**Modelado**  
**Zeyra**  
**SENA**   
**Ficha: 2899747**

**Autor:**

* Juan Sebastián González Horta

**Fecha:** 27/08/2025

**Análisis del Informe de Requisitos**

El documento de **Especificación de Requisitos de Software (SRS)** de Zeyra define las necesidades de tres actores principales:

* **Usuarios**
* **Administradores**

Se identificaron los siguientes **Requisitos Funcionales** principales:

**Funcionalidades para Usuarios**

* RF-01: Inicio de sesión.
* RF-02: Registrarse.
* RF-04: Perfil Usuario.
* RF-06: Productos.

**Funcionalidades para Administrador**

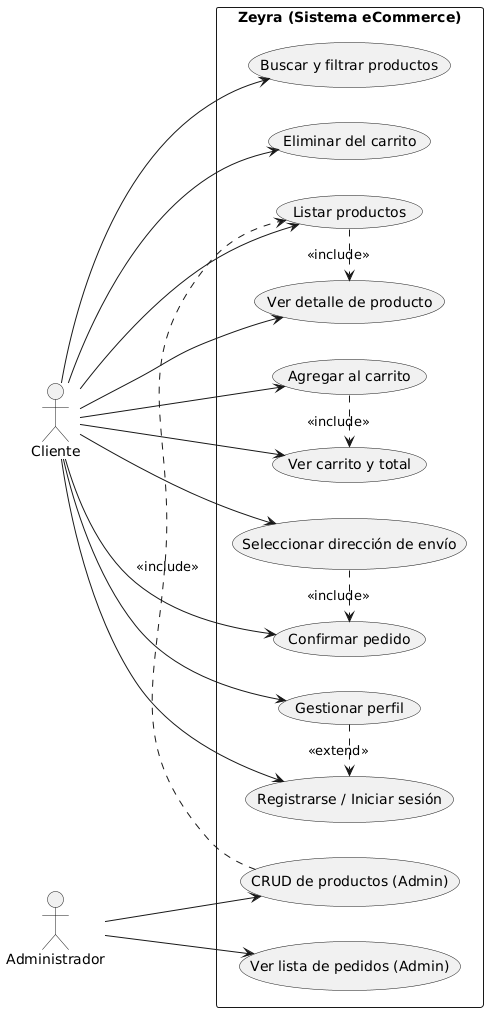
* RF-03: Perfil Admin.
* RF-05: Vistas (Dashboard).

**Mapeo Conceptual de Funciones**

La traducción de requisitos a funciones del sistema se presenta en el siguiente modelo conceptual:

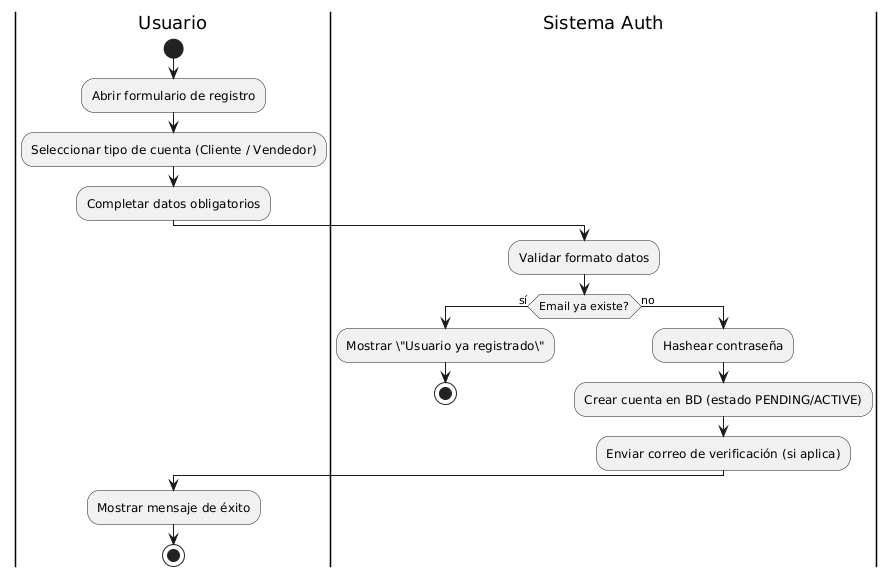
|  |  |  |
| --- | --- | --- |
| **Actor** | **Requisitos Asociados** | **Funciones del Software** |
| **Usuario** | RF-01, RF-02,  RF-04, RF-06 | Inicio sesión · Registrarse · Perfil Usuario · Productos |
| **Administrador** | RF-03, RF-06 | Perfil Admin · Vistas (Dashboard) |

**Diagrama de casos de uso:**

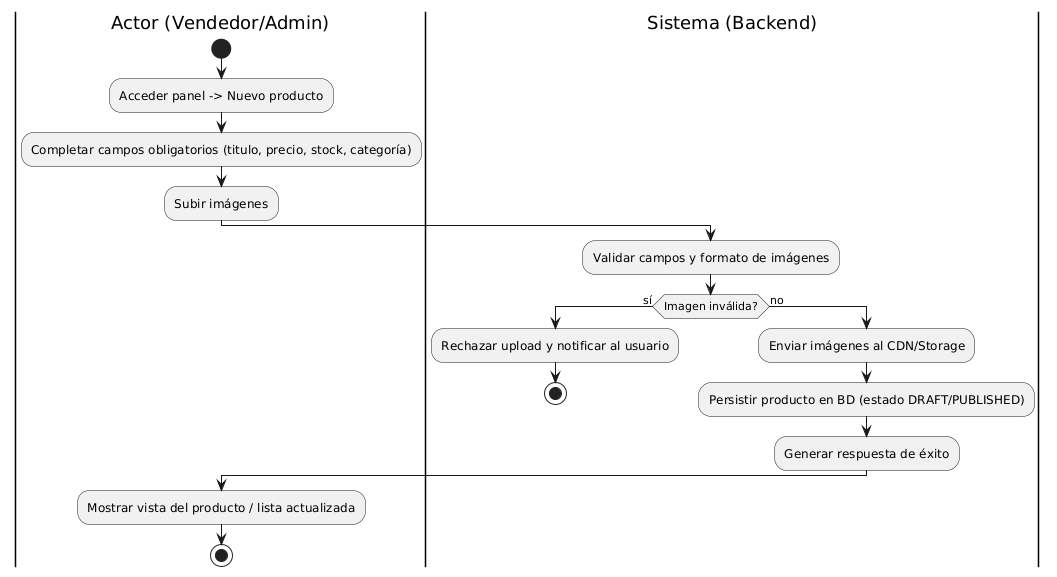


**Diagramas de Actividades de casos de uso:**

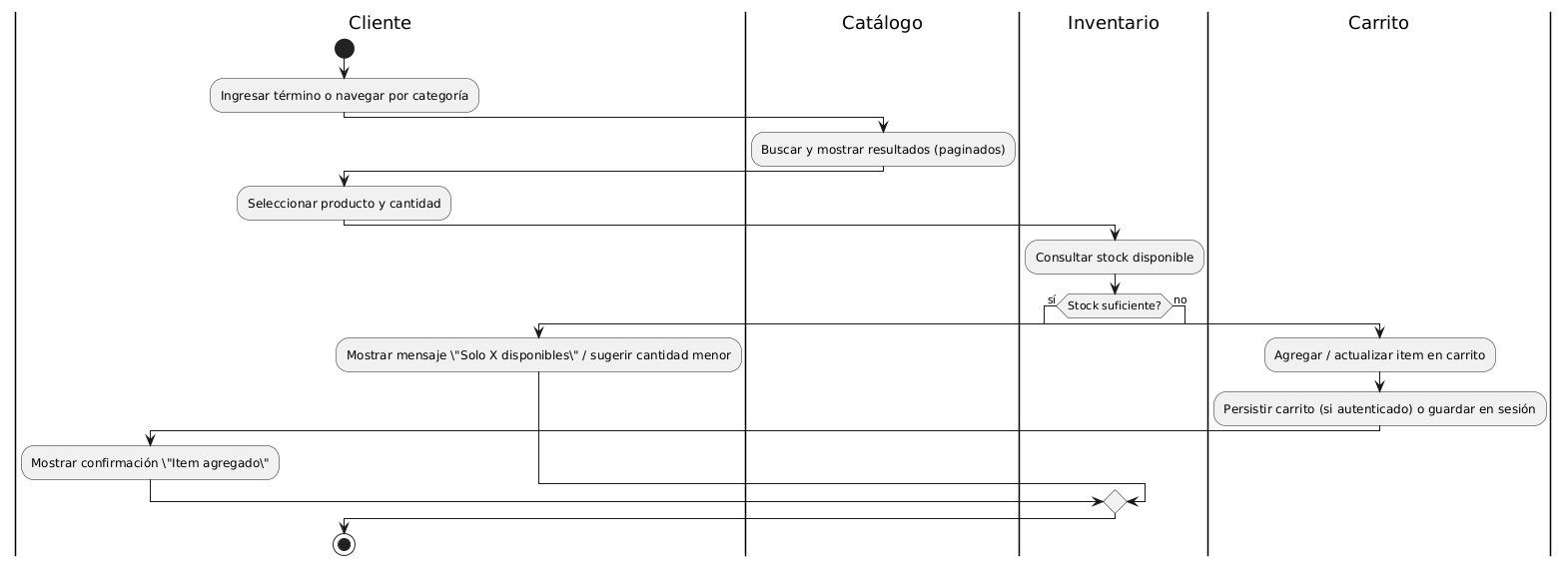
**UC-01 — Registro de Usuario (Cliente / Vendedor)**



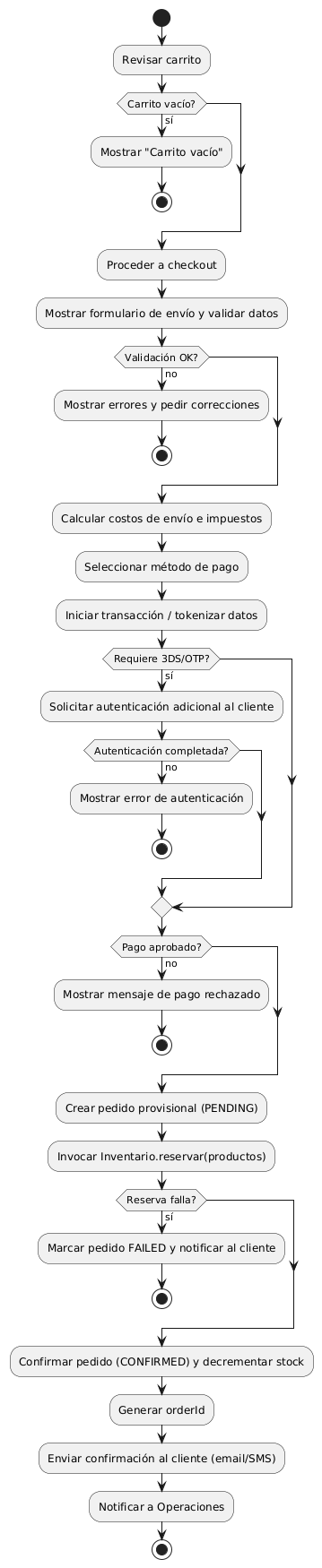
**UC-02 — Publicar / Administrar Producto (Vendedor / Admin)**



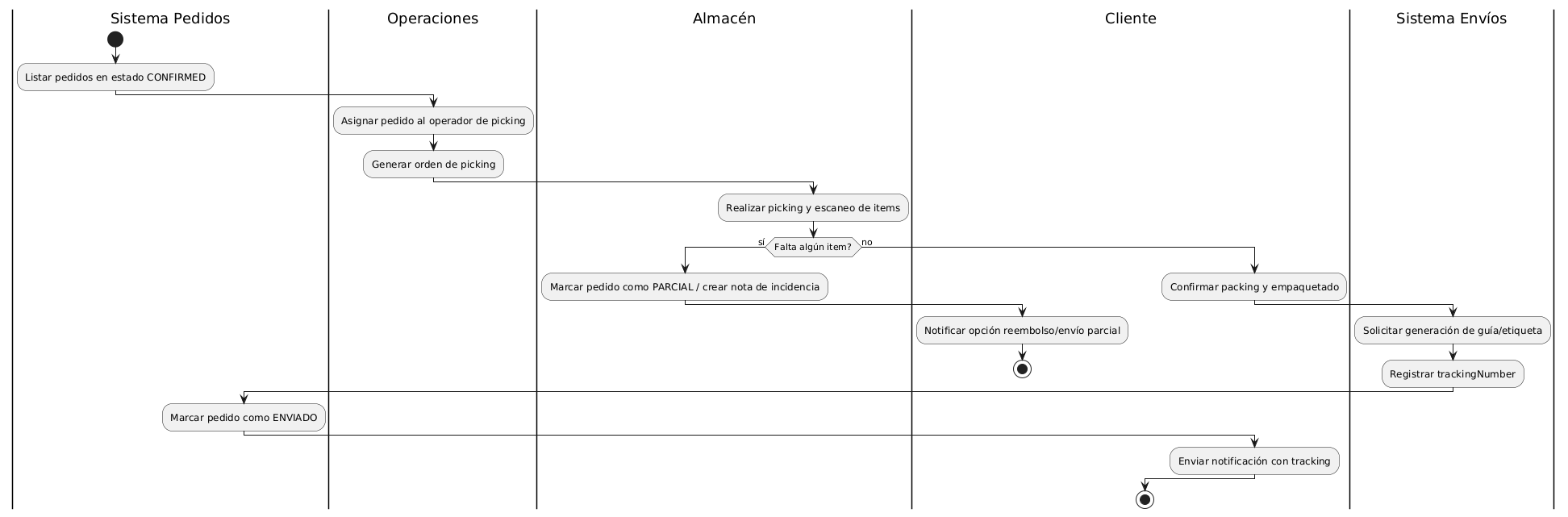
**UC-03 — Buscar, Filtrar y Añadir al Carrito**



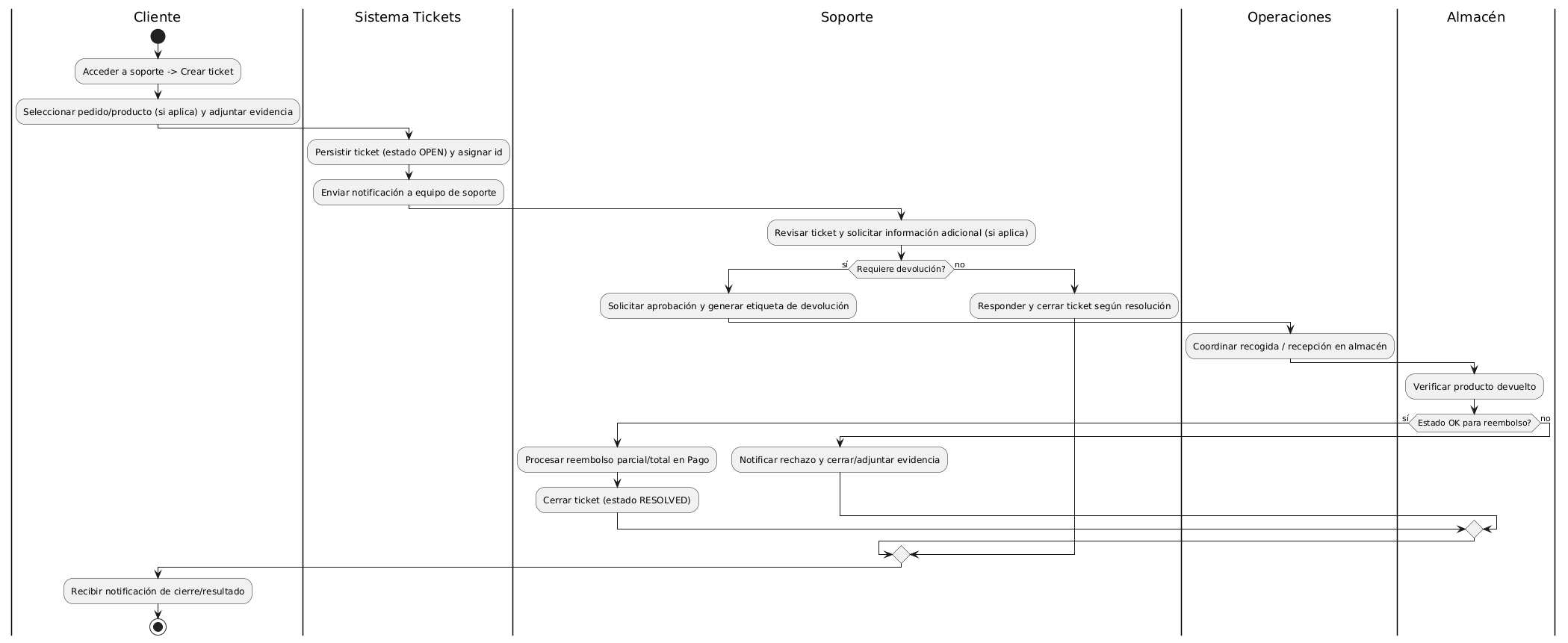
**UC-04 — Checkout y Procesamiento de Pago**



**UC-05 — Gestión de Pedidos y Fulfillment (Operaciones / Almacén)**



**UC-06 — Crear Ticket de Soporte / Gestión de Devoluciones**



**Plantillas Extendidas de Casos de Uso:**

# UC-01 — Registro de Usuario (Cliente / Vendedor)

**ID:** UC-01  
**Título:** Registrar nueva cuenta (Cliente / Vendedor)  
**Actores:** Usuario (Cliente o Vendedor), Sistema (Auth)  
**Prioridad:** Alta

**Descripción:** Permitir que una persona cree una cuenta como cliente o vendedor proporcionando datos básicos; validar y persistir la cuenta.

**Precondiciones:**

* El usuario no tiene sesión iniciada.
* El correo electrónico no debe existir ya en la base de datos.

**Postcondiciones:**

* Cuenta creada en estado PENDING (si requiere verificación) o ACTIVE.
* Envío de correo de confirmación (si aplica).
* Para vendedor: registro queda en REVISIÓN si requiere validación documental.

### Escenario principal (éxito)

1. Usuario abre formulario de registro.
2. Usuario selecciona tipo de cuenta (Cliente / Vendedor).
3. Usuario ingresa: nombre, correo, contraseña (y datos extra si vendedor: NIT, razón social, documentos).
4. Sistema valida formato de correo y fuerza de contraseña.
5. Sistema verifica unicidad de correo.
6. Sistema crea registro en BD con hash de contraseña.
7. Sistema envía correo de verificación (si aplica) y muestra mensaje de confirmación.
8. Usuario confirma correo (si aplica) → cuenta marcada ACTIVE o queda REVISIÓN.

### Flujos alternos / extensiones

* **A1 — Correo ya registrado:** Si el correo existe → mostrar alerta “Usuario ya registrado” y ofrecer recuperación de contraseña.
* **A2 — Validación débil de contraseña:** Si la contraseña no cumple reglas → indicar requisitos mínimos.
* **A3 — Vendedor con documentación incompleta:** Guardar solicitud en REVISIÓN y notificar al equipo comercial.

### Excepciones / errores

* Error en la base de datos → mostrar mensaje genérico “Error al crear cuenta, intente más tarde” y log.
* Fallo en envío de correo → crear cuenta pero marcar notificación pendiente; reintentar envío en background.

**Reglas de negocio:**

* Contraseña mínima 8 caracteres, con mayúscula, minúscula y número.
* Correos duplicados no permitidos.
* Usuarios vendedores pueden requerir aprobación manual.

**Requisitos no funcionales:**

* Respuesta < 2s en verificación de unicidad.
* Hash de contraseñas con algoritmo seguro (bcrypt/argon2).
* Registro disponible 24/7.

**Criterios de aceptación:**

* Usuario Cliente puede registrarse y acceder tras verificación (si aplica).
* Usuario Vendedor queda en estado correcto y recibe notificaciones.

# UC-02 — Publicar / Administrar Producto (Vendedor / Admin)

**ID:** UC-02  
**Título:** Crear, editar y eliminar producto  
**Actores:** Vendedor, Administrador, Sistema (Media/CDN)  
**Prioridad:** Alta

**Descripción:** Permitir al vendedor o admin gestionar productos: título, descripción, precio, imágenes, stock, categorías.

**Precondiciones:**

* Actor autenticado con permisos (vendedor o admin).
* Cuenta verificada (si política lo requiere).

**Postcondiciones:**

* Producto creado/actualizado/eliminado en la base de datos.
* Imágenes almacenadas en CDN y URL guardadas.

### Escenario principal (crear producto)

1. Usuario accede al panel y selecciona “Nuevo producto”.
2. Usuario completa campos obligatorios: título, precio, categoría, stock mínimo.
3. Usuario sube imágenes (mínimo 1).
4. Sistema valida campos y formatos de imágenes (tipo y tamaño).
5. Sistema guarda imágenes en CDN y persiste URLs junto con los datos del producto.
6. Producto queda PUBLISHED o DRAFT según elección.
7. Sistema muestra página del producto o lista de productos con mensaje de éxito.

### Flujos alternos / extensiones

* **A1 — Imagen inválida:** Rechazar upload con motivo (peso/tipo) y pedir reintento.
* **A2 — Campos incompletos:** Mostrar lista de campos obligatorios faltantes.
* **A3 — Edición masiva:** Permitir editar múltiples campos y aplicar cambios en lote (admin).

### Excepciones / errores

* Error al guardar imagen → rollback parcial y mensaje de error.
* Conflicto de SKU/ID único → informar y solicitar cambio.

**Reglas de negocio:**

* Precio >= 0.
* Stock entero no negativo.
* Máximo X imágenes por producto (ej. 8).

**Requisitos no funcionales:**

* Procesar uploads asincrónicamente y devolver respuesta rápida.
* Latencia de visualización < 500ms si imágenes están cacheadas.

**Criterios de aceptación:**

* Productos pueden ser creados con al menos una imagen y aparecer en catálogo.
* Vendedor sin permisos no puede crear productos para otros vendedores.

# UC-03 — Buscar, Filtrar y Añadir al Carrito

**ID:** UC-03  
**Título:** Navegar catálogo, filtrar resultados y añadir producto al carrito  
**Actores:** Cliente (no autenticado o autenticado), Sistema (Catalogo, Inventario)  
**Prioridad:** Alta

**Descripción:** Cliente busca productos, aplica filtros y añade ítems al carrito, con verificación básica de stock.

**Precondiciones:**

* Catálogo poblado con productos visibles.
* Inventario actualizado.

**Postcondiciones:**

* Item agregado al carrito del usuario (persistente si está autenticado).
* Disponible cantidad reservada temporalmente (opcional).

### Escenario principal (añadir al carrito)

1. Usuario busca por palabra/clasificación o navega por categoría.
2. Sistema muestra resultados con paginación y disponibilidad (stock).
3. Usuario selecciona producto y cantidad.
4. Usuario hace clic en “Añadir al carrito”.
5. Sistema verifica stock disponible.
6. Si stock suficiente, se crea/actualiza Carrito y se confirma al usuario.
7. Si usuario autenticado, el carrito se guarda en BD; si no, en sesión/localStorage.

### Flujos alternos / extensiones

* **A1 — Stock insuficiente:** Mostrar mensaje “Solo X disponibles” y ajustar cantidad.
* **A2 — Usuario anónimo → login requerido para checkout:** En checkout, pedir autenticación o invitado.

### Excepciones / errores

* Error de concurrencia al reservar stock → notificar al usuario e indicar stock real.
* Error en sesión → guardar en localStorage y ofrecer sincronizar al login.

**Reglas de negocio:**

* Reservas temporales pueden expirar (p. ej. 15 min).
* Productos con estado inactive no se muestran.

**Requisitos no funcionales:**

* Búsqueda debe responder en < 1s para UX aceptable.
* Paginación y cache para cargas elevadas.

**Criterios de aceptación:**

* Usuario puede añadir producto con stock disponible y verlo en carrito.
* Mensajes claros si stock cambia.

# UC-04 — Checkout y Procesamiento de Pago

**ID:** UC-04  
**Título:** Checkout y pago de pedido  
**Actores:** Cliente, Sistema de Pagos (Pasarela), Inventario, Notificaciones  
**Prioridad:** Crítica

**Descripción:** Completar compra: recopilar dirección, calcular envío, procesar pago y generar pedido.

**Precondiciones:**

* Carrito con al menos un item.
* Usuario autenticado o checkout como invitado permitido.
* Integración con pasarela en modo sandbox o real.

**Postcondiciones:**

* Pedido creado con estado CONFIRMED (si pago exitoso) o PENDING/CANCELED según resultado.
* Stock reservado/decrementado según resultado.
* Notificación enviada al cliente.

### Escenario principal (pago exitoso)

1. Usuario revisa carrito y procede a checkout.
2. Usuario ingresa o confirma dirección de envío y contacto.
3. Sistema calcula costos de envío y total.
4. Usuario selecciona método de pago.
5. Sistema inicia transacción con la pasarela (crear token de pago si aplica).
6. Pasarela devuelve APPROVED.
7. Sistema crea Pedido con estado CONFIRMED.
8. Sistema llama a Inventario.reservar() y luego a Inventario.actualizarStock() (consistente).
9. Sistema genera número de pedido, envía confirmación por email/SMS.
10. Sistema notifica a Operaciones/Almacén.

### Flujos alternos / extensiones

* **A1 — Pago en espera (3DS / OTP):** Marcar pedido PENDING hasta confirmación adicional.
* **A2 — Pago rechazado:** Mostrar motivo y ofrecer reintentar o cambiar método.
* **A3 — Envío no calculable:** Ofrecer contacto manual o selección de punto de recogida.

### Excepciones / errores

* Pago rechazado por fraude → bloquear y marcar para revisión manual.
* Fallo de red durante confirmación → marcar pedido PENDING y programar reintento.
* Error al decrementar stock → iniciar proceso de compensación (rollback o notificación a soporte).

**Reglas de negocio:**

* No confirmar pedido sin confirmar reserva de stock (o usar patrón saga).
* Política de reintentos: 3 intentos en transacciones transitorias.

**Requisitos no funcionales:**

* Tiempo máximo para procesar pago: < 10s (UX).
* Logs de transacciones deben conservarse 6 meses (compliance).

**Criterios de aceptación:**

* Flujo end-to-end: Cliente puede pagar y recibir confirmación.
* Escenarios de rechazo manejados y pedidos no duplicados.

# UC-05 — Gestión de Pedidos y Fulfillment (Operaciones / Almacén)

**ID:** UC-05  
**Título:** Procesamiento y despacho de pedido  
**Actores:** Operaciones, Almacén, Sistema de Pedidos, Sistema de Envíos  
**Prioridad:** Alta

**Descripción:** Transformar pedidos confirmados en envíos: picking, empaquetado, generación de guía y actualización de estado.

**Precondiciones:**

* Pedido en estado CONFIRMED con stock reservado.
* Disponibilidad de recursos logísticos.

**Postcondiciones:**

* Pedido marcado como LISTO\_PARA\_DESPACHO / ENVIADO.
* Envío creado con tracking number (si aplica).
* Inventario actualizado definitivamente.

### Escenario principal (despacho)

1. Sistema lista pedidos CONFIRMED en dashboard de Operaciones.
2. Operaciones crea orden de picking y asigna a un operador.
3. Operador escanea items y confirma packing.
4. Sistema solicita generación de guía a proveedor de envíos o imprime etiqueta interna.
5. Sistema cambia estado del pedido a ENVIADO y guarda trackingNumber.
6. Sistema notifica al cliente con información de envío.

### Flujos alternos / extensiones

* **A1 — Item faltante en picking:** Marcar pedido parcial, notificar al cliente y ofrecer reembolso o envío parcial.
* **A2 — Integración con tercero:** Enviar solicitud y esperar confirmación de la agencia.

### Excepciones / errores

* Error de integración con agencia → marcar incidencia y reintentar.
* Discrepancia de inventario → crear ticket para auditoría.

**Reglas de negocio:**

* Pedidos con parcelas separadas se manejan como sub-pedidos o envíos múltiples.
* SLA de despacho: X horas/días desde confirmación (según prioridad).

**Requisitos no funcionales:**

* Dashboard con refresh en tiempo real para operadores.
* API para integración con transportistas con timeout y reintentos.

**Criterios de aceptación:**

* Pedido cambia su estado correctamente y el cliente recibe tracking.
* Incidencias registradas y visibles para soporte.

# UC-06 — Crear Ticket de Soporte / Gestión de Devoluciones

**ID:** UC-06  
**Título:** Generar ticket de soporte y gestionar devolución/reclamo  
**Actores:** Cliente, Soporte, Sistema de Tickets, Operaciones  
**Prioridad:** Media-Alta

**Descripción:** Cliente genera un ticket relacionado con un pedido o producto; soporte gestiona el caso y coordina devolución o reembolso.

**Precondiciones:**

* Cliente autenticado (o proporcionar datos mínimos para ticket).
* Pedido relacionado existe (si procede).

**Postcondiciones:**

* Ticket creado y asignado a agente.
* Si procede, se inicia proceso de devolución y ajuste de inventario/pago.

### Escenario principal

1. Cliente accede a “Mis pedidos” y selecciona “Reportar problema” o accede a formulario de soporte.
2. Cliente selecciona pedido/producto y describe el problema (motivo, fotos).
3. Sistema crea ticket con estado OPEN y referencia al pedido.
4. Sistema notifica al equipo de soporte.
5. Agente asigna estado (e.g., INVESTIGATING), solicita evidencias si es necesario.
6. Si se aprueba devolución, sistema genera etiqueta de retorno y crea tarea en Operaciones.
7. Tras recepción y verificación, sistema procesa reembolso parcial/total y cierra ticket.

### Flujos alternos / extensiones

* **A1 — Ticket sin relación a pedido:** Se crea ticket general y se asigna área correspondiente.
* **A2 — Reclamación por producto defectuoso:** Envío de instrucciones y reembolso acelerado.

### Excepciones / errores

* Documentos faltantes → ticket en PENDING hasta recibir pruebas.
* Fraude detectado → escalación a equipo de riesgo.

**Reglas de negocio:**

* Plazo para devoluciones: X días desde entrega.
* Reembolso sujeto a verificación del estado del producto.

**Requisitos no funcionales:**

* Adjuntos aceptados hasta X MB/archivo.
* SLA de respuesta inicial de soporte: < 24h (o según acuerdo).

**Criterios de aceptación:**

* Ticket se crea y es visible en el panel de soporte.
* Flujo de devolución ejecutable y reembolso reflejado en pedido/pago.

**Creación del Modelo de Dominio con Clases Abstractas:**

# 1) Visión general

Este es un **modelo de dominio** para el e-commerce Zeyra. Usa clases abstractas para definir plantillas (Cuenta, Producto, Pago, Envío) y luego clases concretas que heredan de ellas (Cliente, Vendedor, ProductoSimple, PagoTarjeta, etc.). Además incluye las entidades operativas (Carrito, Pedido, Inventario, Notificación, SoporteTicket, etc.) y las relaciones entre ellas.

# 2) Clases abstractas y jerarquía (qué significan)

* **Cuenta (abstract)**: plantilla para cualquier tipo de cuenta en el sistema. Define atributos comunes (id, email, passwordHash, fecha creación) y comportamientos básicos (login(), resetPassword()).
  + Heredan: Administrador, Vendedor, Cliente.  
    → Implicación: todas las cuentas comparten autenticación/seguridad; roles añaden comportamiento específico (p. ej. Vendedor publica productos).
* **Producto (abstract)**: atributos comunes a cualquier producto (id, titulo, descripcion, precio, stock, imagenes) y método getPrecio().
  + Heredan: ProductoSimple, ProductoPack (p. ej. packs compuestos).
* **Pago (abstract)**: modelo de pago con monto, fecha, estado y procesar(); concreto: PagoTarjeta, PagoPasarela.
* **Envio (abstract)**: estructura para envíos; concreto: EnvioInterno, EnvioTerceros.

# 3) Entidades operativas principales

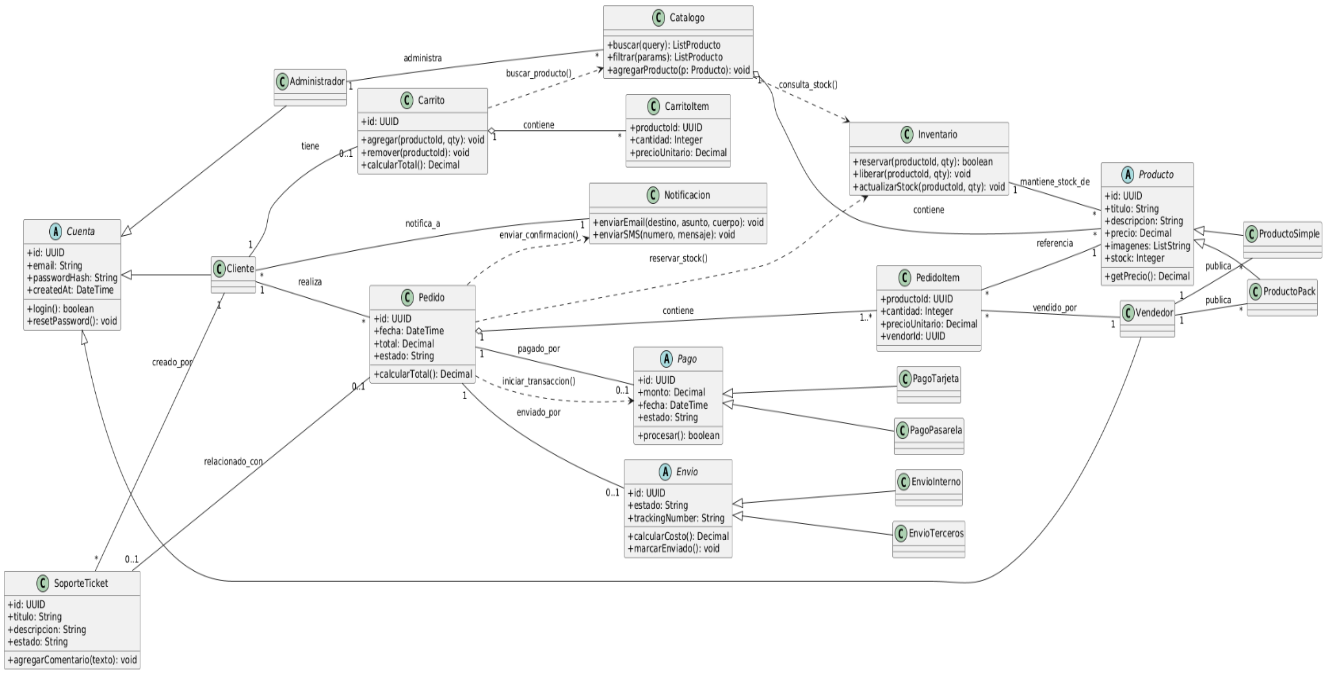
* **Catalogo**: expone operaciones buscar(), filtrar(), agregarProducto(). Agrega/contiene productos (agregación).
* **Carrito** y **CarritoItem**: carrito pertenece a un Cliente (0..1) y contiene CarritoItem (composición → si se destruye carrito, sus items desaparecen).
* **Pedido** y **PedidoItem**: cuando un cliente convierte el carrito, se genera un Pedido que contiene PedidoItem (cada item referencia a un Producto y registra el vendorId).
* **Inventario**: mantiene stock de productos; ofrece operaciones reservar, liberar, actualizarStock.
* **SoporteTicket**: tickets generados por clientes; opcionalmente relacionados con un Pedido.
* **Notificacion**: servicio para enviar emails/SMS.

# 4) Relaciones y multiplicidades clave (interpretación)

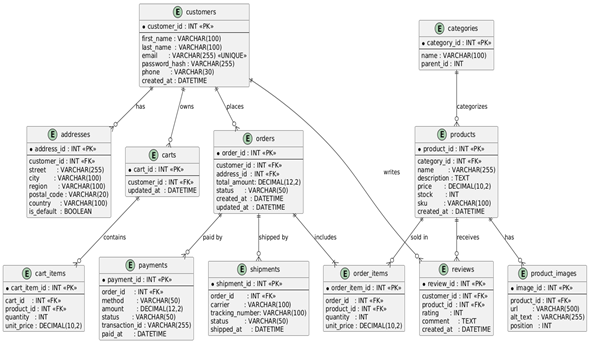
* Vendedor "1" -- "\*" ProductoSimple : publica  
  Un vendedor puede publicar muchos productos; cada producto es publicado por un vendedor (1:N).
* Catalogo "1" o-- "\*" Producto: contiene  
  Agregación: el catálogo referencia productos; los productos pueden existir independientemente (no es composición estricta).
* Cliente "1" -- "0.1" Carrito: tiene  
  Un cliente puede tener 0 o 1 carrito activo. (0 si no ha añadido nada / aún no sesión).
* Carrito "1" o-- "\*" CarritoItem: contiene  
  Composición: items pertenecen al carrito.
* Cliente "1" -- "\*" Pedido: realiza  
  Un cliente puede hacer muchos pedidos (1: N).
* Pedido "1" o-- "1. \*" PedidoItem: contiene  
  Pedido contiene uno o más items — definición de la orden.
* Pedido "1" -- "0.1" Pago: pagado\_por  
  Un pedido puede tener 0 o 1 pago (0 antes de pagar; 1 cuando se completa).
* Pedido "1" -- "0.1" Envio: enviado\_por  
  El envío puede ser creado después de la confirmación y no todos los pedidos están enviados inmediatamente.
* Inventario "1" -- "\*" Producto: mantiene\_stock\_de  
  Inventario administra stocks para múltiples productos.
* SoporteTicket "\*" -- "1" Cliente: creado\_por  
  Muchos tickets pueden ser creados por un cliente.
* Dependencias (.>):
  + Catálogo.> Inventario: consulta stock () — Catalogo necesita consultar inventario para mostrar disponibilidad.
  + Pedido.> Inventario: reservar\_stock () — durante la creación/confirmación de pedido se reserva stock.
  + Pedido.> Pago: iniciar transacción () — flujo del pedido invoca la lógica de pago.
  + Pedido.> Notificacion: enviar confirmación () — al confirmarse, se envía notificación.
  + Carrito.> Catalogo: buscar producto () — el carrito puede validar producto/obtener detalles.

# 5) Flujo típico (ejemplo: checkout)

1. Cliente añade items al Carrito (CarritoItem referenciando productos).
2. Cliente inicia checkout → se crea Pedido con PedidoItem por cada item del carrito.
3. Pedido llama a Inventario.reservar(productoId, qty) para evitar overselling.
4. Pedido inicia la transacción con Pago (crea PagoTarjeta o PagoPasarela según método).
   * Si Pago.procesar() devuelve true → estado del pedido pasa a confirmado.
   * Si falla → pedido queda pendiente o cancelado.
5. Pedido invoca Notificacion.enviarEmail(...) para confirmar al cliente.
6. Si aplica, crear Envio (interno o con terceros) y marcar trackingNumber.
7. Operaciones/Logística procesan fulfillment y actualizan Inventario.actualizarStock() definitivo.



MER:



Formato ajustado según normas APA 7ª edición: interlineado 2.0, títulos en negrita, referencias en orden alfabético, y enlaces formateados correctamente.