**Laravel – Adatbázis kapcsolatok**

Feladat: adott egy POST és egy COMMENT tábla. Az egyes postok alá kommenteket lehet beszúrni.

<https://laravel.com/docs/8.x/eloquent-relationships>

POST tábla:

     \*/

    public function up()

    {

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

            $table->id();

            $table->string('title');

            $table->text('description');

            $table->timestamps();

        });

    }

COMMENT tábla:

   public function up()

    {

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

            $table->id();

            $table->foreignID('post\_id')->constrained()->onUpdate('cascade')->onDelete('cascade')->nullable();

            //post\_id : a Post tábla id mezője

            //a parent megváltoztatása esetén a gyerekelemeken is megváltoznak a parent elem tulajdonságai

            //a parent Post törlése esetén a child elemek (kommentek) is törlődnek

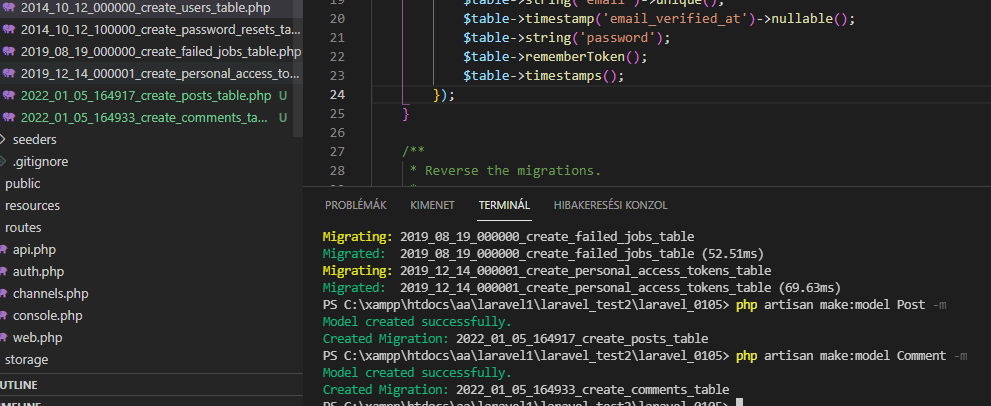
            //engedjük meg a NULL értéket (ha a parent Post törlődik, akkor a kommentek se "lógjanak a levegőben")

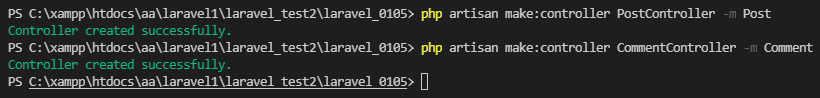
            $table->text('text');

            $table->timestamps();

        });

    }





Amiket le szeretnénk majd kérdezni:

* Összes Post ill. Comment
* Adott id-jú Post ill. Comment

Útvonalak:

Route::get('/api/posts', [PostController::class, 'allposts']);

Route::get('/api/post/{id}', [PostController::class, 'showpost']);

Route::get('/api/comments', [CommentController::class, 'allcomments']);

Route::get('/api/comment/{id}', [CommentController::class, 'showcomment']);

(A meghívott függvényeket most nem részletezzük.)

Adott posthoz tartozó commenteket fogjuk lekérdezni:

//Adott posthoz kikérünk commenteket:

Route::get('/api/post/{id}/comments', [PostController::class, 'comments']);

Először azonban beállítjuk az adatbázis-kapcsolatokat:

Post model 🡪 egy Post példányhoz több Comment példány tartozik (POST : COMMENT = 1:N)

class Post extends Model

{

    use HasFactory;

    public function comments()

    {

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

    }

}

PostController 🡪 adott posthoz tartozó commentek kigyűjtése a $comments változóba:

 public function comments($id)

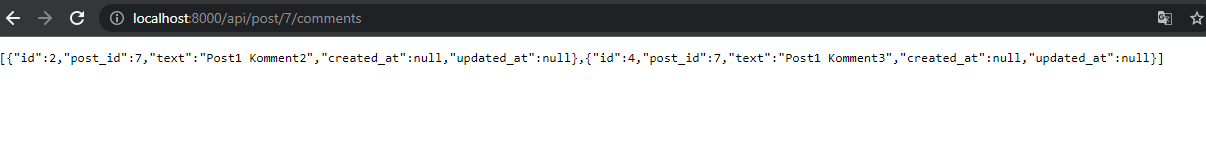
    {

        $post=Post::find($id); //a modelben lévő Post class egy példánya

        $comments = $post->comments;

        return $comments;

    }



Adott commenthez tart. post megkeresése:

Route::get('/api/comment/{id}/post', [CommentController::class, 'post']);

Comment model 🡪 egy adott Comment példány egy Post modelhez tartozik:

class Comment extends Model

{

    use HasFactory;

    public function post()

    {

        return $this->belongsTo(Post::class); //a kommenthez tart. postot adja vissza

    }

}

CommentController:

   public function post($id)

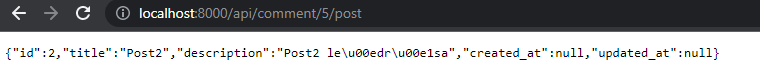
    {

        $comment= Comment::find($id);

        $post = $comment->post; //meghívjuk a Comment példány post függvényét

        return $post;

    }



Adott post kilistázása a hozzá tart. kommentekkel együtt:

Route::get('/api/post/{id}/extended', [PostController::class, 'extended']);

PostController:

  public function extended($id)

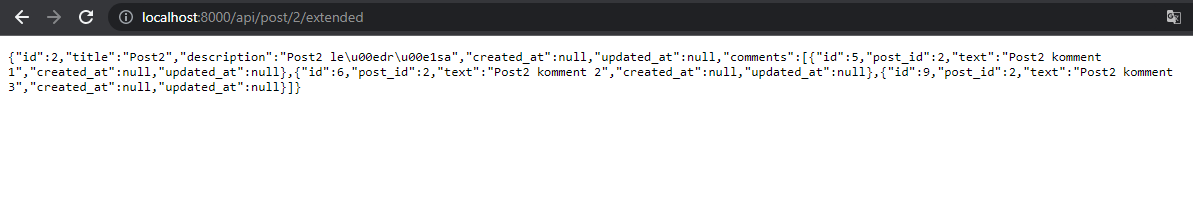
    {

        $post=Post::find($id); //a modelben lévő Post class egy példánya

        /\*$post->comments = \*/$post->comments;

        return $post;

    }



Fordítva:

Route::get('/api/comment/{id}/extended', [CommentController::class, 'extended']);

CommentController:

public function extended($id)

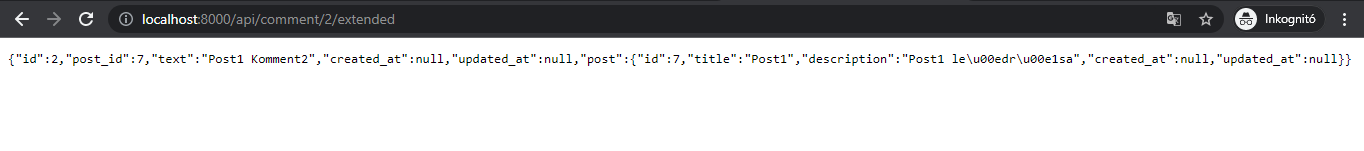
    {

        $comment= Comment::find($id);

        $comment->post;

        return $comment;

    }



Query-k

<https://laravel.com/docs/8.x/queries>

Végpont, ahol átadunk egy kereső stringet, és megnézzük, hogy az mely postok description-jében található meg:

Route::get('/api/post/search', [PostController::class, 'search']);

Felállítunk egy query-t. Megnézzük, hogy szerepel-e a megadott string a Post description-jében:

<http://localhost:8000/api/post/search/?q=a>

PostController:

  public function search(Request $request)

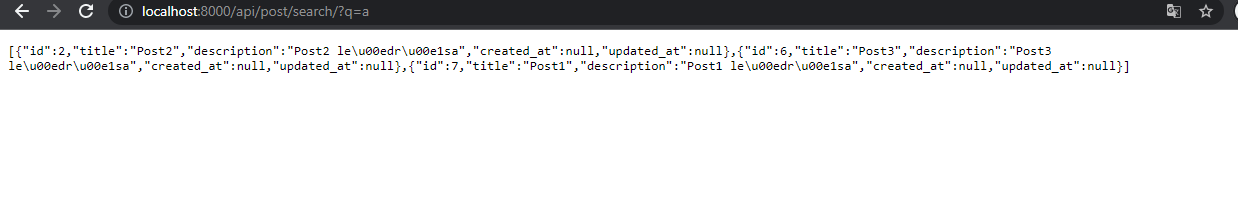
    {

        $queryString= $request->q;//$request->query('q');

        $posts= Post::where('description','like','%'.$queryString.'%')->get();

        return $posts;

    }



public function search(Request $request)

    {

        $queryString= $request->q;//$request->query('q');

        $posts= Post::where('description','like','%'.$queryString.'%')->orderBy('title')->get();

        return $posts;

    }

->where(’role’,’=’,’admin’)->where(’login\_date’,’>’,’now()-1day’)

Több a több kapcsolathoz példa:

<https://medium.com/@afrazahmad090/laravel-many-to-many-relationship-explained-822b554c1973>

<https://appdividend.com/2018/05/17/laravel-many-to-many-relationship-example/>