

## UF 2218 práctica

Para la realización de esta práctica seguimos el tutorial de laravel para la creación de un CMS: <https://www.youtube.com/watch?v=nFKwq2PHJ64>

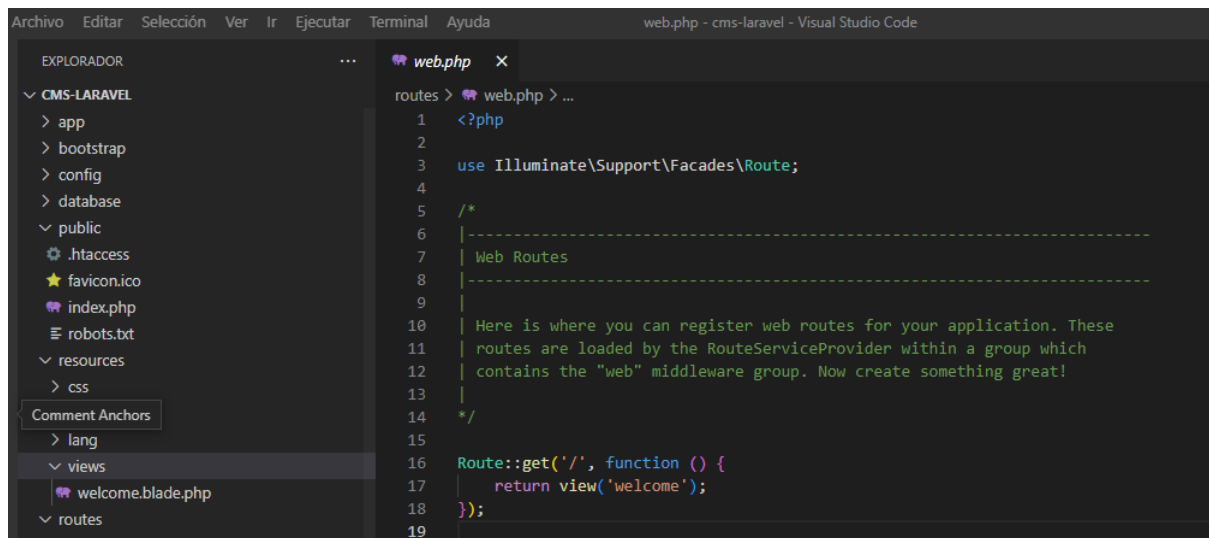
Creamos un sitio nuevo de laravel a través de laragon

C:\WINDOWS\SYSTEM32\cmd.exe - C:\laragon\laragon reload

```
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
7 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.
- @php artisan key:generate --ansi
Application key set successfully.

**** NOTE: Now, you can use pretty url for your awesome project :) ****

Laragon) Project path: C:/laragon/www/cms-laravel
Laragon) Pretty url: http://cms-laravel.test
```

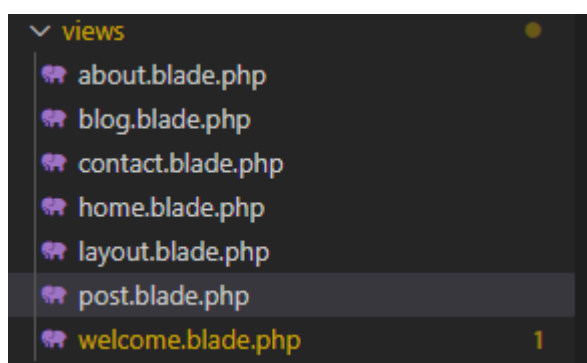


En este caso el archivo `web.php` nos devuelve la vista de la ruta `"welcome.blade.php"`

```
web.php
routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6  |-----
7  | Web Routes
8  |-----
9  |
10 | Here is where you can register web routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | contains the "web" middleware group. Now create something great!
13 |
14 */
15
16 // Route::get('/', function () {
17 //     return view('welcome');
18 // });
19
20 Route::view('/', 'home')->name('home');
21 Route::view('acerca-de', 'about')->name('about');
22
23 Route::get('blog', 'BlogController@index')->name('blog.index');
24 Route::get('blog/{post:slug}', 'BlogController@show')->name('blog.show');
25
26 Route::view('contactos', 'contact')->name('contact');
27
```

De esta forma como se ve en la imagen creamos nuevas rutas. En el primer *route get blog* llamamos a la bbdd

A continuación creamos las nuevas vistas añadiendo los archivos que se ven en la siguiente imagen.



De entre ellos destacamos el archivo *layout.blade.php* que sería como nuestra plantilla.

```
resources > views > layout.blade.php > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>cms laravel</title>
8  </head>
9  <body>
10     Menu
11
12     <div>
13         @yield('content')
14     </div>
15 </body>
16 </html>
17
```

Damos forma a las demás vistas declarando con el siguiente código lo que se vería en los archivos restantes de la carpeta views como se ve en las siguientes imágenes.

```
resources > views > about.blade.php > ...
1  @extends('layout')
2
3  @section('content')
4
5      <h1>Contenido de acerca de...</h1>
6
7  @endsection
8
```

```
home.blade.php layout.blade.php about.blade.php blog.blade.php X
resources > views > blog.blade.php > h1
1 @extends('layout')
2
3 @section('content')
4
5 <h1>Contenido de blog</h1>
6
7 @endsection
8

terminal Ayuda contact.blade.php - cms-laravel - Visual Studio Code
home.blade.php layout.blade.php about.blade.php blog.blade.php contact.blade.php X
resources > views > contact.blade.php > ...
1 @extends('layout')
2
3 @section('content')
4
5 <h1>Contenido de contactos</h1>
6
7 @endsection
8
```

A continuación en el archivo *layout* añadimos el siguiente código, de esta forma generamos el menú.

```
web.php home.blade.php layout.blade.php X about.blade.php blog.blade.php
sources > views > layout.blade.php > html > body > ul
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>cms laravel</title>
8 </head>
9 <body>
10   <ul>
11     <li><a href="{{ route('home') }}">Home</a></li>
12     <li><a href="{{ route('about') }}">Acerca de</a></li>
13     <li><a href="{{ route('blog.index') }}">Blog</a></li>
14     <li><a href="{{ route('contact') }}">Contactos</a></li>
15   </ul>
16
17   <div>
18     @yield('content')
19   </div>
20 </body>
21 </html>
22
```

Una vez hecho esto ejecutamos el comando *npm install* y *npm run dev*. De esta forma se crean nuevos archivos en la carpeta *public*, con especial relevancia el archivo *app.css* dentro de *css*



A continuación si visualizamos nuestro proyecto en el navegador este es el resultado:



Después en el archivo *.env* configuramos el nombre de la base de datos y en el archivo *layout.blade.php* enlazamos la hoja de estilos *app.css*

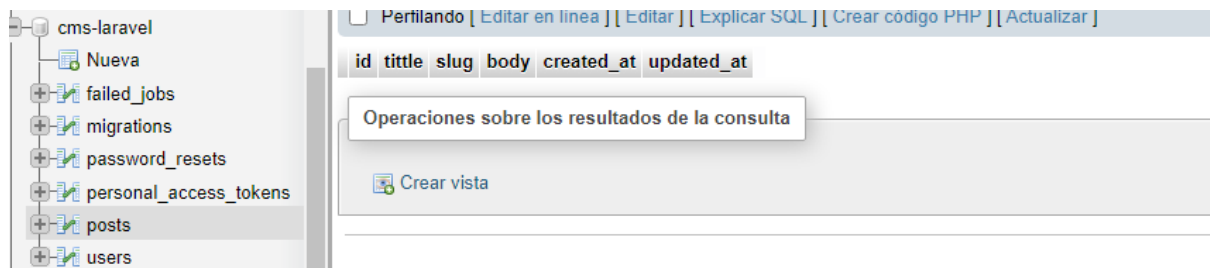
Una vez hecho esto ejecutamos el comando *php artisan make:model Post-msfc*

```
C:\laragon\www\cms-laravel
λ php artisan make:model Post -msfc
Model created successfully.
Factory created successfully.
Created Migration: 2022_05_23_110556_create_posts_table
```

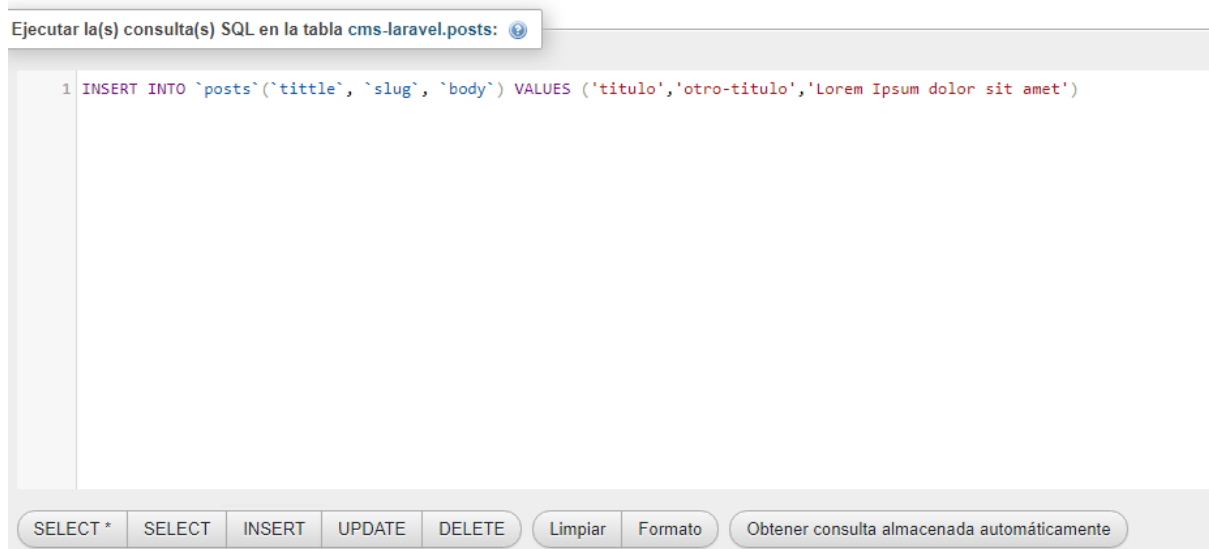
De esta forma en nuestra base de datos se han creado nuevas tablas, tal y como se recoge en el siguiente código.

```
database > migrations > 2022_05_23_110556_create_posts_table.php > CreatePostsTable > up > #Function#bf6b1d3c
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreatePostsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('posts', function (Blueprint $table) {
17             $table->id();
18
19             $table->string('title');
20             $table->string('slug');
21             $table->text('body');
22
23             $table->timestamps();
24         });
25     }
26 }
```

Podemos ver como se encuentra la tabla:



Podemos crear nuevos post a través de *phpmyAdmin*



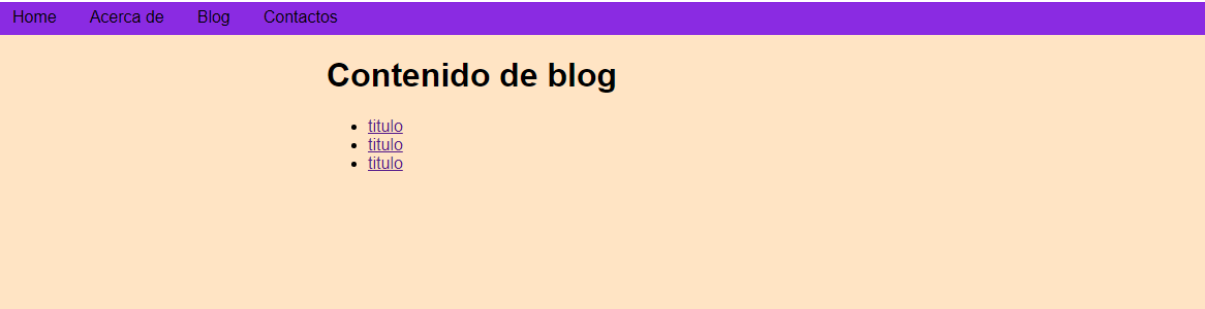
Cambiamos el archivo *PostController.php* y lo llamamos *BlogController.php*

Ejecutamos *php artisan cache:clear*

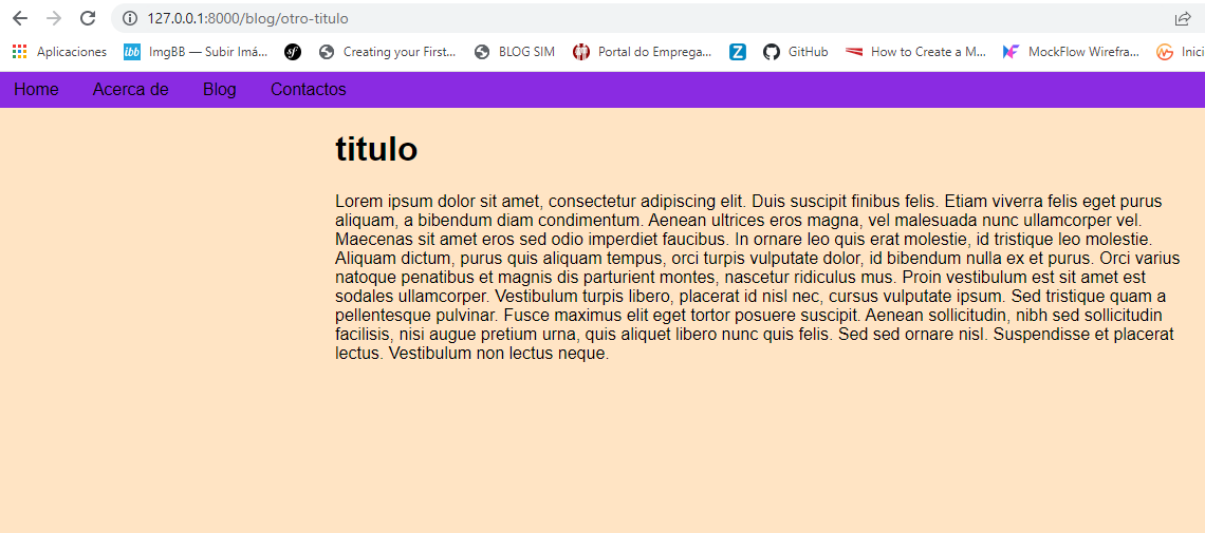
A continuación cambiamos el código en el archivo *PostFactory.php*

```
1 <?php
2
3 namespace Database\Factories;
4
5 use App\Models\Post;
6 use Illuminate\Database\Eloquent\Factories\Factory;
7
8 class PostFactory extends Factory
9 {
10     /**
11      * The name of the factory's corresponding model.
12      *
13      * @var string
14      */
15     protected $model = Post::class;
16
17     /**
18      * Define the model's default state.
19      *
20      * @return array
21      */
22     public function definition()
23     {
24         return [
25             'user_id' => 1,
26             'title' => $this->faker->sentence,
27             'body' => $this->faker->text(3000),
28         ];
29     }
30 }
```

Añadimos estilos en ambos *app.css*



Finalmente, pinchando en cada post podemos ver su contenido:



Para terminar instalamos un framework de estilo en este caso tailwind. El resultado final de la práctica se visualizará de la siguiente forma:







### Fresas

*La dulce fruta roja*

**Recetas** →

☐ Lista de la compra



### Arándanos

*Un pequeño toque dulce*

**Recetas** →

☐ Lista de la compra



### Limones

*Dale un toque ácido a tu día*

**Recetas** →

☐ Lista de la compra

## Formulario de contactos

Nombre:

Apellido:

Receta deseada:

Tiempo de realización:

Regista tu email:

Enviar