Práctica UD-5 Sesión-4

Objetivos	• Aprender conceptos del **modelo y validación** para el desarrollo de aplicaciones webs con Laravel.
Instrucciones de entrega	• No hay que entregar nada.

Instalación de Visual Studio Code

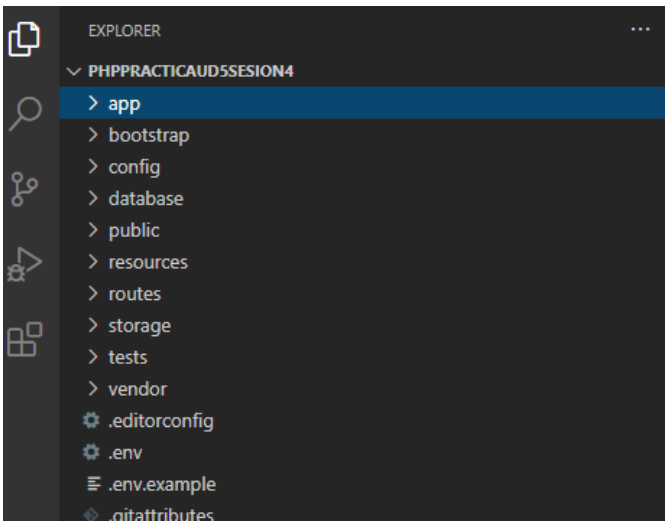
- Descargue la última versión en la URL:
  - https://code.visualstudio.com/

- Extensiones recomendadas para instalar:
  - PHP IntelliSense
  - PHP Intelephense
  - Auto Close Tag
  - Auto Rename Tag
  - Laravel snippets, Laravel blade snippets
  - Visual Studio IntelliCode

- Abre una línea de comandos desde **VS Code** y ve a la ruta donde está el directorio ***htdocs*** de Xampp (**c:\xampp\htdocs**) después crea un nuevo proyecto con nombre ***PHPPracticaUD5Sesion4***:

```
C:\xampp\htdocs>laravel new PHPPracticaUD5Sesion4
```

- Carga el proyecto desde ***VS Code*** simplemente abriendo la carpeta, una vez hecho esto debería aparecer la estructura de carpetas del proyecto:



Modelos

- En Laravel los modelos se gestionan en la carpeta ***"app/Models"***, donde pueden haber subcarpetas también. En una instalación limpia de Laravel se crea un primer modelo por defecto ***User.php***.

- Podemos utilizar un modelo conjuntamente con el ORM Eloquent que usa Laravel para ir a base de datos o bien simplemente como clase sin ninguna vinculación con bbdd y que utilizamos para pasar información a la vista (view models).

- Para crear un modelo podemos hacerlo desde línea de comandos, por ejemplo:
    - *php artisan make:model Link*

Validación en el servidor

- Veamos la validación en el servidor siguiente el siguiente tutorial y la documentación oficial:
    - https://appdividend.com/2019/03/09/laravel-5-8-form-validation-tutorial-with-example/
    - https://laravel.com/docs/8.x/validation

- Crea un nuevo modelo con nombre ***Form***:
    - *php artisan make:model Form*

- Añade el siguiente contenido:

```
class Form extends Model
{
    use HasFactory;

    protected $fillable = ['item_name', 'sku_no', 'price'];
```

```
}
```

- Crea un nuevo controlador con nombre ***FormController***:
  - *php artisan make:controller FormController*

- Añade dos métodos vacíos que luego rellenaremos:

```
    public function create()
    {
    }
    public function store(FieldRequest $request)
    {
    }
```

- Ahora añade las siguientes rutas al fichero ***web.php***:

```
use App\Http\Controllers\FormController;
...

Route::get('form', [FormController::class, 'create'])->name('form.create');
Route::post('form', [FormController::class, 'store'])->name('form.store');
```

- Por último vamos a crear las vistas para ello crea un nuevo directorio en **views** con nombre **layouts** y añade el archivo ***app.blade.php*** con el siguiente contenido:

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>@yield('title')</title>
  <!-- Scripts -->
  <script src="{{ asset('js/app.js') }}" defer></script>
  <!-- Styles -->
  <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>
<body>
  <div class="container">
    @yield('content')
  </div>
</body>
```

```
</html>
```

- Ahora añade una nueva vista a nivel del directorio **views** con nombre ***create.blade.php*** y el siguiente contenido:

```
@extends('layouts.app')

@section('title', 'Laravel, formulario con validación en servidor')

@section('content')
<style>
  .uper {
    margin-top: 40px;
  }
</style>
<div class="card uper">
  <div class="card-header">
    Nuevo Item
  </div>
  <div class="card-body">
    @if ($errors->any())
      <div class="alert alert-danger">
        <ul>
            @foreach ($errors->all() as $error)
              <li>{{ $error }}</li>
            @endforeach
        </ul>
      </div><br />
    @endif
      <form method="post" action="{{ route('form.store') }}">
          <div class="form-group">
              @csrf
              <label for="name">Item Name:</label>
              <input type="text" class="form-control" name="item_name"/>
          </div>
          <div class="form-group">
              <label for="price">SKU Number:</label>
              <input type="text" class="form-control" name="sku_no"/>
          </div>
          <div class="form-group">
              <label for="quantity">Item Price:</label>
```

```
            <input type="text" class="form-control" name="price"/>
        </div>
        <button type="submit" class="btn btn-primary">Create Item</button>
    </form>
  </div>
</div>
@endsection
```

- Modifica el controlador para añadir el siguiente contenido:

```
    public function create()
    {
        return view('create');
    }
```

- Prueba el resultado arrando el servidor de desarrollo de Laravel:
  - *php artisan serve*

- Navega a la URL:
  - http://127.0.0.1:8000/form

Añadir Bootstrap

- Para añadir **bootstrap** a nuestro proyecto ejecute desde línea de comandos los siguientes comandos:
  - *composer require laravel/ui*
  - *php artisan ui bootstrap*
  - *npm install*
  - *npm run dev* (si es necesario otra vez npm run dev)

- Navega de nuevo a la URL:
  - http://127.0.0.1:8000/form

Lógica de validación

- Para añadir la lógica de validación, Laravel nos proporciona unas funciones de ayuda, veamos cómo utilizarlas. Edita el controlador y añade el siguiente código en el método que recibe los datos del formulario:

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

```
        $validatedData = $request->validate([
            'item_name' => 'required|max:255',
            'sku_no' => 'required|alpha_num',
            'price' => 'required|numeric',
        ]);

      //Form::create($validatedData); //Añadir en base de datos
      return response()->json('Formulario validado satisfactoriamente');
    }
```

- Navega de nuevo a la URL y pruebe a enviar el formulario:
  - http://127.0.0.1:8000/form

Parar en el primer error de validación

- Tenemos la opción de parar la validación con el primer error de validación, para ello utilizamos asignar la regla ***bail*** a las reglas de validación del atributo donde queremos parar la validación. Edita de nuevo el contenido de las reglas y añada lo siguiente:

```
        $validatedData = $request->validate([
            'item_name' => 'bail|required|max:255',
            'sku_no' => 'required|alpha_num',
            'price' => 'required|numeric',
        ]);
```

Mostrar los datos previos introducidos

- Para mostrar los datos que el usuario ha introducido en el formulario después de volver a la página con errores de validación, Laravel nos proporciona una función ***old*** que automáticamente extraerá los datos de entrada previamente introducidos de la sesión.

- Edita la página create.blade.php y añade lo siguiente en los *inputs*:

```
<input type="text" class="form-control" name="item_name" value="{{
old('item_name') }}"/>
<input type="text" class="form-control" name="sku_no"  value="{{ old('sku_no')
}}"/>
<input type="text" class="form-control" name="price"  value="{{ old('price')
}}"/>
```

Mostrar errores

- Laravel redirigirá automáticamente al usuario a la ubicación anterior y además todos los errores de validación se incorporarán automáticamente a la sesión y luego los conectará con la vista.

- Para mostrar los errores en la vista se utiliza la variable ***$errors***:

```
@if ($errors->any())
  <div class="alert alert-danger">
    <ul>
        @foreach ($errors->all() as $error)
          <li>{{ $error }}</li>
        @endforeach
    </ul>
  </div><br />
@endif
```

Validación de petición de formulario (Form Request Validation)

- En el ejemplo anterior, hemos escrito las reglas de validación dentro de la función del controlador pero para escenarios de validación más complejos podemos crear un archivo separado y allí escribir las reglas de validación, esto se llama en Laravel ***"form request"***.

- Veamos un ejemplo, crea un *form request* desde línea de comandos:
  - *php artisan make:request FieldRequest*

- El nuevo fichero se habrá creado en la ruta ***app/Http/Requests*** con nombre ***FieldRequest.php***.

- Edita el fichero y añade el contenido siguiente en el método ***rules***:

```
public function rules()
{
    return [
        'item_name' => 'bail|required|max:255',
        'sku_no' => 'required|alpha_num',
        'price' => 'required|numeric',
   ];
}
```

- Como en este ejemplo no utilizamos autorización devuelve true en el método:

```
    public function authorize()
    {
        return true;
    }
```

- Ahora edita el controlador y pon el siguiente contenido:

```
    public function store(Request $request)
    {
        $validatedData = $request->validated();
        //Form::create($validatedData); //Añadir en base de datos
        return response()->json('Form is successfully validated and data has
been saved');
    }
```

- Prueba el resultado en la URL:
    - http://127.0.0.1:8000/form

Personalización de los mensajes de error

- Se pueden personalizar los mensajes de error utilizados por el form request editando el método de mensajes, para ello añade el siguiente código en ***FieldRequest.php***.

```
  public function messages()
  {
      return [
          'item_name.required' => 'El nombre del Ítem es obligatorio',
          'sku_no.required'  => 'El número SKU es obligatorio',
          'price.required'  => 'El precio es obligatorio',
      ];
  }
```

- Prueba el resultado en la URL:
    - http://127.0.0.1:8000/form

Recursos

- https://laraveles.com/la-forma-mas-inteligente-para-que-hagas-validaciones-en-laravel/
- https://richos.gitbooks.io/laravel-5/content/capitulos/chapter12.html

- http://alfredobarron.github.io/smoke/index.html#/