Visoka skola strukovnih studija za informacione I komunikacione tehnologije u Beogradu

Visoka ICT Skola

PHP 2 Dokumentacija za sajt

Aleksandar Marjanovic

Broj Indeksa: 115/17

Maj 2021, Beograd

Sadrzaj:

1. Uvod

1.1. Korišćeni programski i markup jezici

1.2. Opis funkcionalnosti

1.3 Kodovi

2 Baza, I odnosi tabela u bazi

3 Kljucna mesta u kodu

4 Eloquent Modeli

1 Uvod

Sajt je pravljen za predmet web programiranje php 2, koristeci laravel I vue js. Uz njih koriscena je I Spatie Media Library za postavljanje slika.

Sajt resave domen drustvane mreze, sto se tice funkcionalnosti lici na Instagram I fejsbuk

* 1. Korisceni jezici

Laravel I vue Js

Dependencies

**"@fortawesome/fontawesome-free"**: "^5.13.0",

**"@fortawesome/free-solid-svg-icons"**: "^5.13.0",

**"font-awesome"**: "^4.7.0",

**"laravel-vue-pagination"**: "^2.3.1",

**"moment"**: "^2.29.1",

**"vue-carousel"**: "^0.18.0",

**"vue-izitoast"**: "^1.2.1"

* 1. Opis funkcionalnosti

Pre ulaska u aplikaciju, sve kljucne rute su zabranjene za neulogovanog usera, dozvoljeno je samo ici na login, register, about I contact stranice.

Useri su razdvojeni sa rolama, odnosno admin I regular user (ili samo user, u bazi)

Oba usera kada se uloguju okida se isti controller koji na osnovu role ucitava dati pogled

Admin Dashboard ima pristup svim podacima na sajtu, svi postovi, svi useri, uz to moze da pogleda svoj profil I edituje isti kao I sve ostale podatke

Obican user kad se uloguje moze da vidi svoje postove I da pretrazi samo svoje postove, uz to postoji I arhiva postova, nesto slicno kaon a instagramu.

Za obe stranice je koriscen vue zbog axiosa

Oba korisnika imaju pristup svim postovima klikom na logo sajta, sto je slicno kao I na instagramovoj aplikaciji (s tim sto Instagram odmah korisnika posalje tu)

Postove moze, na toj stranici kao I na svim ostalima gde se edituju postovi, da edituje admin I user koji je napravio dati post. Ukoliko user nije napravio post, nece mu se ni prikazati dugme za edit, a ako pokusa nekako da pristupi toj ruti za edit, dobice 403 unauthorized.

Na stranici obicnog posta, umesto post id-ja ucitan je njegov slug, prikazan je post koji moze da se lajkuje, arhivira I da se komentarise na njega.

Ostavljanje komentara je takodje asinhrono I komentare kao I postove moze da edituje samo admin I user koji ih je napisao. Ceo krud tu radi aisnhrono (to ukljucuje komentarisanje, editovanje, brisanje komentara, arhiviranje I lajkovanje)

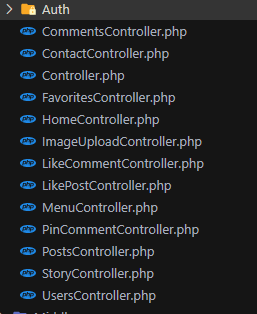
Ruta za kreiranje posta je dozvoljena svim autorizovanim userima, dozvoljeno je postavljanje slike, ali nije I obavezno. Kod editovanja posta, ukoliko se slike ne edituju nista se ne desava, kada se doda nova slika ili slike stare se brisu.

Tokom registracije usera I pravljenja posta dodat je exception za usere koji ne zele sliku u oba slucaja, tada se postavlja defaultna slika za post I za usera

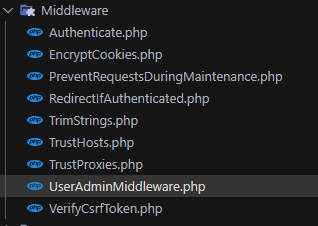
Contact I about stranicama mogu da pristupe svi, contact stranica bi trebala da salje mejl ako je sajt na nekom serveru (procitao sam da zbog toga ne radi generalno slanje mejla ali se podaci upisuju u bazu svakako)

* 1. Kodovi I css

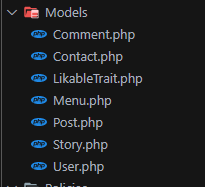
Korisceni kontroleri:



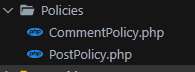
Koriscen middleware:



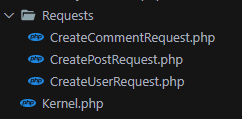
Modeli:



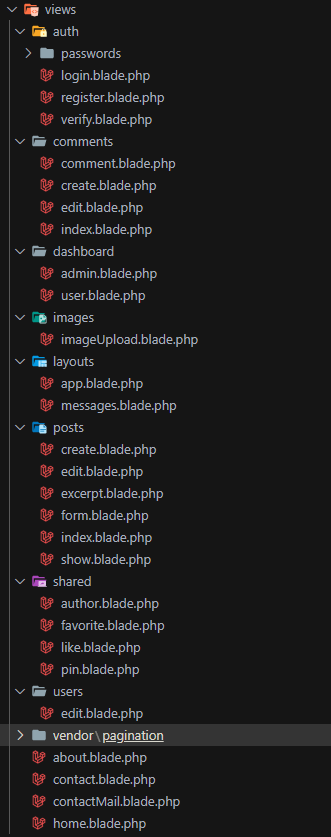
Polise:



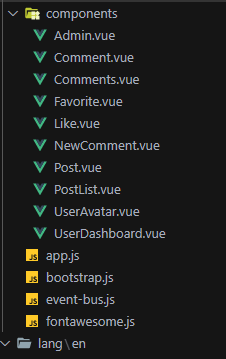
Requestovi:



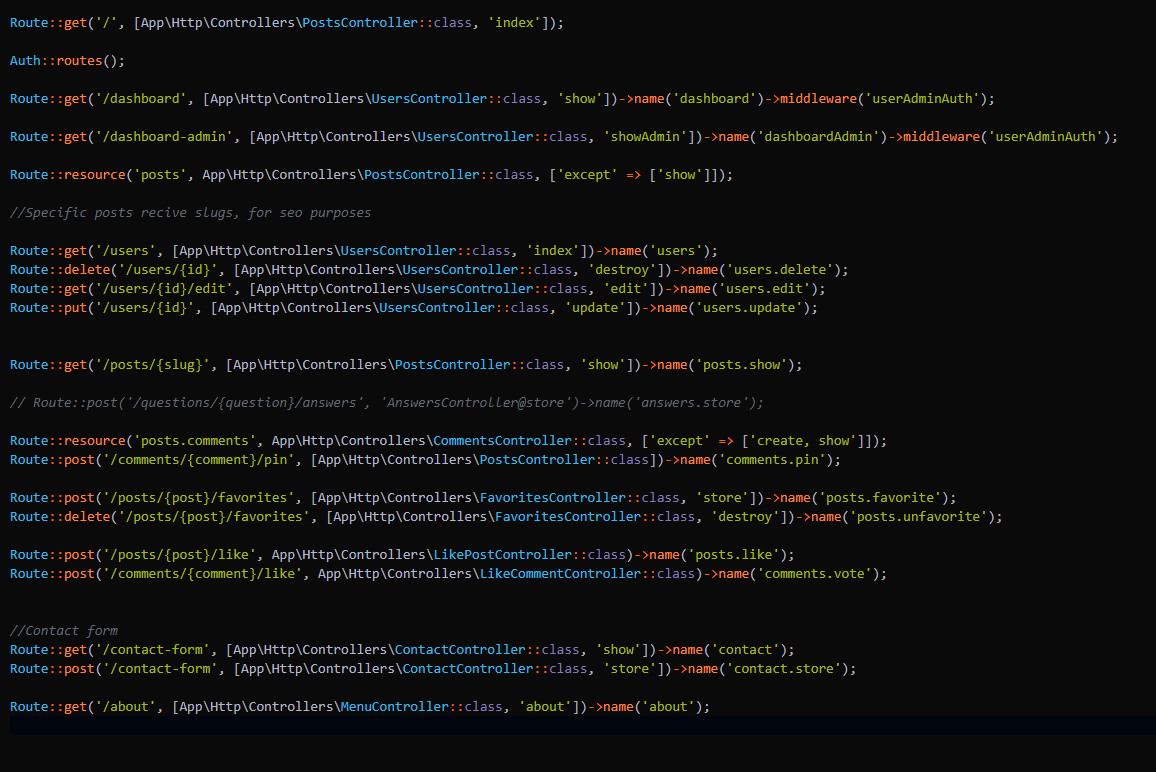
Views folder:



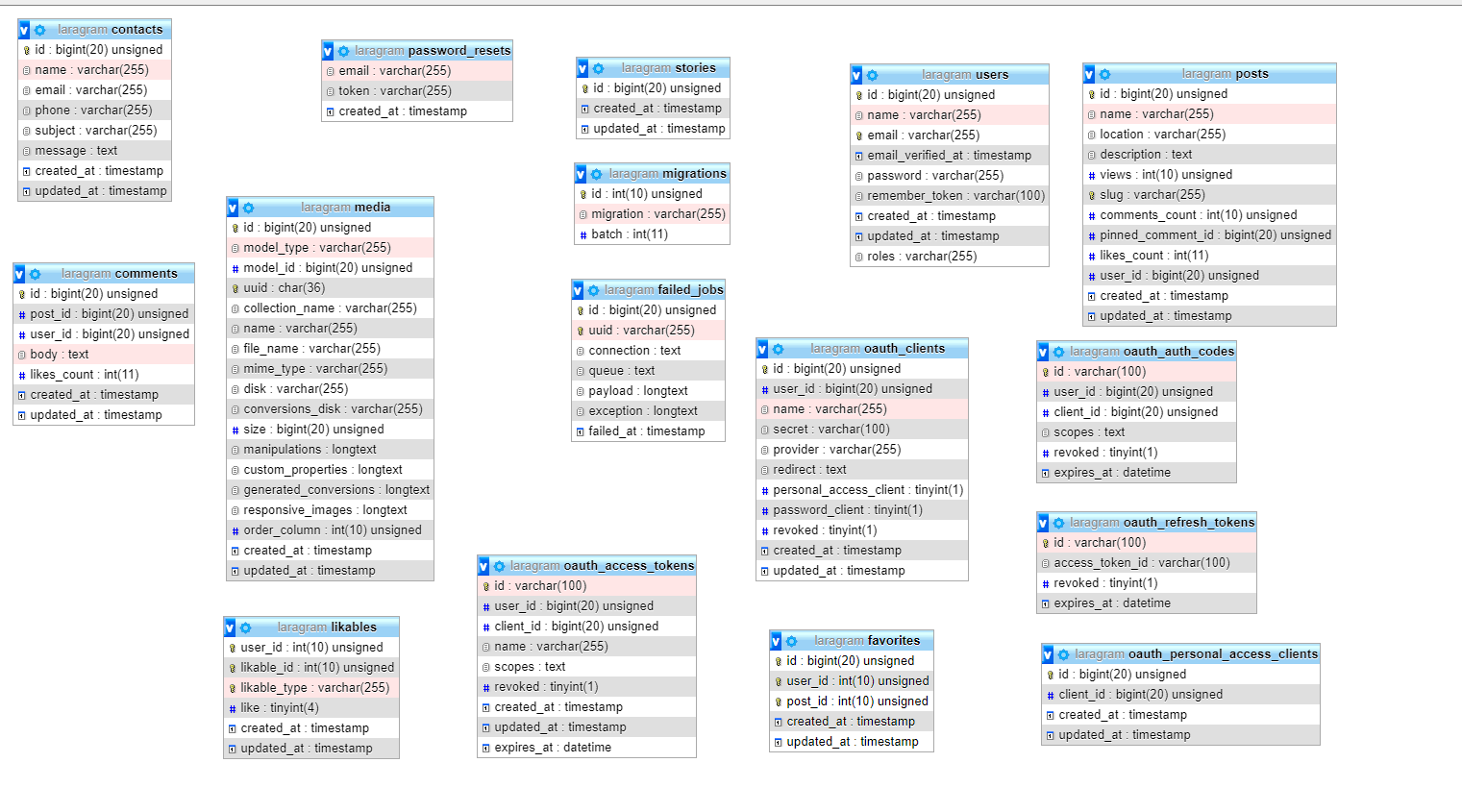
Vue views:



Registrovane rute:



1. Baza, slika odnosa tabela u bazi



1. Kljucna mesta u kodu:

Admin.vue Pretraga



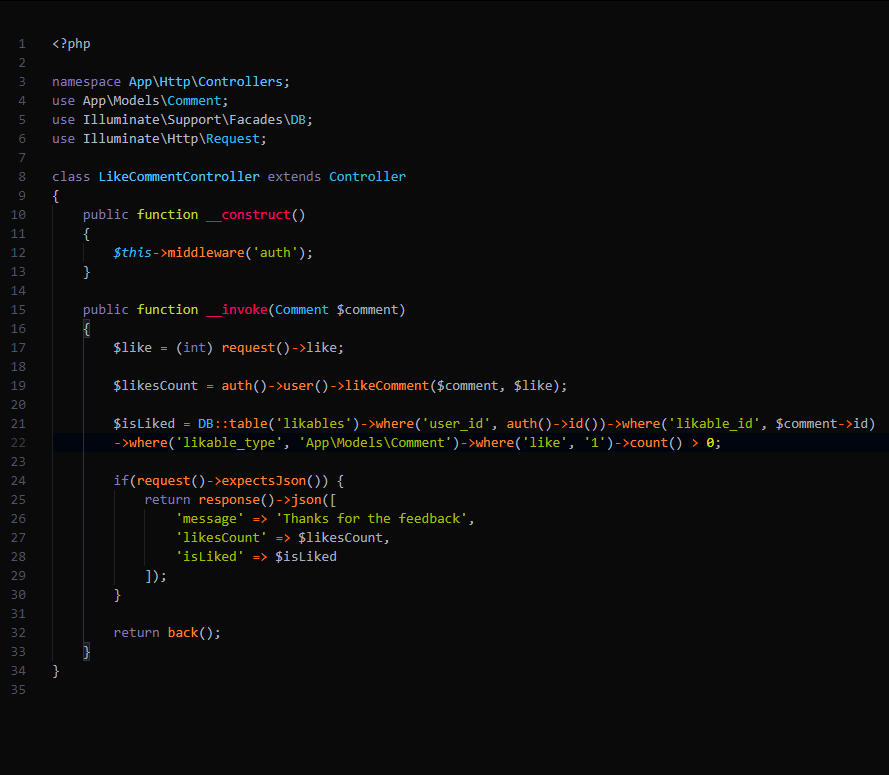
Posts controller vracanje rezultata pretrage



Likes.vue Lajkovanje



Like Controller





Favorite.vue (ili arhivacija, pomesao sam ta 2 naziva u aplikaciji par puta ali se odnosi na istu stvar)

<template>

  <a

    title="Click to mark as favorite post (Click again to undo)"

    :class="classes"

    @click.prevent="toggle"

  >

    <svg

      fill="gray"

      style="height: 55px; width: 55px"

      version="1.1"

      viewBox="0 0 48 48"

      xml:space="preserve"

      xmlns="http://www.w3.org/2000/svg"

      xmlns:xlink="http://www.w3.org/1999/xlink"

    >

      <g class="st0" id="Padding\_\_x26\_\_Artboard" />

      <g id="Icons">

        <g>

          <path

            class="st1"

            d="M30.37337,33.9599l-5.96166-8.79194c-0.19733-0.29102-0.62609-0.29102-0.82342,0l-5.96166,8.79194    c-0.27508,0.40568-0.90915,0.21098-0.90915-0.27917V14.31835c0-0.27473,0.22271-0.49744,0.49744-0.49744h13.57016    c0.27473,0,0.49744,0.22271,0.49744,0.49744v19.36238C31.28252,34.17088,30.64845,34.36558,30.37337,33.9599z"

          />

          <g>

            <line

              class="st1"

              x1="19.16389"

              x2="24.19888"

              y1="16.12722"

              y2="16.12722"

            />

            <line

              class="st1"

              x1="25.54154"

              x2="26.5905"

              y1="16.12722"

              y2="16.12722"

            />

          </g>

          <path

            class="st1"

            d="M30.37337,33.9599l-5.96166-8.79194c-0.19733-0.29102-0.62609-0.29102-0.82342,0l-5.96166,8.79194    c-0.27508,0.40568-0.90915,0.21098-0.90915-0.27917V14.31835c0-0.27473,0.22271-0.49744,0.49744-0.49744h13.57016    c0.27473,0,0.49744,0.22271,0.49744,0.49744v19.36238C31.28252,34.17088,30.64845,34.36558,30.37337,33.9599z"

          />

        </g>

      </g>

    </svg>

  </a>

</template>

<script>

export default {

  props: ["post"],

  data() {

    return {

      isFavorited: *this*.post.is\_favorited,

      count: *this*.post.favorites\_count,

      id: *this*.post.id,

    };

  },

  computed: {

    classes() {

      return [

        "favorite",

        "mt-2",

        !*this*.signedIn ? "off" : *this*.isFavorited ? "favorited" : "",

      ];

    },

    endpoint() {

      return `/posts/${*this*.id}/favorites`;

    },

  },

  methods: {

    toggle() {

      if (!*this*.signedIn) {

*this*.$toast.warning("Please login to favorite this post", "Warning", {

          timeout: 3000,

          position: "bottomLeft",

        });

        return;

      }

*this*.isFavorited ? *this*.destroy() : *this*.create();

    },

    destroy() {

      axios.delete(*this*.endpoint).then((res) => {

*this*.count--;

*this*.isFavorited = false;

      });

    },

    create() {

      axios.post(*this*.endpoint).then((res) => {

*this*.count++;

*this*.isFavorited = true;

      });

    },

  },

};

</script>

Favorites Controller

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Post;

class FavoritesController extends Controller

{

    public function \_\_construct()

    {

*$this*->middleware('auth');

    }

    public function store(Post $post)

    {

        $post->favorites()->attach(auth()->id());

        if(request()->expectsJson()) {

            return response()->json(null, 204);

        }

        return back();

    }

    public function destroy(Post $post)

    {

        $post->favorites()->detach(auth()->id());

        if(request()->expectsJson()) {

            return response()->json(null, 204);

        }

        return back();

    }

}

4 Eloquent Modeli

User

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Passport\HasApiTokens;

use Spatie\MediaLibrary\HasMedia;

use Spatie\MediaLibrary\InteractsWithMedia;

class User extends Authenticatable implements HasMedia

{

    use HasApiTokens, HasFactory, Notifiable;

    use InteractsWithMedia;

*/\*\**

*\* The attributes that are mass assignable.*

*\**

*\* @var array*

*\*/*

    protected $fillable = [

        'name',

        'email',

        'password',

    ];

    protected $appends = [

        'avatar'

    ];

*/\*\**

*\* The attributes that should be hidden for arrays.*

*\**

*\* @var array*

*\*/*

    protected $hidden = [

        'password',

        'remember\_token',

    ];

*/\*\**

*\* The attributes that should be cast to native types.*

*\**

*\* @var array*

*\*/*

    protected $casts = [

        'email\_verified\_at' => 'datetime',

    ];

    public function getAvatarAttribute()

    {

        return *$this*->avatar();

    }

    public function avatar() {

        $mediaItems = *$this*->getMedia('profileImages');

        if(count($mediaItems) > 0) {

            $url = $mediaItems[0]->getFullUrl();

        } else {

            $url = env('APP\_URL') . '/public/storage/notsetProfile.png';

        }

        return $url;

    }

    public function posts() {

        return *$this*->hasMany(Post::class);

    }

    public function comments()

    {

        return *$this*->hasMany(Comment::class);

    }

    public function favorites()

    {

        return *$this*->belongsToMany(Post::class, 'favorites')->withTimestamps(); *//, 'user\_id', 'question\_id');*

    }

    public function likePosts()

    {

        return *$this*->morphedByMany(Post::class, 'likable');

    }

    public function likeComments()

    {

        return *$this*->morphedByMany(Comment::class, 'likable');

    }

    public function likePost(Post $post, $like)

    {

        $likePosts = *$this*->likePosts();

        return *$this*->\_like($likePosts, $post, $like);

    }

    public function likeComment(Comment $comment, $like)

    {

        $likeComments = *$this*->likeComments();

        return *$this*->\_like($likeComments, $comment, $like);

    }

    private function \_like($relationship, $model, $like)

    {

        if($relationship->where('likable\_id', $model->id)->exists()) {

            $relationship->updateExistingPivot($model, ['like' => $like]);

        } else {

            $relationship->attach($model, ['like' => $like]);

        }

        $model->load('likes');

        $unlike = (int) $model->unlike()->sum('like');

        $addLike = (int) $model->addLike()->sum('like');

        $model->likes\_count = $addLike + $unlike;

        $model->save();

        return $model->likes\_count;

    }

}

Post

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Support\Str;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Support\Facades\DB;

use Spatie\MediaLibrary\HasMedia;

use Spatie\MediaLibrary\InteractsWithMedia;

class Post extends Model implements HasMedia

{

    use HasFactory;

    use LikableTrait;

    use InteractsWithMedia;

    protected $fillable = [

        'name',

        'location',

        'description'

    ];

    protected $appends = ['trimmed\_name', 'created\_date', 'is\_favorited', 'favorites\_count', 'body\_html', 'is\_liked', 'images'];

    public function images() {

        $mediaItems = *$this*->getMedia();

        if(count($mediaItems) > 0) {

            $images = $mediaItems[0]->getUrl();

        } else {

            $images = env('APP\_URL') . '/public/storage/notset.jpg';

        }

        return $images;

    }

    public function getImagesAttribute() {

        return *$this*->images();

    }

    public function user()

    {

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

    }

    public function setNameAttribute($value)

    {

*$this*->attributes['name'] = $value;

*$this*->attributes['slug'] = *$this*->str\_slug($value);

    }

    private function str\_slug($text) {

*// replace non letter or digits by -*

        $text = preg\_replace('~[^\pL\d]+~u', '-', $text);

*// transliterate*

        $text = iconv('utf-8', 'us-ascii//TRANSLIT', $text);

*// remove unwanted characters*

        $text = preg\_replace('~[^-\w]+~', '', $text);

*// trim*

        $text = trim($text, '-');

*// remove duplicate -*

        $text = preg\_replace('~-+~', '-', $text);

*// lowercase*

        $text = strtolower($text);

        if (empty($text)) {

            return 'n-a';

        }

        return $text;

    }

    public function getUrlAdminAttribute()

    {

        return route('posts.show', *$this*->slug);

    }

    public function getUrlAttribute()

    {

        return route('posts.show', *$this*->slug);

    }

    public function getTrimmedNameAttribute() {

        $string = substr(*$this*->name, 0, 25);

        return $string . '...';

    }

    public function getCreatedDateAttribute()

    {

        return *$this*->created\_at->diffForHumans();

    }

    public function getStatusAttribute()

    {

        if (*$this*->comments\_count > 0)

        {

           if(*$this*->pinned\_comment\_id)

           {

                return "comment-pinned";

           }

        }

    }

    public function getBodyHtmlAttribute()

    {

        return strip\_tags(*$this*->bodyHtml());

    }

    public function comments() {

        return *$this*->hasMany(Comment::class)->orderBy('likes\_count', 'DESC');

    }

    public function pinComment(Comment $comment) {

*$this*->pinned\_comment\_id = $comment->id;

*$this*->save();

    }

    public function favorites() {

        return *$this*->belongsToMany(User::class, 'favorites')->withTimestamps();

    }

*//For more info kruks40 has how this works, fun to talk to him*

    public function isFavorited()

    {

        $favorites = DB::table('favorites')->where('user\_id', auth()->id())->where('post\_id', *$this*->id)->count() > 0;

        return $favorites;

*// return $this->favorites->where('user\_id', Auth::id())->count() > 0;*

    }

    public function getIsFavoritedAttribute()

    {

        return *$this*->isFavorited();

    }

    public function isLiked()

    {

        $likes = DB::table('likables')->where('user\_id', auth()->id())->where('likable\_id', *$this*->id)->where('likable\_type', 'App\Models\Post')->where('like', '1')->count() > 0;

        return $likes;

*// return $this->favorites->where('user\_id', Auth::id())->count() > 0;*

    }

    public function getIsLikedAttribute()

    {

        return *$this*->isLiked();

    }

    public function getFavoritesCountAttribute()

    {

        return *$this*->favorites->count();

    }

    public function getExcerptAttribute()

    {

        return *$this*->excerpt(250);

    }

    public function excerpt($len)

    {

        return Str::limit(strip\_tags(*$this*->bodyHtml()), $len);

    }

    private function bodyHtml()

    {

        return \Parsedown::instance()->text(*$this*->description);

    }

}

Comment

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Support\Facades\DB;

class Comment extends Model

{

    use HasFactory;

    use LikableTrait;

    protected $fillable = [

        'body',

        'user\_id'

    ];

    protected $appends = ['created\_date', 'body\_html', 'is\_pinned', 'is\_liked'];

    public function post() {

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

    }

    public function user() {

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

    }

    public function getBodyHtmlAttribute()

    {

        return filter\_var(\Parsedown::instance()->text(*$this*->body), *FILTER\_SANITIZE\_STRING*);

    }

    public function getCreatedDateAttribute()

    {

        return *$this*->created\_at->diffForHumans();

    }

    public function getStatusAttribute()

    {

        return *$this*->isPinned() ? 'comment-pinned' : '';

    }

    public function getIsPinnedAttribute() {

        return *$this*->isPinned();

    }

    public function isPinned() {

        return *$this*->id == *$this*->post->pinned\_comment\_id;

    }

    public function isLiked()

    {

        $likes = DB::table('likables')->where('user\_id', auth()->id())->where('likable\_id', *$this*->id)->where('likable\_type', 'App\Models\Comment')->where('like', '1')->count() > 0;

        return $likes;

*// return $this->favorites->where('user\_id', Auth::id())->count() > 0;*

    }

    public function getIsLikedAttribute()

    {

        return *$this*->isLiked();

    }

    public static function boot()

    {

        parent::boot();

        static::created(function($comment) {

            $comment->post->increment('comments\_count');

        });

        static::deleted(function($comment){

            $comment->post->decrement('comments\_count');

        });

    }

}

Likable Trait

<?php

namespace App\Models;

trait LikableTrait

{

  public function likes()

  {

      return *$this*->morphToMany(User::class, 'likable');

  }

  public function addLike() {

      return *$this*->likes()->wherePivot('like', 1);

  }

  public function unlike() {

      return *$this*->likes()->wherePivot('like', 0);

  }

}