

# 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
1  <?php
2  namespace App\ViewModels;
3
4  class ProductViewModel
5  {
6      public string $name;
7      public float $price;
8      public string $description;
9
10     public function __construct(string $name = '', float $price = 0, string $description = '')
11     {
12         $this->name = $name;
13         $this->price = $price;
14         $this->description = $description;
15     }
16
17     public static function fromRequest(array $data): self
18     {
19         return new self(
20             $data['name'] ?? '',
21             (float)($data['price'] ?? 0),
22             $data['description'] ?? ''
23         );
24     }
25 }
26
```

- Langkah 3 : Buat controller  
`php artisan make:controller ProductController`  
ini akan membuat app/Http/Controllers/ProductController.php

dan isi dengan kode :

```
app > Http > Controllers > ProductController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\ViewModels\ProductViewModel;
7
8  class ProductController extends Controller
9  {
10     public function create()
11     {
12         return view('product.create');
13     }
14
15     public function result(Request $request)
16     {
17         $product = ProductViewModel::fromRequest($request->all());
18         return view('product.result', compact('product'));
19     }
20 }
21
```

- Langkah 4: Definisikan route

Buka Routes/Web.php Dan isi dengan code berikut

```
routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\ProductController;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10
11 Route::get('/product/create', [ProductController::class, 'create'])->name('product.create');
12 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
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Create Product</title>
5      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
6  </head>
7  <body class="container py-5">
8      <h2>Create Product (No Database)</h2>
9      <form method="POST" action="{{ route('product.result') }}>
10         @csrf
11         <div class="mb-3">
12             <label class="form-label">Name</label>
13             <input name="name" class="form-control" required>
14         </div>
15         <div class="mb-3">
16             <label class="form-label">Price</label>
17             <input name="price" type="number" step=".01" class="form-control" required>
18         </div>
19         <div class="mb-3">
20             <label class="form-label">Description</label>
21             <textarea name="description" class="form-control"></textarea>
22         </div>
23         <button type="submit" class="btn btn-primary">Submit Product</button>
24     </form>
25 </body>
26 </html>
```

- Buat File Baru di Direktori resources/views/product/result.blade.php :

Isi Dengan Code Berikut

```
resources > views > product > 🗄 result.blade.php
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Product Result</title>
5      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
6  </head>
7  <body class="container py-5">
8      <h2>Submitted Product Details</h2>
9      <ul class="list-group">
10         <li class="list-group-item"><strong>Name:</strong> {{ $product->name }}</li>
11         <li class="list-group-item"><strong>Price:</strong> ${{ number_format($product->price, 2) }}</li>
12         <li class="list-group-item"><strong>Description:</strong> {{ $product->description }}</li>
13     </ul>
14     <a href="{{ route('product.create') }}" class="btn btn-link mt-3">Submit Another Product</a>
15 </body>
16 </html>
```

- Langkah 5: Uji Rute

Mulai Php **Artisan serve**

Buka Browser Dan Kunjungi

<http://127.0.0.1:8000/product/create>

Create Product (No Database)

Name

Price

Description

## 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
1  <?php
2
3  namespace App\DTO;
4
5  class ProductDTO
6  {
7      public string $name;
8      public float $price;
9      public string $description;
10
11     public function __construct(string $name, float $price, string $description)
12     {
13         $this->name = $name;
14         $this->price = $price;
15         $this->description = $description;
16     }
17
18     public static function fromRequest(array $data): self
19     {
20         return new self(
21             $data['name'] ?? '',
22             (float)($data['price'] ?? 0),
23             $data['description'] ?? ''
24         );
25     }
26 }
27
```

- Langkah 3: Buat service layer

Buat ProductService.php di dalam direktori app/Services

lalu isi kode :

```
app > Services > ProductService.php
1  <?php
2
3  namespace App\Services;
4
5  use App\DTO\ProductDTO;
6
7  class ProductService
8  {
9      public function display(ProductDTO $product): array
10     {
11         return [
12             'name' => $product->name,
13             'price' => $product->price,
14             'description' => $product->description,
15         ];
16     }
17 }
```

- Langkah 4: Buat Sebuah Controller

```
php artisan make:controller ProductController
```

ini akan membuat app/Http/Controllers/ProductController.php. lalu isi kode :

```
app > Http > Controllers > ProductController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\DTO\ProductDTO;
7  use App\Services\ProductService;
8
9  class ProductController extends Controller
10 {
11     public function create()
12     {
13         return view('product.create');
14     }
15
16     public function result(Request $request)
17     {
18         $dto = ProductDTO::fromRequest($request->all());
19         $service = new ProductService();
20         $product = $service->display($dto);
21
22         return view('product.result', compact('product'));
23     }
24 }
```

- Langkah 5: Tambahkan route baru

Buka routes/web.php dan tambahkan:

```
routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\ProductController;
5
6  Route::get('/', function () {
7      return view('welcome');
8 });
9
10 Route::get('/product/create', [ProductController::class, 'create'])->name('product.create');
11 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
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Create Product DTO</title>
5      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
6  </head>
7  <body class="container py-5">
8      <div class="row justify-content-center">
9          <div class="col-md-6">
10         <h2 class="mb-4">Create Product</h2>
11         <form method="POST" action="{{ route('product.result') }}>
12             @csrf
13             <div class="mb-3">
14                 <label class="form-label">Name</label>
15                 <input name="name" class="form-control" required>
16             </div>
17             <div class="mb-3">
18                 <label class="form-label">Price</label>
19                 <input name="price" type="number" step="0.01" class="form-control" required>
20             </div>
21             <div class="mb-3">
22                 <label class="form-label">Description</label>
23                 <textarea name="description" class="form-control" rows="3"></textarea>
24             </div>
25             <button type="submit" class="btn btn-primary">Submit Product</button>
26         </form>
27     </div>
28 </div>
29 </body>
30 </html>
```

- Buat File Baru di Direktori resources/views/product/result.blade.php :

Isi Dengan Code Berikut

```
resources > views > product > 📄 result.blade.php
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Product Result</title>
5      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
6  </head>
7  <body class="container py-5">
8      <div class="row justify-content-center">
9          <div class="col-md-6">
10         <h2 class="mb-4">Product DTO Result</h2>
11         <div class="card">
12             <div class="card-header">
13                 <h5 class="card-title mb-0">Product Details</h5>
14             </div>
15             <ul class="list-group list-group-flush">
16                 <li class="list-group-item">
17                     <strong>Name:</strong> {{ $product['name'] }}<br>
18                 </li>
19                 <li class="list-group-item">
20                     <strong>Price:</strong> ${{ number_format($product['price'], 2) }}<br>
21                 </li>
22                 <li class="list-group-item">
23                     <strong>Description:</strong> {{ $product['description'] }}<br>
24                 </li>
25             </ul>
26         </div>
27         <a href="{{ route('product.create') }}" class="btn btn-secondary mt-3">Submit Another Product</a>
28     </div>
29 </div>
30 </body>
31 </html>
```

- Langkah 7: Jalankan Aplikasi

Jalankan Server Pengembangan

`php artisan serve`

Akses aplikasi di:

<http://127.0.0.1:8000/product/create>

Create Product

Name

Price

Description

Submit Product

### 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 dependency 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
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      public function up()
9      {
10         Schema::create('todos', function (Blueprint $table) {
11             $table->id();
12             $table->string('task');
13             $table->boolean('completed')->default(false);
14             $table->timestamps();
15         });
16     }
17
18     public function down()
19     {
20         Schema::dropIfExists('todos');
21     }
22 };
```

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
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8  use Carbon\Carbon;
9
10 class TodoSeeder extends Seeder
11 {
12     public function run()
13     {
14         DB::table('todos')->insert([
15             [
16                 'task' => 'Belanja bahan makanan',
17                 'completed' => false,
18                 'created_at' => Carbon::now(),
19                 'updated_at' => Carbon::now()
20             ],
21             [
22                 'task' => 'Beli buah-buahan',
23                 'completed' => false,
24                 'created_at' => Carbon::now(),
25                 'updated_at' => Carbon::now()
26             ],
27             [
28                 'task' => 'Selesaikan proyek Laravel',
29                 'completed' => true,
30                 'created_at' => Carbon::now(),
31                 'updated_at' => Carbon::now()
32             ],
33         ]);
34     }
35 }
```

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

- 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 :
```

<pre>app &gt; Http &gt; Controllers &gt; TodoController.php 1  &lt;?php 2 3  namespace App\Http\Controllers; 4 5  use Illuminate\Http\Request; 6  use App\Models\Todo; 7 8  class TodoController extends Controller 9  { 10     public function index() 11     { 12         \$todos = Todo::all(); 13         return view('todos.index', compact('todos')); 14     } 15 16     public function create() 17     { 18         return view('todos.create'); 19     } 20 21     public function store(Request \$request) 22     { 23         \$request-&gt;validate(['task' =&gt; 'required string']); 24         Todo::create(['task' =&gt; \$request-&gt;task]); 25         return redirect()-&gt;route('todos.index')-&gt;with('success', 'Ta 26     } 27 28     public function show(Todo \$todo) 29     { 30         return view('todos.show', compact('todo')); 31     } 32 33     public function edit(Todo \$todo) 34     { 35         return view('todos.edit', compact('todo')); 36     } 37 38     public function update(Request \$request, Todo \$todo) 39     { 40         \$request-&gt;validate(['task' =&gt; 'required string']); 41         \$todo-&gt;update(['task' =&gt; \$request-&gt;task]); 42         return redirect()-&gt;route('todos.index')-&gt;with('success', 'Ta 43     } 44 45     public function destroy(Todo \$todo) 46     { 47         \$todo-&gt;delete(); 48         return redirect()-&gt;route('todos.index')-&gt;with('success', 'Ta 49     } 50 }</pre>	<pre>app &gt; Http &gt; Controllers &gt; TodoController.php 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 }</pre>
--	--

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

```
routes > 🗃 web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\TodoController;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10 Route::get('/', [TodoController::class, 'index'])->name('todos.index');
11 Route::get('/todos/create', [TodoController::class, 'create'])->name('todos.create');
12 Route::post('/todos', [TodoController::class, 'store'])->name('todos.store');
13 Route::get('/todos/{todo}', [TodoController::class, 'show'])->name('todos.show');
14 Route::get('/todos/{todo}/edit', [TodoController::class, 'edit'])->name('todos.edit');
15 Route::patch('/todos/{todo}', [TodoController::class, 'update'])->name('todos.update');
16 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
1  !DOCTYPE html
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>@yield('title', 'Todo App')</title>
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
8  </head>
9  <body class="container mt-4">
10     <h1 class="text-center mb-4">Laravel 12 Todo App</h1>
11
12     @if(session('success'))
13         <div class="alert alert-success">{{ session('success') }}</div>
14     @endif
15
16     <nav class="mb-3">
17         <a href="{{ route('todos.index') }}" class="btn btn-primary">Todo List</a>
18         <a href="{{ route('todos.create') }}" class="btn btn-success">Add New Task</a>
19     </nav>
20
21     @yield('content')
22
23     </body>
24
25 </html>
```

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

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

```
resources > views > todos > index.blade.php
1  @extends('layouts.app')
2
3  @section('title', 'Daftar Todo')
4
5  @section('content')
6      <h2>Daftar Todo</h2>
7
8      <ul class="list-group">
9          @foreach($todos as $todo)
10             <li class="list-group-item d-flex justify-content-between align-items-center">
11                 {{ $todo->task }}
12                 <div>
13                     <form action="{{ route('todos.show', $todo->id) }}" method="GET" class="d-inline">
14                         <button type="submit" class="btn btn-info btn-sm">Detail</button>
15                     </form>
16                     <form action="{{ route('todos.edit', $todo->id) }}" method="GET" class="d-inline">
17                         <button type="submit" class="btn btn-warning btn-sm">Edit</button>
18                     </form>
19                     <form action="{{ route('todos.destroy', $todo->id) }}" method="POST" class="d-inline">
20                         @csrf
21                         @method('DELETE')
22                         <button class="btn btn-danger btn-sm">Hapus</button>
23                     </form>
24                 </div>
25             </li>
26         @endforeach
27     </ul>
28 @endsection
```

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

```
resources > views > todos > create.blade.php
1  @extends('layouts.app')
2
3  @section('title', 'Buat Task Baru')
4
5  @section('content')
6      <h2>Buat Task Baru</h2>
7
8      <form action="{{ route('todos.store') }}" method="POST" class="mt-3">
9          @csrf
10         <div class="mb-3">
11             <label for="task" class="form-label">Nama Task</label>
12             <input type="text" name="task" id="task" class="form-control" required>
13         </div>
14         <button type="submit" class="btn btn-success">Tambah Task</button>
15         <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
16     </form>
17
18 @endsection
```

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

```
resources > views > todos > edit.blade.php
1  @extends('layouts.app')
2  @section('title', 'Edit Task')
3
4  @section('content')
5      <h2>Edit Task</h2>
6
7      <form action="{{ route('todos.update', $todo->id) }}" method="POST" class="mt-3">
8          @csrf
9          @method('PATCH')
10         <div class="mb-3">
11             <label for="task" class="form-label">Nama Task</label>
12             <input type="text" name="task" id="task" class="form-control" value="{{ $todo->task }}" required>
13         </div>
14         <button type="submit" class="btn btn-warning">Update Task</button>
15         <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
16     </form>
17
18 @endsection
```

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

```

resources > views > todos > 📄 show.blade.php
1  @extends('layouts.app')
2  @section('title', 'Detail Task')
3  @section('content')
4      <h2>Detail Task</h2>
5
6      <div class="card mt-3">
7          <div class="card-body">
8              <h5 class="card-title">{{ $todo->task }}</h5>
9              <p class="card-text">Status: {{ $todo->completed ? 'Selesai' : 'Belum Selesai' }}</p>
10             <a href="{{ route('todos.edit', $todo->id) }}" class="btn btn-warning">Edit</a>
11             <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
12         </div>
13     </div>
14 @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

The screenshot shows the Laravel 12 Todo App dashboard. At the top, there are two buttons: "Todo List" (blue) and "Add New Task" (green). Below this is a section titled "Daftar Todo" (Todo List). It displays three tasks in a table:

Belanja bahan makanan	<a href="#">Detail</a>	<a href="#">Edit</a>	<a href="#">Hapus</a>
Beli buah-buahan	<a href="#">Detail</a>	<a href="#">Edit</a>	<a href="#">Hapus</a>
Selesaikan proyek Laravel	<a href="#">Detail</a>	<a href="#">Edit</a>	<a href="#">Hapus</a>

## Laravel 12 Todo App

The screenshot shows the Laravel 12 Todo App form for creating a new task. At the top, there are two buttons: "Todo List" (blue) and "Add New Task" (green). Below this is a section titled "Buat Task Baru" (Create New Task). It has a text input field labeled "Nama Task" and two buttons at the bottom: "Tambah Task" (green) and "Kembali ke Daftar" (green).

## 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](#)

The screenshot shows a web application titled "Laravel 12 Todo App". At the top, there is a green banner with the message "Task deleted successfully!". Below the banner, there are two buttons: "Todo List" (blue) and "Add New Task" (green). The main content area is titled "Daftar Todo" and contains a table with two rows of tasks:

Task	Action
Belanja bahan makanan	Detail Edit Hapus
Selesaikan proyek Laravel	Detail Edit Hapus

A red box highlights the "Hapus" button for the second task row. In the bottom half of the screenshot, there is another identical view of the "Laravel 12 Todo App" interface, but the "Todo List" button is highlighted with a red box.

### 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>
-