Informe Técnico Proyecto Academia

11 de agosto de 2022

Procedimientos en Laravel versión 9.21.5

Presentado por:

Rafael José Arenas Restrepo

Presentado al supervisor:

Carlos Andrés Mora Agudelo

Empresa de Desarrollo CDITI Ltda.

Dosquebradas Risaralda

Agosto 2022

Actividad del 11 de ago. de 22

Módulo Docentes

Para crear el módulo de Docentes, seguimos los siguientes pasos:

1. En la terminal de Visual Studio Code, creamos el controlador de recursos TeacherController para generar el CRUD donde estaremos trabajando más adelante usando el comando **php artisan make:controller nombreControlador –resource**, como se puede ver a continuación:

```
PS E:\TGO ADSI 2341240\5to Trimestre\CARLOS MORA\CONSTRUIR -- LUNES-VIERNES\Laravel\Laravel 2341240\Ejercicio1> php artisan make:controller TeacherCont
roller --resource
```

Captura 1 Creando el controlador de recursos

2. Creamos el modelo Teacher para generar el fichero de migración donde vamos a crear nuestra tabla Teachers usando el comando **php artisan make:model NOMBREMODELO –m**, como podemos ver en la siguiente captura:

```
PS E:\TGO ADSI 2341240\5to Trimestre\CARLOS MORA\CONSTRUIR -- LUNES-VIERNES\Laravel\Laravel 2341240\Ejercicio1> php artisan make:model Teacher -m

  INFO  Model created successfully.

  INFO  Created migration [2022_08_11_080617_create_teachers_table].
```

Captura 2 Creando el modelo Teacher con su respectiva migración

3. Dentro del fichero de migración que se creó en el paso anterior, añadimos las siguientes líneas de código para crear nuestra tabla Teachers:

```
14      public function up()
15      {
16          Schema::create('teachers', function (Blueprint $table) {
17              $table->id();
18              $table->timestamps();
19              $table->string('name');
20              $table->string('last_name');
21              $table->string('college_degree');
22              $table->integer('age');
23              $table->timestamps('contract_date');
24              $table->string('imagen');
25              $table->string('identify_document');
26          });
27      }
28
```

Captura 3 Creando la tabla Teachers

4. Generamos la migración de la tabla Teachers hacia la BD, con el siguiente comando **php artisan migrate**, de la siguiente forma:

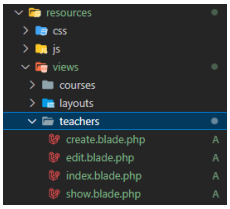
```
PS E:\TGO ADSI 2341240\5to Trimestre\CARLOS MORA\CONSTRUIR -- LUNES-VIERNES\Laravel\Laravel 2341240\Ejercicio1> php artisan m
igrate

  INFO  Running migrations.

  2022_08_11_080617_create_teachers_table ..................................................... 32ms DONE
```

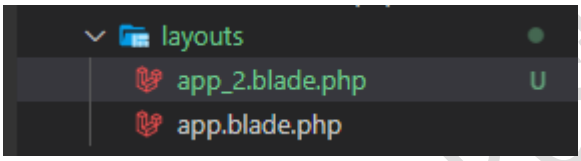
Captura 4 Migrando la tabla Teachers hacia la BD

5. Dentro del directorio **views**, ubicado dentro de la carpeta **resources**, creamos una nueva carpeta llamada **teachers**, y dentro de este creamos las vistas create, edit, show, index:



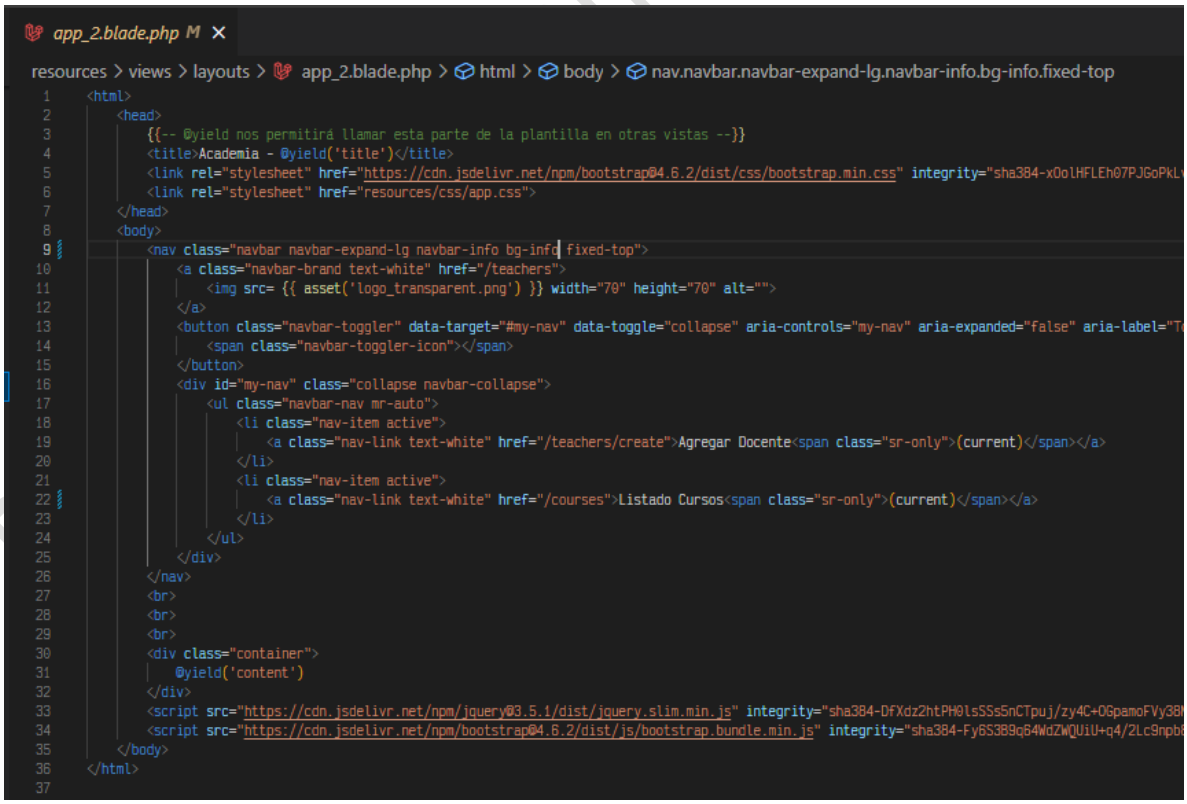
Captura 5 Carpeta y ficheros de vistas para Teachers

6. Dentro del directorio layouts, creamos el fichero app_2.blade.php para hacer la plantilla correspondiente a Teachers:



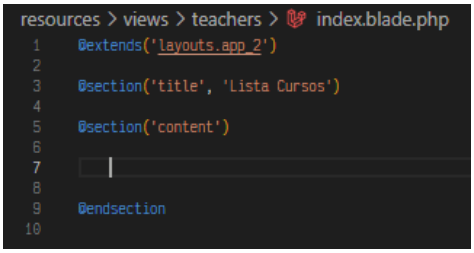
Captura 6 Creando el fichero app_2

7. En el fichero app_2 agregamos las siguientes líneas de código para crear la plantilla que será usada en el módulo docentes:



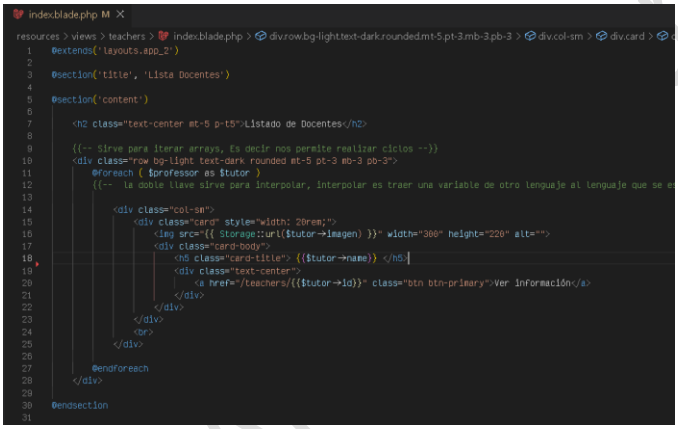
Captura 7 Plantilla app_2

8. Dentro del fichero index.blade.php invocamos la plantilla de app.blade.php ubicada en el directorio layouts.



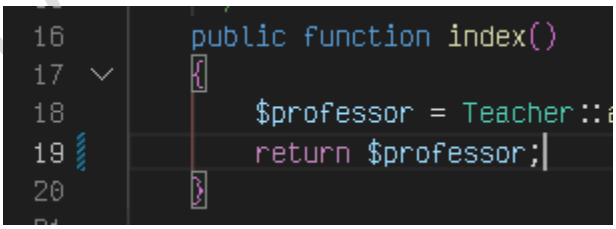
Captura 8 Llamando la plantilla app_2

9. Añadimos las siguientes líneas de código dentro del fichero index.blade.php:



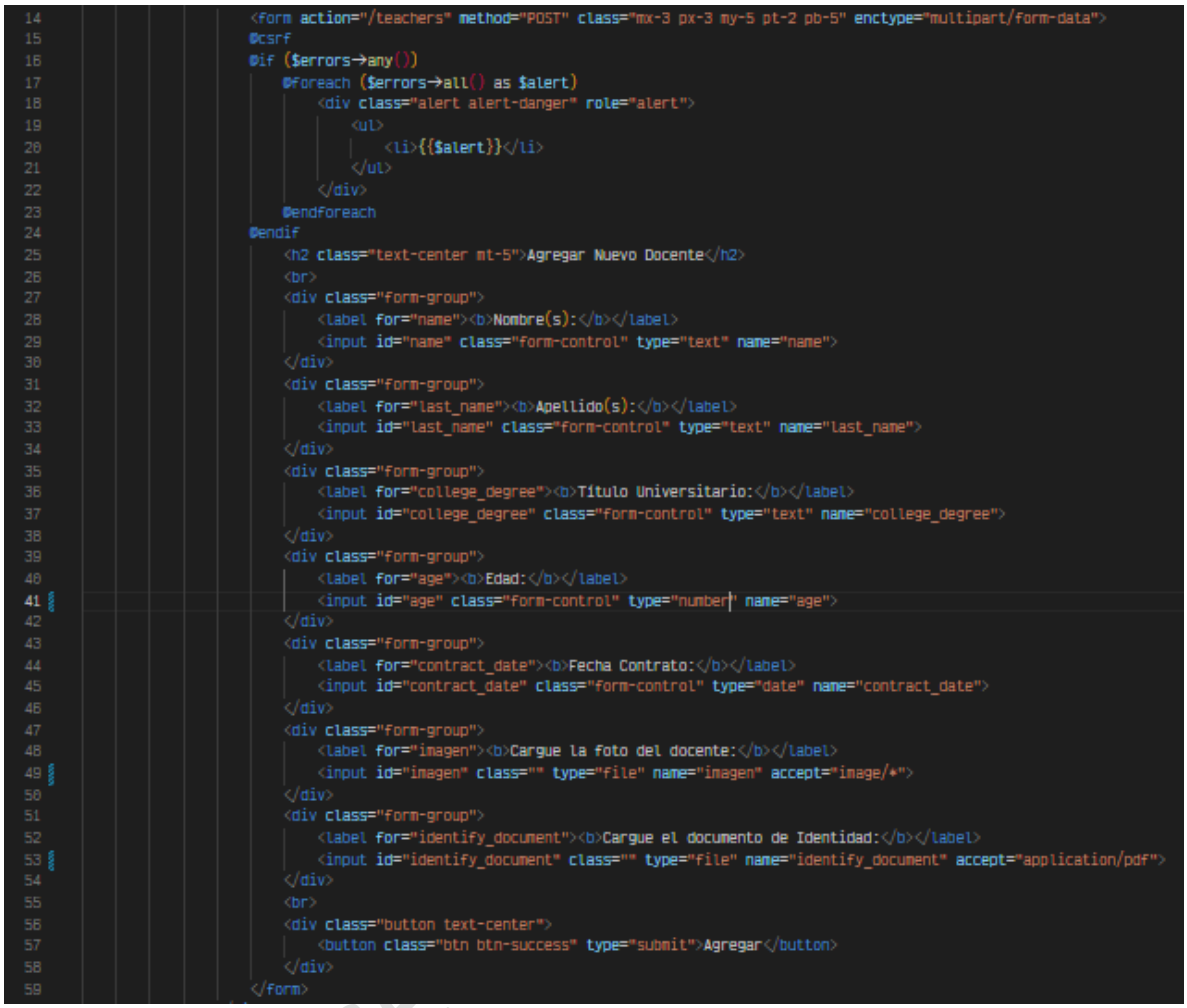
Captura 9 Vista index

10. Dentro del controlador de recursos TeacherController, ubicamos el método index y añadimos las siguientes líneas:



Captura 10 Prueba para saber que el request está enviando la información

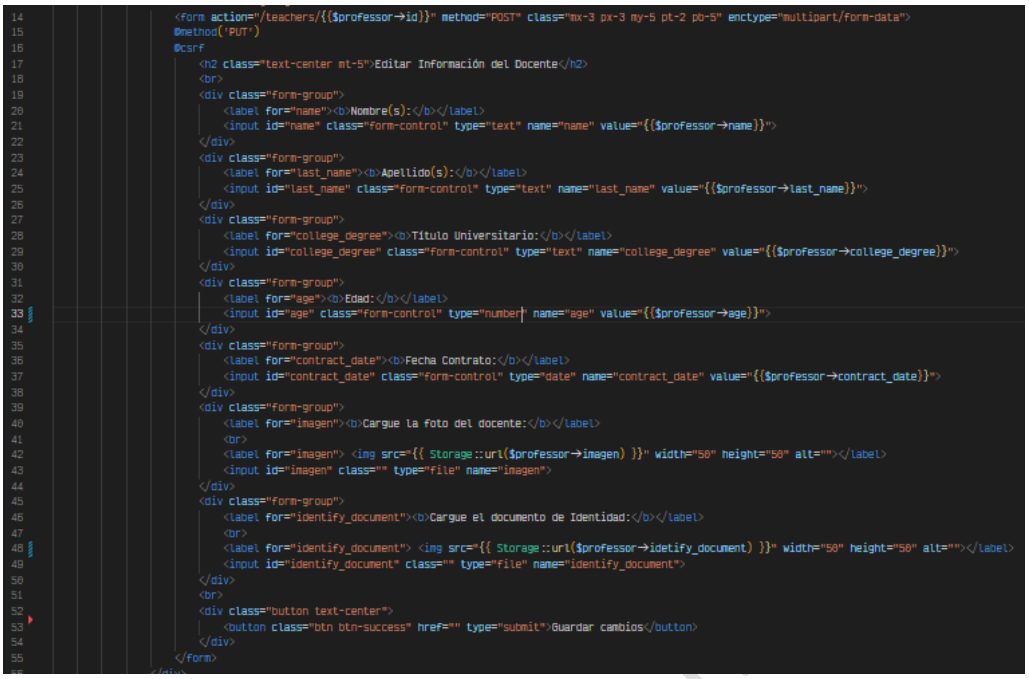
11. En el fichero create hacemos un formulario para ingresar los datos solicitados, quedando de la siguiente forma el código:



```
<form action="/teachers" method="POST" class="mx-3 px-3 my-5 pt-2 pb-5" enctype="multipart/form-data">
@csrf
@if ($errors->any())
    @foreach ($errors->all() as $alert)
        <div class="alert alert-danger" role="alert">
            <ul>
                <li>{{$alert}}</li>
            </ul>
        </div>
    @endforeach
@endif
    <h2 class="text-center mt-5">Agregar Nuevo Docente</h2>
    <br>
    <div class="form-group">
        <label for="name"><b>Nombre(s):</b></label>
        <input id="name" class="form-control" type="text" name="name">
    </div>
    <div class="form-group">
        <label for="last_name"><b>Apellido(s):</b></label>
        <input id="last_name" class="form-control" type="text" name="last_name">
    </div>
    <div class="form-group">
        <label for="college_degree"><b>Título Universitario:</b></label>
        <input id="college_degree" class="form-control" type="text" name="college_degree">
    </div>
    <div class="form-group">
        <label for="age"><b>Edad:</b></label>
        <input id="age" class="form-control" type="number" name="age">
    </div>
    <div class="form-group">
        <label for="contract_date"><b>Fecha Contrato:</b></label>
        <input id="contract_date" class="form-control" type="date" name="contract_date">
    </div>
    <div class="form-group">
        <label for="imagen"><b>Cargue la foto del docente:</b></label>
        <input id="imagen" class="" type="file" name="imagen" accept="image/*">
    </div>
    <div class="form-group">
        <label for="identify_document"><b>Cargue el documento de Identidad:</b></label>
        <input id="identify_document" class="" type="file" name="identify_document" accept="application/pdf">
    </div>
    <br>
    <div class="button text-center">
        <button class="btn btn-success" type="submit">Agregar</button>
    </div>
</form>
```

Captura 11 Vista create

12. En la vista edit, copiamos el mismo código del formulario anterior, además le hacemos unas modificaciones en los imputs para que nos muestre en pantalla la información del docente que queremos editar:



```
<form action="/teachers/{{$professor->id}}" method="POST" class="mx-3 px-3 my-5 pt-2 pb-5" enctype="multipart/form-data">
@method('PUT')
@csrf
    <h2 class="text-center mt-5">Editar Información del Docente</h2>
    <br>
    <div class="form-group">
        <label for="name"><b>Nombre(s):</b></label>
        <input id="name" class="form-control" type="text" name="name" value="{{$professor->name}}">
    </div>
    <div class="form-group">
        <label for="last_name"><b>Apellido(s):</b></label>
        <input id="last_name" class="form-control" type="text" name="last_name" value="{{$professor->last_name}}">
    </div>
    <div class="form-group">
        <label for="college_degree"><b>Titulo Universitario:</b></label>
        <input id="college_degree" class="form-control" type="text" name="college_degree" value="{{$professor->college_degree}}">
    </div>
    <div class="form-group">
        <label for="age"><b>Edad:</b></label>
        <input id="age" class="form-control" type="number" name="age" value="{{$professor->age}}">
    </div>
    <div class="form-group">
        <label for="contract_date"><b>Fecha Contrato:</b></label>
        <input id="contract_date" class="form-control" type="date" name="contract_date" value="{{$professor->contract_date}}">
    </div>
    <div class="form-group">
        <label for="imagen"><b>Cargue la foto del docente:</b></label>
        <br>
        <label for="imagen"> <img src="{{ Storage::url($professor->imagen) }}" width="50" height="50" alt=""></label>
        <input id="imagen" class="" type="File" name="imagen">
    </div>
    <div class="form-group">
        <label for="identify_document"><b>Cargue el documento de Identidad:</b></label>
        <br>
        <label for="identify_document"> <img src="{{ Storage::url($professor->idetify_document) }}" width="50" height="50" alt=""></label>
        <input id="identify_document" class="" type="File" name="identify_document">
    </div>
    <br>
    <div class="button text-center">
        <button class="btn btn-success" href="" type="submit">Guardar cambios</button>
    </div>
</form>
```

Captura 12 Vista edit

13. Ubicamos el método store en el controlador de recursos y digitamos el siguiente código para que se almacenen los datos introducidos en los inputs dentro de la BD:

```
public function store(storeTeacherRequest $request)
{
    if($request->hasFile( key: 'imagen')){
        $file = $request->file( key: 'imagen');
    }

    // return $request->all();
    $professor = new Teacher();
    $professor->name = $request->input( key: 'name');
    $professor->last_name = $request->input( key: 'last_name');
    $professor->college_degree = $request->input( key: 'college_degree');
    $professor->age = $request->input( key: 'age');
    $professor->contract_date = $request->input( key: 'contract_date');
    if($request->hasFile( key: 'imagen')){
        $professor->imagen = $request->file( key: 'imagen')->store( path: 'public/teachers');
    }
    if($request->hasFile( key: 'identify_document')){
        $professor->identify_document = $request->file( key: 'identify_document')->store( path: 'public/identify_document');
    }
    $professor->save();//Comando para registrar la info en la bd
    // return 'El docente se ha agregado exitosamente';
    return view( view: 'teachers.add_teacher');
}
```

Captura 13 Método store en el controlador de recursos

14. Ubicamos el método edit y update en el controlador de recursos y añadimos lo siguiente:

```php
public function edit($id)
{
    $professor = Teacher::find($id);
    // return 'El id de este curso es: ' . $id;
    // return 'La información que ud quiere actualizar, se vería en formato array...' . $professor;
    return view( view: 'teachers.edit', data: compact( var_name: 'professor'));
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $professor = Teacher::find($id);
    // return $professor;
    $professor->fill($request->except( keys: 'imagen'));
    if($request->hasFile( key: 'imagen')){
        $professor->imagen = $request->file( key: 'imagen')->store( path: 'public/courses');
    }
    if($request->hasFile( key: 'identify_document')){
        $professor->identify_document = $request->file( key: 'identify_document')->store( path: 'public/identify_document');
    }
    $professor->save();
    // return $request;
    // return 'La información del docente se ha actualizado exitosamente';
    return view( view: 'teachers.edit_teacher');
}
```

Captura 14 Método edit y update

15. Posteriormente en la vista show generamos el siguiente código para mostrar en pantalla la información requerida, haciendo la consulta en la BD en importándola a la vista:

show.blade.php M

resources > views > teachers > show.blade.php > div.bg-light.text-dark.rounded.mt-5.pt-5.mb-5.pb-5.text-center

```blade
@section('content')

    <div class="bg-light text-dark rounded mt-5 pt-5 mb-5 pb-5 text-center">
        <img src="{{ Storage::url($professor->imagen) }}" width="350" height="310" class="mb-5">
        <p class="card-text"> <b>Nombre(s):</b> {{$professor->name}} </p>
        <p class="card-text"> <b>Apellido(s):</b> {{$professor->last_name}} </p>
        <p class="card-text"> <b>Titulo Universitario:</b> {{$professor->college_degree}} </p>
        <p class="card-text"> <b>Edad:</b> {{$professor->age}} </p>
        <p class="card-text"> <b>Fecha Contrato:</b> {{$professor->contract_date}} </p>
        <p class="card-text"> <b>Documento de Identidad:</b> <button type="button" href="/storage/app/public/identify_document/" class="btn btn-info">Ver</button>
        {{-- {{$professor->identify_document}} --}}
        <div class="text-center p-3">
            <a href="/teachers/{{$professor->id}}/edit" class="btn btn-warning">Editar</a>
            <br>
            <br>
            {{-- Para este caso no se necesita escribir destroy en la ruta como si escribimos
            edit en la ruta para obtener el formulario de edición. Aquí creamos un formulario
            simplemente para poder incluir el botón para eliminar --}}
            <form class="form-group" action="/teachers/{{$professor->id}}" method="POST">
                @csrf
                @method('DELETE')
                <button type="submit" class="btn btn-danger">Eliminar</button>
            </form>
        </div>
    </div>

@endsection
```

Captura 15 Vista show

16. Luego ubicamos el método show en el controlador de recursos, para retornar la vista en el navegador:

```php
public function show($id)
{
    $professor = Teacher::find($id);
    return view( view: 'teachers.show', data: compact( var_name: 'professor'));
    // return 'El id de este curso es: ' . $id;
}
```

Captura 16 Método show

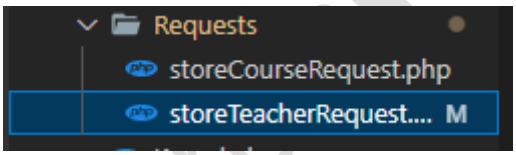
17. En el método destroy agregamos el siguiente código para que, al momento de eliminar la información de un docente, también se elimine la imagen y el documento PDF que han sido cargados previamente:

```php
    public function destroy($id)
    {
        $professor = Teacher::find($id);
        // return $professor;
        $urlImagenBD = $professor->imagen;
        // return $urlImagenBD;
        $imageName = str_replace( search: 'public/', replace: '\storage\\', subject: $urlImagenBD);
        $fullRoute = public_path() . $imageName;
        unlink( filename: $fullRoute);
        // $professor->delete();
        // return 'El curso se ha eliminado exitosamente';
        // return $fullRoute;

        $urlDocument = $professor->identify document;
        $documentName = str_replace( search: 'public/', replace: '\storage\\', subject: $urlDocument);
        $fullRoute = public_path() . $documentName;
        unlink( filename: $fullRoute);
        $professor->delete();
        return view( view: 'teachers.del_teacher');
    }
}
```

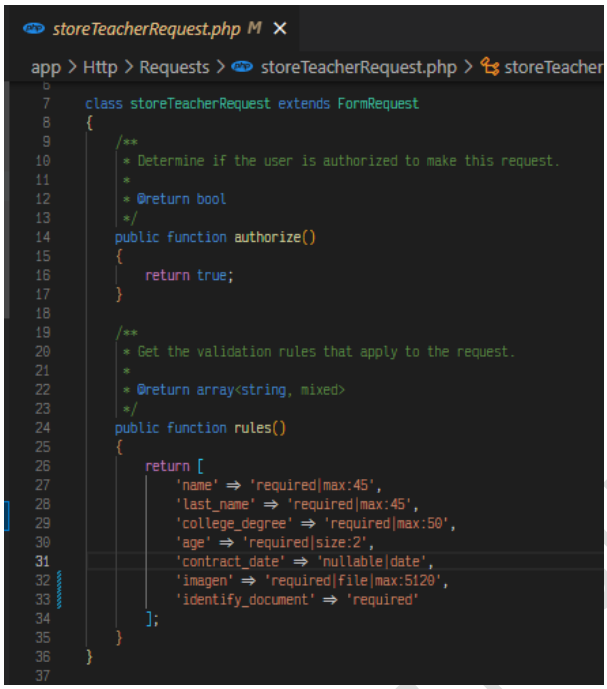
Captura 17 Método destroy

18. Con el comando php artisan make:request storeTeacherRequest generamos el fichero para las validaciones del formulario de agregar docente, quedando este fichero en la ubicación siguiente como podemos ver:



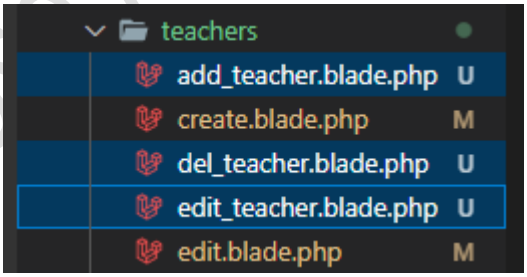
Captura 18 Fichero para las validaciones del formulario de agregar docente

19. Dentro del fichero creado en el paso anterior, hicimos las siguientes normas para validar:



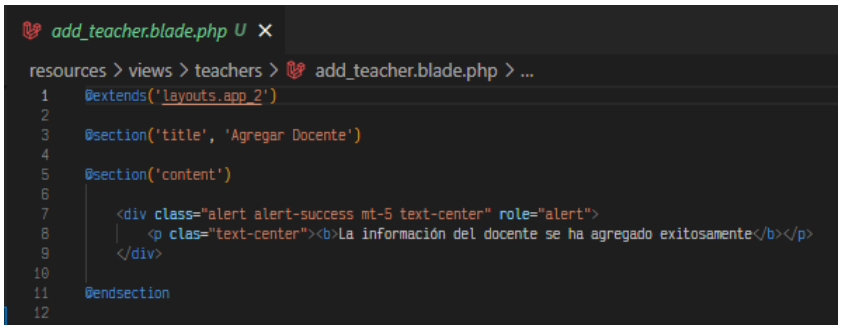
Captura 19 Normas para validar el formulario de agregar docente

20. Dentro de la carpeta views, ubicada en el directorio resources, agregamos las siguientes vistas:
    a. add_teacher.blade.php
    b. edit_teacher.blade.php
    c. del_teacher.blade.php



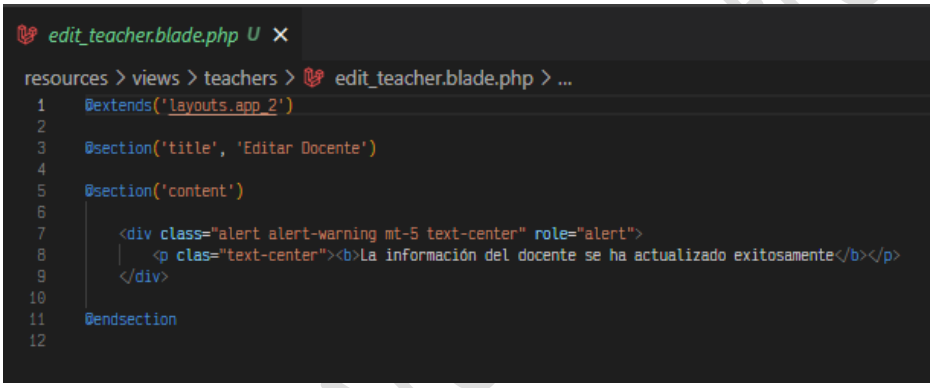
Captura 20 Vistas para las alertas de creado, editado y eliminado

21. Dentro de la vista add_teacher digitamos el siguiente código para que genere una alerta tipo Bootstrap al momento de haber añadido la información de un docente:



Captura 21 Vista add_teacher

22. Dentro de la vista edit_teacher digitamos el siguiente código para que genere una alerta tipo Bootstrap al momento de haber actualizado la información un docente:



Captura 22 Vista edit_teacher

23. Dentro de la vista del_teacher digitamos el siguiente código para que genere una alerta tipo Bootstrap al momento de haber eliminado la información un docente:

Captura 23 Vista del_teacher

24. En el fichero web.php ubicado en el directorio routes generamos la ruta para el módulo de docentes, quedando así:

```
61
62    Route::resource( name: 'teachers', controller: TeacherController::class);
63
```

Captura 24 Ruta para el módulo de docentes

25. En la vista show se añadió el siguiente código para implementar un visor de PDF:

```
<p class="card-text"> <b>Documento de Identidad:</b>
    <br>
    <div class="viewr-pdf text-center">
        <iframe src="{{ Storage::url($professor->identify_document) }}" width="400" height="400"></iframe>
    </div>
</p>
```

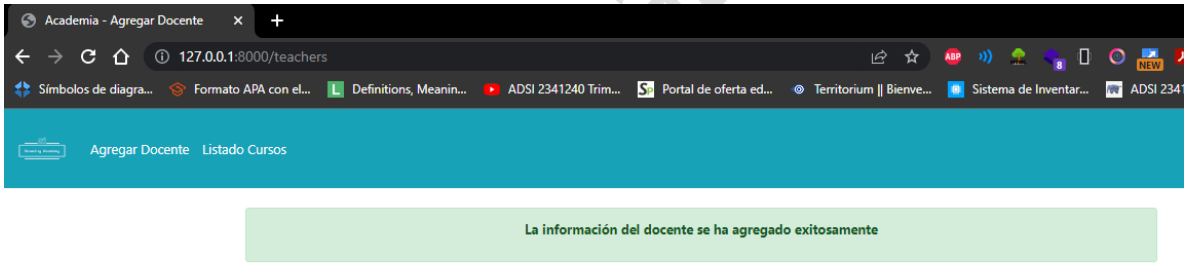
Captura 25 Implementando el visor de PDF

26. En la vista edit, también añadimos la siguiente línea para mostrar el visor de PDF:

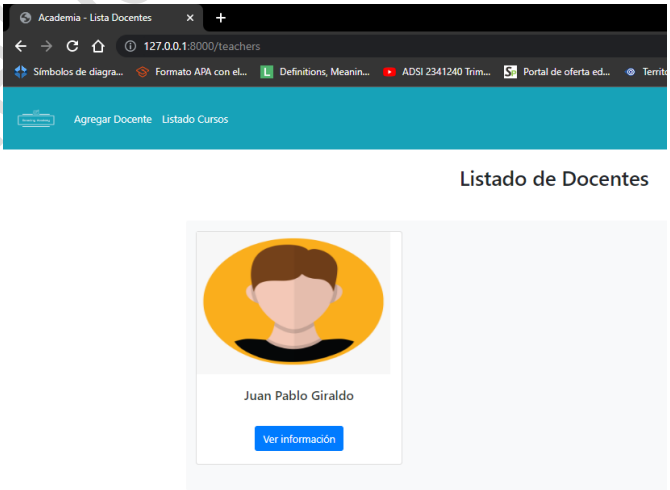
```
<div class="form-group">
    <label for="identify_document"><b>Cargue el documento de Identidad:</b></label>
    <br>
    <label for="identify_document"> <iframe src="{{ Storage::url($professor->identify_document) }}" width="100" height="100"></iframe> </label>
    <input id="identify_document" class="" type="file" name="identify_document">
</div>
```

Captura 26 Implementando el visor de PDF en la vista editar

27. Realizamos unas pruebas creando docentes y este fue el resultado:

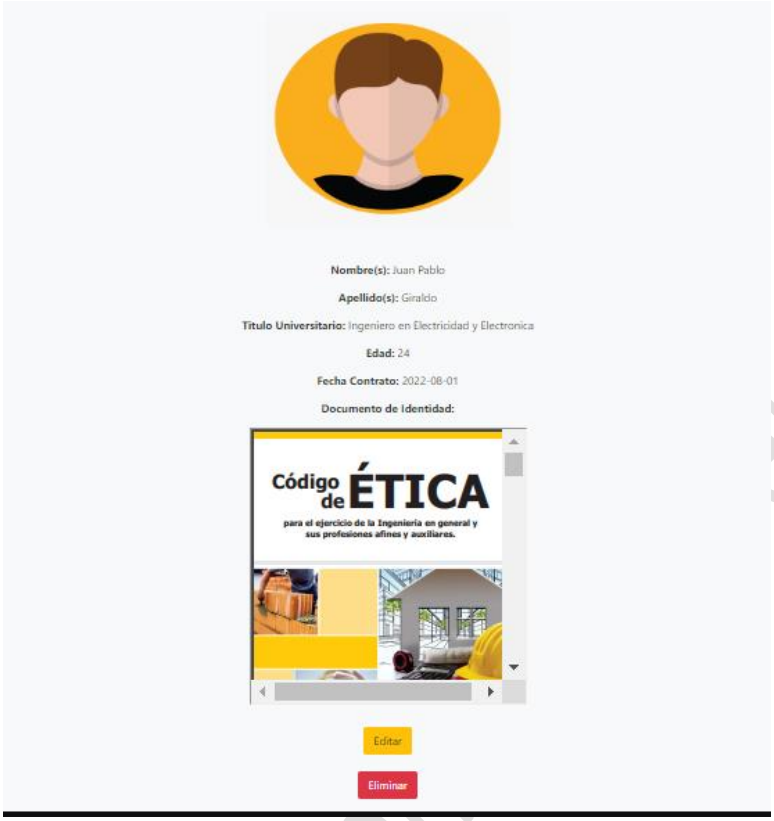


Captura 27 Alerta después de agregar un docente



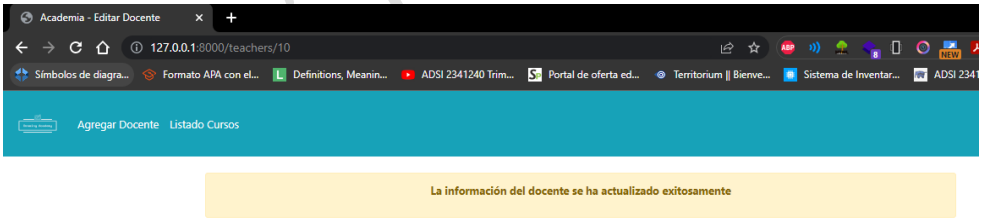
Captura 28 Resultado en el index después de agregar un docente

28. Dimos clic en ver información y este fue el resultado en el navegador:

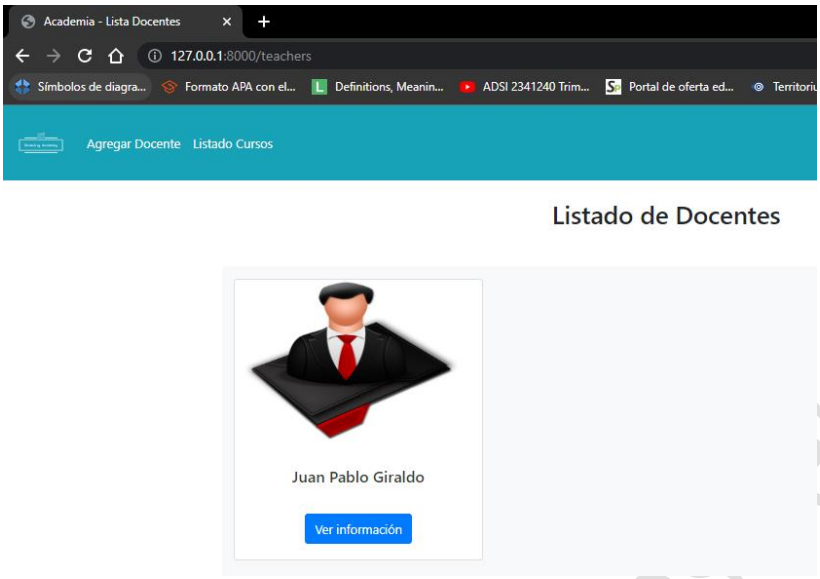


Captura 29 Vista show en el navegador

29. La siguiente es la alerta que nos muestra cuando editamos la información de un docente:

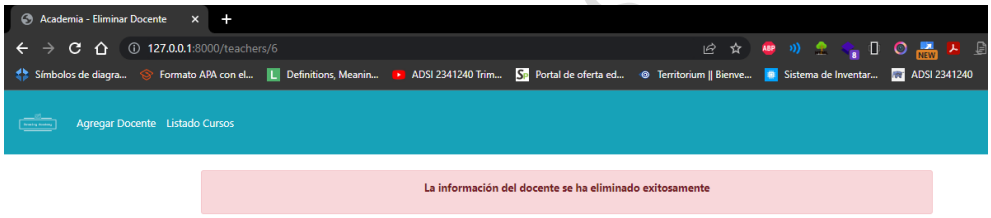


Captura 30 Alerta después de editar la información de un docente



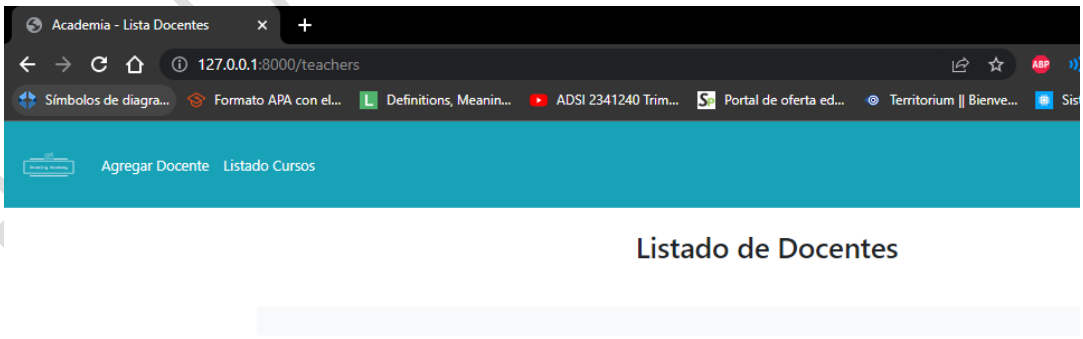
Captura 31 Resultado en el navegador después de cambiar la foto de un docente

30. Después de eliminar un docente está es la alerta que nos muestra el sistema:



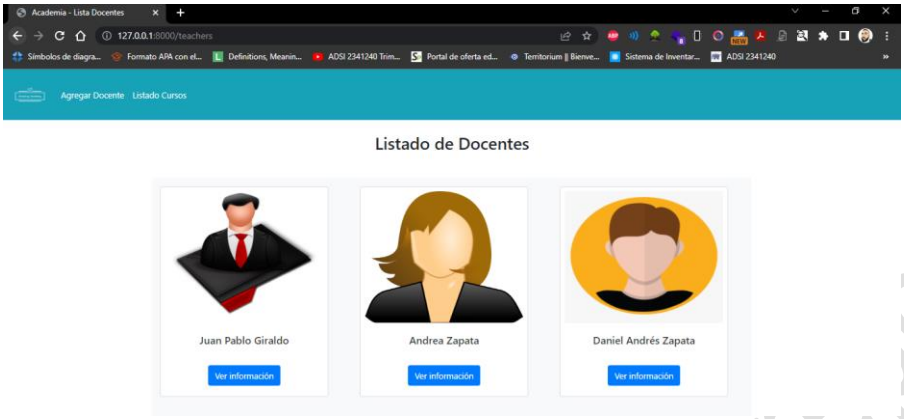
Captura 32 Alerta después de eliminar el docente

31. Esto fue lo que el navegador nos mostró después de eliminar el docente que habíamos agregado:



Captura 33 Resultado después de eliminar el docente que estaba almacenado en la BD

32. Resultado después de añadir varios docentes:



Captura 34 Resultado en el navegador después de agregar varios docentes