

# LATIHAN PRAKTIKUM PENGEMBANGAN APLIKASI WEB

BAB : *FRAMEWORK* LARAVEL (2)  
NAMA : DANI ADRIAN  
NIM : 225150201111009  
ASISTEN : - IBAR HUTTAQI SULTHON  
          - MUHAMMAD AMMAR  
TANGGAL PRAKTIKUM : 16/05/2024

## 1. Subdirektori View

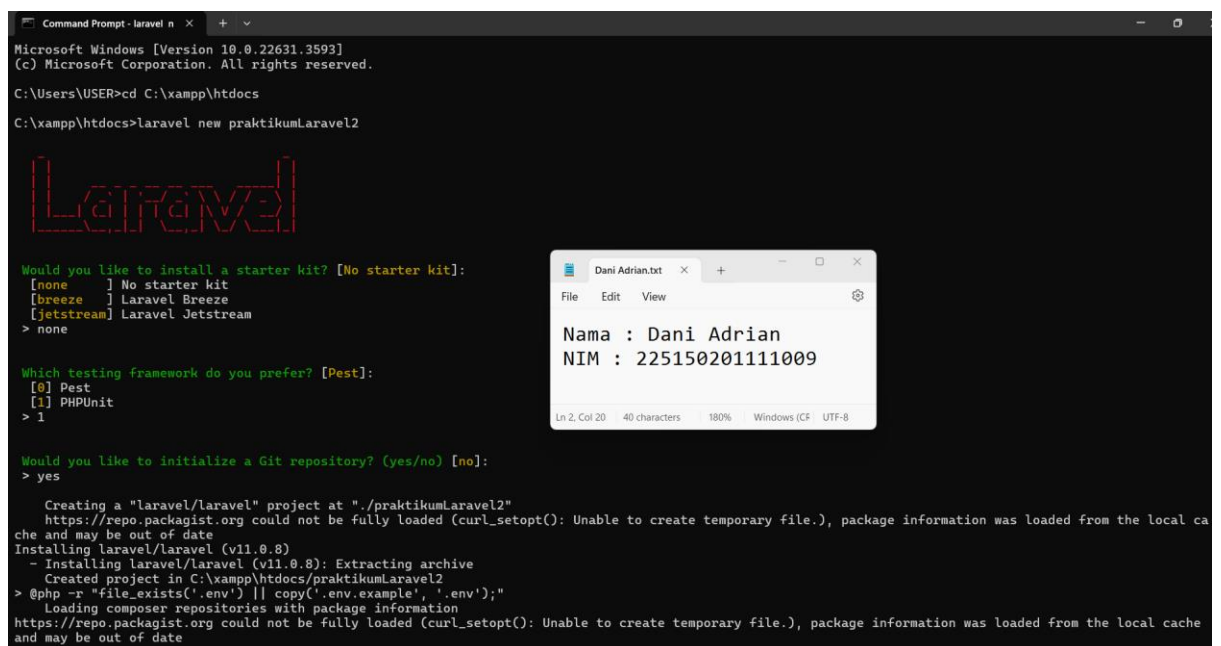
### LATIHAN 1

#### A. Soal

Buatlah sebuah proyek baru Laravel dengan perintah sbb:

```
laravel new praktikumLaravel2
```

#### B. Screenshoot



```
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>cd C:\xampp\htdocs
C:\xampp\htdocs>laravel new praktikumLaravel2

Laravel

Would you like to install a starter kit? [No starter kit]:
[none] ] No starter kit
[breeze] ] Laravel Breeze
[jetstream] ] Laravel Jetstream
> none

Which testing framework do you prefer? [Pest]:
[0] Pest
[1] PHPUnit
> 1

Would you like to initialize a Git repository? (yes/no) [no]:
> yes

Creating a "laravel/laravel" project at "./praktikumLaravel2"
https://repo.packagist.org could not be fully loaded (curl_setopt(): Unable to create temporary file.), package information was loaded from the local cache and may be out of date
Installing laravel/laravel (v11.0.8)
- Installing laravel/laravel (v11.0.8): Extracting archive
Created project in C:\xampp\htdocs\praktikumLaravel2
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
https://repo.packagist.org could not be fully loaded (curl_setopt(): Unable to create temporary file.), package information was loaded from the local cache and may be out of date
```

#### C. Syntax

--

#### D. Penjelasan

Dengan menggunakan perintah `laravel new praktikumLaravel2`, Laravel akan membuat struktur dasar aplikasi web, termasuk direktori untuk kontroler, model, view, dan rute sehingga akan memberikan kerangka kerja yang terorganisir dan standar untuk memulai developpe.

Selain itu, perintah ini juga menginstal semua dependensi yang diperlukan melalui Composer, termasuk kerangka kerja Laravel itu sendiri serta paket-paket pendukung lainnya yang dibutuhkan untuk menjalankan aplikasi Laravel.

## LATIHAN 2

### A. Soal

Pada folder `resources/views`, buatlah direktori baru dengan nama `praktikum2`. Kemudian pada direktori tersebut buatlah sebuah file Blade dengan nama `index.blade.php` dan isi dengan kode sbb:

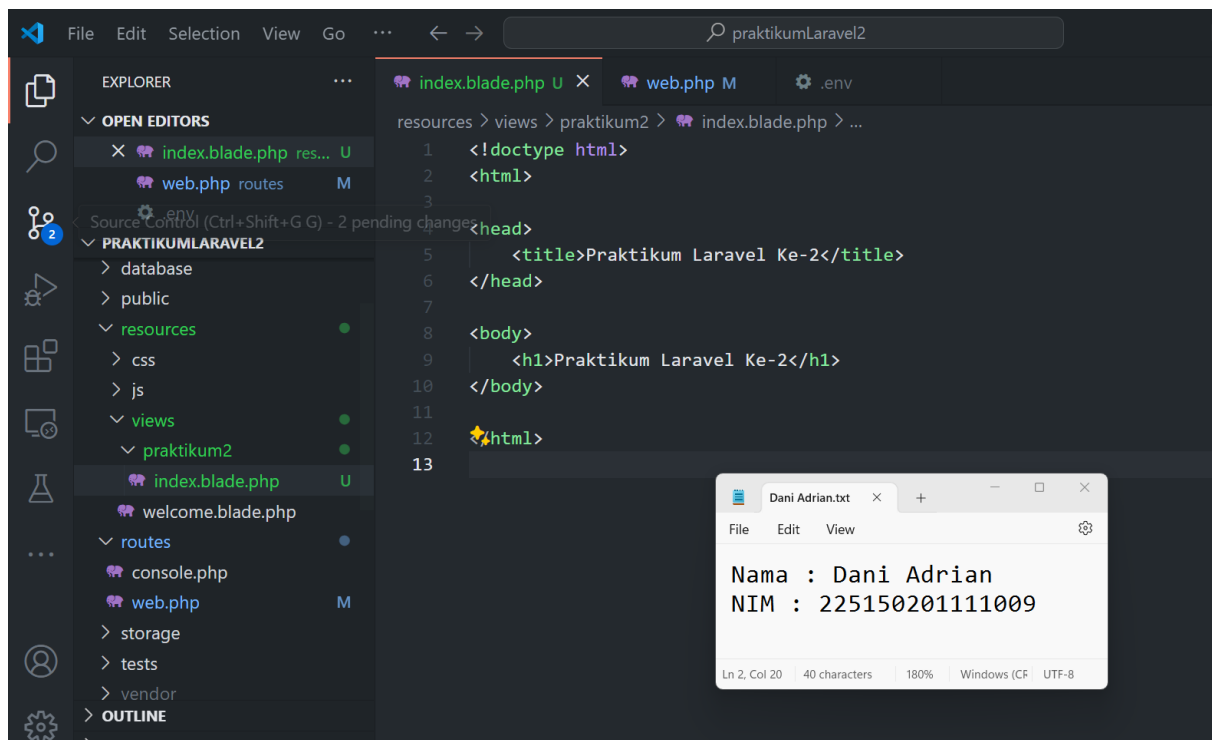
```
<!doctype html>
<html>

<head>
    <title>Praktikum Laravel Ke-2</title>
</head>

<body>
    <h1>Praktikum Laravel Ke-2</h1>
</body>

</html>
```

### B. Screenshoot



### C. Syntax

--	--

### D. Penjelasan

File `index.blade.php` merupakan file tampilan (view) yang akan di-render ketika pengguna mengakses rute tertentu dan merupakan bagian dari arsitektur MVC (Model-View-Controller) Laravel.

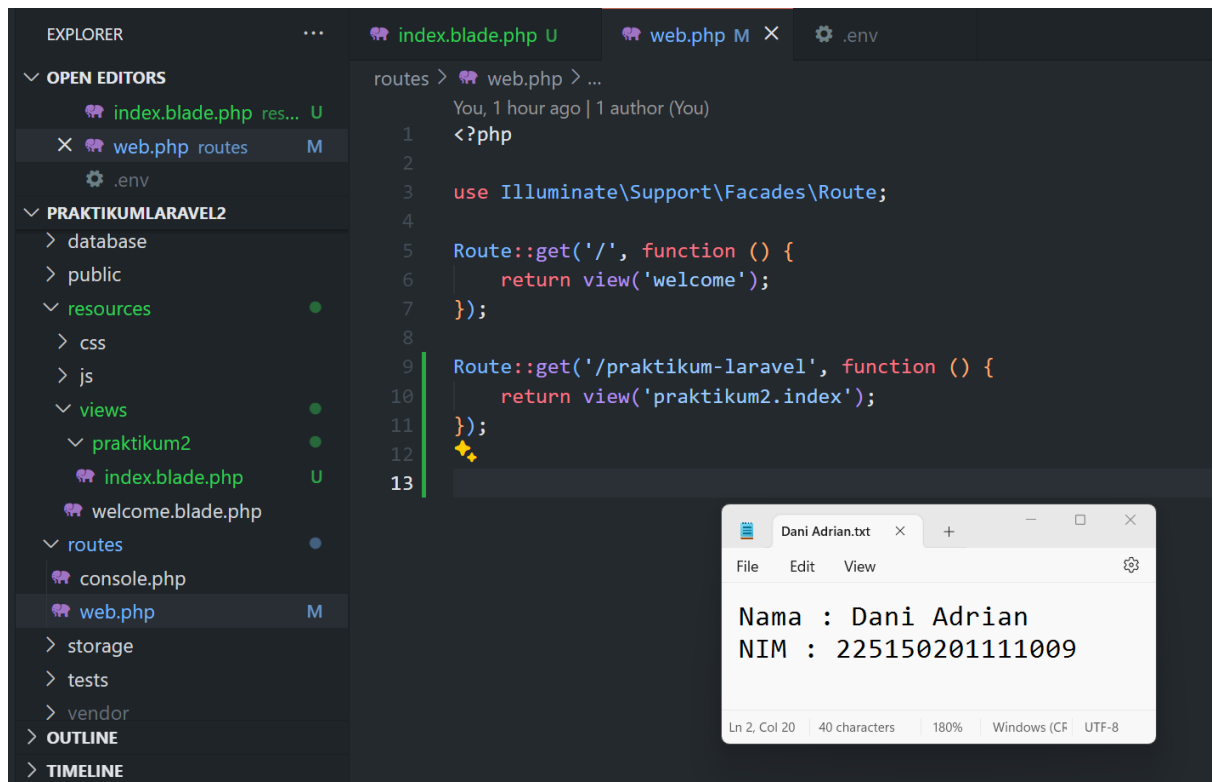
## LATIHAN 3

### A. Soal

Pada file `web.php` dalam folder `routes`, tambahkan rute sbb:

```
Route::get('/praktikum-laravel', function () {  
    return view('praktikum2.index');  
});
```

### B. Screenshoot



### C. Syntax

<?php
-------

```
use Illuminate\Support\Facades\Route;

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

Route::get('/praktikum-laravel', function () {
    return view('praktikum2.index');
});
```

#### D. Penjelasan

Rute ini mendefinisikan bahwa aplikasi akan merespon permintaan HTTP GET ke URL /praktikum-laravel.

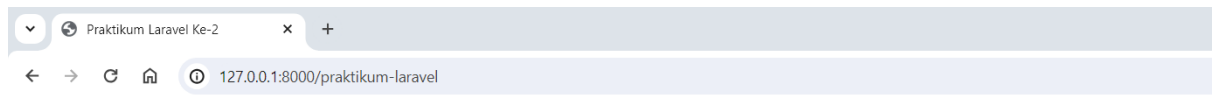
Perintah `return view('praktikum2.index');` menunjukkan bahwa ketika rute /praktikum-laravel diakses, Laravel akan mengembalikan tampilan yang berada di file `resources/views/praktikum2/index.blade.php`.

## LATIHAN 4

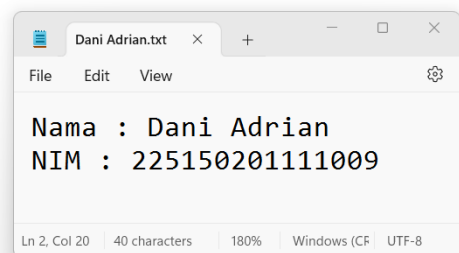
#### A. Soal

Apakah URL untuk mengakses halaman tersebut?

#### B. Screenshoot



### Praktikum Laravel Ke-2



#### C. Syntax

```
http://127.0.0.1:8000/praktikum-laravel
```

#### D. Penjelasan

URL untuk mengakses halaman tersebut adalah :

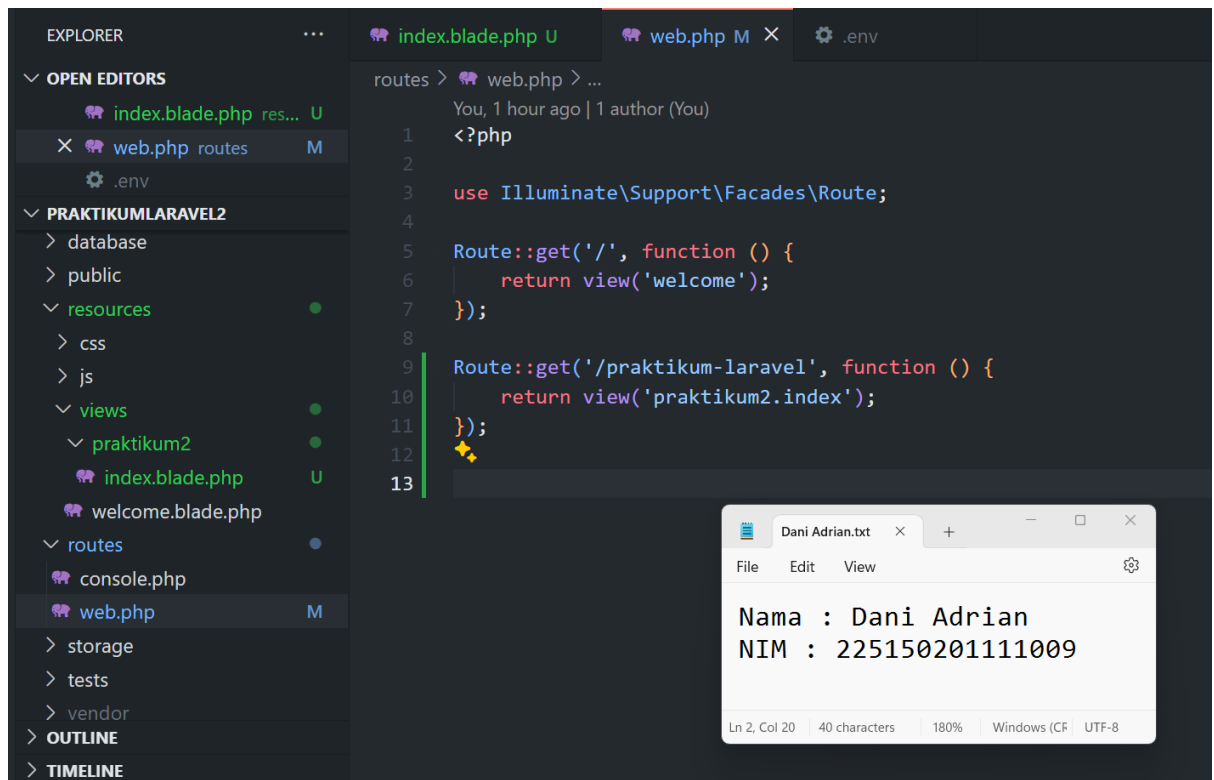
`http://127.0.0.1:8000/praktikum-laravel`

## LATIHAN 5

### A. Soal

Mengapa pada *method* view menggunakan parameter 'praktikum2.index'?

### B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

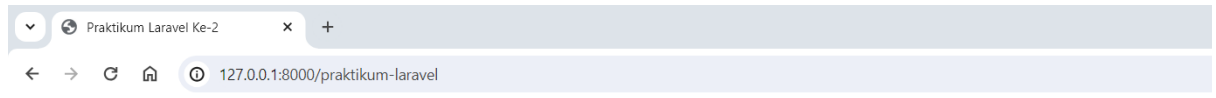
Parameteer 'praktikum2.index' memiliki makna khusus dalam konteks ini yaitu menunjukkan tampilan yang ingin dirender berada di dalam direktori praktikum2 dan memiliki nama file index.blade.php. Laravel menggunakan notasi titik (dot notation) untuk mengakses file dalam sub-direktori.

## LATIHAN 6

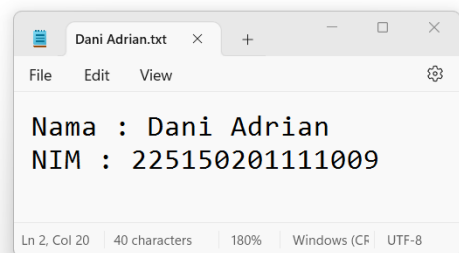
### A. Soal

Jalankan halaman tersebut, tempelkan *screenshot*-nya pada tempat yang telah disediakan dibawah ini.

### B. Screenshoot



## Praktikum Laravel Ke-2



### C. Syntax

1	
---	--

### D. Penjelasan

## 2. Perulangan dengan Sintaks Blade

## LATIHAN 1

### A. Soal

Ubahlah isi *file* `index.blade.php` menjadi sbb:

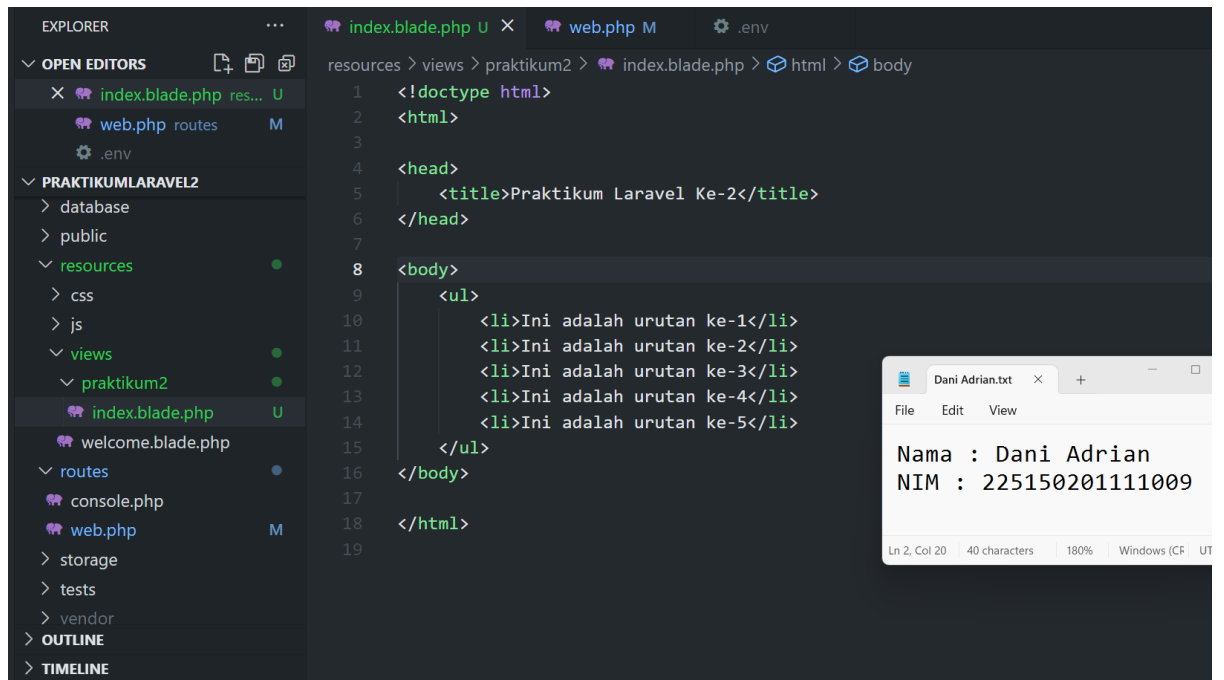
```
<!doctype html>
<html>
<head>
    <title>Praktikum Laravel Ke-2</title>
</head>
<body>
```

```

        <ul>
            <li>Ini adalah urutan ke-1</li>
            <li>Ini adalah urutan ke-2</li>
            <li>Ini adalah urutan ke-3</li>
            <li>Ini adalah urutan ke-4</li>
            <li>Ini adalah urutan ke-5</li>
        </ul>
    </body>
</html>

```

## B. Screenshoot



## C. Syntax

1	
---	--

## D. Penjelasan

Dengan mengubah isi file index.blade.php dan menambahkan elemen HTML baru ke dalam tampilan. Dalam konteks ini, menambahkan elemen `<ul>` (unordered list) yang berisi beberapa item `<li>` (list item).

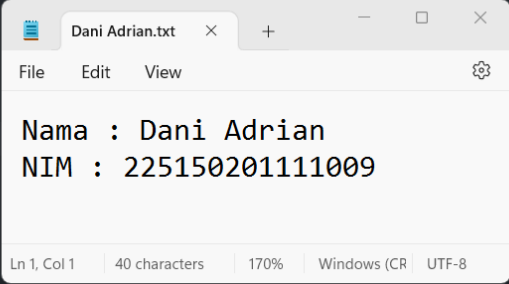
# LATIHAN 2

## A. Soal

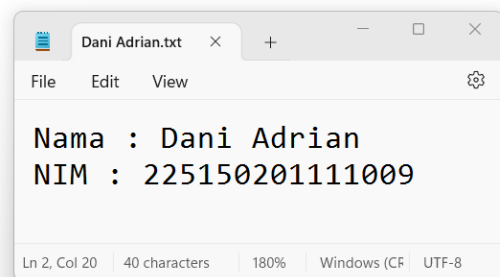
Kemudian ubahlah lagi menggunakan *loop* dengan sintaks Blade. Tempelkan kode yang saudara buat pada tempat yang telah disediakan di bawah ini

## B. Screenshoot

```
resources > views > praktikum2 > index.blade.php > ...
1  <!doctype html>
2  <html>
3
4  <head>
5      <title>Praktikum Laravel Ke-2</title>
6  </head>
7
8  <body>
9      <ul>
10         @for ($i = 1; $i <= 5; $i++)
11             <li>Ini adalah urutan ke-{{ $i }}</li>
12         @endfor
13     </ul>
14 </body>
15
16 <html>
17
```



- Ini adalah urutan ke-1
- Ini adalah urutan ke-2
- Ini adalah urutan ke-3
- Ini adalah urutan ke-4
- Ini adalah urutan ke-5



## C. Syntax

```
<!doctype html>
<html>

<head>
```



```

<title>Praktikum Laravel Ke-2</title>
</head>

<body>
  <ul>
    @for ($i = 1; $i <= 5; $i++)
      <li>Ini adalah urutan ke-{{ $i }}</li>
    @endfor
  </ul>
</body>

</html>

```

#### D. Penjelasan

Penggunaan @for memungkinkan Anda menghasilkan elemen HTML yang berulang secara dinamis. Dalam kasus ini, loop digunakan untuk membuat daftar berurutan dari 1 hingga 5.

### 3. Menampilkan Data Dari Route Parameter

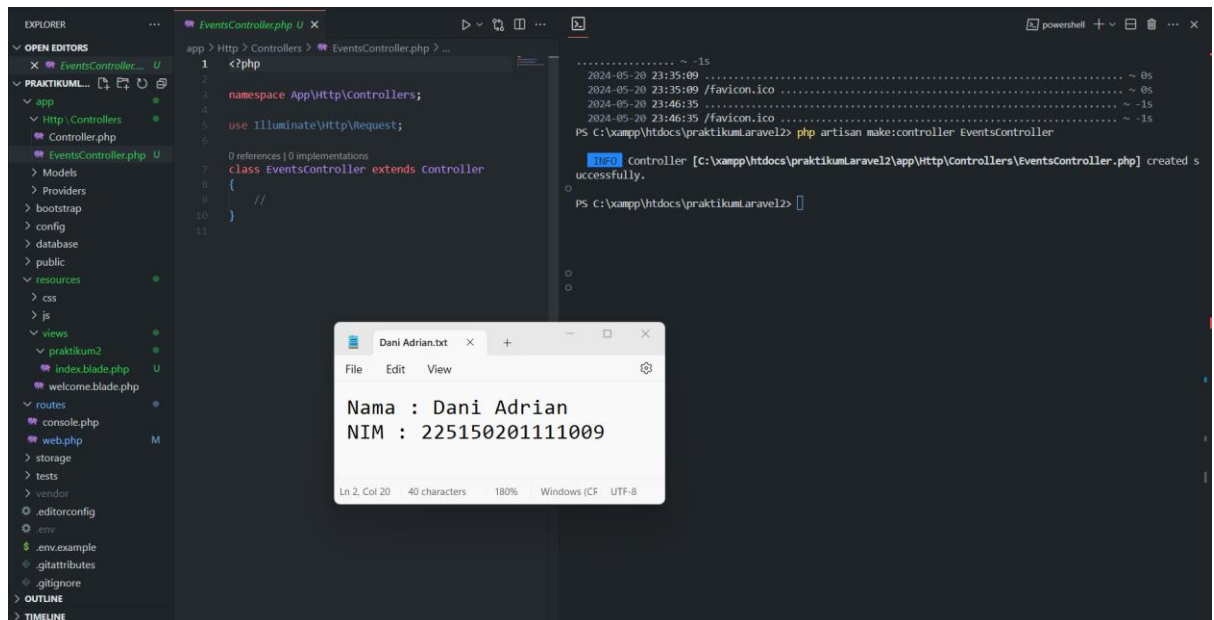
#### LATIHAN 1

##### A. Soal

Buatlah sebuah *controller* baru dengan perintah sbb:

```
php artisan make:controller EventsController
```

##### B. Screenshoot



##### C. Syntax

## D. Penjelasan

Perintah `php artisan make:controller EventsController` digunakan dalam Laravel untuk membuat sebuah controller baru dengan nama `EventsController`. Laravel akan membuat file `EventsController.php` di dalam direktori `app/Http/Controllers`.

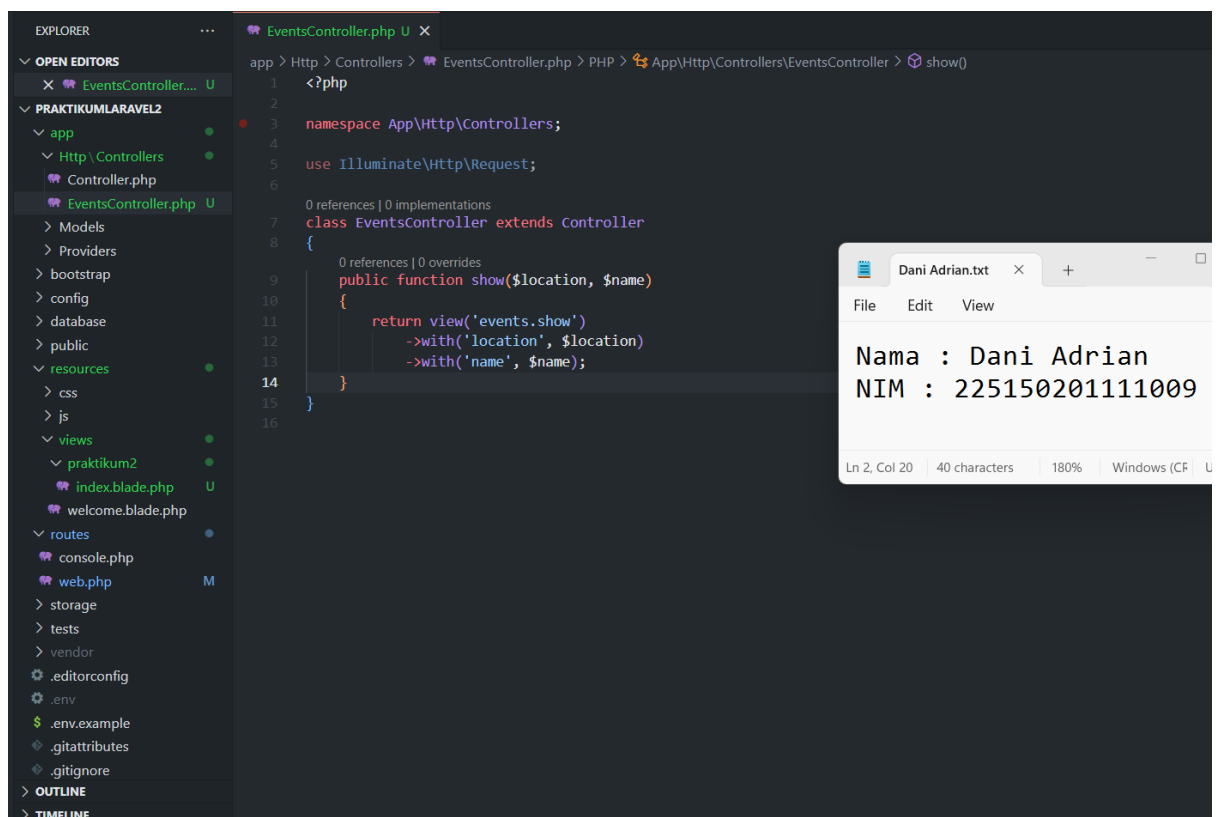
## LATIHAN 2

### A. Soal

Buka *file controller* yang baru saja di-generate pada folder `app\Http\Controllers` dan tambahkan sebuah *method* pada *class controller* tersebut seperti kode berikut:

```
public function show($location, $name)
{
    return view('events.show')
        ->with('location', $location)
        ->with('name', $name);
}
```

### B. Screenshoot



### C. Syntax

#### D. Penjelasan

`public function show($location, $name)` mendefinisikan sebuah method publik bernama `show` dengan dua parameter, yaitu `$location` dan `$name`. `return view('events.show')->with('location', $location)->with('name', $name);` adalah method `show` yang mengembalikan view dengan nama `events.show`. Dua variabel, `$location` dan `$name`, akan dilewatkan ke view tersebut menggunakan metode `with`. Ini memungkinkan data ini dapat diakses di dalam view.

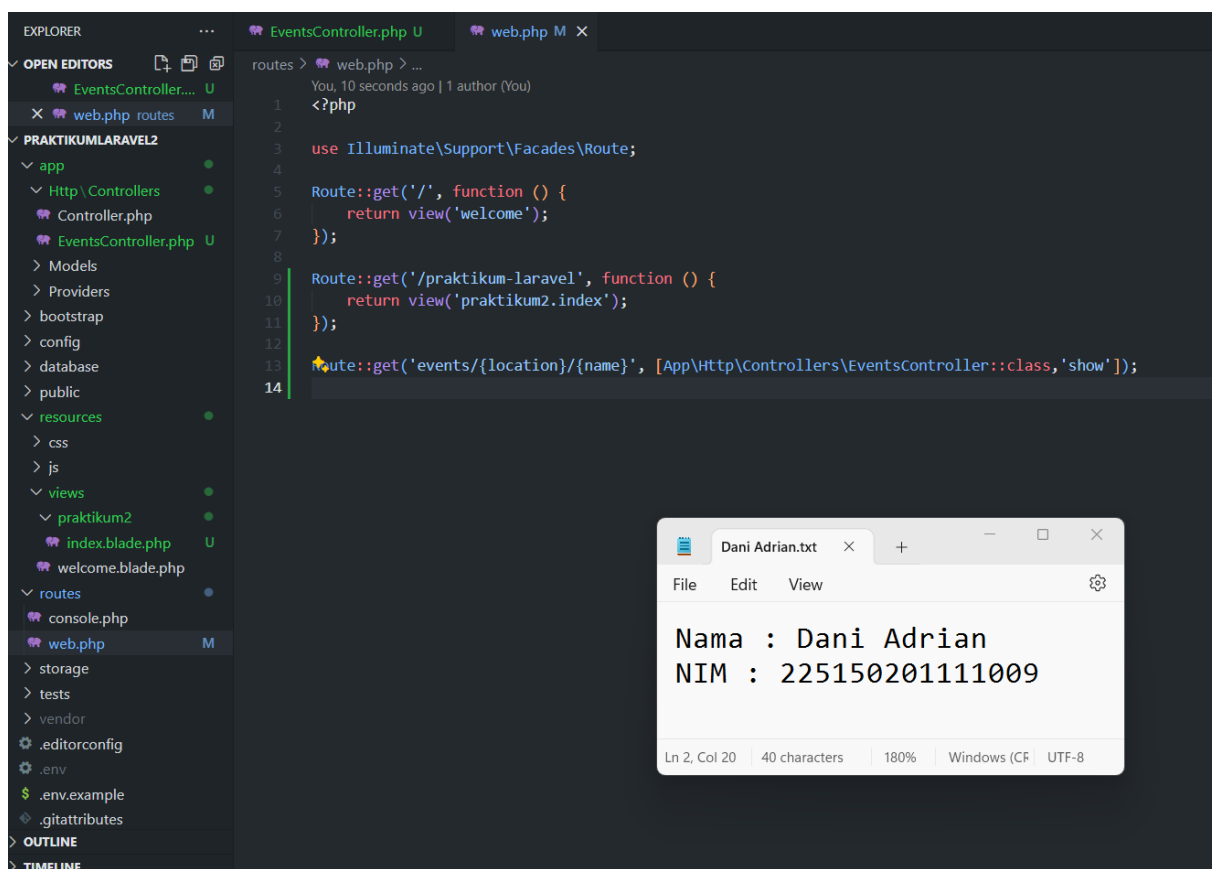
### LATIHAN 3

#### A. Soal

Tambahkan rute berikut:

```
Route::get('events/{location}/{name}',
[App\Http\Controllers\EventsController::class
'show']);
```

#### B. Screenshoot



### C. Syntax

```
<?php

use Illuminate\Support\Facades\Route;

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

Route::get('/praktikum-laravel', function () {
    return view('praktikum2.index');
});

Route::get('events/{location}/{name}',
[App\Http\Controllers\EventsController::class, 'show']);
```

### D. Penjelasan

Route ini menghubungkan URL `events/{location}/{name}` dengan method `show` pada controller `Event Controller`.

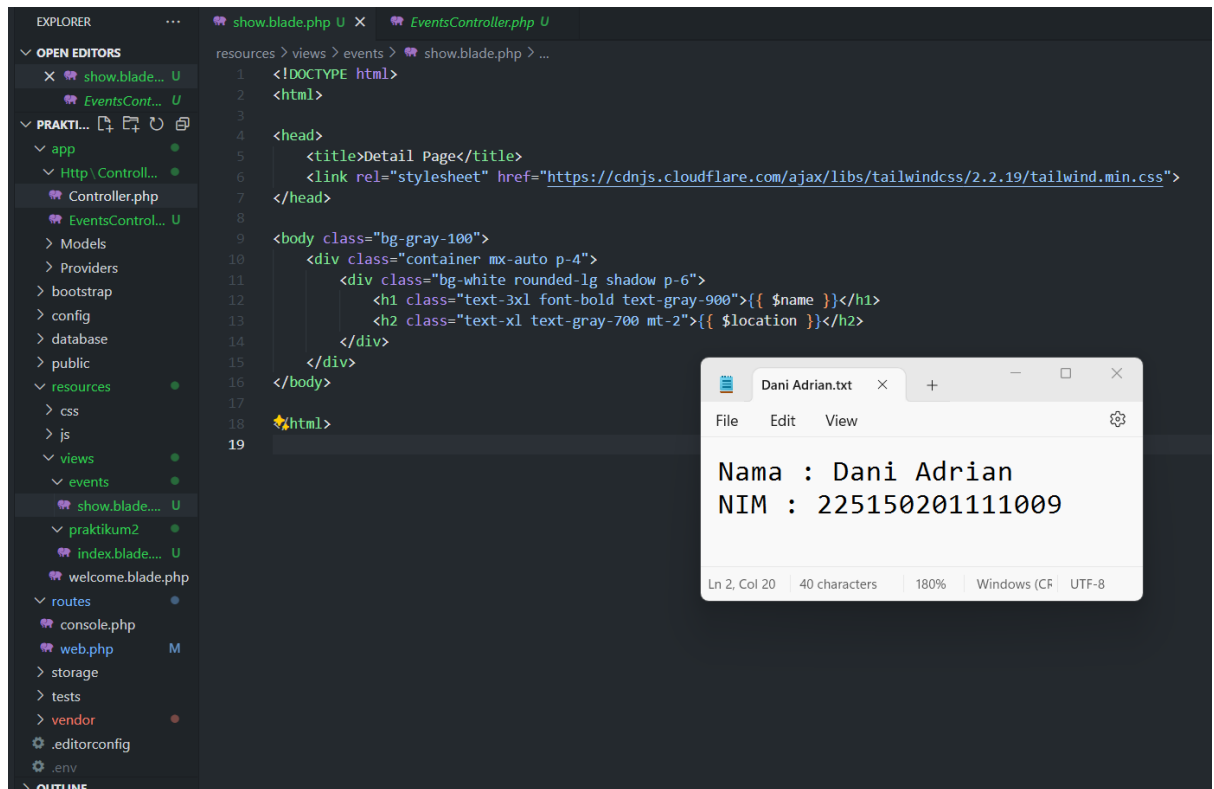
Ketika URL tersebut diakses melalui permintaan HTTP GET, Laravel akan menjalankan method `show` pada controller `EventsController`, sambil memberikan nilai dari `{location}` dan `{name}` sebagai parameter.

## LATIHAN 4

### A. Soal

Apakah nama file *view* (Blade) yang harus dibuat?

### B. Screenshoot



### C. Syntax

```
<!DOCTYPE html>
<html>

<head>
  <title>Detail Page</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2
.2.19/tailwind.min.css">
</head>

<body class="bg-gray-100">
  <div class="container mx-auto p-4">
    <div class="bg-white rounded-lg shadow p-6">
      <h1 class="text-3xl font-bold text-gray-
900">{{ $name }}</h1>
      <h2 class="text-xl text-gray-700 mt-2">{{
$location }}</h2>
    </div>
  </div>
</body>

</html>
```

### D. Penjelasan

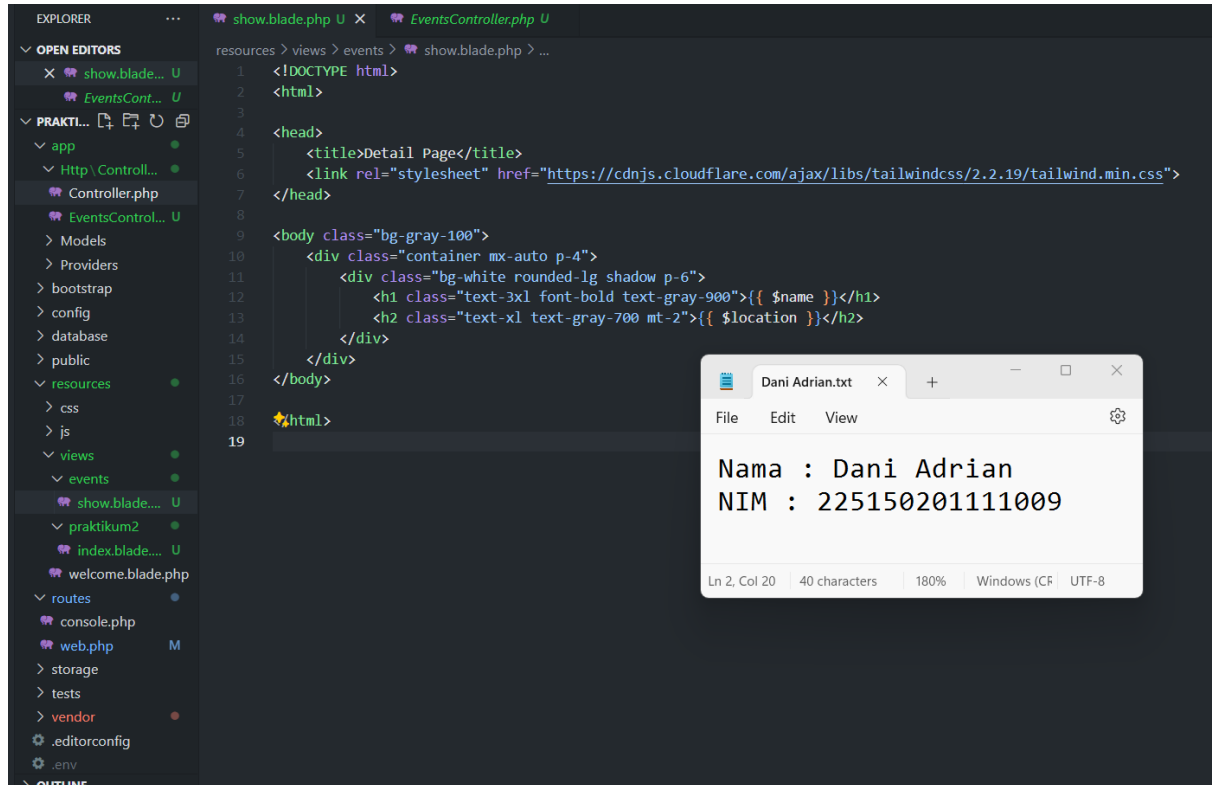
File view (Blade) dengan nama `show.blade.php`

## LATIHAN 5

## A. Soal

Pada direktori apakah *file view* tersebut harus diletakkan?

## B. Screenshoot



## C. Syntax

1	
---	--

## D. Penjelasan

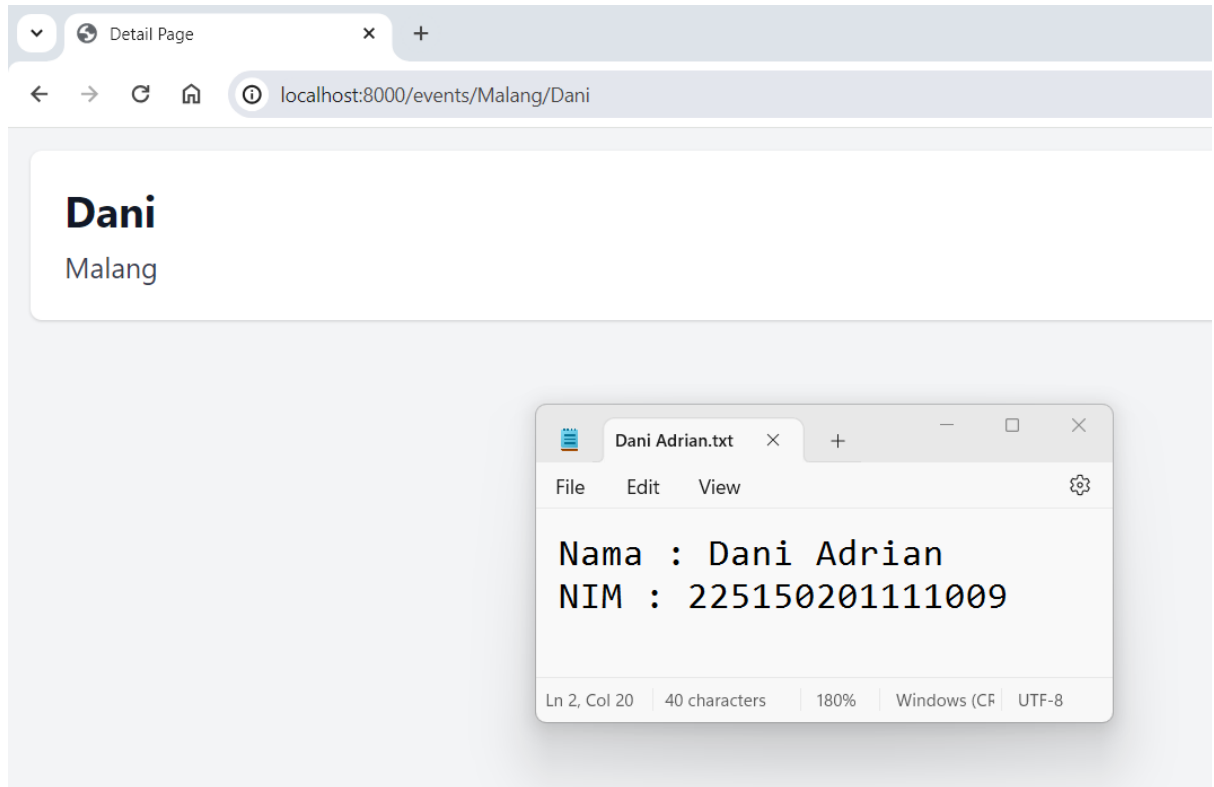
file view `show.blade.php` diletakkan di dalam direktori `resources/views/events`, agar Laravel dapat menemukannya dan merender konten dengan benar saat rute tersebut diakses.

# LATIHAN 6

## A. Soal

Tuliskan sebuah contoh URL untuk mengakses halaman tersebut.

## B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

URL untuk mengakses halamannya adalah :

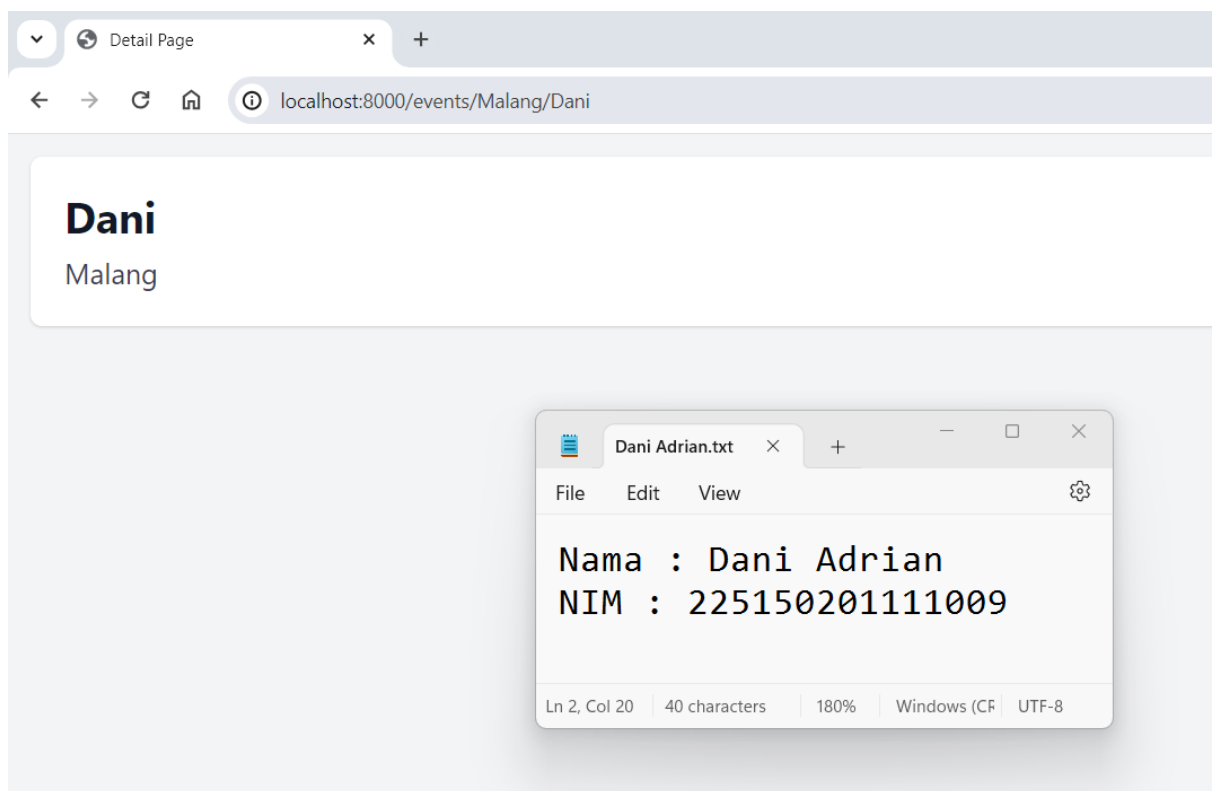
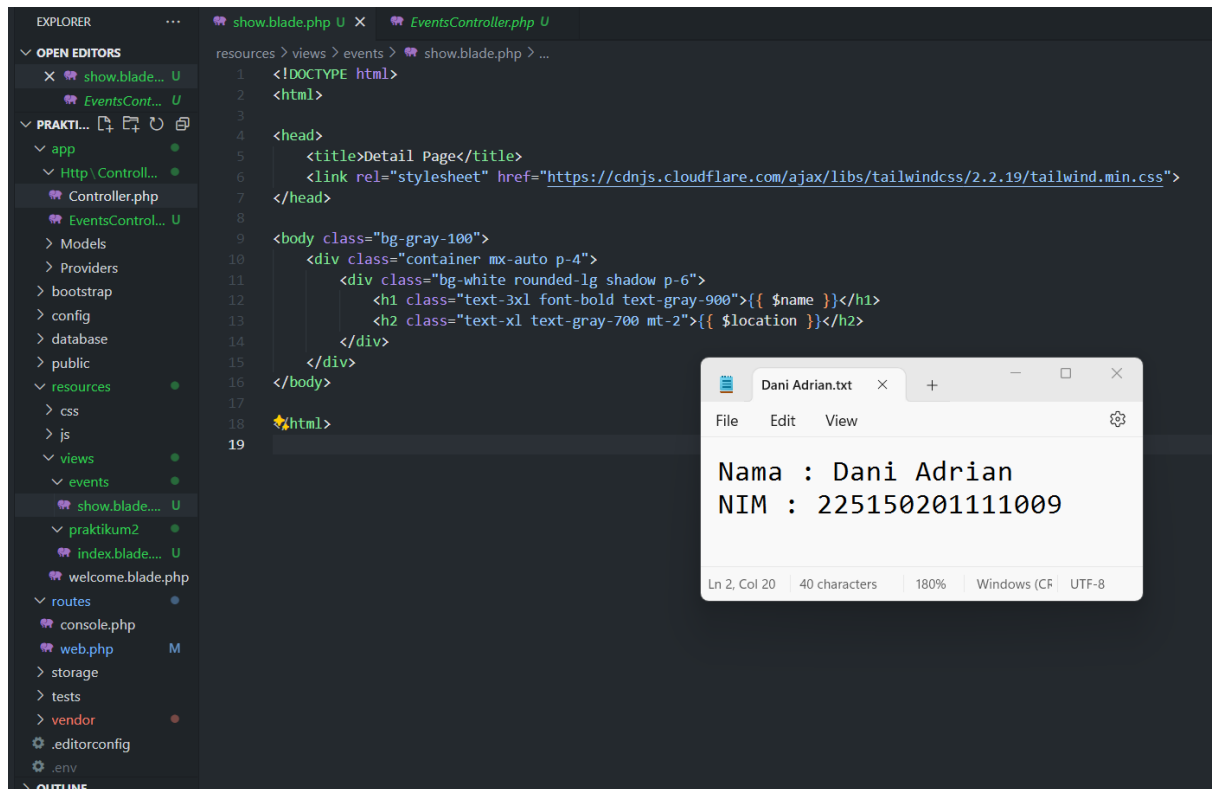
`praktikumaravel2.test/events/Malang/Dani`

## LATIHAN 7

### A. Soal

Buatlah file *view* (Blade) tersebut sehingga dapat menampilkan nilai variabel `location` dan `name`. Tempel kode yang dihasilkan pada tempat yang telah disediakan di bawah ini

### B. Screenshoot



### C. Syntax

```
<!DOCTYPE html>
<html>

<head>
```



```

<title>Detail Page</title>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2
.2.19/tailwind.min.css">
</head>

<body class="bg-gray-100">
  <div class="container mx-auto p-4">
    <div class="bg-white rounded-lg shadow p-6">
      <h1 class="text-3xl font-bold text-gray-
900">{{ $name }}</h1>
      <h2 class="text-xl text-gray-700 mt-2">{{
$location }}</h2>
    </div>
  </div>
</body>

</html>

```

#### **D. Penjelasan**

Kode tersebut menghasilkan halaman dengan nama dan lokasi yang ditampilkan dengan menggunakan nilai variabel `$name` dan `$location`. Styling halaman didasarkan pada framework CSS Tailwind CSS untuk tampilan yang menarik dan responsif

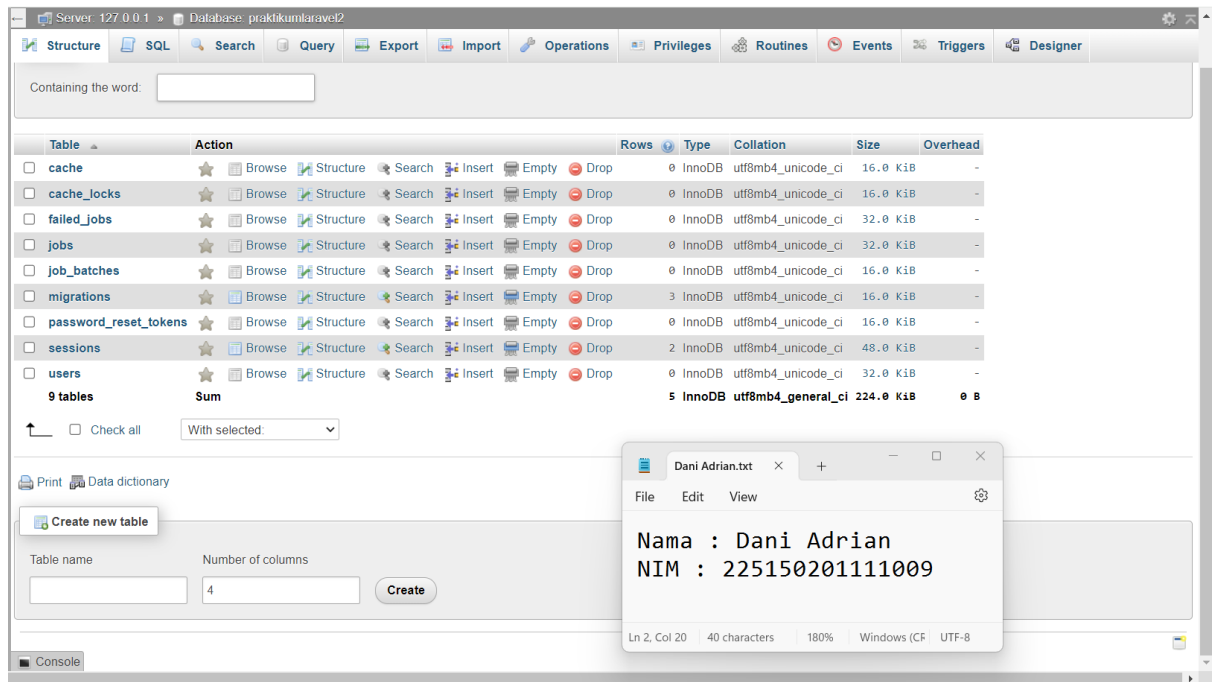
## **4. Model Untuk Menampilkan Data Dari Database**

### **LATIHAN 1**

#### **A. Soal**

Jalankan MySQL dan buatlah database baru dengan nama “praktikumlaravel2”. (Bila saudara menggunakan XAMPP, jalankan Apache dan MySQL kemudian klik tombol Admin pada baris MySQL untuk membuka aplikasi phpMyAdmin. Melalui phpMyAdmin, saudara bisa membuat database baru tersebut).

#### **B. Screenshoot**



### C. Syntax

1

### D. Penjelasan

Membuka phpMyAdmin, antarmuka web untuk mengelola database MySQL. Kemudian membuat database baru dengan memberikan nama "praktikumlaravel2".

## LATIHAN 2

### A. Soal

Pada file `.env` yang berada pada direktori `root` proyek Laravel, ubahlah nama database MySQL sesuai nama yang tadi telah dibuat (jika namanya berbeda). Bila diperlukan, atur pula username, password atau konfigurasi lainnya untuk mengakses MySQL.

### B. Screenshoot

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:y/drTa/H916Dtf2nCN1IhCdFoSK718FK889CsJzQKTg=
4 APP_DEBUG=true
5 APP_TIMEZONE=UTC
6 APP_URL=http://praktikumlaravel2.test
7
8 APP_LOCALE=en
9 APP_FALLBACK_LOCALE=en
10 APP_FAKER_LOCALE=en_US
11
12 APP_MAINTENANCE_DRIVER=file
13 APP_MAINTENANCE_STORE=database
14
15 BCRYPT_ROUNDS=12
16
17 LOG_CHANNEL=stack
18 LOG_STACK=single
19 LOG_DEPRECATIONS_CHANNEL=null
20 LOG_LEVEL=debug
21
22 DB_CONNECTION=mysql
23 DB_HOST=127.0.0.1
24 DB_PORT=3306
25 DB_DATABASE=praktikumlaravel2
26 DB_USERNAME=admin
27 DB_PASSWORD=1234
28
29 SESSION_DRIVER=database
30 SESSION_LIFETIME=120
31 SESSION_ENCRYPT=false
32 SESSION_PATH=/
33 SESSION_DOMAIN=null
34
35 BROADCAST_CONNECTION=log
36 FILESYSTEM_DISK=local
37 QUEUE_CONNECTION=database
```

Dani Adrian.txt  
File Edit View  
Nama : Dani Adrian  
NIM : 225150201111009  
Ln 2, Col 20 | 40 characters | 180% | Windows (CF) | UTF-8

### C. Syntax

1	
---	--

### D. Penjelasan

DB_CONNECTION=mysql
Menentukan jenis koneksi database yang akan digunakan, dalam hal ini MySQL.
DB_HOST=127.0.0.1
Menentukan alamat host dari server database MySQL. 127.0.0.1 adalah alamat loopback untuk localhost.
DB_PORT=3306
Baris ini menentukan port yang digunakan oleh server database MySQL. Port default untuk MySQL adalah 3306.
DB_DATABASE=praktikumlaravel2

Menentukan nama database yang akan digunakan oleh proyek Laravel.

DB\_USERNAME=Admin

Menentukan nama pengguna (username) yang akan digunakan untuk mengakses database MySQL.

DB\_PASSWORD=1234

Menentukan kata sandi (password) yang akan digunakan untuk mengautentikasi koneksi ke database MySQL. Kata sandi yang diatur sesuai dengan kata sandi pengguna yang ditentukan untuk akses database.

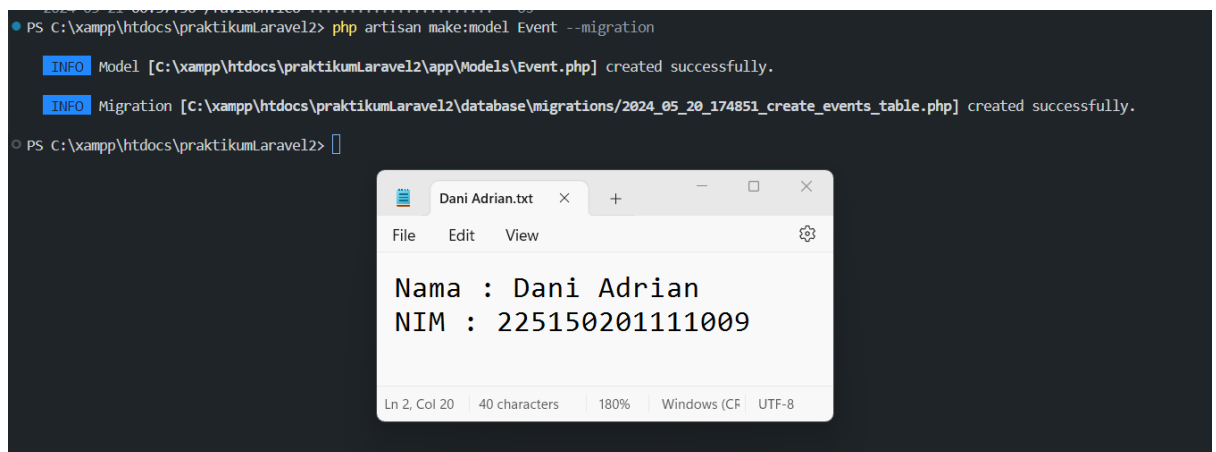
## LATIHAN 3

### A. Soal

Buatlah model dengan menjalankan perintah sbb:

```
php artisan make:model Event --migration
```

### B. Screenshoot



### C. Syntax

1

### D. Penjelasan

Dengan menjalankan perintah `php artisan make:model Event --migration`, akan membuat model Event yang siap digunakan dalam proyek Laravel, serta migrasi yang sesuai untuk membuat tabel database yang diperlukan.

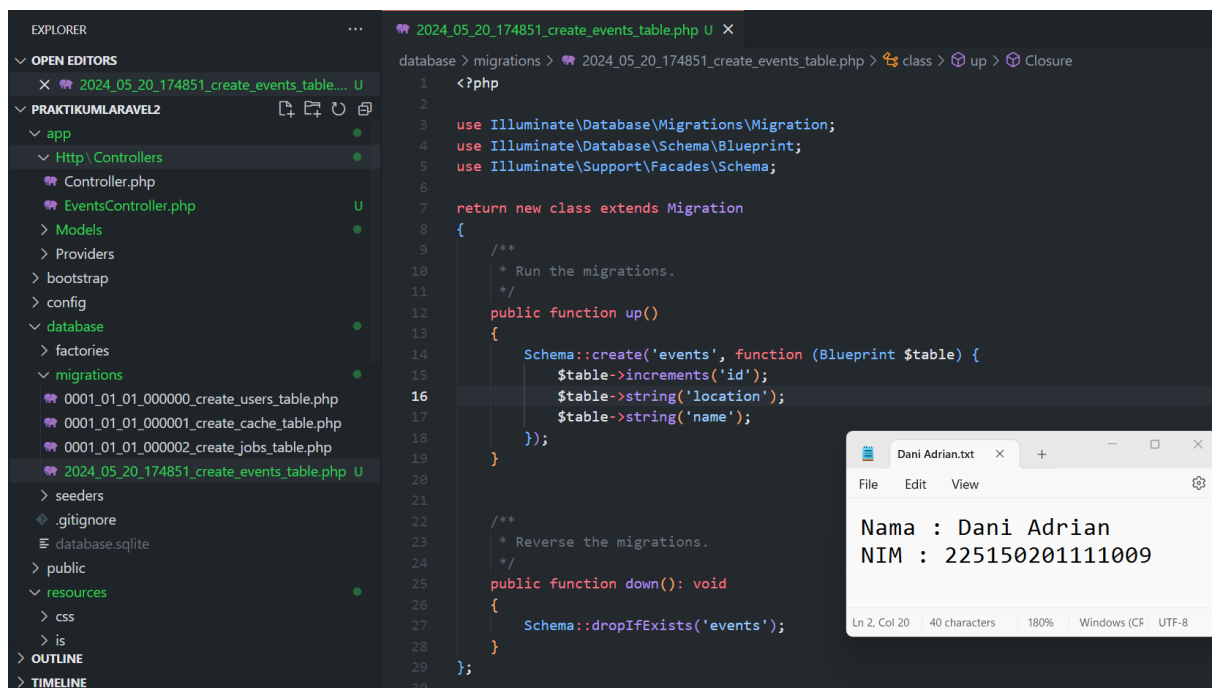
## LATIHAN 4

## A. Soal

Buka file *migration* yang di-generate oleh Artisan pada direktori `"/database/migration"` pilih file yang sesuai dengan model yang baru saja dibuat (nama *file* berakhiran `...._create_events_table`). Kemudian ubah method `up()` menjadi sbb:

```
public function up()
{
    Schema::create('events', function (Blueprint $table) {
        $table->increments('id');
        $table->string('location');
        $table->string('name');
    });
}
```

## B. Screenshoot



## C. Syntax

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up()
```

```

    {
        Schema::create('events', function (Blueprint
$table) {
            $table->increments('id');
            $table->string('location');
            $table->string('name');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('events');
    }
};

```

#### D. Penjelasan

Schema::create('events', function (Blueprint \$table) { ... }) adalah bagian utama yang mendefinisikan pembuatan tabel events. closure function untuk menentukan struktur tabel. \$table->increments('id') menambahkan kolom id sebagai primary key dengan tipe data integer yang secara otomatis bertambah nilainya (auto-increment). \$table->string('location') dan \$table->string('name') menambahkan dua kolom lainnya, location dan name, dengan tipe data string.

## LATIHAN 5

#### A. Soal

Lakukan migrasi dengan menjalankan perintah sbb:

```
php artisan migrate
```

#### B. Screenshoot

```

PS C:\xampp\htdocs\praktikumLaravel2> php artisan migrate

  INFO  Running migrations.

 2024_05_20_174851_create_events_table ..... 8.59ms DONE

PS C:\xampp\htdocs\praktikumLaravel2>

```

Dani Adrian.txt

File Edit View

Nama : Dani Adrian  
NIM : 225150201111009

### C. Syntax

1	
---	--

### D. Penjelasan

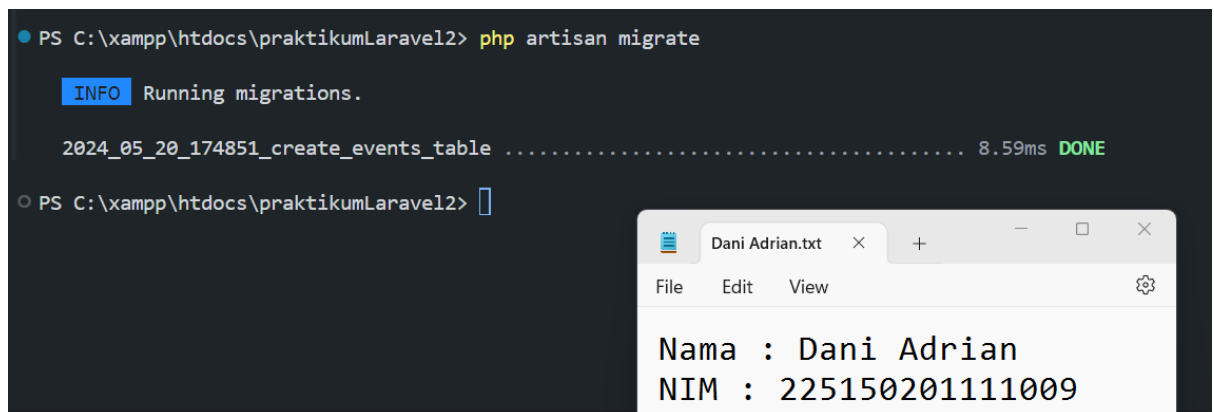
`php artisan migrate` digunakan untuk menjalankan semua migrasi.

## LATIHAN 6

### A. Soal

Tempel *screenshot* CMD atau Terminal setelah perintah di atas dijalankan pada tempat yang disediakan di bawah ini.

### B. Screenshoot



The screenshot shows a Windows Command Prompt window with the following text:

```
PS C:\xampp\htdocs\praktikumLaravel2> php artisan migrate
```

Below this, there is a blue box with the text "INFO Running migrations." and a line of text indicating the migration of a table:

```
2024_05_20_174851_create_events_table ..... 8.59ms DONE
```

Below this, there is a prompt for the next command:

```
PS C:\xampp\htdocs\praktikumLaravel2> 
```

To the right of the Command Prompt, there is a text editor window titled "Dani Adrian.txt" with the following text:

```
Nama : Dani Adrian
NIM : 225150201111009
```

### C. Syntax

1	
---	--

### D. Penjelasan

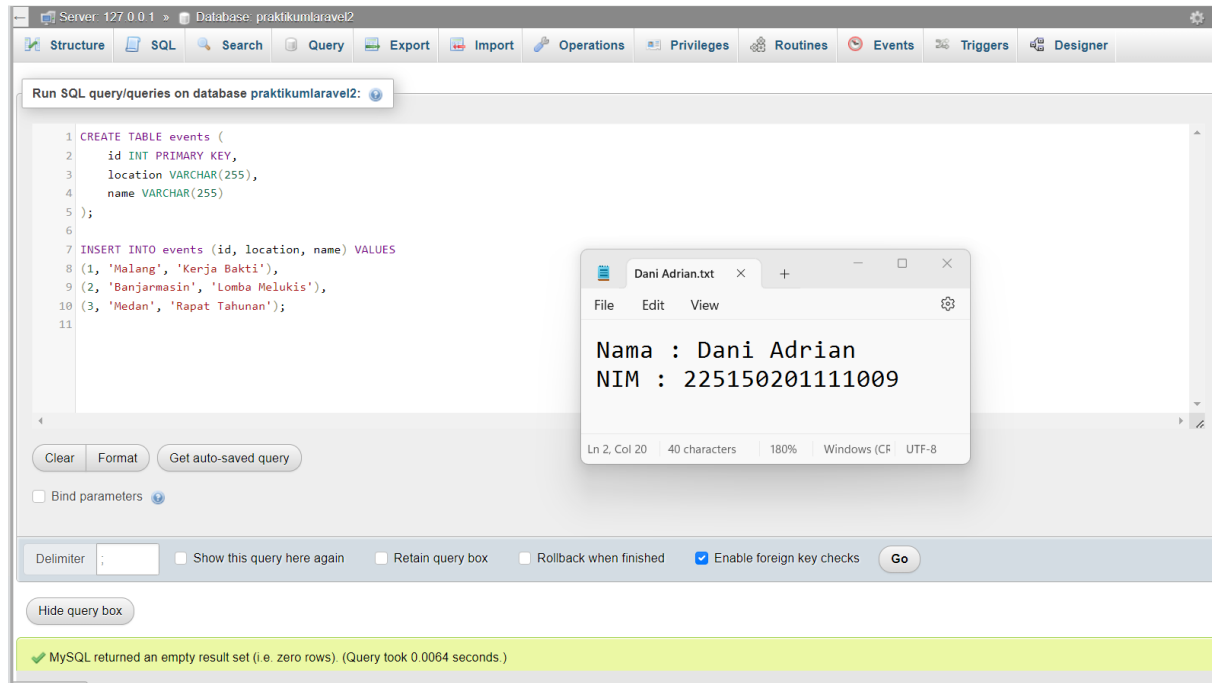
## LATIHAN 7

### A. Soal

Isi tabel `events` dengan data sebagai berikut (Saudara bisa menggunakan aplikasi phpMyAdmin atau sejenisnya):

id	location	name
1	Malang	Kerja Bakti
2	Banjarmasin	Lomba Melukis
3	Medan	Rapat Tahunan

## B. Screenshoot



## C. Syntax

```
CREATE TABLE events ( id INT PRIMARY KEY, location
VARCHAR(255), name VARCHAR(255)
);
INSERT INTO events (id, location, name) VALUES (1,
'Malang', 'Kerja Bakti'), (2, 'Banjarmasin', 'Lomba
Melukis'), (3, 'Medan', 'Rapat Tahunan');
```

## D. Penjelasan

- `CREATE TABLE events` untuk membuat sebuah tabel baru bernama events.
- `id INT PRIMARY KEY` untuk mendefinisikan kolom id sebagai primary key dengan tipe data integer.
- `location VARCHAR(255)` untuk mendefinisikan kolom location sebagai kolom string dengan panjang maksimum 255 karakter.
- `name VARCHAR(255)` untuk mendefinisikan kolom name sebagai kolom string dengan panjang maksimum 255 karakter.



- INSERT INTO events untuk memasukkan beberapa baris data ke dalam tabel events dengan VALUES (1, 'Malang', 'Kerja Bakti'), (2, 'Banjarmasin', 'Lomba Melukis'), (3, 'Medan', 'Rapat Tahunan')

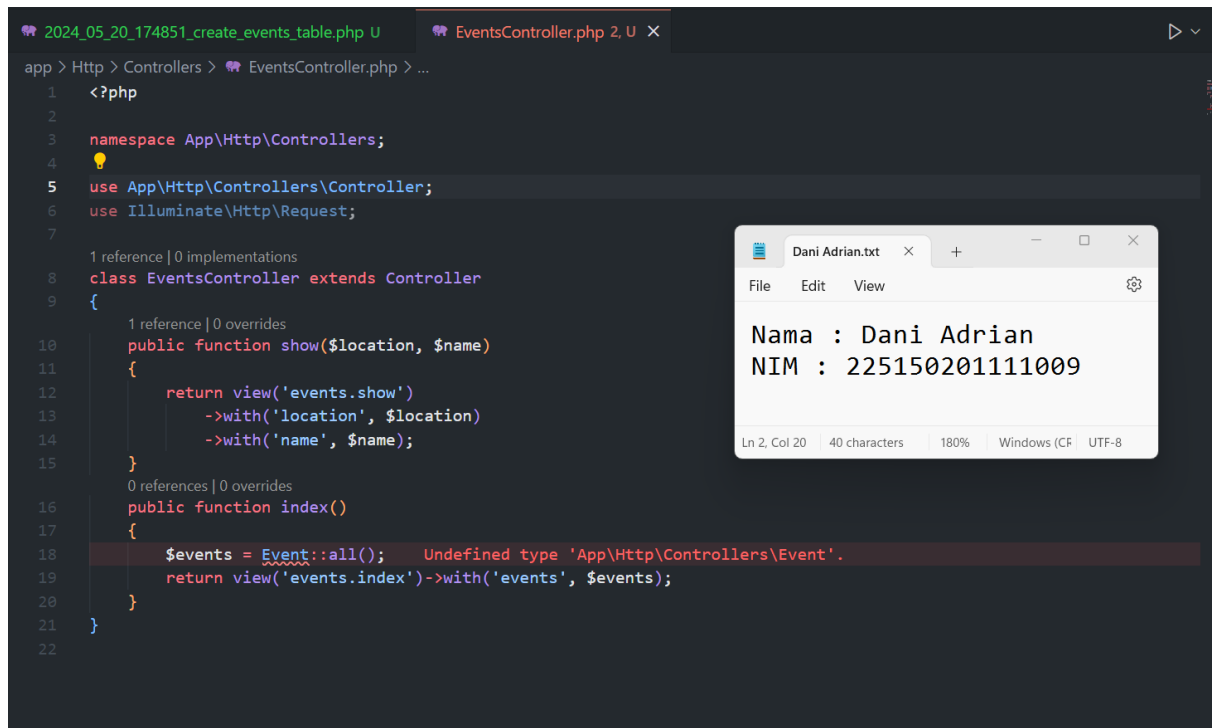
## LATIHAN 8

### A. Soal

Pada *class* EventsController, tambahkan method index() sbb:

```
public function index()
{
    $events = Event::all();
    return view('events.index')->with('events', $events);
}
```

### B. Screenshoot



### C. Syntax

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class EventsController extends Controller
```

```
{
    public function show($location, $name)
    {
        return view('events.show')
            ->with('location', $location)
            ->with('name', $name);
    }
    public function index()
    {
        $events = Event::all();
        return view('events.index')->with('events',
$events);
    }
}
```

#### D. Penjelasan

Menambahkan method `index()` ke dalam class `EventsController`

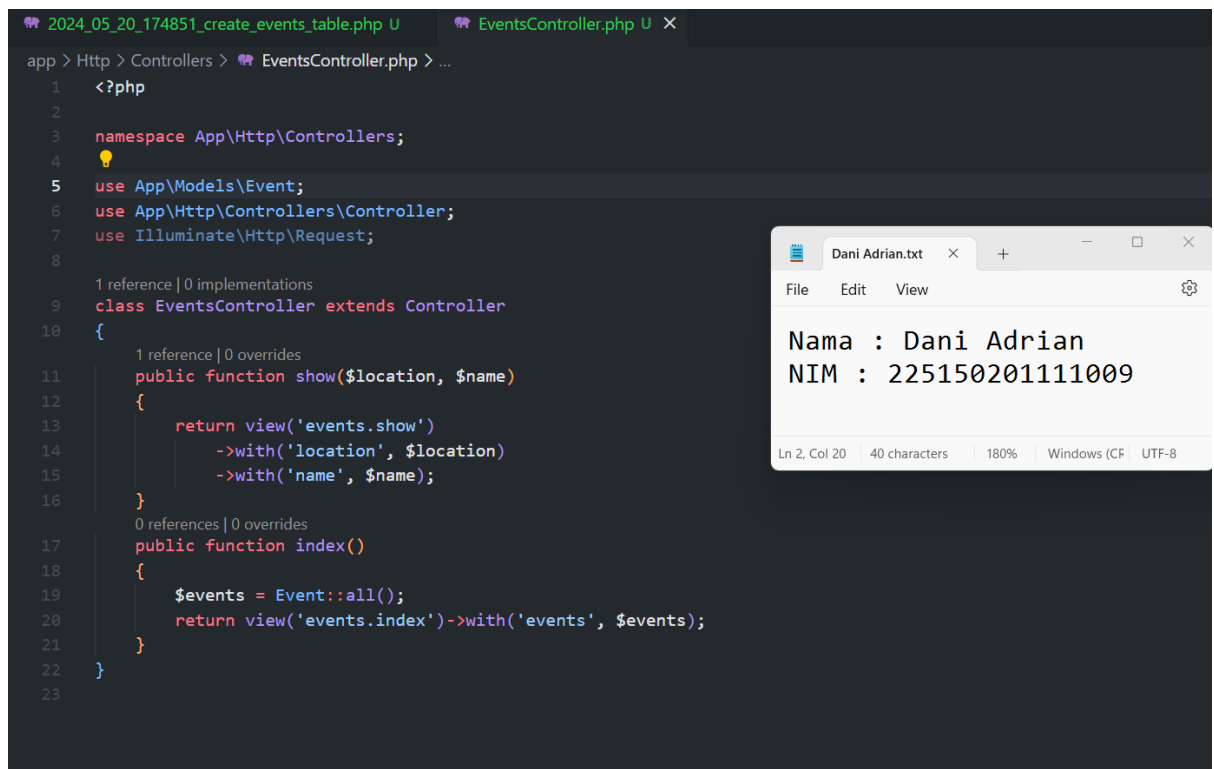
## LATIHAN 9

#### A. Soal

Pada kode `EventsController.php`, tambahkan pula pernyataan untuk *import* sbb:

```
use App\Models\Event;
```

#### B. Screenshoot



### C. Syntax

```
<?php

namespace App\Http\Controllers;

use App\Models\Event;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class EventsController extends Controller
{
    public function show($location, $name)
    {
        return view('events.show')
            ->with('location', $location)
            ->with('name', $name);
    }
    public function index()
    {
        $events = Event::all();
        return view('events.index')->with('events',
        $events);
    }
}
```

### D. Penjelasan

Mengimpor (import) kelas Event yang berada dalam namespace App\Models ke dalam file EventsController.php

## LATIHAN 10

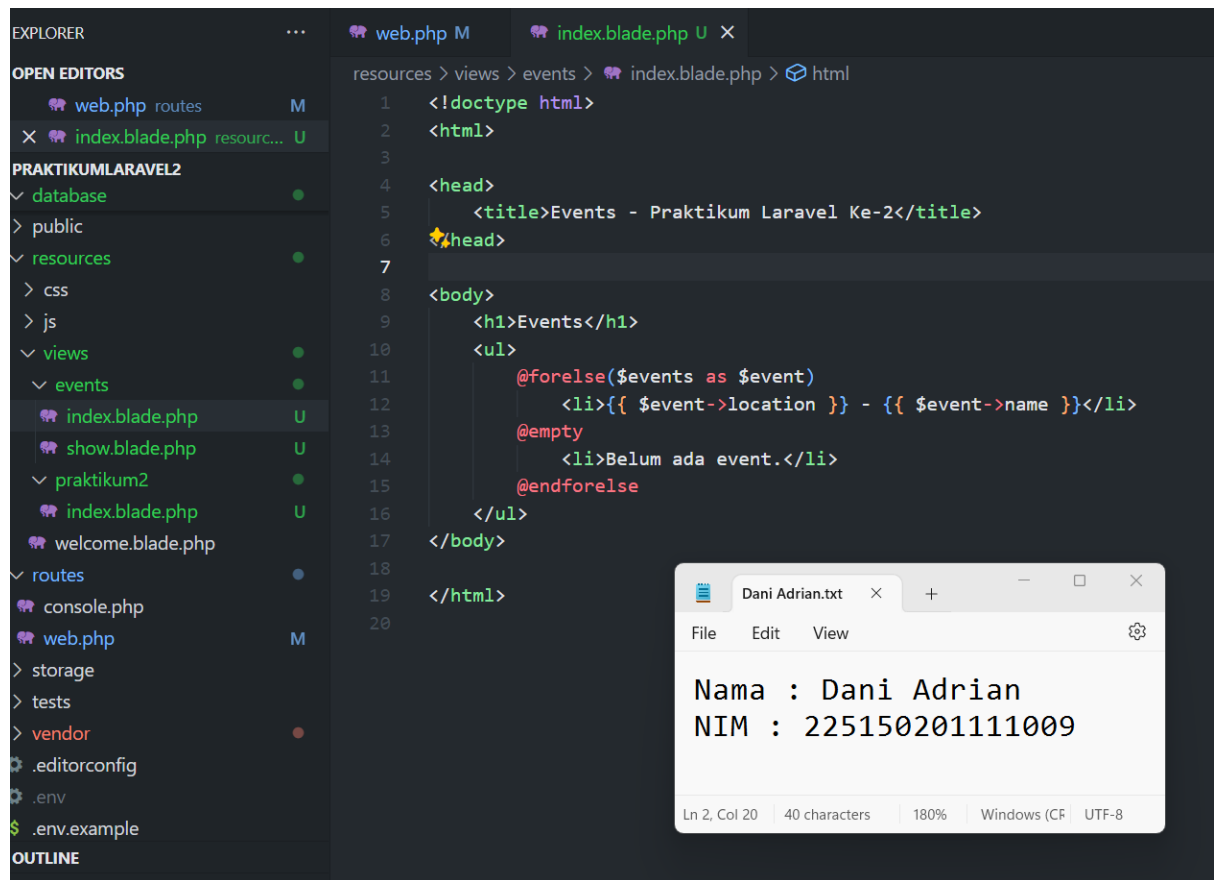
### A. Soal

Kemudian buatlah view dengan lokasi dan nama sesuai pada method index() di atas, isi dengan kode sbb:

```
<!doctype html>
<html>
<head>
    <title>Events - Praktikum Laravel Ke-2</title>
</head>
<body>
    <h1>Events</h1>
    <ul>
        @forelse($events as $event)
            <li>{{ $event->location }} - {{ $event->name }}</li>
        @empty
            <li>Belum ada event.</li>
        @endforelse
    </ul>
```

```
</body>
</html>
```

## B. Screenshoot



## C. Syntax

1

## D. Penjelasan

View ini menampilkan daftar acara yang diambil dari controller dan dilewatkan ke view melalui variabel `$events`. Jika tidak ada acara yang tersedia, pesan "Belum ada event." akan ditampilkan.

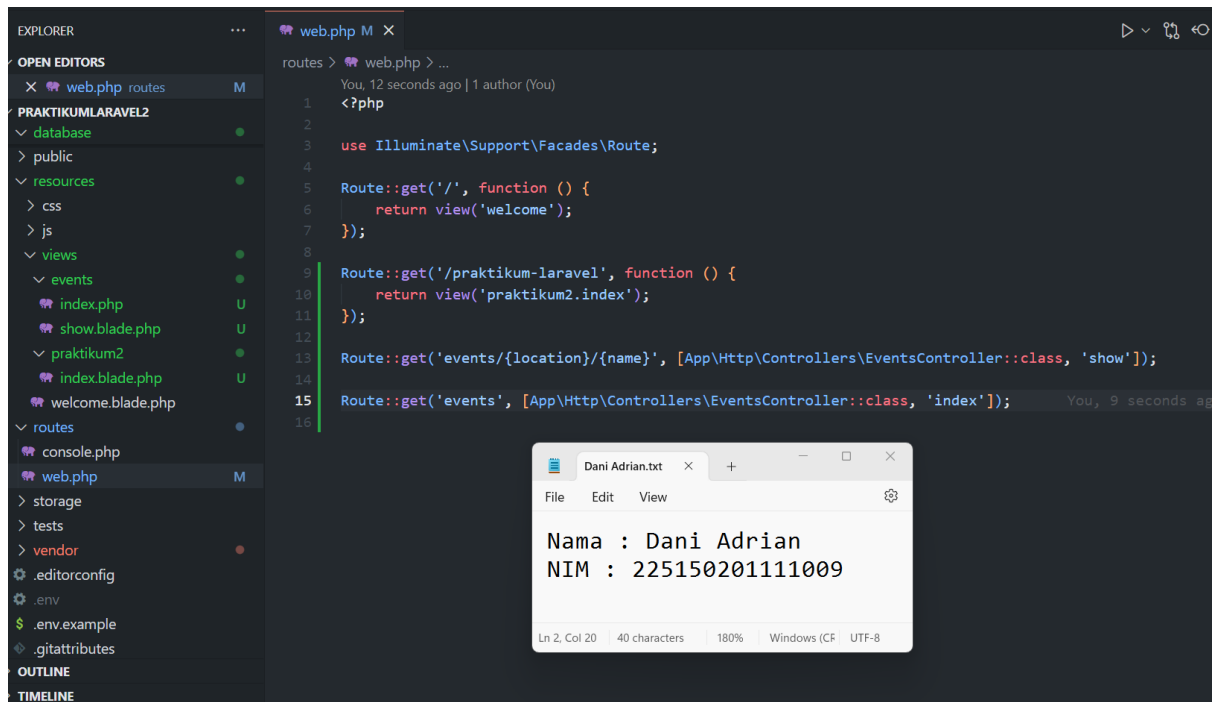
# LATIHAN 11

## A. Soal

Tambahkan rute berikut:

```
Route::get('events',
[App\Http\Controllers\EventsController::class, 'index']);
```

## B. Screenshoot



## C. Syntax

1	
---	--

## D. Penjelasan

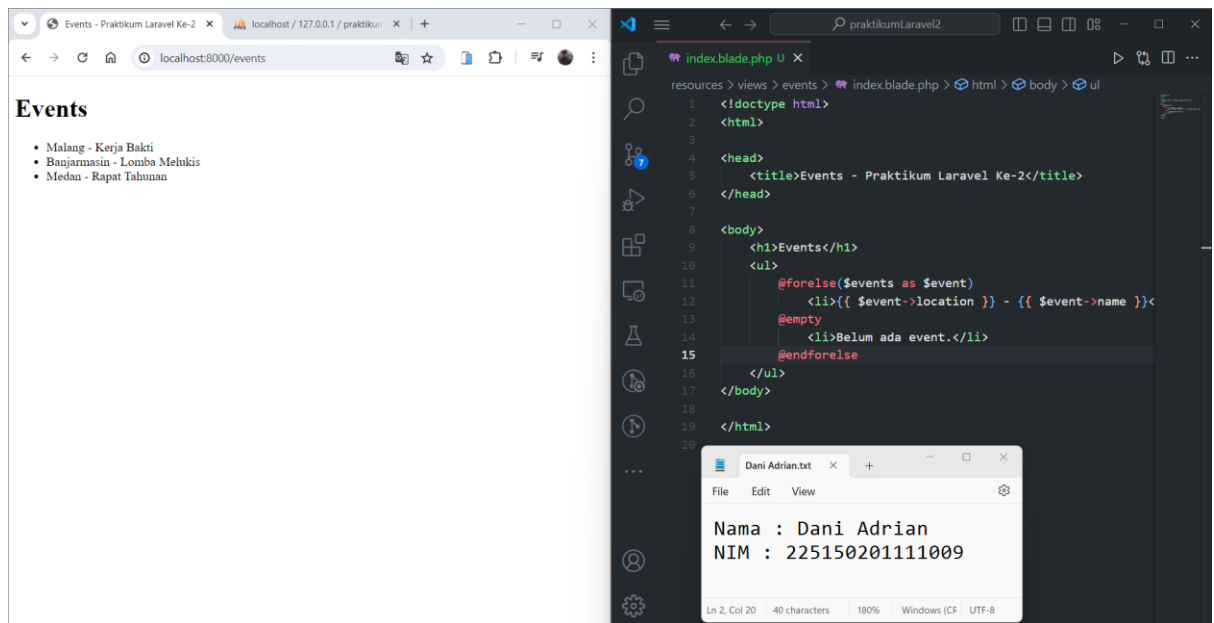
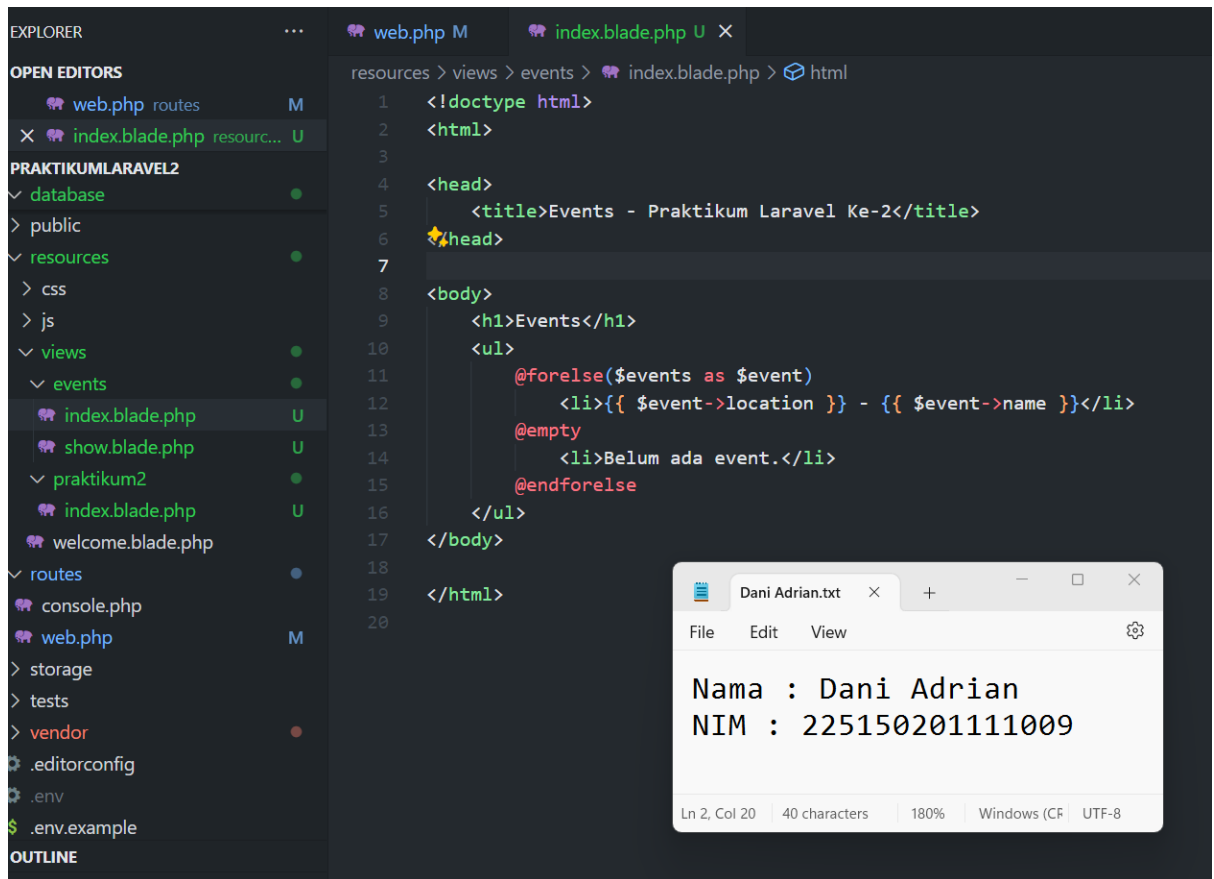
`Route::get('events', [App\Http\Controllers\EventsController::class, 'index'])`; adalah rute yang menentukan bahwa ketika pengguna melakukan permintaan HTTP GET ke URL '/events', Laravel akan menjalankan method `index()` yang ada dalam controller `EventsController`.

## LATIHAN 12

### A. Soal

Buka halaman tersebut pada *web browser* dan tempelkan *screenshot*-nya pada tempat yang disediakan di bawah ini:

### B. Screenshoot



## C. Syntax

1	
---	--

## D. Penjelasan

## 5. Flash Message Untuk Menampilkan Pesan

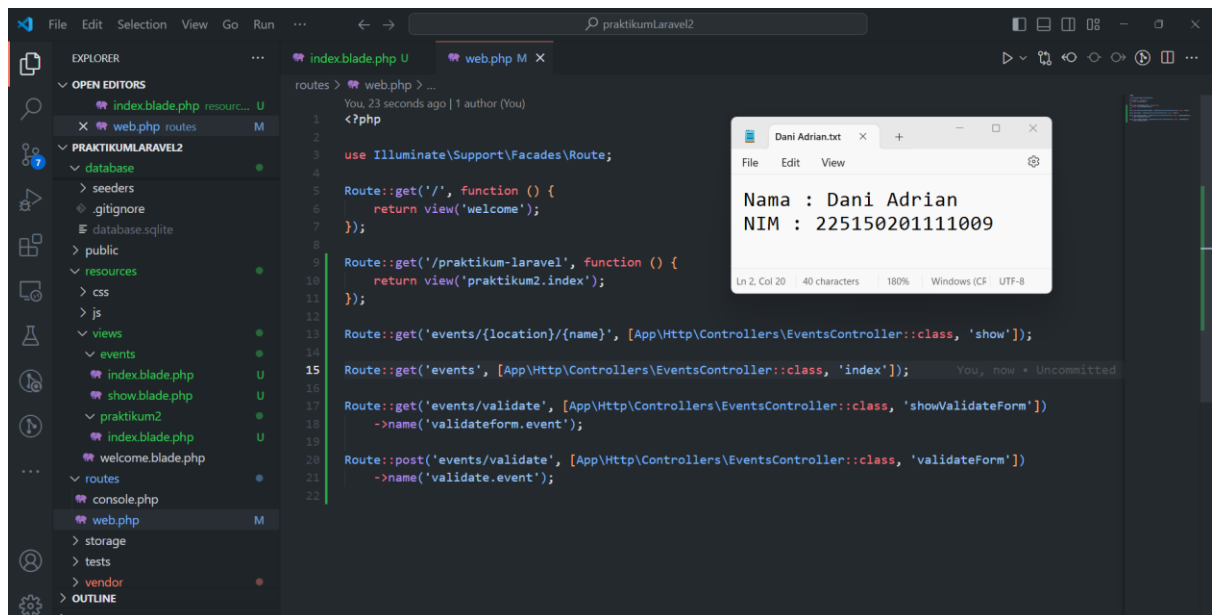
### LATIHAN 1

#### A. Soal

Tambahkan rute berikut:

```
Route::get('events/validate',  
[App\Http\Controllers\EventsController::class,  
'showValidateForm'])  
->name('validateform.event');  
  
Route::post('events/validate',  
[App\Http\Controllers\EventsController::class,  
'validateForm'])  
->name('validate.event');
```

#### B. Screenshoot



#### C. Syntax

1	
---	--

#### D. Penjelasan

Route untuk menampilkan form validasi :

```
Route::get('events/validate',  
[App\Http\Controllers\EventsController::class,  
'showValidateForm'])  
->name('validateform.event');
```

Route untuk memproses form validasi :

```
Route::post('events/validate',  
[App\Http\Controllers\EventsController::class,  
'validateForm'])  
->name('validate.event');
```

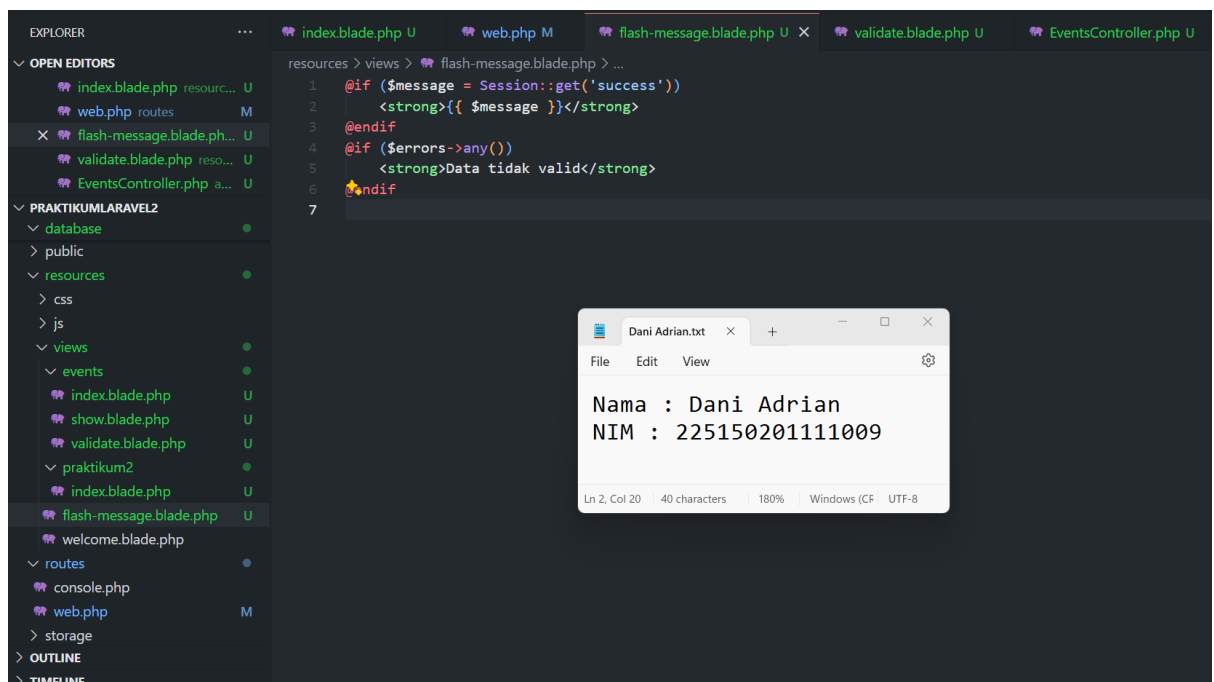
## LATIHAN 2

### A. Soal

Pada folder view, buatlah *file* baru bernama "flash-message.blade.php", kemudian isi dengan kode sbb:

```
@if ($message = Session::get('success'))  
    <strong>{{ $message }}</strong>  
@endif  
  
@if ($errors->any())  
    <strong>Data tidak valid</strong>  
@endif
```

### B. Screenshoot



### C. Syntax

1	
---	--



#### D. Penjelasan

- @if (\$message = Session::get('success')) adalah kondisional Blade yang mengecek apakah ada pesan sukses dalam session. Jika ada, pesan tersebut akan ditampilkan dengan menggunakan variabel \$message.
- @if (\$errors->any()) adalah kondisional Blade lainnya yang mengecek apakah ada kesalahan validasi. Jika ada, pesan "Data tidak valid" akan ditampilkan.

## LATIHAN 3

#### A. Soal

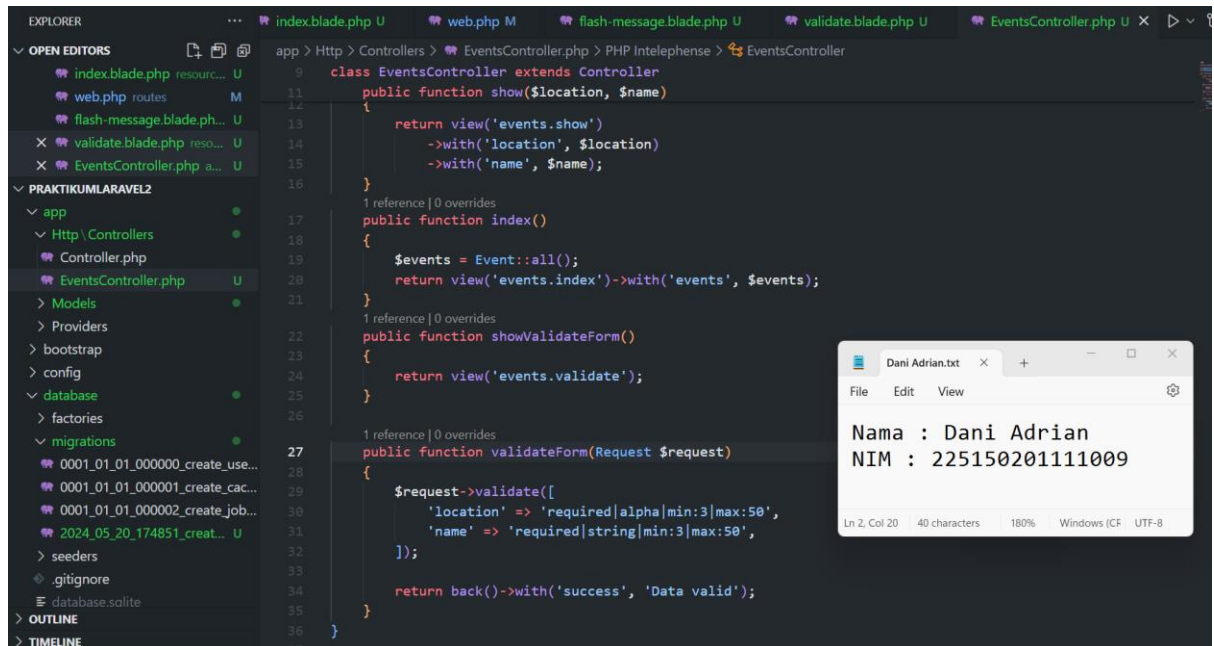
Pada *class* EventsController, tambahkan dua *method* sebagai berikut:

```
public function showValidateForm()
{
    return view('events.validate');
}

public function validateForm(Request $request)
{
    $request->validate([
        'location' => 'required|alpha|min:3|max:50',
        'name' => 'required|string|min:3|max:50',
    ]);

    return back()->with('success', 'Data valid');
}
```

#### B. Screenshoot



#### C. Syntax

1	
---	--

#### D. Penjelasan

- Method `showValidateForm()` digunakan untuk menampilkan formulir validasi. Ketika method ini dipanggil, ia akan mengembalikan view bernama `'events.validate'`.
- Method `validateForm(Request $request)` digunakan untuk memvalidasi data yang diterima dari formulir

## LATIHAN 4

#### A. Soal

Jelaskan validasi apa saja yang dilakukan pada data `location` dan `name`.

#### B. Screenshoot

#### C. Syntax

1	
---	--

#### D. Penjelasan

Location:

- `required`: Menentukan bahwa input field location harus diisi. Jika tidak diisi, validasi akan gagal.
- `alpha`: Menentukan bahwa input field location harus berupa huruf. Jika terdapat karakter selain huruf (misalnya angka atau karakter khusus), validasi akan gagal.
- `min:3`: Menentukan bahwa input field location harus memiliki panjang minimal 3 karakter. Jika kurang dari 3 karakter, validasi akan gagal.
- `max:50`: Menentukan bahwa input field location harus memiliki panjang maksimal 50 karakter. Jika lebih dari 50 karakter, validasi akan gagal.

Name:

- `required`: Menentukan bahwa input field name harus diisi. Jika tidak diisi, validasi akan gagal.
- `string`: Menentukan bahwa input field name harus berupa string. Jika bukan string, validasi akan gagal.
- `min:3`: Menentukan bahwa input field name harus memiliki panjang minimal 3 karakter. Jika kurang dari 3 karakter, validasi akan gagal.

- `max:50`: Menentukan bahwa input field name harus memiliki panjang maksimal 50 karakter. Jika lebih dari 50 karakter, validasi akan gagal.

Dengan aturan validasi ini, memastikan bahwa data yang diterima dari formulir memenuhi kriteria yang diinginkan, seperti format data dan panjang teks. Jika data tidak memenuhi aturan validasi, Laravel akan secara otomatis menangani validasi tersebut dan memberikan pesan kesalahan yang sesuai kepada pengguna.

## LATIHAN 5

### A. Soal

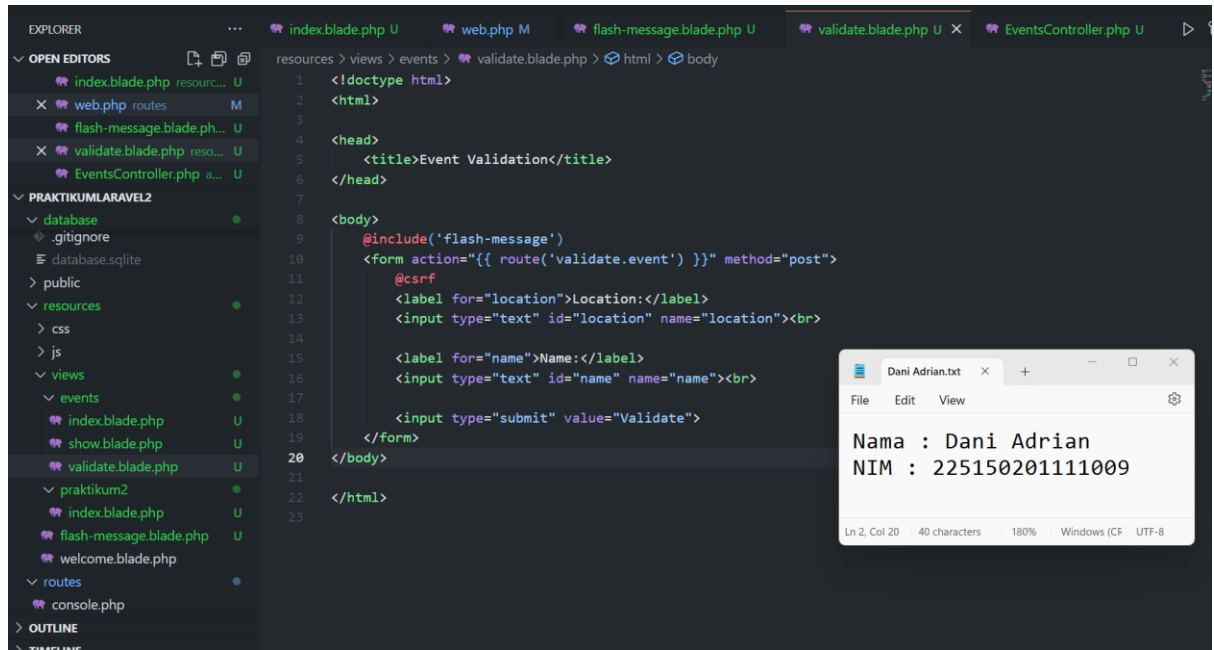
Sesuai isi *method* `showValidateForm` yang baru saja ditambahkan, buatlah file Blade dengan isi sbb:

```
<!doctype html>
<html>
<head>
    <title>Event Validation</title>
</head>
<body>
    @include('flash-message')
    <form action="{{ route('validate.event') }}"
method="post">
        @csrf
        <label for="location">Location:</label>
        <input type="text" id="location"
name="location"><br>

        <label for="name">Name:</label>
        <input type="text" id="name" name="name"><br>

        <input type="submit" value="Validate">
    </form>
</body>
</html>
```

### B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

Dengan menggunakan kode Blade diatas akan membuat halaman untuk menampilkan formulir validasi. Pengguna dapat memasukkan data 'location' dan 'name' di dalam formulir, dan data tersebut akan divalidasi saat formulir disubmit. Jika terjadi kesalahan validasi, pesan kesalahan akan ditampilkan.

## LATIHAN 6

### A. Soal

Pada formulir di atas, apakah fungsi @csrf? Apa yang terjadi jika baris tersebut (@csrf) dihapus?

### B. Screenshoot

### C. Syntax

1	
---	--

### D. Penjelasan

Directive `@csrf` dalam Blade adalah singkatan dari "Cross-Site Request Forgery". Ini adalah token keamanan yang digunakan oleh Laravel untuk melindungi aplikasi dari serangan CSRF (Cross-Site Request Forgery).

Jika baris `@csrf` dihapus dari formulir, maka Laravel akan menganggap permintaan tersebut tidak valid dan kemungkinan akan mengembalikan kesalahan "419 Page Expired". Hal ini terjadi karena Laravel secara default memerlukan token CSRF pada setiap permintaan POST, PUT, PATCH, atau DELETE. Tanpa token CSRF, Laravel tidak akan mengizinkan permintaan tersebut untuk dijalankan.

Jadi, jika menghapus `@csrf` dari formulir, mungkin akan mengalami kesalahan CSRF ketika mencoba mengirimkan formulir, dan permintaan kemungkinan akan ditolak oleh Laravel. Penting untuk selalu menyertakan `@csrf` dalam formulir untuk melindungi aplikasi dari serangan CSRF.

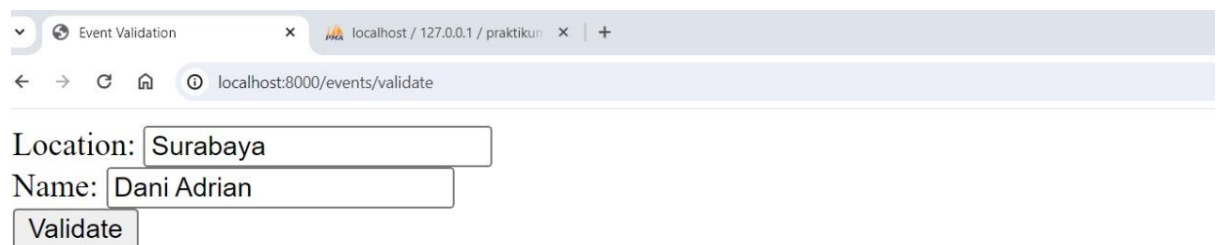
## LATIHAN 7

### A. Soal

Jalankan aplikasi, tempelkan tiga buah screenshot, yaitu: screenshot formulir validasi, screenshot ketika data tidak valid, dan screenshot ketika data valid.

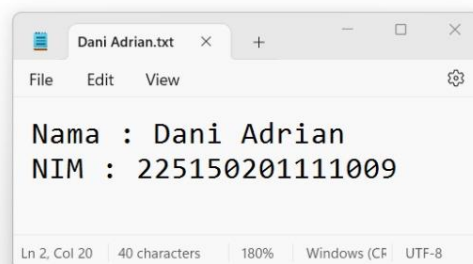
### B. Screenshoot

Formulir Validasi



Location:

Name:



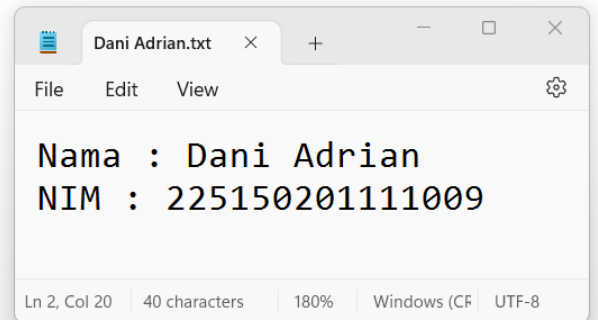
Data tidak Valid

## Data tidak valid

Location:

Name:

Validate



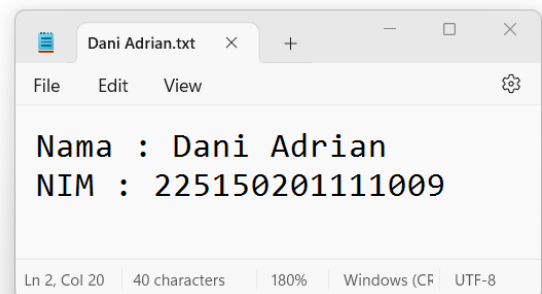
Data Valid

## Data valid

Location:

Name:

Validate



## C. Syntax

1	
---	--

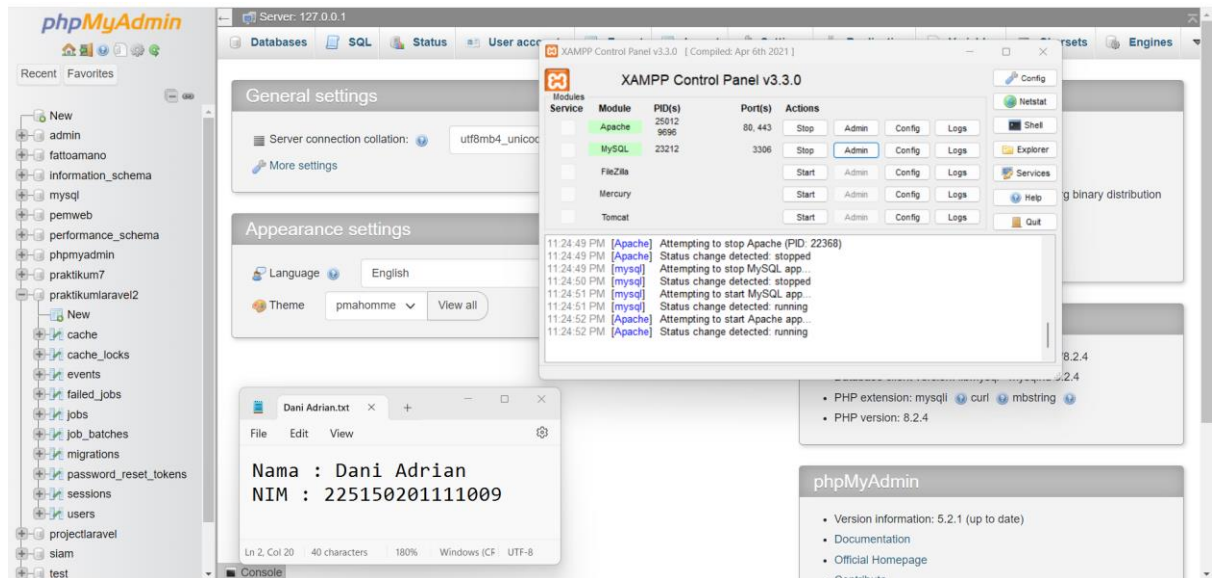
## D. Penjelasan

## 6. Menambahkan Authentication dengan Breeze

### LATIHAN 1

#### A. Soal

Pastikan MySQL Server dalam keadaan berjalan.



#### B. Screenshoot

Jika Ada

#### C. Syntax

1	
---	--

#### D. Penjelasan

### LATIHAN 2

#### A. Soal

Buatlah salinan file rute (web.php) untuk mengantisipasi hilangnya rute-rute yang telah didefinisikan sebelumnya.

#### B. Screenshoot

#### C. Syntax

1	
---	--

## D. Penjelasan

## LATIHAN 3

### A. Soal

Buka CMD/Terminal dan jalankan perintah-perintah berikut secara berurutan pada project root directory:

```
composer require laravel/breeze --dev
php artisan breeze:install
npm install
npm run dev
php artisan migrate
```

### B. Screenshoot

composer require laravel/breeze --dev

The screenshot shows a terminal window with the following output for the command `composer require laravel/breeze --dev`:

```
C:\xampp\htdocs\praktikumLaravel2>composer require laravel/breeze --dev
./composer.json has been updated
Running composer update laravel/breeze
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/breeze (v2.0.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/breeze (v2.0.3)
- Installing laravel/breeze (v2.0.3): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

laravel/breeze ..... DONE
laravel/sail ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE

$5 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

 INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^2.0 for laravel/breeze
C:\xampp\htdocs\praktikumLaravel2>
```

Overlaid on the terminal is a text box titled "Dani Adrian.txt" containing the following text:

```
Nama : Dani Adrian
NIM : 225150201111009
```

php artisan breeze:install



```
C:\xampp\htdocs\praktikumLaravel2>php artisan breeze:install

Which Breeze stack would you like to install?
Blade with Alpine ..... blade
Livewire (Volt Class API) with Alpine ..... livewire
Livewire (Volt Functional API) with Alpine ..... livewire-functional
React with Inertia ..... react
Vue with Inertia ..... vue
API only ..... api
> blade

Would you like dark mode support? (yes/no) [no]
> yes

Which testing framework do you prefer? [PHPUnit]
Pest ..... 0
PHPUnit ..... 1
> PHPUnit

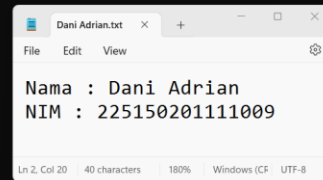
[INFO] Installing and building Node dependencies.

added 143 packages, and audited 144 packages in 23s
36 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

> build
> vite build

    vite v5.2.11 building for production...
transforming...
  ✓ 54 modules transformed.
rendering chunks...
computing gzip size...
public/build/manifest.json 0.27 kB | gzip: 0.15 kB
```



npm install

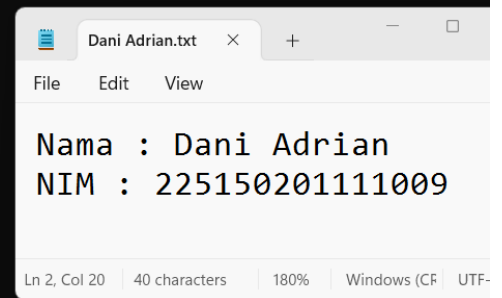
```
C:\xampp\htdocs\praktikumLaravel2>npm install

up to date, audited 144 packages in 646ms

36 packages are looking for funding
  run `npm fund` for details

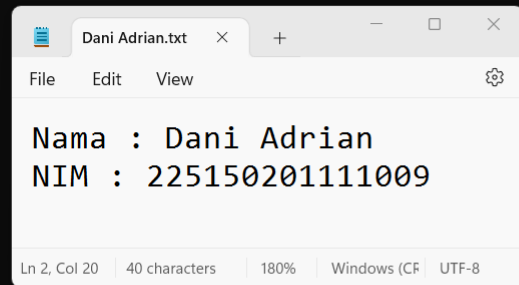
found 0 vulnerabilities

C:\xampp\htdocs\praktikumLaravel2>
```



npm run dev

```
VITE v5.2.11 ready in 180 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help  
LARAVEL v11.7.0 plugin v1.0.4  
→ APP_URL: http://praktikumlaravel2.test
```

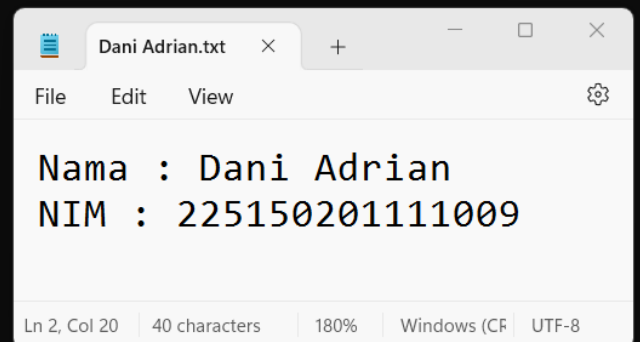


php artisan migrate

```
C:\xampp\htdocs\praktikumLaravel2>php artisan migrate
```

```
INFO Nothing to migrate.
```

```
C:\xampp\htdocs\praktikumLaravel2>
```



### C. Syntax

#### D. Penjelasan

Penjelasan dari poin C menjelaskan alur program dan bagian-bagian dari code program (per modul/method)

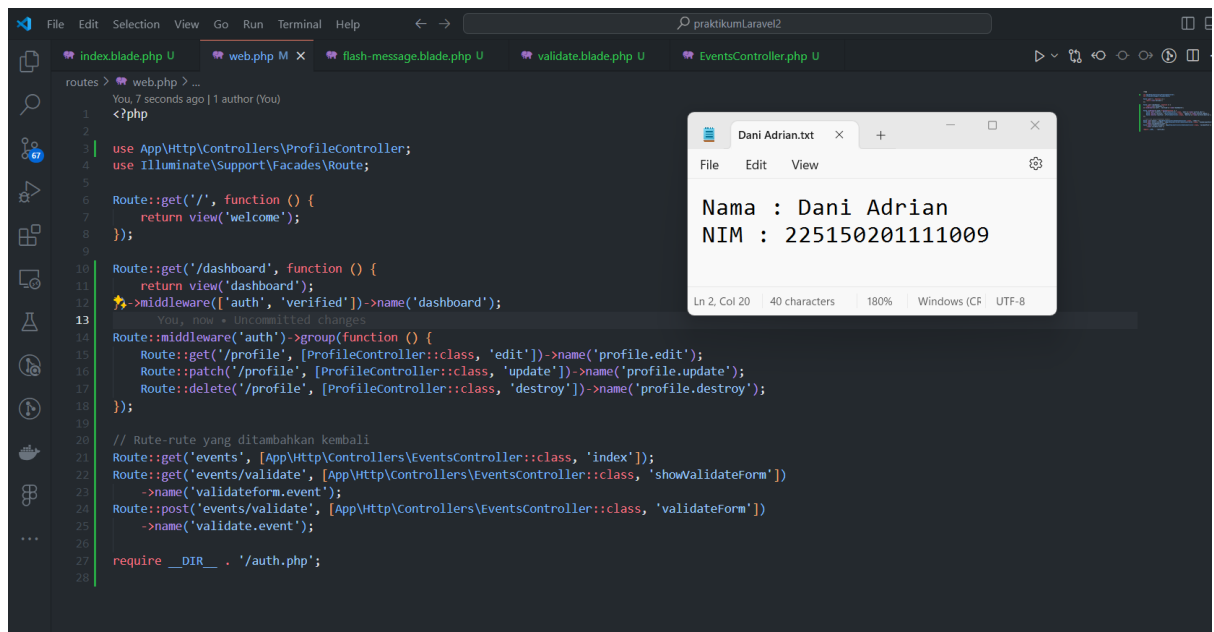
- `composer require laravel/breeze --dev`  
Perintah ini menginstal Laravel Breeze sebagai dependensi pengembangan (development dependency) dalam proyek Laravel. Laravel Breeze adalah paket yang menyediakan otentikasi dan manajemen sesi sederhana untuk proyek Laravel.
- `php artisan breeze:install`  
Setelah menginstal Laravel Breeze, kemudian jalankan perintah ini untuk menginstal dan mengkonfigurasi otentikasi Breeze dalam proyek. Ini akan menambahkan rute-rute otentikasi, mengkonfigurasi autentikasi dan manajemen sesi, serta membuat beberapa file Blade untuk tampilan otentikasi.
- `npm install`  
Ini adalah perintah untuk menginstal semua dependensi Node.js yang didefinisikan dalam file `package.json` proyek. Dependensi ini umumnya terkait dengan pengembangan front-end, seperti Laravel Mix, Vue.js, atau Bootstrap.
- `npm run dev`  
Setelah menginstal dependensi Node.js, kemudian gunakan perintah ini untuk menjalankan proses kompilasi asset untuk pengembangan. Ini akan mengompilasi file-file CSS dan JavaScript ke dalam direktori `public` proyek.
- `php artisan migrate`  
Perintah untuk menjalankan semua migrasi database yang tertunda. Migrasi digunakan untuk membuat dan memodifikasi skema database sesuai dengan definisi yang diberikan dalam file migrasi. Dalam konteks ini, perintah ini akan membuat tabel-tabel yang diperlukan untuk menyimpan informasi otentikasi seperti pengguna, sesi, dan token otentikasi.

## LATIHAN 4

### A. Soal

Buka file `web.php` pada folder `routes`, periksa apakah rute-rute yang sebelumnya dibuat telah hilang atau masih ada. Bila hilang, maka tambahkan kembali rute-rute yang hilang tersebut.

## B. Screenshoot



## C. Syntax

1	
---	--

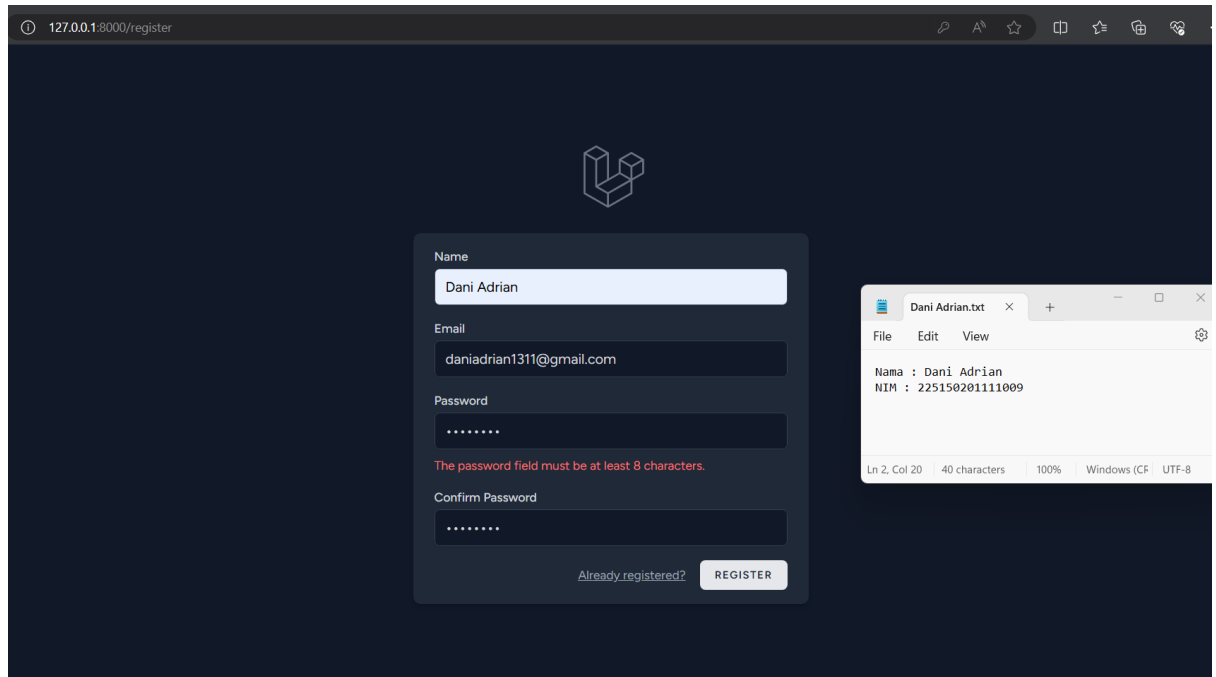
## D. Penjelasan

## LATIHAN 5

### A. Soal

Jalankan aplikasi, lakukan registrasi pengguna baru melalui path /register.

### B. Screenshoot



### C. Syntax

1	
---	--

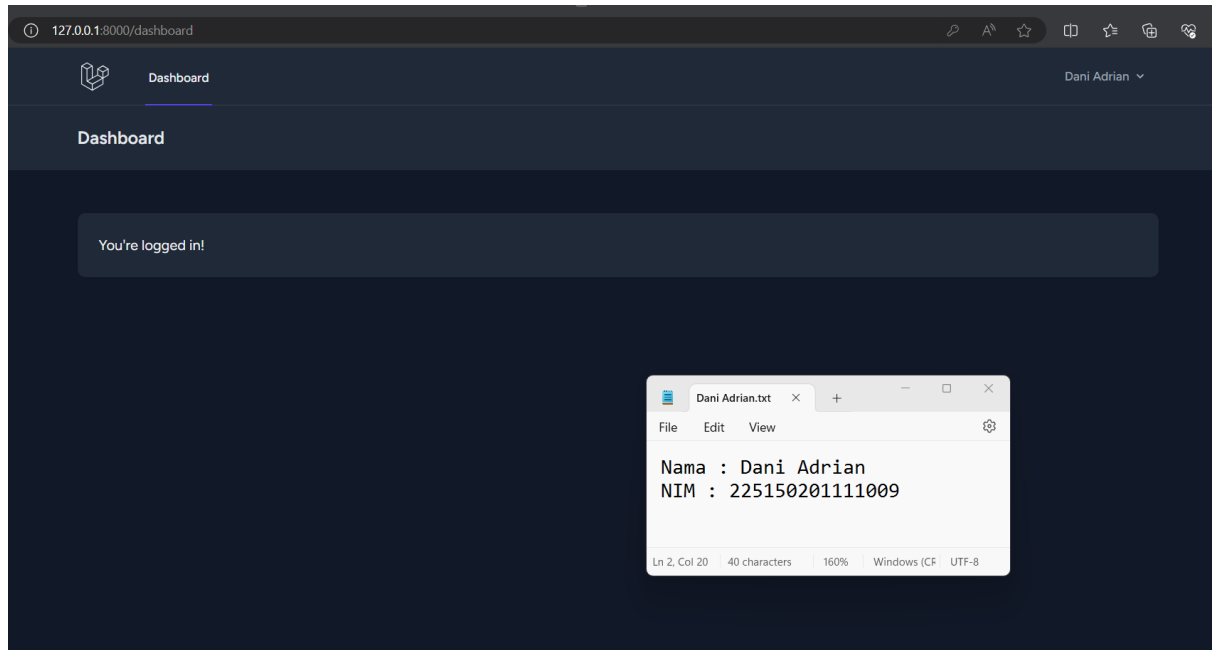
### D. Penjelasan

## LATIHAN 6

### A. Soal

Tempelkan screenshot halaman Dashboard yang tampil setelah proses registrasi berhasil.

### B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

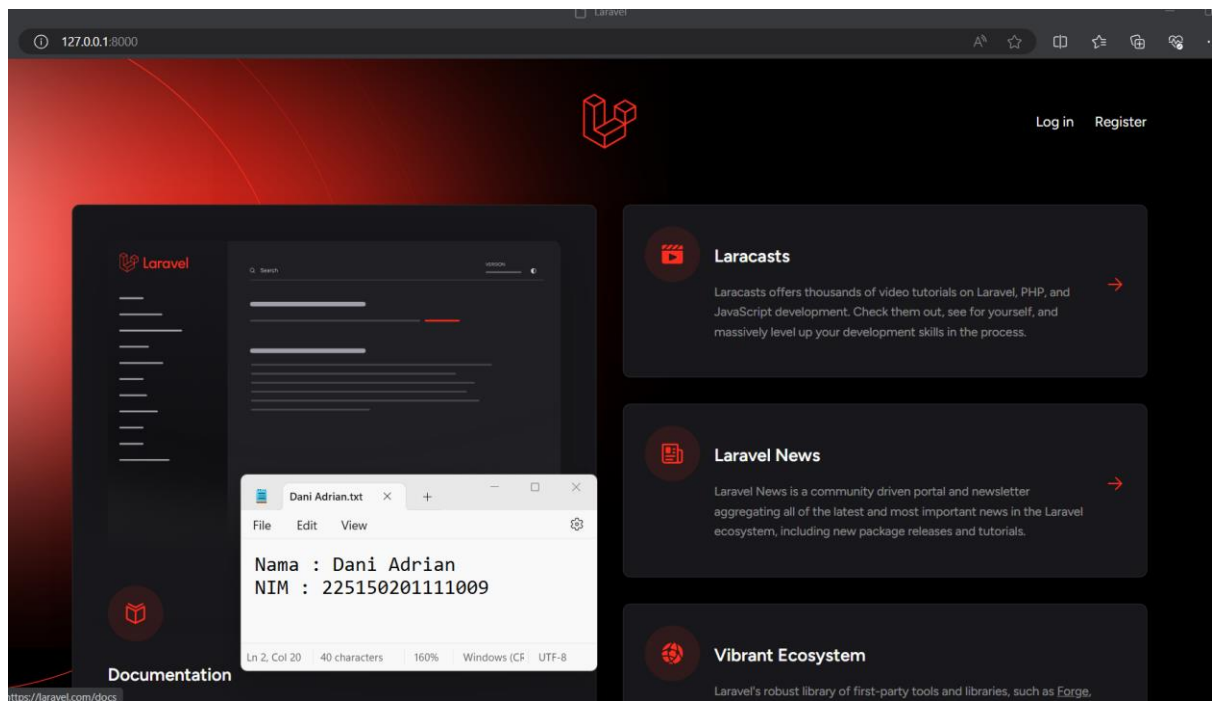
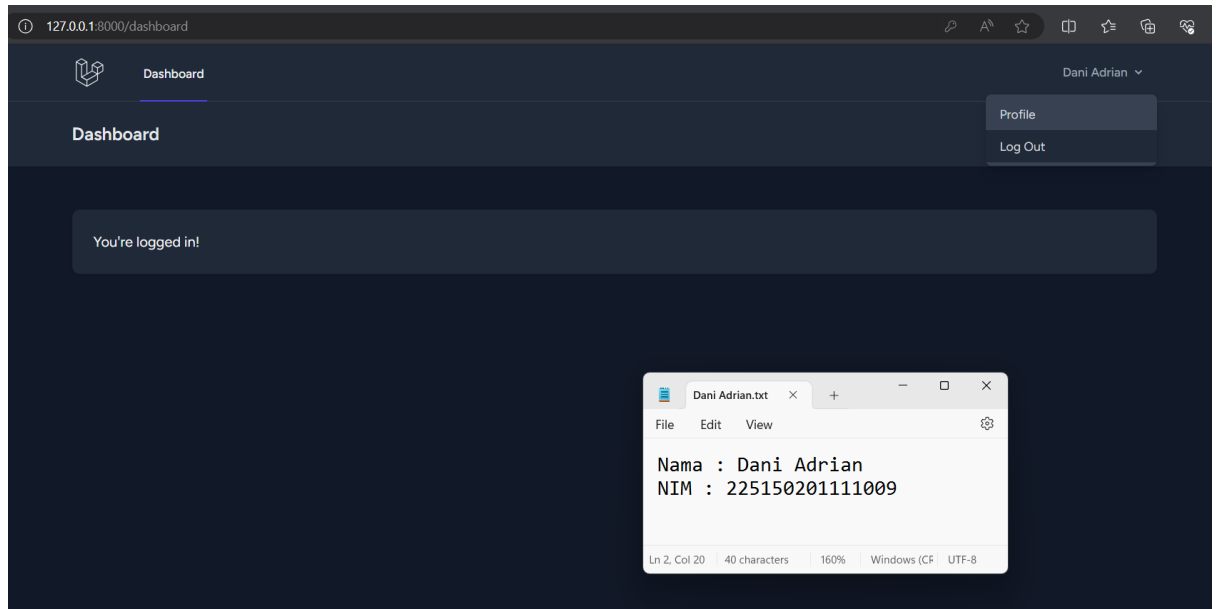
Penjelasan dari poin C menjelaskan alur program dan bagian-bagian dari code program (per modul/method)

## LATIHAN 7

### A. Soal

Lakukan proses logout melalui menu yang ada pada halaman Dashboard tersebut.

### B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

## LATIHAN 8

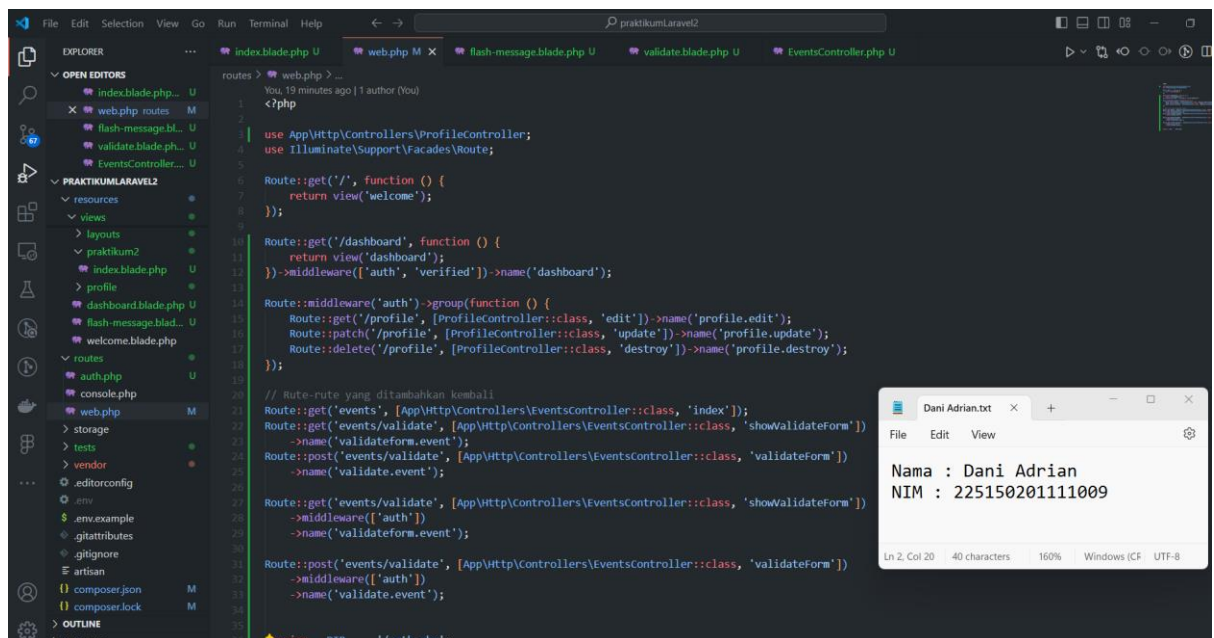
### A. Soal

Selanjutnya kita akan melindungi halaman validasi yang sebelumnya telah kita buat (validateform.event dan validate.event) sehingga hanya bisa diakses oleh pengguna yang sedang logged in. Sisipkan kode yang bertanda khusus berikut ini pada rute:

```
Route::get('events/validate',
[App\Http\Controllers\EventsController::class,
'showValidateForm'])
->middleware(['auth'])
->name('validateform.event');

Route::post('events/validate',
[App\Http\Controllers\EventsController::class,
'validateForm'])
->middleware(['auth'])
->name('validate.event');
```

## B. Screenshoot



## C. Syntax

1
---

## D. Penjelasan

`->middleware(['auth'])` adalah middleware yang menentukan bahwa rute tersebut hanya dapat diakses oleh pengguna yang sudah terautentikasi (logged in). Middleware auth adalah bagian dari Laravel yang memastikan bahwa pengguna hanya dapat mengakses rute tertentu jika mereka telah login ke aplikasi.

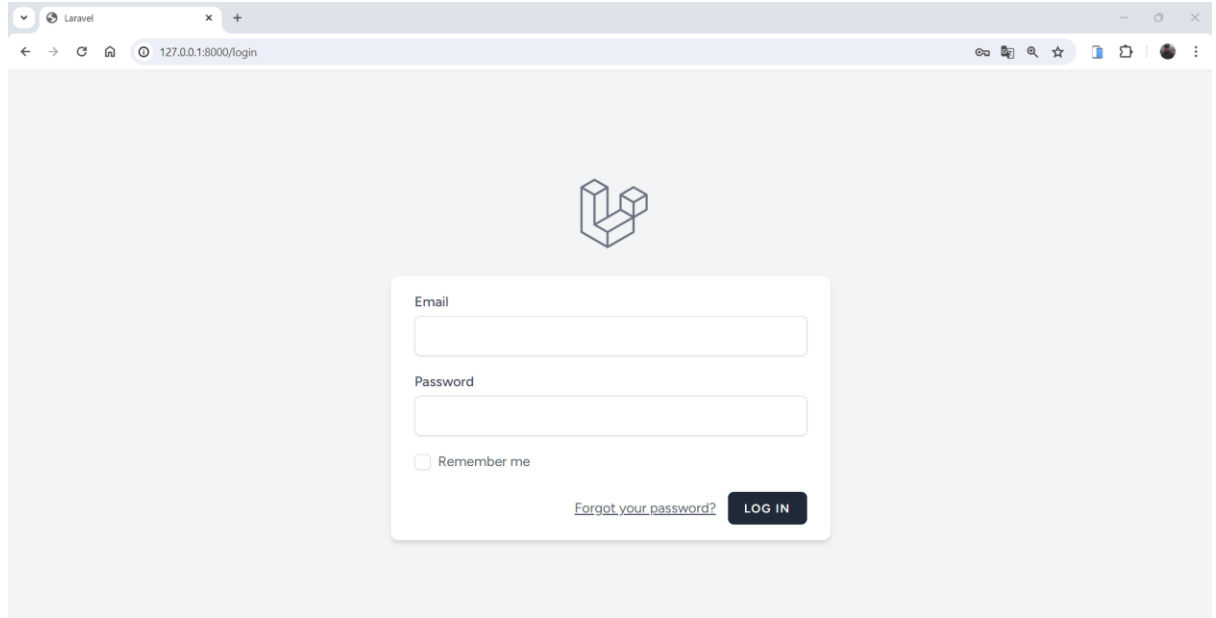
## LATIHAN 9



### A. Soal

Dalam posisi tidak logged in, akses halaman formulir validasi. Apa yang terjadi?

### B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

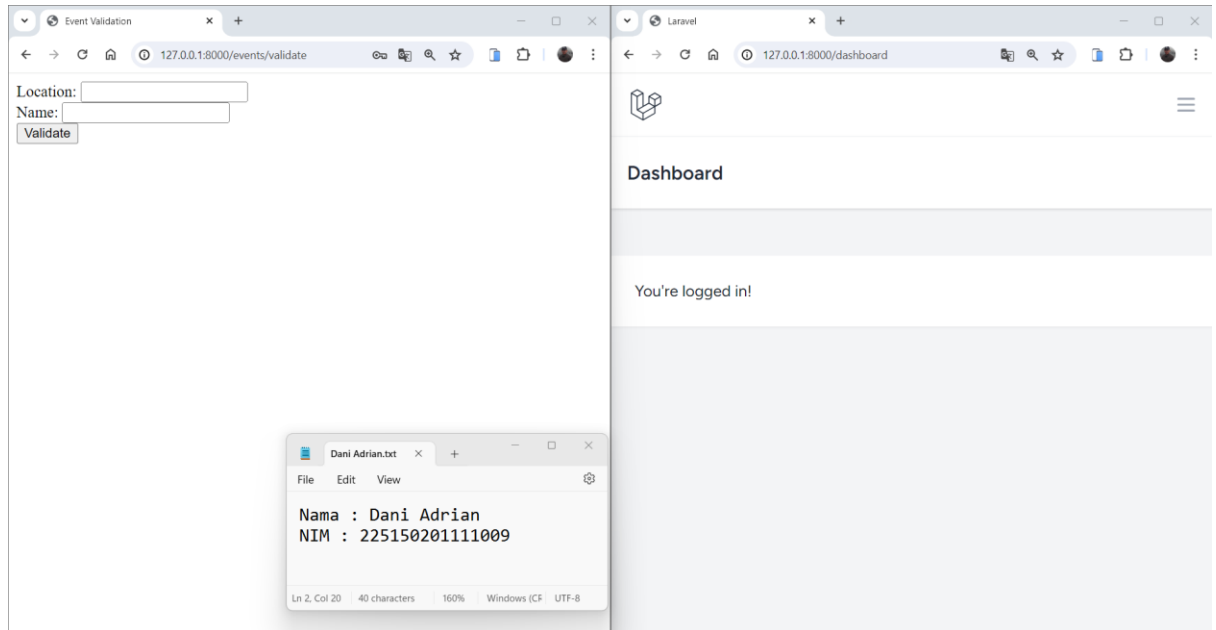
Diarahkan ke halaman login.

## LATIHAN 10

### A. Soal

Dalam posisi logged in, akses halaman formulir validasi. Apa yang terjadi?

### B. Screenshoot



### C. Syntax

1	
---	--

### D. Penjelasan

Dapat memvalidate location dan name