

LARAVEL 1

Profesores y asignaturas



José Antonio García Torrecillas
2ºDAW

Contenido

| | |
|---|----|
| 1.- TABLAS | 2 |
| 2.-CREAR EL PROYECTO LARAVEL Y BASE DE DATOS SQLITE | 2 |
| 2.1- CREAR PROYECTO, MODELOS Y COMANDOS | 2 |
| 2.2- BASE DE DATOS SQLITE | 3 |
| 2.3- RETOQUES Y CAMBIO DE IDIOMA EN EL PROYECTO | 3 |
| 3.- TABLA PROFESOR | 4 |
| 3.1- FACTORY | 4 |
| 3.2- MIGRATION | 4 |
| 3.3- SEEDER | 5 |
| 3.4-INDEX | 8 |
| 3.4.1.-Controlador | 9 |
| 3.5-CREATE | 9 |
| 3.5.1.-Controlador | 10 |
| 3.6-UPDATE | 10 |
| 3.6.1.-Controlador | 11 |
| 3.7-DELETE..... | 12 |
| 3.7.1.-Controlador | 12 |
| 3.8-SHOW | 12 |
| 3.8.1.-Controlador | 13 |
| 4.- TABLA ASIGNATURA..... | 13 |
| 4.1- FACTORY..... | 13 |
| 4.2- MIGRATION | 14 |
| 4.3- SEEDER | 14 |
| 4.4- INDEX | 15 |
| 4.4.1.-Controlador | 16 |
| 4.5- CREATE | 16 |
| 4.5.1.-Controlador | 17 |
| 4.6- UPDATE | 17 |
| 4.6.1.-Controlador | 18 |
| 4.7- DELETE..... | 19 |
| 4.7.1.-Controlador | 19 |
| 4.8- SHOW | 20 |
| 4.8.1.-Controlador | 20 |
| 5.- ROUTES Y LINKS DE INTERES | 20 |
| 5.1- LINKS DE INTERES..... | 20 |

1.- TABLAS

Creamos las tablas profesores y asignaturas (1:N). Donde profesor (nombre, apellidos, email único y localidad) tiene varias asignaturas (nombre, descripción, créditos int).

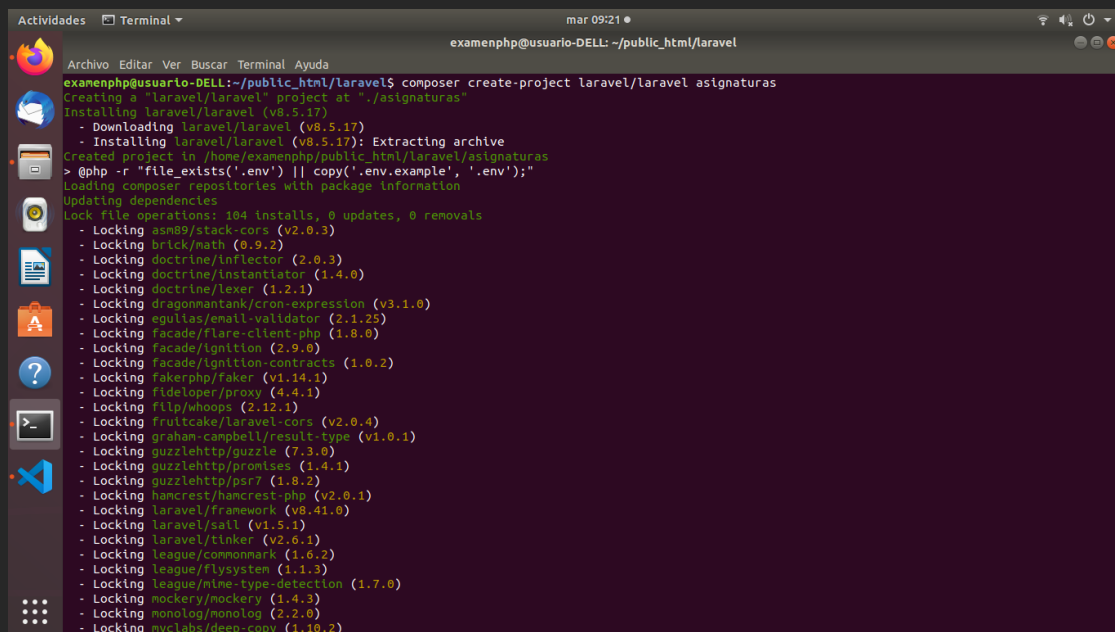
```
profesores(nombre, apellidos, email único, localidad)
asignaturas(nombre, descripcion, créditos unsigned int)
```

2.-CREAR EL PROYECTO LARAVEL Y BASE DE DATOS SQLITE

2.1- CREAR PROYECTO, MODELOS Y COMANDOS

Dentro de cualquier carpeta desde el terminal escribiremos

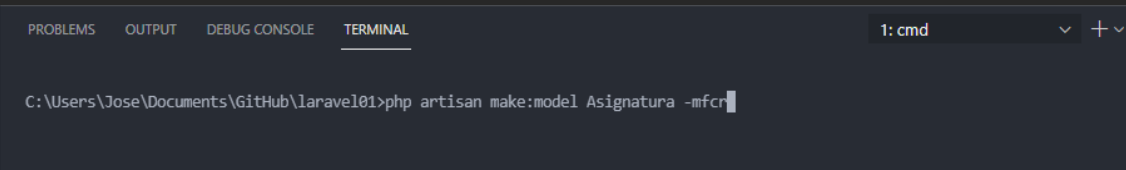
```
composer create-project laravel/laravel proyecto
```



```
Actividades Terminal mar 09:21
examenphp@usuario-DELL: ~/public_html/laravel$ composer create-project laravel/laravel asignaturas
Creating a "laravel/laravel" project at "./asignaturas"
Installing laravel/laravel (v8.5.17)
- Downloading laravel/laravel (v8.5.17)
- Installing laravel/laravel (v8.5.17): Extracting archive
Created project in /home/examenphp/public_html/laravel/asignaturas
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 104 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.0.3)
- Locking brick/math (0.9.2)
- Locking doctrine/inflector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)
- Locking dragonmantank/cron-expression (v3.1.0)
- Locking egulias/email-validator (2.1.25)
- Locking facade/flare-client-php (1.8.0)
- Locking facade/ignition (2.9.0)
- Locking facade/ignition-contracts (1.0.2)
- Locking fakerphp/faker (v1.14.1)
- Locking fideloper/proxy (4.4.1)
- Locking filp/whoops (2.12.1)
- Locking fruitcake/laravel-cors (v2.0.4)
- Locking graham-campbell/result-type (v1.0.1)
- Locking guzzlehttp/guzzle (7.3.0)
- Locking guzzlehttp/promises (1.4.1)
- Locking guzzlehttp/psr7 (1.8.2)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v8.41.0)
- Locking laravel/sail (v1.5.1)
- Locking laravel/tinker (v2.6.1)
- Locking league/commonmark (1.6.2)
- Locking league/flysystem (1.1.3)
- Locking league/mime-type-detection (1.7.0)
- Locking mockery/mockery (1.4.3)
- Locking monolog/monolog (2.2.0)
- Locking myclabs/deep-copy (1.10.2)
```

Crearemos los modelos asignatura y profesor con sus controladores, factories, seeders

```
php artisan make:model Objeto -mfc
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: cmd
C:\Users\Jose\Documents\GitHub\laravel01>php artisan make:model Asignatura -mfc
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  1: cmd  +
C:\Users\Jose\Documents\GitHub\laravel01>php artisan make:model Profesor -mfc
```

2.2- BASE DE DATOS SQLITE

Dentro de la carpeta database crearemos el archivo **database.sqlite** para que nuestro archivo .env reconozca la base de datos sqlite y así poder usarla sin complicaciones

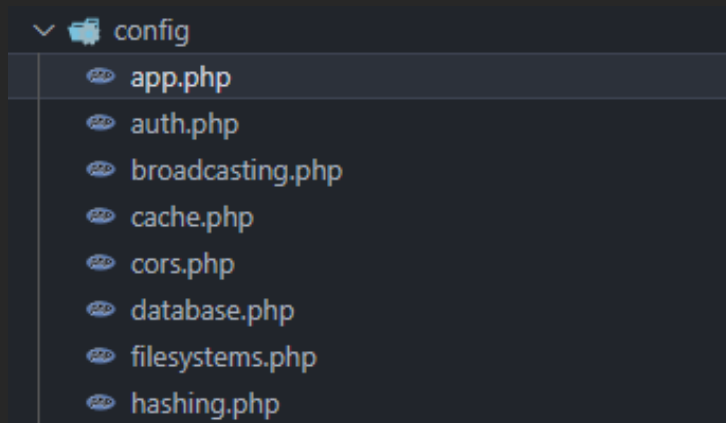
```
laravel > asignaturas > database > database.sqlite
1
```

Dentro del archivo .env modificaremos los parámetros del db y pondremos lo siguiente

```
laravel > asignaturas > .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:RivppuRYzaNtd4Hm
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_LEVEL=debug
9
10 DB_CONNECTION=sqlite
11 DB_FOREIGN_KEYS=true
12
```

2.3- RETOQUES Y CAMBIO DE IDIOMA EN EL PROYECTO

Para que nuestro proyecto no tenga problemas a la hora de crear datos, tenemos que cambiar los siguientes campos. Del inglés al español



```
'locale' => 'es',
```

```
'faker_locale' => 'es_ES',
```

3.- TABLA PROFESOR

3.1- FACTORY

Insertaremos los atributos de profesor en su archivo factory. Factory se encarga de crear aquellos objetos profesor con atributos aleatorios.

'atributo' => \$this->faker->metodoCualquiera

Métodos de faker

```
public function definition()
{
    return [
        'nombre'=>$this->faker->firstName(),
        'apellidos'=>$this->faker->lastName(),
        'email'=>$this->faker->unique()->freeEmail(),
        'localidad'=>$this->faker->city
    ];
}
```

laravel01 / database / factories / ProfesorFactory.php



3.2- MIGRATION

Insertaremos los atributos de profesor en su archivo migration. Migration se encarga de crear la tabla profesor con los atributos recogidos de factory. En la creación de la tabla se crea su id autoincrementado,

nombre, apellidos, email (que va a ser único para establecer la relación con la tabla asignatura), localidad.

```
$table->string/ integer('nombreDelAtributo');
```

Migrations y sus declaraciones

```
public function up()
{
    Schema::create('profesors', function (Blueprint $table) {
        $table->id();
        $table->string('nombre');
        $table->string('apellidos');
        $table->string('email')->unique();
        $table->string('localidad');
        $table->timestamps();
    });
}
```

laravel01 / database / migrations / 2021_05_21_110208_create_profesors_table.php



3.3- SEEDER

En el factory para que inserte 5 objetos de tipo profesor dentro de la tabla proporcionada por migration.

```
\App\Models\Objeto::factory(5)->create();
```

```
public function run()
{
    \App\Models\Profesor::factory(5)->create();
}
```

laravel01 / database / seeders / DatabaseSeeder.php



Una vez hecho las migraciones, seeders y factories. Ejecutamos la migración

```
php artisan migrate:fresh --seed
```

```

C:\Users\Jose\Documents\GitHub\laravel01>php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (129.01ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (31.77ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (41.12ms)
Migrating: 2021_05_18_072859_create_asignaturas_table
Migrated: 2021_05_18_072859_create_asignaturas_table (88.88ms)
Database seeding completed successfully.

```

Antes de proceder debemos descargar la librería para que me traiga los x-form y los x-select para que funcionen los campos correctamente dentro de .blade.php en views. Con este comando:

composer require protonemia/laravel-form-components

```

C:\Users\Jose\Documents\GitHub\laravel01>composer require protonemia/laravel-form-components
Using version ^2.5 for protonemia/laravel-form-components
./composer.json has been updated
Running composer update protonemia/laravel-form-components
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking protonemia/laravel-form-components (2.5.4)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading protonemia/laravel-form-components (2.5.4)
- Installing protonemia/laravel-form-components (2.5.4): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/sail
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: protonemia/laravel-form-components
Package manifest generated successfully.
75 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

C:\Users\Jose\Documents\GitHub\laravel01>

```

Una vez traído accedemos al índice 10 (formcomponent), con el comando

php artisan vendor:publish

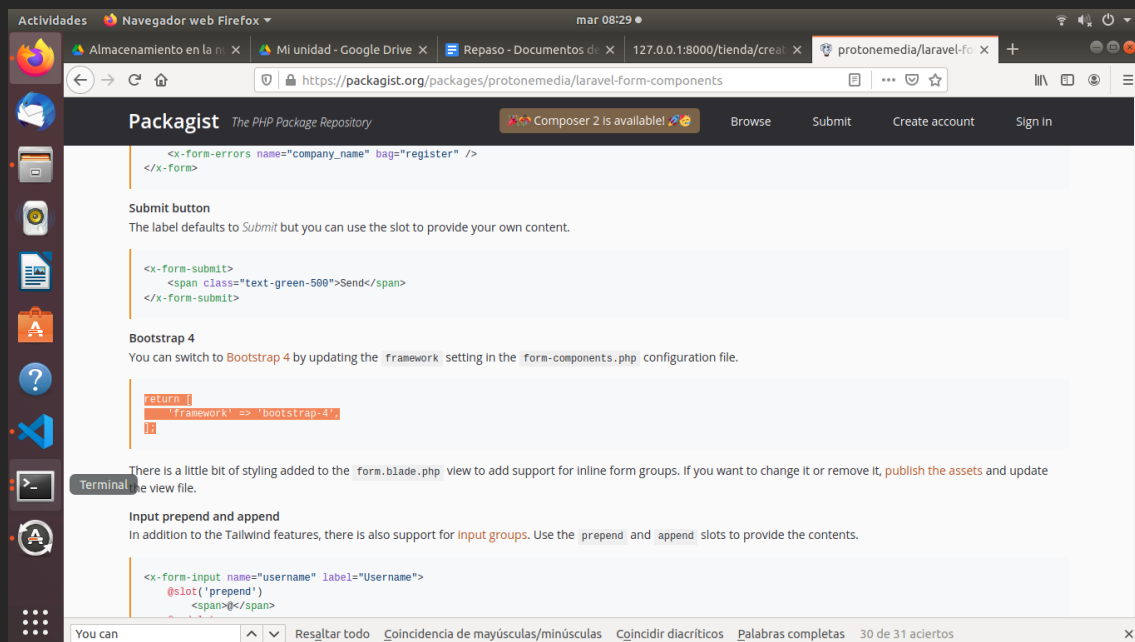
```
C:\Users\Jose\Documents\GitHub\laravel01>php artisan vendor:publish

Which provider or tag's files would you like to publish?:
[0 ] Publish files from all providers and tags listed below
[1 ] Provider: Facade\Ignition\IgnitionServiceProvider
[2 ] Provider: Fideloper\Proxy\TrustedProxyServiceProvider
[3 ] Provider: Fruitcake\Cors\CorsServiceProvider
[4 ] Provider: Illuminate\Foundation\Providers\FoundationServiceProvider
[5 ] Provider: Illuminate\Mail\MailServiceProvider
[6 ] Provider: Illuminate\Notifications\NotificationServiceProvider
[7 ] Provider: Illuminate\Pagination\PaginationServiceProvider
[8 ] Provider: Laravel\Sail\SailServiceProvider
[9 ] Provider: Laravel\Tinker\TinkerServiceProvider
[10] Provider: ProtoneMedia\LaravelFormComponents\Support\ServiceProviders
[11] Tag: config
[12] Tag: cors
[13] Tag: flare-config
[14] Tag: ignition-config
[15] Tag: laravel-errors
[16] Tag: laravel-mail
[17] Tag: laravel-notifications
[18] Tag: laravel-pagination
[19] Tag: sail
[20] Tag: views
> 10

Copied File [\vendor\protonemedia\laravel-form-components\config\config.php] To [\config\form-components.php]
Copied Directory [\vendor\protonemedia\laravel-form-components\resources\views] To [\resources\views\vendor\form-components]
Publishing complete.

C:\Users\Jose\Documents\GitHub\laravel01>
```

Para terminar con la instalación de bootstrap 4 dirigiéndonos a la carpeta siguiente y cambiamos de tailwind a bootstrap-4 (formcomponent), con el comando.



laravel01 / config / form-components.php

'framework' => 'bootstrap-4',

framework

3.4-INDEX

En este código se mostrará la tabla profesores con los atributos nombre, apellidos, email, localidad con los botones editar, borrar, mostrar con el botón crear. Con la paginación gracias a los links.



```
@foreach($profesor as $item)

<tr>

    <th scope="row">

        <a href="{{route('profesor.show', $item)}}" class="btn btn-info"><i
class="fas fa-info-circle"> Detalle</i>

    </th>

    <td>{{$item->nombre}}</td>

    <td>{{$item->apellidos}}</td>

    <td>{{$item->email}}</td>

    <td>{{$item->localidad}}</td>

    <td class="center">

        <a href="{{route('profesor.edit', $item)}}" class="btn btn-warning"><i
class="fas fa-edit"> Actualizar</i></a>

    </td>

    <td class="center">

        <form name="as" method="POST" action="{{route('profesor.destroy', $item)}}">

            @csrf

            @method("DELETE")

            <button type="submit" class="btn btn-danger"><i class="fas fa-trash">
Borrar</i></button>

        </form>

    </td>

</tr>

@endforeach
```

Los links para paginar se deben hacer lo siguiente después de crear la tabla

```
{{ $profesor->links() }}
```

Debemos escribir **Paginator::useBootstrap();** para que traiga perfectamente la paginación de las páginas entre tablas.

laravel01 / app / Providers / AppServiceProvider.php

```
public function boot()
```

```
{
    Paginator::useBootstrap();
}
```



3.4.1.-Controlador

Controlador para mostrar la tabla ordenada por nombre y que muestre 10 filas por cada paginación y que devuelva la vista con los profesores creados gracias a sqlite.

```
public function index()
{
    $profesor = Profesor::orderBy('nombre')->paginate(10);
    return view('profesor.index', compact('profesor'));
}
```



3.5-CREATE

Para ello, dedicaremos un botón para que nos lleve a una vista nueva donde index.php

```
<a href="{{route('profesor.create')}}" class="mt-3 btn btn-light"><i
class="fas fa-plus-square"></i> Crear Profesor</a>
```



En este código se mostrará los campos con los atributos. Los atributos nombre, apellidos, email, localidad. Con el formulario de tipo POST y llamando al método store del controlador profesor

```
<form name="sd" method="POST" action="{{route('profesor.store')}}" class="p-4
text-light">

@csrf

<x-form-input name="nombre" label="Nombre profesor" />
<x-form-input name="apellidos" label="Apellidos" />
<x-form-input name="email" label="Email" />
<x-form-input name="localidad" label="Localidad" />
```

```

<div class="mt-3">

    <button type="submit" class="mt-3 btn btn-info"><i class="fas fa-
save"></i>Enviar</button>

    <button type="reset" class="mt-3 btn btn-warning"><i class="fas fa-
broom"></i>Limpiar</button>

    <a href="{{route('profesor.index')}}" class='btn btn-primary mt-3'><i class="fas
fa-undo"></i>Volver</a>

</div>

</form>

```



3.5.1.-Controlador

Controlador para verificar si la información traída de los campos es válida y pueda crear un objeto profesor sin problemas.

```

public function store(Request $request)
{
    $request->validate([
        "nombre" => ['required', 'string', 'min:3', 'max:30'],
        "apellidos" => ['required', 'string', 'min:3', 'max:50'],
        "email" => ['required', 'string', 'unique:profesors,email'],
        "localidad" => ['required', 'string', 'min:3', 'max:30']
    ]);
    try {
        Profesor::create($request->all());
    } catch (Exception) {
        return redirect()->route('profesor.index');
    }
    return redirect()->route('profesor.index');
}

```



3.6-UPDATE

Para ello, dedicaremos un botón para que nos lleve a una vista nueva desde index.php. Cada fila traerá el id del profesor recogido desde dicha fila donde se pulso el botón editar \$item

```
<a href="{{route('profesor.edit', $item)}}" class="btn btn-warning"><i class="fas fa-edit"> Actualizar</i></a>
```



En este código se mostrará los campos con los atributos. Los atributos nombre, apellidos, email, localidad. Con la diferencia de que el formulario tendrá el método PUT con el \$item traído desde index.blade.php. Y nos traerá la información gracias al método @bind(\$item) si coincide con el nombre de los form con los atributos de cada profesor

```
<form name="sd" method="POST" action="{{route('profesor.update', $profesor)}}"
class="p-4 text-light">

    @csrf

    @method("PUT")

    @bind($profesor)

    <x-form-input name="nombre" label="Nombre profesor" />

    <x-form-input name="apellidos" label="Apellidos" />

    <x-form-input name="email" label="Email" />

    <x-form-input name="localidad" label="Localidad" />


    <div class="mt-3">

        <button type="submit" class="mt-3 btn btn-info"><i class="fas fa-edit"></i>Actualizar</button>

        <button type="reset" class="mt-3 btn btn-warning"><i class="fas fa-broom"></i>Limpiar</button>

        <a href="{{route('profesor.index')}}" class="btn btn-primary mt-3"><i class="fas fa-undo"></i>Volver</a>

    </div>

</form>
```



3.6.1.-Controlador

Controlador para verificar si la información traída de los campos es válida y pueda updatear un objeto profesor con profesor traído por \$item sin problemas.

```
public function update(Request $request, Profesor $profesor)
{
    $request->validate([
        "nombre" => ['required', 'string', 'min:3', 'max:30'],
        "apellidos" => ['required', 'string', 'min:3', 'max:50'],
        "email" => ['required', 'string', 'unique:profesors,email'.$profesor->id],
    ];
```

```

        "localidad" => ['required', 'string', 'min:3', 'max:50']
    ]);
    try{
        $profesor->update($request->all());
    }catch(Exception){
        return redirect()->route('profesor.index');
    }
    return redirect()->route('profesor.index');
}

```



3.7-DELETE

Para ello, dedicaremos un botón a cada fila y borrara aquella fila seleccionada con dicho botón.

```

@csrf

@method("DELETE")

<button type="submit" class="btn btn-danger"><i class="fas fa-trash">
Borrar</i></button>

```



3.7.1.-Controlador

```

public function destroy(Profesor $profesor){
    try{
        $profesor->delete();
    }catch(Exception){
        return redirect()->route('profesor.index');
    }
    return redirect()->route('profesor.index');
}

```



3.8-SHOW

Para ello, dedicaremos un botón para que nos lleve a una vista nueva desde index.php. Cada fila traerá el id del profesor recogido desde dicha fila donde se pulso el botón show \$item

```

<a href="{{route('profesor.show', $item)}}" class="btn btn-info"><i class="fas
fa-info-circle"> Detalle</i>

```



3.8.1.-Controlador

Para enviarnos del index al show gracias al id traído desde index
\$profesor

```
public function show(Profesor $profesor)
{
    return view('profesor.show', compact('profesor'));
}
```



4.- TABLA ASIGNATURA

4.1- FACTORY

Insertaremos los atributos de asignatura en el archivo factory. Factory se encarga de crear aquellos objetos asignatura con atributos aleatorios.

Con la diferencia de que tenemos que traernos todos los profesores con sus ids y llamarlos por cada asignatura (profesor_id)

'atributo' => \$this->faker->metodoCualquiera

Métodos de faker

```
public function definition()
{
    $profesors=Profesor::all('id');
    return [
        'nombre'=>$this->faker->unique()->randomElement(['Matematicas', 'Lengua',
        'Geografia', 'Ciencias', 'Informatica', 'Economia', 'Base de datos', 'Astronomia',
        'Teatro', 'Musica']),
        'descripcion'=>$this->faker->text(),
        'creditos'=>$this->faker->numberBetween(1,250),
        'profesor_id'=>$profesors->get(rand(0, count($profesors)-1))
    ];
}
```

laravel01 / database / factories / AsignaturaFactory.php



4.2- MIGRATION

Insertaremos los atributos de asignatura en su archivo migration. Migration se encarga de crear la tabla asignatura con los atributos recogidos de factory. En la creación de la tabla se crea su id autoincrementado, nombre, descripción, créditos, profesor_id.

En profesor_id haremos referencia al id del profesor y el borrado y la actualización lo haremos a cascada.

```
$table->string('nombreDelAtributo');
```

Migrations y sus declaraciones

```
public function up()
{
    Schema::create('asignaturas', function (Blueprint $table) {
        $table->id();
        $table->string('nombre')->unique();
        $table->string('descripcion');
        $table->integer('creditos')->unsigned();
        $table->foreignId('profesor_id');
        $table->foreign('profesor_id')
            ->references("id")->on('profesors')
            ->onDelete("cascade")->onUpdate('cascade');
        $table->timestamps();
    });
}
```

laravel01 / database / migrations / 2021_05_18_072859_create_asignaturas_table.php



4.3- SEEDER

En el factory para que inserte 5 objetos de tipo profesor dentro de la tabla proporcionada por migration.

```
\App\Models\Objeto::factory(5)->create();
```

```
\App\Models\Asignatura::factory(10)->create();
```

laravel01 / database / seeders / DatabaseSeeder.php



Una vez hecho las migraciones, seeders y factorys. Ejecutamos la migración

```
php artisan migrate:fresh --seed
```

4.4- INDEX

En este código se mostrará la tabla asignaturas con los atributos nombre, descripción, créditos con los botones editar, borrar, mostrar con el botón crear. Con la paginación gracias a los links (que se vieron anteriormente)

```
@foreach($asignatura as $item)

<tr>

    <th scope="row">

        <a href="{{route('asignatura.show', $item)}}" class="btn btn-info"><i
class="fas fa-info-circle"> Detalle</i>

    </th>

    <td>{{$item->nombre}}</td>

    <td>{{$item->descripcion}}</td>

    <td>{{$item->creditos}}</td>

    <td class="center">

        <a href="{{route('asignatura.edit', $item)}}" class="btn btn-warning"><i
class="fas fa-edit"> Actualizar</i></a>

    </td>

    <td class="center">

        <form name="as" method="POST" action="{{route('asignatura.destroy',
$item)}}">

            @csrf

            @method("DELETE")

            <button type="submit" class="btn btn-danger"><i class="fas fa-trash">
Borrar</i></button>

        </form>

    </td>

</tr>

@endforeach
```

laravel01 / resources / views / asignatura / index.blade.php



4.4.1.-Controlador

Controlador para mostrar la tabla ordenada por nombre y que muestre 5 filas por cada paginación y que devuelva la vista con las asignaturas creadas gracias a sqlite.

```
public function index()
{
    $asignatura=Asignatura::orderBy('nombre')->paginate(5);
    return view('asignatura.index', compact('asignatura'));
}
```

laravel01 / app / Http / Controllers / AsignaturaController.php



4.5- CREATE

Para ello, dedicaremos un botón para que nos lleve a una vista nueva donde index.php

```
<a href="{{route('asignatura.create')}}" class="mt-3 btn btn-light"><i class="fas fa-plus-square"></i> Crear Asignatura</a>
```



En este código se mostrará los campos con los atributos. Los atributos nombre, descripción, créditos. Con el formulario de tipo POST y llamando al método store del controlador asignatura

```
<form name="sd" method="POST" action="{{route('asignatura.store')}}" class="p-4 text-light">

    @csrf

    <x-form-input name="nombre" label="Nombre asignatura" />
    <x-form-input name="descripcion" label="Descripcion" />
    <x-form-input name="creditos" label="Creditos" />
    <x-form-select name="profesor_id" :options="$profesors" label="Profesor"/>


    <div class="mt-3">

        <button type="submit" class="mt-3 btn btn-info"><i class="fas fa-save"></i>Enviar</button>

        <button type="reset" class="mt-3 btn btn-warning"><i class="fas fa-broom"></i>Limpiar</button>
```

```

        <a href="{{route('asignatura.index')}}" class='btn btn-primary mt-3'><i
class="fas fa-undo"></i>Volver</a>

    </div>

</form>

```

laravel01 / resources / views / asignatura / create.blade.php



4.5.1.-Controlador

Controlador para verificar si la información traída de los campos es válida y pueda crear un objeto asignatura sin problemas

```

public function store(Request $request)
{
    $request->validate([
        "nombre"=>['required','string','min:3', 'max:12',
'unique:asignaturas,nombre'],
        "descripcion"=>['required','string','min:3', 'max:300'],
        "creditos"=>['required','integer','digits_between:1,250'],
        'profesor_id' => ['required']
    ]);
    try{
        Asignatura::create($request->all());
    }catch(Exception){
        return redirect()->route('asignatura.index');
    }
    return redirect()->route('asignatura.index');
}

```

laravel01 / app / Http / Controllers / AsignaturaController.php



4.6- UPDATE

Para ello, dedicaremos un botón para que nos lleve a una vista nueva. Cada fila traerá el id de asignatura recogido desde dicha fila donde se pulso el botón editar \$item

```

        <a href="{{route('asignatura.edit', $item)}}" class="btn btn-
warning"><i class="fas fa-edit"> Actualizar</i></a>

```



En este código se mostrará los campos con los atributos. Los atributos nombre, descripción, créditos. Con la diferencia de que el formulario tendrá el método PUT con el \$item traído desde index.blade.php. Y nos traerá la información gracias al método

@bind(\$item) si coincide con el nombre de los form con los atributos de cada asignatura

```
<form name="sd" method="POST" action="{{route('asignatura.update',
$asignatura)}}" class="p-4 text-light">

    @csrf

    @method("PUT")

    @bind($asignatura)

    <x-form-input name="nombre" label="Nombre asignatura" />

    <x-form-input name="descripcion" label="Descripcion" />

    <x-form-input name="creditos" label="Creditos" />

    <x-form-select name="profesor_id" :options="$profesors" label="Profesor"/>


    <div class="mt-3">

        <button type="submit" class="mt-3 btn btn-info"><i class="fas fa-
edit"></i>Actualizar</button>

        <button type="reset" class="mt-3 btn btn-warning"><i class="fas fa-
broom"></i>Limpiar</button>

        <a href="{{route('asignatura.index')}}" class='btn btn-primary mt-3'><i
class="fas fa-undo"></i>Volver</a>

    </div>

</form>
```

laravel01 / resources / views / asignatura / edit.blade.php



4.6.1.-Controlador

Controlador para verificar si la información traída de los campos es válida y pueda updatear un objeto profesor con asignatura traído por \$item sin problemas.

```
public function update(Request $request, Asignatura $asignatura)
{
    $request->validate([
        "nombre"=>['required','string','min:3', 'max:12',
'unique:asignaturas,nombre'.$asignatura->id],
        "descripcion"=>['required','string','min:3', 'max:300'],
        "creditos"=>['required','integer','digits_between:1,250'],
        'profesor_id' => ['required']
    ]);
}
```

```

try{
    $asignatura->update($request->all());
}catch(Exception){
    return redirect()->route('asignatura.index');
}
return redirect()->route('asignatura.index');
}

```

laravel01 / app / Http / Controllers / AsignaturaController.php



4.7- DELETE

Para ello, dedicaremos un botón a cada fila y borrara aquella fila seleccionada con dicho botón.

```

<form name="as" method="POST" action="{{route('asignatura.destroy',
$item)}}">
    @csrf
    @method("DELETE")
    <button type="submit" class="btn btn-danger"><i class="fas fa-trash">
Borrar</i></button>
</form>

```

laravel01 / resources / views / asignatura / index.blade.php



4.7.1.-Controlador

```

public function destroy(Asignatura $asignatura)
{
    try{
        $asignatura->delete();
    }catch(Exception){
        return redirect()->route('asignatura.index');
    }
    return redirect()->route('asignatura.index');
}

```

laravel01 / app / Http / Controllers / AsignaturaController.php



4.8- SHOW

Para ello, dedicaremos un botón para que nos lleve a una vista nueva desde index.php. Cada fila traerá el id del profesor recogido desde dicha fila donde se pulso el botón show \$item

```
<a href="{{route('asignatura.show', $item)}}" class="btn btn-info"><i class="fas fa-info-circle"> Detalle</i>
```



4.8.1.-Controlador

Para enviarnos del index al show gracias al id traído desde index \$asignatura

```
public function show(Asignatura $asignatura)
{
    return view('asignatura.show', compact('asignatura'));
}
```

laravel01 / app / Http / Controllers / AsignaturaController.php



5.- ROUTES Y LINKS DE INTERES

Para que recoja la información de las vistas y proyectarlas a través de estas rutas

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

Route::resource('asignatura', AsignaturaController::class);
Route::resource('profesor', ProfesorController::class);
```

laravel01 / routes / web.php



5.1- LINKS DE INTERES

[Packagist](#)

[Laravel-lang \(idiomas\)](#)

[Validation](#)

[Bootstrap](#)