📘 Documentación Técnica - Tienda Muebles Laravel

Nombre: Izan Celis, Marisa Peña, Diego Cadiz, Saúl Espino, Gabriel Sanchez
Curso: 2ºA DAW
Módulo: Desarrollo Web en Entorno Servidor

📌Índice

1. Introducción

Este documento proporciona una visión general técnica del proyecto "Tienda Muebles", desarrollado en Laravel. El sistema permite la gestión de un catálogo de productos, categorías, y un sistema de carrito de compras con persistencia de pedidos.

La base de datos **tienda_muebles** está diseñada para gestionar una tienda de muebles en línea, permitiendo administrar usuarios, roles, productos, categorías, carritos de compra, galerías e imágenes asociadas. El modelo relacional se basa en una estructura modular que facilita la escalabilidad y mantiene una clara separación entre funcionalidades.

Base de Datos:

🧩 1. Modelo Entidad–Relación (Descripción General)

El diseño de la base de datos se organiza en tres grandes áreas:

A. Gestión de usuarios

- Usuarios
- Roles
- Sesiones (soporte Laravel)

B. Gestión del catálogo

- Productos
- Categorías

- categoria_producto (Tabla pivote N:M)
- Galerías
- Imágenes

C. Gestión del proceso de compra

- Carritos
- carrito_items

Además, incluye tablas auxiliares generadas por Laravel (migrations, jobs, cache, etc.) que no forman parte del núcleo del negocio pero soportan el funcionamiento interno del sistema.

🗂️ 2. Descripción de tablas principales

🔹 2.1. Tabla roles

Propósito: Define los tipos de roles que pueden tener los usuarios del sistema (administrador, cliente, invitado, etc.).

Campos principales:

- `id`
- `nombre`
- timestamps

Relaciones:

- **1:N → usuarios**
  Un rol puede ser asignado a múltiples usuarios.

🔹 2.2. Tabla usuarios

Propósito: Almacena los datos de los usuarios registrados.

Campos principales:

- `id`
- `rol_id` (FK)
- `nombre`, `apellidos`
- `email`, `password`
- `intentos_fallidos`, `bloqueado_hasta` (control de seguridad)
- timestamps

Relaciones:

- **N:1 → roles** (cada usuario tiene un rol)
- **1:N → carritos**

🔹 2.3. Tabla carritos

Propósito: Representa el carrito de compra activo de un usuario o visitante.

Campos principales:

- `id`
- `usuario_id` (FK)
- `session_id` (para usuarios invitados)
- timestamps

Relaciones:

- **N:1 → usuarios**
- **1:N → carrito_items**

🔹 2.4. Tabla carrito_items

Propósito: Contiene los productos que un usuario agrega al carrito.

Campos:

- `id`
- `carrito_id` (FK)
- `producto_id` (FK)
- `cantidad`
- `precio_unitario`
- timestamps

Relaciones:

- **N:1 → carritos**
- **N:1 → productos**

🔹 2.5. Tabla productos

Propósito: Almacena la información de los productos disponibles en la tienda.

Campos principales:

- `id`
- `nombre`
- `descripcion`
- `precio`
- `stock`
- `materiales`
- `dimensiones`
- `color_principal`
- `destacado` (booleano)

- `imagen_principal`
- timestamps

Relaciones:

- **N:M** → **categorias** (vía tabla pivote)
- **1:1** → **galerias**
- **1:N** → **carrito_items**

🔹 2.6. Tabla categorias

Propósito: Clasifica los productos por categorías (sofás, mesas, decoración, etc.).

Campos principales:

- `id`
- `nombre`
- `descripcion`
- timestamps

Relaciones:

- **N:M** → **productos** (vía categoria_producto)

🔹 2.7. Tabla categoria_producto (pivote)

Propósito: Implementa la relación de muchos a muchos entre productos y categorías.

Campos:

- `categoria_id`
- `producto_id`
- timestamps

Relaciones:

- **N:1 → categorias**
- **N:1 → productos**

🔹 2.8. Tabla galerias

Propósito: Contiene las galerías asociadas a cada producto.

Campos:

- `id`
- `producto_id` (FK)
- timestamps

Relaciones:

- **1:1 → productos**
- **1:N → imagenes**

🔹 2.9. Tabla `imagenes`

Propósito: Guarda las imágenes que componen la galería de un producto.

Campos:

- `id`
- `galeria_id` (FK)
- `ruta`
- `es_principal` (tinyint)
- `orden` (posición dentro de la galería)
- timestamps

Relaciones:

- **N:1 → galerias**

🔗 3. Relaciones entre las entidades principales

Las relaciones clave del modelo son:

1. **ROLES → USUARIOS (1:N)**
   Un rol puede tener varios usuarios.

2. **USUARIOS → CARRITOS (1:N)**
   Un usuario puede tener múltiples carritos a lo largo de su ciclo.

3. **PRODUCTOS ↔ CATEGORIAS (N:M)**
   Un producto puede pertenecer a varias categorías y viceversa.

4. **PRODUCTOS → GALERIAS (1:1)**
   Cada producto tiene una única galería.

5. **GALERIAS → IMAGENES (1:N)**
   Una galería puede contener múltiples imágenes.

6. **CARRITOS → CARRITO_ITEMS (1:N)**
   Cada carrito contiene muchos items.

7. **PRODUCTOS → CARRITO_ITEMS (1:N)**
   Un producto puede estar presente en varios items de carritos.

🧾 4. Explicación del funcionamiento global

La base de datos permite modelar todo el proceso de una tienda:

Usuarios y roles

Los usuarios se registran y se les asigna un rol que determina sus permisos (administrador o cliente). Cada usuario puede mantener carritos persistentes gracias al almacenamiento por sesión o UID.

Catálogo de productos

Los productos se clasifican en categorías mediante una relación N:M.
Cada producto posee una galería que contiene múltiples imágenes ordenadas, lo que permite una presentación visual completa.

Carrito de compras

El carrito se crea al iniciar una sesión o navegar como invitado.
Los productos añadidos generan registros en carrito_items, permitiendo calcular subtotales, cantidades y totales finales.

Soporte del framework

Tablas como migrations, cache, sessions o jobs son generadas automáticamente por Laravel y permiten:

- Control de migraciones
- Gestión de colas
- Persistencia de sesiones
- Cacheo de información

---

2. Estructura del Proyecto

El proyecto sigue la arquitectura MVC (Modelo-Vista-Controlador) estándar de Laravel.

Directorios Principales

- **`app/Http/Controllers`**: Contiene la lógica de negocio y manejo de peticiones.
- **`app/Models`**: Define la estructura de datos y relaciones.
- **`resources/views`**: Contiene las plantillas Blade para la interfaz de usuario.
- **`routes`**: Define las rutas web y API del sistema.

---

3. Rutas (`routes/web.php`)

El archivo de rutas define los puntos de entrada de la aplicación.

Grupos de Rutas

- **Públicas:**
  - `/`: Página de inicio (`HomeController@index`).
  - `/products`: Listado de productos (`ProductController@index`).

    - `/products/{product}`: Detalle de producto (`ProductController@show`).
    - `/carrito`: Vista del carrito (`CarritoController@index`).
- **Autenticación:**
    - `/login`, `/register`: Rutas para inicio de sesión y registro (`AuthController`).
    - `/logout`: Cierre de sesión.
- **Administración (Middleware `auth` y `admin`):**
    - Permite crear, editar y eliminar productos (`ProductController`).
    - Gestión de categorías (`CategoryController`).
    - Gestión de imágenes de productos (`ImagenProductoController`).
- **Usuario Autenticado (Middleware `auth`):**
    - `/carrito/guardar-en-bd`: Convierte el carrito de sesión en un pedido persistente.
    - `/carrito/historial`: Visualización de pedidos anteriores.
    - `/preferences`: Gestión de preferencias de usuario (tema, moneda, etc.).

---

4. Controladores (`app/Http/Controllers`)

ProductController

Gestiona el catálogo de productos.

- `index()`: Lista productos con paginación.
- `create()` / `store()`: Muestra formulario y guarda nuevos productos, incluyendo subida de imágenes.
- `edit()` / `update()`: Modifica productos existentes.
- `destroy()`: Elimina productos del sistema.
- **Validaciones:** Asegura que los campos obligatorios (nombre, precio, stock) sean correctos antes de guardar.

CarritoController

Maneja la lógica del carrito de compras.

- **Gestión de Sesión:** Los productos se almacenan temporalmente en la sesión del usuario (`Session::get('carrito')`).
- `agregar()`: Añade productos al carrito, verificando disponibilidad de stock.
- `actualizar()`: Modifica la cantidad de productos, validando que no exceda el stock real.
- `guardarEnBD()`:
    - Verifica autenticación.

    - Valida stock final.
    - Inicia una transacción de base de datos (`DB::beginTransaction`).
    - Crea registros en las tablas `carritos` y `carrito_items`.
    - Descuenta el stock de los productos (`$producto->reducirStock()`).
    - Limpia el carrito de la sesión.

AuthController

Gestiona la seguridad y acceso.

- `login()`: Autentica usuarios y migra preferencias guardadas en cookies a la base de datos.
- `register()`: Crea nuevos usuarios con rol de "Cliente" por defecto.
- `migrarPreferenciasDeCookies()`: Sincroniza configuraciones visuales (tema oscuro, etc.) cuando un usuario inicia sesión.

CategoryController

CRUD simple para gestionar las categorías de los muebles (ej. Salas, Dormitorios).

---

5. Modelos (`app/Models`)

Producto

Representa un artículo en venta.

- **Relaciones:**
    - `categorias()`: Relación muchos a muchos (N:M) con `Categoria`.
    - `galeria()`: Relación uno a uno con `Galeria` (imágenes adicionales).
- **Métodos Helper:**
    - `tieneStock($cantidad)`: Verifica disponibilidad.
    - `reducirStock($cantidad)`: Disminuye el inventario de forma segura.
- **Scopes:**
    - `scopeDestacados()`, `scopeDisponibles()`: Filtros predefinidos para consultas rápidas.

Carrito

Representa un pedido finalizado (guardado en BD).

- **Relaciones:**
    - `usuario()`: El cliente que realizó el pedido.
    - `items()`: Los detalles de los productos comprados (`CarritoItem`).
- **Lógica:** Calcula el total del pedido sumando sus items.

User

El usuario del sistema.

- Contiene campos para preferencias de interfaz (tema, paginacion, moneda) que se sincronizan con `AuthController`.

---

6. Vistas (`resources/views`)

Layout Principal (`layout/cabecera.blade.php`)

Define la estructura común (HTML, HEAD, estilos globales) y la barra de navegación. Todas las vistas principales extienden de este archivo o lo incluyen.

Home (`home.blade.php`)

Página de aterrizaje con un banner principal, categorías destacadas y una grilla de productos destacados.

Productos (`productos/index.blade.php`)

Tabla de administración para listar, editar y borrar productos. Muestra imágenes en miniatura y botones de acción.

Carrito (`carrito/index.blade.php`)

Interfaz del carrito de compras.

- Muestra tabla con productos, precios y subtotales.
- Calcula impuestos y total general.
- Botón para "Finalizar Compra" que invoca a `guardarEnBD`.

---

7. Middleware y Seguridad

- **Auth:** Protege rutas que requieren inicio de sesión (historial de pedidos).
- **Admin:** Middleware personalizado que restringe el acceso a la gestión de productos solo a administradores.

- **Validación de Stock:** Implementada tanto al añadir al carrito como al finalizar la compra para evitar inconsistencias.

---

8. Base de Datos (Migraciones)

El esquema de base de datos se define mediante migraciones en `database/migrations`.

Tablas Principales

- **`productos`**: Almacena el catálogo completo.
  - **Campos clave:** `precio` (decimal), `stock` (entero), `destacado` (booleano).
  - **Detalles específicos:** `materiales` (texto), `dimensiones` (string), `color_principal`.
  - **Índices:** En `destacado`, `precio` y `color_principal` para optimizar filtros.
- **`carritos`**: Tabla transaccional para pedidos.
  - **`usuario_id`**: Relación con `users` (nullable, permite carritos de invitados si se implementara).
  - **`sesion_id`**: Identificador único para rastrear carritos de sesión antes de guardarlos.
  - **`total`**: Monto final de la compra.
- **`carrito_items`**: Detalle de cada pedido.
  - Relaciona `carrito_id` con `producto_id`.
  - Guarda el **`precio_unitario` histórico** (el precio al momento de la compra) y la `cantidad`.

---

9. Datos de Prueba (Seeders y Factories)

El sistema incluye herramientas para poblar la base de datos con información ficticia pero realista para desarrollo y pruebas.

Factories (`database/factories`)

Definen cómo se generan los modelos de forma automática.

- **`ProductoFactory`**:
  - Genera nombres compuestos (ej. "Mesa Moderna de Roble").
  - Crea descripciones y materiales coherentes.

    - **Lógica Inteligente:** Asigna dimensiones realistas según el tipo de mueble (ej. una "Cama" tendrá dimensiones de cama, diferentes a una "Silla").
    - **Estados:** Métodos como `destacado()`, `sinStock()` o `muchoStock()` para variar los datos generados.

Seeders (`database/seeders`)

Scripts para ejecutar la carga de datos.

- `DatabaseSeeder`: Orquestador principal. Llama a otros seeders en orden secuencial (Roles -> Usuarios -> Productos -> Categorías).
- `DemoSeeder`: Crea usuarios fijos para pruebas:
    - **Cliente:** `cliente@demo.com` / `password`
    - **Admin:** `admin@demo.com` / `password`
    - Genera 5 categorías y 20 productos aleatorios.
    - Asigna imágenes de placeholder a los productos y crea galerías asociadas.