



Análisis y desarrollo de software.

ID: 2644590



www.sena.edu.co

@SENAComunica

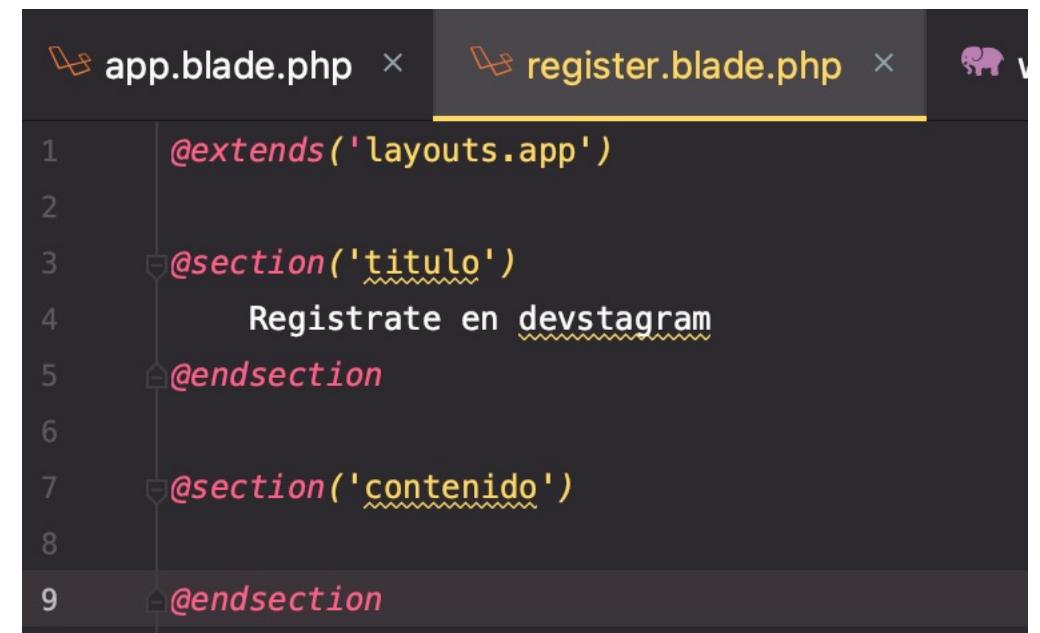
Instructor

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

Creando mi primer formulario

En este punto ya se creó el primer hola mundo y también nuestro primer controlador, así que es hora de crear nuestro primer formulario para empezar a conectar nuestro modelo con nuestro controlador.

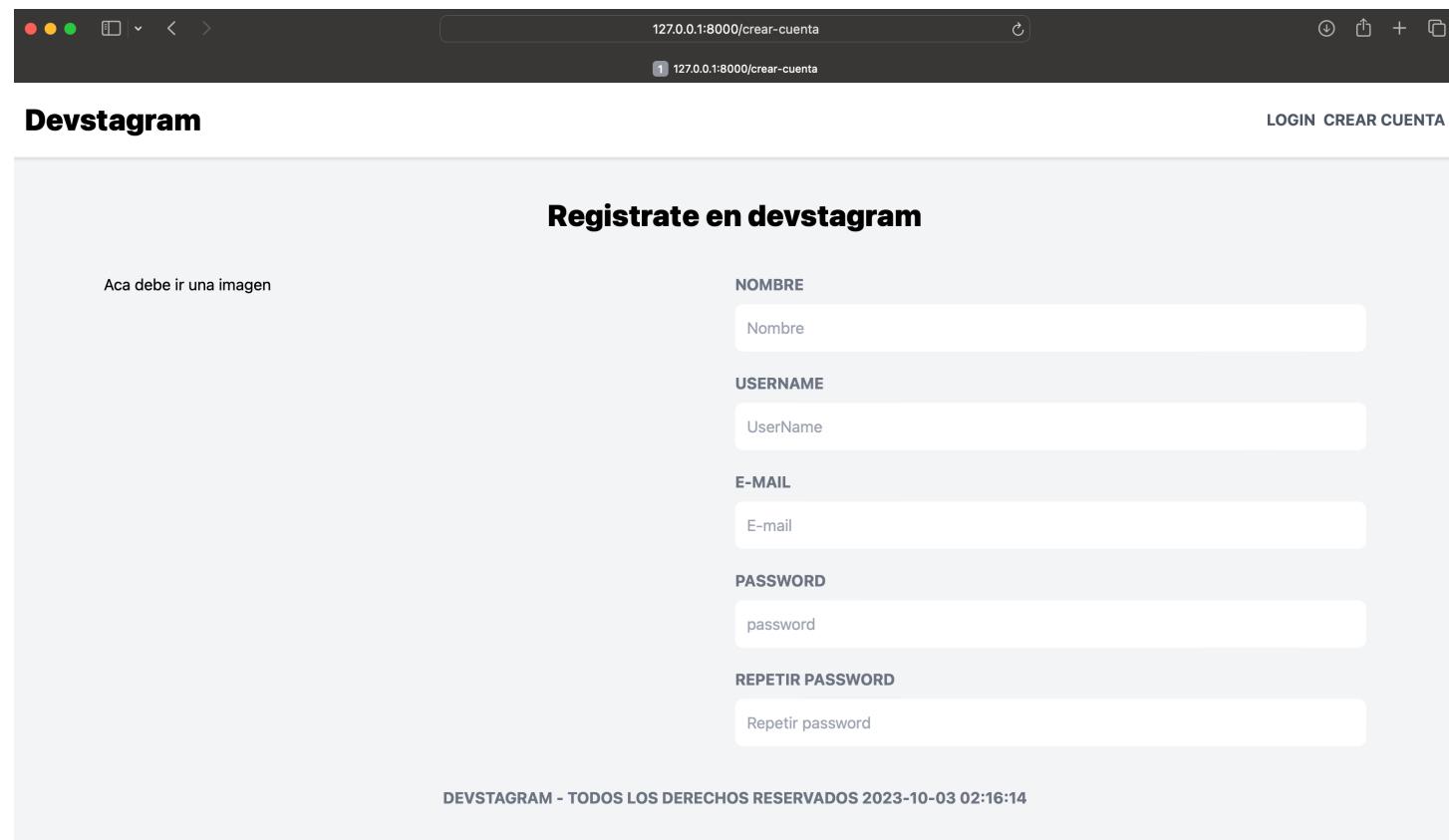
Partiremos trabajando en la vista de registro, así que crearemos una sección para contenido:



```
1 @extends('layouts.app')
2
3 @section('titulo')
4     Regístrate en devstagram
5 @endsection
6
7 @section('contenido')
8
9 @endsection
```

Creando mi primer formulario

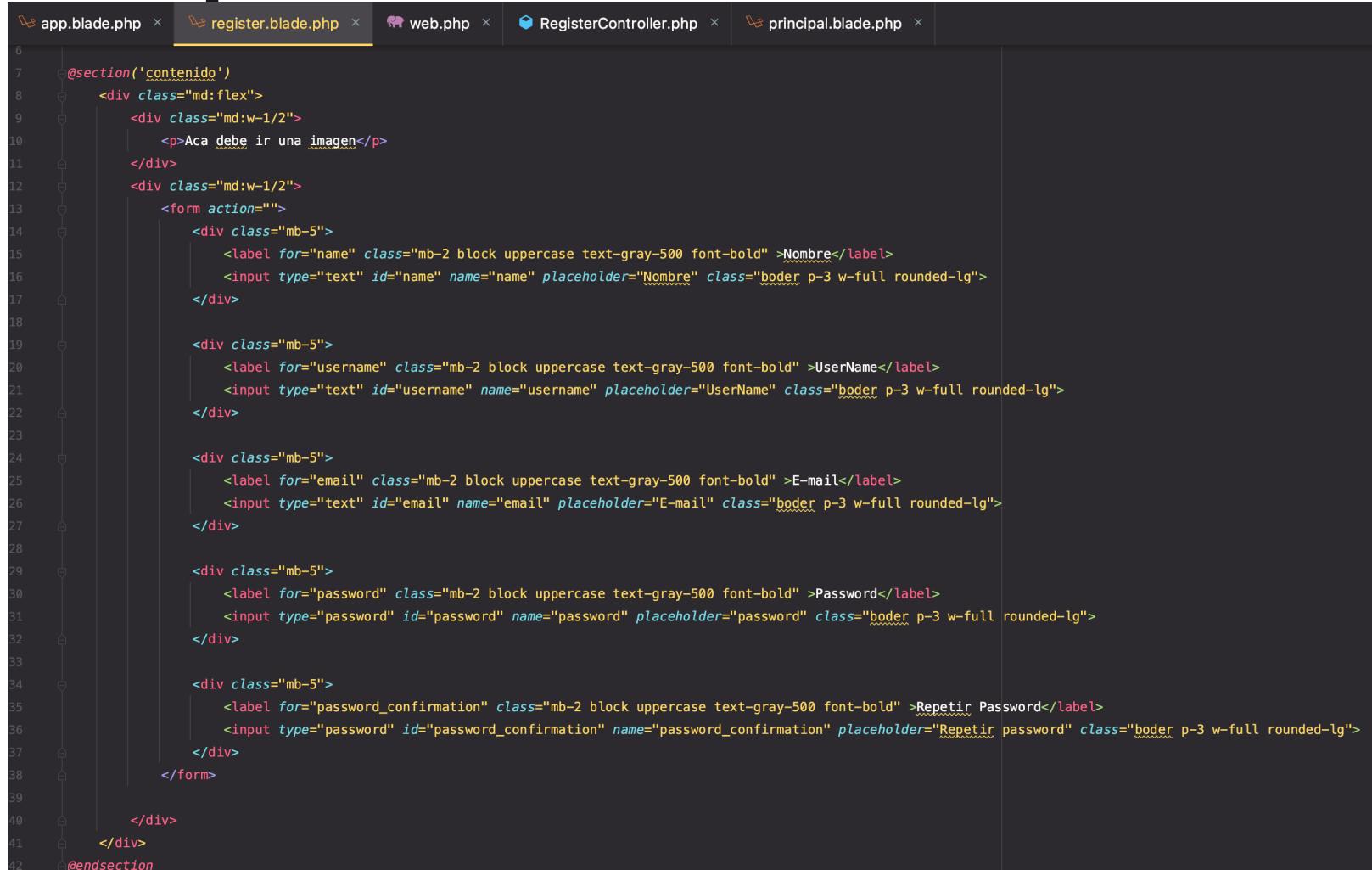
Seguidamente empezamos a crear los campos del formulario tales como nombre, Nick, password, confirm password y todos los que creamos necesarios. Para este ejemplo va a crear:



Creando mi primer formulario

Seguidamente empezamos a crear los campos del formulario tales como nombre, Nick, password, confirm password y todos los que creamos necesarios. Para este ejemplo va a crear:

Nota: En confirmar password encontramos “password_confirmation”; este comando le indica a laravel que debe ir a revisar el input anterior que contenga password y así poder hacer la validación, si se pone de otra forma, tendrá que implementar dicha lógica.



The screenshot shows a code editor with several tabs open at the top: app.blade.php, register.blade.php (which is currently selected), web.php, RegisterController.php, and principal.blade.php. The register.blade.php file contains the following blade template code:

```
<@section('contenido')>
    <div class="md:flex">
        <div class="md:w-1/2">
            <p>Aca debe ir una imagen</p>
        </div>
        <div class="md:w-1/2">
            <form action="">
                <div class="mb-5">
                    <label for="name" class="mb-2 block uppercase text-gray-500 font-bold" >Nombre</label>
                    <input type="text" id="name" name="name" placeholder="Nombre" class="border p-3 w-full rounded-lg">
                </div>

                <div class="mb-5">
                    <label for="username" class="mb-2 block uppercase text-gray-500 font-bold" >UserName</label>
                    <input type="text" id="username" name="username" placeholder="UserName" class="border p-3 w-full rounded-lg">
                </div>

                <div class="mb-5">
                    <label for="email" class="mb-2 block uppercase text-gray-500 font-bold" >E-mail</label>
                    <input type="text" id="email" name="email" placeholder="E-mail" class="border p-3 w-full rounded-lg">
                </div>

                <div class="mb-5">
                    <label for="password" class="mb-2 block uppercase text-gray-500 font-bold" >Password</label>
                    <input type="password" id="password" name="password" placeholder="password" class="border p-3 w-full rounded-lg">
                </div>

                <div class="mb-5">
                    <label for="password_confirmation" class="mb-2 block uppercase text-gray-500 font-bold" >Repetir Password</label>
                    <input type="password" id="password_confirmation" name="password_confirmation" placeholder="Repetir password" class="border p-3 w-full rounded-lg">
                </div>
            </form>
        </div>
    </div>
<@endsection>
```

Creando mi primer formulario

Finalmente agrego un botón con un input así:

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

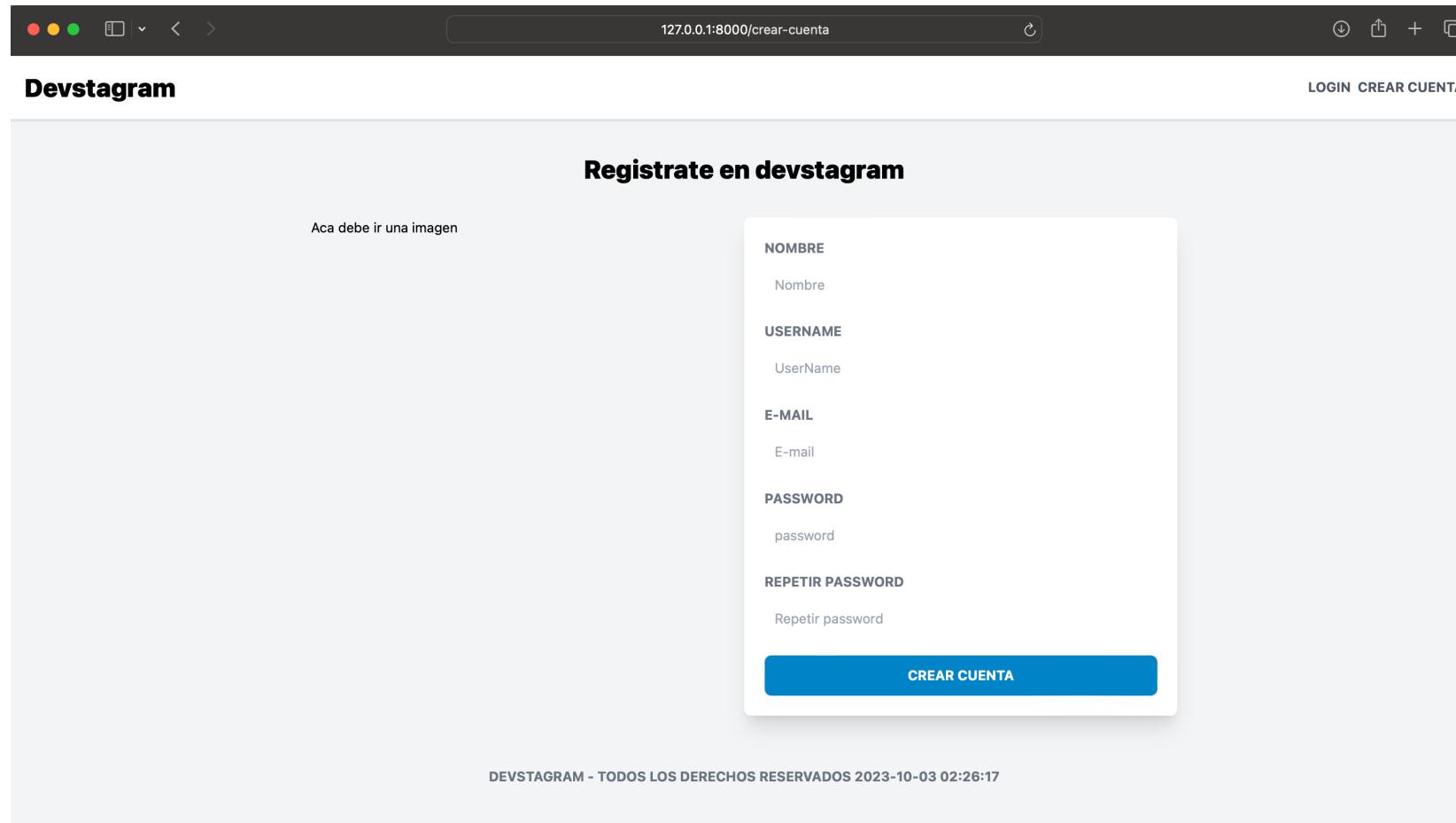
Y se ve de la siguiente forma



CREAR CUENTA

Creando mi primer formulario

Hasta el momento, y basado en su conocimiento debe crear una vista que este de esta forma:



The screenshot shows a web browser window with the URL `127.0.0.1:8000/crear-cuenta`. The page title is **Devstagram**. On the left, there is a placeholder text "Aca debe ir una imagen". On the right, there is a registration form titled **Regístrate en devstagram**. The form fields are as follows:

Label	Placeholder
NOMBRE	Nombre
USERNAME	UserName
E-MAIL	E-mail
PASSWORD	password
REPETIR PASSWORD	Repetir password

At the bottom of the form is a blue button labeled **CREAR CUENTA**.

At the very bottom of the page, the footer text reads: **DEVSTAGRAM - TODOS LOS DERECHOS RESERVADOS 2023-10-03 02:26:17**.

Creando mi primer formulario

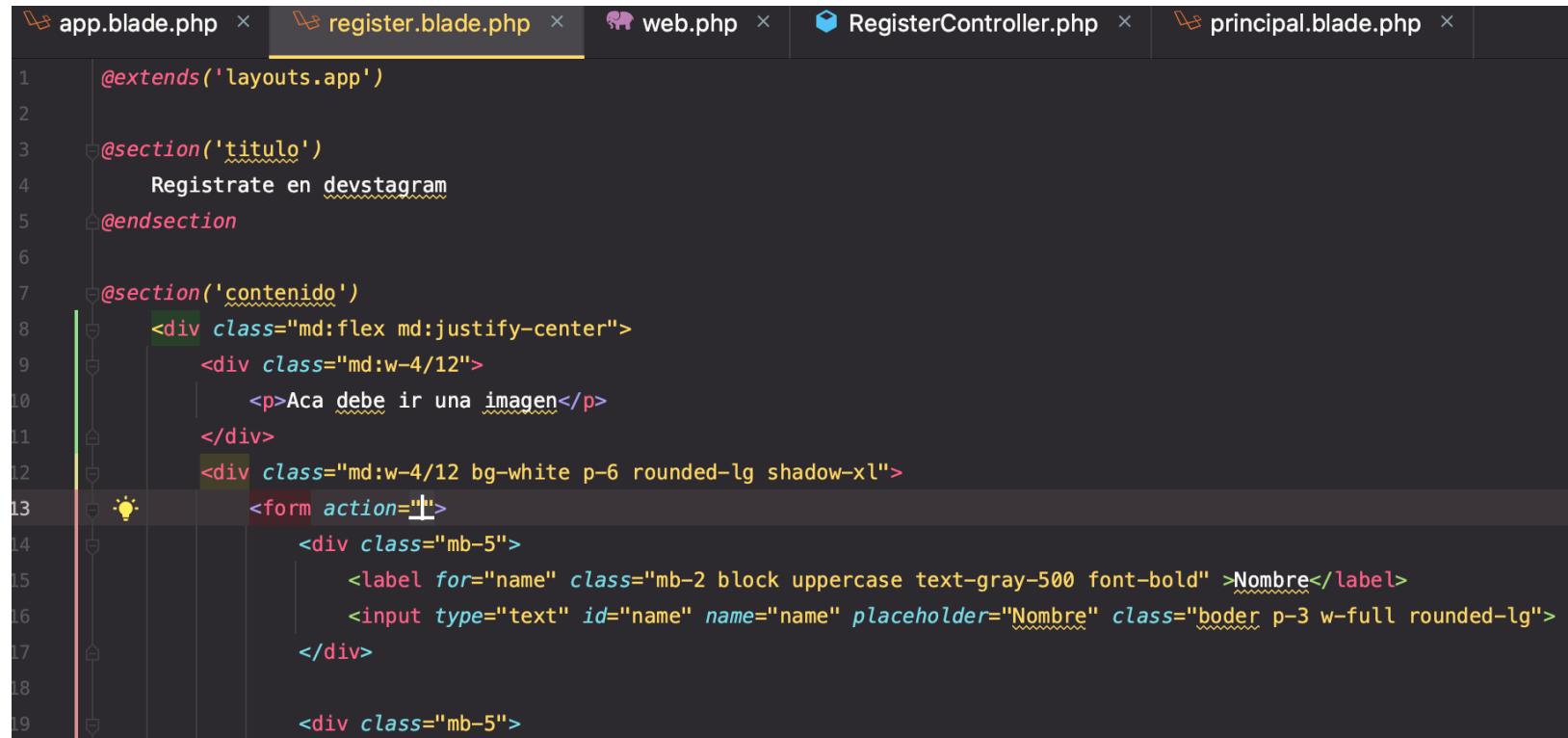
Ahora vamos a cargar una imagen que le dé lucides a nuestro formulario y vista de registro:

Para esto debemos copiar las imágenes compartidas por el instructor en la carpeta del proyecto en la ruta

Devstagram/public/img

Si no existe el directorio img, debe crearlo y luego pegar las imágenes dentro.

Reemplazamos “Aca debe ir una imagen”



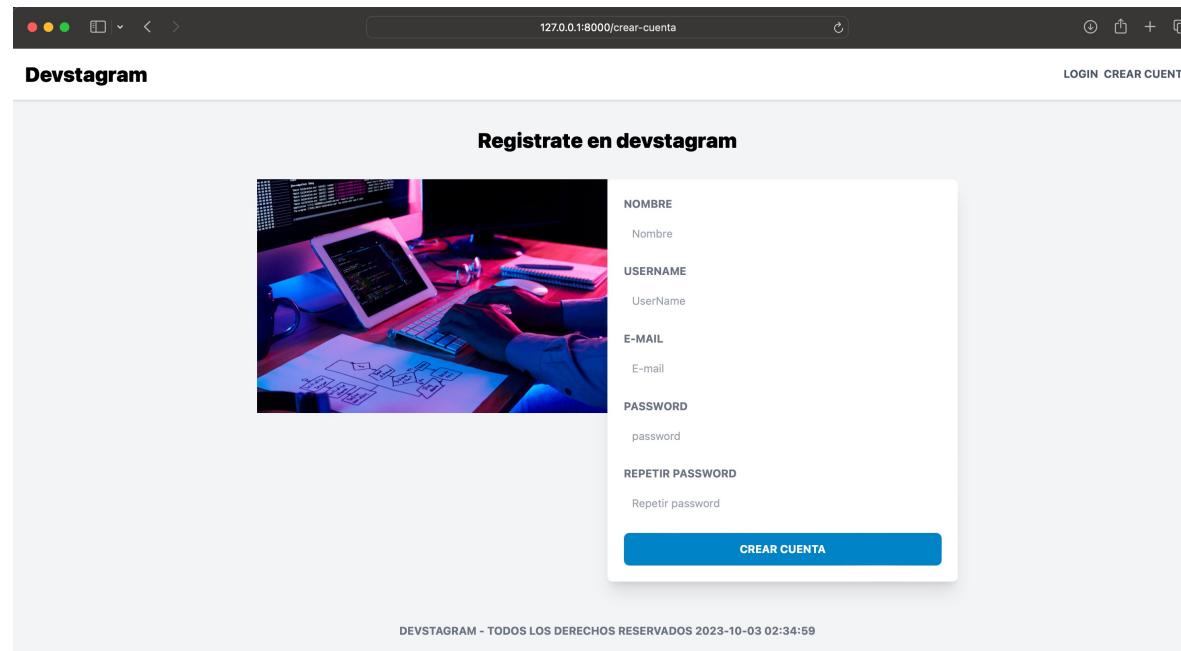
```
app.blade.php x register.blade.php x web.php x RegisterController.php x principal.blade.php x
1 @extends('layouts.app')
2
3 @section('titulo')
4     Registrate en devstagram
5 @endsection
6
7 @section('contenido')
8     <div class="md:flex md:justify-center">
9         <div class="md:w-4/12">
10            <p>Aca debe ir una imagen</p>
11        </div>
12        <div class="md:w-4/12 bg-white p-6 rounded-lg shadow-xl">
13            <form action="#">
14                <div class="mb-5">
15                    <label for="name" class="mb-2 block uppercase text-gray-500 font-bold">Nombre</label>
16                    <input type="text" id="name" name="name" placeholder="Nombre" class="border p-3 w-full rounded-lg">
17                </div>
18
19                <div class="mb-5">
```

Creando mi primer formulario

Por:

```
<div class="md:w-4/12">
    
</div>
```

Teniendo este resultado:



Framework

Creando

Devstagram



Dándole un poco de estilo a la imagen se tiene:



Regístrate en devstagram

NOMBRE

Nombre

USERNAME

UserName

E-MAIL

E-mail

PASSWORD

password

REPETIR PASSWORD

Repetir password

CREAR CUENTA

Creando mi primer formulario

Y nuestro código se debe ver así:

```
  app.blade.php x register.blade.php x web.php x RegisterController.php x principal.blade.php x A17
6
7     @section('contenido')
8         <div class="md:flex md:justify-center md:gap-10 md:items-center">
9             <div class="md:w-6/12 p-5">
10                 
11             </div>
12             <div class="md:w-4/12 bg-white p-6 rounded-lg shadow-xl">
13                 <form action="">
14                     <div class="mb-5">
15                         <label for="name" class="mb-2 block uppercase text-gray-500 font-bold" >Nombre</label>
16                         <input type="text" id="name" name="name" placeholder="Nombre" class="border p-3 w-full rounded-lg">
17                     </div>
18                     <div class="mb-5">
19                         <label for="username" class="mb-2 block uppercase text-gray-500 font-bold" >UserName</label>
20                         <input type="text" id="username" name="username" placeholder="UserName" class="border p-3 w-full rounded-lg">
21                     </div>
22                     <div class="mb-5">
23                         <label for="email" class="mb-2 block uppercase text-gray-500 font-bold" >E-mail</label>
24                         <input type="text" id="email" name="email" placeholder="E-mail" class="border p-3 w-full rounded-lg">
25                     </div>
26                     <div class="mb-5">
27                         <label for="password" class="mb-2 block uppercase text-gray-500 font-bold" >Password</label>
28                         <input type="password" id="password" name="password" placeholder="password" class="border p-3 w-full rounded-lg">
29                     </div>
30                     <div class="mb-5">
31                         <label for="password_confirmation" class="mb-2 block uppercase text-gray-500 font-bold" >Repetir Password</label>
32                         <input type="password" id="password_confirmation" name="password_confirmation" placeholder="Repetir password" class="border p-3 w-full rounded-lg">
33                     </div>
34
35                         <input type="submit" value="Crear cuenta" class="bg-sky-600 hover:bg-sky-700 transition-colors cursor-pointer uppercase font-bold w-full p-3 text-white rounded-lg">
36                     </form>
37                 </div>
38             </div>
39         @endsection
```

¿Qué tipos de request existen?

En HTTP existen diferentes tipos de Request o tipos de Petición: GET, POST, PUT, PATCH y DELETE.

GET es el más simple; cuando visitas un sitio web por default es un GET, y el método solo se utiliza para recuperar datos, pero nunca debe enviar datos.

POST se utiliza cuando mandas datos a un servidor; esto incluye información que llenas en un formulario o buscador

PUT es utilizado para actualizar un elemento; pero si no existe crea uno nuevo. PUT es un reemplazo total de un registro.

PATCH es utilizado para actualizar parcialmente un elemento o recurso.

DELETE se utiliza para eliminar un recurso o elemento.



Creando mi primer formulario

El lugar en donde validamos el tipo de solicitud es en el archivo web.php, es por esto que vamos a crear una nueva ruta hacia el método store pero esta vez usando el método post y no get así:

A su vez, en el controlador debemos crear el método store así:

Por ahora si en el action del formulario de registro nos dirigimos a /crear-cuenta y usamos el método POST encontraremos un error 419 | PAGE EXPIRED

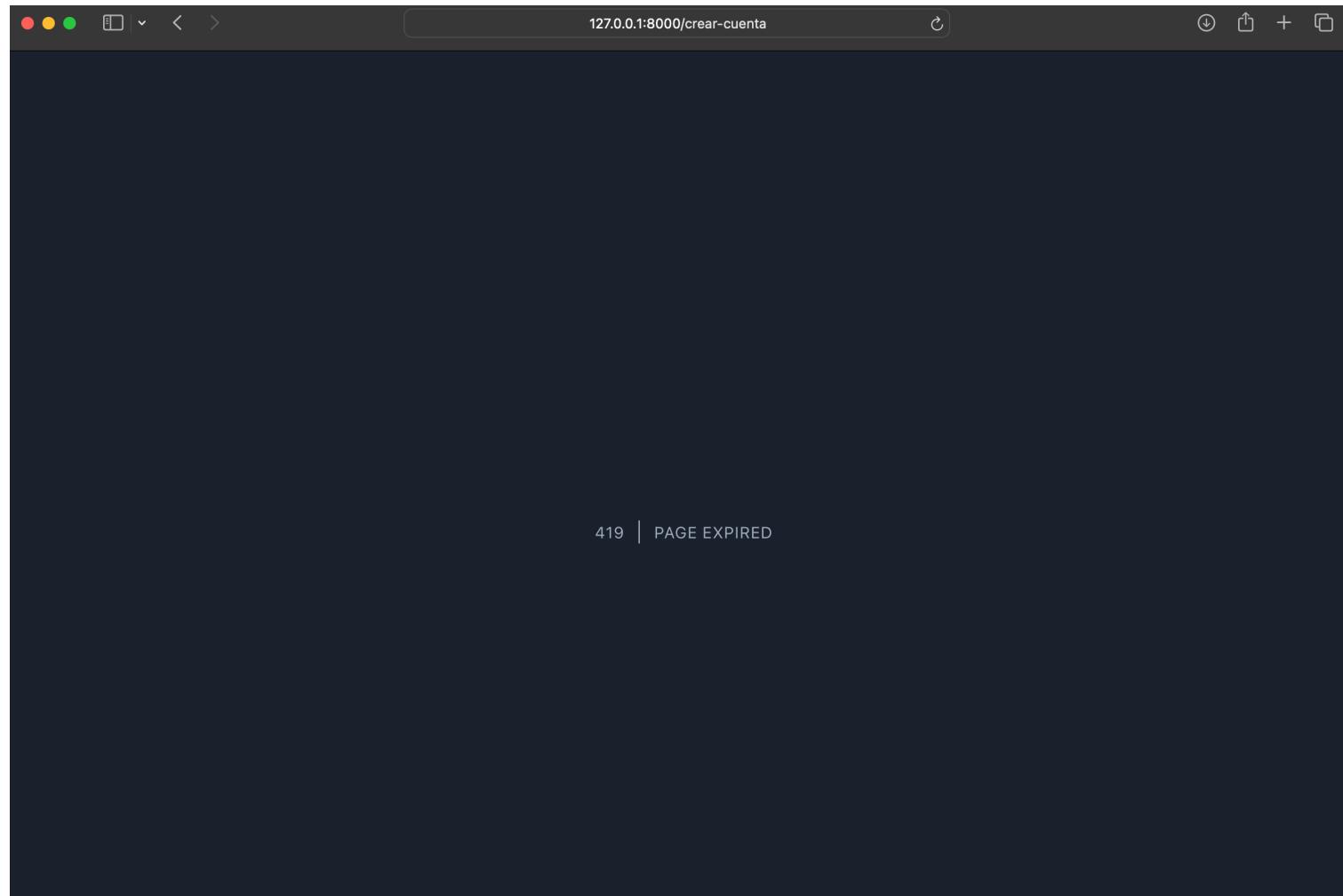
```
Route::get('/', function () {
    return view('principal');
});
Route::get('/crear-cuenta', [RegisterController::class, 'index']);
Route::post('/crear-cuenta', [RegisterController::class, 'store']);
```

```
class RegisterController extends Controller
{
    public function index()
    {
        return view('auth.register');
    }
    public function store()
    {
        dd(...vars: 'Post...');

    }
}
```

Creando mi primer formulario

El error 419 | PAGE EXPIRED es conocido como cross site request y es un ataque básico generado por usuarios mal intencionados, pero gracias a que laravel es robusto, nosotros como desarrolladores tenemos la ventaja de estar protegidos, aun así, veamos cómo protegernos y corregir este fallo.



Creando mi primer formulario

Para corregir error 419 | PAGE EXPIRED
Debemos ir a register.Blade.php y debajo del formulario agregar la etiqueta:

@csrf

Si recargamos la página y le damos inspeccionar, podremos ver como en donde se crea el formulario, se crea un token de validación, junto de un campo vacío.

```
<div class="md:w-4/12 bg-white p-6 rounded-lg shadow-xl">
  <form action="/crear-cuenta" method="POST">
    @csrf
```

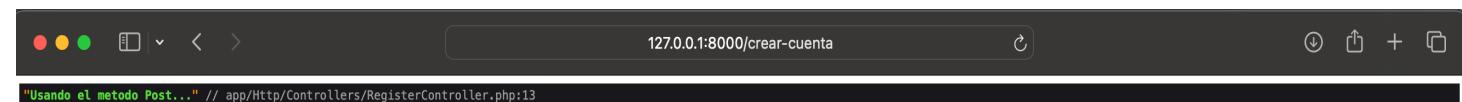
```
  <input type="hidden" name="_token" value="pQEmNwMpWxio5TYhcD0SM0zWCeXZhV04L4J2gZ5" autocomplete="off">
  <div class="mb-5">...</div>
  <div class="mb-5">...</div>
```

Creando mi primer formulario

Si recargamos la pagina del formulario nuevamente y le damos enviar, veremos como no solo pasa la primera validación, si no que nos trae el contenido del controlador store.

```
class RegisterController extends Controller
{
    1 usage
    public function index() {
        return view('auth.register');
    }
    1 usage
    public function store() {
        dd(...vars: 'Usando el metodo Post...');

    }
}
```



Hablemos un poco de los controllers

Los Controllers van a ayudarte a tener un código mejor organizado, además de una separación mayor en la funcionalidad de tus aplicaciones y sitios web.

Laravel tiene una convención a la hora de nombrar los métodos de tus controllers conocida como Resource Controllers. Esta convención ayuda bastante para tener todo mejor organizado.



Verbo Http	URI	Acción	Ruta
GET	/clientes	index	clientes.index
POST	/clientes	store	clientes.store
DELETE	/clientes/{cliente}	destroy	clientes.destroy

<https://laravel.com/docs/10.x/controllers#actions-handled-by-resource-controller>

Darles nombre a las rutas

A medida que va creciendo nuestro código, vemos como cada vez que necesitamos hacer un cambio es mas complejo ya que debemos cambiar o renombrar en muchas partes los tags que hemos venido asignado, es por esto, que laravel nos permite darle nombre a las rutas para que no importe como llamemos la url a la que queramos acceder, podremos ingresar usando el nombre de la ruta, ejemplo:

Crearemos el nombre de la ruta en el archivo web.php

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

Route::get( uri: '/crear-cuenta', [RegisterController::class, 'index'])->name( name: 'register');

Route::post( uri: '/crear-cuenta', [RegisterController::class, 'store']);
```

Luego para este ejemplo, cambiaremos la ruta en app.Blade.php de /crear-cuenta por:

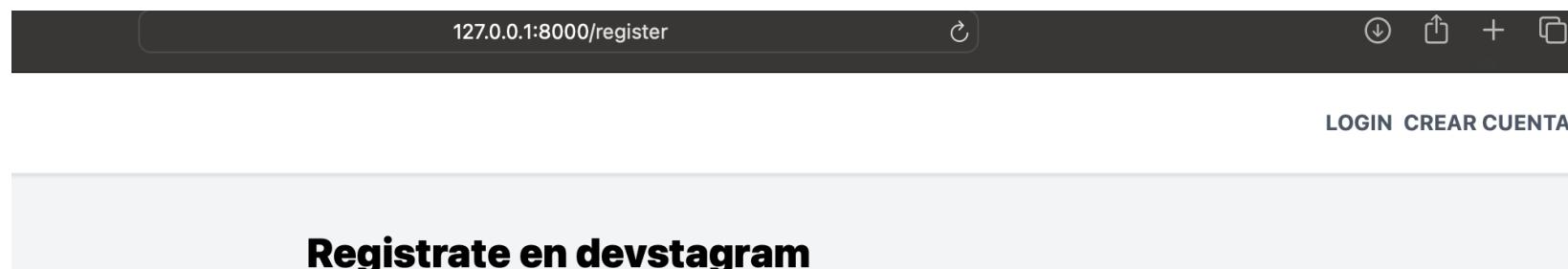
```
<a class="font-bold uppercase text-gray-600" href="{{route('register')}}>
    Crear cuenta
</a>
```

Darles nombre a las rutas

Ahora, si recargamos la página y vamos al inicio, veremos que es posible volver a nuestra página de registro sin ningún problema. Vamos a cambiar donde exista la ruta /crear-cuenta por {{ route('register') }} (app, web y register), para finalizar este ejemplo, voy a cambiar en nuestra ruta /crear-cuenta y recargo la página.

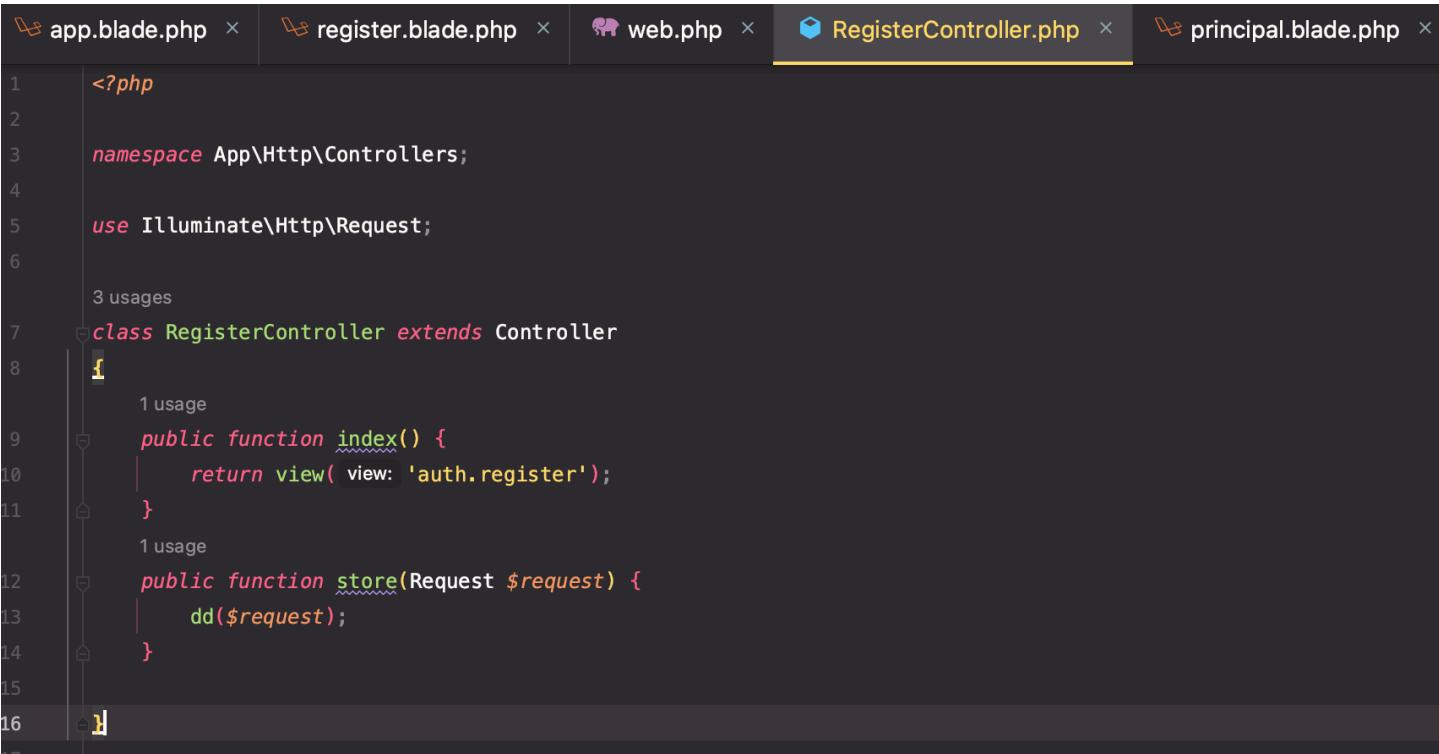
```
Route::get( uri: '/', function () {
    return view( view: 'principal');
});
Route::get( uri: '/register', [RegisterController::class, 'index'])->name( name: 'register');
Route::post( uri: '/register', [RegisterController::class, 'store']);
```

Evidenciaremos que lo único que cambia es la url, sin ningún problema en algún otro lado.



¿Cómo leer los datos del formulario?

Primero iremos hasta el controlador y en el método store, le pasaremos el método Request y crearemos una variable \$request y usando el comando dd que sirve para debug, observemos que resultado se obtiene.



```
<?php

namespace App\Http\Controllers;

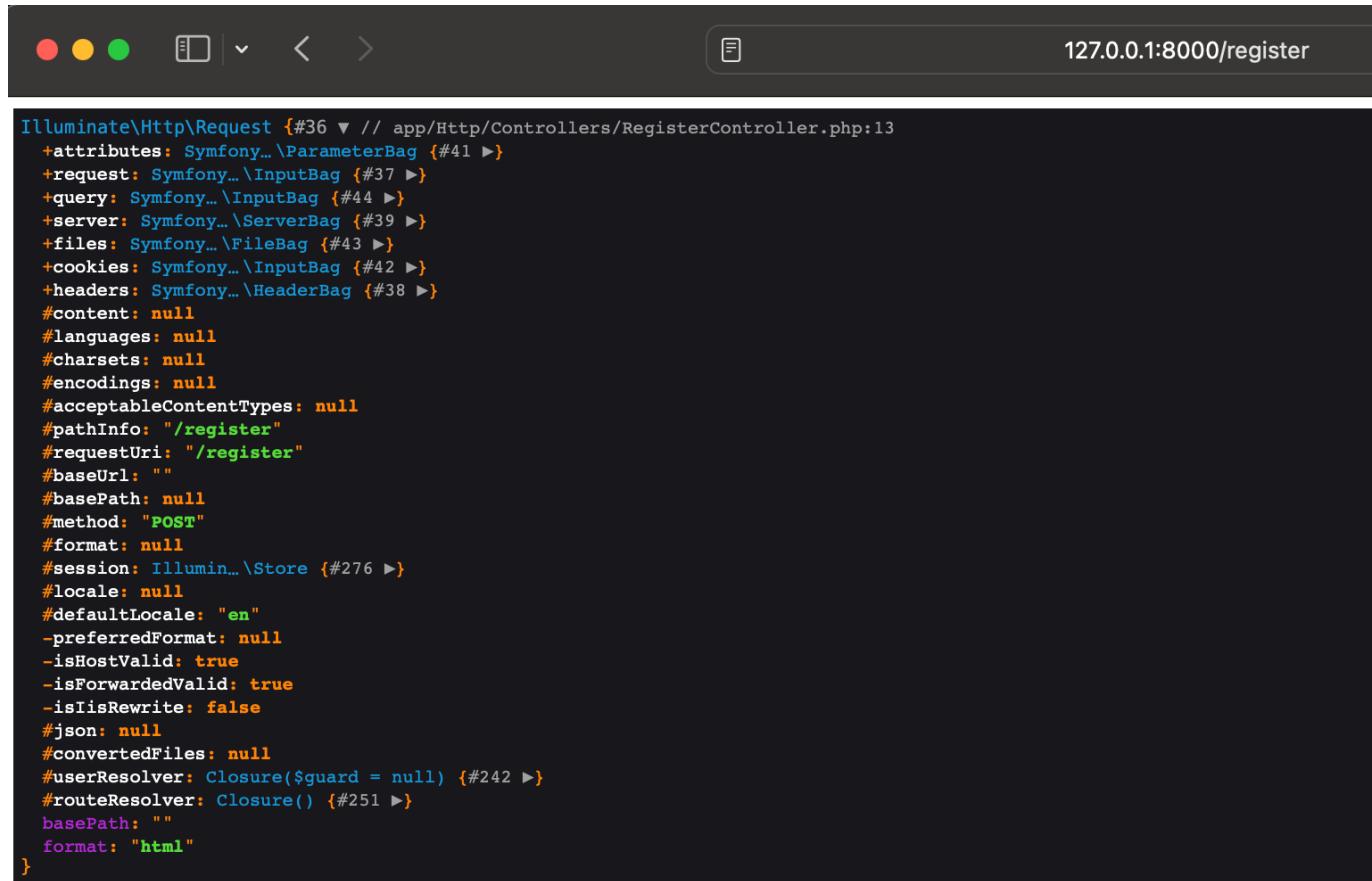
use Illuminate\Http\Request;

3 usages
class RegisterController extends Controller
{
    public function index() {
        return view('auth.register');
    }

    public function store(Request $request) {
        dd($request);
    }
}
```

¿Cómo leer los datos del formulario?

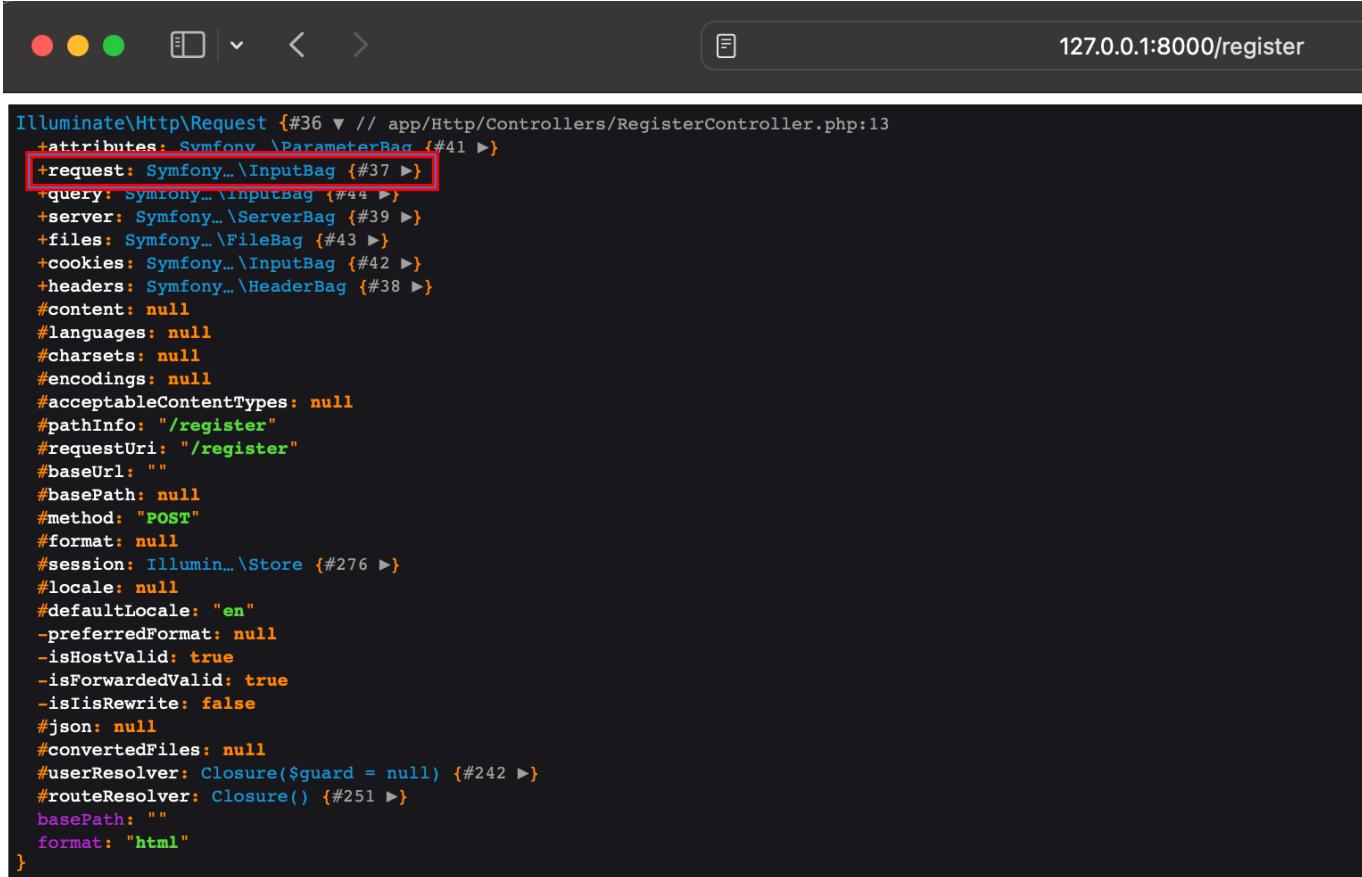
Al hacer click en "CREAR CUENTA" nos abre el método store y observamos todo lo que contiene la variable \$request



```
Illuminate\Http\Request {#36 ▼ // app/Http/Controllers/RegisterController.php:13
  +attributes: Symfony...\\ParameterBag {#41 ▶}
  +request: Symfony...\\InputBag {#37 ▶}
  +query: Symfony...\\InputBag {#44 ▶}
  +server: Symfony...\\ServerBag {#39 ▶}
  +files: Symfony...\\FileBag {#43 ▶}
  +cookies: Symfony...\\InputBag {#42 ▶}
  +headers: Symfony...\\HeaderBag {#38 ▶}
  #content: null
  #languages: null
  #charsets: null
  #encodings: null
  #acceptableContentTypes: null
  #pathInfo: "/register"
  #requestUri: "/register"
  #baseUrl: ""
  #basePath: null
  #method: "POST"
  #format: null
  #session: Illumin...\\Store {#276 ▶}
  #locale: null
  #defaultLocale: "en"
  -preferredFormat: null
  -isHostValid: true
  -isForwardedValid: true
  -isIisRewrite: false
  #json: null
  #convertedFiles: null
  #userResolver: Closure($guard = null) {#242 ▶}
  #routeResolver: Closure() {#251 ▶}
  basePath: ""
  format: "html"
}
```

¿Cómo leer los datos del formulario?

De toda la información que nos retorna, nos fijaremos un momento en request:



```
Illuminate\Http\Request {#36 ▼ // app/Http/Controllers/RegisterController.php:13
  +attributes: Symfony...\\ParameterBag {#41 ▶}
  +request: Symfony...\\InputBag {#37 ▶} (highlighted)
  +query: Symfony...\\InputBag {#44 ▶}
  +server: Symfony...\\ServerBag {#39 ▶}
  +files: Symfony...\\FileBag {#43 ▶}
  +cookies: Symfony...\\InputBag {#42 ▶}
  +headers: Symfony...\\HeaderBag {#38 ▶}
  #content: null
  #languages: null
  #charsets: null
  #encodings: null
  #acceptableContentTypes: null
  #pathInfo: "/register"
  #requestUri: "/register"
  #baseUrl: ""
  #basePath: null
  #method: "POST"
  #format: null
  #session: Illumin...\\Store {#276 ▶}
  #locale: null
  #defaultLocale: "en"
  -preferredFormat: null
  -isHostValid: true
  -isForwardedValid: true
  -isIisRewrite: false
  #json: null
  #convertedFiles: null
  #userResolver: Closure($guard = null) {#242 ▶}
  #routeResolver: Closure() {#251 ▶}
  basePath: ""
  format: "html"
}
```

¿Cómo leer los datos del formulario?

De toda la información que nos retorna, nos fijaremos un momento en request:

```
Illuminate\Http\Request {#36 ▼ // app/Http/Controllers/RegisterController.php:13
    +attributes: Symfony... \ParameterBag {#41 ▶}
    +request: Symfony... \InputBag {#37 ▼
        #parameters: array:6 [▼
            "_token" => "pQEmNwMpWxio5TYhcD0SMOzWCeXzhxVo4L4J2gZ5"
            "name" => "Diego"
            "username" => "diegoCalderon"
            "email" => "diegokld@info.com"
            "password" => "123"
            "password_confirmation" => "123"
        ]
    }
}
```

Como podemos observar, nos muestra la información que enviamos a través de nuestro formulario, así que para acceder a esta información usaremos el name que se crea en el input y envía la información al servidor así:

```
public function store(Request $request) {
    dd($request->get('key: 'name'));
}
```

Validación de datos

Para validar hacemos uso dentro del controlador en el método store o el que corresponda de la palabra reservada:
Validate

La forma de usarla es:

```
public function store(Request $request) {
    //dd($request->get('name'));
    $this->validate($request, [
        'name'=>'required'
    ]);
}
```

Como se observa en la imagen, validate recibe la variable \$request y un array en donde se organizará el nombre y la regla que le demos a ese nombre, para este caso vamos a hacer la validación del primer campo del formulario, es decir, el nombre.

Validación de datos

Para esto, luego de crear el campo validate en el controlador nos dirigimos a register.Blade.php y debajo del input donde está el nombre se abrirán y cerrarán las etiquetas:

```
<div class="mb-5">
    <label for="name" class="mb-2 block uppercase text-gray-500 font-bold" >Nombre</label>
    <input type="text" id="name" name="name" placeholder="Nombre" class="border p-3 w-full rounded-lg">
    @error('name')
        @enderror
    </div>
```

Y dentro hará un texto simple así:

```
@error('name')
    <p class="bg-red-500 text-white my-2 rounded-lg text-sm p-2 text-center">Error, debe ingresar el nombre</p>
@enderror
```

Teniendo este resultado en el navegador:

NOMBRE

Nombre

Error, debe ingresar el nombre

Validación de datos

Si bien la forma anterior nos enseña a crear textos de error y validación, no es la forma más optima ya que laravel tiene variables destinadas a este fin, por ejemplo:

```
@error('name')
    <p class="bg-red-500 text-white my-2 rounded-lg text-sm p-2 text-center">{{ $message }}</p>
@enderror
```

Y si le agrego una restricción a la validación obtengo lo siguiente:

```
public function store(Request $request) {
    //dd($request->get('name'));
    $this->validate($request, [
        'name'=>'required | min:5'
    ]);
}
```

NOMBRE

Nombre

The name field must be at least 5 characters.

Validación de datos

Evidentemente el retorno del mensaje esta en ingles, veamos como ponerlo en español:

Dentro del directorio resources, crearemos un directorio llamado lang y pegaremos mediante git o descargando directamente de:

<https://github.com/MarcoGomesr/laravel-validation-en-espanol.git>

Si usa git: git clone https://github.com/MarcoGomesr/laravel-validation-en-espanol.git resources/lang

Si no simplemente se descarga la carpeta y se pega dentro del directorio creado.

Seguidamente, se dirige al archivo config/app.php a la línea 85 y cambia de en a es

A screenshot of a code editor showing a portion of the config/app.php file. The line 85, which contains the code 'locale' => 'en', has a yellow lightbulb icon positioned above it, indicating a code suggestion or hint. The code editor interface shows line numbers 84, 85, 86, 87, and 88, with the cursor positioned at the start of line 88.

Validación de datos

Recarga la página, vuelve a poner un nombre de 4 dígitos y enviar:

NOMBRE

Nombre

El campo name debe tener al menos 5 caracteres.

Hemos cambiado el idioma de nuestra app.

Validación de datos

Cree el resto de las validaciones teniendo en cuenta que el userName es único ya que no puede haber dos cuentas iguales. También desarrolle y descubra la forma en que se deben hacer migraciones e insertar estos datos a una tabla en la base de datos.



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