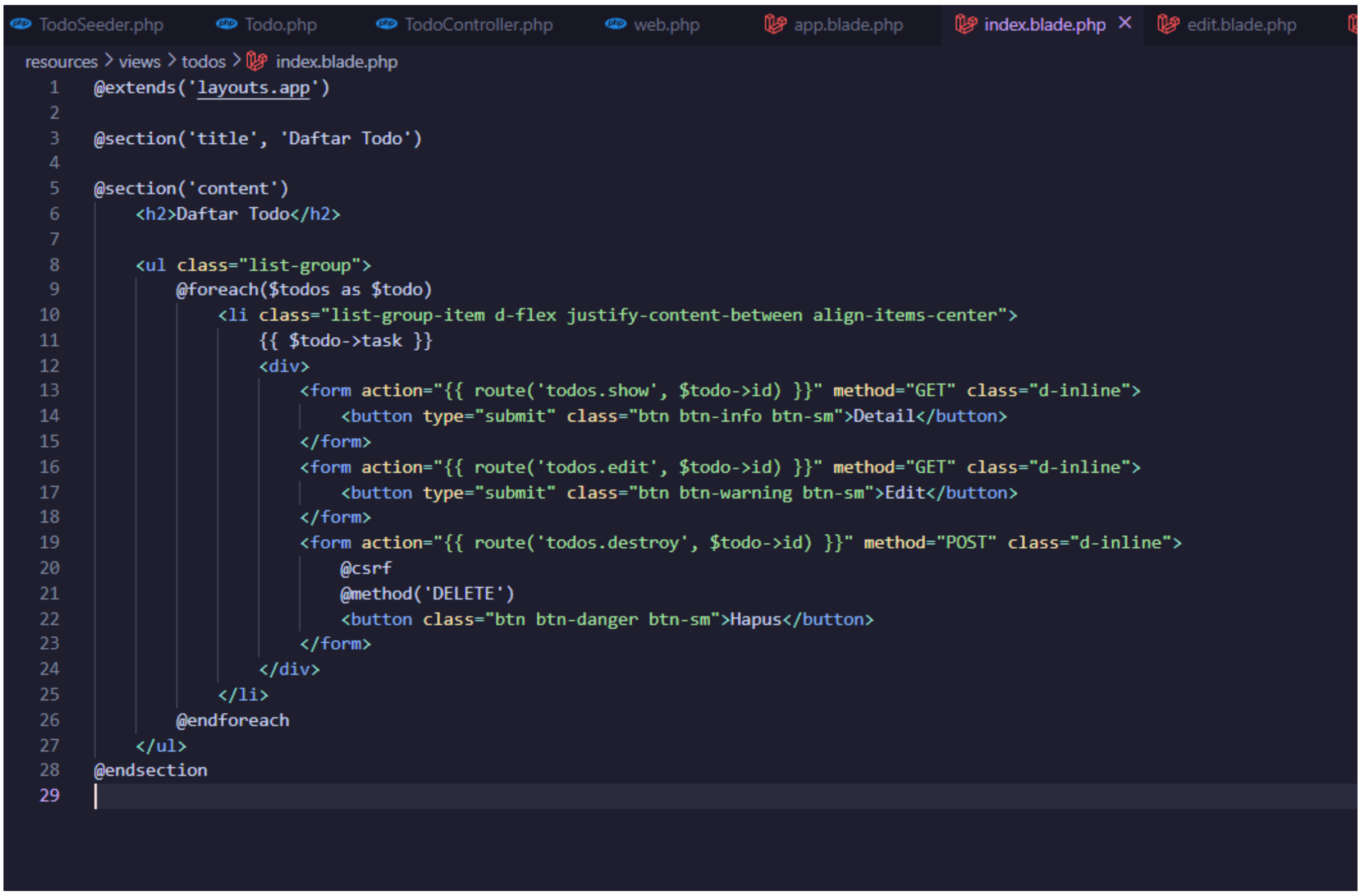

2025_10_28_075610_create_todos_table.php
TodoSeeder.php
Todo.php
TodoController.php
web.php
app.blade.php
resources > views > layouts > app.blade.php

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', 'Todo App')</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 Todo App</h1>

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

    <nav class="mb-3">
        <a href="{{ route('todos.index') }}" class="btn btn-primary">Todo List</a>
        <a href="{{ route('todos.create') }}" class="btn btn-success">Add New Task</a>
    </nav>

    @yield('content')

</body>
</html>
```

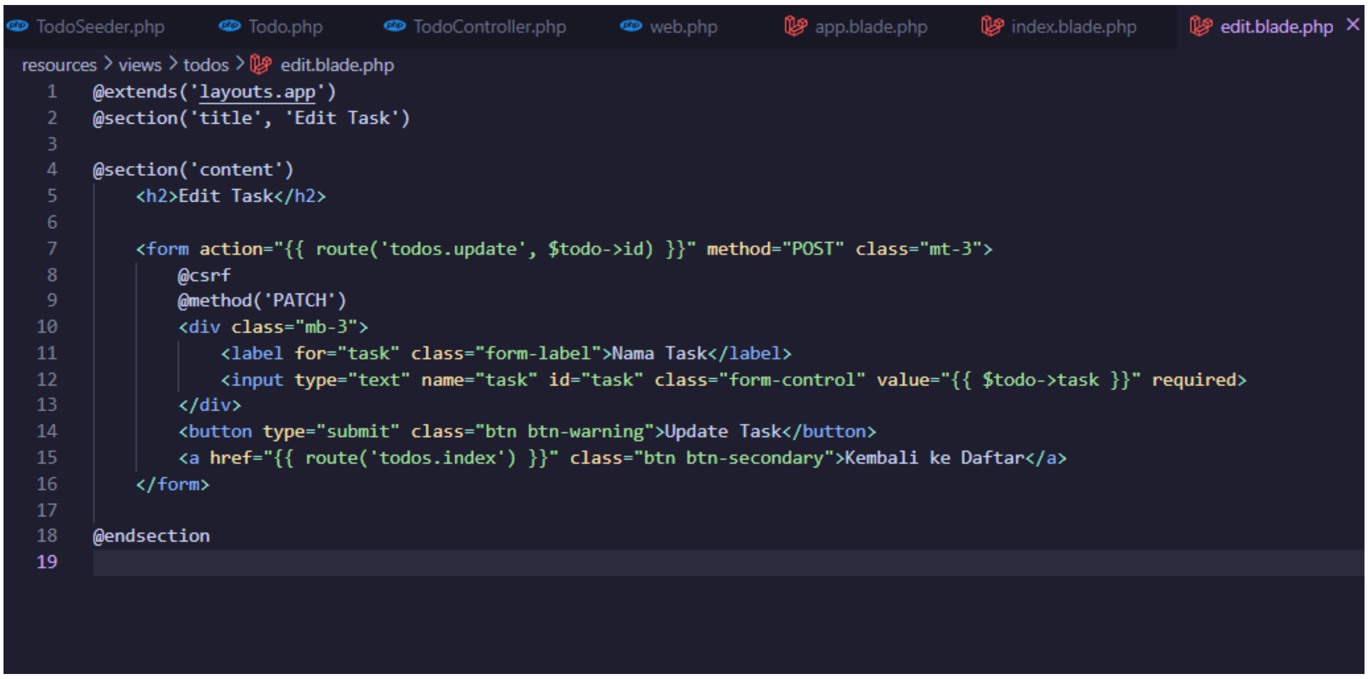
Kemudian di dalam views bikin file

- Index.blade.php
- Create.blade.php
- edit.blade.php
- show.blade.php

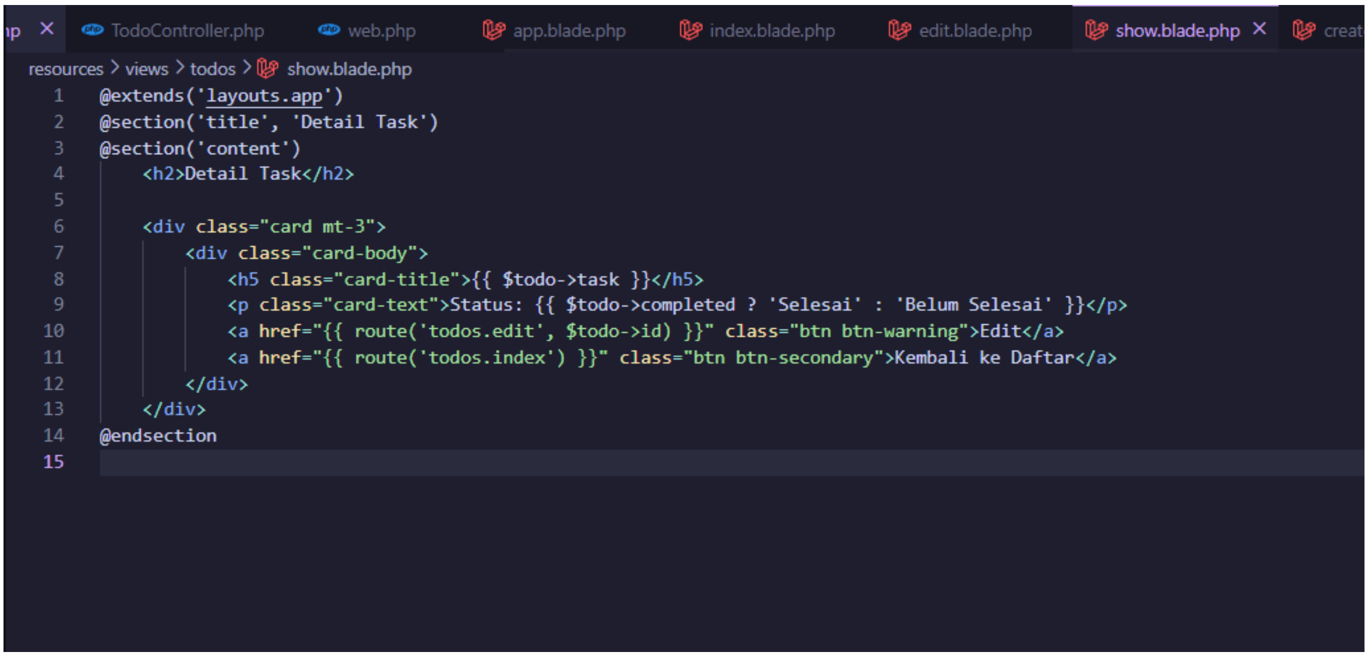
isi dengan code berikut:



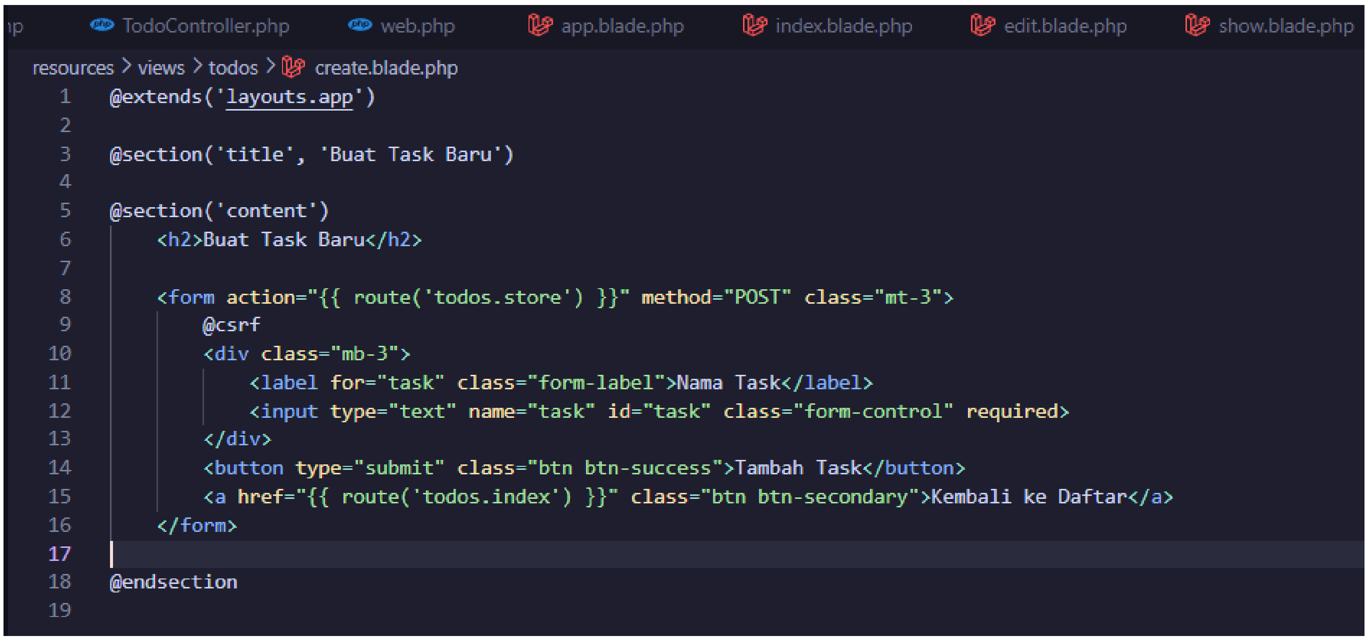
```
@extends('layouts.app')

@section('title', 'Daftar Todo')

@section('content')
    <h2>Daftar Todo</h2>

    <ul class="list-group">
        @foreach($todos as $todo)
            <li class="list-group-item d-flex justify-content-between align-items-center">
                {{ $todo->task }}
                <div>
                    <form action="{{ route('todos.show', $todo->id) }}" method="GET" class="d-inline">
                        <button type="submit" class="btn btn-info btn-sm">Detail</button>
                    </form>
                    <form action="{{ route('todos.edit', $todo->id) }}" method="GET" class="d-inline">
                        <button type="submit" class="btn btn-warning btn-sm">Edit</button>
                    </form>
                    <form action="{{ route('todos.destroy', $todo->id) }}" method="POST" class="d-inline">
                        @csrf
                        @method('DELETE')
                        <button class="btn btn-danger btn-sm">Hapus</button>
                    </form>
                </div>
            </li>
        @endforeach
    </ul>
@endsection

```



```
@extends('layouts.app')
@section('title', 'Edit Task')

@section('content')
    <h2>Edit Task</h2>

    <form action="{{ route('todos.update', $todo->id) }}" method="POST" class="mt-3">
        @csrf
        @method('PATCH')
        <div class="mb-3">
            <label for="task" class="form-label">Nama Task</label>
            <input type="text" name="task" id="task" class="form-control" value="{{ $todo->task }}" required>
        </div>
        <button type="submit" class="btn btn-warning">Update Task</button>
        <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
    </form>

@endsection

```



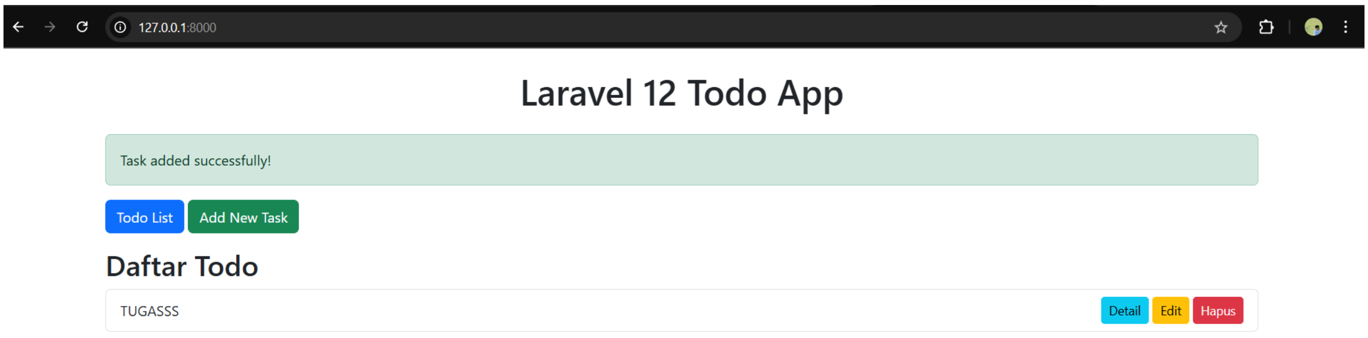
```
@extends('layouts.app')
@section('title', 'Detail Task')
@section('content')
    <h2>Detail Task</h2>

    <div class="card mt-3">
        <div class="card-body">
            <h5 class="card-title">{{ $todo->task }}</h5>
            <p class="card-text">Status: {{ $todo->completed ? 'Selesai' : 'Belum Selesai' }}</p>
            <a href="{{ route('todos.edit', $todo->id) }}" class="btn btn-warning">Edit</a>
            <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
        </div>
    </div>
@endsection
```



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

@section('title', 'Buat Task Baru')

@section('content')
    <h2>Buat Task Baru</h2>

    <form action="{{ route('todos.store') }}" method="POST" class="mt-3">
        @csrf
        <div class="mb-3">
            <label for="task" class="form-label">Nama Task</label>
            <input type="text" name="task" id="task" class="form-control" required>
        </div>
        <button type="submit" class="btn btn-success">Tambah Task</button>
        <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
    </form>

@endsection
```

- langkah 7: Jalankan Dan Uji Aplikasi jalankan Perintah : php artisan serve kemudian,kunjungi link http://127.0.0.1:8000

lakukan ujicoba berikut: Klik Tambah Task Baru untuk membuat task baru Klik Detail untuk melihat detail task Klik Edit untuk memperbarui task Klik Hapus untuk menghapus task Klik Kembali ke Daftar untuk kembali ke daftar Todo



127.0.0.1:8000

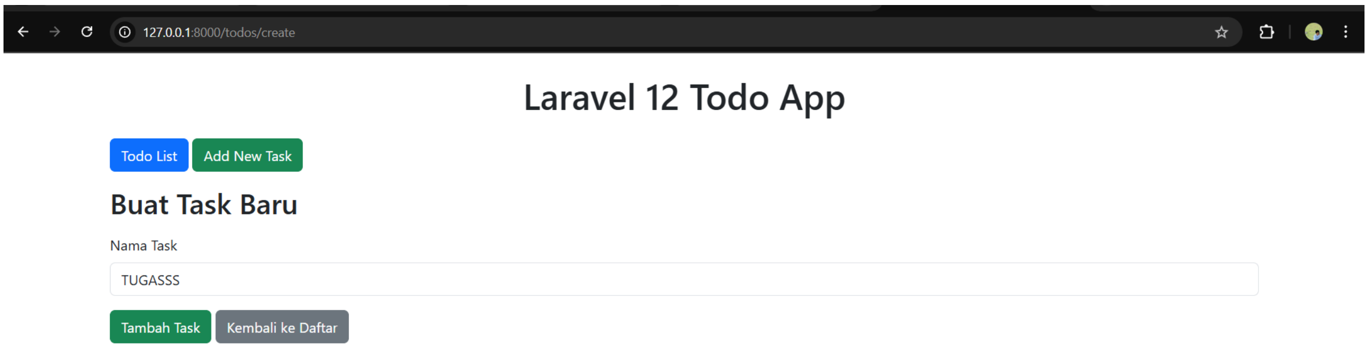
Laravel 12 Todo App

Task added successfully!

Todo List Add New Task

Daftar Todo

TUGASSS Detail Edit Hapus



127.0.0.1:8000/todos/create

Laravel 12 Todo App

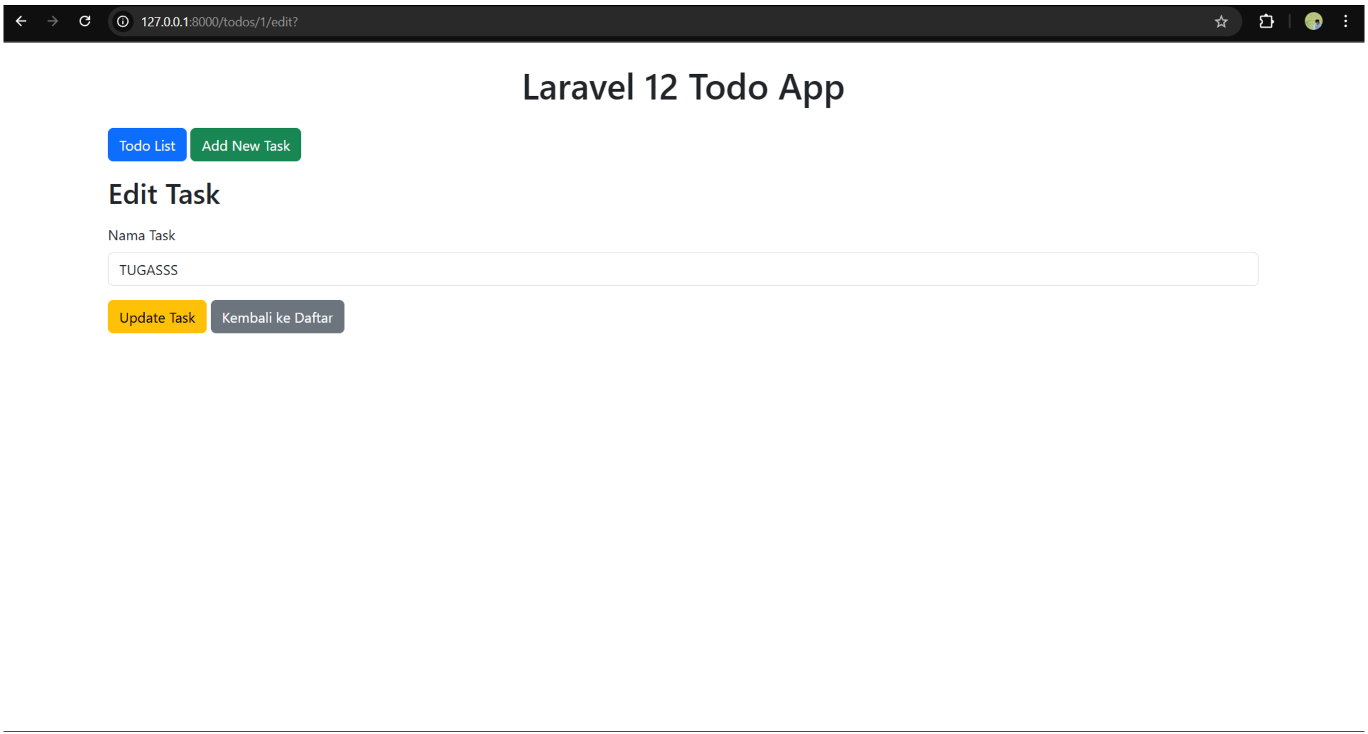
Todo List Add New Task

Buat Task Baru

Nama Task

TUGASSS

Tambah Task Kembali ke Daftar

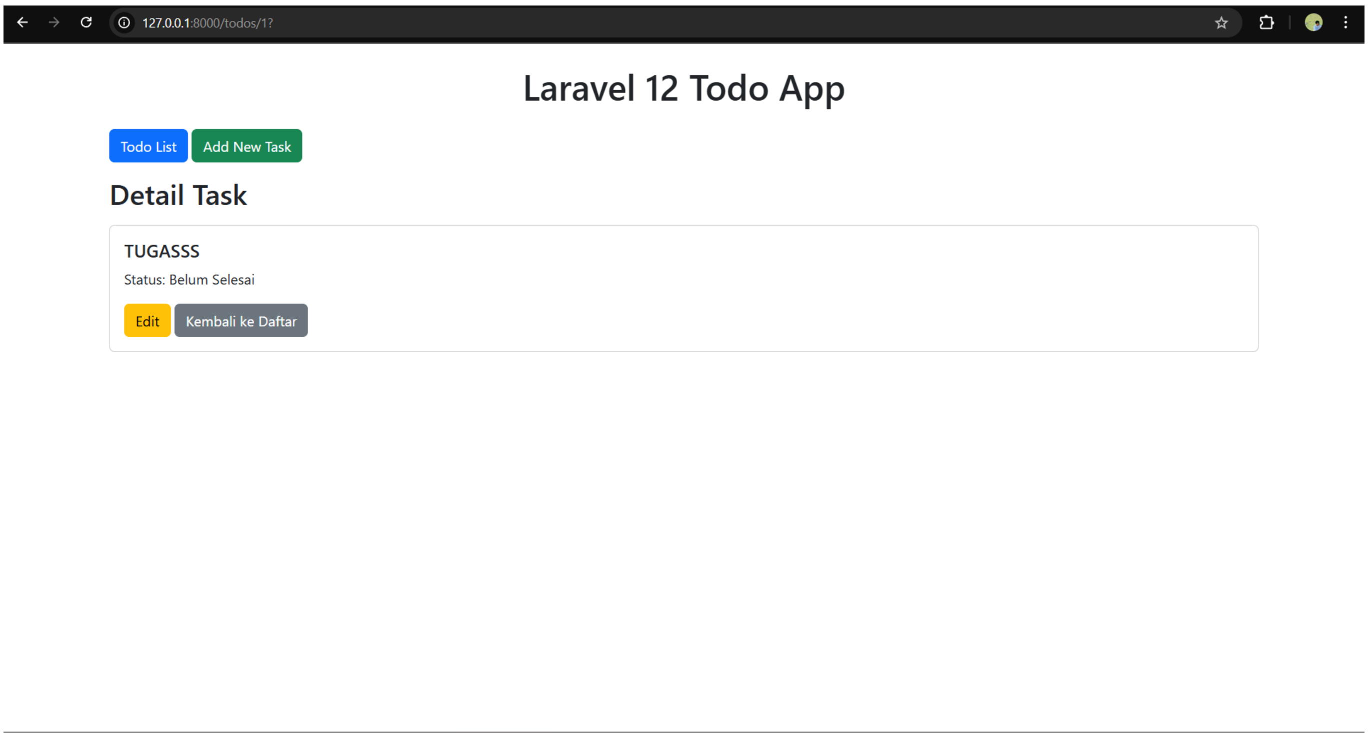

127.0.0.1:8000/todos/1/edit?

Laravel 12 Todo App

Edit Task

Nama Task

TUGASSS

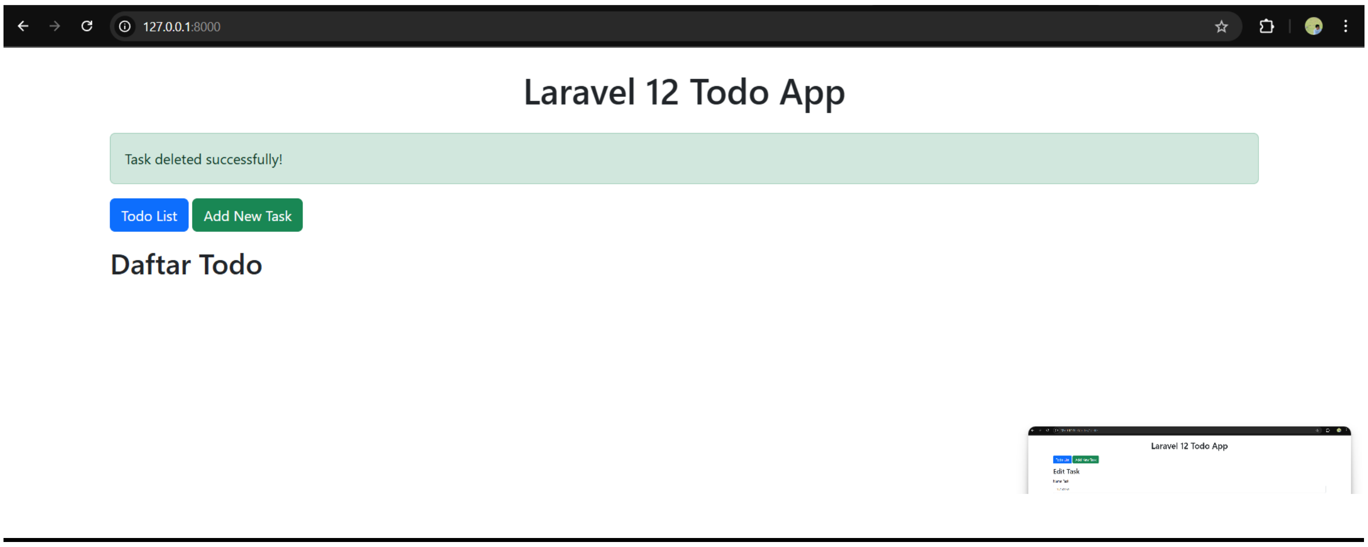

127.0.0.1:8000/todos/1?

Laravel 12 Todo App

Detail Task

TUGASSS

Status: Belum Selesai


127.0.0.1:8000

Laravel 12 Todo App

Task deleted successfully!

Daftar Todo

Laravel 12 Todo App

Edit Task

3. Hasil dan Pembahasan

Dari ketiga praktikum yang dilakukan, diperoleh hasil sebagai berikut:

Praktikum 1: Sistem dapat menampilkan form input data dan menampilkan hasilnya di halaman baru menggunakan Request dan Response View. Tidak ada database yang digunakan karena model hanya berfungsi sebagai penampung data sementara.

Praktikum 2: Aplikasi berhasil menerapkan validasi kustom menggunakan Form Request dan menampilkan pesan error di halaman view. Fitur ini penting agar data yang masuk ke sistem terjamin validitasnya.

Praktikum 3: Implementasi multi-step form berjalan dengan baik. Setiap langkah form menyimpan data ke dalam session sehingga pengguna dapat mengisi data bertahap hingga proses selesai. Hasil akhirnya menampilkan ringkasan data yang sudah dimasukkan sebelumnya.

Secara keseluruhan, ketiga praktikum ini memperkuat pemahaman mahasiswa terhadap alur kerja MVC Laravel, pengelolaan data melalui model, serta penerapan Eloquent ORM dan session untuk kebutuhan aplikasi web yang dinamis.

---

4. Kesimpulan

Dari hasil praktikum dapat disimpulkan bahwa:

- Model berfungsi sebagai penghubung antara controller dan data, baik yang berasal dari form maupun database.
- Laravel Eloquent memudahkan proses interaksi dengan database menggunakan konsep ORM tanpa perlu menulis query SQL secara manual.
- Validasi data sangat penting untuk menjaga keakuratan input pengguna sebelum disimpan.
- Multi-step form dengan session memberikan pengalaman input data yang lebih terstruktur dan interaktif.
- Secara keseluruhan, penerapan Model dan Eloquent ORM di Laravel membantu pengembangan aplikasi web menjadi lebih efisien, terorganisir, dan mudah dikembangkan.

---

5. Referensi

- HackMD.io — Modul Laravel Model
- ChatGPT (chat.openai.com)
- Laravel Documentation — Validation
- SantriKoding — Validasi Data di Laravel
