|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| Name | Varchar (25) |  |
| Username | Varchar (25) |  |
| Email | Varchar (25) |  |
| Password | Varchar (Max) |  |

Table users

|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_doktrin | Varchar (25) |  |
| Nama\_doktrin | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |

Table doktrin operasi

Table doktrin fungsi umum

|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_doktrin | Varchar (25) |  |
| Nama\_doktrin | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |

Table doktin fungsi khusus

|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_doktrin | Varchar (25) |  |
| Nama\_doktrin | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |

Table Jukgar

|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_petunjuk | Varchar (25) |  |
| Nama\_petunjuk | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |

Table Juknis

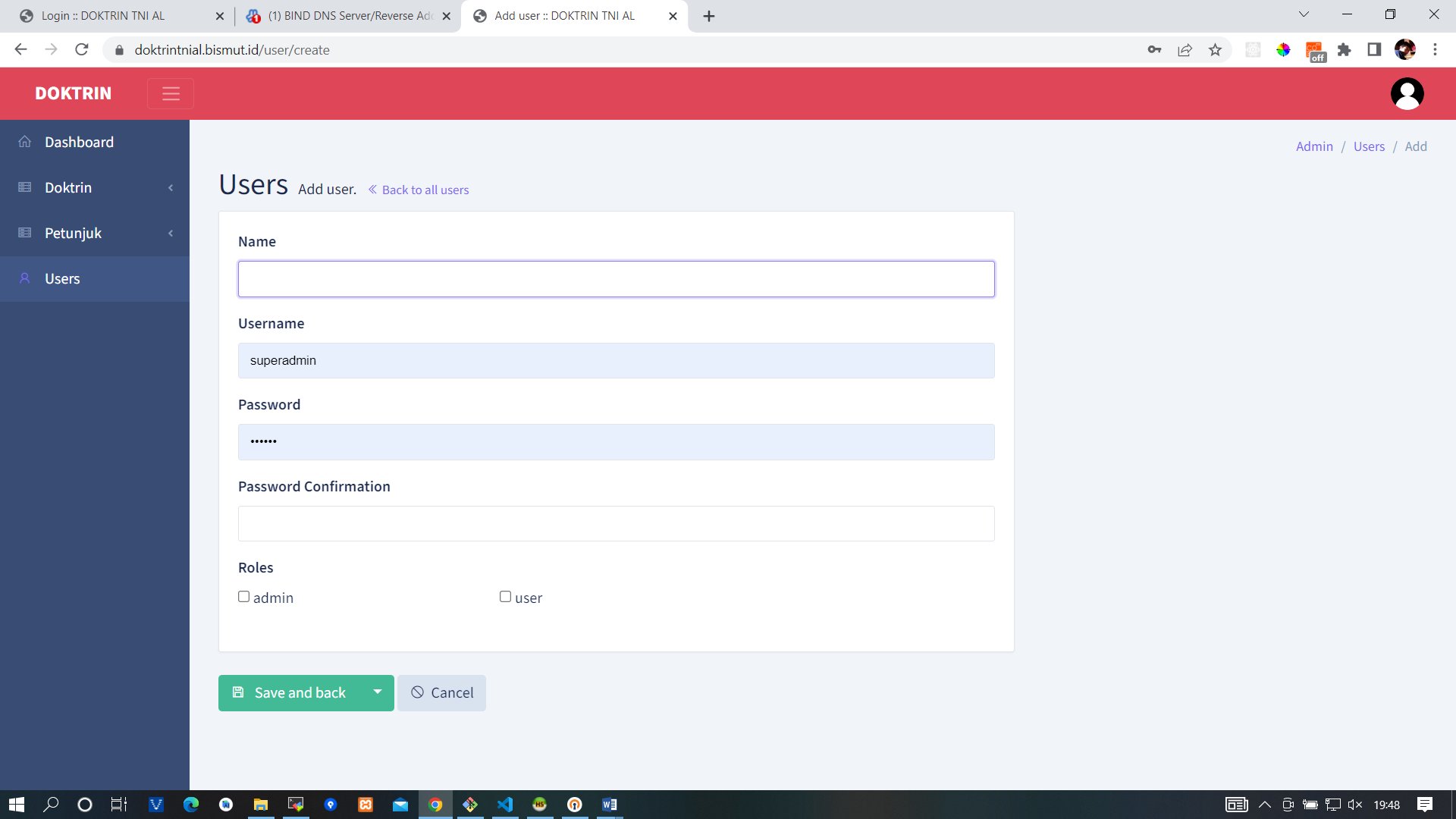
|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_petunjuk | Varchar (25) |  |
| Nama\_petunjuk | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |

Table Jukref

|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_petunjuk | Varchar (25) |  |
| Nama\_petunjuk | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |

Table Protap

|  |  |  |
| --- | --- | --- |
| Nama field | Tipe data | Keterangan |
| Id | Int | Primary key |
| No\_petunjuk | Varchar (25) |  |
| Nama\_petunjuk | Varchar (25) |  |
| Tanggal | Date |  |
| File | Varchar (Max) |  |
| Pembina | Varchar (25) |  |



**database\migrations\2014\_10\_12\_000000\_create\_users\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('users', function (Blueprint $table) {

            $table->id();

            $table->string('name');

            $table->string('username')->unique();

            $table->string('password');

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('users');

    }

}

**app\Models\User.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

use Spatie\Permission\Traits\HasRoles;

use Backpack\CRUD\app\Models\Traits\CrudTrait;

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable, HasRoles, CrudTrait;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'username',

        'password',

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

    ];

}

**app\Http\Controllers\Admin\UserCrudController.php**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Requests\UserRequest;

use Backpack\CRUD\app\Http\Controllers\CrudController;

use Backpack\CRUD\app\Library\CrudPanel\CrudPanelFacade as CRUD;

use Illuminate\Support\Facades\Hash;

/\*\*

 \* Class UserCrudController

 \* @package App\Http\Controllers\Admin

 \* @property-read \Backpack\CRUD\app\Library\CrudPanel\CrudPanel $crud

 \*/

class UserCrudController extends CrudController

{

    use \Backpack\CRUD\app\Http\Controllers\Operations\ListOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\CreateOperation {store as traitStore;}

    use \Backpack\CRUD\app\Http\Controllers\Operations\UpdateOperation {update as traitUpdate;}

    use \Backpack\CRUD\app\Http\Controllers\Operations\DeleteOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\ShowOperation;

    /\*\*

     \* Configure the CrudPanel object. Apply settings to all operations.

     \*

     \* @return void

     \*/

    public function setup()

    {

        CRUD::setModel(\App\Models\User::class);

        CRUD::setRoute(config('backpack.base.route\_prefix') . '/user');

        CRUD::setEntityNameStrings('user', 'users');

    }

    /\*\*

     \* Define what happens when the List operation is loaded.

     \*

     \* @see  https://backpackforlaravel.com/docs/crud-operation-list-entries

     \* @return void

     \*/

    protected function setupListOperation()

    {

        $this->crud->denyAccess('show');

        CRUD::column('name');

        CRUD::column('username');

        /\*\*

         \* Columns can be defined using the fluent syntax or array syntax:

         \* - CRUD::column('price')->type('number');

         \* - CRUD::addColumn(['name' => 'price', 'type' => 'number']);

         \*/

    }

    /\*\*

     \* Define what happens when the Create operation is loaded.

     \*

     \* @see https://backpackforlaravel.com/docs/crud-operation-create

     \* @return void

     \*/

    protected function setupCreateOperation()

    {

        CRUD::setValidation(UserRequest::class);

        $this->crud->addFields([

            [

                'name' => 'name',

                'label' => trans('backpack::permissionmanager.name'),

                'type' => 'text',

            ],

            [

                'name' => 'username',

                'label' => 'Username',

                'type' => 'text',

            ],

            [

                'name' => 'password',

                'label' => trans('backpack::permissionmanager.password'),

                'type' => 'password',

            ],

            [

                'name' => 'password\_confirmation',

                'label' => trans('backpack::permissionmanager.password\_confirmation'),

                'type' => 'password',

            ],

            [

                'label' => 'Roles',

                'type' => 'checklist',

                'name' => 'roles',

                'entity' => 'roles',

                'attribute' => 'name',

                'model' => "Backpack\PermissionManager\app\Models\Role",

                'pivot' => true,

                'options' => (function ($query) {

                    return $query->select('name','id')->where('name', '<>', 'superadmin')->pluck('name','id');

                }),

            ],

        ]);

        /\*\*

         \* Fields can be defined using the fluent syntax or array syntax:

         \* - CRUD::field('price')->type('number');

         \* - CRUD::addField(['name' => 'price', 'type' => 'number']));

         \*/

    }

    /\*\*

     \* Define what happens when the Update operation is loaded.

     \*

     \* @see https://backpackforlaravel.com/docs/crud-operation-update

     \* @return void

     \*/

    protected function setupUpdateOperation()

    {

        $this->setupCreateOperation();

    }

     /\*\*

     \* Store a newly created resource in the database.

     \*

     \* @return \Illuminate\Http\RedirectResponse

     \*/

    function store()

    {

        $this->crud->setRequest($this->crud->validateRequest());

        $this->crud->setRequest($this->handlePasswordInput($this->crud->getRequest()));

        $this->crud->unsetValidation(); // validation has already been run

        return $this->traitStore();

    }

    /\*\*

     \* Update the specified resource in the database.

     \*

     \* @return \Illuminate\Http\RedirectResponse

     \*/

    function update()

    {

        $this->crud->setRequest($this->crud->validateRequest());

        $this->crud->setRequest($this->handlePasswordInput($this->crud->getRequest()));

        $this->crud->unsetValidation(); // validation has already been run

        return $this->traitUpdate();

    }

    /\*\*

     \* Handle password input fields.

     \*/

    function handlePasswordInput($request)

    {

        // Remove fields not present on the user.

        $request->request->remove('password\_confirmation');

        $request->request->remove('roles\_show');

        $request->request->remove('permissions\_show');

        // Encrypt password if specified.

        if ($request->input('password')) {

            $request->request->set('password', Hash::make($request->input('password')));

        } else {

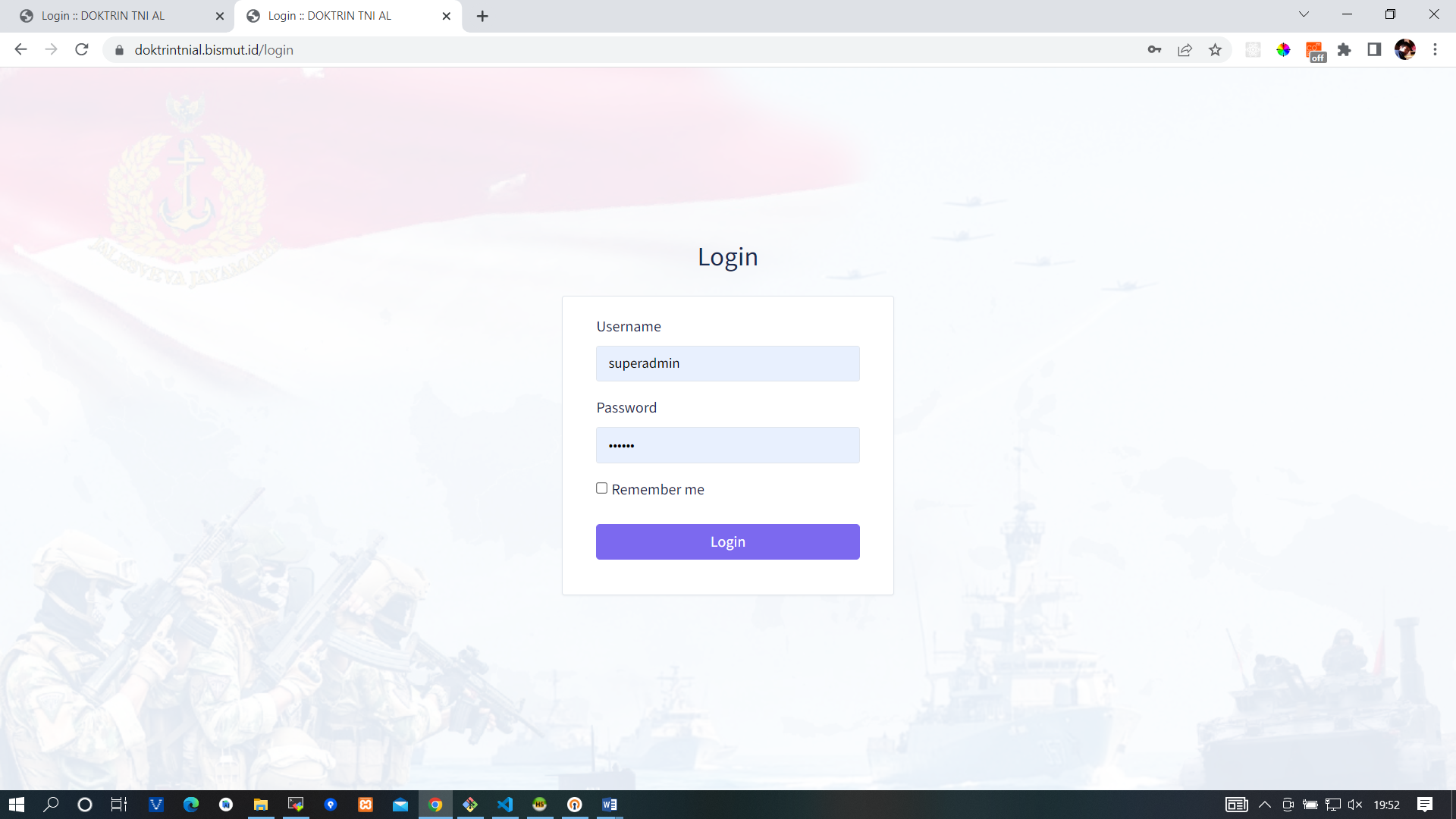
            $request->request->remove('password');

        }

        return $request;

    }

}



**resources\views\vendor\backpack\base\auth\login.blade.php**

@extends(backpack\_view('layouts.plain'))

@section('content')

    <div class="row justify-content-center">

        <div class="col-12 col-md-8 col-lg-4">

            <h3 class="text-center mb-4">Login</h3>

            <div class="card">

                <div class="card-body">

                    <form class="col-md-12 p-t-10" role="form" method="POST" action="{{ route('backpack.auth.login') }}">

                        {!! csrf\_field() !!}

                        <div class="form-group">

                            <label class="control-label" for="{{ $username }}">{{ config('backpack.base.authentication\_column\_name') }}</label>

                            <div>

                                <input type="text" class="form-control{{ $errors->has($username) ? ' is-invalid' : '' }}" name="{{ $username }}" value="{{ old($username) }}" id="{{ $username }}">

                                @if ($errors->has($username))

                                    <span class="invalid-feedback">

                                        <strong>{{ $errors->first($username) }}</strong>

                                    </span>

                                @endif

                            </div>

                        </div>

                        <div class="form-group">

                            <label class="control-label" for="password">Password</label>

                            <div>

                                <input type="password" class="form-control{{ $errors->has('password') ? ' is-invalid' : '' }}" name="password" id="password">

                                @if ($errors->has('password'))

                                    <span class="invalid-feedback">

                                        <strong>{{ $errors->first('password') }}</strong>

                                    </span>

                                @endif

                            </div>

                        </div>

                        <div class="form-group">

                            <div>

                                <div class="checkbox">

                                    <label>

                                        <input type="checkbox" name="remember"> Remember me

                                    </label>

                                </div>

                            </div>

                        </div>

                        <div class="form-group">

                            <div>

                                <button type="submit" class="btn btn-block btn-primary">

                                    Login

                                </button>

                            </div>

                        </div>

                    </form>

                </div>

            </div>

            @if (backpack\_users\_have\_email() && config('backpack.base.setup\_password\_recovery\_routes', true))

                <div class="text-center"><a href="{{ route('backpack.auth.password.reset') }}">{{ trans('backpack::base.forgot\_your\_password') }}</a></div>

            @endif

            @if (config('backpack.base.registration\_open'))

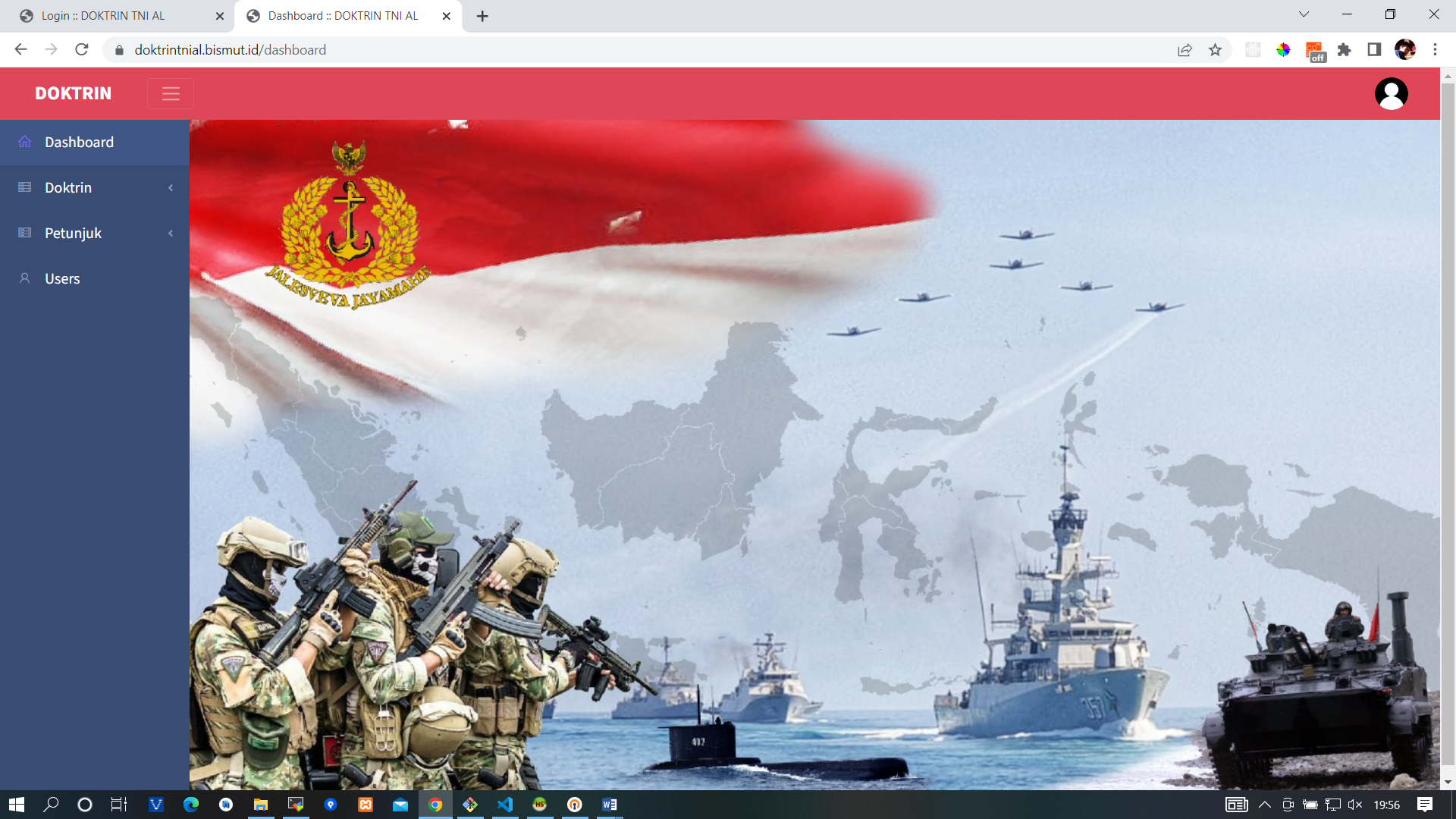
                <div class="text-center"><a href="{{ route('backpack.auth.register') }}">{{ trans('backpack::base.register') }}</a></div>

            @endif

        </div>

    </div>

@endsection



**resources\views\vendor\backpack\base\dashboard.blade.php**

@extends(backpack\_view('blankds'))

@section('content')

<style>

    #bg {

  /\* Preserve aspet ratio \*/

  min-width: 100%;

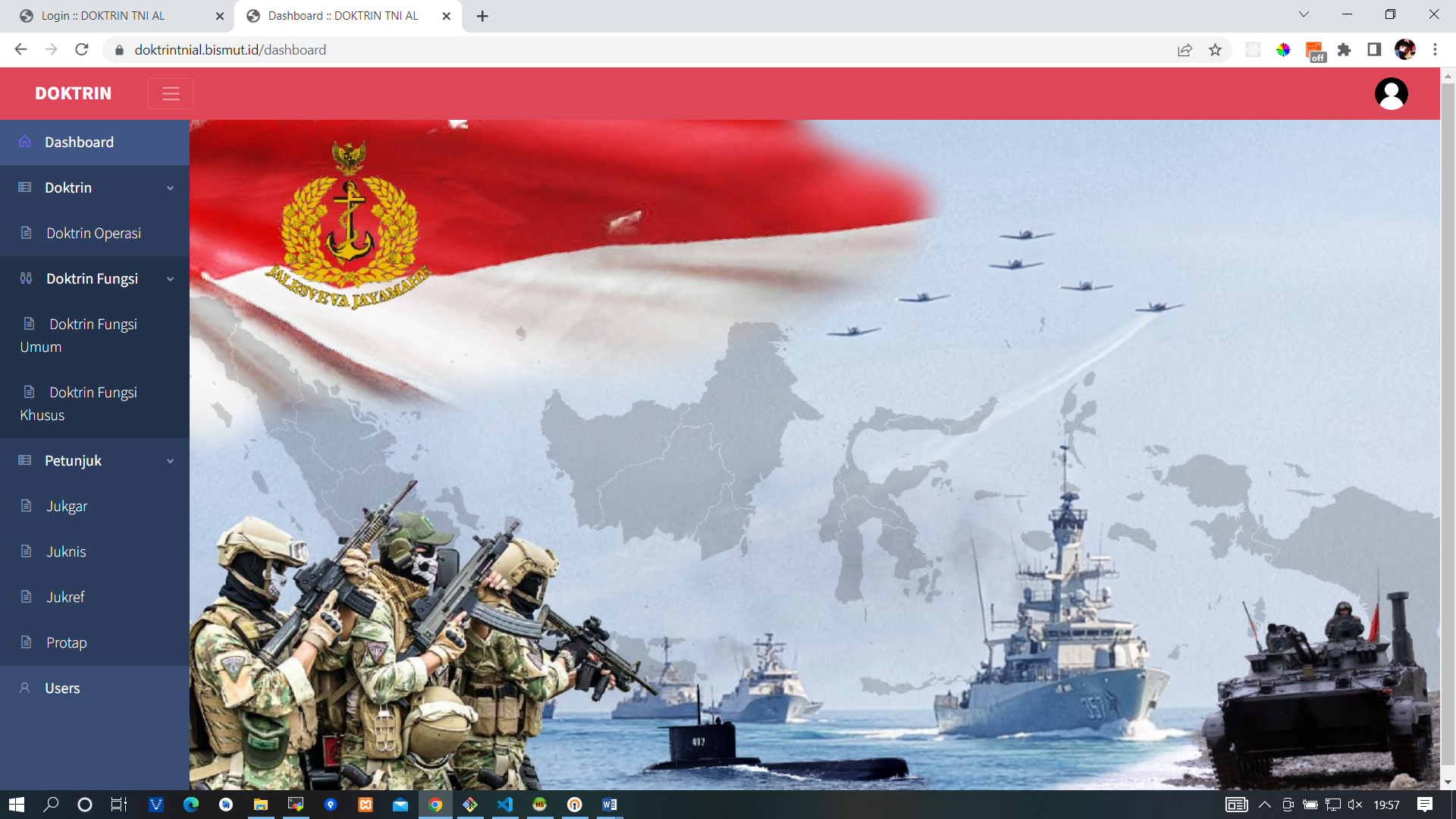
  min-height: 100%;

}

</style>

<img src="{{asset('tni.jpeg')}}" id="bg" alt="">

@endsection



**resources\views\vendor\backpack\base\inc\sidebar\_content.blade.php**

<li class="nav-item"><a class="nav-link" href="{{ backpack\_url('dashboard') }}"><i class="la la-home nav-icon"></i>

        {{ trans('backpack::base.dashboard') }}</a></li>

<li class="nav-item nav-dropdown"><a class="nav-link nav-dropdown-toggle" href="#"><i

            class="nav-icon las la-server"></i> Doktrin</a>

    <ul class="nav-dropdown-items">

        <li class='nav-item' style="padding-left: 0.1rem; font-weight: 300;"><a class='nav-link'

                href='{{ backpack\_url('doktrin') }}'><i class='nav-icon las la-file-alt'></i>

                Doktrin Operasi</a></li>

        <li class="nav-item nav-dropdown"><a class="nav-link nav-dropdown-toggle" href="#"

                style="padding-left: 1.1rem;"><i class="nav-icon las la-restroom"></i> Doktrin Fungsi</a>

            <ul class="nav-dropdown-items">

                <li class='nav-item' style="padding-left: 0.3rem; font-weight: 300;"><a class='nav-link'

                        href='{{ backpack\_url('doktrin-fungsi-umum') }}'><i class='nav-icon las la-file-alt'></i>

                        Doktrin

                        Fungsi Umum</a></li>

                <li class='nav-item' style="padding-left: 0.3rem; font-weight: 300;"><a class='nav-link'

                        href='{{ backpack\_url('doktrin-fungsi-khusus') }}'><i class='nav-icon las la-file-alt'></i>

                        Doktrin

                        Fungsi Khusus</a></li>

            </ul>

        </li>

    </ul>

</li>

<li class="nav-item nav-dropdown"><a class="nav-link nav-dropdown-toggle" href="#"><i

            class="nav-icon las la-server"></i> Petunjuk</a>

    <ul class="nav-dropdown-items">

        <li class='nav-item' style="padding-left: 0.1rem; font-weight: 300;"><a class='nav-link'

                href='{{ backpack\_url('jukgar') }}'><i class='nav-icon las la-file-alt'></i>

                Jukgar</a></li>

        <li class='nav-item' style="padding-left: 0.1rem; font-weight: 300;"><a class='nav-link'

                href='{{ backpack\_url('juknis') }}'><i class='nav-icon las la-file-alt'></i>

                Juknis</a></li>

        <li class='nav-item' style="padding-left: 0.1rem; font-weight: 300;"><a class='nav-link'

                href='{{ backpack\_url('jukref') }}'><i class='nav-icon las la-file-alt'></i>

                Jukref</a></li>

        <li class='nav-item' style="padding-left: 0.1rem; font-weight: 300;"><a class='nav-link'

                href='{{ backpack\_url('protap') }}'><i class='nav-icon las la-file-alt'></i>

                Protap</a></li>

    </ul>

</li>

@role('superadmin')

<li class="nav-item"><a class="nav-link" href="{{ backpack\_url('user') }}"><i class="nav-icon la la-user"></i>

        <span>Users</span></a></li>

@endrole

**database\migrations\2022\_02\_19\_042743\_doktrin.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class Doktrin extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('doktrins', function (Blueprint $table) {

            $table->id();

            $table->string('no\_doktrin');

            $table->string('name');

            $table->date('tgl');

            $table->string('file');

            $table->string('pembina')->nullable();

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('doktrins');

    }

}

**app\Models\Doktrin.php**

<?php

namespace App\Models;

use Backpack\CRUD\app\Models\Traits\CrudTrait;

use Illuminate\Database\Eloquent\Model;

class Doktrin extends Model

{

    use CrudTrait;

    /\*

    |--------------------------------------------------------------------------

    | GLOBAL VARIABLES

    |--------------------------------------------------------------------------

    \*/

    protected $table = 'doktrins';

    // protected $primaryKey = 'id';

    // public $timestamps = false;

    protected $guarded = ['id'];

    // protected $fillable = [];

    // protected $hidden = [];

    // protected $dates = [];

    /\*

    |--------------------------------------------------------------------------

    | FUNCTIONS

    |--------------------------------------------------------------------------

    \*/

    public function setFileAttribute($value)

    {

        $attribute\_name = "file";

        $disk = "public\_beneran";

        $destination\_path = 'file/' . date("Ymd");

        $this->uploadFileToDisk($value, $attribute\_name, $disk, $destination\_path);

    }

}

**app\Http\Controllers\Admin\DoktrinCrudController.php**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Requests\DoktrinRequest;

use App\Models\Doktrin;

use Backpack\CRUD\app\Http\Controllers\CrudController;

use Backpack\CRUD\app\Library\CrudPanel\CrudPanelFacade as CRUD;

/\*\*

 \* Class DoktrinCrudController

 \* @package App\Http\Controllers\Admin

 \* @property-read \Backpack\CRUD\app\Library\CrudPanel\CrudPanel $crud

 \*/

class DoktrinCrudController extends CrudController

{

    use \Backpack\CRUD\app\Http\Controllers\Operations\ListOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\CreateOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\UpdateOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\DeleteOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\ShowOperation;

    /\*\*

     \* Configure the CrudPanel object. Apply settings to all operations.

     \*

     \* @return void

     \*/

    public function setup()

    {

        CRUD::setModel(\App\Models\Doktrin::class);

        CRUD::setRoute(config('backpack.base.route\_prefix') . '/doktrin');

        CRUD::setEntityNameStrings('doktrin operasi', 'doktrin operasi');

    }

    /\*\*

     \* Define what happens when the List operation is loaded.

     \*

     \* @see  https://backpackforlaravel.com/docs/crud-operation-list-entries

     \* @return void

     \*/

    protected function setupListOperation()

    {

        $this->crud->addButtonFromView('line', 'preview', 'preview', 'end');

        $this->crud->addButtonFromView('line', 'download', 'download', 'end');

        $this->crud->denyAccess('show');

        CRUD::column('id')->label('No');

        CRUD::column('no\_doktrin');

        CRUD::column('name');

        CRUD::column('tgl');

        CRUD::column('pembina');

        // CRUD::column('file');

        // CRUD::column('created\_at');

        // CRUD::column('updated\_at');

        /\*\*

         \* Columns can be defined using the fluent syntax or array syntax:

         \* - CRUD::column('price')->type('number');

         \* - CRUD::addColumn(['name' => 'price', 'type' => 'number']);

         \*/

    }

    /\*\*

     \* Define what happens when the Create operation is loaded.

     \*

     \* @see https://backpackforlaravel.com/docs/crud-operation-create

     \* @return void

     \*/

    protected function setupCreateOperation()

    {

        CRUD::setValidation(DoktrinRequest::class);

        CRUD::field('no\_doktrin');

        CRUD::field('name');

        CRUD::field('tgl')->type('date');

        CRUD::field('pembina');

        CRUD::field('file')->type('upload')->upload(true)->disk('public\_beneran');

        /\*\*

         \* Fields can be defined using the fluent syntax or array syntax:

         \* - CRUD::field('price')->type('number');

         \* - CRUD::addField(['name' => 'price', 'type' => 'number']));

         \*/

    }

    /\*\*

     \* Define what happens when the Update operation is loaded.

     \*

     \* @see https://backpackforlaravel.com/docs/crud-operation-update

     \* @return void

     \*/

    protected function setupUpdateOperation()

    {

        $this->setupCreateOperation();

    }

    public function download($id)

    {

        $doktrin = Doktrin::find($id);

        $file\_path = public\_path($doktrin->file);

        return response()->download($file\_path);

    }

    public function previewpdf($id)

    {

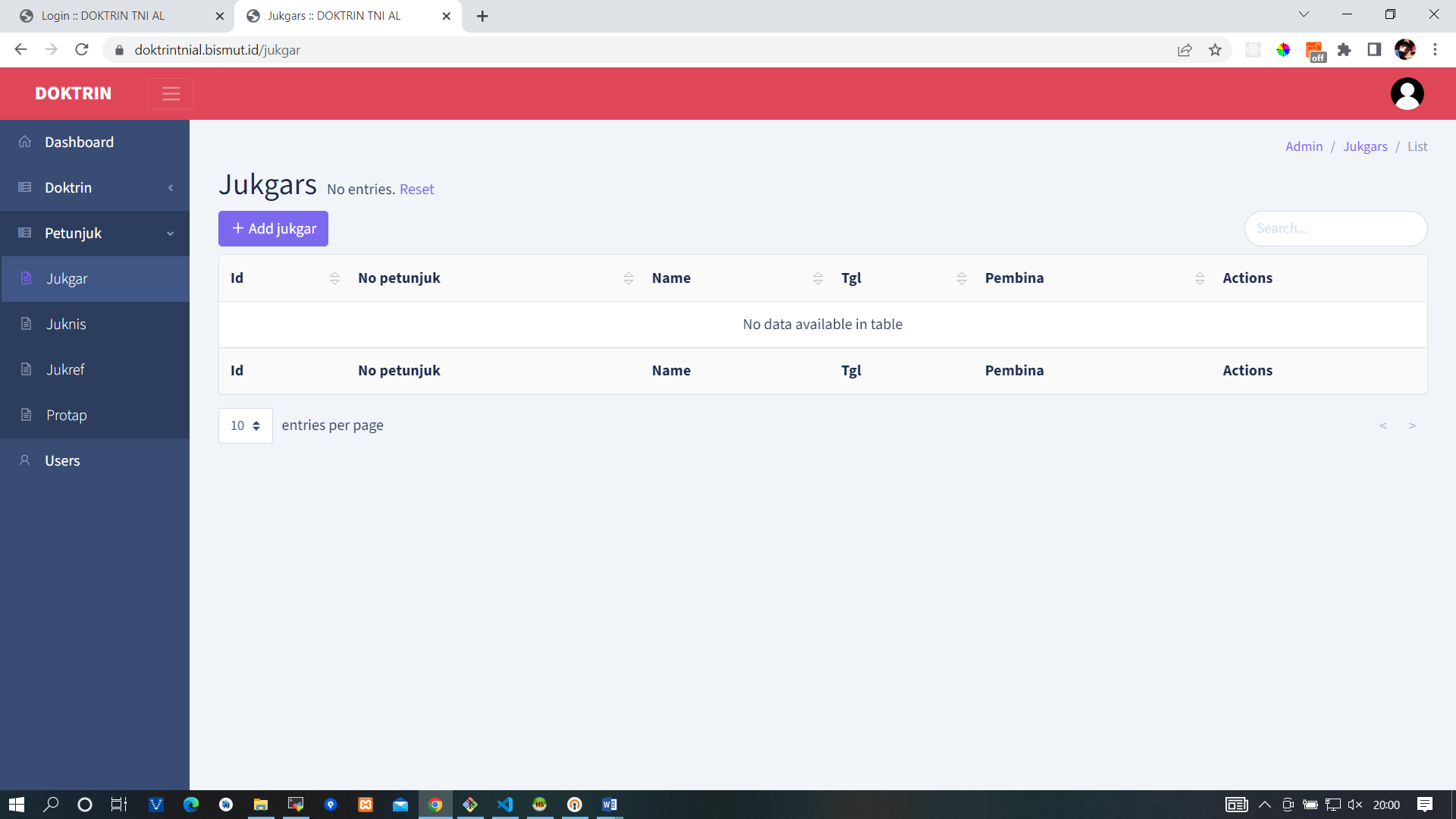
        $doktrin = Doktrin::find($id);

        $file\_path = public\_path($doktrin->file);

        return response()->file($file\_path);

    }

}



**database\migrations\2022\_04\_06\_172102\_jukgar.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class Jukgar extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('jukgars', function (Blueprint $table) {

            $table->id();

            $table->string('no\_petunjuk');

            $table->string('name');

            $table->date('tgl');

            $table->string('file');

            $table->string('pembina')->nullable();

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('jukgars');

    }

}

**app\Models\Jukgar.php**

<?php

namespace App\Models;

use Backpack\CRUD\app\Models\Traits\CrudTrait;

use Illuminate\Database\Eloquent\Model;

class Jukgar extends Model

{

    use CrudTrait;

    /\*

    |--------------------------------------------------------------------------

    | GLOBAL VARIABLES

    |--------------------------------------------------------------------------

    \*/

    protected $table = 'jukgars';

    // protected $primaryKey = 'id';

    // public $timestamps = false;

    protected $guarded = ['id'];

    // protected $fillable = [];

    // protected $hidden = [];

    // protected $dates = [];

    public function setFileAttribute($value)

    {

        $attribute\_name = "file";

        $disk = "public\_beneran";

        $destination\_path = 'file/' . date("Ymd");

        $this->uploadFileToDisk($value, $attribute\_name, $disk, $destination\_path);

    }

}

**app\Http\Controllers\Admin\JukgarCrudController.php**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Requests\JukgarRequest;

use App\Models\Jukgar;

use Backpack\CRUD\app\Http\Controllers\CrudController;

use Backpack\CRUD\app\Library\CrudPanel\CrudPanelFacade as CRUD;

/\*\*

 \* Class JukgarCrudController

 \* @package App\Http\Controllers\Admin

 \* @property-read \Backpack\CRUD\app\Library\CrudPanel\CrudPanel $crud

 \*/

class JukgarCrudController extends CrudController

{

    use \Backpack\CRUD\app\Http\Controllers\Operations\ListOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\CreateOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\UpdateOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\DeleteOperation;

    use \Backpack\CRUD\app\Http\Controllers\Operations\ShowOperation;

    /\*\*

     \* Configure the CrudPanel object. Apply settings to all operations.

     \*

     \* @return void

     \*/

    public function setup()

    {

        CRUD::setModel(\App\Models\Jukgar::class);

        CRUD::setRoute(config('backpack.base.route\_prefix') . '/jukgar');

        CRUD::setEntityNameStrings('jukgar', 'jukgars');

    }

    /\*\*

     \* Define what happens when the List operation is loaded.

     \*

     \* @see  https://backpackforlaravel.com/docs/crud-operation-list-entries

     \* @return void

     \*/

    protected function setupListOperation()

    {

        $this->crud->addButtonFromView('line', 'preview', 'preview', 'end');

        $this->crud->addButtonFromView('line', 'download', 'download', 'end');

        $this->crud->denyAccess('show');

        CRUD::column('id');

        CRUD::column('no\_petunjuk');

        CRUD::column('name');

        CRUD::column('tgl');

        // CRUD::column('file');

        CRUD::column('pembina');

        // CRUD::column('created\_at');

        // CRUD::column('updated\_at');

        /\*\*

         \* Columns can be defined using the fluent syntax or array syntax:

         \* - CRUD::column('price')->type('number');

         \* - CRUD::addColumn(['name' => 'price', 'type' => 'number']);

         \*/

    }

    /\*\*

     \* Define what happens when the Create operation is loaded.

     \*

     \* @see https://backpackforlaravel.com/docs/crud-operation-create

     \* @return void

     \*/

    protected function setupCreateOperation()

    {

        CRUD::setValidation(JukgarRequest::class);

        // CRUD::field('id');

        CRUD::field('no\_petunjuk');

        CRUD::field('name');

        CRUD::field('tgl')->type('date');

        CRUD::field('file')->type('upload')->upload(true)->disk('public\_beneran');

        CRUD::field('pembina');

        // CRUD::field('created\_at');

        // CRUD::field('updated\_at');

        /\*\*

         \* Fields can be defined using the fluent syntax or array syntax:

         \* - CRUD::field('price')->type('number');

         \* - CRUD::addField(['name' => 'price', 'type' => 'number']));

         \*/

    }

    /\*\*

     \* Define what happens when the Update operation is loaded.

     \*

     \* @see https://backpackforlaravel.com/docs/crud-operation-update

     \* @return void

     \*/

    protected function setupUpdateOperation()

    {

        $this->setupCreateOperation();

    }

    public function download($id)

    {

        $doktrin = Jukgar::find($id);

        $file\_path = public\_path($doktrin->file);

        return response()->download($file\_path);

    }

    public function previewpdf($id)

    {

        $doktrin = Jukgar::find($id);

        $file\_path = public\_path($doktrin->file);

        return response()->file($file\_path);

    }

}

