



Análisis y desarrollo de software.

ID: 2644590



www.sena.edu.co

@SENAComunica

Instructor

Diego Fernando Calderón Silva
Correo: dfcalderon@sena.edu.co

Enlaces para crear Posts



Vamos a comenzar por crear los Posts, por lo tanto, pondremos un botón en el dashboard. Para esto nos dirigimos al app.Blade.php e iremos al bloque de @auth ya que el usuario debe estar autenticado para poder crear un post.

```
@auth()  
  <nav class="flex gap-2 items-center">  
    <a class="flex items-center gap-2 bg-white border p-2 text-gray-600 rounded text-sm uppercase font-bold cursor-pointer" href="">  
      Crear  
    </a>
```

Y tendremos un botón como este:

CREAR HOLA DIEGOCALDERON

Enlaces para crear Posts



Junto a ese botón quiero poner un icono, por lo haremos una búsqueda en la web:

<https://heroicons.com>

Aquí encontraremos bastantes iconos de los mismos creadores de TailwindCSS los cuales son compatibles con react y vueJS y se pueden instalar vía npm o dando click sobre Copy svg.

A screenshot of a search results page from heroicons.com. A search bar at the top contains the text "photo". Below it, a card for the "photo" icon is shown. The card has a title "Outline 24x24, 1.5px stroke", a description "For primary navigation and marketing sections, with an outlined appearance.", and two buttons: "Copy SVG" with a copy icon and "Copy JSX" with a copy icon. To the right of this card are two smaller cards: one labeled "photo" with a camera icon and another labeled "camera" with a camera icon.

Enlaces para crear Posts



Una vez copiado, pegamos en el código del proyecto antes del texto del botón.

```
@auth()
<nav class="flex gap-2 items-center">
  <a class="flex items-center gap-2 bg-white border p-2 text-gray-600 rounded text-sm uppercase font-bold cursor-pointer" href="">
    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="w-6 h-6">
      <path stroke-linecap="round" stroke-linejoin="round" d="M6.827 6.175A2.31 2.31 0 015.186 7.23c-.38.054-.757.112-1.134.175C2.999 7.58 2.25 8.507 2.25 9.574V18a2 .25 2.25 0 002.25 2.25h15A2.25 2.25 0 0021.75 18V9.574c0-1.067-.75-1.994-1.802-2.169a47.865 47.865 0 00-1.134-.175 2.31 2.31 0 01-1.64-1.055l-.822-1.316a2.192 2.192 0 00-1.736-1.039 48.774 48.774 0 00-5.232 0 2.192 2.192 0 00-1.736 1.039l-.821 1.316z" />
      <path stroke-linecap="round" stroke-linejoin="round" d="M16.5 12.75a4.5 4.5 0 11-9 0 4.5 4.5 0 019 0zM18.75 10.5h.008v.008h-.008V10.5z" />
    </svg>
    Crear
  </a>
```

Enlaces para crear Posts



Por ahora el enlace no hace nada por que no tiene href, por lo tanto, lo redireccionaremos hacia un controlador teniendo en cuenta que el Verb es GET, url es /photos, Action es index y RouteName es photos.index

# Actions Handled By Resource Controller			
Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

Enlaces para crear Posts



Pero para crear un nuevo post el url debe ser /photos/create.

Actions Handled By Resource Controller

Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

Enlaces para crear Posts



Por esta razón iremos a web.php a crear la nueva ruta que nos lleva al controlador PostController.

```
Route::get('uri: '/posts/create', [PostController::class, 'create'])->name('name: post.create');
```

Ahora en el controlador crearemos el método 'create' así:

```
public function create(){
    dd(...vars: 'Creando posts');
}
```

Y en el app.Blade.php podremos dar href al botón de crear así:

```
<a class="flex items-center gap-2 bg-white border p-2 text-gray-600 rounded text-sm uppercase font-bold cursor-pointer" href="{{ route('post.create') }}>
```

Recargando y dando click sobre el botón crear verificamos la utilidad del endpoint.

Template de creación de publicaciones



Continuando con el proyecto, creamos un formulario para agregar las publicaciones. Para esto crearemos un directorio llamado `posts` y dentro una vista `create.blade.php`. Lo bueno de esto es que todo tiene el nombre de create, el controlador, el método, la vista, por lo tanto, será mas fácil identificar cada paso que queramos dar. Ahora, cambiaremos el dd por el retorno y nos dirigiremos hacia `posts.create`.

```
public function create(){
    return view( view: 'posts.create');
}
```

Seguidamente, iremos a heredar de `layouts.app` todo, y escribiremos en la sección de título, Crear una nueva publicación.

Recargamos la página, observamos y nos disponemos a crear el código que creara los posts.

Template de creación de publicaciones



Creare un par de div y copiare el formulario de register.Blade.php para que nuestro diseño se vea igual y se tiene el siguiente resultado:

```
app.blade.php x web.php x PostController.php x create.blade.php x register.blade.php x

1 @extends('layouts.app')
2
3 @section('titulo')
4     Crea una nueva publicacion
5 @endsection
6
7 @section('contenido')
8     <div class="md:flex md:items-center" >
9         <div class="md:w-6/12 px-10 ">
10            Imagen Aqui
11        </div>
12
13         <div class="md:w-6/12 px-10 bg-white p-6 rounded-lg shadow-xl">
14             <form action="{{route('register')}}" method="POST" novalidate>
15                 @csrf
16                 <div class="mb-5">
17                     <label for="name" class="mb-2 block uppercase text-gray-500 font-bold" >Nombre</label>
18                     <input type="text" id="name" name="name" placeholder="Nombre" class="border p-3 w-full rounded-lg @error('name') border-red-500 @enderror" value="{{ old('name') }}>
19                     @error('name')
20                         <p class="bg-red-500 text-white my-2 rounded-lg text-sm p-2 text-center">{{ $message }}</p>
21                     @enderror
22
23                 </div>
24             </form>
25         </div>
26     </div>
27 @endsection
```

Template de creación de publicaciones



Anteriormente, la imagen esta tal cual como se copió de register, ahora modificaremos el formulario para adecuarlo a nuestra necesidad.

Ahora crearemos un textarea debajo de este div teniendo en cuenta que acá no existe el value.

```
<div class="mb-5">
  <label for="titulo" class="mb-2 block uppercase text-gray-500 font-bold" >Titulo</label>
  <input type="text" id="titulo" name="titulo" placeholder="Titulo" class="border p-3 w-full rounded-lg @error('titulo') border-red-500 @enderror" value="{{ old('titulo') }}>
  @error('name')
    <p class="bg-red-500 text-white my-2 rounded-lg text-sm p-2 text-center">{{ $message }}</p>
  @enderror
</div>
```

```
<div class="mb-5">
  <label for="descripcion" class="mb-2 block uppercase text-gray-500 font-bold" >Descripcion de la publicacion</label>

  <textarea id="descripcion" name="descripcion" placeholder="Descripcion de la publicacion" class="border p-3 w-full rounded-lg @error('titulo') border-red-500 @enderror" > {{ old('titulo') }} </textarea>
  @error('name')
    <p class="bg-red-500 text-white my-2 rounded-lg text-sm p-2 text-center">{{ $message }}</p>
  @enderror
</div>

<input type="submit" value="Publicar" class="bg-sky-600 hover:bg-sky-700 transition-colors cursor-pointer uppercase font-bold w-full p-3 text-white rounded-lg">
```

Template de creación de publicaciones



Cierre sesión y compruebe el funcionamiento completo de su app.

Template de creación de publicaciones



The screenshot shows a Laravel application error page. The URL in the browser is `127.0.0.1:8000/login`. The error message is: `Missing required parameter for [Route: post.index] [URI: {user}] [Missing parameter: user].` The error is categorized under `Illuminate\Routing\Exceptions\UrlGenerationException`. The page includes navigation links for `STACK`, `CONTEXT`, `DEBUG`, and `FLARE`, along with sharing and documentation options. The stack trace on the right side of the page shows the following code from `app/Http/Controllers/LoginController.php:24`:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use function Laravel\Prompts\password;
7
8 class LoginController extends Controller
9 {
```

Solucionando inicio de sesión.



Recordando un poco, en LoginController, tenemos que, si el usuario se autentica, lo enviamos a post.index pero si vemos web.php nos encontramos que index requiere el username y laravel no sabe en realidad cual es, por eso debemos pasarle esa información.

En

LoginController enviamos:

A screenshot of a code editor showing a portion of a Laravel application. The tabs at the top include 'app.blade.php', 'web.php', 'PostController.php', 'create.blade.php', and 'LoginController.php'. The 'LoginController.php' tab is active, highlighted with a yellow bar. The code itself is a PHP file with syntax highlighting. It defines a class 'LoginController' extending 'Controller'. It contains two methods: 'index()' which returns a view for login, and 'store(Request \$request)' which validates the request for email and password, attempts to authenticate the user, and if successful, redirects to 'post.index' route with the authenticated user's username.

Solucionando inicio de sesión.



Antes de avanzar, vamos a app.Blade, en el span de hola donde imprimimos el nombre de usuario y vamos a crear la funcionalidad que lo lleve a su propio muro.

```
    Crear
  </a>
  <a class="font-bold uppercase text-gray-600" href="{{ route('post.index', auth()>user()>username) }}>
    Hola <span class="font-bold" > {{ auth()>user()>username }} </span>
  </a>

  <form action="{{ route('logout') }}" method="post">
    @csrf
    <button type="submit" class="font-bold uppercase text-gray-600" >
      Cerrar sesión
    </button>
  </form>
```

Solucionando inicio de sesión.



Antes de avanzar, vamos a app.Blade, en el span de hola donde imprimimos el nombre de usuario y vamos a crear la funcionalidad que lo lleve a su propio muro.

Verifiquemos que todo esté funcionando correctamente y verifiquemos que pasa si no ponemos auth()->user()->username

```
    Crear
  </a>
<a class="font-bold uppercase text-gray-600" href="{{ route('post.index', auth()>user()>username) }}>
  Hola <span class="font-bold" > {{ auth()>user()>username }} </span>
</a>

<form action="{{ route('logout') }}" method="post">
  @csrf
  <button type="submit" class="font-bold uppercase text-gray-600" >
    Cerrar sesión
  </button>
</form>
```

Instalemos Dropzone



¿Qué es dropzone?

Es una biblioteca o una característica de interfaz de usuario que permite al cliente arrastrar y soltar archivos desde su computadora a una zona específica de una página web.

La documentación la encontraremos en:

<https://www.dropzone.dev>

The screenshot shows the official website for Dropzone.js. The header features the 'Dropzone' logo and social media links. The main title 'File uploads made easy' is prominently displayed in large white text against a red-to-yellow gradient background. Below the title, a paragraph explains what Dropzone.js is: 'Dropzone.js is one of the most popular drag and drop JavaScript libraries. It is free, fully open source, and makes it easy for you to handle dropped files on your website.' A note below states, 'It's meant to look good by default, and is highly customizable.' At the bottom of the main section are 'Documentation >' and 'Download >'. To the right, a white callout box contains the heading 'Try it out!', a placeholder 'Drag and drop files here', and a small note: 'This is just a demo Dropzone. Dropped files are not actually uploaded.'



[Source code on GitHub](#)

You can get all the source code on GitHub, as well as installation instructions. If you encounter an issue with this library, this is the place to create an issue.

[GitHub >](#)



[Documentation](#)

All the documentation about Dropzone, and the multiple ways to configure and customise it, can be found on GitBook.

[Docs >](#)



[Questions and Support](#)

If you need help, there are GitHub Discussions and Stackoverflow. Use the tag dropzonejs and there'll be plenty of people helping you out.

[Stack Overflow >](#)

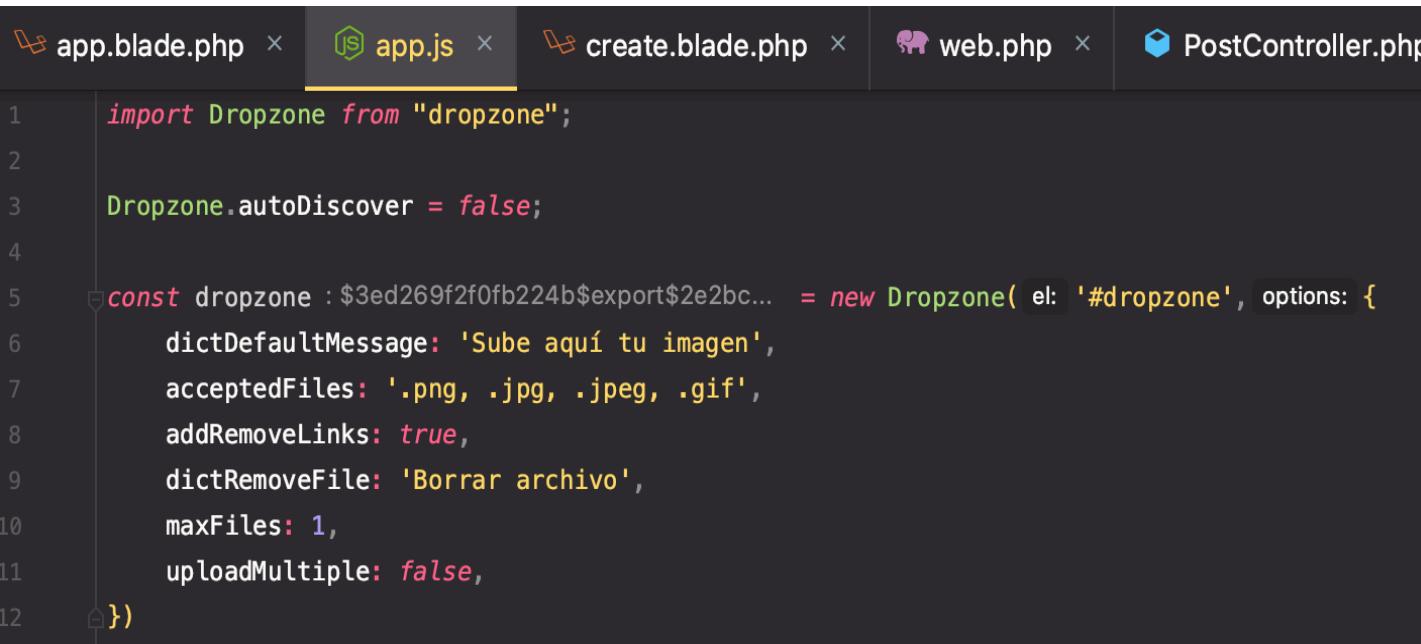
Instalemos Dropzone



Para instalar vamos a la documentación y en el apartado de instalación nos dice que con npm podemos instalar:

```
npm yarn  
$ npm install --save dropzone
```

Bajando un poco mas vemos que viene la opción de css **Stand-alone file**. y copiaremos el primer link que encontramos al inicio para copiarlo. Primero eliminaremos /resources/js/Bootstrap.js y en /resources/js/app.js, le daremos forma al dropzone seteando los valores que queremos por defecto. Recuerda agregar en app.Blade.php **@vite('resources/js/app.js')**



The screenshot shows a code editor with several tabs at the top: app.blade.php, app.js (which is currently selected), create.blade.php, web.php, and PostController.php. The app.js tab contains the following code:

```
import Dropzone from "dropzone";  
  
Dropzone.autoDiscover = false;  
  
const dropzone : $3ed269f2f0fb224b$export$2e2bc... = new Dropzone( el: '#dropzone', options: {  
    dictDefaultMessage: 'Sube aquí tu imagen',  
    acceptedFiles: '.png, .jpg, .jpeg, .gif',  
    addRemoveLinks: true,  
    dictRemoveFile: 'Borrar archivo',  
    maxFiles: 1,  
    uploadMultiple: false,  
});
```

Instalemos Dropzone



Ahora para visualizar el dropzone, debemos dirigirnos a `create.blade.php` y agregar:

```
@section('contenido')
    <div class="md:flex md:items-center" >
        <div class="md:w-6/12 px-10 ">
            <form action="" id="dropzone" class="dropzone border-dashed border-2 w-full h-96 rounded flex-col justify-center items-center">
                </form>
            </div>
```

Si recargamos la página no observaremos los valores por defecto y esto se debe a que no hay una ruta en `action`, la pondremos con fines demostrativos (`/img`) y tendremos:

The screenshot shows a user interface for creating a new post on a platform called Devstagram. At the top right, there are buttons for 'CREAR' (Create) and 'CERRAR SESIÓN' (Close Session). The main area has a title 'Crea una nueva publicación'. On the left, there is a dashed box labeled 'Sube aquí tu imagen' (Upload your image here). On the right, there are input fields for 'TITULO' (Title) containing 'Título' and 'DESCRIPCION DE LA PUBLICACION' (Post Description). A large blue button at the bottom right is labeled 'PUBLICAR' (Publish).



G R A C I A S

Línea de atención al ciudadano: 01 8000 910270

Línea de atención al empresario: 01 8000 910682



www.sena.edu.co