Technisch Ontwerp

Shape

Description automatically generated

Inhoudsopgave

1. Inleiding
   1. Summary
2. Architectuur
   1. Subsystemen
   2. Hardware
   3. Software
   4. Opslag
   5. Dependecies
3. Model
   1. Database
   2. MVC
4. Interface
   1. Schermen
5. Processen
   1. Gebruikersfuncties
   2. Systeemfuncties
6. Migrations
   1. Migrations
7. Controllers
8. Controllers

Inleiding

*Summary*

Een opdracht van school voor A5. De opdracht is het leren samen werken en leren van PHP framework ‘Laravel’ als zowel dingen omheen als Bootstrap, HTML etc. In deze opdracht maken we een EPK hierin moeten we bands kunnen aanmaken in die bands moeten mensen hun muziek (youtube embed) foto’s en beschrijving kunnen uploaden.

Architectuur

*Subsystemen*

We gebruiken PHP, Laravel, HTML & CSS, Bootstrap en MySQL voor de database

*Hardware*

We gebruiken gewoon onze laptops en PC’s van thuis

*Software*

Alles word ook gemaakt in Windows

*Opslag*

Alle opslag dat word gebruikt in de site word op MySQL opgeslagen in je local database

*Database*

We gebruiken MySQL als Database voor ons project :

*Dependecies*

We hebben de dependecies: BenSampo / laravel-embed en cviebrock/eloquent-sluggable.

**laravel-embed** ervoor dat youtube links automatische ge-embed wordt.

**eloquent-sluggable** zorgt ervoor dat de url van een band de naam van de band is dus als ik een band aanmaakt die heet : Rolling Stones dan wordt de url http://127.0.0.1:8000/EPK/rolling-stones

Models

We hebben 2 models in dit project “User” en “Post”

class Post extends Model

{

    use HasFactory;

    use Sluggable;

    protected $fillable = ['title', 'slug', 'description', 'image\_path', 'user\_id', 'youtube\_url','youtube\_url2','youtube\_url3','biography'];

    public function user()

    {

        return **$this**->belongsTo(User::class);

    }

    public function sluggable():array

    {

        return [

            'slug' => [

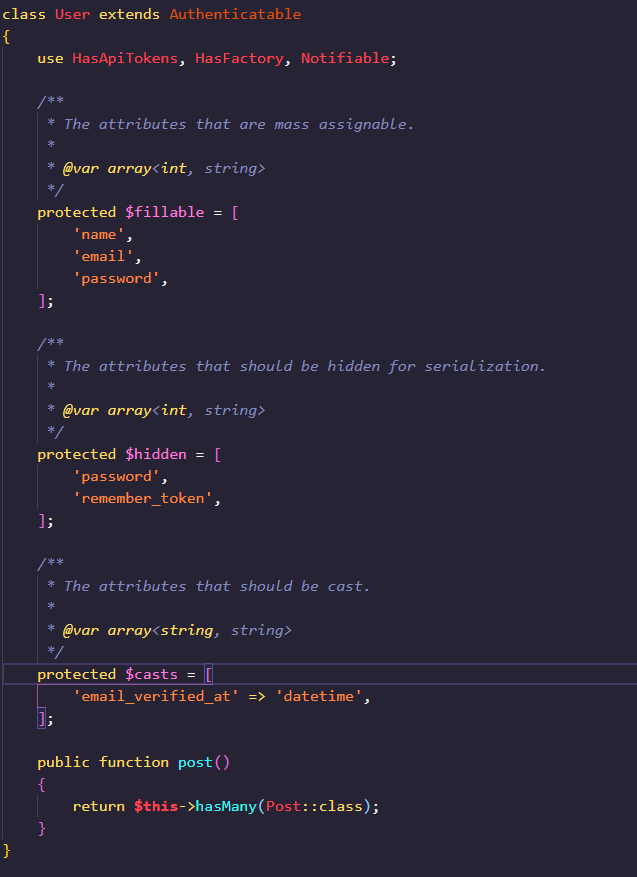
                'source' => 'title'

            ]

        ];

    }

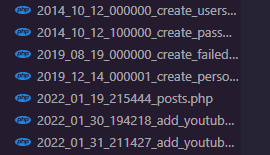
Hier wordt alles gedefineerd wat er in een band komt te staan, ook hebben we een belongsTo naar de user table deze worden gelinkt met user\_id en id



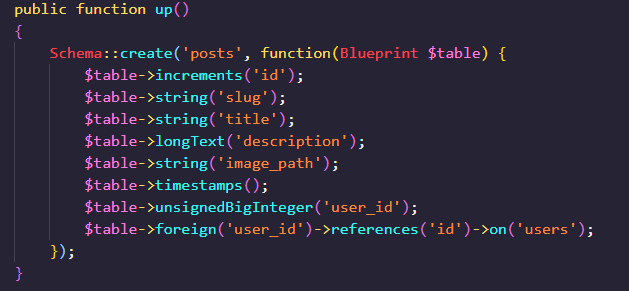
De user class is ge autogenerate doormiddel van laravel-ui we hebben zelf een One to many relationship toegevoegd met Post

Migrations

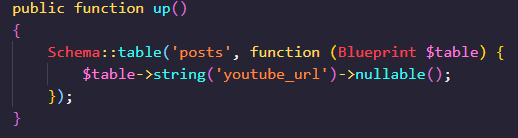
We hebben totaal 7 migrations de eerste 4 zijn auto generated doormiddel van laravel ui, in de eerste 4 migrations worden de user tables gemaakt.



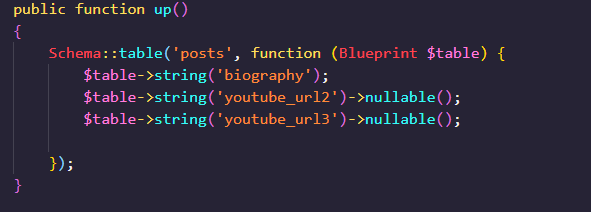
Eerste migration Post : hierin wordt alles gemaakt wat met er in een band komt te staan, de slug is de url wat in de browser komt te staan de slug is gebaseerd op de titel.



In de tweede migration wordt alleen de youtube url toegevoegd aan de database deze link wordt ge-embed doormiddel van de dependecy “BenSampo / laravel-embed”



In de laatste migration heb ik youtube links toegevoegd een een kolom voor een biografie.



Processen

*Routes*

 Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

Route::resource('/', PostsController::class);

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

Route::resource('/EPK', PostsController::class);

Route::get('/search', [App\Http\Controllers\HomeController::class, 'search'])->name('EPK.search');

Controllers

We hebben 2 controllers een HomeController een PostsController



In de HomeController heb ik 3 dingen staan de \_\_construct function maakt middleware aan voor de homepagina, in de index function wordt de standaard pagina gedefineerd, en in de search function maak ik een functie voor een zoekbalk voor de homepagina.

Postcontroller

In de \_\_construct function wordt middleware aangeroepen die controleerd als je bent ingelogd behalve in index en show.

class PostsController extends Controller

{

    public function \_\_construct()

    {

**$this**->middleware('auth', ['except' => ['index', 'show']]);

    }

Index function haalt de posts op en wordt gesorteerd op de datum

public function index()

    {

        return view('index')->with('posts', Post::orderBy('updated\_at', 'DESC')->get());

    }

Roept epk.create.blade.php aan

 public function create()

    {

        return view('EPK.create');

    }

 public function store(Request $request)

    {

        $request->validate([

            'title' => 'required',

            'description' => 'required',

            'image' => 'required|mimes:jpg,png,jpeg|max:5048'

        ]);

        $newImageName = uniqid() . '-' . $request->title . '.' .

        $request->image->extension();

        $request->image->move(public\_path('images'), $newImageName);

        $slug = SlugService::createSlug(Post::class, 'slug', $request->title);

        Post::create([

            'title' => $request->input('title'),

            'description' => $request->input('description'),

            'slug' => SlugService::createSlug(Post::class, 'slug', $request->title),

            'image\_path' => $newImageName,

            'user\_id' => auth()->user()->id,

            'youtube\_url' => $request->input('youtube\_url'),

            'youtube\_url2' => $request->input('youtube\_url2'),

            'youtube\_url3' => $request->input('youtube\_url3'),

            'biography' => $request->input('biography')

        ]);

        return redirect('/create')->with('message', 'Je band is gepubliceerd');

    }

$request->validate checkt als je de titel description en een foto hebt toegevoegd

$newImageName generated een random naam voor de image die je upload

$slug is wat je ziet in de url, de slug is gebaseerd op de naam van de band die je invult.

Function show laat epk.show.blade zien.

public function show($slug)

    {

        return view('epk.show')

        ->with('post', Post::where('slug', $slug)->first());

    }

Met function edit kan je de post editen

public function edit($slug)

    {

        return view('epk.edit')

        ->with('post', Post::where('slug', $slug)->first());

    }

Function update lijkt op store hiermee kun je de post updaten.

public function update(Request $request, $slug)

    {

        $request->validate([

            'title' => 'required',

            'description' => 'required',

        ]);

        Post::where('slug', $slug)

            ->update([

                'title' => $request->input('title'),

                'description' => $request->input('description'),

                'slug' => SlugService::createSlug(Post::class, 'slug', $request->title),

                'user\_id' => auth()->user()->id

            ]);

        return redirect('/EPK')

            ->with('message', 'Your post has been updated!');

    }

In de destroy function verwijdert hij de post

 public function destroy($slug)

    {

        $post = Post::where('slug', $slug);

        $post->delete();

        return redirect('/EPK')

            ->with('message', 'Your post has been deleted!');

    }

}