

# **MANUAL DEL PROGRAMADOR**



## **UNISANGIL**

**Fundación Universitaria de San Gil Unisangil –  
Sede Yopal**

## Tabla de contenido

1	Introducción.....	6
2	Modulo Configuración .....	10
2.1	Estado programa .....	10
2.1.1	Modelo .....	10
2.1.2	Controlador.....	10
2.1.3	Vistas.....	13
2.2	Departamento.....	15
2.2.1	Modelo .....	15
2.2.2	Controlador.....	15
2.2.3	Vistas.....	17
2.3	Municipio .....	20
2.3.1	Modelo .....	20
2.3.2	Controlador.....	20
2.3.3	Vistas.....	23
2.4	Facultad .....	25
2.4.1	Modelo .....	25
2.4.2	Controlador.....	25
2.4.3	Vistas.....	28
2.5	Nivel formación.....	30
2.5.1	Modelo .....	30
2.5.2	Controlador.....	30
2.5.3	Vistas.....	33
2.6	Metodología.....	36
2.6.1	Modelo .....	36
2.6.2	Controlador.....	36
2.6.3	Vistas.....	39
3	Modulo Sistema.....	41

3.1	Programas.....	42
3.1.1	Modelo .....	42
3.1.2	Controlador.....	43
3.1.3	Vistas.....	48
3.1.4	Export.....	51
3.2	Estudiantes .....	53
3.2.1	Modelo .....	53
3.2.2	Controlador.....	54
3.2.3	Vistas.....	59
3.2.4	Export.....	62
3.3	Trabajo de grado.....	62
3.3.1	Modelo .....	62
3.3.2	Controlador.....	63
3.3.3	Vistas.....	67
3.3.4	Export.....	69
3.4	Docentes .....	70
3.4.1	Modelo .....	70
3.4.2	Controlador.....	71
3.4.3	Vistas.....	75
3.4.4	Export.....	76
3.5	Pruebas Saber Pro .....	78
3.5.1	Modelo .....	78
3.5.2	Controlador.....	78
3.5.3	Vistas.....	82
3.5.4	Export.....	84
3.6	TIC'S .....	86
3.6.1	Modelo .....	86
3.6.2	Controlador.....	86

3.6.3	Vistas.....	90
3.6.4	Export.....	92
3.7	Extensión e Internacionalización.....	94
3.7.1	Modelo .....	94
3.7.2	Controlador.....	94
3.7.3	Vistas.....	97
3.7.4	Export.....	98
3.8	Prácticas de grado.....	99
3.8.1	Modelo .....	99
3.8.2	Controlador.....	100
3.8.3	Vistas.....	103
3.8.4	Export.....	105
3.9	Laboratorios .....	106
3.9.1	Modelo .....	106
3.9.2	Controlador.....	106
3.9.3	Vistas.....	109
3.9.4	Export.....	111
3.10	Convenios .....	113
3.10.1	Modelo .....	113
3.10.2	Controlador.....	113
3.10.3	Vistas.....	116
3.10.4	Export.....	118
3.11	Redes Académicas.....	118
3.11.1	Modelo .....	118
3.11.2	Controlador.....	119
3.11.3	Vistas.....	121
3.11.4	Export.....	123
3.12	Movilidad.....	124

3.12.1	Modelo .....	124
3.12.2	Controlador.....	125
3.12.3	Vistas.....	129
3.12.4	Export.....	131
3.13	Investigación .....	133
3.13.1	Modelo .....	133
3.13.2	Controlador.....	133
3.13.3	Vistas.....	137
3.13.4	Export.....	138
3.14	Egresados .....	140
3.14.1	Modelo .....	140
3.14.2	Controlador.....	140
3.14.3	Vistas.....	142
3.14.4	Export.....	143

## **1 INTRODUCCIÓN**

A continuación, encontraran el manual del programador que dará a conocer y entender la forma en que se programaron los diferentes módulos y su lógica utilizada, también se mostraran los pasos.



## 2 CONFIGURACIÓN

Los siguientes pasos se realizaron para subir el proyecto del framework laravel llamado **GECI** al hosting Colombia con el dominio [www.labratoriosunisangil.com](http://www.labratoriosunisangil.com).

Como primer paso se debe compilar el proyecto de laravel mediante el siguiente código **npm run prod** dicho código debe ejecutarse directamente de la raíz del proyecto mediante la consola cmd o terminal de su preferencia. Luego se comprime el proyecto a formato **.zip** para posteriormente cargarlo. Por otra parte exportamos la base de datos creado para el proyecto en formato **SQL** desde el gestor de base de datos que se implementó en este caso **heidisql**.

Como siguiente paso se debe entrar al **Cpanel** del **hostingcolombia** mediante la siguiente dirección

[https://nury.colombiahosting.com.co:2083/cpsess6444280329/frontend/paper\\_lantern/index.html](https://nury.colombiahosting.com.co:2083/cpsess6444280329/frontend/paper_lantern/index.html)

Ingresar con las credenciales de acceso al **Cpanel**

Nombre de usuario: laborato

Contraseña: K64#~K85Cyz-7vb

Una vez hallamos ingresado debemos crear una base de datos en MYSQL desde el Cpanel en nuestro caso se le dieron las siguientes credenciales de acceso.

Base de datos = geci\_unisangil – laborato\_geci

Usuario = geci\_root

Contraseña = ^Z51X8b9(&SaiG\$-

Ingresamos al **phpmyadmin** del **Cpanel** e importamos la base de datos del proyecto.

Para subir el proyecto al Cpanel debemos entrar al menú de administrador de archivos y en la raíz debemos cargar nuestro archivo .zip y luego descomprimirlo debemos mover los archivos de la carpeta <>public></> a la carpeta publica del hosting <>public\_html></>

Posteriormente configuramos nuestro index la línea de código # 22 y # 36 ya que ahora se encuentra en la carpeta public\_html quedando así:

```
22 require __DIR__.'/../laravel/bootstrap/autoload.php';  
22
```

```
36 $app = require_once __DIR__.'/../laravel/bootstrap/app.php';
```

Por ultimo configuramos nuestro archivo **.env** de la siguiente forma, este archivo se encuentra en la carpeta raíz del proyecto llamada **laravel**.

~~En este momento nuestro proyecto ya debe estar funcionando en el siguiente dominio [www.laboratoriosunisangil.com](http://www.laboratoriosunisangil.com)~~

Credenciales de acceso para súper administrador  
Correo electrónico: michaelrodriguezhernandez@unisangil.edu.co

Contraseña: 123456789.

### 3 MODULO CONFIGURACIÓN

#### 3.1 Estado programa

##### 3.1.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```
app > Models > Departamento.php > Departamento
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Departamento extends Model
9  {
10     use HasFactory;
11
12     protected $table = "departamentos";
13
14     protected $fillable = [
15         'id',
16         'dep_nombre',
17         'dep_status'
18     ];
19 }
```

##### 3.1.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestran los controladores que se crearon para la ejecución del proyecto directa o indirectamente con el módulo de configuración. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > EstadoProgramaController.php > EstadoProgramaController > index
  1 <?php
  2 namespace App\Http\Controllers;
  3
  4 use App\Models\EstadoPrograma;
  5 use Illuminate\Http\Request;
  6 use App\Models>Status;
  7 use PDF;
  8 use RealRashid\SweetAlert\Facades\Alert;
  9 use Illuminate\Support\Facades\Auth;
 10
 11 class EstadoProgramaController extends Controller
 12 {
 13     public function index()
 14     {
 15         $user = Auth::user();
 16         if ($user->per_tipo_usuario == 1) [
 17             $estadoprogramas = EstadoPrograma::all();
 18             return view('configuracion/estadoprograma.index')->with('estadoprogramas', $estadoprogramas);
 19         } else {
 20             return redirect('/home');
 21         }
 22     }
 23
 24     public function create()
 25     {
 26         $user = Auth::user();
 27         if ($user->per_tipo_usuario == 1) {
 28             $estadoprogramas = Status::all();
 29             return view('configuracion/estadoprograma.create')->with('estadoprogramas', $estadoprogramas);
 30         } else {
 31             return redirect('/home');
 32         }
 33     }
 34
 35     public function store(Request $request)
 36     {
 37         $estadoprograma = new EstadoPrograma();
 38         $estadoprograma->est_nombre = $request->get('est_nombre');
 39         $estadoprograma->est_status = $request->get('est_status');
 40
 41         $rules = [
 42             'est_nombre' => 'required',
 43             'est_status' => 'required'
 44         ];
 45
 46         $messages = [
 47             'est_nombre.required' => 'El campo nombre del estado es requerido'
 48         ];
 49
 50         $this->validate($request, $rules, $messages);
 51
 52         $estadoprograma->save();
 53
 54         Alert::success('Registro Exitoso');
 55
 56         return redirect('/estadoprograma');
 57     }

```

```
59     public function show($id)
60     {
61         $user = Auth::user();
62         if ($user->per_tipo_usuario == 1) {
63             $status = Status::all();
64             $estadoprograma = EstadoPrograma::find($id);
65             return view('configuracion/estadoprograma.show')
66                 ->with('status', $status)
67                 ->with('estadoprograma', $estadoprograma);
68         } else {
69             return redirect('/home');
70         }
71     }
72
73     public function edit($id)
74     {
75         $user = Auth::user();
76         if ($user->per_tipo_usuario == 1) {
77             $status = Status::all();
78             $estadoprograma = EstadoPrograma::find($id);
79             return view('configuracion/estadoprograma.edit')
80                 ->with('status', $status)
81                 ->with('estadoprograma', $estadoprograma);
82         } else {
83             return redirect('/home');
84         }
85     }
```

```
87     public function update(Request $request, $id)
88     {
89         $estadoprograma = EstadoPrograma::find($id);
90         $estadoprograma->est_nombre = $request->get('est_nombre');
91         $estadoprograma->est_status = $request->get('est_status');
92
93         $estadoprograma->save();
94
95         Alert::success('Registro Actualizado');
96         return redirect('/estadoprograma');
97     }
98
99     public function destroy($id)
100    {
101        $estadoprograma = EstadoPrograma::find($id);
102        $estadoprograma->delete();
103        Alert::success('Registro Eliminado');
104        return redirect('/estadoprograma');
105    }
```

```
99     public function destroy($id)
100    {
101        $estadoprograma = EstadoPrograma::find($id);
102        $estadoprograma->delete();
103        Alert::success('Registro Eliminado');
104        return redirect('/estadoprograma');
105    }
106
107    public function pdf(Request $request)
108    {
109        $estadoprogramas = EstadoPrograma::all();
110        $view = \view('configuracion/estadoprograma.pdf', compact('estadoprogramas'))->render();
111        $pdf = \App::make('dompdf.wrapper');
112        $pdf->loadHTML($view);
113        return $pdf->stream('reporte.pdf');
114    }
115 }
```

### 3.1.3 Vistas

A continuación se muestran las vistas creadas para el modelo **estadoprograma**.

#### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda **nuevo**, además de actualizar y eliminar un registro almacenado.

```
resources > views > configuracion > estadoprograma > index.blade.php
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-toggle-on"></i> Configuración / estado programa</h1>
4      <p>Estado de programas: Indica el estado del programa académico registrado</p>
5  @endsection
6  @section('content')
7      <div class="container-fluid">
8          <div class="tile col-md-12 p-3">
9              <div class="card-body">...
61             </div>
62         </div>
63     </div>
64  @endsection
65
```

#### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevo estados para los programas.

```
resources > views > configuracion > estadoprograma > create.blade.php
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
4  @section('message')
5      <p>Formulario de registro nuevo estado programa</p>
6  @endsection
7  @endsection
8  @section('content')
9      <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title"><i class="fab fa-wpforms"></i> Nuevo estado</h4>
12         <form action="/estadoprograma" method="post">...
33             </form>
34         </div>
35     </div>
36  @endsection
```

#### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```
resources > views > configuracion > estadoprograma > edit.blade.php
1 @extends('layouts.app')
2 @section('title')
3     <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
4     <p>Formulario de actualización de la información</p>
5 @endsection
6 @section('content')
7     <div class="container-fluid">
8         <div class="tile card__config">
9             <h4 class="tile title">Actualizar información</h4>
10            <form action="/estadoprograma/{{ $estadoprograma->id }}" method="POST">...
11                </form>
12            </div>
13        </div>
14    @endsection
```

#### d. show.blade.php

Esta vista permite la visualización de un registro específico.

```
resources > views > configuracion > estadoprograma > show.blade.php
1 @extends('layouts.app')
2 @section('title')
3     <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
4 @section('message')
5     <p>Información almacenada en el sistema</p>
6 @endsection
7 @endsection
8 @section('content')
9     <div class="container-fluid">
10        <div class="tile card__config">
11            <h4 class="tile title"><i class="far fa-question-circle"></i> Vista registro</h4>
12        <div class="row mb-3">...
13            </div>
14        </div>
15    </div>
16 @endsection
```

## 3.2 Departamento

### 3.2.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.



```
app > Models > Departamento.php > Departamento > $fillable
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Departamento extends Model
9  {
10     use HasFactory;
11
12     protected $table = "departamentos";
13
14     protected $fillable = [
15         'id',
16         'dep_nombre',
17         'dep_status'
18     ];
19 }
20 }
```

### 3.2.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestran los controladores que se crearon para la ejecución del proyecto directa o indirectamente con el módulo de configuración. Su ubicación es en la carpeta **App\Http\Controllers**.

```
app > Http > Controllers > DepartamentoController.php > DepartamentoController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Departamento;
6  use App\Models>Status;
7  use App\Models\User;
8  use Illuminate\Http\Request;
9  use RealRashid\SweetAlert\Facades\Alert;
10 use Illuminate\Support\Facades\Auth;
11
12
13 class DepartamentoController extends Controller
14 [
15     /**
16      * Display a listing of the resource.
17      *
18      * @return \Illuminate\Http\Response
19     */
20    public function index()
21    {
22        $user = Auth::user();
23        if ($user->per_tipo_usuario == 1) {
24            $departamentos = Departamento::all();
25            return view('configuracion/departamento.index')->with('departamentos', $departamentos);
26        } else {
27            return redirect('/home');
28        }
29    }

```

```
36    public function create()
37    {
38        $user = Auth::user();
39        if ($user->per_tipo_usuario == 1) {
40            $estadoprogramas = Status::all();
41            return view('configuracion/departamento.create')->with('estadoprogramas', $estadoprogramas);
42        } else {
43            return redirect('/home');
44        }
45    }

```

```
53    public function store(Request $request)
54    {
55        $departamentos = new Departamento();
56        $departamentos->dep_nombre = $request->get('dep_nombre');
57        $departamentos->dep_status = $request->get('dep_status');

58        $rules = [
59            'dep_nombre' => 'required'
60        ];
61
62        $messages = [
63            'dep_nombre.required' => 'El campo nombre del departamento es requerido'
64        ];
65
66        $this->validate($request, $rules, $messages);

67        $departamentos->save();

68        Alert::success('Registro Exitoso');

69        return redirect('/departamento');
70    }
71
72
73
74 }

```

```
82    public function show($id)
83    {
84        $user = Auth::user();
85        if ($user->per_tipo_usuario == 1) {
86            $status = Status::all();
87            $departamento = Departamento::find($id);
88            return view('configuracion/departamento.show')
89                ->with('status', $status)
90                ->with('departamento', $departamento);
91        } else {
92            return redirect('/home');
93        }
94    }

```

```

102 |     public function edit($id)
103 |     {
104 |         $user = Auth::user();
105 |         if ($user->per_tipo_usuario == 1) {
106 |             $status = Status::all();
107 |             $departamento = Departamento::find($id);
108 |             return view('configuracion/departamento.edit')
109 |                 ->with('departamento', $departamento)
110 |                 ->with('status', $status);
111 |         } else {
112 |             return redirect('/home');
113 |         }
114 |     }
115 |
116 |
117 |
118 |
119 |
120 |
121 |
122 |
123 |     public function update(Request $request, $id)
124 |     {
125 |         $departamento = Departamento::find($id);
126 |         $departamento->dep_nombre = $request->get('dep_nombre');
127 |         $departamento->dep_status = $request->get('dep_status');
128 |
129 |         $departamento->save();
130 |
131 |         Alert::success('Registro Actualizado');
132 |
133 |         return redirect('/departamento');
134 |     }
135 |
136 |
137 |
138 |
139 |
140 |
141 |
142 |     public function destroy($id)
143 |     {
144 |         $departamento = Departamento::find($id);
145 |         $departamento->delete();
146 |
147 |         Alert::success('Registro Eliminado');
148 |
149 |         return redirect('/departamento');
150 |     }
151 |
152 |     public function pdf(Request $request)
153 |     {
154 |         $departamentos = Departamento::all();
155 |         $view = \view('configuracion/departamento.pdf', compact('departamentos'))->render();
156 |         $pdf = \App::make('dompdf.wrapper');
157 |         $pdf->loadHTML($view);
158 |         return $pdf->stream('departamentos-report.pdf');
159 |     }
160 |
161 |

```

### 3.2.3 Vistas

A continuación se muestran las vistas creadas para el modelo **departamento**.

#### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda **nuevo**, además de actualizar y eliminar un registro almacenado.

```

resources > views > configuracion > departamento > index.blade.php
1  @extends('layouts.app')
2  @section('title')
3  <h1 class="titulo"><i class="fas fa-globe-americas"></i> Configuración / departamentos</h1>
4  @section('message')
5  |   <p>Departamentos: Listado departamentos registrados</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10 >     <div class="tile col-md-12 p-3">...
11 >         <div class="card-body">...
12 >             </div>
13 >         </div>
14 >     </div>
15 >     @endsection

```

### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevo estados para los programas.

```

resources > views > configuracion > departamento > create.blade.php
1  @if (Auth::user()->per_tipo_usuario == 1)
2  @extends('layouts.app')
3  @section('title')
4  |   <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
5  @section('message')
6  |   <p>Formulario de registro nuevo departamento</p>
7  @endsection
8  @endsection
9  @section('content')
10 >     <div class="container-fluid">
11 >         <div class="tile card_config">
12 >             <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro departamento</h4>
13 >             <form action="/departamento" method="post">...
14 >             </form>
15 >         </div>
16 >     </div>
17 >     @endsection
18 @else
19 @php
20     return view('home');
21 @endphp
22 @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > configuracion > departamento > edit.blade.php
1  @extends('layouts.app')
2  @section('title')
3  <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
4  @section('message')
5  <p>Formulario de actualización de la información</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10 >     <div class="tile card_config">
11 >         <h4 class="tile title">Actualizar información</h4>
12 >         <form action="/departamento/{{ $departamento->id }}" method="POST">...
13 >         </form>
14 >     </div>
15 >     @endsection

```

d. show.blade.php

Esta vista permite la visualización de un registro específico.

```
resources > views > configuracion > departamento >  show.blade.php
1  @extends('layouts.app')
2  @section('title')
3    <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
4  @section('message')
5    <p>Información almacenada en el sistema</p>
6  @endsection
7  @endsection
8  @section('content')
9    <div class="container-fluid">
10   <div class="tile card__config">...
11   ...
12   ...
13   ...
14   ...
15   ...
16   ...
17   ...
18   ...
19   ...
20   ...
21   ...
22   ...
23   ...
24   ...
25   ...
26   ...
27  @endsection
```

## 3.3 Municipio

### 3.3.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```
app > Models > Municipio.php > ...
1 <?php
2 namespace App\Models;
3
4 use Illuminate\Database\Eloquent\Factories\HasFactory;
5 use Illuminate\Database\Eloquent\Model;
6
7 class Municipio extends Model
8 {
9     use HasFactory;
10
11     protected $table = "municipios";
12
13     protected $fillable = [
14         'id',
15         'mun_nombre',
16         'mun_departamento'
17     ];
18
19     public function investigacion(){
20         return $this->hasMany(Investigacion::class, 'id');
21     }
22 }
23 }
```

### 3.3.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestran los controladores que se crearon para la ejecución del proyecto directa o indirectamente con el módulo de configuración. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > MunicipioController.php
  1  <?php
  2  namespace App\Http\Controllers;
  3
  4  use App\Models\Departamento;
  5  use App\Models\Municipio;
  6  use App\Models>Status;
  7  use Illuminate\Http\Request;
  8  use RealRashid\SweetAlert\Facades\Alert;
  9  use Illuminate\Support\Facades\Auth;
10
11 class MunicipioController extends Controller
12 {
13     /**
14      * Display a listing of the resource.
15      *
16      * @return \Illuminate\Http\Response
17     */
18     public function index()
19     {
20         $user = Auth::user();
21         if ($user->per_tipo_usuario == 1) {
22             $municipios = Municipio::all();
23             return view('configuracion/municipio.index')->with('municipios', $municipios);
24         } else {
25             return redirect('/home');
26         }
27     }
28
29
30
31
32
33
34     public function create()
35     {
36         $user = Auth::user();
37         if ($user->per_tipo_usuario == 1) {
38             $status = Status::all();
39             $departamentos = Departamento::all();
40             return view('configuracion/municipio.create')
41                 ->with('departamentos', $departamentos)
42                 ->with('status', $status);
43         } else {
44             return redirect('/home');
45         }
46     }
47
48     /**
49      * Store a newly created resource in storage.
50      *
51      * @param \Illuminate\Http\Request $request
52      * @return \Illuminate\Http\Response
53     */
54     public function store(Request $request)
55     {
56         $municipios = new Municipio();
57         $municipios->mun_departamento = $request->get('mun_departamento');
58         $municipios->mun_nombre = $request->get('mun_nombre');
59         $municipios->mun_status = $request->get('mun_status');
60
61         $rules = [
62             'mun_nombre' => 'required'
63         ];
64
65         $messages = [
66             'mun_nombre.required' => 'El campo nombre del municipio es requerido'
67         ];
68
69         $this->validate($request, $rules, $messages);
70
71         $municipios->save();
72     }

```

```

84     public function show($id)
85     {
86         $user = Auth::user();
87         if ($user->per_tipo_usuario == 1) {
88             $departamentos = Departamento::all();
89             $status = Status::all();
90             $municipio = Municipio::find($id);
91
92             return view('configuracion/municipio.show')
93                 ->with('municipio', $municipio)
94                 ->with('departamentos', $departamentos)
95                 ->with('status', $status);
96         } else {
97             return redirect('/home');
98         }
99     }
100
101 /**
102 * Show the form for editing the specified resource.
103 *
104 * @param int $id
105 * @return \Illuminate\Http\Response
106 */
107 public function edit($id)
108 {
109     $user = Auth::user();
110     if ($user->per_tipo_usuario == 1) {
111         $departamentos = Departamento::all();
112         $status = Status::all();
113         $municipio = Municipio::find($id);
114
115         return view('configuracion/municipio.edit')
116             ->with('municipio', $municipio)
117             ->with('departamentos', $departamentos)
118             ->with('status', $status);
119     } else {
120         return redirect('/home');
121     }
122 }
123
124
125 public function update(Request $request, $id)
126 {
127     $municipio = Municipio::find($id);
128     $municipio->mun_departamento = $request->get('mun_departamento');
129     $municipio->mun_nombre = $request->get('mun_nombre');
130     $municipio->mun_status = $request->get('mun_status');
131
132     $municipio->save();
133
134     Alert::success('Registro Actualizado');
135
136     return redirect('/municipio');
137 }
138
139 /**
140 * Remove the specified resource from storage.
141 *
142 * @param int $id
143 * @return \Illuminate\Http\Response
144 */
145 public function destroy($id)
146 {
147     $municipio = Municipio::find($id);
148     $municipio->delete();
149
150     Alert::success('Registro Eliminado');
151
152     return redirect('/municipio');
153 }
154
155
156 public function pdf(Request $request)
157 {
158     $municipios = Municipio::all();
159     $view = \view('configuracion/municipio.pdf', compact('municipios'))->render();
160     $pdf = \App::make('dompdf.wrapper');
161     $pdf->loadHTML($view);
162     return $pdf->stream('municipio-report.pdf');
163 }
164
165
166
167
168
169 }
```

### 3.3.3 Vistas

A continuación se muestran las vistas creadas para el modelo municipio.

#### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > configuracion > municipio > index.blade.php
1 @extends('layouts.app')
2 @section('title')
3     <h1 class="titulo"><i class="fas fa-globe-americas"></i> Configuración / municipio</h1>
4     <p>Municipios: Listado municipios o sedes registrados</p>
5 @endsection
6 @section('content')
7     <div class="container-fluid">
8         <div class="tile col-md-12 p-3">
9             <div class="card-body">...
60         </div>
61     </div>
62 </div>
63 @endsection
```

#### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevos municipios o sedes.

```
resources > views > configuracion > municipio > create.blade.php
1 @extends('layouts.app')
2 @section('title')
3     <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
4 @section('message')
5     <p>Formulario de registro nuevo municipio</p>
6 @endsection
7 @endsection
8 @section('content')
9     <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro municipio</h4>
12         <form action="/municipio" method="post">...
44         </form>
45     </div>
46 </div>
47 @endsection
```

#### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > configuracion > municipio > edit.blade.php
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
4      <p>Formulario de actualización de la información</p>
5  @endsection
6  @section('content')
7      <div class="container-fluid">
8          <div class="tile card_config">
9              <h4 class="tile title">Actualizar información</h4>
10         <form action="/municipio/{{ $municipio->id }}" method="POST">...
11         </form>
12     </div>
13 </div>
14 @endsection

```

#### d. show.blade.php

Esta vista permite la visualización de un registro específico.

```

resources > views > configuracion > municipio > show.blade.php > ...
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
4  @section('message')
5      <p>Información almacenada en el sistema</p>
6  @endsection
7  @endsection
8  @section('content')
9      <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title"><i class="far fa-question-circle"></i> Vista registro</h4>
12             <div class="row mb-3">
13                 <label for="mun_departamento">{{ __('Departamento *') }}</label>
14                 <div class="col-md-12">
15                     <select class="js-example-placeholder-single form-select" name="mun_departamento" id="mun_departamento" disabled>
16                         <option value="">---- SELECCIONE ----</option>
17                         @foreach ($departamentos as $departamento)
18                             <option value="{{ $departamento->id }}>
19                                 {{ $departamento->id == $municipio->mun_departamento ? 'selected' : '' }}>
20                                 {{ $departamento->dep_nombre }}</option>
21                         @endforeach
22                     </select>
23                 </div>
24             </div>
25             <div class="row mb-3">
26                 <label for="mun_nombre">{{ __('Municipio *') }}</label>
27                 <div class="col-md-12">
28                     <input id="mun_nombre" type="text" class="form-control @error('mun_nombre') is-invalid @enderror" name="mun_nombre" value="{{ $municipio->mun_nombre }}" disabled required autocomplete="mun_nombre" autofocus>
29                     @error('mun_nombre')
30                         <span class="invalid-feedback" role="alert">
31                             <strong>{{ $message }}</strong>
32                         </span>
33                     @enderror
34                 </div>
35             </div>
36         </div>
37     </div>
38 
```

## 3.4 Facultad

Este módulo pertenece al módulo de configuración el cual permite el registro de facultades de la institución, las cuales son necesarias para el correcto funcionamiento de otros módulos del sistemas ya que es información requerida para el registro de nuevos programas, estudiantes, docentes etc.

### 3.4.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```
app > Models > Facultad.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Facultad extends Model
9  {
10     use HasFactory;
11
12     protected $table = "facultades";
13
14     protected $fillable = [
15         'id',
16         'fac_nombre',
17         'fac_status'
18     ];
19
20 }
```

### 3.4.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestran los controladores que se crearon para la ejecución del proyecto directa o indirectamente con el módulo de configuración. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > FacultadController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Facultad;
6 use App\Models>Status;
7 use Illuminate\Http\Request;
8 use RealRashid\SweetAlert\Facades\Alert;
9 use Illuminate\Support\Facades\Auth;
10
11 class FacultadController extends Controller
12 {
13     /**
14      * Display a listing of the resource.
15      *
16      * @return \Illuminate\Http\Response
17      */
18     public function index()
19     {
20         $user = Auth::user();
21         if ($user->per_tipo_usuario == 1) {
22             $facultades = Facultad::all();
23             return view('configuracion/facultad.index')->with('facultades', $facultades);
24         } else {
25             return redirect('/home');
26         }
27     }
28
29
30
31
32
33
34     public function create()
35     {
36         $user = Auth::user();
37         if ($user->per_tipo_usuario == 1) {
38             $status = Status::all();
39             return view('configuracion/facultad.create')->with('status', $status);
40         } else {
41             return redirect('/home');
42         }
43     }
44
45
46
47
48
49
50
51     public function store(Request $request)
52     {
53         $facultades = new Facultad();
54         $facultades->fac_nombre = $request->get('fac_nombre');
55         $facultades->fac_status = $request->get('fac_status');
56
57         $rules = [
58             'fac_nombre' => 'required'
59         ];
56
57         $messages = [
58             'fac_nombre.required' => 'El campo nombre de la facultad es requerido'
59         ];
60
61         $this->validate($request, $rules, $messages);
62
63         $facultades->save();
64
65         Alert::success('Registro Exitoso');
66
67         return redirect('/facultad');
68     }
69
70
71
72 }

```

```
80     public function show($id)
81     {
82         $user = Auth::user();
83         if ($user->per_tipo_usuario == 1) {
84             $status = Status::all();
85             $facultad = Facultad::find($id);
86             return view('configuracion/facultad.show')
87                 ->with('facultad', $facultad)
88                 ->with('status', $status);
89         } else {
90             return redirect('/home');
91         }
92     }
93
94     /**
95      * Show the form for editing the specified resource.
96      *
97      * @param int $id
98      * @return \Illuminate\Http\Response
99      */
100    public function edit($id)
101    {
102        $user = Auth::user();
103        if ($user->per_tipo_usuario == 1) {
104            $status = Status::all();
105            $facultad = Facultad::find($id);
106            return view('configuracion/facultad.edit')
107                ->with('facultad', $facultad)
108                ->with('status', $status);
109        } else {
110            return redirect('/home');
111        }
112    }

```

```
121    public function update(Request $request, $id)
122    {
123        $facultad = Facultad::find($id);
124        $facultad->fac_nombre = $request->get('fac_nombre');
125        $facultad->fac_status = $request->get('fac_status');
126
127        $facultad->save();
128
129        Alert::success('Registro Actualizado');
130
131        return redirect('/facultad');
132    }

```

```
140    public function destroy($id)
141    {
142        $facultad = Facultad::find($id);
143        $facultad->delete();
144
145        Alert::success('Registro Eliminado');
146
147        return redirect('/facultad');
148    }
149
150    public function pdf(Request $request)
151    {
152        $facultades = Facultad::all();
153        $view = \view('configuracion/facultad.pdf', compact('facultades'))->render();
154        $pdf = \App::make('dompdf.wrapper');
155        $pdf->loadHTML($view);
156        return $pdf->stream('facultad-reporte.pdf');
157    }
158 }
```

### 3.4.3 Vistas

A continuación se muestran las vistas creadas para el modelo facultad.

#### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > configuracion > facultad > index.blade.php
1  @extends('layouts.app')
2  @section('title')
3  <h1 class="titulo"><i class="fas fa-laptop-house"></i> Configuración / facultad</h1>
4  @section('message')
5  <p>Facultades: Listado facultades registradas</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10 <div class="tile col-md-12 p-3">
11 >   <div class="card-body">...
12 </div>
13 </div>
14 </div>
15 @endsection
```

#### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevas facultades.

```
resources > views > configuracion > facultad > create.blade.php
1  @extends('layouts.app')
2  @section('title')
3  <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
4  @section('message')
5  <p>Formulario de registro nueva facultad</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10 <div class="tile card_config">
11 <h4 class="tile_title"><i class="fab fa-wpforms"></i> Registro facultad</h4>
12 >   <form action="/facultad" method="post">...
13 </form>
14 </div>
15 </div>
16 @endsection
```

#### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > configuracion > facultad > edit.blade.php
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
4  @section('message')
5      <p>Formulario de actualización de la información</p>
6  @endsection
7  @endsection
8  @section('content')
9      <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title">Actualizar información</h4>
12         <form action="/facultad/{{ $facultad->id }}" method="POST"> ...
13             </form>
14         </div>
15     </div>
16     @endsection

```

#### d. show.blade.php

Esta vista permite la visualización de un registro específico.

```

resources > views > configuracion > facultad > show.blade.php > ...
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
4  @section('message')
5      <p>Información almacenada en el sistema</p>
6  @endsection
7  @endsection
8  @section('content')
9      <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title"><i class="far fa-question-circle"></i> Vista registro</h4>
12             <div class="row mb-3">
13                 <label for="fac_nombre">{{ __('Facultad *') }}</label>
14                 <div class="col-md-12">
15                     <input id="fac_nombre" type="text" class="form-control" @error('fac_nombre') is-invalid @enderror
16                     name="fac_nombre" value="{{ $facultad->fac_nombre }}" disabled required autocomplete="fac_nombre"
17                     autofocus>
18                     @error('fac_nombre')
19                         <span class="invalid-feedback" role="alert">
20                             <strong>{{ $message }}</strong>
21                         </span>
22                     @enderror
23                 </div>
24             </div>
25         </div>
26     </div>
27     @endsection

```

## 3.5 Nivel formación

Este módulo pertenece al módulo de configuración el cual permite el registro de nivel de formación de los programas, el cual es requerido para el correcto funcionamiento del módulo programa, ya que al momento de registrar un nuevo programa es necesario conocer el nivel de formación.

### 3.5.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta app/models.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de Eloquent llamada model la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo \$fillable.

Este contiene un método llamado programas() el cual nos permite acceder a la información almacenada en nuestra base de datos a la entidad investigación mediante la relación de ambas tablas.

```
app > Models > NivelFormacion.php > ...
1 <?php
2 namespace App\Models;
3
4 use Illuminate\Database\Eloquent\Factories\HasFactory;
5 use Illuminate\Database\Eloquent\Model;
6
7 class NivelFormacion extends Model
8 {
9     use HasFactory;
10
11     protected $table = "nivelformacion";
12
13     protected $fillable = [
14         'id',
15         'niv_nombre',
16     ];
17
18     public function programas(){
19         return $this->hasMany(Programa::class, 'id');
20     }
21
22 }
```

### 3.5.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestran los

controladores que se crearon para la ejecución del proyecto directa o indirectamente con el módulo de configuración. Su ubicación es en la carpeta App\Http\Controllers.

```
app > Http > Controllers > NivelFormacionController.php > NivelFormacionController > export
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Exports\NivelFormacionExport;
6  use App\Models\NivelFormacion;
7  use App\Models>Status;
8  use Illuminate\Http\Request;
9  use RealRashid\SweetAlert\Facades\Alert;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\DB;
12 use Matwebsite\Excel\Facades\Excel;
13
14 class NivelFormacionController extends Controller
15 {
16     /**
17      * Display a listing of the resource.
18      *
19      * @return \Illuminate\Http\Response
20     */
21     public function index()
22     {
23         $user = Auth::user();
24         if ($user->per_tipo_usuario == 1) {
25             $niveformaciones = NivelFormacion::all();
26             return view('configuracion/nivelformacion.index')->with('niveformaciones', $niveformaciones);
27         } else {
28             return redirect('/home');
29         }
30     }
31
32
33
34
35
36
37     public function create()
38     {
39         $user = Auth::user();
40         if ($user->per_tipo_usuario == 1) {
41             return view('configuracion/nivelformacion.create');
42         } else {
43             return redirect('/home');
44         }
45     }
46
47
48
49
50
51
52
53     public function store(Request $request)
54     {
55         $niveles = new NivelFormacion();
56         $niveles->niv_nombre = $request->get('niv_nombre');
57
58         $rules = [
59             'niv_nombre' => 'required'
60         ];
61
62         $message = [
63             'niv_nombre.required' => 'El campo nivel formación es requerido'
64         ];
65
66         $this->validate($request, $rules, $message);
67
68         $niveles->save();
69
70         DB::table('acciones_plataforma')->insert([
71             'usuario' => Auth::user()->id,
72             'accion' => 'insertar',
73             'modulo' => 'nivelformacion'
74         ]);
75
76         Alert::success('Registro Exitoso');
77
78         return redirect('/nivelformacion');
79     }
80 }
```

```
87     public function show($id)
88     {
89         $user = Auth::user();
90         if ($user->per_tipo_usuario == 1) {
91             $nivelformacion = NivelFormacion::find($id);
92             return view('configuracion/nivelformacion.show')
93                 ->with('nivelformacion', $nivelformacion);
94         } else {
95             return redirect('/home');
96         }
97     }
```

```
105    public function edit($id)
106    {
107        $user = Auth::user();
108        if ($user->per_tipo_usuario == 1) {
109            $nivelformacion = NivelFormacion::find($id);
110            return view('configuracion/nivelformacion.edit')
111                ->with('nivelformacion', $nivelformacion);
112        } else {
113            return redirect('/home');
114        }
115    }
116
117 /**
118 * Update the specified resource in storage.
119 *
120 * @param \Illuminate\Http\Request $request
121 * @param int $id
122 * @return \Illuminate\Http\Response
123 */
124 public function update(Request $request, $id)
125 {
126     $nivelformacion = NivelFormacion::find($id);
127     $nivelformacion->niv_nombre = $request->get('niv_nombre');
128
129     $nivelformacion->save();
130
131     DB::table('acciones_plataforma')->insert([
132         'usuario' => Auth::user()->id,
133         'accion' => 'editar',
134         'modulo' => 'nivelformacion'
135     ]);
136
137     Alert::success('Registro Actualizado');
138
139     return redirect('/nivelformacion');
140 }
```

```
148 public function destroy($id)
149 {
150     $nivel = NivelFormacion::find($id);
151
152     $veriNiv = DB::table('programas')->where('pro_nivel_formacion', $id);
153
154     if ($veriNiv->count() > 0) {
155         Alert::warning('No se pude eliminar el nivel formación, debido a que esta asociado a otra entidad');
156         return redirect('/nivelformacion');
157     } else {
158         $nivel->delete();
159
160         DB::table('acciones_plataforma')->insert([
161             'usuario' => Auth::user()->id,
162             'accion' => 'eliminar',
163             'modulo' => 'nivelformacion'
164         ]);
165
166         Alert::success('Registro Eliminado');
167
168         return redirect('/nivelformacion');
169     }
170 }
```

```

172     public function pdf(Request $request)
173     {
174         $nivelformaciones = NivelFormacion::all();
175         if ($nivelformaciones->count() <= 0) {
176             Alert::warning('No hay registros');
177             return redirect('/nivelformacion');
178         } else {
179             $view = \view('configuracion/nivelformacion.pdf', compact('nivelformaciones'))->render();
180             $pdf = \App::make('dompdf.wrapper');
181             $pdf->loadHTML($view);
182
183             DB::table('acciones_plataforma')->insert([
184                 'usuario' => Auth::user()->id,
185                 'accion' => 'pdf',
186                 'modulo' => 'nivelformacion'
187             ]);
188
189             return $pdf->stream('nivelformacion-reporte.pdf');
190         }
191     }
192
193     public function export()
194     {
195         $nivelformaciones = NivelFormacion::all();
196         if ($nivelformaciones->count() <= 0) {
197             Alert::warning('No hay registros');
198             return redirect('/nivelformacion');
199         } else {
200
201             DB::table('acciones_plataforma')->insert([
202                 'usuario' => Auth::user()->id,
203                 'accion' => 'excel',
204                 'modulo' => 'nivelformacion'
205             ]);
206
207             return Excel::download(new NivelFormacionExport, 'nivelformacion.xlsx');
208         }
209     }
210 }
```

### 3.5.3 Vistas

A continuación se muestran las vistas creadas para el modelo nivel formación.

#### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > configuracion > nivelformacion > index.blade.php
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-terminal"></i> Configuración / nivel de formación programas</h1 class="titulo">
4  @section('message')
5      <p>Nivel de formación: Listado niveles de formación registradas</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10     <div class="tile col-md-12 p-3">
11         <div class="card-body">...
12     </div>
13     </div>
14     </div>
15     @endsection
16 
```

#### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevos niveles de formación.

```
resources > views > configuracion > nivelformacion > 🗂 create.blade.php
1  @extends('layouts.app')
2  @section('title')
3  |   <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
4  @section('message')
5  |   <p>Formulario de registro nuevo nivel de formación programado</p>
6  @endsection
7  @endsection
8  @section('content')
9  |   <div class="container-fluid">
10 |       <div class="tile card_config">
11 |           <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro nivel de formación</h4>
12 |           <form action="/nivelformacion" method="post">...
13 |               </form>
14 |           </div>
15 |       </div>
16 |   @endsection
```

#### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```
resources > views > configuracion > nivelformacion > 🗂 edit.blade.php
1  @extends('layouts.app')
2  @section('title')
3  |   <h1 class="titulo"><i class="fa-edit"></i> Formulario de edición de datos</h1>
4  @section('message')
5  |   <p>Formulario de actualización de la información</p>
6  @endsection
7  @endsection
8  @section('content')
9  |   <div class="container-fluid">
10 |       <div class="tile card_config">
11 |           <h4 class="tile title"> Actualizar información</h4>
12 |           <form action="/nivelformacion/{{ $nivelformacion->id }}" method="POST">...
13 |               </form>
14 |           </div>
15 |       </div>
16 |   @endsection
```

#### d. show.blade.php

Esta vista permite la visualización de un registro específico.

```
resources > views > configuracion > nivelformacion > show.blade.php > ...
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
4  @section('message')
5      <p>Información almacenada en el sistema</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10     <div class="tile card_config">
11         <h4 class="tile title"><i class="far fa-question-circle"></i> Vista registro</h4>
12         <div class="row mb-3">
13             <label for="niv_nombre">{{ __('Nivel Formación') }}</label>
14             <div class="col-md-12">
15                 <input id="niv_nombre" type="text" class="form-control" @error('niv_nombre') is-invalid @enderror
16                 name="niv_nombre" value="{{ $nivelformacion->niv_nombre }}" autocomplete="niv_nombre" readonly
17                 autofocus>
18                 @error('niv_nombre')
19                     <span class="invalid-feedback" role="alert">
20                         <strong>{{ $message }}</strong>
21                     </span>
22                 @enderror
23             </div>
24         </div>
25     </div>
26 </div>
27 @endsection
```

## 3.6 Metodología

Este módulo pertenece al módulo de configuración el cual permite el registro de metodologías de la institución, el cual es necesario para el funcionamiento del módulo programas ya que este es requerido al momento de registrar un programas.

### 3.6.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta app/models.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de Eloquent llamada model la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo \$fillable.

```
app > Models > Metodologia.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Metodologia extends Model
9  {
10     use HasFactory;
11
12     protected $table = "metodologia";
13
14     protected $fillables = [
15         'id',
16         'met_nombre',
17     ];
18
19 }
```

### 3.6.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestran los controladores que se crearon para la ejecución del proyecto directa o indirectamente con el módulo de configuración. Su ubicación es en la carpeta App\Http\Controllers.

```

app > Http > Controllers > MetodologiaController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Exports\MetodologiaExport;
6  use App\Models\Metodologia;
7  use Illuminate\Http\Request;
8  use RealRashid\SweetAlert\Facades\Alert;
9  use Illuminate\Support\Facades\Auth;
10 use Illuminate\Support\Facades\DB;
11 use Maatwebsite\Excel\Facades\Excel;
12
13 class MetodologiaController extends Controller
14 {
15
16     public function index()
17     {
18         $user = Auth::user();
19         if ($user->per_tipo_usuario == 1) {
20             $metodologias = Metodologia::all();
21             return view('configuracion/metodologia.index')->with('metodologias', $metodologias);
22         } else {
23             return redirect('/home');
24         }
25     }
26
27
28     public function create()
29     {
30         $user = Auth::user();
31         if ($user->per_tipo_usuario == 1) {
32             return view('configuracion/metodologia.create');
33         } else {
34             return redirect('/home');
35         }
36     }
37
38     public function store(Request $request)
39     {
40         $metodologias = new Metodologia();
41         $metodologias->met_nombre = $request->get('met_nombre');
42
43         $metodologias->save();
44
45         DB::table('acciones_plataforma')->insert([
46             'usuario' => Auth::user()->id,
47             'accion' => 'insertar',
48             'modulo' => 'metodologia'
49         ]);
50
51         Alert::success('Registro Exitoso');
52
53         return redirect('/metodologia');
54     }
55
56     public function show($id)
57     {
58         $user = Auth::user();
59         if ($user->per_tipo_usuario == 1) {
60             $metodologia = Metodologia::find($id);
61             return view('configuracion/metodologia.show')
62                 ->with('metodologia', $metodologia);
63         } else {
64             return redirect('/home');
65         }
66     }
67
68     public function edit($id)
69     {
70         $user = Auth::user();
71         if ($user->per_tipo_usuario == 1) {
72             $metodologia = Metodologia::find($id);
73             return view('configuracion/metodologia.edit')
74                 ->with('metodologia', $metodologia);
75         } else {
76             return redirect('/home');
77         }
78     }

```

```

79     public function update(Request $request, $id)
80     {
81         $metodologia = Metodologia::find($id);
82         $metodologia->met_nombre = $request->get('met_nombre');
83
84         $metodologia->save();
85
86         DB::table('acciones_plataforma')->insert([
87             'usuario' => Auth::user()->id,
88             'accion' => 'editar',
89             'modulo' => 'metodologia'
90         ]);
91
92         Alert::success('Registro Actualizado');
93
94         return redirect('/metodologia');
95     }
96
97
98     public function destroy($id)
99     {
100         $metodologia = Metodologia::find($id);
101
102         $veriMet = DB::table('programas')->where('pro_metodologia', $id);
103
104         if ($veriMet->count() > 0) {
105             Alert::warning('No se pudo eliminar la metodologia, debido a que esta asociado a otra entidad');
106             return redirect('/metodologia');
107         } else {
108             $metodologia->delete();
109
110             DB::table('acciones_plataforma')->insert([
111                 'usuario' => Auth::user()->id,
112                 'accion' => 'eliminar',
113                 'modulo' => 'metodologia'
114             ]);
115
116             Alert::success('Registro Eliminado');
117
118             return redirect('/metodologia');
119         }
120     }
121
122     public function pdf(Request $request)
123     {
124         $metodologias = Metodologia::all();
125         if ($metodologias->count() <= 0) {
126             Alert::warning('No hay registros');
127             return redirect('/metodologia');
128         } else {
129             $view = \view('configuracion/metodologia.pdf', compact('metodologias'))->render();
130             $pdf = \App::make('dompdf.wrapper');
131             $pdf->loadHTML($view);
132
133             DB::table('acciones_plataforma')->insert([
134                 'usuario' => Auth::user()->id,
135                 'accion' => 'pdf',
136                 'modulo' => 'metodologia'
137             ]);
138
139             return $pdf->stream('metodologia.pdf');
140         }
141
142     public function export()
143     {
144         $metodologias = Metodologia::all();
145         if ($metodologias->count() <= 0) {
146             Alert::warning('No hay registros');
147             return redirect('/metodologia');
148         } else {
149
150             DB::table('acciones_plataforma')->insert([
151                 'usuario' => Auth::user()->id,
152                 'accion' => 'excel',
153                 'modulo' => 'metodologia'
154             ]);
155
156             return Excel::download(new MetodologiaExport, 'metodologias.xlsx');
157
158         }
159     }

```

### 3.6.3 Vistas

A continuación se muestran las vistas creadas para el modelo metodología.

#### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > configuracion > metodologia > index.blade.php
1  @extends('layouts.app')
2  @section('title')
3  |  <h1 class="titulo"><i class="fas fa-wave-square"></i> Configuración / metodología del programa</h1>
4  @section('message')
5  |  <p>Metodología: Listado metodologías del programa registrados</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10 |  <div class="tile col-md-12 p-3">
11 |  |  <div class="card-body"> ...
12 |  |  </div>
13 |  </div>
14 |  </div>
15 @endsection
```

#### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevas metodologías.

```
resources > views > configuracion > metodologia > create.blade.php
1  @extends('layouts.app')
2  @section('title')
3  |  <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
4  @section('message')
5  |  <p>Formulario de registro nueva metodología</p>
6  @endsection
7  @endsection
8  @section('content')
9  <div class="container-fluid">
10 |  <div class="tile card_config">
11 |  |  <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro metodología</h4>
12 |  |  <form action="/metodologia" method="post"> ...
13 |  |  </form>
14 |  </div>
15 </div>
16 @endsection
```

#### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```
resources > views > configuracion > metodologia > edit.blade.php
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
4  @section('message')
5      <p>Formulario de actualización de la información</p>
6  @endsection
7  @endsection
8  @section('content')
9      <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title">Actualizar información</h4>
12         <form action="/metodologia/{{ $metodologia->id }}" method="POST">...
13         </form>
14     </div>
15     </div>
16     </div>
17     @endsection
```

#### d. show.blade.php

Esta vista permite la visualización de un registro específico.

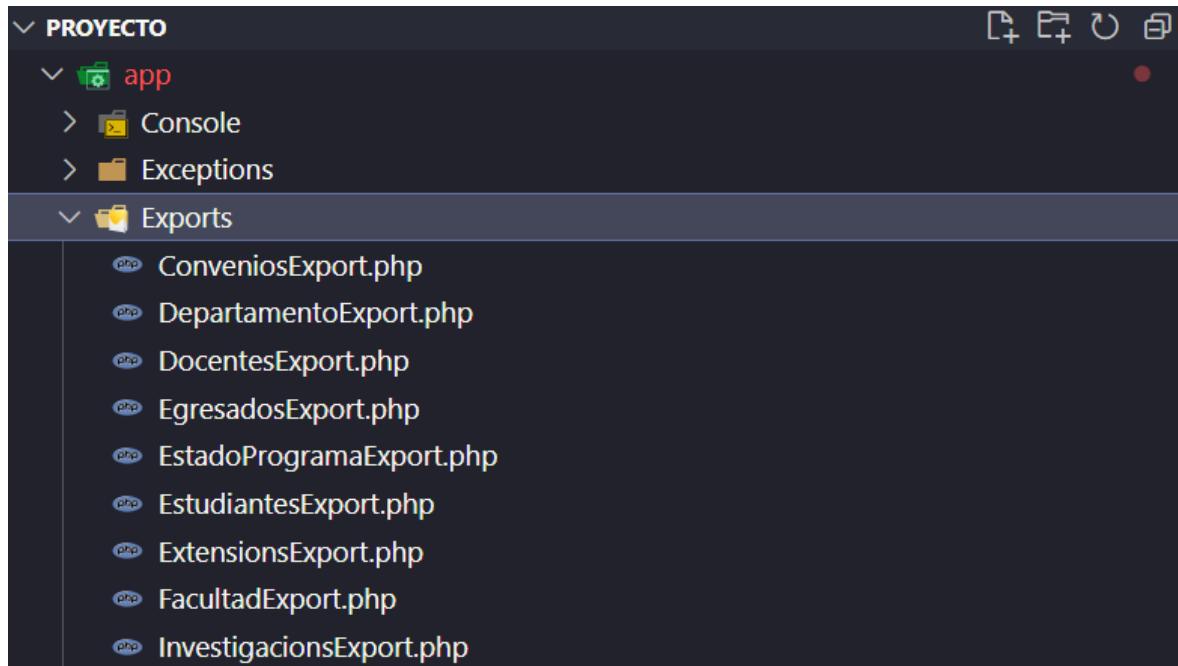
```
resources > views > configuracion > metodología > show.blade.php > ...
1  @extends('layouts.app')
2  @section('title')
3      <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
4  @section('message')
5      <p>Información almacenada en el sistema</p>
6  @endsection
7  @endsection
8  @section('content')
9      <div class="container-fluid">
10         <div class="tile card_config">
11             <h4 class="tile title"><i class="far fa-question-circle"></i> Vista registro</h4>
12             <div class="row mb-3">
13                 <label for="met_nombre">{{ __('Metodología') }}</label>
14                 <div class="col-md-12">
15                     <input id="met_nombre" type="text" class="form-control" @error('met_nombre') is-invalid @enderror
16                     name="met_nombre" value="{{ $metodologia->met_nombre }}" required autocomplete="met_nombre"
17                     autofocus disabled>
18                     @error('met_nombre')
19                         <span class="invalid-feedback" role="alert">
20                             <strong>{{ $message }}</strong>
21                         </span>
22                     @enderror
23                 </div>
24             </div>
25         </div>
26     </div>
27     @endsection
```

## 4 MODULO SISTEMA

Los módulos que se van a mencionar a continuación permiten el registro de diferente información la cual es necesaria para la generación de los reportes en los formatos admitidos .pdf y .xlsx.

ID	Módulo	Métodos creados
1	Programas	
2	Estudiantes	
3	Trabajo de grado	
4	Docentes	
5	Pruebas Saber Pro	
6	Tic's	
7	Extensión e Internacionalización	
8	Prácticas de grado	
9	Laboratorios	
10	Convenios	
11	Redes Académicas	
12	Movilidad	
13	Investigación	
14	Egresados	

Estos módulos aparte de tener cada uno su modelo y controlador, contiene un archivo almacenado en la ruta App/Exports. Estos archivos contienen la lógica para la generación del reporte en formato .xlsx teniendo en cuenta los valores establecidos allí.



## 4.1 Programas

### 4.1.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de Eloquent llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relación de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.

```

app > Models > Programa.php > ...
1 > <?php
2   namespace App\Models;
3
4   use Illuminate\Database\Eloquent\Factories\HasFactory;
5   use Illuminate\Database\Eloquent\Model;
6
7   class Programa extends Model
8   {
9     use HasFactory;
10
11   protected $table = "programas";
12
13   protected $fillable = [
14     'id','pro_estado_programa','pro_departamento','pro_municipio','pro_facultad','pro_nombre',
15     'pro_titulo','pro_codigosnies','pro_resolucion','pro_fecha_ult',
16     'pro_fecha_prox','pro_nivel_formacion','pro_programa_ciclos','pro_metodologia',
17     'pro_duracion','pro_periodo','pro_creditos','pro_asignaturas','pro_norma','pro_director_programa'
18   ];
19
20   public function directorprograma(){
21     return $this->belongsTo(Docente::class, 'pro_director_programa');
22   }
23
24   public function niveles(){
25     return $this->belongsTo(NivelFormacion::class, 'pro_nivel_formacion');
26   }
27
28   public function pruebassaber(){
29     return $this->belongsTo(PruebasSaber::class, 'id');
30   }
31
32   public function laboratorios(){
33     return $this->hasMany(Laboratorio::class, 'id');
34   }
35
36   public function investigacion(){
37     return $this->hasMany(Investigacion::class, 'id');
38   }
39
40   public function estudiantes(){
41     return $this->hasMany(Estudiante::class, 'id');
42   }
43
44 }

```

#### 4.1.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo programas. Su ubicación es en la carpeta App\Http\Controllers.

```

app > Http > Controllers > ProgramaController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Departamento;
6  use App\Models\EstadoPrograma;
7  use App\Models\Facultad;
8  use App\Models\Metodologia;
9  use App\Models\Municipio;
10 use App\Models\Programa;
11 use App\Models\NivelFormacion;
12 use App\Models\Periodo;
13 use App\Models\ProgramaCiclo;
14 use App\Models\TiempoList;
15 use App\Models\Docente;
16 use RealRashid\SweetAlert\Facades\Alert;
17 use Illuminate\Http\Request;
18 use App\Exports\ProgramasExport;
19 use Illuminate\Support\Facades\Auth;
20 use Illuminate\Support\Facades\DB;
21 use Maatwebsite\Excel\Facades\Excel;

23 class ProgramaController extends Controller
24 {
25
26     public function index()
27     {
28         $programas = Programa::all();
29         $facultades = Facultad::all();
30         $niveles = NivelFormacion::all();
31         $metodologias = Metodologia::all();
32         $docentes = Docente::all();
33         $users = DB::table('users')->where('per_tipo_usuario', 2);
34
35         if (Auth::user()->per_tipo_usuario == '3') {
36             return view('programa.index', compact('programas'));
37         } else {
38             if (($niveles->count() > 0) && ($metodologias->count() > 0) && ($facultades->count() > 0)) {
39                 return view('programa.index', compact('programas'));
40             } else {
41                 Alert::warning('Requisitos', 'Es necesario que agregue información en los módulos mencionados en la dashboard');
42                 return redirect('/home');
43             }
44         }
45     }
46
47     public function create()
48     {
49         $estadoprogramas = EstadoPrograma::all();
50         $departamentos = Departamento::all();
51         $municipios = Municipio::all();
52         $facultades = Facultad::all();
53         $niveles = NivelFormacion::all();
54         $metodologias = Metodologia::all();
55         $tiemposlist = TiempoList::all();
56         $programasCiclo = ProgramaCiclo::all();
57         $periodos = Periodo::all();
58         $docentes = Docente::all();
59
60         return view('programa.create')->with('estadoprogramas', $estadoprogramas)
61             ->with('departamentos', $departamentos)
62             ->with('municipios', $municipios)
63             ->with('facultades', $facultades)
64             ->with('niveles', $niveles)
65             ->with('metodologias', $metodologias)
66             ->with('tiemposList', $tiemposList)
67             ->with('programasCiclo', $programasCiclo)
68             ->with('periodos', $periodos)
69             ->with('docentes', $docentes);
70     }
71

```

```

72    public function store(Request $request)
73    {
74        $programas = new Programa();
75        $programas->pro_estado_programa = $request->get('pro_estado_programa');
76        $programas->pro_departamento = $request->get('pro_departamento');
77        $programas->pro_municipio = $request->get('pro_municipio');
78        $programas->pro_facultad = $request->get('pro_facultad');
79        $programas->pro_nombre = $request->get('pro_nombre');
80        $programas->pro_titulo = $request->get('pro_titulo');
81        $programas->pro_codigosnies = $request->get('pro_codigosnies');
82        $programas->pro_resolucion = $request->get('pro_resolucion');
83        $programas->pro_fecha_ult = $request->get('pro_fecha_ult');
84        $programas->pro_fecha_prox = $request->get('pro_fecha_prox');
85        $programas->pro_nivel_formacion = $request->get('pro_nivel_formacion');
86        $programas->pro_programa_ciclos = $request->get('pro_programa_ciclos');
87        $programas->pro_metodologia = $request->get('pro_metodologia');
88        $programas->pro_duraccion = $request->get('pro_duraccion');
89        $programas->pro_periodo = $request->get('pro_periodo');
90        $programas->pro_creditos = $request->get('pro_creditos');
91        $programas->pro_asignaturas = $request->get('pro_asignaturas');
92        $programas->pro_norma = $request->get('pro_norma');
93        $programas->pro_director_programa = $request->get('pro_director_programa');
94
95        $rules = [
96            'pro_nombre' => 'required',
97            'pro_titulo' => 'required',
98            'pro_codigosnies' => 'required',
99            'pro_resolucion' => 'required',
100           'pro_fecha_ult' => 'required',
101           'pro_fecha_prox' => 'required',
102           'pro_creditos' => 'required',
103           'pro_asignaturas' => 'required',
104           'pro_norma' => 'required'
105        ];
106
107        $messages = [
108            'pro_nombre.required' => 'El campo programa es requerido',
109            'pro_titulo.required' => 'El campo titulo es requerido',
110            'pro_codigosnies.required' => 'El campo codigo snies es requerido',
111            'pro_resolucion.required' => 'El campo resolución es requerido',
112            'pro_fecha_ult.required' => 'El campo fecha ultimo registro es requerido',
113            'pro_fecha_prox.required' => 'El campo fecha próximo registro es requerido',
114            'pro_creditos.required' => 'El campo número creditos es requerido',
115            'pro_asignaturas.required' => 'El campo número de asignaturas es requerido',
116            'pro_norma.required' => 'El campo norma de creación programa es requerido'
117        ];
118
119        $this->validate($request, $rules, $messages);
120
121        $valiDi = DB::table('programas')->where('pro_director_programa', $request->get('pro_director_programa'));
122        $sniesDi = DB::table('programas')->where('pro_codigosnies', $request->get('pro_codigosnies'));
123
124        if ($valiDi->count() > 0) {
125            Alert::warning('El docente elegido como director de programa, ya esta asociado a un programa');
126            return back()->withInput();
127        } else if ($sniesDi->count() > 0) {
128            Alert::warning('El programa que esta intentando inscribir, ya se encuentra registrado');
129            return back()->withInput();
130        } else {
131            $programas->save();
132
133            DB::table('acciones_plataforma')->insert([
134                'usuario' => Auth::user()->id,
135                'accion' => 'insertar',
136                'modulo' => 'programas'
137            ]);
138
139            Alert::success('Registro Exitoso');
140
141            return redirect('/programa');
142        }

```

```

145    public function show($id)
146    {
147        $estadoprogramas = EstadoPrograma::all();
148        $departamentos = Departamento::all();
149        $municipios = Municipio::all();
150        $facultades = Facultad::all();
151        $niveles = NivelFormacion::all();
152        $metodologias = Metodologia::all();
153        $tiemposList = TiempoList::all();
154        $programasCiclo = ProgramaCiclo::all();
155        $periodos = Periodo::all();
156        $docentes = Docente::all();
157        $programa = Programa::find($id);
158        return view('programa.show')->with('programa', $programa)
159            ->with('estadoprogramas', $estadoprogramas)
160            ->with('departamentos', $departamentos)
161            ->with('municipios', $municipios)
162            ->with('facultades', $facultades)
163            ->with('niveles', $niveles)
164            ->with('metodologias', $metodologias)
165            ->with('tiemposList', $tiemposList)
166            ->with('programasCiclo', $programasCiclo)
167            ->with('periodos', $periodos)
168            ->with('docentes', $docentes);
169    }

```

```

171    public function edit($id)
172    {
173        $estadoprogramas = EstadoPrograma::all();
174        $departamentos = Departamento::all();
175        $municipios = Municipio::all();
176        $facultades = Facultad::all();
177        $niveles = NivelFormacion::all();
178        $metodologias = Metodologia::all();
179        $tiemposlist = Tiempolist::all();
180        $programasCiclo = ProgramaCiclo::all();
181        $periodos = Periodo::all();
182        $docentes = Docente::all();
183        $programa = Programa::find($id);
184        return view('programa.edit')->with('programa', $programa)
185            ->with('estadoprogramas', $estadoprogramas)
186            ->with('departamentos', $departamentos)
187            ->with('municipios', $municipios)
188            ->with('facultades', $facultades)
189            ->with('niveles', $niveles)
190            ->with('metodologias', $metodologias)
191            ->with('tiemposlist', $tiemposlist)
192            ->with('programasCiclo', $programasCiclo)
193            ->with('periodos', $periodos)
194            ->with('docentes', $docentes);
195    }

```

```

197    public function update(Request $request, $id)
198    {
199        $programa = Programa::find($id);
200        $programa->pro_estado_programa = $request->get('pro_estado_programa');
201        $programa->pro_departamento = $request->get('pro_departamento');
202        $programa->pro_municipio = $request->get('pro_municipio');
203        $programa->pro_facultad = $request->get('pro_facultad');
204        $programa->pro_nombre = $request->get('pro_nombre');
205        $programa->pro_titulo = $request->get('pro_titulo');
206        $programa->pro_codigosnies = $request->get('pro_codigosnies');
207        $programa->pro_resolucion = $request->get('pro_resolucion');
208        $programa->pro_fecha_ult = $request->get('pro_fecha_ult');
209        $programa->pro_fecha_prox = $request->get('pro_fecha_prox');
210        $programa->pro_nivel_formacion = $request->get('pro_nivel_formacion');
211        $programa->pro_programa_ciclos = $request->get('pro_programa_ciclos');
212        $programa->pro_metodologia = $request->get('pro_metodologia');
213        $programa->pro_duracion = $request->get('pro_duracion');
214        $programa->pro_período = $request->get('pro_período');
215        $programa->pro_creditos = $request->get('pro_creditos');
216        $programa->pro_asignaturas = $request->get('pro_asignaturas');
217        $programa->pro_norma = $request->get('pro_norma');
218        $programa->pro_director_programa = $request->get('pro_director_programa');
219
220        $programa->save();
221
222        DB::table('acciones_plataforma')->insert([
223            'usuario' => Auth::user()->id,
224            'accion' => 'editar',
225            'modulo' => 'programas'
226        ]);
227
228        Alert::success('Registro Actualizado');
229
230        return redirect('/programa');
231    }

```

```

233    public function destroy($id)
234    {
235        $programa = Programa::find($id);
236
237        $veriEstu = DB::table('estudiantes')->where('estu_programa', $id);
238        $veriInve = DB::table('investigacion')->where('inv_programa', $id);
239        $veriPrue = DB::table('pruebas_saber')->where('pr_id_programa', $id);
240        $verilab = DB::table('uso_laboratorio')->where('lab_id_programa', $id);
241
242        if ($veriEstu->count() > 0) {
243            Alert::warning('No se pudo eliminar el programa, debido a que esta asociado a otra entidad');
244            return redirect('/programa');
245        } else if ($veriInve->count() > 0) {
246            Alert::warning('No se pudo eliminar el programa, debido a que esta asociado a otra entidad');
247            return redirect('/programa');
248        } else if ($veriPrue->count() > 0) {
249            Alert::warning('No se pudo eliminar el programa, debido a que esta asociado a otra entidad');
250            return redirect('/programa');
251        } else if ($verilab->count() > 0) {
252            Alert::warning('No se pudo eliminar el programa, debido a que esta asociado a otra entidad');
253            return redirect('/programa');
254        } else {
255            $programa->delete();
256
257            DB::table('acciones_plataforma')->insert([
258                'usuario' => Auth::user()->id,
259                'accion' => 'eliminar',
260                'modulo' => 'programas'
261            ]);
262
263            Alert::success('Registro Eliminado');
264
265            return redirect('/programa');
266        }
267    }

```

```

269    public function pdf(Request $request)
270    {
271        $programas = Programa::all();
272        if ($programas->count() <= 0) {
273            Alert::warning('No hay registros');
274            return redirect('/programa');
275        } else {
276            $view = \view('programa.pdf', compact('programas'))->render();
277            $pdf = \App::make('dompdf.wrapper');
278            $pdf->setPaper('A4', 'landscape');
279            $pdf->loadHTML($view);
280
281            DB::table('acciones_plataforma')->insert([
282                'usuario' => Auth::user()->id,
283                'accion' => 'pdf',
284                'modulo' => 'programas'
285            ]);
286
287            return $pdf->stream('programas-reporte.pdf');
288        }
289    }

```

```

291    public function export()
292    {
293        $programas = Programa::all();
294        if ($programas->count() <= 0) {
295            Alert::warning('No hay registros');
296            return redirect('/programa');
297        } else {
298
299            DB::table('acciones_plataforma')->insert([
300                'usuario' => Auth::user()->id,
301                'accion' => 'excel',
302                'modulo' => 'programas'
303            ]);
304
305            return Excel::download(new ProgramasExport, 'users.xlsx');
306        }
307    }
308}

```

### 4.1.3 Vistas

A continuación se muestran las vistas creadas para el modulo programas.

### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

39             <th>Director de programa</th>
40             @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
41                 <th>Acciones</th>
42             @endif
43         </tr>
44     </thead>
45     <tbody>
46         <?php $i = 1; ?>
47         @foreach ($programas as $programa)
48             <tr>
49                 <td>{{ $i++ }}</td>
50                 <td>{{ $programa->pro_nombre }}</td>
51                 <td>{{ $programa->pro_titulo }}</td>
52                 <td>{{ $programa->pro_codigosnies }}</td>
53                 <td>{{ $programa->niveles->niv_nombre }}</td>
54                 <td>{{ $programa->pro_creditos }}</td>
55                 <td>{{ $programa->directorprograma->doc_nombre . ' ' . $programa->directorprograma->doc_apellido }}</td>
56                 <td>
57                     @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
58                         <form action="{{ route('programa.destroy', $programa->id) }}"
59                             method="POST">
60                             <div class="d-flex">
61                                 <a class="btn btn-sm" href="/programa/{{ $programa->id }}">i
62                                     class="fas fa-folder"></i></a>
63                                 <a class="btn btn-outline-info btn-sm"
64                                     href="/programa/{{ $programa->id }}/edit">i
65                                     class="fas fa-sync-alt"></i></a>
66                             </div>
67                         <@csrf
68                         @method('DELETE')
69                         <button type="submit" class="btn btn-danger btn-sm">i
70                             class="fas fa-trash-alt"></i></button>
71                     </form>
72                 </td>
73             </tr>
74         @endforeach
75     </tbody>
76     </table>
77     </div>
78     </div>
79     </div>
80     </div>
81     </div>
82     <@endif
83     <@section
84     <@endif

```

## b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevos programas.

```

resources > views > programa > create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7          @section('message') <p>Diligenciar los campos requeridos, para el debido registro del docente.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100 mx-auto">
12                 <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro programa</h4>
13                 <form action="/programa" method="post">
14                     @csrf
15                     <div class="row mb-3">
16                         <div class="col-md-6">
17                             <label for="pro_estado_programa">{{ __('Estado programa') }}</label>
18                             <div class="row">
19                                 <div class="col-md-9">
20                                     <select class="js-example-placeholder-single form-select" name="pro_estado_programa"
21                                         id="pro_estado_programa">
22                                         <option selected>----- SELECCIONE -----</option>
23                                         @foreach ($estadoprogramas as $estadoprograma)
24                                             <option value="{{ $estadoprograma->id }}>{{ $estadoprograma->est_nombre }}</option>
25                                         @endforeach
26                                     @endforeach
27                                     </select>
28                                 </div>
29                                 <div class="col-md-2">
30                                     <a class="btn btn-outline-primary" href="/estadoprograma/create">Agregar</a>
31                                 </div>
32                             </div>
33                         </div>
34                         <div class="col-md-6">
35                             <label for="pro_departamento">{{ __('Departamento') }}</label>
36                             <div class="row">
37                                 <div class="col-md-9">
38                                     <select class="js-example-placeholder-single form-select" name="pro_departamento"
39                                         id="pro_departamento">
40                                         <option selected>----- SELECCIONE -----</option>
41                                         @foreach ($departamentos as $departamento)
42                                             <option value="{{ $departamento->id }}>{{ $departamento->dep_nombre }}</option>
43                                         @endforeach
44                                     </select>
45                                 </div>
46                                 <div class="col-md-2">
47                                     <a class="btn btn-outline-primary" href="/departamento/create">Agregar</a>
48                                 </div>
49                             </div>
50                         </div>
51                     </div>
52                 </div>

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > programa > edit.blade.php > ...
1 <?php
2     @if (!Auth::check())
3         @include('home')
4     @else
5         @extends('layouts.app')
6         @section('title')
7             <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
8             @section('message') <p>Actualizar información del programa.</p> @endsection
9         @endsection
10        @section('content')
11            <div class="container-fluid">
12                <div class="tile w-100">
13                    <h4 class="tile title">Actualizar información</h4>
14                    <form action="/programa/{{ $programa->id }}" method="post">...
15                    </form>
16                </div>
17            </div>
18            <br>
19        @endsection
20    @endif

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de un programa específico.

```

resources > views > programa > show.blade.php > ...
1 <?php
2     @if (!Auth::check())
3         @include('home')
4     @else
5         @extends('layouts.app')
6         @section('title')
7             <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
8             @section('message') <p>Información almacenada en el sistema</p> @endsection
9         @endsection
10        @section('content')
11            <div class="container-fluid">
12                <div class="tile w-100">...
13                </div>
14            </div>
15            <br>
16        @endsection
17    @endif

```

#### 4.1.4 Export

Este archivo llamado **ExportPrograma** contiene los parámetros establecidos para generar el reporte en formato **.xlsx**.

```

app > Exports > ProgramasExport.php > ProgramasExport > headings
1  <?php
2
3  namespace App\Exports;
4
5  use DB;
6  use Maatwebsite\Excel\Concerns\FromCollection;
7  use Maatwebsite\Excel\Concerns\WithHeadings;
8  use PhpParser\Node\Expr\AssignOp\Concat;
9
10 class ProgramasExport implements FromCollection, WithHeadings
11 {
12     public function headings(): array
13     {
14         return [
15             'Id',
16             'Estado programa',
17             'Departamento',
18             'Municipio',
19             'Facultad',
20             'Nombre',
21             'Titulo',
22             'Codigo SNIES',
23             'Resolución',
24             'Fecha ultimo registro',
25             'Fecha próximo registro',
26             'Nivel formación',
27             'Programa por ciclos (si/no)',
28             'Metodología',
29             'Duración (semestres)',
30             'Periodo',
31             'Número total de creditos',
32             'Norma',
33             'Director de programa'
34         ];
35     }

```

```

36     public function collection()
37     {
38         $programas = DB::table('programas')->select(
39             'programas.id',
40             'est_nombre',
41             'dep_nombre',
42             'mun_nombre',
43             'fac_nombre',
44             'pro_nombre',
45             'pro_titulo',
46             'pro_codigosnies',
47             'pro_resolucion',
48             'pro_fecha_ult',
49             'pro_fecha_prox',
50             'niv_nombre',
51             'prc_nombre',
52             'met_nombre',
53             'pro_duracion',
54             'per_nombre',
55             'pro_creditos',
56             'pro_norma',
57             DB::raw("CONCAT(doc_nombre, ' ', doc_apellido)")
58         )
59         ->join('estado_programas', 'programas.pro_estado_programa', '=', 'estado_programas.id')
60         ->join('departamentos', 'programas.pro_departamento', '=', 'departamentos.id')
61         ->join('municipios', 'programas.pro_municipio', '=', 'municipios.id')
62         ->join('facultades', 'programas.pro_facultad', '=', 'facultades.id')
63         ->join('nivelformacion', 'programas.pro_nivel_formacion', '=', 'nivelformacion.id')
64         ->join('programa_ciclos', 'programas.pro_programa_ciclos', '=', 'programa_ciclos.id')
65         ->join('metodologia', 'programas.pro_metodologia', '=', 'metodologia.id')
66         ->join('periodo', 'programas.pro_periodo', '=', 'periodo.id')
67         ->join('docentes', 'programas.pro_director_programa', '=', 'docentes.id')
68         ->get();
69     }
70 }

```

## 4.2 Estudiantes

### 4.2.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.

```
app > Models > Estudiante.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\hasMany;
8
9  class Estudiante extends Model
10 {
11     use HasFactory;
12
13     protected $table = "estudiantes";
14
15     protected $fillable = [
16         'id','estu_programa','estu_tipo_documento','estu_numero_documento','estu_nombre',
17         'estu_apellido','estu_telefono1','estu_telefono2','estu_direccion','estu_correo',
18         'estu_estrato','estu_departamento','estu_municipio','estu_fecha_nacimiento','estu_ingreso',
19         'estu_ult_periodo','estu_semestre','estu_financiamiento','estu_beca','estu_estado',
20         'estu_matricula','estu_pga','estu_grado','estu_reconocimiento','estu_egresado'
21     ];
22
23     public function tipodocumento(){
24         return $this->belongsTo(TipoDocumento::class, 'estu_tipo_documento');
25     }
26
27     public function programas(){
28         return $this->belongsTo(Programa::class, 'estu_programa');
29     }

```

```

31     public function pruebassaaber(){
32         return $this->hasMany(PruebasSaber::class, 'id');
33     }
34
35     public function practicas(){
36         return $this->hasMany(Practica::class, 'id');
37     }
38
39     public function movilidad(){
40         return $this->hasMany(Movilidad::class, 'id');
41     }
42
43     public function egresado(){
44         return $this->hasMany(Egresado::class, 'id');
45     }
46
47 }

```

#### 4.2.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo estudiantes. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > EstudianteController.php > EstudianteController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Departamento;
6  use App\Models\EstadoPrograma;
7  use App\Models\Estudiante;
8  use App\Models\Municipio;
9  use App\Models\Programa;
10 use App\Models\TiempoList;
11 use App\Models\TipoDocumento;
12 use RealRashid\SweetAlert\Facades\Alert;
13 use Illuminate\Http\Request;
14 use App\Exports\EstudiantesExport;
15 use Illuminate\Support\Facades\Auth;
16 use Illuminate\Support\Facades\DB;
17 use Maatwebsite\Excel\Facades\Excel;
18
19
20 class EstudianteController extends Controller
21 {
22     public function index()
23     {
24         $estudiantes = Estudiante::all();
25         $programas = Programa::all();
26
27         if (Auth::user()->per_tipo_usuario == '3') {
28             return view('estudiante.index', compact('estudiantes'));
29         } else {
30             if ($programas->count() > 0) {
31                 return view('estudiante.index', compact('estudiantes'));
32             } else {
33                 Alert::warning('Requisitos', 'Primero registre un programa académico');
34                 return redirect('/home');
35             }
36         }
37     }
38
39 }

```

```

39 |     public function create()
40 |     {
41 |         $programas = Programa::all();
42 |         $departamentos = Departamento::all();
43 |         $municipios = Municipio::all();
44 |         $tiemposList = TiempoList::all();
45 |         $tipos = TipoDocumento::all();
46 |         $estadoprogramas = EstadoPrograma::all();
47 |         return view('estudiante.create')->with('programas', $programas)
48 |             ->with('departamentos', $departamentos)
49 |             ->with('municipios', $municipios)
50 |             ->with('tiemposList', $tiemposList)
51 |             ->with('estadoprogramas', $estadoprogramas)
52 |             ->with('tipos', $tipos);
53 |     }

```

```

55 |     public function store(Request $request)
56 |     {
57 |
58 |         $rules = [
59 |             'estu_numero_documento' => 'required',
60 |             'estu_nombre' => 'required',
61 |             'estu_apellido' => 'required',
62 |             'estu_telefono1' => 'required',
63 |             'estu_correo' => 'required',
64 |             'estu_estrato' => 'required',
65 |             'estu_fecha_nacimiento' => 'required',
66 |             'estu_ingreso' => 'required',
67 |             'estu_ult_periodo' => 'required',
68 |             'estu_financiamiento' => 'required',
69 |             'estu_beca' => 'required',
70 |             'estu_estado' => 'required',
71 |             'estu_matricula' => 'required'
72 |         ];
73 |
74 |         $messages = [
75 |             'estu_numero_documento.required' => 'El campo número de documento es requerido',
76 |             'estu_nombre.required' => 'El campo nombre(s) es requerido',
77 |             'estu_apellido.required' => 'El campo apellido(s) es requerido',
78 |             'estu_telefono1.required' => 'El campo telefono 1 es requerido',
79 |             'estu_correo.required' => 'El campo correo electronico es requerido',
80 |             'estu_estrato.required' => 'El campo estrato es requerido',
81 |             'estu_fecha_nacimiento.required' => 'El campo fecha de nacimiento es requerido',
82 |             'estu_ingreso.required' => 'El campo año de ingreso es requerido',
83 |             'estu_ult_periodo.required' => 'El campo ultimo periodo es requerido',
84 |             'estu_financiamiento.required' => 'El campo financiamiento es requerido',
85 |             'estu_beca.required' => 'El campo beca es requerido',
86 |             'estu_matricula.required' => 'El campo matricula es requerido'
87 |         ];

```

```

89     $this->validate($request, $rules, $messages);
90
91     if (
92         $request->get('estu_programa') == "" || $request->get('estu_tipo_documento') == "" || $request->get('estu_departamento') == ""
93         || $request->get('estu_municipio') == "" || $request->get('estu_semestre') == "" || $request->get('estu_estado') == ""
94     ) {
95         Alert::warning('El valor seleccionado es incorrecto');
96         return back()->withInput();
97     }
98
99     $estudiantes = new Estudiante();
100    $estudiantes->estu_programa = $request->get('estu_programa');
101    $estudiantes->estu_tipo_documento = $request->get('estu_tipo_documento');
102    $estudiantes->estu_numero_documento = $request->get('estu_numero_documento');
103    $estudiantes->estu_nombre = $request->get('estu_nombre');
104    $estudiantes->estu_apellido = $request->get('estu_apellido');
105    $estudiantes->estu_telefono1 = $request->get('estu_telefono1');
106    $estudiantes->estu_telefono2 = $request->get('estu_telefono2');
107    $estudiantes->estu_direccion = $request->get('estu_direccion');
108    $estudiantes->estu_correo = $request->get('estu_correo');
109    $estudiantes->estu_estrato = $request->get('estu_estrato');
110    $estudiantes->estu_departamento = $request->get('estu_departamento');
111    $estudiantes->estu_municipio = $request->get('estu_municipio');
112    $estudiantes->estu_fecha_nacimiento = $request->get('estu_fecha_nacimiento');
113    $estudiantes->estu_ingreso = $request->get('estu_ingreso');
114    $estudiantes->estu_ult_periodo = $request->get('estu_ult_periodo');
115    $estudiantes->estu_semestre = $request->get('estu_semestre');
116    $estudiantes->estu_financiamiento = $request->get('estu_financiamiento');
117    $estudiantes->estu_beca = $request->get('estu_beca');
118    $estudiantes->estu_estado = $request->get('estu_estado');
119    $estudiantes->estu_matricula = $request->get('estu_matricula');
120    $estudiantes->estu_pga = $request->get('estu_pga');
121    $estudiantes->estu_grado = $request->get('estu_grado');
122    $estudiantes->estu_reconocimiento = $request->get('estu_reconocimiento');
123    $estudiantes->estu_egresado = $request->get('estu_egresado');
124
125
126     $ExistEstu = DB::table('estudiantes')->where('estu_numero_documento', $request->get('estu_numero_documento'));
127
128     if ($ExistEstu->count() > 0) {
129         Alert::warning('El estudiante que intenta registrar ya existe en el sistema');
130         return back()->withInput();
131     } else {
132         $estudiantes->save();
133
134         DB::table('acciones_plataforma')->insert([
135             'usuario' => Auth::user()->id,
136             'accion' => 'insertar',
137             'modulo' => 'estudiantes'
138         ]);
139
140         Alert::success('Registro Exitoso');
141
142         return redirect('/estudiante');
143     }

```

```

145    public function show($id)
146    {
147        $programas = Programa::all();
148        $tipos = TipoDocumento::all();
149        $departamentos = Departamento::all();
150        $municipios = Municipio::all();
151        $tiemposlist = TiempoList::all();
152        $estadoprogramas = EstadoPrograma::all();
153        $estudiante = Estudiante::find($id);
154        return view('estudiante.show')->with('estudiante', $estudiante)
155            ->with('programas', $programas)
156            ->with('tipos', $tipos)
157            ->with('departamentos', $departamentos)
158            ->with('municipios', $municipios)
159            ->with('tiemposList', $tiemposList)
160            ->with('estadoprogramas', $estadoprogramas);
161    }
162
163    public function edit($id)
164    {
165        $programas = Programa::all();
166        $tipos = TipoDocumento::all();
167        $departamentos = Departamento::all();
168        $municipios = Municipio::all();
169        $tiemposlist = TiempoList::all();
170        $estadoprogramas = EstadoPrograma::all();
171        $estudiante = Estudiante::find($id);
172        return view('estudiante.edit')->with('estudiante', $estudiante)
173            ->with('programas', $programas)
174            ->with('tipos', $tipos)
175            ->with('departamentos', $departamentos)
176            ->with('municipios', $municipios)
177            ->with('tiemposList', $tiemposList)
178            ->with('estadoprogramas', $estadoprogramas);
179    }
180
181    public function update(Request $request, $id)
182    {
183        $estudiante = Estudiante::find($id);
184        $estudiante->estu_programa = $request->get('estu_programa');
185        $estudiante->estu_tipo_documento = $request->get('estu_tipo_documento');
186        $estudiante->estu_numero_documento = $request->get('estu_numero_documento');
187        $estudiante->estu_nombre = $request->get('estu_nombre');
188        $estudiante->estu_apellido = $request->get('estu_apellido');
189        $estudiante->estu_telefono1 = $request->get('estu_telefono1');
190        $estudiante->estu_telefono2 = $request->get('estu_telefono2');
191        $estudiante->estu_direccion = $request->get('estu_direccion');
192        $estudiante->estu_correo = $request->get('estu_correo');
193        $estudiante->estu_estrato = $request->get('estu_estrato');
194        $estudiante->estu_departamento = $request->get('estu_departamento');
195        $estudiante->estu_municipio = $request->get('estu_municipio');
196        $estudiante->estu_fecha_nacimiento = $request->get('estu_fecha_nacimiento');
197        $estudiante->estu_ingreso = $request->get('estu_ingreso');
198        $estudiante->estu_ult_periodo = $request->get('estu_ult_periodo');
199        $estudiante->estu_semestre = $request->get('estu_semestre');
200        $estudiante->estu_financiamiento = $request->get('estu_financiamiento');
201        $estudiante->estu_beca = $request->get('estu_beca');
202        $estudiante->estu_estado = $request->get('estu_estado');
203        $estudiante->estu_matricula = $request->get('estu_matricula');
204        $estudiante->estu_pga = $request->get('estu_pga');
205        $estudiante->estu_grado = $request->get('estu_grado');
206        $estudiante->estu_reconocimiento = $request->get('estu_reconocimiento');
207        $estudiante->estu_egresado = $request->get('estu_egresado');
208
209        $estudiante->save();

```

```

222     public function destroy($id)
223     {
224         $estudiante = Estudiante::find($id);
225
226         $veriMovi = DB::table('movilidad')->where('mov_id_estudiante', $id);
227         $veriPru = DB::table('pruebas_saber')->where('pr_id_estudiante', $id);
228         $veriPra = DB::table('practicas')->where('pra_id_estudiante', $id);
229
230         if ($veriMovi->count() > 0) {
231             Alert::warning('No se pudo eliminar el estudiante, debido a que esta asociado a otra entidad');
232             return redirect('/estudiante');
233         } else if ($veriPru->count() > 0) {
234             Alert::warning('No se pudo eliminar el estudiante, debido a que esta asociado a otra entidad');
235             return redirect('/estudiante');
236         } else if ($veriPra->count() > 0) {
237             Alert::warning('No se pudo eliminar el estudiante, debido a que esta asociado a otra entidad');
238             return redirect('/estudiante');
239         } else {
240             $estudiante->delete();
241
242             DB::table('acciones_plataforma')->insert([
243                 'usuario' => Auth::user()->id,
244                 'accion' => 'eliminar',
245                 'modulo' => 'estudiantes'
246             ]);
247
248             Alert::success('Registro Eliminado');
249
250             return redirect('/estudiante');
251         }
252     }

```

```

254     public function pdf()
255     {
256         $estudiantes = Estudiante::all();
257         if ($estudiantes->count() <= 0) {
258             Alert::warning('No hay registros');
259             return redirect('/estudiante');
260         } else {
261             $view = \view('estudiante.pdf', compact('estudiantes'))->render();
262             $pdf = \App::make('dompdf.wrapper');
263             $pdf->setPaper('A4', 'landscape');
264             $pdf->loadHTML($view);
265
266             DB::table('acciones_plataforma')->insert([
267                 'usuario' => Auth::user()->id,
268                 'accion' => 'pdf',
269                 'modulo' => 'estudiantes'
270             ]);
271
272             return $pdf->stream('estudiantes-reporte.pdf');
273         }
274     }

```

```

276     public function export()
277     {
278         $estudiantes = Estudiante::all();
279         if ($estudiantes->count() <= 0) {
280             Alert::warning('No hay registros');
281             return redirect('/estudiante');
282         } else {
283
284             DB::table('acciones_plataforma')->insert([
285                 'usuario' => Auth::user()->id,
286                 'accion' => 'excel',
287                 'modulo' => 'estudiantes'
288             ]);
289
290             return Excel::download(new EstudiantesExport, 'estudiantes.xlsx');
291         }
292     }
293 }

```

#### 4.2.3 Vistas

A continuación se muestran las vistas creadas para el modulo estudiantes.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > estudiante > index.blade.php > ...
1  @if (Auth::check())
2    @include('home')
3  @else
4    @extends('layouts.app')
5    @section('title')
6      <h1 class="titulo"><i class="fas fa-users"></i> Módulo Estudiantes</h1>
7    @section('message')
8      <p>Lista de registro estudiantes</p>
9    @endsection
10   @endsection
11   @section('content')
12     <div class="container-fluid">
13       <div class="tile col-md-12 p-3">
14         <div class="card-body">
15           <div class="row">
16             <div class="col-md-7">
17               <h3>Lista de registros</h3>
18               <a class="btn btn-outline-danger btn-radius" href="{{ url('estudiante/pdf') }}"
19                 title="Generar reporte pdf" target="_blank"><i class="fas fa-file-pdf"></i></a>
20               <a class="btn btn-outline-primary btn-radius" href="{{ url('estudiante/export') }}"
21                 title="Generar reporte excel" target="_blank"><i class="fas fa-file-excel"></i></a>
22             </div>
23             <div class="col-md-5 d-flex justify-content-end align-items-center">
24               @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
25                 <a class="btn btn-outline-success" href="{{ url('estudiante/create') }}"><i
26                   class="fas fa-plus-circle"></i></a>
27                 Nuevo</a>
28               @endif
29             </div>
30           </div>
31           <br>
32           <div class="table-responsive">
```

```

33         <table class="display mt-4 p-3" id="tables">
34             <thead>
35                 <tr>
36                     <th>Nº</th>
37                     <th>Tipo Documento</th>
38                     <th>Número Documento</th>
39                     <th>Nombre(s)</th>
40                     <th>Apellido (s)</th>
41                     <th>Correo electronico</th>
42                     <th>Semestre</th>
43                     @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
44                         <th>Acciones</th>
45                     @endif
46                 </tr>
47             </thead>
48             <tbody>
49                 <?php $i = 1; ?>
50                 @foreach ($estudiantes as $estudiante)
51                     <tr>
52                         <td>{{ $i++ }}</td>
53                         <td>{{ $estudiante->tipodocumento->nombre }}</td>
54                         <td>{{ $estudiante->estu_numero_documento }}</td>
55                         <td>{{ $estudiante->estu_nombre }}</td>
56                         <td>{{ $estudiante->estu_apellido }}</td>
57                         <td>{{ $estudiante->estu_correo }}</td>
58                         <td>{{ $estudiante->estu_semestre }}</td>
59                         @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
60                             <td>
61                                 <form action="{{ route('estudiante.destroy', $estudiante->id) }}" method="POST">
62                                     <div class="d-flex">
63                                         <a class="btn btn-sm"
64                                             href="/estudiante/{{ $estudiante->id }}"
65                                             class="fas fa-folder"></a>
66                                         <a class="btn btn-outline-info btn-sm" " 

```

## b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevos estudiantes.

```

resources > views > estudiante > create.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7         @section('message') <p>Diligenciar los campos requeridos, para el debido registro del estudiante.</p> @endsection
8     @endsection
9     @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title"><i class="fas fa-plus-square"></i> Registro estudiante</h4>
13                 <form action="/estudiante" method="post">
14                     @csrf
15                     <div class="row mb-3">
16                         <div class="col-md-6">
17                             <div class="row">
18                                 <label for="estu_programa">{{ __('Programa *') }}</label>
19                                 <div class="col-md-9">
20                                     <select class="js-example-placeholder-single form-select" name="estu_programa"
21                                         id="estu_programa">
22                                         <option value="">---- SELECCIONE ----</option>
23                                         @foreach ($programas as $programa)
24                                             <option value="{{ $programa->id }}>{{ $programa->pro_nombre }}</option>
25                                         @endforeach
26                                     </select>
27                                 </div>
28                             <div class="col-md-2">
29                                 <a class="btn btn-outline-primary" href="/programa/create">Agregar</a>
30                             </div>
31                         </div>
32                     </div>

```

```

33     <div class="col-md-6">
34         <label for="estu_tipo_documento">{{ __('Tipo Documento *') }}</label>
35         <select class="js-example-placeholder-single form-select" name="estu_tipo_documento"
36             id="estu_tipo_documento">
37             <option value="">---- SELECCIONE ----</option>
38             @foreach ($tipos as $tipo)
39                 <option value="{{ $tipo->id }}>{{ $tipo->nombre }}</option>
40             @endforeach
41         </select>
42     </div>
43 
```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > estudiante > edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del estudiante.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13             <form action="/estudiante/{{ $estudiante->id }}" method="post">...
393             </form>
394         </div>
395     <br>
396     @endsection
397     @endif
398 
```

### d. show.blade.php

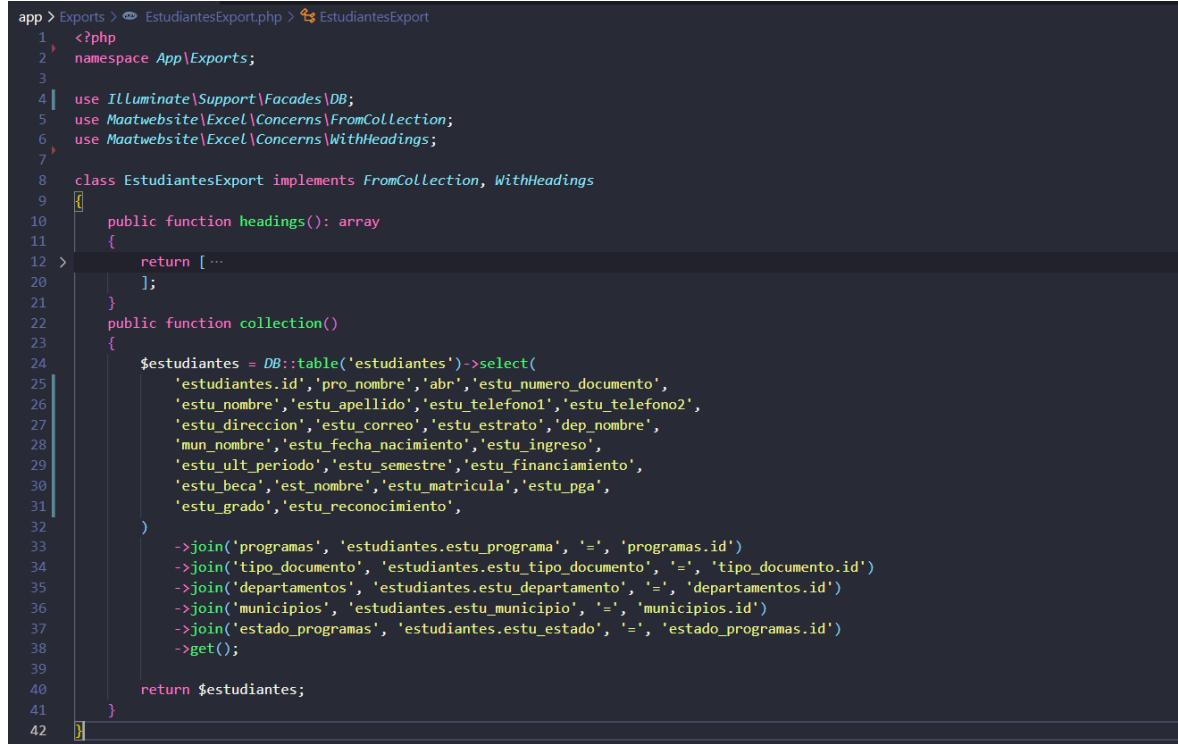
Esta vista permite la visualización de los datos registrados, de un estudiante específico.

```

resources > views > estudiante > show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile">...
320             </div>
321         </div>
322         <br>
323     @endsection
324     @endif
325 
```

#### 4.2.4 Export

Este archivo llamado ExportPrograma contiene los parámetros establecidos para generar el reporte en formato .xlsx.



```
app > Exports > EstudiantesExport.php > EstudiantesExport
1 <?php
2 namespace App\Exports;
3
4 use Illuminate\Support\Facades\DB;
5 use Maatwebsite\Excel\Concerns\FromCollection;
6 use Maatwebsite\Excel\Concerns\WithHeadings;
7
8 class EstudiantesExport implements FromCollection, WithHeadings
9 {
10     public function headings(): array
11     {
12         return [...];
13     }
14     public function collection()
15     {
16         $estudiantes = DB::table('estudiantes')->select(
17             'estudiantes.id', 'pro_nombre', 'abr', 'estu_numero_documento',
18             'estu_nombre', 'estu_apellido', 'estu_telefono1', 'estu_telefono2',
19             'estu_direccion', 'estu_correo', 'estu_estrato', 'dep_nombre',
20             'mun_nombre', 'estu_fecha_nacimiento', 'estu_ingreso',
21             'estu_ult_periodo', 'estu_semestre', 'estu_financiamiento',
22             'estu_beca', 'est_nombre', 'estu_matricula', 'estu_pga',
23             'estu_grado', 'estu_reconocimiento',
24         )
25         ->join('programas', 'estudiantes.estu_programa', '=', 'programas.id')
26         ->join('tipo_documento', 'estudiantes.estu_tipo_documento', '=', 'tipo_documento.id')
27         ->join('departamentos', 'estudiantes.estu_departamento', '=', 'departamentos.id')
28         ->join('municipios', 'estudiantes.estu_municipio', '=', 'municipios.id')
29         ->join('estado_programas', 'estudiantes.estu_estado', '=', 'estado_programas.id')
30         ->get();
31
32         return $estudiantes;
33     }
34 }
35
36
37
38
39
40
41
42 ]|
```

### 4.3 Trabajo de grado

#### 4.3.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```
app > Models > Trabajo.php > Trabajo > $fillable
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Trabajo extends Model
9  {
10     use HasFactory;
11
12     protected $table = "trabajo_grado";
13
14     protected $fillable = [
15         'id',
16         'tra_fecha_ejecucion',
17         'tra_nombre',
18         'tra_id_estudiante',
19         'tra_director',
20         'tra_jurado',
21         'tra_fecha_inicio',
22         'tra_fecha_fin',
23         'tra_documento'
24     ];
25
26 }
```

#### 4.3.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo trabajo de grado. Su ubicación es en la carpeta App\Http\Controllers.

```
app > Http > Controllers > TrabajoController.php > ...
1  <?php
2  namespace App\Http\Controllers;
3
4  use App\Exports\TrabajosExport;
5  use App\Models\Trabajo;
6  use App\Models\Estudiante;
7  use App\Models\Docente;
8  use RealRashid\SweetAlert\Facades\Alert;
9  use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\DB;
12 use Maatwebsite\Excel\Facades\Excel;
```

```

14 class TrabajoController extends Controller
15 {
16     public function index()
17     {
18         $trabajos = Trabajo::all();
19         $estudiantes = Estudiante::all();
20         $docentes = Docente::all();
21
22         if (Auth::user()->per_tipo_usuario == '3') {
23             return view('trabajo.index')->with('trabajos', $trabajos);
24         } else {
25             if ($estudiantes->count() <= 0 && $docentes->count() <= 0) {
26                 Alert::warning('Requisitos', 'Registre estudiantes y docentes');
27                 return redirect('/home');
28             } else if ($estudiantes->count() <= 0) {
29                 Alert::warning('Requisitos', 'Registre estudiantes');
30                 return redirect('/home');
31             } else if ($docentes->count() <= 0) {
32                 Alert::warning('Requisitos', 'Registre docentes');
33                 return redirect('/home');
34             } else {
35                 return view('trabajo.index')->with('trabajos', $trabajos);
36             }
37         }
38     }
39
40     public function create()
41     {
42         $estudiantes = Estudiante::all();
43         $docentes = Docente::all();
44
45         return view('trabajo.create')->with('estudiantes', $estudiantes)
46             ->with('docentes', $docentes);
47     }
48
49     public function store(Request $request)
50     {
51
52         $rules = [
53             'tra_fecha_ejecucion' => 'required',
54             'tra_nombre' => 'required',
55             'tra_director' => 'required',
56             'tra_fecha_inicio' => 'required',
57             'tra_fecha_fin' => 'required'
58         ];
59
60         $message = [
61             'tra_fecha_ejecucion.required' => 'El campo año de ejecución es requerido',
62             'tra_nombre.required' => 'El campo título del proyecto es requerido',
63             'tra_director.required' => 'El campo director es requerido',
64             'tra_fecha_inicio.required' => 'El campo fecha inicio proyecto es requerido',
65             'tra_fecha_fin.required' => 'El campo fecha fin proyecto es requerido'
66         ];
67
68         $this->validate($request, $rules, $message);

```

```

70 ~    if ($request->get('tra_id_estudiante') == "" || $request->get('tra_jurado') == "") {
71     Alert::warning('El valor seleccionado es incorrecto');
72     return back()->withInput();
73 }
74
75 ~    if ($request->file('tra_documento') == "") {
76     Alert::warning('Debe adjuntar documento en formato .PDF');
77     return back()->withInput();
78 }
79
80     $trabajos = new Trabajo();
81     $trabajos->tra_fecha_ejecucion = $request->get('tra_fecha_ejecucion');
82     $trabajos->tra_nombre = $request->get('tra_nombre');
83     $trabajos->tra_id_estudiante = implode(';', $request->get('tra_id_estudiante'));
84     $trabajos->tra_director = $request->get('tra_director');
85     $trabajos->tra_jurado = implode(';', $request->get('tra_jurado'));
86     $trabajos->tra_fecha_inicio = $request->get('tra_fecha_inicio');
87     $trabajos->tra_fecha_fin = $request->get('tra_fecha_fin');
88
89
90 ~    if ($request->file('tra_documento')) {
91     $file = $request->file('tra_documento');
92     $name = 'pdf_' . $trabajos->tra_nombre . '.' . $file->extension();
93
94     $ruta = public_path('pdf/' . $name);
95
96     $trabajos->tra_documento = $name;

```

```

98         if ($file->extension() == 'pdf') {
99             copy($file, $ruta);
100        } else {
101            Alert::warning('El formato del documento no es .PDF');
102            return back()->withInput();
103        }
104    }
105
106    $trabajos->save();
107
108    DB::table('acciones_plataforma')->insert([
109        'usuario' => Auth::user()->id,
110        'accion' => 'insertar',
111        'modulo' => 'trabajo_grado'
112    ]);
113
114    Alert::success('Registro Exitoso');
115
116    return redirect('/trabajo');
117 }
118
119 public function show($id)
{
    $estudiantes = Estudiante::all();
    $docentes = Docente::all();
    $trabajo = Trabajo::find($id);
    return view('trabajo.show')->with('estudiantes', $estudiantes)
        ->with('docentes', $docentes)
        ->with('trabajo', $trabajo);
}

```

```

129 public function edit($id)
{
    $estudiantes = Estudiante::all();
    $docentes = Docente::all();
    $trabajo = Trabajo::find($id);
    return view('trabajo.edit')->with('estudiantes', $estudiantes)
        ->with('docentes', $docentes)
        ->with('trabajo', $trabajo);
}

```

```

139     public function update(Request $request, $id)
140     {
141         $trabajo = Trabajo::find($id);
142         $trabajo->tra_fecha_ejecucion = $request->get('tra_fecha_ejecucion');
143         $trabajo->tra_nombre = $request->get('tra_nombre');
144         $trabajo->tra_id_estudiante = implode(';', $request->get('tra_id_estudiante'));
145         $trabajo->tra_director = $request->get('tra_director');
146         $trabajo->tra_jurado = implode(';', $request->get('tra_jurado'));
147         $trabajo->tra_fecha_inicio = $request->get('tra_fecha_inicio');
148         $trabajo->tra_fecha_fin = $request->get('tra_fecha_fin');
149
150         if ($trabajo->tra_documento = $request->file('tra_documento')) {
151             $file = $request->file('tra_documento');
152             $name = 'pdf_' . $trabajo->tra_nombre . '.' . $file->extension();
153
154             $ruta = public_path('pdf/' . $name);
155
156             if ($file->extension() == 'pdf') {
157                 copy($file, $ruta);
158             } else {
159                 Alert::warning('El formato del documento no es .PDF');
160                 return back()->withInput();
161             }
162         }
163
164         $trabajo->save();
165
166         DB::table('acciones_plataforma')->insert([
167             'usuario' => Auth::user()->id,
168             'accion' => 'editar',
169             'modulo' => 'trabajo_grado'
170         ]);
171
172         Alert::success('Registro Actualizado');
173
174         return redirect('/trabajo');
175     }
176
177     public function destroy($id)
178     {
179         $trabajo = Trabajo::find($id);
180         $trabajo->delete();
181
182         DB::table('acciones_plataforma')->insert([
183             'usuario' => Auth::user()->id,
184             'accion' => 'eliminar',
185             'modulo' => 'trabajo_grado'
186         ]);
187
188         Alert::success('Registro Eliminado');
189         return redirect('/trabajo');
190     }
191
192     public function pdf(Request $request)
193     {
194         $trabajos = Trabajo::all();
195         if ($trabajos->count() <= 0) {
196             Alert::warning('No hay registros');
197             return redirect('/trabajo');
198         } else {
199             $view = \view('trabajo.pdf', compact('trabajos'))->render();
200             $pdf = \App::make('dompdf.wrapper');
201             $pdf->loadHTML($view);
202
203             DB::table('acciones_plataforma')->insert([
204                 'usuario' => Auth::user()->id,
205                 'accion' => 'pdf',
206                 'modulo' => 'trabajo_grado'
207             ]);
208
209             return $pdf->stream('trabajos-grado-report.pdf');
210         }
211     }

```

```

213     public function export()
214     {
215         $trabajos = Trabajo::all();
216         if ($trabajos->count() <= 0) {
217             Alert::warning('No hay registros');
218             return redirect('/trabajo');
219         } else {
220
221             DB::table('acciones_plataforma')->insert([
222                 'usuario' => Auth::user()->id,
223                 'accion' => 'excel',
224                 'modulo' => 'trabajo_grado'
225             ]);
226
227             return Excel::download(new TrabajosExport, 'trabajos_grado.xlsx');
228         }
229     }
230 }

```

#### 4.3.3 Vistas

A continuación se muestran las vistas creadas para el modulo trabajo de grado.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > trabajo > index.blade.php
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-laptop-house"></i> Módulo trabajo de grado</h1>
7          @section('message')<p>Lista de registro trabajo de grado</p>@endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile col-md-12 p-3">
12                 <div class="card-body">
13                     <div class="row">
14                         <div class="col-md-7">
15                             <h3>Lista de registros</h3>
16                             <a class="btn btn-outline-danger btn-radius" href="{{ url('trabajo/pdf') }}"
17                                 title="Generar reporte pdf" target="_blank"><i class="fas fa-file-pdf"></i></a>
18                             <a class="btn btn-outline-primary btn-radius" href="{{ url('trabajo/export') }}"
19                                 title="Generar reporte excel" target="_blank"><i class="fas fa-file-excel"></i>
20                         </div>
21                         <div class="col-md-5 d-flex justify-content-end align-items-center">
22                             @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
23                                 <a class="btn btn-outline-success " href="{{ url('trabajo/create') }}"><i
24                                     class="fas fa-plus-circle"></i>
25                                     Nuevo</a>
26                             @endif
27                         </div>
28                     </div>
29                     <br>

```

## b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevo trabajo de grado.

```
resources > views > trabajo >  create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7      @section('message') <p>Diligenciar los campos requeridos, para el debido registro del trabajo de grado.</p>
8      @endsection
9  @endsection
10 @section('content')
11     <div class="container-fluid">
12         <div class="tile w-100">
13             <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro trabajo de grado</h4>
14         <form action="/trabajo" method="post" enctype="multipart/form-data"> ...
142        </form>
143     </div>
144     </div>
145 @endsection
146 @endif
```

## c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```
resources > views > trabajo >  edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del trabajo de grado.</p> @endsection
8      @endsection
9  @section('content')
10     <div class="container-fluid">
11         <div class="tile w-100">
12             <h4 class="tile title">Actualizar información</h4>
13             <div class="card-body">
14                 <form action="/trabajo/{{ $trabajo->id }}" method="post" enctype="multipart/form-data"> ...
142                </form>
143            </div>
144            </div>
145        @endsection
146    @endif
```

## d. show.blade.php

Esta vista permite la visualización de los datos registrados, de un trabajo de grado específico.

```

resources > views > trabajo > show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">...
116        </div>
117    @endsection
118 @endif
119

```

#### 4.3.4 Export

Este archivo llamado ExportTrabajos contiene los parámetros establecidos para generar el reporte en formato .xlsx.

```

app > Exports > TrabajosExport.php > ...
1  <?php
2  namespace App\Exports;
3
4  use DB;
5  use Maatwebsite\Excel\Concerns\FromCollection;
6  use Maatwebsite\Excel\Concerns\WithHeadings;
7
8  class TrabajosExport implements FromCollection, WithHeadings
9  {
10     public function headings(): array
11     {
12         return [
13             'Id',
14             'Fecha ejecución',
15             'Titulo del proyecto',
16             'Estudiante (s)',
17             'Director',
18             'Jurado (s)',
19             'Fecha inicio',
20             'Fecha fin',
21             'Url documento',
22         ];
23     }
24     public function collection()
25     {
26         $trabajos = DB::table('trabajo_grado')->select(
27             'trabajo_grado.id',
28             'tra_fecha_ejecucion',
29             'tra_nombre',
30             'tra_id_estudiante',
31             'tra_director',
32             'tra_jurado',
33             'tra_fecha_inicio',
34             'tra_fecha_fin',
35             'tra_documento',
36         )->get();
37         return $trabajos;
38     }
39 }

```

## 4.4 Docentes

### 4.4.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.

```
app > Models > Docente.php > Docente > $fillable
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Docente extends Model
9  {
10     use HasFactory;
11
12     protected $table = "docentes";
13
14     protected $fillable = [
15         'id','doc_tipo_documento','doc_numero_documento','doc_nombre','doc_apellido',
16         'doc_telefono1','doc_telefono2','doc_correo_personal','doc_correo_institucional','doc_departamento',
17         'doc_ciudad','doc_direccion','doc_estudios','doc_estudio_complementarios','doc_fecha_inicio_contra',
18         'doc_fecha_fin_contra','doc_experiencia','doc_categoria','doc_becas','doc_reconocimiento',
19         'doc_organos_colegiales','doc_asignaturas','doc_evaluacion_docente'
20     ];
21
22     public function programas(){
23         return $this->hasMany(Programa::class, 'id');
24     }
25
26     public function tipodocumento(){
27         return $this->belongsTo(TipoDocumento::class, 'doc_tipo_documento');
28     }
```

```

30     public function softwares(){
31         return $this->hasMany(Software::class, 'id');
32     }
33
34     public function laboratorios(){
35         return $this->hasMany(Laboratorio::class, 'id');
36     }
37
38     public function movilidad(){
39         return $this->hasMany(Movilidad::class, 'id');
40     }
41
42     public function investigacion(){
43         return $this->hasMany(Investigacion::class, 'id');
44     }
45
46 }

```

#### 4.4.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo docentes. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > DocenteController.php > DocenteController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Exports\DocentesExport;
6  use App\Models\Docente;
7  use App\Models\tipoDocumento;
8  use RealRashid\SweetAlert\Facades\Alert;
9  use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\DB;
12 use Maatwebsite\Excel\Facades\Excel;
13
14 class DocenteController extends Controller
15 {
16     public function index()
17     {
18         $docentes = Docente::all();
19         return view('docente.index')->with('docentes', $docentes);
20     }
21
22     public function create()
23     {
24         $tipos = TipoDocumento::all();
25         return view('docente.create')->with('tipos', $tipos);
26     }

```

```

28     public function store(Request $request)
29 {
30
31     $rules = [
32         'doc_numero_documento' => 'required',
33         'doc_nombre' => 'required',
34         'doc_apellido' => 'required',
35         'doc_telefono1' => 'required',
36         'doc_correo_personal' => 'required',
37         'doc_correo_institucional' => 'required',
38         'doc_departamento' => 'required',
39         'doc_ciudad' => 'required',
40         'doc_direccion' => 'required',
41         'doc_estudios' => 'required',
42         'doc_fecha_inicio_contra' => 'required',
43         'doc_fecha_fin_contra' => 'required'
44     ];
45
46     $message = [
47         'doc_numero_documento.required' => 'El campo número de documento es requerido',
48         'doc_nombre.required' => 'El campo nombre (s) es requerido',
49         'doc_apellido.required' => 'El campo apellido (s) es requerido',
50         'doc_telefono1.required' => 'El campo telefono es requerido',
51         'doc_correo_personal.required' => 'El campo correo personal es requerido',
52         'doc_correo_institucional.required' => 'El campo correo institucional es requerido',
53         'doc_departamento.required' => 'El campo departamento es requerido',
54         'doc_ciudad.required' => 'El campo ciudad es requerido',
55         'doc_direccion.required' => 'El campo dirección es requerido',
56         'doc_estudios.required' => 'El campo nivel de estudio es requerido',
57         'doc_fecha_inicio_contra.required' => 'El campo fecha inicio contrato es requerido',
58         'doc_fecha_fin_contra.required' => 'El campo fecha fin contrato es requerido'
59     ];
60
61     $this->validate($request, $rules, $message);
62
63     if($request->get('doc_tipo_documento') == "" || $request->get('doc_categoria') == ""){
64         Alert::warning('El valor seleccionado es incorrecto');
65         return back()->withInput();
66     }
67
68     $docentes = new Docente();
69     $docentes->doc_tipo_documento = $request->get('doc_tipo_documento');
70     $docentes->doc_numero_documento = $request->get('doc_numero_documento');
71     $docentes->doc_nombre = $request->get('doc_nombre');
72     $docentes->doc_apellido = $request->get('doc_apellido');
73     $docentes->doc_telefono1 = $request->get('doc_telefono1');
74     $docentes->doc_telefono2 = $request->get('doc_telefono2');
75     $docentes->doc_correo_personal = $request->get('doc_correo_personal');
76     $docentes->doc_correo_institucional = $request->get('doc_correo_institucional');
77     $docentes->doc_departamento = $request->get('doc_departamento');
78     $docentes->doc_ciudad = $request->get('doc_ciudad');
79     $docentes->doc_direccion = $request->get('doc_direccion');
80     $docentes->doc_estudios = $request->get('doc_estudios');
81     $docentes->doc_fecha_inicio_contra = $request->get('doc_fecha_inicio_contra');
82     $docentes->doc_fecha_fin_contra = $request->get('doc_fecha_fin_contra');
83     $docentes->doc_categoria = $request->get('doc_categoria');
84     $docentes->doc_reconocimiento = $request->get('doc_reconocimiento');
85
86
87     $ExistDoc = DB::table('docentes')->where('doc_numero_documento', $request->get('doc_numero_documento'));
88     $ExistCoI = DB::table('docentes')->where('doc_correo_institucional', $request->get('doc_correo_institucional'));
89     if ($ExistDoc->count() > 0) {
90         Alert::warning('El número de documento que esta intentado registrar, ya se encuentra en el sistema');
91         return back()->withInput();
92     } else if ($ExistCoI->count() > 0) {
93         Alert::warning('El correo ya se encuentra en el sistema');
94         return back()->withInput();
95     } else {
96         $docentes->save();

```

```

98     DB::table('acciones_plataforma')->insert([
99         'usuario' => Auth::user()->id,
100        'accion' => 'insertar',
101        'modulo' => 'docentes'
102    ]);
103
104    Alert::success('Registro Exitoso');
105
106    return redirect('/docente');
107 }
108 }
109
110 public function show($id)
111 {
112     $tipos = TipoDocumento::all();
113     $docente = Docente::find($id);
114     return view('docente.show')->with('tipos', $tipos)
115         ->with('docente', $docente);
116 }
117
118 public function edit($id)
119 {
120     $tipos = TipoDocumento::all();
121     $docente = Docente::find($id);
122     return view('docente.edit')->with('tipos', $tipos)
123         ->with('docente', $docente);
124 }

125
126 public function update(Request $request, $id)
127 {
128     $docente = Docente::find($id);
129     $docente->doc_tipo_documento = $request->get('doc_tipo_documento');
130     $docente->doc_numero_documento = $request->get('doc_numero_documento');
131     $docente->doc_nombre = $request->get('doc_nombre');
132     $docente->doc_apellido = $request->get('doc_apellido');
133     $docente->doc_telefono1 = $request->get('doc_telefono1');
134     $docente->doc_telefono2 = $request->get('doc_telefono2');
135     $docente->doc_correo_personal = $request->get('doc_correo_personal');
136     $docente->doc_correo_institucional = $request->get('doc_correo_institucional');
137     $docente->doc_departamento = $request->get('doc_departamento');
138     $docente->doc_ciudad = $request->get('doc_ciudad');
139     $docente->doc_direccion = $request->get('doc_direccion');
140     $docente->doc_estudios = $request->get('doc_estudios');
141     $docente->doc_fecha_inicio_contra = $request->get('doc_fecha_inicio_contra');
142     $docente->doc_fecha_fin_contra = $request->get('doc_fecha_fin_contra');
143     $docente->doc_categoria = $request->get('doc_categoria');
144     $docente->doc_reconocimiento = $request->get('doc_reconocimiento');

145     $docente->save();

146     DB::table('acciones_plataforma')->insert([
147         'usuario' => Auth::user()->id,
148         'accion' => 'editar',
149         'modulo' => 'docentes'
150     ]);

151     Alert::success('Registro Actualizado');
152
153
154     return redirect('/docente');
155
156 }
157

```

```

159     public function destroy($id)
160     {
161         $docente = Docente::find($id);
162
163         $veriMovi = DB::table('movilidad')->where('mov_id_docente', $id);
164         $veriInv = DB::table('investigacion')->where('inv_director', $id);
165         $veriPra = DB::table('programas')->where('pro_director_programa', $id);
166         $veriLab = DB::table('uso_laboratorio')->where('lab_id_docente', $id);
167
168         if ($veriMovi->count() > 0) {
169             Alert::warning('No se pudo eliminar el docente, debido a que esta asociado a otra entidad');
170             return redirect('/docente');
171         } else if ($veriInv->count() > 0) {
172             Alert::warning('No se pudo eliminar el docente, debido a que esta asociado a otra entidad');
173             return redirect('/docente');
174         } else if ($veriPra->count() > 0) {
175             Alert::warning('No se pudo eliminar el docente, debido a que esta asociado a otra entidad');
176             return redirect('/docente');
177         } else if ($veriLab->count() > 0) {
178             Alert::warning('No se pudo eliminar el docente, debido a que esta asociado a otra entidad');
179             return redirect('/docente');
180         } else {
181
182             $docente->delete();
183
184             DB::table('acciones_plataforma')->insert([
185                 'usuario' => Auth::user()->id,
186                 'accion' => 'eliminar',
187                 'modulo' => 'docentes'
188             ]);
189
190             Alert::success('Registro Eliminado');
191
192             return redirect('/docente');
193         }
194     }

```

```

196     public function pdf(){
197         $docentes = Docente::all();
198         if ($docentes->count() <= 0) {
199             Alert::warning('No hay registros');
200             return back()->withInput();
201         } else {
202             $view = \view('docente.pdf', compact('docentes'))->render();
203             $pdf = \App::make('dompdf.wrapper');
204             $pdf->setPaper('A4', 'landscape');
205             $pdf->loadHTML($view);
206
207             DB::table('acciones_plataforma')->insert([
208                 'usuario' => Auth::user()->id,
209                 'accion' => 'pdf',
210                 'modulo' => 'docentes'
211             ]);
212
213             return $pdf->stream('docentes-reporte.pdf');
214         }
215     }
216
217     public function export(){
218         $docentes = Docente::all();
219         if($docentes->count()<=0){
220             Alert::warning('No hay registros');
221             return back()->withInput();
222         }else{
223
224             DB::table('acciones_plataforma')->insert([
225                 'usuario' => Auth::user()->id,
226                 'accion' => 'excel',
227                 'modulo' => 'docentes'
228             ]);
229
230             return Excel::download(new DocentesExport, 'docentes.xlsx');
231         }
232     }
233 }

```

#### 4.4.3 Vistas

A continuación se muestran las vistas creadas para el modulo docentes.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > docente > index.blade.php > ...
1  @if (!Auth::check())
2    @include('home')
3  @else
4    @extends('layouts.app')
5    @section('title')
6      <h1 class="titulo"><i class="fas fa-users"></i> Módulo docentes</h1>
7      @section('message')<p>Lista de registro docentes</p>@endsection
8    @endsection
9    @section('content')
10      <div class="container-fluid">
11        <div class="tile col-md-12 p-3">
12          <div class="card-body">
13            <div class="row">
14              <div class="col-md-7">
15                <h2>Lista de registros</h2>
16                <a class="btn btn-outline-danger btn-radius" href="{{ url('docente/pdf') }}"
17                  title="Generar reporte pdf" target="_blank"><i class="fas fa-file-pdf"></i></a>
18                <a class="btn btn-outline-primary btn-radius" href="{{ url('docente/export') }}"
19                  title="Generar reporte excel" target="_blank"><i class="fas fa-file-excel"></i></a>
20              </div>
21              <div class="col-md-5 d-flex justify-content-end align-items-center">
22                @if (Auth::user()->per_tipo_usuario == 1 || Auth::user()->per_tipo_usuario == 2)
23                  <a class="btn btn-outline-success" href="{{ url('docente/create') }}"
24                    class="fas fa-plus-circle"><i></i>
25                  Nuevo</a>
26                @endif
27              </div>
28            </div>
29          </div>
```

##### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nuevo docente.

```
resources > views > docente > create.blade.php > ...
1  @if (!Auth::check())
2    @include('home')
3  @else
4    @extends('layouts.app')
5    @section('title')
6      <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7      @section('message')<p>Diligenciar los campos requeridos, para el debido registro del docente.</p>@endsection
8    @endsection
9    @section('content')
10      <div class="container-fluid">
11        <div class="tile w-100">
12          <h4 class="title title"><i class="fab fa-wpforms"></i> Registro docente</h4>
13          <form action="/docente" method="post">...
14          </form>
15        </div>
16      </div>
17      <br>
18    @endsection
19  @endif
20
```

##### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```
resources > views > docente > edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del docente.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile-title">Actualizar información</h4>
13                 <form action="/docente/{{ $docente->id }}" method="post">...
14                     ...
15                 </form>
16             </div>
17             <br>
18         @endsection
19     @endif
20 
```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de un docente específico.

```
resources > views > docente > show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">...
12             </div>
13             <br>
14         @endsection
15     @endif
16 
```

#### 4.4.4 Export

Este archivo llamado ExportDocentes contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase DocentesExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > DocentesExport.php > DocentesExport > collection
  2  namespace App\Exports;
  3
  4  use Illuminate\Support\Facades\DB;
  5  use Maatwebsite\Excel\Concerns\FromCollection;
  6  use Maatwebsite\Excel\Concerns\WithHeadings;
  7
  8  class DocentesExport implements FromCollection, WithHeadings
  9  {
10      public function headings(): array
11      {
12          return [
13              'Id', 'Tipo de documento', 'Número de documento',
14              'Nombre (s)', 'Apellido (s)', 'Teléfono 1',
15              'Teléfono 2', 'Dirección', 'Correo electrónico personal',
16              'Correo electrónico institucional', 'Departamento',
17              'Ciudad', 'Estudios', 'Fecha inicio contrato',
18              'Fecha fin contrato', 'Categoría', 'Reconocimientos'
19          ];
20      }
21      public function collection()
22      {
23          $docentes = DB::table('docentes')->select(
24              'docentes.id', 'abrv', 'doc_numero_documento',
25              'doc_nombre', 'doc_apellido', 'doc_telefono1',
26              'doc_telefono2', 'doc_direccion', 'doc_correo_personal',
27              'doc_correo_institucional', 'doc_departamento', 'doc_ciudad',
28              'doc_estudios', 'doc_fecha_inicio_contra',
29              'doc_fecha_fin_contra', 'doc_categoria', 'doc_reconocimiento'
30          )
31          ->join('tipo_documento', 'docentes.doc_tipo_documento', '=', 'tipo_documento.id')
32          ->get();
33
34          return $docentes;
35      }
36  }
```

## 4.5 Pruebas Saber Pro

### 4.5.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.

```
app > Models > PruebasSaber.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class PruebasSaber extends Model
9  {
10     use HasFactory;
11
12     protected $table = "pruebas_saber";
13
14     protected $fillable = [
15         'id',
16         'pr_id_estudiante',
17         'pr_codigo_prueba',
18         'pr_fecha_presentacion',
19         'pr_programa',
20         'pr_sede',
21         'pr_grupo_referencial'
22     ];
23
24     public function estudiantes(){
25         return $this->belongsTo(Estudiante::class, 'pr_id_estudiante');
26     }
27
28     public function programas(){
29         return $this->belongsTo(Programa::class, 'pr_id_programa');
30     }
31
32 }
33
```

### 4.5.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo pruebas saber pro. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > PruebaController.php > ...
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Exports\PruebasExport;
5 use App\Models\PruebasSaber;
6 use App\Models\Estudiante;
7 use App\Models\Programa;
8 use Illuminate\Http\Request;
9 use RealRashid\SweetAlert\Facades\Alert;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\DB;
12 use Maatwebsite\Excel\Facades\Excel;
13
14 class PruebaController extends Controller
15 {
16     public function index()
17     {
18         $pruebas = PruebasSaber::all();
19         $estudiantes = Estudiante::all();
20         $programas = Programa::all();
21         if (Auth::user()->per_tipo_usuario == '3') {
22             return view('pruebas.index')->with('pruebas', $pruebas);
23         } else {
24             if ($estudiantes->count() <= 0 && $programas->count() <= 0) {
25                 Alert::warning('Requisitos', 'Registre programas y estudiantes');
26                 return redirect('/home');
27             } else if ($estudiantes->count() <= 0) {
28                 Alert::warning('Requisitos', 'Registre estudiantes');
29                 return redirect('/home');
30             } else {
31                 return view('pruebas.index')->with('pruebas', $pruebas);
32             }
33         }
34     }

```

```

36     public function create()
37     {
38         $estudiantes = Estudiante::all();
39         $programas = Programa::all();
40         return view('pruebas.create')->with('estudiantes', $estudiantes)
41             ->with('programas', $programas);
42     }
43
44     public function store(Request $request)
45     {
46         if ($request->get('pr_id_estudiante') == "" || $request->get('pr_id_programa')) {
47             Alert::warning('El valor seleccionado es incorrecto');
48             return back()->withInput();
49         }
50
51         $rules = [
52             'pr_codigo_prueba' => 'required',
53             'pr_fecha_presentacion' => 'required',
54             'pr_grupo_referencial' => 'required',
55             'pr_grupo_referencia' => 'required'
56         ];
57
58         $messages = [
59             'pr_codigo_prueba.required' => 'El campo código prueba es requerido',
60             'pr_fecha_presentacion.required' => 'El campo año presentación prueba es requerido',
61             'pr_grupo_referencial.required' => 'El campo grupo referencial es requerido',
62             'pr_grupo_referencia.required' => 'El campo grupo referencia NBC es requerido'
63         ];
64
65         $this->validate($request, $rules, $messages);
66
67         $pruebas = new PruebasSaber();
68         $pruebas->pr_id_estudiante = $request->get('pr_id_estudiante');
69         $pruebas->pr_codigo_prueba = $request->get('pr_codigo_prueba');
70         $pruebas->pr_id_programa = $request->get('pr_id_programa');
71         $pruebas->pr_fecha_presentacion = $request->get('pr_fecha_presentacion');
72         $pruebas->pr_grupo_referencial = $request->get('pr_grupo_referencial');
73         $pruebas->pr_grupo_referencia = $request->get('pr_grupo_referencia');
74

```

```
75     $ExistPruId = DB::table('pruebas_saber')->where('pr_id_estudiante', $request->get('pr_id_estudiante'));
76
77     if ($ExistPruId->count() > 0) {
78         Alert::warning('El estudiante ya registra prueba saber Pro en el sistema');
79         return back()->withInput();
80     } else {
81         $pruebas->save();
82
83         DB::table('acciones_plataforma')->insert([
84             'usuario' => Auth::user()->id,
85             'accion' => 'insertar',
86             'modulo' => 'pruebas_saber'
87         ]);
88
89         Alert::success('Registro Exitoso');
90
91         return redirect('/prueba');
92     }
93 }
94
95 public function show($id)
96 {
97     $prueba = PruebasSaber::find($id);
98     $estudiantes = Estudiante::all();
99     $programas = Programa::all();
100    return view('pruebas.show')->with('prueba', $prueba)
101        ->with('estudiantes', $estudiantes)
102        ->with('programas', $programas);
103 }
104
105 public function edit($id)
106 {
107     $prueba = PruebasSaber::find($id);
108     $estudiantes = Estudiante::all();
109     $programas = Programa::all();
110    return view('pruebas.edit')->with('prueba', $prueba)
111        ->with('estudiantes', $estudiantes)
112        ->with('programas', $programas);
113 }
```

```
115     public function update(Request $request, $id)
116     {
117         $prueba = PruebasSaber::find($id);
118         $prueba->pr_id_estudiante = $request->get('pr_id_estudiante');
119         $prueba->pr_codigo_prueba = $request->get('pr_codigo_prueba');
120         $prueba->pr_id_programa = $request->get('pr_id_programa');
121         $prueba->pr_fecha_presentacion = $request->get('pr_fecha_presentacion');
122         $prueba->pr_grupo_referencial = $request->get('pr_grupo_referencial');
123         $prueba->pr_grupo_referencia = $request->get('pr_grupo_referencia');
124
125         $prueba->save();
126
127         DB::table('acciones_plataforma')->insert([
128             'usuario' => Auth::user()->id,
129             'accion' => 'editar',
130             'modulo' => 'pruebas_saber'
131         ]);
132
133         Alert::success('Registro Actualizado');
134
135         return redirect('/prueba');
136     }
137
138     public function destroy($id)
139     {
140         $prueba = PruebasSaber::find($id);
141         $prueba->delete();
142
143         DB::table('acciones_plataforma')->insert([
144             'usuario' => Auth::user()->id,
145             'accion' => 'eliminar',
146             'modulo' => 'pruebas_saber'
147         ]);
148
149         Alert::success('Registro Eliminado');
150
151         return redirect('/prueba');
152     }

```

```

154     public function pdf(Request $request)
155     {
156         $pruebas = PruebasSaber::all();
157         if ($pruebas->count() <= 0) {
158             Alert::warning('No hay registros');
159             return redirect('/prueba');
160         } else {
161             $view = \view('pruebas.pdf', compact('pruebas'))->render();
162             $pdf = \App::make('dompdf.wrapper');
163             $pdf->loadHTML($view);
164
165             DB::table('acciones_plataforma')->insert([
166                 'usuario' => Auth::user()->id,
167                 'accion' => 'pdf',
168                 'modulo' => 'pruebas_saber'
169             ]);
170
171             return $pdf->stream('pruebas-report.pdf');
172         }
173     }
174
175     public function export()
176     {
177         $pruebas = PruebasSaber::all();
178         if ($pruebas->count() <= 0) {
179             Alert::warning('No hay registros');
180             return redirect('/prueba');
181         } else {
182
183             DB::table('acciones_plataforma')->insert([
184                 'usuario' => Auth::user()->id,
185                 'accion' => 'excel',
186                 'modulo' => 'pruebas_saber'
187             ]);
188
189             return Excel::download(new PruebasExport, 'pruebas-saber.xlsx');
190         }
191     }
192 }

```

#### 4.5.3 Vistas

A continuación se muestran las vistas creadas para el modulo pruebas saber pro.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > pruebas > index.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-users"></i> Módulo pruebas saber pro</h1>
7         @section('message')<p>Lista de registro pruebas saber pro</p>@endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile col-md-12 p-3">
12                <div class="card-body">...
13            </div>
14        </div>
15    @endsection
16 @endif

```

### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nueva prueba saber pro.

```

resources > views > pruebas > create.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7         @section('message')<p>Diligenciar los campos requeridos, para el debido registro del docente.</p>@endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile w-100">
12                <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro prueba saber pro</h4>
13                <form action="/prueba" method="post">...
14            </div>
15        </div>
16    @endsection
17 @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > pruebas > edit.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fa-edit"></i> Formulario de edición de datos</h1>
7         @section('message')<p>Actualizar información de prueba saber pro.</p>@endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile w-100">
12                <h4 class="tile title">Actualizar información</h4>
13                <form action="/prueba/{{ $prueba->id }}" method="post">...
14            </div>
15        </div>
16    @endsection
17 @endif

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de una prueba saber específica.

```
resources > views > pruebas > show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">...
109            </div>
110        </div>
111    @endsection
112 @endif
```

#### 4.5.4 Export

Este archivo llamado ExportPruebas contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase PruebasExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > PruebasExport.php > PruebasExport > collection
  2  namespace App\Exports;
  3
  4  use Illuminate\Support\Facades\DB;
  5  use Maatwebsite\Excel\Concerns\FromCollection;
  6  use Maatwebsite\Excel\Concerns\WithHeadings;
  7
  8  class PruebasExport implements FromCollection, WithHeadings
  9  {
10      public function headings(): array
11      {
12          return [
13              'Id', 'Tipo de documento',
14              'Número de documento', 'Nombre (s)',
15              'Apellido (s)', 'Codigo prueba',
16              'Programa', 'Fecha presentación prueba',
17              'Grupo referencial', 'Grupo referencial NBC',
18          ];
19      }
20      public function collection()
21      {
22          $pruebas = DB::table('pruebas_saber')->select(
23              'pruebas_saber.id', 'abr',
24              'estu_numero_documento', 'estu_nombre',
25              'estu_apellido', 'pr_codigo_prueba',
26              'pro_nombre', 'pr_fecha_presentacion',
27              'pr_grupo_referencial', 'pr_grupo_referencia'
28          )
29          ->join('estudiantes', 'pruebas_saber.pr_id_estudiante', '=', 'estudiantes.id')
30          ->join('tipo_documento', 'estudiantes.estu_tipo_documento', '=', 'tipo_documento.id')
31          ->join('programas', 'estudiantes.estu_programa', '=', 'programas.id')
32          ->get();
33
34          return $pruebas;
35      }
36  }
```

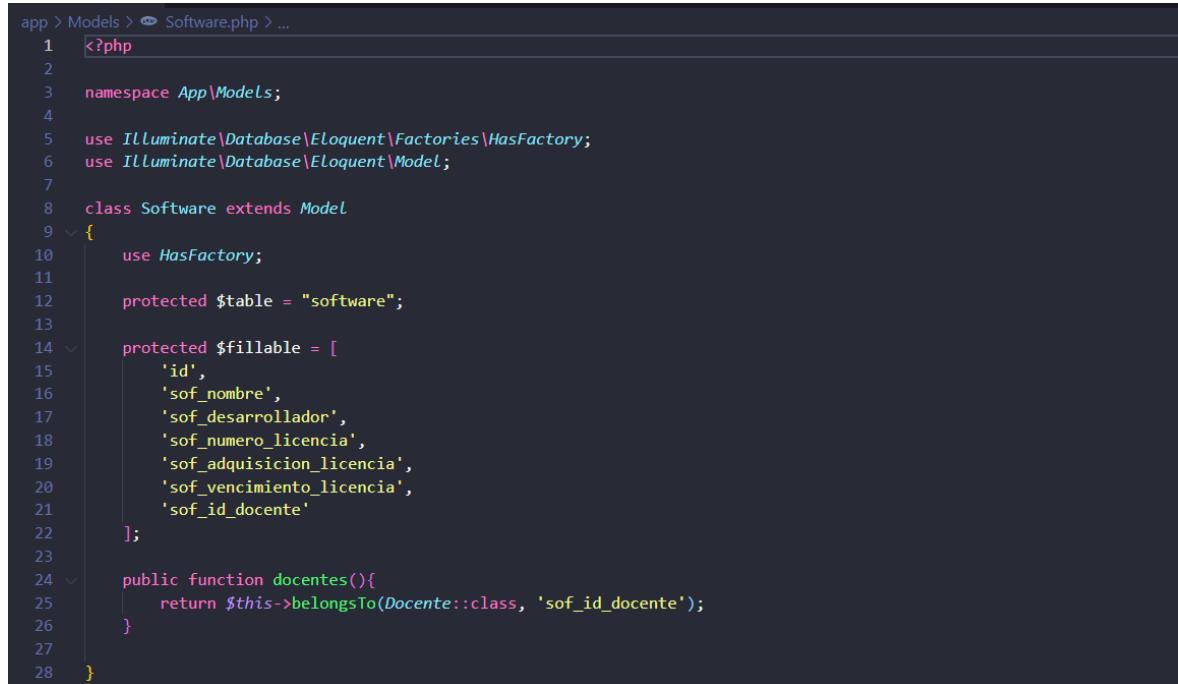
## 4.6 TIC'S

### 4.6.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.



```
app > Models > Software.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Software extends Model
9  {
10     use HasFactory;
11
12     protected $table = "software";
13
14     protected $fillable = [
15         'id',
16         'sof_nombre',
17         'sof_desarrollador',
18         'sof_numero_licencia',
19         'sof_adquisicion_licencia',
20         'sof_vencimiento_licencia',
21         'sof_id_docente'
22     ];
23
24     public function docentes(){
25         return $this->belongsTo(Docente::class, 'sof_id_docente');
26     }
27
28 }
```

### 4.6.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo tic's. Su ubicación es en la carpeta **App\Http\Controllers**.

```
app > Http > Controllers > SoftwareController.php > ...
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Exports\SoftwareExport;
5 use App\Models\Software;
6 use App\Models\Docente;
7 use Illuminate\Http\Request;
8 use Illuminate\Support\Facades\Auth;
9 use Illuminate\Support\Facades\DB;
10 use Maatwebsite\Excel\Facades\Excel;
11 use RealRashid\SweetAlert\Facades\Alert;
12
13 class SoftwareController extends Controller
14 {
15     public function index()
16     {
17         $softwares = Software::all();
18         $docentes = Docente::all();
19         if (Auth::user()->per_tipo_usuario == '3') {
20             return view('software.index')->with('softwares', $softwares);
21         } else {
22             if ($docentes->count() <= 0) {
23                 Alert::warning('Requisitos', 'Registre docentes');
24                 return redirect('/home');
25             } else {
26                 return view('software.index')->with('softwares', $softwares);
27             }
28         }
29     }
30
31     public function create()
32     {
33         $docentes = Docente::all();
34         return view('software.create')->with('docentes', $docentes);
35     }
}
```

```

37    public function store(Request $request)
38    {
39        if ($request->get('sof_id_docente') == "") {
40            Alert::warning('El valor seleccionado es incorrecto');
41            return back()->withInput();
42        }
43
44        $rules = [
45            'sof_nombre' => 'required',
46            'sof_desarrollador' => 'required',
47            'sof_numero_licencia' => 'required',
48            'sof_adquisicion_licencia' => 'required',
49            'sof_vencimiento_licencia' => 'required'
50        ];
51
52        $messages = [
53            'sof_nombre.required' => 'El campo nombre del software es requerido',
54            'sof_desarrollador.required' => 'El campo nombre del desarrollador es requerido',
55            'sof_numero_licencia.required' => 'El campo número de licencia es requerido',
56            'sof_adquisicion_licencia.required' => 'El campo fecha de adquisición es requerido',
57            'sof_vencimiento_licencia.required' => 'El campo fecha de vencimiento es requerido'
58        ];
59
60        $this->validate($request, $rules, $messages);
61
62        $softwares = new Software();
63        $softwares->sof_nombre = $request->get('sof_nombre');
64        $softwares->sof_desarrollador = $request->get('sof_desarrollador');
65        $softwares->sof_numero_licencia = $request->get('sof_numero_licencia');
66        $softwares->sof_adquisicion_licencia = $request->get('sof_adquisicion_licencia');
67        $softwares->sof_vencimiento_licencia = $request->get('sof_vencimiento_licencia');
68        $softwares->sof_id_docente = $request->get('sof_id_docente');
69
70        $softwares->save();
71
72        DB::table('acciones_plataforma')->insert([
73            'usuario' => Auth::user()->id,
74            'accion' => 'insertar',
75            'modulo' => 'software'
76        ]);
77
78        Alert::success('Registro Exitoso');
79
80        return redirect('/software');
81    }
82
83    public function show($id)
84    {
85        $software = Software::find($id);
86        $docentes = Docente::all();
87        return view('software.show')->with('software', $software)
88            ->with('docentes', $docentes);
89    }
90
91    public function edit($id)
92    {
93        $software = Software::find($id);
94        $docentes = Docente::all();
95        return view('software.edit')->with('software', $software)
96            ->with('docentes', $docentes);
97    }
98

```

```
99     public function update(Request $request, $id)
100    {
101        $software = Software::find($id);
102        $software->sof_nombre = $request->get('sof_nombre');
103        $software->sof_desarrollador = $request->get('sof_desarrollador');
104        $software->sof_numero_licencia = $request->get('sof_numero_licencia');
105        $software->sof_adquisicion_licencia = $request->get('sof_adquisicion_licencia');
106        $software->sof_vencimiento_licencia = $request->get('sof_vencimiento_licencia');
107        $software->sof_id_docente = $request->get('sof_id_docente');
108
109        $software->save();
110
111        DB::table('acciones_plataforma')->insert([
112            'usuario' => Auth::user()->id,
113            'accion' => 'editar',
114            'modulo' => 'software'
115        ]);
116
117        Alert::success('Registro Actualizado');
118
119        return redirect('/software');
120    }
121
122    public function destroy($id)
123    {
124        $software = Software::find($id);
125        $software->delete();
126
127        DB::table('acciones_plataforma')->insert([
128            'usuario' => Auth::user()->id,
129            'accion' => 'eliminar',
130            'modulo' => 'software'
131        ]);
132
133        Alert::success('Registro Eliminado');
134
135        return redirect('/software');
136    }
137
```

```

138     public function pdf(Request $request)
139     {
140         $softwares = Software::all();
141         if ($softwares->count() <= 0) {
142             Alert::warning('No hay registros');
143             return redirect('/software');
144         } else {
145             $view = \view('software.pdf', compact('softwares'))->render();
146             $pdf = \App::make('dompdf.wrapper');
147             $pdf->loadHTML($view);
148
149             DB::table('acciones_plataforma')->insert([
150                 'usuario' => Auth::user()->id,
151                 'accion' => 'pdf',
152                 'modulo' => 'software'
153             ]);
154
155             return $pdf->stream('software-report.pdf');
156         }
157     }
158
159     public function export()
160     {
161         $softwares = Software::all();
162         if ($softwares->count() <= 0) {
163             Alert::warning('No hay registros');
164             return redirect('/software');
165         } else {
166
167             DB::table('acciones_plataforma')->insert([
168                 'usuario' => Auth::user()->id,
169                 'accion' => 'excel',
170                 'modulo' => 'software'
171             ]);
172
173             return Excel::download(new SoftwareExport, 'softwares.xlsx');
174         }
175     }
176 }

```

#### 4.6.3 Vistas

A continuación se muestran las vistas creadas para el modulo tic's.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > software > index.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-users"></i> Módulo TIC'S</h1>
7          @section('message')<p>Lista de registro softwares en uso</p>@endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="col-md-12 shadow-sm p-3 bg-white"> ...
12         </div>
13     </div>
14     @endsection
15 @endif

```

### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de nueva herramienta tic's en uso por un docente.

```

resources > views > software > create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7          @section('message') <p>Diligenciar los campos requeridos, para el debido registro del software.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro software</h4>
13             <form action="/software" method="post"> ...
14         </div>
15     </div>
16     @endsection
17 @endif
18

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > software > edit.blade.php > div.container-fluid > div.tile.w-100
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del software.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13             <form action="/software/{{ $software->id }}" method="post"> ...
14         </div>
15     </div>
16     @endsection
17 @endif

```

### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de un software en uso específico.

```
resources > views > software > show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">...
107         </div>
108     </div>
109     @endsection
110 @endif
111
```

#### 4.6.4 Export

Este archivo llamado ExportSoftware contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase SoftwaresExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > SoftwareExport.php > ...
1  <?php
2
3  namespace App\Exports;
4
5  use DB;
6  use Maatwebsite\Excel\Concerns\FromCollection;
7  use Maatwebsite\Excel\Concerns\WithHeadings;
8
9  class SoftwareExport implements FromCollection, WithHeadings
10 {
11     public function headings(): array
12     {
13         return [
14             'Id',
15             'Nombre software',
16             'Nombre desarrollador',
17             'Número de licencia',
18             'Fecha adquisición de licencia',
19             'Fecha vencimiento de licencia',
20             'Docente',
21         ];
22     }
23     public function collection()
24     {
25         $softwares = DB::table('software')->select(
26             'software.id',
27             'sof_nombre',
28             'sof_desarrollador',
29             'sof_numero_licencia',
30             'sof_adquisicion_licencia',
31             'sof_vencimiento_licencia',
32             DB::raw("CONCAT(doc_nombre, ' ', doc_apellido)")
33         )
34         ->join('docentes', 'software.sof_id_docente', '=', 'docentes.id')
35         ->get();
36
37         return $softwares;
38     }
39 }
40
```

## 4.7 Extensión e Internacionalización

### 4.7.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```
app > Models > Extension.php > Extension > $fillable
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Extension extends Model
9  {
10     use HasFactory;
11
12     protected $table = "extension";
13
14     protected $fillable = [
15         'id',
16         'ext_nombre',
17         'ext_tipo_evento',
18         'ext_fecha_realizacion',
19         'ext_publico_objeto',
20         'ext_ponentes',
21         'ext_pais',
22         'ext_participantes',
23         'ext_img'
24     ];
25
26 }
```

### 4.7.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo extensión e internacionalización. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > ExtensionController.php > ExtensionController
  1  <?php
  2  namespace App\Http\Controllers;
  3
  4  use App\Exports\ExtensionsExport;
  5  use App\Models\Extension;
  6  use Illuminate\Http\Request;
  7  use Illuminate\Support\Facades\Auth;
  8  use Illuminate\Support\Facades\DB;
  9  use Maatwebsite\Excel\Facades\Excel;
 10 use RealRashid\SweetAlert\Facades\Alert;
 11
 12 class ExtensionController extends Controller
 13 {
 14     public function index()
 15     {
 16         $extensions = Extension::all();
 17         return view('extension.index')->with('extensions', $extensions);
 18     }
 19
 20     public function create()
 21     {
 22         return view('extension.create');
 23     }
 24
 25     public function store(Request $request)
 26     {
 27         $rules = [
 28             'ext_nombre' => 'required',
 29             'ext_tipo_evento' => 'required',
 30             'ext_fecha_realizacion' => 'required',
 31             'ext_publico_objeto' => 'required',
 32             'ext_participantes' => 'required'
 33         ];
 34
 35         $mesagges = [
 36             'ext_nombre.required' => 'El campo nombre del evento es requerido',
 37             'ext_tipo_evento.required' => 'El campo tipo del evento es requerido',
 38             'ext_fecha_realizacion.required' => 'El campo fecha realización es requerido',
 39             'ext_publico_objeto.required' => 'El campo público objeto es requerido',
 40             'ext_participantes.required' => 'El campo número de participantes es requerido'
 41         ];
 42
 43         $this->validate($request, $rules, $mesagges);
 44
 45         if ($request->get('ext_ponentes') == "" || $request->get('ext_pais') == "") {
 46             Alert::warning('Diligenciar los campos faltantes');
 47             return back()->withInput();
 48         }
 49
 50         $extensions = new Extension();
 51         $extensions->ext_nombre = $request->get('ext_nombre');
 52         $extensions->ext_tipo_evento = $request->get('ext_tipo_evento');
 53         $extensions->ext_fecha_realizacion = $request->get('ext_fecha_realizacion');
 54         $extensions->ext_publico_objeto = $request->get('ext_publico_objeto');
 55         $extensions->ext_ponentes = $request->get('ext_ponentes');
 56         $extensions->ext_pais = $request->get('ext_pais');
 57         $extensions->ext_participantes = $request->get('ext_participantes');
 58         $extensions->ext_img = $request->get('ext_img');
 59
 60         $extensions->save();
 61
 62         DB::table('acciones_plataforma')->insert([
 63             'usuario' => Auth::user()->id,
 64             'accion' => 'insertar',
 65             'modulo' => 'extension'
 66         ]);
 67
 68         Alert::success('Registro Exitoso');
 69
 70         return redirect('/extension');
 71     }
 72
 73     public function show($id)
 74     {
 75         $extension = Extension::find($id);
 76         return view('extension.show')->with('extension', $extension);
 77     }
 78
 79     public function edit($id)
 80     {
 81         $extension = Extension::find($id);
 82         return view('extension.edit')->with('extension', $extension);
 83     }

```

```

85     public function update(Request $request, $id)
86     {
87         $extension = Extension::find($id);
88         $extension->ext_nombre = $request->get('ext_nombre');
89         $extension->ext_tipo_evento = $request->get('ext_tipo_evento');
90         $extension->ext_fecha_realizacion = $request->get('ext_fecha_realizacion');
91         $extension->ext_publico_objeto = $request->get('ext_publico_objeto');
92         $extension->ext_ponentes = $request->get('ext_ponentes');
93         $extension->ext_pais = $request->get('ext_pais');
94         $extension->ext_participantes = $request->get('ext_participantes');
95         $extension->ext_img = $request->get('ext_img');
96
97         $extension->save();
98
99         DB::table('acciones_plataforma')->insert([
100             'usuario' => Auth::user()->id,
101             'accion' => 'editar',
102             'modulo' => 'extension'
103         ]);
104
105         Alert::success('Registro Actualizado');
106
107         return redirect('/extension');
108     }
109
110     public function destroy($id)
111     {
112         $extension = Extension::find($id);
113         $extension->delete();
114
115         DB::table('acciones_plataforma')->insert([
116             'usuario' => Auth::user()->id,
117             'accion' => 'eliminar',
118             'modulo' => 'extension'
119         ]);
120
121         Alert::success('Registro Eliminado');
122
123         return redirect('/extension');
124     }
125
126     public function pdf(Request $request)
127     {
128         $extensions = Extension::all();
129         if ($extensions->count() <= 0) {
130             Alert::warning('No hay registros');
131             return back()->withInput();
132         } else {
133             $view = \view('extension.pdf', compact('extensions'))->render();
134             $pdf = \App::make('dompdf.wrapper');
135             $pdf->loadHTML($view);
136
137             DB::table('acciones_plataforma')->insert([
138                 'usuario' => Auth::user()->id,
139                 'accion' => 'pdf',
140                 'modulo' => 'extension'
141             ]);
142
143             return $pdf->stream('extension-reporte.pdf');
144         }
145     }
146
147     public function export(){
148         $extensions = Extension::all();
149         if ($extensions->count() <= 0) {
150             Alert::warning('No hay registros');
151             return back()->withInput();
152         }else{
153
154             DB::table('acciones_plataforma')->insert([
155                 'usuario' => Auth::user()->id,
156                 'accion' => 'excel',
157                 'modulo' => 'extension'
158             ]);
159
160             return Excel::download(new ExtensionsExport, 'extension.xlsx');
161         }
162     }

```

#### 4.7.3 Vistas

A continuación se muestran las vistas creadas para el modulo extensión e internacionalización.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > extension > index.blade.php > ...
1  @if (!Auth::check())
2    @include('home')
3  @else
4    @extends('layouts.app')
5    @section('title')
6      <h1 class="fas fa-users"></i> Módulo extensión e internacionalización</h1>
7      @section('message')<p>Lista de registro extensión e internacionalización</p>@endsection
8    @endsection
9    @section('content')
10   <div class="container-fluid">
11     <div class="tile col-md-12 p-3">...
12   </div>
13   </div>
14   @endsection
15 @endif
16
```

##### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de un nuevo evento.

```
resources > views > extension > create.blade.php > ...
1  @if (!Auth::check())
2    @include('home')
3  @else
4    @extends('layouts.app')
5    @section('title')
6      <h1 class="fas fa-vector-square"></i> Formulario de registro</h1>
7      @section('message')<p>Diligenciar los campos requeridos, para el debido registro del docente.</p>@endsection
8    @endsection
9    @section('content')
10   <div class="container-fluid">
11     <div class="tile w-100">
12       <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro extensión e internacionalización</h4>
13       <form action="/extension" method="post">...
14     </div>
15   </div>
16   @endsection
17 @endif
18
```

##### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > extension > edit.blade.php > ...
1 ~ @if (!Auth::check())
2   @include('home')
3 ~ @else
4   @extends('layouts.app')
5 ~ @section('title')
6   <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7   @section('message') <p>Actualizar información del programa.</p> @endsection
8 @endsection
9 ~ @section('content')
10 ~   <div class="container-fluid">
11   <div class="tile w-100">
12     <h4 class="tile title">Actualizar información</h4>
13   <form action="/extension/{{ $extension->id }}" method="post"> ...
142   </form>
143   </div>
144   @endsection
145 @endif
146

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de un evento específico.

```

resources > views > extension > show.blade.php > ...
1 ~ @if (!Auth::check())
2   @include('home')
3 ~ @else
4   @extends('layouts.app')
5 ~ @section('title')
6   <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7   @section('message') <p>Información almacenada en el sistema</p> @endsection
8 @endsection
9 ~ @section('content')
10 ~   <div class="container-fluid">
11   <div class="tile w-100">...
130   </div>
131   @endsection
132 @endif
133

```

#### 4.7.4 Export

Este archivo llamado ExportExtension contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase ExtensionsExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```

app > Exports > ExtensionsExport.php > ExtensionsExport > collection
1  <?php
2
3  namespace App\Exports;
4
5  use Illuminate\Support\Facades\DB;
6  use Maatwebsite\Excel\Concerns\FromCollection;
7  use Maatwebsite\Excel\Concerns\WithHeadings;
8
9  class ExtensionsExport implements FromCollection, WithHeadings
10 {
11     public function headings(): array
12     {
13         return [
14             'Id', 'Nombre evento',
15             'Tipo evento', 'Fecha realización',
16             'Público objeto', 'Ponente (s)',
17             'País', 'Número de participantes',
18             'Url imágenes',
19         ];
20     }
21     public function collection()
22     {
23         $extensions = DB::table('extension')->select(
24             'id', 'ext_nombre',
25             'ext_tipo_evento', 'ext_fecha_realizacion',
26             'ext_publico_objeto', 'ext_ponentes',
27             'ext_pais', 'ext_participantes',
28             'ext_img',
29         )->get();
30
31         return $extensions;
32     }
33 }

```

## 4.8 Prácticas de grado

### 4.8.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.

```
app > Models > Practica.php > Practica > $fillable
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Practica extends Model
9  {
10     use HasFactory;
11
12     protected $table = "practicas";
13
14     protected $fillable = [
15         'id',
16         'pra_id_estudiante',
17         'pra_razon_social',
18         'pra_nit_empresa',
19         'pra_telefono',
20         'pra_direccion',
21         'pra_fecha_inicio',
22         'pra_fecha_fin'
23     ];
24
25     public function estudiantes(){
26         return $this->belongsTo(Estudiante::class, 'pra_id_estudiante');
27     }
28
29 }
```

#### 4.8.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo prácticas de grado. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > PracticaController.php > PracticaController
  1  <?php
  2
  3  namespace App\Http\Controllers;
  4
  5  use App\Models\Practica;
  6  use App\Models\Estudiante;
  7  use RealRashid\SweetAlert\Facades\Alert;
  8  use Illuminate\Http\Request;
  9  use App\Exports\PracticasExport;
 10 use Illuminate\Support\Facades\Auth;
 11 use Illuminate\Support\Facades\DB;
 12 use Maatwebsite\Excel\Facades\Excel;
 13
 14 class PracticaController extends Controller
 15 {
 16     public function index()
 17     {
 18         $practicas = Practica::all();
 19         $estudiantes = Estudiante::all();
 20         if (Auth::user()->per_tipo_usuario == '3') {
 21             return view('practica.index')->with('practicas', $practicas);
 22         } else {
 23             if ($estudiantes->count() <= 0) {
 24                 Alert::warning('Requisitos', 'Registre estudiantes');
 25                 return redirect('/home');
 26             } else {
 27                 return view('practica.index')->with('practicas', $practicas);
 28             }
 29         }
 30     }
 31
 32     public function create()
 33     {
 34         $estudiantes = Estudiante::all();
 35         return view('practica.create')->with('estudiantes', $estudiantes);
 36     }
 37
 38     public function store(Request $request)
 39     {
 40         $practicas = new Practica();
 41         $practicas->pra_id_estudiante = $request->get('pra_id_estudiante');
 42         $practicas->pra_razon_social = $request->get('pra_razon_social');
 43         $practicas->pra_nit_empresa = $request->get('pra_nit_empresa');
 44         $practicas->pra_telefono = $request->get('pra_telefono');
 45         $practicas->pra_direccion = $request->get('pra_direccion');
 46         $practicas->pra_fecha_inicio = $request->get('pra_fecha_inicio');
 47         $practicas->pra_fecha_fin = $request->get('pra_fecha_fin');

 48         $rules = [
 49             'pra_razon_social' => 'required',
 50             'pra_nit_empresa' => 'required',
 51             'pra_telefono' => 'required',
 52             'pra_direccion' => 'required',
 53             'pra_fecha_inicio' => 'required',
 54             'pra_fecha_fin' => 'required'
 55         ];
 56
 57         $messages = [
 58             'pra_razon_social.required' => 'El campo razón social es requerido',
 59             'pra_nit_empresa.required' => 'El campo nit de la empresa es requerido',
 60             'pra_telefono.required' => 'El campo telefono empresa es requerido',
 61             'pra_direccion.required' => 'El campo dirección empresa es requerido',
 62             'pra_fecha_inicio.required' => 'El campo fecha inicio laboral es requerido',
 63         ];
 64
 65         $this->validate($request, $rules, $messages);

 66
 67         if ($request->get('pra_id_estudiante') == "") {
 68             Alert::warning('El valor seleccionado es incorrecto');
 69             return back()->withInput();
 70         }

 71         $ExistEstu = DB::table('practicas')->where('pra_id_estudiante', $request->get('pra_id_estudiante'));

 72         if ($ExistEstu->count() > 0) {
 73             Alert::warning('El estudiante ya registra practicas laborales en el sistema');
 74             return back()->withInput();
 75         } else {
 76             $practicas->save();

 77             DB::table('acciones_plataforma')->insert([
 78                 'usuario' => Auth::user()->id,
 79                 'accion' => 'insertar',
 80                 'modulo' => 'practicas'
 81             ]);

 82             Alert::success('Registro Exitoso');

 83             return redirect('/practica');
 84         }
 85     }
 86
 87
 88
 89
 90
 91

```

```

93    public function show($id)
94    {
95        $estudiantes = Estudiante::all();
96        $practica = Practica::find($id);
97        return view('practica.show')->with('practica', $practica)
98            ->with('estudiantes', $estudiantes);
99    }
100
101   public function edit($id)
102   {
103       $estudiantes = Estudiante::all();
104       $practica = Practica::find($id);
105       return view('practica.edit')->with('practica', $practica)
106           ->with('estudiantes', $estudiantes);
107   }
108
109   public function update(Request $request, $id)
110   {
111       $practica = Practica::find($id);
112       $practica->pra_id_estudiante = $request->get('pra_id_estudiante');
113       $practica->pra_razon_social = $request->get('pra_razon_social');
114       $practica->pra_nit_empresa = $request->get('pra_nit_empresa');
115       $practica->pra_telefono = $request->get('pra_telefono');
116       $practica->pra_direccion = $request->get('pra_direccion');
117       $practica->pra_fecha_inicio = $request->get('pra_fecha_inicio');
118       $practica->pra_fecha_fin = $request->get('pra_fecha_fin');
119
120       $practica->save();
121
122       DB::table('acciones_plataforma')->insert([
123           'usuario' => Auth::user()->id,
124           'accion' => 'editar',
125           'modulo' => 'practicas'
126       ]);
127
128       Alert::success('Registro Actualizado');
129
130       return redirect('/practica');
131   }

```

```

133   public function destroy($id)
134   {
135       $practica = Practica::find($id);
136       $practica->delete();
137
138       DB::table('acciones_plataforma')->insert([
139           'usuario' => Auth::user()->id,
140           'accion' => 'eliminar',
141           'modulo' => 'practicas'
142       ]);
143
144       Alert::success('Registro Eliminado');
145
146       return redirect('/practica');
147   }
148
149   public function pdf(Request $request)
150   {
151       $practicas = Practica::all();
152       if ($practicas->count() > 0) {
153           Alert::warning('No hay registros');
154           return back()->withInput();
155       } else {
156           $view = \view('practica.pdf', compact('practicas'))->render();
157           $pdf = \App::make('dompdf.wrapper');
158           $pdf->loadHTML($view);
159
160           DB::table('acciones_plataforma')->insert([
161               'usuario' => Auth::user()->id,
162               'accion' => 'pdf',
163               'modulo' => 'practicas'
164           ]);
165
166           return $pdf->stream('practicas-reporte.pdf');
167       }
168   }

```

```

170     public function export()
171     {
172         $practicas = Practica::all();
173         if ($practicas->count() <= 0) {
174             Alert::warning('No hay registros');
175             return back()->withInput();
176         } else {
177
178             DB::table('acciones_plataforma')->insert([
179                 'usuario' => Auth::user()->id,
180                 'accion' => 'excel',
181                 'modulo' => 'practicas'
182             ]);
183
184             return Excel::download(new PracticasExport, 'practicas.xlsx');
185
186         }
187     }

```

#### 4.8.3 Vistas

A continuación se muestran las vistas creadas para el modulo practicas laborales.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > practica > index.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-users"></i> Módulo practicas de grado</h1>
7          @section('message')<p>Lista de registro practicas de grado</p>@endsection
8      @endsection
9
10     @section('content')
11         <div class="container-fluid">
12             <div class="tile col-md-12 p-3">
13                 <div class="card-body">...
14             </div>
15         </div>
16     @endsection
17     @endif

```

##### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de una nueva práctica de grado.

```

resources > views > practica >  create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7      @section('message') <p>Diligenciar los campos requeridos, para el debido registro de las prácticas de grado.</p>
8      @endsection
9      @endsection
10     @section('content')
11         <div class="container-fluid">
12             <div class="tile w-100">
13                 <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro practicas de grado</h4>
14             <form action="/practica" method="post"> ...
15             </form>
16         </div>
17     </div>
18     @endsection
19     @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > practica >  edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información práctica de grado.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13             <form action="/practica/{{ $practica->id }}" method="post"> ...
14             </form>
15         </div>
16     </div>
17     @endsection
18     @endif

```

### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de una práctica laboral específica.

```

resources > views > practica >  show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100"> ...
12         </div>
13     </div>
14     @endsection
15     @endif

```

#### 4.8.4 Export

Este archivo llamado ExportPractica contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase PracticasExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > ↗ PracticasExport.php > ↗ PracticasExport > headings
1  <?php
2  namespace App\Exports;
3
4  use Illuminate\Support\Facades\DB;
5  use Maatwebsite\Excel\Concerns\FromCollection;
6  use Maatwebsite\Excel\Concerns\WithHeadings;
7
8  class PracticasExport implements FromCollection, WithHeadings
9  {
10      public function headings(): array
11      {
12          return [
13              'Id','Tipo documento',
14              'Número de documento','Nombre (s)',
15              'Apellido (s)','Razón social',
16              'Nit','Teléfono',
17              'Dirección','Fecha inicio prácticas','Fecha fin prácticas'
18          ];
19      }
20      public function collection()
21      {
22          $practicas = DB::table('practicas')->select(
23              'practicas.id','abn',
24              'estu_numero_documento','estu_nombre',
25              'estu_apellido','pra_razon_social',
26              'pra_nit_empresa','pra_telefono',
27              'pra_direccion','pra_fecha_inicio',
28              'pra_fecha_fin'
29          )
30          ->join('estudiantes', 'practicas.pra_id_estudiante', '=', 'estudiantes.id')
31          ->join('tipo_documento', 'estudiantes.estu_tipo_documento', '=', 'tipo_documento.id')
32          ->get();
33
34      }
35  }
```

## 4.9 Laboratorios

### 4.9.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista **index.blade.php** del módulo.

```
app > Models > Laboratorio.php > ...
1 <?php
2 namespace App\Models;
3
4 use Illuminate\Database\Eloquent\Factories\HasFactory;
5 use Illuminate\Database\Eloquent\Model;
6
7 class Laboratorio extends Model
8 {
9     use HasFactory;
10
11     protected $table = "uso_laboratorio";
12
13     protected $fillable = [
14         'id',
15         'lab_nombre',
16         'lab_lugar',
17         'lab_id_docente',
18         'lab_caracteristicas',
19         'lab_fecha_laboratorio',
20         'lab_id_programa'
21     ];
22
23     public function docentes(){
24         return $this->belongsTo(Docente::class, 'lab_id_docente');
25     }
26
27     public function programas(){
28         return $this->belongsTo(Programa::class, 'lab_id_programa');
29     }
30 }
31 }
```

### 4.9.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo laboratorios. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > LaboratorioController.php > LaboratorioController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Exports\LaboratoriosExport;
6  use App\Models\Docente;
7  use App\Models\Laboratorio;
8  use App\Models\Programa;
9  use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\DB;
12 use Maatwebsite\Excel\Facades\Excel;
13 use RealRashid\SweetAlert\Facades\Alert;
14
15
16 class LaboratorioController extends Controller
17 {
18     public function index()
19     {
20         $laboratorios = Laboratorio::all();
21         $docentes = Docente::all();
22         $programas = Programa::all();
23         if (Auth::user()->per_tipo_usuario == '3') {
24             return view('laboratorio.index')->with('laboratorios', $laboratorios);
25         } else {
26             if ($docentes->count() <= 0 && $programas->count() <= 0) {
27                 Alert::warning('Requisitos', 'Registre programas y docentes');
28                 return redirect('/home');
29             } else if ($programas->count() <= 0) {
30                 Alert::warning('Requisitos', 'Registre programas');
31                 return redirect('/home');
32             } else if ($docentes->count() <= 0) {
33                 Alert::warning('Requisitos', 'Registre docentes');
34                 return redirect('/home');
35             } else {
36                 return view('laboratorio.index')->with('laboratorios', $laboratorios);
37             }
38         }
39     }
40
41     public function create()
42     {
43         $docentes = Docente::all();
44         $programas = Programa::all();
45         return view('laboratorio.create')->with('docentes', $docentes)
46             ->with('programas', $programas);
47     }
48
49     public function store(Request $request)
50     {
51         $laboratorios = new Laboratorio();
52         $laboratorios->lab_nombre = $request->get('lab_nombre');
53         $laboratorios->lab_lugar = $request->get('lab_lugar');
54         $laboratorios->lab_id_docente = $request->get('lab_id_docente');
55         $laboratorios->lab_caracteristicas = $request->get('lab_caracteristicas');
56         $laboratorios->lab_fecha_laboratorio = $request->get('lab_fecha_laboratorio');
57         $laboratorios->lab_id_programa = $request->get('lab_id_programa');
58
59         $rules = [
60             'lab_nombre' => 'required',
61             'lab_lugar' => 'required',
62             'lab_fecha_laboratorio' => 'required'
63         ];
64
65         $messages = [
66             'lab_nombre.required' => 'El campo nombre del laboratorio es requerido',
67             'lab_lugar.required' => 'El campo lugar es requerido',
68             'lab_fecha_laboratorio.required' => 'El campo fecha laboratorio es requerido'
69         ];
70
71         $this->validate($request, $rules, $messages);
72
73         if ($request->get('lab_id_docente') == "" || $request->get('lab_id_programa') == "" || $request->get('lab_caracteristicas') == "") {
74             Alert::warning('El valor seleccionado es incorrecto');
75             return back()->withInput();
76         }

```

```

78     $laboratorios->save();
79
80     DB::table('acciones_plataforma')->insert([
81         'usuario' => Auth::user()->id,
82         'accion' => 'insertar',
83         'modulo' => 'uso_laboratorio'
84     ]);
85
86     Alert::success('Registro Exitoso');
87
88     return redirect('/laboratorio');
89 }
90
91 public function show($id)
92 {
93     $laboratorio = Laboratorio::find($id);
94     $docentes = Docente::all();
95     $programas = Programa::all();
96     return view('laboratorio.show')->with('laboratorio', $laboratorio)
97         ->with('docentes', $docentes)
98         ->with('programas', $programas);
99 }
100
101 public function edit($id)
102 {
103     $laboratorio = Laboratorio::find($id);
104     $docentes = Docente::all();
105     $programas = Programa::all();
106     return view('laboratorio.edit')->with('laboratorio', $laboratorio)
107         ->with('docentes', $docentes)
108         ->with('programas', $programas);
109 }
110
111 public function update(Request $request, $id)
112 {
113     $laboratorio = Laboratorio::find($id);
114     $laboratorio->lab_nombre = $request->get('lab_nombre');
115     $laboratorio->lab_lugar = $request->get('lab_lugar');
116     $laboratorio->lab_id_docente = $request->get('lab_id_docente');
117     $laboratorio->lab_caracteristicas = $request->get('lab_caracteristicas');
118     $laboratorio->lab_fecha_laboratorio = $request->get('lab_fecha_laboratorio');
119     $laboratorio->lab_id_programa = $request->get('lab_id_programa');
120
121     $laboratorio->save();
122
123     DB::table('acciones_plataforma')->insert([
124         'usuario' => Auth::user()->id,
125         'accion' => 'editar',
126         'modulo' => 'uso_laboratorio'
127     ]);
128
129     Alert::success('Registro Actualizado');
130
131     return redirect('/laboratorio');
132 }
133
134 public function destroy($id)
135 {
136     $laboratorio = Laboratorio::find($id);
137     $laboratorio->delete();
138
139     DB::table('acciones_plataforma')->insert([
140         'usuario' => Auth::user()->id,
141         'accion' => 'eliminar',
142         'modulo' => 'uso_laboratorio'
143     ]);
144
145     Alert::success('Registro Eliminado');
146
147     return redirect('/laboratorio');
148 }

```

```

150     public function pdf(Request $request)
151     {
152         $laboratorios = Laboratorio::all();
153         if ($laboratorios->count() <= 0) {
154             Alert::warning('No hay registros');
155             return redirect('/laboratorio');
156         } else {
157             $view = \view('laboratorio.pdf', compact('laboratorios')->render());
158             $pdf = \App::make('dompdf.wrapper');
159             $pdf->loadHTML($view);
160
161             DB::table('acciones_plataforma')->insert([
162                 'usuario' => Auth::user()->id,
163                 'accion' => 'pdf',
164                 'modulo' => 'uso_laboratorio'
165             ]);
166
167             return $pdf->stream('laboratorios-report.pdf');
168         }
169     }
170
171     public function export()
172     {
173         $laboratorios = Laboratorio::all();
174         if ($laboratorios->count() <= 0) {
175             Alert::warning('No hay registros');
176             return redirect('/laboratorio');
177         } else {
178
179             DB::table('acciones_plataforma')->insert([
180                 'usuario' => Auth::user()->id,
181                 'accion' => 'excel',
182                 'modulo' => 'uso_laboratorio'
183             ]);
184
185             return Excel::download(new LaboratoriosExport, 'laboratorios.xlsx');
186         }
187     }

```

#### 4.9.3 Vistas

A continuación se muestran las vistas creadas para el modulo laboratorios.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > laboratorio > index.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-users"></i> Módulo laboratorios</h1>
7          @section('message')<p>Lista de registro laboratorios</p>@endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile col-md-12 p-3">
12                 <div class="card-body">...
13             </div>
14         </div>
15     @endsection
16 @endif

```

### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de un nuevo uso del laboratorio.

```

resources > views > laboratorio > create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7          @section('message') <p>Diligenciar los campos requeridos, para el debido registro del laboratorio.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro laboratorio</h4>
13                 <form action="/laboratorio" method="post">...
14             </div>
15         </div>
16     @endsection
17 @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > laboratorio > edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del laboratorio.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13                 <form action="/laboratorio/{{ $laboratorio->id }}" method="post">...
14             </div>
15         </div>
16     @endsection
17 @endif

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de uso de laboratorio específico.

```
resources > views > laboratorio > show.blade.php > ...
1 <?php if (!Auth::check()) {
2     return redirect()->route('home');
3 } ?>
4 <?php @extends('layouts.app') ?>
5 <?php @section('title') ?>
6     <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7     <?php @section('message') ?> <p>Información almacenada en el sistema</p> <?php @endsection ?>
8 <?php @endsection ?>
9 <?php @section('content') ?>
10 <?php <div class="container-fluid">
11     <?php <div class="tile w-100">...</div>
12 <?php </div>
13 <?php @endsection ?>
14 <?php @endif ?>
```

#### 4.9.4 Export

Este archivo llamado ExportLaboratorio contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase LaboratoriosExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > LaboratoriosExport.php > LaboratoriosExport > collection
  2  namespace App\Exports;
  3
  4  use Illuminate\Support\Facades\DB;
  5  use Maatwebsite\Excel\Concerns\FromCollection;
  6  use Maatwebsite\Excel\Concerns\WithHeadings;
  7
  8  class LaboratoriosExport implements FromCollection, WithHeadings
  9  {
10      public function headings(): array
11      {
12          return [
13              'Id', 'Nombre laboratorio',
14              'Lugar', 'Tipo documento',
15              'Número de documento', 'Nombre (s)',
16              'Apellido (s)', 'Características lab',
17              'Fecha realización', 'Programa'
18          ];
19      }
20      public function collection()
21      {
22          $laboratorios = DB::table('uso_laboratorio')->select([
23              'uso_laboratorio.id', 'lab_nombre',
24              'lab_lugar', 'abr',
25              'doc_numero_documento', 'doc_nombre',
26              'doc_apellido', 'lab_caracteristicas',
27              'lab_fecha_laboratorio', 'pro_nombre'
28          ])
29          ->join('docentes', 'uso_laboratorio.lab_id_docente', '=', 'docentes.id')
30          ->join('tipo_documento', 'docentes.doc_tipo_documento', '=', 'tipo_documento.id')
31          ->join('programas', 'uso_laboratorio.lab_id_programa', '=', 'programas.id')
32          ->get();
33
34          return $laboratorios;
35      }
36  }
```

## 4.10 Convenios

### 4.10.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```
app > Models > Convenio.php > Convenio > $fillable
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Convenio extends Model
9  {
10     use HasFactory;
11
12     protected $table = "convenio";
13
14     protected $fillable = [
15         'id',
16         'con_nombre',
17         'con_pais',
18         'con_objetivo',
19         'con_fecha_convenio'
20     ];
21 }
22 }
```

### 4.10.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo convenios. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > ConvenioController.php > ConvenioController
  1  <?php
  2
  3  namespace App\Http\Controllers;
  4
  5  use App\Exports\ConveniosExport;
  6  use App\Models\Convenio;
  7  use Illuminate\Http\Request;
  8  use Illuminate\Support\Facades\Auth;
  9  use Illuminate\Support\Facades\DB;
10  use Maatwebsite\Excel\Facades\Excel;
11  use RealRashid\SweetAlert\Facades\Alert;
12
13  class ConvenioController extends Controller
14 {
15      public function index()
16      {
17          $convenios = Convenio::all();
18          return view('convenio.index')->with('convenios', $convenios);
19      }
20
21      public function create()
22      {
23          return view('convenio.create');
24      }
25
26      public function store(Request $request)
27      {
28          $convenios = new Convenio();
29          $convenios->con_nombre = $request->get('con_nombre');
30          $convenios->con_pais = $request->get('con_pais');
31          $convenios->con_objetivo = $request->get('con_objetivo');
32          $convenios->con_fecha_convenio = $request->get('con_fecha_convenio');
33
34          $rules = [
35              'con_nombre' => 'required',
36              'con_pais' => 'required',
37              'con_fecha_convenio' => 'required'
38          ];
39
40          $messages = [
41              'con_nombre.required' => 'El campo nombre de la institución o entidad cooperante es requerido',
42              'con_pais.required' => 'El campo país es requerido',
43              'con_fecha_convenio.required' => 'El campo fecha del convenio es requerido'
44          ];
45
46          $this->validate($request, $rules, $messages);
47
48          if ($request->get('con_objetivo') == "") {
49              Alert::warning('Diligenciar el campo faltante');
50              return redirect('convenio');
51          }
52
53          $convenios->save();
54
55          DB::table('acciones_plataforma')->insert([
56              'usuario' => Auth::user()->id,
57              'accion' => 'insertar',
58              'modulo' => 'convenio'
59          ]);
56

```

```

61     Alert::success('Registro Exitoso');
62
63     return redirect('/convenio');
64 }
65
66 public function show($id)
67 {
68     $convenio = Convenio::find($id);
69     return view('convenio.show')->with('convenio', $convenio);
70 }
71
72 public function edit($id)
73 {
74     $convenio = Convenio::find($id);
75     return view('convenio.edit')->with('convenio', $convenio);
76 }
77
78 public function update(Request $request, $id)
79 {
80     $convenio = Convenio::find($id);
81     $convenio->con_nombre = $request->get('con_nombre');
82     $convenio->con_pais = $request->get('con_pais');
83     $convenio->con_objetivo = $request->get('con_objetivo');
84     $convenio->con_fecha_convenio = $request->get('con_fecha_convenio');

85     $convenio->save();

86     DB::table('acciones_plataforma')->insert([
87         'usuario' => Auth::user()->id,
88         'accion' => 'editar',
89         'modulo' => 'convenio'
90     ]);

91     Alert::success('Registro Actualizado');

92
93     return redirect('/convenio');
94 }
95
96
97 }
```

```

99
100 public function destroy($id)
101 {
102     $convenio = Convenio::find($id);
103     $convenio->delete();

104     DB::table('acciones_plataforma')->insert([
105         'usuario' => Auth::user()->id,
106         'accion' => 'eliminar',
107         'modulo' => 'convenio'
108     ]);

109     Alert::success('Registro Eliminado');

110     return redirect('/convenio');
111 }

112 public function pdf(Request $request)
113 {
114     $convenios = Convenio::all();
115     if ($convenios->count() <= 0) {
116         Alert::warning('No hay registros');
117         return back()->withInput();
118     } else {
119         $view = \view('convenio.pdf', compact('convenios'))->render();
120         $pdf = \App::make('dompdf.wrapper');
121         $pdf->loadHTML($view);

122         DB::table('acciones_plataforma')->insert([
123             'usuario' => Auth::user()->id,
124             'accion' => 'pdf',
125             'modulo' => 'convenio'
126         ]);

127         return $pdf->stream('convenios.pdf');
128     }
129 }
130
131
132
133
134 }
```

```

136     public function export(){
137         $convenios = Convenio::all();
138         if ($convenios->count() <= 0) {
139             Alert::warning('No hay registros');
140             return back()->withInput();
141         }else{
142
143             DB::table('acciones_plataforma')->insert([
144                 'usuario' => Auth::user()->id,
145                 'accion' => 'excel',
146                 'modulo' => 'convenio'
147             ]);
148
149             return Excel::download(new ConveniosExport, 'convenios.xlsx');
150         }
151     }
152 }
153

```

#### 4.10.3 Vistas

A continuación se muestran las vistas creadas para el modulo convenios.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > convenio > index.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-users"></i> Módulo convenios nacionales e internacionales</h1>
7         @section('message')<p>Lista de registro convenios nacionales e internacionales</p>@endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile col-md-12 p-3">
12                <div class="card-body">...
13            </div>
14        </div>
15    @endsection
16 @endif

```

##### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de un nuevo convenio.

```

resources > views > convenio >  create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7          @section('message') <p>Diligenciar los campos requeridos, para el debido registro del convenio.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro convenio</h4>
13             <form action="/convenio" method="post"> ...
14             </form>
15         </div>
16     @endsection
17 @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > convenio >  edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del convenio.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13             <form action="/convenio/{{ $convenio->id }}" method="post"> ...
14             </form>
15         </div>
16     @endsection
17 @endif

```

### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de un convenio específico.

```

resources > views > convenio >  show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100"> ...
12         </div>
13     @endsection
14 @endif

```

#### 4.10.4 Export

Este archivo llamado ExportConvenio contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase ConveniosExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabecal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > ConveniosExport.php > ...
1 <?php
2 namespace App\Exports;
3
4 use Illuminate\Support\Facades\DB;
5 use Maatwebsite\Excel\Concerns\FromCollection;
6 use Maatwebsite\Excel\Concerns\WithHeadings;
7
8 class ConveniosExport implements FromCollection, WithHeadings
9 {
10     public function headings(): array
11     {
12         return [
13             'Id', 'Nombre convenio',
14             'País', 'Objetivo convenio',
15             'Fecha convenio',
16         ];
17     }
18     public function collection()
19     {
20         $convenios = DB::table('convenio')->select(
21             'convenio.id', 'con_nombre',
22             'con_pais', 'con_objetivo',
23             'con_fecha_convenio',
24         )
25             ->get();
26
27         return $convenios;
28     }
29 }
```

### 4.11 Redes Académicas

#### 4.11.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

```

app > Models > RedAcademica.php > ...
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class RedAcademica extends Model
9 {
10     use HasFactory;
11
12     protected $table = "redes_academicas";
13
14     protected $fillable = [
15         'id',
16         'red_nombre',
17         'red_accion',
18         'red_fecha_afiliacion',
19         'red_programa',
20     ];
21 }
22

```

#### 4.11.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo redes académicas. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > RedAcademicaController.php > ...
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Exports\RedesExport;
5 use App\Models\RedAcademica;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\Auth;
8 use Illuminate\Support\Facades\DB;
9 use Maatwebsite\Excel\Facades\Excel;
10 use RealRashid\SweetAlert\Facades\Alert;
11
12 class RedAcademicaController extends Controller
13 {
14     public function index()
15     {
16         $reds = RedAcademica::all();
17         return view('red.index')->with('reds', $reds);
18     }
19
20     public function create()
21     {
22         return view('red.create');
23     }

```

```

25    public function store(Request $request)
26    {
27        $reds = new RedAcademica();
28        $reds->red_nombre = $request->get('red_nombre');
29        $reds->red_accion = $request->get('red_accion');
30        $reds->red_fecha_afiliacion = $request->get('red_fecha_afiliacion');
31        $reds->red_programa = $request->get('red_programa');
32
33        $rules = [
34            'red_nombre' => 'required',
35            'red_accion' => 'required',
36            'red_fecha_afiliacion' => 'required',
37            'red_programa' => 'required'
38        ];
39
40        $messages = [
41            'red_nombre.required' => 'El campo nombre red es requerido',
42            'red_accion.required' => 'El campo objetivo o accion que realiza es requerido',
43            'red_fecha_actualizacion.required' => 'El campo fecha de afiliación es requerido',
44            'red_programa.required' => 'El campo programa afiliado es requerido'
45        ];
46
47        $this->validate($request, $rules, $messages);
48
49        $reds->save();
50
51        DB::table('acciones_plataforma')->insert([
52            'usuario' => Auth::user()->id,
53            'accion' => 'insertar',
54            'modulo' => 'redes_academicas'
55        ]);
56
57        Alert::success('Registro Exitoso');
58
59        return redirect('/red');
60    }

```

```

62    public function show($id)
63    {
64        $red = RedAcademica::find($id);
65        return view('red.show')->with('red', $red);
66    }
67
68    public function edit($id)
69    {
70        $red = RedAcademica::find($id);
71        return view('red.edit')->with('red', $red);
72    }
73
74    public function update(Request $request, $id)
75    {
76        $red = RedAcademica::find($id);
77        $red->red_nombre = $request->get('red_nombre');
78        $red->red_accion = $request->get('red_accion');
79        $red->red_fecha_afiliacion = $request->get('red_fecha_afiliacion');
80        $red->red_programa = $request->get('red_programa');
81
82        $red->save();
83
84        DB::table('acciones_plataforma')->insert([
85            'usuario' => Auth::user()->id,
86            'accion' => 'editar',
87            'modulo' => 'redes_academicas'
88        ]);
89
90        Alert::success('Registro Actualizado');
91
92        return redirect('/red');
93    }

```

```

95     public function destroy($id)
96     {
97         $red = RedAcademica::find($id);
98         $red->delete();
99
100        DB::table('acciones_plataforma')->insert([
101            'usuario' => Auth::user()->id,
102            'accion' => 'eliminar',
103            'modulo' => 'redes_academicas'
104        ]);
105
106        Alert::success('Registro Eliminado');
107
108        return redirect('/red');
109    }
110
111    public function pdf(Request $request)
112    {
113        $redes = RedAcademica::all();
114        if ($redes->count() <= 0) {
115            Alert::warning('No hay registros');
116            return back()->withInput();
117        } else {
118            $view = \view('red.pdf', compact('redes'))->render();
119            $pdf = \App::make('dompdf.wrapper');
120            $pdf->loadHTML($view);
121
122            DB::table('acciones_plataforma')->insert([
123                'usuario' => Auth::user()->id,
124                'accion' => 'pdf',
125                'modulo' => 'redes_academicas'
126            ]);
127
128            return $pdf->stream('redes-academicas-report.pdf');
129        }
130    }
131
132    public function export(){
133        $redes = RedAcademica::all();
134        if ($redes->count() <= 0) {
135            Alert::warning('No hay registros');
136            return back()->withInput();
137        }else{
138
139            DB::table('acciones_plataforma')->insert([
140                'usuario' => Auth::user()->id,
141                'accion' => 'excel',
142                'modulo' => 'redes_academicas'
143            ]);
144
145            return Excel::download(new RedesExport, 'redes.xlsx');
146        }
147    }
148
149 }

```

#### 4.11.3 Vistas

A continuación se muestran las vistas creadas para el modulo redes academicas.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > red > index.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-users"></i> Módulo redes académicas</h1>
7         @section('message')<p>Lista de registro redes académicas</p>@endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile col-md-12 p-3">
12                <div class="card-body">...
13            </div>
14        </div>
15    @endsection
16 @endif

```

### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de una nueva red académica.

```

resources > views > red > create.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7         @section('message') <p>Diligenciar los campos requeridos, para el debido registro red académica.</p> @endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile w-100">
12                <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro red académica</h4>
13                <form action="/red" method="post">...
14            </div>
15        </div>
16    @endsection
17 @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > red > edit.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7         @section('message') <p>Actualizar información del programa.</p> @endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile w-100">
12                <h4 class="tile title">Actualizar información</h4>
13                <form action="/red/{{ $red->id }}" method="post">...
14            </div>
15        </div>
16    @endsection
17 @endif

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de una red específica.

```
resources > views > red > show.blade.php > ...
1 @if (!Auth::check())
2     @include('home')
3 @else
4     @extends('layouts.app')
5     @section('title')
6         <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7         @section('message') <p>Información almacenada en el sistema</p> @endsection
8     @endsection
9     @section('content')
10        <div class="container-fluid">
11            <div class="tile w-100">...
12        </div>
13    @endsection
14 @endif
```

#### 4.11.4 Export

Este archivo llamado ExportRedes contiene los parámetros establecidos para generar el reporte en formato **.xlsx**.

La estructura de la clase RedesExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > RedesExport.php > ...
1  <?php
2
3  namespace App\Exports;
4
5  use DB;
6  use Maatwebsite\Excel\Concerns\FromCollection;
7  use Maatwebsite\Excel\Concerns\WithHeadings;
8
9  class RedesExport implements FromCollection, WithHeadings
10 {
11     public function headings(): array
12     {
13         return [
14             'Id',
15             'Nombre red',
16             'Acción o objetivo',
17             'Fecha de afiliación',
18             'Programa',
19         ];
20     }
21     public function collection()
22     {
23         $redes = DB::table('redes_academicas')->select(
24             'redes_academicas.id',
25             'red_nombre',
26             'red_accion',
27             'red_fecha_afiliacion',
28             'red_programa',
29         )
30         ->get();
31
32         return $redes;
33     }
34 }
```

## 4.12 Movilidad

### 4.12.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista index.blade.php del módulo.

```
app > Models > Movilidad.php > ...
1 <?php
2 namespace App\Models;
3
4 use Illuminate\Database\Eloquent\Factories\HasFactory;
5 use Illuminate\Database\Eloquent\Model;
6
7 class Movilidad extends Model
8 {
9     use HasFactory;
10
11     protected $table = "movilidad";
12
13     protected $fillable = [
14         'id',
15         'mov_fecha_movilidad',
16         'mov_id_docente',
17         'mov_id_estudiante',
18         'mov_tipo',
19         'mov_evento',
20         'mov_ciudad_pais'
21     ];
22
23     public function docentes(){
24         return $this->belongsTo(Docente::class, 'mov_id_docente');
25     }
26
27     public function estudiantes(){
28         return $this->belongsTo(Estudiante::class, 'mov_id_estudiante');
29     }
30 }
31 }
```

#### 4.12.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo movilidad. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > MovilidadController.php > ...
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Exports\MovilidadesExport;
5 use App\Models\Docente;
6 use App\Models\Estudiante;
7 use App\Models\Movilidad;
8 use Illuminate\Http\Request;
9 use Illuminate\Support\Facades\Auth;
10 use Illuminate\Support\Facades\DB;
11 use Maatwebsite\Excel\Facades\Excel;
12 use RealRashid\SweetAlert\Facades\Alert;
13
14 class MovilidadController extends Controller
15 {
16     public function index()
17     {
18         $movilidades = Movilidad::all();
19         $estudiantes = Estudiante::all();
20         $docentes = Docente::all();
21         if (Auth::user()->per_tipo_usuario == '3') {
22             return view('movilidad.index')->with('movilidades', $movilidades);
23         } else {
24             if ($estudiantes->count() <= 0 && $docentes->count() <= 0) {
25                 Alert::warning('Requisitos', 'Registre docentes y estudiantes');
26                 return redirect('/home');
27             } else if ($docentes->count() <= 0) {
28                 Alert::warning('Requisitos', 'Registre docentes');
29                 return redirect('/home');
30             } else if ($estudiantes->count() <= 0) {
31                 Alert::warning('Requisitos', 'Registre estudiantes');
32                 return redirect('/home');
33             } else {
34                 return view('movilidad.index')->with('movilidades', $movilidades);
35             }
36         }
37     }
38
39     public function create()
40     {
41         $docentes = Docente::all();
42         $estudiantes = Estudiante::all();
43         return view('movilidad.create')->with('docentes', $docentes)
44             ->with('estudiantes', $estudiantes);
45     }
46
47     public function store(Request $request)
48     {
49         $movilidad = new Movilidad();
50         $movilidad->mov_fecha_movilidad = $request->get('mov_fecha_movilidad');
51         $movilidad->mov_id_docente = $request->get('mov_id_docente');
52         $movilidad->mov_id_estudiante = $request->get('mov_id_estudiante');
53         $movilidad->mov_tipo = $request->get('mov_tipo');
54         $movilidad->mov_evento = $request->get('mov_evento');
55         $movilidad->mov_ciudad_pais = $request->get('mov_ciudad_pais');
56
57         $rules = [
58             'mov_fecha_movilidad' => 'required',
59             'mov_tipo' => 'required',
60             'mov_evento' => 'required',
61             'mov_ciudad_pais' => 'required'
62         ];
63
64         $messages = [
65             'mov_fecha_movilidad.required' => 'El campo fecha movilidad es requerido',
66             'mov_tipo.required' => 'El campo tipo de movilidad es requerido',
67             'mov_evento.required' => 'El campo evento o actividad es requerido',
68             'mov_ciudad_pais.required' => 'El campo ciudad o pais es requerido'
69         ];
70
71         $this->validate($request, $rules, $messages);
72
73         if ($request->get('mov_id_docente') == "" || $request->get('mov_id_estudiante') == "") {
74             Alert::warning('Diligenciar los campos faltantes');
75             return back()->withInput();
76         }

```

```
78     $movilidads->save();
79
80     DB::table('acciones_plataforma')->insert([
81         'usuario' => Auth::user()->id,
82         'accion' => 'insertar',
83         'modulo' => 'movilidad'
84     ]);
85
86     Alert::success('Registro Exitoso');
87
88     return redirect('/movilidad');
89 }
90
91 public function show($id)
92 {
93     $docentes = Docente::all();
94     $estudiantes = Estudiante::all();
95     $movilidad = Movilidad::find($id);
96     return view('movilidad.show')->with('docentes', $docentes)
97             ->with('estudiantes', $estudiantes)
98             ->with('movilidad', $movilidad);
99 }
100
101 public function edit($id)
102 {
103     $docentes = Docente::all();
104     $estudiantes = Estudiante::all();
105     $movilidad = Movilidad::find($id);
106     return view('movilidad.edit')->with('docentes', $docentes)
107             ->with('estudiantes', $estudiantes)
108             ->with('movilidad', $movilidad);
109 }
```

```
111  public function update(Request $request, $id)
112  {
113      $movilidad = Movilidad::find($id);
114      $movilidad->mov_fecha_movilidad = $request->get('mov_fecha_movilidad');
115      $movilidad->mov_id_docente = $request->get('mov_id_docente');
116      $movilidad->mov_id_estudiante = $request->get('mov_id_estudiante');
117      $movilidad->mov_tipo = $request->get('mov_tipo');
118      $movilidad->mov_evento = $request->get('mov_evento');
119      $movilidad->mov_ciudad_pais = $request->get('mov_ciudad_pais');
120
121      $movilidad->save();
122
123      DB::table('acciones_plataforma')->insert([
124          'usuario' => Auth::user()->id,
125          'accion' => 'editar',
126          'modulo' => 'movilidad'
127      ]);
128
129      Alert::success('Registro Actualizado');
130
131      return redirect('/movilidad');
132  }
133
134  public function destroy($id)
135  {
136      $movilidad = Movilidad::find($id);
137      $movilidad->delete();
138
139      DB::table('acciones_plataforma')->insert([
140          'usuario' => Auth::user()->id,
141          'accion' => 'eliminar',
142          'modulo' => 'movilidad'
143      ]);
144
145      Alert::success('Registro Eliminado');
146
147      return redirect('/movilidad');
148  }
```

```

150     public function pdf(Request $request)
151     {
152         $movilidades = Movilidad::all();
153         if ($movilidades->count() <= 0) {
154             Alert::warning('No hay registros');
155             return redirect('/movilidad');
156         } else {
157             $view = \view('movilidad.pdf', compact('movilidades'))->render();
158             $pdf = \App::make('dompdf.wrapper');
159             $pdf->loadHTML($view);
160
161             DB::table('acciones_plataforma')->insert([
162                 'usuario' => Auth::user()->id,
163                 'accion' => 'pdf',
164                 'modulo' => 'movilidad'
165             ]);
166
167             return $pdf->stream('movilidades-reporte.pdf');
168         }
169     }
170
171     public function export()
172     {
173         $movilidades = Movilidad::all();
174         if ($movilidades->count() <= 0) {
175             Alert::warning('No hay registros');
176             return redirect('/movilidad');
177         } else {
178
179             DB::table('acciones_plataforma')->insert([
180                 'usuario' => Auth::user()->id,
181                 'accion' => 'excel',
182                 'modulo' => 'movilidad'
183             ]);
184
185             return Excel::download(new MovilidadesExport, 'movilidades.xlsx');
186         }
187     }
188 }

```

#### 4.12.3 Vistas

A continuación se muestran las vistas creadas para el modulo movilidad.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > movilidad > index.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-users"></i> Módulo movilidad</h1>
7          @section('message')<p>Lista de registro movilidad</p>@endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile col-md-12 p-3">
12                 <div class="card-body">...
13             </div>
14         </div>
15     @endsection
16 @endif

```

### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de una nueva movilidad.

```

resources > views > movilidad > create.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
7          @section('message') <p>Diligenciar los campos requeridos, para el debido registro movilidad.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro movilidad</h4>
13             <form action="/movilidad" method="post">...
14         </div>
15     @endsection
16 @endif

```

### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > movilidad > edit.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información de la movilidad.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13                 <form action="/movilidad/{{ $movilidad->id }}" method="post">...
14             </div>
15         </div>
16     @endsection
17 @endif

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de una movilidad específica.

```
resources > views > movilidad > show.blade.php > ...
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">...
100
101
102     @endsection
103 @endif
```

#### 4.12.4 Export

Este archivo llamado ExportMovilidad contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase MovilidadesExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ():** Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ():** Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports > MovilidadesExport.php > ...
1 <?php
2 namespace App\Exports;
3
4 use Illuminate\Support\Facades\DB;
5 use Maatwebsite\Excel\Concerns\FromCollection;
6 use Maatwebsite\Excel\Concerns\WithHeadings;
7
8 class MovilidadesExport implements FromCollection, WithHeadings
9 {
10     public function headings(): array
11     {
12         return [
13             'Id', 'Fecha movilidad', 'Tipo documento docente',
14             'Número de documento docente', 'Nombre (s) docente',
15             'Apellido (s) docente', 'Tipo documento estudiante',
16             'Número de documento estudiante', 'Nombre (s) estudiante',
17             'Apellido (s) estudiante', 'Tipo de movilidad',
18             'Evento', 'País o ciudad',
19         ];
20     }
21     public function collection()
22     {
23         $movilidades = DB::table('movilidad')->select(
24             'movilidad.id', 'abrv',
25             'doc_numero_documento',
26             'doc_nombre', 'doc_apellido',
27             'abrv', 'estu_numero_documento',
28             'estu_nombre', 'estu_apellido',
29             'mov_tipo', 'mov_evento', 'mov_ciudad_pais',
30         )
31         ->join('docentes', 'movilidad.mov_id_docente', '=', 'docentes.id')
32         ->join('estudiantes', 'movilidad.mov_id_estudiante', '=', 'estudiantes.id')
33         ->join('tipo_documento', 'docentes.doc_tipo_documento', '=', 'tipo_documento.id')
34         ->get();
35
36         return $movilidades;
37     }
38 }
```

## 4.13 Investigación

### 4.13.1 Modelo

Los modelos utilizados para el desarrollo del software están ubicados en la carpeta **app/models**.

Los modelos a mostrarse creados para el desarrollo modelan una tabla de la base de datos y se le extiende una clase de **Eloquent** llamada **model** la cual nos permite interactuar con los datos de nuestra base de datos, los atributos que pueden ser accedidos se encuentran en nuestro arreglo **\$fillable**.

Por otra parte este modelo cuenta con unos métodos lo cuales establecen una relacionan de llaves primarias a foráneas con las tablas de las cuales requiere información a mostrar en la vista index.blade.php del módulo.

```
app > Models > Investigacion.php > Investigacion > $fillable
1  <?php
2  namespace App\Models;
3
4  use Illuminate\Database\Eloquent\Factories\HasFactory;
5  use Illuminate\Database\Eloquent\Model;
6
7  class Investigacion extends Model
8  {
9      use HasFactory;
10     protected $table = "investigacion";
11
12     protected $fillable = [
13         'id', 'inv_titulo_proyecto',
14         'inv_grupo_semillero', 'inv_director',
15         'inv_sede', 'inv_programa',
16         'inv_estado_proyecto', 'inv_fecha_inicio'
17     ];
18
19     public function docentes(){
20         return $this->belongsTo(Docente::class, 'inv_director');
21     }
22
23     public function sedes(){
24         return $this->belongsTo(Municipio::class, 'inv_sede');
25     }
26
27     public function programas(){
28         return $this->belongsTo(Programa::class, 'inv_programa');
29     }
30 }
31 }
```

### 4.13.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el

controlador con sus respectivos métodos creados para el funcionamiento del módulo investigación. Su ubicación es en la carpeta **App\Http\Controllers**.

```
app > Http > Controllers > InvestigacionController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Exports\InvestigacionesExport;
6  use App\Models\Docente;
7  use App\Models\Investigacion;
8  use App\Models\Municipio;
9  use App\Models\Programa;
10 use Illuminate\Http\Request;
11 use Illuminate\Support\Facades\Auth;
12 use Illuminate\Support\Facades\DB;
13 use Maatwebsite\Excel\Facades\Excel;
14 use RealRashid\SweetAlert\Facades\Alert;
15
16 class InvestigacionController extends Controller
17 {
18     public function index()
19     {
20         $investigaciones = Investigacion::all();
21         $programas = Programa::all();
22         if (Auth::user()->per_tipo_usuario == '3') {
23             return view('investigacion.index')->with('investigaciones', $investigaciones);
24         } else {
25             if ($programas->count() <= 0) {
26                 Alert::warning('Requisitos', 'Registre programas');
27                 return redirect('/home');
28             } else {
29                 return view('investigacion.index')->with('investigaciones', $investigaciones);
30             }
31         }
32     }
}
```

```

34     public function create()
35     {
36         $docentes = Docente::all();
37         $programas = Programa::all();
38         $sedes = Municipio::all();
39         return view('investigacion.create')->with('docentes', $docentes)
40             ->with('programas', $programas)
41             ->with('sedes', $sedes);
42     }
43
44     public function store(Request $request)
45     {
46         $investigaciones = new Investigacion();
47         $investigaciones->inv_titulo_proyecto = $request->get('inv_titulo_proyecto');
48         $investigaciones->inv_grupo_semillero = $request->get('inv_grupo_semillero');
49         $investigaciones->inv_director = $request->get('inv_director');
50         $investigaciones->inv_sede = $request->get('inv_sede');
51         $investigaciones->inv_programa = $request->get('inv_programa');
52         $investigaciones->inv_estado_proyecto = $request->get('inv_estado_proyecto');
53         $investigaciones->inv_fecha_inicio = $request->get('inv_fecha_inicio');
54
55         $rules = [
56             'inv_titulo_proyecto' => 'required',
57             'inv_grupo_semillero' => 'required',
58             'inv_estado_proyecto' => 'required',
59             'inv_fecha_inicio' => 'required'
60         ];
61
62         $messages = [
63             'inv_titulo_proyecto.required' => 'El campo titulo del proyecto es requerido',
64             'inv_grupo_semillero.required' => 'El campo grupo de semillero es requerido',
65             'inv_estado_proyecto.required' => 'El campo estado del proyecto es requerido',
66             'inv_fecha_inicio.required' => 'El campo fecha de inicio es requerido'
67         ];
68
69         $this->validate($request, $rules, $messages);

```

```

89     public function show($id)
90     {
91         $docentes = Docente::all();
92         $programas = Programa::all();
93         $sedes = Municipio::all();
94         $investigacion = Investigacion::find($id);
95         return view('investigacion.show')->with('docentes', $docentes)
96             ->with('programas', $programas)
97             ->with('sedes', $sedes)
98             ->with('investigacion', $investigacion);
99     }
100
101    public function edit($id)
102    {
103        $docentes = Docente::all();
104        $programas = Programa::all();
105        $sedes = Municipio::all();
106        $investigacion = Investigacion::find($id);
107        return view('investigacion.edit')->with('docentes', $docentes)
108            ->with('programas', $programas)
109            ->with('sedes', $sedes)
110            ->with('investigacion', $investigacion);
111    }

```

```

113     public function update(Request $request, $id)
114     {
115         $investigacion = Investigacion::find($id);
116         $investigacion->inv_titulo_proyecto = $request->get('inv_titulo_proyecto');
117         $investigacion->inv_grupo_semillero = $request->get('inv_grupo_semillero');
118         $investigacion->inv_director = $request->get('inv_director');
119         $investigacion->inv_sede = $request->get('inv_sede');
120         $investigacion->inv_programa = $request->get('inv_programa');
121         $investigacion->inv_estado_proyecto = $request->get('inv_estado_proyecto');
122         $investigacion->inv_fecha_inicio = $request->get('inv_fecha_inicio');

123         $investigacion->save();

124         DB::table('acciones_plataforma')->insert([
125             'usuario' => Auth::user()->id,
126             'accion' => 'editar',
127             'modulo' => 'investigacion'
128         ]);

129         Alert::success('Registro Actualizado');

130         return redirect('/investigacion');
131     }

132     public function destroy($id)
133     {
134         $investigacion = Investigacion::find($id);
135         $investigacion->delete();

136         DB::table('acciones_plataforma')->insert([
137             'usuario' => Auth::user()->id,
138             'accion' => 'eliminar',
139             'modulo' => 'investigacion'
140         ]);

141         Alert::success('Registro Eliminado');

142         return redirect('/investigacion');
143     }

144     public function pdf(Request $request)
145     {
146         $investigaciones = Investigacion::all();
147         if ($investigaciones->count() <= 0) {
148             Alert::warning('No hay registros');
149             return redirect('/investigacion');
150         } else {
151             $view = \view('investigacion.pdf', compact('investigaciones'))->render();
152             $pdf = \App::make('dompdf.wrapper');
153             $pdf->loadHTML($view);

154             DB::table('acciones_plataforma')->insert([
155                 'usuario' => Auth::user()->id,
156                 'accion' => 'pdf',
157                 'modulo' => 'investigacion'
158             ]);

159             return $pdf->stream('investigacion-reporte.pdf');
160         }
161     }

162     public function export()
163     {
164         $investigaciones = Investigacion::all();
165         if ($investigaciones->count() <= 0) {
166             Alert::warning('No hay registros');
167             return redirect('/investigacion');
168         } else {
169             DB::table('acciones_plataforma')->insert([
170                 'usuario' => Auth::user()->id,
171                 'accion' => 'excel',
172                 'modulo' => 'investigacion'
173             ]);

174             return Excel::download(new InvestigacionesExport, 'investigaciones.xlsx');
175         }
176     }

```

#### 4.13.3 Vistas

A continuación se muestran las vistas creadas para el modulo investigación.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```
resources > views > investigacion > index.blade.php
1 <?php
2     @if (!Auth::check())
3         @include('home')
4     @else
5         @extends('layouts.app')
6         @section('title')
7             <h1 class="titulo"><i class="fas fa-users"></i> Módulo investigación</h1>
8             @section('message')<p>Lista de registro investigación</p>@endsection
9         @endsection
10        @section('content')
11            <div class="container-fluid">
12                <div class="col-md-12 shadow-sm p-3 bg-white">
13                    <div class="card-body">...
14                </div>
15            </div>
16        @endsection
17    @endif
```

##### b. create.blade.php

Esta vista contiene un formulario, el cual permite realizar el registro de una nueva investigación.

```
resources > views > investigacion > create.blade.php
1 <?php
2     @if (!Auth::check())
3         @include('home')
4     @else
5         @extends('layouts.app')
6         @section('title')
7             <h1 class="titulo"><i class="fas fa-vector-square"></i> Formulario de registro</h1>
8             @section('message') <p>Diligenciar los campos requeridos, para el debido registro investigación.</p> @endsection
9         @endsection
10        @section('content')
11            <div class="container-fluid">
12                <div class="tile w-100">
13                    <h4 class="tile title"><i class="fab fa-wpforms"></i> Registro investigación</h4>
14                    <form action="/investigacion" method="post">...
15                </div>
16            </div>
17        @endsection
18    @endif
```

##### c. edit.blade.php

Esta vista contiene un formulario, el cual permite actualizar la información ya registrada de un registro específico.

```

resources > views > Investigacion > edit.blade.php
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información investigación.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">
12                 <h4 class="tile title">Actualizar información</h4>
13                 <form action="/investigacion/{{ $investigacion->id }}" method="post">...
130                </form>
131            </div>
132        </div>
133    @endsection
134 @endif

```

#### d. show.blade.php

Esta vista permite la visualización de los datos registrados, de una investigación específica.

```

resources > views > investigacion > show.blade.php
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-headset"></i> Vista registro</h1>
7          @section('message') <p>Información almacenada en el sistema</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile w-100">...
119            </div>
120        </div>
121    @endsection
122 @endif

```

#### 4.13.4 Export

Este archivo llamado ExportInvestigacion contiene los parámetros establecidos para generar el reporte en formato .xlsx.

La estructura de la clase InvestigacionsExport es la siguiente:

Contiene 2 métodos llamados heading () / collection ()

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

```
app > Exports >  InvestigacionesExport.php
 1  <?php
 2
 3  namespace App\Exports;
 4
 5  use DB;
 6  use Maatwebsite\Excel\Concerns\FromCollection;
 7  use Maatwebsite\Excel\Concerns\WithHeadings;
 8
 9  class InvestigacionesExport implements FromCollection, WithHeadings
10 {
11     public function headings(): array
12     {
13         return [
14             'Id', 'Titulo del proyecto',
15             'Grupo de semillero', 'Director',
16             'Sede', 'Programa',
17             'Estado', 'Fecha inicio',
18         ];
19     }
20     public function collection()
21     {
22         $investigaciones = DB::table('investigacion')->select([
23             'investigacion.id', 'inv_titulo_proyecto',
24             'inv_grupo_semillero',
25             DB::raw("CONCAT(doc_nombre, ' ', doc_apellido)"),
26             'mun_nombre', 'pro_nombre',
27             'inv_estado_proyecto', 'inv_fecha_inicio',
28         ])
29             ->join('docentes', 'investigacion.inv_director', '=', 'investigacion.id')
30             ->join('tipo_documento', 'docentes.doc_tipo_documento', '=', 'tipo_documento.id')
31             ->join('municipios', 'investigacion.inv_sede', '=', 'municipios.id')
32             ->join('programas', 'investigacion.inv_programa', '=', 'programas.id')
33             ->get();
34
35         return $investigaciones;
36     }
37 }
```

## 4.14 Egresados

### 4.14.1 Modelo

Este módulo está asociado al modelo estudiantes, donde nos permite mediante la relación entre entidades mostrar los registros de cada uno de los estudiantes.

Mediante el módulo nos da la opción de actualizar el estado de los estudiantes, es decir si es egresado o no y agregar fecha de grado.

Mediante el controlador **EgresadosController** se actualiza el estado del estudiante seleccionado. De igual manera cuenta con el archivo que contiene la clase **EgresadosExport** que permite generar el reporte de los estudiantes graduados.

La estructura de la clase **EgresadosExport** es la siguiente:

Contiene 2 métodos llamados **heading () / collection ()**

**heading ()**: Este método es el que indica el cabezal o fila superior del archivo Excel.

**collection ()**: Este método es el que realiza la solicitud a la base de datos para la obtención de los registros.

### 4.14.2 Controlador

Los controladores son los encargados de ordenar la lógica que se muestra en las vistas o las diferentes peticiones a la base de datos. A continuación, se muestra el controlador con sus respectivos métodos creados para el funcionamiento del módulo egresados. Su ubicación es en la carpeta **App\Http\Controllers**.

```

app > Http > Controllers > EgresadoController.php
1  <?php
2  namespace App\Http\Controllers;
3
4  use App\Exports\EgresadosExport;
5  use App\Models\Estudiante;
6  use App\Models\Programa;
7  use Illuminate\Http\Request;
8  use DB;
9  use Illuminate\Support\Facades\Auth;
10 use Maatwebsite\Excel\Facades\Excel;
11 use RealRashid\SweetAlert\Facades\Alert;
12
13 class EgresadoController extends Controller
14 {
15     public function index()
16     {
17         $estudiantes = Estudiante::all();
18         if (Auth::user()->per_tipo_usuario == '3') {
19             return view('egresado.index')->with('estudiantes', $estudiantes);
20         } else {
21             if ($estudiantes->count() <= 0) {
22                 Alert::warning('Requisitos', 'Registre estudiantes');
23                 return redirect('/home');
24             } else {
25                 return view('egresado.index')->with('estudiantes', $estudiantes);
26             }
27         }
28     }
29
30     public function edit($id)
31     {
32         $estudiante = Estudiante::find($id);
33         return view('egresado.edit')->with('estudiante', $estudiante);
34     }

```

```

36     public function update(Request $request, $id)
37     {
38         $estudiante = Estudiante::find($id);
39         $estudiante->estu_egresado = $request->get('estu_egresado');
40         $estudiante->estu_grado = $request->get('estu_grado');
41
42         $rules = [
43             'estu_grado' => 'required'
44         ];
45
46         $message = [
47             'estu_grado.required' => 'El campo fecha de grado es requerido'
48         ];
49
50         $this->validate($request, $rules, $message);
51
52         if ($request->get('estu_egresado') == "") {
53             Alert::warning('El valor seleccionado no es correcto');
54             return back()->withInput();
55         }
56
57         $estudiante->save();
58
59         Alert::success('Estado Actualizado');
60         return redirect('/egresado');
61     }

```

```

63     public function pdf()
64     {
65         $estudiantes = DB::table('estudiantes')
66             ->join('tipo_documento', 'estudiantes.estu_tipo_documento', '=', 'tipo_documento.id')
67             ->join('programas', 'estudiantes.estu_programa', '=', 'programas.id')
68             ->where('estu_egresado', 1)
69             ->get();
70         if ($estudiantes->count() <= 0) {
71             Alert::warning('No hay registros');
72             return redirect('/egresado');
73         } else {
74             $view = \view('egresado.pdf', compact('estudiantes'))->render();
75             $pdf = \App::make('dompdf.wrapper');
76             $pdf->setPaper('A4', 'landscape');
77             $pdf->loadHTML($view);
78             return $pdf->stream('egresados-report.pdf');
79         }
80     }
81
82     public function export()
83     {
84         $estudiantes = DB::table('estudiantes')->where('estu_egresado', '=', '1');
85         if ($estudiantes->count() <= 0) {
86             Alert::warning('No hay registros');
87             return redirect('/egresado');
88         } else {
89             return Excel::download(new EgresadosExport, 'egresados.xlsx');
90         }
91     }
92 }

```

#### 4.14.3 Vistas

A continuación se muestran las vistas creadas para el modulo egresados.

##### a. Index.blade.php

Esta vista contiene el listado de los registros realizados por el o los administradores, almacenados en la base de datos. Por otra parte nos permite agregar nuevos registros mediante el botón ubicado arriba de la barra de búsqueda nuevo, además de actualizar y eliminar un registro almacenado.

```

resources > views > egresado > index.blade.php
1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-users"></i> Módulo egresados</h1>
7          @section('message')<p>Lista de registro egresados</p>@endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="col-md-12 shadow-sm p-3 bg-white">
12                 ...
13             </div>
14         </div>
15     @endsection
16 @endif

```

##### b. edit.blade.php

Mediante la vista edit.blade.php se actualiza el estado del estudiante egresado.

```

1  @if (!Auth::check())
2      @include('home')
3  @else
4      @extends('layouts.app')
5      @section('title')
6          <h1 class="titulo"><i class="fas fa-edit"></i> Formulario de edición de datos</h1>
7          @section('message') <p>Actualizar información del estudiante.</p> @endsection
8      @endsection
9      @section('content')
10         <div class="container-fluid">
11             <div class="tile card__config">
12                 <h4 class="tile title">Actualizar información</h4>
13             <form action="/egresado/{{ $estudiante->id }}" method="POST">...
14             </form>
15         </div>
16     @endsection
17     @endif

```

#### 4.14.4 Export

```

app > Exports > EgresadosExport.php
1  <?php
2  namespace App\Exports;
3
4  use DB;
5  use Maatwebsite\Excel\Concerns\FromCollection;
6  use Maatwebsite\Excel\Concerns\WithHeadings;
7
8  class EgresadosExport implements FromCollection, WithHeadings
9  {
10     public function headings(): array
11     {
12         return [
13             'Id','Estado estudiante','Tipo de documento','Número de documento',
14             'Nombre (s)','Apellido (s)','Telefono 1','Telefono 2',
15             'Dirección','Correo electrónico','Estrato','Departamento','Municipio',
16             'Fecha de nacimiento','Año de ingreso','Último periodo inscrito',
17             'Semestre','Tipo de financiamiento','Beca','Estado',
18             'Matrícula','Promedio general acumulado','Fecha grado','Reconocimientos'
19         ];
20     }
21     public function collection()
22     {
23         $estudiantes = DB::table('estudiantes')->select(
24             'estudiantes.id','pro_nombre','abr','estu_numero_documento',
25             'estu_nombre','estu_apellido','estu_telefono1','estu_telefono2',
26             'estu_direccion','estu_correo','estu_estrato','dep_nombre',
27             'mun_nombre','estu_fecha_nacimiento','estu_ingreso','estu_ult_periodo',
28             'estu_semestre','estu_financiamiento','estu_beca',
29             'est_nombre','estu_matricula','estu_pga','estu_grado','estu_reconocimiento',
30         )
31         ->join('programas', 'estudiantes.estu_programa', '=', 'programas.id')
32         ->join('tipo_documento', 'estudiantes.estu_tipo_documento', '=', 'tipo_documento.id')
33         ->join('departamentos', 'estudiantes.estu_departamento', '=', 'departamentos.id')
34         ->join('municipios', 'estudiantes.estu_municipio', '=', 'municipios.id')
35         ->join('estado_programas', 'estudiantes.estu_estado', '=', 'estado_programas.id')
36         ->where('estu_egresado', 1)
37         ->get();
38
39         return $estudiantes;

```