



Análisis y desarrollo de software.

ID: 2556456



www.sena.edu.co

@SENAComunica

Instructor

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

Autenticación



Laravel en su infinita sabiduría nos provee de métodos o funciones para generar autenticación exitosa y segura atravez de una serie de helpers y esto lo evidenciaremos en PostController, donde se reemplazará el dd con desde muro por:

```
class PostController extends Controller
{
    public function index(){
        dd(auth()->user());
    }
}
```

Actualice su vista de muro.

```
null // app/Http/Controllers/PostController.php:10
```

Y esto se debe a que aun no sabemos si el usuario esta autenticado.

Autenticación

Vamos a dirigirnos al controlador RegisterController y veremos que, así como existen métodos para redireccionar, también existen para autenticar; en este caso lo haremos mediante attempt lo cual intentara autenticar de forma exitosa así:

```
//Autenticar

auth()->attempt([
    'email' => $request->email,
    'password' => $request->password

]);
//Redireccionar
return redirect()->route( route: 'post.index');
```

En este caso intentaremos autenticar mediante email y password teniendo en cuenta que attempt retorna un valor booleano.

Autenticación



Intente crear un usuario nuevo con las siguientes condiciones:

1. Que exista su email en la bd
2. Que no exista ningún dato en la bd

Autenticación

Dado que existe el registro en la bd deberíamos tener como respuesta que ya existe, pero de lo contrario nuestra pantalla debería cargar la siguiente información:

```
App\Models\User {#686 ▼ // app/Http/Controllers/PostController.php:10
  #connection: "mysql"
  #table: "users"
  #primaryKey: "id"
  #keyType: "int"
  +incrementing: true
  #with: []
  #withCount: []
  +preventsLazyLoading: false
  #perPage: 15
  +exists: true
  +wasRecentlyCreated: false
  #escapeWhenCastingToString: false
  #attributes: array:9 [▶]
  #original: array:9 [▶]
  #changes: []
  #casts: array:2 [▶]
  #classCastCache: []
  #attributeCastCache: []
  #dateFormat: null
  #appends: []
  #dispatchesEvents: []
  #observables: []
  #relations: []
  #touches: []
  +timestamps: true
  +usesUniqueIds: false
  #hidden: array:2 [▶]
  #visible: []
  #fillable: array:4 [▶]
  #guarded: array:1 [▶]
  #rememberTokenName: "remember_token"
  #accessToken: null
}
```

Autenticación

Y nos centraremos en attributes:

```
App\Models\User {#686 ▼ // app/Http/Controllers/PostController.php:10
  #connection: "mysql"
  #table: "users"
  #primaryKey: "id"
  #keyType: "int"
  +incrementing: true
  #with: []
  #withCount: []
  +preventsLazyLoading: false
  #perPage: 15
  +exists: true
  +wasRecentlyCreated: false
  #escapeWhenCastingToString: false
  #attributes: array:9 [▼
    "id" => 10
    "name" => "cami"
    "email" => "cami@informacion.com"
    "email_verified_at" => null
    "password" => "$2y$10$xIRcKurxhftcQtZBwTPUmezFi7FH/oFXWUF2pmnsz1rzdyw6Lxc7S"
    "remember_token" => null
    "created_at" => "2023-10-13 00:37:15"
    "updated_at" => "2023-10-13 00:37:15"
    "username" => "cccmi basto riascos"
  ]
  #original: array:9 [▶]
  #changes: []
  #casts: array:2 [▶]
  #classCastCache: []
  #attributeCastCache: []
  #dateFormat: null
  #appends: []
  #dispatchesEvents: []
  #observables: []
  #relations: []
  #touches: []
  +timestamps: true
  +usesUniqueIds: false
  #hidden: array:2 [▶]
  #visible: []
  #fillable: array:4 [▶]
  #guarded: array:1 [▶]
  #rememberTokenName: "remember_token"
  #accessToken: null
}
```

Autenticación



Otra forma de autenticar es haciendo uso del mismo método, pero escrito de la siguiente forma:

```
//Auutenticar

/*auth()->attempt([
    'email' => $request->email,
    'password' => $request->password
]);*/

//otra forma de autenticar
auth()->attempt($request->only( keys: 'email', 'password'));
```

Con esto le indicamos que de \$request solamente tome email y password.

Autenticación



Ahora lo que debemos hacer es crear un muro y proteger los datos del usuario.

Escucho propuestas.

Autenticación



Debemos entonces, crear una vista para el muro llamada dashboard.Blade.php

Una vez creada se incorporará app.Blade.php y se recargará la pagina

A screenshot of a web application interface. At the top left is the logo 'Devstagram'. At the top right are two buttons: 'LOGIN' and 'CREAR CUENTA'. Below the header is a light gray content area containing the text 'Hola desde el dashboard.blade'. At the bottom of the content area, there is a footer bar with the text 'DEVSTAGRAM - TODOS LOS DERECHOS RESERVADOS 2023-10-13 01:53:16'.

Autenticación



Ahora dentro de contenido crearemos todo lo necesario para nuestra vista de muro

Devstagram

[LOGIN](#) [CREAR CUENTA](#)

Hola desde el dashboard

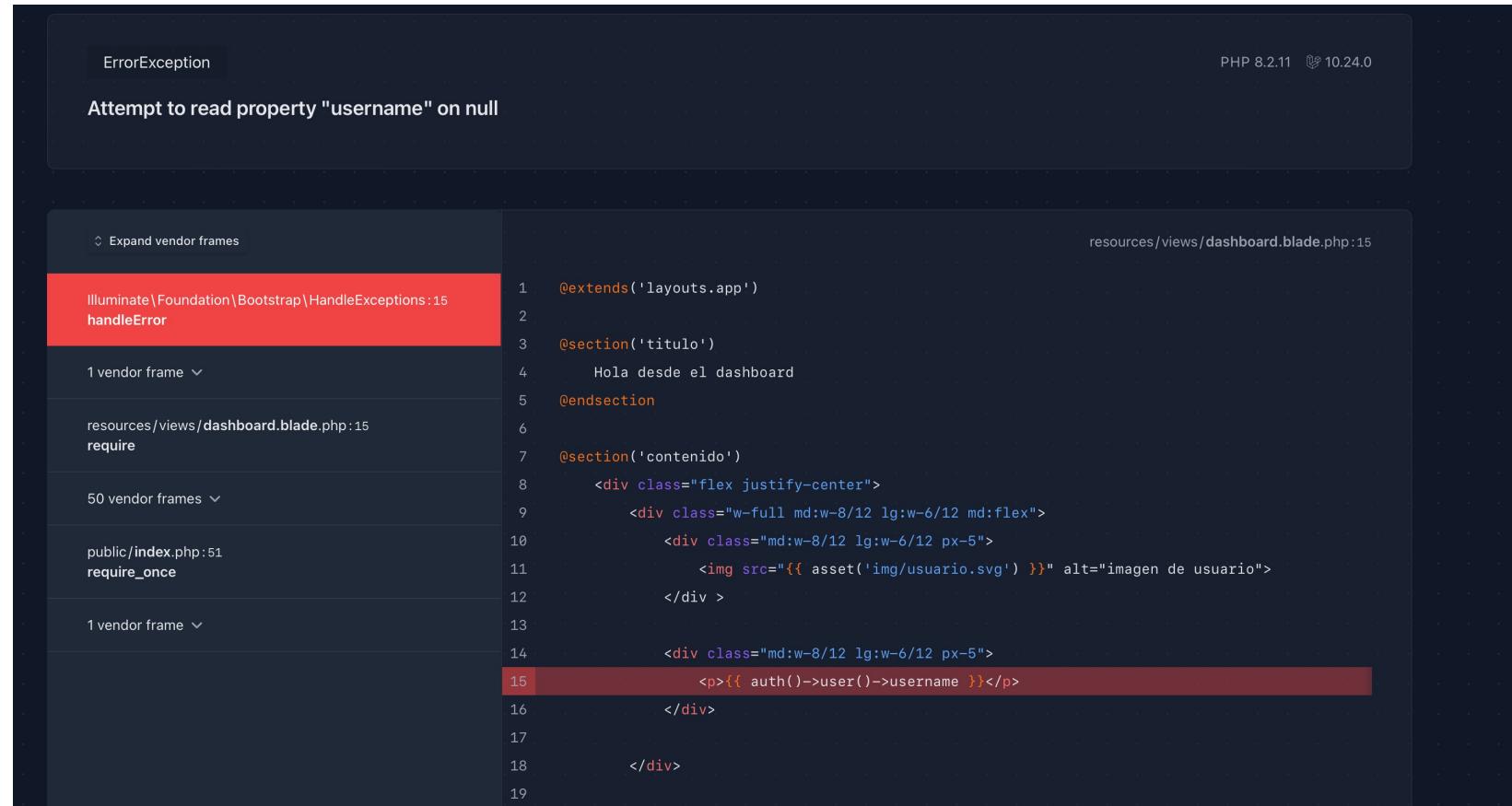
diegokld



DEVSTAGRAM - TODOS LOS DERECHOS RESERVADOS 2023-10-13 02:38:37

Autenticación

Miremos que pasa si intentamos acceder a la ruta de nuestro muro directamente desde nuestra url en una pestaña nueva o incognito donde no se haya iniciado sesión:



The screenshot shows a Laravel error page with the following details:

- Error Type:** `ErrorException`
- Message:** `Attempt to read property "username" on null`
- Environment:** PHP 8.2.11 | MySQL 10.24.0
- File:** `resources/views/dashboard.blade.php:15`
- Stack Trace:**

```
Illuminate\Foundation\Bootstrap\HandleExceptions::15
handleError
```

1 vendor frame ▾

```
resources/views/dashboard.blade.php:15
require
```

50 vendor frames ▾

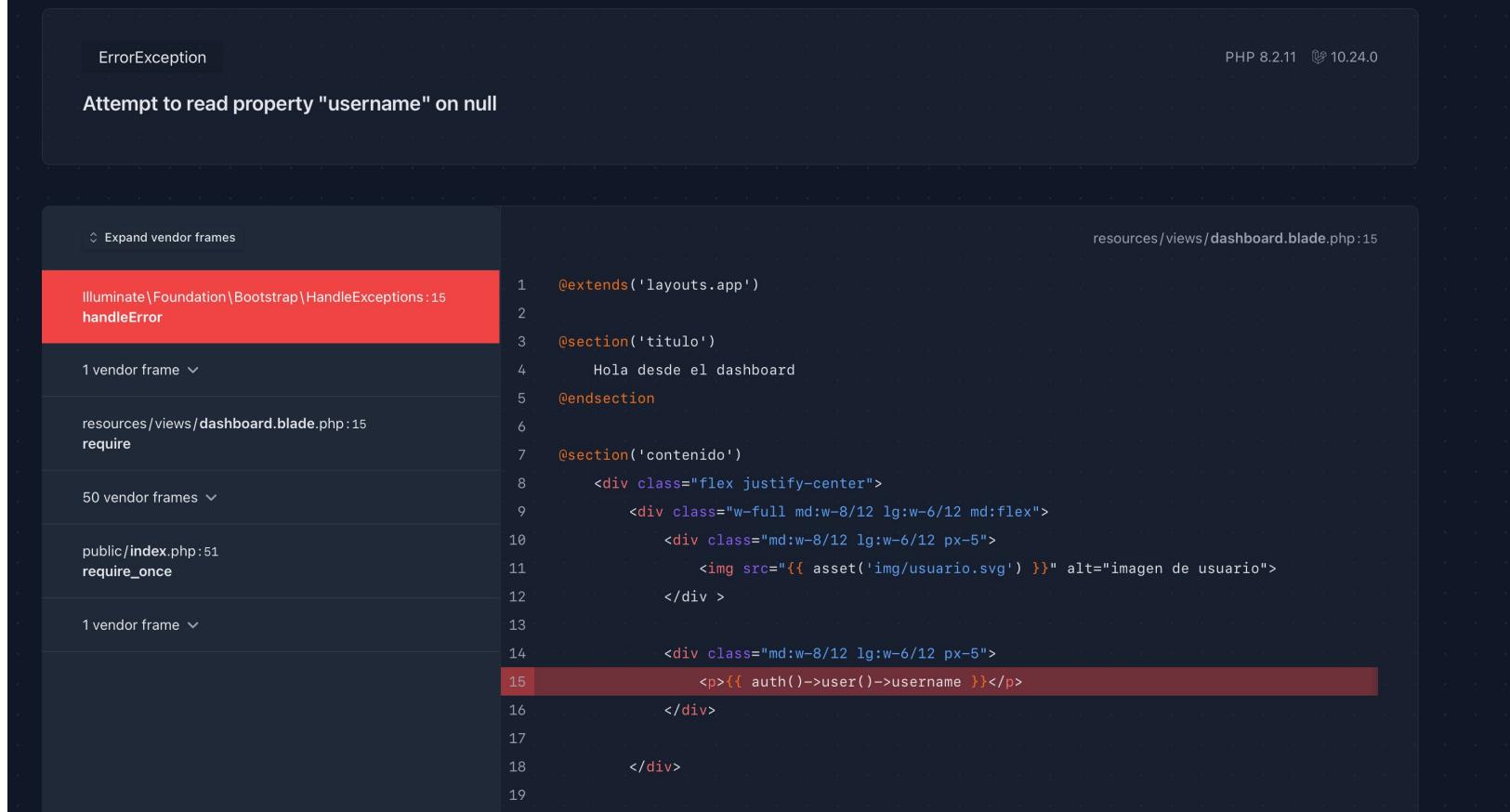
```
public/index.php:51
require_once
```

1 vendor frame ▾
- Code Snippet:**

```
1 @extends('layouts.app')
2
3 @section('titulo')
4     Hola desde el dashboard
5 @endsection
6
7 @section('contenido')
8     <div class="flex justify-center">
9         <div class="w-full md:w-8/12 lg:w-6/12 md:flex">
10            <div class="md:w-8/12 lg:w-6/12 px-5">
11                
12            </div>
13
14            <div class="md:w-8/12 lg:w-6/12 px-5">
15                <p>{{ auth()>user()>username }}</p>
16            </div>
17
18        </div>
```

Autenticación

Nos dice que está intentando leer username de un usuario autenticado, pero no hemos autenticado a nadie en esta pestaña del navegador



The screenshot shows a Laravel error page with the following details:

- Error Type:** ErrorException
- Message:** Attempt to read property "username" on null
- Environment:** PHP 8.2.11 | 10.24.0
- File:** resources/views/dashboard.blade.php:15
- Stack Trace:** Illuminate\Foundation\Bootstrap\HandleExceptions::15 handleError
- Code Snippet:**

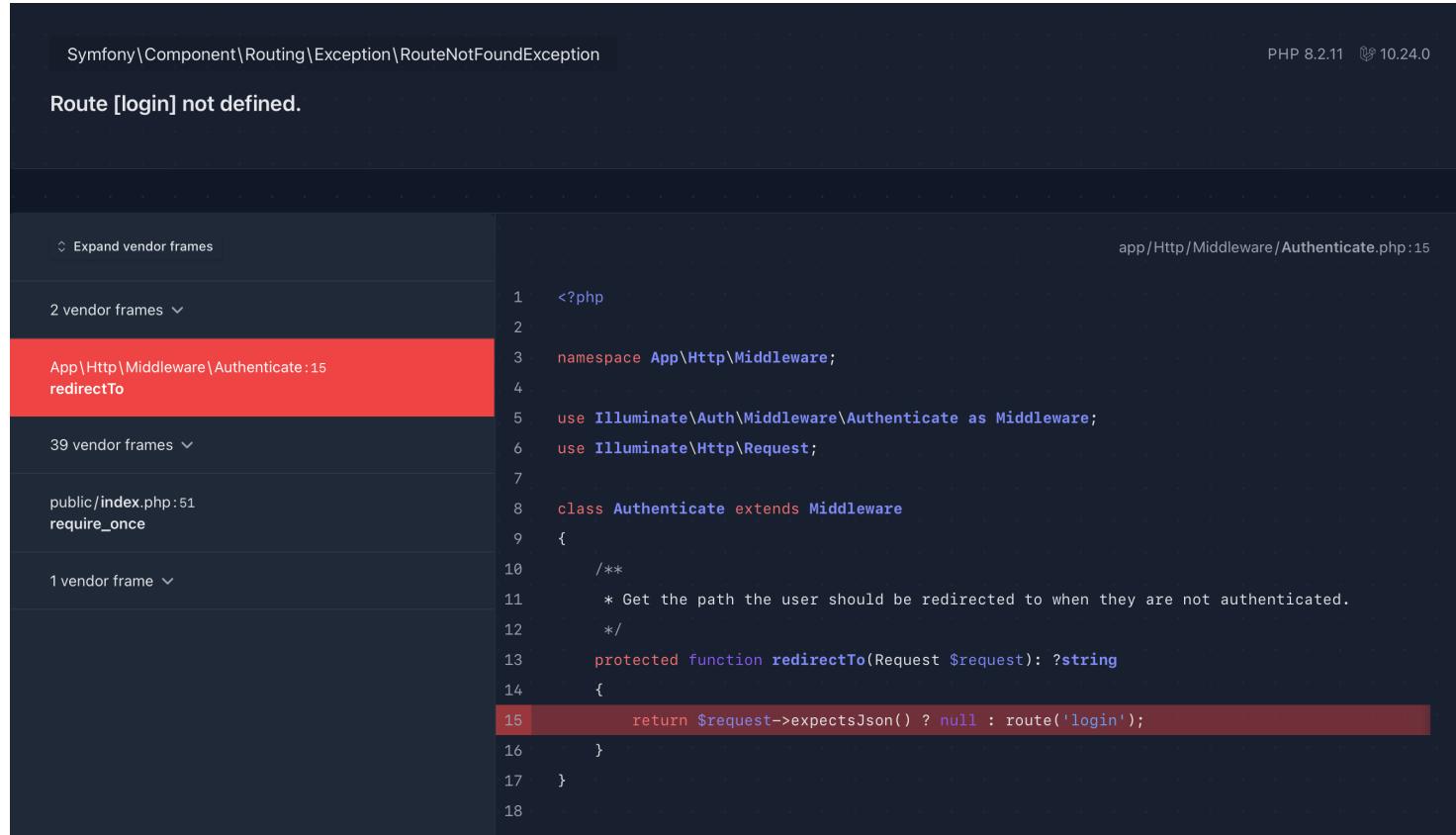
```
1 @extends('layouts.app')
2
3 @section('titulo')
4     Hola desde el dashboard
5 @endsection
6
7 @section('contenido')
8     <div class="flex justify-center">
9         <div class="w-full md:w-8/12 lg:w-6/12 md:flex">
10            <div class="md:w-8/12 lg:w-6/12 px-5">
11                
12            </div>
13
14            <div class="md:w-8/12 lg:w-6/12 px-5">
15                <p>{{ auth()>user()>username }}</p>
16            </div>
17
18        </div>
```

Autenticación

Así que debemos proteger el controlador PostController para que el usuario pueda crear su contenido de forma tranquila. Es por esto que vamos a crear un constructor y dentro usaremos la propiedad :

```
$this->middleware('auth');
```

Con esto le indicamos al código que antes de ejecutar el index, proteja la autenticación, es decir, que el usuario este autenticado, si validamos en la pagina privada, evidenciaremos que el error ahora es que no existe la vista de login.



Symfony\Component\Routing\Exception\RouteNotFoundException
Route [login] not defined.

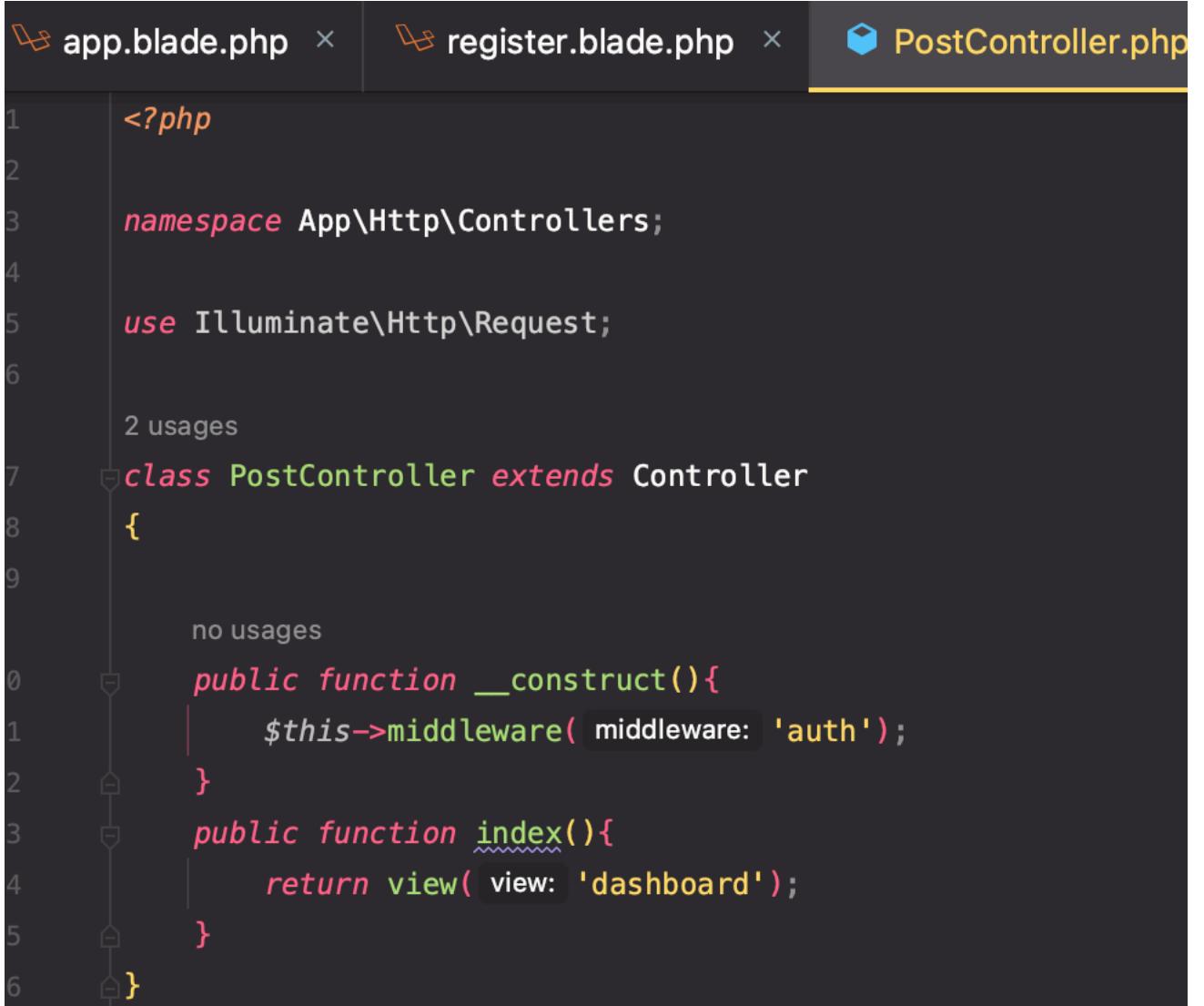
PHP 8.2.11 | 10.24.0

app\Http\Middleware\Authenticate.php:15

```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Illuminate\Auth\Middleware\Authenticate as Middleware;
6 use Illuminate\Http\Request;
7
8 class Authenticate extends Middleware
9 {
10     /**
11      * Get the path the user should be redirected to when they are not authenticated.
12      */
13     protected function redirectTo(Request $request): ?string
14     {
15         return $request->expectsJson() ? null : route('login');
16     }
17 }
18 }
```

Autenticación

Hasta ahora nuestro controlador se ve así:



```
<?php  
  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
2 usages  
class PostController extends Controller  
{  
  
    no usages  
    public function __construct(){  
        $this->middleware( middleware: 'auth' );  
    }  
    public function index(){  
        return view( view: 'dashboard' );  
    }  
}
```

Autenticación



Tal como se observó hace un momento hay que crear la ruta, es por esto que en web.php se creara la ruta asi:

```
Route::get('/login', [\App\Http\Controllers\LoginController::class, 'index'])->name('login');
```

Ahora en el controlador, se creará el método de index:

```
class LoginController extends Controller
{
    public function index(){
        return view( view: 'auth.login');
    }
}
```

Y en vistas crearemos auth/login.Blade.php. Como el login es similar al registro, copiare todo y modificaré para que se vea solo lo que quiero autenticar en el login.

Autenticación

Nuestro login se debe observar asi:

Devstagram

LOGIN CREAR CUENTA

Inicia sesión en devstagram



E-MAIL

PASSWORD

INICIAR SESIÓN

Validación en el formulario login



Hasta el momento cuando creamos un nuevo usuario automáticamente este queda logueado en nuestro sistema, así que crearemos esta misma funcionalidad para el login que acabamos de crear.

Para esto vamos a abrir web.php y duplicare la línea que se creo con la ruta de login pero en lugar de usar el método get, voy a usar el método post:

```
Route::post('/login', [\App\Http\Controllers\LoginController::class, 'store']);
```

Solo que el método en este caso será store y debido a que estamos trabajando bajo la misma url, también eliminare el name.

Ahora en el LoginController, creare el método store con un dd que diga autenticando. Recuerden que ese dd es como si usaran un console.log en js, simplemente lo usamos para autenticar que el endpoint se este comunicando correctamente.

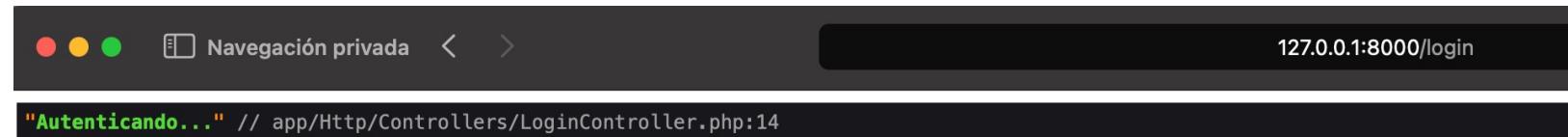
Validación en el formulario login



Seguidamente iremos a login.blade.php y en el formulario escribiremos method post y agregaremos un action que dirija el código hacia {{ route('login') }}

```
<form method="post" action="{{ route('login') }}" novalidate>
```

Recargando la pagina y pinchando en iniciar sesión:



Por lo que se entiende que nuestro endpoint está funcionando correctamente.

Una vez logramos esta verificación eliminamos el dd y le pasamos Request

Validación en el formulario login



Pasando Request se ve así nuestro controlador:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class LoginController extends Controller
{
    public function index(){
        return view( view: 'auth.login');
    }

    public function store(Request $request){
        $this->validate($request, [
            'email' =>'required|email',
            'password' =>'required'
        ]);
    }
}
```

Validación en el formulario login



Y si intentamos iniciar sesión sin poner ningún dato en los campos de escritura de login:

Devstagram

LOGIN CREAR CUENTA

A screenshot of a web browser displaying the Devstagram login page. The page has a header with the Devstagram logo and navigation links for LOGIN and CREAR CUENTA. Below the header is a sub-header "Inicia sesión en devstagram". The main content area features a dark-themed background image of a person working at a computer. To the right is a form with two required fields: "E-MAIL" and "PASSWORD". Both fields have red validation error messages below them: "El campo email es requerido." and "El campo password es requerido." A blue "INICIAR SESIÓN" button is at the bottom of the form. At the very bottom of the page, a footer note reads "DEVSTAGRAM - TODOS LOS DERECHOS RESERVADOS 2023-10-13 03:11:19".

Inicia sesión en devstagram

E-MAIL

E-mail

El campo email es requerido.

PASSWORD

password

El campo password es requerido.

INICIAR SESIÓN

DEVSTAGRAM - TODOS LOS DERECHOS RESERVADOS 2023-10-13 03:11:19

Validación en el formulario login



Finalmente vamos a autenticar a los usuarios desde el formulario ya que anteriormente lo hicimos, pero cuando el usuario creaba su cuenta por medio del attempt, iniciaba. ¿Pero y que si el usuario escribe mal sus credenciales? O ¿que si alguien intenta ingresar por él?

Para esto iremos a LoginController y utilizaremos un if que comprueba si las credenciales son correctas, para este caso, si el usuario no se puede loguear:

```
public function store(Request $request){
    $this->validate($request, [
        'email' =>'required|email',
        'password' =>'required'
    ]);

    if (!auth()->attempt($request->only( keys: 'email', 'password'))){
        return back()->with('mensaje', 'Credenciales incorrectas');
    }
}
```

Validación en el formulario login

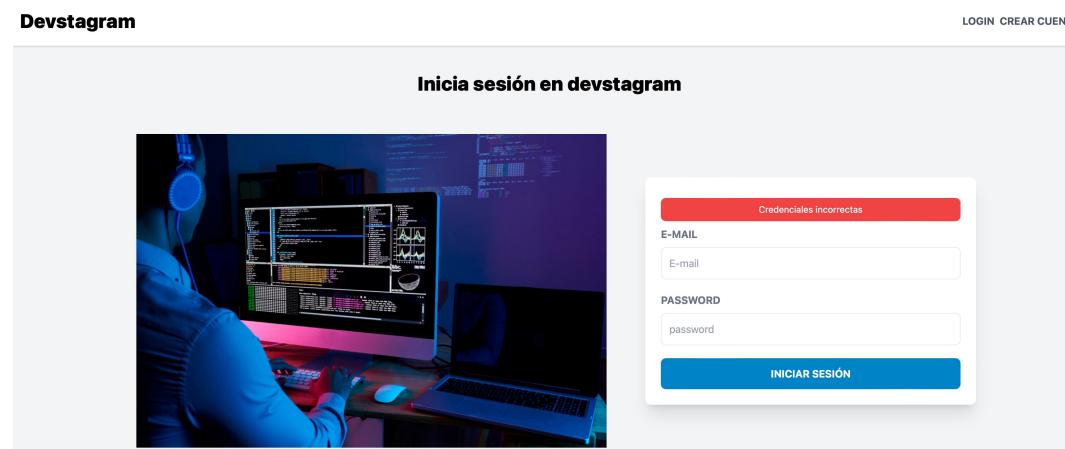


El código anterior colocara el mensaje en una sesión, por lo que vamos a ir a login.blade.php y luego del token crearemos una section para un if que valide lo siguiente:

```
<form method="post" action="{{ route('login') }}" novalidate>
    @csrf

    @if(session('mensaje'))
        <p class="bg-red-500 text-white my-2 rounded-lg text-sm p-2 text-center">{{ session('mensaje') }}</p>
    @endif
```

Copiando los estilos de error le digo que si hay un mensaje, simplemente lo imprima.



Validación en el formulario login



Ahora, que es el with() que usamos en el controlador:

Es una forma de llenar los valores de la sesión que usamos en la vista, por lo que en mi template puedo comprobar la sesión y mostrar mensaje; son bastante útiles por que los puedo crear en mi controlador y consumirlos desde la vista.

Por otro lado, que es el back():

Cuando llenamos el formulario usamos el metodo post para llevar datos a la url, es por esto que si usamos el back devolveremos los valores a la página anterior, es decir, a la página donde estamos llenando el formulario; es demasiado útil ya que no tenemos que redireccionar al usuario, sino que lo devolvemos a la vista, pero en esta ocasión con un mensaje de error.

Asi que básicamente con el siguiente código, le indicamos al software que debe regresar a la pagina anterior con un mensaje específico

```
return back()->with('mensaje','Credenciales incorrectas');
```

Validación en el formulario login



Ya vimos que pasa si no se logra autenticar el usuario, pero y ¿que si lo hace de forma correcta?

Deberíamos permitirle ingresar a su muro, así que luego del if que creamos anteriormente, retornaremos el caso contrario

```
public function store(Request $request){
    $this->validate($request, [
        'email' =>'required|email',
        'password' =>'required'
    ]);

    if (!auth()->attempt($request->only( keys: 'email', 'password'))){
        return back()->with('mensaje', 'Credenciales incorrectas');
    }

    return redirect()->route( route: 'post.index');
}
```

Validación en el formulario login



Si todo va bien hasta ahora, deberíamos ver esto:

The screenshot shows a user interface for a social media platform named 'Devstagram'. At the top left is the brand name 'Devstagram' in a bold, black, sans-serif font. On the far right, there are two links: 'LOGIN' and 'CREAR CUENTA', both in a smaller, black, sans-serif font. The main content area has a light gray background. In the center, the text 'Hola desde el dashboard' is displayed in a bold, black, sans-serif font. Below this text is a large, circular, gray placeholder icon containing a white silhouette of a person's head and shoulders. To the right of the icon, the text 'diego-kld-23' is written in a small, black, sans-serif font. At the bottom of the page, a thin horizontal bar contains the text 'DEVSTAGRAM - TODOS LOS DERECHOS RESERVADOS 2023-10-13 03:28:50' in a very small, black, sans-serif font.

Aun así, sigue saliendo login sabiendo que ya ingresamos, por lo que debemos cambiar eso por cerrar sesión



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