3- Creando el layout

Si hemos seguido la serie hasta aquí ya tenemos el front configurado. Si quieres ver el articulo anterior has clic aqui.

En la carpeta public deberían haber dos carpetas (**css** y **js**) dentro de **css** dos archivos **app.css** y **style.css** y dentro de **js** el único archivo por ahora sería **app.js.**

Crearemos dos vistas. Una será el **layout** que permanecerá igual durante todo el proyecto y la siguiente será **home** que es donde iremos agregando y modificando código a medida que avancemos.

Primero revisemos cómo debería quedar el layout. (En este caso estará en **resources/view/template/layout.blade.php**)

```
<!DOCTYPE html>
<html lang="{{ config('app.locale') }}">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>{{ config('app.name', 'Laravel') }} | @yield('title') </title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="//fonts.googleapis.com/css?family=Roboto:300,400,500,700,400italic">
    <link rel="stylesheet" href="//fonts.googleapis.com/icon?family=Material+Icons">
    <link href="{{ asset('public/css/app.css') }}" rel="stylesheet">
    <link href="{{ asset('public/css/style.css') }}" rel="stylesheet">
    <script>
        window.Laravel = {!! json_encode([
            'csrfToken' => csrf_token(),
        ]) !!};
    </script>
    @yield('style')
</head>
<body>
<div class="app">
    @yield('content')
</div>
<script src="{{ asset('public/js/app.js') }}"></script>

@yield('script')
</body>
</html>
```

Una vez más el código es apenas una modificación de los estándar de laravel. Pero lo revisaremos un poco.

```
<html lang="{{ config('app.locale') }}">
```

Aquí estamos usando la configuración local de laravel para definir el idioma de la página. Hacer esto y no poner el idioma directamente es buena práctica ya que el archivo **config/app.php** debería estar configurado según el idioma de nuestro sitio. Sobre todo la línea:

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

Que aquí está configurado en Español.

Para que **Axios** funcione correctamente y no tenga problemas de token en las llamadas ajax se se configura la cabecera como hemos visto en el post anterior

```
let token = document.head.querySelector('meta[name="csrf-token"]');

if (token) {
    window.axios.defaults.headers.common['X-CSRF-TOKEN'] = token.content;
} else {
    console.error('CSRF token not found: https://laravel.com/docs/csrf#csrf-x-csrf-token');
}
```

Esto ya lo tenemos hecho. No hay que cambiar nada de aquí pero lo repito desde el post anterior porque aquí vemos que el token lo recoge del meta de la web. Esto hace que nuestro layout deba incluir

```
<meta name="csrf-token" content="{{ csrf_token() }}">
```

para que todo funcione perfectamente.

Las siguientes 3 líneas del layout

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="//fonts.googleapis.com/css?family=Roboto:300,400,500,700,400italic">
<link rel="stylesheet" href="//fonts.googleapis.com/icon?family=Material+Icons">
```

Son de iconos font-awesome y fuentes de letras de Google. Menos los iconos lo demás queda a criterio personal usarlo o no. En este proyecto los iconos los usaremos.

Importante

Ahora veremos el link de los **css** y el **js**. Mi proyecto esta configurado para usar en **miproyecto.com/** por lo tanto los assets deben incluir la carpeta **public** en el path pero si tu proyecto es **miproyecto.com/public/** entonces la carpeta **public** debe ser eliminada de la ruta. Este cambio en particular se explica al final de este post.

```
<link href="{{ asset('public/css/app.css') }}" rel="stylesheet">
<link href="{{ asset('public/css/style.css') }}" rel="stylesheet">
<script src="{{ asset('public/js/app.js') }}"></script>
```

Si este punto no quedó claro poner en los comentarios las dudas.

Ahora solo un repaso a los **@yield** y el lugar escogido para cada uno

```
@yield('style')
</head>
<body>
<div class="app">
    @yield('content')
</div>
<script src="{{ asset('public/js/app.js') }}"></script>

@yield('script')
</body>
```

El **yield style** está puesto de manera que si en una página en particular tengo que usar un estilo diferente en algo pueda crear un section son un **<style></style>** dentro y definir mis propias reglas css

El **yield content** está contenido dentro de un **div con class app.** Esto es indispensable en este proyecto para el correcto funcionamiento de vue y cuando hagamos los primeros códigos repasaremos porque.

El **yield script** está puesto estratégicamente para que todo el código javascript que escribamos este después de cargar nuestros **js** (en este caso uno solo) y no le falte información. Esta forma de este yield es recomendado siempre que escribamos código js en nuestras páginas ya sea que usemos Vue, Angular, React, jQuery o cualquier otro framework / librería js.

Carpeta public

Cuando hacemos un proyecto comercial nuestro proyecto no es **www.loquesea.com/public** sino que es **www.loquesea.com.**

La carpeta public que viene en Laravel tiene varias maneras de no estar presente en la url del navegador. Una tecnica es apuntar el servidor directamente a public o diferentes configuraciones que permiten mantener la estructura de laravel sin tocar nada de su configuración principal.

En mi caso no suelo trabajar siempre con los servidores y eso queda en manos de terceros, en ocasiones que no tienen mucha idea de como hacer las cosas por lo que prefiero preparar el proyecto para entregarlo de forma que quede configurado la carpeta raiz como carpeta principal.

Para eso tenemos que seguir los siguiente pasos:

- Renombrar **/server.php** como **index.php**
- Cambiar en ese archivo

```
require_once __DIR__ . '/public/index.php';

por

require_once __DIR__ . '/public/server.php';
```

- Renombrar **/public/index.php** por **/public/server.php**
- Mover **/public/.htaccess** a **/.htaccess**
- Colocar el contenido de **.htaccess** a:

```
IfModule mod_rewrite.c>
    <IfModule mod_negotiation.c>
        Options -Indexes
    </IfModule>
 <FilesMatch "\.(json|xml|lock|env|example|gitattributes|gitignore)$">
    Order allow,deny
    Deny from all
    </FilesMatch>
    <Files artisan>
    Order allow,deny
    Deny from all
    </Files>

    RewriteEngine On

    # Redirect Trailing Slashes If Not A Folder...
    RewriteRule ^(.*)/$ /$1 [L,R=301]
    # Handle Front Controller...
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
    # Handle Authorization Header
    RewriteCond %{HTTP:Authorization} .
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

- Cambiar los permisos de **webpack.mix.js** para que solo el dueño (owner) pueda leer o escribir en ese archivo.

Con esta configuración (Que es la que esta en el repositorio) el sistema funciona en raiz. y los assets se ven así:

```
"{{ asset('public/js/app.js') }}"
```

OJO: Uso esta configuración por el tema ya explicado pero siempre es mejor manejar la configuración desde el servidor y apuntar a la carpeta public. Eso da mas seguridad. Si se usan entornos como Homestead la soluciones son mas sencillas.

En estos casos se debe eliminar la carpeta public de asset. Para el ejemplo anterior quedaria asi

```
"{{ asset('js/app.js') }}"
```

Código: Github

En el siguiente post. Trabajando la home