PROIEKTU BAT SORTU - AULA ADIBIDEA

1. Paginazioa

Laravel bere 9. bertsioan paginazioa egiteko oso modu erraza sortu zuen. Oraingoan gure ikasleak jasotzerakoan all() metodoari deitu beharrean, paginate() metodoari deituko diogu

1. AlumnosControler, index metodoan all() metodoa paginte() metodoarekin aldatuko dugu:

```
$alumnos = Alumno::paginate();
```

Aurrekoarekin soilik esan diogu gure ikasleak taldeka pasatzea, berez 15-naka pasatuko ditu datu baseko erregistroak, gure kasuan ikasleenak.

Baina hori aldatu dezakegu, paginate metodoari parametroz zenbat erregistro jaso nahi ditugu esanez:

```
$alumnos = Alumno::paginate(10);
```

Beste metodo bat erabili dezakeguna orderBy izango da, hau beste metodoekin eraili dezakegu, adibidez:

```
$alumnos = Alumno::orderBy('id','desc')->paginate();
```

2. indez.blade.php bistan paginazioaren linkak jarriko ditugu hurrengo direktiba erabiliz::

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

3. Ikusi dezakegunez, paginazioa eginda dago jada, baina sortzen dituen botoiek eta estekek Tailwineko klaseak erabiltzen dituzte, beraz pizkatxo bat txukunago jartzeko https://tailwindcss.com/ webgunera sartuko gara eta "Installation" barruan hartuko CDN-eko kodea gure txantiloian sartzeko. Modu hau garapen momentuan gaudenerako bakarrik erabiliko dugu.

txantiloia.blade.php

```
<script src="https://cdn.tailwindcss.com"></script>
```

4. Orain bai, gure paginazioa askoz txukunago ikus dezakegu.

2. Balidazioa

Laravelek baditu hainbat funtzio datuen balidazioak egiteko lagungarri egingo zaizkigunak.

Ezer baino lehen, begiratuko dugu gure datu basean zeintzuk datu dira derrigorrezkoak, zeintzuk dira datu horien mota etab.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	bigint(20)		UNSIGNED	No	*Ninguna*		AUTO_INCREM
2	nombre	varchar(75)	utf8mb4_unicode_ci		No	*Ninguna*		
3	edad	int(11)			No	*Ninguna*		
4	telefono	varchar(35)	utf8mb4_unicode_ci		Sí	*NULL*		
5	direccion	varchar(75)	utf8mb4_unicode_ci		Sí	*NULL*		
6	created_at	timestamp			Sí	*NULL*		
7	updated_at	timestamp			Sí	*NULL*		

1. Akatsak datu basean datuak sartzean egingo ditugu, gure kasuan *store*() eta *update()* metodoen barruan kontrolatuko ditugu. Formularioko eremu guztiak dituen **store** funtzioaren parametro gisa iragaten den **request** aldagaia erabiliko dugu. **Validate()** metodoari array bat emango diogu, formularioko eremuak eta baliozkotzeak dituena.

Laravelek adierazi egiten du, momentu baten baten balidazioa betetzen ez baldin bada, ez da hurrengo kode lerrora pasatuko eta akatsa emango digu. Aldiz, balidazio guztiak ondo daudenean aurrera jarraituko du.

Gure kasuan, store() metodoan idatziko dugu, izena eta adina balidatuko ditugu:

```
        $request->validate([
            'nombre' => 'required|max:75',
            'edad' => 'required|integer',
        ]);
```

Akatsak mezuetan adierazi

2. Orain, gure inprimakiak ikusiko ditugu. Hor inprimatuko ditugu mezuak formularioko eremuetan, gure kontrolatzaileetan definitzen ditugun baliozkotze-akatsak daudenean.

3. Inprimakiak baliozkotzen hasi aurretik, css kodea erantsiko diegu gure ikuspegi guztiei, errore-mezuei estiloa emateko.

```
    p { color: red; }
```

Inprimakia ikasleen ikuspegian (bistan) baliozkotzea

4. **create.blade.php** bistan sortuko ditugu balidazio guztiak.

Input elementuaren azpiko *Izen-deiturak* eremuan, eremu honetako baliozkotze-kodea gehitzen dugu Bladek eskaintzen digun zuzentarauarekin. **@error** aukerak eremu honi dagokion errore-mezua inprimatzeko aukera ematen digu.

```
        @error('nombre')
                <p>{{ $message }}</p>
        @enderror
```

5. Gure input-ari gehituko diogu **old()** metodoa, blade-ko direktiba bat dena. Honegaz lortuko dugu balidazioa egin baino lehen sartutako datua mantentzea.

```
<input type="text" name="nombre" value="{{ old('nombre') }}">
```

6. Gauza berdina adina elementaun egingo dugu.

7. Sartutako telefonoa eta helbidea ere mantenduko ditugu, beraz old metodoari deituko diogu kasu guztietan.

8. Gauza berdina update metodoan eta bistan egingo dugu. Aldaketa bakarra old metodoari deitzerakoan izango dugu, kasu honetan old metodoak 2 parametro izango ditu, akatsa ematen duenerako eta akatsa ematen ez duenerako:

```
<input type="text" name="nombre" value="{{ old('nombre',$alumno->nombre) }}">
```

3. Form Request

Aurreko balidazioak hainbat lekutan jarri beharrean, hoberena leku bakar batean izatea izango litzateke, horretarako, balidazioak kontroladoretik kendu ahal izateko FormRequest dugu.

1. Lehenengo gauza egingo duguna beraz gure FormRequest generatzea izango da, horretarako terminalean hurrengo direktiba idatziko dugu

```
  php artisan make:request StoreAlumno
```

2. Ikusten dugunez Http>Request karpetan esandako fitxategia sortu du. Momentuz bi metodo dituen StoreAlumno klase bat duena.
3. Oraingoz authorize() metodoan ez dugu ezer egingo, baina hau izango denez lehenengo metodoa deituko dena, hurrengoari pasa dadin "return true; " jarriko diogu.
4. Eta rules() metodoan aldiz, gure balidazioak idatziko ditugu:

```
    public function rules(): array
    {
        return [
            'nombre' => 'required|min:5',
            'edad' => 'required'
        ];
    }
```

5. Eta azkenik, gure store eta update metodoan $request Request klasearen instantzia bat dela esan beharrean, StoreAlumno-ren intantzia izango dela esango diogu. Gogoratu ere klase hori importatu behar dugula, hau namespace idatzi behar dugu:

```
  public function store(StoreAlumno $request){
```

Eta:

```
  use App\Http\Requests\StoreAlumno;
```

6. FormRequest-en ere mezuak eta atributuak pertsonalizatu ditzakegu messages() eta atributes() metodoak erabiliz

```
public function messages(): array
    {
         return [
             'nombre.required' => 'El nombre es obligatorio',
             'edad' => 'La edad es obligatoria'
         ];
     }
```

4. Hizkuntza

Gure formularioarekin froga egiten badugu, ikusiko dugu zelan akats guztiak ingeleraz ateratzen diren, hori aldatu ahal izateko, hizkuntza aldatuko genuke.

1. Momentuz naiz eta gure mezuak euskaraz nahi izan, gazteleraz egingo dugu hasiera batean.
2. Lehenengo gauza egingo duguna, gure lang karpeta publikatzea izango da:

```
  php artisan lang:publish
```

3. Ikusten dugunez "en" karpeta dago soilik, eta guk "es" karpeta nahi dugu. Horretarako aukera bat kopiatzea izango litzateke eta hor dauden direktiba guztiak itzuli beharko genituzke; baina badago beste modu errezago bat
4. Sartuko gara https://laravel-lang.com/ webgunean, eta han Installation atalean ikusiko dugun Quick Start-eko direktibak kopiatuko ditugu gure terminalean, baina kontuz, guk ez dugu "fr" instalatu nahi, beraz "es" jarri behar dugu horren ordez:

```
  composer require laravel-lang/common --dev
  php artisan lang:add es
```

5. Orain badugu jada gure "es" karpeta sortuta, ikusten dugunez, direktiba guztiak gazteleraz daude.
6. Baina ondino ez diogu Laraveleri esan kapeta hori erabili behar duela, hori esateko, config karpetan app.php fitxategira sartu behar gara eta "locale" bilatu:

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

Jone Martinez jone.m@fpizurdinaga.com