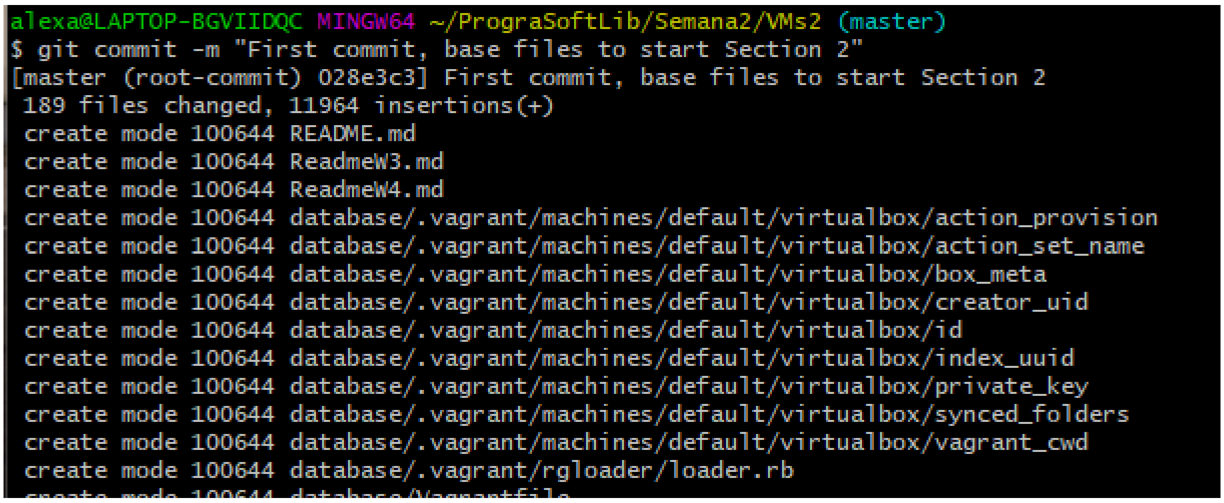
Laravel From Scratch

My first commit

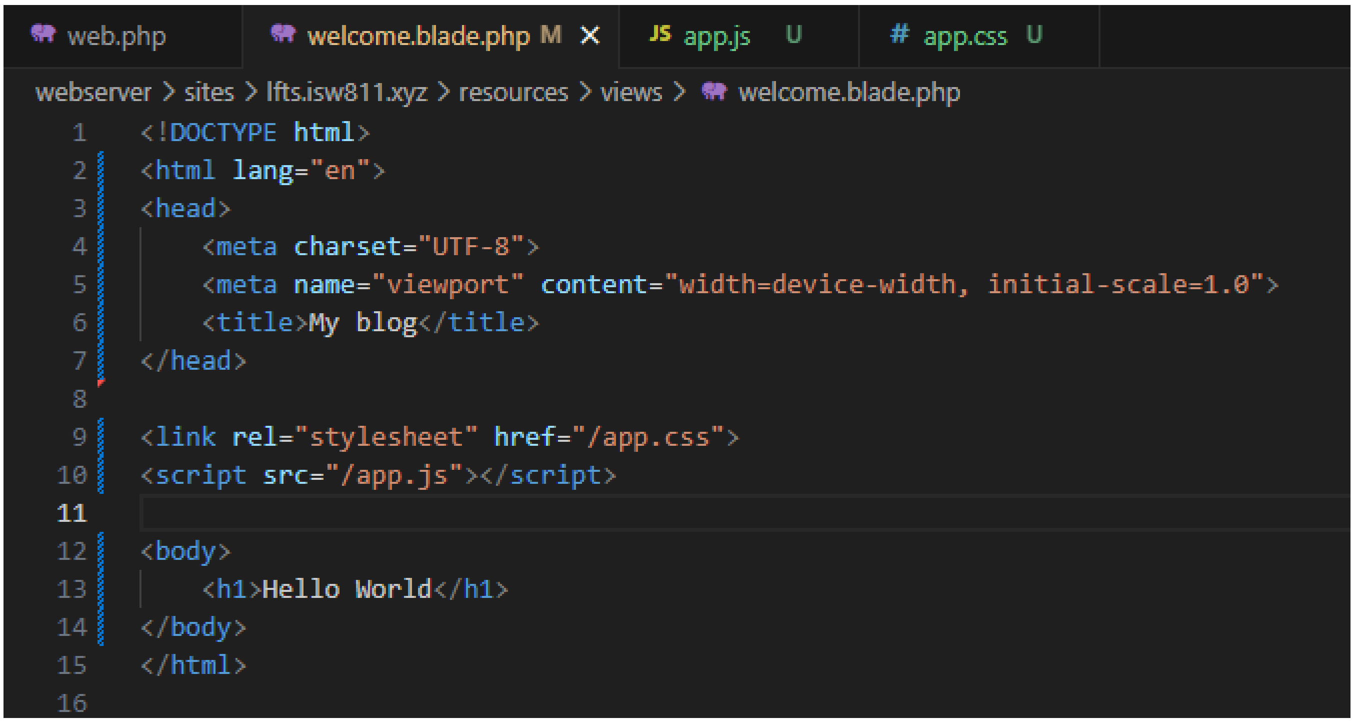
Archivos base para empezar el curso LFTS a partir de la Section 2



```
alexa@LAPTOP-BGVIIDQC MINGW64 ~/PrograSoftLib/Semana2/VMs2 (master)
$ git commit -m "First commit, base files to start Section 2"
[master (root-commit) 028e3c3] First commit, base files to start Section 2
 189 files changed, 11964 insertions(+)
 create mode 100644 README.md
 create mode 100644 ReadmeW3.md
 create mode 100644 ReadmeW4.md
 create mode 100644 database/.vagrant/machines/default/virtualbox/action_provision
 create mode 100644 database/.vagrant/machines/default/virtualbox/action_set_name
 create mode 100644 database/.vagrant/machines/default/virtualbox/box_meta
 create mode 100644 database/.vagrant/machines/default/virtualbox/creator_uid
 create mode 100644 database/.vagrant/machines/default/virtualbox/id
 create mode 100644 database/.vagrant/machines/default/virtualbox/index_uuid
 create mode 100644 database/.vagrant/machines/default/virtualbox/private_key
 create mode 100644 database/.vagrant/machines/default/virtualbox/synced_folders
 create mode 100644 database/.vagrant/machines/default/virtualbox/vagrant_cwd
 create mode 100644 database/.vagrant/rgloader/loader.rb
```

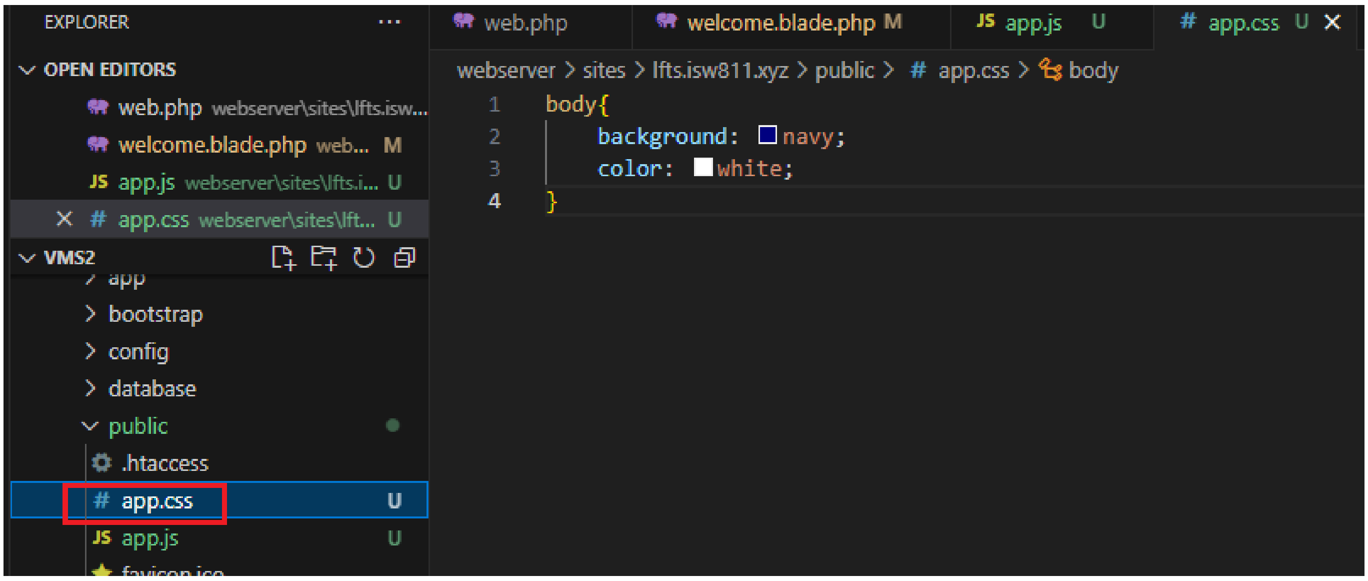
Include CSS and Javascript

Modificamos el welcom.blade.php que cargaba la vista basica de laravel creando nuestro propio código


web.php
welcome.blade.php M
app.js U
app.css U
webserver > sites > lfts.isw811.xyz > resources > views > welcome.blade.php

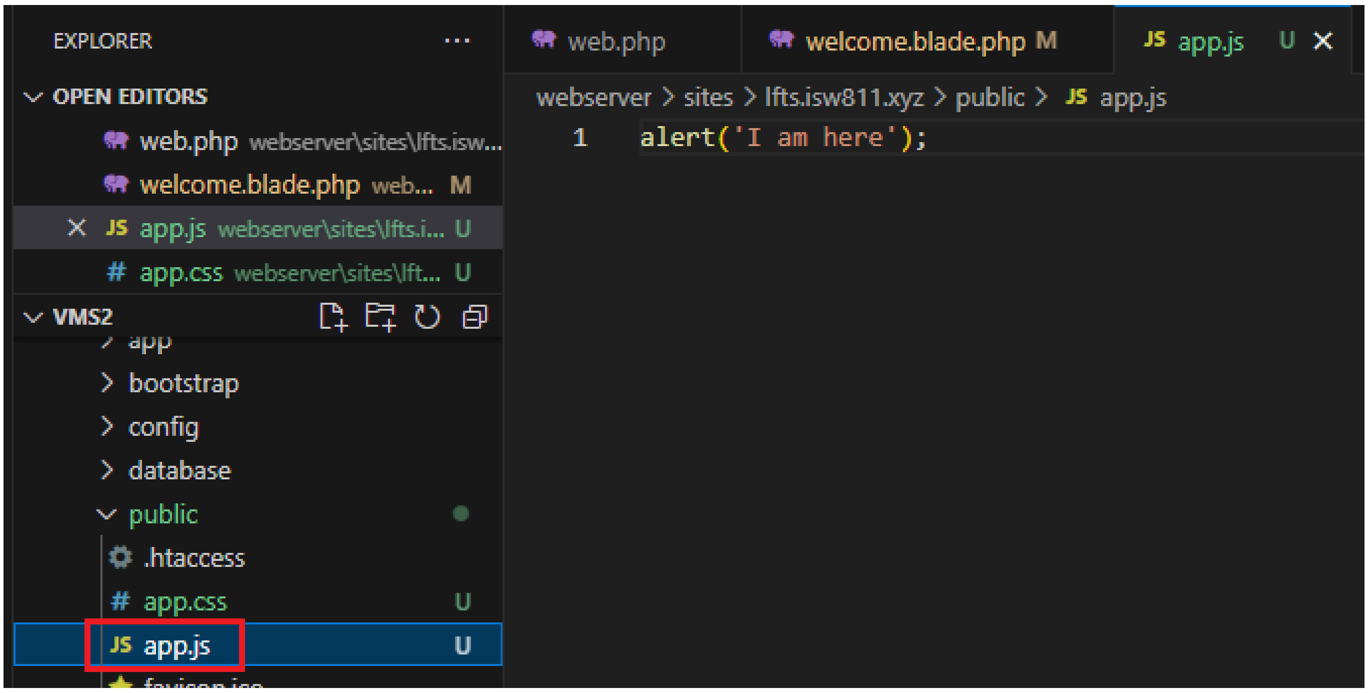
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">
<script src="/app.js"></script>

<body>
    <h1>Hello World</h1>
</body>
</html>

```

Agregamos un archivo app.css para dar estilo por medio de css


EXPLORER
OPEN EDITORS
web.php webserver\sites\lfts.isw...
welcome.blade.php web... M
app.js webserver\sites\lfts.i... U
app.css webserver\sites\lft... U
VMS2
app
bootstrap
config
database
public
.htaccess
app.css U
app.js U
favicon.ico
web.php
welcome.blade.php M
app.js U
app.css U
webserver > sites > lfts.isw811.xyz > public > app.css > body

```css
body{
    background: navy;
    color: white;
}
```

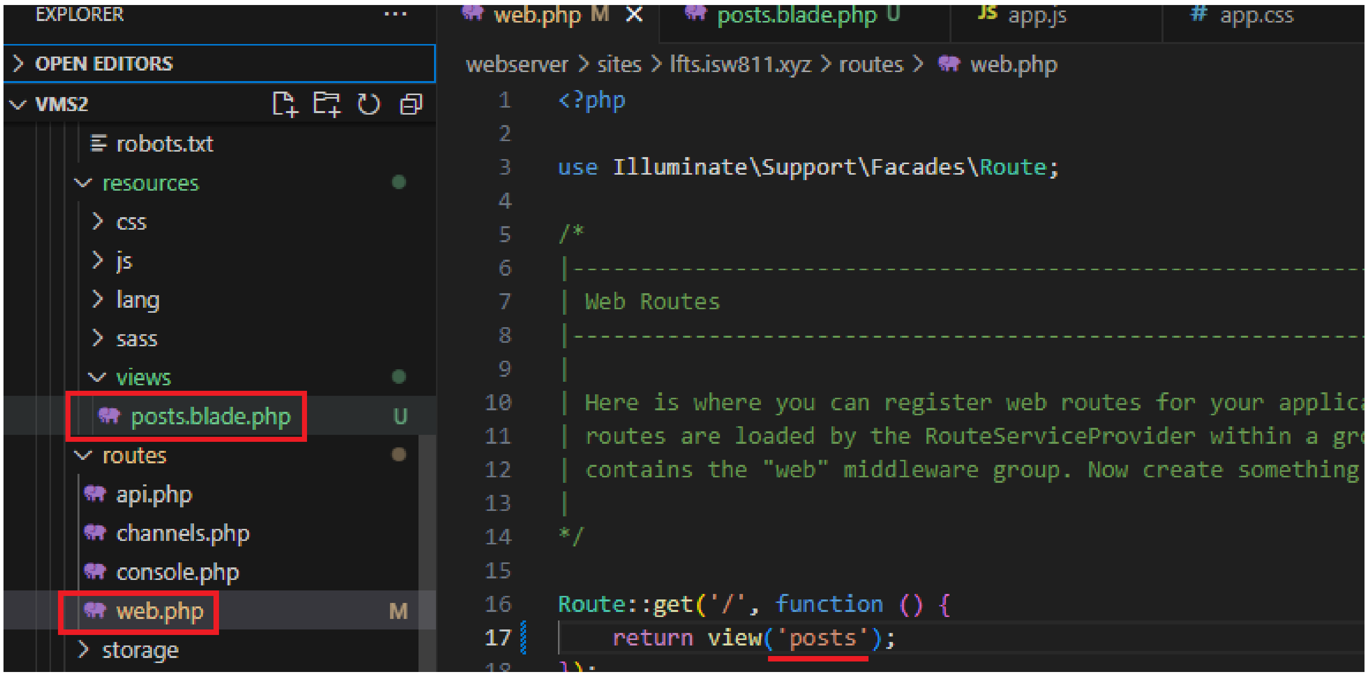
Agregamos un archivo app.js para crear código javascript


EXPLORER
OPEN EDITORS
web.php webserver\sites\lfts.isw...
welcome.blade.php web... M
app.js webserver\sites\lfts.i... U
app.css webserver\sites\lft... U
VMS2
app
bootstrap
config
database
public
.htaccess
app.css U
app.js U
favicon.ico
web.php
welcome.blade.php M
app.js U
webserver > sites > lfts.isw811.xyz > public > app.js

```javascript
alert('I am here');
```

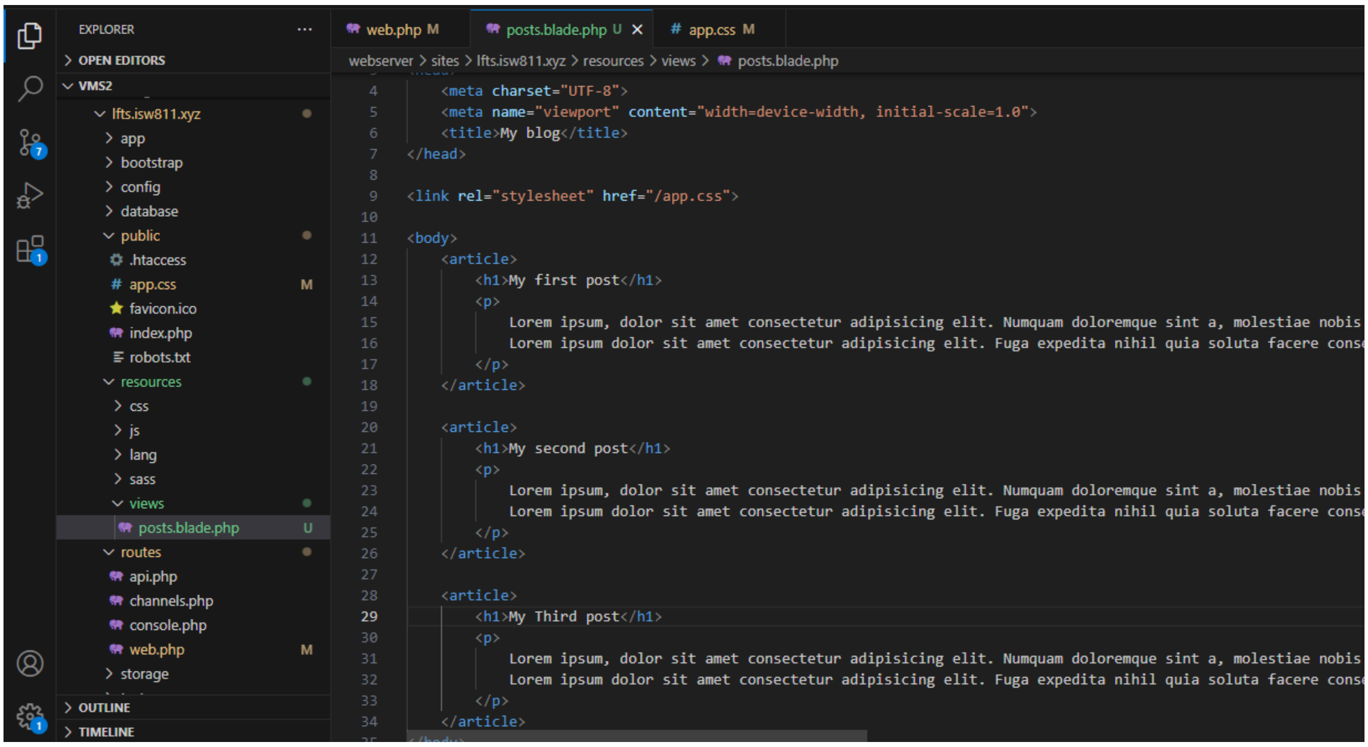
Make a Route and Link to it

Modificamos el nombre de la vista en la carpeta views y el llamado en el archivo web.php de la carpeta routes



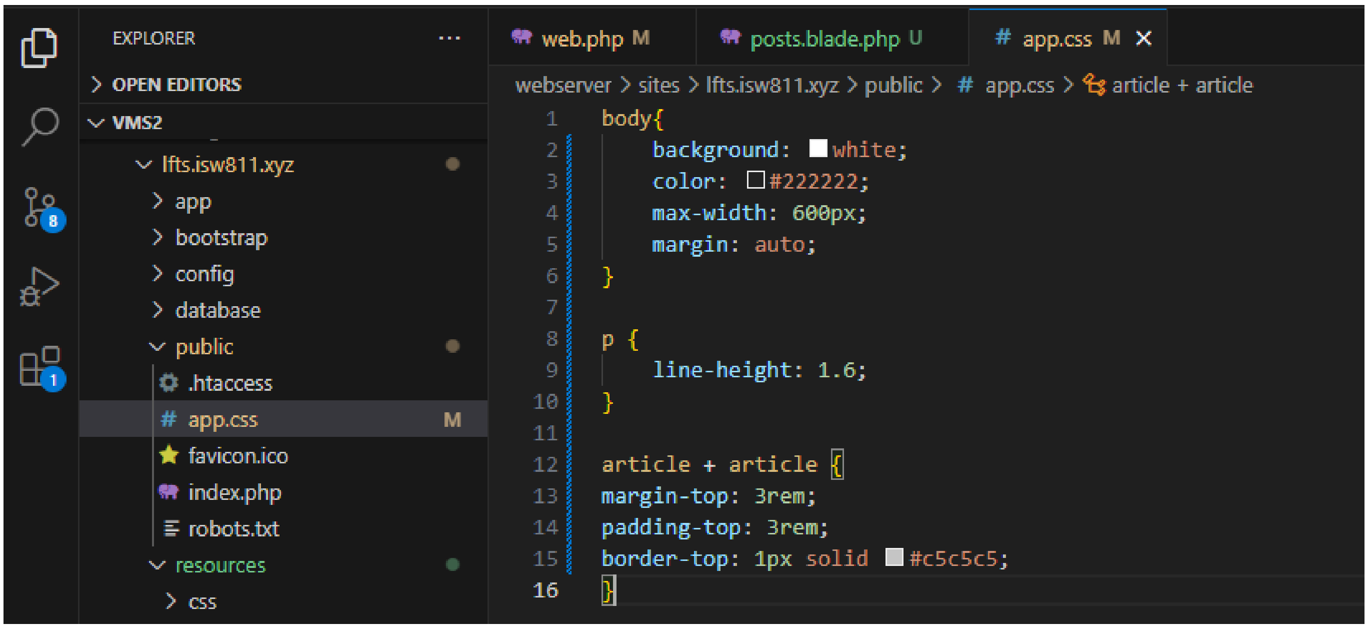
```php
<?php

use Illuminate\Support\Facades\Route;

/*
|-----------------------------------------------------------------
| Web Routes
|-----------------------------------------------------------------
|
| Here is where you can register web routes for your applica
| routes are loaded by the RouteServiceProvider within a gro
| contains the "web" middleware group. Now create something
|
*/

Route::get('/', function () {
    return view('posts');
```

Cambiamos el contenido del html de posts.blade.php



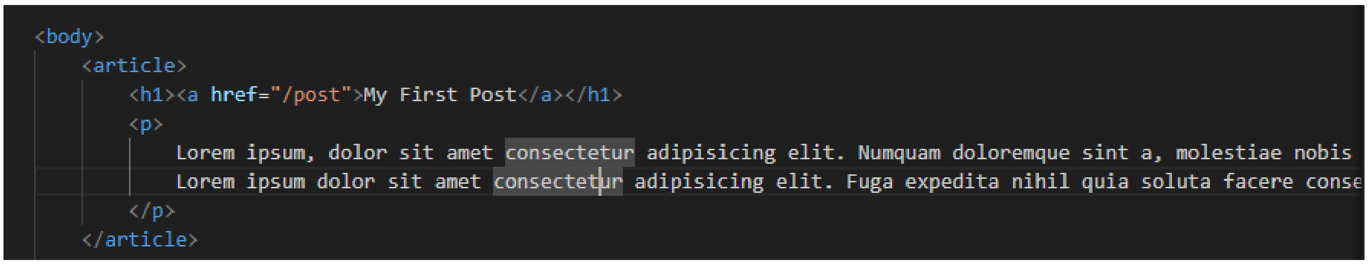
```html
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">

<body>
    <article>
        <h1>My first post</h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere conse
        </p>
    </article>

    <article>
        <h1>My second post</h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere conse
        </p>
    </article>

    <article>
        <h1>My Third post</h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere conse
        </p>
    </article>
```

Junto con el css

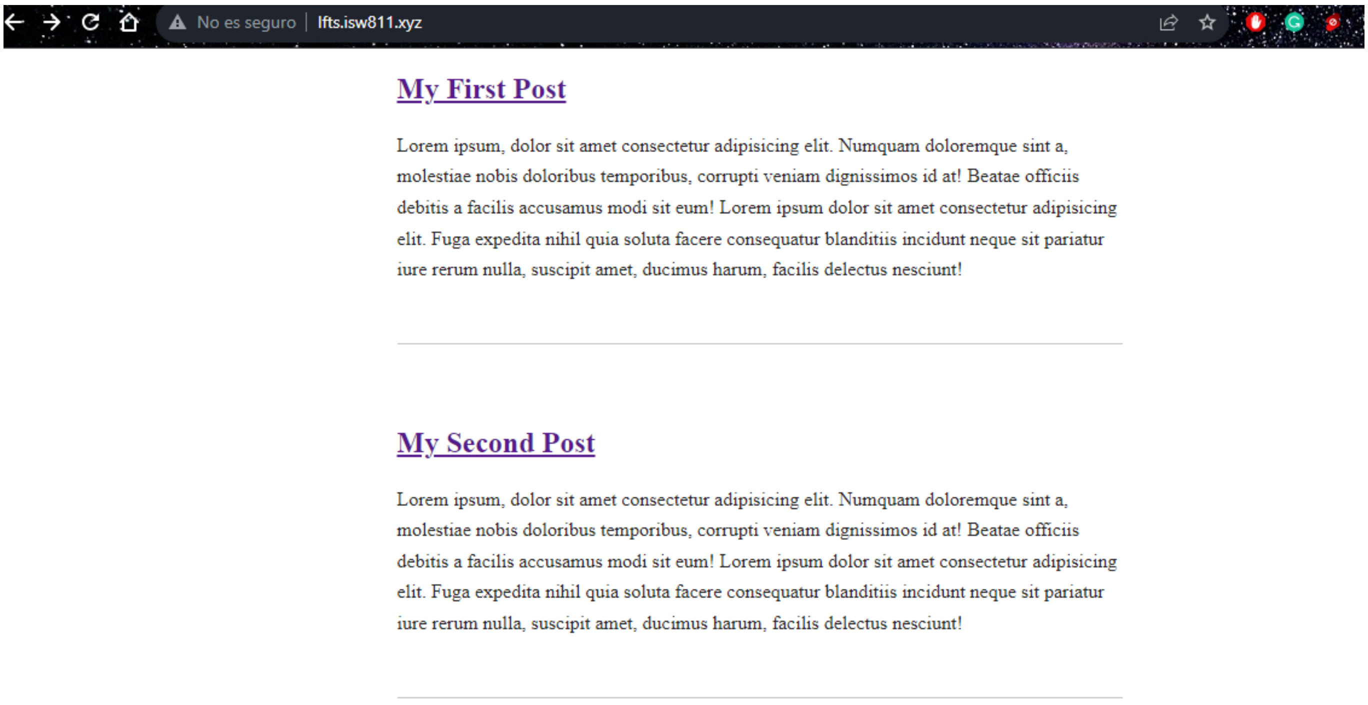


```
body{
    background: white;
    color: #222222;
    max-width: 600px;
    margin: auto;
}

p {
    line-height: 1.6;
}

article + article {
margin-top: 3rem;
padding-top: 3rem;
border-top: 1px solid #c5c5c5;
}
```

Hacemos los titulos de cada post cliqueables y con un link que redireccione a otra página



```
<body>
    <article>
        <h1><a href="/post">My First Post</a></h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere conse
        </p>
    </article>
```



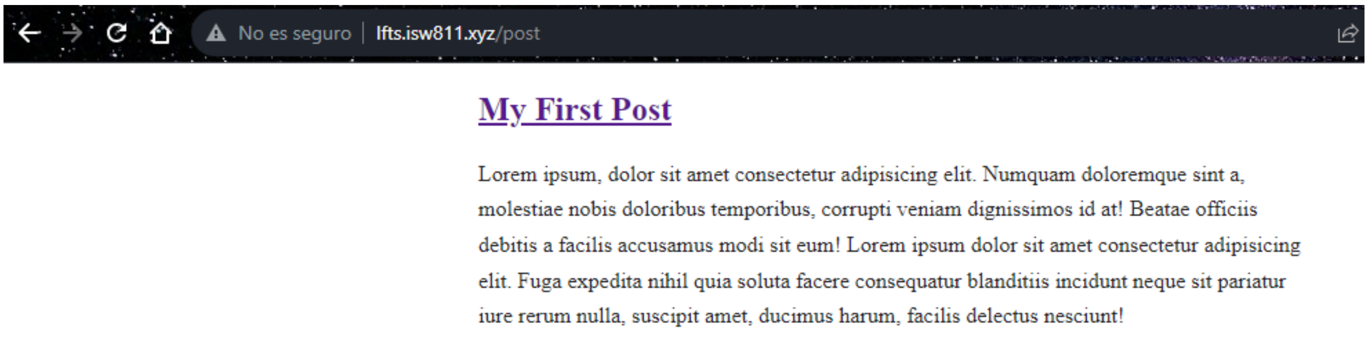
No es seguro | lfts.isw811.xyz

My First Post

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!

My Second Post

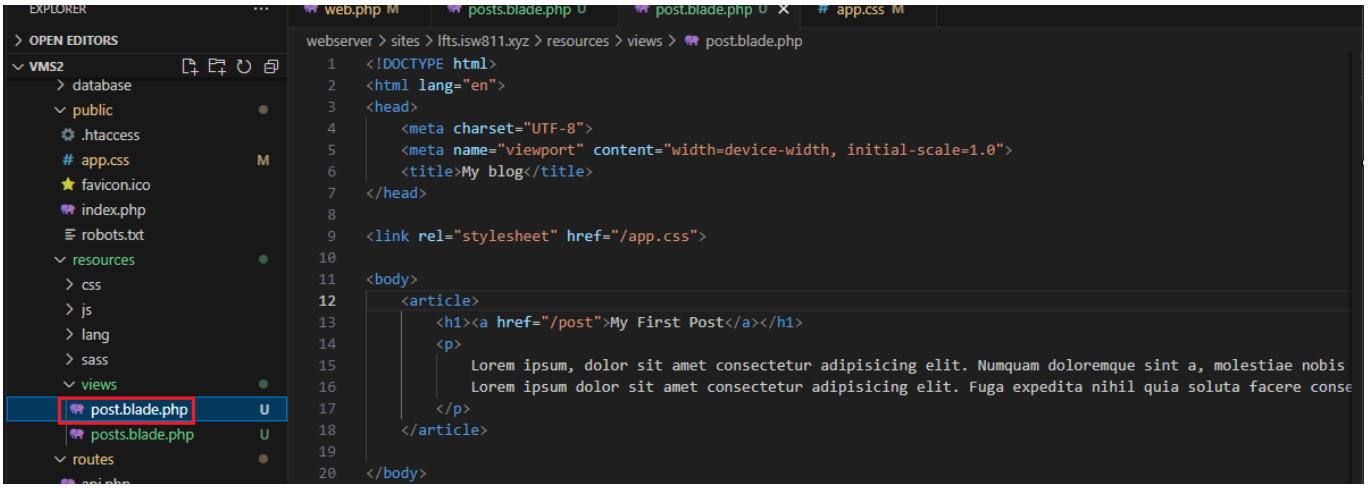
Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!



My First Post

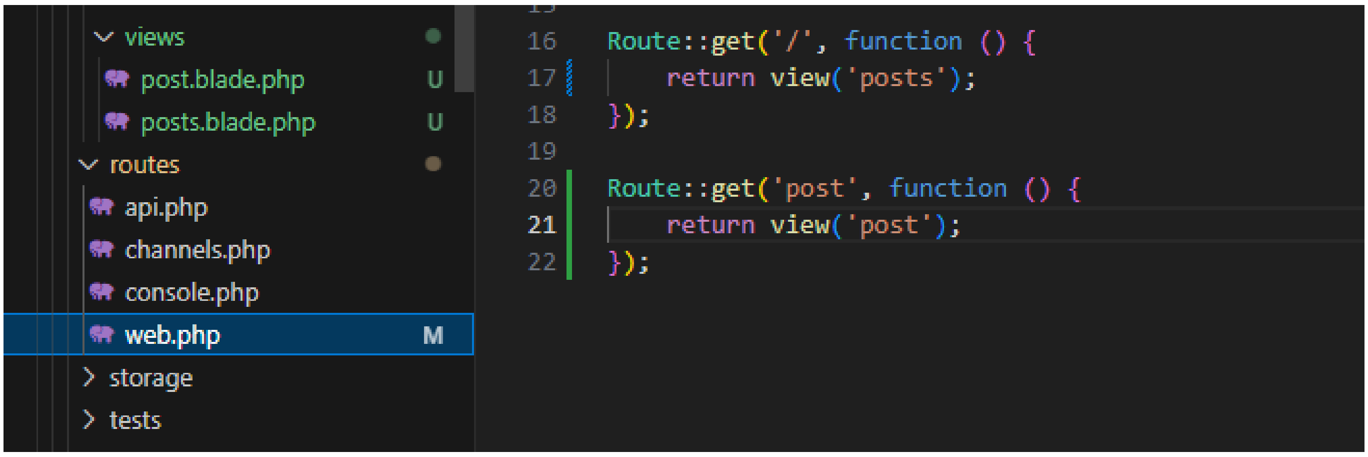
Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!

Esto se logra creando una nueva vista en la carpeta views



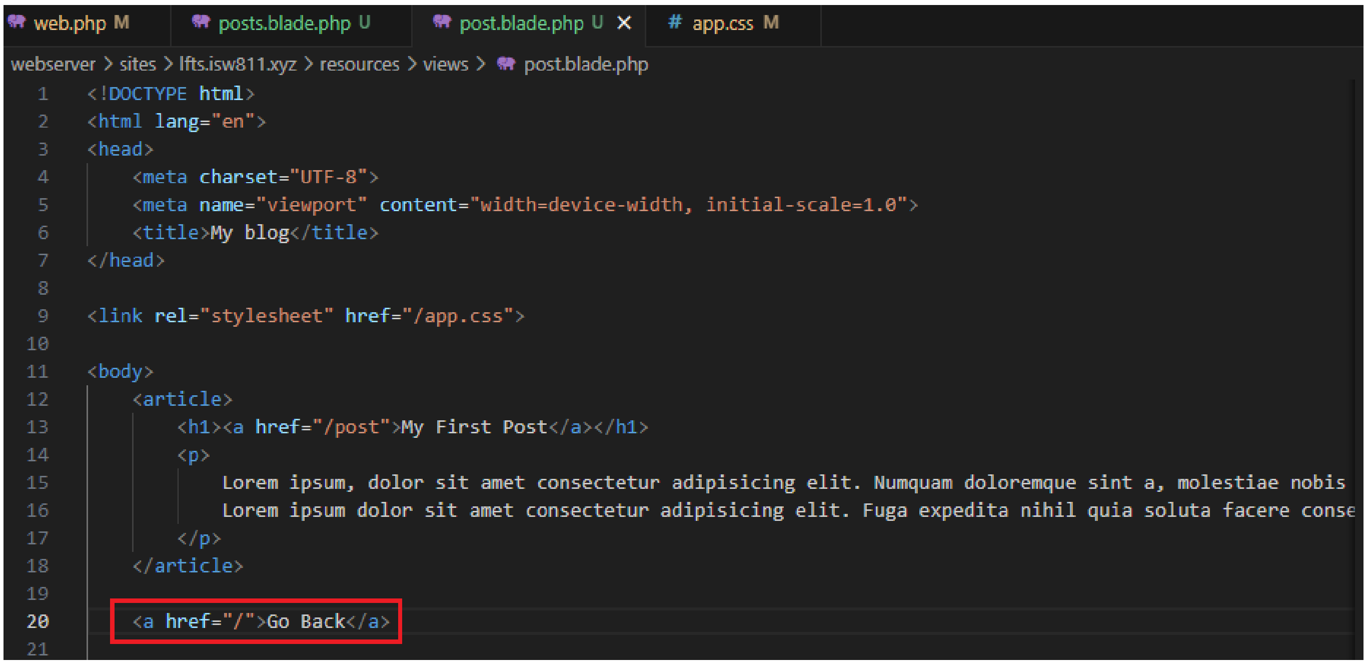
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">

<body>
    <article>
        <h1><a href="/post">My First Post</a></h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere conse
        </p>
    </article>

</body>
```

Y una nueva ruta en web.php

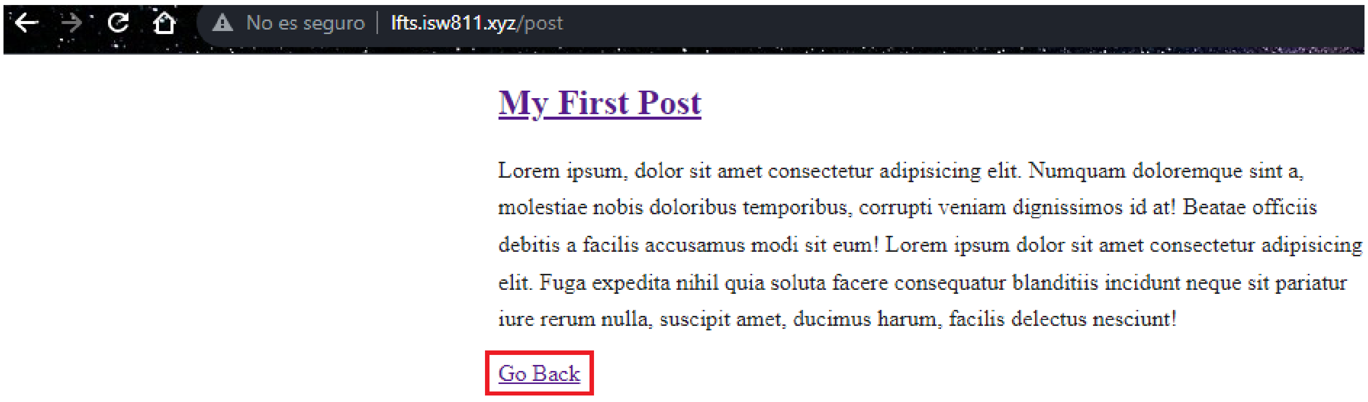


```
Route::get('/', function () {
    return view('posts');
});

Route::get('post', function () {
    return view('post');
});
```

Una vez creada la nueva vista del post individual, creamos un boton para regresar al Home


web.php M
posts.blade.php U
post.blade.php U
app.css M
webserver > sites > lfts.isw811.xyz > resources > views > post.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">

<body>
    <article>
        <h1><a href="/post">My First Post</a></h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere conse
        </p>
    </article>

    <a href="/">Go Back</a>
```



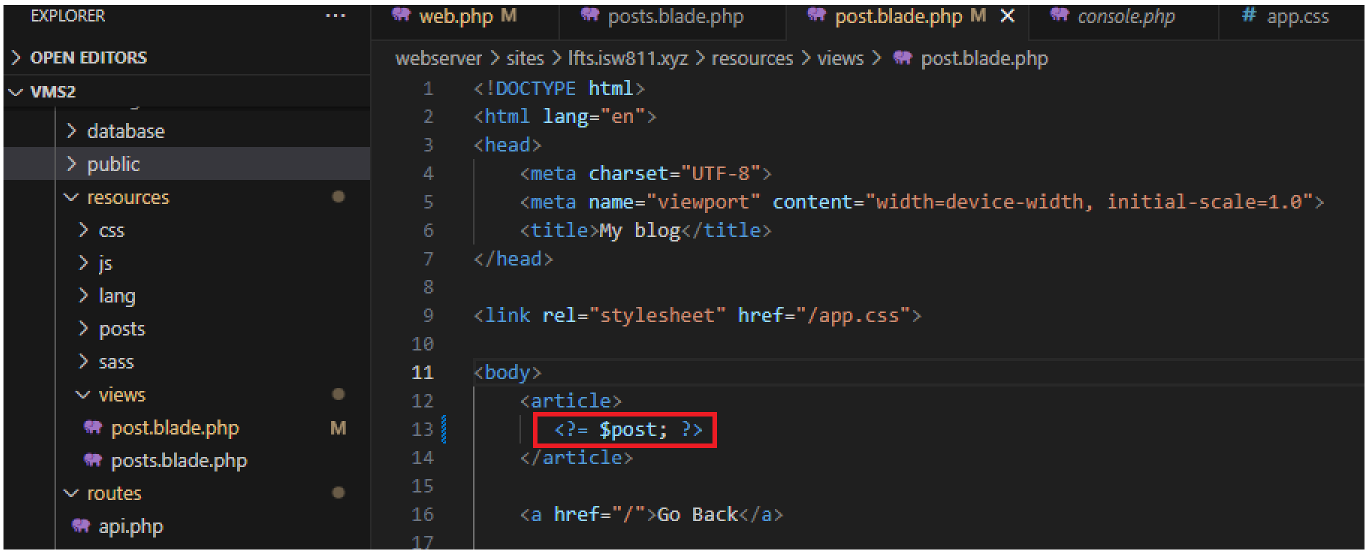
My First Post

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!

Go Back

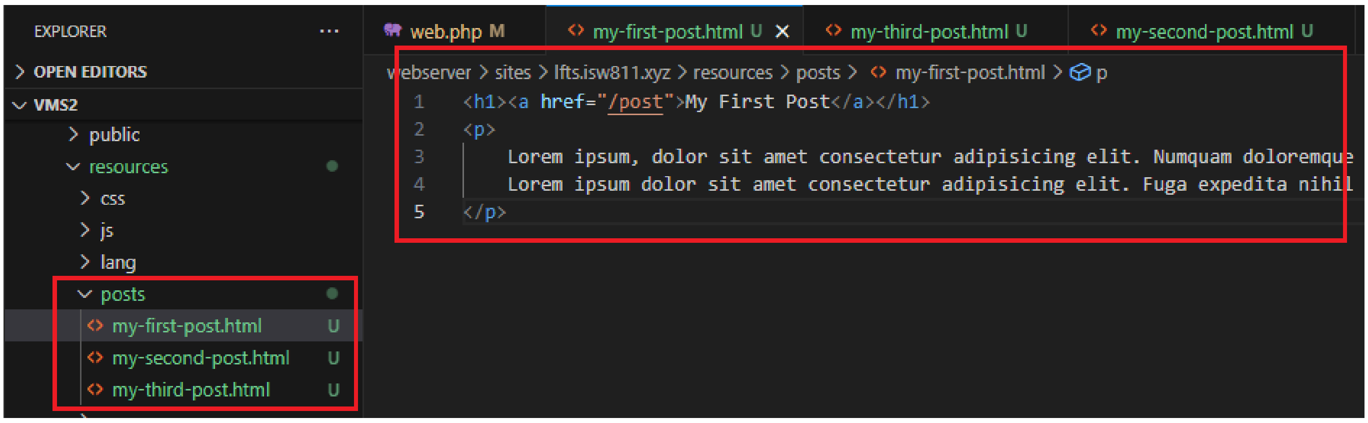
Store Blog Posts as HTML Files

Creamos una variable $post que podamos llamar



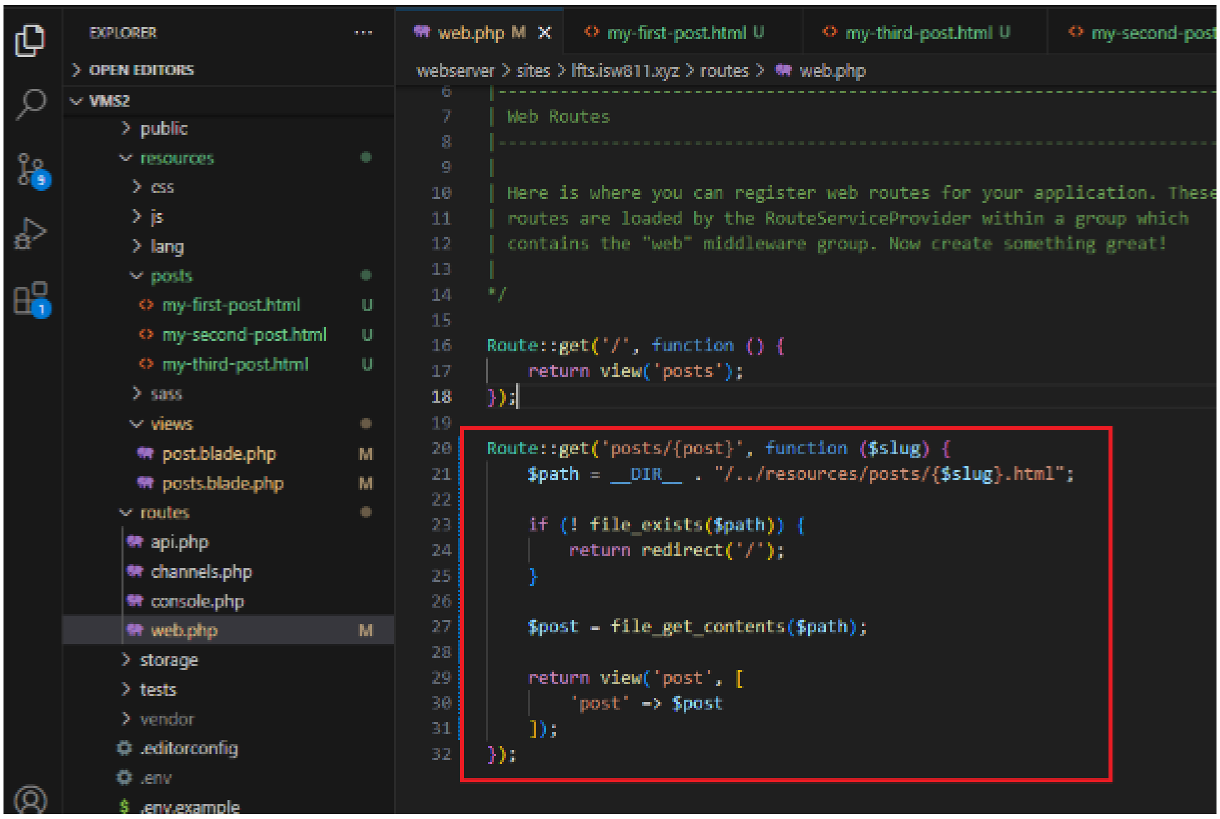
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">

<body>
    <article>
        <?= $post; ?>
    </article>

    <a href="/">Go Back</a>
```

Creamos un folder llamado Posts con un archivo html por cada post



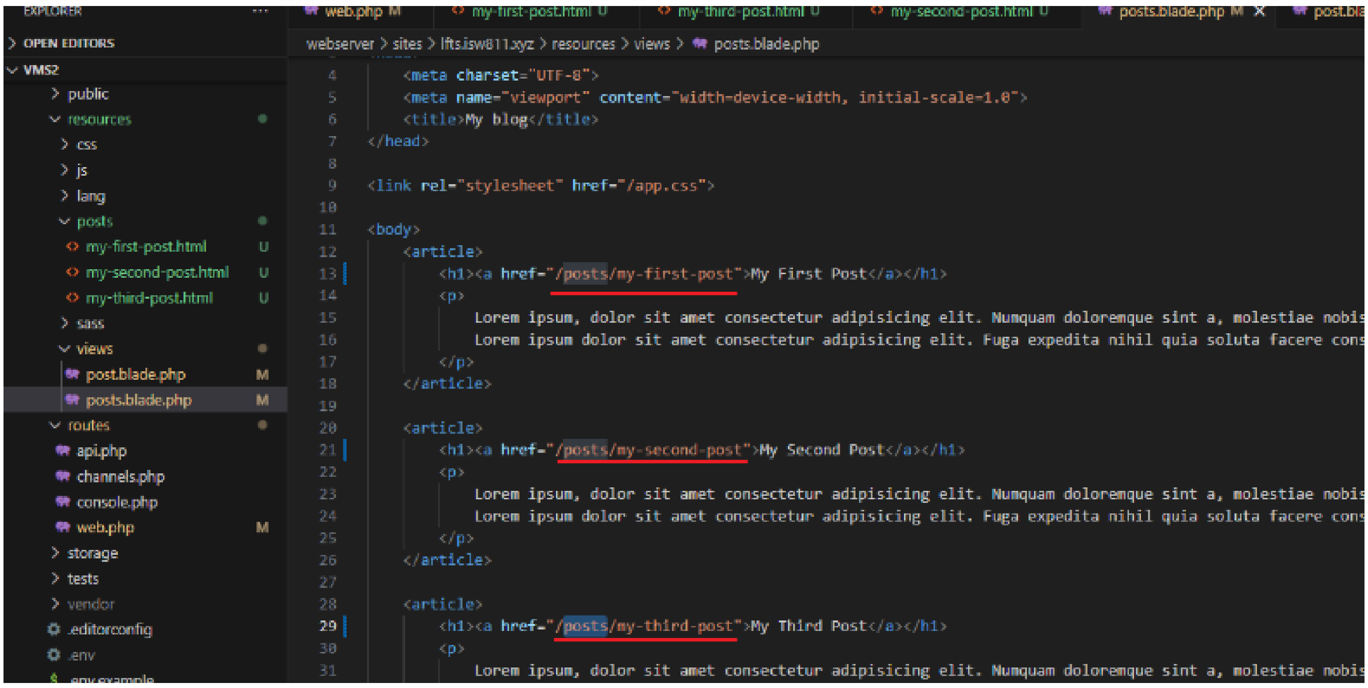
```html
<h1><a href="/post">My First Post</a></h1>
<p>
    Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil
</p>
```

Pero para que funcione la variable $post debemos crearla en el archivo de rutas web.php cambiando un poco el get de la ruta, vamos a crear variables que obtengan la ruta del post seleccionado y vamos a evitar errores cuando se digite una ruta inexistente en la url



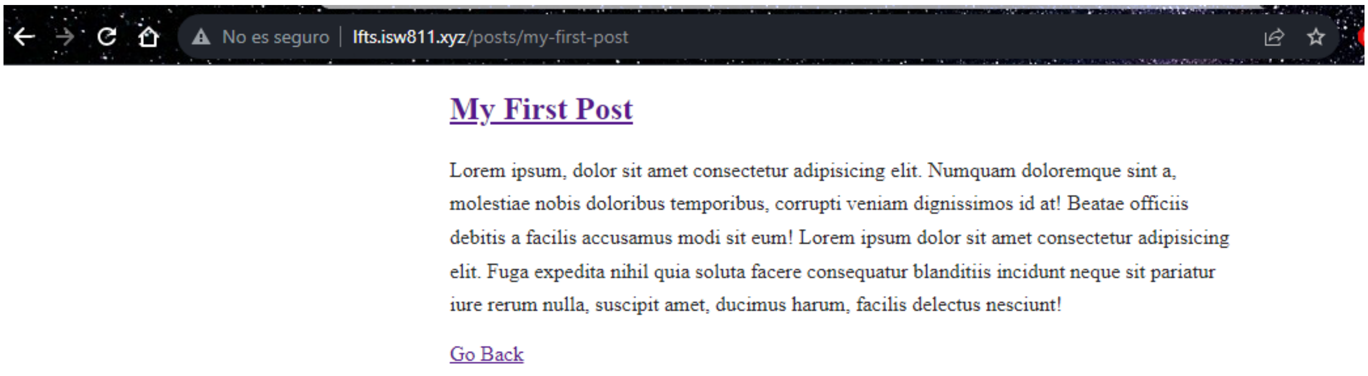
```php
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('posts');
});

Route::get('posts/{post}', function ($slug) {
    $path = __DIR__ . "/../resources/posts/{$slug}.html";

    if (! file_exists($path)) {
        return redirect('/');
    }

    $post = file_get_contents($path);

    return view('post', [
        'post' => $post
    ]);
});
```

Antes de probar las rutas primero debemos modificarlas en el archivo posts.blade.php



```html
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">

<body>
    <article>
        <h1><a href="/posts/my-first-post">My First Post</a></h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere cons
        </p>
    </article>

    <article>
        <h1><a href="/posts/my-second-post">My Second Post</a></h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere cons
        </p>
    </article>

    <article>
        <h1><a href="/posts/my-third-post">My Third Post</a></h1>
        <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis
```

A nivel de la página se vería de la siguiente manera



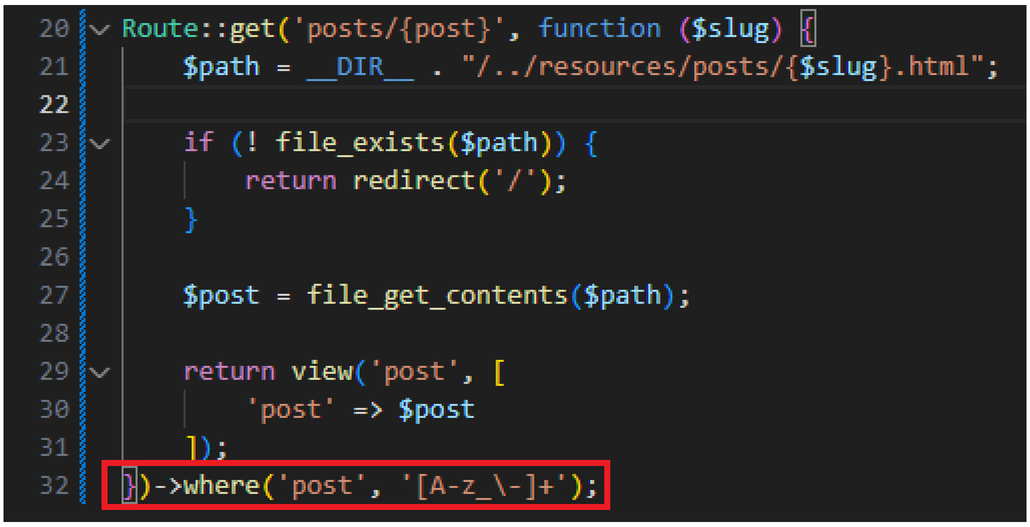
My First Post

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!

Go Back

Route Wildcard Constraints

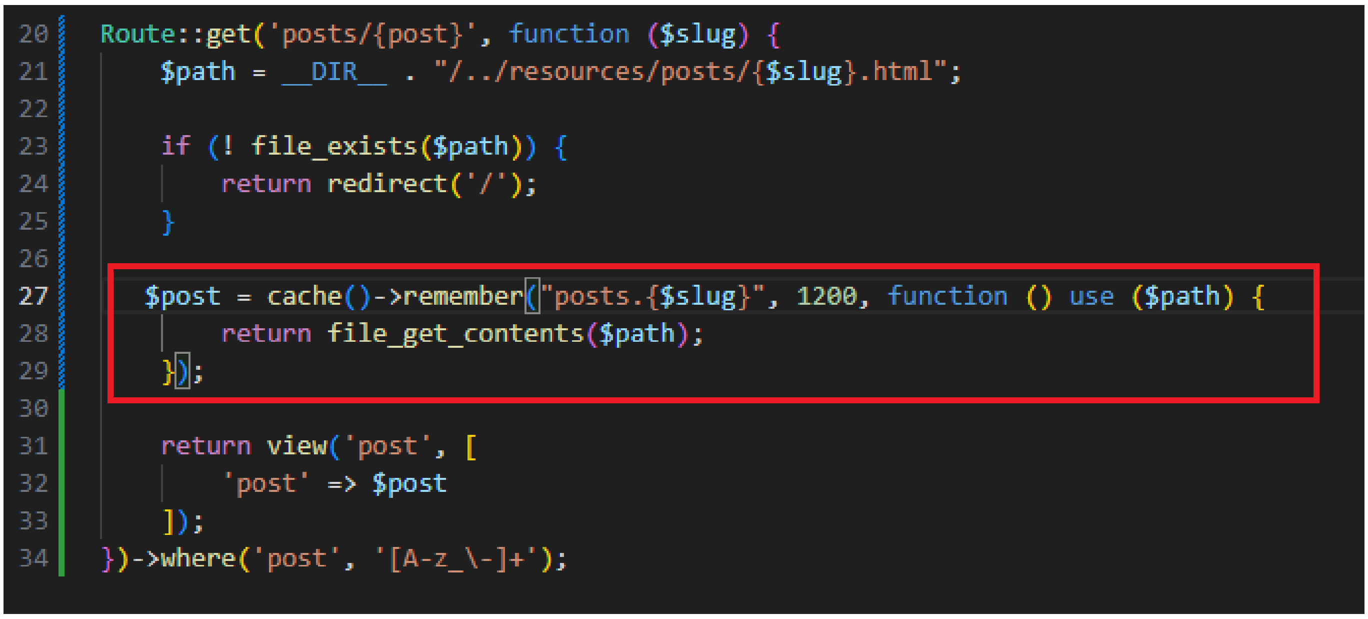
Para delimitar lo que se puede o no poner en la ruta utilizamos un where para una expresión regular



```php
Route::get('posts/{post}', function ($slug) {
    $path = __DIR__ . "/../resources/posts/{$slug}.html";

    if (! file_exists($path)) {
        return redirect('/');
    }

    $post = file_get_contents($path);

    return view('post', [
        'post' => $post
    ]);
})->where('post', '[A-z_\-]+');
```

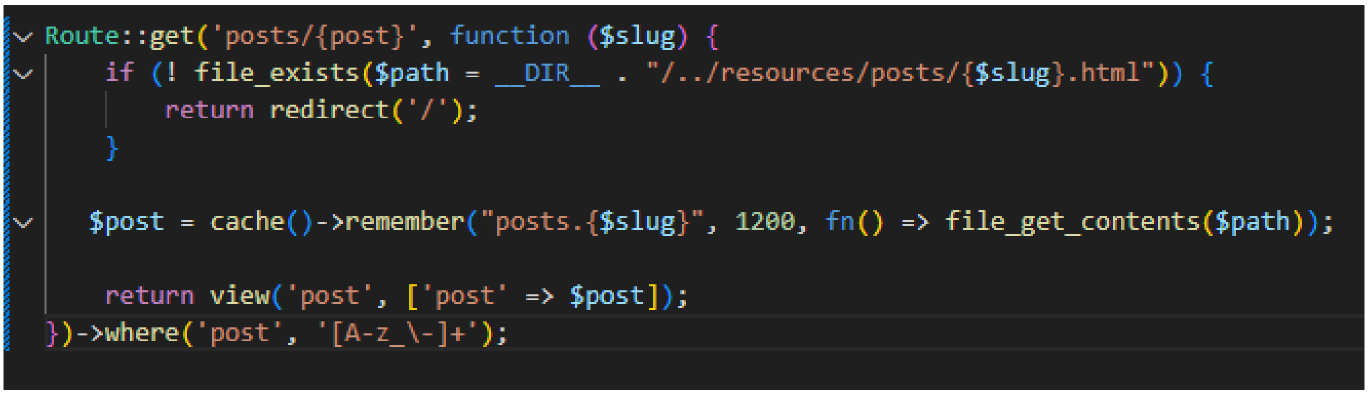
Use Caching for Expensive Operations

Vamos a crear código para que cada vez que se acceda a la ruta nuevamente, la cargue desde la memoria cache y no tenga que pasar por el sistema de archivos en cada hit.



```php
Route::get('posts/{post}', function ($slug) {
    $path = __DIR__ . "/../resources/posts/{$slug}.html";

    if (! file_exists($path)) {
        return redirect('/');
    }

    $post = cache()->remember("posts.{$slug}", 1200, function () use ($path) {
        return file_get_contents($path);
    });

    return view('post', [
        'post' => $post
    ]);
})->where('post', '[A-z_\-]+');
```

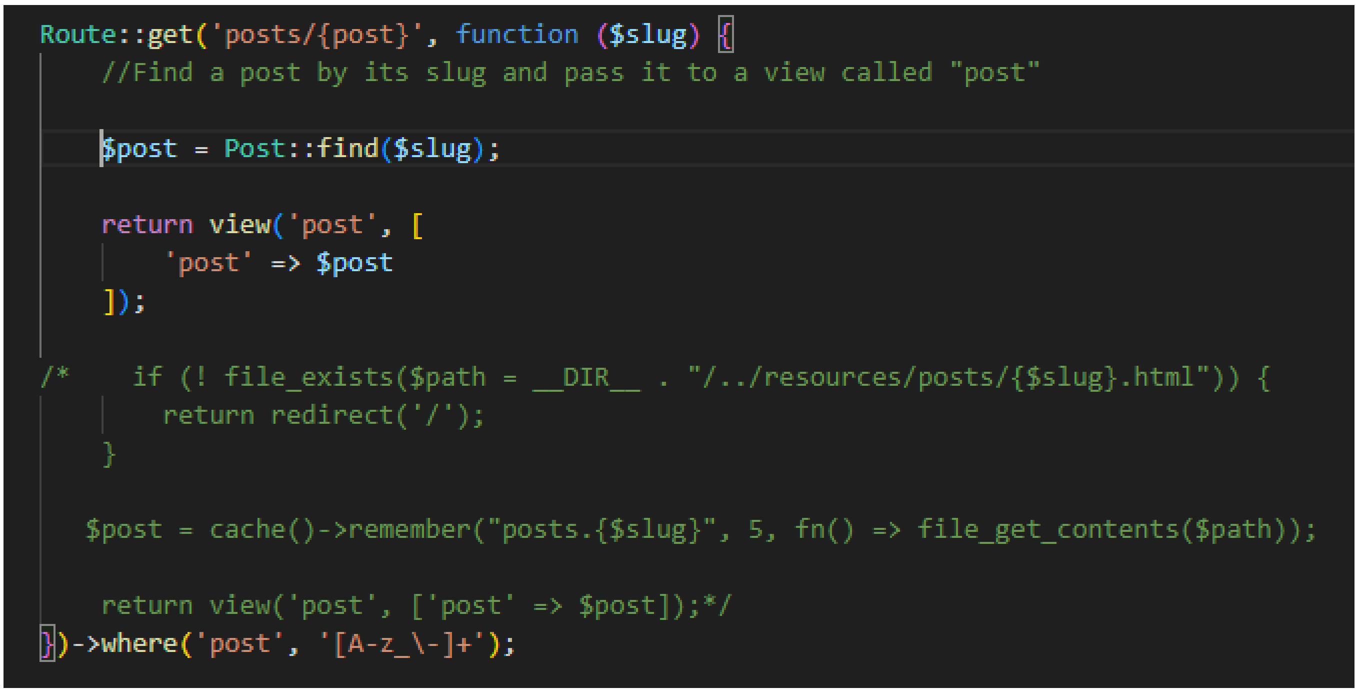
Así se vería el código de la ruta un poco mas limpio



```php
Route::get('posts/{post}', function ($slug) {
    if (! file_exists($path = __DIR__ . "/../resources/posts/{$slug}.html")) {
        return redirect('/');
    }

    $post = cache()->remember("posts.{$slug}", 1200, fn() => file_get_contents($path));

    return view('post', ['post' => $post]);
})->where('post', '[A-z_\-]+');
```

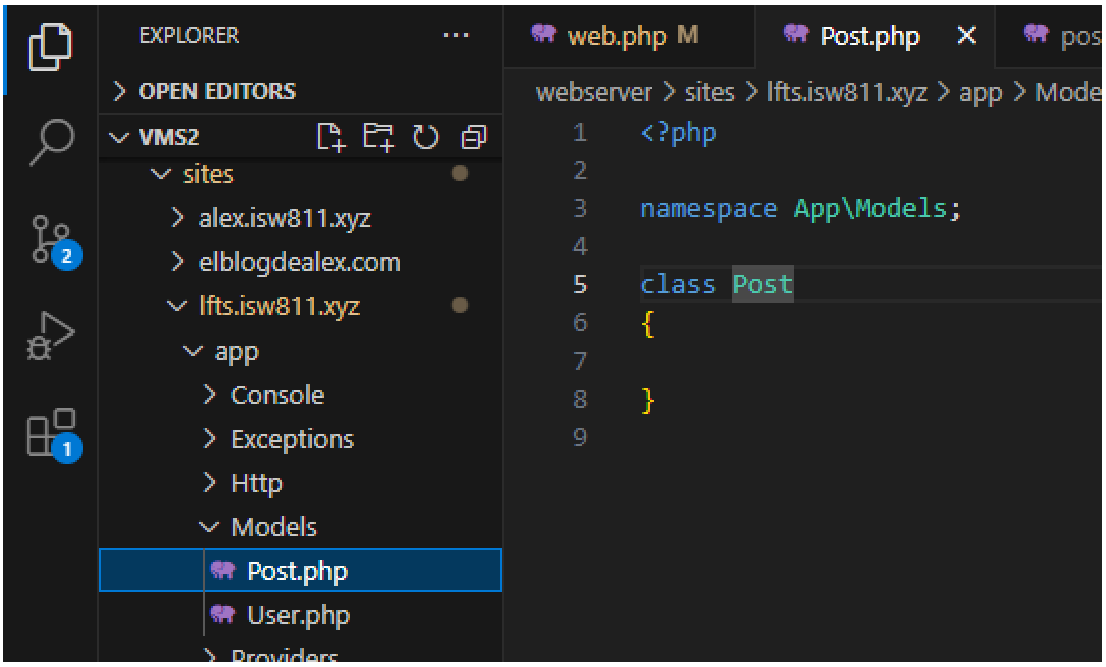
Use the Filesystem Class to Read a Directory

Vamos a cambiar el codigo de la ruta para poder encontrar post especificos y pasarlos a la vista "post". Demos tomar en cuenta que la clase Post no está creada, es el siguiente paso.



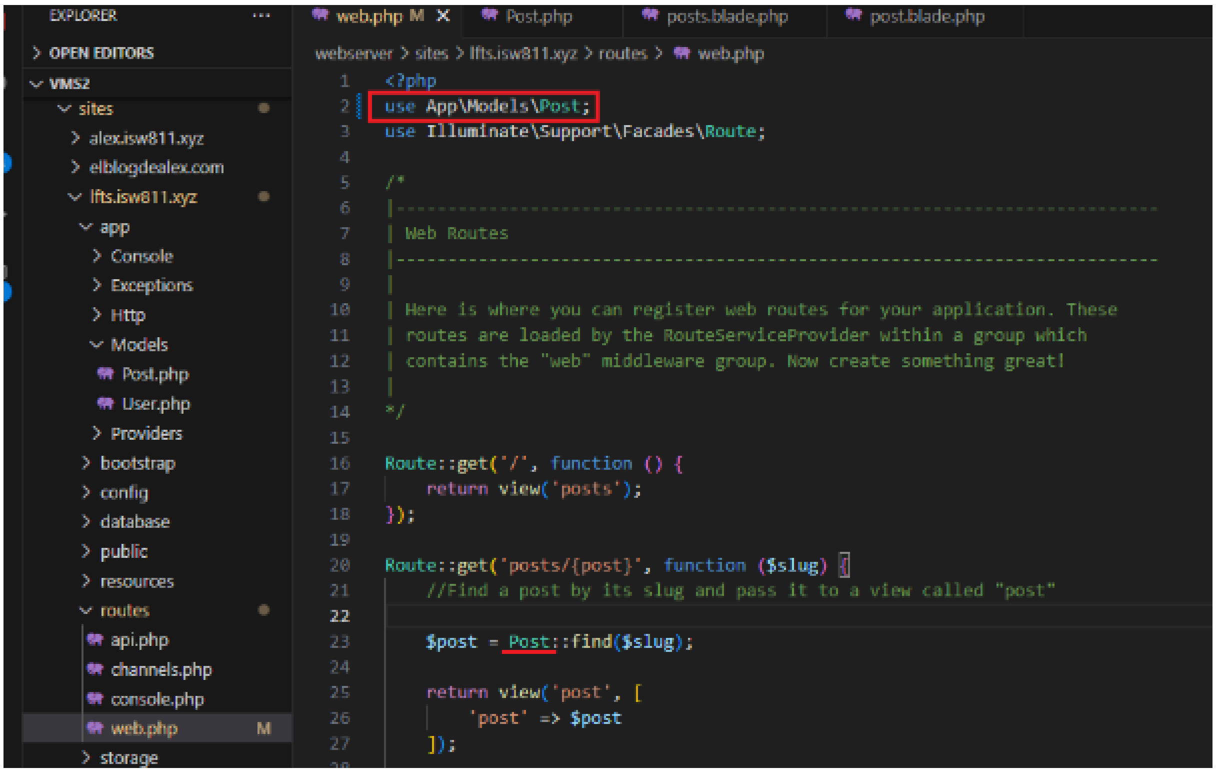
```php
Route::get('posts/{post}', function ($slug) {
    //Find a post by its slug and pass it to a view called "post"

    $post = Post::find($slug);

    return view('post', [
        'post' => $post
    ]);

/*    if (! file_exists($path = __DIR__ . "/../resources/posts/{$slug}.html")) {
        return redirect('/');
    }

   $post = cache()->remember("posts.{$slug}", 5, fn() => file_get_contents($path));

    return view('post', ['post' => $post]);*/
})->where('post', '[A-z_\-]+');
```

Creamos la clase o modelo Post dentro de la carpeta App / Models y llamamos al modelo desde route en web.php



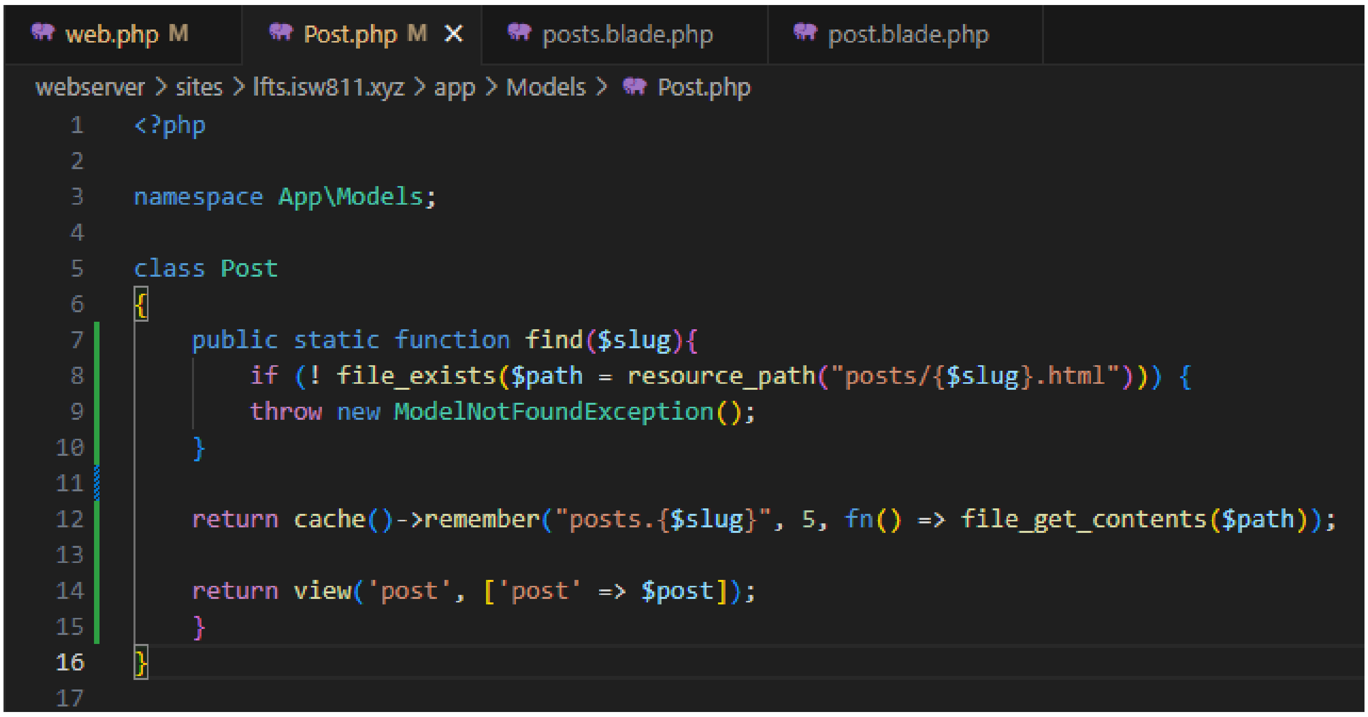
```php
<?php

namespace App\Models;

class Post
{

}
```

Llamado del modelo Post



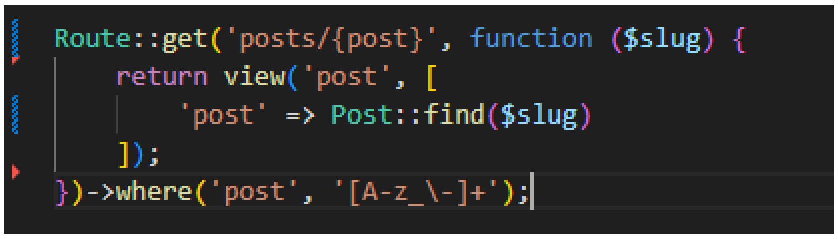
```php
<?php
use App\Models\Post;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('posts');
});

Route::get('posts/{post}', function ($slug) {
    //Find a post by its slug and pass it to a view called "post"

    $post = Post::find($slug);

    return view('post', [
        'post' => $post
    ]);
```

Ahora movemos el codigo anterior que teniamos en nuestro archivo de rutas(web.php) a la clase Post, con algunas modificaciones


web.php M
Post.php M
posts.blade.php
post.blade.php
webserver > sites > lfts.isw811.xyz > app > Models > Post.php

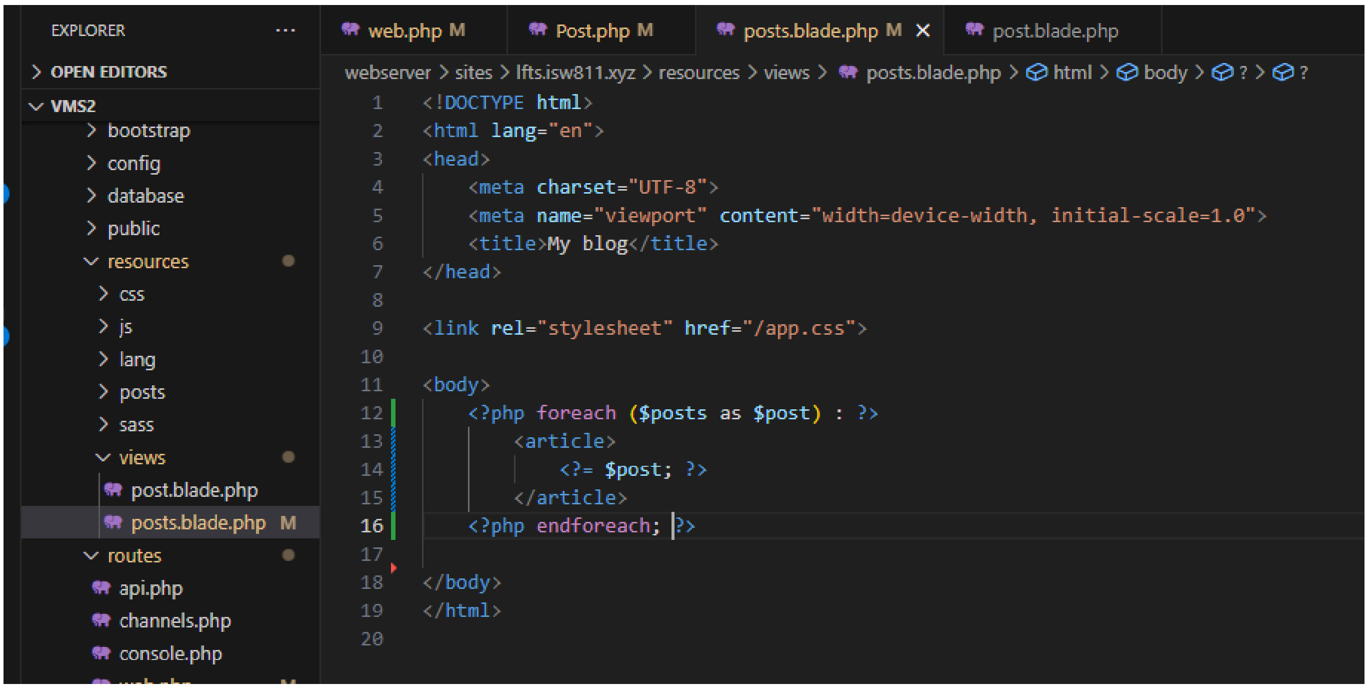
```php
<?php

namespace App\Models;

class Post
{
    public static function find($slug){
        if (! file_exists($path = resource_path("posts/{$slug}.html"))) {
        throw new ModelNotFoundException();
    }

    return cache()->remember("posts.{$slug}", 5, fn() => file_get_contents($path));

    return view('post', ['post' => $post]);
    }
}
```

Y volvemos a modificar el archivo de rutas(web.php) para mejorar el codigo



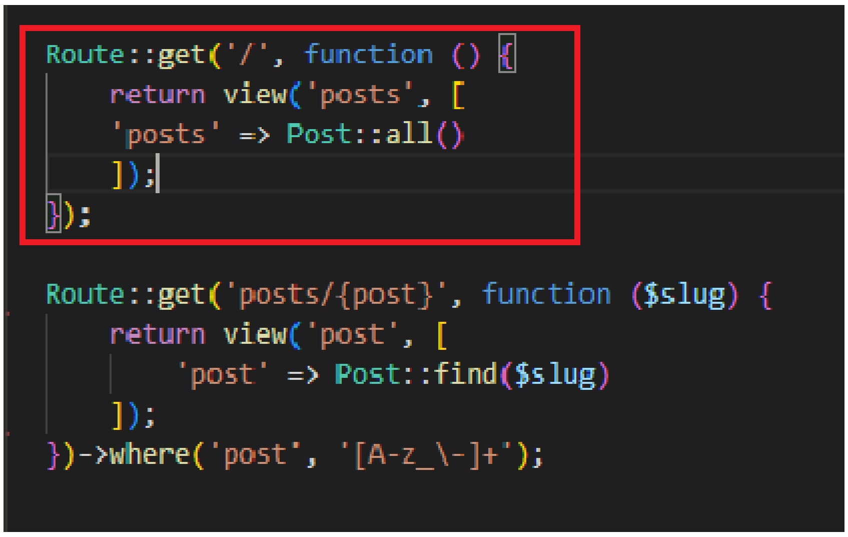
```php
Route::get('posts/{post}', function ($slug) {
    return view('post', [
        'post' => Post::find($slug)
    ]);
})->where('post', '[A-z_\-]+');
```

Modificamos los posts quemados anteriormente por el un foreach



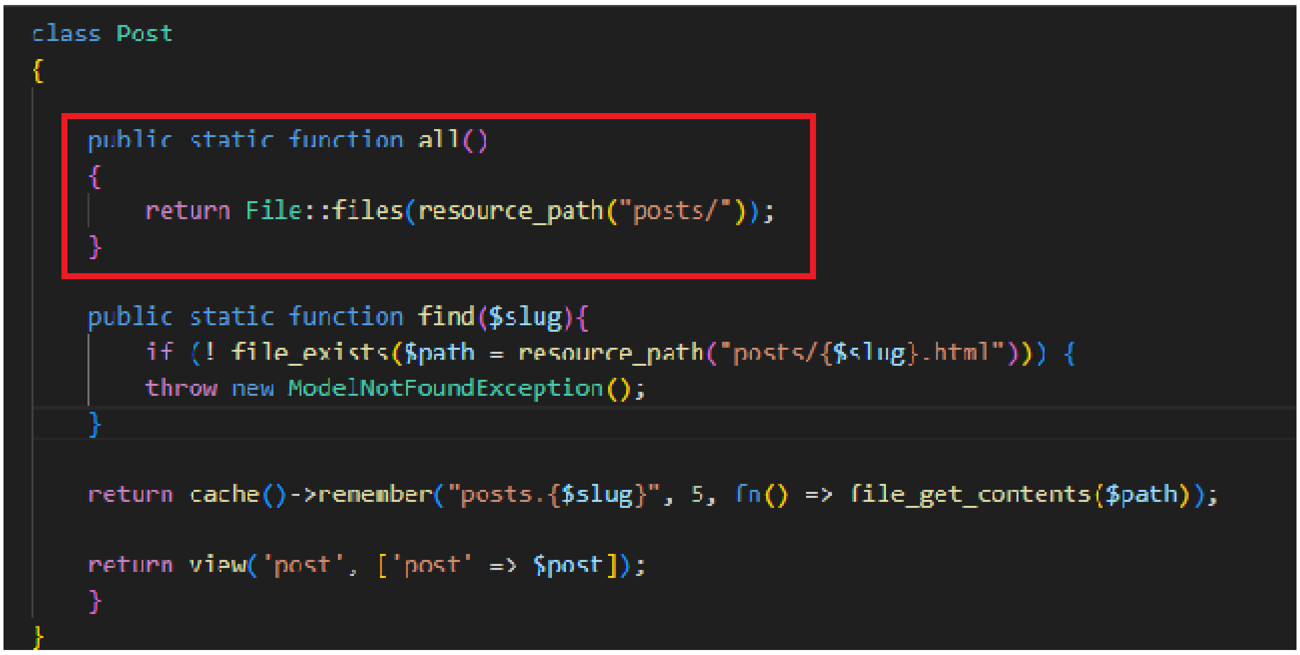
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My blog</title>
</head>

<link rel="stylesheet" href="/app.css">

<body>
    <?php foreach ($posts as $post) : ?>
        <article>
            <?= $post; ?>
        </article>
    <?php endforeach; ?>

</body>
</html>
```

Luego modificamos en el archivo de rutas, la ruta del home



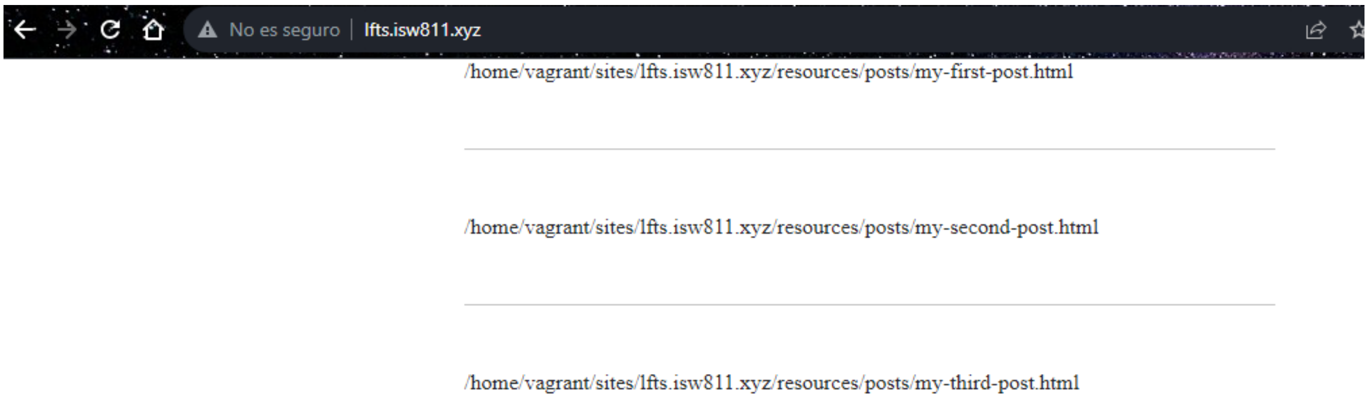
```
Route::get('/', function () {
    return view('posts', [
    'posts' => Post::all()
    ]);
});

Route::get('posts/{post}', function ($slug) {
    return view('post', [
        'post' => Post::find($slug)
    ]);
})->where('post', '[A-z_\-]+');
```

Y ya que no temos un metodo all, debemos crearlo



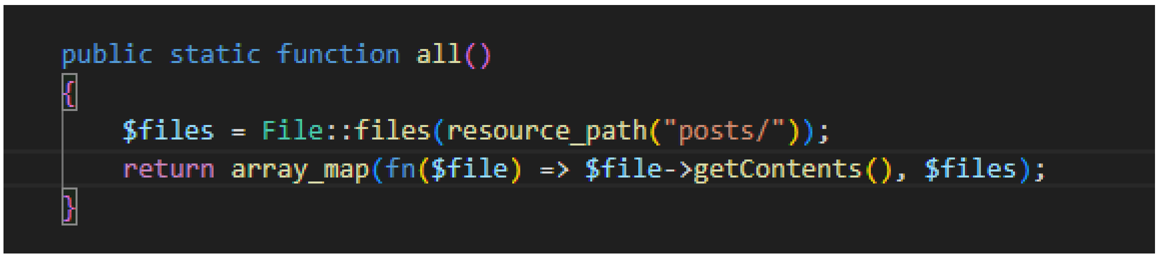
```php
class Post
{
    public static function all()
    {
        return File::files(resource_path("posts/"));
    }

    public static function find($slug){
        if (! file_exists($path = resource_path("posts/{$slug}.html"))) {
        throw new ModelNotFoundException();
    }

    return cache()->remember("posts.{$slug}", 5, fn() => file_get_contents($path));

    return view('post', ['post' => $post]);
    }
}
```

Asi se ve el home de momento



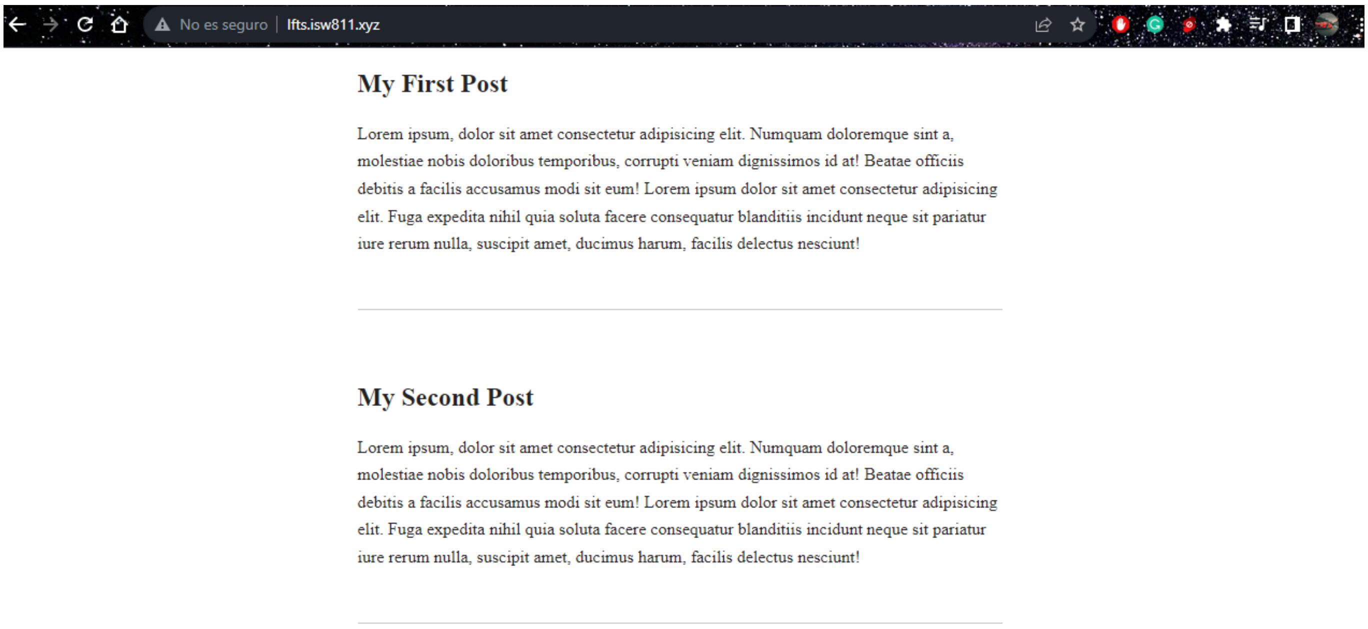
```
/home/vagrant/sites/lfts.isw811.xyz/resources/posts/my-first-post.html

/home/vagrant/sites/lfts.isw811.xyz/resources/posts/my-second-post.html

/home/vagrant/sites/lfts.isw811.xyz/resources/posts/my-third-post.html
```

Por lo que tenemos que modificar el objeto all



```php
public static function all()
{
    $files = File::files(resource_path("posts/"));
    return array_map(fn($file) => $file->getContents(), $files);
}
```

Y ya se verán nuestros posts



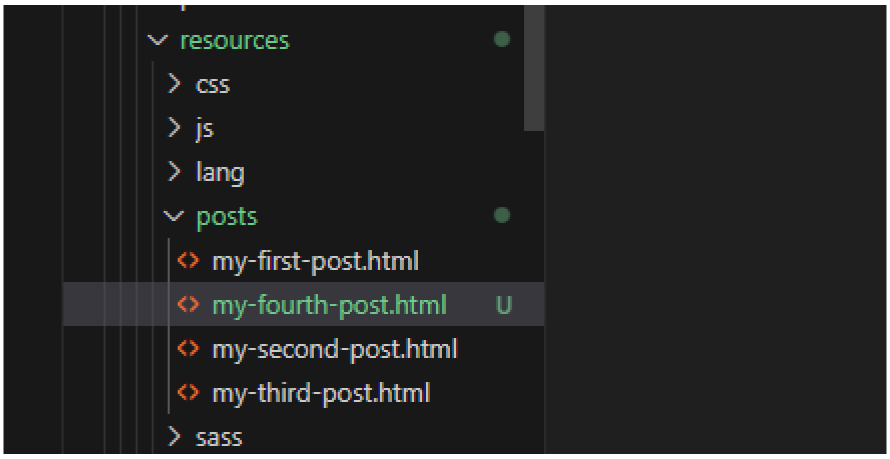
My First Post

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!

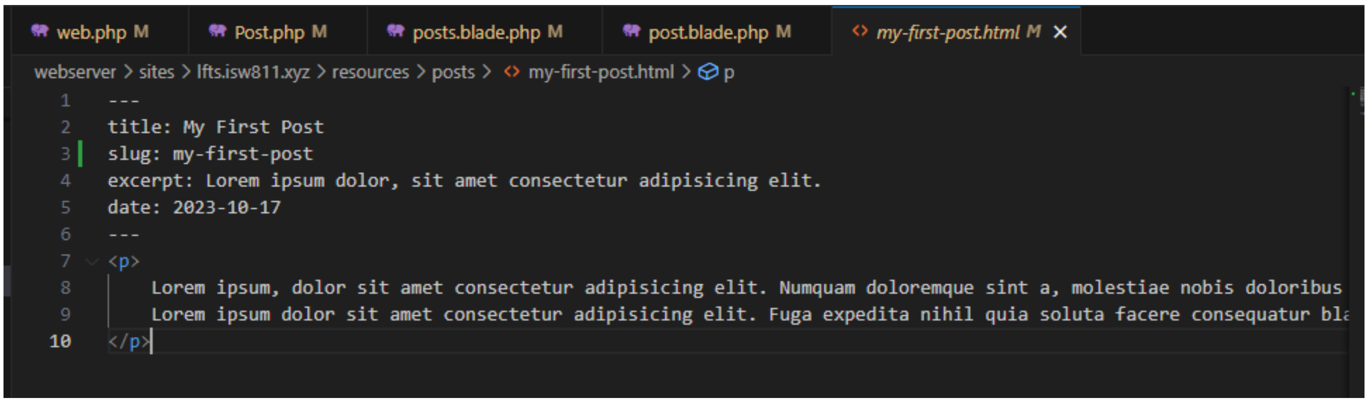
My Second Post

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis debitis a facilis accusamus modi sit eum! Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit amet, ducimus harum, facilis delectus nesciunt!

Agregamos un nuevo archivo post



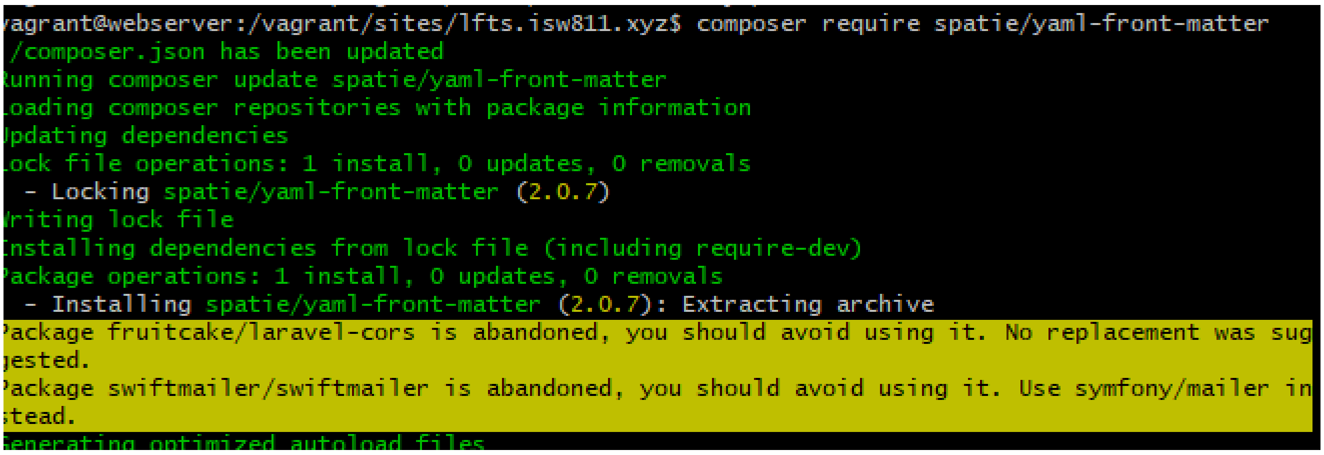
Y les colocamos metadata al inicio


web.php M
Post.php M
posts.blade.php M
post.blade.php M
my-first-post.html M
webserver > sites > lfts.isw811.xyz > resources > posts > my-first-post.html > p

```
---
title: My First Post
slug: my-first-post
excerpt: Lorem ipsum dolor, sit amet consectetur adipisicing elit.
date: 2023-10-17
---
<p>
    Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sint a, molestiae nobis doloribus
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil quia soluta facere consequatur bla
</p>
```

Vamos a instalar yaml-front-matter para manejar la metadata y el body

```
composer require spatie/yaml-front-matter
```



```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ composer require spatie/yaml-front-matter
./composer.json has been updated
Running composer update spatie/yaml-front-matter
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking spatie/yaml-front-matter (2.0.7)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing spatie/yaml-front-matter (2.0.7): Extracting archive
Package fruitcake/laravel-cors is abandoned, you should avoid using it. No replacement was sug
gested.
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mailer in
stead.
Generating optimized autoload files
```

Se agregan variables, un constructor y se modifican los metodos all y find del post.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\ModelNotFoundException;
use Illuminate\Support\Facades\File;
use Illuminate\Support\HigherOrderCollectionProxy;
use Spatie\YamlFrontMatter\YamlFrontMatter;
class Post
{
    public $title;
    public $excerpt;
    public $date;
    public $body;
    public $slug;
    public function __construct($title, $excerpt, $date, $body, $slug)
    {
        $this->title = $title;
        $this->excerpt = $excerpt;
        $this->date = $date;
        $this->body = $body;
        $this->slug = $slug;
    }

    public static function all()
    {
        return collect(File::files(resource_path("posts")))
        ->map(fn($file) => YamlFrontMatter::parseFile($file))
        ->map(fn($document) => new Post(
            $document->title,
            $document->excerpt,
            $document->date,
            $document->body(),
            $document->slug
        ));
    }

    public static function find($slug){
        return static::all()->firstWhere("slug", $slug);
    }
}
```

El web.php donde van nuestras rutas queda de la siguiente manera

```
<?php
use Illuminate\Support\Facades\Route;
use Spatie\YamlFrontMatter\YamlFrontMatter;
use App\Models\Post;

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

Route::get('posts/{post}', function ($slug) {
    return view('post', [
        'post' => Post::find($slug)
    ]);
})->where('post', '[A-z_\-]+');
```

El posts.blade.php que carga nuestros posts, siendo este el home basicamente quedaria así

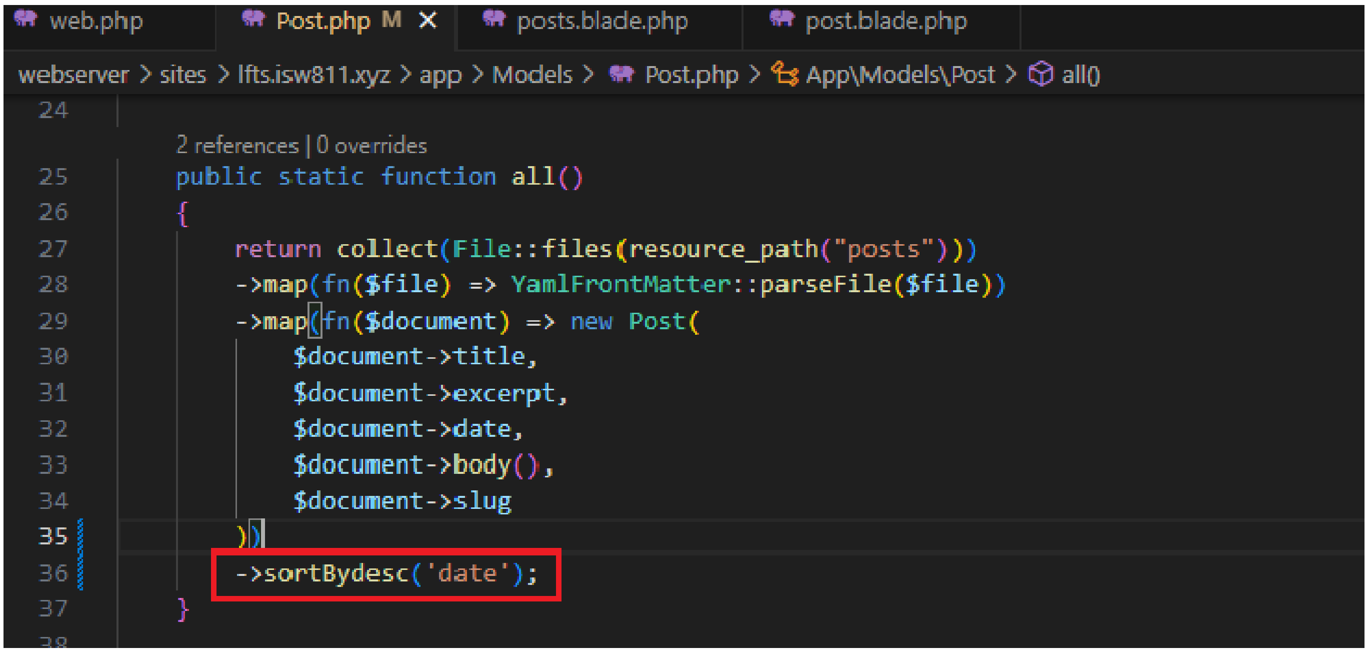
```
<!DOCTYPE html>

    <title>My blog</title>
    <link rel="stylesheet" href="/app.css">

<body>
    <?php foreach ($posts as $post) : ?>

        <article>
            <h1>
                <a href="/posts/<?= $post->slug; ?>">
                    <?= $post->title; ?>
                </a>
            </h1>

            <div>
                <?= $post->excerpt;?>
            </div>
        </article>

    <?php endforeach; ?>

</body>
```

Y para que cargue cada uno de los posts selecionados individualmente debemos tener nustro codigo en post.blade.php de la siguiente manera

```
<!DOCTYPE html>

    <title>My blog</title>
    <link rel="stylesheet" href="/app.css">

<body>
    <?php foreach ($posts as $post) : ?>

        <article>
            <h1>
                <a href="/posts/<?= $post->slug; ?>">
                    <?= $post->title; ?>
                </a>
            </h1>

            <div>
                <?= $post->excerpt;?>
            </div>
        </article>

    <?php endforeach; ?>

</body>
```

Collection Sorting and Caching Refresher

Ahora vamos a acomodar los post por fecha de manera descendente y los vamos a guadar en la cache para que no tenga que cargar cada vez que se accede a la pagina


web.php
Post.php M
posts.blade.php
post.blade.php
webserver > sites > lfts.isw811.xyz > app > Models > Post.php > App\Models\Post > all()
2 references | 0 overrides

```
public static function all()
{
    return collect(File::files(resource_path("posts")))
    ->map(fn($file) => YamlFrontMatter::parseFile($file))
    ->map(fn($document) => new Post(
        $document->title,
        $document->excerpt,
        $document->date,
        $document->body(),
        $document->slug
    ))
    ->sortBydesc('date');
}
```

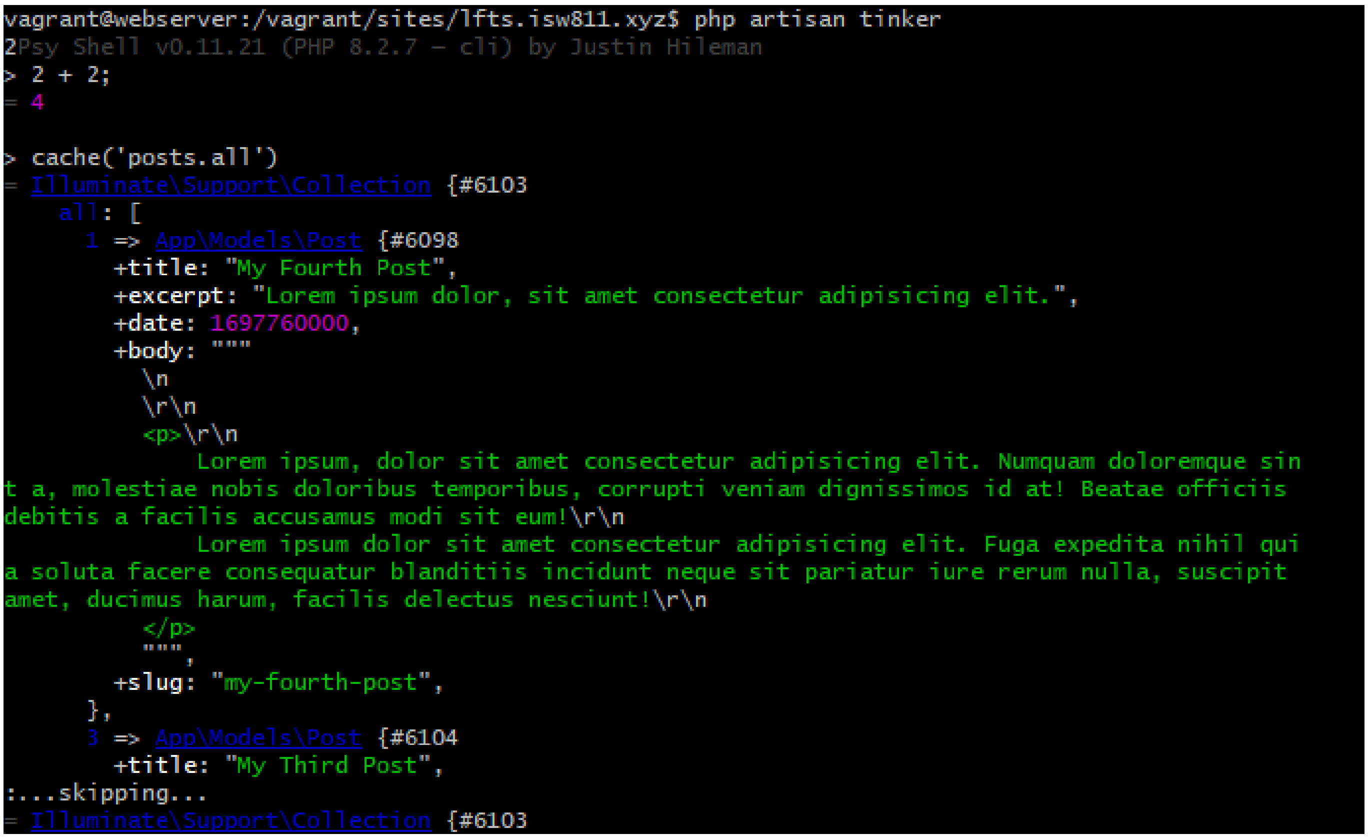
```
->sortBydesc('date');
```

Y para guardar en cache los post seria colocar todo el metodo all de la siguiente manera

```
  public static function all()
    {
        return cache()->rememberForever("posts.all", function () {
            return collect(File::files(resource_path("posts")))
            ->map(fn($file) => YamlFrontMatter::parseFile($file))
            ->map(fn($document) => new Post(
                $document->title,
                $document->excerpt,
                $document->date,
                $document->body(),
                $document->slug
            ))
            ->sortBydesc('date');
        });
    }
```

Para validar que si guarda los post en cache podemos acceder a ella por medio del siguente comando

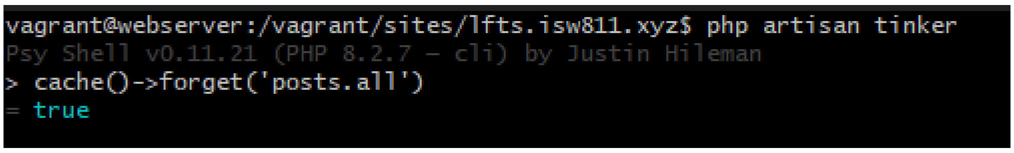
```
php artisan tinker
```



```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan tinker
2Psy Shell v0.11.21 (PHP 8.2.7 — cli) by Justin Hileman
> 2 + 2;
= 4

> cache('posts.all')
= Illuminate\Support\Collection {#6103
    all: [
      1 => App\Models\Post {#6098
        +title: "My Fourth Post",
        +excerpt: "Lorem ipsum dolor, sit amet consectetur adipisicing elit.",
        +date: 1697760000,
        +body: """
          \n
          \r\n
          <p>\r\n
              Lorem ipsum, dolor sit amet consectetur adipisicing elit. Numquam doloremque sin
t a, molestiae nobis doloribus temporibus, corrupti veniam dignissimos id at! Beatae officiis
debitis a facilis accusamus modi sit eum!\r\n
              Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita nihil qui
a soluta facere consequatur blanditiis incidunt neque sit pariatur iure rerum nulla, suscipit
amet, ducimus harum, facilis delectus nesciunt!\r\n
          </p>
          """,
        +slug: "my-fourth-post",
      },
      3 => App\Models\Post {#6104
        +title: "My Third Post",
:...skipping...
= Illuminate\Support\Collection {#6103
```

Para eliminar la cache ejecutamos el comando

```
cache()->forget('posts.all')
```



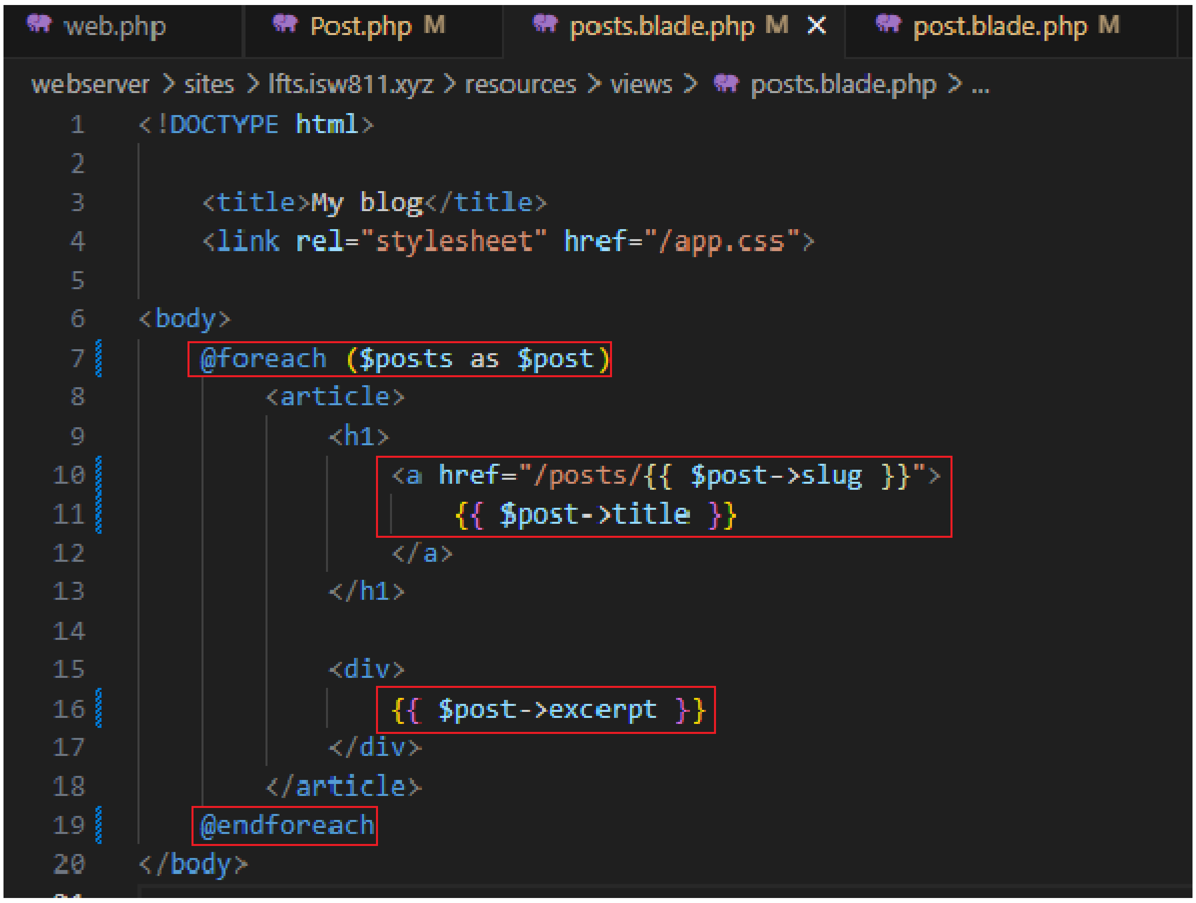
```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan tinker
Psy Shell v0.11.21 (PHP 8.2.7 — cli) by Justin Hileman
> cache()->forget('posts.all')
= true
```

Blade: The Absolute Basics

Blade es especifico para las vistas, nos facilita el codigo php dentro de ellas

```
Antes   -   <?= $post->title; ?>
Despues -   {{ $post->title }}
```

Asi quedaría la pagina de posts.blade.php y una vez aprendido esto podemos crear Layouts



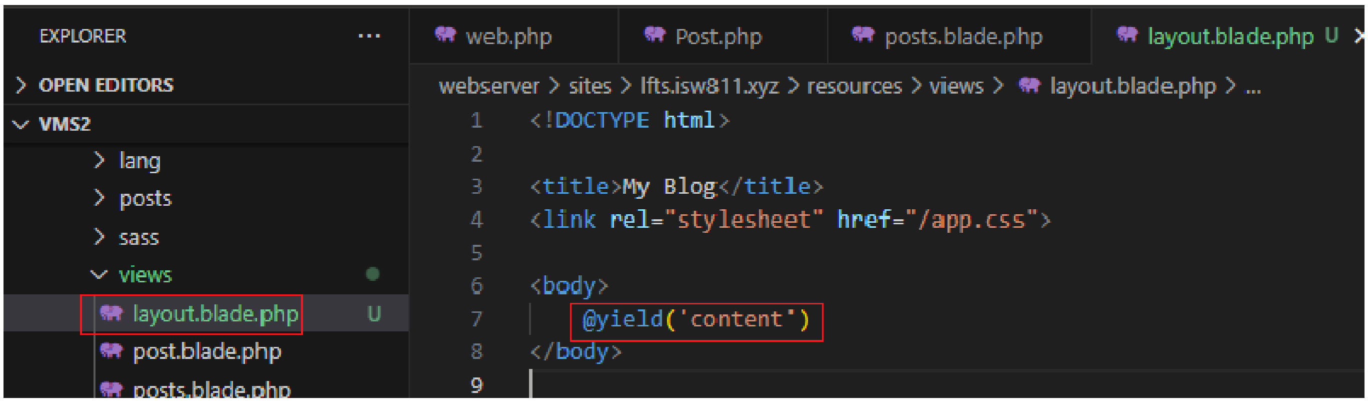
```
web.php   Post.php M   posts.blade.php M ×   post.blade.php M
webserver > sites > lfts.isw811.xyz > resources > views > posts.blade.php > ...
<!DOCTYPE html>

    <title>My blog</title>
    <link rel="stylesheet" href="/app.css">

<body>
    @foreach ($posts as $post)
        <article>
            <h1>
                <a href="/posts/{{ $post->slug }}">
                    {{ $post->title }}
                </a>
            </h1>

            <div>
                {{ $post->excerpt }}
            </div>
        </article>
    @endforeach
</body>
```

Blade: The Absolute Basics

Vamos a crear layouts para poder utilizarlos en nuestras vistas y asi no tener que hacer imports en cada view por cada nuevo archivo que vayamos a crear.

Creamos una nueva vista llamada layout.blade.php con el codigo

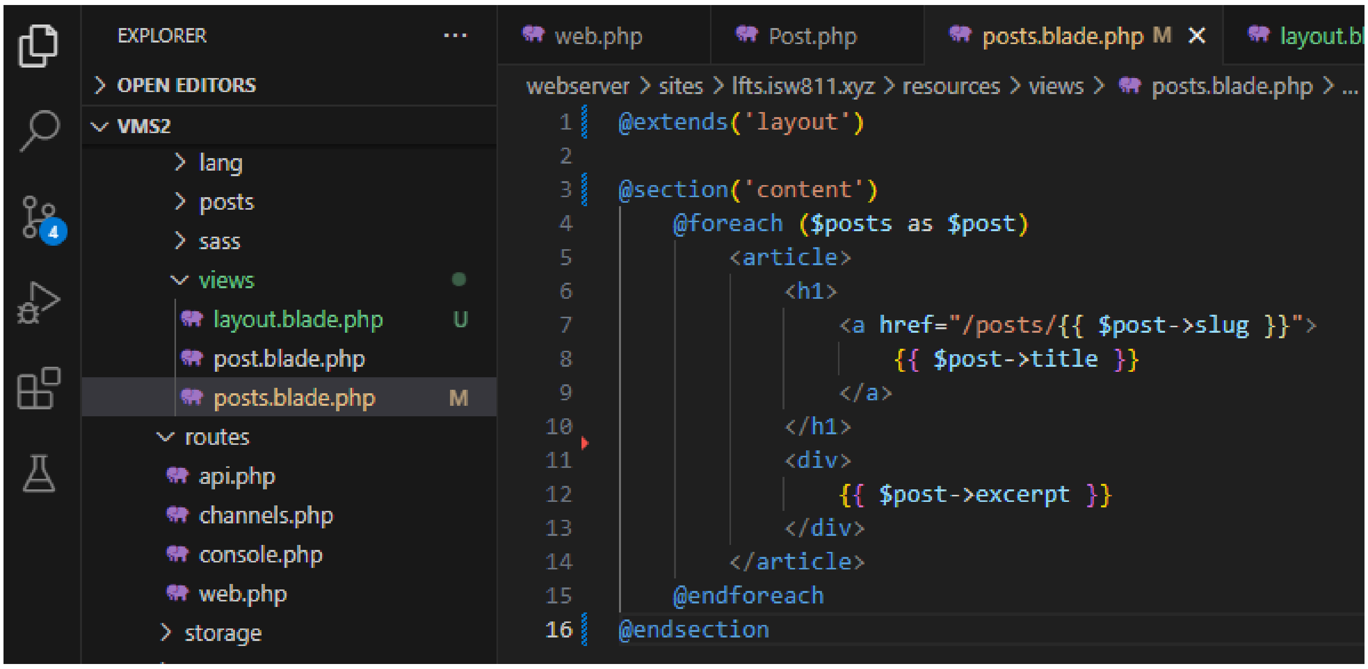
```
@yield('content')
```



```
EXPLORER
> OPEN EDITORS
v VMS2
    > lang
    > posts
    > sass
    v views
      layout.blade.php   U
      post.blade.php
      posts.blade.php
```

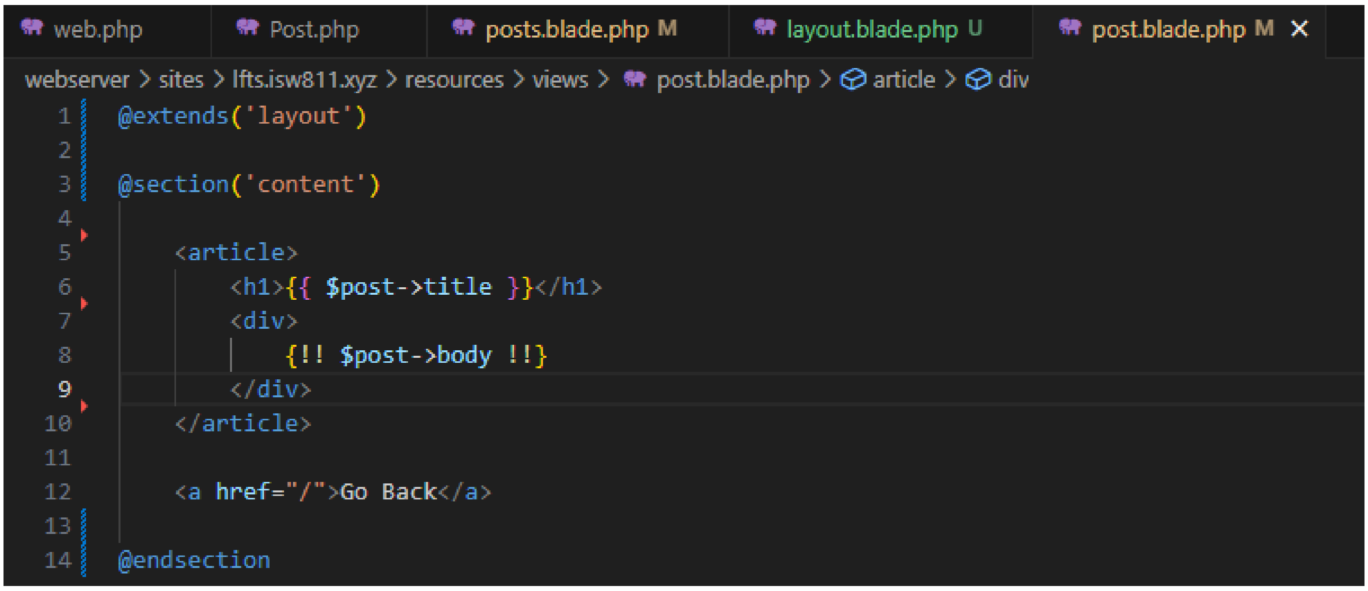
```
web.php   Post.php   posts.blade.php   layout.blade.php U
webserver > sites > lfts.isw811.xyz > resources > views > layout.blade.php > ...
<!DOCTYPE html>

<title>My Blog</title>
<link rel="stylesheet" href="/app.css">

<body>
    @yield('content')
</body>
```

Al crear este layout, podemos modificar nuestro codigo del archivo posts.blade.php, quedando sin las etiquetas html y links.



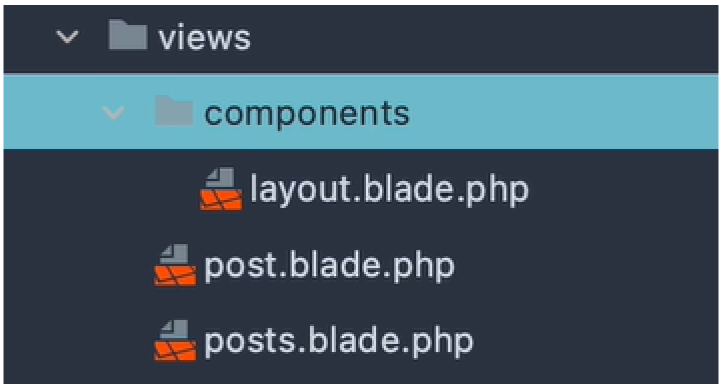
```
@extends('layout')

@section('content')
    @foreach ($posts as $post)
        <article>
            <h1>
                <a href="/posts/{{ $post->slug }}">
                    {{ $post->title }}
                </a>
            </h1>
            <div>
                {{ $post->excerpt }}
            </div>
        </article>
    @endforeach
@endsection
```

El post.blade.php igualmente


web.php
Post.php
posts.blade.php M
layout.blade.php U
post.blade.php M
webserver > sites > lfts.isw811.xyz > resources > views > post.blade.php > article > div

```
@extends('layout')

@section('content')

    <article>
        <h1>{{ $post->title }}</h1>
        <div>
            {!! $post->body !!}
        </div>
    </article>

    <a href="/">Go Back</a>

@endsection
```

Una manera alternativa de crear layouts es creando dentro de la carpeta views una subcarpeta llamada components, ahi metemos el archivo layout,blade.php y realizamos algunos cambios de codigo



A Few Tweaks and Considerations

web.php

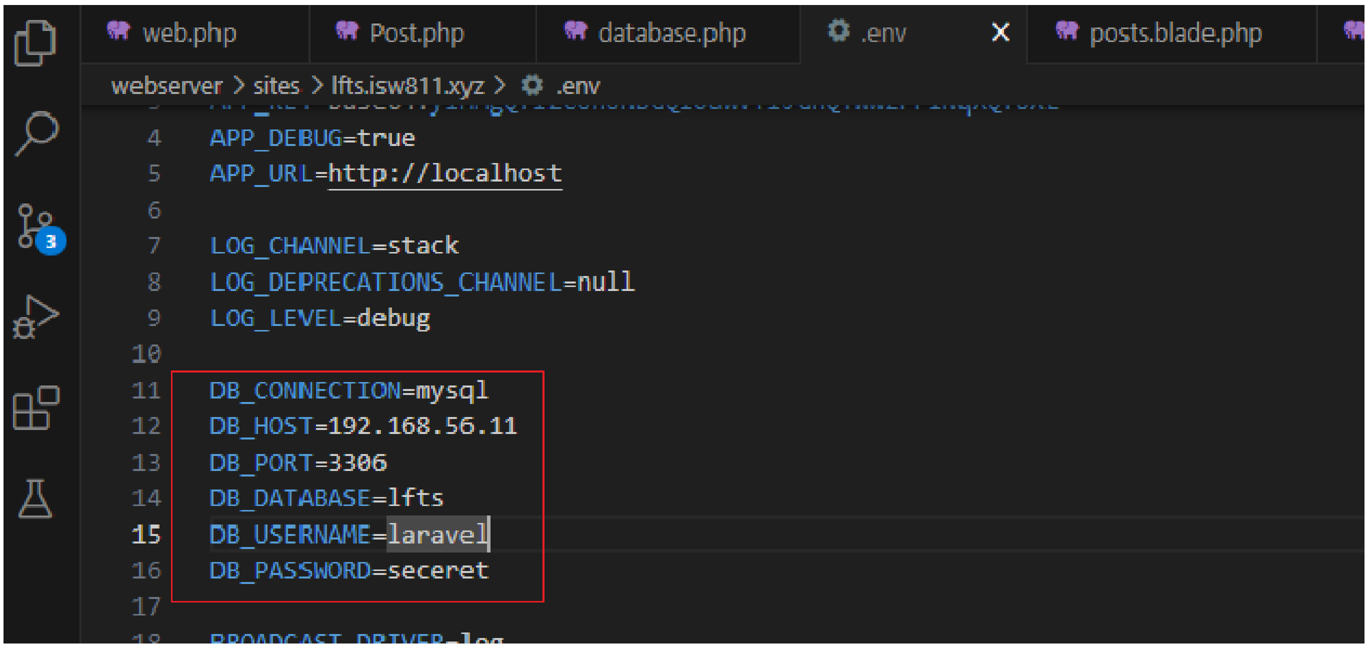
```php
<?php
use Illuminate\Support\Facades\Route;
use Spatie\YamlFrontMatter\YamlFrontMatter;
use App\Models\Post;

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

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

post.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\ModelNotFoundException;
use Illuminate\Support\Facades\File;
use Spatie\YamlFrontMatter\YamlFrontMatter;
class Post
{
    public $title;
    public $excerpt;
    public $date;
    public $body;
    public $slug;
    public function __construct($title, $excerpt, $date, $body, $slug)
    {
        $this->title = $title;
        $this->excerpt = $excerpt;
        $this->date = $date;
        $this->body = $body;
        $this->slug = $slug;
    }

    public static function all()
    {
        return cache()->rememberForever("posts.all", function () {
            return collect(File::files(resource_path("posts")))
            ->map(fn($file) => YamlFrontMatter::parseFile($file))
            ->map(fn($document) => new Post(
                $document->title,
                $document->excerpt,
                $document->date,
                $document->body(),
                $document->slug
            ))
            ->sortBydesc('date');
        });
    }

    public static function find($slug)
    {
        return static::all()->firstWhere("slug", $slug);
    }

    public static function findOrFail($slug)
    {
        $post = static::find($slug);

        if (! $post) {
        throw new ModelNotFoundException();
        }
        return $post;
    }
}
```

Environment Files and Database Connections

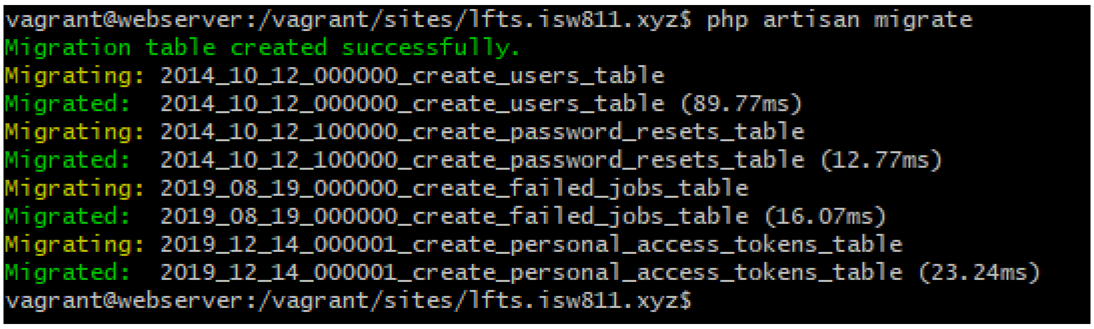
Realizamos la migracion de la BD, configurando el .env



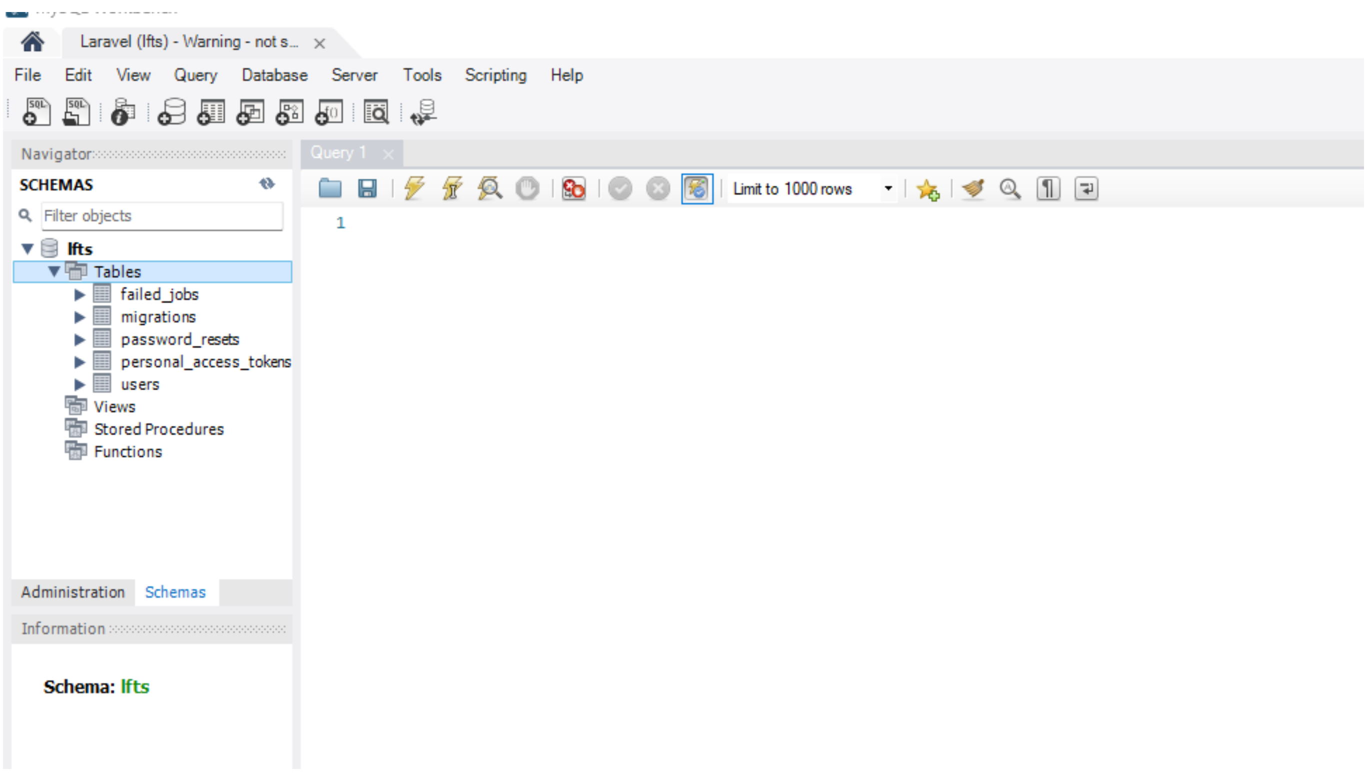
```
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=192.168.56.11
DB_PORT=3306
DB_DATABASE=lfts
DB_USERNAME=laravel
DB_PASSWORD=seceret

BROADCAST_DRIVER=log
```

Y ejecutando el comando

```
php artisan migrate
```



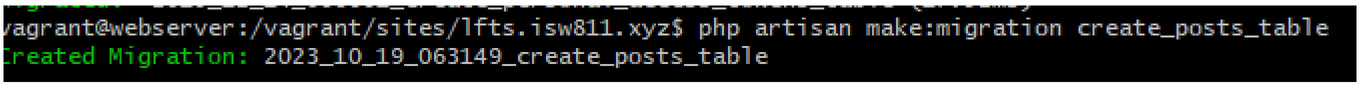
```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (89.77ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (12.77ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (16.07ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (23.24ms)
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$
```



Make a Post Model and Migration

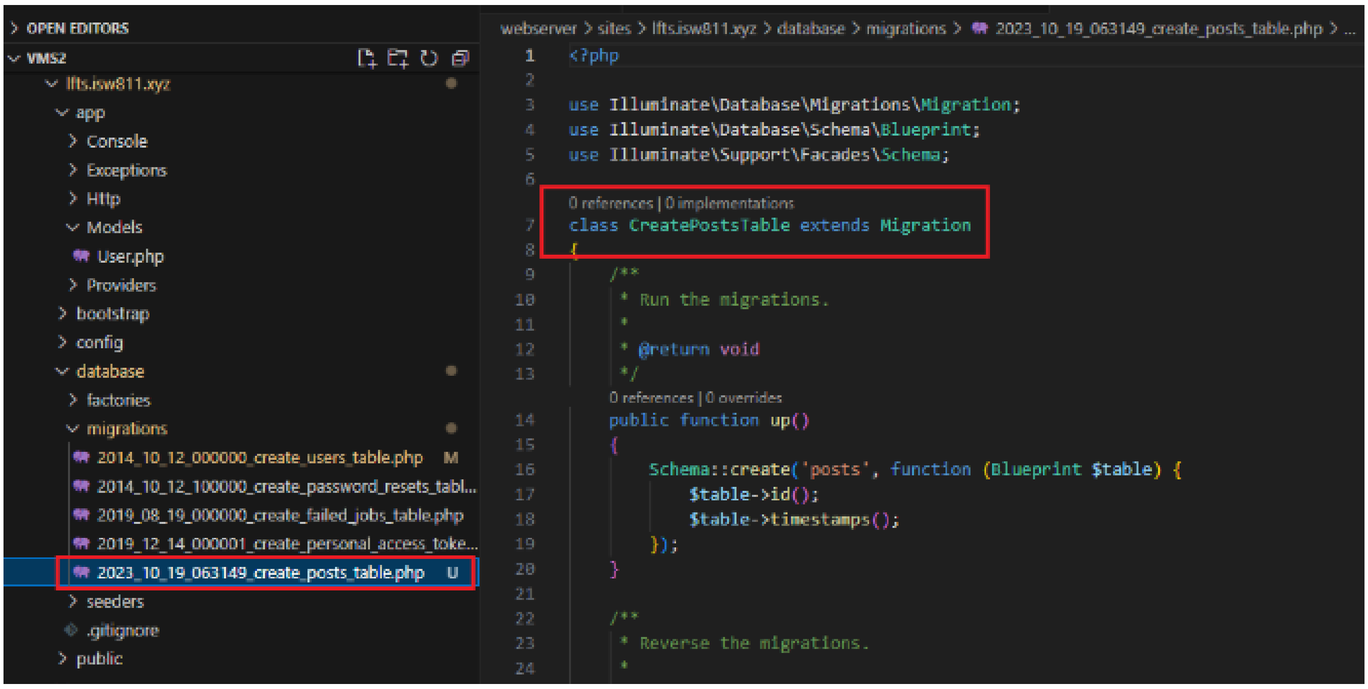
Borramos el modelo Post.php porque ahora vamos a transicionar a un post con Eloquent Model.

Creamos la migracion de los posts



```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan make:migration create_posts_table
Created Migration: 2023_10_19_063149_create_posts_table
```

```
php artisan make:migration create_posts_table
```

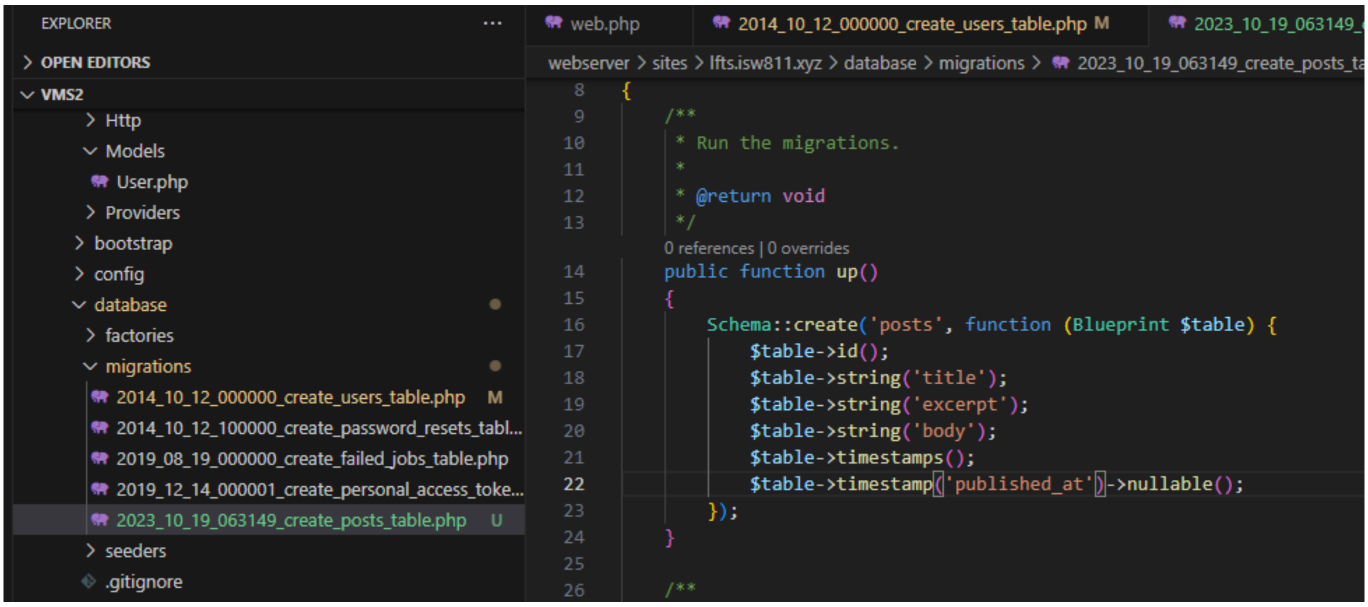


```
webserver > sites > lfts.isw811.xyz > database > migrations > 2023_10_19_063149_create_posts_table.php > ...
```

```
> OPEN EDITORS
v VMS2
  v lfts.isw811.xyz
    v app
      > Console
      > Exceptions
      > Http
      v Models
          User.php
      > Providers
    > bootstrap
    > config
    v database
      > factories
      v migrations
          2014_10_12_000000_create_users_table.php  M
          2014_10_12_100000_create_password_resets_tabl...
          2019_08_19_000000_create_failed_jobs_table.php
          2019_12_14_000001_create_personal_access_toke...
          2023_10_19_063149_create_posts_table.php  U
      > seeders
        .gitignore
    > public
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

0 references | 0 implementations
class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    0 references | 0 overrides
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
```

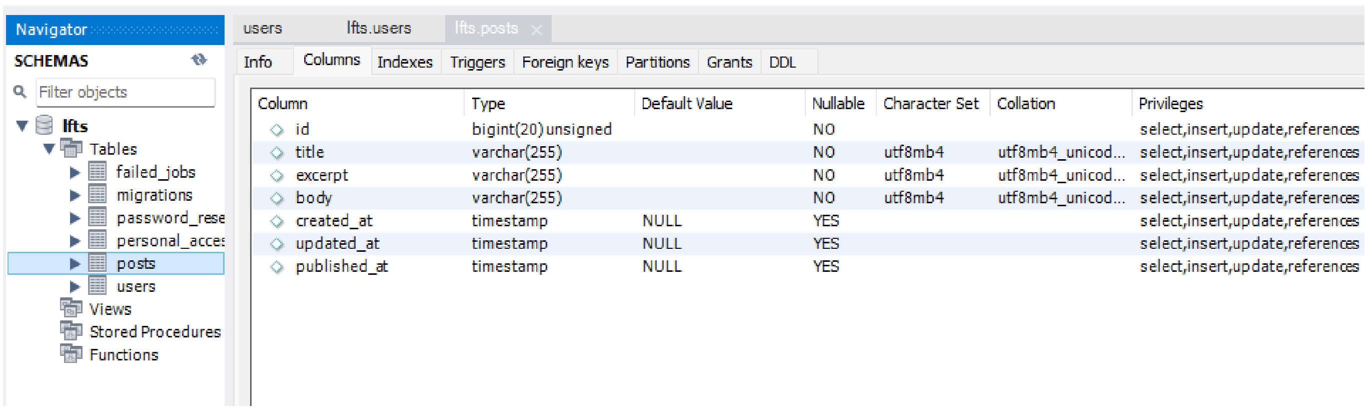
Borramos la carpeta con los post creados manualmente y modificamos la migracion de create_posts_table para que cuente con los campos que necesitamos



```
web.php   2014_10_12_000000_create_users_table.php M   2023_10_19_063149_
webserver > sites > lfts.isw811.xyz > database > migrations > 2023_10_19_063149_create_posts_ta
```

```
EXPLORER
> OPEN EDITORS
v VMS2
    > Http
    v Models
        User.php
    > Providers
  > bootstrap
  > config
  v database
    > factories
    v migrations
        2014_10_12_000000_create_users_table.php  M
        2014_10_12_100000_create_password_resets_tabl...
        2019_08_19_000000_create_failed_jobs_table.php
        2019_12_14_000001_create_personal_access_toke...
        2023_10_19_063149_create_posts_table.php  U
    > seeders
      .gitignore
```

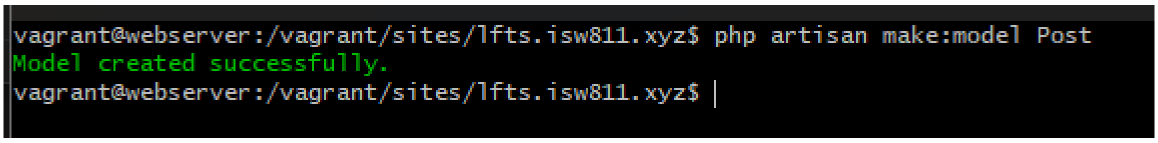
```php
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    0 references | 0 overrides
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->string('excerpt');
            $table->string('body');
            $table->timestamps();
            $table->timestamp('published_at')->nullable();
        });
    }

    /**
```



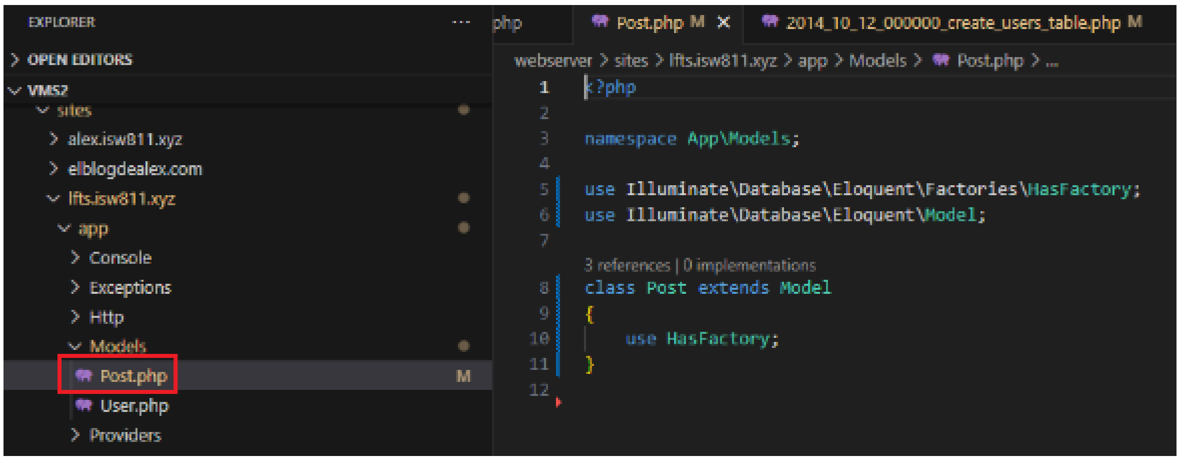
Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
id	bigint(20) unsigned		NO			select,insert,update,references
title	varchar(255)		NO	utf8mb4	utf8mb4_unicod...	select,insert,update,references
excerpt	varchar(255)		NO	utf8mb4	utf8mb4_unicod...	select,insert,update,references
body	varchar(255)		NO	utf8mb4	utf8mb4_unicod...	select,insert,update,references
created_at	timestamp	NULL	YES			select,insert,update,references
updated_at	timestamp	NULL	YES			select,insert,update,references
published_at	timestamp	NULL	YES			select,insert,update,references

Debemos crear un Eloquent Model

```
php artisan make:model Post
```

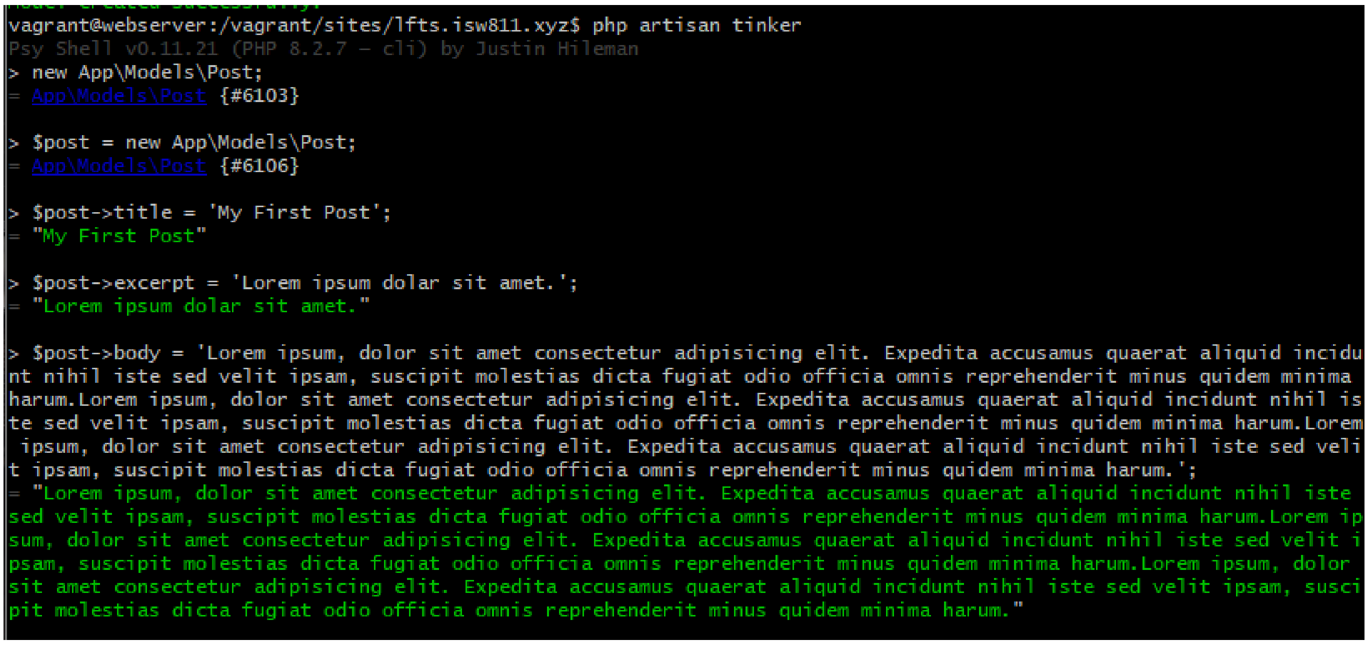


```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan make:model Post
Model created successfully.
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ |
```



```
<?php

namespace App\Models;

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

class Post extends Model
{
    use HasFactory;
}
```

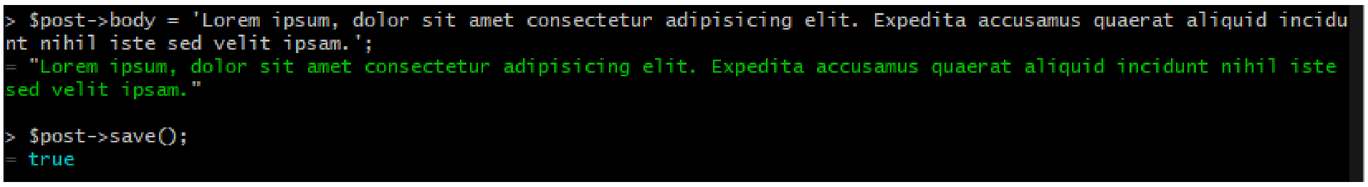
Con php atrtisan tinker creamos un post



```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan tinker
Psy Shell v0.11.21 (PHP 8.2.7 — cli) by Justin Hileman
> new App\Models\Post;
= App\Models\Post {#6103}

> $post = new App\Models\Post;
= App\Models\Post {#6106}

> $post->title = 'My First Post';
= "My First Post"

> $post->excerpt = 'Lorem ipsum dolar sit amet.';
= "Lorem ipsum dolar sit amet."

> $post->body = 'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidu
nt nihil iste sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima
harum.Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil is
te sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima harum.Lorem
 ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed veli
t ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima harum.';
= "Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste
sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima harum.Lorem ip
sum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit i
psam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima harum.Lorem ipsum, dolor
sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit ipsam, susci
pit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima harum."
```

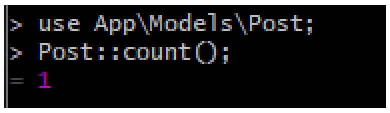
```
php artisan tinker

new App\Models\Post;

$post = new App\Models\Post;

$post->slug = 'my-first-post';

$post->title = 'My First Post';

$post->excerpt = 'Lorem ipsum dolar sit amet consectetur adipisicing elit.';
```

```
$post->body = 'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima.odio officia omnis reprehenderit minus';

$post->save();
```



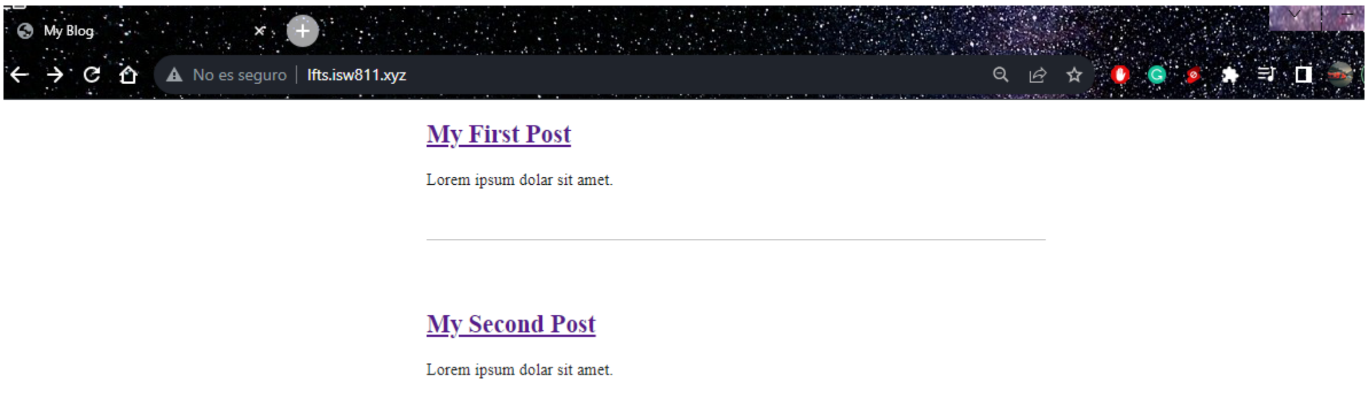
```
> $post->body = 'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidu
nt nihil iste sed velit ipsam.';
= "Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste
sed velit ipsam."

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

```
use App\Models\Post;
Post::count();
```



```
> use App\Models\Post;
> Post::count();
= 1
```

Asi se va viendo el site

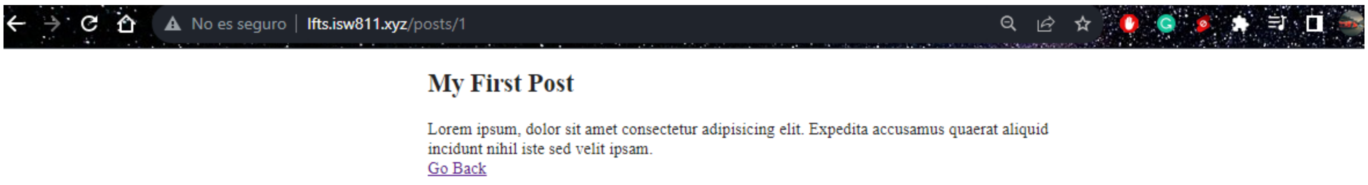


My First Post

Lorem ipsum dolar sit amet.

My Second Post

Lorem ipsum dolar sit amet.



My First Post

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit ipsam.
Go Back

Eloquent Updates and HTML Escaping

Actualizacion de blog post existente, cambiamos el body a html

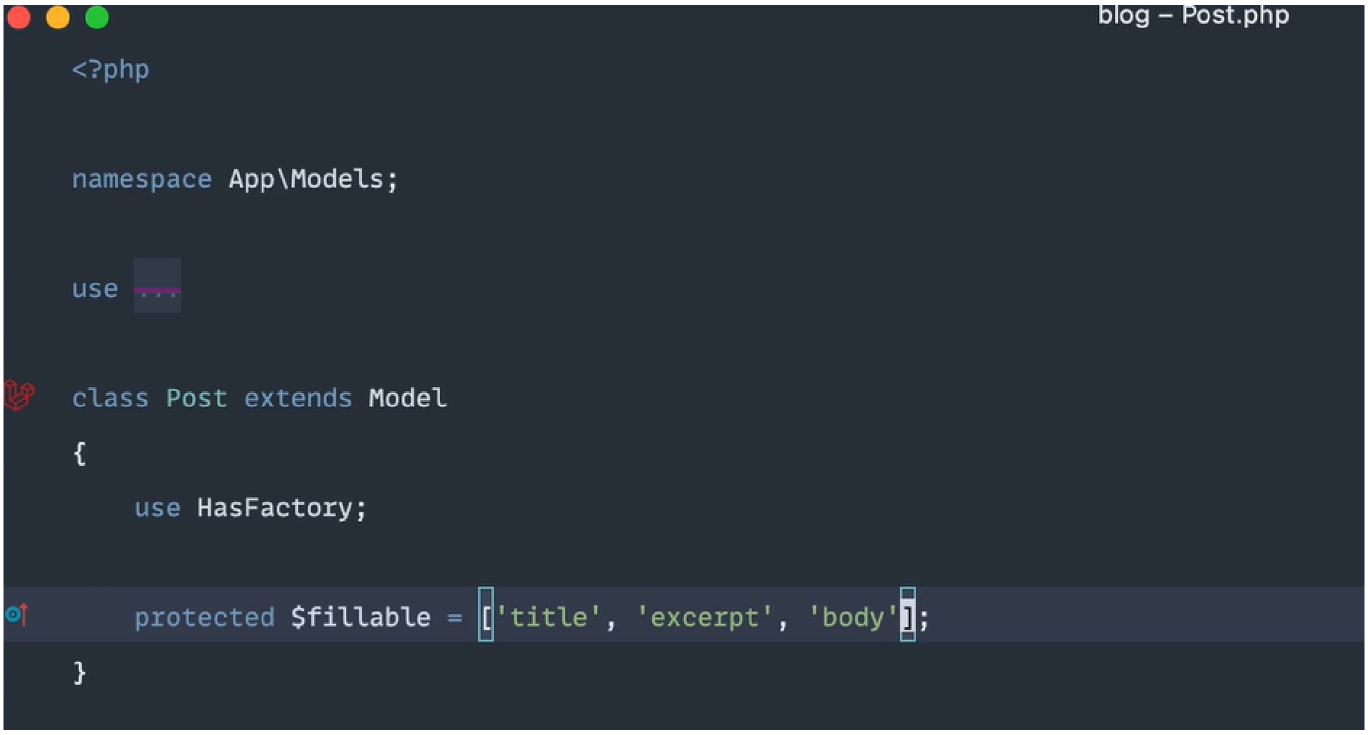
```
php artisan tinker
$post = App\Models\Post::first();
$post->body;
$post->body = '<p>' . $post->body . '</p>'
$post
$post->save();

$post = App\Models\Post::find(2); //Para actualizar los demas por id
```

3 Ways to Mitigate Mass Assignment Vulnerabilities

1- Para asignar en masa un post como en el codigo de abajo, debemos tener en cuenta la seguridad, por eso solo vamos a permitir que se asignen los datos deseados en el archivo Post.php

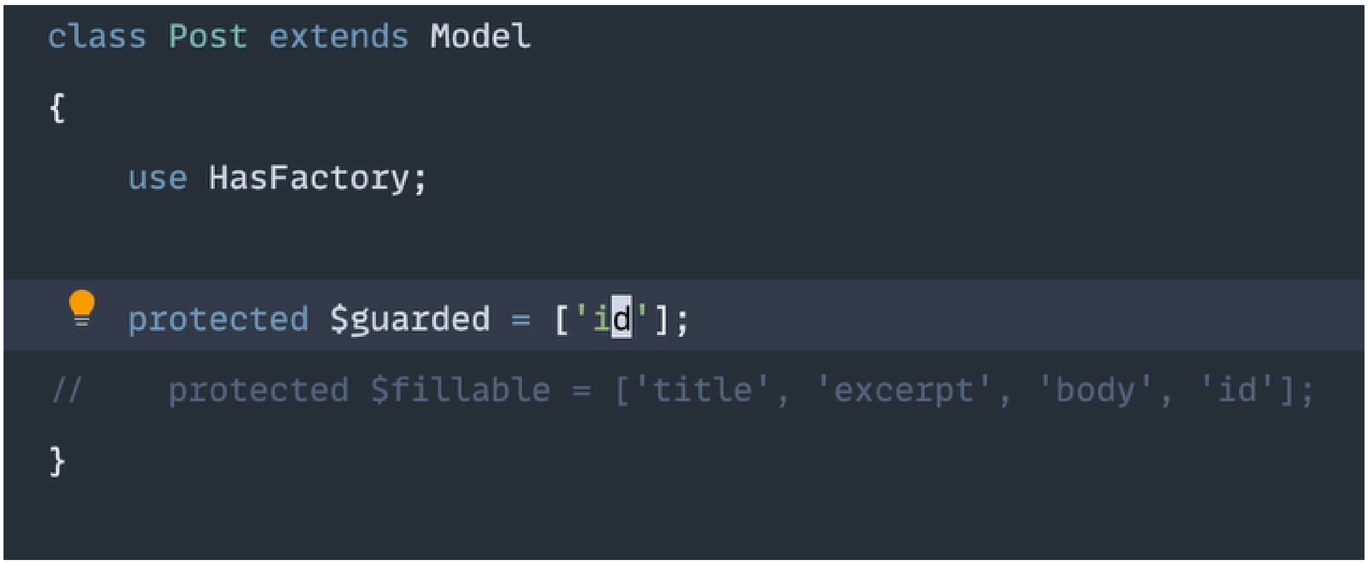
```
Post::create(['title' => 'My Fourth Post', 'excerpt' => 'Lorem ipsum dolar sit amet consectetur adipisicing elit.', 'body' => 'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem minima.odio officia omnis reprehenderit minus quidem minima.' ]);
```

Archivo Post.php, si se agregan mas datos en el inster masivo entonces laravel lo rechaza



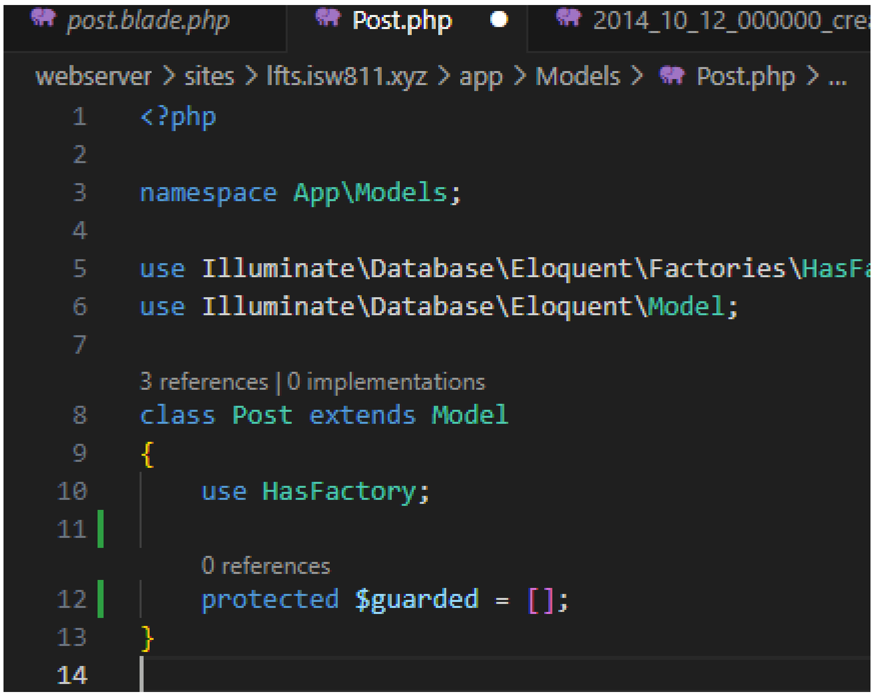
```
<?php

namespace App\Models;

use ...

class Post extends Model
{
    use HasFactory;

    protected $fillable = ['title', 'excerpt', 'body'];
}
```

2- Por medio de protected podemos evitar que se agreguen datos indeseados como puede ser el id del post, entonces a la hora de hacer el mass asign con un id, el sistema si crea el post pero ignorando el id.



```
class Post extends Model
{
    use HasFactory;

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

3- La tarcera opcion es nunca permitir el mass asign, simplemente dejando el array vacio



post.blade.php | Post.php | 2014_10_12_000000_cre

webserver > sites > lfts.isw811.xyz > app > Models > Post.php > ...

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFa
use Illuminate\Database\Eloquent\Model;

3 references | 0 implementations
class Post extends Model
{
    use HasFactory;

    0 references
    protected $guarded = [];
}

```

Route Model Binding

Le agregamos al archivo de migration, una variable slug

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

Luego en nuestro archivo de rutas web.php modificamos el get de los posts quedando así

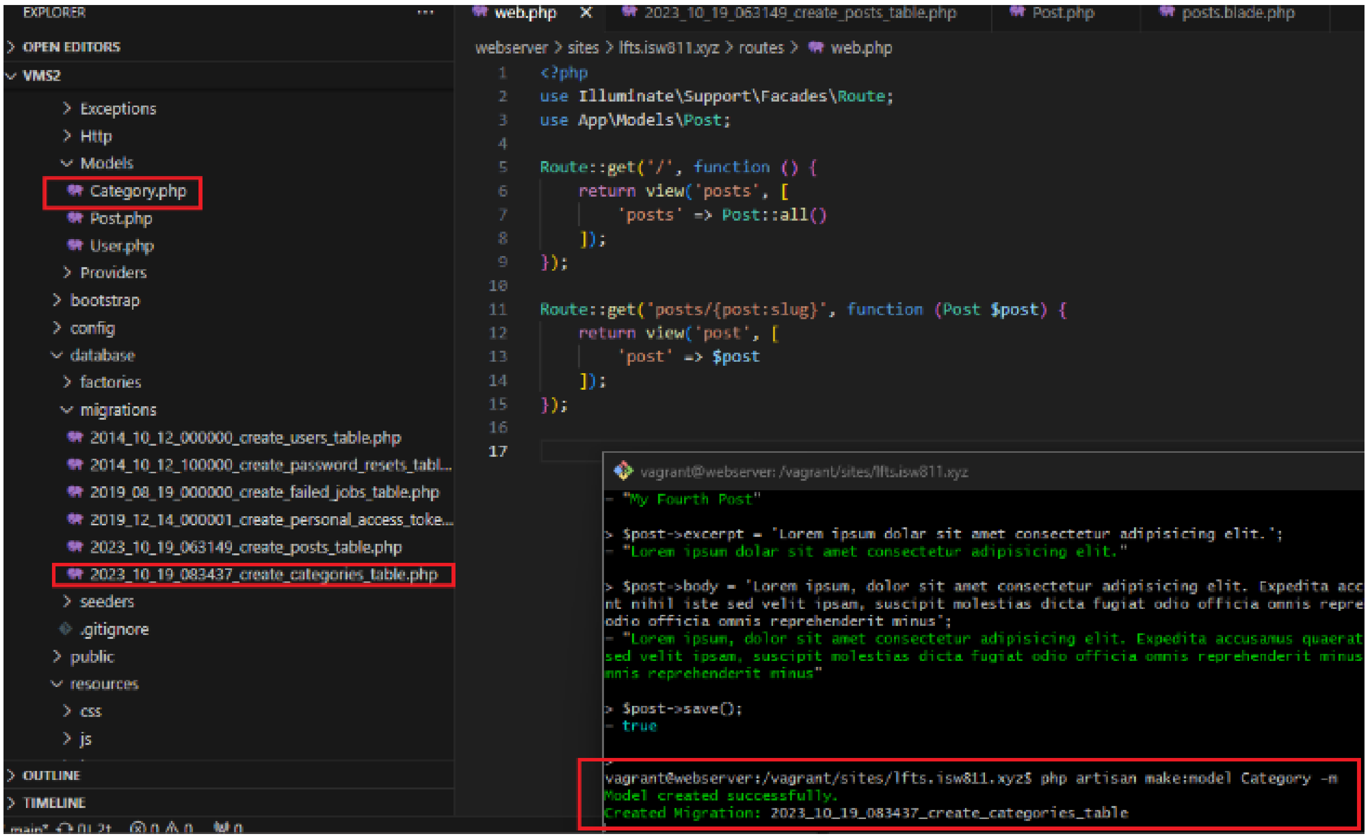
```
Route::get('posts/{post:slug}', function (Post $post) {
    return view('post', [
        'post' => $post
    ]);
});
```

Your First Eloquent Relationship

Vamos a agregar una categoria a cada post

Creacion de modelo y su migration

```
php artisan make:model Category -m
```


EXPLORER
OPEN EDITORS
VMS2
Exceptions
Http
Models
Category.php
Post.php
User.php
Providers
bootstrap
config
database
factories
migrations
2014_10_12_000000_create_users_table.php
2014_10_12_100000_create_password_resets_tabl...
2019_08_19_000000_create_failed_jobs_table.php
2019_12_14_000001_create_personal_access_toke...
2023_10_19_063149_create_posts_table.php
2023_10_19_083437_create_categories_table.php
seeders
.gitignore
public
resources
css
js
OUTLINE
TIMELINE
web.php
2023_10_19_063149_create_posts_table.php
Post.php
posts.blade.php
webserver > sites > lfts.isw811.xyz > routes > web.php
vagrant@webserver: /vagrant/sites/lfts.isw811.xyz

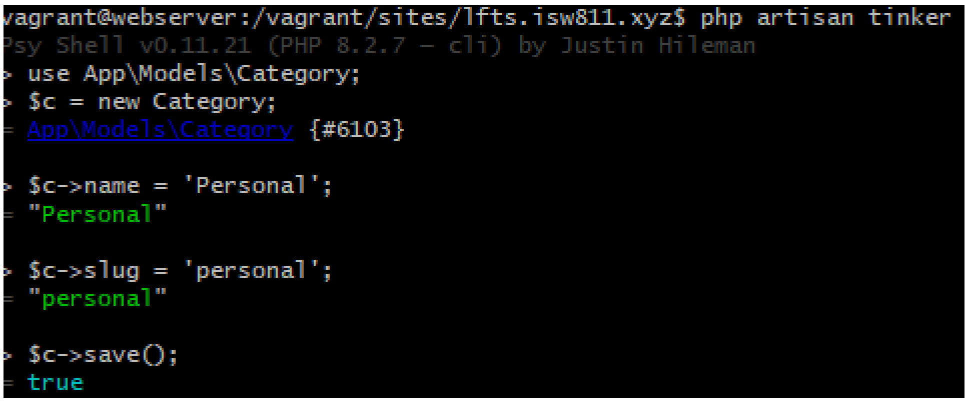
```
<?php
use Illuminate\Support\Facades\Route;
use App\Models\Post;

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

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

```
= "My Fourth Post"

> $post->excerpt = 'Lorem ipsum dolar sit amet consectetur adipisicing elit.';
= "Lorem ipsum dolar sit amet consectetur adipisicing elit."

> $post->body = 'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita acc
nt nihil iste sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis repre
odio officia omnis reprehenderit minus';
= "Lorem ipsum, dolor sit amet consectetur adipisicing elit. Expedita accusamus quaerat
sed velit ipsam, suscipit molestias dicta fugiat odio officia omnis reprehenderit minus
mnis reprehenderit minus"

> $post->save();
= true

vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan make:model Category -m
Model created successfully.
Created Migration: 2023_10_19_083437_create_categories_table
```

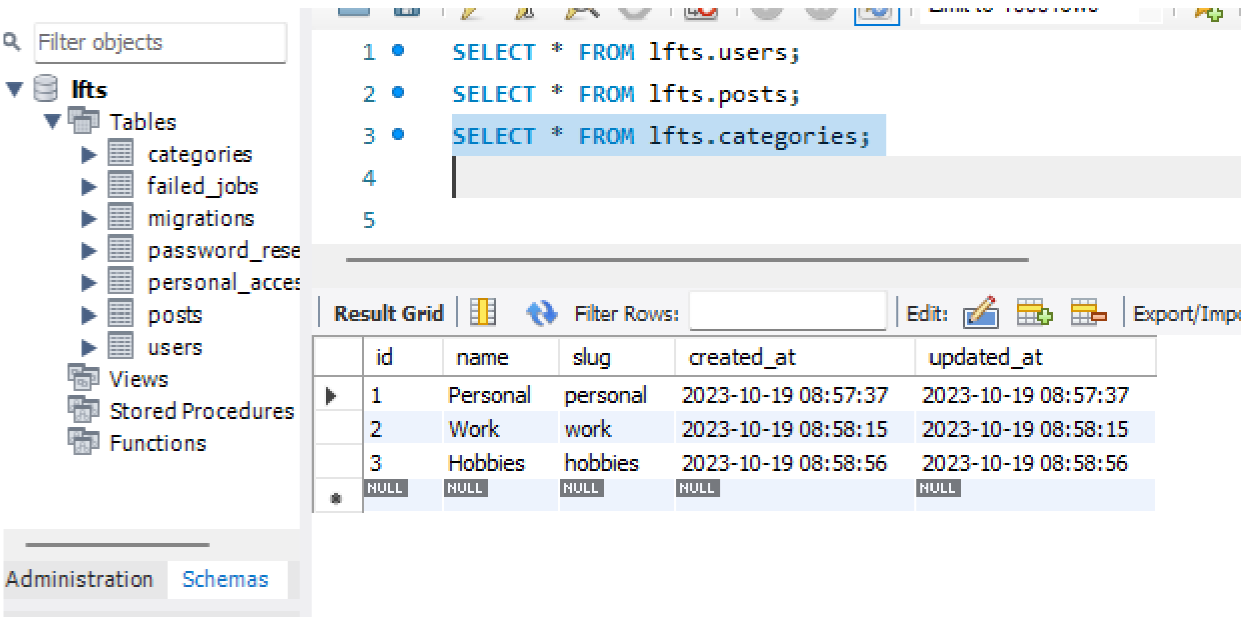
Creamos varias categorias despues de ralizar un php artisan migrate:fresh

```
php artisan tinker
use App\Models\Category;
```

```
$c = new Category;
$c->name = 'Personal';
$c->slug = 'personal';
$c->save();
```



```
agrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan tinker
sy Shell v0.11.21 (PHP 8.2.7 — cli) by Justin Hileman
 use App\Models\Category;
 $c = new Category;
 App\Models\Category {#6103}

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

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

 $c->save();
 true
```



```
SELECT * FROM lfts.users;
SELECT * FROM lfts.posts;
SELECT * FROM lfts.categories;
```

```
lfts
  Tables
    categories
    failed_jobs
    migrations
    password_rese
    personal_acces
    posts
    users
  Views
  Stored Procedures
  Functions
```

id	name	slug	created_at	updated_at
1	Personal	personal	2023-10-19 08:57:37	2023-10-19 08:57:37
2	Work	work	2023-10-19 08:58:15	2023-10-19 08:58:15
3	Hobbies	hobbies	2023-10-19 08:58:56	2023-10-19 08:58:56
NULL	NULL	NULL	NULL	NULL

Ahora vamos a crear posts

```
use App\Models\Post;
Post::create([
    'title' => 'My Hobby Post',
    'excerpt' => 'Excerpt for my post',
    'body' => 'Lorem ipsum dolar sit amet ameis og.',
    'slug' => 'my-hobby-post',
    'category_id' => 3
]);
```

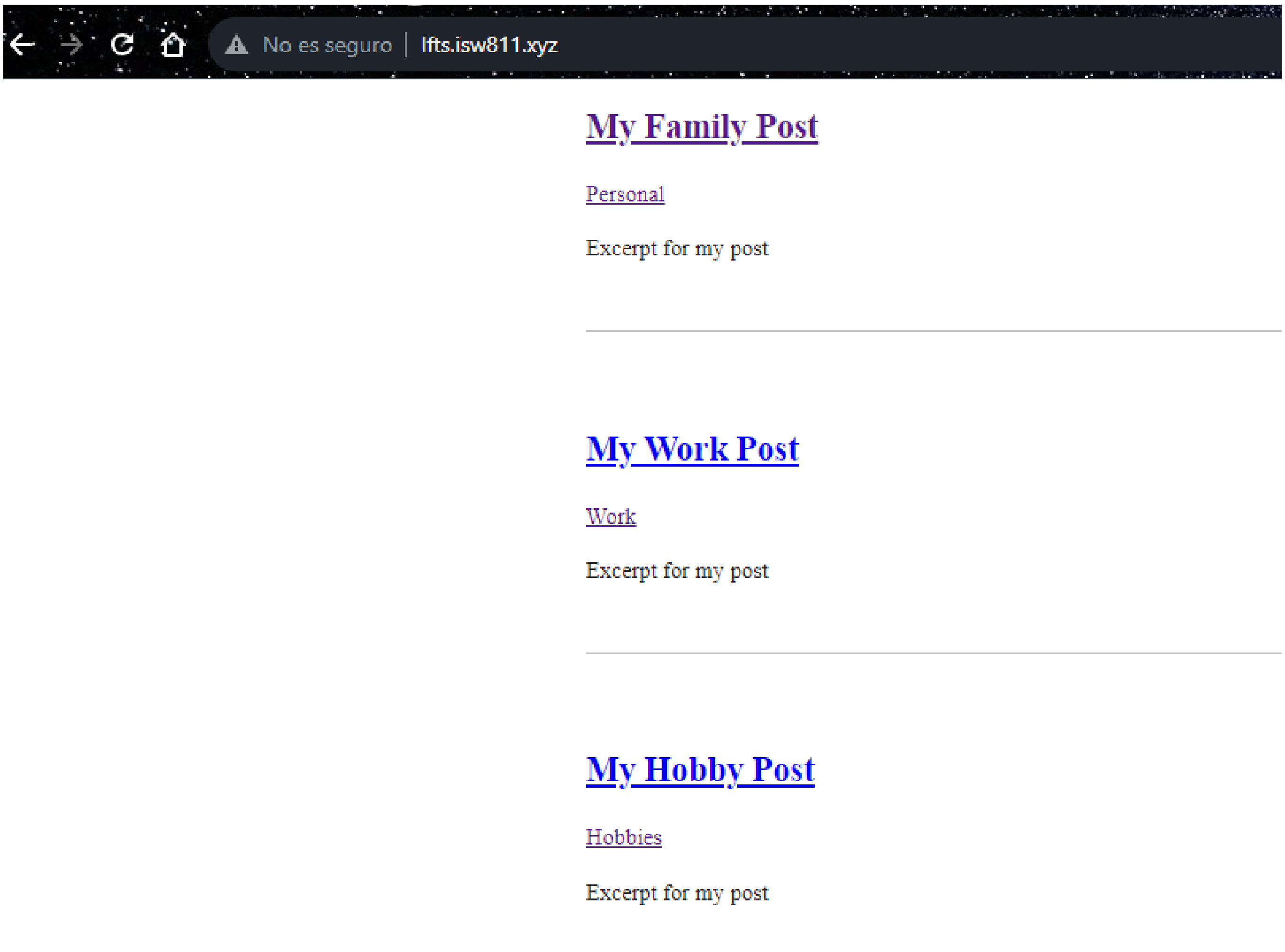
Nuestro Post.php se veria asi, lo que quiere decir que ahora tenemos nuestra primera Eloquent Relationship

```
<?php

namespace App\Models;
```

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

class Post extends Model
{
    use HasFactory;

    protected $guarded = [];
    public function category()
    {
        return $this->belongsTo(Category::class);
    }
}
```

Agregamos a nuestro post.blade.php y posts.blade.php el siguiente codigo para poder ver las categorias

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



My Family Post

Personal

Excerpt for my post

My Work Post

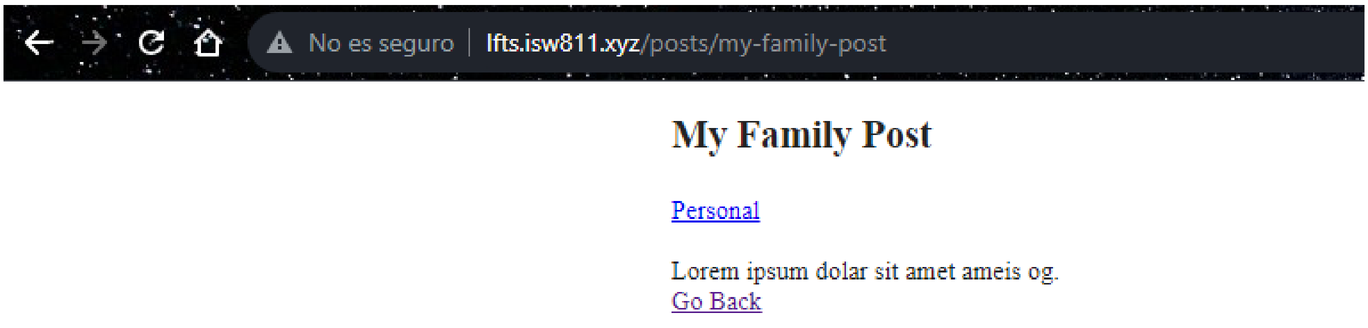
Work

Excerpt for my post

My Hobby Post

Hobbies

Excerpt for my post


No es seguro | lfts.isw811.xyz/posts/my-family-post

My Family Post

Personal

Lorem ipsum dolar sit amet ameis og.
Go Back

Show All Posts Associated With a Category

Agregamos un nuevo metodo a Web.php

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

Agregamos una nueva funcion a nuestro Category.php

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

En post.blade.php y posts.blade.php colocamos el link de la categoria de esta manera

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

Clockwork, and the N+1 Problem

Para no ejecutar un query cada vez que se carga una categoria vamos a resolverlo de esta manera

Para esto debemos modificar el get de los posts

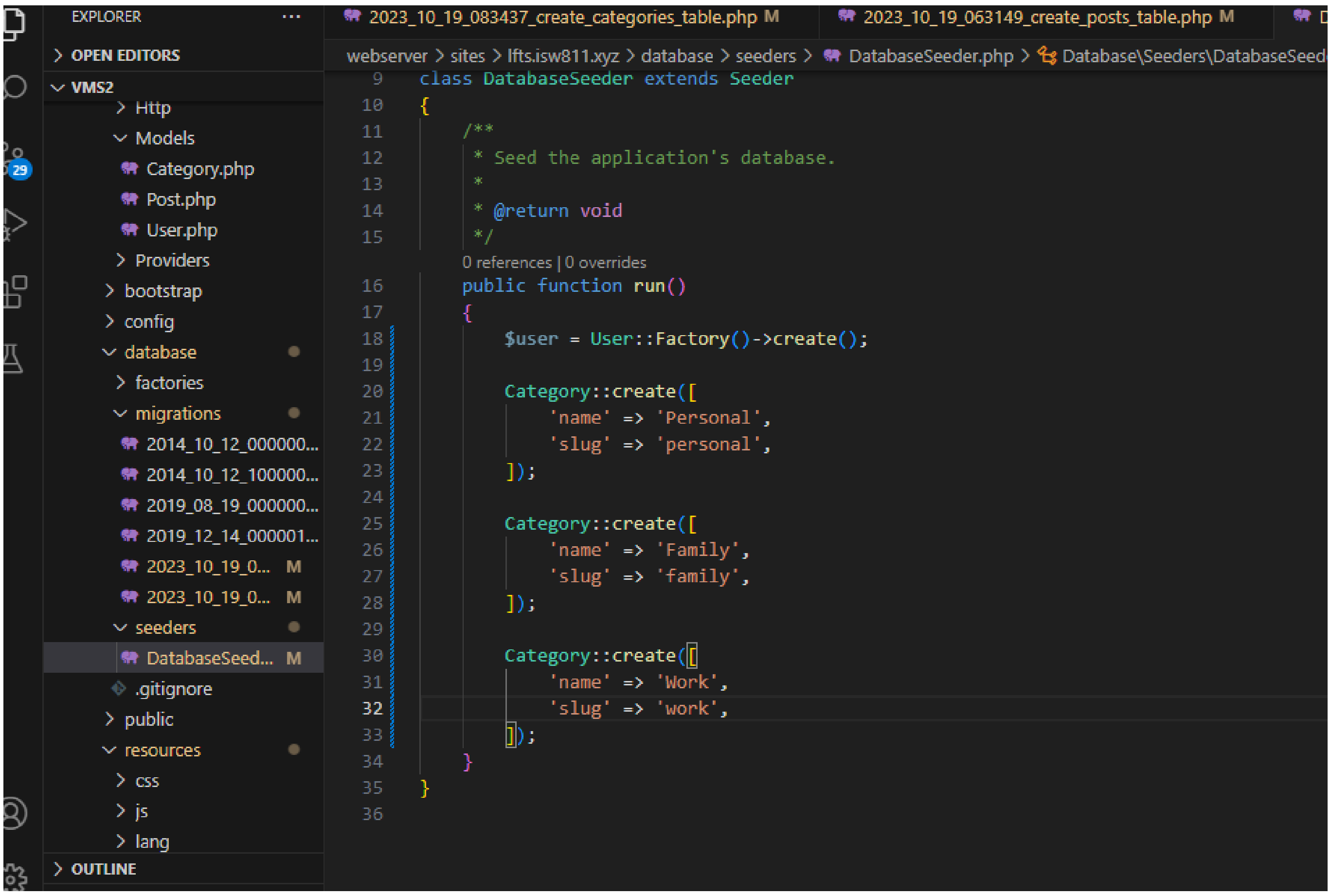
```
Route::get('/', function () {
    return view('posts', [
        'posts' => Post::with('category')->get()
    ]);
});
```

Database Seeding Saves Time

Agregamos un nuevo campo a la migration de posts

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

Accedemos al archivo de la carpeta seeder y creamos lo siguiente



```
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $user = User::Factory()->create();

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

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

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

Corremos el migrate seed para actualizar la BD y repoblarla

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

Actualizamos el databaseseeder y aplicamos php artisan migrate:fresh --seed

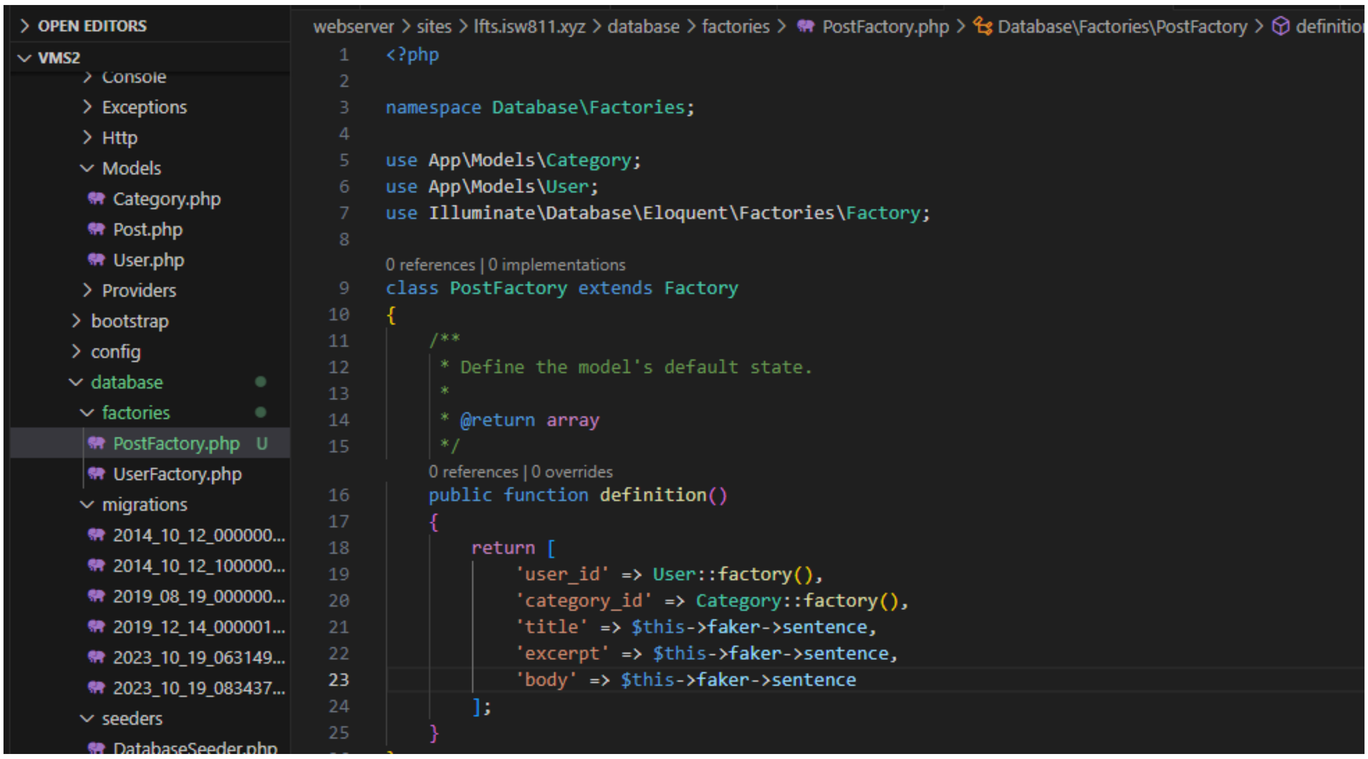
```
<?php

namespace Database\Seeders;

use App\Models\Category;
use App\Models\Post;
use App\Models\User;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        User::truncate();
        Category::truncate();
        Post::truncate();

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

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

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

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

        Post::create([
            'user_id'=> $user->id,
            'category_id' => $family->id,
            'title' => 'My Family Post',
            'slug' => 'my-family-post',
            'excerpt' => '<p>Lorem ipsum, dolor sit amet consectetur adipisicing
elit.</p>',
            'body' => '<p>Lorem ipsum, dolor sit amet consectetur adipisicing
elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit ipsam,
suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem
minima.odio officia</p>',
        ]);

        Post::create([
            'user_id'=> $user->id,
            'category_id' => $work->id,
            'title' => 'My Work Post',
            'slug' => 'my-work-post',
            'excerpt' => '<p>Lorem ipsum, dolor sit amet consectetur adipisicing
elit.</p>',
            'body' => '<p>Lorem ipsum, dolor sit amet consectetur adipisicing
elit. Expedita accusamus quaerat aliquid incidunt nihil iste sed velit ipsam,
suscipit molestias dicta fugiat odio officia omnis reprehenderit minus quidem
minima.odio officia</p>',
        ]);

    }
}
```

Turbo Boost With Factories

Creamos un post de factory

```
php artisan make:factory PostFactory
```

Lo modificamos



```php
<?php

namespace Database\Factories;

use App\Models\Category;
use App\Models\User;
use Illuminate\Database\Eloquent\Factories\Factory;

0 references | 0 implementations
class PostFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    0 references | 0 overrides
    public function definition()
    {
        return [
            'user_id' => User::factory(),
            'category_id' => Category::factory(),
            'title' => $this->faker->sentence,
            'excerpt' => $this->faker->sentence,
            'body' => $this->faker->sentence
        ];
    }
```

Creamos un factory CategoryFactory

```
php artisan make:factory CategoryFactory;
```

Y lo modificamos

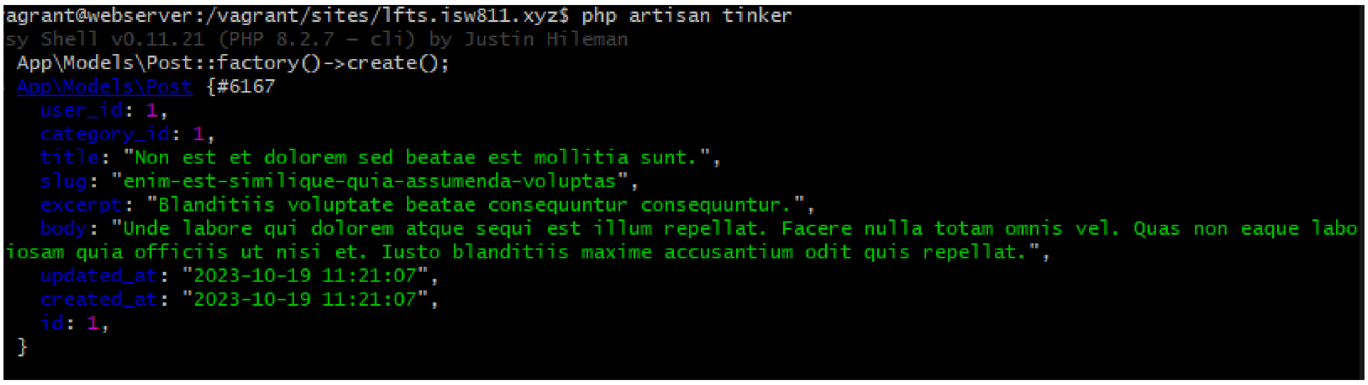
```php
public function definition()
    {
        return [
            'name' => $this->faker->word,
            'slug'=> $this->faker->slug,
        ];
    }
```

Tambien modificamos el postFactory

```php
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
        ];
    }
```

Ingresamos a tinker

```
php artisan tinker
App\Models\Post::factory()->create();
```



```
agrant@webserver:/vagrant/sites/lfts.isw811.xyz$ php artisan tinker
sy Shell v0.11.21 (PHP 8.2.7 — cli) by Justin Hileman
 App\Models\Post::factory()->create();
 App\Models\Post {#6167
   user_id: 1,
   category_id: 1,
   title: "Non est et dolorem sed beatae est mollitia sunt.",
   slug: "enim-est-similique-quia-assumenda-voluptas",
   excerpt: "Blanditiis voluptate beatae consequuntur consequuntur.",
   body: "Unde labore qui dolorem atque sequi est illum repellat. Facere nulla totam omnis vel. Quas non eaque labo
iosam quia officiis ut nisi et. Iusto blanditiis maxime accusantium odit quis repellat.",
   updated_at: "2023-10-19 11:21:07",
   created_at: "2023-10-19 11:21:07",
   id: 1,
 }
```

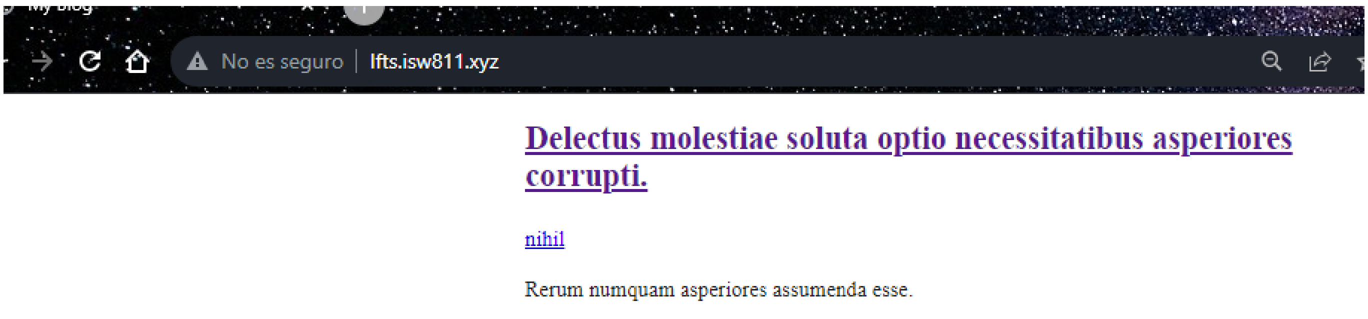
Modificamos nuevamente el databaseSeeder

```
<?php

namespace Database\Seeders;

use App\Models\Category;
use App\Models\Post;
use App\Models\User;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        Post::factory()->create();
    }
}
```

Ejecutamos el comando

```
php artisan db:seed
```

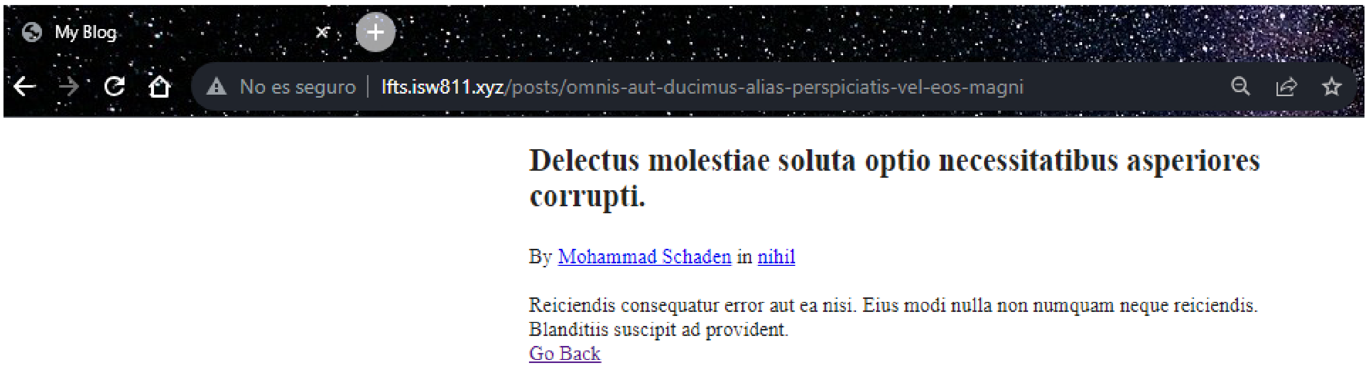
Y ya podemos ver un post creado automaticamente en todos sus campos



Delectus molestiae soluta optio necessitatibus asperiores corrupti.

nihil

Rerum numquam asperiores assumenda esse.


My Blog
No es seguro
lfts.isw811.xyz/posts/omnis-aut-ducimus-alias-perspiciatis-vel-eos-magni

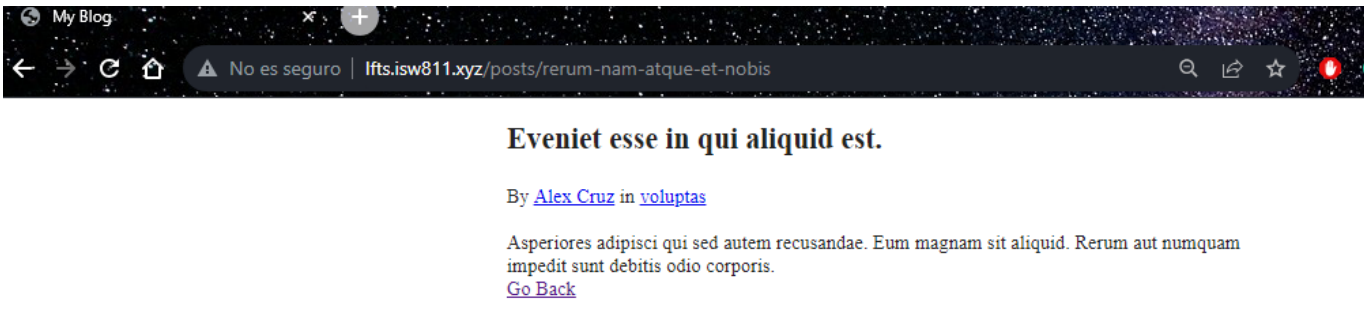
Delectus molestiae soluta optio necessitatibus asperiores corrupti.

By Mohammad Schaden in nihil

Reiciendis consequatur error aut ea nisi. Eius modi nulla non numquam neque reiciendis. Blanditiis suscipit ad provident.
Go Back

Modificamos nuevamente el databaseSeeder para que el usuario siempre sea Alex Cruz

```
<?php

namespace Database\Seeders;

use App\Models\Category;
use App\Models\Post;
use App\Models\User;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $user = User::factory()->create([
            'name' => 'Alex Cruz',
        ]);


        Post::factory(5)->create([
            'user_id'=> $user->id,
        ]);
    }
}
```



Eveniet esse in qui aliquid est.

By Alex Cruz in voluptas

Asperiores adipisci qui sed autem recusandae. Eum magnam sit aliquid. Rerum aut numquam impedit sunt debitis odio corporis.
Go Back

En cada cambio se recomienda correr

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

View All Posts By An Author

En web.phph modificamos el get home posts para que los acomode de manera desc

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

En web.php creamos una ruta para poder traer todos los post de un autor

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

Para evitar utilizar el id vamos a crear lo siguiente. Primero agregamos un username a la migracion de user

```
 $table->string('username')->unique();
```

En userfactory tambien agregamos el campo

```
'username' => $this->faker->unique()->username,
```

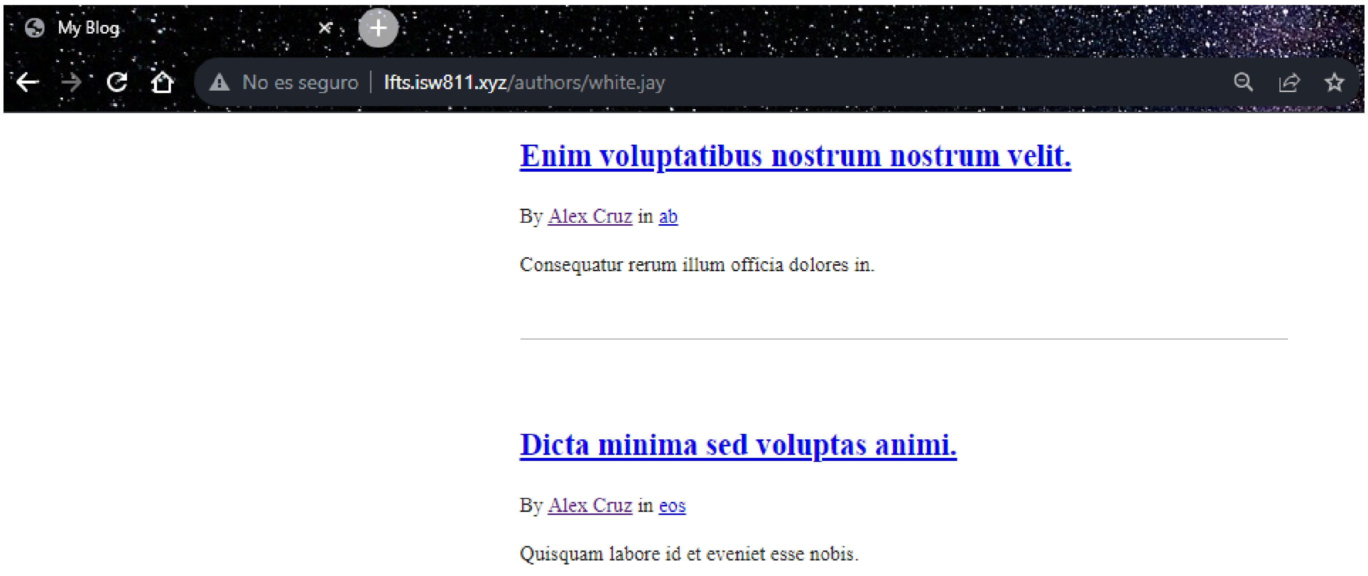
El metodo en web.php quedaria de la siguiente manera

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

Corremos el migrate

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

Y asi se ve la pagina



Enim voluptatibus nostrum nostrum velit.

By Alex Cruz in ab

Consequatur rerum illum officia dolores in.

Dicta minima sed voluptas animi.

By Alex Cruz in eos

Quisquam labore id et eveniet esse nobis.

Eager Load Relationships on an Existing Model

Agregamos a post.php el siguiente codigo que sirve por defecto para cada post query que se realice

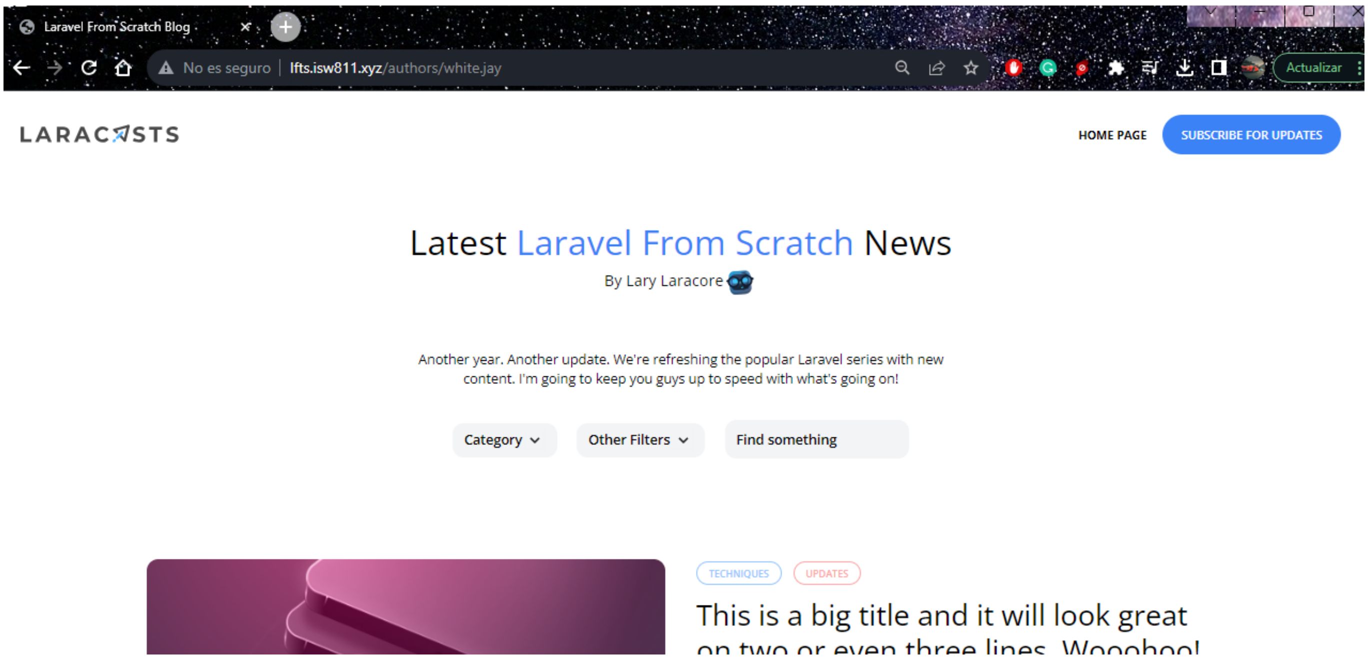
```
    protected $with = ['category', 'author'];
```

A raiz de esa linea entonces nuestro archivo de rutas web.php puede verse asi

```
<?php
use App\Models\Category;
use App\Models\User;
use Illuminate\Support\Facades\Route;
use App\Models\Post;

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

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

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

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

Convert the HTML and CSS to Blade

Descargamos el repositorio de git y pegamos el contenido del index en nuestro layout.blade.php de la carpeta components Pasamos la carpeta de imagenes a nuestro public y nada mas cambiamos las rutas del index y vemos la pagina asi



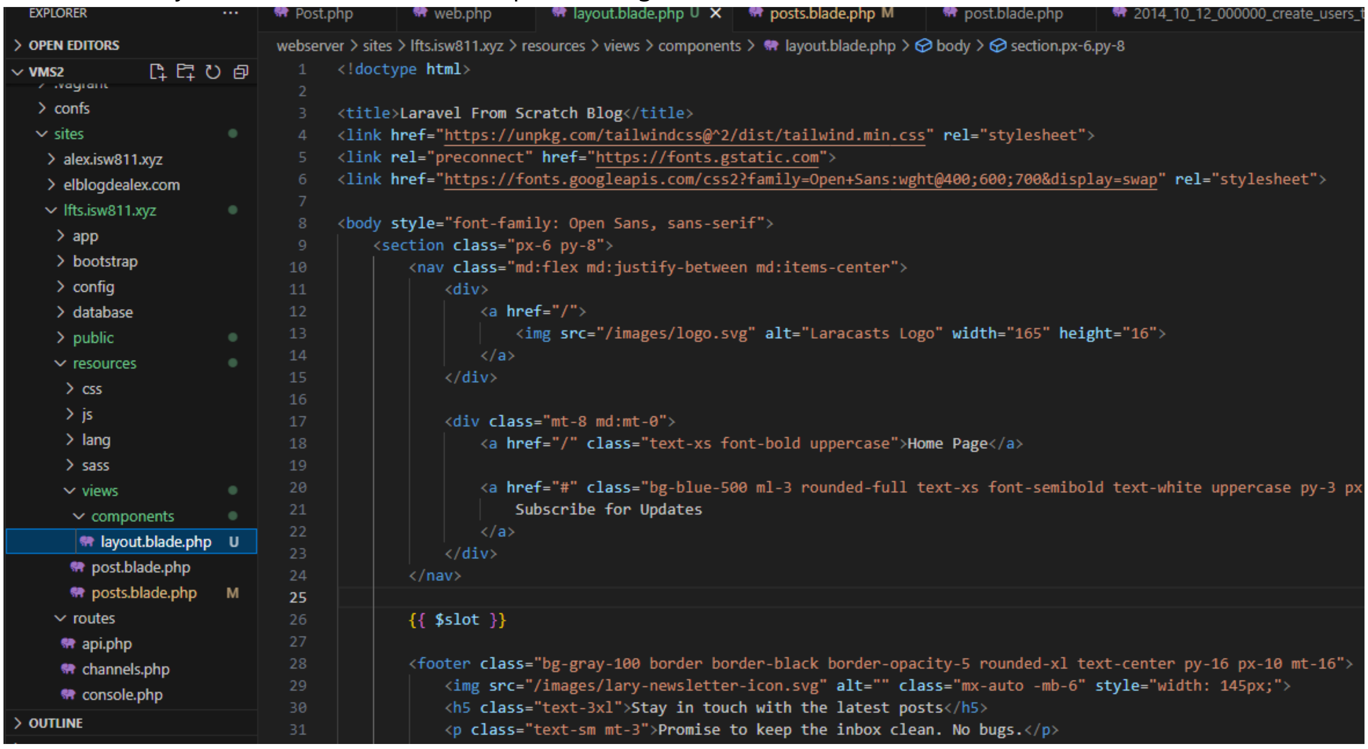
Latest Laravel From Scratch News

By Lary Laracore

Another year. Another update. We're refreshing the popular Laravel series with new content. I'm going to keep you guys up to speed with what's going on!

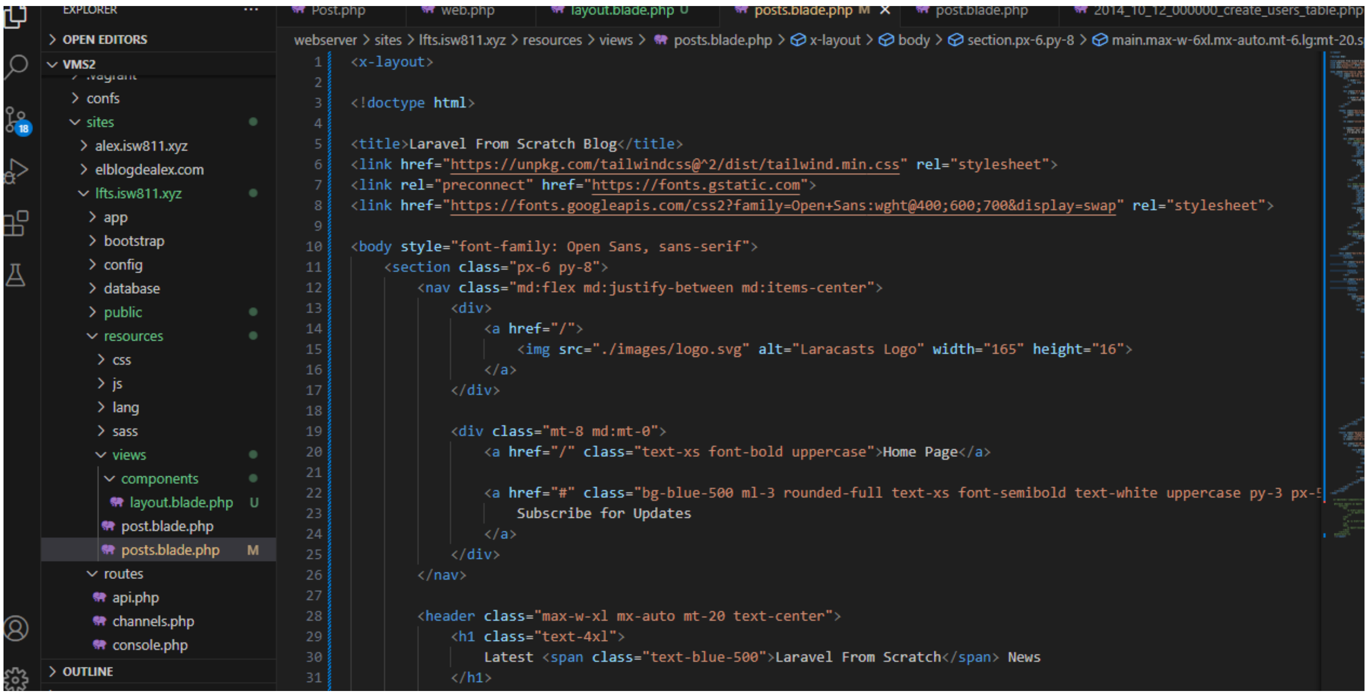
This is a big title and it will look great

Editamos el layout con el html del index que descargamos



```html
<!doctype html>

<title>Laravel From Scratch Blog</title>
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600;700&display=swap" rel="stylesheet">

<body style="font-family: Open Sans, sans-serif">
    <section class="px-6 py-8">
        <nav class="md:flex md:justify-between md:items-center">
            <div>
                <a href="/">
                    <img src="/images/logo.svg" alt="Laracasts Logo" width="165" height="16">
                </a>
            </div>

            <div class="mt-8 md:mt-0">
                <a href="/" class="text-xs font-bold uppercase">Home Page</a>

                <a href="#" class="bg-blue-500 ml-3 rounded-full text-xs font-semibold text-white uppercase py-3 px
                    Subscribe for Updates
                </a>
            </div>
        </nav>

        {{ $slot }}

        <footer class="bg-gray-100 border border-black border-opacity-5 rounded-xl text-center py-16 px-10 mt-16">
            <img src="/images/lary-newsletter-icon.svg" alt="" class="mx-auto -mb-6" style="width: 145px;">
            <h5 class="text-3xl">Stay in touch with the latest posts</h5>
            <p class="text-sm mt-3">Promise to keep the inbox clean. No bugs.</p>
```

Igualmente el post



```html
<x-layout>

<!doctype html>

<title>Laravel From Scratch Blog</title>
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600;700&display=swap" rel="stylesheet">

<body style="font-family: Open Sans, sans-serif">
    <section class="px-6 py-8">
        <nav class="md:flex md:justify-between md:items-center">
            <div>
                <a href="/">
                    <img src="./images/logo.svg" alt="Laracasts Logo" width="165" height="16">
                </a>
            </div>

            <div class="mt-8 md:mt-0">
                <a href="/" class="text-xs font-bold uppercase">Home Page</a>

                <a href="#" class="bg-blue-500 ml-3 rounded-full text-xs font-semibold text-white uppercase py-3 px-5
                    Subscribe for Updates
                </a>
            </div>
        </nav>

        <header class="max-w-xl mx-auto mt-20 text-center">
            <h1 class="text-4xl">
                Latest <span class="text-blue-500">Laravel From Scratch</span> News
            </h1>
```