Compra Venta Laravel - Mostrar errores en login

Para mostrar un mensaje de error cuando el correo no exista o la contraseña sea incorrecta, puedes utilizar `@error` en Blade y también asegurarte de que en tu `LoginController` los errores se devuelvan correctamente.

1. Modificar la Vista para Mostrar Mensajes de Error

Debes agregar un `@if(session('status'))` y `@error` en tu formulario de login.

blade

```
<form action="{{ route('login') }}" method="POST">
    @csrf

    @if(session('status'))
        <div class="alert alert-danger">
            {{ session('status') }}
        </div>
    @endif

    <div class="mb-3">
        <label for="email" class="form-label">Correo Electrónico</label>
        <input type="email" class="form-control @error('email') is-invalid
@enderror" id="email" name="email" placeholder="Ingresa tu correo electrónico"
required>
        @error('email')
            <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>

    <div class="mb-3">
        <div class="float-end">
            <a href="{{ route('password.request') }}" class="text-muted">¿Olvidaste
tu contraseña?</a>
        </div>
        <label class="form-label" for="password">Contraseña</label>
        <div class="position-relative auth-pass-inputgroup mb-3">
            <input type="password" class="form-control pe-5 @error('password') is-
invalid @enderror" name="password" id="password" placeholder="Ingresa tu contraseña"
required>
```

```
            <button class="btn btn-link position-absolute end-0 top-0 text-
decoration-none text-muted" type="button" id="password-addon">
                <i class="ri-eye-fill align-middle" id="toggle-password-icon"></i>
            </button>
        </div>
        @error('password')
            <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>

    <div class="form-check">
        <input class="form-check-input" type="checkbox" name="remember" id="auth-
remember-check">
        <label class="form-check-label" for="auth-remember-check">Recuérdame</label>
    </div>

    <div class="mt-4">
        <button class="btn btn-success w-100" type="submit">Iniciar Sesión</button>
    </div>

</form>
```

---

2. Modificar el Controlador de Login para Personalizar los Mensajes de Error

Si estás usando Laravel Breeze, el controlador de autenticación suele ser `AuthenticatedSessionController.php` ubicado en `app/Http/Controllers/Auth`.

Modifica el método `store()` para manejar errores y redirigir con un mensaje de error:

php

```php
namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\ValidationException;
```

```php
class AuthenticatedSessionController extends Controller
{
    public function store(Request $request)
    {
        $credentials = $request->validate([
            'email' => ['required', 'email'],
            'password' => ['required'],
        ]);

        if (!Auth::attempt($credentials, $request->filled('remember'))) {
            return back()->with('status', 'Las credenciales no son correctas.
Verifica tu correo y contraseña.');
        }

        $request->session()->regenerate();
        return redirect()->intended(route('dashboard')); // Ajusta la ruta según tu
proyecto
    }
}
```

Explicación de los Cambios

1. **Vista Blade (`login.blade.php`):**
   - Se agregaron mensajes de error para `email` y `password`.
   - Se muestra un mensaje de error general en un `div.alert-danger` cuando las credenciales no coinciden.
2. **Controlador (`AuthenticatedSessionController.php`):**
   - Se valida el email y la contraseña antes de intentar autenticarse.
   - Si `Auth::attempt()` falla, se redirige con un mensaje de error.
   - Si el login es exitoso, se regenera la sesión y se redirige al `dashboard`.

Con esto, los usuarios verán un mensaje claro cuando ingresen un email incorrecto o una contraseña equivocada. 🚀

Aquí tienes la migración para agregar los tres nuevos campos (`barcode`, `qrcode` y `fechvencimiento`) después del campo `status` en la tabla `products`:

Generar la Migración

Ejecuta el siguiente comando en la terminal para crear la migración:

sh

```
php artisan make:migration add_barcode_qrcode_fechvencimiento_to_products_table --table=products
```

Editar la Migración Generada

Abre el archivo de migración recién creado en `database/migrations/` y actualízalo con el siguiente código:

php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    public function up()
    {
        Schema::table('products', function (Blueprint $table) {
            $table->string('barcode')->nullable()->after('status');
            $table->string('qrcode')->nullable()->after('barcode');
            $table->date('fechvencimiento')->nullable()->after('qrcode');
        });
    }

    public function down()
    {
        Schema::table('products', function (Blueprint $table) {
            $table->dropColumn(['barcode', 'qrcode', 'fechvencimiento']);
        });
    }
};
```

Ejecutar la Migración

Una vez editada la migración, ejecútala con:

```sh
php artisan migrate
```

Explicación

- `barcode`: Un `string` que almacenará el código de barras.
- `qrcode`: Un `string` que almacenará el código QR.
- `fechvencimiento`: Un `date` para la fecha de vencimiento del producto.
- `nullable()`: Permite que estos campos puedan quedar vacíos.
- `after('status')`: Asegura que los campos se agreguen después del campo `status`.

Con esto, ya tendrás los nuevos campos listos en la tabla `products`. 🚀

Si deseas eliminar todas las migraciones y seeders y volver a generarlos desde cero, puedes seguir estos pasos:

1. Eliminar todas las tablas y volver a ejecutar las migraciones

Ejecuta el siguiente comando para eliminar y volver a ejecutar todas las migraciones:

```sh
php artisan migrate:fresh
```

🔹 **¿Qué hace?**

- Elimina todas las tablas de la base de datos.
- Ejecuta nuevamente todas las migraciones desde cero.

---

2. Volver a ejecutar los Seeders

Después de correr las migraciones, ejecuta los seeders con:

```sh
php artisan db:seed
```

Si tienes un `DatabaseSeeder.php` que llama a todos los seeders, este comando ejecutará todos ellos.

Si quieres ejecutar un seeder específico, usa:

```sh
php artisan db:seed --class=NombreDelSeeder
```

Por ejemplo, si tienes un seeder llamado `ProductsTableSeeder`, puedes correrlo así:

```sh
php artisan db:seed --class=ProductsTableSeeder
```

---

3. Opcional: Ejecutar Migraciones y Seeders en un Solo Comando

Si deseas hacerlo todo en un solo paso (migraciones + seeders), puedes ejecutar:

```sh
php artisan migrate:fresh --seed
```

🔹 **¿Qué hace?**

- Borra todas las tablas.
- Ejecuta las migraciones desde cero.
- Corre los seeders automáticamente.

---

4. Si Necesitas Reiniciar los Archivos de Migración y Seeders

Si deseas **eliminar completamente** los archivos de migraciones y seeders para generarlos de nuevo, puedes hacer lo siguiente:

```sh
rm database/migrations/*.php
rm database/seeders/*.php
```

Luego, genera nuevamente las migraciones con:

```sh
php artisan make:migration create_products_table
php artisan make:seeder ProductsTableSeeder
```

Con estos pasos, puedes limpiar completamente tu base de datos y reconstruirla desde cero con migraciones y seeders actualizados. 🚀

Para generar automáticamente el código de barras y el código QR al momento de guardar el producto, puedes actualizar tu método `store` en el `ProductController` de la siguiente manera:

---

1. Modificar el Método `store` en `ProductController`

Actualiza tu método para generar el código de barras y el QR automáticamente:

```php
use SimpleSoftwareIO\QrCode\Facades\QrCode;
use Picqer\Barcode\BarcodeGeneratorPNG;
use Illuminate\Support\Facades\Storage;

public function store(Request $request)
{
    $request->validate([
        'name' => 'required|string|max:255',
```

```
            'description' => 'nullable|string',
            'price' => 'required|numeric|min:0',
            'quantity' => 'required|integer|min:1',
            'category_id' => 'required|exists:categories,id',
            'status' => 'required|in:available,sold,archived',
            'fechvencimiento' => 'nullable|date',
            'images.*' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ]);

        $data = $request->only(['name', 'description', 'price', 'quantity',
'category_id', 'status', 'fechvencimiento']);
        $data['user_id'] = auth()->id(); // Agrega el ID del usuario autenticado

        // Crear el producto para obtener el ID
        $product = Product::create($data);

        // 1. Generar código de barras (usando el ID del producto)
        $barcodeGenerator = new BarcodeGeneratorPNG();
        $barcodeData = $barcodeGenerator->getBarcode($product->id,
BarcodeGeneratorPNG::TYPE_CODE_128);
        $barcodePath = "barcodes/{$product->id}.png";
        Storage::disk('public')->put($barcodePath, $barcodeData);
        $product->barcode = $barcodePath;

        // 2. Generar código QR con la información del producto
        $qrCodeData = "Producto: {$product->name}\nPrecio: {$product->price}\nCantidad:
{$product->quantity}\nEstado: {$product->status}";
        $qrCodePath = "qrcodes/{$product->id}.png";
        Storage::disk('public')->put($qrCodePath, QrCode::format('png')->size(200)-
>generate($qrCodeData));
        $product->qrcode = $qrCodePath;

        // Guardar los datos actualizados en la base de datos
        $product->save();

        // 3. Guardar imágenes del producto (si existen)
        if ($request->hasFile('images')) {
            foreach ($request->file('images') as $image) {
                $imagePath = $image->store('products', 'public');
                ProductImage::create([
                    'product_id' => $product->id,
                    'image_path' => $imagePath,
```

```
            ]);
        }
    }

    return redirect()->route('products.index')->with('success', 'Producto creado
exitosamente.');
}
```

---

2. Explicación de los Cambios

1. **Generación del Código de Barras:**
   - Se usa la librería `Picqer\Barcode\BarcodeGeneratorPNG` para generar el código de barras a partir del `id` del producto.
   - Se almacena la imagen en `storage/app/public/barcodes/ID.png`.
   - Se guarda la ruta en el campo `barcode` de la base de datos.
2. **Generación del Código QR:**
   - Se genera un QR con la información del producto (nombre, precio, cantidad, estado).
   - Se almacena la imagen en `storage/app/public/qrcodes/ID.png`.
   - Se guarda la ruta en el campo `qrcode` de la base de datos.
3. **Guardar los Cambios en la Base de Datos:**
   - Se actualiza el producto con la ruta del código de barras y el código QR antes de finalizar.

---

3. Asegurar que los Directorios Existan

Si es la primera vez que usas códigos de barras y QR, asegúrate de que los directorios `storage/app/public/barcodes` y `storage/app/public/qrcodes` existen. Si no, puedes crearlos manualmente o ejecutar:

```sh
```

```
php artisan storage:link
```

Esto asegurará que los archivos almacenados en `storage/app/public/` sean accesibles desde `public/storage/`.

---

4. Instalación de la Librería de Código de Barras

Si no tienes instalada la librería para generar códigos de barras (`picqer/php-barcode-generator`), instálala con:

sh

```
composer require picqer/php-barcode-generator
```

Con estos cambios, cada vez que guardes un nuevo producto, se generarán automáticamente su código de barras y código QR. 🚀

Sí, el error `"You need to install the imagick extension to use this back end"` ocurre porque la librería `QrCode` usa `imagick` por defecto para generar imágenes PNG. Para evitar este problema, podemos cambiar el formato de imagen a **SVG o Base64** en lugar de PNG, o usar GD en lugar de Imagick.

Solución 1: Generar el QR en formato SVG (Recomendado)

Si no quieres instalar `imagick`, puedes cambiar la generación del código QR a **SVG**, que no requiere ninguna extensión adicional:

Modificación en el Método `store`

php

```
```

```php
// 2. Generar código QR en formato SVG para evitar problemas con Imagick
$qrCodeData = "Producto: {$product->name}\nPrecio: {$product->price}\nCantidad: {$product->quantity}\nEstado: {$product->status}";
$qrCodeSvg = QrCode::format('svg')->size(200)->generate($qrCodeData);
$qrCodePath = "qrcodes/{$product->id}.svg";
Storage::disk('public')->put($qrCodePath, $qrCodeSvg);
$product->qrcode = $qrCodePath;
```

🔹 **Ventaja:** No requiere la extensión `imagick` y se puede visualizar en navegadores modernos.

Solución 2: Guardar el QR en Base64 (Alternativa)

Si no necesitas almacenar el QR como un archivo, puedes guardarlo en la base de datos en formato **Base64**:

Modificación en el Método `store`

php

```php
$qrCodeBase64 = base64_encode(QrCode::format('png')->size(200)->generate($qrCodeData));
$product->qrcode = $qrCodeBase64;
```

🔹 **Ventaja:** No almacena archivos adicionales, pero necesitarás mostrarlo en la vista con `<img src="{{ $product->qrcode }}">`.

Solución 3: Forzar el Uso de GD en Lugar de Imagick

Si prefieres seguir usando PNG pero sin `imagick`, puedes decirle a `QrCode` que use GD en lugar de Imagick:

Modificación en `store`

php

```php

```

```
$qrCodePath = "qrcodes/{$product->id}.png";
Storage::disk('public')->put($qrCodePath, QrCode::format('png')->size(200)->generate($qrCodeData));
$product->qrcode = $qrCodePath;
```

Y agrega esta línea en `config/qrcode.php` (si existe este archivo de configuración):

php

```
'driver' => 'gd',
```

- **Ventaja:** Sigue usando PNG sin necesidad de `imagick`.

Conclusión

Si no quieres instalar `imagick`, la mejor opción es usar **SVG (Solución 1)**. 🚀

Si prefieres evitar archivos y guardar el QR en la base de datos, usa **Base64 (Solución 2)**.

Si quieres seguir con PNG sin `imagick`, **forza el uso de GD (Solución 3)**.

Elige la que mejor se adapte a tu proyecto y dime si necesitas más ayuda. 😊

Aquí está el método `update` modificado para incluir la generación y actualización del **código de barras (barcode)** y **código QR (qrcode)**, siguiendo la misma lógica del método `store` que proporcionaste.

Método `update` con Barcode y QRCode

php

```
use SimpleSoftwareIO\QrCode\Facades\QrCode;
use Picqer\Barcode\BarcodeGeneratorPNG;
use Illuminate\Support\Facades\Storage;

public function update(Request $request, string $id)
{
```

```
    $request->validate([
        'name' => 'required|string|max:255',
        'description' => 'nullable|string',
        'price' => 'required|numeric|min:0',
        'quantity' => 'required|integer|min:1',
        'category_id' => 'required|exists:categories,id',
        'status' => 'required|in:available,sold,archived',
        'fechvencimiento' => 'nullable|date',
        'images.*' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
    ]);

    $product = Product::findOrFail($id);
    $product->update($request->only(['name', 'description', 'price', 'quantity',
'category_id', 'status', 'fechvencimiento']));

    // 1. Generar código de barras (usando el ID del producto)
    $barcodeGenerator = new BarcodeGeneratorPNG();
    $barcodeData = $barcodeGenerator->getBarcode($product->id,
BarcodeGeneratorPNG::TYPE_CODE_128);
    $barcodePath = "barcodes/{$product->id}.png";
    Storage::disk('public')->put($barcodePath, $barcodeData);
    $product->barcode = $barcodePath;

    // 2. Generar código QR en formato SVG para evitar problemas con Imagick
    $qrCodeData = "Producto: {$product->name}\nPrecio: {$product->price}\nCantidad:
{$product->quantity}\nEstado: {$product->status}";
    $qrCodeSvg = QrCode::format('svg')->size(200)->generate($qrCodeData);
    $qrCodePath = "qrcodes/{$product->id}.svg";
    Storage::disk('public')->put($qrCodePath, $qrCodeSvg);
    $product->qrcode = $qrCodePath;

    // Guardar los datos actualizados en la base de datos
    $product->save();

    // 3. Manejo de imágenes del producto (si se suben nuevas imágenes)
    if ($request->hasFile('images')) {
        foreach ($request->file('images') as $image) {
            $imagePath = $image->store('products', 'public');
            ProductImage::create([
                'product_id' => $product->id,
                'image_path' => $imagePath,
            ]);
```

```
        }
    }

    return redirect()->route('products.index')->with('success', 'Producto
actualizado correctamente.');
}
```

Explicación de los Cambios

1. **Se agregaron los campos `barcode` y `qrcode` al actualizar un producto.**
   - Se genera un **código de barras** basado en el ID del producto y se guarda en `storage/app/public/barcodes/ID.png`.
   - Se genera un **código QR en formato SVG** con los datos del producto y se guarda en `storage/app/public/qrcodes/ID.svg`.
2. **Se usa `Storage::disk('public')->put()` para guardar las imágenes de código de barras y QR.**
   - No depende de `imagick`, evitando errores de compatibilidad.
3. **Las imágenes se manejan correctamente si el usuario sube nuevas imágenes al actualizar el producto.**
   - Se almacenan en `storage/app/public/products/`.

Conclusión

Este código garantiza que cada vez que se **actualiza un producto**, también se regeneren **su código de barras y su código QR**, sin generar conflictos con `imagick`. 🚀

Si necesitas hacer que el **código de barras o QR solo se regeneren si cambian ciertos datos**, dime y podemos optimizarlo aún más. 😊

El error `"Call to a member function isNotEmpty() on null"` ocurre porque `barcode` es una **cadena de texto (string)** y no una colección, por lo que `isNotEmpty()` no se puede usar en este caso.

Corrección del Código

Cambia la condición en la vista para verificar si `barcode` no está vacío usando `!empty($product->barcode)`:

blade

```
<td>
    @if (!empty($product->barcode))
        <img src="{{ asset('storage/' . $product->barcode) }}" class="img-thumbnail"
width="50" alt="Código de Barras de {{ $product->name }}">
    @else
        <span class="text-muted">Sin imagen</span>
    @endif
</td>
```

Corrección para Mostrar el Código QR

Si también deseas mostrar el **código QR** en la tabla, agrégalo de manera similar:

blade

```
<td>
    @if (!empty($product->qrcode))
        <img src="{{ asset('storage/' . $product->qrcode) }}" class="img-thumbnail"
width="50" alt="QR de {{ $product->name }}">
    @else
        <span class="text-muted">Sin QR</span>
    @endif
</td>
```

Explicación del Cambio

1. `$product->barcode` **es un string, no una colección**, por lo que `isNotEmpty()` **no es válido**.
2. **Se usa** `!empty($product->barcode)` para verificar si tiene un valor antes de mostrar la imagen.
3. **Se muestra el código de barras y el QR correctamente usando** `asset('storage/...')`, asumiendo que usaste `php artisan storage:link` para acceder a los archivos almacenados en `storage/app/public/`.

Asegurar que los Archivos sean Accesibles

Si las imágenes no se muestran correctamente, asegúrate de haber ejecutado:

```sh
php artisan storage:link
```

Esto creará un enlace simbólico desde `storage/app/public` a `public/storage`, permitiendo acceder a los archivos subidos.

Resultado Final

Con estos cambios, el código de barras y el código QR se mostrarán correctamente en la vista sin generar errores. 🚀

Aquí tienes el código modificado para cambiar el color de la **fecha de vencimiento** según las condiciones que mencionaste:

- **Rojo (** `text-danger` **)**: Si la fecha de vencimiento ya pasó (es menor a la fecha actual).
- **Naranja (** `text-warning` **)**: Si la fecha de vencimiento está dentro de los próximos 30 días.
- **Sin color adicional**: Si la fecha de vencimiento es mayor a 30 días.

Código Modificado

blade

```
<td>
    @php
        $today = \Carbon\Carbon::today();
        $expirationDate = \Carbon\Carbon::parse($product->fechvencimiento);
        $daysDifference = $today->diffInDays($expirationDate, false);
    @endphp

    <span class="
        @if ($daysDifference < 0) text-danger
        @elseif ($daysDifference <= 30) text-warning
        @endif
    ">
        {{ $product->fechvencimiento ? $product->fechvencimiento : 'No registrada'
}}
    </span>
</td>
```

Explicación del Código

1. **Obtenemos la fecha actual** con `Carbon::today()`.
2. **Convertimos la fecha de vencimiento del producto** en un objeto `Carbon` para poder compararlo.
3. **Calculamos la diferencia en días** entre la fecha actual y la fecha de vencimiento.
    - Si `daysDifference < 0` → La fecha ya pasó, entonces se muestra en **rojo (** `text-danger` **)**.
    - Si `daysDifference <= 30` → La fecha está por vencerse dentro de 30 días, entonces se muestra en **naranja (** `text-warning` **)**.
    - Si la fecha de vencimiento es mayor a 30 días, se muestra normal.
4. **Si no hay fecha de vencimiento**, se muestra el texto `"No registrada"`.

Ejemplo de Resultado

Fecha de Vencimiento	Color
2024-02-01 (pasada)	🔴 Rojo
2024-02-29 (en 22 días)	🟠 Naranja
2024-04-01 (más de 30 días)	⚪ Normal

Asegurar que Carbon está Importado

Si no has importado `Carbon`, agrégalo al inicio de tu archivo Blade con:

blade

```
@php
    use Carbon\Carbon;
@endphp
```

¡Con este código, la fecha de vencimiento se resaltará correctamente según el estado del producto! 🚀

Aquí tienes el código actualizado para mostrar la **fecha de vencimiento** con los estilos de **badge** que mencionaste:

Código Modificado

blade

```
<td>
    @php
        $today = \Carbon\Carbon::today();
        $expirationDate = $product->fechvencimiento ?
\Carbon\Carbon::parse($product->fechvencimiento) : null;
        $daysDifference = $expirationDate ? $today->diffInDays($expirationDate,
false) : null;
```

```
    @endphp

    @if ($expirationDate)
        @if ($daysDifference < 0)
            <span class="badge badge-soft-danger p-2">Vencido</span>
        @elseif ($daysDifference <= 30)
            <span class="badge badge-soft-warning p-2">Por vencer</span>
        @else
            <span class="badge badge-soft-success p-2">Vigente</span>
        @endif
        <br>
        <small>{{ $expirationDate->format('d/m/Y') }}</small>
    @else
        <span class="badge badge-soft-secondary p-2">Sin fecha</span>
    @endif
</td>
```

Explicación de los Cambios

1. **Se usa `Carbon` para manejar la fecha de vencimiento**:
   - Si el producto **no tiene fecha de vencimiento**, se muestra `"Sin fecha"`.
   - Se usa `format('d/m/Y')` para mostrar la fecha en un formato más legible (`DD/MM/YYYY`).
2. **Se asignan los estilos de badge según la diferencia en días**:
   - **Rojo (`badge-soft-danger`)** → `"Vencido"` (Fecha menor a la actual).
   - **Naranja (`badge-soft-warning`)** → `"Por vencer"` (Menos de 30 días para vencer).
   - **Verde (`badge-soft-success`)** → `"Vigente"` (Más de 30 días para vencer).
   - **Gris (`badge-soft-secondary`)** → `"Sin fecha"` (Si no hay fecha de vencimiento).

Ejemplo de Resultado

Fecha de Vencimiento	Estilo
2024-02-01 (pasada)	🔴 **Vencido** (`badge-soft-danger`)
2024-02-29 (en 22 días)	🟠 **Por vencer** (`badge-soft-warning`)
2024-04-01 (más de 30 días)	🟢 **Vigente** (`badge-soft-success`)
No tiene fecha	⚪ **Sin fecha** (`badge-soft-secondary`)

Esto hará que la fecha de vencimiento se vea mucho más clara y visualmente atractiva en la tabla. 🚀

Para agregar la opción de **seleccionar columnas visibles** en tu **DataTable**, necesitas incluir los archivos correctos de la extensión **Buttons** y habilitar la funcionalidad de **column visibility**.

1️⃣ Corrección de la Versión y Agregado de Botones

Tu código actual no incluye los archivos necesarios para la funcionalidad de selección de columnas. **Corrige esto agregando las siguientes librerías:**

- 🔹 **Agregar los Estilos y Scripts Correctos**

```html
<!-- DataTables CSS -->
<link rel="stylesheet"
href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css">
<link rel="stylesheet"
href="https://cdn.datatables.net/buttons/2.3.6/css/buttons.bootstrap5.min.css">

<!-- DataTables JS -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js">
</script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js">
</script>
```

```
<!-- Extensiones para Buttons -->
<script src="https://cdn.datatables.net/buttons/2.3.6/js/dataTables.buttons.min.js">
</script>
<script src="https://cdn.datatables.net/buttons/2.3.6/js/buttons.bootstrap5.min.js">
</script>
<script src="https://cdn.datatables.net/buttons/2.3.6/js/buttons.colVis.min.js">
</script>
```

2️⃣ Configuración del DataTable con Botón para Columnas Visibles

Modifica tu script de inicialización para incluir los botones de **selección de columnas**:

js

```
$(document).ready(function () {
    $('#productsTable').DataTable({
        language: {
            url: '//cdn.datatables.net/plug-ins/1.13.6/i18n/es-ES.json'
        },
        responsive: true,
        autoWidth: false,
        lengthMenu: [10, 25, 50, 75, 100],
        pageLength: 10,
        order: [[0, 'desc']],

        // Habilitar botones
        dom: 'Bfrtip',
        buttons: [
            {
                extend: 'colvis',
                text: 'Seleccionar Columnas',
                className: 'btn btn-primary'
            }
        ]
    });
});
```

3️⃣ Explicación del Código

1. **Se agregaron las librerías correctas:**
   - `dataTables.buttons.min.js`: Para habilitar los botones en DataTable.
   - `buttons.bootstrap5.min.js`: Para que los botones sigan el estilo de Bootstrap 5.
   - `buttons.colVis.min.js`: Para agregar la funcionalidad de **selección de columnas**.
2. `dom: 'Bfrtip'`:
   - `B` → Activa los botones en la tabla.
   - `f` → Habilita la barra de búsqueda.
   - `r` → Muestra la tabla en formato responsivo.
   - `t` → Muestra la tabla.
   - `ip` → Permite la paginación.
3. **Se agregó el botón de visibilidad de columnas:**

```js
buttons: [
    {
        extend: 'colvis',
        text: 'Seleccionar Columnas',
        className: 'btn btn-primary'
    }
]
```

   - `extend: 'colvis'` → Activa la selección de columnas.
   - `text: 'Seleccionar Columnas'` → Nombre del botón.
   - `className: 'btn btn-primary'` → Aplica el estilo de Bootstrap.

4️⃣ Resultado Final

✅ **Se verá un botón llamado** `"Seleccionar Columnas"` **en la parte superior** de la tabla.
✅ **Al hacer clic, permitirá ocultar/mostrar columnas** según lo necesites.
✅ **Se mantienen las demás funcionalidades del DataTable, como paginación y búsqueda.**

Esto asegurará que el **DataTable con selección de columnas** funcione correctamente sin problemas de versión. 🚀🎯

Para que en el **listado de selección de columnas** se resalten correctamente las columnas activas e inactivas, debes aplicar estilos adicionales para que las opciones se vean con diferentes colores o efectos visuales.

🔹 Solución: Personalizar el Estilo del Selector de Columnas

DataTables usa una clase llamada `.dt-button-collection` para la lista de selección de columnas. Debes agregar estilos en tu archivo CSS o en un `<style>` dentro de la vista:

html

```
<style>
    /* Estilos personalizados para el menú de selección de columnas */
    .dt-button-collection .dt-button {
        background-color: white !important;
        color: black !important;
        border: 1px solid #ccc;
        padding: 5px 10px;
        display: block;
        text-align: left;
    }

    /* Cuando una columna está activa (visible) */
    .dt-button-collection .dt-button.active {
        background-color: #28a745 !important; /* Verde */
        color: white !important;
        font-weight: bold;
    }

    /* Cuando una columna está desactivada (oculta) */
    .dt-button-collection .dt-button:not(.active) {
        background-color: #dc3545 !important; /* Rojo */
        color: white !important;
        font-weight: bold;
```

```
    }
</style>
```

🔹 Ajuste en el Código JavaScript

Agrega la configuración para que los botones de visibilidad de columnas cambien de color según su estado:

js

```
$(document).ready(function () {
    var table = $('#productsTable').DataTable({
        language: {
            url: '//cdn.datatables.net/plug-ins/1.13.6/i18n/es-ES.json'
        },
        responsive: true,
        autoWidth: false,
        lengthMenu: [10, 25, 50, 75, 100],
        pageLength: 10,
        order: [[0, 'desc']],

        // Habilitar botones
        dom: 'Bfrtip',
        buttons: [
            {
                extend: 'colvis',
                text: 'Seleccionar Columnas',
                className: 'btn btn-primary',
                postfixButtons: ['colvisRestore']
            }
        ]
    });

    // Personalizar los estilos al cambiar visibilidad de columnas
    table.on('buttons-action', function (e, buttonApi, dataTable, node, config) {
        setTimeout(function () {
            $('.dt-button-collection .dt-button').each(function () {
                var buttonText = $(this).text();
                var isActive = $(this).hasClass('active');
```

```
                if (isActive) {
                    $(this).css({
                        "background-color": "#28a745", // Verde si está activo
                        "color": "white",
                        "font-weight": "bold"
                    });
                } else {
                    $(this).css({
                        "background-color": "#dc3545", // Rojo si está desactivado
                        "color": "white",
                        "font-weight": "bold"
                    });
                }
            });
        }, 10);
    });
});
```

🔹 Explicación de los Cambios

1. **Se agregaron estilos personalizados en CSS**:
   - `active` **(Verde -** `#28a745`**)**: Indica que la columna está visible.
   - **No** `active` **(Rojo -** `#dc3545`**)**: Indica que la columna está oculta.
2. **Se agregó un evento en JavaScript (** `buttons-action` **)**:
   - Detecta cuando el usuario hace clic en un botón de visibilidad de columna.
   - Aplica los estilos correspondientes a cada botón.
3. **Se usa** `setTimeout` :
   - Para asegurarnos de que los cambios se apliquen después de que DataTables actualice el estado de los botones.

🔹 Resultado Final

✅ **Las columnas activas se verán en VERDE** en el listado de selección.
✅ **Las columnas ocultas se verán en ROJO**, facilitando la identificación.
✅ **El diseño es más intuitivo y fácil de usar.**

Con estos cambios, ahora tendrás una mejor experiencia de usuario al seleccionar qué columnas quieres ver en el DataTable. 🚀🎯

Sí, para manejar **grandes volúmenes de datos** (como 10,000 registros o más) en **DataTables**, la mejor práctica es utilizar **Server-side Processing** en lugar de cargar todos los registros en el navegador. Esto optimiza el rendimiento y evita que la página se vuelva lenta.

1️⃣ Configurar DataTables para Server-side Processing

En tu **JavaScript**, activa `serverSide: true` y configura la URL que servirá los datos:

js

```
$(document).ready(function () {
    $('#productsTable').DataTable({
        processing: true,
        serverSide: true,
        ajax: "{{ route('products.data') }}", // Ruta a la API que devolverá los
datos
        language: {
            url: '//cdn.datatables.net/plug-ins/1.13.6/i18n/es-ES.json'
        },
        responsive: true,
        autoWidth: false,
        lengthMenu: [10, 25, 50, 100, 500, 1000], // Opciones para mostrar más datos
        pageLength: 10, // Número de registros por página
        order: [[0, 'desc']],
        columns: [
            { data: 'id', name: 'id' },
            { data: 'images', name: 'images', orderable: false, searchable: false },
            { data: 'name', name: 'name' },
            { data: 'category', name: 'category' },
            { data: 'price', name: 'price' },
            { data: 'quantity', name: 'quantity' },
            { data: 'status', name: 'status' },
```

```
            { data: 'fechvencimiento', name: 'fechvencimiento' },
            { data: 'barcode', name: 'barcode', orderable: false, searchable: false
},
            { data: 'qrcode', name: 'qrcode', orderable: false, searchable: false },
            { data: 'actions', name: 'actions', orderable: false, searchable: false
}
        ]
    });
});
```

2️⃣ Crear la Ruta en `web.php`

Define una ruta para procesar los datos en Laravel:

php

```php
Route::get('/products/data', [ProductController::class, 'getData'])-
>name('products.data');
```

3️⃣ Implementar la Lógica en el Controlador

En tu `ProductController`, agrega un nuevo método `getData` para manejar la paginación en el servidor:

php

```php
use Illuminate\Http\Request;
use App\Models\Product;
use Yajra\DataTables\Facades\DataTables;

public function getData(Request $request)
{
    $query = Product::with('category')
        ->select('products.*');
```

```
    return DataTables::of($query)
        ->addColumn('images', function ($product) {
            if ($product->images->isNotEmpty()) {
                return '<img src="' . asset('storage/' . $product->images->first()->image_path) . '" class="img-thumbnail" width="50" />';
            }
            return '<span class="text-muted">Sin imagen</span>';
        })
        ->addColumn('category', function ($product) {
            return $product->category->name ?? 'Sin categoría';
        })
        ->addColumn('barcode', function ($product) {
            return !empty($product->barcode) ? '<img src="' . asset('storage/' . $product->barcode) . '" class="img-thumbnail" width="80" />' : '<span class="text-muted">Sin imagen</span>';
        })
        ->addColumn('qrcode', function ($product) {
            return !empty($product->qrcode) ? '<img src="' . asset('storage/' . $product->qrcode) . '" class="img-thumbnail" width="80" />' : '<span class="text-muted">Sin imagen</span>';
        })
        ->addColumn('actions', function ($product) {
            return '
                <a href="' . route('products.edit', $product->id) . '" class="btn btn-warning btn-sm">Editar</a>
                <button class="btn btn-danger btn-sm delete-btn" data-id="' . $product->id . '">Eliminar</button>
                <form id="delete-form-' . $product->id . '" action="' . route('products.destroy', $product->id) . '" method="POST" class="d-none">
                    ' . csrf_field() . '
                    ' . method_field('DELETE') . '
                </form>
            ';
        })
        ->rawColumns(['images', 'barcode', 'qrcode', 'actions']) // Permitir HTML en estas columnas
        ->make(true);
}
```

4️⃣ Explicación del Código

🔹 Backend (Laravel)

1. **Se usa `Product::with('category')` para optimizar la consulta y evitar el problema de N+1 queries.**
2. **Se agregan columnas personalizadas (`images`, `barcode`, `qrcode`, `actions`) para renderizar HTML.**
3. **Se usa `->rawColumns(['images', 'barcode', 'qrcode', 'actions'])` para que se muestren correctamente las imágenes y botones.**
4. **El método `make(true)` devuelve los datos en JSON, lo que permite a DataTables manejar la paginación en el servidor.**

🔹 Frontend (JavaScript)

1. **Se define `serverSide: true`** para que la paginación y búsqueda se hagan en el backend.
2. **Se configura `ajax: "{{ route('products.data') }}"`** para obtener los datos de la API de Laravel.
3. **Las columnas se configuran en el arreglo `columns`, asegurando que algunas (`images`, `barcode`, `qrcode`, `actions`) no sean ordenables ni buscables.**

5️⃣ Resultado Final

✅ **DataTable cargará solo los registros necesarios por página.**
✅ **Optimizado para manejar 10K+ registros sin afectar el rendimiento.**
✅ **Búsqueda y paginación en el servidor, sin recargar toda la página.**
✅ **Menor consumo de memoria en el navegador y mejor experiencia de usuario.**

¡Con esta implementación, tu **DataTable JS soportará grandes volúmenes de datos sin problemas!** 🚀🎯

Corrección y Mejora de la Implementación

Voy a corregir y optimizar tu configuración **server-side** en **DataTables** para la vista `transactions.index`.

🔹 **Problemas detectados y mejoras:**

1. **Inconsistencia en la columna `user.name`:**
   - En el controlador, se usa `user_name`, pero en la tabla se llama `user.name`.
   - **Solución:** Unificarlo en `user_name`.
2. **Las columnas de "Ver Detalle" y "PDF" no usan el formato del backend.**
   - En el backend ya generaste `detalle` y `pdf`, pero en el frontend los vuelves a crear.
   - **Solución:** Usar los valores ya procesados en el backend.
3. **Formato de fecha optimizado desde el backend.**
   - En el JS, eliminamos `render` para `created_at`, ya que en el backend se formatea correctamente.

1️⃣ Modificación en el `TransactionController`

Modifica la lógica del método `index` para **asegurar la correcta relación con `user`** y optimizar la estructura del JSON devuelto.

php

```php
public function index(Request $request)
{
    $startDate = $request->input('start_date');
    $endDate = $request->input('end_date');

    $query = Transaction::with('user');

    if ($startDate && $endDate) {
        $query->whereBetween('created_at', [
            Carbon::parse($startDate)->startOfDay(),
            Carbon::parse($endDate)->endOfDay()
        ]);
    }

    if ($request->ajax()) {
        return DataTables::of($query)
            ->editColumn('created_at', function ($transaction) {
```

```
                return Carbon::parse($transaction->created_at)->format('d/m/Y H:i');
            })
            ->addColumn('detalle', function ($transaction) {
                return '<button class="btn btn-info btn-sm view-detail" data-id="' .
$transaction->id . '">Ver Detalle</button>';
            })
            ->addColumn('pdf', function ($transaction) {
                $pdfUrl = $this->getPdfUrl($transaction);
                return $pdfUrl ? '<a href="' . $pdfUrl . '" target="_blank"
class="btn btn-danger btn-sm">PDF</a>' : '<span class="text-muted">No
disponible</span>';
            })
            ->addColumn('user_name', function ($transaction) {
                return $transaction->user ? $transaction->user->name : '<span
class="text-muted">Sin usuario</span>';
            })
            ->rawColumns(['detalle', 'pdf', 'user_name'])
            ->toJson();
    }

    return view('transactions.index');
}
```

✅ **Se asegura que `user_name` sea correcto y no haya errores en la vista.**
✅ **Se devuelve `detalle` y `pdf` ya formateados en el backend, sin procesarlos de nuevo en el frontend.**

2️⃣ Configuración del DataTable en `transactions.index`

Ahora **aprovechamos los datos ya formateados en el backend** en lugar de repetir la lógica en JavaScript.

js

```
$(document).ready(function () {
    let table = $('#transactionsTable').DataTable({
        processing: true,
        serverSide: true,
        ajax: {
```

```
            url: "{{ route('transactions.index') }}",
            data: function (d) {
                d.start_date = $('#start_date').val();
                d.end_date = $('#end_date').val();
            }
        },
        columns: [
            { data: 'id', name: 'id' },
            { data: 'type', name: 'type' },
            { data: 'amount', name: 'amount' },
            { data: 'reference_id', name: 'reference_id' },
            { data: 'description', name: 'description' },
            { data: 'user_name', name: 'user.name', orderable: false },
            { data: 'created_at', name: 'created_at' },
            { data: 'detalle', name: 'detalle', orderable: false, searchable: false
},
            { data: 'pdf', name: 'pdf', orderable: false, searchable: false }
        ],
        language: {
            url: '//cdn.datatables.net/plug-ins/1.13.6/i18n/es-ES.json'
        },
        responsive: true,
        autoWidth: false,
        lengthMenu: [10, 25, 50, 75, 100],
        pageLength: 10,
        order: [[0, 'desc']],
        dom: 'Bfrtip',
        buttons: [
            {
                extend: 'colvis',
                text: 'Seleccionar Columnas',
                className: 'btn btn-primary',
                postfixButtons: ['colvisRestore']
            }
        ]
    });

    // Filtrar por fecha cuando se cambian los inputs de fecha
    $('#start_date, #end_date').change(function () {
        table.draw();
    });
});
```

3️⃣ Explicación de los Cambios

1. **Se unificó** `user_name` **en el backend y el frontend** para evitar errores en `user.name`.
2. **Se eliminó la lógica de renderizado en JS para** `detalle` **y** `pdf`, ya que el backend ya devuelve los valores listos en HTML.
3. **Se agregó el evento** `$('#start_date, #end_date').change()`
   - Esto **hace que el DataTable se recargue automáticamente** cuando se selecciona un nuevo rango de fechas.
4. **Eliminamos el** `render` **en** `created_at` porque ya lo formateamos en el backend con `Carbon`.

4️⃣ ¿Cómo se ve ahora?

✅ **Carga los datos por AJAX de manera eficiente.**
✅ **La paginación y búsqueda ahora son server-side, optimizando grandes volúmenes de datos.**
✅ **Filtro de fechas funcional sin necesidad de recargar la página.**
✅ **Selección de columnas activada sin errores de visualización.**

🚀 ¡Con esto, tu vista `transactions.index` ahora es 100% optimizada para grandes volúmenes de datos! 🎯

Para ejecutar el `DatabaseSeeder.php` en **Laravel**, sigue estos pasos:

1️⃣ Asegúrate de que el Seeder está Configurado Correctamente

Abre el archivo `database/seeders/DatabaseSeeder.php` y verifica que tenga la estructura correcta. Debería verse algo así:

```php
```

```php
namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        // Llamar a otros Seeders
        $this->call([
            UsersTableSeeder::class,
            CategoriesTableSeeder::class,
            ProductsTableSeeder::class,
            TransactionsTableSeeder::class,
        ]);
    }
}
```

Asegúrate de que en `DatabaseSeeder.php` estás llamando correctamente a los seeders individuales.

2️⃣ Ejecutar el Seeder Principal

En la terminal, ejecuta el siguiente comando para ejecutar **todos los seeders**:

```sh
php artisan db:seed
```

✅ Esto ejecutará `DatabaseSeeder.php` y llamará a todos los seeders listados dentro.

3️⃣ Opcional: Ejecutar un Seeder Específico

Si solo quieres ejecutar un **Seeder individual**, usa:

sh

```
```

Para **mantener el estado del modo oscuro (dark mode) y fullscreen** entre vistas en Laravel, podemos usar **localStorage** en JavaScript. Esto nos permitirá recordar la configuración del usuario incluso después de cambiar de página.

1️⃣ Solución para Mantener el Dark Mode

Cada vez que el usuario active el **modo oscuro**, debemos guardar la configuración en `localStorage`. Cuando se cargue la página, revisamos si el usuario había activado **dark mode** previamente y aplicamos los cambios.

🔹 Modificación del Script para Dark Mode

js

```
document.addEventListener("DOMContentLoaded", function () {
    const htmlElement = document.documentElement;
    const darkModeToggle = document.getElementById("darkModeToggle"); // Reemplaza
con el ID real del botón

    // Cargar configuración guardada en localStorage
    if (localStorage.getItem("darkMode") === "enabled") {
        htmlElement.setAttribute("data-layout-mode", "dark");
    }

    // Evento para cambiar el modo
    darkModeToggle.addEventListener("click", function () {
        if (htmlElement.getAttribute("data-layout-mode") === "dark") {
            htmlElement.setAttribute("data-layout-mode", "light");
            localStorage.setItem("darkMode", "disabled");
        } else {
            htmlElement.setAttribute("data-layout-mode", "dark");
            localStorage.setItem("darkMode", "enabled");
        }
```

```
    });
});
```

✅ **¿Qué hace este código?**

1. Al cargar la página, revisa si `localStorage.getItem("darkMode")` está activado.
2. Si el modo oscuro estaba activado, aplica `data-layout-mode="dark"`.
3. Al hacer clic en el botón de dark mode, cambia la configuración y **guarda el estado en localStorage**.

2️⃣ Solución para Mantener el Fullscreen

Cuando el usuario activa **fullscreen**, se le agrega la clase `fullscreen-enable` al `<body>`. Para recordar este estado, guardamos en `localStorage` y restauramos la clase cuando la página se recarga.

🔹 Modificación del Script para Fullscreen

```js
document.addEventListener("DOMContentLoaded", function () {
    const body
```

Aquí tienes el código corregido para que **el modo oscuro y el fullscreen se mantengan al navegar entre páginas** utilizando **localStorage**.

1️⃣ Solución para Mantener el Modo Oscuro

El botón de **modo oscuro** tiene la clase `light-dark-mode`. Cuando el usuario haga clic, se cambiará el atributo `data-layout-mode` del `<html>` y se guardará la configuración en **localStorage**.

🔹 Código Mejorado para Dark Mode

```js
```

```
document.addEventListener("DOMContentLoaded", function () {
    const htmlElement = document.documentElement;
    const darkModeToggle = document.querySelector(".light-dark-mode"); // Botón de
Dark Mode
    const icon = darkModeToggle.querySelector("i");

    // Comprobar si hay una configuración guardada
    if (localStorage.getItem("darkMode") === "enabled") {
        htmlElement.setAttribute("data-layout-mode", "dark");
        icon.classList.remove("bx-moon");
        icon.classList.add("bx-sun"); // Cambia el icono a sol
    } else {
        htmlElement.setAttribute("data-layout-mode", "light");
        icon.classList.remove("bx-sun");
        icon.classList.add("bx-moon"); // Cambia el icono a luna
    }

    // Evento para alternar entre modos
    darkModeToggle.addEventListener("click", function () {
        if (htmlElement.getAttribute("data-layout-mode") === "dark") {
            htmlElement.setAttribute("data-layout-mode", "light");
            localStorage.setItem("darkMode", "disabled");
            icon.classList.remove("bx-sun");
            icon.classList.add("bx-moon");
        } else {
            htmlElement.setAttribute("data-layout-mode", "dark");
            localStorage.setItem("darkMode", "enabled");
            icon.classList.remove("bx-moon");
            icon.classList.add("bx-sun");
        }
    });
});
```

✅ **¿Qué hace este código?**

1. **Verifica el estado guardado en** `localStorage` y aplica el modo oscuro si estaba activado.
2. **Cambia el icono del botón** (`bx-moon` → `bx-sun` y viceversa) según el estado del dark mode.

3. **Guarda la configuración en** `localStorage` para mantener el estado al cambiar de página.

2️⃣ Solución para Mantener el Fullscreen

El botón de **pantalla completa** tiene el atributo `data-toggle="fullscreen"`. Cuando el usuario lo activa, el `<body>` recibe la clase `fullscreen-enable`, pero se pierde al cambiar de vista.

🔹 Código Mejorado para Fullscreen

```js
document.addEventListener("DOMContentLoaded", function () {
    const fullscreenToggle = document.querySelector("[data-toggle='fullscreen']");
    const bodyElement = document.body;

    // Comprobar si estaba en fullscreen antes de cambiar de página
    if (localStorage.getItem("fullscreen") === "enabled") {
        bodyElement.classList.add("fullscreen-enable");
        if (!document.fullscreenElement) {
            document.documentElement.requestFullscreen();
        }
    }

    // Evento para alternar fullscreen
    fullscreenToggle.addEventListener("click", function () {
        if (!document.fullscreenElement) {
            document.documentElement.requestFullscreen();
            bodyElement.classList.add("fullscreen-enable");
            localStorage.setItem("fullscreen", "enabled");
        } else {
            document.exitFullscreen();
            bodyElement.classList.remove("fullscreen-enable");
            localStorage.setItem("fullscreen", "disabled");
        }
    });

    // Detectar salida de fullscreen y actualizar el localStorage
```

```
    document.addEventListener("fullscreenchange", function () {
        if (!document.fullscreenElement) {
            bodyElement.classList.remove("fullscreen-enable");
            localStorage.setItem("fullscreen", "disabled");
        }
    });
});
```

✅ **¿Qué hace este código?**

1. **Guarda el estado de fullscreen en** `localStorage`.
2. **Si el usuario estaba en fullscreen antes de cambiar de página, lo reactiva automáticamente.**
3. **Detecta si el usuario sale de fullscreen manualmente** y actualiza `localStorage`.

3️⃣ Implementación Final

📌 **Coloca este código dentro de un archivo** `app.js` **o en un** `<script>` **en tu plantilla base (**`layout.blade.php`**).**
📌 **Asegúrate de incluir** `localStorage.clear()` **solo si necesitas resetear configuraciones en algún punto.**

Con esto, el **modo oscuro y fullscreen se mantendrán aunque el usuario navegue entre páginas.** 🚀🎯

Es posible que el **template tenga código propio que maneje el modo oscuro y fullscreen**, lo que podría estar interfiriendo con el código que estamos agregando.

Pasos para Revisar Conflictos

Para encontrar el conflicto, sigue estos pasos:

1️⃣ Revisa el Código del Template en el Navegador

1. **Abre la Consola del Desarrollador (**`F12` **en Chrome/Firefox).**

2. **Ve a la pestaña "Elements"** y busca la etiqueta `<html>`.
   - ¿Cambia `data-layout-mode` cuando haces clic en el botón?
   - ¿Se modifica automáticamente al cambiar de vista?
3. **Ve a la pestaña "Console"** e ingresa este comando:

```js
document.documentElement.getAttribute("data-layout-mode");
```

   - Si devuelve `"dark"` y la página sigue en **light**, hay un **script que está forzando light mode**.

2️⃣ Revisa los Scripts que Controlan el Modo Oscuro

1. **Ve a la pestaña "Sources" en las herramientas de desarrollador.**
2. **Busca archivos JS del template** (por ejemplo, `theme.js`, `layout.js` o `settings.js`).
3. **Filtra por "darkMode" en los scripts:**
   - Presiona `CTRL + F` y busca `"darkMode"` o `"data-layout-mode"`.
   - Si encuentras un código que **sobrescribe** el atributo en `<html>`, anótalo.

3️⃣ Verifica si el Template Usa `localStorage`

Ejecuta en la consola:

```js
console.log(localStorage);
```

Si ves algo como:

```json
```

```
{
    "layout-mode": "light"
}
```

Eso significa que el template ya está **guardando y sobrescribiendo el modo oscuro**.

Prueba eliminando su configuración con:

```js
localStorage.removeItem("layout-mode");
```

Luego recarga la página y prueba de nuevo el botón de **dark mode**.

4️⃣ Solución: Deshabilitar el Código del Template

Si descubres que el template está sobrescribiendo el modo oscuro, intenta **anular su código** con este snippet:

```js
document.addEventListener("DOMContentLoaded", function () {
    // Eliminar cualquier interferencia del template
    localStorage.removeItem("layout-mode");

    const htmlElement = document.documentElement;
    const darkModeToggle = document.querySelector(".light-dark-mode");
    const icon = darkModeToggle.querySelector("i");

    if (localStorage.getItem("darkMode") === "enabled") {
        htmlElement.setAttribute("data-layout-mode", "dark");
        icon.classList.replace("bx-moon", "bx-sun");
    } else {
        htmlElement.setAttribute("data-layout-mode", "light");
        icon.classList.replace("bx-sun", "bx-moon");
    }

    darkModeToggle.addEventListener("click", function () {
        if (htmlElement.getAttribute("data-layout-mode") === "dark") {
            htmlElement.setAttribute("data-layout-mode", "light");
```

```
            localStorage.setItem("darkMode", "disabled");
            icon.classList.replace("bx-sun", "bx-moon");
        } else {
            htmlElement.setAttribute("data-layout-mode", "dark");
            localStorage.setItem("darkMode", "enabled");
            icon.classList.replace("bx-moon", "bx-sun");
        }
    });
});
```

✅ **¿Qué hace este código?**

- **Elimina `localStorage['layout-mode']` que el template usa para forzar `light`.**
- **Evita que el template sobrescriba `data-layout-mode`.**
- **Mantiene la configuración de dark mode al cambiar de vista.**

Prueba esto y dime si sigues viendo problemas. 🚀🔥

Parece que el problema con el **modo oscuro y fullscreen** está relacionado con el `app.js` **del template**, que maneja dinámicamente los atributos en `<html>` y usa `sessionStorage` para persistir configuraciones.

Voy a explicarte **cómo modificarlo correctamente** sin afectar otras funcionalidades del template.

🔍 Problema Identificado

1. **Modo Oscuro (Dark Mode)**
   - El template está usando `sessionStorage` para almacenar `data-layout-mode` en lugar de `localStorage`.
   - La función `F("data-layout-mode", "dark/light", ..., 1)` está sobrescribiendo el valor y restableciendo el modo **"light"** al cambiar de página.
2. **Fullscreen**
   - El `app.js` usa un evento para **agregar** `fullscreen-enable` al `<body>`, pero no lo guarda en `sessionStorage` ni `localStorage`.

- Esto provoca que **fullscreen se desactive al cambiar de página.**

✅ Solución para Mantener Dark Mode

1️⃣ Modificar `app.js` para Usar `localStorage` en Lugar de `sessionStorage`

Busca esta parte en `app.js` (alrededor de donde se maneja `data-layout-mode`):

```js
(i = document.querySelectorAll(".light-dark-mode")) &&
    i.length &&
    i[0].addEventListener("click", function (e) {
        l.hasAttribute("data-layout-mode") &&
        "dark" == l.getAttribute("data-layout-mode")
            ? F("data-layout-mode", "light", "layout-mode-light", l)
            : F("data-layout-mode", "dark", "layout-mode-dark", l);
    });
```

Reemplázalo con este código para persistir el modo oscuro en `localStorage`:

```js
document.addEventListener("DOMContentLoaded", function () {
    const htmlElement = document.documentElement;
    const darkModeToggle = document.querySelector(".light-dark-mode");

    // Comprobar si hay una configuración guardada
    if (localStorage.getItem("darkMode") === "enabled") {
        htmlElement.setAttribute("data-layout-mode", "dark");
    } else {
        htmlElement.setAttribute("data-layout-mode", "light");
    }

    // Evento para cambiar el modo oscuro
    darkModeToggle.addEventListener("click", function () {
        if (htmlElement.getAttribute("data-layout-mode") === "dark") {
            htmlElement.setAttribute("data-layout-mode", "light");
            localStorage.setItem("darkMode", "disabled");
```

```
        } else {
            htmlElement.setAttribute("data-layout-mode", "dark");
            localStorage.setItem("darkMode", "enabled");
        }
    });
});
```

✅ **¿Qué hace este código?**

- Almacena el estado del **modo oscuro en** `localStorage`, evitando que el template lo sobrescriba con `sessionStorage`.
- Se ejecuta en `DOMContentLoaded` para **restaurar la configuración antes de que el template sobrescriba valores.**

✅ Solución para Mantener Fullscreen

Busca esta parte en `app.js`:

js

```
(i = document.querySelector('[data-toggle="fullscreen"]')) &&
    i.addEventListener("click", function (e) {
        e.preventDefault(),
            document.body.classList.toggle("fullscreen-enable"),
            document.fullscreenElement ||
            document.mozFullScreenElement ||
            document.webkitFullscreenElement
                ? document.cancelFullScreen
                    ? document.cancelFullScreen()
                    : document.mozCancelFullScreen
                    ? document.mozCancelFullScreen()
                    : document.webkitCancelFullScreen &&
                      document.webkitCancelFullScreen()
                : document.documentElement.requestFullscreen
                ? document.documentElement.requestFullscreen()
                : document.documentElement.mozRequestFullScreen
                ? document.documentElement.mozRequestFullScreen()
                : document.documentElement.webkitRequestFullscreen &&
                  document.documentElement.webkitRequestFullscreen(
```

```
                    Element.ALLOW_KEYBOARD_INPUT
                );
    });
```

Reemplázalo con este código:

```js
document.addEventListener("DOMContentLoaded", function () {
    const fullscreenToggle = document.querySelector("[data-toggle='fullscreen']");
    const bodyElement = document.body;

    // Comprobar si estaba en fullscreen antes de cambiar de página
    if (localStorage.getItem("fullscreen") === "enabled") {
        bodyElement.classList.add("fullscreen-enable");
        if (!document.fullscreenElement) {
            document.documentElement.requestFullscreen();
        }
    }

    // Evento para alternar fullscreen
    fullscreenToggle.addEventListener("click", function () {
        if (!document.fullscreenElement) {
            document.documentElement.requestFullscreen();
            bodyElement.classList.add("fullscreen-enable");
            localStorage.setItem("fullscreen", "enabled");
        } else {
            document.exitFullscreen();
            bodyElement.classList.remove("fullscreen-enable");
            localStorage.setItem("fullscreen", "disabled");
        }
    });

    // Detectar salida de fullscreen y actualizar el localStorage
    document.addEventListener("fullscreenchange", function () {
        if (!document.fullscreenElement) {
            bodyElement.classList.remove("fullscreen-enable");
            localStorage.setItem("fullscreen", "disabled");
        }
    });
});
```

✅ **¿Qué hace este código?**

- Guarda el estado de **fullscreen en** `localStorage`.
- Si el usuario estaba en **fullscreen antes de cambiar de página**, lo reactiva automáticamente.
- Detecta si el usuario **sale de fullscreen manualmente** y actualiza `localStorage`.

📌 Implementación Final

1. **Edita** `app.js` y reemplaza las partes indicadas con los códigos corregidos.
2. **Guarda los cambios y recarga la página** (`Ctrl + Shift + R` para forzar la recarga).
3. **Prueba los botones**:
    - Activa el **modo oscuro** y cambia de página. **Debe mantenerse activado**.
    - Activa **fullscreen** y cambia de página. **Debe seguir en fullscreen**.

🔎 ¿Por qué Funciona?

- El template estaba usando `sessionStorage`, lo cual **reseteaba los valores al cambiar de página**.
- Ahora usamos `localStorage`, que **persiste entre sesiones y navegación**.
- Se ejecuta en `DOMContentLoaded`, asegurando que los valores se restauren **antes** de que el template sobrescriba cualquier ajuste.

🚀 **¡Con esta solución, el dark mode y fullscreen se mantendrán entre páginas sin conflictos!** 🎯

El problema con **fullscreen** es que **los navegadores no permiten mantenerlo entre páginas automáticamente**, debido a restricciones de seguridad. Sin embargo, podemos hacer que **se reactive automáticamente cuando el usuario cambia de vista**.

✅ Solución para Mantener Fullscreen

El navegador **no permite mantener fullscreen entre páginas**, pero podemos:

1. **Guardar en `localStorage` si el usuario activó fullscreen.**
2. **Detectar cambios de página y volver a activarlo automáticamente.**

🔹 1️⃣ Código Mejorado para Fullscreen

Reemplaza el código de fullscreen en `app.js` con este:

```js
document.addEventListener("DOMContentLoaded", function () {
    const fullscreenToggle = document.querySelector("[data-toggle='fullscreen']");
    const bodyElement = document.body;

    // Comprobar si fullscreen estaba activado antes de cambiar de página
    if (localStorage.getItem("fullscreen") === "enabled") {
        if (!document.fullscreenElement) {
            document.documentElement.requestFullscreen().then(() => {
                bodyElement.classList.add("fullscreen-enable");
            }).catch(err => {
                console.warn("No se pudo activar fullscreen automáticamente:", err);
            });
        }
    }

    // Evento para alternar fullscreen manualmente
    fullscreenToggle.addEventListener("click", function () {
        if (!document.fullscreenElement) {
            document.documentElement.requestFullscreen();
            bodyElement.classList.add("fullscreen-enable");
            localStorage.setItem("fullscreen", "enabled");
        } else {
            document.exitFullscreen();
            bodyElement.classList.remove("fullscreen-enable");
            localStorage.setItem("fullscreen", "disabled");
        }
    });
```

```
    // Detectar salida de fullscreen y actualizar el estado en localStorage
    document.addEventListener("fullscreenchange", function () {
        if (!document.fullscreenElement) {
            bodyElement.classList.remove("fullscreen-enable");
            localStorage.setItem("fullscreen", "disabled");
        }
    });
});
```

🔹 2️⃣ ¿Cómo Funciona?

1. **Guarda en `localStorage` si fullscreen estaba activado.**
2. **Cuando se carga la página, intenta volver a activar fullscreen.**
3. **Si el usuario sale manualmente de fullscreen, lo desactiva en `localStorage`.**
4. **Muestra un `console.warn` si el navegador bloquea la activación automática (seguridad del navegador).**

🔎 ¿Qué Problema Puede Ocurrir?

Algunos navegadores **bloquean la reactivación automática** de fullscreen **sin interacción del usuario**.
Si esto sucede:

- El usuario debe hacer clic en algún botón para reactivar fullscreen manualmente.

✅ **Pero al menos guardará el estado y recordará que estaba en fullscreen.**

🚀 **¡Ahora fullscreen se mantendrá entre páginas siempre que el navegador lo permita!** 🎯

Para personalizar la página **404 (Página No Encontrada)** en **Laravel 10**, sigue estos pasos:

1️⃣ Crear la Vista Personalizada

Laravel maneja los errores en `resources/views/errors/`, por lo que debemos **crear o editar el archivo 404.blade.php**.

📌 Ubicación:

📂 `resources/views/errors/404.blade.php`

Si el archivo no existe, créalo con el siguiente contenido **personalizable**:

blade

```
@extends('layouts.app') {{-- O tu layout base --}}

@section('title', 'Página No Encontrada')

@section('content')
<div class="d-flex flex-column align-items-center justify-content-center"
style="height: 100vh;">
    <h1 class="display-1 fw-bold text-danger">404</h1>
    <h3 class="mb-4">Oops! Página no encontrada.</h3>
    <p class="text-muted">La página que buscas no existe o fue movida.</p>
    <a href="{{ url('/') }}" class="btn btn-primary mt-3">Volver al inicio</a>
</div>
@endsection
```

✅ **Puedes personalizar el diseño como desees.**

2️⃣ Configurar Laravel para Usar la Página Personalizada

Laravel maneja errores desde el `Handler.php`. Para asegurarnos de que nuestra vista **404.blade.php** sea utilizada, edita el archivo:

📂 `app/Exceptions/Handler.php`

- Busca la función `render()` y agrega o modifica el siguiente código:

php

```php
public function render($request, Throwable $exception)
{
    if ($this->isHttpException($exception)) {
        if ($exception->getStatusCode() == 404) {
            return response()->view('errors.404', [], 404);
        }
    }
    return parent::render($request, $exception);
}
```

✅ **Esto asegurará que Laravel use nuestra página personalizada cuando ocurra un error 404.**

3️⃣ Limpiar Caché y Probar

Ejecuta los siguientes comandos para asegurarte de que los cambios se reflejen correctamente:

```sh
php artisan cache:clear
php artisan view:clear
```

Luego, prueba ingresando a una **URL inexistente**, por ejemplo:

```perl
http://tudominio.com/esto-no-existe
```

👀 **Debería mostrar la página personalizada 404.**

🎨 Opcional: Agregar Estilos o Animaciones

Si deseas agregar **animaciones o mejoras visuales**, puedes usar CSS o Bootstrap.

Ejemplo con una imagen de error y animación:

blade

```blade
<div class="text-center">
    <img src="{{ asset('images/404.png') }}" alt="404" class="img-fluid w-50">
    <h1 class="display-4 text-danger">404</h1>
    <p class="lead">¡Oops! No encontramos esta página.</p>
    <a href="{{ url('/') }}" class="btn btn-outline-primary">Regresar al Inicio</a>
</div>
```

📌 **Ubica la imagen en:**
📂 `public/images/404.png`

✅ Resumen Final

1. **Crear la vista personalizada en:** `resources/views/errors/404.blade.php`
2. **Modificar** `Handler.php` **para usar la nueva vista.**
3. **Limpiar caché y probar con una URL inexistente.**

🚀 **¡Listo! Ahora tu Laravel tiene una página 404 personalizada y estilizada.** 🎨🎯

Aquí tienes el código del template corregido con **rutas dinámicas de Laravel** y **texto en español**.

📌 Correcciones y Mejoras

✅ **Usa** `{{ asset() }}` **para rutas de CSS, JS e imágenes**
✅ **Traducción completa a español**
✅ **Meta descripción mejorada**

📝 Código Mejorado

blade

```blade
<!DOCTYPE html>
<html lang="es">
```

```
<head>
    <meta charset="utf-8">
    <title>404 | AnderCode</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Página no encontrada | AnderCode">
    <meta name="author" content="AnderCode">
    <!-- Icono del sitio -->
    <link rel="shortcut icon" href="{{ asset('assets/images/favicon.ico') }}">
    <!-- Configuración del layout -->
    <script src="{{ asset('assets/js/layout.js') }}"></script>
    <!-- Estilos Bootstrap -->
    <link href="{{ asset('assets/css/bootstrap.min.css') }}" rel="stylesheet"
type="text/css">
    <!-- Iconos -->
    <link href="{{ asset('assets/css/icons.min.css') }}" rel="stylesheet"
type="text/css">
    <!-- Estilos principales -->
    <link href="{{ asset('assets/css/app.min.css') }}" rel="stylesheet"
type="text/css">
    <!-- Estilos personalizados -->
    <link href="{{ asset('assets/css/custom.min.css') }}" rel="stylesheet"
type="text/css">
</head>
<body>
    <div class="d-flex flex-column align-items-center justify-content-center"
style="height: 100vh;">
        <h1 class="display-1 fw-bold text-danger">404</h1>
        <h3 class="mb-4">¡Página no encontrada!</h3>
        <p class="text-muted">Lo sentimos, la página que buscas no existe o ha sido
movida.</p>
        <a href="{{ url('/') }}" class="btn btn-primary mt-3">Volver al Inicio</a>
    </div>
</body>
</html>
```

🔹 Explicación de Mejoras

1. **Uso de `{{ asset() }}`** ✅

- **Carga correcta de archivos estáticos en Laravel**.

2. **Meta Descripción Mejorada ✅**
    - **Descripción optimizada para SEO** en español.
3. **Texto en Español ✅**
    - Mensajes claros y bien traducidos.
4. **Estructura más Clara ✅**
    - Código **limpio y organizado**.

🚀 ¡Listo! Ahora tu página 404 está bien estructurada en Laravel con rutas dinámicas y en español! 🎯

Aquí tienes el código corregido con **rutas dinámicas en Laravel** y **traducción al español**:

📌 Correcciones

✅ `asset()` para cargar imágenes correctamente
✅ Traducción al español
✅ Uso de `{{ url('/') }}` en el botón para redirigir correctamente a la página de inicio

---

📝 Código Mejorado

blade

```
<div class="auth-page-content">
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="text-center pt-4">
                    <div>
                        <img src="{{ asset('assets/images/error.svg') }}" alt="Error
404" class="error-basic-img move-animation">
                    </div>
```

```
                    <div class="mt-n4">
                        <h1 class="display-1 fw-medium">404</h1>
                        <h3 class="text-uppercase">Lo sentimos, página no encontrada
😢</h3>
                        <p class="text-muted mb-4">La página que buscas no está
disponible.</p>
                        <a href="{{ url('/') }}" class="btn btn-success"><i
class="mdi mdi-home me-1"></i> Volver al inicio</a>
                    </div>
                </div>
            </div>
        </div>
        <!-- end row -->
    </div>
    <!-- end container -->
</div>
```

🔹 Explicación de Mejoras

1. **Rutas dinámicas de Laravel (`asset()` y `url()`) ✅**
   - Asegura que la imagen y los enlaces **funcionen correctamente en producción**.
2. **Traducción correcta al español ✅**
   - Mensajes más claros y en el tono adecuado.
3. **Código limpio y ordenado ✅**
   - Mantiene la estructura original con solo los cambios necesarios.

🚀 **¡Listo! Ahora tu página 404 es funcional y está en Laravel correctamente.** 🎯

Aquí tienes la sección **JAVASCRIPT** corregida con rutas dinámicas en **Laravel**:

📌 Correcciones

✅ **Uso de `{{ asset() }}` para cargar los archivos correctamente**
✅ **Mantiene la estructura original** sin cambios innecesarios

---

📝 Código Mejorado

blade

```
<!-- JAVASCRIPT -->
<script src="{{ asset('assets/libs/bootstrap/js/bootstrap.bundle.min.js') }}">
</script>
<script src="{{ asset('assets/libs/simplebar/simplebar.min.js') }}"></script>
<script src="{{ asset('assets/libs/node-waves/waves.min.js') }}"></script>
<script src="{{ asset('assets/libs/feather-icons/feather.min.js') }}"></script>
<script src="{{ asset('assets/js/pages/plugins/lord-icon-2.1.0.js') }}"></script>
<script src="{{ asset('assets/js/plugins.js') }}"></script>
<!-- particles js -->
<script src="{{ asset('assets/libs/particles.js/particles.js') }}"></script>
<!-- particles app js -->
<script src="{{ asset('assets/js/pages/particles.app.js') }}"></script>
```

🔹 Explicación de Mejoras

1. **Uso de `{{ asset() }}`** ✅
   - Laravel manejará las rutas correctamente **tanto en local como en producción.**
2. **Estructura Original Mantendida** ✅
   - Solo se cambiaron las rutas sin afectar la lógica del template.

🚀 **¡Listo! Ahora el template carga correctamente los scripts en Laravel.** 🎯

Para validar el acceso a cada vista según el **rol del usuario** en **Laravel 10**, puedes seguir estos pasos:

1️⃣ Agregar Middleware de Autorización

Laravel proporciona **middlewares** para restringir acceso a rutas según el **rol del usuario**.

📌 **Crea un middleware personalizado** para manejar los permisos:
Ejecuta el siguiente comando en la terminal:

sh

```sh
php artisan make:middleware RoleMiddleware
```

Esto generará un archivo en:
📂 `app/Http/Middleware/RoleMiddleware.php`

2️⃣ Configurar el Middleware

Edita `RoleMiddleware.php` y **modifica el** `handle()` **para validar el rol**:

php

```php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class RoleMiddleware
{
    /**
     * Maneja la solicitud entrante.
     */
    public function handle(Request $request, Closure $next, ...$roles)
    {
        // Verifica si el usuario está autenticado
        if (!Auth::check()) {
            return redirect('/login'); // Redirige si no está logueado
        }

        // Obtiene el rol del usuario autenticado
        $userRole = Auth::user()->role; // Asegúrate de que el campo 'role' existe
```

```
en la tabla 'users'

        // Verifica si el rol del usuario está permitido
        if (!in_array($userRole, $roles)) {
            return abort(403, 'No tienes permiso para acceder a esta página.');
        }

        return $next($request);
    }
}
```

3️⃣ Registrar el Middleware

Abre 📂 `app/Http/Kernel.php` y **agrega el middleware en** `$routeMiddleware`:

php

```php
protected $routeMiddleware = [
    'role' => \App\Http\Middleware\RoleMiddleware::class,
];
```

4️⃣ Aplicar Middleware en Rutas

Ahora puedes **proteger rutas específicas** en `routes/web.php` usando `middleware('role:admin')`.

📌 **Ejemplo:**

php

```php
use App\Http\Controllers\AdminController;
use App\Http\Controllers\MantenimientoController;

//  🔹 Rutas accesibles solo para ADMINISTRADOR
Route::middleware(['auth', 'role:admin'])->group(function () {
    Route::get('/admin/dashboard', [AdminController::class, 'index'])-
```

```
>name('admin.dashboard');
});

// 🔹 Rutas accesibles solo para el MANTENEDOR
Route::middleware(['auth', 'role:mantenedor'])->group(function () {
    Route::get('/mantenimientos', [MantenimientoController::class, 'index'])-
>name('mantenimientos.index');
});

// 🔹 Rutas accesibles para ADMINISTRADOR y MANTENEDOR
Route::middleware(['auth', 'role:admin,mantenedor'])->group(function () {
    Route::get('/gestion', [MantenimientoController::class, 'gestionar'])-
>name('gestion.index');
});
```

✅ **El Administrador (`admin`) tiene acceso a todo.**
✅ **El Mantenedor (`mantenedor`) solo puede acceder a los mantenimientos.**
✅ **Podemos agregar más roles fácilmente (`'role:admin,mantenedor'`).**

5️⃣ Restringir Acceso en Vistas Blade

Para ocultar **botones o secciones en las vistas**, usa:

blade

```
@can('admin')
    <a href="{{ route('admin.dashboard') }}" class="btn btn-primary">Panel Admin</a>
@endcan
```

O usando `Auth::user()->role`:

blade

```
@if(Auth::user()->role === 'admin')
    <a href="{{ route('admin.dashboard') }}" class="btn btn-primary">Panel Admin</a>
@endif
```

6️⃣ Probar los Accesos

- **Accede con un usuario ADMIN:** Puede ver todas las rutas.
- **Accede con un usuario MANTENEDOR:** Solo ve las rutas de mantenimiento.
- **Si un usuario sin permiso intenta acceder: Error 403** (sin acceso).

✅ Resumen

1. **Crear un middleware (`RoleMiddleware`) para validar roles.**
2. **Registrar el middleware en `Kernel.php`.**
3. **Usar el middleware en las rutas (`middleware('role:admin')`).**
4. **Proteger elementos en las vistas con `@can()` o `Auth::user()->role`.**

🚀 **¡Listo! Ahora el acceso a las vistas está restringido por roles y podemos seguir agregando más fácilmente.** 🎯

En **Laravel 10**, la propiedad `$routeMiddleware` **ha sido eliminada**, y ahora todos los middlewares personalizados deben registrarse en la propiedad `$middlewareAliases`.

📌 Solución

En lugar de agregar el middleware en `$routeMiddleware`, **debes registrarlo en** `$middlewareAliases` dentro de `app/Http/Kernel.php`.

🔹 **Abre el archivo 📂 `app/Http/Kernel.php` y agrega esto dentro de `$middlewareAliases`:**

```php
protected $middlewareAliases = [
    'role' => \App\Http\Middleware\RoleMiddleware::class,
];
```

📌 ¿Cómo Usarlo en las Rutas?

Después de registrar el middleware, úsalo en `routes/web.php` de la siguiente manera:

```php
use App\Http\Controllers\AdminController;
use App\Http\Controllers\MantenimientoController;

// 🔹 Rutas accesibles solo para ADMINISTRADOR
Route::middleware(['auth', 'role:admin'])->group(function () {
    Route::get('/admin/dashboard', [AdminController::class, 'index'])->name('admin.dashboard');
});

// 🔹 Rutas accesibles solo para el MANTENEDOR
Route::middleware(['auth', 'role:mantenedor'])->group(function () {
    Route::get('/mantenimientos', [MantenimientoController::class, 'index'])->name('mantenimientos.index');
});

// 🔹 Rutas accesibles para ADMINISTRADOR y MANTENEDOR
Route::middleware(['auth', 'role:admin,mantenedor'])->group(function () {
    Route::get('/gestion', [MantenimientoController::class, 'gestionar'])->name('gestion.index');
});
```

✅ **Con esta corrección, el middleware funcionará perfectamente en Laravel 10.** 🚀

Para adaptar el código a tu estructura, donde los usuarios tienen un campo `role_id` y este se relaciona con una tabla `roles`, debemos hacer algunos cambios en el middleware y en las rutas.

📌 1️⃣ Actualizar el Middleware `RoleMiddleware`

📂 **Ubicación:** `app/Http/Middleware/RoleMiddleware.php`

php

```php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class RoleMiddleware
{
    /**
     * Maneja la solicitud entrante.
     */
    public function handle(Request $request, Closure $next, ...$roles)
    {
        // Verifica si el usuario está autenticado
        if (!Auth::check()) {
            return redirect('/login'); // Redirige si no está logueado
        }

        // Obtiene el nombre del rol del usuario autenticado (suponiendo que hay una relación con roles)
        $userRole = Auth::user()->role->name ?? null; // Asegúrate de tener la relación con roles en el modelo User

        // Verifica si el rol del usuario está permitido
        if (!in_array($userRole, $roles)) {
            return abort(403, 'No tienes permiso para acceder a esta página.');
        }

        return $next($request);
    }
}
```

📌 2️⃣ Asegurar la Relación en el Modelo `User.php`

📂 **Ubicación:** `app/Models/User.php`

Asegúrate de que el modelo `User` tenga la relación con la tabla `roles`:

```php
namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name', 'email', 'password', 'role_id'
    ];

    /**
     * Relación con la tabla roles.
     */
    public function role()
    {
        return $this->belongsTo(Role::class, 'role_id');
    }
}
```

📌 3️⃣ Registrar el Middleware en `Kernel.php`

📂 **Ubicación:** `app/Http/Kernel.php`

Agrega esto en `$middlewareAliases`:

```php
protected $middlewareAliases = [
    'role' => \App\Http\Middleware\RoleMiddleware::class,
];
```

📌 4️⃣ Aplicar Middleware en las Rutas

📂 **Ubicación:** `routes/web.php`

Ahora podemos **usar el middleware en las rutas** basándonos en el `role->name`:

```php
use App\Http\Controllers\AdminController;
use App\Http\Controllers\MantenimientoController;

// 🔹 Rutas accesibles solo para ADMINISTRADOR
Route::middleware(['auth', 'role:Administrador'])->group(function () {
    Route::get('/admin/dashboard', [AdminController::class, 'index'])->name('admin.dashboard');
});

// 🔹 Rutas accesibles solo para el MANTENEDOR
Route::middleware(['auth', 'role:Mantenedor'])->group(function () {
    Route::get('/mantenimientos', [MantenimientoController::class, 'index'])->name('mantenimientos.index');
});

// 🔹 Rutas accesibles para ADMINISTRADOR y MANTENEDOR
Route::middleware(['auth', 'role:Administrador,Mantenedor'])->group(function () {
    Route::get('/gestion', [MantenimientoController::class, 'gestionar'])->name('gestion.index');
});
```

✅ **Con estos cambios ahora Laravel validará el acceso basándose en la relación** `role_id -> roles.name`. 🚀

Para permitir el acceso a las rutas de recursos (`Route::resource`) según el rol del usuario, debemos aplicar el **middleware de roles** correctamente en **web.php**.

📌 1️⃣ Solución: Aplicar Middleware a las Rutas de Recursos

Si los **mantenedores** deben acceder solo a los mantenimientos, entonces aplica el middleware `role:Mantenedor,Administrador` a esas rutas.

📂 **Ubicación:** `routes/web.php`

php

```php
use App\Http\Controllers\CategoryController;
use App\Http\Controllers\CustomerController;
use App\Http\Controllers\SupplierController;
use App\Http\Controllers\UserController;
use App\Http\Controllers\ProductController;

// 🔹 Agrupar rutas con middleware para MANTENEDOR y ADMINISTRADOR
Route::middleware(['auth', 'role:Administrador,Mantenedor'])->group(function () {
    Route::resource('categorias', CategoryController::class);
    Route::resource('customers', CustomerController::class);
    Route::resource('suppliers', SupplierController::class);
    Route::resource('products', ProductController::class);

    // Ruta para eliminar imágenes de productos
    Route::delete('product-images/{id}', [ProductController::class,
'destroyImage'])->name('product_images.destroy');
});

// 🔹 Solo el ADMINISTRADOR puede gestionar usuarios
Route::middleware(['auth', 'role:Administrador'])->group(function () {
    Route::resource('users', UserController::class);
});
```

✅ **Con esta configuración:**

- **Los MANTENEDORES y ADMINISTRADORES** pueden acceder a los mantenimientos (`categorias, customers, suppliers, products`).
- **Solo el ADMINISTRADOR** puede gestionar `users`.

📌 2️⃣ Verifica que el Usuario tenga su `role_id` Asignado

Si Laravel sigue bloqueando el acceso, verifica en la base de datos que el usuario tenga un `role_id` **válido** en la tabla `users` y que su **relación con la tabla** `roles` **esté correcta**.

Ejecuta en **Tinker** (`php artisan tinker`):

```php
$user = \App\Models\User::find(1); // Reemplaza 1 con el ID del usuario que pruebas
$user->role; // Verifica si devuelve un objeto con el nombre del rol
```

Si `null`, entonces asegúrate de que el usuario tenga un `role_id` válido.

📌 3️⃣ Opcional: Verifica Middleware en `RoleMiddleware.php`

📂 **Ubicación:** `app/Http/Middleware/RoleMiddleware.php`

Confirma que el middleware está **verificando correctamente el nombre del rol**:

```php
public function handle(Request $request, Closure $next, ...$roles)
{
    if (!Auth::check()) {
        return redirect('/login'); // Redirige si no está logueado
    }

    // Obtiene el nombre del rol del usuario autenticado
    $userRole = Auth::user()->role->name ?? null;

    // Verifica si el rol del usuario está en los permitidos
    if (!in_array($userRole, $roles)) {
        return abort(403, 'No tienes permiso para acceder a esta página.');
    }

    return $next($request);
}
```

📌 4️⃣ Prueba con una Sesión Activa

Si el problema persiste, intenta cerrar sesión y volver a iniciar sesión con un usuario con **rol de Administrador o Mantenedor**.

🚀 **¡Listo! Con estos ajustes, los accesos estarán bien restringidos por roles y solo quienes tengan permiso podrán acceder!** 🎯

🚀 Solución: Middleware en `Route::resource` no Funciona

El problema ocurre porque **el middleware `role` se ejecuta antes de cargar los controladores**, pero las rutas de `Route::resource` **ya están registradas** en Laravel. Esto provoca que **se bloqueen todos los accesos**.

✅ Solución: Aplicar Middleware en los Controladores

En lugar de aplicar el middleware en `web.php`, lo aplicamos dentro de los **controladores**, asegurándonos de que cada controlador tenga **el acceso correcto** según el rol.

📌 1️⃣ Agregar Middleware en los Controladores

📂 **Ubicación:** `app/Http/Controllers/CategoryController.php` (Repetir para otros controladores)

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CategoryController extends Controller
{
```

```
    public function __construct()
    {
        // 🔹 Solo el ADMINISTRADOR y MANTENEDOR pueden acceder a este controlador
        $this->middleware(['auth', 'role:Administrador,Mantenedor']);
    }

    public function index()
    {
        return view('categories.index');
    }

    // Otros métodos CRUD...
}
```

🔹 **Repite esto para:**

✅ `CustomerController.php`
✅ `SupplierController.php`
✅ `UserController.php` (solo `'role:Administrador'`)
✅ `ProductController.php`
✅ `CompraController.php`, `VentaController.php`, etc.

📌 2️⃣ Rutas `web.php` sin Middleware

Después de mover el middleware a los controladores, **elimina los `role` en `web.php`**, manteniendo solo `auth`.

php

```
Route::middleware(['auth'])->group(function () {
    Route::resource('categorias', CategoryController::class);
    Route::resource('customers', CustomerController::class);
    Route::resource('suppliers', SupplierController::class);
    Route::resource('users', UserController::class); // 🔹 Usuarios solo para ADMIN
    Route::resource('products', ProductController::class);
    Route::delete('product-images/{id}', [ProductController::class,
'destroyImage'])->name('product_images.destroy');

    Route::resource('compras', CompraController::class);
    Route::get('compras/pdf/{id}', [CompraController::class, 'downloadPDF'])-
```

```
>name('compras.pdf');
    Route::get('compras/detalle/{id}', [CompraController::class, 'detalle'])-
>name('compras.detalle');

    Route::resource('ventas', VentaController::class);
    Route::get('ventas/pdf/{id}', [VentaController::class, 'downloadPDF'])-
>name('ventas.pdf');
    Route::get('ventas/detalle/{id}', [VentaController::class, 'detalle'])-
>name('ventas.detalle');

    Route::get('inventories', [InventoryController::class, 'index'])-
>name('inventories.index');
    Route::get('inventories/detalle/{id}', [InventoryController::class, 'detalle'])-
>name('inventories.detalle');
    Route::get('inventories/export', [InventoryController::class, 'export'])-
>name('inventories.export');

    Route::get('transactions', [TransactionController::class, 'index'])-
>name('transactions.index');
    Route::get('transactions/detalle/{id}', [TransactionController::class,
'detalle'])->name('transactions.detalle');
    Route::get('transactions/export', [TransactionController::class, 'export'])-
>name('transactions.export');
});
```

📌 3️⃣ Verificar Accesos

Ahora los **mantenedores y administradores** pueden acceder según el **middleware en cada controlador**.

✅ **Ventajas de este método:**

1. **Mejor control:** Cada controlador maneja sus propios accesos.
2. **Evita problemas con** `Route::resource` en `web.php`.
3. **Fácil de extender** a más roles en el futuro.

🚀 **¡Listo! Ahora los accesos funcionan correctamente y puedes seguir agregando más roles según sea necesario!** 🎯

✅ Solución: Ocultar Menús Según el Rol del Usuario

Para ocultar los **menús en la vista** según el rol del usuario, puedes hacerlo en **Blade** (`.blade.php`) con la ayuda de la autenticación (`Auth::user()`).

📌 1️⃣ Verificar el Rol del Usuario en la Vista

📂 **Ubicación:** `resources/views/layouts/sidebar.blade.php` (o el archivo donde está el menú)

```html
<ul class="sidebar-menu">
    <!-- 🔹 Menú visible solo para ADMINISTRADORES -->
    @if (Auth::check() && Auth::user()->role->name === 'Administrador')
        <li><a href="{{ route('users.index') }}"><i class="bx bx-user"></i>
Usuarios</a></li>
    @endif

    <!-- 🔹 Menú visible para ADMIN y MANTENEDOR -->
    @if (Auth::check() && in_array(Auth::user()->role->name, ['Administrador',
'Mantenedor']))
        <li><a href="{{ route('categorias.index') }}"><i class="bx bx-list-ul"></i>
Categorías</a></li>
        <li><a href="{{ route('customers.index') }}"><i class="bx bx-group"></i>
Clientes</a></li>
        <li><a href="{{ route('suppliers.index') }}"><i class="bx bx-truck"></i>
Proveedores</a></li>
        <li><a href="{{ route('products.index') }}"><i class="bx bx-box"></i>
Productos</a></li>
    @endif

    <!-- 🔹 Menú visible para todos los usuarios autenticados -->
    @if (Auth::check())
        <li><a href="{{ route('transactions.index') }}"><i class="bx bx-transfer">
</i> Transacciones</a></li>
    @endif
</ul>
```

📌 2️⃣ Optimizar con un Helper (`isRole()`)

Si quieres hacer el código más limpio, puedes crear un **helper** en Laravel.

📂 **Ubicación:** `app/Helpers/RoleHelper.php`

```php
<?php

use Illuminate\Support\Facades\Auth;

if (!function_exists('isRole')) {
    function isRole($role)
    {
        return Auth::check() && Auth::user()->role->name === $role;
    }
}
```

✅ Luego, **cárgalo automáticamente** en `composer.json`:

📂 **Ubicación:** `composer.json`

```json
"autoload": {
    "files": [
        "app/Helpers/RoleHelper.php"
    ]
}
```

Ejecuta este comando para actualizar:

```sh
composer dump-autoload
```

📌 3️⃣ Usarlo en Blade

Ahora, en tu **Blade**, puedes usar `isRole('Administrador')`:

```html
```

```
<ul class="sidebar-menu">
    @if (isRole('Administrador'))
        <li><a href="{{ route('users.index') }}"><i class="bx bx-user"></i>
Usuarios</a></li>
    @endif

    @if (isRole('Administrador') || isRole('Mantenedor'))
        <li><a href="{{ route('categorias.index') }}"><i class="bx bx-list-ul"></i>
Categorías</a></li>
        <li><a href="{{ route('customers.index') }}"><i class="bx bx-group"></i>
Clientes</a></li>
        <li><a href="{{ route('suppliers.index') }}"><i class="bx bx-truck"></i>
Proveedores</a></li>
        <li><a href="{{ route('products.index') }}"><i class="bx bx-box"></i>
Productos</a></li>
    @endif

    @if (Auth::check())
        <li><a href="{{ route('transactions.index') }}"><i class="bx bx-transfer">
</i> Transacciones</a></li>
    @endif
</ul>
```

🚀 Ahora el menú solo mostrará las opciones según el rol del usuario.

🎯 Ventajas de este método:

- **Código más limpio** con el helper `isRole()`.
- **Mayor seguridad**, ya que las opciones no aparecen para roles no autorizados.
- **Fácil mantenimiento** si agregas más roles en el futuro.

🔹 **¡Listo! Ahora el menú se ocultará automáticamente según el rol del usuario.** 🚀

¡Genial! Envíame el código HTML del template y lo iré modificando y mejorando con la información del curso. Solo genera código basado en lo que me proporciones. ¡Estoy listo! 🚀

Aquí está la versión modificada de tu código con la información del curso "Desarrolla un Sistema de Compra y Venta con Laravel + ChatGPT". He cambiado el **título**, **descripción** y **favicon** para que reflejen la identidad del curso.

```html
<meta charset="utf-8" />
<title>Desarrolla un Sistema de Compra y Venta con Laravel + ChatGPT</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Aprende a desarrollar un sistema de compra y venta profesional con Laravel 10, MySQL y ChatGPT como asistente inteligente. Domina autenticación, CRUDs, gestión de inventarios, compras, ventas y reportes dinámicos." />
<meta name="author" content="AnderCode" />

<!-- Favicon personalizado -->
<link rel="shortcut icon" href="assets/images/favicon-course.ico">

<!-- Swiper CSS para carruseles -->
<link href="assets/libs/swiper/swiper-bundle.min.css" rel="stylesheet" type="text/css" />

<!-- Configuración de Layout -->
<script src="assets/js/layout.js"></script>

<!-- Bootstrap y Estilos Principales -->
<link href="assets/css/bootstrap.min.css" rel="stylesheet" type="text/css" />
<link href="assets/css/icons.min.css" rel="stylesheet" type="text/css" />
<link href="assets/css/app.min.css" rel="stylesheet" type="text/css" />
<link href="assets/css/custom.min.css" rel="stylesheet" type="text/css" />
```

✅ Mejoras Realizadas:

1. **Título ajustado** para reflejar el nombre del curso.
2. **Descripción optimizada** para mejorar SEO y atraer a más estudiantes.
3. **Autor cambiado a "AnderCode"** para reflejar tu marca.
4. **Favicon cambiado a** `favicon-course.ico` (puedes reemplazarlo por tu logo personalizado).

Si tienes un **logo específico**, dímelo y ajustaré el favicon. Ahora, **envíame más código del template** y lo iré personalizando. 🚀

Aquí está el código actualizado con las **rutas corregidas** para Laravel 10, asegurándome de que los archivos estáticos **se carguen correctamente** desde la carpeta `public/`:

```html
<meta charset="utf-8" />
<title>Desarrolla un Sistema de Compra y Venta con Laravel + ChatGPT</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Aprende a desarrollar un sistema de compra y venta profesional con Laravel 10, MySQL y ChatGPT como asistente inteligente. Domina autenticación, CRUDs, gestión de inventarios, compras, ventas y reportes dinámicos." />
<meta name="author" content="AnderCode" />

<!-- Favicon personalizado -->
<link rel="shortcut icon" href="{{ asset('assets/images/favicon-course.ico') }}">

<!-- Swiper CSS para carruseles -->
<link href="{{ asset('assets/libs/swiper/swiper-bundle.min.css') }}" rel="stylesheet" type="text/css" />

<!-- Configuración de Layout -->
<script src="{{ asset('assets/js/layout.js') }}"></script>

<!-- Bootstrap y Estilos Principales -->
<link href="{{ asset('assets/css/bootstrap.min.css') }}" rel="stylesheet" type="text/css" />
<link href="{{ asset('assets/css/icons.min.css') }}" rel="stylesheet" type="text/css" />
<link href="{{ asset('assets/css/app.min.css') }}" rel="stylesheet" type="text/css" />
<link href="{{ asset('assets/css/custom.min.css') }}" rel="stylesheet" type="text/css" />
```

✅ Mejoras Realizadas:

1. **Rutas corregidas para Laravel 10:**
   - Se reemplazó `"assets/..."` por `{{ asset('assets/...') }}` para que Laravel cargue correctamente los archivos desde `public/`.
2. **Favicon personalizado:**
   - Asegúrate de que `public/assets/images/favicon-course.ico` existe o reemplázalo por otro favicon que tengas.
3. **Carga optimizada de archivos estáticos:**
   - Ahora los estilos y scripts se cargarán correctamente en Laravel.

🔥 **¡Listo! Ahora el código es completamente funcional en Laravel 10.**
📌 **Envíame más partes del template HTML y lo iré adaptando al curso.** 🚀

Aquí tienes la versión **mejorada y adaptada** de la **sección hero** para el curso "**Desarrolla un Sistema de Compra y Venta con Laravel + ChatGPT**". He cambiado los textos, las rutas y los enlaces para que reflejen el contenido del curso. 🚀

---

html

```
<section class="section pb-0 hero-section" id="hero">
    <div class="bg-overlay bg-overlay-pattern"></div>
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-lg-8 col-sm-10">
                <div class="text-center mt-lg-5 pt-5">
                    <h1 class="display-6 fw-semibold mb-3 lh-base">
                        Desarrolla un <span class="text-success">Sistema de Compra y
Venta</span> con Laravel 10 + ChatGPT
                    </h1>
                    <p class="lead text-muted lh-base">
                        Aprende a construir un sistema completo desde cero,
integrando autenticación, gestión de productos, clientes, compras y ventas con
reportes en PDF y Excel.
```

```
                        Optimiza tu código con ChatGPT y mejora tu productividad como desarrollador.
                    </p>

                    <div class="d-flex gap-2 justify-content-center mt-4">
                        <a href="{{ route('register') }}" class="btn btn-primary">
                            Inscribirse Ahora <i class="ri-arrow-right-line align-middle ms-1"></i>
                        </a>
                        <a href="{{ route('pricing') }}" class="btn btn-danger">
                            Ver Planes <i class="ri-eye-line align-middle ms-1"></i>
                        </a>
                    </div>
                </div>

                <div class="mt-4 mt-sm-5 pt-sm-5 mb-sm-n5 demo-carousel">
                    <div class="demo-img-patten-top d-none d-sm-block">
                        <img src="{{ asset('assets/images/landing/img-pattern.png') }}" class="d-block img-fluid" alt="Patrón superior">
                    </div>
                    <div class="demo-img-patten-bottom d-none d-sm-block">
                        <img src="{{ asset('assets/images/landing/img-pattern.png') }}" class="d-block img-fluid" alt="Patrón inferior">
                    </div>
                    <div class="carousel slide carousel-fade" data-bs-ride="carousel">
                        <div class="carousel-inner shadow-lg p-2 bg-white rounded">
                            <div class="carousel-item active" data-bs-interval="2000">
                                <img src="{{ asset('assets/images/demos/dashboard.png') }}" class="d-block w-100" alt="Dashboard del sistema">
                            </div>
                            <div class="carousel-item" data-bs-interval="2000">
                                <img src="{{ asset('assets/images/demos/auth.png') }}" class="d-block w-100" alt="Autenticación Laravel Breeze">
                            </div>
                            <div class="carousel-item" data-bs-interval="2000">
                                <img src="{{ asset('assets/images/demos/inventario.png') }}" class="d-block w-100" alt="Gestión de Inventarios">
                            </div>
```

```
                            <div class="carousel-item" data-bs-interval="2000">
                                <img src="{{
asset('assets/images/demos/reportes.png') }}" class="d-block w-100" alt="Reportes
PDF y Excel">
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <!-- end row -->
    </div>
    <!-- end container -->
    <div class="position-absolute start-0 end-0 bottom-0 hero-shape-svg">
        <svg xmlns="http://www.w3.org/2000/svg" version="1.1"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 1440 120">
            <g mask="url(&quot;#SvgjsMask1003&quot;)" fill="none">
                <path d="M 0,118 C 288,98.6 1152,40.4 1440,21L1440 140L0 140z">
</path>
            </g>
        </svg>
    </div>
    <!-- end shape -->
</section>
```

✅ Mejoras Realizadas:

1. **Título y descripción adaptados** al curso de Laravel 10 y ChatGPT.
2. **Botones actualizados**:
   - **"Inscribirse Ahora"** apunta a `{{ route('register') }}`.
   - **"Ver Planes"** apunta a `{{ route('pricing') }}` (puedes cambiarlo si usas otra ruta).
3. **Carousel mejorado** con capturas relevantes del curso:
   - Dashboard
   - Autenticación con Laravel Breeze

- Gestión de inventarios
- Reportes en PDF y Excel

4. **Corrección de rutas en Laravel 10** para los assets:
   - `{{ asset('assets/...') }}` en imágenes y CSS.

🔥 **¡Listo! La sección principal del landing está personalizada y lista para Laravel 10!**
📌 **Envíame la siguiente parte del template y lo seguiré mejorando.** 🚀

Aquí tienes la sección **"Servicios"** adaptada para tu curso **"Desarrolla un Sistema de Compra y Venta con Laravel + ChatGPT"**. He cambiado los textos para reflejar las características del curso, **manteniendo el nombre de las imágenes e iconos**. 🚀

---

html

```
<section class="section" id="services">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-lg-8">
                <div class="text-center mb-5">
                    <h1 class="mb-3 ff-secondary fw-semibold lh-base">
                        Todo lo que necesitas para desarrollar un sistema de compra
y venta profesional
                    </h1>
                    <p class="text-muted">
                        Aprende a construir un sistema completo con Laravel 10 y
MySQL, optimizando tu flujo de trabajo con ChatGPT. Desde la gestión de productos
hasta reportes avanzados, este curso te guiará en cada paso.
                    </p>
                </div>
            </div>
            <!-- end col -->
        </div>
        <!-- end row -->
```

```
        <div class="row g-3">
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-pencil-ruler-2-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Diseño Profesional</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Implementa una interfaz atractiva y funcional con un
diseño responsive basado en Bootstrap.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-stack-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Gestión Completa</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Crea y administra productos, categorías, clientes y
proveedores con un CRUD dinámico y eficiente.
                        </p>
                        <div>
```

```
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-lightbulb-flash-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Estrategia y Flujo de Trabajo</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Aprende a estructurar y organizar tu código siguiendo
buenas prácticas de desarrollo en Laravel.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-customer-service-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
```

```
                        <h5 class="fs-18">Soporte y Comunidad</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Únete a nuestra comunidad para resolver dudas y
compartir experiencias con otros desarrolladores.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-settings-2-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Configuración Sencilla</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Aprende a instalar y configurar Laravel 10, MySQL y
herramientas adicionales de manera rápida.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
```

```
rounded-circle">
                                <i class="ri-google-fill fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Optimización con ChatGPT</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Descubre cómo usar ChatGPT para mejorar tu productividad
y agilizar el desarrollo.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-slideshow-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Dashboard Interactivo</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Implementa gráficos y reportes interactivos con
ApexCharts para visualizar estadísticas clave.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
```

```
            <!-- end col -->
            <div class="col-lg-4">
                <div class="d-flex p-3">
                    <div class="flex-shrink-0 me-3">
                        <div class="avatar-sm icon-effect">
                            <div class="avatar-title bg-transparent text-success
rounded-circle">
                                <i class="ri-briefcase-5-line fs-36"></i>
                            </div>
                        </div>
                    </div>
                    <div class="flex-grow-1">
                        <h5 class="fs-18">Reportes en PDF y Excel</h5>
                        <p class="text-muted my-3 ff-secondary">
                            Genera reportes dinámicos en distintos formatos para
analizar la información de tu sistema.
                        </p>
                        <div>
                            <a href="#" class="fs-13 fw-medium">Ver más <i
class="ri-arrow-right-s-line align-bottom"></i></a>
                        </div>
                    </div>
                </div>
            </div>
            <!-- end col -->
        </div>
        <!-- end row -->
    </div>
    <!-- end container -->
</section>
```

✅ **Cambios y mejoras realizadas:**

- **Traducción y adaptación** de los servicios al curso de Laravel 10 + ChatGPT.
- **Mantenimiento de los nombres de las imágenes** y iconos.
- **Optimización del contenido** para reflejar lo que realmente aprenderán los estudiantes.
- **Estructura organizada** para que cada servicio se enfoque en un punto clave del curso.

📌 **Envíame la siguiente sección y la seguimos mejorando!** 🚀

Aquí tienes el código corregido para **Laravel 10**, asegurándome de que las rutas sean correctas y respetando la estructura del framework:

```html
<!-- JAVASCRIPT -->
<script src="{{ asset('assets/libs/bootstrap/js/bootstrap.bundle.min.js') }}">
</script>
<script src="{{ asset('assets/libs/simplebar/simplebar.min.js') }}"></script>
<script src="{{ asset('assets/libs/node-waves/waves.min.js') }}"></script>
<script src="{{ asset('assets/libs/feather-icons/feather.min.js') }}"></script>
<script src="{{ asset('assets/js/pages/plugins/lord-icon-2.1.0.js') }}"></script>
<script src="{{ asset('assets/js/plugins.js') }}"></script>

<!-- Swiper slider js -->
<script src="{{ asset('assets/libs/swiper/swiper-bundle.min.js') }}"></script>

<!-- landing init -->
<script src="{{ asset('assets/js/pages/landing.init.js') }}"></script>
```

✅ Correcciones realizadas:

1. **Uso de `{{ asset() }}` en Laravel 10**
   - Laravel usa `asset()` para generar correctamente las rutas de archivos estáticos, asegurando que funcionen incluso si el proyecto está en un subdirectorio.
2. **Compatibilidad con Laravel Mix o Vite (opcional)**
   - Si estás usando **Vite**, tendrías que migrar a `@vite('resources/js/app.js')` en lugar de `asset()`.

🔹 **Listo para usar en Laravel 10.** 🚀 Ahora, si tienes más secciones, envíamelas y las adaptamos. 😊

Aquí tienes el código actualizado para **Laravel 10**, agregando la validación de sesión para mostrar el botón de inicio de sesión o redirigir al **dashboard** si el usuario ya ha iniciado sesión:

```html
```

```
<div class="">
    @auth
        <!-- Si el usuario ya ha iniciado sesión, redirigir al dashboard -->
        <a href="{{ route('dashboard') }}" class="btn btn-primary">Ir al
Dashboard</a>
    @else
        <!-- Si el usuario no ha iniciado sesión, mostrar opciones -->
        <a href="{{ route('login') }}" class="btn btn-link fw-medium text-
decoration-none text-dark">Iniciar Sesión</a>
        <a href="{{ route('register') }}" class="btn btn-primary">Registrarse</a>
    @endauth
</div>
```

✅ Cambios realizados:

1. **Uso de `@auth` y `@else` en Blade**
   - Si el usuario está autenticado (`@auth`), se muestra el botón para ir al **Dashboard**.
   - Si no está autenticado (`@else`), se muestran las opciones para **Iniciar Sesión** o **Registrarse**.
2. **Corrección de rutas en Laravel 10**
   - `route('login')` → Redirige a la vista de inicio de sesión.
   - `route('register')` → Redirige a la vista de registro.
   - `route('dashboard')` → Redirige al Dashboard después de iniciar sesión.

🔹 **Ahora el sistema manejará correctamente la sesión del usuario.** 🚀
Si necesitas personalizar la redirección del dashboard, dime qué ruta específica deseas usar. 😊