

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 TeacherController --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
```

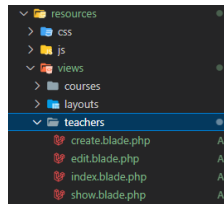
Captura 3 Creando la tabla Teachers

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 migrate
INFO: Running migrations.
2022_08_11_080617_create_teachers_table ..... 32ms DONE
```

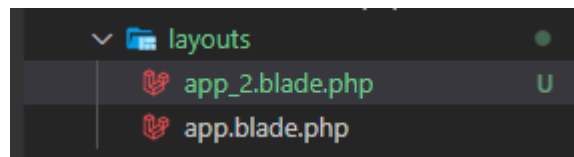
Captura 4 Migrando la tabla Teachers hacia la BD

5. Dentro del directorio **views**, ubicado dentro de la carpeta **resources**, creamos una nueva carpeta llamada **teachers**, y dentro de esta 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:

```
resources > views > layouts > app_2.blade.php > html > body > nav.navbar.navbar-expand-lg.navbar-info.bg-info.fixed-top
1 <html>
2 <head>
3     {{-- @yield nos permitirá llamar esta parte de la plantilla en otras vistas --}}
4     <title>Academia - @yield('title')</title>
5     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css" integrity="sha384-x0oLHfLEh07PJGoPKL
6     <link rel="stylesheet" href="resources/css/app.css">
7 </head>
8 <body>
9     <nav class="navbar navbar-expand-lg navbar-info bg-info fixed-top">
10         <a class="navbar-brand text-white" href="/teachers">
11             
12         </a>
13         <button class="navbar-toggler" data-target="#my-nav" data-toggle="collapse" aria-controls="my-nav" aria-expanded="false" aria-label="T
14             <span class="navbar-toggler-icon"></span>
15         </button>
16         <div id="my-nav" class="collapse navbar-collapse">
17             <ul class="navbar-nav mr-auto">
18                 <li class="nav-item active">
19                     <a class="nav-link text-white" href="/teachers/create">Agregar Docente<span class="sr-only">(current)</span></a>
20                 </li>
21                 <li class="nav-item active">
22                     <a class="nav-link text-white" href="/courses">Listado Cursos<span class="sr-only">(current)</span></a>
23                 </li>
24             </ul>
25         </div>
26     </nav>
27     <br>
28     <br>
29     <br>
30     <div class="container">
31         @yield('content')
32     </div>
33     <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38
34     <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-Fy6S3B9q64WdZwQUiU+q4/2Lc9npb
35 </body>
36 </html>
37
```

Captura 7 Plantilla app_2

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

```
resources > views > teachers > index.blade.php
1 @extends('layouts.app_2')
2
3 @section('title', 'Lista Cursos')
4
5 @section('content')
6
7
8
9 @endsection
10
```

Captura 8 Llamando la plantilla app_2

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

```
index.blade.php M X
resources > views > teachers > index.blade.php > div.row.bg-light.text-dark.rounded mt-5 pt-3 mb-3 pb-3 > div.col-sm-6 > div.card >
1 @extends('layouts.app_2')
2
3 @section('title', 'Lista Docentes')
4
5 @section('content')
6
7 <h2 class="text-center mt-5 pt-5">Listado de Docentes</h2>
8
9 <!-- Sirve para iterar arrays, es decir nos permite realizar ciclos -->
10 <div class="row bg-light text-dark rounded mt-5 pt-3 mb-3 pb-3">
11     @foreach ($professor as $tutor)
12         <!-- la doble llave sirve para interpolar, interpolar es traer una variable de otro lenguaje al lenguaje que se
13
14         <div class="col-sm-6">
15             <div class="card" style="width: 20rem;">
16                 
17                 <div class="card-body">
18                     <h3 class="card-title">{{ $tutor->name }}</h3>
19                     <div class="text-center">
20                         <a href="/teachers/{{ $tutor->id }}" class="btn btn-primary">Ver Información</a>
21                     </div>
22                 </div>
23             </div>
24         </div>
25     @endforeach
26 </div>
27
28 @endsection
29
30
31
```

Captura 9 Vista index

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

```
16 public function index()
17 {
18     $professor = Teacher::all();
19     return $professor;
20 }
```

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

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

```

14 <form action="/teachers" method="POST" class="mx-3 px-3 my-5 pt-2 pb-5" enctype="multipart/form-data">
15 @csrf
16 @if ($errors->any())
17     @foreach ($errors->all() as $alert)
18         <div class="alert alert-danger" role="alert">
19             <ul>
20                 <li>{{ $alert }}</li>
21             </ul>
22         </div>
23     @endforeach
24 @endif
25 <h2 class="text-center mt-5">Agregar Nuevo Docente</h2>
26 <br>
27 <div class="form-group">
28     <label for="name"><b>Nombre(s):</b></label>
29     <input id="name" class="form-control" type="text" name="name">
30 </div>
31 <div class="form-group">
32     <label for="last_name"><b>Apellido(s):</b></label>
33     <input id="last_name" class="form-control" type="text" name="last_name">
34 </div>
35 <div class="form-group">
36     <label for="college_degree"><b>Título Universitario:</b></label>
37     <input id="college_degree" class="form-control" type="text" name="college_degree">
38 </div>
39 <div class="form-group">
40     <label for="age"><b>Edad:</b></label>
41     <input id="age" class="form-control" type="number" name="age">
42 </div>
43 <div class="form-group">
44     <label for="contract_date"><b>Fecha Contrato:</b></label>
45     <input id="contract_date" class="form-control" type="date" name="contract_date">
46 </div>
47 <div class="form-group">
48     <label for="imagen"><b>Cargue la foto del docente:</b></label>
49     <input id="imagen" class="" type="file" name="imagen" accept="image/*">
50 </div>
51 <div class="form-group">
52     <label for="identify_document"><b>Cargue el documento de Identidad:</b></label>
53     <input id="identify_document" class="" type="file" name="identify_document" accept="application/pdf">
54 </div>
55 <br>
56 <div class="button text-center">
57     <button class="btn btn-success" type="submit">Agregar</button>
58 </div>
59 </form>

```

Captura 11 Vista create

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

```

14 <form action="/teachers/{{ $professor->id }}" method="POST" class="mx-3 px-3 my-5 pt-2 pb-5" enctype="multipart/form-data">
15 <method("PUT")>
16 <csrf>
17 <h2 class="text-center mt-5">Editar Información del Docente</h2>
18 <br>
19 <div class="form-group">
20 <label for="name"><b>Nombre(s):</b></label>
21 <input id="name" class="form-control" type="text" name="name" value="{{ $professor->name }}">
22 </div>
23 <div class="form-group">
24 <label for="last_name"><b>Apellido(s):</b></label>
25 <input id="last_name" class="form-control" type="text" name="last_name" value="{{ $professor->last_name }}">
26 </div>
27 <div class="form-group">
28 <label for="college_degree"><b>Título Universitario:</b></label>
29 <input id="college_degree" class="form-control" type="text" name="college_degree" value="{{ $professor->college_degree }}">
30 </div>
31 <div class="form-group">
32 <label for="age"><b>Edad:</b></label>
33 <input id="age" class="form-control" type="number" name="age" value="{{ $professor->age }}">
34 </div>
35 <div class="form-group">
36 <label for="contract_date"><b>Fecha Contrato:</b></label>
37 <input id="contract_date" class="form-control" type="date" name="contract_date" value="{{ $professor->contract_date }}">
38 </div>
39 <div class="form-group">
40 <label for="imagen"><b>Cargue la foto del docente:</b></label>
41 <br>
42 <label for="imagen"></label>
43 <input id="imagen" class="" type="file" name="imagen">
44 </div>
45 <div class="form-group">
46 <label for="identify_document"><b>Cargue el documento de Identidad:</b></label>
47 <br>
48 <label for="identify_document"></label>
49 <input id="identify_document" class="" type="file" name="identify_document">
50 </div>
51 <br>
52 <div class="button text-center">
53 <button class="btn btn-success" href="" type="submit">Guardar cambios</button>
54 </div>
55 </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:

```

40 public function store(storeTeacherRequest $request)
41 {
42     if($request->hasFile( key: 'imagen')){
43         $file = $request->file( key: 'imagen');
44     }
45
46     // return $request->all();
47     $professor = new Teacher();
48     $professor->name = $request->input( key: 'name');
49     $professor->last_name = $request->input( key: 'last_name');
50     $professor->college_degree = $request->input( key: 'college_degree');
51     $professor->age = $request->input( key: 'age');
52     $professor->contract_date = $request->input( key: 'contract_date');
53     if($request->hasFile( key: 'imagen')){
54         $professor->imagen = $request->file( key: 'imagen')->store( path: 'public/teachers');
55     }
56     if($request->hasFile( key: 'identify_document')){
57         $professor->identify_document = $request->file( key: 'identify_document')->store( path: 'public/identify_document');
58     }
59     $professor->save(); //Comando para registrar la info en la bd
60     // return 'El docente se ha agregado exitosamente';
61     return view( view: 'teachers.add_teacher');
62 }
63

```

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:

```
64 public function edit($id)
65 {
66     $professor = Teacher::find($id);
67     // return 'El id de este curso es: ' . $id;
68     // return 'La información que ud quiere actualizar, se vería en formato array...' . $professor;
69     return view('view: teachers.edit', data: compact( var_name: 'professor'));
70 }
71
72 /**
73  * Update the specified resource in storage.
74  *
75  * @param \Illuminate\Http\Request $request
76  * @param int $id
77  * @return \Illuminate\Http\Response
78  */
79 public function update(Request $request, $id)
80 {
81     $professor = Teacher::find($id);
82     // return $professor;
83     $professor->fill($request->except( key: 'imagen'));
84     if($request->hasFile( key: 'imagen')){
85         $professor->imagen = $request->file( key: 'imagen')->store( path: 'public/courses');
86     }
87     if($request->hasFile( key: 'identify_document')){
88         $professor->identify_document = $request->file( key: 'identify_document')->store( path: 'public/identify_document');
89     }
90     $professor->save();
91     // return $request;
92     // return 'La información del docente se ha actualizado exitosamente';
93     return view( view: 'teachers.edit_teacher');
94 }
95
```

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 X
resources > views > teachers > show.blade.php > div.bg-light.text-dark.rounded.mt-5.pt-5.mb-5.pb-5.text-center
5 @section('content')
6
7 <div class="bg-light text-dark rounded mt-5 pt-5 mb-5 pb-5 text-center">
8 
9 <p class="card-text"> <b>Nombre(s):</b> {{ $professor->name }} </p>
10 <p class="card-text"> <b>Apellido(s):</b> {{ $professor->last_name }} </p>
11 <p class="card-text"> <b>Titulo Universitario:</b> {{ $professor->college_degree }} </p>
12 <p class="card-text"> <b>Edad:</b> {{ $professor->age }} </p>
13 <p class="card-text"> <b>Fecha Contrato:</b> {{ $professor->contract_date }} </p>
14 <p class="card-text"> <b>Documento de Identidad:</b> <button type="button" href="/storage/app/public/identify_document/" class="btn btn-info">Ver</button>
15 {{ $professor->identify_document }} </p>
16 <div class="text-center p-3">
17 <a href="/teachers/{{ $professor->id }}/edit" class="btn btn-warning">Editar</a>
18 <br>
19 <br>
20 {{!-- Para este caso no se necesita escribir destroy en la ruta como si escribimos
21 edit en la ruta para obtener el formulario de edición. Aquí creamos un formulario
22 simplemente para poder incluir el botón para eliminar --}}
23 <form class="form-group" action="/teachers/{{ $professor->id }}" method="POST">
24 @csrf
25 @method('DELETE')
26 <button type="submit" class="btn btn-danger">Eliminar</button>
27 </form>
28 </div>
29 </div>
30 @endsection
31
```

Captura 15 Vista show

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

```
71 public function show($id)
72 {
73     $professor = Teacher::find($id);
74     return view( view: 'teachers.show', data: compact( var_name: 'professor'));
75     // return 'El id de este curso es: ' . $id;
76 }
77
```

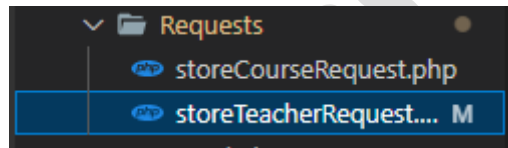
Captura 16 Método show

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:

```
119 public function destroy($id)
120 {
121     $professor = Teacher::find($id);
122     // return $professor;
123     $urlImagenBD = $professor->imagen;
124     // return $urlImagenBD;
125     $imageName = str_replace( search: 'public/', replace: '\storage\', subject: $urlImagenBD);
126     $fullRoute = public_path() . $imageName;
127     unlink( filename: $fullRoute);
128     // $professor->delete();
129     // return 'El curso se ha eliminado exitosamente';
130     // return $fullRoute;
131
132     $urlDocument = $professor->identify document;
133     $documentName = str_replace( search: 'public/', replace: '\storage\', subject: $urlDocument);
134     $fullRoute = public_path() . $documentName;
135     unlink( filename: $fullRoute);
136     $professor->delete();
137     return view( view: 'teachers.del_teacher');
138 }
139 }
```

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:


```

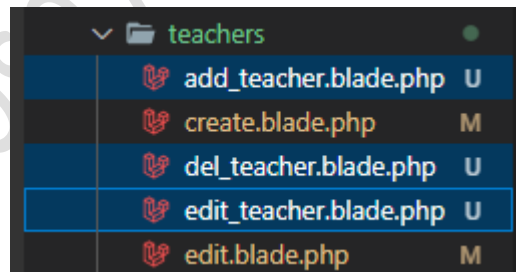
storeTeacherRequest.php M X
app > Http > Requests > storeTeacherRequest.php > storeTeacher
7 class storeTeacherRequest extends FormRequest
8 {
9     /**
10      * Determine if the user is authorized to make this request.
11      *
12      * @return bool
13      */
14     public function authorize()
15     {
16         return true;
17     }
18
19     /**
20      * Get the validation rules that apply to the request.
21      *
22      * @return array<string, mixed>
23      */
24     public function rules()
25     {
26         return [
27             'name' => 'required|max:45',
28             'last_name' => 'required|max:45',
29             'college_degree' => 'required|max:50',
30             'age' => 'required|size:2',
31             'contract_date' => 'nullable|date',
32             'imagen' => 'required|file|max:5120',
33             'identify_document' => 'required'
34         ];
35     }
36 }
37

```

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:

- add_teacher.blade.php
- edit_teacher.blade.php
- 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:

```

add_teacher.blade.php U X
resources > views > teachers > add_teacher.blade.php > ...
1 @extends('layouts.app_2')
2
3 @section('title', 'Agregar Docente')
4
5 @section('content')
6
7     <div class="alert alert-success mt-5 text-center" role="alert">
8         <p class="text-center"><b>La información del docente se ha agregado exitosamente</b></p>
9     </div>
10
11 @endsection
12

```

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:

```

edit_teacher.blade.php U X
resources > views > teachers > edit_teacher.blade.php > ...
1 @extends('layouts.app_2')
2
3 @section('title', 'Editar Docente')
4
5 @section('content')
6
7     <div class="alert alert-warning mt-5 text-center" role="alert">
8         <p class="text-center"><b>La información del docente se ha actualizado exitosamente</b></p>
9     </div>
10
11 @endsection
12

```

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:

```

del_teacher.blade.php U X
resources > views > teachers > del_teacher.blade.php > ...
1 @extends('layouts.app_2')
2
3 @section('title', 'Eliminar Docente')
4
5 @section('content')
6
7     <div class="alert alert-danger mt-5 text-center" role="alert">
8         <p class="text-center"><b>La información del docente se ha eliminado exitosamente</b></p>
9     </div>
10
11 @endsection
12

```

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:

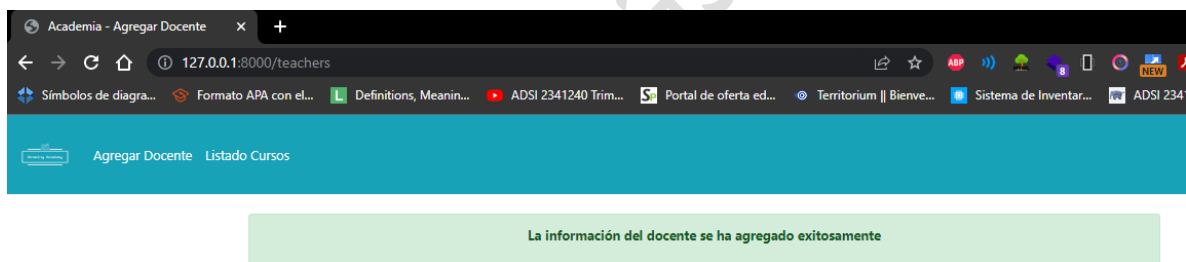
```

<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>

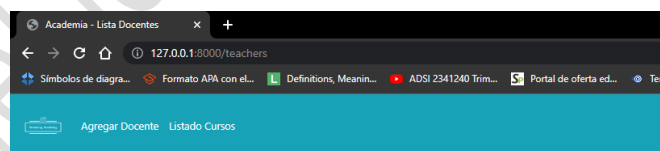
```

Captura 26 Implementando el visor de PDF en la vista editar

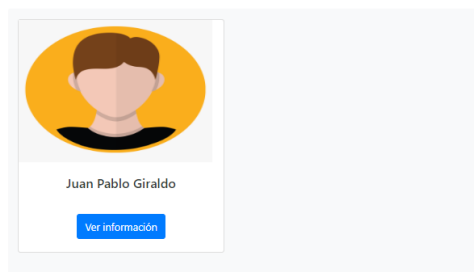
27. Realizamos unas pruebas creando docentes y este fue el resultado:



Captura 27 Alerta después de agregar un docente

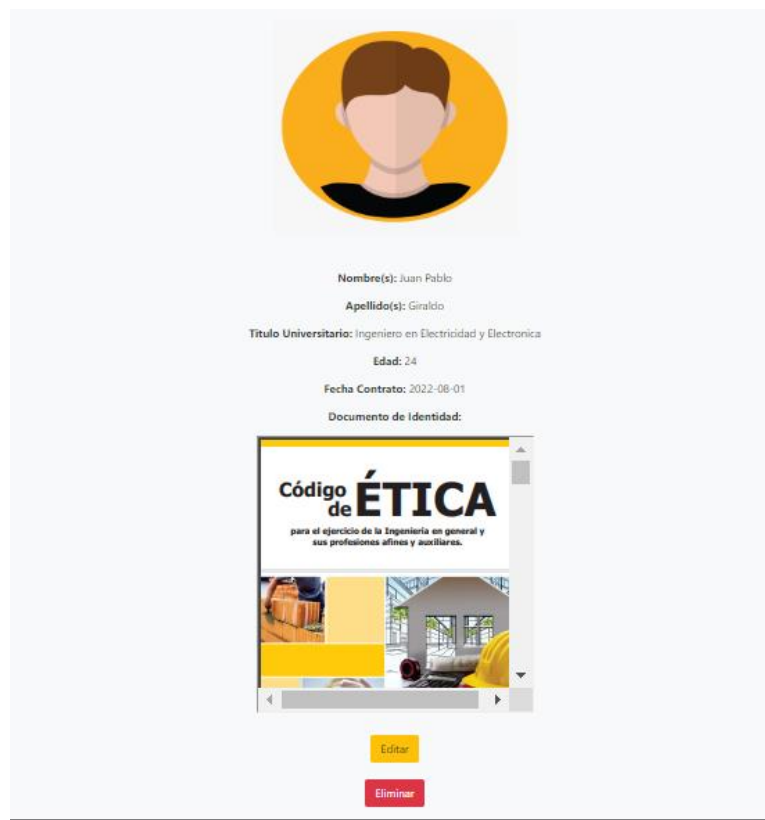


Listado de Docentes



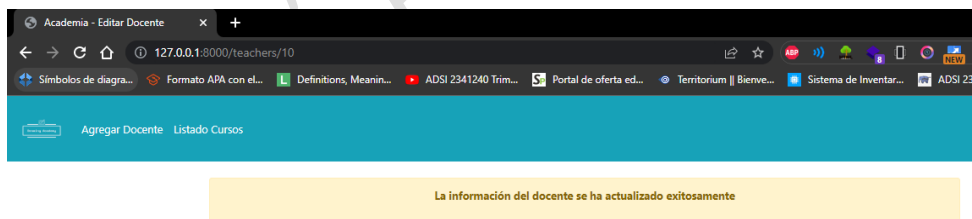
Captura 28 Resultado en el index después de agregar un docente

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

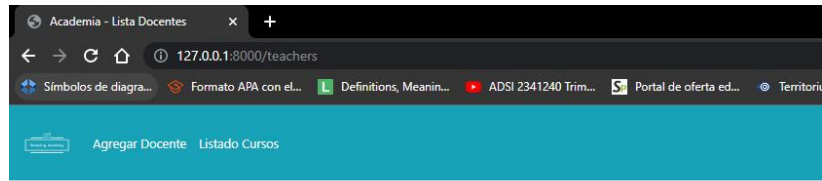


Captura 29 Vista show en el navegador

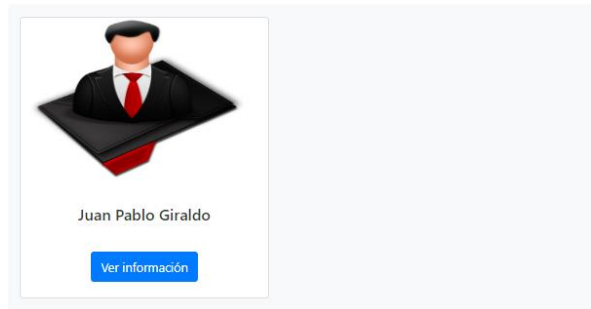
29. La siguiente es la alerta que nos muestra cuando editamos la información de un docente:



Captura 30 Alerta después de editar la información de un docente

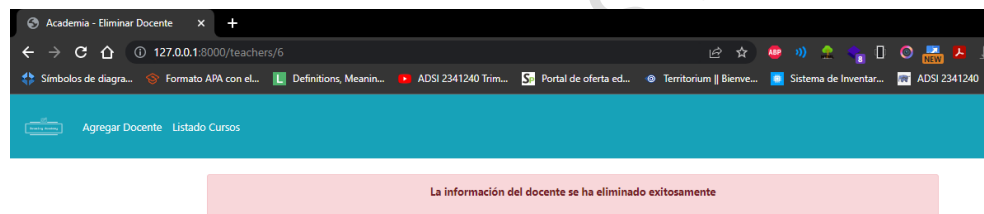


Listado de Docentes



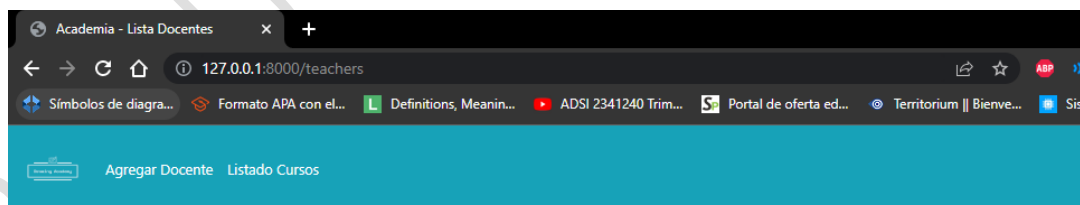
Captura 31 Resultado en el navegador después de cambiar la foto de un docente

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



Captura 32 Alerta después de eliminar el docente

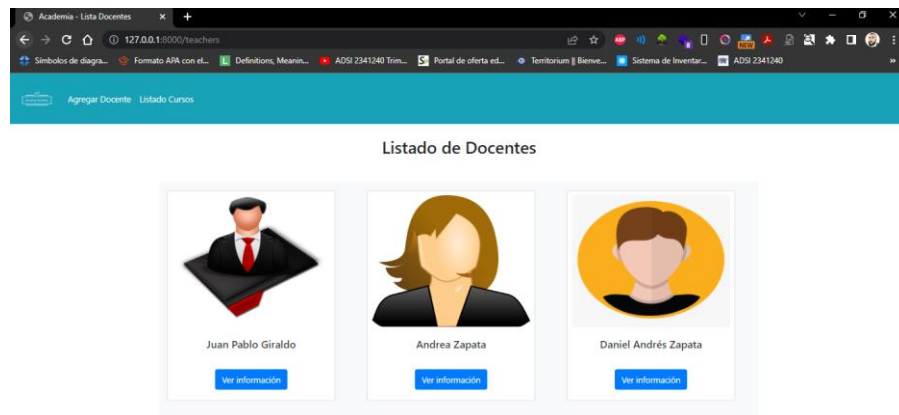
31. Esto fue lo que el navegador nos mostró después de eliminar el docente que habíamos agregado:



Listado de Docentes

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