

# Laporan Modul 2: Laravel Fundamentasl

---

**Mata Kuliah:** Workshop Web Lanjut

**Nama:** Nayla Mutia Silvia Dina

**NIM:** 2024573010106

**Kelas:** TI-2B

---

## Abstrak

Laporan ini membahas konsep dasar framework Laravel, termasuk pembuatan route, controller, dan Blade view, serta penerapan validasi input. Laporan ini bertujuan untuk memberikan pemahaman mengenai alur kerja Laravel dalam mengelola request dan response, serta memperkenalkan implementasi konsep arsitektur MVC melalui pembuatan aplikasi kalkulator sederhana.

---

## 1. Dasar Teori

1.1 MVC (Model, View, Controller) MVC adalah pola arsitektur yang memisahkan aplikasi menjadi tiga komponen utama:

1. Model : bagian yang mengatur data, misalnya mengambil data dari database, menyimpan data baru, atau mengubah data yang sudah ada.
2. View : merupakan bagian yang menampilkan informasi ke pengguna (antarmuka pengguna).
3. Controller : bagian yang menjadi penghubung antara Model dan View. Controller menerima permintaan dari pengguna, mengambil data dari Model, lalu mengirimkan data tersebut ke View untuk ditampilkan.

Dengan pembagian ini, kode menjadi lebih teratur, mudah diperbaiki, dan dikembangkan.

1.2 Routing di Laravel Routing adalah mekanisme untuk menentukan bagaimana aplikasi merespons sebuah request. Di Laravel, routing didefinisikan pada file `routes/web.php` menggunakan metode seperti `Route::get()` atau `Route::post()`.

1.3 Middleware Middleware bertindak sebagai perantara antara request dan response. Fungsinya antara lain untuk autentikasi, logging, atau memodifikasi request sebelum diteruskan ke controller.

1.4 Request dan Response di Laravel Laravel memproses request dengan mencocokkan URL ke route yang sesuai, menjalankan middleware, memanggil controller yang ditentukan, dan akhirnya mengirim response kembali ke browser berupa view atau data.

1.5 Peran Controller dan View Controller menangani logika aplikasi, memproses input dari user, dan mengirimkan data ke view. View bertanggung jawab menampilkan data dalam format HTML yang mudah dibaca pengguna.

1.6 Fungsi Blade Templating Engine Blade adalah template engine Laravel yang memungkinkan penggunaan sintaks sederhana seperti `{{ $variable }}`, serta mendukung layout, komponen, dan control structure seperti `@if` atau `@foreach` untuk membuat tampilan lebih dinamis.

---

## 2. Langkah-Langkah Praktikum

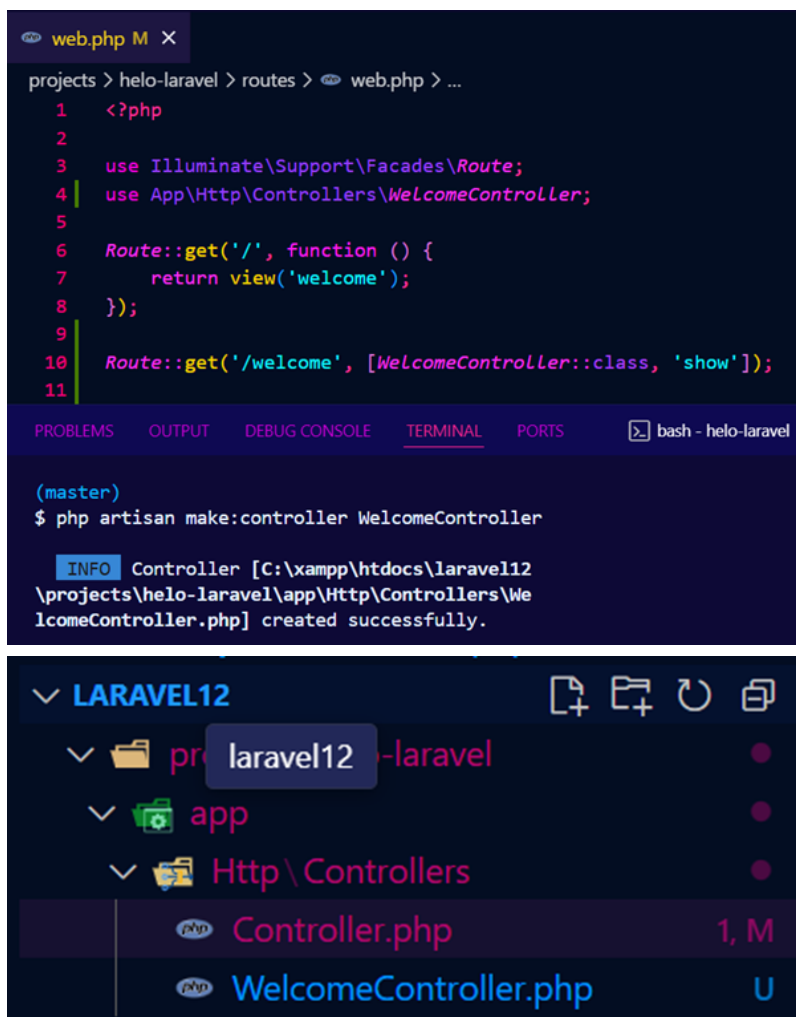
### 2.1 Praktikum 1 – Route, Controller, dan Blade View

- Menambahkan route pada routes/web.php Untuk menambahkan Route, tambahkan kode :  
Route::get('/welcome', [WelcomeController::class, 'index']);



```
web.php M X
projects > helo-laravel > routes > web.php > ...
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\WelcomeController;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10 Route::get('/welcome', [WelcomeController::class, 'show']);
11
12
```

- Membuat controller WelcomeController Buat file WelcomeController.php di folder app/Http/Controllers/



```
web.php M X
projects > helo-laravel > routes > web.php > ...
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\WelcomeController;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10 Route::get('/welcome', [WelcomeController::class, 'show']);
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - helo-laravel

```
(master)
$ php artisan make:controller WelcomeController

INFO Controller [C:\xampp\htdocs\laravel12\projects\helo-laravel\app\Http\Controllers>WelcomeController.php] created successfully.
```

LARAVEL12

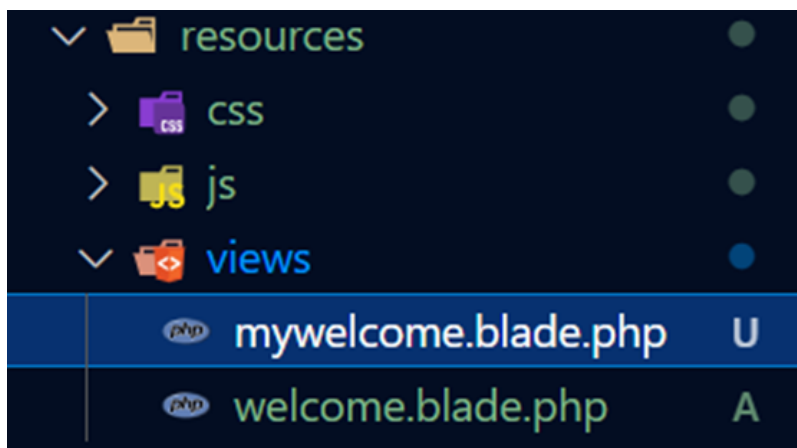
- laravel12 -laravel
- app
- Http\ Controllers
  - Controller.php 1, M
  - WelcomeController.php U

Kemudian isi:

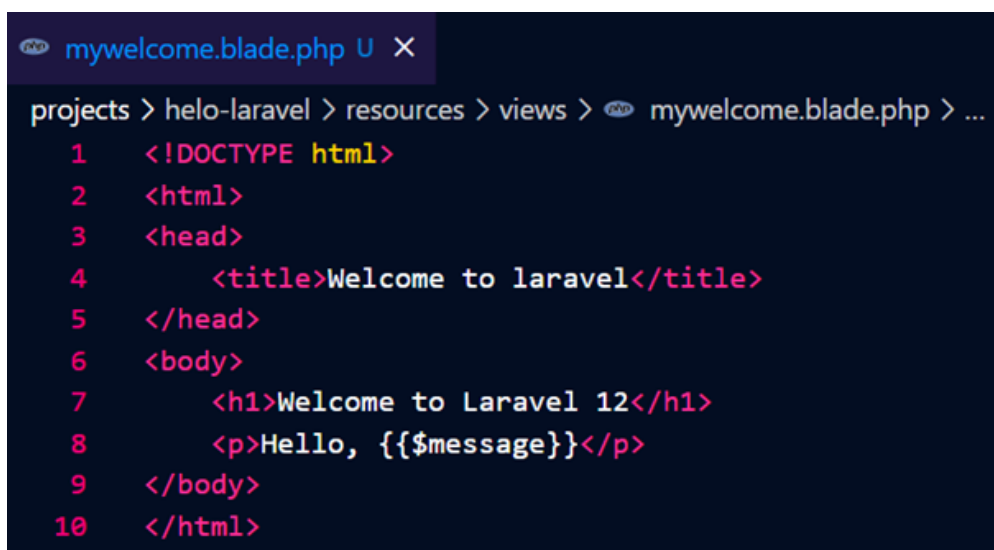


```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class WelcomeController extends Controller
8 {
9     public function show()
10     {
11         $message = "Welcome to Laravel!";
12         return view('mywelcome', ['message' => $message]);
13     }
14 }
```

- Membuat view mywelcome.blade.php Buka resources/views kemudian buat file dengan nama mywelcome.blade.php



isi sederhana seperti:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Welcome to laravel</title>
5 </head>
6 <body>
7     <h1>Welcome to Laravel 12</h1>
8     <p>Hello, {{$message}}</p>
9 </body>
10 </html>
```

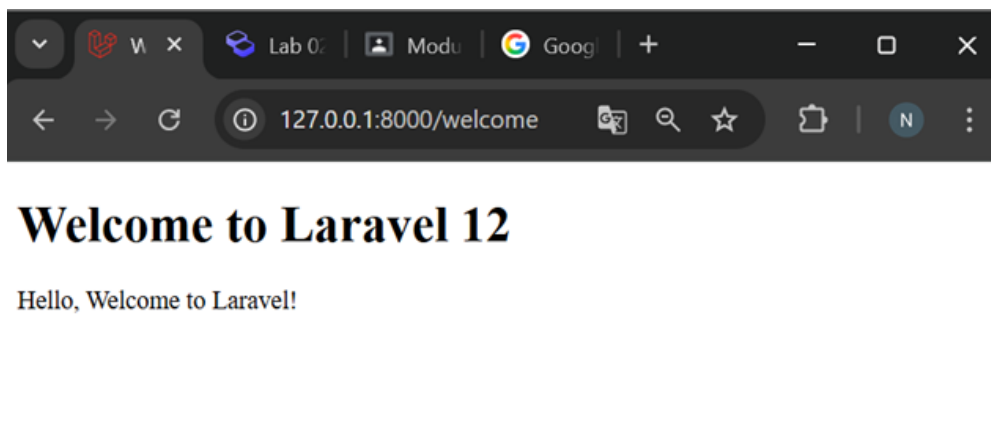
- Menjalankan aplikasi dan menunjukkan hasil di browser Buka terminal dan jalankan perintah : php artisan serve

```
HP ProBook 440 G7@DESKTOP-C8T2EVH MINGW64 /c:/xampp/htdocs/
laravel12/projects/helo-laravel (master)
$ php artisan serve

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```

buka browser dan akses <http://127.0.0.1:8000/welcome> untuk melihat hasilnya



## 2.2 Praktikum 2 – Membuat Aplikasi Sederhana "Calculator"

- Menambahkan route untuk kalkulator Tambahkan kode berikut di routes/web.php:  
Route::get('/calculator', [CalculatorController::class, 'index']); Route::post('/calculator', [CalculatorController::class, 'calculate']);

```
web.php 2 U x
projects > calculator > routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\CalculatorController;
5
6  // Route pertama: tampilkan form kalkulator
7  Route::get('/calculator', [CalculatorController::class, 'index']);
8
9  // Route kedua: proses data dari form
10 Route::post('/calculator', [CalculatorController::class, 'calculate'])->name('calculator.calculate');
11
```

- Membuat controller CalculatorController Buat file CalculatorController.php di folder app/Http/Controllers/

```
HP ProBook 440 G7@DESKTOP-C8T2EVH MINGW64 /c:/xampp/htdocs/laravel12/projects (master)
• $ cd /c:/xampp/htdocs/laravel12/projects/calculator

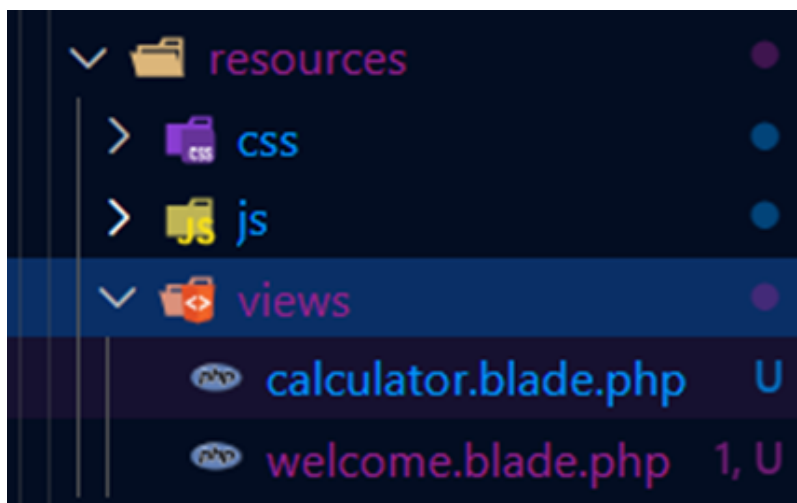
HP ProBook 440 G7@DESKTOP-C8T2EVH MINGW64 /c:/xampp/htdocs/laravel12/projects/calculator (master)
• $ php artisan make:controller CalculatorController

INFO Controller [C:\xampp\htdocs\laravel12\projects\calculator\app\Http\Controllers\CalculatorController.php]
created successfully.
```

Kemudian isi file seperti berikut:

```
CalculatorController.php U X
projects > calculator > app > Http > Controllers > CalculatorController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class CalculatorController extends Controller
8  {
9      public function index()
10     {
11         return view('calculator');
12     }
13
14     public function calculate(Request $request)
15     {
16         $validated = $request->validate([
17             'number1' => 'required|numeric',
18             'number2' => 'required|numeric',
19             'operator' => 'required|in:add,sub,mul,div'
20         ]);
21
22         $result = match ($validated['operator']) {
23             'add' => $validated['number1'] + $validated['number2'],
24             'sub' => $validated['number1'] - $validated['number2'],
25             'mul' => $validated['number1'] * $validated['number2'],
26             'div' => $validated['number1'] != 0 ? ($validated['number1'] / $validated['number2']) : 'Error : Division by 0',
27         };
28
29         return view('calculator', [
30             'result' => $result,
31             'number1' => $validated['number1'],
32             'number2' => $validated['number2'],
33             'operator' => $validated['operator'],
34         ]);
35     }
36 }
37
38
```

- Menambahkan view calculator.blade.php Buat file resources/views/calculator.blade.php



Kemudian isi file seperti berikut:

```
calculator.blade.php U X
projects > calculator > resources > views > calculator.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Laravel Calculator</title>
5 </head>
6 <body>
7 <h1>Kalkulator Sederhana</h1>
8
9 @if ($errors->any())
10 <div style="color: red;">
11 <ul>
12 @foreach ($errors->all() as $error)
13 <li>{{ $error }}</li>
14 @endforeach
15 </ul>
16 </div>
17 @endif
18
19 <form method="POST" action="{{ route('calculator.calculate') }}">
20 @csrf
21 <input type="number" name="number1" value="{{ old('number1', $number1 ?? '') }}" placeholder="Angka Pertama" required>
22
23 <select name="operator" required>
24 <option value="add" {{ ($operator ?? '') == 'add' ? 'selected' : '' }}></option>
25 <option value="sub" {{ ($operator ?? '') == 'sub' ? 'selected' : '' }}></option>
26 <option value="mul" {{ ($operator ?? '') == 'mul' ? 'selected' : '' }}></option>
27 <option value="div" {{ ($operator ?? '') == 'div' ? 'selected' : '' }}></option>
28 </select>
29
30 <input type="number" name="number2" value="{{ old('number2', $number2 ?? '') }}" placeholder="Angka Kedua" required>
31 <button type="submit">Hitung</button>
32 </form>
33
34 @isset($result)
35 <h3>Hasil: {{ $result }}</h3>
36 @endisset
37 </body>
38 </html>
```

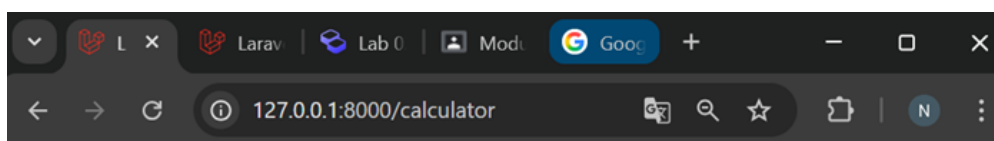
- Menjalankan aplikasi dan coba dengan beberapa input berbeda Jalankan perintah php artisan serve

```
HP ProBook 440 G7@DESKTOP-C8T2EVH MINGW64 /c:/xampp/htdocs/laravel12/projects
/calculator (master)
$ php artisan serve

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```

Buka <http://127.0.0.1:8000/calculator>

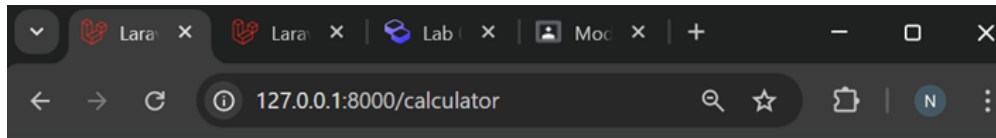


## Kalkulator Sederhana

Angka Pertama  +  Angka Kedua

Masukkan angka, pilih operator, lalu klik tombol Hitung:

- Operator Penjumlahan

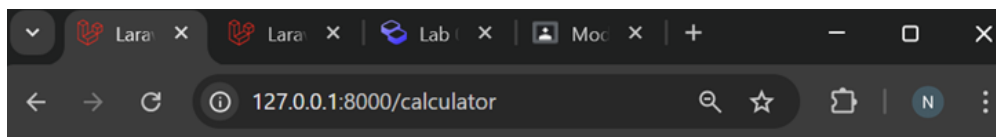


## Kalkulator Sederhana

2  +  3

Hasil: 5

### 2. Operator Pengurangan

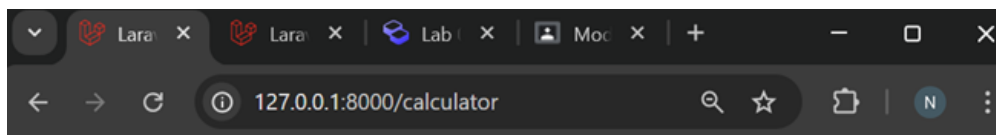


## Kalkulator Sederhana

5  -  1

Hasil: 4

### 3. Operator Perkalian

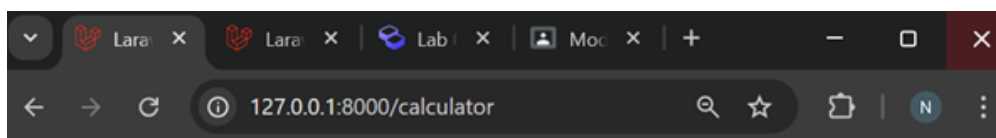


## Kalkulator Sederhana

2  \*  3

Hasil: 6

### 4. Operator Pembagian

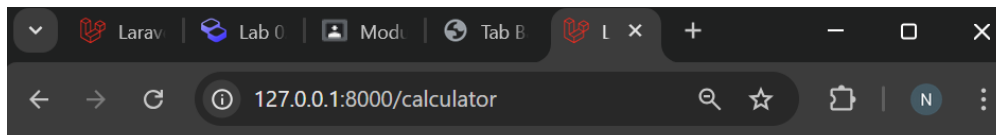


## Kalkulator Sederhana

15  /  5

Hasil: 3

### 5. Pembagian dengan 0



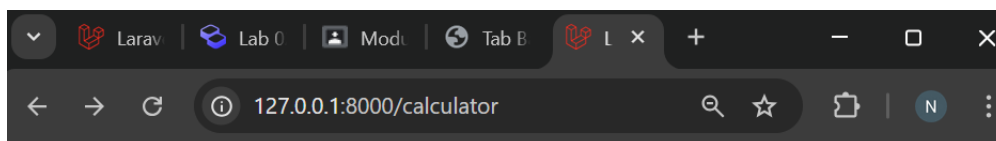
## Kalkulator Sederhana

0 / 6 Hitung

**Hasil: Error : Division by 0**

Jika pengguna melakukan pembagian dengan angka nol, aplikasi akan menampilkan pesan "Error: Division by 0" pada halaman, sehingga tidak terjadi internal server error.

### 6. Menginput String



## Kalkulator Sederhana

6 / e Hitung

Masukkan nomor.

Jika pengguna mencoba memasukkan karakter non-angka, browser akan menampilkan peringatan "Masukkan nomor." dan input tidak diproses.

---

## 3. Hasil dan Pembahasan

Hasil dari praktikum menunjukkan bahwa aplikasi berjalan sesuai harapan. Halaman welcome dapat ditampilkan melalui route yang sudah dibuat. Pada aplikasi kalkulator, setiap input angka dan operator dapat diproses oleh controller dan ditampilkan pada view.

Jika terjadi kesalahan input, seperti memasukkan nilai non-numerik (misalnya huruf), browser akan menampilkan peringatan "Masukkan nomor." sehingga input tidak bisa dikirim. Sedangkan jika dilakukan pembagian dengan angka nol, aplikasi tidak mengalami error, tetapi menampilkan pesan "Error : Division by 0" pada halaman hasil. Hal ini membantu pengguna mengetahui adanya kesalahan tanpa membuat aplikasi berhenti.

Validasi input di Laravel bekerja dengan memeriksa data yang dikirim pengguna sebelum dijalankan logika perhitungannya. Jika data tidak sesuai aturan (misalnya bukan angka atau operator yang tidak valid), Laravel otomatis mengembalikan pesan error dan tidak melanjutkan proses perhitungan.

Peran komponen MVC dalam program:



Route bertugas mengarahkan request ke controller yang sesuai. Pada praktikum ini, kedua route memastikan halaman kalkulator dapat diakses dan data yang dikirim dari form diproses oleh CalculatorController.

Controller berperan dalam menangani logika aplikasi, memproses input, dan mengembalikan hasil ke view. Pada praktikum, kode tersebut memvalidasi input, menghitung hasil, dan mengirim data ke view.

View berperan dalam menampilkan form input dan hasil perhitungan kepada pengguna. Pada praktikum ini berguna untuk menampilkan hasil perhitungan atau pesan error secara rapi di halaman.

---

## 4. Kesimpulan

Berdasarkan praktikum yang dilakukan, dapat disimpulkan bahwa Laravel mempermudah pengembangan aplikasi web dengan pendekatan MVC. Pemisahan antara Route, Controller, dan View membuat kode lebih terstruktur, mudah dipahami, dan mudah dipelihara.

Penerapan pada aplikasi kalkulator sederhana menunjukkan bahwa setiap komponen memiliki peran penting: Route mengarahkan permintaan pengguna, Controller memproses logika perhitungan dan validasi, sedangkan View menampilkan hasil atau pesan error dengan jelas. Selain itu, mekanisme validasi input Laravel membantu mencegah kesalahan perhitungan seperti memasukkan data non-numerik atau pembagian dengan nol sehingga aplikasi tetap berjalan stabil.

---

## 5. Referensi

[1] "Modul 2 - Laravel Fundamentals" — <https://hackmd.io/@mohdrzu/B1zwKEK5xe>

[2] "Building MVC Applications in PHP Laravel: Part 2" — <https://www.codemag.com/Article/2207041/Building-MVC-Applications-in-PHP-Laravel-Part-2>

[3] "Cara menggunakan Laravel bagi pemula" — [https://www.hostinger.com/id/tutorial/cara-menggunakan-laravel?utm\\_medium=ppc&utm\\_campaign=Generic-Tutorials-DSA](https://www.hostinger.com/id/tutorial/cara-menggunakan-laravel?utm_medium=ppc&utm_campaign=Generic-Tutorials-DSA)

---