

LFTS - Working With Database

17. Environment Files and Database Connections

este paso es basicamente lo ultimo que hicimos en la ultima clase, configuracion de la basede datos y migracion tablas

Configuracion de .env

Configuramos el espacio que tiene preasignado el .env con los datos de nuestra base de datos

```
DB_CONNECTION=mysql
DB_HOST=192.168.146.11
DB_PORT=3306
DB_DATABASE=lfts
DB_USERNAME=laravel
DB_PASSWORD=secret
```

creamos el esquema la base de datos

```
-> ;
+-----+
| Database |
+-----+
| information_schema |
| lfts |
+-----+
2 rows in set (0.001 sec)
```

Migracion de la tablas de artisan

```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (86.31ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (10.49ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (12.10ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (22.69ms)
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$
```

18. Migrations: The Absolute Basics

Nos enseñan conceptos basicos de las migraciones la mayoria de ellos en torno al:

```
'''
```

```
php artisan migrate:fresh
```

```
'''
```

Esto basicamente no importa los cambios que podamos hacer ya sea en el archivo de

compilacion de laravel o desde el administrador de base de datos siempre restablece todo a la migracion original

Tambien nos muestra un pequeño detalle pero que me parecio muy conveniente que es que basicamente si nos encontramos en el ambiente de produccion antes de correr este codigo nos muestra una alerta, avisandonos que estamos en el entorno de produccion.

tambien nos enseña cosas como

```
'''
php artisan migrate:rollback
'''
```

```
migrate
migrate:fresh      Drop all tables and re-run all migrations
migrate:install    Create the migration repository
migrate:refresh    Reset and re-run all migrations
migrate:reset      Rollback all database migrations
migrate:rollback   Rollback the last database migration
migrate:status     Show the status of each migration
model
```

19. Eloquent and the Active Record Pattern

Utilizando el php artisan tinker creamos y visualizamos usuarios

Creacion de usuario

Utilizando tinker es posible crear usuarios en la base de datos

```
> $user = new User;
[!] Aliasing 'User' to 'App\Models\User' for this Tinker session.
= App\Models\User {#6117}

> $user->name = 'Cristiam';
= "Cristiam"

> $user->email = 'Cristiam@gmail.com';
= "Cristiam@gmail.com"

> $user->password = bcrypt('!password');
= "$2y$10$IVnZE35J/TJCa5PfGFTQw0cAJ3d0Eb5uQ6MbGNfOX26sLx1.eAxwG"

PARSE ERROR PHP Parse error: Syntax error, unexpected ';', expecting
or/psy/psysh/src/Exception/ParseException.php on line 38.

> $user->password = bcrypt('!password');
= "$2y$10$IVnZE35J/TJCa5PfGFTQw0cAJ3d0Eb5uQ6MbGNfOX26sLx1.eAxwG"

> $user->save();
= true
```

Visualizar usuarios

Tambien es posible ver los usuarios que hemos creado

```
> User::all()
= Illuminate\Database\Eloquent\Collection {#7080
  all: [
    App\Models\User {#7078
      id: 1,
      name: "Cristiam",
      email: "Cristiam@gmail.com",
      email_verified_at: null,
      #password: "$2y$10$IVnZE35J/TJCa5PfGfTQwOcAJ3d0Eb5uQ6MbGNfOX26sLx1.eAxwG",
      #remember_token: null,
      created_at: "2023-10-11 15:33:57",
      updated_at: "2023-10-11 15:33:57",
    },
    App\Models\User {#7079
      id: 2,
      name: "Luis",
      email: "luis@gmail.com",
      email_verified_at: null,
      #password: "$2y$10$/Zm7uEtfmXjpZaE3tgh.9.AwyGxqy6tvZhefhmbfHmu3eif04PbK0",
      #remember_token: null,
      created_at: "2023-10-11 15:39:18",
      updated_at: "2023-10-11 15:39:18",
    },
  ],
}
```

Y obviamente estos usuarios no son exclusivos del php artisan tinker, esto afecta la base de datos y por ende los podemos ver desde nuestro administrador de base de datos

id	name	email	email_verified_at	password	remember_token
1	Cristiam	Cristiam@gmail.com	NULL	\$2y\$10\$IVnZE35J/TJCa5PfGfTQwOcAJ3d0Eb5...	NULL
2	Luis	luis@gmail.com	NULL	\$2y\$10\$/Zm7uEtfmXjpZaE3tgh.9.AwyGxqy6tv...	NULL

20. Make a Post Model and Migration

Ahora vamos a crear la tabla y el modelo post para almacenar post

Creacion de tabla y modelo

vamos a crear la tabla y el modelo de post para esto vamos a utilizar el siguiente comando

```
php artisan make:migration create_posts_table
php artisan make:model Post
```

Asignar columnas a la tabla

Vamos a asignar unas columnas a la tabla que acabamos de crear

id	name	email	email_verified_at	password	remember_token
1	Cristiam	Cristiam@gmail.com	NULL	\$2y\$10\$IVnZE35J/TJCa5PfGfTQwOcAJ3d0Eb5...	NULL
2	Luis	luis@gmail.com	NULL	\$2y\$10\$/Zm7uEtfmXjpZaE3tgh.9.AwyGxqy6tv...	NULL

despues de esto ocupamos migrar estas columnas con el comando

```
php artisan migrate
```

Creacion de datos para Posts

De la misma manera en la que creamos los usuarios en el ep. 19 vamos a crear datos para Post, esto lo hacemos 2 veces, simplemente para tener 2 datos de prueba

```
> $post = new App\Models\Post  
= App\Models\Post {#6927}  
  
> $post-> title = 'My Second Post'  
= "My Second Post"  
  
> $post-> excerpt = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.';  
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
  
> $post-> body = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque  
ec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas  
ravidia orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio  
eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.';  
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Ph  
asellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id  
posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis n  
isl. Aliquam ultrices finibus metus a consectetur."  
  
> $post-> save();  
= true
```

Resultado

```
> $post::all()
= Illuminate\Database\Eloquent\Collection {#7075
  all: [
    App\Models\Post {#7076
      id: 1,
      title: "My First Post",
      excerpt: "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
      body: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.",
      created_at: "2023-10-11 16:07:48",
      updated_at: "2023-10-11 16:07:48",
      published_at: null,
    },
    App\Models\Post {#7077
      id: 2,
      title: "My Second Post",
      excerpt: "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
      body: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.",
      created_at: "2023-10-11 16:12:15",
      updated_at: "2023-10-11 16:12:15",
      published_at: null,
    },
  ],
}
```

Configuración de código para que funcione con base de datos

Ahora nuestro programa en vez de funcionar con archivos, funciona con base de datos y por lo mismo hay que hacer unos pequeños cambios que básicamente consisten en cambiar `$slug` por `$id`

```
Route::get('posts/{post}', function ($id){
    return view('post',[
        'post' => Post::findOrFail($id)
    ]);
});
```

```
<h1>
  <a href="/posts/{{ $post->id }}">
    {{ $post->title }}
  </a>
```

21. Eloquent Updates and HTML Escaping

En este módulo le damos un poco más de formato a el HTML y también nos muestran los cuidados que hay que tener a la hora de "Escape" cierta información, para evitar errores, o problemas de seguridad

formato extra para el HTML

le agregamos el tag de `<p>` a el body para que la informacion que se muestra a continuacion no este pegado, en este caso el boton de volver a la pagina principal

```
> $post = App\Models\Post::find(2);
= App\Models\Post {#6455
  id: 2,
  title: "My Second Post",
  excerpt: "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
  body: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.",
  created_at: "2023-10-11 16:12:15",
  updated_at: "2023-10-11 16:12:15",
  published_at: null,
}

> $post->body;
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur."

> $post->body = '<p>' . $post->body . '</p>';
= "<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.</p>"

> $post-> save();
= true

> $post
= App\Models\Post {#6455
  id: 2,
  title: "My Second Post",
  excerpt: "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
  body: "<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.</p>",
  created_at: "2023-10-11 16:12:15",
  updated_at: "2023-10-11 17:02:37",
  published_at: null,
}
```

HTML Escaping info

Tambien se nos explica que hay que tener cuidado a la hora de hacer esto porque por ejemplo en el titulo se podria hacer esto, pero al hacer esto podemos causar un problema de seguridad ya que le damos la posibilidad de generar un error, por utilizar el mismo ejemplo meter codigo JS en el titulo, entonces cada vez que se cargue el post ejecutaria el JS, lo cual no es de nuestro interes



22. 3 Ways to Mitigate Mass Assignment Vulnerabilities

Nos enseñan 3 formas de migrar datos utilizando el php artisan tinker

Una de estas formas ya la conocíamos, es como habíamos estado introduciendo datos hasta ahora

```
> $post = new App\Models\Post
= App\Models\Post {#6927}

> $post-> title = 'My Second Post'
= "My Second Post"

> $post-> excerpt = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.';
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit."

> $post-> body = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.';
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur."

> $post-> save();
= true
```

Otra forma es metiendo todo en una misma línea

```
> $post = new App\Models\Post
= App\Models\Post {#6927}

> $post-> title = 'My Second Post'
= "My Second Post"

> $post-> excerpt = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.';
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit."

> $post-> body = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.';
= "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur."

> $post-> save();
= true
```

Pero para esto es importante modificar nuestro modelo de Post

```
use Illuminate\Database\Eloquent\Factories\HasFactory;

protected $guarded = [];
// protected $fillable = ['title', 'excerpt', 'body', 'id'];
```

Tambien podemos actualizar informacion

```
> $post = Post::first();
= App\Models\Post {#7076
  id: 1,
  title: "My First Post",
  excerpt: "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
  body: "<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.</p>",
  created_at: "2023-10-11 16:07:48",
  updated_at: "2023-10-11 17:01:57",
  published_at: null,
}

> $post-> excerpt = 'changed'
= "changed"

> $post->save();
= true
```

Y con el siguiente comando lo volvemos a su estado original

```
> $post->fresh();
= App\Models\Post {#6124}
```

O con un metodo mas directo con el cual podemos actualizar

```
> $post-> update(['excerpt' => 'Changedd']);
= true
```

23. Route Model Binding

Nos enseñan a utilizar las wildcart de otra mananera y obtener un URL mas completo

Migracion de la tabla Posts

Para esto tenemos que agregarle un parametro a la tabla de posts para esto nos vamos a la carpeta de migracion y lo agregamos

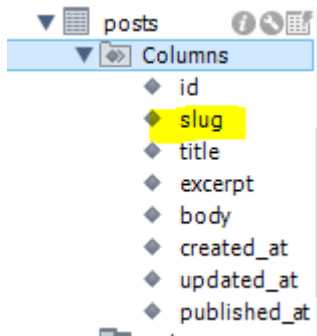
```
{
  Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->string('slug')->unique();
    $table->string('title');
    $table->text('excerpt');
    $table->text('body');
    $table->timestamps();
    $table->timestamp('published_at')->nullable();
  });
}
```

Pero para que se guarde hay que aplicar la migracion, pero vamos a crear todo desde 0 con el parametro recién creado

```
```
php artisan migration:fresh
```
```

```
vagrant@webserver:~/sites/lfts.isw811.xyz$ php artisan migrate:fresh
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (164.65ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (14.53ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (17.12ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (29.73ms)
Migrating: 2023_10_11_155212_create_posts_table
Migrated: 2023_10_11_155212_create_posts_table (11.66ms)
vagrant@webserver:~/sites/lfts.isw811.xyz$
```

Resultado



Y despues simplemente en nuestro archivo de rutas modificamos la funcion

```
Route::get('posts/{post:slug}', function (Post $post){  
    return view('post', [  
        'post' => $post  
    ]);  
});
```

Tambien esta la posibilidad de hacerlo de la siguiente manera

archivo de rutas

```
Route::get('posts/{post}', function (Post $post){  
    return view('post', [  
        'post' => $post  
    ]);  
});
```

Modelo Post

```
public function getRouteKeyName()  
{  
    return 'slug';  
}
```

Resultado de ULR



My First Post

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque nec est orci. Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egestas gravida orci id posuere. Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sagittis nisl. Aliquam ultrices finibus metus a consectetur.

[Go Back](#)

24. Your First Eloquent Relationship

Asiganamos una categoria a cada post

Creacion de las categorias

Para crear las categorias primero tenemos que crear la tabla y Modelo de categorias, para esto utilizamos el artisan

```
vagrant@webserver:~/sites/lfts.isw811.xyz$ php artisan make:model Category -m
Model created successfully.
Created Migration: 2023_10_16_233447_create_categories_table
vagrant@webserver:~/sites/lfts.isw811.xyz$ php artisan migrate:fresh
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (23.15ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (9.60ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (10.19ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (16.75ms)
Migrating: 2023_10_11_155212_create_posts_table
Migrated: 2023_10_11_155212_create_posts_table (13.32ms)
Migrating: 2023_10_16_233447_create_categories_table
Migrated: 2023_10_16_233447_create_categories_table (4.80ms)
vagrant@webserver:~/sites/lfts.isw811.xyz$ php artisan tinker
Psy Shell v0.11.21 (PHP 8.2.7 - cli) by Justin Hileman
```

Ahora creamos las categorias

```

> use App\Models\Category;
> $c = new Category;
= App\Models\Category {#6115}

> $c->name = 'Personal';
= "Personal"

> $c->slug = 'personal';
= "personal"

> $c->save();
= true

> $c = new Category;
= App\Models\Category {#6861}

> $c->name = 'Work';
= "Work"

> $c->slug = 'work';
= "work"

> $c->save();
= true

> $c = new Category;
= App\Models\Category {#6116}

> $c->name = 'Hobbies';
= "Hobbies"

> $c->slug = 'hobbies';
= "hobbies"

> $c->save();

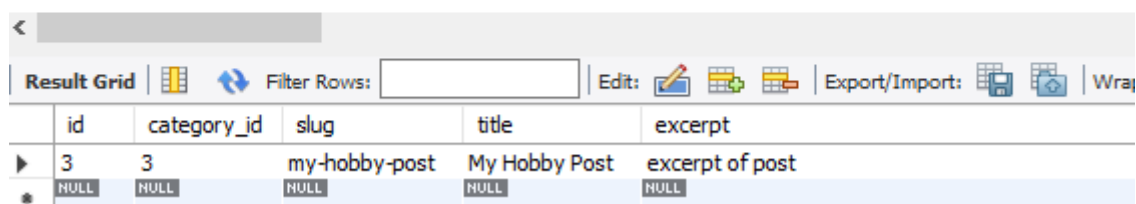
```

Utilizando nuestro administrador de base de datos ahora podemos ver la relacion que establecimos, obviamente a la hora de crear los post nuevos hay que agregar una nueva columna que va a hacer una llave foranea que va a ser nuestra manera de establecer la conexion entra las tablas

```

1 • select * from posts where category_id = 3;
2
3
4
5

```



	id	category_id	slug	title	excerpt
▶	3	3	my-hobby-post	My Hobby Post	excerpt of post
*	NULL	NULL	NULL	NULL	NULL

Creacion de funcion relacional

Si probamos en el tinkers la relacion

```

...

```

```

$post->category

```

```

...

```

nuestra null, para arreglar esto vamos a ir a el Modelo Post

```
public function category()
{
    return $this->belongsTo(Category::class);
}
```

Testeo de Relaciones

Con el php artisan tinker probamos nuestras relaciones

```
> $post = App\Models\Post::first();
= App\Models\Post {#6817
  id: 1,
  category_id: 1,
  slug: "my-family-post",
  title: "My Family Post",
  excerpt: "excerpt of post",
  body: "<p>Lorem ipsum dolor sit amet, consectetur adipiscing e
hasellus semper a tellus ut pulvinar. Sed non urna sem. Cras egest
Nullam scelerisque molestie pretium. Fusce nec pretium odio, eu sa
s finibus metus a consectetur.</p>",
  created_at: "2023-10-11 16:07:48",
  updated_at: "2023-10-16 22:35:10",
  published_at: null,
}

>
> $post->category();
= Illuminate\Database\Eloquent\Relations\BelongsTo {#6113}

> $post->category;
= App\Models\Category {#6166
  id: 1,
  name: "Personal",
  slug: "personal",
  created_at: "2023-10-16 23:41:53",
  updated_at: "2023-10-16 23:41:53",
}
```

Modificacion de las paginas

Modificamos la pagina principal como la pagina de cada post para que se muestre la categoria

```
<x-layout>
  @foreach($posts as $post)
    <article class="{{ $loop->even ? 'foobar' : '' }}">
      <h1>
        <a href="/posts/{{ $post->slug }}">
          {{ $post->title }}
        </a>
      </h1>
      <p>
        <a href="#">{{ $post->category->name }}</a>
      </p>
      <div>
        {{ $post->excerpt }}
      </div>
    </article>
  @endforeach
</x-layout>
```

```
<x-layout>
  <article>
    <h1>{{ $post->title }}</h1>
    <p>
      <a href="#">{{ $post->category->name }}</a>
    </p>
    <div>
      {!! $post->body !!}
    </div>
  </article>
  <a href="/">Go Back</a>
</x-layout>
```

Resultado final

My Family Post

Personal

excerpt of post

My Work Post

Work

excerpt of post

My Hobby Post

Hobbies

excerpt of post

My Family Post

Personal

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque
Phasellus semper a tellus ut pulvinar. Sed non urna sem. Cras
id posuere. Nullam scelerisque molestie pretium. Fusce nec
nisl. Aliquam ultrices finibus metus a consectetur.

Go Back

25. Show All Posts Associated With a Category

Ahora vamos a crear el la ruta para acceder a los post por categoria

Creacion de ruta

Primero en nuestro archivo de rutas vamos a crear el "path" hacia lo que ocupamos

```
Route::get('categories/{category:slug}', function (Category $category){  
    return view('posts', [  
        'posts' => $category->posts  
    ]);  
});
```

Creacion de funcion para cargar los post

Ahora vamos a la funcion para que se cargen todos los post que pertenecen a una categoria

```
public function posts()  
{  
    return $this->hasMany(Post::class);  
}
```

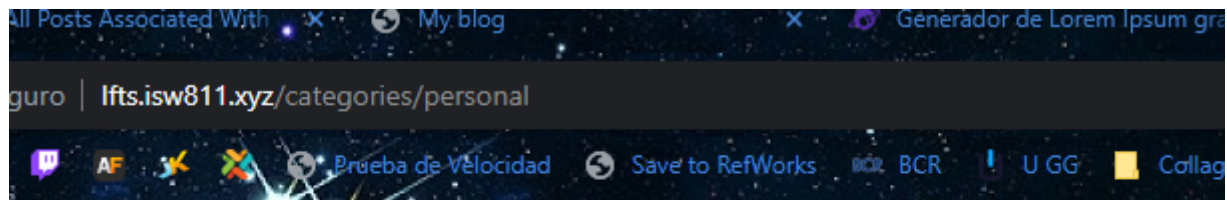
Y Como ultimo le asignamos la ruta en la vista

Le asignamos la ruta en la vista para que sea cargue la pagina que buscamos

```
<p>  
    <a href="/categories/{ {{ $post->category->slug }}}">{{ $post->category->name }}</a>  
</p>
```

```
<p>  
    <a href="/categories/{ {{ $post->category->slug }}}">{{ $post->category->name }}</a>  
</p>
```

Resultado



My Family Post

Personal

excerpt of post

26. Clockwork, and the N+1 Problem

Basicamente nos explican que hay un problema con las solicitudes, y que que estan ejecutando varias queries de manera innecesaria

Instalar clockwork

Instalamos clockwork junto con su extension para ver los queries solicitados

```
agrant@webserver:~/sites/lfts.isw811.xyz$ composer require itsgoingd/clockwork
./composer.json has been updated
Running composer update itsgoingd/clockwork
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking itsgoingd/clockwork (v5.1.12)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading itsgoingd/clockwork (v5.1.12)
  - Installing itsgoingd/clockwork (v5.1.12): Extracting archive
Package fruitcake/laravel-cors is abandoned, you should avoid using it. No replacement was suggested.
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mail instead.
Generating optimized autoload files
Illuminate\Foundation\ComposerScripts::postAutoloadDump
@php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fruitcake/laravel-cors
Discovered Package: itsgoingd/clockwork
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
9 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
@php artisan vendor:publish --tag=laravel-assets --ansi --force
No publishable resources for tag [laravel-assets].
Publishing complete.
No security vulnerability advisories found.
Using version ^5.1 for itsgoingd/clockwork
```

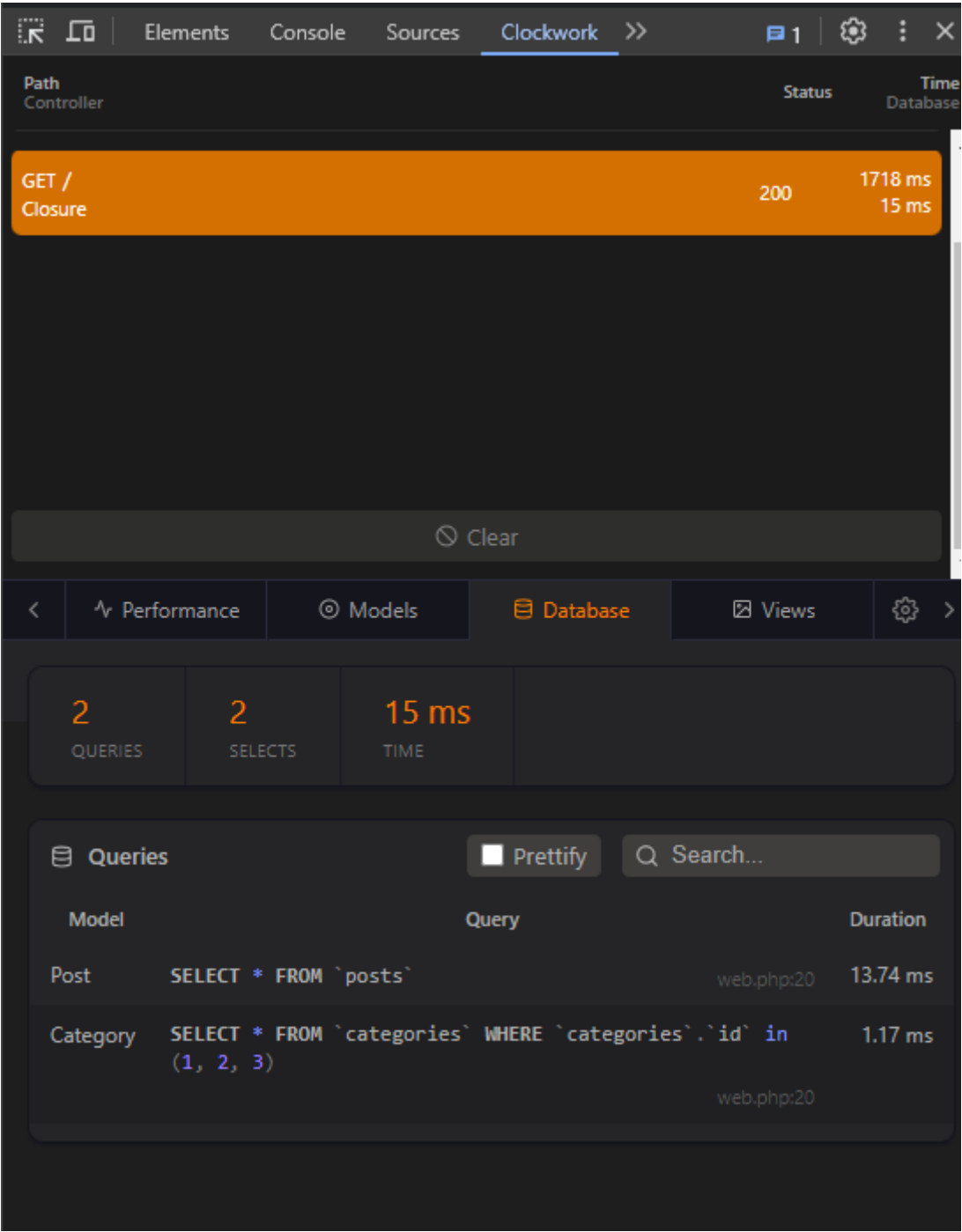
Solucion

Hacemos una pequeña modificacion en el archivo de rutas

```
Route::get('/', function () {
    return view('posts', [
        'posts' => Post::with('category')->get()
    ]);
});
```

Resultado

Pasamos de tener 4 querys para cargar la vista a solo tener 2



27. Database Seeding Saves Time

Nos enseñan a crear un seeder para no tener que estar creando datos de prueba cada vez que hacemos una migración

Modificación de tabla posts

Ocupamos agregarle una llave foránea a posts para que tenga una relación con el usuario que se creó

```

Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id');
    $table->foreignId('category_id');
    $table->string('slug')->unique();
    $table->string('title');
    $table->text('excerpt');
    $table->text('body');
    $table->timestamps();
    $table->timestamp('published_at')->nullable();
});

```

Creacion del seeder

Utilizamos el archivo seeder que esta por defecto en la carpeta de base de datos, aqui le vamos a meter todos los datos que queremos que el seeder cree

```

User::truncate();
Post::truncate();
Category::truncate();

$user = User::factory()->create();

$personal = Category::create([
    'name' => 'Personal',
    'slug' => 'personal'
]);

$work = Category::create([
    'name' => 'Work',
    'slug' => 'work'
]);

$family = Category::create([
    'name' => 'Family',
    'slug' => 'family'
]);

Post::create([
    'user_id' => $user->id,
    'category_id' => $family->id,
    'title' => 'My Family Post',
    'slug' => 'my-family-post',
    'excerpt' => '<p>excerpt for this post</p>',
    'body' => '<p>lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nisl tincidunt eget nullam non.</p>'
]);

Post::create([
    'user_id' => $user->id,
    'category_id' => $work->id,
    'title' => 'My Work Post',
    'slug' => 'my-work-post',
    'excerpt' => '<p>excerpt for this post</p>',
    'body' => '<p>lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nisl tincidunt eget nullam non.</p>'
]);

```

Ahora cada vez que migramos la base de datos le podemos agregar "--seed" para que ademas de remigrar las tablas ejecute el seeder y cree los datos por defecto

```
php artisan migrate::fresh --seed
```

Modificacion de modelos

```
public function posts()  
{  
    return $this->hasMany(Post::class);  
}
```

```
public function user()  
{  
    return $this->belongsTo(User::class);  
}
```

Modificacion de vistas

```
<p>  
By <a href="#">{{ $post->user->name }}</a> in <a href="/categories/{{ $post->category->slug }}">{{ $post->category->name }}</a>  
</p>
```

```
<p>  
By <a href="#">{{ $post->user->name }}</a> in <a href="/categories/{{ $post->category->slug }}">{{ $post->category->name }}</a>  
</p>
```

28. Turbo Boost With Factories

Creacion de datos aleatorios

Creacion de factories

Utilizando el artizan creamos los archivos de factory

```
Database seeding completed successfully.  
vagrant@webserver:~/sites/lfts.isw811.xyz$ php artisan make:factory PostFactory  
Factory created successfully.  
vagrant@webserver:~/sites/lfts.isw811.xyz$ php artisan make:factory CategoryFactory  
Factory created successfully.
```

Asignacion de datos a los factories

```
public function definition()  
{  
    return [  
        'user_id' => User::factory(),  
        'category_id' => Category::factory(),  
        'title' => $this->faker->sentence,  
        'slug' => $this->faker->slug,  
        'excerpt' => $this->faker->sentence,  
        'body' => $this->faker->paragraph  
    ];  
}
```

```
public function definition()  
{  
    return [  
        'name' => $this->faker->sentence,  
        'slug' => $this->faker->slug,  
    ];  
}
```

Modificacion del Seeder

Ahora los datos del seeder se pueden fabricar de manera aleatoria y cuantos nosotros queramos, en este caso vamos a crear 5 posts, que todos provengan del mismo usuario

```
public function run()  
{  
    {  
        $user = User::factory()->create([  
            'name' => 'Lesley Kautzer'  
        ]);  
  
        Post::factory(5)->create([  
            'user_id' => $user->id  
        ]);  
    }  
}
```

Resultado

[Ducimus cumque voluptas ut enim amet.](#)

By [Lesley Kautzer](#) in [Eligendi sapiente quod et consequatur reiciendis.](#)

Nobis ipsa ex facere deserunt repellendus et.

Et aut consequuntur ut ut totam.

By [Lesley Kautzer](#) in [Sed cumque at nulla voluptatem ut nam.](#)

Ut voluptatem possimus eos sed sint temporibus iure molestiae.

Soluta soluta doloribus quos adipisci aut nihil magnam et.

By [Lesley Kautzer](#) in [Eos enim voluptas eaque nemo.](#)

Modi aspernatur et labore sunt.

Ipsa rerum facere maxime vel.

By [Lesley Kautzer](#) in [Delectus amet accusantium sit facilis deleniti sed et.](#)

Voluptas velit sed beatae magni consequatur ipsum fugiat.

Cumque dolorum repudiandae quia.

By [Lesley Kautzer](#) in [Et et itaque ad nulla molestiae laborum expedita.](#)

Quas aut pariatur non cumque cupiditate architecto.

Vamos a crear la rutas para poder acceder a todos los post de un mismo autor

Creacion de la ruta

```
Route::get('authors/{author:username}', function (User $author){
    return view('posts',[
        'posts' => $author->posts
    ]);
});
```

Modificacion de vistas

```
<p>
    By <a href="/authors/{{ $post->author->username }}">{{ $post->author->name }}</a> in <a href="/categories/{{ $post->category->name }}">{{ $post->category->name }}</a>
</p>
```

```
<p>
    By <a href="/authors/{{ $post->author->username }}">{{ $post->author->name }}</a> in <a href="/categories/{{ $post->category->name }}">{{ $post->category->name }}</a>
</p>
```

Resultado



30. Eager Load Relationships on an Existing Model

Basicamente nos explican que al cargar los posts por categorias y por autor incurrimos en un pequeño sobre cargo de queries, que en un programa mas grande puede ser un problema, para esto vamos a incluir en cada post los datos de la categoria y del autor y de esta manera pasamos de 24 queries a solo 4

```
protected $with = ['category', 'author'];
```