1. Rutas: routes/web.php

Laravel utiliza el archivo routes/web.php para definir las rutas que apuntan a los diferentes controladores y métodos que controlan la lógica de las páginas.

```php
<?php
use App\Http\Controllers\ProductController;
use Illuminate\Support\Facades\Route;

// Ruta principal que muestra los tres catálogos: Ofertas, Selección, y Top Ventas.
Route::get('/', [ProductController::class, 'index']);

// Ruta que muestra los productos de ofertas
Route::get('/ofertas', [ProductController::class, 'ofertas']);

// Ruta que muestra los productos de selección
Route::get('/seleccion', [ProductController::class, 'seleccion']);

// Ruta que muestra los productos top ventas
Route::get('/top-ventas', [ProductController::class, 'topVentas']);

// Ruta que muestra un producto específico por su ID
Route::get('/producto/{productoid}', [ProductController::class, 'verProducto']);
```

Explicación:

- **Route::get('/', [ProductController::class, 'index']);**: Esta es la ruta que se llama cuando accedes al home (/) de la aplicación. Muestra todos los catálogos juntos (Ofertas, Selección, Top Ventas).
- **Route::get('/ofertas', [ProductController::class, 'ofertas']);**: Cuando accedes a /ofertas, muestra solo los productos que están en el catálogo de "Ofertas".
- **Route::get('/seleccion', [ProductController::class, 'seleccion']);**: Muestra los productos en la categoría de "Selección".
- **Route::get('/top-ventas', [ProductController::class, 'topVentas']);**: Muestra los productos del catálogo "Top Ventas".
- **Route::get('/producto/{productoid}', [ProductController::class, 'verProducto']);**: Esta ruta recibe el ID de un producto en la URL y muestra el detalle del producto. El {productoid} es una variable dinámica que será procesada por el controlador.

2. Controlador: ProductController.php

El controlador maneja la lógica de cómo se organizan y muestran los productos para cada catálogo y el detalle de cada producto.

Código Completo del Controlador:

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

use App\Models\Producto;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    private $ofertas;
    private $seleccion;
    private $topventas;

    // Constructor: se llama automáticamente al instanciar el controlador.
    public function __construct()
    {
        // Inicializa las variables con los datos de los catálogos
        $this->init_variables();
    }

    // Método para inicializar las variables de los catálogos
    public function init_variables()
    {
        // Ofertas guardadas en un array asociativo
        $this->ofertas = [
            ['id' => 1, 'nombre' => 'Producto A', 'descripcion' => 'Descripción de Producto A', 'precio' => 100, 'imagen' => 'producto1.jpeg'],
            ['id' => 2, 'nombre' => 'Producto B', 'descripcion' => 'Descripción de Producto B', 'precio' => 150, 'imagen' => 'producto2.jpeg']
        ];

        // Selección guardada como un string JSON
        $jsonSeleccion = '[
            {"id": 3, "nombre": "Producto C", "descripcion": "Descripción de Producto C", "precio": 200, "imagen": "producto3.jpeg"},
            {"id": 4, "nombre": "Producto D", "descripcion": "Descripción de Producto D", "precio": 250, "imagen": "producto4.jpeg"}
        ]';
        $this->seleccion = json_decode($jsonSeleccion, true); // Decodificamos el JSON

        // Top Ventas guardado como un array de objetos Producto
        $this->topventas = [
            new Producto(5, 'Producto E', 'Descripción de Producto E', 300, 'producto5.jpeg'),
            new Producto(6, 'Producto F', 'Descripción de Producto F', 350, 'producto6.jpeg')
        ];
    }

    // Muestra los tres catálogos juntos (Ofertas, Selección, Top Ventas)
    public function index()
    {
        return view('all', [
            'ofertas' => $this->ofertas,
            'seleccion' => $this->seleccion,
            'topventas' => $this->topventas,
        ]);
    }

    // Muestra solo los productos de Ofertas
    public function ofertas()
    {
        return view('ofertas', ['productos' => $this->ofertas]);
    }

    // Muestra solo los productos de Selección
    public function seleccion()
    {
        return view('seleccion', ['productos' => $this->seleccion]);
    }

    // Muestra solo los productos de Top Ventas
    public function topVentas()
    {
        return view('topventas', ['productos' => $this->topventas]);
    }

    // Muestra un producto específico por su ID
    public function verProducto($productoid)
    {
        // Buscamos el producto en todos los catálogos
        $producto = collect($this->ofertas)
            ->merge($this->seleccion)
            ->merge($this->topventas)
            ->firstWhere('id', $productoid);

        if (!$producto) {
            abort(404); // Si no se encuentra el producto, se lanza un error 404
        }

        return view('producto', ['producto' => (object)$producto]);
    }
}
```

Explicación Detallada:

- **Constructor (__construct)**: Al inicializar la clase ProductController, el constructor llama al método init_variables() para inicializar las variables de los catálogos de productos (Ofertas, Selección y Top Ventas).

- **init_variables()**: Este método define los datos de los tres catálogos:

    - Ofertas se almacena como un array asociativo.
    - Selección se guarda en formato JSON que luego se decodifica con json_decode().
    - Top Ventas se almacena como un array de objetos de la clase Producto.
- **index()**: Este método se usa para la página principal (/) y devuelve la vista all.blade.php, pasando los datos de los tres catálogos (Ofertas, Selección, Top Ventas).

- ofertas(), seleccion(), topVentas(): Cada uno de estos métodos maneja una página que muestra solo un catálogo específico, pasando los productos correspondientes a la vista.

- verProducto($productoid): Busca un producto por su ID a través de los tres catálogos combinados usando collect() y firstWhere(). Si el producto no se encuentra, se lanza un error 404.

3. Vistas

Las vistas definen cómo se presentan los datos en la aplicación. Laravel usa el motor de plantillas Blade para facilitar la inserción de lógica en el HTML.

Vista Principal (resources/views/all.blade.php)

```
@extends('layouts.app')

@section('content')
    <h1>Todos los Catálogos</h1>

    <h2>Ofertas</h2>
    @include('partials.catalogo', ['productos' => $ofertas])

    <h2>Selección</h2>
    @include('partials.catalogo', ['productos' => $seleccion])

    <h2>Top Ventas</h2>
    @include('partials.catalogo', ['productos' => $topventas])
@endsection
```

- **@extends('layouts.app')**: Extiende el layout base que contiene la estructura general de la página (cabecera, pie de página, etc.).
- **@section('content')**: Define el contenido que irá dentro de la sección content del layout.
- **@include('partials.catalogo')**: Incluye la vista parcial catalogo.blade.php, que es responsable de mostrar los productos en un formato de tarjeta.

Vista de Producto (resources/views/producto.blade.php)

```
@extends('layouts.app')

@section('content')
    <div class="producto-detalle">
        <h1>{{ is_object($producto) ? $producto->nombre : $producto['nombre'] }}</h1>
        <img src="{{ asset('images/' . (is_object($producto) ? $producto->imagen : $producto['imagen'])) }}" alt=
"{{ is_object($producto) ? $producto->nombre : $producto['nombre'] }}">
        <p>Precio: ${{ is_object($producto) ? $producto->precio : $producto['precio'] }}</p>
    </div>
@endsection
```

- **@extends('layouts.app')**: Extiende el layout base.
- **@section('content')**: Define el contenido de la página de detalle del producto. Aquí se muestra el título, la imagen y el precio del producto.

Vista Parcial para Catálogo de Productos
(resources/views/partials/catalogo.blade.php)

```
<div class="catalogo">
    @foreach($productos as $producto)
        <div class="producto">
            <img src="{{ asset('images/' . (is_object($producto) ? $producto->imagen : $producto['imagen'])) }}" alt=
"{{ is_object($producto) ? $producto->nombre : $producto['nombre'] }}">
            <h3>{{ is_object($producto) ? $producto->nombre : $producto['nombre'] }}</h3>
            <p>{{ is_object($producto) ? $producto->descripcion : $producto['descripcion'] }}</p>
            <p>Precio: ${{ is_object($producto) ? $producto->precio : $producto['precio'] }}</p>
            <a href="/producto/{{ is_object($producto) ? $producto->id : $producto['id'] }}">Ver Producto</a>
        </div>
    @endforeach
</div>
```

- **@foreach($productos as $producto)**: Recorre el array de productos y los muestra como tarjetas.
- **{{ asset('images/...') }}**: Inserta la ruta de la imagen del producto usando el helper asset() de Laravel, que genera la URL correcta para los archivos estáticos en public/images.

4. Layout Principal (resources/views/layouts/app.blade.php)

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', 'Catálogo de Productos')</title>
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">
</head>
<body>
    <div class="wrapper" style="min-height: 100vh; display: flex; flex-direction: column;">
        <div class="content" style="flex: 1;">
            @yield('content')
        </div>
        @include('partials.footer')
    </div>
</body>
</html>
```

- @yield('content'): El contenido de las vistas individuales se inserta aquí.
- @include('partials.footer'): Incluye el pie de página en todas las páginas automáticamente.

Conclusión

El proyecto está dividido en partes lógicas que permiten organizar y visualizar productos en diferentes catálogos. Las rutas controlan el flujo de las solicitudes, el controlador maneja la lógica del negocio, y las vistas se encargan de presentar los datos al usuario.

- Controlador: Centraliza la lógica de los productos.
- Vistas: Se encargan de mostrar los datos formateados y estructurados.
- Rutas: Definen cómo los usuarios navegan en la aplicación.