Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ciencias de la Ingeniería Ingeniería en Ciencias y Sistemas.
Laboratorio Archivos

ECOMMERCE GT Manual Tecnico

Octubre 27 del 2,025

Enmer Oswaldo Sandoval Mazariegos **202031773**

Manual Tecnico

Plataforma web de comercio electrónico con Angular (SPA) + Spring Boot 3 y PostgreSQL (esquema 'ecommerce_gt'). Incluye autenticación JWT, control de roles (COMUN, MODERADOR, LOGISTICA, ADMIN), catálogo, carrito, ventas, calificaciones y reportes.

1) Resumen del proyecto

eCommerce GT es una plataforma de comercio electrónico con frontend Angular 17–19 y backend Spring Boot 3. La base de datos es PostgreSQL utilizando el esquema 'ecommerce_gt'. El sistema cubre autenticación, autorización por roles, catálogo de productos, carrito de compras, proceso de venta, calificaciones/comentarios y reportes de negocio.

2) Arquitectura

2.1 Vista lógica

- Cliente (Angular SPA): rutas protegidas por guard e interceptor (Authorization: Bearer <token>).
- API REST (Spring Boot): controllers, services, repositories, domain (entities/DTOs), security (JWT/CORS).
- Base de Datos: PostgreSQL (esquema 'ecommerce_gt'), índices en FKs y unicidad en campos críticos.

2.2 Seguridad

- Login con emisión de JWT (expiración configurable).
- Autorización por roles en endpoints.
- Contraseñas con BCrypt.
- CORS restringido por origen (Angular).
- Validación y sanitización en DTOs (Bean Validation).

3) Entorno y despliegue

3.1 Requisitos

- Node 20+, Angular CLI 17–19
- JDK 21, Maven 3.9+
- PostgreSQL 15+ (UTF8)
- Git

3.2 Variables de entorno (backend)

- SPRING_DATASOURCE_URL=jdbc:postgresql://<host>:5432/<db>
- SPRING_DATASOURCE_USERNAME=<user>
- SPRING_DATASOURCE_PASSWORD=<pass>
- SPRING_JPA_HIBERNATE_DDL_AUTO=validate
- JWT_SECRET=<clave-secreta-256bits>
- JWT_EXPIRATION_MS=3600000
- CORS_ALLOWED_ORIGINS=https://<frontend>

3.3 Construcción

Frontend

- cd frontend
- npm ci
- ng build --configuration=production

Backend

- cd backend
- mvn -DskipTests clean package
- java -jar target/ecommerce-gt.jar

3.4 Despliegue de referencia

- Frontend: hosting estático (Netlify/S3/NGINX).
- Backend: VM/Container (EC2/Docker) detrás de NGINX o ALB.
- BD: RDS PostgreSQL o instancia administrada.
- Almacenamiento de imágenes: URL externas firmadas/host público (guardar solo el link en BD).

4) Modelo de datos (PostgreSQL – esquema 'ecommerce_gt')

Resumen de entidades más relevantes. Los nombres pueden ajustarse a tu script actual; mantener índices en FKs y unicidad donde aplique.

4.1 Usuarios, acceso y páginas

- roles (id, nombre UNIQUE)
- users (id, nombre, username UNIQUE, email UNIQUE, password, is_active, role_id FK, created_at)
- modules (id, nombre UNIQUE, path UNIQUE, is_enabled)
- pages (id, nombre UNIQUE, path UNIQUE, module_id FK, is_enabled)
- role_has_pages (role_id FK, page_id FK, can_create, can_update, can_delete)

4.2 Catálogo

- genders (id, nombre)
- items (id, title, artist, price, dimensions, gender_id FK, publication YEAR, stock, is_active)
- vinyls (id, item_id FK, size, is_special, color, brand)
- cassettes (id, item_id FK, category)
- cds (id, item_id FK)
- item_has_images (id, item_id FK, url)

4.3 Interacción y eventos

- comments (id, item_id FK, user_id FK, content, created_at, is_deleted)
- qualifications (id, item_id FK, user_id FK, stars CHECK 1..5)
- events (id, title, description, date, item_id FK)
- inscriptions (id, event_id FK, user_id FK)
- event_chat (id, event_id FK, user_id FK, message, created_at)

4.4 Carrito y ventas

- carts (id, user_id FK, created_at, is_available)
- cart_has_items (id, cart_id FK, item_id FK, amount)
- sales (id, user_id FK, total, created_at)

- sales_detail (id, sale_id FK, item_id FK, amount, price)

4.5 Preventas y promociones

- pre_sales (id, item_id FK, is_active)
- user_has_pre_sale (id, pre_sale_id FK, user_id FK)
- promotions (id, is_random, discount, created_at, date_end)
- cd_has_promotion (id, promotion_id FK, cd_id FK)

4.6 OTP y correo

- otps (id, user_id UNIQUE FK, otp_code UNIQUE, expires_at)
- mail_templates (id, html)

Lineamientos: Soft delete en items.is_active; transacciones ACID en checkout/stock; unicidad en users.email/username, roles.nombre, modules/pages.path.

5) Endpoints principales (resumen)

5.1 Autenticación

- POST /auth/login → JWT
- POST /auth/refresh
- POST /auth/logout
- GET /auth/me

5.2 Catálogo

- GET /items / GET /items/{id}
- POST /items (ADMIN/MOD)
- PATCH /items/{id}/stock
- PATCH /items/{id}/active
- GET /vinyls, GET /cassettes, GET /cds

5.3 Interacción

- POST /comments / GET /items/{id}/comments
- POST /qualifications (upsert por usuario/item)
- GET /items/{id}/rating (promedio + conteo)

5.4 Carrito y ventas

- GET /cart / POST /cart/items / PATCH /cart/items/{id} / DELETE /cart/items/{id} / DELETE /cart
- POST /checkout → sales + sales_detail
- GET /sales / GET /sales/{id}

5.5 Preventas y eventos

- GET /pre-sales / POST /pre-sales
- POST /pre-sales/{id}/subscribe
- GET /events / GET /events/active / POST /events / POST /events/{id}/chat

Todos los endpoints protegidos exigen Authorization: Bearer <token> y permisos por rol.

6) Flujos críticos

6.1 Login + sesión

1. Usuario envía credenciales.
2. Backend valida y emite JWT.
3. Frontend guarda token (almacenamiento seguro).
4. Interceptor adjunta token en cada request.

6.2 Carrito → Checkout

5. POST /cart/items (validar stock).
6. POST /checkout (transacción): revalida stock, crea sales + sales_detail, descuenta stock, limpia carrito.

6.3 Calificación y comentarios

- Un (1) rating por usuario/item (upsert).
- Comentarios con eliminación lógica (is_deleted).

7) Semillas y datos de prueba

- Roles: COMUN, MODERADOR, LOGISTICA, ADMIN.
- Usuarios: 10 COMUN, 5 MODERADOR, 3 LOGISTICA, 1 ADMIN.
- Password hash (BCrypt): $2a$10$Ta8nxCEjBBAEle2vEsMzz.VYnPcJ6GQRbnj4t0MxFqY7dKvRbDJIG
- Productos: ≥10 por cada usuario COMUN (como vendedor).
- Imágenes: guardar solo link externo. Archivo DML.sql con inserts por entidad.

8) Calidad, pruebas y observabilidad

- Pruebas unitarias (Junit5 + Mockito) y de integración (WebMvcTest / Testcontainers).
- Validaciones con Bean Validation y ControllerAdvice para errores.
- Logs estructurados y métricas (Micrometer) opcional.

9) Rendimiento y buenas prácticas

- Paginación en listados.
- Índices en FKs y campos de búsqueda.
- Evitar N+1 (fetch join / @EntityGraph).
- Cache HTTP para assets/listados estáticos (frontend/CDN).
- Subida de imágenes fuera del API (presigned URL o servicio dedicado).

10) Mapeo Angular (alto nivel)

- Auth: login/recovery/signup, guard y token-interceptor.
- Catálogo: item.service (lista/detalle), dialogs (vinyl/cassette/cd).
- Carrito: vista de carrito con totales; badge en navbar.
- Admin: usuarios, productos, comentarios, preventas, reportes.
- Eventos: listado, detalle, chat (WebSocket opcional).

11) Plan de trabajo pendiente

- Completar DML.sql definitivo (usuarios, productos, relaciones).
- Endpoint de rating agregado por producto.
- Reportes (top items, top clientes, ventas por período).
- Endpoints/UI de preventa.

- Hardening CORS y expiración/rotación de tokens.
- Pipeline CI/CD (build, test, deploy).

12) Anexos

12.1 Convenciones de código (backend)

- controllers: nombres RESTful (ItemController, CartController…).
- DTOs: *ReqDto, *RespDto.
- Repos: findBy*, countBy*.

12.2 Comandos útiles

- createdb ecommerce_gt
- psql -d ecommerce_gt -c "CREATE SCHEMA IF NOT EXISTS ecommerce_gt;"
- mvn -Dflyway.user=<user> -Dflyway.password=<pass> -Dflyway.url=jdbc:postgresql://... flyway:migrate
- psql -d ecommerce_gt -f DML.sql