Documentación Técnica del Proyecto: COURSEMARKET

1. Información General

Nombre del Proyecto: CourseMarket Laravel
Descripción: Plataforma web que permite la publicación, administración y compra de productos. Se contemplan distintos roles de usuario: cliente y administrador.
Tecnologías utilizadas:

- Laravel 10
- PHP 8
- Blade (motor de plantillas)
- Bootstrap 4
- MySQL
- Composer y Artisan

Realizamos esta sección para definir las bases del proyecto, identificar las tecnologías a utilizar y establecer una visión clara del sistema.

---

2. Estructura del Proyecto

Directorio principal:

```
app/
  Http/
    AdminAuthMiddleware.php
  Models/
    Categoria.php
    Producto.php
    Usuario.php
    User.php
    Venta.php
resources/
  views/
    productos/
      create.blade.php
      edit.blade.php
      index.blade.php
      show.blade.php
    admin_login.blade.php
    admin.blade.php
    inicio.blade.php
    login.blade.php
    register.blade.php
routes/
```

```
  web.php
  console.php
config/
  *.php (archivos de configuración de Laravel)
```

Esta estructura refleja cómo organizamos los archivos del proyecto para mantener el orden entre modelos, vistas, controladores y rutas.

---

3. Middleware Personalizado: `AdminAuthMiddleware`

Ubicación: `app/Http/AdminAuthMiddleware.php`
Descripción: Middleware que restringe el acceso a las rutas de administrador.

Lógica del Middleware:

- Permite acceso libre a `/admin/login` y su submit.
- Verifica si hay sesión iniciada con la clave `admin_auth`.
- Valida que el usuario tenga rol `admin`.
- Redirige con error si el usuario no está autenticado o no tiene permisos.

Resumen del Código:

```php
if (!$request->session()->has('admin_auth')) {
    return redirect('/admin/login');
}
$usuario = Usuario::find($request->session()->get('admin_auth'));
if (!$usuario || $usuario->rol !== 'admin') {
    return redirect('/admin/login')->with('error', 'No tienes permisos.');
}
```

Implementamos este middleware para proteger las rutas administrativas y asegurar que solo usuarios autorizados puedan acceder a ellas.

---

4. Modelos Eloquent

`Categoria.php`

- Tabla: `categorias`
- Clave primaria: `id_categoria`
- Campos fillable: `nombre_categoria`, `descripcion`
- Relaciones: Tiene muchos productos

```php
public function productos() {
    return $this->hasMany(Producto::class, 'id_categoria', 'id_categoria');
}
```

Creamos este modelo para representar las categorías de productos y gestionar su relación con los productos.

`Producto.php`

- Tabla: `productos`
- Clave primaria: `id_producto`
- Campos fillable: `nombre`, `precio`, `descripcion`, `id_categoria`, etc.
- Relaciones: Pertenece a una categoría

```
public function categoria() {
    return $this->belongsTo(Categoria::class, 'id_categoria',
'id_categoria');
}
```

Este modelo permite manipular los datos de los productos y vincularlos con su categoría correspondiente.

`Usuario.php`

- Tabla: `usuarios`
- Campos fillable: `nombre`, `email`, `password`, `rol`
- Roles posibles: `admin`, `cliente`
- Relaciones: Puede tener muchas ventas

Creamos este modelo para registrar y autenticar tanto a clientes como administradores.

`Venta.php`

- Tabla: `ventas`
- Relaciones:

```
public function usuario() {
    return $this->belongsTo(Usuario::class, 'id_usuario');
}
```

Este modelo representa las compras realizadas por los usuarios y su relación con ellos.

---

5. Rutas Web

Archivo: `routes/web.php`

Rutas de administración

```
Route::get('/admin/login', [AdminController::class, 'login']);
Route::post('/admin/login/submit', [AdminController::class, 'submit']);
```

Rutas protegidas por middleware admin

```
Route::middleware(['admin.auth'])->group(function() {
    Route::resource('productos', ProductoController::class);
});
```

Rutas para usuarios

```
Route::get('/login', [UsuarioController::class, 'login']);
Route::post('/login', [UsuarioController::class, 'auth']);
Route::get('/register', [UsuarioController::class, 'register']);
Route::post('/register', [UsuarioController::class, 'store']);
```

> Definimos estas rutas para manejar la navegación del sistema, dividiendo claramente las funciones administrativas y de usuario.

---

6. Vistas Blade

Carpeta: `resources/views/`

Productos (CRUD):

- `index.blade.php`: Lista de productos.
- `create.blade.php`: Formulario para crear un producto.
- `edit.blade.php`: Edición de producto.
- `show.blade.php`: Vista detallada del producto.

Autenticación:

- `login.blade.php`: Inicio de sesión de usuario.
- `register.blade.php`: Registro de usuario.
- `admin_login.blade.php`: Login de administradores.

Paneles:

- `admin.blade.php`: Panel de control de admin.
- `inicio.blade.php`: Vista inicial o dashboard para usuarios.

> Estas vistas fueron diseñadas para ofrecer una experiencia de usuario clara y diferenciada entre clientes y administradores.

---

7. Validaciones y Seguridad

- Validaciones de formularios usando `Request` y `validate()`.
- Middleware para proteger rutas según rol.
- Protección CSRF automática con `@csrf`.
- Uso de sesiones Laravel.

> Aplicamos estas medidas para garantizar la seguridad de los datos y evitar accesos no autorizados.

---

8. Base de Datos (Estructura principal)

Tabla: `usuarios`

Campo	Tipo	Descripción
id_usuario	INT	PK
nombre	VARCHAR	Nombre del usuario
email	VARCHAR	Correo electrónico
password	VARCHAR	Contraseña cifrada
rol	ENUM	admin / cliente

Tabla: `categorias`

Campo	Tipo
id_categoria	INT
nombre_categoria	VARCHAR
descripcion	TEXT

Tabla: `productos`

Campo	Tipo
id_producto	INT
nombre	VARCHAR
precio	DECIMAL
descripcion	TEXT
id_categoria	INT (FK)

Tabla: `ventas`

Campo	Tipo
id_venta	INT
id_usuario	INT
total	DECIMAL
fecha	DATETIME

> Diseñamos esta base de datos para organizar adecuadamente las relaciones entre usuarios, productos, ventas y categorías.

---

9. Consideraciones Finales

- Se sigue el patrón MVC de Laravel.
- El código está modularizado y usa buenas prácticas.
- Hay separación de lógica de negocio, vistas y rutas.
- Middleware personalizado permite control de accesos robusto.

Estas decisiones facilitaron el desarrollo colaborativo y el mantenimiento del proyecto.

---

10. Posibles Mejoras a Futuro

- Agregar autenticación con Laravel Breeze o Jetstream.
- Usar Laravel Policies para autorizaciones.
- Agregar sistema de carrito y pagos.
- Tests automatizados (PHPUnit).
- Implementar API REST para acceso desde apps móviles.

Estas mejoras permitirían escalar el proyecto y optimizar su funcionalidad en el futuro.
