Framework Web

Praktikum 1

Materi :

1. Pendahuluan Laravel
2. Pengenalan Arsitektur MVC (*Model View Controller*)
3. Instalasi Laravel
4. Route
5. View
6. Blade

Tujuan Praktikum :

1. Mahasiswa mengenal *framework* Laravel dan memahami cara meng*install* Laravel
2. Mahasiswa mampu memahami cara penggunaan *route* untuk menentukan kemana proses akan dibawa, apakah ke *controller* atau *view*
3. Mahasiswa mampu memahami *view* untuk menampilkan halaman
4. Mahasiswa mampu memahami penggunaan *blade* yang merupakan sebuah template *engine* bawaan Laravel

A. Konsep

1. Pendahuluan Laravel

1.1 Pengertian Framework

Framework merupakan sekumpulan kode program yang siap digunakan dengan mengikuti aturan penulisan tertentu yang memiliki tujuan untuk memudahkan penggunannya serta dapat mempercepat pembuatan website. Dalam bahasa pemrograman PHP terdapat banyak sekali *framework* yang bisa digunakan yaitu Laravel, CodeIgniter, Symfony, Yii, dsb.

1.2 Apa Itu Laravel

Laravel adalah sebuah *framework* PHP yang gratis dan *open source* untuk mengembangkan website modern dengan menggunakan bahasa pemrograman PHP. Laravel telah menyediakan berbagai macam sintaks yang elegan, memiliki berbagai

macam fitur yang siap digunakan dan berbagai fitur yang kompatibel dengan *library* ataupun *ekstensions*

2. Pengenalan Arsitektur MVC (*Model View Controller*)

Model-View-Controller atau biasa disingkat MVC adalah sebuah arsitektur dalam membuat sebuah aplikasi/website dengan cara memisahkan kode menjadi tiga bagian yang terdiri dari:

a. Model

Menyiapkan, mengatur, memanipulasi, dan mengorganisasikan data yang ada di *database* merupakan bagian tugas dari *Model*.

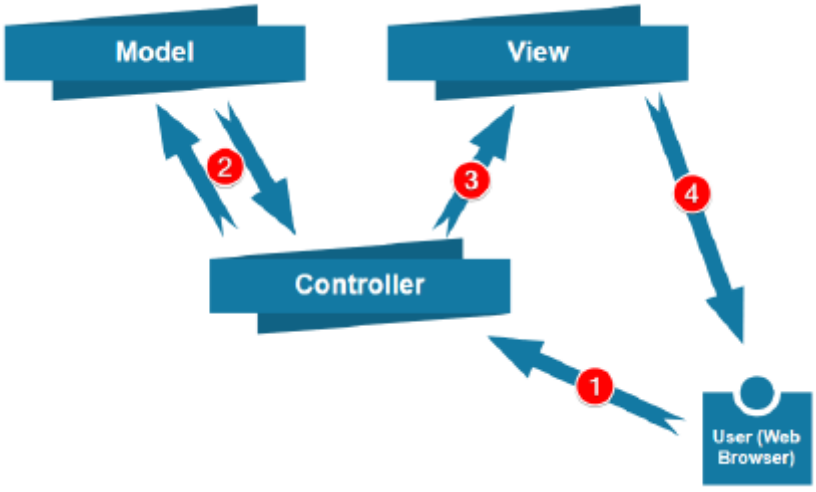
b. View

Menampilkan informasi dalam bentuk *User Interface* merupakan bagian tugas dari *View*.

c. Controller

Menghubungkan serta mengatur *Model* dan *View* agar dapat saling terhubung merupakan tugas dari *Controller*

Jika dibuat dalam bentuk diagram, maka alur proses dari sebuah arsitektur *Model-View-Controller* adalah sebagai berikut:



3. Instalasi Laravel

3.1 Requirement

Untuk menggunakan Laravel ada beberapa persyaratan yang harus dipenuhi:

a. Xampp

XAMPP merupakan perangkat lunak *open source* berbasis web server yang berisi berbagai program. Aplikasi ini mendukung berbagai sistem operasi seperti Linux, Windows, MacOS, dan Solaris. Fungsi XAMPP adalah sebagai server lokal/localhost, di dalamnya sudah mencakup program Apache, MySQL dan PHP. Xampp bisa di unduh melalui halaman resminya di https://www.apachefriends.org/download.html

b. Composer

Composer merupakan alat *dependency manager* yang digunakan untuk bahasa pemrograman PHP. Dengan kata lain, Composer adalah aplikasi yang diinstal ke perangkat untuk memfasilitasi developer menggunakan *library open source* milik orang lain ke dalam project yang sedang dibangun. Composer bisa di unduh melalui halaman resminya di https://getcomposer.org/download/

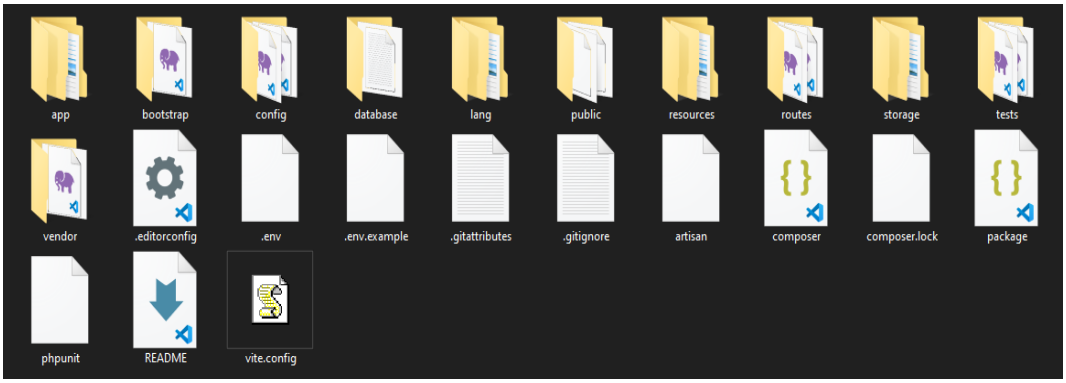
3.2 Cara Instalasi

Terdapat 2 cara yang bisa kita lakukan untuk melakukan instalasi Laravel:

a. Menggunakan perintah *Composer Create-Project*
b. Menggunakan *Laravel Installer*

3.3 Struktur Folder Laravel

Berikut ini merupakan struktur folder laravel dan penjelasan nya:



a. App: berisi berbagai file untuk aplikasi yang akan di bangun. Dalam folder inilah membuat controller dan model.
b. Bootstrap: berisi file app.php yang berfungsi sebagai bootstrap atau file pengaturan awal dari Laravel. Selain itu terdapat juga folder cache untuk meningkatkan performa aplikasi.
c. Config: berisi berbagai file konfigurasi Laravel
d. Database: berisi folder dan file yang 'mengurusi' database seperti migrations,factories dan seeds.
e. Lang: berisi konfigurasi *language* yang digunakan untuk mengatur bahasa pada aplikasi yang kita buat
f. Public: berisi file index.php sebagai file awal dari semua request ke aplikasi Laravel
g. Resources: berisi file resource 'mentah' seperti file CSS dan JavaScript. Di sini juga nantinya kita membuat view.
h. Route: berisi file yang akan menangani proses routing
i. Storage: berisi tempat penyimpanan file yang di *generate* oleh Laraval

j. Test: berisi file untuk proses testing seperti PHPUnit
k. Vendor: Berisi berbagai file internal framework Laravel

4. Route

   Route atau Routing berperan sebagai penghubung antara user dengan keseluruhan framework. Dalam Laravel, setiap alamat web yang kita ketik di web browser akan melewati route terlebih dahulu. Route-lah yang menentukan ke mana proses akan dibawa, apakah ke Controller atau ke View.

5. View

   View adalah komponen MVC yang menangani tampilan. Di dalam viewlah kode HTML, CSS dan juga JavaScript berada

6. Blade

   Blade adalah template engine bawaan Laravel. Secara sederhana, template engine berisi perintah tambahan untuk mempermudah pembuatan template. Template sendiri bisa disebut sebagai kerangka dasar tampilan.

   Secara garis besar, terdapat 2 fungsi utama blade di dalam Laravel:

   a. Mempersingkat penulisan perintah PHP.
   b. Pemecahan file template (proses pembuatan layout).

B. Latihan Konsep

1. Apa yang dimaksud dengan *framework*?
2. Jelaskan dengan menggunakan bahasa sendiri mengenai arsitektur mvc!
3. Sebutkan dan jelaskan bagaimana cara melakukan instalasi Laravel?
4. Sebutkan dan jelaskan struktur folder Laravel!
5. Apa yang dimaksud dengan *route*, *view*, dan *blade*?

C. Praktik

Asisten lab menjelaskan terkait praktikum:

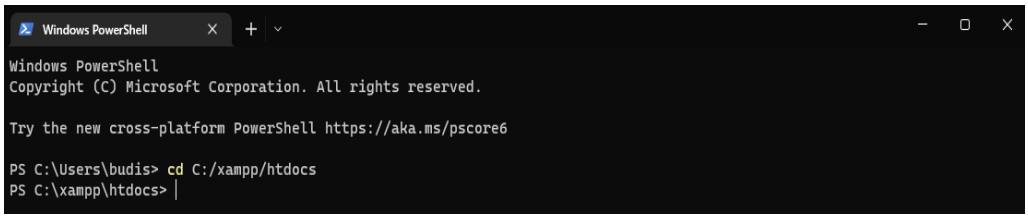
1. Cara Instalasi Laravel

   Terdapat 2 cara yang bisa kita lakukan untuk melakukan instalasi Laravel:

   a. Menggunakan perintah *Composer Create-Project*

      - Untuk menggunakan perintah *Composer Create-Project*. Silahkan untuk buka ***command prompt(cmd)***

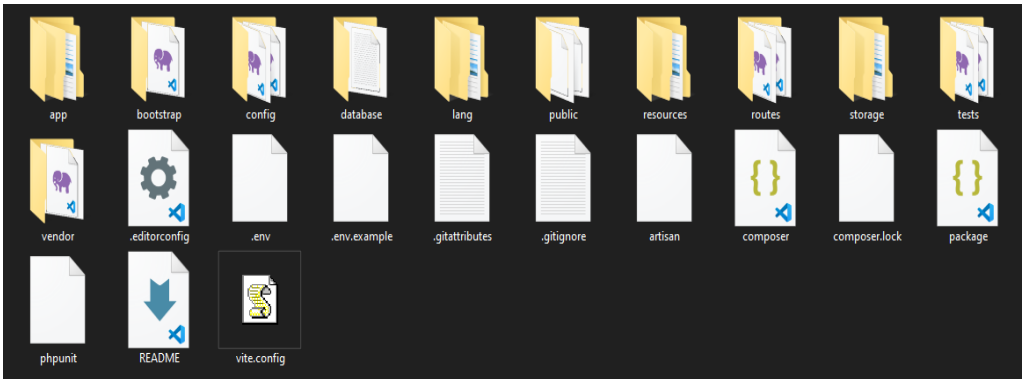
- Kemudian, pindah ke folder *htdocs* Xampp dengan menggunakan perintah ***cd C:/xampp/htdocs***



- Kemudian, masukkan perintah berikut:

Composer create-project –prefer-dist laravel/laravel framework-web1

- Pastikan komputer/laptop anda memiliki koneksi dan tunggu hingga proses instalasi selesai



b. Menggunakan *Laravel Installer*

- Masukkan perintah ***composer global require laravel/installer***

- Setelah berhasil, langkah selanjutnya kita menginstall laravel dengan memasukkan perintah ***laravel new framework-web1***

- Pastikan komputer/laptop anda memiliki koneksi dan tunggu hingga proses instalasi selesai

2. Route

2.1 Cara Membuat Route

Untuk membuat route, kita cukup menuliskan method static di dalam file ***routes/web.php***, seperti berikut ini:

```
Route::get( uri: '/store', action: function(){
    return "Selamat datang di halaman store Fasilkom";
});
```

Ketika alamat http://localhost:8000/store diakses, maka perintah return ‘Selamat datang di halaman store fasilkom' akan dijalankan.

2.2 Route Parameter

Dalam keadaan tertentu, terkadang kita menginginkan untuk mengambil nilai yang terdapat di dalam URL seperti mengambil nilai kode produk seperti berikut ini:

```
Route::get( uri: 'store/{kode_produk}', action: function($kode_produk){
    return "Laptop Macbook Pro M1 2021";
});
```

Ketika alamat http://localhost:8000/store/if2001 diakses, maka perintah return ‘Laptop Macbook Pro M1 2021' akan dijalankan. Selain 1 parameter kita juga bisa memasukkan 2 parameter atau bahkan lebih dan kita bisa mengambil nilai yang ada di URL dengan cara seperti berikut:

```
Route::get( uri: 'store/{kode_produk}/{nama_produk}', action: function($kode_produk, $nama_produk){
    return "Kode Produk: $kode_produk , Nama produk: $nama_produk";
});
```

2.3 Route Redirect

Untuk mengalihkan route satu ke route yang lain nya kita bisa menggunakan method ***Route::redirect*** seperti berikut:

```
Route::get( uri: '/about', action: function(){
    return "Ini merupakan halaman about";
});

Route::redirect( uri: '/tentang-kami', destination: '/about');
```

2.4 Route Group

Dalam suatu kondisi terdapat beberapa route yang ingin kita kelompokkan, untuk mengelompokkan route tersebut kita bisa menggunakan method ***group*** seperti berikut:

```
Route::prefix( prefix: '/store')->group( callback: function(){
    Route::get( uri: '/product', action: function(){
        return "Daftar Produk";
    });

    Route::get( uri: '/category/{nama_kategori}', action: function($nama_kategori){
        return "Nama kategori $nama_kategori";
    });
}); ← #35-43 Route::prefix('/store')->group
```

2.5 Route Fallback

Laravel memiliki fitur route khusus yang apabila dijalankan route tersebut tidak ditemukan maka secara *default* laravel akan mengarahkan ke halaman ***404 Not Found*** namun kita dapat menimpa halaman tersebut dengan menggunakan method ***fallback***

```
Route::fallback( action: function(){
    return "Halaman yang anda cari tidak ditemukan";
});
```

3. View

Laravel sudah memiliki *view* bawaan yang terlihat saat kita mengakses halaman awal. Pemanggilan *view* berasal dari kode yang ada di ***routes/web.php*** seperti berikut:

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

3.1 Cara Membuat View

Untuk membuat view terdapat 2 cara yang harus kita lakukan yakni:

a. Buat route yang akan mengembalikan view
b. Buat file view di folder ***resources\views*** dengan format ***nama.blade.php***

Sebagai contoh ingin menampilkan view store

a. Membuat route terlebih dahulu

```
Route::get( uri: '/store', action: function(){
    return view( view: 'store');
});
```

b. Membuat view ***store.blade.php***

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Store</title>
</head>
<body>
    <h1>Ini halaman store</h1>
</body>
</html>
```

3.2 Cara Mengirim Data ke View

Untuk mengirim data ke view terdapat 2 cara yang dapat kita lakukan yakni:

a. Menulis data sebagai argument kedua dari function ***view()***

b. Menggunakan method ***with()***

Berikut ini merupakan cara mengirim data ke view sebagai argument:

```
Route::get( uri: '/store',  action: function(){
    return view( view: 'store', data: [
        "produk01" => "Macbook"
    ]);
}); ← #20-24 Route::get
```

Selanjutnya agar data yang dikirim bisa di baca oleh *view* maka kita bisa memasukkan kode berikut di dalam file ***store.blade.php***

```
<body>
    <h1>Ini halaman store</h1>
    <p><?php echo $produk01; ?></p>
</body>
```

Berikut ini merupakan cara mengirim data ke view dengan menggunakan method ***with()***

```
Route::get( uri: '/store',  action: function(){
    $list_product = ["Macbook","Xiaomi","Iphone","Acer"];
    return view( view: 'store')->with( key: 'product',  value: $list_product);
});
```

Selanjutnya agar data yang dikirim bisa di baca oleh *view* maka kita bisa memasukkan kode berikut di dalam file ***store.blade.php***

```
<body>
    <h1>Ini halaman store</h1>
    <ol>
        <?php
        foreach ($product as $value) {
            echo "<li>$value</li>";
        }
        ?>
    </ol>
</body>
```

4. Blade

4.1 Menampilkan Data Dengan Blade

Silahkan buat route dengan kode seperti berikut:

```php
Route::get( uri: '/store',  action: function(){
    $kode_produk = "IF2001";
    $nama_produk = "Laptop";
    $harga_produk = 125000;
    return view( view: 'store',  data: compact(
        var_name: 'kode_produk',
        var_names[0]: 'nama_produk',
        var_names[1]: 'harga_produk'
    ));
});
```

Dan buat tampilan view dengan memasukkan kode seperti berikut:

```php
<body>
    <div class="container text-center mt-3 py-3 bg-white">
        <h1 class="bg-dark px-3 py-1 text-white d-inline-block">
            {{$kode_produk}}
        </h1>
        <h1 class="bg-dark px-3 py-1 text-white d-inline-block">
            {{$nama_produk}}
        </h1>
        <h1 class="bg-dark px-3 py-1 text-white d-inline-block">
            {{$harga_produk}}
        </h1>
    </div>
</body>
```

4.2 If Else

Selain menampilkan data, blade juga menyediakan penulisan singkat untuk struktur kondisi PHP lain seperti if else, switch

```php
<body>
    <div class="container text-center mt-3 pt-3 bg-white">
        <h1 class="bg-dark px-3 py-1 text-white d-inline-block">{{ $kode_produk }}</h1>
        <h1 class="bg-dark px-3 py-1 text-white d-inline-block">{{ $nama_produk }}</h1>
        <h1 class="bg-dark px-3 py-1 text-white d-inline-block">{{ $harga_produk }}</h1>
        <br>
    @if ($harga_produk ≥ 100000)
        <div class="alert alert-danger d-inline-block">
            Harga produk terlalu mahal
        </div>
    @else
        <div class="alert alert-success d-inline-block">
            Harga Produk Murah
        </div>
    @endif
    </div>
</body>
```

4.3 Foreach

Perulangan foreach lebih banyak kita pakai dalam Laravel. Alasannya karena nilai yang dikirim dari route (dan nantinya dari controller), lebih banyak berbentuk array. Perulangan foreach sangat cocok untuk memproses array dibandingkan for dan while.

```
<body>
    <h1>Ini halaman store</h1>
    <ol>
        @foreach ($products as $product)
            <li>{{$product}}</li>
        @endforeach
    </ol>
</body>
</html>
```

4.4 Forelse

Forelse adalah perulangan khusus yang disediakan blade dengan struktur @forelse, @empty dan @endforelse. Ini sebenarnya gabungan dari proses pemeriksaan apakah sebuah array berisi nilai atau tidak

```
<body>
    <h1>Ini halaman store</h1>
    <ol>
        @forelse ($products as $product)
            <li>{{$product}}</li>
        @empty
            Tidak ada produk yang tersedia
        @endforelse
    </ol>
</body>
```

4.5 Membuat Layout Dengan Blade

1. Pertama membuat route seperti berikut

```
Route::get( uri: '/product',  action: function(){
    $list_product = ["Macbook","Xiaomi","Iphone","Acer"];
    return view( view: 'store')->with( key: 'products',  value: $list_product);
});

Route::get( uri: '/category',  action: function () {
    $list_category = ["Laptop", "Handphone", "Komputer"];
    return view( view: 'store')->with( key: 'products',  value: $list_category);
});
```

2. Membuat file blade **product.blade.php**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Product</title>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
        crossorigin="anonymous">
</head>
<body>
    <nav class="navbar navbar-expand-sm navbar-dark bg-dark">
        <div class="container">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link active" href="/product">Produk</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/category">Kategori</a>
                </li>
            </ul>
        </div>
    </nav>
```

```
    <div class="container text-center mt-3 p-4 bg-white">
        <h1>Data Produk</h1>
        <div class="row">
            <div class="col-sm-12 col-md-8 m-auto">
                <ol class="list-group">
                    @forelse ($products as $product)
                    <li class="list-group-item">{{$product}}</li>
                    @empty
                    <div class="alert alert-dark d-inline-block">Tidak ada data...</div>
                    @endforelse
                </ol>
            </div>
        </div>
    </div>
</body>
</html>
```

3. Membuat file blade **category.blade.php**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Category</title>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
        crossorigin="anonymous">
</head>
<body>
    <nav class="navbar navbar-expand-sm navbar-dark bg-dark">
        <div class="container">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="/product">Produk</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link category" href="/category">Kategori</a>
                </li>
            </ul>
        </div>
    </nav>
```

```
    <div class="container text-center mt-3 p-4 bg-white">
        <h1>Kategori</h1>
        <div class="row">
            <div class="col-sm-12 col-md-8 m-auto">
                <ol class="list-group">
                    @forelse ($categories as $category)
                    <li class="list-group-item">{{$category}}</li>
                    @empty
                    <div class="alert alert-dark d-inline-block">Tidak ada data...</div>
                    @endforelse
                </ol>
            </div>
        </div>
    </div>
</body>

</html>
```

4.6 View Include

Untuk mempermudah proses pembuatan layout atau template adalah memecah bagian yang berulang menjadi file terpisah, kemudian di satukan kembali di view yang membutuhkan dengan cara seperti berikut:

1. Buatlah folder dengan nama layout, kemudian buat file ***header.blade.php*** di dalam folder tersebut
2. Masukkan kode berikut di dalam file ***header.blade.php***

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Category</title>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.
    min.css" rel="stylesheet"
        integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/
        et3URy9Bv1WTRi" crossorigin="anonymous">
</head>
<body>
    <nav class="navbar navbar-expand-sm navbar-dark bg-dark">
        <div class="container">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="/product">Produk</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link category" href="/category">Kategori</a>
                </li>
            </ul>
        </div>
    </nav>
```

3. Kemudian pada file ***category.blade.php*** ubah menjadi seperti berikut

```
@include('layout.header')
    <div class="container text-center mt-3 p-4 bg-white">
        <h1>Kategori</h1>
        <div class="row">
            <div class="col-sm-12 col-md-8 m-auto">
                <ol class="list-group">
                    @forelse ($categories as $category)
                    <li class="list-group-item">{{$category}}</li>
                    @empty
                    <div class="alert alert-dark d-inline-block">Tidak ada
                    data...</div>
                    @endforelse
                </ol>
            </div>
        </div>
    </div>
</body>

</html>
```

4.7 Layout Extends

Blade menyediakan teknik alternatif yang dalam kebanyakan situasi lebih praktis daripada menggunakan teknik include. Caranya adalah dengan "menurunkan sebuah file view ke view lain, atau bisa juga disebut teknik men-extends sebuah view

1. Hapus file ***header.blade.php*** sebelumnya, kemudian buat file ***master.blade.php*** di dalam folder ***layouts***

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>@yield('title')</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.
    min.css" rel="stylesheet"
        integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/
        et3URy9Bv1WTRi" crossorigin="anonymous">
</head>
<body>
    <nav class="navbar navbar-expand-sm navbar-dark bg-dark">
        <div class="container">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="/product">Produk</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link category" href="/category">Kategori</a>
                </li>
            </ul>
        </div>
    </nav>
    @yield('content')
    </body>
</html>
```

2. Kemudian pada file ***category.blade.php*** ubah kode menjadi seperti berikut

```
@extends('layout.master')
@section('title','Category')

@section('content')
<div class="container text-center mt-3 p-4 bg-white">
    <h1>Kategori</h1>
    <div class="row">
        <div class="col-sm-12 col-md-8 m-auto">
            <ol class="list-group">
                @forelse ($categories as $category)
                <li class="list-group-item">{{$category}}</li>
                @empty
                <div class="alert alert-dark d-inline-block">Tidak ada data...</
                div>
                @endforelse
            </ol>
        </div>
    </div>
</div>
@endsection
```

D. Tugas Praktik

1. Buatlah sebuah project laravel baru dengan ketentuan sebagai berikut:
    a. Nama project **sistem_informasi_akademik_npm**
    b. Terdiri dari 3 route (dosen, mahasiswa, dan matakuliah)
    c. Membuat 4 view yang berbeda dan menggunakan layout blade *engine* (1 view sebagai template master)
    d. Masing-masing view harus memiliki 10 minimal data yang dibuat dengan menggunakan array di dalam *route* dan dikirim ke *view* (data bebas)
    e. Buat tampilan website semenarik mungkin menggunakan bootstrap