**[Cambioteca]**

**(SAD) Software Architecture Document**

**Versión 1.0**

**Identificación de Documento**

| **Identificación** | DAS00024 |
| --- | --- |
| **Proyecto** | Cambioteca |
| **Versión** | 2.0 |

| **Documento mantenido por** | Nicolás Rebolledo - Vicente Vargas |
| --- | --- |
| **Fecha de última revisión** | 21/10/2025 |
| **Fecha de próxima revisión** | 22/10/2025 |

| **Documento aprobado por** |  |
| --- | --- |
| **Fecha de última aprobación** |  |

**Historia de Revisiones**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 21/10/2025 | 1.1 |  | Nicolás Rebolledo, Vicente Vargas |

**Tabla de Contenidos**

[**1. Introducción 3**](#_heading=h.wfhwte7xhegr)

[1.1. Contexto del Problema 3](#_heading=h.c5jfdf4ffcir)

[1.2. Propósito 3](#_heading=h.jamvidkwvlxt)

[1.3. Ámbito 4](#_heading=h.fxckk8hkxzn2)

[1.4. Definiciones, acrónimos y abreviaciones 6](#_heading=h.stari1vippjj)

[1.5. Resumen ejecutivo 7](#_heading=h.tnvvn0llh9pl)

[**2. Representación 8**](#_heading=h.s9psabevh7yz)

[2.1. Vista de casos de uso 8](#_heading=h.6d3uasl7eu00)

[**Modelo 4 + 1 66**](#_heading=h.3vu5voz8chb)

[**3. Vista Logica 66**](#_heading=h.9k4dsuj178hd)

[3.1. Diagrama de Clases 66](#_heading=h.ppz7tmmkau2r)

[3.2. Modelo E-R 69](#_heading=h.na8k822jz6tr)

[3.3. Diagramas de Secuencia 73](#_heading=h.d2h9ri66nbys)

[3.3.1. 1 Negociación y cierre de intercambio 73](#_heading=h.7tjm8g9pcvfm)

[3.3.2. Búsqueda de libros con filtros y paginación 75](#_heading=h.kbjcaikrv6pm)

[3.3.3. Registro y verificación por correo 77](#_heading=h.wz4h914miyaa)

[3.3.4. Mensajería y “visto” en conversación 79](#_heading=h.k8ejvhxi9bj2)

[3.3.5. Publicación de libro e imágenes 80](#_heading=h.f2s4kfp76bjt)

[3.3.6. Recuperación de contraseña por token 82](#_heading=h.jwtub6irs21l)

[**4. Vista de despliegue 84**](#_heading=h.feam8m6or6nu)

[4.1. Diagrama de Componentes 84](#_heading=h.es773i4clhbm)

[4.2. Diagrama de Paquetes 86](#_heading=h.n9xx3gdl9uc)

[**5. Vista de procesos 88**](#_heading=h.c3e6q680odq)

[5.1. Diagrama de Actividad 88](#_heading=h.6k5j0s1y4fmv)

[5.1.1. Registro y verificación de cuenta 88](#_heading=h.qip9hh38wih3)

[5.1.2. Solicitud de intercambio y aceptación 88](#_heading=h.8lcdbzkv2xee)

[5.1.3. Pactar fecha y lugar, generar código 89](#_heading=h.9t0mx3anw0u0)

[5.1.4. Búsqueda y visualización de detalle de libro 91](#_heading=h.b6s3e8rcbryf)

[5.1.5. Publicación y edición de libro con imágenes 92](#_heading=h.e6a1b7dqwsx)

[**6. Vista Fisica 93**](#_heading=h.69o6r2t7j1ux)

[6.1. Diagrama de Despliegue 93](#_heading=h.hylljnv21slv)

[**7. Vista de Escenario 95**](#_heading=h.xib5nzwsv)

[7.1. Diagrama de Caso de uso 95](#_heading=h.t1d6zuw1pjfk)

[7.1.1. Cuenta y seguridad 95](#_heading=h.ag7r1z32gkmh)

[7.1.2. Intercambio End to End 96](#_heading=h.uxr6ivwqwrx2)

[7.1.3. Publicar Libro 97](#_heading=h.u3643cc12hb2)

[7.1.4. Gestionar solicitud de intercambio 98](#_heading=h.wvxop6rpvd0k)

[7.1.5. Registro de Usuarios 101](#_heading=h.vziqxdk2uauy)

# Introducción

## Contexto del Problema

Cambioteca surge como respuesta a la fragmentación e ineficiencia del intercambio y la compraventa de libros físicos en Chile. Hoy, estudiantes, lectores y coleccionistas dependen de grupos informales (redes sociales, chats dispersos y ferias puntuales), lo que genera problemas recurrentes:

* Baja visibilidad y desorden del catálogo (difícil filtrar por título, autor, edición, estado, región/comuna).
* Poca confianza y trazabilidad de las transacciones (no hay reputación de usuarios, ni historial).
* Coordinación compleja de lugar/fecha/hora para encuentros.
* Falta de notificaciones y seguimiento, lo que deriva en intercambios caídos.

Este escenario afecta el acceso a libros a buen precio, ralentiza la rotación de inventario personal y frena una práctica sostenible de reutilización. Cambioteca propone unificar catálogo, conversación, coordinación y cierre de transacciones en un solo flujo, con foco en seguridad, usabilidad y alcance nacional (regiones/comunas).

## Propósito

Implementar un sistema de software multi-plataforma (web + móvil) que:

* Centralice la publicación, búsqueda y negociación de libros usados.
* Entregue chat y notificaciones integradas para coordinar intercambios/ventas.
* Asista a la definición de punto de encuentro (mapa) y fecha/hora.
* Opcionalmente soporte pagos/confirmaciones y comprobantes de transacciones.
* Mejorar la confianza mediante perfiles y reputación (calificaciones).
* Escale de forma sostenible en usuarios, publicaciones y ciudades.

## Ámbito

**Nombre del proyecto:** Cambioteca

El sistema de software **realizará**:

* Autenticación y seguridad: registro, login, recuperación de contraseña, cierre de sesión en todos los dispositivos, manejo de sesiones con JWT.
* Gestión de perfiles: datos básicos, imagen, reputación (calificaciones/reseñas), métricas de actividad.
* Publicación de libros: crear/editar/eliminar publicaciones con fotos, estado del libro, intención de intercambiar, tags, autor, editorial, ISBN, categoría/género.
* Catálogo y búsqueda avanzada: filtros por título/autor/ISBN, categoría, precio, región/comuna, estado del libro, modalidad (venta/intercambio).
* Detalle de publicación: fotos, descripción, condiciones, disponibilidad, ubicación general.
* Negociación y mensajería: chat uno-a-uno desde la publicación; notificaciones por nuevos mensajes y cambios de estado.
* Flujo de intercambio: creación de solicitudes, aceptación/rechazo, propuesta/confirmación de lugar y fecha (mapa), estado de la operación (pendiente, confirmada, realizada, cancelada).
* Puntos de encuentro en mapa: selección de lugares (región/comuna), con edición y confirmación.
* Calificaciones post-transacción: una sola calificación por usuario y por intercambio.
* Notificaciones: eventos clave (nueva solicitud, propuesta de lugar/fecha, confirmaciones, cambios de estado).
* Panel básico de actividad: mis publicaciones, mis solicitudes, mis intercambios/ventas, histórico.
* Administración básica (rol admin): moderación de contenidos y usuarios, gestión de categorías y reportes operativos.

El sistema de software **no realizará** (versión inicial):

* Logística/envíos ni tracking de paquetería.
* Arbitraje de disputas legales ni mediación avanzada (solo reportes básicos).
* Marketplace de otros rubros (foco exclusivo en libros).

**Beneficios**

* Acceso y asequibilidad: oferta amplia de libros sin costo.
* Eficiencia: publicación y búsqueda en minutos, coordinación simplificada.
* Confianza: reputación, trazabilidad y notificaciones.
* Sostenibilidad: reutilización y economía circular del libro físico.
* Alcance territorial: filtros por región/comuna y puntos de encuentro.

**Objetivos y metas**

Objetivo general: Optimizar la experiencia de intercambio de libros físicos, asegurando facilidad, confianza y trazabilidad de punta a punta.

Metas (indicativas – versión MVP y primeras iteraciones):

* Reducir el tiempo medio de publicar un libro a < 3 minutos.
* Alcanzar ≥ 80% de solicitudes con lugar/fecha confirmados en < 24 h.
* Lograr ≥ 70% de operaciones exitosas (con cierre verificado) en los primeros 90 días.
* Mantener disponibilidad ≥ 99,5% mensual del servicio.
* Cero incidentes críticos de exposición de datos personales; cumplimiento de Ley 19.628.
* Obtener NPS ≥ 40 en usuarios activos al cuarto mes.

## Definiciones, acrónimos y abreviaciones

| **ACRÓNIMO** | **DESCRIPCIÓN** |
| --- | --- |
| **Usuario** | Persona qué publica o intercambia libros. |
| **ID** | Identificador de un caso de uso, historia o requerimiento. |
| **Publicación** | Ficha de un libro disponible (venta o intercambio). |
| **Chat** | Mensajería integrada para negociar y coordinar. |
| **Calificación** | Evaluación post-transacción entre usuarios. |
| **API/REST** | Interfaz de programación; estilo de integración usado por el backend. |
| **JWT** | JSON Web Token para autenticación/autorización. |
| **WEB** | Referencia a la aplicación web utilizada para la toma de citas médicas. |
| **HTML** | HyperText Markup Language (se basa en un lenguaje de marcas para crear documentos que puedan ser distribuidos por Internet). |
| **CSS** | Cascading Style Sheets (Hojas de estilo en cascada, es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado). |
| **SQL** | Structured Query Language (es un lenguaje de programación para almacenar y procesar información en una base de datos relacional). |
| **HTTPS** | Hyper text Transfer Protocol Secure (es la versión segura de HTTP, el protocolo bajo el que se envían los datos entre su navegador y el sitio web al que está usted conectado). |
| **JSON** | JavaScript Object Notation (es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript). |

## 

## Resumen ejecutivo

Este **DAS de Cambioteca** presenta la arquitectura de una plataforma **web + móvil** para intercambio y venta de libros físicos. La solución adopta una **arquitectura por capas**:

* **Frontend móvil (Ionic + Angular)** y **Frontend web** para la experiencia de usuario.
* **Backend** en **Django + Django REST Framework**, con **JWT** para autenticación y servicios para publicaciones, catálogo, intercambio/venta, chat y notificaciones.
* **Base de datos relacional (MySQL)** para persistencia de usuarios, libros, transacciones, calificaciones, regiones/comunas, etc.

El documento detalla **metas y restricciones**, **casos de uso** (publicar, buscar, negociar, coordinar, calificar), **escenarios de calidad** (seguridad, disponibilidad, rendimiento), **vista lógica** (módulos/autenticación/catálogo/chat/mapa), **vista de procesos** (flujos de intercambio/venta con propuestas y confirmaciones), y las **decisiones de diseño** (p. ej., JWT, filtros en catálogo, control de estado de operaciones, una sola calificación por intercambio). También discute **riesgos** (fraude, contenido inapropiado, privacidad), **mitigaciones** (moderación, reputación, reportes, logs) y **lineamientos de implementación** para el despliegue y la escalabilidad.

# Representación

La arquitectura del sistema Gamelegest está representada siguiendo el enfoque del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

## Vista de casos de uso

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 001* | Registrar usuario |
| **Descripción** | |
| Permite crear una cuenta para publicar libros e intercambiar. | |
| **Actores** | |
| Visitante, Sistema, Servicio de Email. | |
| **Precondiciones** | |
| No haber iniciado sesión. | |
| **Postcondiciones** | |
| Usuario creado; sesión iniciada; email de bienvenida/verificación enviado. | |
| **Flujo Principal** | |
| * El visitante ingresa nombre, email, contraseña y acepta términos. * El sistema valida formato y unicidad del email. * Crea el usuario con contraseña hasheada. * Crea una sesión activa. * Envía correo de bienvenida/verificación. * Muestra “Registro exitoso”. | |
| **Flujo Alternativo** | |
| A1: Email ya registrado → Mostrar “El email ya está en uso”.  A2: Contraseña no cumple política → Mostrar detalle de requisitos.  A3: Falla el envío de email → Registrar usuario igual y mostrar aviso “No pudimos enviar el correo, puedes reintentarlo desde tu perfil”. | |
| **Reglas de Negocio** | |
| RN01: Email único obligatorio (índice único).  RN02: Política de contraseña: ≥8 caracteres, 1 mayúscula, 1 número.  RN03: Debe aceptar Términos y Política de Privacidad.  RN04: Throttling: máx. 5 altas por IP por hora.  RN05: El hash de contraseña debe ser PBKDF2/bcrypt/Argon2 (no texto plano).  RN02- Para iniciar sesión los campos nombre de usuario y contraseña deben estar correctos. | |
| **Criterios de Aceptación** | |
| * CA01: Si email único + contraseña válida → 201, sesión creada. * CA02: Si email ocupado → 409 y mensaje claro. * CA03: El correo de verificación se registra como “pendiente” sin bloquear el acceso básico. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 002* | Iniciar sesión |
| **Descripción** | |
| Autenticar usuario. | |
| **Actores** | |
| Usuario, Sistema | |
| **Precondiciones** | |
| Usuario registrado. | |
| **Postcondiciones** | |
| Sesión activa registrada. | |
| **Flujo Principal** | |
| * Usuario ingresa email y contraseña. * Sistema valida credenciales. * Registra sesión (dispositivo, expiración). * Devuelve token/credenciales y redirige. | |
| **Flujo Alternativo** | |
| **A1**: Credenciales inválidas → 401 “Usuario o contraseña incorrectos”.  **A2**: Cuenta bloqueada o inactiva → 423 con motivo. | |
| **Reglas de Negocio** | |
| * RN06: Bloqueo tras N intentos fallidos (ej. 5/15min). * RN07: Forzar uso de HTTPS. * RN08: Invalidate tokens al cambiar contraseña. | |
| **Criterios de Aceptación** | |
| * CA04: Con credenciales válidas → sesión creada y usable. * CA05: Tras bloqueo → rechaza logins hasta ventana de desbloqueo. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 003* | Recuperar contraseña |
| **Descripción** | |
| Restablecer contraseña con token temporal. | |
| **Actores** | |
| *Usuario, Sistema, Servicio de Email* | |
| **Precondiciones** | |
| Email registrado. | |
| **Postcondiciones** | |
| Contraseña actualizada; sesiones previas revocadas. | |
| **Flujo Principal** | |
| * Usuario solicita “Olvidé mi contraseña”. * Sistema crea password\_reset\_token (único, con expiración). * Envía email con enlace. * Usuario define nueva contraseña. * Sistema valida token, cambia contraseña, marca token como usado y elimina sesiones activas. | |
| **Flujo Alternativo** | |
| A1: Token inválido/expirado/ya usado → error y opción de reenviar. | |
| **Reglas de Negocio** | |
| * RN09: Token de un uso, vence (ej. 60 min). * RN10: Nueva contraseña debe cumplir RN02. | |
| **Criterios de Aceptación** | |
| * CA06: Tras reset → login previo invalido (401). * CA07: Auditoría registra IP/fecha del cambio. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 004* | Ver catálogo |
| **Descripción** | |
| Listar libros disponibles. | |
| **Actores** | |
| *Usuario* | |
| **Precondiciones** | |
| No aplica | |
| **Postcondiciones** | |
| No aplica | |
| **Flujo Principal** | |
| * Cargar catálogo. * Sistema lista libro con disponible=1 (portada y metadatos). * Muestra paginación/orden. | |
| **Flujo Alternativo** | |
| A1: Filtros inválidos → normalizar a valores permitidos. | |
| **Reglas de Negocio** | |
| * **RN14: Texto busca en título/autor/editorial.** * **RN15: Filtros válidos: id\_genero, estado, tipo\_tapa, rango de año.** * **RN16: No incluir disponible=0.** | |
| **Criterios de Aceptación** | |
| * CA10: El backend retorna conteo total y número de página coherentes. * CA11: El ordenamiento es estable (mismo orden entre páginas con mismos criterios). | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 006* | Ver detalle de libro |
| **Descripción** | |
| Mostrar ficha del libro, imágenes y dueño. | |
| **Actores** | |
| *Usuario* | |
| **Precondiciones** | |
| id\_libro válido. | |
| **Post Condiciones** | |
| N/A | |
| **Flujo Principal** | |
| * Usuario abre libro/ID. * Sistema retorna datos del libro, dueño, género e imágenes (portada primero). * Muestra botón “Proponer intercambio | |
| **Flujo Alternativo** | |
| A1: Libro no existe → 404.  A2: Libro disponible=0 → mostrar estado “No disponible”. | |
| **Reglas de Negocio** | |
| * RN17: Mostrar is\_portada=1 al inicio. * RN18: El dueño no puede solicitar su propio libro. | |
| **Criterios de Aceptación** | |
| * CA12: Carga < 2s en condiciones normales (no N+1 queries). * CA13: Al cambiar portada, se refleja al refrescar. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 007* | Publicar libro |
| **Descripción** | |
| Alta de un libro. | |
| **Actores** | |
| *Secretario y Médicos* | |
| **Precondiciones** | |
| *Sesión iniciada.* | |
| **Post Condiciones** | |
| Libro visible en catálogo. | |
| **Flujo Principal** | |
| * Completar formulario: título, autor, ISBN, año, editorial, estado, tapa, género, descripción. * Validación de campos. * Insertar libro con disponible=1. * Mostrar “Publicación creada”. | |
| **Flujo Alternativo** | |
| A1: Género inválido → error.  A2: Año fuera de rango → error. | |
| **Reglas de Negocio** | |
| * RN19: Género debe existir (FK). * RN20: Año razonable (>= 1800 y <= año actual). * RN21: ISBN opcional pero si viene, validar largo (10/13 sin guiones). | |
| **Criterios de Aceptación** | |
| * CA14: Devuelve id\_libro y fecha de subida. * CA15: El libro aparece en catálogo de inmediato. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 008* | Subir imágenes y elegir portada |
| **Descripción** | |
| Adjuntar imágenes y definir portada. | |
| **Actores** | |
| *Usuario (dueño del libro)* | |
| **Precondiciones** | |
| Libro propio. | |
| **Postcondiciones** | |
| Registros en imagen\_libro. | |
| **Flujo Principal** | |
| * Seleccionar archivos. * Sistema valida tipo (jpeg/png/webp) y peso (≤5MB). * Sube a almacenamiento y guarda URL con orden. * Marcar una como portada. | |
| **Flujo Alternativo** | |
| * A1: Archivo inválido → rechazar y detallar causa. * A2: Intento de múltiples portadas → sistema fuerza una sola **is\_portada=1**. | |
| **Reglas de Negocio** | |
| * RN22: Una sola portada por libro. * RN23: Máx. 8 imágenes por libro. * RN24: Orden numérico sin huecos (reordenamiento normaliza). | |
| **Criterios de Aceptación** | |
| * CA16: La portada aparece primera en el detalle. * CA17: El orden mostrado coincide con orden. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 009* | Editar libro |
| **Descripción** | |
| Modificar datos de un libro propio. | |
| **Actores** | |
| *Usuario* | |
| **Precondiciones** | |
| Ser dueño; libro existente. | |
| **Postcondiciones** | |
| Datos actualizados. | |
| **Flujo Principal** | |
| * Abrir “Editar”. * Cambiar campos permitidos. * Guardar. * Confirmación. | |
| **Flujo Alternativo** | |
| A1: Validaciones fallan → mostrar errores.  A2: Libro marcado no disponible por intercambio → permitir editar descripción, no disponibilidad. | |
| **Reglas de Negocio** | |
| * RN25: No se puede cambiar id\_usuario. * RN26: Mantener integridad de FK (género). | |
| **Criterios de Aceptación** | |
| * CA18: Los cambios se ven al refrescar catálogo/detalle. * CA19: No se rompe la portada ni las imágenes existentes. | |

| *RF 010* | Eliminar libro |
| --- | --- |
| **Descripción** | |
| Borrar un libro si no está involucrado en un intercambio completado. | |
| **Actores** | |
| Dueño del libro. | |
| **Postcondiciones** | |
| Libro borrado o rechazo con explicación. | |
| **Flujo Alternativo** | |
| * A1: Existe intercambio completado asociado → “No se puede eliminar: libro con intercambio Completado”. | |
| **Reglas de Negocio** | |
| * RN27: En línea con trigger de protección (no eliminar si esta completado). * RN28: Cascada coherente (imágenes, favoritos, ofertas). | |
| **Criterios de Aceptación** | |
| * CA20: Si viola RN27 → 409 con mensaje claro. * CA21: Si no, 204 sin contenido. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *R06* | Cerrar sesión |
| **Descripción** | |
| Funcionalidad que permite a un usuario cerrar sesión en el sistema, para garantizar la privacidad y seguridad de los datos. | |
| **Actores** | |
| *Todos los usuarios que interactúan con el sistema* | |
| **Precondiciones** | |
| * RF001- Iniciar sesión * RF005- Mostrar menú secretaria * RF030- Mostrar menú médico * RF039- Home cajero * RF007- Mostrar menú paciente | |
| **Postcondiciones** | |
| No aplica | |
| **Flujo Principal** | |
| * Iniciar sesión * El usuario selecciona la opción "Cerrar Sesión" en la interfaz del sistema. * El sistema finaliza la sesión del médico. * El sistema redirige al usuario a la página de inicio de sesión. | |
| **Flujo Alternativo** | |
| *No aplica* | |
| **Reglas de Negocio** | |
| * Solo los usuarios que han iniciado sesión pueden utilizar la opción de cerrar sesión. * Cerrar sesión debe garantizar que el usuario ya no tenga acceso a las funciones protegidas del sistema. | |
| **Criterios de Aceptación** | |
| * Un usuario puede cerrar sesión en el sistema. * Después de cerrar sesión, el usuario no debe poder acceder a las funciones protegidas sin volver a iniciar sesión. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 011* | Crear solicitud de intercambio |
| **Descripción** | |
| Pedir un libro a su dueño. | |
| **Actores** | |
| *Usuario , Sistema* | |
| **Precondiciones** | |
| Libro disponible=1, no ser dueño. | |
| **Postcondiciones** | |
| solicitud\_intercambio en “Pendiente”. | |
| **Flujo Principal** | |
| * Click “Solicitar intercambio”. * Sistema crea solicitud (solicitante, receptor, libro deseado). * Notifica al receptor. | |
| **Flujo Alternativo** | |
| A1: Ya hay una solicitud abierta del mismo solicitante al mismo libro → denegar.  A2: Libro dejó de estar disponible → error “No disponible”. | |
| **Reglas de Negocio** | |
| RN29: Una solicitud activa por (solicitante, libro).  RN30: Estado inicial “Pendiente”. | |
| **Criterios de Aceptación** | |
| * CA22: Se crea una sola fila válida y se notifica. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 012* | Ofertar libro a una solicitud |
| **Descripción** | |
| Receptor, Sistema | |
| **Actores** | |
| *Paciente, Secretaría* | |
| **Precondiciones** | |
| Solicitud “Pendiente”; libro propio disponible. | |
| **Postcondiciones** | |
| solicitud\_oferta creada. | |
| **Flujo Principal** | |
| * Abrir solicitud recibida. * Elegir libro propio disponible. * Enviar oferta. * Sistema guarda oferta y notifica al solicitante. | |
| **Flujo Alternativo** | |
| * A1: Oferta duplicada (misma solicitud y libro) → rechazar por unicidad. * A2: Libro ofrecido ya no disponible → error. | |
| **Reglas de Negocio** | |
| * RN31: Unicidad (id\_solicitud, id\_libro\_ofrecido). * RN32: No permitir ofrecer libros disponible=0. | |
| **Criterios de Aceptación** | |
| * CA23: Rechaza duplicados con 409; oferta válida crea registro. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 013* | Aceptar oferta y crear intercambio |
| **Descripción** | |
| Confirmar oferta y crear intercambio + conversacion. | |
| **Actores** | |
| *Solicitante, Sistema* | |
| **Precondiciones** | |
| Solicitud “Pendiente”; existe al menos una oferta válida. | |
| **Postcondiciones** | |
| Solicitud “Aceptada”; intercambio y conversacion creados. | |
| **Flujo Principal** | |
| * Solicitante selecciona una oferta. * Presiona “Aceptar”. * Sistema actualiza solicitud a “Aceptada”. * Crea intercambio (con libro ofrecido aceptado) y conversacion. * Notifica al receptor. | |
| **Flujo Alternativo** | |
| * A1: Oferta retirada/caducada → abortar y mostrar mensaje. * A2: Error creando conversación → rollback total. | |
| **Reglas de Negocio** | |
| * RN33: Al aceptar una oferta, invalidar el resto. * RN34: Estados permitidos: Pendiente → Aceptada. | |
| **Criterios de Aceptación** | |
| * CA24: Devuelve id\_intercambio y id\_conversacion creados. * CA25: No quedan ofertas huérfanas activas. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 014* | Enviar mensaje por chat |
| **Descripción** | |
| Intercambiar mensajes en la conversación del intercambio. | |
| **Actores** | |
| *Solicitante, Receptor, Sistema (WebSocket)* | |
| **Precondiciones** | |
| conversacion activa. | |
| **Postcondiciones** | |
| Mensaje guardado; actualización de “último mensaje” y no leídos. | |
| **Flujo Principal** | |
| * Usuario escribe y envía mensaje. * Sistema inserta en conversacion\_mensaje. * Trigger/servicio actualiza conversacion.ultimo\_id\_mensaje y actualizado\_en. * Entrega en tiempo real a la contraparte. | |
| **Flujo Alternativo** | |
| A1: Conversación cerrada/archivada → bloquear envío.  A2: Conexión WS caída → fallback por HTTP, luego reintento WS. | |
| **Reglas de Negocio** | |
| * RN35: Mensajes eliminados no son entregados. * RN36: Límite de longitud (ej. 2000 chars). * RN37: Antispam (rata máxima de mensajes por minuto). | |
| **Criterios de Aceptación** | |
| * CA26: Entrega ≤1s via WS bajo condiciones normales. * CA27: unread\_count sube para el receptor hasta “marcar visto”. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 015* | Pactar fecha y lugar |
| **Descripción** | |
| Definir encuentro del intercambio. | |
| **Actores** | |
| *Cualquiera de las partes* | |
| **Precondiciones** | |
| Intercambio creado. | |
| **Postcondiciones** | |
| fecha\_intercambio\_pactada y lugar\_intercambio definidos. | |
| **Flujo Principal** | |
| * Abrir detalles del intercambio. * Ingresar fecha futura y lugar. * Guardar; sistema actualiza y notifica. | |
| **Flujo Alternativo** | |
| A1: Fecha pasada → error de validación.  A2: Reprogramación → se registra nueva fecha y se notifica. | |
| **Reglas de Negocio** | |
| * RN38: Fecha en zona horaria del usuario. * RN39: Lugar con longitud mínima (ej. ≥5 chars). | |
| **Criterios de Aceptación** | |
| * CA28: Ambos usuarios ven fecha/lugar actualizados. * CA29: Reprogramaciones quedan auditadas con timestamp. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 016* | Generar código de intercambio |
| **Descripción** | |
| Crear código único para validar el encuentro. | |
| **Actores** | |
| *Sistema* | |
| **Precondiciones** | |
| Intercambio con fecha pactada. | |
| **Postcondiciones** | |
| intercambio\_codigo creado con expira\_en. | |
| **Flujo Principal** | |
| * Usuario solicita “Generar código”. * Sistema genera código único (ej. 12 chars) y fecha de expiración. * Guarda y muestra a ambas partes. | |
| **Flujo Alternativo** | |
| A1: Ya hay código vigente → reutilizar; si caducado, generar uno nuevo.  A2: Solicitudes repetidas en <1 min → throttling. | |
| **Reglas de Negocio** | |
| * RN40: Código único (índice uq\_codigo). * RN41: Expira en N días/horas tras generación. | |
| **Criterios de Aceptación** | |
| * CA30: Código no visible a terceros (solo partes). * CA31: No se crean duplicados por condiciones de carrera. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 017* | Validar código en el encuentro |
| **Descripción** | |
| Verificar código en el lugar y momento acordado. | |
| **Actores** | |
| *Cualquiera de las partes, Sistema* | |
| **Precondiciones** | |
| Código vigente y no usado. | |
| **Postcondiciones** | |
| usado\_en registrado si válido. | |
| **Flujo Principal** | |
| * Usuario ingresa el código. * Sistema busca el código y verifica expiración/uso. * Si válido, marca usado\_en y confirma. | |
| **Flujo Alternativo** | |
| * A1: Código inexistente/erróneo → error y reintento. * A2: Código expirado → permitir regeneración desde flujo RF 016 (si política lo permite). | |
| **Reglas de Negocio** | |
| RN42: Un solo uso por intercambio.  RN43: Tolerancia de X minutos respecto a la hora pactada (opcional). | |
| **Criterios de Aceptación** | |
| * CA32: Validaciones concurrentes no marcan dos veces. * CA33: Se registra auditoría (quién validó, cuándo, IP). | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 018* | Completar intercambio |
| **Descripción** | |
| Marcar como “Completado” y actualizar inventario/contadores. | |
| **Actores** | |
| *Receptor o Solicitante (según política), Sistema* | |
| **Precondiciones** | |
| Código validado o política de completado manual. | |
| **Postcondiciones** | |
| intercambio.estado=Completado; libros pasan a disponible=0; contadores de usuarios incrementan. | |
| **Flujo Principal** | |
| * Usuario presiona “Completar”. * Sistema ejecuta sp\_marcar\_intercambio\_completado(id, fecha). * Marca libros como no disponibles. * Incrementa usuario.numero\_intercambios de ambas partes. * Dispara notificaciones para calificar. | |
| **Flujo Alternativo** | |
| * A1: Algún libro ya está disponible=0 por otro cierre → SP aborta y retorna error. * A2: Inconsistencia detectada (libro ya usado en otro completado) → SP aborta y explica. | |
| **Reglas de Negocio** | |
| RN44: SP debe ser atómico (COMMIT/ROLLBACK).  RN45: No duplicar cierres sobre el mismo intercambio.  . | |
| **Criterios de Aceptación** | |
| * CA34: Estado del intercambio = “Completado” y disponibilidad de ambos libros = 0. * CA35: Contadores de las partes aumentan en 1. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 019* | Calificar usuario |
| **Descripción** | |
| Registrar puntuación y comentario tras un intercambio. | |
| **Actores** | |
| *Solicitante, Receptor* | |
| **Precondiciones** | |
| Intercambio completado. | |
| **Postcondiciones** | |
| calificación creada de forma única por (intercambio, calificador). | |
| **Flujo Principal** | |
| * Usuario abre pantalla de calificación. * Ingresa puntuación (1–5) y comentario (opcional). * Guarda; sistema inserta y confirma. | |
| **Flujo Alternativo** | |
| * A1: Intento de calificar dos veces el mismo intercambio → rechazo por unicidad. * A2: Puntuación fuera de rango → error. | |
| **Reglas de Negocio** | |
| RN46: Unicidad por (id\_intercambio, id\_usuario\_calificador).  RN47: Comentario máx. 500 caracteres; sin enlaces prohibidos.  RN48: El promedio mostrado en perfil se recalcula al vuelo. | |
| **Criterios de Aceptación** | |
| * CA36: La calificación aparece en el historial del perfil. * CA37: El promedio cambia acorde a la nueva entrada. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 020* | Marcar/Quitar favorito |
| **Descripción** | |
| Guardar o quitar libros favoritos del usuario. | |
| **Actores** | |
| *Usuario* | |
| **Precondiciones** | |
| Sesión iniciada. | |
| **Postcondiciones** | |
| Registro en favorito agregado o eliminado. | |
| **Flujo Principal** | |
| * Usuario pulsa “Favorito”. * Si no existe, crear registro; si existe, eliminar. * Actualizar icono/contador. | |
| **Flujo Alternativo** | |
| A1: Dueño marca su propio libro → permitido (opcional) o denegado según política. | |
| **Reglas de Negocio** | |
| RN49: Unicidad (id\_usuario, id\_libro).  RN50: Idempotencia: repetir acción alterna el estado. | |
| **Criterios de Aceptación** | |
| * CA38: Jamás se duplica un favorito. * CA39: El estado visual coincide con el registro real. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 021* | Ver y gestionar notificaciones |
| **Descripción** | |
| Listar notificaciones, marcarlas como leídas o limpiarlas. | |
| **Actores** | |
| *Usuario, Sistema* | |
| **Precondiciones** | |
| Sesión iniciada. | |
| **Postcondiciones** | |
| Estados de lectura actualizados en notificación. | |
| **Flujo Principal** | |
| * El usuario abre “Notificaciones”. * El sistema lista notificaciones ordenadas por estado de intercambios. * El usuario marca una o varias como leídas. * El sistema actualiza estado . | |
| **Flujo Alternativo** | |
| A1: No hay notificaciones → mostrar “Sin novedades”.  A2: Selecciona “Marcar todas como leídas” → el sistema actualiza en lote. | |
| **Reglas de Negocio** | |
| RN51: Solo el dueño puede cambiar el estado de sus notificaciones.  RN52: El listado es paginado (por ejemplo, 20 ítems).  RN53: Badge del header refleja conteo de no leídas. | |
| **Criterios de Aceptación** | |
| * CA40: Tras marcar leídas, el badge disminuye inmediatamente. * CA41: La paginación refleja total, page, pageSize. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 022* | Editar perfil y ubicación |
| **Descripción** | |
| Cambiar nombre, imagen, comuna y región. | |
| **Actores** | |
| *Usuario, Sistema* | |
| **Precondiciones** | |
| Sesión iniciada. | |
| **Postcondiciones** | |
| Datos actualizados en usuario. | |
| **Flujo Principal** | |
| * Usuario abre “Perfil”. * Edita nombres, imagen, comuna/region, teléfono. * Guarda cambios. * Sistema valida y persiste. | |
| **Flujo Alternativo** | |
| A1: Comuna no existente → error (FK). | |
| **Reglas de Negocio** | |
| RN64: Email único; si cambia, revalidar correo.  RN65: Imagen de perfil con peso y tipo soportados. | |
| **Criterios de Aceptación** | |
| * CA52: El perfil refleja cambios al refrescar. * CA53: La comuna y región corresponden a FKs válidas. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 023* | Reprogramar intercambio |
| **Descripción** | |
| Cambiar fecha y/o lugar de un intercambio activo. | |
| **Actores** | |
| *Cualquiera de las partes, Sistema* | |
| **Precondiciones** | |
| Intercambio en “Aceptado”. | |
| **Postcondiciones** | |
| Nueva fecha/lugar registrados y notificados. | |
| **Flujo Principal** | |
| * Usuario abre el intercambio. * Propone nueva fecha y lugar válidos. * El sistema actualiza campos y notifica a la contraparte. | |
| **Flujo Alternativo** | |
| A1: Fecha pasada → rechazo.  A2: Cambio de último minuto (configurable) → requiere confirmación de ambas partes. | |
| **Reglas de Negocio** | |
| RN66: Fecha futura obligatoria; TZ del usuario.  RN67: Reprogramaciones auditadas (quién/cuándo/desde). | |
| **Criterios de Aceptación** | |
| * CA54: La otra parte ve el cambio y recibe notificación. * CA55: Las reprogramaciones aparecen en el historial del intercambio. | |

| **Identificador** | **Nombre** |
| --- | --- |
| *RF 024* | Cancelar intercambio |
| **Descripción** | |
| Cancelar un intercambio antes de completarse. | |
| **Actores** | |
| *Intercambio en “Aceptado”.* | |
| **Precondiciones** | |
| Intercambio en “Aceptado”. | |
| **Postcondiciones** | |
| estado\_intercambio=Cancelado, notificaciones enviadas. | |
| **Flujo Principal** | |
| * Usuario selecciona “Cancelar”. * Confirma el motivo. * El sistema marca “Cancelado”, notifica a la otra parte. | |
| **Flujo Alternativo** | |
| A1: Ya “Completado” → no se permite cancelar. | |
| **Reglas de Negocio** | |
| * RN68: Política de cancelación puede afectar reputación (opcional). * RN69: Tras cancelar, se puede reabrir la solicitud original (opcional). | |
| **Criterios de Aceptación** | |
| * CA56: Estado cambia a “Cancelado” y queda visible en el detalle. * CA57: Se registra auditoría y notificaciones. | |

# Modelo 4 + 1

# Vista Logica

La vista lógica modela el dominio de Cambioteca sin amarrarse a tecnología. Define entidades, responsabilidades y relaciones que soportan publicar libros, negociar y concretar intercambios, mensajear y calificar.

## Diagrama de Clases

**Áreas del modelo**

* **Usuarios y ubicación**: Usuario concentra identidad, reputación y estado; se asocia a Comuna y Region.
* **Catálogo**: Libro (disponibilidad, estado de conservación, metadatos) es raíz del agregado; ImagenLibro (composición, portada/orden) y Genero clasifican.
* **Negociación e intercambio**: SolicitudIntercambio (ciclo Pendiente/Aceptada/Rechazada/Cancelada) puede tener SolicitudOferta. La aceptación deriva en Intercambio (Pendiente/Completado/Cancelado) y, opcionalmente, IntercambioCodigo para confirmar en sitio.
* **Comunicación**: Conversacion (vinculada a **solicitud o intercambio**) agrupa ConversacionMensaje y ConversacionParticipante (rol, mute, último visto).
* **Reputación y engagement**: Calificacion (una por usuario y por intercambio), Favorito (único por usuario+libro), Notificacion. LibroSolicitudesVistas optimiza contadores/badges.

**Relaciones clave (cardinalidades)**

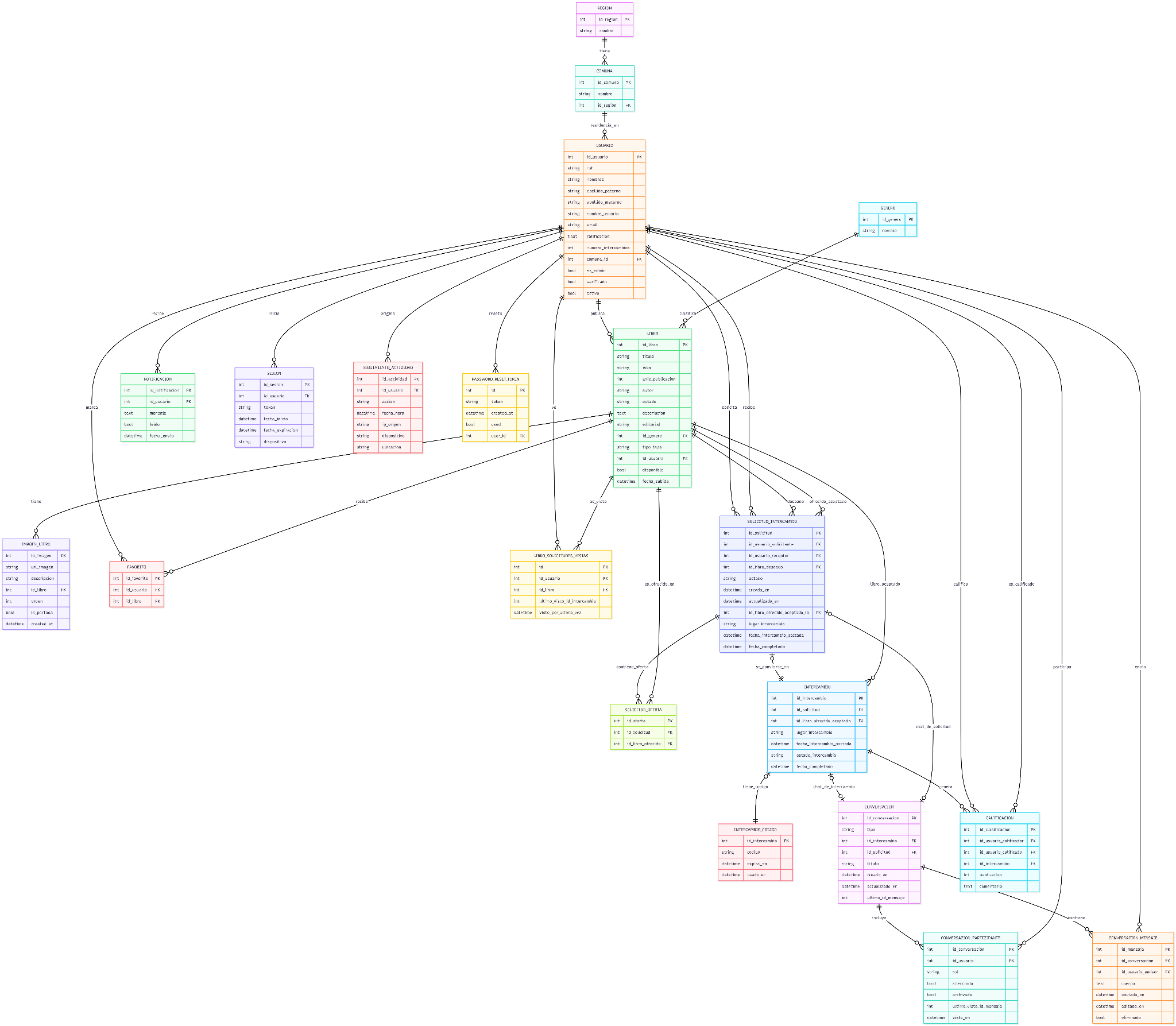
* Usuario 1 — 0..\* Libro; Libro 1 \*— 0..\* ImagenLibro (composición); Libro 1 — 0..\* Favorito y Usuario 1 — 0..\* Favorito.
* Usuario 1 — 0..\* SolicitudIntercambio como solicitante y receptor.
* SolicitudIntercambio 1 — 0..\* SolicitudOferta; SolicitudIntercambio 1 — 0..1 Intercambio.
* Intercambio 1 — 0..1 IntercambioCodigo; Intercambio 1 — 0..\* Calificacion.
* Conversacion 1 — 0..\* ConversacionMensaje; Conversacion 1 — 0..\* ConversacionParticipante.
* Usuario 0..\* — 1 Comuna; Comuna 0..\* — 1 Region.

**Reglas de negocio reflejadas**

* Al **completar** un intercambio se marcan **no disponibles** los libros involucrados y se incrementa numero\_intercambios.
* **Unicidad**: favorito (usuario+libro), calificación (intercambio+calificador) y oferta (solicitud+libro\_ofrecido).
* **Estados consistentes**: no se acepta si la solicitud está rechazada/cancelada; no se cancela un intercambio completado.
* **Portada única** por libro; mensajes con control de edición/eliminación y “último visto”.

**Racional de diseño** Agregados claros (Libro, Solicitud/Intercambio, Conversación) reducen acoplamiento y facilitan transacciones coherentes; el modelo está normalizado y preparado para ORM relacional (claves surrogate, FKs y restricciones). La confirmación con código y el chat quedan **desacoplados** del núcleo de intercambio, permitiendo evolucionar transporte y políticas sin afectar el dominio.

## Modelo E-R



**Acá se visualiza mejor el modelo ER:** [**Enlace Github**](https://github.com/Nicofelipe/cambioteca/blob/main/Fase%202/Documentos/Seguimiento%20PPT/Diagramas/Modelo%20ER.png)

El esquema está normalizado y centrado en usuario y libro, desde donde se articulan los procesos de solicitud, intercambio y mensajería.

**Núcleo transaccional**

* **solicitud\_intercambio**: inicio del flujo (estados: Pendiente/Aceptada/Rechazada/Cancelada). Claves foráneas a solicitante, receptor y libro deseado.
* **solicitud\_oferta**: libros ofrecidos a una solicitud (único por solicitud + libro\_ofrecido).
* **intercambio**: se crea al aceptar una solicitud; guarda lugar/fecha y estado (*Pendiente/Completado/Cancelado*).
* **intercambio\_codigo**: código de confirmación 1:1 con expiración y uso único.

**Catálogo y preferencias**

* **libro**: metadatos, estado de conservación y disponible.
* **imagen\_libro**: URLs de imágenes con orden y bandera de portada (relación fuerte con libro).
* **genero**: catálogo de géneros.
* **favorito**: marca de interés (único por usuario+libro).
* **libro\_solicitudes\_vistas**: último intercambio visto por usuario en la ficha del libro (optimiza badges).

**Comunicación y reputación**

* **conversacion**, **conversacion\_mensaje**, **conversacion\_participante**: chat asociado a **una** solicitud o **un** intercambio (excluyentes). Controlan último visto, mute/archivo y conteo de no leídos.
* **calificacion**: puntaje y comentario por intercambio; restricción única por (id\_intercambio, id\_usuario\_calificador).
* **notificacion**: avisos del sistema a usuarios.

**Ubicación y seguridad**

* **comuna** ↔ **region** (jerarquía territorial); usuario referencia comuna.
* Soporte de autenticación/administración (tablas auth\_\*, django\_\*), **sesion** y **password\_reset\_token** para recuperación de clave.

**Integridad y reglas implementadas**

* **FKs y cascadas**: eliminaciones en cascada donde corresponde (p. ej., mensajes al borrar conversación).
* **Unicidades**: favorito (usuario+libro), oferta (solicitud+libro), calificación (intercambio+calificador).
* **Triggers**:  
  + ai\_conversacion\_mensaje: actualiza conversacion.ultimo\_id\_mensaje y actualizado\_en.
  + bu\_conversacion\_mensaje\_editado: marca editado\_en al modificar cuerpo.
  + bu\_libro\_prevent\_delete\_completed: bloquea borrar libros usados en intercambios **Completados**.
* **Procedimiento** sp\_marcar\_intercambio\_completado: operación atómica que valida código/estado, marca intercambio como **Completado**, pone ambos libros **no disponibles** y aumenta numero\_intercambios.
* **Vistas**: v\_user\_chats (conteo de no leídos por participante) e intercambio\_legacy (compatibilidad histórica).

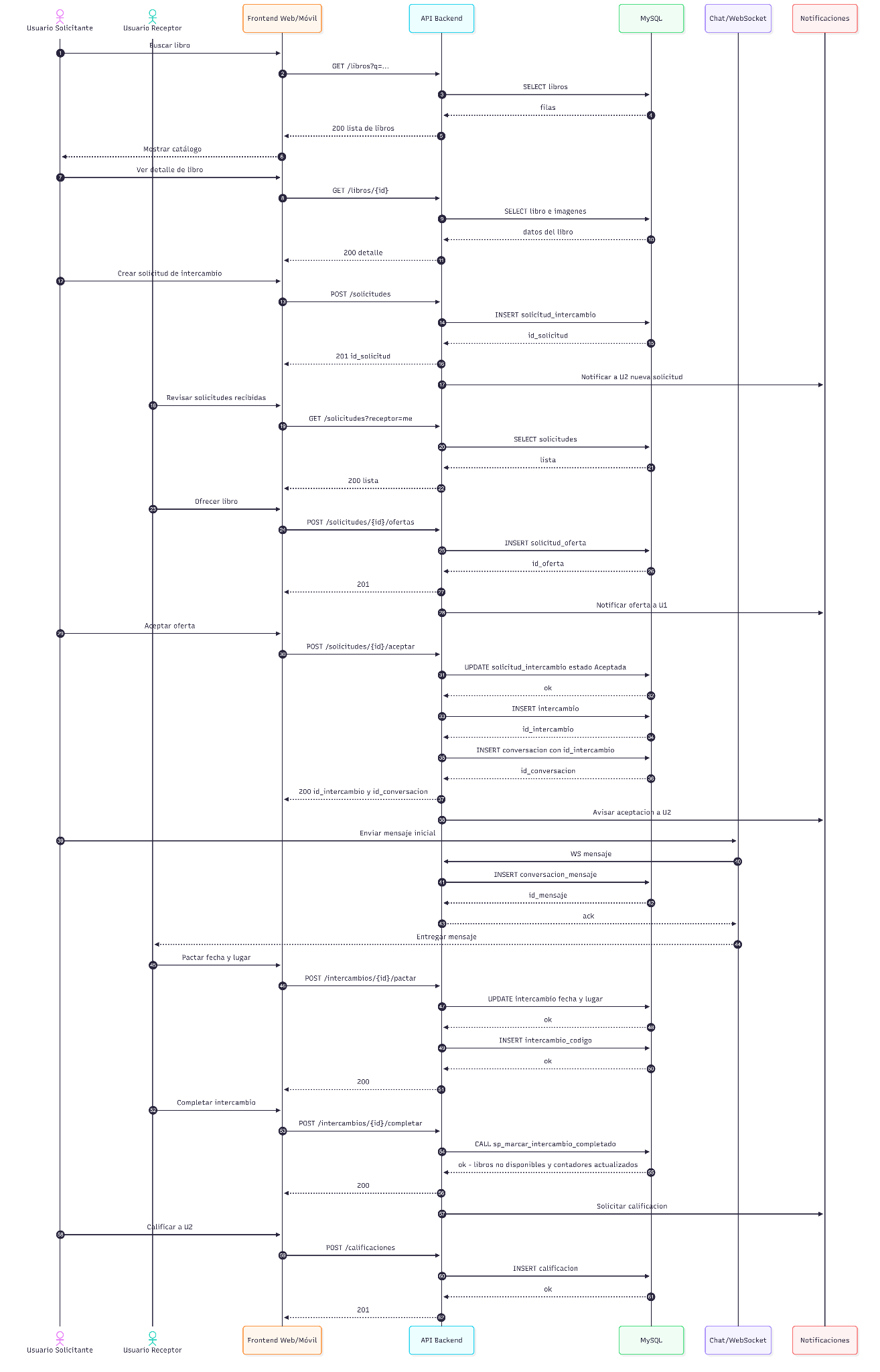
**Rendimiento (índices destacados)**

* idx\_libro\_disponible\_genero, idx\_libro\_usuario\_disponible para catálogos/búsquedas.
* idx\_conversacion\_actualizado, idx\_cm\_conversacion\_orden para chat.
* idx\_cp\_usuario, idx\_cp\_conversacion\_visto para “último visto” y bandejas.

**En síntesis**, el diagrama físico refleja los casos de uso principales: publicación, búsqueda, negociación, coordinación, confirmación y calificación; preserva integridad mediante FKs, unicidades y triggers; y expone puntos de extensión claros (notificaciones, media, recuperación de clave) sin acoplarlos al núcleo del intercambio.

## Diagramas de Secuencia

### 1 Negociación y cierre de intercambio



**Alcance.** Flujo “feliz” de punta a punta entre **Usuario Solicitante (U1)** y **Usuario Receptor (U2)**, con **Frontend**, **API**, **DB (MySQL)**, **Chat WebSocket** y **Notificaciones**.

**Resumen del flujo.**

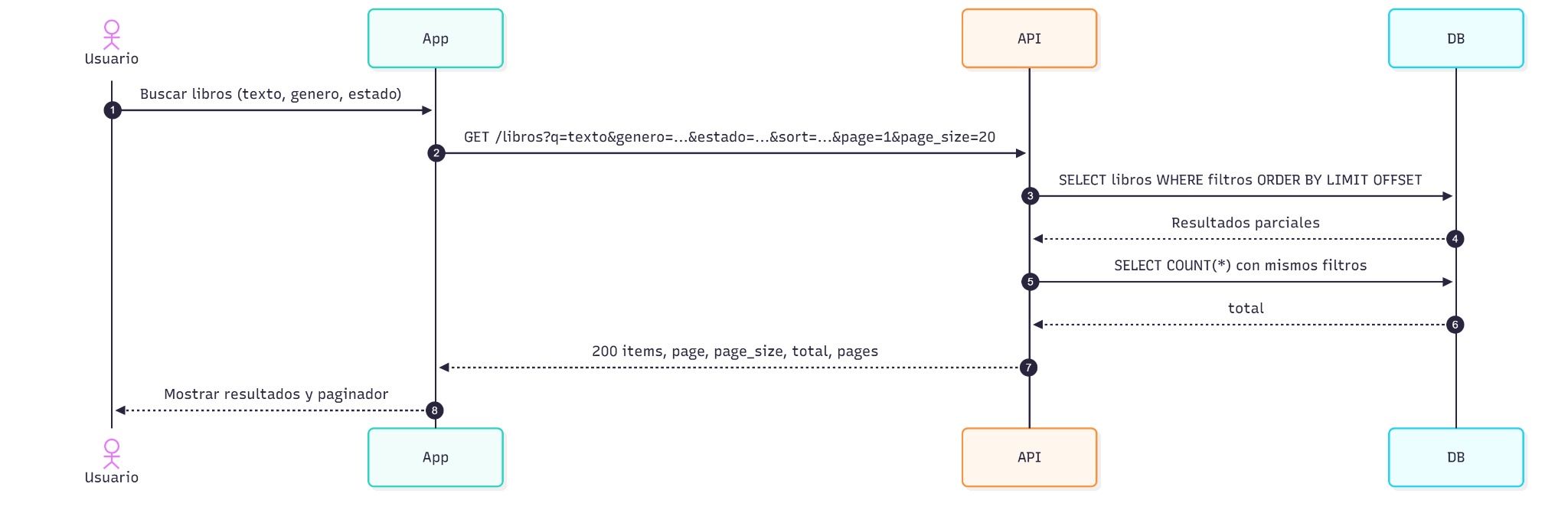
1. U1 busca y abre el detalle del libro.
2. U1 crea **solicitud**; la API persiste y **notifica** a U2.
3. U2 ofrece un libro y luego **acepta**; la API crea **Intercambio** y **Conversación**.
4. Mensajería en tiempo real por WebSocket (persistencia y ack).
5. Se **pacta** fecha/lugar y se emite **código** de confirmación.
6. En el encuentro, U2 **completa**; procedimiento transaccional marca libros **no disponibles** y actualiza contadores.
7. Ambos **califican** (una por participante).

**Reglas clave.** Estados válidos, operación de cierre transaccional, calificación única por intercambio.

**Figuras relacionadas.**

* 3.2.b **Rechazar solicitud** (U2 cambia a *Rechazada* y se notifica a U1).
* 3.2.c **Cancelar solicitud** (U1, sólo *Pendiente*).
* 3.2.d **Cancelar intercambio** (cualquiera, no *Completado*).
* 3.2.e **Recuperar contraseña** (request → validate → confirm; token usado).
* 3.2.f **Búsqueda en favoritos** (GET con filtros y paginación).

### Búsqueda de libros con filtros y paginación



**Alcance.** Consulta del catálogo por **texto, género y estado**, con **paginación** y metadatos.

**Resumen del flujo.**

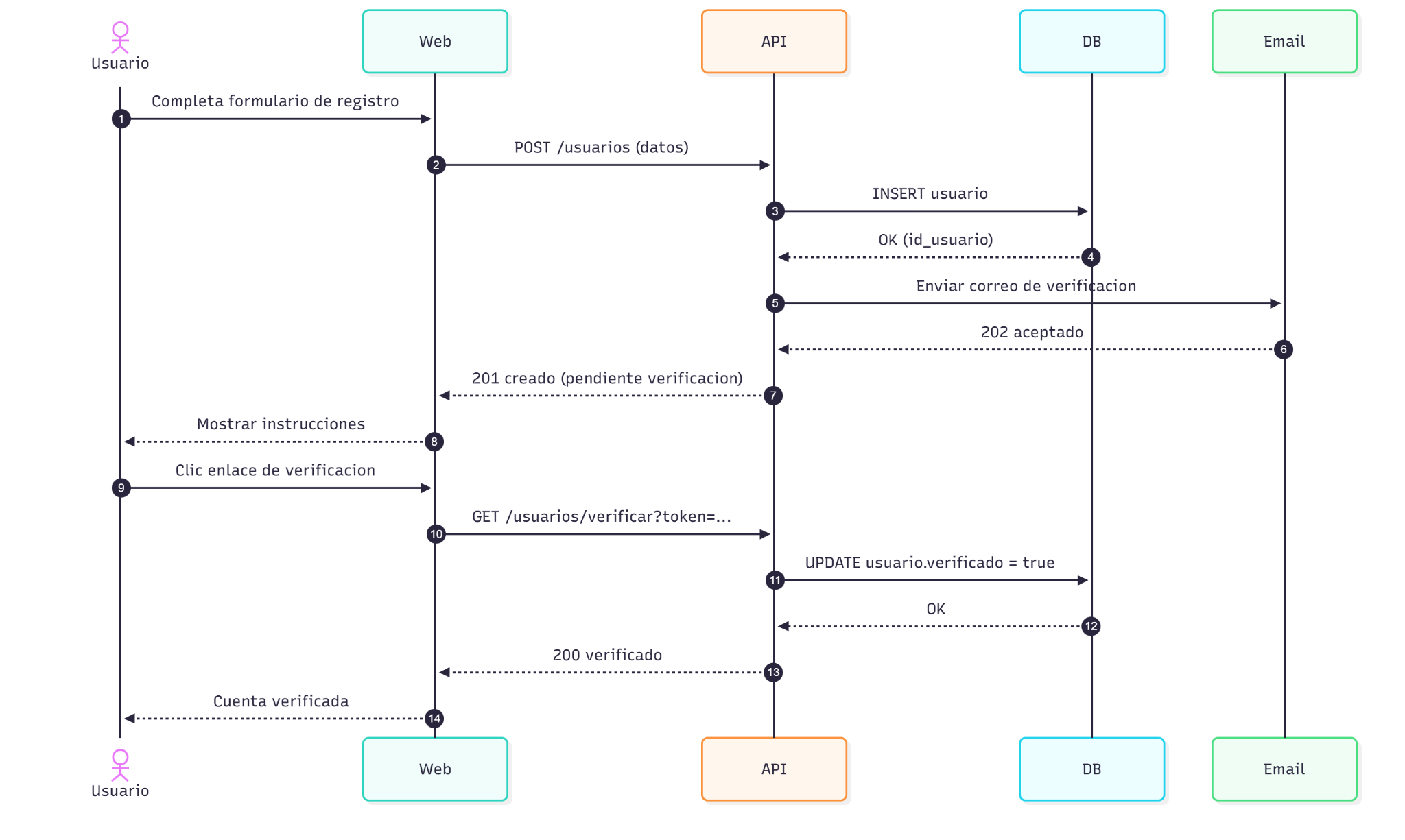
1. **Usuario → App:** inicia búsqueda con filtros.
2. **App → API:** GET /libros?q=…&genero=…&estado=…&sort=…&page=1&page\_size=20.
3. **API → DB:** SELECT … WHERE <filtros> ORDER BY <sort> LIMIT/OFFSET → devuelve **items** de la página.
4. **API → DB:** SELECT COUNT(\*) con **los mismos filtros** → obtiene **total**.
5. **API → App:** 200 con {items, page, page\_size, total, pages}.
6. **App:** muestra resultados y paginador.

**Reglas y criterios.**

* Filtros y conteo deben usar **la misma condición**; ordenamiento **determinístico** por updated\_at, id.
* Límite de página (p. ej., 1–100); valores fuera de rango → 400.
* Índices recomendados: (disponible, id\_genero), titulo/autor para búsqueda y updated\_at para orden.
* Resultados vacíos → 200 con items=[] y total=0

.

### Registro y verificación por correo



**Alcance.** Alta de usuario y activación por email. Actores: **Usuario**, **Web**, **API**, **DB**, **Email**.

**Flujo resumido.**

1. Usuario completa formulario.
2. Web → API: POST /usuarios.
3. API inserta en **DB** (verificado=false, password hasheado) y genera **token** firmado.
4. API → Email: envío del enlace de verificación.
5. API → Web: 201 (pendiente de verificación) y la UI muestra instrucciones.
6. Usuario hace clic en el enlace.
7. Web → API: GET /usuarios/verificar?token=....
8. API valida token y marca verificado=true en **DB**.
9. API → Web: 200 y redirección a inicio de sesión / cuenta.

**Reglas y criterios.**

* **Email único**; responder mismo mensaje si ya existe (evita enumeración).
* **Token** de un solo uso, con expiración (p.ej., 24 h); errores: 400/410 inválido/expirado, 409 ya verificado.
* **Idempotencia**: repetir GET /verificar sobre token usado debe devolver 200 si la cuenta ya está verificada.
* **Seguridad**: contraseñas con hash fuerte (p.ej., PBKDF2/bcrypt/argon2), rate-limit en registro y verificación.
* **Trazabilidad**: registrar envío y verificación (IP, timestamp) para auditoría.

### Mensajería y “visto” en conversación

**Alcance.** Envío de mensajes en tiempo real entre usuarios y confirmación de lectura. Actores: **App A/B**, **WebSocket**, **API**, **DB**.

**Resumen del flujo.**

1. Usuario A envía texto → **App A** publica por **WebSocket**.
2. **API** recibe, **persiste** en conversacion\_mensaje y devuelve id\_mensaje.
3. **WebSocket** hace **fan-out** a los participantes: ack al emisor y entrega a **App B**.
4. Usuario B lee; **App B** llama POST /conversaciones/{id}/visto con id\_mensaje.
5. **API** actualiza conversacion\_participante.ultimo\_visto\_id\_mensaje en **DB** y responde 200.

**Reglas y criterios.**

* Solo **participantes** autorizados pueden publicar y marcar visto.
* Fan-out **después** de persistir (no se pierden mensajes).
* ultimo\_visto\_id\_mensaje es **monotónico** por usuario; idempotente.
* Ordenación por id\_mensaje/timestamp; reintentos si WS cae (fallback REST).
* Mensaje editable/eliminable se registra como tal; aplicar rate-limit y saneo de contenido.

### Publicación de libro e imágenes

**Alcance.** Alta de un libro y carga de sus imágenes. Actores: **Usuario**, **Web/App**, **API**, **DB**, **Almacén de medios**.

**Flujo resumido.**

1. Usuario inicia **nueva publicación**.
2. Web/App → API: POST /libros (título, autor, isbn, género, estado, tapa…).
3. API → DB: INSERT libro → devuelve id\_libro.
4. **Por cada imagen**: Web/App → API: POST /libros/{id}/imagenes (archivo).
5. API → Storage: guarda archivo → obtiene **URL pública**.
6. API → DB: INSERT imagen\_libro (id\_libro, url, orden[, is\_portada]).
7. API → Web/App: 201 por imagen.
8. La publicación queda **visible** con sus imágenes.

**Reglas y criterios.**

* Campos obligatorios: título, autor, género, estado, tipo de tapa; **ISBN** válido (único opcional).
* **Permisos**: sólo el propietario puede subir/editar imágenes del libro.
* Límite imágenes (p.ej., **máx. 8**, tamaño y formatos permitidos); la **primera** se marca portada si no se especifica.
* Orden **incremental** y una sola **portada** por libro.
* Validar contenido (mime/antivirus) y aplicar **rate-limit** a cargas.
* Respuestas idempotentes ante reintentos; si falla una imagen no afecta el alta del libro.

### 

### Recuperación de contraseña por token



**Alcance.** Restablecer la contraseña mediante enlace enviado por correo.  
 **Actores.** Usuario, Frontend, API Backend, MySQL, Correo.

**Flujo resumido.**

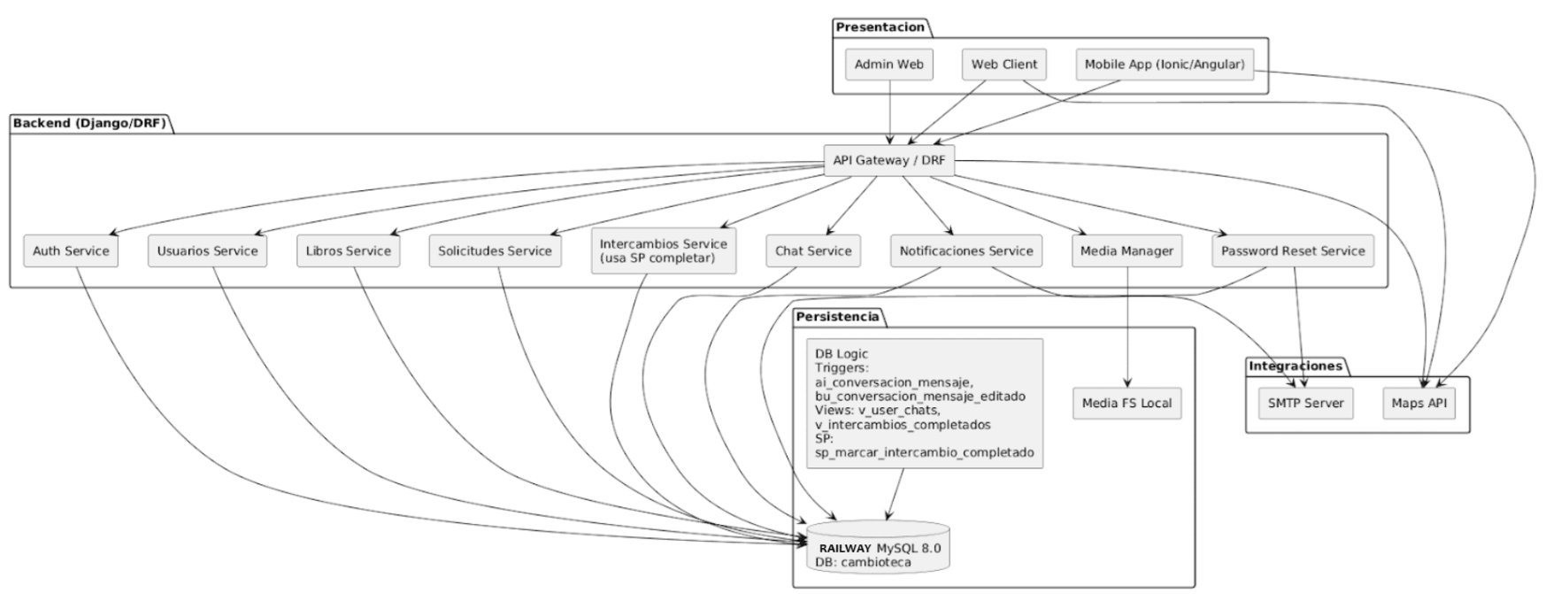
1. Usuario solicita recuperación.
2. **Frontend → API:** POST /auth/forgot (email).
3. **API → DB:** crear password\_reset\_token (user\_id, token, created\_at, expira\_en, used=false).
4. **API → Correo:** envío de enlace firmado con el token (**202 Accepted**).
5. Usuario abre el enlace y define nueva clave.
6. **Frontend → API:** POST /auth/reset {token, new\_password}.
7. **API:** valida token **vigente** y **no usado**; verifica pertenencia al usuario.
8. **API → DB:** actualiza contraseña hasheada, **marca token como usado** y **revoca sesiones** activas del usuario.
9. **API → Frontend:** 200 “contraseña actualizada”.

**Reglas y criterios.**

* **Anti–enumeración:** /auth/forgot siempre responde **202**, exista o no el email (se registra internamente).
* **Token**: aleatorio/no predecible, **un solo uso** y **TTL** (30 min–24 h); opcional almacenar **hash del token**.
* **Seguridad de clave:** política mínima y hash fuerte (Argon2id/bcrypt/PBKDF2 con costo alto).
* **Revocación:** eliminar sesiones (django\_session/tokens) tras el reset; enviar aviso “tu contraseña fue cambiada”.
* **Idempotencia y errores:** si el token ya fue usado o expiró → 410/400; repetir un reset exitoso puede devolver **200** sin cambios.
* **Rate-limit** por IP/email para forgot y reset; trazabilidad (IP, timestamps, resultado).

# Vista de despliegue

## Diagrama de Componentes

Alcance. Arquitectura lógica a nivel de módulos: frontends, capa API y servicios, persistencia y sistemas externos.

Componentes clave.

* Presentación: *Web Client*, *Mobile App (Ionic/Angular)* y *Admin Web* consumen un API Gateway / DRF.
* Servicios (Django/DRF): Auth, Usuarios, Libros, Solicitudes, Intercambios *(usa SP para completar)*, Chat, Notificaciones, Media Manager, Password Reset.
* Persistencia: Railway MySQL 8.0 (db cambioteca) + lógica cercana a datos: *triggers* (ai\_conversacion\_mensaje, bu\_conversacion\_mensaje\_editado), *views* (v\_user\_chats, v\_intercambios\_completados) y SP sp\_marcar\_intercambio\_completado.
* Medios: *Media FS Local* para imágenes de libros.
* Integraciones: SMTP Server (correo/avisos) y Maps API (lugares de intercambio).

Conectividad.

* Todos los clientes entran por el API Gateway, que centraliza autenticación, rate-limit y versionado.
* Cada servicio accede a MySQL vía ORM; Intercambios invoca el SP para ejecutar el cierre de forma atómica.
* Media Manager persiste archivos en Media FS y guarda metadatos en DB.
* Password Reset y Notificaciones publican al SMTP Server; frontends consumen mapas desde Maps API.

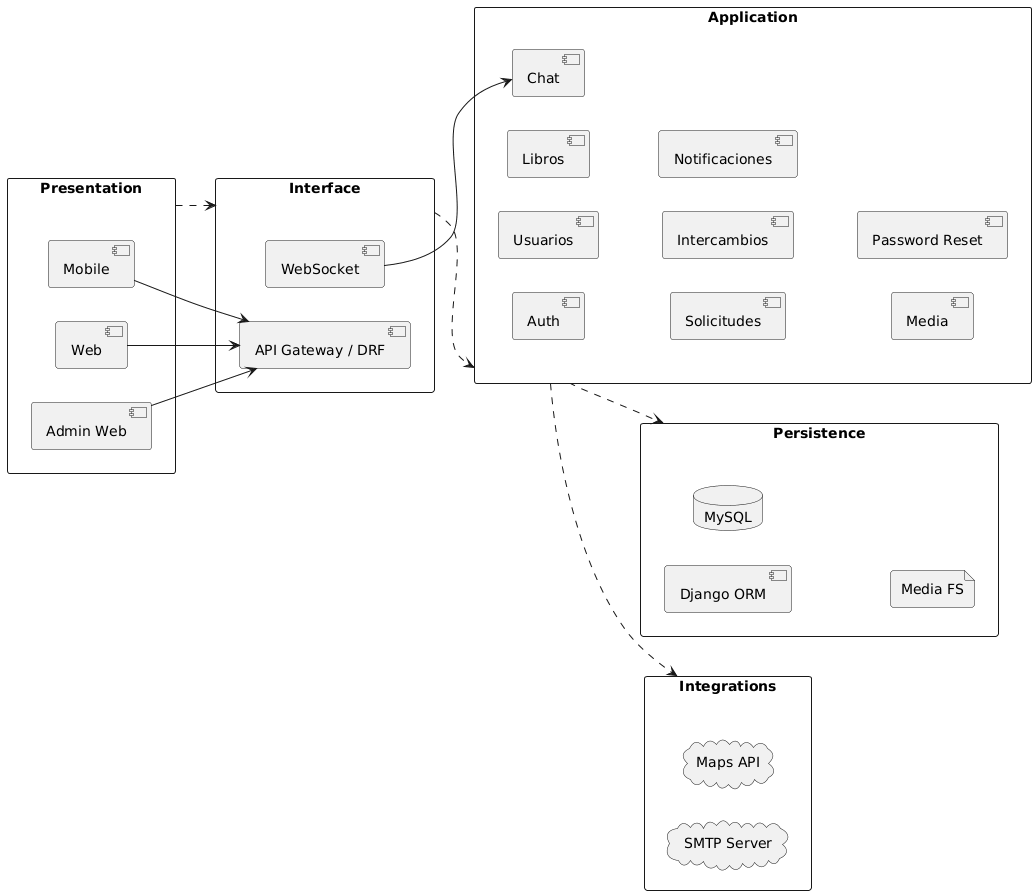
Decisiones de diseño.

* Servicios estateless detrás del gateway para escalar horizontalmente.
* Lógica crítica cerca de los datos (SP/triggers) para garantizar consistencia y simplificar transacciones.
* Chat persiste antes de difundir (evita pérdida de mensajes); notificaciones fuera del camino crítico.
* Separación por dominios funcionales (usuarios, libros, solicitudes, intercambios) que facilita pruebas y mantenimiento.

Evolutivo.

* Media FS puede migrar a S3 o similar sin tocar dominio.
* Posible cacheo de lecturas frecuentes (catálogo) y colas para email/notificaciones.

## Diagrama de Paquetes

****

**Descripción general.** El sistema se organiza en **cinco paquetes** con dependencias **unidireccionales**:  
 **Presentation → Interface → Application → Persistence / Integrations.** Las flechas **sólidas** representan llamadas síncronas (HTTP/WS) y las **discontinuas** dependencias técnicas o invocaciones indirectas.

**Paquetes y responsabilidades**

* **Presentation** (Web, Mobile, Admin Web)  
   Renderiza UI, gobierna navegación y estados de vista. No contiene reglas de negocio ni accede a datos; **siempre** consume la capa *Interface*.
* **Interface** (API Gateway/DRF, WebSocket)  
   Termina peticiones REST y conexiones WS, aplica autenticación/autorización, *rate limits* y validación superficial. Encapsula la firma de endpoints y eventos.  
  + **WebSocket** se usa sólo para mensajería de **Chat** (eventos y *fan-out*).
* **Application** (Auth, Usuarios, Libros, Solicitudes, Intercambios, Chat, Notificaciones, Media, Password Reset)  
   Implementa **casos de uso** y reglas: creación y búsqueda de libros, ciclo de solicitudes/intercambios, mensajería, calificaciones, notificaciones y recuperación de contraseña. Servicios **sin estado**; coordinan transacciones y publican eventos.
* **Persistence** (Django ORM, MySQL, Media FS)  
   Acceso a datos y almacenamiento de ficheros. Los servicios de *Application* acceden a la BD **exclusivamente** a través del ORM. *Media Manager* persiste y lee imágenes desde **Media FS**.
* **Integrations** (SMTP Server, Maps API)  
   Conectores externos. **Notificaciones** y **Password Reset** envían correos vía SMTP. **Intercambios** puede consultar **Maps API** para validaciones de lugar/distancias.

**Reglas de acoplamiento**

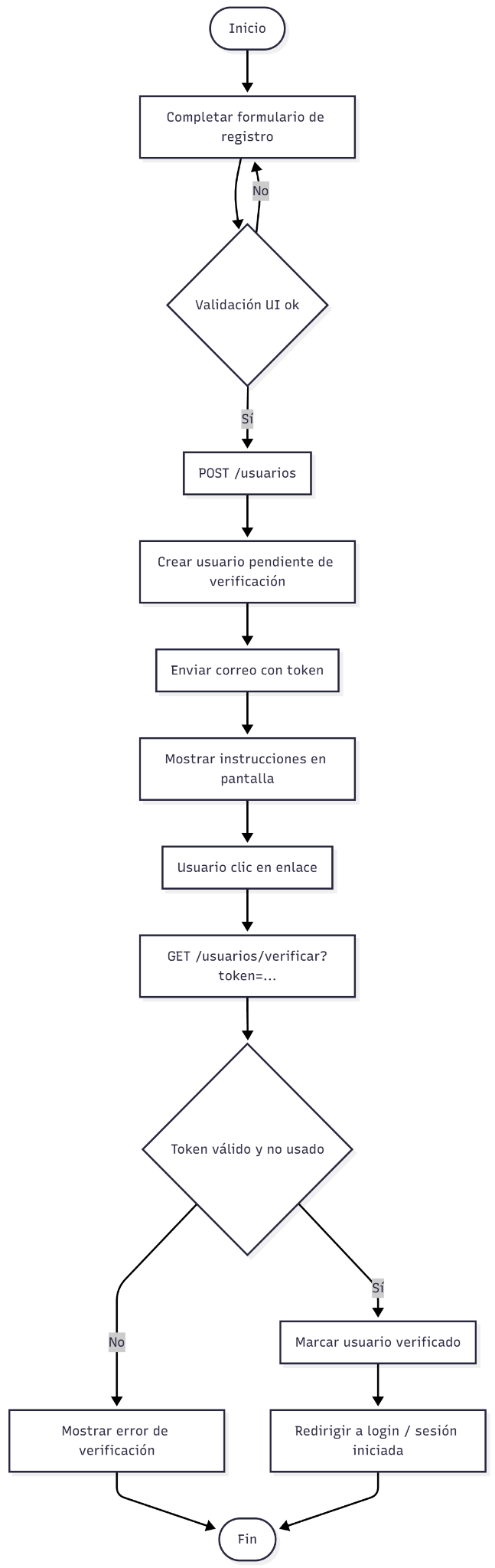
1. UI no llama servicios directamente: **sólo Interface** expone contratos.
2. *Application* no conoce detalles de MySQL/FS/SMTP/Maps; depende de **abstracciones** (ORM/clients).
3. Las validaciones de negocio y transacciones viven en **Application**; los *triggers* y SPs de apoyo permanecen en **Persistence**.
4. Cualquier nuevo módulo debe insertarse como servicio en *Application* y publicarse por *Interface*.

**Beneficios** Separa responsabilidades, reduce acoplamiento, facilita pruebas y despliegue (cada paquete puede escalarse o evolucionar de forma independiente) y mantiene claros los puntos de extensión (nuevos servicios o integraciones).

# Vista de procesos

## Diagrama de Actividad

### Registro y verificación de cuenta



El diagrama describe el alta y verificación de cuenta. El usuario completa el formulario en la web/app; la interfaz valida formato y obligatoriedad de los campos y, si todo está correcto, envía un POST al backend. La API crea el registro con estado “pendiente”, genera un token de verificación con expiración y lo persiste, y a continuación solicita al servicio de correo el envío del enlace. En pantalla se muestran instrucciones para continuar. Cuando el usuario hace clic en el enlace, la aplicación invoca el endpoint de verificación; si el token es válido, vigente y no usado, la API marca la cuenta como verificada, invalida el token y redirige a inicio de sesión (o abre sesión). Si el token es inválido o caducó, se informa el error y se ofrece reintentar.

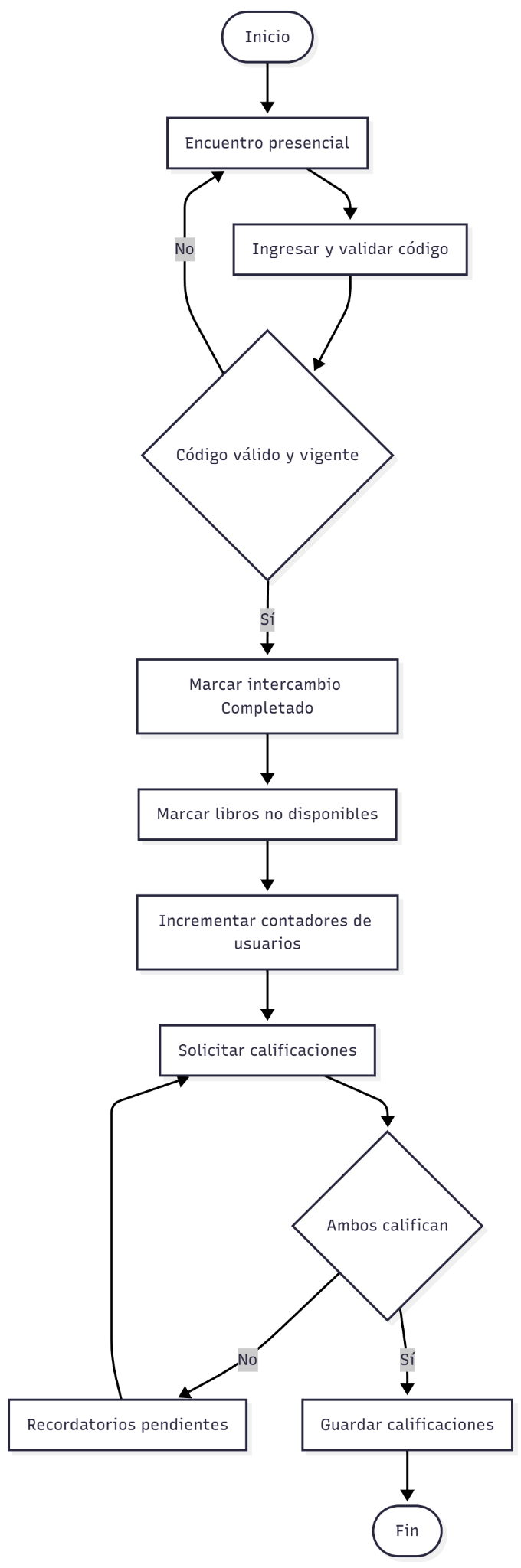
En cuanto a políticas, se exige correo único y contraseña conforme a la política definida; el token es aleatorio, de un solo uso y con vencimiento, y la verificación es idempotente (si la cuenta ya está verificada, se responde sin duplicar efectos). Se registran eventos relevantes (IP/UA y timestamps) y se aplican límites de tasa a creación de cuentas y reenvíos de correo. A nivel de experiencia, se muestran validaciones en vivo, confirmación de cuenta verificada y opción de “reenviar verificación” cuando corresponda.

### Solicitud de intercambio y aceptación

El flujo representa la gestión de una solicitud de intercambio. El Usuario A abre la ficha del libro publicado por B y envía la solicitud (POST /solicitudes). El sistema registra la solicitud con estado *Pendiente* y notifica a B (bandeja de “recibidas” y push/email). B entra a revisar y decide: ¿aceptar o no?

Si rechaza, la solicitud pasa a *Rechazada* y se notifica a A con el motivo (si se indicó).  
 Si acepta, la solicitud cambia a *Aceptada*, el backend crea el Intercambio ligado a la solicitud y abre la Conversación (chat) entre A y B para coordinar; ambos reciben notificación y el intercambio queda en estado “en coordinación” hasta pactar fecha y lugar.

### Pactar fecha y lugar, generar código



El flujo comienza en el encuentro presencial. Cualquiera de las partes ingresa el código único del intercambio; el backend lo valida (pertenece al intercambio, no expirado, no usado). Si la validación falla, el intercambio permanece “en coordinación”. Si el código es válido y vigente, el sistema marca el intercambio como Completado, registra la fecha, pone ambos libros como no disponibles y actualiza los contadores de intercambio de los usuarios. Esta transición se realiza de forma atómica (vía POST /intercambios/{id}/completar, que invoca el SP sp\_marcar\_intercambio\_completado) y es idempotente: reintentos no duplican efectos.

Al completar, se solicitan calificaciones a ambos participantes (push/email) y quedan pendientes hasta su envío. El flujo termina cuando ambos califican: se persisten las entradas de calificacion (una por usuario por intercambio, respetando unicidad), se recalculan promedios y se cierra la tarea. Si alguno no califica, el sistema emite recordatorios periódicos hasta el límite definido; una vez recibidas todas las calificaciones —o alcanzado el vencimiento— el proceso concluye.

### Búsqueda y visualización de detalle de libro

El flujo comienza cuando el usuario ingresa texto libre, filtros (género, estado, etc.) y orden en la interfaz. La aplicación invoca GET /libros con esos parámetros y paginación. Si la consulta no devuelve filas, se muestra un estado vacío con sugerencias (ajustar filtros, limpiar búsqueda). Cuando hay resultados, el cliente renderiza la lista junto con el paginador (número de página, tamaño y total). Al seleccionar un elemento, se ejecuta GET "/libros/{id}" y se presenta la ficha completa del libro: metadatos, imágenes, disponibilidad y acciones asociadas (por ejemplo, ofertar/intercambiar o añadir a favoritos).

El proceso contempla validación básica de entradas (filtros y orden válidos), límites de paginación razonables, ordenamiento estable para evitar saltos entre páginas y manejo de errores de red/servidor con mensajes claros. El backend optimiza la respuesta paginada (consulta de datos + conteo total) y entrega información suficiente para construir la navegación. Con ello, el usuario puede explorar el catálogo eficientemente y profundizar en el detalle de un libro concreto sin abandonar el contexto de búsqueda.

### Publicación y edición de libro con imágenes

El diagrama modela el flujo completo para publicar un libro y para editar una publicación existente. En la rama *Publicar*, el usuario completa los metadatos (título, autor, género, estado, etc.) y la UI valida los campos; si la validación falla, vuelve a corrección. Con validación OK se ejecuta POST /libros y se crea el registro. Luego, opcionalmente, el usuario sube imágenes: selecciona archivo, se valida formato y tamaño, y si es válido se envía POST /libros/:id/imagenes. El backend guarda el archivo en el almacén de medios, registra en BD la URL, orden y portada, y el usuario puede repetir hasta no agregar más. Al terminar, la publicación queda visible.

En la rama *Editar*, el usuario puede modificar metadatos, reordenar imágenes, cambiar portada o eliminar archivos. Los cambios pasan por validación; si están correctos, el sistema aplica las operaciones PATCH/DELETE correspondientes y actualiza la publicación, manteniendo la visibilidad. El flujo enfatiza controles de integridad (validaciones UI/servidor, formatos/peso permitidos, límite de imágenes, una sola portada y orden consistente) y tolerancia a fallos: un error subiendo una imagen no revierte el alta del libro y permite reintentar.

# Vista Fisica

## Diagrama de Despliegue

Arquitectura. La solución se organiza en capas dentro de una red privada en la nube. La única pieza pública es un balanceador/reverse proxy (HTTPS) que recibe todo el tráfico y lo distribuye hacia la API (Django/DRF) y, cuando corresponde, a un gateway WebSocket para mensajería en tiempo real. En la red privada residen también los workers de tareas (Celery) y Redis, usado como caché y cola de trabajos. La base de datos MySQL gestionada y el almacén de medios tipo S3 completan la capa de persistencia: datos estructurados en MySQL; imágenes/archivos fuera del servidor de aplicación.

Flujo de tráfico. Usuario web/móvil → HTTPS → balanceador. Este enruta a la API (resto de operaciones) o a WebSocket (chat en vivo). La API lee/escribe en MySQL y publica/consulta archivos en S3. Operaciones no interactivas (envío de correos, notificaciones, procesamiento de imágenes) se colocan en Redis y las ejecutan los workers sin bloquear la respuesta al usuario.

Seguridad y operación. Solo el balanceador es accesible desde Internet; el resto de los servicios operan en red privada. Todo el tráfico externo viaja cifrado (HTTPS). Secretos y credenciales se gestionan por variables de entorno/secret manager. La base de datos cuenta con copias de seguridad automáticas y monitoreo del proveedor. El diseño es escalable horizontalmente: se pueden añadir instancias de aplicación sin cambios en clientes, manteniendo datos y archivos centralizados en MySQL/S3.

Beneficios. Separación clara de responsabilidades, menor exposición de activos sensibles, mejor experiencia en tiempo real (WebSocket) y capacidad de crecimiento progresivo (API y workers escalan de forma independiente). Esta topología es una evolución natural desde un VPS único hacia un entorno más robusto, manteniendo la complejidad bajo control.

# Vista de Escenario

## Diagrama de Caso de uso

### Cuenta y seguridad

**Actores:** Visitante (no autenticado) y Usuario (autenticado).  
 Alcance: registro, login/logout, recuperación de acceso y gestión de perfil.

**Casos clave:**

* Registrar Usuario *(incluye → Verificar Email)*: crea y activa la cuenta.
* Iniciar Sesión: valida credenciales; requiere email verificado.
* Cerrar Sesión: invalida sesión/tokens.
* Recuperar contraseña *(incluye → Enviar Email de Reset)*: genera enlace temporal y actualiza clave.
* Cambiar contraseña: desde perfil; exige clave actual y fuerza re-login.

* Editar Perfil y Ubicación: actualiza datos y guarda.

**Reglas:** email verificado obligatorio; política de contraseñas (≥8, mixtas, hash); tokens/enlaces con expiración y registro de auditoría.

### Intercambio End to End

**Actores:** Usuario Solicitante, Usuario Receptor y Servicio de Notificaciones.  
 Objetivo: concretar el intercambio de un libro y cerrar la operación con actualización de inventario y reputación.

**Flujo base:**

1. Aceptar oferta → Crear Intercambio *(pre: estado “Aceptado”; post: contrapartes notificadas)*.
2. Crear conversación y enviar mensajes.
3. Pactar fecha y lugar (con opción de reprogramar).
4. En el encuentro: generar y validar código (uso único).
5. Completar intercambio → actualizar stock/contadores y notificar para calificar.
6. Cada parte califica al usuario contraparte.

**Alternativas:** Cancelar intercambio antes de completar; si el código no es válido, no se cierra.  
 Efectos: notificaciones automáticas, inventario actualizado y registro de eventos/auditoría.

### Publicar Libro

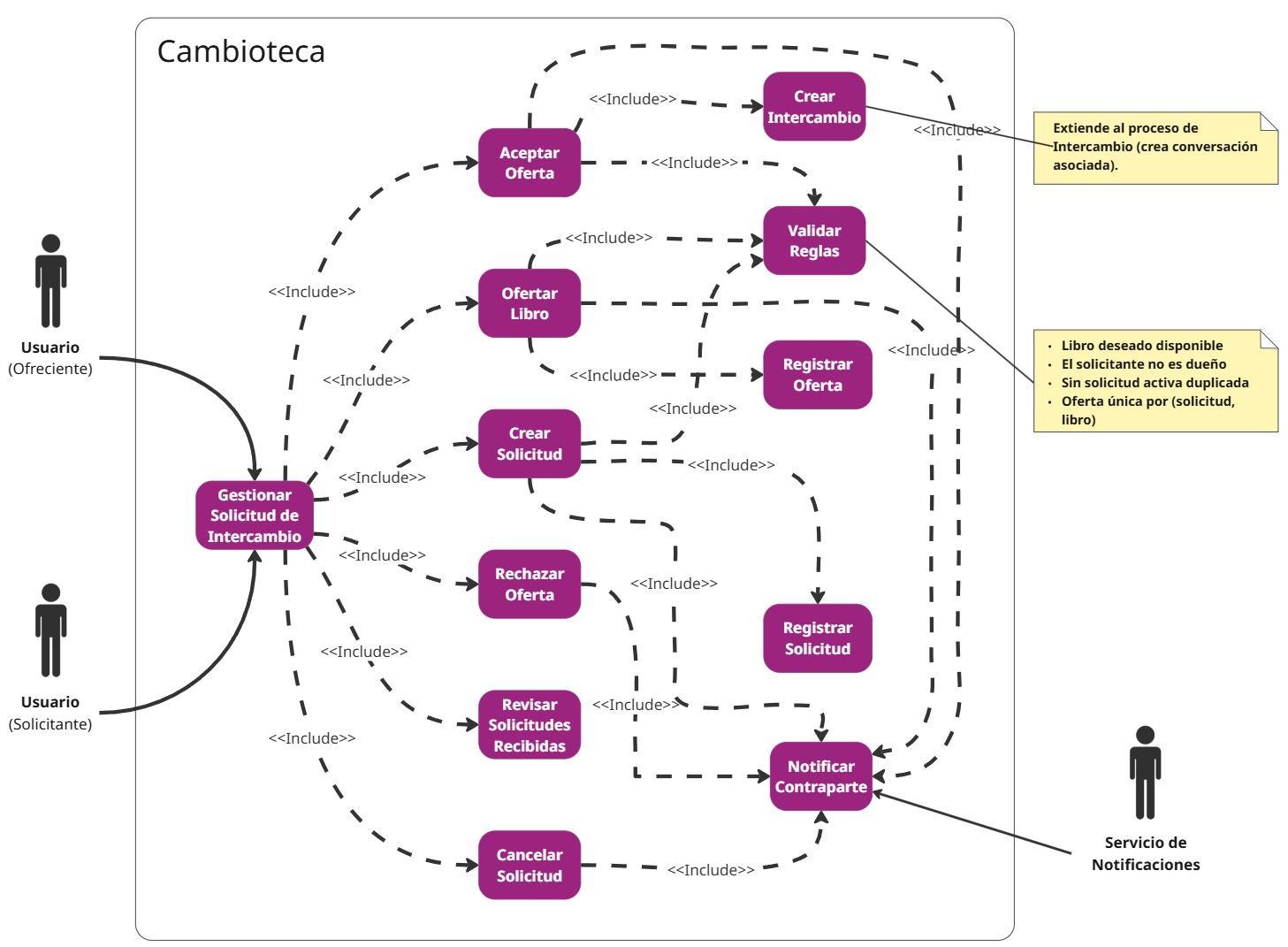
**Actor:** Usuario (autenticado).  
 Objetivo: crear una publicación visible en el catálogo.

**Flujo base:**

1. (Opcional) Gestionar/Subir/Reordenar imágenes — hasta 8 (.jpg/.png/.webp), una portada y orden sin huecos.
2. Publicar Libro → *incluye* Validar metadatos (título, autor, género válido, año razonable, ISBN opcional válido 10/13, estado y encuadernado) y Persistir libro (guardar registro e imágenes).

**Resultado:** publicación creada con disponible = 1 y visible en el catálogo.

### Gestionar solicitud de intercambio



**Actores:** Usuario **Solicitante**, Usuario **Oferente**, **Servicio de Notificaciones**.  
**Objetivo:** tramitar una solicitud hasta crear el intercambio o cerrarla (rechazo/cancelación).

**Reglas (validación):** libro disponible, el solicitante no es dueño, sin solicitud activa duplicada, **oferta única** por solicitud/libro.

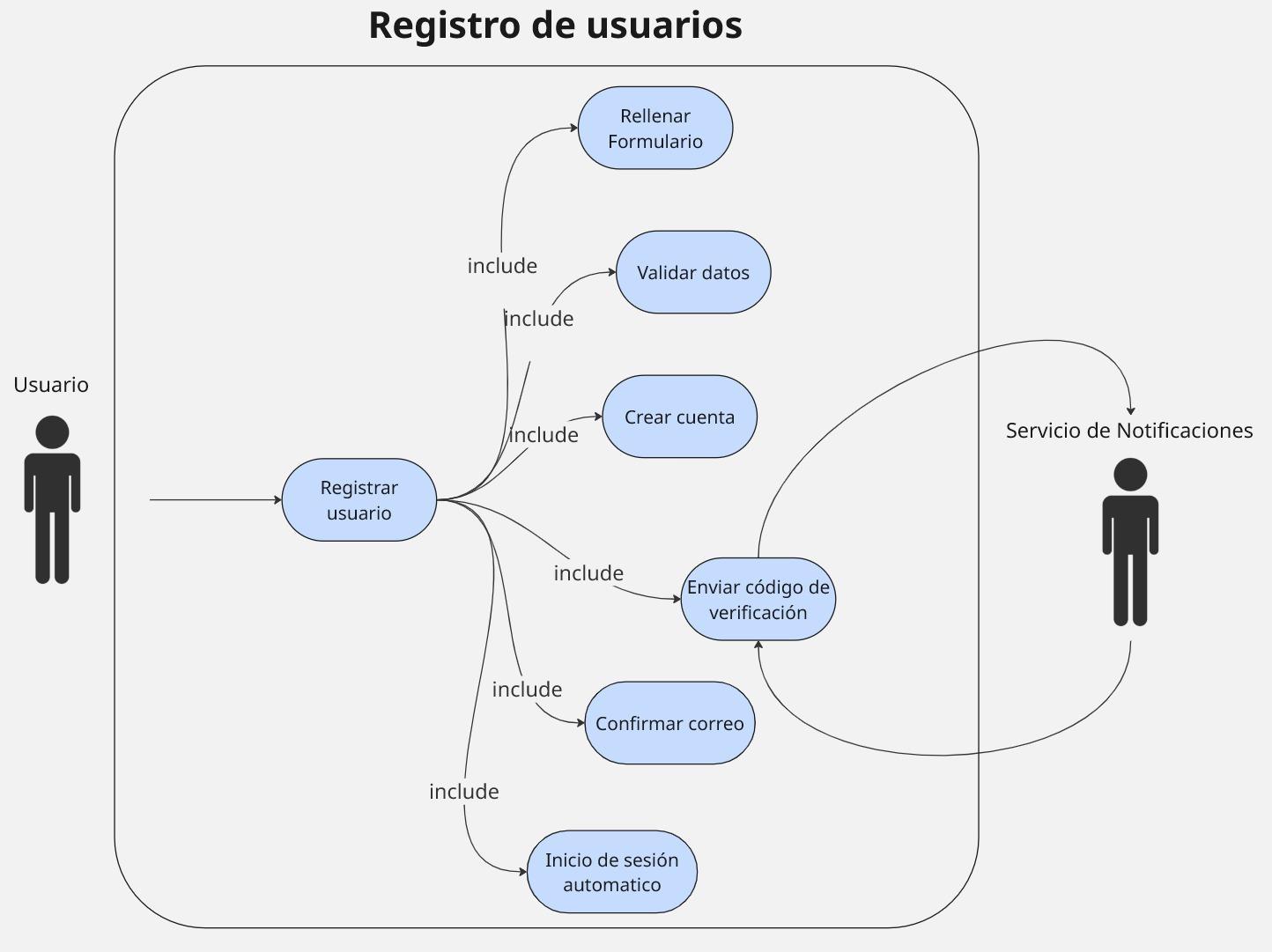
**Flujo base:**

1. **Crear Solicitud** → *incluye* **Validar Reglas** → **Registrar Solicitud** → **Notificar Contraparte**.
2. **Revisar Solicitudes Recibidas** (oferente).
3. Decisión del oferente:  
   * **Aceptar Oferta** → *incluye* **Registrar Oferta** → **Validar Reglas** → **Crear Intercambio** *(extiende al proceso de Intercambio y crea conversación)*.
   * **Ofertar Libro** (contrapropuesta) → *incluye* **Registrar Oferta** → **Validar Reglas**.
   * **Rechazar Oferta**.

**Alternativas:** **Cancelar Solicitud** (antes de aceptación); si fallan reglas, se rechaza automáticamente.

**Resultado:** **Intercambio creado** con conversación asociada **o** **solicitud cerrada**; contrapartes notificadas y eventos registrados.

### Registro de Usuarios

****

**Actores:** Usuario; Servicio de Notificaciones.  
**Objetivo:** crear y activar una cuenta.

**Flujo base:**

1. Rellenar formulario → 2) Validar datos → 3) Crear cuenta (pendiente) →
2. Enviar código de verificación (notificación) → 5) Confirmar correo (activa) →
3. *(Opcional)* Inicio de sesión automático.

**Reglas clave:** email válido y único; contraseña conforme a política.  
**Resultado:** cuenta activa y acceso habilitado.