**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.
* RF-07: Pasarela de pagos

**Funcionalidades para Administrador**

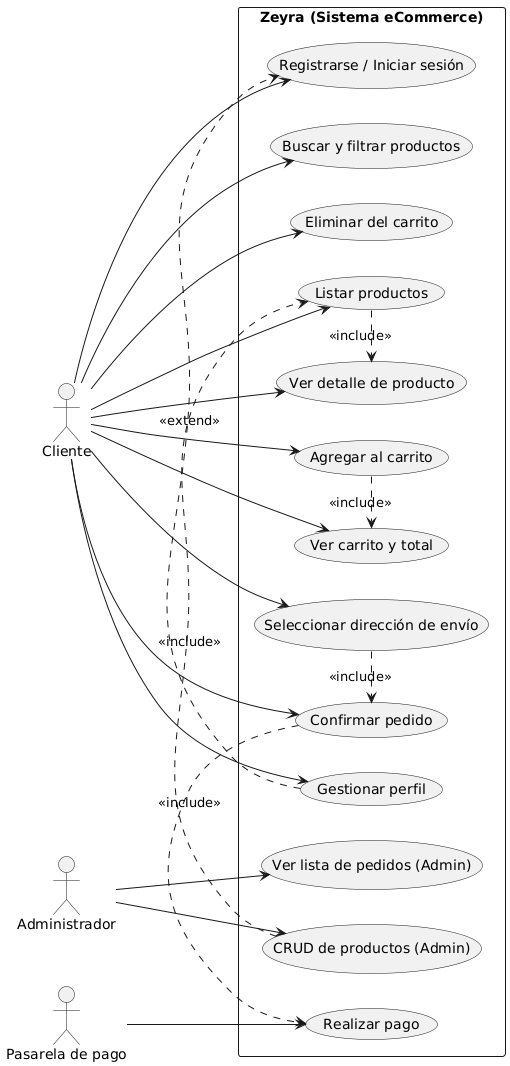
* RF-03: Perfil Admin.
* RF-05: Vistas (Dashboard).

**Mapeo Conceptual de Funciones**

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

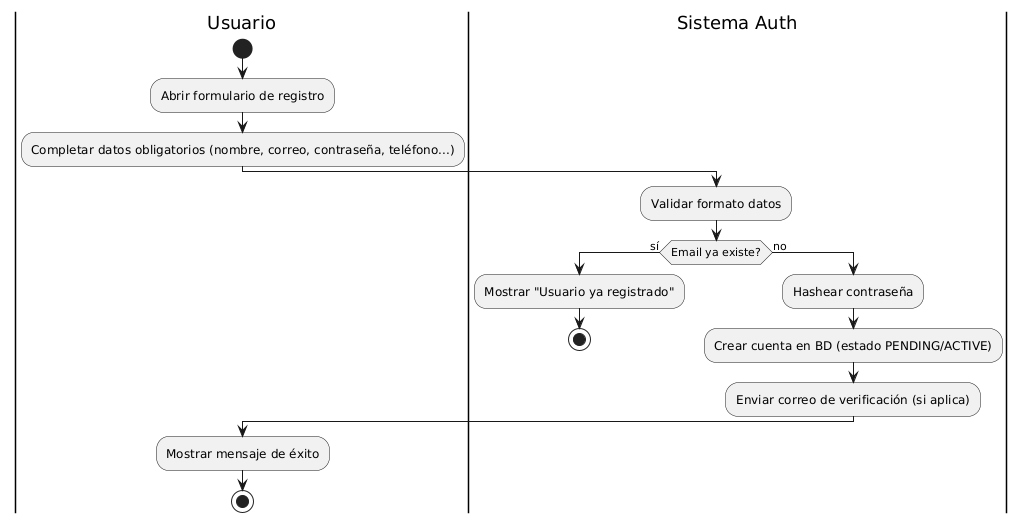
|  |  |  |
| --- | --- | --- |
| **Actor** | **Requisitos Asociados** | **Funciones del Software** |
| **Usuario** | RF-01, RF-02,  RF-04, RF-06, RF-07 | Inicio sesión · Registrarse · Perfil Usuario · Productos · Pasarela de pagos |
| **Administrador** | RF-03, RF-06 | Perfil Admin · Vistas (Dashboard) |

**Diagrama de casos de uso:**

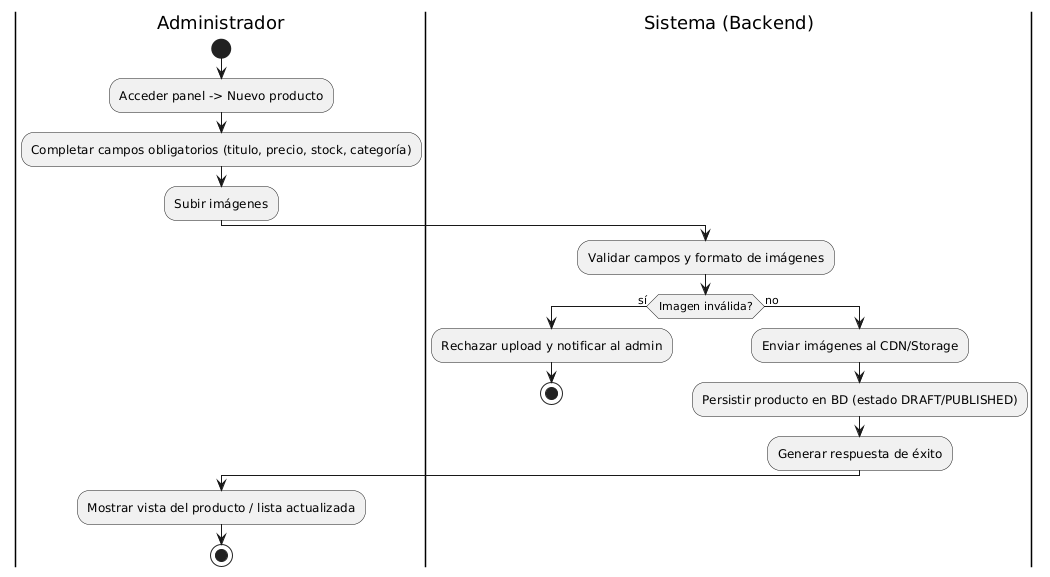


**Diagramas de Actividades de casos de uso:**

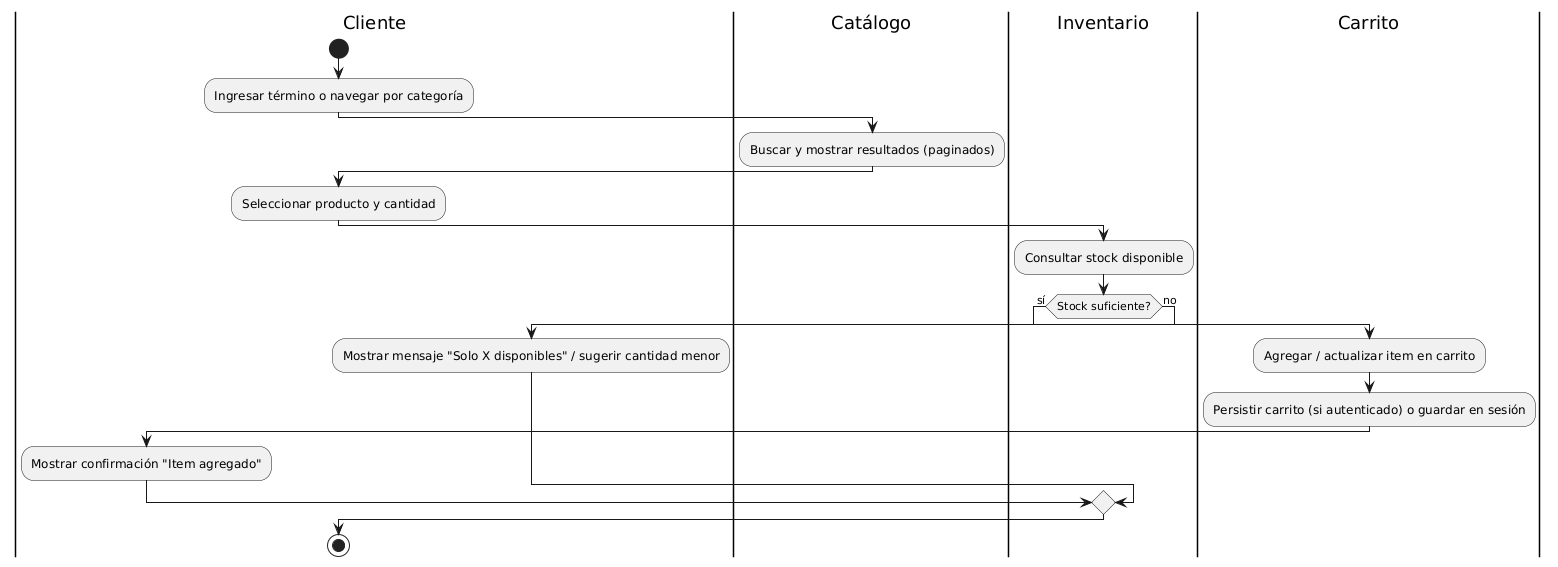
**UC-01 — Registro de Usuario**



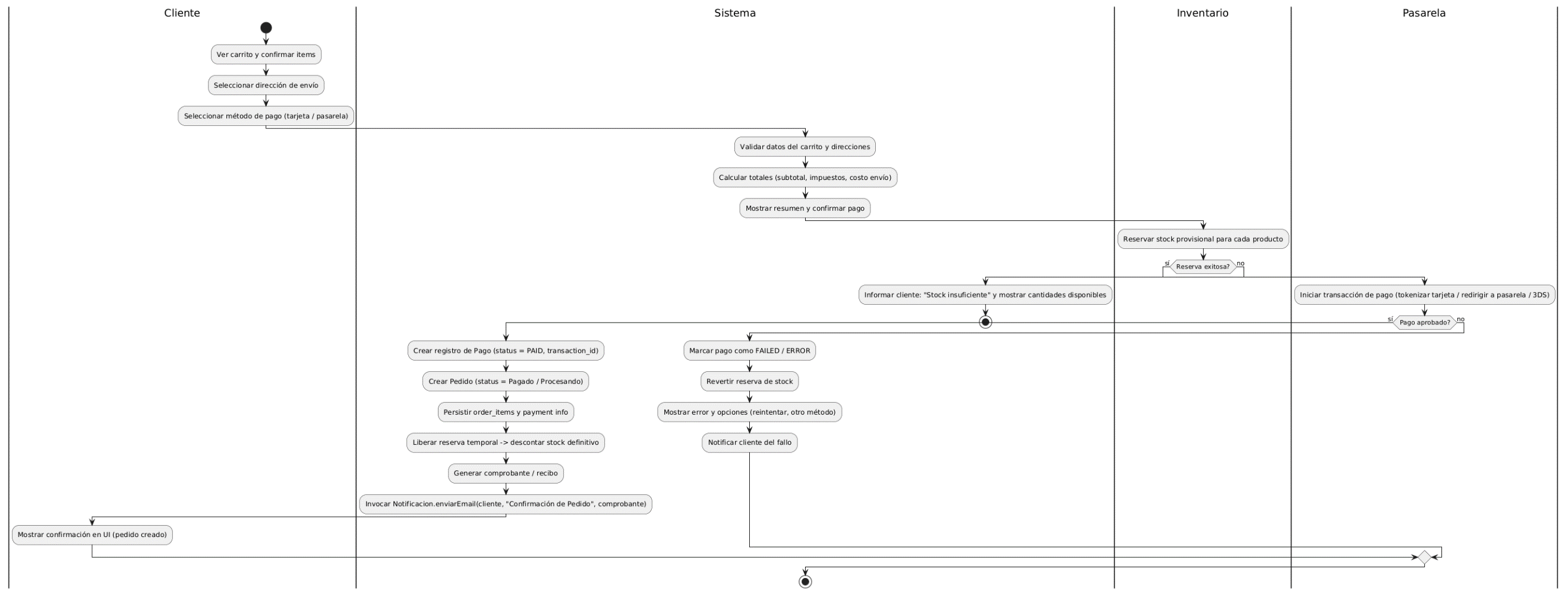
**UC-02 — Publicar - Administrar Producto**



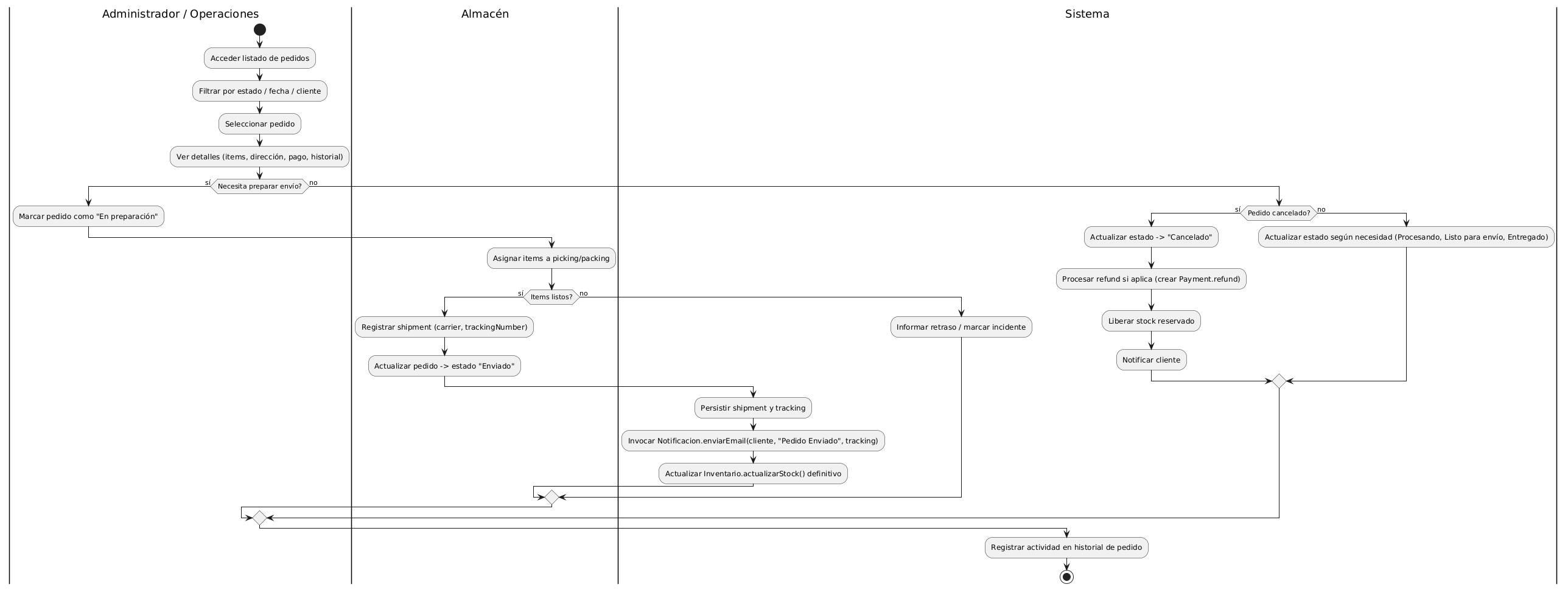
**UC-03 — Buscar, Filtrar y Añadir al Carrito**



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



**UC-05 — Gestión de Pedidos y Fulfillment**



**Plantillas Extendidas de Casos de Uso:**

**UC-01 — Registrar nueva cuenta (Cliente)**

* **ID:** UC-01
* **Título:** Registrar nueva cuenta (Cliente)
* **Actores:** Usuario (Cliente), Sistema de Autenticación (Auth)
* **Prioridad:** Alta
* **Descripción:** Permitir que una persona cree una cuenta de cliente proporcionando datos básicos; validar la información, crear el registro y (opcionalmente) enviar verificación por correo.
* **Precondiciones:** El usuario no tiene sesión iniciada; el correo electrónico introducido no debe existir en la base de datos.
* **Postcondiciones:** Se crea la cuenta en la BD en estado PENDING o ACTIVE según configuración; si aplica, se envía correo de verificación.
* **Escenario principal:**
  1. Usuario abre formulario de registro.
  2. Usuario completa campos obligatorios (nombre, correo, contraseña, teléfono opcional, dirección opcional).
  3. Sistema valida formato de los campos (email, contraseña, teléfono).
  4. Sistema verifica unicidad del correo en la BD.
  5. Sistema hashea la contraseña y crea el registro de usuario.
  6. Sistema devuelve confirmación y, si aplica, envía correo de verificación.
  7. Usuario recibe mensaje de éxito; puede iniciar sesión tras verificar (si aplica).
* **Flujos alternos / Excepciones:**
  1. A1: Si el correo ya existe → sistema muestra “Usuario ya registrado” y sugiere recuperación de contraseña o iniciar sesión.
  2. A2: Si la contraseña no cumple requisitos → sistema muestra requisitos mínimos (e.g., longitud, complejidad).
  3. A3: Error en envío de correo de verificación → el usuario queda con cuenta creada; se registra el error en logs y se permite reenvío manual.
* **Reglas de negocio:**
  1. Contraseña mínima: 8 caracteres.
  2. Correos duplicados no permitidos (UNIQUE(email)).
  3. Verificación por correo configurable (ON/OFF).
* **Criterio de aceptación:** Un usuario nuevo puede registrarse; si la verificación está activada, el correo se envía y la cuenta no permite acceso hasta verificación.

**UC-02 — Crear, editar y eliminar producto (Administrador)**

* **ID:** UC-02
* **Título:** Crear / Editar / Eliminar Producto (Administrador)
* **Actores:** Administrador, Servicio de Almacenamiento/Media (CDN)
* **Prioridad:** Alta
* **Descripción:** El administrador gestiona el catálogo de productos: crear, actualizar y eliminar productos con sus atributos (título, descripción, precio, stock, SKU, categoría, imágenes).
* **Precondiciones:** Actor autenticado y autorizado como administrador.
* **Postcondiciones:** Producto persistido/modificado/eliminado en BD; imágenes almacenadas en CDN; catálogo actualizado.
* **Escenario principal (Crear producto):**
  1. Admin accede al panel y elige “Nuevo producto”.
  2. Admin completa campos obligatorios (nombre, precio, stock, categoría) y opcionales (descripción, SKU, atributos adicionales).
  3. Admin sube imágenes; sistema valida formatos y tamaño.
  4. Sistema envía imágenes al CDN y obtiene URLs.
  5. Sistema valida datos (no nulos, precios >= 0) y persiste product con product\_images.
  6. Sistema muestra confirmación y el producto aparece en el catálogo (si publicado).
* **Flujos alternos / Excepciones:**
  1. A1: Imagen inválida → sistema rechaza upload, informa al admin y no persiste hasta corrección.
  2. A2: SKU duplicado → sistema alerta y obliga a corregir.
  3. A3: Error al persistir en BD → sistema registra error y notifica al admin.
* **Reglas de negocio:**
  1. Precio >= 0; stock >= 0.
  2. Al menos 1 imagen recomendada (puede permitirse 0 pero con placeholder en UI).
  3. Categoría válida y existente.
* **Criterio de aceptación:** Admin puede crear/editar/eliminar un producto y éste aparece correctamente listado en el catálogo con imágenes accesibles.

**UC-03 — Buscar, filtrar y añadir al carrito (Cliente)**

* **ID:** UC-03
* **Título:** Buscar / Filtrar / Añadir al Carrito (Cliente)
* **Actores:** Cliente, Catálogo, Inventario
* **Prioridad:** Alta
* **Descripción:** Cliente explora el catálogo mediante búsqueda y filtros, selecciona productos y los añade al carrito; el sistema verifica stock y persiste el carrito.
* **Precondiciones:** Productos disponibles en catálogo; si el cliente está autenticado, existe un carrito asociado o se crea uno en sesión.
* **Postcondiciones:** Ítems añadidos al carrito; carrito persistido (si cliente autenticado) o guardado en sesión.
* **Escenario principal:**
  1. Cliente usa barra de búsqueda o filtros por categoría/precio.
  2. Sistema consulta catálogo e inventario para mostrar resultados paginados con disponibilidad.
  3. Cliente selecciona producto y cantidad.
  4. Sistema consulta Inventario para verificar stock disponible.
  5. Si stock suficiente, sistema agrega CartItem y actualiza total; si autenticado, persiste carrito en BD.
  6. Sistema muestra confirmación “Item agregado”.
* **Flujos alternos / Excepciones:**
  1. A1: Stock insuficiente → sistema muestra mensaje y sugiere ajustar cantidad o notificar cuando esté disponible.
  2. A2: Producto despublicado durante navegación → sistema informa y remueve producto del listado.
* **Reglas de negocio:**
  1. No permitir añadir más cantidad que el stock disponible.
  2. Si usuario no autenticado, el carrito se mantiene en sesión y se migra a cuenta al iniciar sesión.
* **Criterio de aceptación:** El cliente puede buscar/filtrar productos y añadir ítems al carrito; las verificaciones de stock funcionan y el total se actualiza correctamente.

**UC-04 — Checkout y Procesamiento de Pago (Cliente → Pasarela)**

* **ID:** UC-04
* **Título:** Checkout y Procesamiento de Pago
* **Actores:** Cliente, Sistema (Checkout), Inventario, Pasarela de pago (externa), Notificaciones
* **Prioridad:** Crítica
* **Descripción:** Flujo de finalización de compra: desde la revisión del carrito hasta la confirmación del pago y creación del pedido. Incluye validación de stock, reserva temporal, integración con pasarela, persistencia de Order y Payment, y notificaciones al cliente.
* **Precondiciones:** Cliente con ítems en el carrito; dirección de envío válida; método de pago disponible (tarjeta o pasarela).
* **Postcondiciones:** Si pago exitoso: se crea Order con estado Pagado, Payment registrado (con transaction\_id) y stock actualizado; cliente recibe confirmación. Si falla: orden no creada o queda en estado Pendiente/Cancelado, reserva de stock liberada y se notifica al cliente.
* **Escenario principal (pasos):**
  1. Cliente accede a “Checkout” y revisa resumen (items, subtotal, envío, impuestos, total).
  2. Cliente selecciona o ingresa dirección de envío y método de envío.
  3. Cliente elige método de pago (tarjeta nueva, tarjeta guardada o pasarela externa).
  4. Sistema valida disponibilidad de stock para cada CartItem.
  5. Si stock suficiente, sistema llama a Inventario.reservar(productoId, qty) para reservar temporalmente.
  6. Sistema calcula total final (incluye envío/impuestos) y prepara la transacción.
  7. Sistema inicia transacción con la pasarela (tokenización, 3DS si aplica) o procesa pago interno; recibe respuesta con transaction\_id y status.
  8. Si la pasarela confirma PAID:
     + Sistema crea Order con PedidoItems (copiando precio unitario), marca estado Pagado.
     + Crea registro Payment (method, amount, transaction\_id, paid\_at).
     + Llama a Inventario.actualizarStock() para decrementar stock definitivo.
     + Crea Shipment si corresponde y desencadena Notificacion.enviarConfirmacion().
  9. Si la pasarela devuelve fallo (FAILED, DECLINED):
     + Sistema registra Payment con estado FAILED y libera reservas (Inventario.liberar()).
     + Cliente recibe mensaje de error y opciones (reintentar, cambiar método, cancelar).
* **Flujos alternos / Excepciones:**
  1. A1: Durante reserva se detecta falta de stock → mostrar error y permitir ajuste de cantidades.
  2. A2: Error de comunicación con pasarela → registrar y mostrar mensaje “Error en procesar pago, intenta más tarde”. Permitir reintento o cancelar.
  3. A3: Pago pendiente (p.ej. revisión manual) → Order creado con estado Pendiente; se notifica a operaciones para revisión.
* **Reglas de negocio:**
  1. No confirmar Order como Pagado sin transaction\_id válido proveniente de la pasarela.
  2. Reservas temporales deben expirar tras X minutos si no se completa la transacción.
  3. Los reembolsos se procesan mediante Payment.refund() y deben quedar registrados con referencia a order\_id.
* **Criterio de aceptación:** Al completar un pago exitoso: existe un Order en BD con estado Pagado, un Payment con transaction\_id y amount correcto, stock decrecido, y el cliente recibe confirmación por email/in-app.

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

* **ID:** UC-05
* **Título:** Gestión de Pedidos y Fulfillment (Operaciones / Almacén)
* **Actores:** Administrador (Operaciones / Almacén), Sistema de Pedidos, Notificaciones, Inventario, Servicio de Envíos (Shipment)
* **Prioridad:** Alta
* **Descripción:** Permite a personal autorizado gestionar pedidos: revisar detalles, actualizar estados, registrar envíos (tracking), procesar devoluciones/reembolsos y notificar al cliente. Incluye validaciones de transición de estados y sincronización con inventario y pagos.
* **Precondiciones:** El pedido existe en la BD; el actor está autenticado y autorizado como Administrador/Operaciones.
* **Postcondiciones:** Estado del pedido actualizado; si aplica, Shipment creado/actualizado con trackingNumber; en reembolsos Payment actualizado y stock ajustado. Cliente notificado de cambios.
* **Escenario principal:**
  1. Admin accede al panel de pedidos y busca/filtra por estado, fecha, cliente o id.
  2. Admin selecciona un pedido y visualiza detalle (items, direcciones, pago, historial de estados).
  3. Si procede, Admin actualiza el estado (p. ej. Preparando → Enviado).
  4. Sistema valida transición de estado según reglas (p. ej. no pasar de Cancelado a Enviado).
  5. Admin puede registrar envío: ingresa carrier y tracking\_number.
  6. Sistema crea/actualiza Shipment con shipped\_at y cambia estado del pedido si aplica.
  7. Sistema envía notificación al cliente con el nuevo estado y/o tracking.
* **Flujos alternos / Excepciones (devoluciones / reembolsos):**
  1. A1: Procesar devolución → Admin registra devolución, sistema inicia Payment.refund() (si aplica) y crea registro de reembolso; sistema actualiza estado del pedido (Refunded o similar) y restaura stock si la devolución lo exige (Inventario.actualizarStock()). Notificación enviada al cliente.
  2. A2: Error al solicitar reembolso a la pasarela → sistema registra el fallo y deja el pedido en estado Reembolso Pendiente; notifica al equipo financiero.
  3. A3: Intento inválido de transición de estado → sistema rechaza la actualización y muestra regla de negocio explicando por qué.
* **Reglas de negocio:**
  1. Transiciones de estado deben cumplir una máquina de estados predefinida (p.ej. Pagado → Preparando → Enviado → Entregado).
  2. Solo admins con permiso fulfillment pueden marcar como Enviado y registrar tracking.
  3. Reembolsos solo proceden si el Payment original lo permite; debe quedar referencia de refund\_transaction\_id.
* **Criterio de aceptación:** Admin puede actualizar estados válidos, registrar Shipment con trackingNumber, procesar reembolsos cuando proceda y todas las acciones quedan registradas en el historial del pedido; el cliente recibe notificaciones coherentes.

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

1. **Visión general**  
   Este es el modelo de dominio para el e-commerce Zeyra. Se utilizan clases abstractas para definir plantillas generales (por ejemplo, Cuenta, Producto, Pago, Envio) y luego clases concretas que heredan de ellas (por ejemplo, Cliente, Administrador, ProductoConcrete, PagoTarjeta, PagoPasarela). Además incluye las entidades operativas necesarias para la operación del comercio (Carrito, Pedido, Inventario, Notificación, Shipment) y las relaciones entre ellas. Se han eliminado deliberadamente las entidades/roles no deseados.
2. **Clases abstractas y jerarquía (significado y alcance)**

* **Cuenta (abstract)**: plantilla para cualquier cuenta del sistema. Define atributos comunes (id, email, passwordHash, createdAt) y comportamientos básicos (login(), resetPassword()).
  + **Heredan:** Administrador, Cliente.
  + **Implicación:** todas las cuentas comparten autenticación y seguridad; los roles definen permisos y acciones específicas (el Administrador asume todas las funciones de gestión que antes podían atribuirse al vendedor).
* **Producto (abstract)**: define atributos comunes a todos los productos (id, titulo, descripcion, precio, stock, imagenes) y métodos asociados (getPrecio()).
  + **Heredan:** ProductoConcrete (implementación concreta del producto único del sistema).
  + **Implicación:** no se gestiona una jerarquía de variantes complejas; en la capa de dominio se trabaja con una única entidad Producto que contiene la información necesaria para listado, carrito y pedidos.
* **Pago (abstract)**: modelo base para pagos (id, monto, fecha, estado, procesar()), con implementaciones concretas según el método.
  + **Heredan:** PagoTarjeta, PagoPasarela.
  + **Implicación:** el flujo de checkout interactúa con instancias de Pago que encapsulan la integración con pasarelas externas o métodos directos.
* **Envio (abstract)**: plantilla para envíos (id, estado, trackingNumber, calcularCosto(), marcarEnviado()), con extensiones concretas como EnvioInterno o EnvioTerceros según el proveedor/logística elegida.

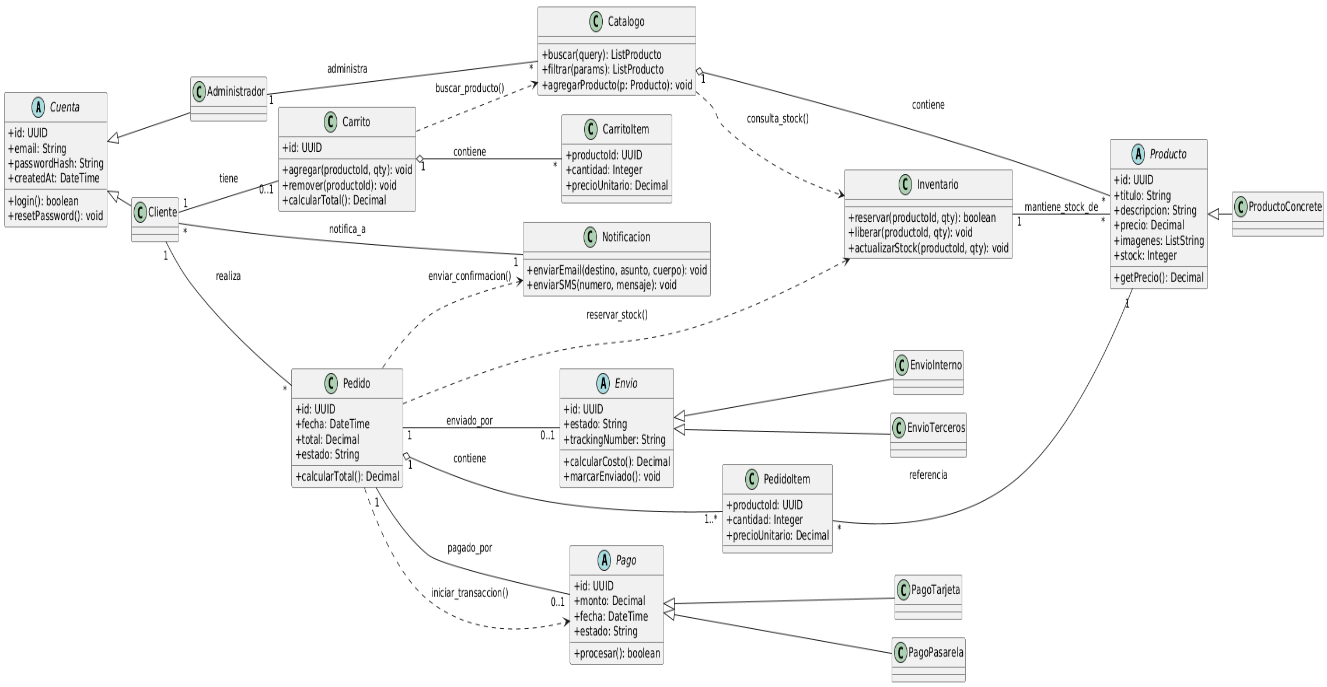
1. **Entidades operativas principales**

* **Catalogo:** expone operaciones buscar(), filtrar(), agregarProducto(). Representa una agregación de productos; el catálogo referencia productos que pueden existir independientemente.
* **Carrito / CarritoItem:** el Carrito pertenece a un Cliente (0..1) y contiene CarritoItem en composición (si se destruye el carrito, sus items desaparecen). Carrito provee agregar(), remover(), calcularTotal().
* **Pedido / PedidoItem:** al convertir un carrito en pedido se crea un Pedido que contiene PedidoItem. Cada PedidoItem referencia un Producto y registra cantidad y precio unitario al momento de la compra. **No se registra vendorId.**
* **Inventario:** mantiene el stock de productos y provee operaciones clave: reservar(productoId, qty), liberar(productoId, qty), actualizarStock(productoId, qty).
* **Notificacion:** servicio para enviar comunicaciones (email, in-app, SMS) en eventos relevantes (confirmación de pedido, cambios de estado, envío, reembolsos).
* **Shipment (Envio / Fulfillment):** entidad que contiene trackingNumber, carrier, shipped\_at y estado del envío; vinculada a Pedido cuando corresponda.

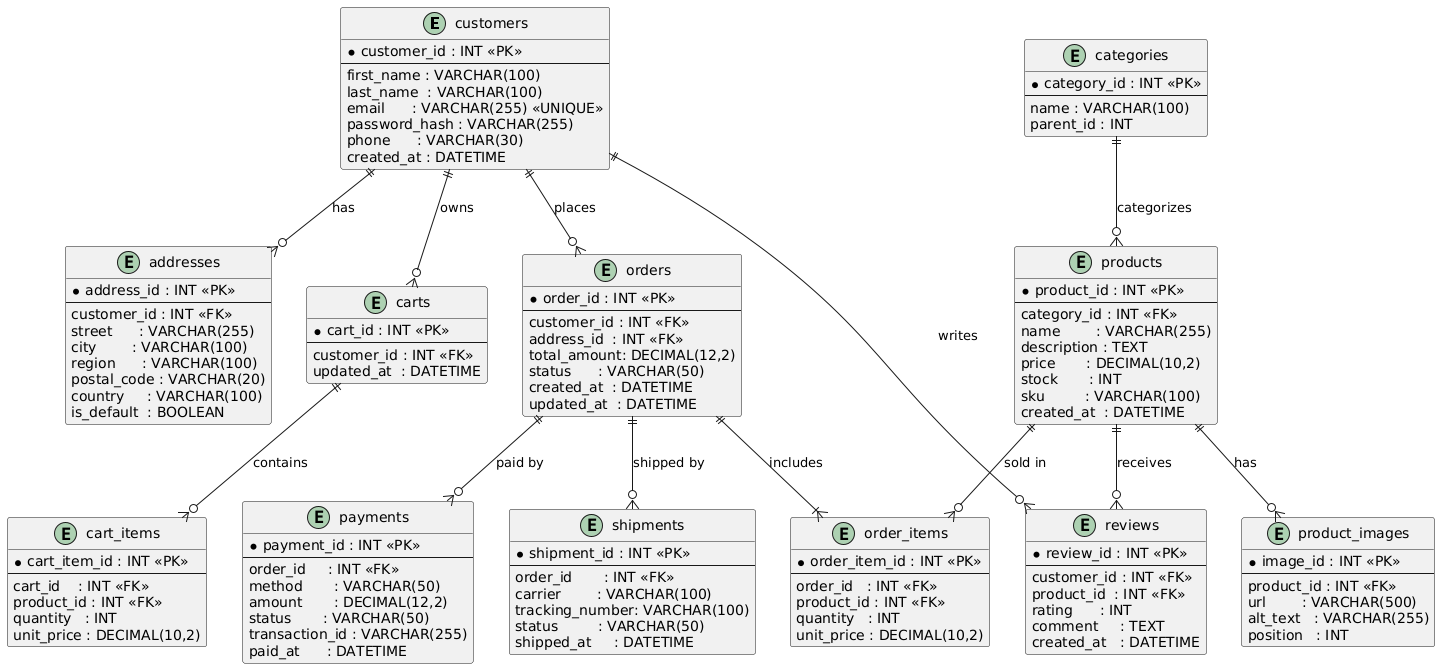
1. **Relaciones y multiplicidades clave (interpretación)**

* Catalogo "1" o-- "\*" Producto : agregación; el catálogo referencia múltiples productos.
* Cliente "1" -- "0..1" Carrito : un cliente puede tener 0 o 1 carrito activo.
* Carrito "1" o-- "\*" CarritoItem : composición; los items son propiedad del carrito.
* Cliente "1" -- "\*" Pedido : un cliente puede realizar varios pedidos.
* Pedido "1" o-- "1..\*" PedidoItem : cada pedido contiene uno o más items.
* Pedido "1" -- "0..1" Payment : un pedido puede tener 0 o 1 pago registrado (0 cuando aún no se ha pagado).
* Pedido "1" -- "0..1" Shipment : el envío se registra si aplica, con su trackingNumber.
* Inventario "1" -- "\*" Producto : el inventario gestiona el stock de múltiples productos.
* **Dependencias (comportamiento / métodos)**:
  + Catalogo .> Inventario : consultaStock() — para mostrar disponibilidad.
  + Pedido .> Inventario : reservar\_stock() — reserva durante checkout para evitar overselling.
  + Pedido .> Pago : iniciar\_transaccion() — el flujo de pedido invoca la lógica de pago.
  + Pedido .> Notificacion : enviar\_confirmacion() — cuando se confirma el pago/crea el pedido.
  + Carrito .> Catalogo : buscar\_producto() — validaciones de producto desde el carrito.

1. **Flujo típico (ejemplo: checkout)**
2. El cliente añade items al Carrito (cada CarritoItem referencia un Producto).
3. El cliente inicia el checkout → se crea un Pedido con PedidoItem por cada item del carrito.
4. El Pedido invoca Inventario.reservar(productoId, qty) para bloquear stock temporal y evitar overselling.
5. El Pedido inicia la transacción con Pago (se crea una instancia PagoTarjeta o PagoPasarela según el método elegido).
   * Si Pago.procesar() devuelve true → el Pedido pasa a estado Pagado/Confirmado.
   * Si el pago falla → el pedido queda en estado Pendiente o Cancelado según las reglas de negocio; se libera la reserva de stock.
6. Tras pago exitoso, se invoca Notificacion.enviarEmail(...) para confirmar al cliente y se crea (si aplica) un Shipment con trackingNumber.
7. El área de operaciones/fulfillment procesa el envío y, cuando corresponda, actualiza Inventario.actualizarStock() de forma definitiva.
8. En caso de devolución o reembolso, se gestiona la lógica de Payment.refund() y la actualización de stock y estados del pedido.



MER:



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