Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División Ciencias de la Ingeniería

Manejo e Implementación de Archivos

Sección “A”

“MANUAL DE TÉCNICO”

Kenny Marco Augusto López Salazar

202031554

Quetzaltenango 28 de Octubre 2025

“E-Commerce_GT”

A continuación se detallan todas herramientas utilizadas para la realización de este proyecto y como una explicación de cómo las clases interactúan entre sí

1. DATOS GENERALES DEL PROYECTO

Nombre del sistema: E-Commerce GT
Backend: Spring Boot versión **3.5.6**
Frontend: Angular versión **17.0.3.17**
Base de datos: PostgreSQL
Despliegue Backend: Ngrok (exposición local con HTTPS)
Despliegue Frontend: Netlify

2. ARQUITECTURA GENERAL

El sistema está dividido en dos partes principales:

1. **Backend (API REST)**: desarrollado con Spring Boot, expone los servicios y gestiona la lógica de negocio.
2. **Frontend (Interfaz Web)**: construido en Angular, consume los endpoints del backend.

El backend usa JWT (JSON Web Token) para autenticar usuarios y manejar roles.
Cada módulo del sistema está separado por carpetas:

- **Auth:** login, registro y generación del token
- **Carrito:** carrito de compras, pagos y pedidos
- **Producto:** gestión de productos y moderación
- **Notificación:** alertas y avisos a usuarios
- **Admin:** reportes y estadísticas
- **Seguridad:** validación del token y roles

3. RELACIÓN ENTRE CLASES PRINCIPALES

3. 1 Módulo de Autenticación (auth)

- **AuthController:** recibe las peticiones /api/auth/login y /api/auth/register.
- **AuthService:** valida las credenciales del usuario en la base de datos y genera el JWT.
- **JwtUtil:** crea, valida y lee los tokens JWT.

- **JwtFiltro:** intercepta las peticiones y, si el token es válido, coloca al usuario autenticado en el contexto de seguridad de Spring.
- **UsuarioRepository / RolRepository:** acceden a la base de datos para obtener información del usuario y su rol.

Flujo de autenticación:

1. El usuario inicia sesión desde Angular: /api/auth/login
2. El backend valida el correo y contraseña.
3. Si son correctos, genera un token JWT con información del usuario (nombre, rol, id).
4. Angular guarda este token en localStorage.
5. En cada petición siguiente, Angular lo envía en el encabezado: Authorization: Bearer <token>.
6. El filtro JwtFiltro lo intercepta, válida y permite el acceso si el token es válido.

3.2 Módulo de Productos (producto)

- **ProductoController:** gestiona la creación, actualización y listado de productos.
- **ProductoService:** contiene la lógica para registrar productos nuevos, guardar imágenes y asociar categorías.
- **ProductoRepository:** maneja las consultas SQL (con JPA).
- **ProductoImagenRepository:** guarda y obtiene imágenes de los productos.
- **ModeracionService / ModeradorController:** permiten que los moderadores aprueben o rechacen productos antes de publicarse.
- **ProductoReviewController / ProductoReviewRepository:** controlan los comentarios y calificaciones de productos.

Relación:
Cada Producto tiene una Categoría, una Imagen principal, un Vendedor y un Estado de moderación (Pendiente, Aprobado o Rechazado).
Los moderadores cambian ese estado desde su panel de control.

3.3 Módulo de Carrito y Pedidos (carrito)

- **CarritoController:** permite agregar, actualizar o eliminar productos del carrito.
- **CarritoService:** realiza toda la lógica del carrito (sumar totales, revisar stock, crear pedidos).

- **PedidoRepository / PedidoItemRepository:** guardan la información de las compras realizadas.
- **PagoRepository:** almacena los datos de pago.
- **EstadoPedidoRepository:** gestiona los distintos estados de un pedido (en curso, entregado, cancelado, etc.).
- **GananciasService / GananciasController:** calculan las comisiones y ganancias de cada vendedor.

Relación:
Un Usuario tiene un Carrito, y dentro de este hay CarritoItems (productos + cantidad).
Cuando se confirma la compra, se crea un Pedido y sus PedidoItems.
El Pago queda asociado al pedido, y se actualizan los estados automáticamente.

3.4 Módulo de Notificaciones (notificacion)

- **NotificacionController:** devuelve las notificaciones o mantiene una conexión abierta (SSE).
- **NotificacionService:** crea y envía notificaciones nuevas.
- **NotificacionHub:** mantiene las conexiones activas con los usuarios conectados.
- **NotificacionRepository:** guarda el historial de notificaciones en la base de datos.

Relación:
Cuando un pedido cambia de estado o se aprueba un producto, el sistema genera una notificación que se envía al usuario por correo y también aparece en tiempo real en el frontend.

3.5 Módulo de Administración (admin)

- **GananciasAdminController / Service:** calculan los ingresos totales del sistema en un rango de fechas.
- **PedidoReportesRepository / ProductoReportesRepository:** generan estadísticas como los productos más vendidos o los clientes con más compras.
- **NotificacionReportesRepository:** genera reportes del historial de notificaciones enviadas.

3.6 Módulo de Seguridad (seguridad)

- **SecurityConfig:** define las rutas protegidas y los roles que pueden acceder.
- **JwtFiltro:** intercepta las peticiones y valida el token.
- **JwtUtil:** genera y verifica los tokens firmados.

4. FUNCIONAMIENTO DE LA AUTENTICACIÓN JWT

1. El usuario inicia sesión con su correo y contraseña.
2. El backend valida las credenciales usando UsuarioRepository.login().
3. Si son correctas, el AuthService llama a JwtUtil.generarToken().

El token incluye datos como: { “rol”: “COMUN”, “nombre”: ”Kenny”, “userId”: 15 }

4. Este token se envía al cliente (Angular), que lo guarda localmente.
5. En cada petición protegida, Angular agrega el token en los encabezados.
6. El JwtFiltro intercepta la solicitud y.
    1. Verifica la firma del token.
    2. Extrae el rol y el correo.
    3. Coloca los datos en el contexto de seguridad (SecurityContextHolder)
7. Spring Security usa esa información para decidir si el usuario puede acceder a la ruta solicitada.
8. Esto son genera algo como: Authorization: Bearer eyJhbGciOiJIUzI1NiJ9…
9. Con esto no se necesita mantener sesiones en el servidor; el token es autosuficiente y seguro mientras no expire.

5. DESPLIEGUE

5.1 Backend con Ngrok:

1. Ejecutar el proyecto con: **mvn spring-boot:run**
2. Levantar Ngrok para exponer el backend: **ngrok http 8080**

5.2 Frontend en Netlify:

1. Generar la build de Angular: **ng build --configuration production**
2. Subir la carpeta dist/ al panel de Netlify.
3. Crear un archivo _redirects en src/ con este contenido: /* /index.html 200
4. Esto permite que Angular maneje las rutas sin error 404.

6. CONCLUSIÓN

El sistema E-Commerce GT está construido sobre una arquitectura limpia y modular, con seguridad basada en JWT, y una comunicación fluida entre frontend y backend.

Spring Boot maneja toda la lógica de negocio, PostgreSQL almacena la información estructurada, y Angular ofrece una interfaz moderna y responsiva.

Ngrok y Netlify permiten desplegar el sistema fácilmente sin necesidad de servidores dedicados.