MODUL PRAKTIKUM PEMROGRAMAN WEB

Laravel 13

Breeze + Blade + MySQL

Implementasi Authentication, Middleware Admin/User, Dashboard, dan CRUD User

PROGRAM KEAHLIAN

PPLG / RPL

KELAS

XI

MODEL PEMBELAJARAN

Praktikum Berbasis Proyek

Modul Praktikum Pemrograman Web — Kelas XI RPL/PPLG

01 Identitas Modul

Mata Pelajaran	Pemrograman Web
Kelas	XI RPL / PPLG
Materi	Laravel 13, Breeze, Blade, MySQL, Authentication, Middleware, dan CRUD
Model Pembelajaran	Project Based Learning dan Praktikum
Alokasi Waktu	10–12 Pertemuan

02 Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, peserta didik diharapkan mampu memahami konsep dasar Laravel dan menerapkannya untuk membangun aplikasi web dengan sistem autentikasi dan hak akses.

Kompetensi yang Diharapkan

- Menjelaskan konsep framework Laravel.
- Membuat project Laravel 13.
- Menghubungkan Laravel dengan MySQL.
- Menginstal Laravel Breeze.
- Menggunakan Blade sebagai template engine.
- Membuat authentication.
- Membuat role Admin dan User.
- Membuat middleware.
- Membatasi halaman berdasarkan role.
- Membuat dashboard Admin dan User.
- Membuat menu Manajemen User.
- Membuat CRUD User.
- Menerapkan validasi form.
- Menerapkan keamanan dasar aplikasi.

03

Daftar Isi

Materi Modul

04 Konsep Aplikasi

Pada praktikum ini kita akan membangun aplikasi sederhana bernama **RPL User Management** .

Aplikasi memiliki dua jenis pengguna, yaitu Admin dan User. Keduanya memiliki halaman dan hak akses yang berbeda.

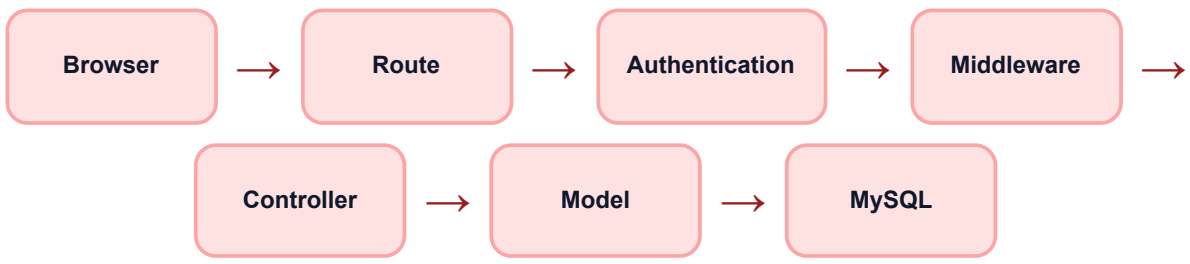
Administrator

- Login
- Membuka Dashboard Admin
- Melihat data user
- Menambah user
- Mengedit user
- Menghapus user

User

- Login
- Membuka Dashboard User
- Melihat halaman user
- Tidak dapat membuka halaman admin
- Tidak dapat mengakses CRUD User

Arsitektur Aplikasi



Authentication digunakan untuk menentukan apakah pengguna telah login.

Authorization digunakan untuk menentukan apa saja yang boleh dilakukan oleh pengguna.

Middleware dapat digunakan untuk menyaring request sebelum masuk ke controller.

05 Persiapan Perangkat

Software yang Dibutuhkan

Software	Fungsi
PHP	Menjalankan Laravel
Composer	Package manager PHP
Node.js dan NPM	Mengelola asset frontend
MySQL	Database aplikasi
VS Code	Editor program
Browser	Mengakses aplikasi

Mengecek Instalasi

TERMINAL

```
php -v

composer -V

node -v

npm -v

mysql --version
```

Pastikan seluruh perintah dapat dijalankan tanpa error sebelum melanjutkan praktikum.

06 Membuat Project Laravel 13

1 Membuat Project

```
TERMINAL
composer create-project laravel/laravel:^13.0 rpl-user-management
```

2 Masuk ke Folder Project

```
TERMINAL
cd rpl-user-management
```

3 Menjalankan Server

```
TERMINAL
php artisan serve
```

Jika server Laravel berhasil berjalan, aplikasi dapat dibuka melalui alamat server lokal yang ditampilkan terminal.

07 Konfigurasi MySQL

Membuat Database

Buat database MySQL dengan nama:

Nama Database

```
rpl_laravel13
```

Konfigurasi File .env

.env

```
APP_NAME="RPL User Management"

APP_ENV=local
APP_KEY=
APP_DEBUG=true

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=rpl_laravel13
DB_USERNAME=root
DB_PASSWORD=
```

Menjalankan Migration

TERMINAL

```
php artisan migrate
```

> Jika tidak terdapat error, Laravel telah berhasil terhubung dengan MySQL.

08 Instalasi Breeze + Blade

Laravel Breeze menyediakan implementasi authentication sederhana yang dapat digunakan sebagai dasar aplikasi.

Install Laravel Breeze

```
TERMINAL
composer require laravel/breeze --dev
```

Install Breeze dengan Blade

```
TERMINAL
php artisan breeze:install blade
```

Install Dependency NPM

```
TERMINAL
npm install
```

Build Asset

```
TERMINAL
npm run build
```

Jalankan Migration

```
TERMINAL
php artisan migrate
```

Login

Pengguna dapat masuk menggunakan email dan password.

Register

Pengguna dapat membuat akun baru.

Profile

Pengguna dapat mengelola profil.

09 Database, Migration dan Model User

Tabel Users

Laravel Breeze telah menyediakan migration untuk tabel **users**.

Struktur Tabel Users

```
users
|
├── id
├── name
├── email
├── email_verified_at
├── password
├── remember_token
├── created_at
└── updated_at
```

Menambahkan Kolom Role

ARTISAN

```
php artisan make:migration add_role_to_users_table --table=users
```

Migration Role

database/migrations/xxxx_add_role_to_users_table.php

```php
<?php

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

return new class extends Migration
{
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {

            $table->string('role')
                ->default('user')
                ->after('password');

        });
    }

    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {

            $table->dropColumn('role');

        });
    }
};
```

Jalankan Migration

TERMINAL

```
php artisan migrate
```

Model User

Edit file:

app/Models/User.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
        'role',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```

Perhatikan $fillable. Kolom **role** harus dimasukkan agar dapat diisi melalui mass assignment.

10 Membuat Role Admin dan User

Kita menggunakan dua role sederhana:

Role	Hak Akses
admin	Dashboard Admin dan CRUD User
user	Dashboard User

Membuat Seeder Admin

TERMINAL

```
php artisan make:seeder AdminUserSeeder
```

Isi Seeder

database/seeders/AdminUserSeeder.php

```
<?php

namespace Database\Seeders;

use App\Models\User;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;

class AdminUserSeeder extends Seeder
{
    public function run(): void
    {
        User::updateOrCreate(
            [
                'email' => 'admin@rpl.test',
            ],
            [
                'name' => 'Administrator',
                'password' => Hash::make('password'),
                'role' => 'admin',
            ]
        );
    }
}
```

Jalankan Seeder

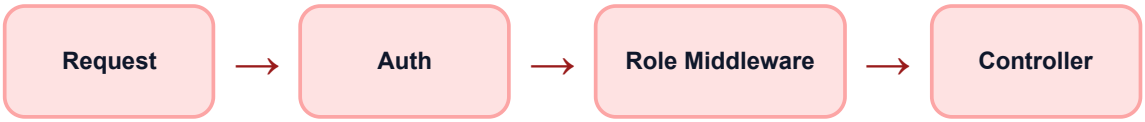
```
TERMINAL
php artisan db:seed --class=AdminUserSeeder
```

Data	Nilai
Email	admin@rpl.test
Password	password
Role	admin

> Password di atas hanya untuk keperluan praktikum. Pada aplikasi sebenarnya gunakan password yang kuat dan jangan menyimpan password dalam bentuk teks biasa.

11 Membuat Middleware Role

Middleware adalah lapisan yang berada di antara request dan aplikasi. Middleware dapat digunakan untuk memeriksa apakah pengguna memiliki hak akses tertentu.



Membuat Middleware

```
TERMINAL
php artisan make:middleware RoleMiddleware
```

Isi RoleMiddleware

app/Http/Middleware/RoleMiddleware.php

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class RoleMiddleware
{
    public function handle(
        Request $request,
        Closure $next,
        string $role
    ): Response {

        if (
            ! $request->user() ||
            $request->user()->role !== $role
        ) {
            abort(
                403,
                'Anda tidak memiliki akses.'
            );
        }

        return $next($request);
    }
}
```

Mendaftarkan Alias Middleware

Pada Laravel versi modern, alias middleware dapat didaftarkan pada file:

bootstrap/app.php

```
use App\Http\Middleware\RoleMiddleware;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(
    basePath: dirname(__DIR__)
)
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )

    ->withMiddleware(function (
        Middleware $middleware
    ): void {

        $middleware->alias([
            'role' => RoleMiddleware::class,
        ]);

    })

    ->withExceptions(function (
        Exceptions $exceptions
    ): void {
        //
    })
    ->create();
```

Setelah alias didaftarkan, middleware dapat digunakan sebagai:

role:admin

atau:

role:user

12 Dashboard Admin dan User

Membuat Controller

TERMINAL

```
php artisan make:controller DashboardController
```

DashboardController

app/Http/Controllers/DashboardController.php

```
<?php

namespace App\Http\Controllers;

class DashboardController extends Controller
{
    public function admin()
    {
        return view(
            'admin.dashboard'
        );
    }


    public function user()
    {
        return view(
            'user.dashboard'
        );
    }
}
```

Route Dashboard

routes/web.php

```php
<?php

use App\Http\Controllers\DashboardController;
use Illuminate\Support\Facades\Route;


Route::middleware('auth')->group(function () {

    Route::get(
        '/admin/dashboard',
        [DashboardController::class, 'admin']
    )
    ->middleware('role:admin')
    ->name('admin.dashboard');


    Route::get(
        '/user/dashboard',
        [DashboardController::class, 'user']
    )
    ->middleware('role:user')
    ->name('user.dashboard');

});
```

Dashboard Admin

resources/views/admin/dashboard.blade.php

```
<x-app-layout>

    <x-slot name="header">

        <h2 class="font-semibold text-xl">

            Dashboard Admin

        </h2>

    </x-slot>


    <div class="p-6">

        <h3>

            Selamat datang,
            {{ auth()->user()->name }}

        </h3>


        <p>

            Anda login sebagai Administrator.

        </p>

    </div>

</x-app-layout>
```

Dashboard User

resources/views/user/dashboard.blade.php

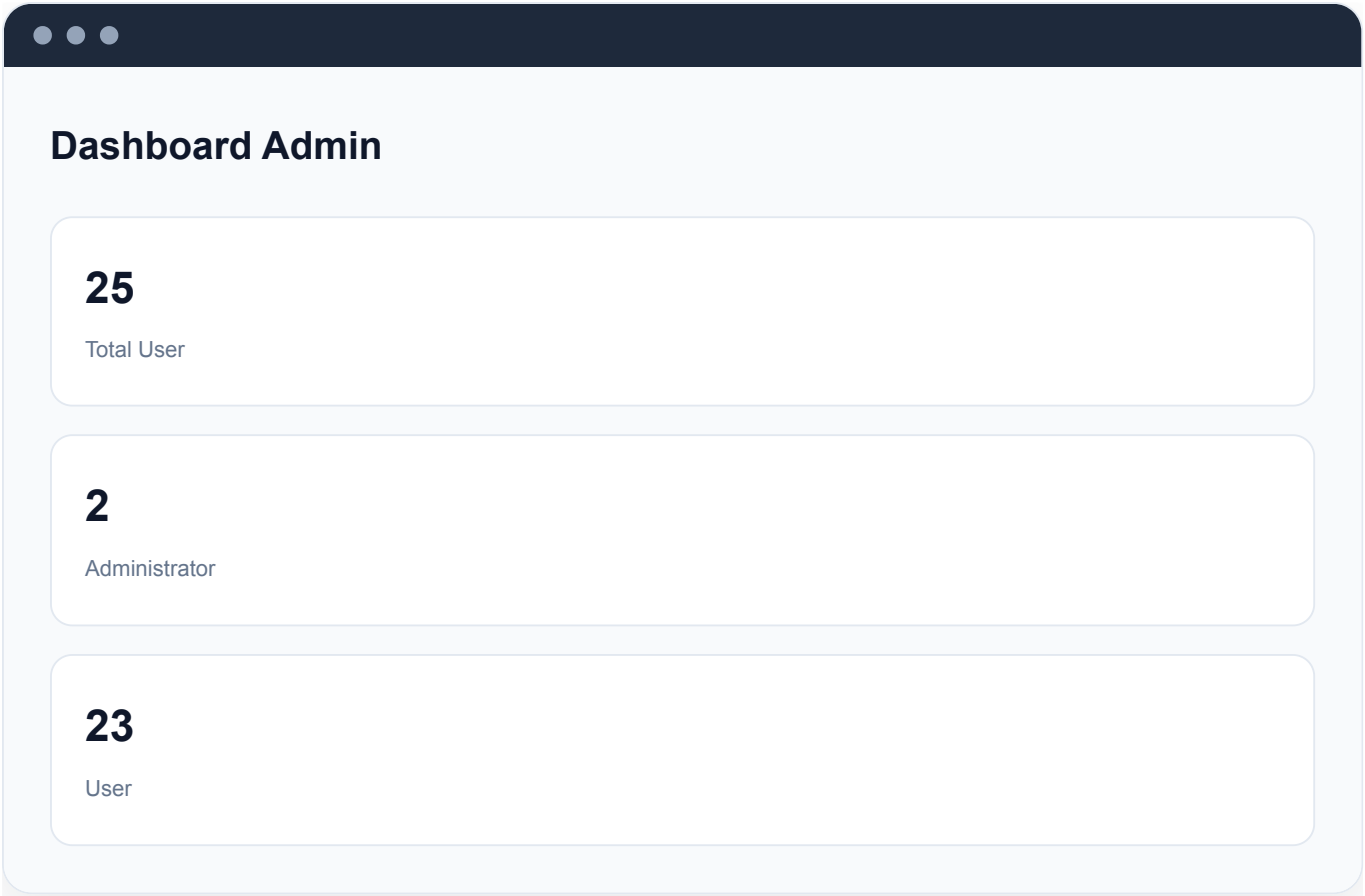
```
<x-app-layout>

    <x-slot name="header">

        <h2 class="font-semibold text-xl">

            Dashboard User

        </h2>

    </x-slot>


    <div class="p-6">

        <h3>

            Selamat datang,
            {{ auth()->user()->name }}

        </h3>


        <p>

            Anda login sebagai User.

        </p>

    </div>

</x-app-layout>
```

Redirect Dashboard Berdasarkan Role

Agar pengguna tidak perlu menentukan URL dashboard, kita dapat membuat route dashboard utama.

routes/web.php

```php
Route::get('/dashboard', function () {

    return match (
        auth()->user()->role
    ) {

        'admin' =>
            redirect()->route(
                'admin.dashboard'
            ),

        'user' =>
            redirect()->route(
                'user.dashboard'
            ),

        default =>
            abort(403),

    };

})
->middleware('auth')
->name('dashboard');
```



13 Membuat Menu Manajemen User

Menu Manajemen User hanya boleh ditampilkan kepada pengguna dengan role Admin.

Kondisi Blade

resources/views/layouts/navigation.blade.php

```
@if(auth()->user()->role === 'admin')

    <a
        href="{{ route(
            'admin.users.index'
        ) }}"
    >

        Manajemen User

    </a>

@endif
```

Ingat! Menyembunyikan menu bukanlah sistem keamanan.

Route tetap harus menggunakan middleware **role:admin**.

14 Membuat CRUD User

CRUD merupakan singkatan dari:

Operasi	Arti	Fungsi
Create	Membuat	Menambahkan user
Read	Membaca	Menampilkan user
Update	Memperbarui	Mengubah user
Delete	Menghapus	Menghapus user

Membuat Resource Controller

TERMINAL

```
php artisan make:controller UserController --resource
```

UserController

app/Http/Controllers/UserController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;

class UserController extends Controller
{
    public function index()
    {
        $users = User::latest()
            ->paginate(10);

        return view(
            'admin.users.index',
            compact('users')
        );
    }


    public function create()
    {
        return view(
            'admin.users.create'
        );
    }


    public function store(Request $request)
    {
        $validated = $request->validate([

            'name' => [
                'required',
                'string',
                'max:255',
            ],

            'email' => [
                'required',
                'email',
                'max:255',
```

```
                'unique:users,email',
            ],

            'role' => [
                'required',
                Rule::in([
                    'admin',
                    'user',
                ]),
            ],

            'password' => [
                'required',
                'string',
                'min:8',
                'confirmed',
            ],

        ]);


        $validated['password'] =
            Hash::make(
                $validated['password']
            );


        User::create($validated);


        return redirect()
            ->route(
                'admin.users.index'
            )
            ->with(
                'success',
                'User berhasil ditambahkan.'
            );
    }


    public function show(User $user)
    {
        return view(
            'admin.users.show',
            compact('user')
        );
    }
```

```
public function edit(User $user)
{
    return view(
        'admin.users.edit',
        compact('user')
    );
}


public function update(
    Request $request,
    User $user
) {

    $validated = $request->validate([

        'name' => [
            'required',
            'string',
            'max:255',
        ],

        'email' => [
            'required',
            'email',
            'max:255',

            Rule::unique(
                'users',
                'email'
            )->ignore($user->id),
        ],

        'role' => [
            'required',
            Rule::in([
                'admin',
                'user',
            ]),
        ],

        'password' => [
            'nullable',
            'string',
            'min:8',
            'confirmed',
```

```
            ],

        ]);


        if (
            !empty(
                $validated['password']
            )
        ) {

            $validated['password'] =
                Hash::make(
                    $validated['password']
                );

        } else {

            unset(
                $validated['password']
            );

        }


        $user->update(
            $validated
        );


        return redirect()
            ->route(
                'admin.users.index'
            )
            ->with(
                'success',
                'User berhasil diperbarui.'
            );
    }


    public function destroy(User $user)
    {
        if (
            $user->id ===
            auth()->id()
        ) {
```

```
            return back()
                ->with(
                    'error',
                    'Tidak dapat menghapus akun sendiri.'
                );
        }


        $user->delete();


        return back()
            ->with(
                'success',
                'User berhasil dihapus.'
            );
    }
}
```

15 Route CRUD User

Seluruh route CRUD user harus dilindungi dengan authentication dan role admin.

routes/web.php

```php
use App\Http\Controllers\UserController;


Route::middleware([
    'auth',
    'role:admin',
])
->prefix('admin')
->name('admin.')
->group(function () {

    Route::resource(
        'users',
        UserController::class
    )
    ->except([
        'show',
    ]);

});
```

Memeriksa Route

TERMINAL

```
php artisan route:list
```

Method	URL	Route Name	Fungsi
GET	/admin/users	admin.users.index	Menampilkan data
GET	/admin/users/create	admin.users.create	Form tambah
POST	/admin/users	admin.users.store	Simpan data
GET	/admin/users/{user}/edit	admin.users.edit	Form edit
PUT/PATCH	/admin/users/{user}	admin.users.update	Update data
DELETE	/admin/users/{user}	admin.users.destroy	Hapus data

16 Membuat View Blade

Buat folder:

```
resources/views/admin/users/
```

Kemudian buat tiga file:

- index.blade.php
- create.blade.php
- edit.blade.php

Index User

resources/views/admin/users/index.blade.php

```
<x-app-layout>

    <x-slot name="header">

        <h2>
            Manajemen User
        </h2>

    </x-slot>


    <div class="p-6">


        @if(session('success'))

            <div>

                {{ session('success') }}

            </div>

        @endif


        <div>

            <a
                href="{{ route(
                    'admin.users.create'
                ) }}"
            >

                + Tambah User

            </a>

        </div>


        <table>

            <thead>

                <tr>
```

```
                <th>
                    Nama
                </th>

                <th>
                    Email
                </th>

                <th>
                    Role
                </th>

                <th>
                    Aksi
                </th>

            </tr>

        </thead>


        <tbody>

            @forelse(
                $users as $user
            )

                <tr>

                    <td>

                        {{ $user->name }}

                    </td>


                    <td>

                        {{ $user->email }}

                    </td>


                    <td>

                        {{ $user->role }}
```

```
                    </td>


                    <td>

                        <a
                            href="{{ route(
                                'admin.users.edit',
                                $user
                            ) }}"
                            >

                            Edit

                        </a>


                        <form
                            action="{{ route(
                                'admin.users.destroy',
                                $user
                            ) }}"
                            method="POST"
                            >

                            @csrf

                            @method('DELETE')


                            <button
                                type="submit"
                                onclick="
                                    return confirm(
                                        'Hapus user?'
                                    )
                                "
                            >

                                Hapus

                            </button>

                        </form>

                    </td>

                </tr>
```

```
                @empty
                    <tr>
                        <td colspan="4">
                            Belum ada data.
                        </td>
                    </tr>
                @endforelse
            </tbody>
        </table>

        {{ $users->links() }}
    </div>
</x-app-layout>
```

Form Tambah User

resources/views/admin/users/create.blade.php

```
<x-app-layout>

    <x-slot name="header">

        <h2>
            Tambah User
        </h2>

    </x-slot>


    <div class="p-6">

        <form
            action="{{ route(
                'admin.users.store'
            ) }}"
            method="POST"
        >

            @csrf


            <div>

                <label>
                    Nama
                </label>

                <input
                    type="text"
                    name="name"
                    value="{{ old('name') }}"
                    required
                >

            </div>


            <div>

                <label>
                    Email
                </label>
```

```
            <input
                type="email"
                name="email"
                value="{{ old('email') }}"
                required
            >

        </div>


        <div>

            <label>
                Role
            </label>

            <select name="role">

                <option value="user">
                    User
                </option>

                <option value="admin">
                    Admin
                </option>

            </select>

        </div>


        <div>

            <label>
                Password
            </label>

            <input
                type="password"
                name="password"
                required
            >

        </div>


        <div>
```

```
                <label>
                    Konfirmasi Password
                </label>

                <input
                    type="password"
                    name="password_confirmation"
                    required
                >

            </div>


            <button type="submit">

                Simpan

            </button>

        </form>

    </div>

</x-app-layout>
```

Form Edit User

resources/views/admin/users/edit.blade.php

```
<x-app-layout>

    <x-slot name="header">

        <h2>
            Edit User
        </h2>

    </x-slot>


    <div class="p-6">

        <form
            action="{{ route(
                'admin.users.update',
                $user
            ) }}"
            method="POST"
        >

            @csrf

            @method('PUT')


            <div>

                <label>
                    Nama
                </label>

                <input
                    type="text"
                    name="name"
                    value="{{ old(
                        'name',
                        $user->name
                    ) }}"
                    required
                >

            </div>
```

```
            <div>

                <label>
                    Email
                </label>

                <input
                    type="email"
                    name="email"
                    value="{{ old(
                        'email',
                        $user->email
                    ) }}"
                    required
                >

            </div>


            <div>

                <label>
                    Role
                </label>

                <select name="role">

                    <option
                        value="user"
                        @selected(
                            $user->role === 'user'
                        )
                    >

                        User

                    </option>


                    <option
                        value="admin"
                        @selected(
                            $user->role === 'admin'
                        )
                    >

                        Admin
```

```
                </option>

            </select>

        </div>


        <div>

            <label>
                Password Baru
            </label>

            <input
                type="password"
                name="password"
            >

            <small>

                Kosongkan jika password
                tidak ingin diubah.

            </small>

        </div>


        <div>

            <label>
                Konfirmasi Password
            </label>

            <input
                type="password"
                name="password_confirmation"
            >

        </div>


        <button type="submit">

            Update

        </button>
```

```
        </form>

    </div>

</x-app-layout>
```

17 Validasi dan Keamanan

1. CSRF Protection

Form Laravel harus menggunakan token CSRF.

```
@csrf
```

2. Hash Password

```
Hash::make(
    $password
);
```

3. Validasi Email

```
'email' => [
    'required',
    'email',
    'unique:users,email',
],
```

4. Validasi Role

```
'role' => [
    'required',
    Rule::in([
        'admin',
        'user',
    ]),
],
```

5. Middleware

```
Route::middleware([
    'auth',
    'role:admin',
])->group(function () {

    // Route admin

});
```

> Jangan hanya mengandalkan tampilan menu untuk membatasi akses.
>
> Keamanan utama harus berada pada route dan middleware.

18 Pengujian Aplikasi

Lakukan pengujian menggunakan skenario berikut.

No	Skenario	Hasil yang Diharapkan	Status
1	Membuka dashboard tanpa login	Dialihkan ke halaman login	☐
2	User membuka /admin/dashboard	Akses ditolak	☐
3	Admin membuka dashboard admin	Berhasil	☐
4	Admin menambah user	Data tersimpan	☐
5	Email duplikat	Validasi gagal	☐
6	Admin mengedit user	Data berubah	☐
7	Admin menghapus user	Data terhapus	☐
8	Admin menghapus akun sendiri	Ditolak	☐

19 Struktur Folder Project

Struktur project setelah praktikum kurang lebih seperti berikut:

Struktur Project

```
rpl-user-management/
|
├── app/
|   |
|   ├── Http/
|   |   |
|   |   ├── Controllers/
|   |   |   ├── Controller.php
|   |   |   ├── DashboardController.php
|   |   |   └── UserController.php
|   |   |
|   |   └── Middleware/
|   |       └── RoleMiddleware.php
|   |
|   └── Models/
|       └── User.php
|
├── bootstrap/
|   └── app.php
|
├── database/
|   |
|   ├── migrations/
|   |   ├── xxxx_create_users_table.php
|   |   └── xxxx_add_role_to_users_table.php
|   |
|   └── seeders/
|       ├── DatabaseSeeder.php
|       └── AdminUserSeeder.php
|
├── resources/
|   |
|   └── views/
|       |
|       ├── admin/
|       |   ├── dashboard.blade.php
|       |   |
|       |   └── users/
|       |       ├── index.blade.php
|       |       ├── create.blade.php
|       |       └── edit.blade.php
|       |
|       └── user/
|           └── dashboard.blade.php
|
```

```
├── routes/
│   └── web.php
│
├── .env
│
├── composer.json
│
└── package.json
```

20 Troubleshooting

Error: Target class [role] does not exist

Periksa apakah alias middleware sudah didaftarkan pada:

```
bootstrap/app.php
```

Error: Unknown column 'role'

Jalankan migration:

```
TERMINAL
php artisan migrate
```

Error Vite Manifest

```
TERMINAL
npm install

npm run build
```

Database Connection Error

Periksa file **.env**. Kemudian bersihkan cache konfigurasi.

```
TERMINAL
php artisan config:clear
```

Route Tidak Ditemukan

```
TERMINAL
php artisan route:list
```

Membersihkan Cache Laravel

```
TERMINAL
php artisan optimize:clear
```

21 Tugas Praktikum

Tugas Utama

Buat aplikasi manajemen user menggunakan Laravel 13, Breeze, Blade dan MySQL berdasarkan langkah-langkah dalam modul.

Ketentuan

1. Aplikasi harus menggunakan Laravel 13.
2. Database menggunakan MySQL.
3. Authentication menggunakan Breeze.
4. Tampilan menggunakan Blade.
5. Terdapat role Admin dan User.
6. Admin dapat mengelola user.
7. User tidak dapat membuka halaman admin.
8. Route admin menggunakan middleware.
9. Password harus di-hash.

10. Form harus mempunyai validasi.

Pengembangan

1. Tambahkan pencarian user.
2. Tambahkan filter berdasarkan role.
3. Tambahkan jumlah Admin dan User pada dashboard.
4. Tambahkan pagination.
5. Tambahkan konfirmasi sebelum menghapus.
6. Buat tampilan dashboard yang lebih menarik.
7. Tambahkan halaman profil.

Checklist Praktikum

- ☐ Project Laravel 13 berhasil dibuat.
- ☐ MySQL berhasil terhubung.
- ☐ Migration berhasil.
- ☐ Breeze berhasil dipasang.
- ☐ Login berhasil.
- ☐ Register berhasil.
- ☐ Role Admin berhasil.
- ☐ Role User berhasil.
- ☐ Middleware berhasil.
- ☐ Dashboard Admin berhasil.
- ☐ Dashboard User berhasil.

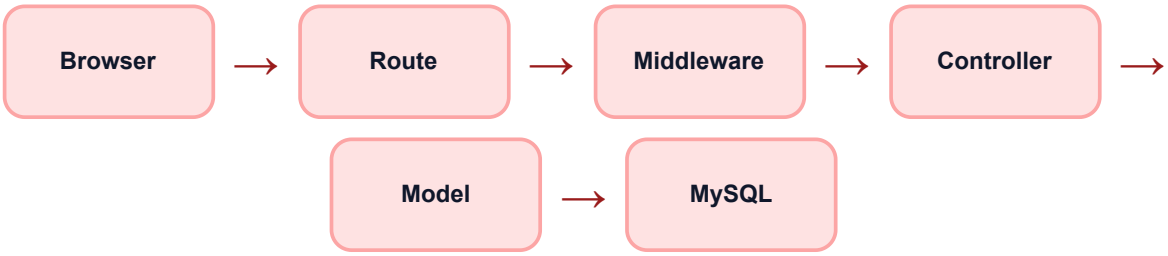
- ☐ Menu Manajemen User berhasil.
- ☐ Create berhasil.
- ☐ Read berhasil.
- ☐ Update berhasil.
- ☐ Delete berhasil.
- ☐ Validasi berhasil.
- ☐ Pengujian berhasil.

22 Rubrik Penilaian

Komponen	Bobot	Indikator
Instalasi	10%	Project Laravel berhasil dibuat.
Database	15%	MySQL dan migration berhasil.
Authentication	15%	Breeze dan login berjalan.
Middleware	20%	Hak akses Admin/User berjalan.
CRUD	25%	Create, Read, Update dan Delete berhasil.
Tampilan	10%	Tampilan rapi dan mudah digunakan.
Pengujian	5%	Siswa melakukan pengujian aplikasi.

Total Nilai = 100%

23 Ringkasan Materi



Laravel

Framework PHP yang digunakan untuk membangun aplikasi web.

Breeze

Starter kit sederhana untuk authentication Laravel.

Blade

Template engine bawaan Laravel untuk membuat tampilan.

Middleware

Lapisan yang digunakan untuk menyaring request dan membatasi akses.

Inti pembelajaran:

Authentication menentukan **siapa pengguna**.

Role menentukan **jenis pengguna**.

Middleware menentukan **siapa yang boleh mengakses route tertentu**.

Controller menangani **logika aplikasi**.

Model menangani **interaksi dengan database**.

Blade menangani **tampilan aplikasi**.

Selamat Belajar!

Jangan hanya menyalin kode. Pahami bagaimana setiap bagian aplikasi saling terhubung.

Mulailah dari Route, pahami Middleware, lanjutkan ke Controller, Model, Database, kemudian tampilkan hasilnya menggunakan Blade.

Selamat berkarya, siswa RPL/PPLG!
