KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 9 (sembilan)

JOBSHEET 09
AUTHENTICATION, MIDDLEWARE

Sebelumnya kita sudah membahas mengenai *Migration, Seeder*, *DB Façade, Query Builder,* dan sedikit tentang *Eloquent ORM* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Dalam pertemuang kali ini kita akan memahami tentang bagainama cara menampilkan data, mengubah data, dan menghapus data menggunakan teknik Eloquent.

Sesuai dengan `Studi Kasus PWL.pdf`.
Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS.**

Project **PWL_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajarai

Laravel Authentication

Laravel Authentication dipergunakan untuk memproteksi halaman atau fitur dari web yang hanya diakses oleh orang tertentu yang diberikan hak. Fitur seperti ini biasanya ditemui di sistem yang memiliki fitur administrator atau sistem yang memiliki pengguna yang boleh menambahkan datanya.

Laravel membuat penerapan otentikasi sangat sederhana dan telah menyediakan berbagai fitur yang dapat dimanfaatkan tanpa perlu melakukan penambahan instalasi modul tertentu. File konfigurasi otentikasi terletak di config / auth.php, yang berisi beberapa opsi yang terdokumentasi dengan baik untuk mengubah konfigurasi dari layanan otentikasi.

Pada intinya, fasilitas otentikasi Laravel terdiri dari "guards" dan "providers". Guards menentukan bagaimana pengguna diautentikasi untuk setiap permintaan. Misalnya, Laravel mengirim dengan guards untuk sesi dengan menggunakan penyimpanan session dan cookie.

Middleware

Middleware adalah lapisan perantara antara permintaan ***route HTTP*** yang masuk dan ***action*** dari `Controller` yang akan dijalankan. **Middleware** memungkinkan kita untuk meakukan berbagai tugas baik itu sebelum ataupun sesudah tindakan dilakukan. Kita juga dapat menggunakan *tool* **CLI** untuk membuat sebuah **Middleware** dalam **Laravel**. Beberapa contoh penggunaan **Middleware** meliputi autentikasi, validasi, manipulasi permintaan, dan lainnya. Berikut di bawah ini adalah manfaat dari **Middleware :**

- **Keamanan** : dalam **Middleware** memungkinkan kita untuk memverifikasi apakah pengguna sudah diautentikasi sebelum mengakses halaman tertentu. Dengan demikian, kita dapat melindungi data sensitif dan mengontrol hak akses pengguna.
- **Pemfilteran Data** : **Middleware** dapat digunakan untuk memanipulasi data permintaan sebelum sebuah ***action*** dalam *controller* dilakukan. Misalnya, kita dapat memeriksa terlebih dahulu data yang dikirim oleh pengguna sebelum data tersebut diproses lebih lanjut atau kita ingin memodifikasi data yang akan dikirim lalu kita dapat memeriksa ulang data yang akan dikirim oleh pengguna sebelum data tersebut diproses.
- **Logging dan Audit** : **Middleware** juga dapat digunakan untuk mencatat aktivitas pengguna atau melakukan audit terhadap permintaan yang masuk. Ini dapat membantu dalam pemantauan dan analisis aplikasi.

Praktikum – Auth dan Middleware

Praktikum ini lanjutan dari pertemuan ke-4. Tabel yang digunakan untuk praktikum ini menggunakan m_user. Sebelum melakukan praktikum jalankan perintah di bawah ini

a. Start Laragon atau Xampp
b. Jalankan "php artisan serve"
c. Jalankan "npm run dev"

Jika 3 tahap diatas sudah dilakukan kemudian lakukan langkah-langkah praktikum di bawah ini

1. Sebelum melangkah lebih lanjut setting file pada **config/auth.php** ubah pada bagian model menjadi model yang telah kita buat sebelumnya untuk menyimpan username dan password

```
    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\Models\UserModel::class,
        ],
```

2. Membuat middleware dengan perintah pada cmd atau terminal

```
php artisan make:middleware Cek_login
```

Setelah itu cek didalam folder app/Http/Middleware apakah sudah ada file bernama **Cek_login.php**

3. Kemudian tambahkan kode dibawah ini, letakkan di function handle()

```
?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades Auth;
use\Symfony\Component\HttpFoundation\Response;

class Cek_login
{
    /**
     * Handle an incoming request.
     *
     * @param   Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */       \
    public function handle(Request $request, Closure $next, $roles): Response
    {
        // cek sudah login atau belum . jika belum kembali ke halaman login
        if   Auth::check()) {
          (!return redirect('login');
        }
        // simpan data user pada variabel $user
        $user = Auth::user();

        // jika user memiliki level sesuai pada kolom pada lanjutkan request
        if ($user->level_id == $roles) {
            return $next($request);
        }

        // jika tidak memiliki akses maka kembalikan ke halaman login
        return redirect('login')->with('error', 'Maaf anda tidak memiliki akses');
    }
}
```

4. Ketika sudah membuat middleware, langkah berikutnya harus registrasikan pada **kernel.php**. Agar middleware dapat dibaca oleh sistem. Buka folder **App/Http/Kernel.php** dan tambahkan code pada variabel $middlewareAliases seperti dibawah

```php
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
    'signed' => \App\Http\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
    // tambahkan middleware alias dengan nama cek_login
    'cek_login' => \App\Http\Middleware\Cek_login::class,
];
```

5. Membuat Controller Auth. Pada langkah ini kita akan membuat controller auth, yang dimana di dalamnya terdapat proses authentikasi dari request login. Jalankan perintah make:controller

```
php artisan make:controller AuthController
```

   Buka file **App/Http/Controllers/AuthController.php** yang sudah kita buat. Lalu kita akan isi perintah code untuk membuat proses authentikasi , verifikasi, dan logout

6. Kemudian buat kode AuthController.php seperti dibawah ini

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades Auth;
use\Illuminate\Support\Facades Hash;
use\Illuminate\Support\Facades Validator;
   \
class AuthController extends Controller
{
    public function index()
    {
        // kita ambil data user lalu simpan pada variable $user
        $user = Auth::user();

        // kondisi jika user nya ada
        if ($user) {
            // jika user nya memiliki level admin
            if ($user->level_id == '1') {
                return redirect()->intended('admin');
            }
            // jika user nya memiliki level manager
            else if ($user->level == '2') {
                return redirect()->intended('manager');
            }
        }
        return view('login');
    }
    //
    public function proses_login(Request $request)
    {

        // kita buat validasi pada saat tombol login di klik
        // validas nya username & password wajib di isi
        $request->validate([
            'username' => 'required',
            'password' => 'required'
        ]);


        // ambil data request username & password saja
        $credential = $request->only('username', 'password');
        // cek jika data username dan password valid (sesuai) dengan data
        if (Auth::attempt($credential)) {

            // kalau berhasil simpan data user ya di variabel $user
            $user =  Auth::user();

            // cek lagi jika level user admin maka arahkan ke halaman admin
            if ($user->level_id == '1') {
                //dd($user->level_id);
                return redirect()->intended('admin');
            }
            // tapi jika level user nya user biasa maka arahkan ke halaman user
            else if ($user->level_id == '2') {
                return redirect()->intended('manager');
            }
            // jika belum ada role maka ke halaman /
            return redirect()->intended('/');
        }
        // jika ga ada data user yang valid maka kembalikan lagi ke halaman login
        // pastikan kirim pesan error juga kalau login gagal ya
        return redirect('login')
            ->withInput()
            ->withErrors(['login_gagal' => 'Pastikan kembali username dan password yang dimasukan sudah benar']);
    }

    public function register()
    {
        // tampilkan view register
        return view('register');
    }

    // aksi form register
    public function proses_register(Request $request)
    {
        //. kita buat validasi nih buat proses register
        // validasinya yaitu semua field wajib di isi
        // validasi username itu harus unique atau tidak boleh duplicate username ya
        $validator =  Validator::make($request->all(), [
            'nama' => 'required',
            'username' => 'required|unique:m_user',
            'password' => 'required'
        ]);

        // kalau gagal kembali ke halaman register dengan munculkan pesan error
        if ($validator->fails()) {
            return redirect('/register')
                ->withErrors($validator)
                ->withInput();
        }
        // kalau berhasil isi level & hash passwordnya ya biar secure
        $request['level_id'] = '2';
        $request['password'] = Hash::make($request->password);

        // masukkan semua data pada request ke table user
        UserModel::create($request->all());

        // kalo berhasil arahkan ke halaman login
        return redirect()->route('login');
    }

    public function logout(Request $request)
    {
        // logout itu harus menghapus session nya
        $request->session()->flush();

        // jalan kan juga fungsi logout pada auth
        Auth::logout();
        // kembali kan ke halaman login
        return Redirect('login');
    }
}
```

7. Kemudian tambahkan kode pada route pada file **routes/web.php** seperti kode program dibawah

```php
Route::get('login', [AuthController::class, 'index'])->name('login');
Route::get('register', [AuthController::class, 'register'])->name('register');
Route::post('proses_login', [AuthController::class, 'proses_login'])->name('proses_login');
Route::get('logout', [AuthController::class, 'logout'])->name('logout');
Route::post('proses_register', [AuthController::class, 'proses_register'])->name('proses_register');

// kita atur juga untuk middleware menggunakan group pada routing
// didalamnya terdapat group untuk mengecek kondisi login
// jika user yang login merupakan admin maka akan diarahkan ke AdminController
// jika user yang login merupakan manager maka akan diarahkan ke UserController
Route::group(['middleware' => ['auth']], function () {

    Route::group(['middleware' => ['cek_login:1']], function () {
        Route::resource('admin', AdminController::class);
    });
    Route::group(['middleware' => ['cek_login:2']], function () {
        Route::resource('manager', ManagerController::class);
    });
});
```

8. Kemudian buat file controller dengan nama AdminController sebagai halaman yang boleh diakses oleh admin saja. Jalankan perintah dibawah

```
php artisan make:controller AdminController
```

9. Buka folder **App/Http/Controllers** dan isi file AdminController seperti dibawah

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AdminController extends Controller
{
    public function index()
    {
        return view('admin');
    }
}
```

10. Kemudian buat file controller dengan nama ManagerController sebagai halaman yang boleh diakses oleh admin saja. Jalankan perintah dibawah

```
php artisan make:controller ManagerController
```

11. Buka folder **App/Http/Controllers** dan isi file **ManagerController** seperti dibawah

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ManagerController extends Controller
{
    public function index()
    {
        return view('manager');
    }
}
```

12. Pada langkah terakhir yaitu membuat layout sederhana
13. Buatlah halaman login dengan membuka folder resources/views dengan nama file l**ogin.blade.php.** Lalu kita isi filenya seperti contoh dibawah

```
@extends('adminlte::auth.auth-page', ['auth_type' => 'login'])

@section('adminlte_css_pre')
    <link rel="stylesheet" href="{{ asset('vendor/icheck-bootstrap/icheck-bootstrap.min.css') }}">
@stop

@php( $login_url = View::getSection('login_url') ?? config('adminlte.login_url', 'login') )
@php( $register_url = View::getSection('register_url') ?? config('adminlte.register_url', 'register') )
@php( $password_reset_url = View::getSection('password_reset_url') ?? config('adminlte.password_reset_url',
'password/reset') )

@if (config('adminlte.use_route_url', false))
    @php( $login_url = $login_url ? route($login_url) : '' )
    @php( $register_url = $register_url ? route($register_url) : '' )
    @php( $password_reset_url = $password_reset_url ? route($password_reset_url) : '' )
@else
    @php( $login_url = $login_url ? url($login_url) : '' )
    @php( $register_url = $register_url ? url($register_url) : '' )
    @php( $password_reset_url = $password_reset_url ? url($password_reset_url) : '' )
@endif

@section('auth_header', __('adminlte::adminlte.login_message'))

@section('auth_body')
    @error('login_gagal')
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
        <span class="alert-inner--text"><strong>Warning!</strong>  {{ $message }}</span>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    </div>
    @enderror
    <form action="{{url('proses_login')}}" method="post">
        @csrf

        {{-- Email field --}}
        <div class="input-group mb-3">
            <input type="text" name="username" class="form-control @error('username') is-invalid @enderror"
                   value="{{ old('username') }}" placeholder="Username" autofocus>

            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-envelope {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>

            @error('username')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>

        {{-- Password field --}}
        <div class="input-group mb-3">
            <input type="password" name="password" class="form-control @error('password') is-invalid @enderror"
                   placeholder="{{ __('adminlte::adminlte.password') }}">

            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-lock {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>

            @error('password')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>

        {{-- Login field --}}
        <div class="row">
            <div class="col-7">
                <div class="icheck-primary" title="{{ __('adminlte::adminlte.remember_me_hint') }}">
                    <input type="checkbox" name="remember" id="remember" {{ old('remember') ? 'checked' : '' }}>

                    <label for="remember">
                        {{ __('adminlte::adminlte.remember_me') }}
                    </label>
                </div>
            </div>

            <div class="col-5">
                <button type=submit class="btn btn-block {{ config('adminlte.classes_auth_btn', 'btn-flat btn-primary') }}
">

                    <span class="fas fa-sign-in-alt"></span>
                    {{ __('adminlte::adminlte.sign_in') }}
                </button>
            </div>
        </div>

    </form>
@stop

@section('auth_footer')
    @if($register_url)
        <p class="my-0">
            <a href="{{ route('register') }}">
                {{ __('adminlte::adminlte.register_a_new_membership') }}
            </a>
        </p>
    @endif
@stop
```

14. Buatlah halaman register dengan membuka folder resources/views dengan nama file **register.blade.php.** Lalu kita isi filenya seperti contoh dibawah

```
@extends('adminlte::auth.auth-page', ['auth_type' => 'register'])

@php( $login_url = View::getSection('login_url') ?? config('adminlte.login_url', 'login') )
@php( $register_url = View::getSection('register_url') ?? config('adminlte.register_url', 'register') )

@if (config('adminlte.use_route_url', false))
    @php( $login_url = $login_url ? route($login_url) : '' )
    @php( $register_url = $register_url ? route($register_url) : '' )
@else
    @php( $login_url = $login_url ? url($login_url) : '' )
    @php( $register_url = $register_url ? url($register_url) : '' )
@endif

@section('auth_header', __('adminlte::adminlte.register_message'))

@section('auth_body')
    <form action="{{route('proses_register')}}" method="post">
        @csrf

        {{-- Nama field --}}
        <div class="input-group mb-3">
            <input type="text" name="nama" class="form-control @error('nama') is-invalid @enderror"
                   value="{{ old('name') }}" placeholder="Nama" autofocus>

            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-user {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>

            @error('nama')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>

        {{-- Username field --}}
        <div class="input-group mb-3">
            <input type="text" name="username" class="form-control @error('username') is-invalid @enderror"
                   value="{{ old('name') }}" placeholder="Username" autofocus>

            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-user {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>

            @error('username')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>

        {{-- Password field --}}
        <div class="input-group mb-3">
            <input type="password" name="password" class="form-control @error('password') is-invalid @enderror"
                   placeholder="{{ __('adminlte::adminlte.password') }}">

            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-lock {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>

            @error('password')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>

        {{-- Register button --}}
        <button type="submit" class="btn btn-block {{ config('adminlte.classes_auth_btn', 'btn-flat btn-primary') }}">
            <span class="fas fa-user-plus"></span>
            {{ __('adminlte::adminlte.register') }}
        </button>

    </form>
@stop

@section('auth_footer')
    <p class="my-0">
        <a href="{{route('login')}}">
            {{ __('adminlte::adminlte.i_already_have_a_membership') }}
        </a>
    </p>
@stop
```

15. Buatlah halaman register dengan membuka folder resources/views dengan nama file **admin.blade.php.** Lalu kita isi filenya seperti contoh dibawah

```
@extends('layout.app')

{{-- Customize layout sections --}}

@section('subtitle', 'Admin')
@section('content_header_title', 'Home')
@section('content_header_subtitle', 'Admin')

@section('content')
<div class="container">
    <div class="card">
        <div class="card-header">Tampilan <?php echo (Auth::user()->level_id == 1)?'Admin':'Manager'; ?>
        <div class="card-body">
            <h1>Login Sebagai:
                <?php echo (Auth::user()->level_id == 1)?'Admin':'Manager'; ?></h1>
            <a href="{{ route('logout') }}">Logout</a>
        </div>
    </div>
</div>
@endsection

@push('js')
@endpush
```

16. Buatlah halaman register dengan membuka folder resources/views dengan nama file **manager.blade.php.** Lalu kita isi filenya seperti contoh dibawah

```
@extends('layout.app')

{{-- Customize layout sections --}}

@section('subtitle', 'Manager')
@section('content_header_title', 'Home')
@section('content_header_subtitle', 'Manager')

@section('content')
<div class="container">
    <div class="card">
        <div class="card-header">Tampilan <?php echo (Auth::user()->level_id == 1)?'Admin':'Manager'; ?>
        <div class="card-body">
            <h1>Login Sebagai:
                <?php echo (Auth::user()->level_id == 1)?'Admin':'Manager'; ?></h1>
            <a href="{{ route('logout') }}">Logout</a>
        </div>
    </div>
</div>
@endsection

@push('js')
@endpush
```

17. Jalankan kode diatas dan buat laporan dengan isi screenshoot hasil dan beri penjelasan pada setiap screenshoot hasil dan *commit* perubahan pada *git*.

**** Sekian, dan selamat belajar ****