# Laporan Modul 6: Model dan Laravel Eloquent

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

## Abstrak

Dalam Praktikum ini membahas konsep Model dan penerapan Eloquent ORM pada framework Laravel. Tujuannya adalah untuk memahami cara berinteraksi dengan database menggunakan pendekatan berbasis objek tanpa harus menulis query SQL secara langsung. Dengan menggunakan Eloquent, setiap tabel di database direpresentasikan sebagai model yang memiliki relasi, atribut, dan fungsi manipulasi data sendiri. Praktikum ini juga mendemonstrasikan operasi dasar seperti create, read, update, dan delete (CRUD), serta implementasi relasi antar tabel seperti one-to-many dan many-to-many. Melalui percobaan ini, mahasiswa memahami cara membangun logika bisnis aplikasi yang efisien, terstruktur, dan mudah dikelola dengan memanfaatkan fitur bawaan Laravel.

## 1. Dasar Teori

### Dasar Teori

Dalam arsitektur MVC (Model–View–Controller), Model berfungsi sebagai penghubung antara aplikasi dan database. Model bertanggung jawab untuk mengelola data, baik dalam hal penyimpanan, pengambilan, maupun pemrosesan logika bisnis.

1. Model Laravel = Kelas yang merepresentasikan tabel di database dan berinteraksi langsung dengan data.
2. Eloquent ORM = Sistem Object-Relational Mapping Laravel yang memungkinkan manipulasi data melalui objek tanpa SQL manual.
3. Konvensi Penamaan = Nama model umumnya singular (misal: User), sedangkan nama tabel plural (users).
4. CRUD Operations = Eloquent menyediakan metode sederhana seperti create(), all(), find(), update(), dan delete().
5. Relasi Antar Model = Mendukung relasi one-to-one, one-to-many, many-to-many, dan hasManyThrough.
6. Mass Assignment Protection = Menggunakan properti $fillable atau $guarded untuk melindungi atribut dari input yang tidak sah.
7. Query Builder Integration = Eloquent dapat dikombinasikan dengan Query Builder untuk operasi data yang lebih kompleks.

## 2. Langkah-Langkah Praktikum

Praktikum 1: Menggunakan Model untuk Binding Form dan Display

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

- Langkah 2: Membuat model data sederhana (POCO)
  Buat ProductViewModel.php di dalam direktori app/ViewModels
  lalu isi kode :

app > ViewModels > ProductViewModel.php

```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 ProductConntroller`
  ini akan membuat app/Http/Controllers/ProductController.php

dan isi dengan kode :

app > Http > Controllers > ProductController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\ViewModels\ProductViewModel;

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 route
  Buka Routes/Web.php Dan isi dengan code berikut

routes > web.php

```php
<?php

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

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


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

- Langkah 5: Buat Blade View menggunakan bootstrap :
  - Buat File Baru di Direktori resources/views/product/create.blade.php :
    Isi Dengan Code Berikut

resources > views > product > create.blade.php

```blade.php
<!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>
```

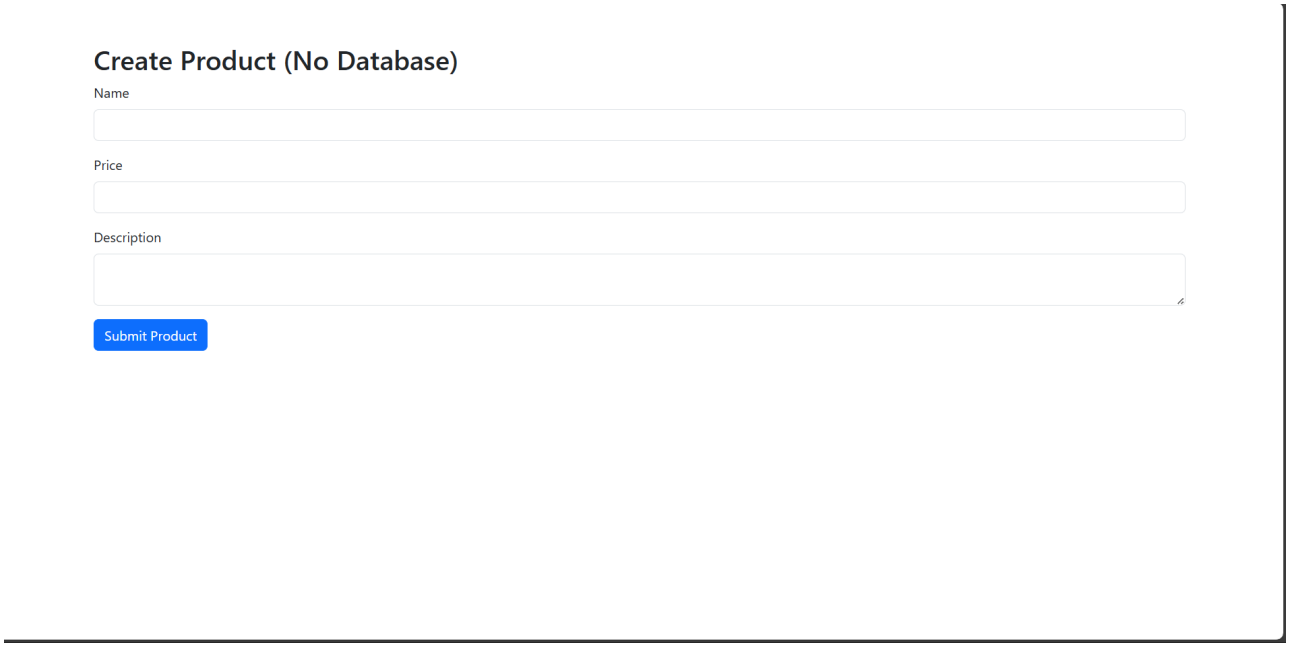
- Buat File Baru di Direktori resources/views/product/result.blade.php :
  Isi Dengan Code Berikut

resources > views > product > result.blade.php

```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">
    <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>
```

- Langkah 5: Uji Rute
  Mulai Php `Artisan serve`
  Buka Browser Dan Kunjungi

http://127.0.0.1:8000/product/create




Praktikum 2: Menggunakan DTO (Data Transfer Object)

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

- Langkah 2: Buat kelas DTO
  Buat ProductDTO.php di dalam direktori app/DTO

lalu isi kode :

app > DTO > ProductDTO.php

```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(
            $data['name'] ?? '',
            (float)($data['price'] ?? 0),
            $data['description'] ?? ''
        );
    }
}
```

- Langkah 3: Buat service layer
  Buat ProductService.php di dalam direktori app/Services
  lalu isi kode :

app > Services > ProductService.php

```php
<?php

namespace App\Services;

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 Sebuah Controller
  `php artisan make:controller ProductConntroller`
  ini akan membuat app/Http/Controllers/ProductController.php. lalu isi kode :

app > Http > Controllers > ProductController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\DTO\ProductDTO;
use App\Services\ProductService;

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

- Langkah 6: Buat Blade View menggunakan bootstrap :
  - Buat File Baru di Direktori resources/views/product/create.blade.php :
    Isi Dengan Code Berikut

resources > views > product > create.blade.php

```blade.php
<!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>
```

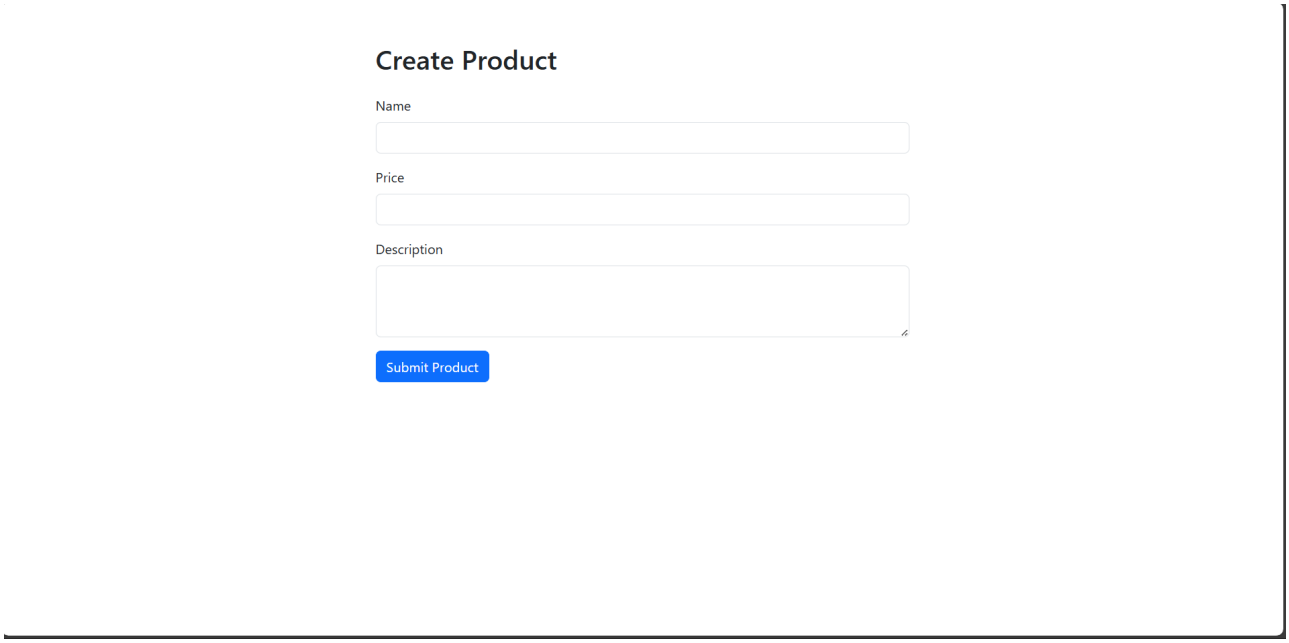
- Buat File Baru di Direktori resources/views/product/result.blade.php :
  Isi Dengan Code Berikut

resources > views > product > result.blade.php

```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>
```

- Langkah 7: Jalankan Aplikasi
  Jalankan Server Pengembangan
  `php php artisan serve`
  Akses aplikasi di:

http://127.0.0.1:8000/product/create




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

- Langkah 1: Buat dan Buka Proyek laravel
  laravel new todo-app-mysql
  cd todo-app-mysql
  code .
  - ketika ditanya default migration, maka pilih **NO**
  - Pastikan ekstensi(`extension=mysqli` & `extension=pdo_mysql`) mysql diaktifkan dalam file `php.ini`
- Langkah 2: Buat database hidupkan mysql dan buka cmd dan masuk(cd) ke folder bin kemudian ketik :
  `mysql -u root -p`
  `Create database tododb`
- Langkah 3: install depedency MySQL
  `composer require doctrine/dbal`
- Langkah 4: Konfigurasi file .env

```
23  DB_CONNECTION=mysql
24  DB_HOST=127.0.0.1
25  DB_PORT=3306
26  DB_DATABASE=tododb
27  DB_USERNAME=root
28  DB_PASSWORD=
```

  Kemudian bersihkan cache
  `php artisan config:clear`
- Langkah 5: buat migration table todos
  `php artisan make:migration create_todos_table`
  buka file baru yg dihasilkan dari perintah diatas dan isi :

database > migrations > 2025_11_03_015244_create_todos_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('todos', function (Blueprint $table) {
            $table->id();
            $table->string('task');
            $table->boolean('completed')->default(false);
            $table->timestamps();
        });
    }

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

Jalankan migrasi

```
php artisan migrate
```

- Langkah 6: buat seeder untuk data dummy

  ```
  php artisan make:seeder TodoSeeder
  ```

  buka file yg ada di direktori database/seeders/TodoSeeder.php dan perbarui :

database > seeders > TodoSeeder.php

```php
<?php

namespace Database\Seeders;

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

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 7: Buat model Todo

```
php artisan make:model Todo
```

Buka file dan perbarui

app > Models > Todo.php

```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'];
}
```

- Langkah 8: Buat TodoController untuk operasi CRUD
  `php artisan make:controller TodoController`
  ini akan membuat app/Http/Controllers/TodoController.php
  dan isikan file nya dengan :

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', 'Ta
    }

    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)
    {
```

app > Http > Controllers > TodoController.php

```php
    {
        $request->validate(['task' => 'required|string']);
        Todo::create(['task' => $request->task]);
        return redirect()->route('todos.index')->with('success', 'Ta
    }

    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', 'Ta
    }

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

- Langkah 9: Definisikan Route
  buka route/web.php dan isi:

routes > web.php

```php
<?php

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

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

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 10: Buat layout dasar
  - Buat direktori layouts di resources/views
    Kemudian, buat resources/views/admin/app.blade.php:
    Dan Isi dengan Code Berikut:

resources > views > layouts > app.blade.php

```php
<!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>
```

  - Buat direktori todos di resources/views Kemudian, buat :

- resources/views/admin/Index.blade.php: dan isi dengan :

resources > views > todos > index.blade.php

```blade
@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
```

- resources/views/admin/Create.blade.php: dan isi dengan :

resources > views > todos > create.blade.php

```blade
@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
```

- resources/views/admin/Edit.blade.php: dan isi dengan :

resources > views > todos > edit.blade.php

```blade
@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
```

- resources/views/admin/Show.blade.php: dan isi dengan :

```
resources > views > todos > show.blade.php
@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
```

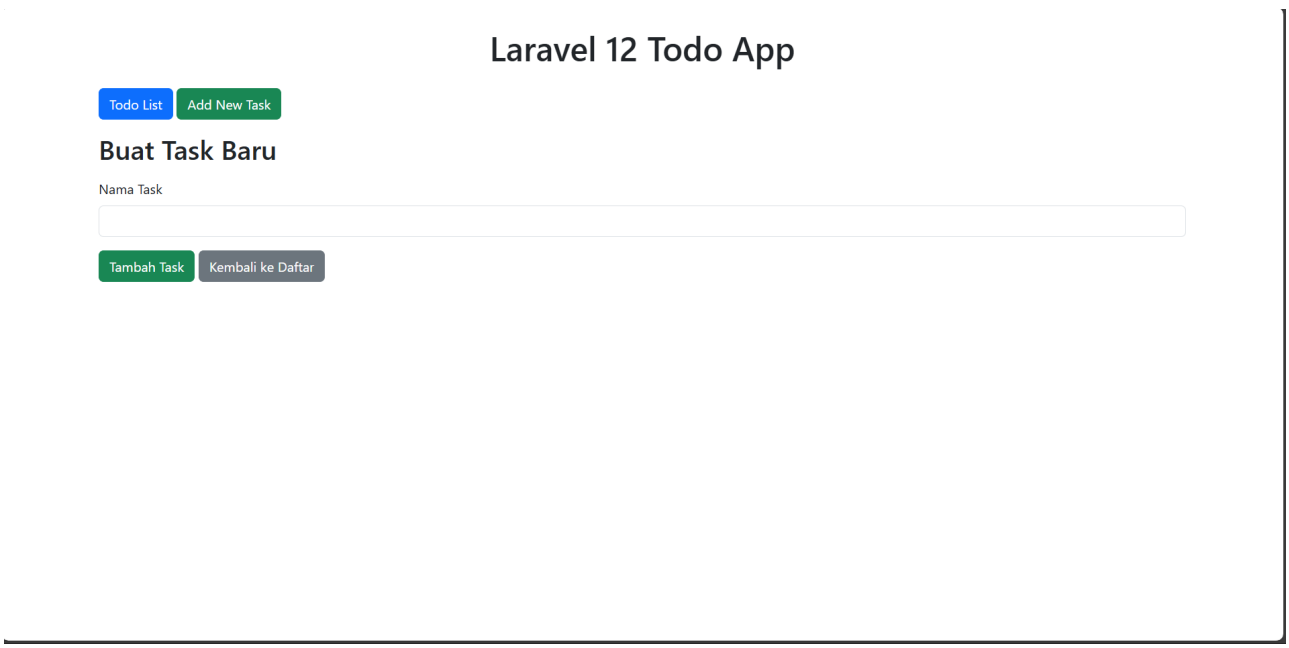
- Langkah 11: Menguji Aplikasi

  `php artisan serve`

  Buka browser dan kunjungi

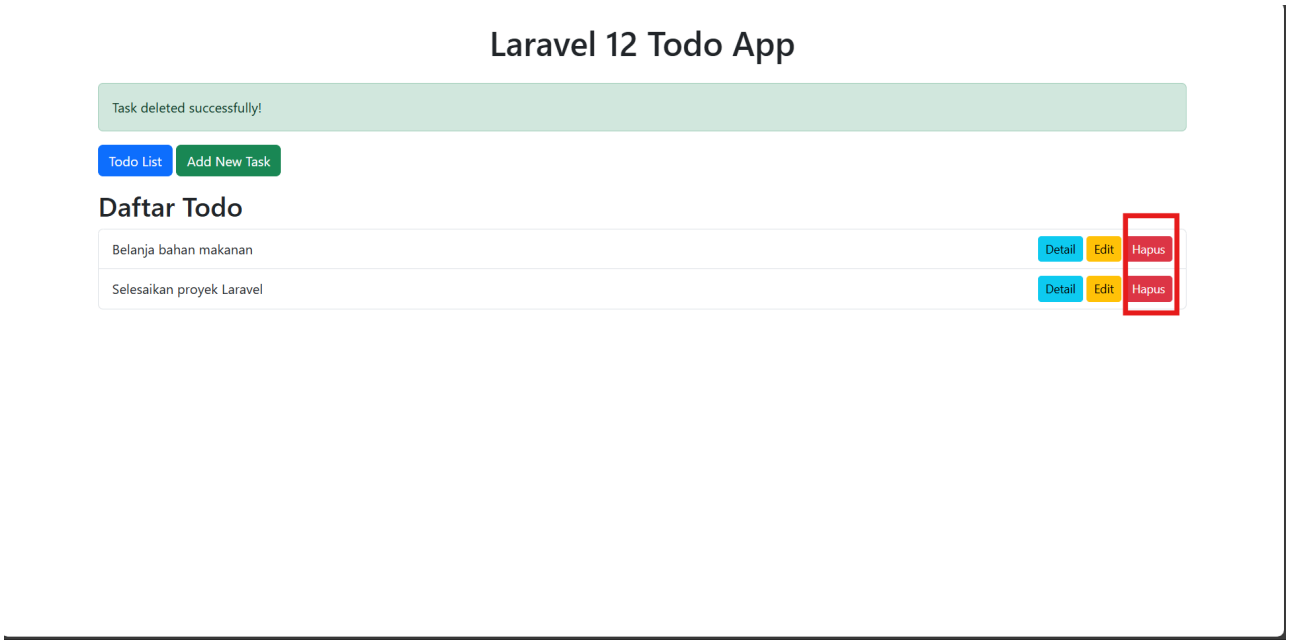
  http://127.0.0.1:8000

  Berikut adalah Hasilnya :


Laravel 12 Todo App
Todo List
Add New Task
Detail Task
Belanja bahan makanan
Status: Belum Selesai
Edit
Kembali ke Daftar


Laravel 12 Todo App
Todo List
Add New Task
Edit Task
Nama Task
Belanja bahan makanan
Update Task
Kembali ke Daftar





## 3. Hasil dan Pembahasan

Pada praktikum 6 ini, Anda telah berhasil:

- Membuat model Laravel menggunakan perintah php artisan make:model.
- Menghubungkan model dengan tabel database melalui konfigurasi otomatis dan konvensi Laravel.
- Menerapkan operasi CRUD menggunakan metode Eloquent seperti create(), find(), update(), dan delete().
- Menambahkan atribut $fillable untuk mengatur kolom yang boleh diisi melalui mass assignment.
- Mengimplementasikan relasi antar model, seperti one-to-many untuk menghubungkan data antar tabel.
- Menampilkan data dari model ke tampilan (view) menggunakan controller.
- Menguji fungsi Eloquent dengan menampilkan hasil query langsung pada halaman web.

Hasil praktikum menunjukkan bahwa penggunaan Model dan Eloquent ORM memudahkan proses pengelolaan data karena sintaksnya sederhana, mudah dibaca, dan tetap efisien tanpa perlu menulis query

SQL secara manual.

---

## 4. Kesimpulan

Praktikum ini berhasil menunjukkan penerapan Model dan Eloquent ORM dalam Laravel sebagai solusi efisien untuk mengelola data di database. Eloquent menyederhanakan proses manipulasi data dengan sintaks yang deklaratif dan mendukung berbagai relasi antar tabel. Melalui model, logika bisnis dapat dipisahkan dari tampilan, menjadikan aplikasi lebih terstruktur dan mudah dikembangkan. Dengan demikian, mahasiswa memahami pentingnya konsep Model dalam arsitektur MVC serta keunggulan Eloquent dalam implementasi berbasis objek.

---

## 5. Referensi


- chatgpt.com
- https://hackmd.io/@mohdrzu/HJWzYp7Reg


---