Implémentation des cruds dans le projet Mini Blog

# Ajout d’article

1 – Ajout du bouton « Ajouter un article » dans la page index.

    <div class="d-flex justify-content-between align-items-center mb-4">

        <h1>Articles du blog</h1>

        <a href="{{ route('posts.create') }}" class="btn btn-success">Ajouter un article</a>

    </div>

2 – Ajouter la route posts.create dans web.php

Route::get('/posts/create', [PostController::class, 'create'])->name('posts.create');

3 – ajouter la méthode create dans PostController

    public function create()

    {

        return view('posts.create');

    }

4 – Ajouter la view create.blade.php dans le dossier posts

@extends('layouts.app')

@section('title', 'Ajouter un article')

@section('content')

<div class="container">

    <h1 class="mb-4">Ajouter un nouvel article</h1>

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

        @csrf

        <div class="mb-3">

            <label for="title" class="form-label">Titre</label>

            <input type="text" class="form-control @error('title') is-invalid @enderror" id="title" name="title"

                value="{{ old('title') }}" required>

            @error('title')

            <div class="invalid-feedback">{{ $message }}</div>

            @enderror

        </div>

        <div class="mb-3">

            <label for="content" class="form-label">Contenu</label>

            <textarea class="form-control @error('content') is-invalid @enderror" id="content" name="content" rows="6"

                required>{{ old('content') }}</textarea>

            @error('content')

            <div class="invalid-feedback">{{ $message }}</div>

            @enderror

        </div>

        <div class="mb-3">

            <button type="submit" class="btn btn-success">Publier l'article</button>

            <a href="{{ route('posts.index') }}" class="btn btn-secondary">Annuler</a>

        </div>

    </form>

</div>

@endsection

5 – Ajouter la route posts.store dans web.php

Route::post('/posts', [PostController::class, 'store'])->name('posts.store');

6 – Ajouter le validateur App/http/requests/ StorePostRequest

Php artisan make :Request StorePostReuqest

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StorePostRequest extends FormRequest

{

    /\*\*

     \* Determine if the user is authorized to make this request.

     \*/

    public function authorize(): bool

    {

        return true;

    }

    /\*\*

     \* Get the validation rules that apply to the request.

     \*

     \* @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>

     \*/

    public function rules(): array

    {

        return [

            'title' => 'required|min:3|max:255',

            'content' => 'required|min:10',

        ];

    }

    /\*\*

     \* Get custom messages for validator errors.

     \*

     \* @return array<string, string>

     \*/

    public function messages(): array

    {

        return [

            'title.required' => 'Le titre est obligatoire',

            'title.min' => 'Le titre doit contenir au moins :min caractères',

            'title.max' => 'Le titre ne peut pas dépasser :max caractères',

            'content.required' => 'Le contenu est obligatoire',

            'content.min' => 'Le contenu doit contenir au moins :min caractères',

        ];

    }

}

7 – Ajouter la méthode store dans PostController

use App\Http\Requests\StorePostRequest;

public function store(StorePostRequest $request)

    {

        // La validation est déjà gérée par StorePostRequest

        // Création du post

        Post::create([

            'title' => $request->title,

            'content' => $request->content,

        ]);

        // Redirection avec message de succès

        return redirect()->route('posts.index')

            ->with('success', 'L\'article a été créé avec succès!');

    }

8 – Ajouter Success div à l’index

@if(session('success'))

        <div class="alert alert-success alert-dismissible fade show" role="alert">

            {{ session('success') }}

            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

        </div>

    @endif

9 – Tester L’application

Modifier l’article

1 – Ajouter le bouton Modifier dans show

   <div class="d-flex justify-content-between align-items-center mb-3">

            <h1 class="card-title mb-0">{{ $post->title }}</h1>

            <div>

                <a href="{{ route('posts.edit', $post) }}" class="btn btn-primary">Modifier</a>

            </div>

        </div>

2 – Ajouter les routes de modification

Route::get('/posts/{post}', [PostController::class, 'show'])->name('posts.show');

Route::get('/posts/{post}/edit', [PostController::class, 'edit'])->name('posts.edit');

3 – Ajouter les methodes du controller et la request

public function edit(Post $post)

    {

        return view('posts.edit', compact('post'));

    }

    public function update(StorePostRequest $request, Post $post)

    {

        // La validation est déjà gérée par StorePostRequest

        // Mise à jour du post

        $post->update([

            'title' => $request->title,

            'content' => $request->content,

        ]);

        // Redirection avec message de succès

        return redirect()->route('posts.show', $post)

            ->with('success', 'L\'article a été modifié avec succès!');

    }

4 – Ajouter la view

@extends('layouts.app')

@section('title', 'Modifier l\'article')

@section('content')

<div class="container">

    <h1 class="mb-4">Modifier l'article</h1>

    <form method="POST" action="{{ route('posts.update', $post) }}">

        @csrf

        @method('PUT')

        <div class="mb-3">

            <label for="title" class="form-label">Titre</label>

            <input type="text" class="form-control @error('title') is-invalid @enderror" id="title" name="title"

                value="{{ old('title', $post->title) }}" required>

            @error('title')

            <div class="invalid-feedback">{{ $message }}</div>

            @enderror

        </div>

        <div class="mb-3">

            <label for="content" class="form-label">Contenu</label>

            <textarea class="form-control @error('content') is-invalid @enderror" id="content" name="content" rows="6"

                required>{{ old('content', $post->content) }}</textarea>

            @error('content')

            <div class="invalid-feedback">{{ $message }}</div>

            @enderror

        </div>

        <div class="mb-3">

            <button type="submit" class="btn btn-primary">Mettre à jour</button>

            <a href="{{ route('posts.show', $post) }}" class="btn btn-secondary">Annuler</a>

        </div>

    </form>

</div>

@endsection

5 – tester la modification

# Supprimer l’article

1 – Ajouter le bouton de suppression acoté du bouton de la modification avec la confirmation dans show

   <div class="d-flex justify-content-between align-items-center mb-3">

            <h1 class="card-title mb-0">{{ $post->title }}</h1>

            <div>

                <a href="{{ route('posts.edit', $post) }}" class="btn btn-primary">Modifier</a>

                <form action="{{ route('posts.destroy', $post) }}" method="POST" class="d-inline">

                    @csrf

                    @method('DELETE')

                    <button type="submit" class="btn btn-danger" onclick="return confirm('Êtes-vous sûr de vouloir supprimer cet article?')">Supprimer</button>

                </form>

            </div>

        </div>

2 – Ajouter la route dans web.php

Route::delete('/posts/{post}', [PostController::class, 'destroy'])->name('posts.destroy');

3 – ajouter la suppression dans le controller

public function destroy(Post $post)

    {

        // Suppression du post

        $post->delete();

        // Redirection avec message de succès

        return redirect()->route('posts.index')

            ->with('success', 'L\'article a été supprimé avec succès!');

    }

Tester la suppression