layouts, Blade y Bootstrap

Los **Layout** son el "molde" o "plantilla principal" de nuestra aplicación. Así como una casa tiene: techo, paredes, puertas , nuestras vistas tienen: cabecera, menú, contenido, scripts y no queremos repetir eso en todas las páginas.

Por eso creamos un **layout principal.**

Para crear y utilizar un layout, Laravel ofrece un ciertas directivas atraves de su motor de plantillas **Blade.** La directiva **@yield** para reservar espacios. Luego, las vistas individuales "extienden" del layout y rellenan esos espacios utilizando la directiva **@section**

Componentes del layout (*Inicio del documento, Head, Body*)

- **Inicio del documento**

```
<!doctype html>
<html lang="es">
```

"Lo que viene es una página HTML escrita en español."

- **Head** (contiene configuraciones de la página)

- **Charset**

```
<meta charset="utf-8">
```

permite mostrar correctamente ñ,tildes y simbolos

- **Viewport**

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Adptá la página al tamaño de la pantalla

- **CSRF Token**

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

El @csrf protege formularios HTML.
El <meta name="csrf-token"> deja ese mismo token disponible para Javascript.
Ambos contienen el mismo valor, pero se usan en contextos distintos.

- **Título**

```
<title>@yield('title','MiApp')</title>
```

Cada página puede definir su propio título.

@section('title','Categorías')

- **Bootstrap**

<link rel="stylesheet" href=

Acá cargamos Boostrap, Framework CSS y que agrega extensiones JavaScript para la interactividad

Botones, formularios , tablas , grillas , menús, listos para usar.

No tenemos que diseñar todo desde cero.

- **Font Awesome**

<link rel="stylesheet" href=

Bibliotecas de Iconos

- **Stack de estilos**

@stack('styles')

Si alguna vista necesita CSS extra, puede agregarlo aquí.

- **Body**

El **body** es el cuerpo de la página.
Todo lo que el usuario ve en pantalla está dentro del body.

- **NavBar**

<nav class="navbar navbar-expand-lg navbar-light bg-light">

Barra superior

- **Container**

Bootstrap usa containers para: centrar contenido, agregar márgenes, mantener orden

- **@yield('contenido')**

Acá es donde se inyectará el contenido de cada vista.

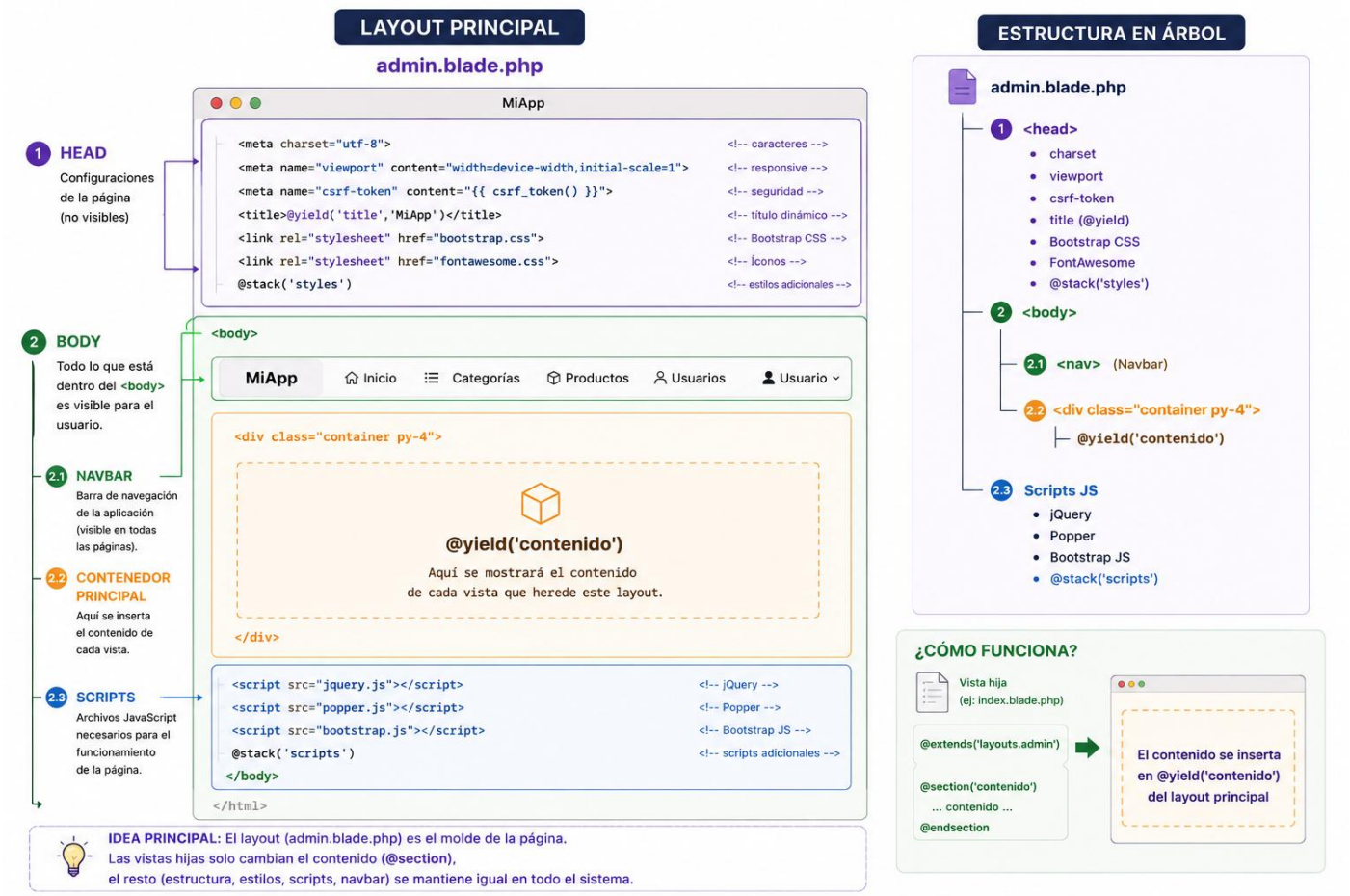
El layout es el marco del cuadro.
@yield('contenido') es el espacio donde colocamos la foto.

-Dependencias Javascript (jquery , Boostrap JS, Stack de Scripts)

Jquery: Biblioteca para manipular HTML y eventos.

Boostrap JS: Activa componentes interactivos: modales, dropdowns , menús, tooltips

Stack de Scripts: Sirve para que una vista agregue Javascript adicional sin tocar el layout.



¿Qué es Bootstrap?

- **Bootstrap** es un **framework de CSS y extensiones de JavaScript** gratuito y de código abierto creado por Twitter orientado a la creación de interfaces web.

- Su objetivo es permitir a los desarrolladores diseñar interfaces web **rápidas, responsivas y modernas con sus componentes predefinidos** sin necesidad de escribir todo el CSS desde cero, como por ejemplo grid, botones, formularios, tablas, columnas, alertas, modales.

- **Ventajas** para el desarrollador:

    - Ahorra tiempo acelerando la productividad con diseños predefinidos
    - Diseño responsivo (se adapta a móvil/pc automáticamente).
    - Compatible con navegadores modernos.
    - Extensible: puedes personalizarlo con tu propio CSS.

El componente NavBar

Es un **componente visual e interactivo** que suele contener:

- El **logo** o nombre de la aplicación (navbar-brand).
- **Enlaces de navegación** (ej: Inicio, Contacto, Login).
  - Menús desplegables.
- Botones (ej: salir, registrarse).
- En algunos casos: buscadores, iconos o perfiles de usuario. por ejemplo:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <!-- Logo o nombre -->
  <a class="navbar-brand" href="#">MiApp</a>

  <!-- Enlaces -->
  <div class="collapse navbar-collapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Inicio</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Contacto</a>
      </li>
    </ul>
  </div>
</nav>
```

Explicación de clases usadas

- navbar → activa el estilo de barra de navegación.
- navbar-expand-lg → hace que los menús se expandan horizontalmente en pantallas grandes (LG) y se colapsen en pantallas chicas (móvil).
- navbar-light → texto oscuro, pensado para fondos claros.
- bg-light → color de fondo gris claro.
- navbar-brand → define el área de logo/nombre.
- nav-link → da estilo a los enlaces de navegación.
- nav-item → define cada elemento dentro del menú.

El sistema de grillas de Bootstrap

- Bootstrap divide el ancho de la página en **12 columnas virtuales**.
- Cuando usás clases como col-md-3, significa:
  "Esta columna ocupa 3 de las 12 partes disponibles en pantallas medianas o más grandes".

Entonces:

- col-md-3 → 3/12 (25% del ancho).

- col-md-6 → 6/12 (50% del ancho).
- col-md-12 → 12/12 (100% del ancho, columna completa).

¿Qué es form-group en Bootstrap?

- Es una **clase de Bootstrap** que sirve para **agrupar un campo de formulario con su etiqueta y/o ayuda** (ej: label, input, textarea, select).
- Su objetivo es **ordenar visualmente** los formularios:
  - Aplica **márgenes inferiores** automáticos.
  - Asegura una **alineación correcta** entre el label y el campo de entrada.
  - Mantiene consistencia en la distribución de todos los campos del formulario.

Resumen de los componentes Bootstrap utilizados

1. **Grillas** (row, col-12, col-md-11, col-md-2)
   - Organizan el contenido en filas y columnas, adaptándose al tamaño de la pantalla.
2. **Utilidades de alineación** (justify-content-center, align-items-center)
   - Centran el contenido horizontal y verticalmente.
3. **Botones** (btn, btn-success, btn-info, btn-warning, btn-danger, btn-block, btn-lg)
   - Distintos colores y tamaños para indicar acciones (nuevo, ver, editar, eliminar).
4. **Tablas** (table, table-striped, table-bordered, table-hover)
   - Mejoran la presentación de los datos con bordes, alternancia de filas y resaltado al pasar el mouse.
5. **Responsividad** (table-responsive)
   - Hace que la tabla se pueda desplazar horizontalmente en pantallas pequeñas.
6. **Íconos de FontAwesome** (fa fa-eye, fa fa-pencil, fa fa-remove)
   - Refuerzan visualmente la acción de cada botón.

Datos Flash

A veces puede desear almacenar elementos en la sesión para la próxima solicitud. Puede hacerlo utilizando el método flash. Los datos almacenados en la sesión utilizando este método estarán disponibles inmediatamente y durante la siguiente solicitud HTTP. Después de la siguiente solicitud HTTP, los datos destellados se eliminarán. Los datos de destello son principalmente útiles para mensajes de estado de corta duración:

```
$request->session()->flash('status', 'Task was successful!');
```

Si necesita persistir sus datos de destello durante varias solicitudes, puede utilizar el método reflash, que mantendrá todos los datos de destello durante una solicitud adicional.

Validacion con FormRequest

Una de nuestras metas como desarrollador profesional es construir un código limpio, fácil de mantener y reusable. Sin embargo, es común que tengamos controladores que se ocupan de muchas tareas y a veces repetidas haciendo que sean extensos o complejos. Para tales casos, puede ser

conveniente usar una clase especial de Laravel llamada Form Request que permiten separar la lógica de validación de datos (validación, mensajes de errores, autorización de usuarios y redirección en caso de fallar) de la lógica del controlador. Esta clase intercepta la solicitud o request y valida los datos que vienen de una petición HTTP antes de pasar al controlador.

```
 public function store(Request $request){
        $request->validate([
            'descripcion' => 'required',
        ],
        [
            'descripcion.required' => 'el campo
descripcion debe ser requerido'
        ]
        );
        $categorias = new Categorias();
```

Visualización de los errores de validación

Entonces, ¿qué sucede si los campos de la solicitud entrante no superan las reglas de validación dadas? Como se mencionó anteriormente, Laravel redirigirá automáticamente al usuario a su ubicación anterior. Además, todos los errores de validación y datos de entrada de la solicitud se almacenarán automáticamente en la sesión.

Se comparte una variable $errors

Por ejemplo si estamos haciendo un alta, el usuario será redirigido al método create de nuestro controlador cuando la validación falle, lo que nos permite mostrar los mensajes de error en la vista.

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

Si no hay errores entonces debemos mostrar el mensaje satisfactorio

```
@if (session('mensaje-success'))
  <p class="alert alert-success">{{ session('mensaje-success') }}</p>
@endif
```

Creación de un Form Request

Para crear una clase Form Request se ejecuta en la consola:

```
php artisan make:request NombreRequest
```

Como por ejemplo:

```
php artisan make:request ProductCreateRequest
```

y creará un nuevo archivo en /app/Http/Requests/ProductCreateRequest.php.

dentro del archivo contiene:

authorize este es un método opcional donde colocamos la lógica para la autorización del usuario que devolverá true si el usuario está autorizado para hacer la solicitud y false en caso contrario. Si se decide dejar la lógica de autorización en otra parte de la aplicación, simplemente se devuelve true de esta manera:

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

rules método que devuelve un array con las reglas de validación que serán aplicadas. Por ejemplo:

```
public function rules()
{
  return [
      'name' => 'required',
      'price' => 'required|min:0',
  ];
}
```

messages es el método encargado de retornar un array con pares atributo.regla y su correspondiente mensaje de error y de esta manera podemos personalizar los errores de la validación. Por ejemplo:

```
public function messages()
{
```

```
    return [
        'name.required' => 'El :attribute es obligatorio.',
        'price.required' => 'Añade un :attribute al producto',
        'price.min' => 'El :attribute debe ser mínimo 0'
    ];

}
```

Reglas de validación disponibles

https://laravel-docs.com/es/docs/10.x/validation#quick-displaying-the-validation-errors