VISTAS EN LARAVEL

En Laravel, la parte de Vista del MVC son archivos .php en la carpeta **resources/views**.

Una vista se llama de la forma:

nombredelavista.blade.php

Creando vistas y métodos personalizados

Nuestro intermediario para todo es el controlador. Por eso, si quiero mostrar una vista creada por mí, tendré que:

1. Añadir un método personalizado al controlador para que retorne la vista
2. Crear la vista en la carpeta views
3. Añadir una ruta con get, cuya función será el método del controlador que retorna esa vista

1.- Método personalizado:

Vamos a crear un método personalizado en nuestro controlador. Para ello, basta con añadirlo al final.

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

La función view buscará un fichero llamado **contacto.blade.php** en la carpeta de view.

Puedo agrupar vistas en subcarpetas dentro de resources. Si el fichero está en una subcarpeta llamada “paginas”, habría que llamar a:

view(‘paginas.contacto’)

2.- Nueva vista:

Creamos un fichero, con la siguiente estructura de nombres:

nombrevista.blade.php

en nuestro caso, se llamará

contacto.blade.php

Laravel ofrece varios “atajos” y otras utilidades que facilitan la escritura de código php. Son una serie de plantillas, y el motor (programa interno de Laravel que permite el uso de estas plantillas) es blade.

Bien, ¿qué escribimos en este php?. Como es una vista, lo lógico es poner **HTML**. Podemos copiar el código de welcome.blade.php y adaptarlo, o escribir nuestro HTML desde cero.

De momento no vamos a definir CSS, vamos a hacer una vista sencilla a modo de prueba: HTML puro.

contacto.blade.php

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Contacto</title>
    </head>
    <body>
        <div class="container">
            <h1>Página de contacto</h1>
        </div>
    </body>
</html>
```

Así, en el método contacto del controlador, se retornará esta vista, una página HTML que muestra el texto “Página de contacto”.

Añadir la ruta en web.php:

La ruta sería como sigue:

```
Route::get('/contacto','\App\Http\Controllers\PostController@contacto');
```

Una vez definida la ruta con get, puedo escribir la URL en el navegador, y ver cómo carga la página:

Pasando datos a las vistas

Vamos a definir la siguiente ruta en web.php:

```
Route::get('/post/{var}','\App\Http\Controllers\PostController@show_post');
```

Para esta ruta, definiremos un método en nuestro controlador, llamado “show_post”, que retornará en el response una vista llamada post (fichero post.blade.php). Recibirá una variable como parámetro.

```
    public function show_post($var) {
        return view('post');
    }
```

La vista la crearemos en la carpeta views, tendrá que llamarse post.blade.php.
Hay dos formas de pasarle la variable como parámetro:

1. Pasar un único parámetro con with
2. Pasar varios parámetros con compact

1.- Paso de parámetro a la vista: Utilizando **with**

Encadenamos a la llamada a view un with de la forma:

```
return view('post')->with('var',$var);
```

En esta opción, view abre la vista y le pasa esa información. Los parámetros del with son:

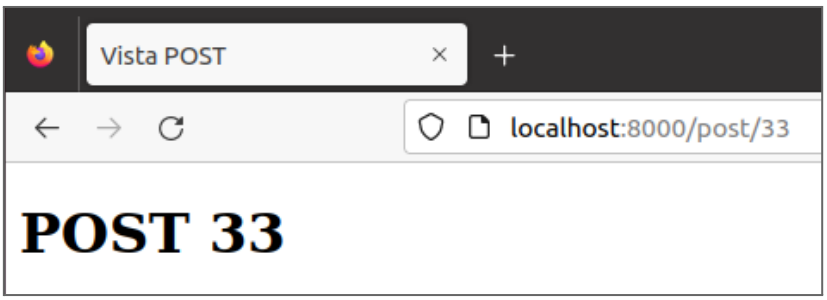
1. El nombre de la variable
2. El valor de la variable

Para mostrar la variable en la vista, utilizaré dobles llaves `{{$var}}`

post.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Vista POST</title>
</head>
<body>
    <div class="container">
        <h1>POST {{$var}}</h1>
    </div>
</body>
</html>
```

Si pongo la ruta como URL en el navegador obtendré:



2.- Paso de parámetro a la vista: Múltiples parámetros

En la opción 1 sólo podía pasarse una variable.

La forma de pasar múltiples parámetros es la siguiente:

```
return view('post', compact('var'));
```

La función **compact**:

1. Cogerá como nombre del parámetro "var"
2. Asignará a ver el valor del parámetro que reciba el método del controlador.

Podrán pasarse tantos parámetros como tenga el método del controlador, separados por comas. Compact generará un array de parámetros para la vista, con sus nombres y sus valores.

EJERCICIO

Crea una ruta "/viaje" con tres parámetros llamados país, provincia y ciudad.

```
/viaje/{pais}/{provincia}/{ciudad}
```

Haz un método llamado muestra_ciudad, dentro del controlador asociado a esa ruta. Recibirá los tres parámetros y retornará una vista llamada ciudad.

Crea la vista llamada ciudad.

El resultado en el navegador será el siguiente: