Laravel

Instalar Laravel

Para instalar Laravel tenemos que tener instalado Composer, y después desde composer creamos un proyecto Laravel. En este tutorial se explica perfectamente:

https://styde.net/instalacion-de-composer-y-laravel-en-windows/

Una vez creado el proyecto entramos en la consola dentro del proyecto y ponemos lo siguiente:

php artisan serve

Esto nos activa un servidor para ejecutar laravel en el puerto 8000. Poniendo lo siguiente:

localhost:8000

Entramos en el proyecto.

Sistema de rutas

Las rutas es una forma de asignar a una URL un código de nuestra aplicación. Estas URL no tienen que asignarse a archivos específicos en un sitio web.

En Laravel 5.5, las rutas se crean dentro de la carpeta de rutas. Las rutas para el sitio web se crean en el archivo web.php. Del mismo modo, las rutas para la API se crean dentro de api.php.

La instalación predeterminada de Laravel 5.5 viene con dos rutas, una para la web y otra para la API. Aquí es cómo se ve la ruta para web en web.php:

```
Route::get('/', function () {

   return view('welcome');

});
```

Aquí lo que estamos diciendo es: cuando nos venga una petición get en la raiz (‘/’) devuelve la vista ‘welcome’. Ya veremos más adelante qué es eso de las vistas.

La estructura de la ruta es muy sencilla. Abrimos el archivo (ya sea `web.php` o` api.php`) e inicie el código con `Route ::` A esto le sigue la solicitud que desea asignar a esa ruta específica y luego viene la función que será ejecutada como resultado de la solicitud.

Laravel ofrece los siguientes métodos de ruta:

- get
- post
- put
- delete
- patch
- options

Vamos a crear una ruta que nos muestre la tabla del 2:

```
Route::get('/table', function () {

  for($i =1; $i <= 10 ; $i++){

     echo "$i * 2 = ". $i*2 ."<br>";

  }

});
```

Ahora cuando pongamos http://localhost:8000/table nos ejecutará la función mostrando la tabla del 2.

También podemos usar parámetros:

```
        Route::get('/table/{number}', function ($number) {

        for($i =1; $i <= 10 ; $i++){

        echo "$i * $number = ". $i* $number ."<br>";

        }

        });
```

En el caso anterior debemos incluir un número en la url, que quedaría así: http://localhost:8000/table/7

Podemos hacer que ese parámetro sea opcional poniendo un ? en el parámetro y un valor por defecto en la función php:

```
  Route::get('/table/{number?}', function ($number=5) {
```

```
for($i =1; $i <= 10 ; $i++){
echo "$i * $number = ". $i* $number ."<br>";
}
});
```

Ya hemos creado una ruta para la tabla, pero ¿cómo asegurarnos de que si el parámetro de la ruta es realmente un número para evitar errores al generar una tabla? En Laravel puede definir una restricción para su parámetro de ruta usando el método `where` en la instancia de ruta. El método `where` toma el nombre del parámetro y una regla de expresión regular para ese parámetro. Ahora definamos una restricción para nuestro parámetro `{number}` para asegurar que solo se pase un número a la función.

```
Route::get('/table/{number}', function ($number) {
for($i =1; $i <= 10 ; $i++){
echo "$i * $number = ". $i* $number ."<br>";
}
})->where('number', '[0-9]+');
```

Si ahora ponemos lo siguiente: http://localhost:8000/table/t nos dirá que no encuentra lo que ponemos.

Controladores

En Laravel, podemos definir un método de controlador para una ruta. Cuando un usuario indica una ruta, las acciones se realizarán según el método del controlador que se haya definido.

Para asignar un método de controlador a la ruta, use el siguiente fragmento de código:

```
Route :: get ('/ home', 'YourController @ functionname');
```

El código comienza con `Route ::` y luego define el método de solicitud para la ruta. A continuación, defina su ruta y luego el controlador junto con el método que desea enlazar a esta ruta agregando el símbolo @ antes del nombre del método. Veremos los controladores más adelante.

En Laravel, puede definir el nombre de su ruta. Este nombre a menudo resulta muy útil en varios escenarios. Por ejemplo, si desea redirigir a un usuario de una ruta a otra, no necesita definir la URL de redireccionamiento completo. Con usar el nombre es suficiente. Puede definir el nombre de la ruta usando el método `name` en la instancia de la ruta.

```
Route::get('/table/{number}', function ($number) {
for($i =1; $i <= 10 ; $i++){
echo "$i * $number = ". $i* $number ."<br>";
```

```
    }
    })->where('number', '[0-9]+')->name('table');
```

Controladores

Los controladores se almacenan en la carpeta app/http/controllers. Los podemos crear a mano o utilizar el siguiente comando (mucho más fácil):

php artisan make:controller NombreDelControlador

El nombre suele seguir las mismas reglas que las clases, letra inicial en mayúsculas. Vamos a hacer lo mismo que antes pero vía controlador. Lo primero crear el controlador:

php artisan make:controller Tabla

Esto nos crea lo siguiente:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Tabla extends Controller
{
    //
}
```

A partir de aquí ya podemos poner las funciones que queramos:

```
class Tabla extends Controller
{
    function multiplicar(){
    $number=2;
    for($i =1; $i <= 10 ; $i++){
    echo "$i * $number = ". $i* $number ."<br>";
    }
    }
}
```

Y tenemos que asignar una ruta al controlador:

```
Route::get('/multiplicar','tabla@multiplicar');
```

Si ahora ponemos la ruta: http://localhost:8000/multiplicar llegamos al controlador que ejecuta el código que hemos puesto.

Si queremos pasar un parámetro es muy parecido a como lo hemos hecho antes:

```
Route::get('/multiplicar/{number}','tabla@multiplicar');

 function multiplicar($number){

        for($i =1; $i <= 10 ; $i++){
        echo "$i * $number = ". $i* $number ."<br>";
        }
        }
```

O con parámetros por defecto:

```
  Route::get('/multiplicar/{number?}','tabla@multiplicar');

  function multiplicar($number=6){

        for($i =1; $i <= 10 ; $i++){
        echo "$i * $number = ". $i* $number ."<br>";
        }
        }
```

En un controlador también podemos llamar a vistas o acceder a los datos. Esto lo veremos más adelante.

Vistas

Las vistas se crean en resources/views, tienen la extensión blade.php y se pueden retornar desde cualquier controlador, pasarles parámetros, etcétera. Vamos a ver un ejemplo. Voy a crear la vista hola.blade.php

```
<!doctype html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
        <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>Saludo</title>
```

```
        </head>
        <body>
        <h1>Hola ¿Qué tal?</h1>
        </body>
</html>
```

Voy a crear un controlador ‘saludo’:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Saludo extends Controller
{
        function hola(){
        return view('hola');
        }
}
```

Y voy a enrutarlo:

```
  Route::get('/hola','saludo@hola');
```

Si ahora pongo: http://localhost:8000/hola me saldrá el saludo de la vista.

Si queremos pasar datos del controlador a la vista lo podemos hacer pasando un array asociativo como segundo parámetro:

```
 function hola(){
        return view('hola',['nombre'=>'Juan']);
        }

        <body>
        <h1>Hola {{$nombre}} ¿Qué tal?</h1>
        </body>
```

Modelos

Hacer un modelo en Laravel es muy sencillo. Basta con poner la siguiente sentencia:

```
php artisan make:model Centros
```

Esto nos crea un modelo vacío, como sigue:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Centro extends Model
{
}
```

Le tendríamos que indicar a qué tabla hace referencia y cual es su clave primaria:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Centro extends Model
{
  protected $table = "centros";
  protected $primaryKey ="idcentro";
}
```

A partir de aquí ya lo podemos usar en nuestros controladores, simplemente usando el modelo y accediendo a los datos. Por ejemplo, si lo ponemos en el controlador saludo de antes:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Centro;
class Saludo extends Controller
{
        function hola(){
        // return view('hola',['nombre'=>'Juan']);
        $centros = Centro::all();
        dd($centros);
        }
}
```

Si quisiéramos ver una lista de todos los centros deberíamos recoger los datos en el controlador y pasarlos a la vista:

```
class Actor extends Controller {

    function index() {
        $actores = Actores::all();
        return view('actor/index', ['actores' => $actores]);
    }

}
```

En la vista:

```
 <body>
      @foreach($actores as $actor)
      <p>{{$actor->first_name}} {{$actor->last_name}}</p>
      @endforeach
    </body>
```

Laravel usa el modelo eloquent para recuperar los datos, podemos ver una lista completa aquí:

https://laravel.com/docs/5.8/eloquent

Veamos algunas en detalle.

Si lo que quieres hacer es buscar un registro en concreto de este modelo existe un método muy sencillo que permite recuperar un único elemento dado un id de clave primaria.

```
Centro::find(18);
```

Esto nos devolverá los datos del centro con identificador 18. Este es un objeto parecido al que hemos usado en la clase, por lo que podemos actualizar, modificar o borrar los elementos de ese centro. Si por ejemplo hacemos:

```
 $centro = Centro::find(18);

        $centro->nombre = "pepe";

        $centro->save();
```

Hemos hecho una actualización del centro. Si hacemos lo siguiente:

$centro = Centro::find(18);

$centro->delete();

Estaremos borrando el centro con el id 18.

Para crear:

$centro = new Centro();

$centro->nombre=”Academia Laia”;

$centro->save()

Podemos realizar búsquedas complejas:

```
$flights = App\Flight::where('active', 1)
               ->orderBy('name', 'desc')
               ->take(10)
               ->get();
```

Aquí tenemos toda la lista de cosas que podemos hacer:

https://laravel.com/docs/5.8/queries

Vamos a realizar el CRUD de la tabla de centros incluyendo modelos, vistas y controladores.

Empezemos por el enrutamiento. Si ponemos:

Route :: resource ('centros', 'Centro');

Asignamos automáticamente todas las rutas estándar del laravel a ese controlador, lo único que tendremos que hacer será poner el código necesario en cada apartado.

Creamos el controlador Centro, que quedará así:

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Centro extends Controller
{
   function index(){

   }
}
```

Para las vistas vamos a crear un layout que tendrá incorporado el bootstrap, para que quede bonito:

en views/layout/layout.blade.php

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min
.css">
        <!-- jQuery library -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
        <!-- Popper JS -->
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper
.min.js"></script>
        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.j
s"></script>
    </head>
```

```
    <body>
        <div class="container-fluid" >
            @yield('content')
        </div>
    </body>
</html>
```

Con esto creamos un layout para todas las vistas. El contenido se pondrá donde ponemos el yield(‘content’).

Empecemos por el controlador. Recuperamos los datos y los enviamos a la vista:

```
 function index() {
      $centros = Centros::orderBy('idcentro', 'DESC')->paginate(3);
      return view('Centro.index', compact('centros'));
   }
```

He añadido un paginate de tres elementos para tener la paginación. La vista index quedará así:

```
{{-- esta plantilla deriva del layout --}}
@extends('layouts.layout')
{{-- este contenido se pone donde esté el content del layout --}}
@section('content')
<div class="row">
   <div>
   <h1>Lista de los centros</h1>
{{-- Botón que enlaza con el create para dar de alta --}}
          <div >
            <a href="{{ route('centros.create') }}" class="btn btn-info" >Añadir Centro</a>
          </div>

         {{-- Los datos los mostramos en una tabla --}}
          <table id="mytable" class="table     table-striped">
           <thead>
             <th>Nombre</th>
             <th>Editar</th>
             <th>Eliminar</th>
           </thead>
```

```
            <tbody>

             @forelse($centros as $centro)
             <tr>
               <td>{{$centro->nombre}}</td>

               <td><a class="btn btn-primary btn-xs" href="{{action('Centro@edit',
$centro->idcentro)}}" ><span class="fa fa-pencil-square-o"></span></a></td>
               <td>
                 <form action="{{action('Centro@destroy', $centro->idcentro)}}" method="post">
                  {{csrf_field()}}
                  <input name="_method" type="hidden" value="DELETE">

                  <button class="btn btn-danger btn-xs" type="submit"><span class="fa
fa-trash"></span></button>
</form>
                </td>
              </tr>
             @empty
              <tr>
               <td colspan="8">No hay registros !!</td>
              </tr>
             @endforelse
            </tbody>

          </table>
        {{-- ENlaces de paginación --}}
      {{ $centros->links() }}
    </div>
</div>
@endsection
```

Vamos a hacer el create. En el controlador simplemente llamaremos a la vista. La vista mandará los datos a una función store que los guardará:

Revisemos el modelo:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
```

```
class Centros extends Model {
    protected $table = "centros";
    protected $primaryKey = "idcentro";
    protected $fillable = ['nombre'];
    public $timestamps = false;
}
```

Indicamos la tabla, la clave primaria, los campos que pueden rellenarse y que la tabla no tiene campos de creación y actualización.

El controlador:

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

    public function store(Request $request) {
        //
        $this->validate($request, ['nombre' => 'required']);
        try {
            Centros::create($request->all());

            return redirect()->route('centros.index')->with('success', 'Registro creado
satisfactoriamente');
        } catch (Exception $e) {
            echo $e->getMessage();
            die();
        }
    }
```

La vista:

```
@extends('layouts.layout')
@section('content')
<div class="row">
    <div class="col-md-12">
        @if (count($errors) > 0)
        <div class="alert alert-danger">
            <strong>Error!</strong> Revise los campos obligatorios.<br><br>
            <ul>
                @foreach ($errors->all() as $error)
```

```
            <li>{{ $error }}</li>
            @endforeach
          </ul>
        </div>
        @endif
        @if(Session::has('success'))
        <div class="alert alert-info">
          {{Session::get('success')}}
        </div>
        @endif

        <h1 class="panel-title">Nuevo Centro</h1>
        <div class="table-container">
          <form method="POST" action="{{ route('centros.store') }}"  role="form">
            {{ csrf_field() }}
            <div class="row">
              <div class="col-xs-6 col-sm-6 col-md-6">
                <div class="form-group">
                  <input type="text" name="nombre" id="nombre" class="form-control "
placeholder="Nombre del centro">
                </div>
              </div>
            </div>
            <div class="row">
              <div class="col-xs-12 col-sm-12 col-md-12">
                <input type="submit"  value="Guardar" class="btn btn-success btn-block">
                <a href="{{ route('centros.index') }}" class="btn btn-info btn-block" >Atrás</a>
              </div>
            </div>
          </form>
        </div>
    </div>

</div>
@endsection
```

Para actualizar copiamos prácticamente la vista anterior:

```
@extends('layouts.layout')
@section('content')
<div class="row">
    <div class="col-md-12">
        @if (count($errors) > 0)
        <div class="alert alert-danger">
```

```
            <strong>Error!</strong> Revise los campos obligatorios.<br><br>
            <ul>
                @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
        @endif
        @if(Session::has('success'))
        <div class="alert alert-info">
            {{Session::get('success')}}
        </div>
        @endif

        <h1 class="panel-title">Nuevo Centro</h1>
        <div class="table-container">
            <form method="POST" action="{{ route('centros.update',$centro->idcentro) }}"
role="form">
                {{ csrf_field() }}
                <input name="_method" type="hidden" value="PATCH">
                <div class="row">
                    <div class="col-xs-6 col-sm-6 col-md-6">
                        <div class="form-group">
                            <input type="text" name="nombre" id="nombre" class="form-control "
value="{{$centro->nombre}}">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 col-sm-12 col-md-12">
                        <input type="submit"  value="Actualizar" class="btn btn-success btn-block">
                        <a href="{{ route('centros.index') }}" class="btn btn-info btn-block" >Atrás</a>
                    </div>
                </div>
            </form>
        </div>
    </div>

</div>
@endsection
```

El controlador será muy sencillo:

```
public function edit($id) {
```

```
        //
        $centro = Centros::find($id);
        return view('centro.edit', compact('centro'));
    }

    public function update(Request $request, $id) {
        //
        $this->validate($request, ['nombre' => 'required']);
        Centros::find($id)->update($request->all());
        return redirect()->route('centros.index')->with('success', 'Registro actualizado
satisfactoriamente');
    }
```

Por último para eliminar sólo nos hace falta poner el controlador:

```
  public function destroy($id) {
         Centros::find($id)->delete();
        return redirect()->route('centros.index')->with('success', 'Registro eliminado
satisfactoriamente');
    }
```