INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTACIÓN

Sistema de búsqueda y recomendación de libros

Unidad de aprendizaje:

Ingeniería de Software

6CV3

Elaborado por:

Luis Gerardo Herrera Avila

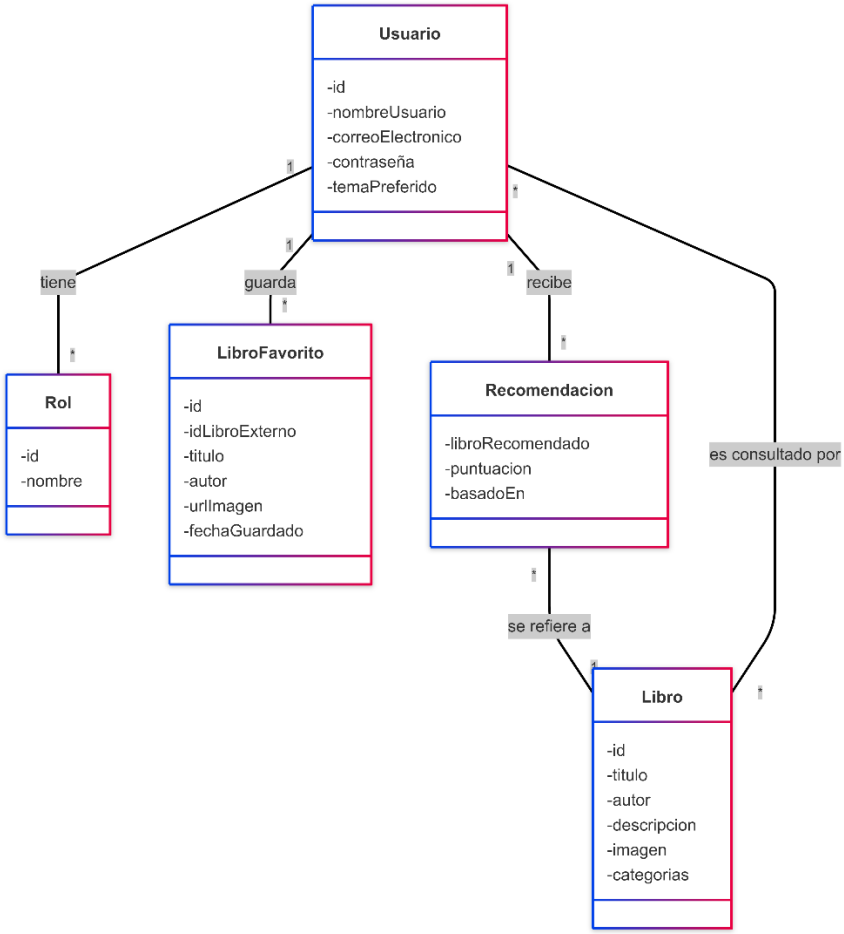
Profesora:

Gabriel Hurtado Avilés

28 de Abril de 2025

Índice

Diagrama de Clases Conceptuales



Entidades y sus Relaciones con Cardinalidades

Usuario

1. **Usuario - Rol** [1..1 a 0..*]
    - Un Usuario tiene exactamente un (1) Rol
    - Un Rol puede estar asignado a cero o muchos (0..*) Usuarios
2. **Usuario - LibroFavorito** [1..1 a 0..*]
    - Un Usuario puede guardar cero o muchos (0..*) LibroFavorito
    - Un LibroFavorito pertenece exactamente a un (1) Usuario
3. **Usuario - Recomendacion** [1..1 a 0..*]

- Un Usuario puede recibir cero o muchas (0..*) Recomendaciones
- Una Recomendacion está asociada exactamente a un (1) Usuario

4. **Usuario - Libro** [0..* a 0..*]
    - Un Usuario puede consultar cero o muchos (0..*) Libros
    - Un Libro puede ser consultado por cero o muchos (0..*) Usuarios

Rol

1. **Rol - Usuario** [0..* a 1..1]
    - Un Rol puede ser asignado a cero o muchos (0..*) Usuarios
    - Un Usuario tiene exactamente un (1) Rol

LibroFavorito

1. **LibroFavorito - Usuario** [1..1 a 0..*]
    - Un LibroFavorito pertenece exactamente a un (1) Usuario
    - Un Usuario puede tener cero o muchos (0..*) LibroFavorito
2. **LibroFavorito - Libro** [1..1 a 0..*]
    - Un LibroFavorito referencia exactamente a un (1) Libro
    - Un Libro puede ser referenciado por cero o muchos (0..*) LibroFavorito

Recomendación

1. **Recomendacion - Usuario** [1..1 a 0..*]
    - Una Recomendacion está asociada exactamente a un (1) Usuario
    - Un Usuario puede recibir cero o muchas (0..*) Recomendaciones
2. **Recomendacion - Libro** [1..* a 0..*]
    - Una Recomendacion se refiere a uno o muchos (1..*) Libros
    - Un Libro puede estar incluido en cero o muchas (0..*) Recomendaciones

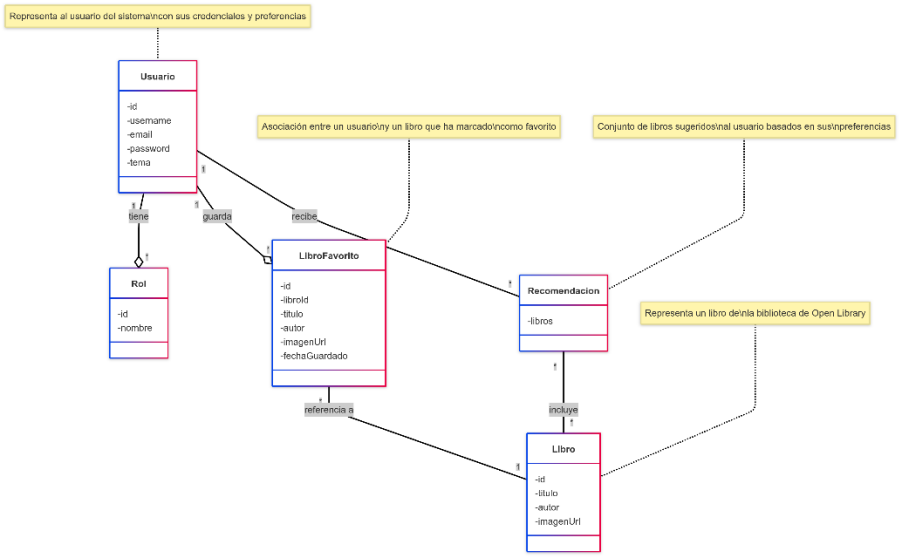
Libro

1. **Libro - LibroFavorito** [0..* a 1..1]
    - Un Libro puede ser referenciado por cero o muchos (0..*) LibroFavorito
    - Un LibroFavorito referencia exactamente a un (1) Libro
2. **Libro - Recomendacion** [0..* a 1..*]
    - Un Libro puede estar incluido en cero o muchas (0..*) Recomendaciones
    - Una Recomendacion incluye uno o muchos (1..*) Libros
3. **Libro - Usuario** [0..* a 0..*]
    - Un Libro puede ser consultado por cero o muchos (0..*) Usuarios
    - Un Usuario puede consultar cero o muchos (0..*) Libros

Notas sobre las Cardinalidades

1. La relación entre Usuario y Rol se muestra en el diagrama como una relación directa (uno a uno), aunque en la implementación física de la base de datos (tercera imagen) se utiliza una tabla de unión (usuario_roles) para permitir la asignación de múltiples roles a un usuario.
2. La relación "es consultado por" entre Libro y Usuario representa una relación muchos a muchos que probablemente se implementaría mediante una tabla de unión en el esquema físico.
3. La relación entre Recomendacion y Libro se establece con cardinalidad mínima de 1 para Recomendacion, ya que una recomendación debe incluir al menos un libro para ser útil.
4. Las cardinalidades reflejan tanto las restricciones técnicas como las reglas de negocio establecidas en el modelo de dominio.

Modelo de Dominio



Reglas de Negocio por Entidad

Entidad: Usuario

1. Cada usuario debe tener un nombre de usuario único en el sistema.
2. El correo electrónico debe ser único para cada usuario y tener un formato válido.
3. Las contraseñas deben almacenarse de forma segura (encriptadas).
4. Un usuario puede tener solo un rol asignado a la vez.
5. Un usuario puede guardar múltiples libros como favoritos.
6. Un usuario puede recibir múltiples recomendaciones basadas en sus preferencias.
7. El tema preferido es opcional pero sirve como base para el sistema de recomendación.

Entidad: Rol

1. Cada rol debe tener un nombre único y descriptivo.
2. Los roles definen los niveles de acceso y permisos dentro del sistema.
3. Un rol puede ser asignado a múltiples usuarios.
4. Los roles básicos del sistema incluyen al menos "Usuario" y "Administrador".

Entidad: LibroFavorito

1. Un usuario puede marcar múltiples libros como favoritos.
2. Un libro puede ser marcado como favorito por múltiples usuarios.
3. La fecha de guardado se establece automáticamente al momento de marcar un libro como favorito.

4. La información básica del libro (título, autor, imagen) se almacena redundantemente por eficiencia.
5. Cada registro de LibroFavorito debe estar asociado a un usuario existente.
6. Debe existir una referencia a un libro real en el sistema de Open Library.

Entidad: Libro

1. Cada libro debe tener un identificador único proveniente de la biblioteca de Open Library.
2. La información del libro (título, autor) es obligatoria.
3. La imagen y la descripción son opcionales pero deseables para la interfaz de usuario.
4. Las categorías ayudan al sistema de recomendación y deben ser consistentes.
5. Los libros son entidades de solo lectura que reflejan datos del API externo.

Entidad: Recomendacion

1. Cada recomendación está asociada a un usuario específico.
2. Una recomendación puede incluir múltiples libros.
3. Las recomendaciones se generan basadas en las preferencias del usuario y sus libros favoritos.
4. El atributo "basadoEn" indica el criterio utilizado para generar la recomendación.
5. La puntuación representa el nivel de coincidencia entre el libro recomendado y los gustos del usuario.

Invariantes del Sistema

1. **Integridad referencial**: Todas las referencias entre entidades deben ser válidas (FK → PK existente).
2. **Unicidad de identidad**: Los usuarios, roles y libros deben ser identificables de manera única.
3. **Consistencia de datos de usuario**: El correo electrónico y nombre de usuario no pueden ser duplicados.
4. **Integridad de autenticación**: Los usuarios solo pueden acceder con credenciales válidas.
5. **Consistencia de roles**: Todo usuario debe tener asignado al menos un rol válido.
6. **Persistencia de favoritos**: Un libro favorito debe mantener su relación con el usuario y el libro original.
7. **Coherencia de recomendaciones**: Las recomendaciones deben estar basadas en datos reales del usuario.
8. **Trazabilidad**: Toda acción del usuario que modifique datos debe ser registrada con fecha y hora.

Restricciones y Condiciones de los Datos

Restricciones de Integridad

1. **Obligatoriedad**: Los campos marcados como no nulos deben contener valores válidos.
    - Usuario: id, nombreUsuario, correoElectronico, contraseña
    - Rol: id, nombre
    - LibroFavorito: id, idLibroExterno, titulo, autor, fechaGuardado, usuario_id
    - Libro: id, titulo, autor
    - Recomendacion: id, libroRecomendado
2. **Unicidad**: Los campos marcados como únicos no pueden tener valores duplicados.
    - Usuario: nombreUsuario, correoElectronico
    - Rol: nombre
3. **Restricciones de valor**:
    - Las contraseñas deben tener una longitud mínima de 8 caracteres.
    - Los correos electrónicos deben seguir un formato estándar (xxx@xxx.xxx).
    - Las fechas deben estar en formato válido y no pueden ser futuras.

Restricciones de Dominio

1. **Valores admisibles**:
    - El campo tema del usuario debe corresponder a categorías predefinidas de libros.
    - Los roles deben ser seleccionados de una lista predefinida.
    - La puntuación de recomendación debe estar en un rango de 0 a 5.
2. **Validación de formato**:
    - Las URL de imágenes deben ser válidas y accesibles.
    - Los identificadores externos de libros deben seguir el formato de Open Library.

Restricciones de Negocio

1. **Operaciones permitidas**:
    - Solo los usuarios autenticados pueden guardar libros como favoritos.
    - Las recomendaciones se generan automáticamente basadas en algoritmos predefinidos.
    - Los usuarios solo pueden ver y modificar sus propios datos y favoritos.
    - Solo los administradores pueden gestionar roles y usuarios.
2. **Restricciones temporales**:
    - Las recomendaciones deben actualizarse periódicamente según la actividad del usuario.
    - Los datos de libros deben sincronizarse con Open Library regularmente.

Restricciones de Seguridad

1. **Protección de datos**:
    - Las contraseñas nunca se almacenan en texto plano.
    - Los datos personales deben estar protegidos según normativas de privacidad.
    - El acceso a la información debe estar limitado según el rol del usuario.
2. **Trazabilidad**:

- Todas las operaciones de modificación de datos deben registrarse.
- El sistema debe mantener un registro de accesos y actividades clave.

Este modelo de dominio proporciona una comprensión integral de las reglas, invariantes y restricciones que gobiernan el sistema de recomendación de libros, asegurando la integridad y consistencia de los datos a lo largo de todas las operaciones.

Diccionario de Datos

Entidad: Usuario

Nombre de la entidad: Usuario
Descripción y propósito: Representa a un usuario registrado en el sistema con sus credenciales y preferencias de lectura.

Atributo	Tipo de Dato	Descripción	Restricciones
id	BIGINT	Identificador único del usuario	PK, Autoincrementable, No nulo
nombreUsuario/username	VARCHAR	Nombre de usuario para acceso al sistema	Único, No nulo
correoElectronico/email	VARCHAR	Correo electrónico del usuario	Único, No nulo
contraseña/password	VARCHAR	Contraseña encriptada del usuario	No nulo
temaPreferido/tema	VARCHAR	Tema o categoría preferida de lectura	Opcional

Relaciones:

- Usuario **tiene** Rol (Relación uno a uno)
- Usuario **guarda** LibroFavorito (Relación uno a muchos)
- Usuario **recibe** Recomendacion (Relación uno a muchos)
- Usuario **es consultado por** Libro (Relación muchos a muchos)

Entidad: Rol

Nombre de la entidad: Rol
Descripción y propósito: Define los diferentes tipos de roles y niveles de acceso en el sistema.

Atributo	Tipo de Dato	Descripción	Restricciones

id	BIGINT	Identificador único del rol	PK, Autoincrementable, No nulo
nombre	VARCHAR	Nombre descriptivo del rol (ej. "Administrador", "Usuario")	Único, No nulo

Relaciones:

- Rol **es asignado a** Usuario (Relación uno a muchos)

Entidad: LibroFavorito

Nombre de la entidad: LibroFavorito
Descripción y propósito: Asociación entre un usuario y un libro que ha marcado como favorito.

Atributo	Tipo de Dato	Descripción	Restricciones
id	BIGINT	Identificador único del registro	PK, Autoincrementable, No nulo
idLibroExterno/libroId	VARCHAR	Identificador externo del libro	No nulo, FK (referencia a Libro)
titulo	VARCHAR	Título del libro guardado como favorito	No nulo
autor	VARCHAR	Autor del libro guardado como favorito	No nulo
urlImagen/imagenUrl	VARCHAR	URL de la imagen de portada del libro	Opcional
fechaGuardado	TIMESTAMP	Fecha y hora cuando el libro fue guardado como favorito	No nulo, Valor por defecto: fecha actual
usuario_id	BIGINT	Identificador del usuario que guardó el libro	FK (referencia a Usuario), No nulo

Relaciones:

- LibroFavorito **referencia a** Libro (Relación muchos a uno)
- LibroFavorito **es guardado por** Usuario (Relación muchos a uno)

Entidad: Libro

Nombre de la entidad: Libro
Descripción y propósito: Representa un libro de la biblioteca de Open Library.

Atributo	Tipo de Dato	Descripción	Restricciones

id	VARCHAR	Identificador único del libro	PK, No nulo
titulo	VARCHAR	Título del libro	No nulo
autor	VARCHAR	Autor o autores del libro	No nulo
descripcion	TEXT	Descripción o sinopsis del libro	Opcional
imagen/imagenUrl	VARCHAR	URL de la imagen de portada	Opcional
categorias	VARCHAR	Categorías o géneros a los que pertenece el libro	Opcional

Relaciones:

- Libro **es consultado por** Usuario (Relación muchos a muchos)
- Libro **es referenciado por** LibroFavorito (Relación uno a muchos)
- Libro **es incluido en** Recomendacion (Relación muchos a muchos)

Entidad: usuario_roles

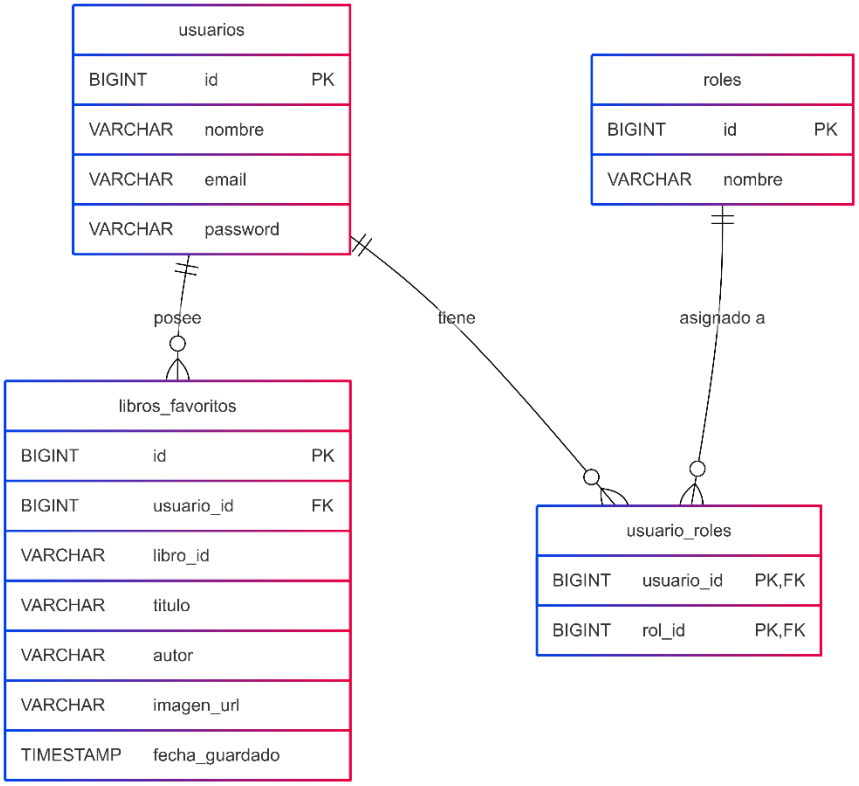
Nombre de la entidad: usuario_roles
Descripción y propósito: Tabla de unión que relaciona usuarios con sus roles asignados.

Atributo	Tipo de Dato	Descripción	Restricciones
usuario_id	BIGINT	Identificador del usuario	PK, FK (referencia a Usuario), No nulo
rol_id	BIGINT	Identificador del rol	PK, FK (referencia a Rol), No nulo

Relaciones:

- Implementa la relación muchos a muchos entre Usuario y Rol

Diagrama Entidad-Relación



Relación entre usuarios y libros_favoritos

- **Tipo de relación:** Uno a muchos (1)
- **Descripción:** Un usuario puede tener múltiples libros favoritos, pero cada libro favorito pertenece a un solo usuario.
- **Implementación técnica:** La tabla libros_favoritos contiene una clave foránea usuario_id que referencia la clave primaria id de la tabla usuarios.
- **Cardinalidad:** Un usuario (1) puede poseer muchos (N) libros favoritos.
- **Restricciones:** La integridad referencial debe asegurar que el usuario_id en libros_favoritos corresponda a un usuario existente en la tabla usuarios.

Relación entre usuarios y usuario_roles

- **Tipo de relación:** Uno a muchos (1)
- **Descripción:** Un usuario puede tener asignados múltiples roles a través de la tabla de unión usuario_roles.

- **Implementación técnica:** La tabla usuario_roles contiene una clave foránea usuario_id que referencia la clave primaria id de la tabla usuarios.
- **Cardinalidad:** Un usuario (1) puede tener muchos (N) registros en la tabla de unión.
- **Restricciones:** La integridad referencial debe asegurar que el usuario_id en usuario_roles corresponda a un usuario válido en la tabla usuarios.

Relación entre roles y usuario_roles

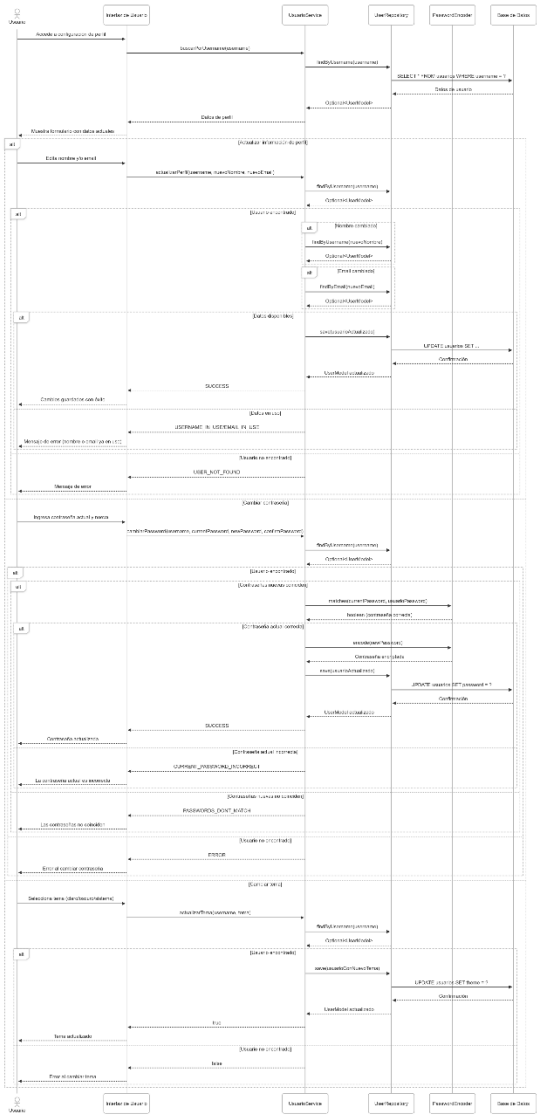
- **Tipo de relación:** Uno a muchos (1)
- **Descripción:** Un rol puede ser asignado a múltiples usuarios a través de la tabla de unión usuario_roles.
- **Implementación técnica:** La tabla usuario_roles contiene una clave foránea rol_id que referencia la clave primaria id de la tabla roles.
- **Cardinalidad:** Un rol (1) puede estar asignado a muchos (N) registros en la tabla de unión.
- **Restricciones:** La integridad referencial debe asegurar que el rol_id en usuario_roles corresponda a un rol válido en la tabla roles.

Relación entre usuarios y roles

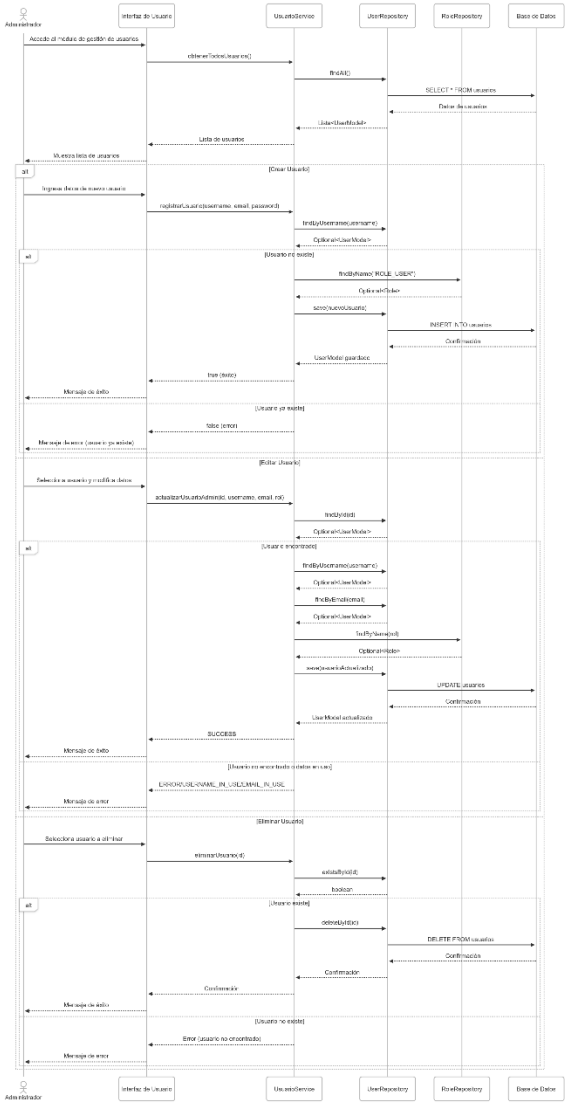
- **Tipo de relación:** Muchos a muchos (M)
- **Descripción:** Un usuario puede tener múltiples roles, y un rol puede ser asignado a múltiples usuarios.
- **Implementación técnica:** La relación se implementa mediante la tabla de unión usuario_roles, que contiene claves foráneas a ambas tablas.
- **Cardinalidad:** Muchos usuarios (M) pueden tener muchos roles (N).

Diagramas de Secuencia

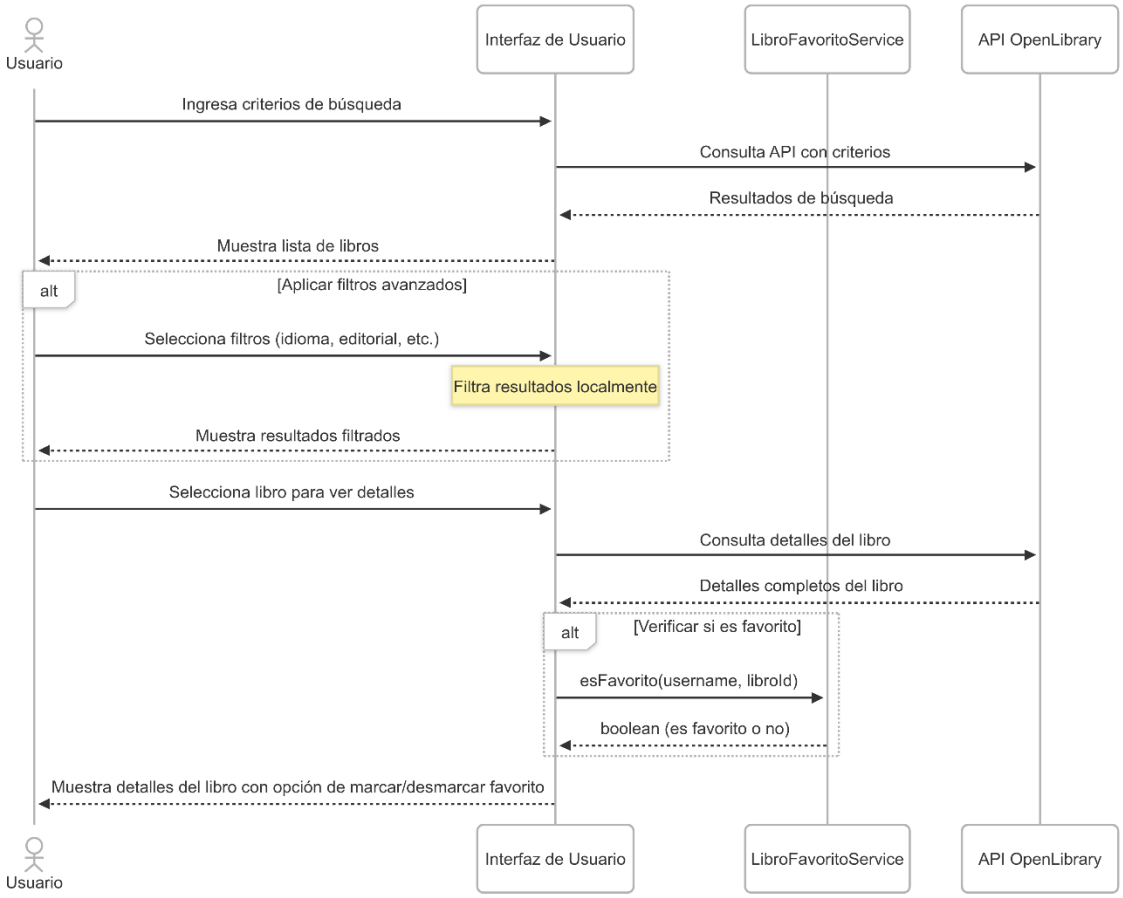
Gestión de Perfil



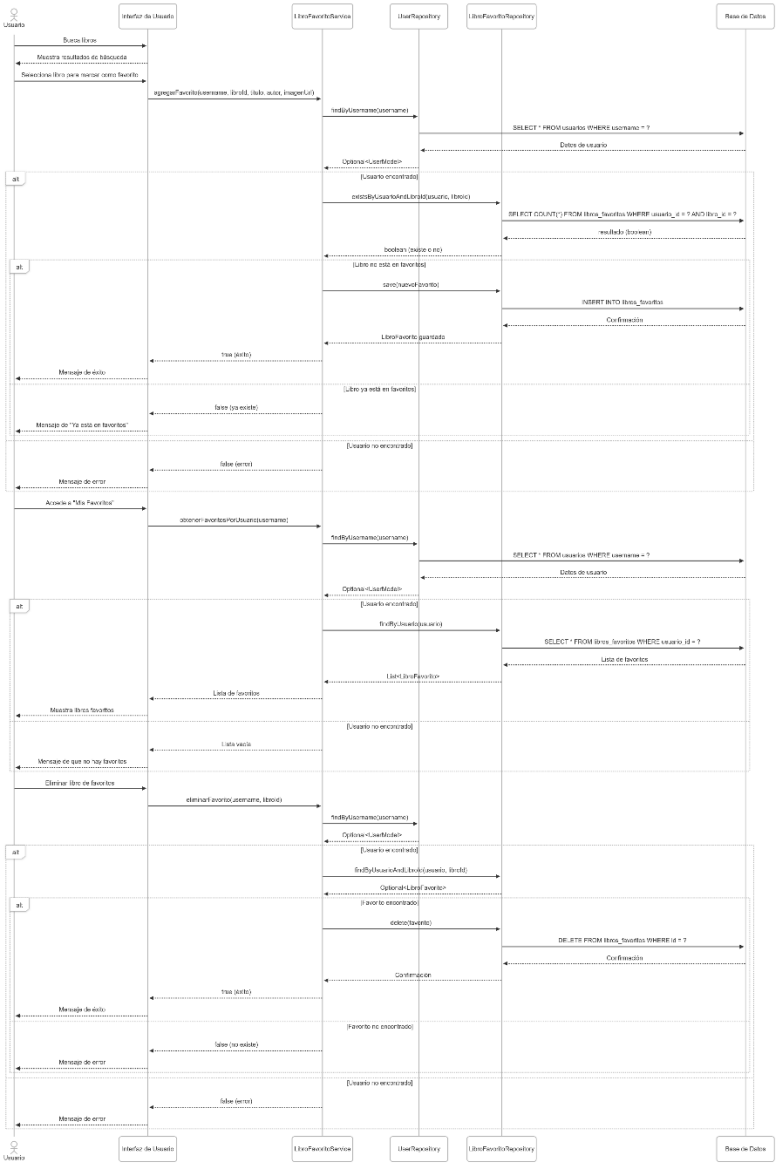
Gestión de Usuarios


Administrador
Interfaz de Usuario
UsuarioService
UserRepository
RoleRepository
Base de Datos
Accede al módulo de gestión de usuarios
obtenerTodosUsuarios()
findAll()
SELECT * FROM usuarios
Datos de usuarios
List<UserModel>
Lista de usuarios
Muestra lista de usuarios
alt
[Crear Usuario]
Ingresa datos de nuevo usuario
registrarUsuario(username, email, password)
findByUsername(username)
Optional<UserModel>
alt
[Usuario no existe]
findByName('ROLE_USER')
Optional<Role>
new(nuevoUsuario)
INSERT INTO usuarios
Confirmación
UserModel guardado
true (éxito)
Mensaje de éxito
[Usuario ya existe]
false (error)
Mensaje de error ('usuario ya existe')
[Editar Usuario]
Selecciona usuario y modifica datos
actualizarUsuario(idUsuario, username, email, rol)
findById(id)
Optional<UserModel>
alt
[Usuario encontrado]
findByUsername(username)
Optional<UserModel>
findByEmail(email)
Optional<UserModel>
findByName(rol)
Optional<Role>
save(usuarioActualizado)
UPDATE usuarios
Confirmación
UserModel actualizado
SUCCESS
Mensaje de éxito
[Usuario no encontrado o datos en uso]
ERROR:USERNAME_IN_USE/EMAIL_IN_USE
Mensaje de error
[Eliminar Usuario]
Selecciona usuario a eliminar
eliminarUsuario(id)
existsById(id)
boolean
alt
[Usuario existe]
deleteById(id)
DELETE FROM usuarios
Confirmación
Confirmación
Confirmación
Mensaje de éxito
[Usuario no existe]
Error (usuario no encontrado)
Mensaje de error

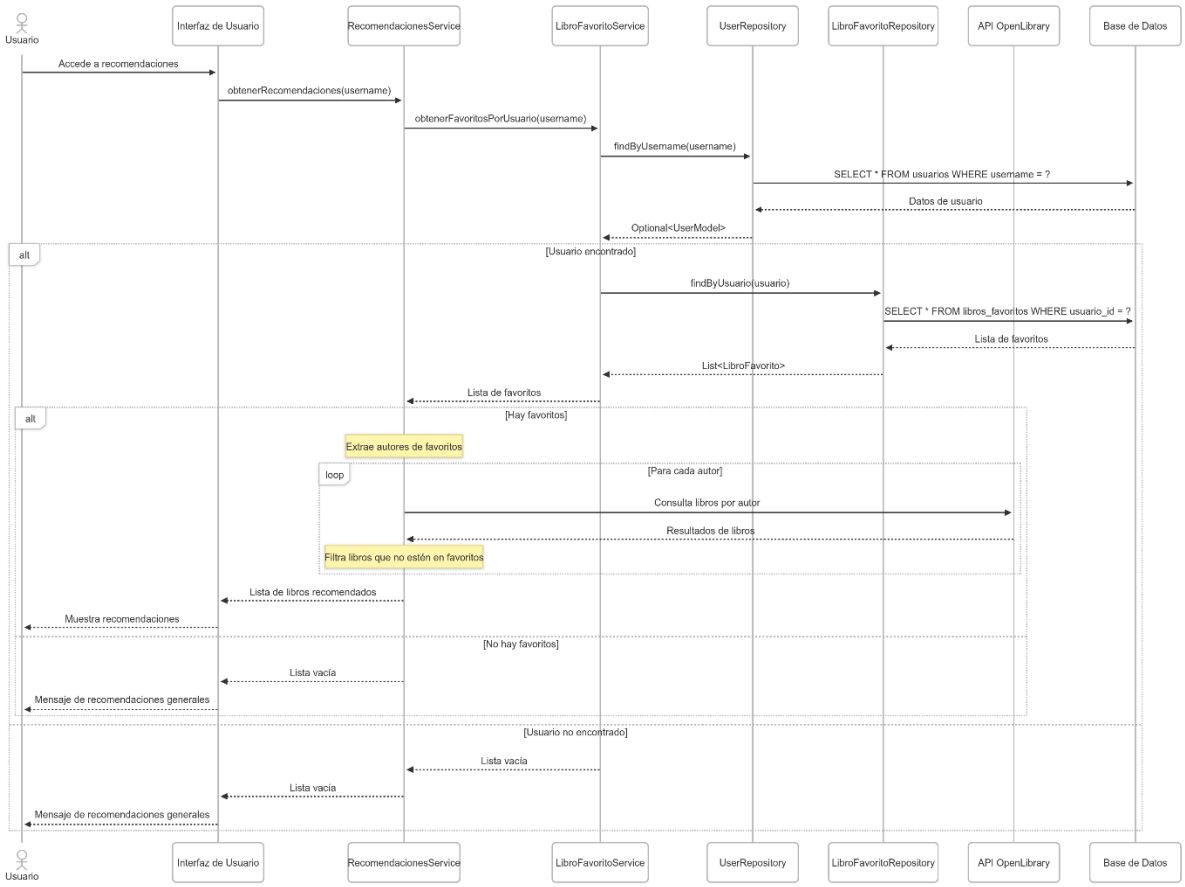
Búsqueda



Libros Favoritos

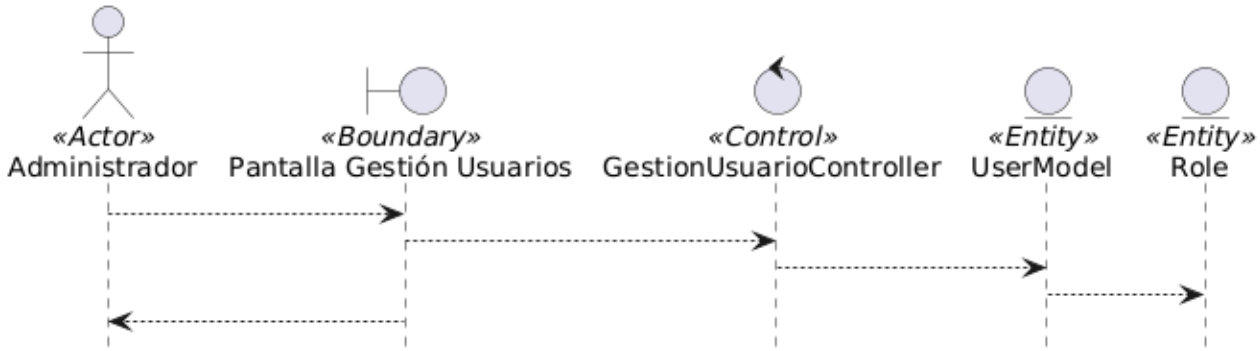


Recomendaciones

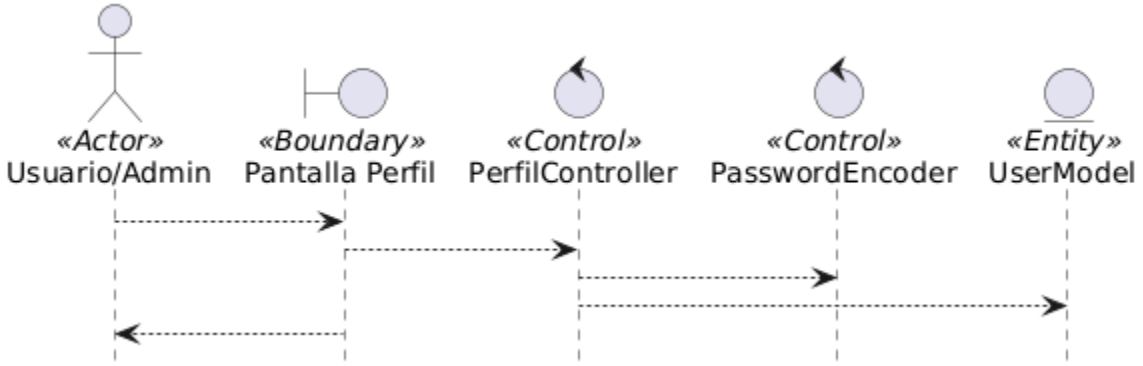


Diagramas de Robustez

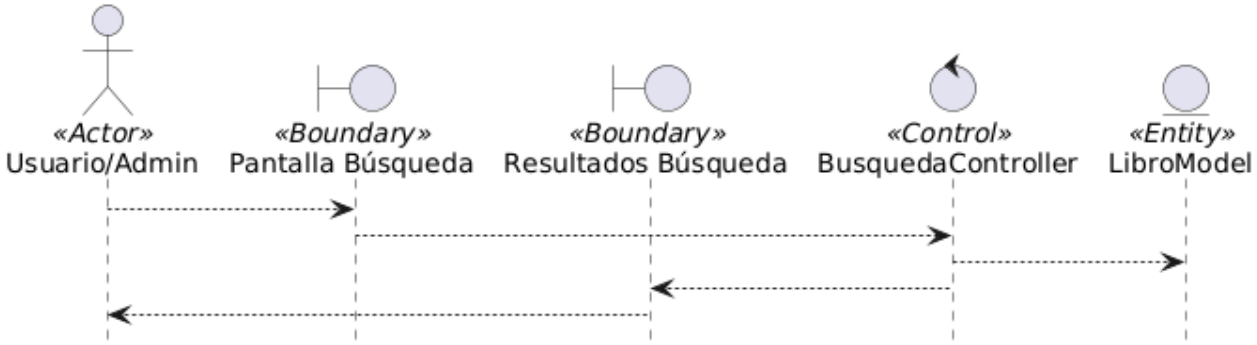
Gestión de Perfil



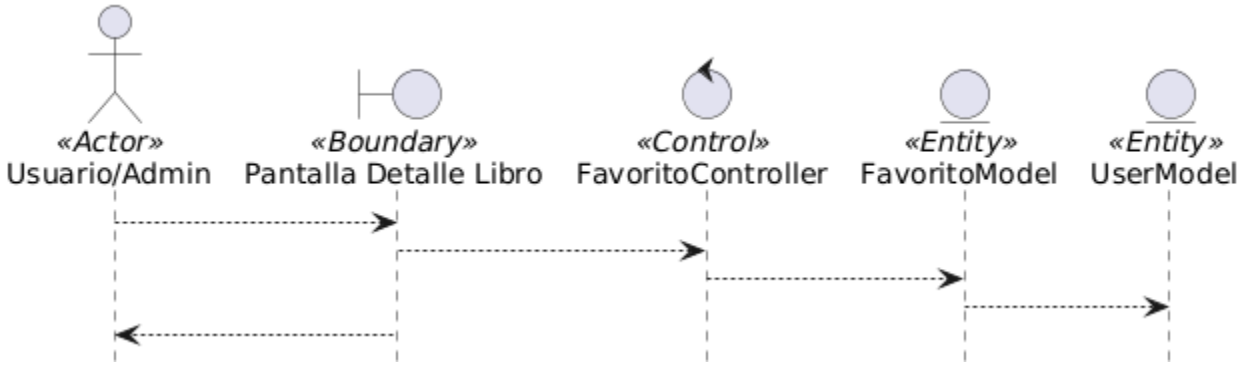
Gestión de Usuarios



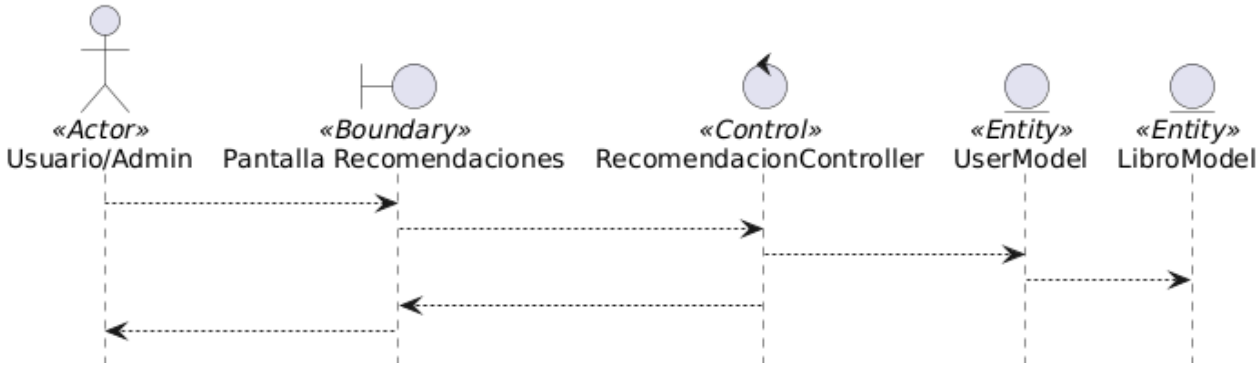
Búsqueda



Libros Favoritos



Recomendaciones



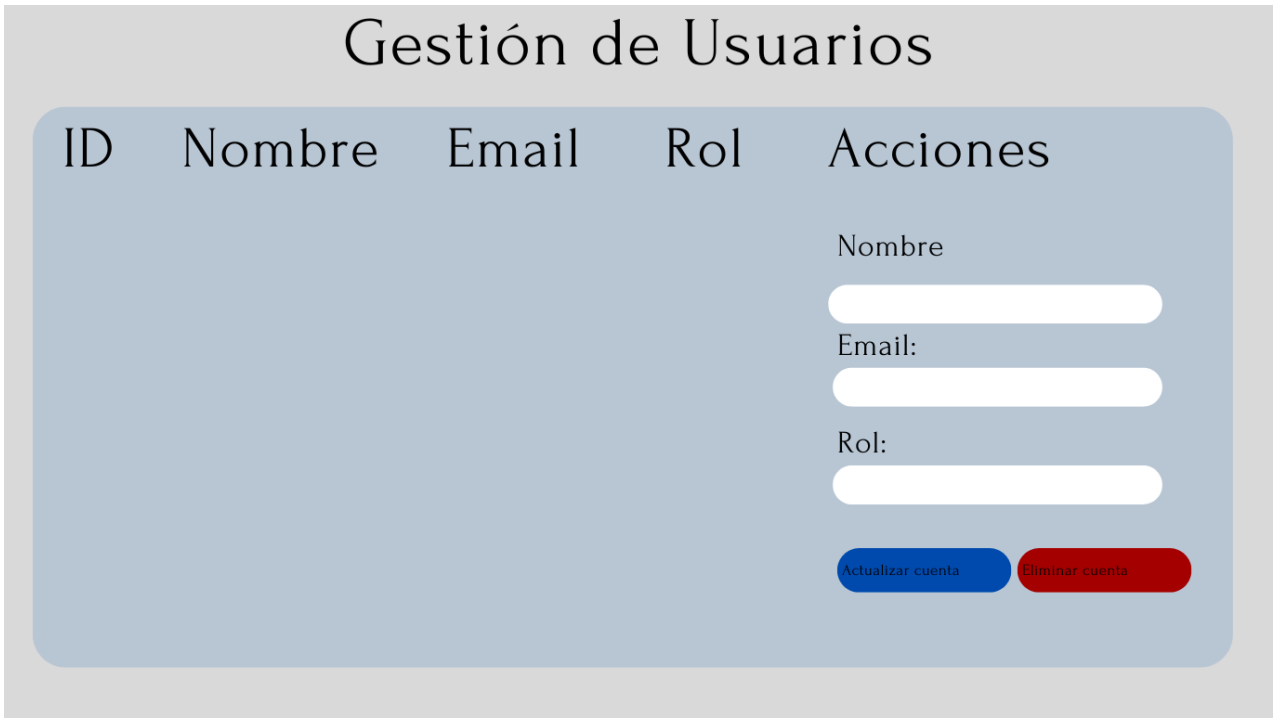
Modelo de Interfaz y Navegación

Login

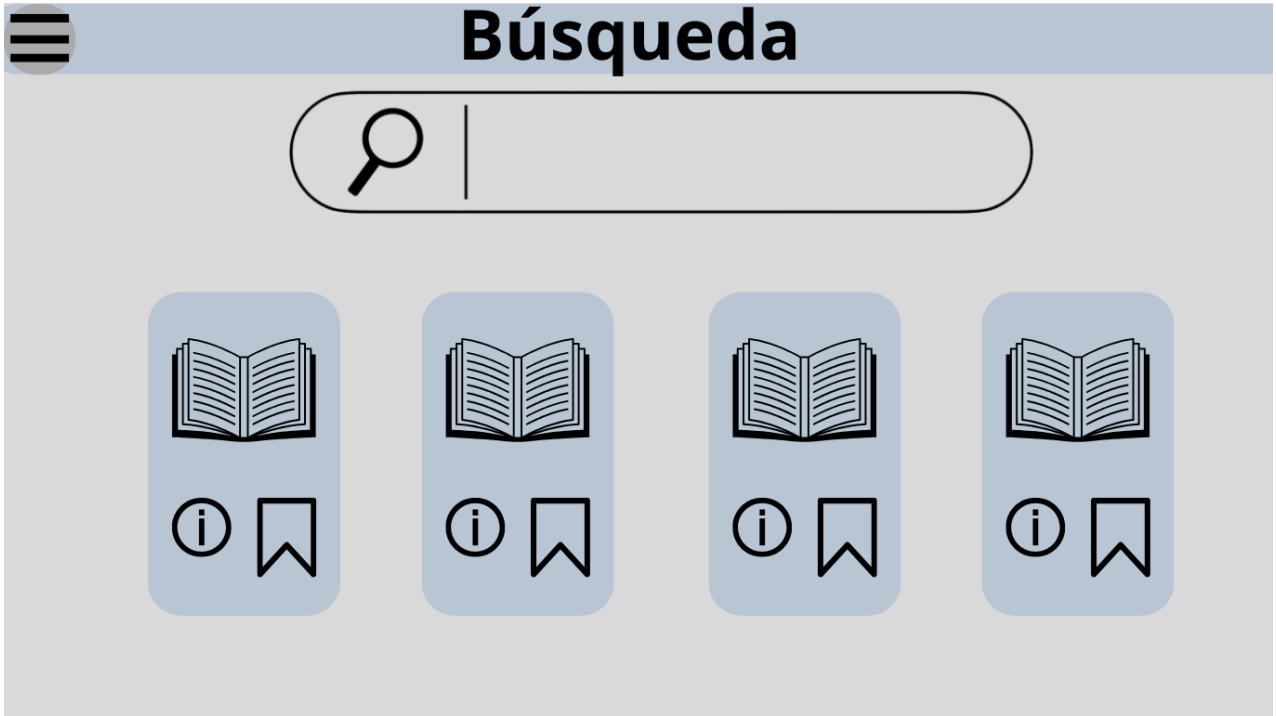
Registro

Perfil

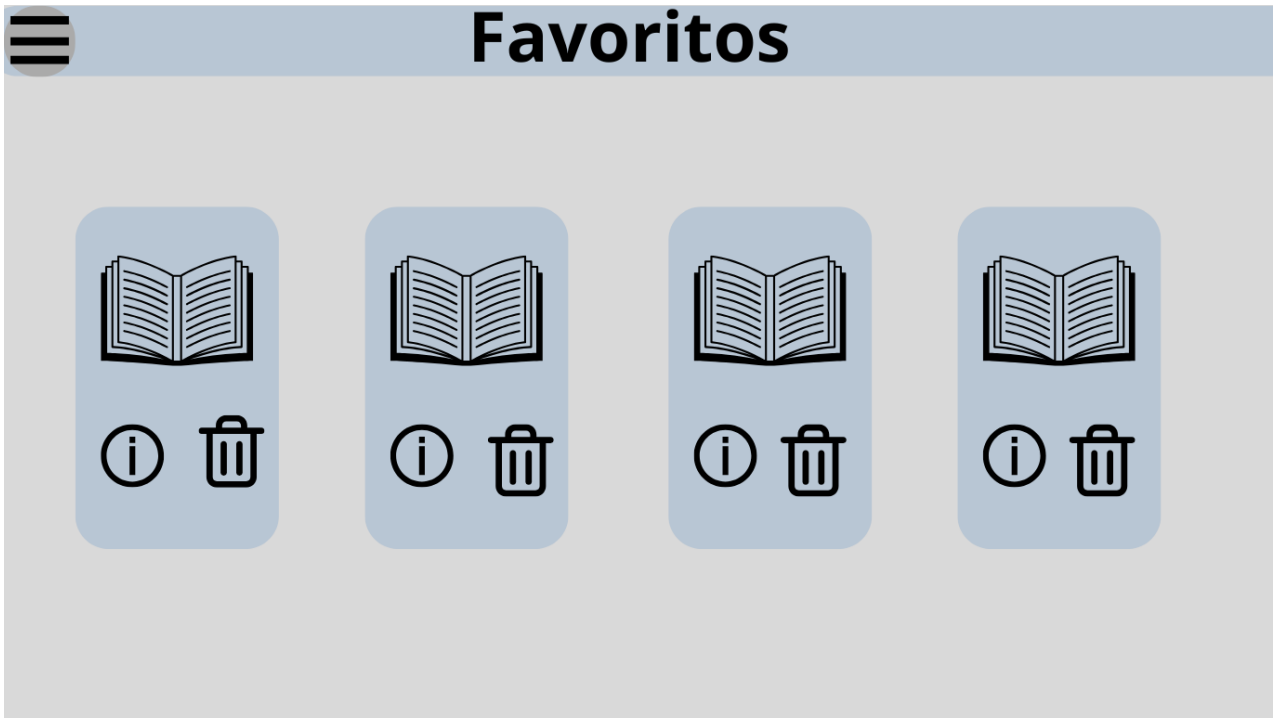
Gestión de Usuarios



Búsqueda



Favoritos



Recomendaciones

Flujo de Navegación

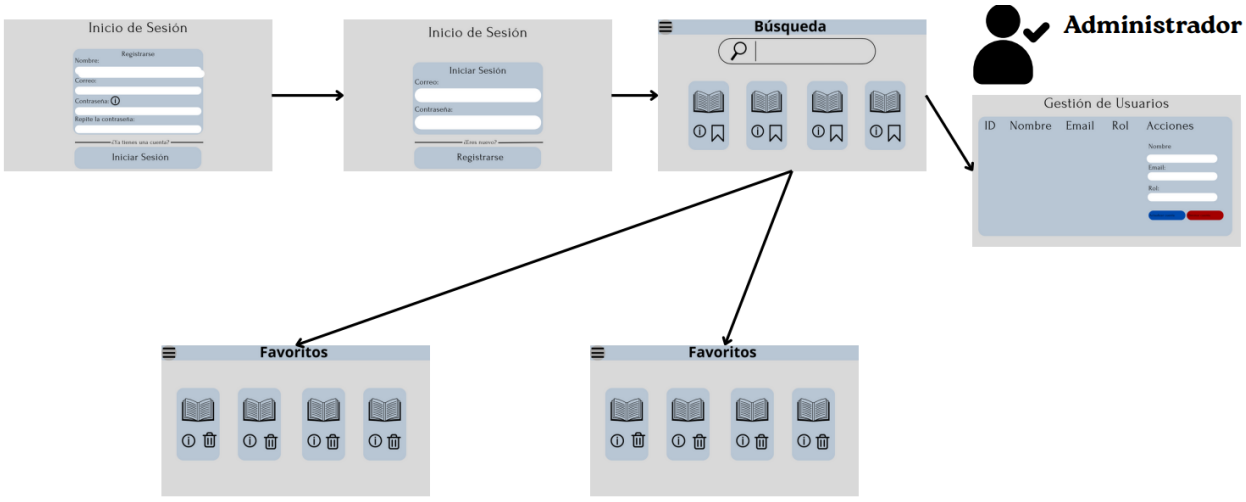
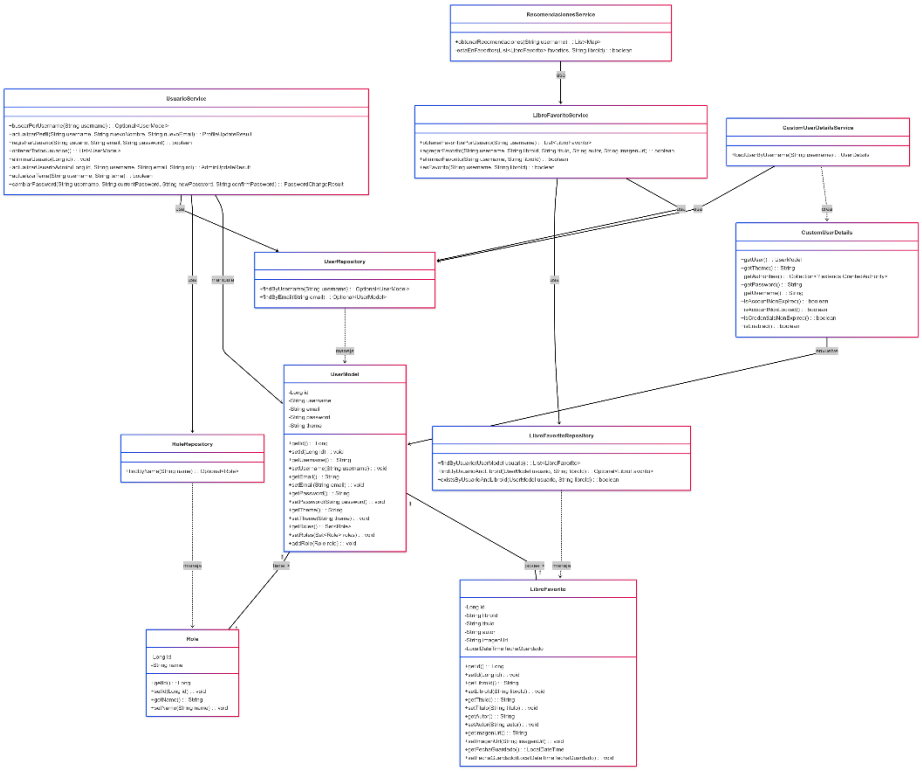


Diagrama de Clases de Diseño



El sistema está organizado en torno a varios módulos principales, cada uno compuesto por servicios, repositorios y modelos de datos. Estos módulos permiten la gestión de usuarios, el manejo de favoritos en la biblioteca y la generación de recomendaciones.

La arquitectura sigue el patrón **Servicio-Repositorio**, donde:

- Los **servicios** implementan la lógica de negocio.
- Los **repositorios** gestionan el acceso a los datos.
- Las **clases de modelo** representan las entidades del dominio.

El flujo de dependencias es:

Servicios → Repositorios → Modelos

El sistema está orientado a una aplicación de biblioteca o de medios, proporcionando funcionalidades de gestión de usuarios, favoritos y recomendaciones.

Módulos y Componentes

1. Módulo Principal de Gestión de Usuarios

- **UserService**
  Encargado de las operaciones principales de usuario, como:
  - Autenticación
  - Actualización de perfil
  - Gestión de contraseñas
- **UserRepository**
  Capa de acceso a datos que administra la información de los usuarios.
- **UserModel**
  Entidad del dominio que representa los datos de usuario, con propiedades y comportamientos asociados.
- **RoleRepository**
  Gestiona el acceso y persistencia de los datos relacionados con los roles de usuario.
- **Role**
  Entidad del dominio que define los roles de los usuarios dentro del sistema.

2. Módulo de Favoritos de la Biblioteca

- **LibraryFavoritesService**
  Servicio responsable de gestionar los elementos favoritos de la biblioteca.
- **LibraryFavoriteRepository**
  Capa de acceso a datos que administra los favoritos registrados por los usuarios.
- **LibraryFavorite**
  Entidad del dominio que representa un elemento marcado como favorito en la biblioteca.

3. Módulo de Detalles de Usuario

- **CustomUserDetailsService**
  Servicio que proporciona información extendida sobre los usuarios.
- **CustomUserDetails**
  Clase que almacena información adicional del usuario más allá de los datos básicos.

4. Módulo de Recomendaciones

- **RecommendationsService**
  Servicio encargado de generar recomendaciones personalizadas basadas en la actividad y preferencias del usuario.

Conexiones y Dependencias

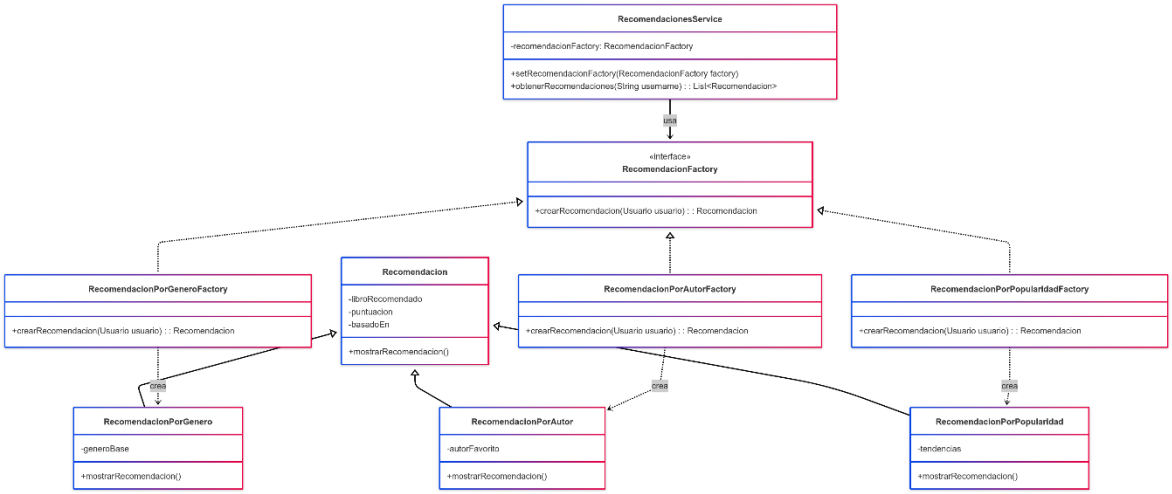
- Cada **servicio** depende de uno o varios **repositorios** para realizar sus operaciones.
- Los **repositorios** interactúan directamente con las **entidades del dominio** (modelos) para la persistencia de datos.
- Las dependencias están organizadas de manera que la lógica de negocio esté separada del acceso a datos, promoviendo una arquitectura limpia y mantenible.

Patrón de Diseño

Selección e Implementación del Patrón Factory Method

Para el sistema de gestión de libros, se ha decidido implementar el patrón **Factory Method** para la creación de recomendaciones de libros.

Diagrama UML del Patrón Factory Method



Justificación de la Selección del Patrón

El patrón Factory Method resulta apropiado para el sistema por las siguientes razones:

- **Variabilidad de algoritmos de recomendación:** El sistema requiere generar recomendaciones basadas en diferentes criterios, como géneros favoritos, autores preferidos o libros más populares.
- **Extensibilidad:** A futuro, será posible agregar nuevos tipos de algoritmos de recomendación sin necesidad de modificar el código existente.
- **Desacoplamiento:** Permite separar la lógica de creación de objetos de recomendación de su uso, respetando el principio de responsabilidad única (SRP).
- **Facilidad de pruebas:** La estructura facilita realizar pruebas unitarias, permitiendo inyectar fábricas de recomendación diferentes para probar diversos escenarios.

Mejora en la Calidad del Diseño

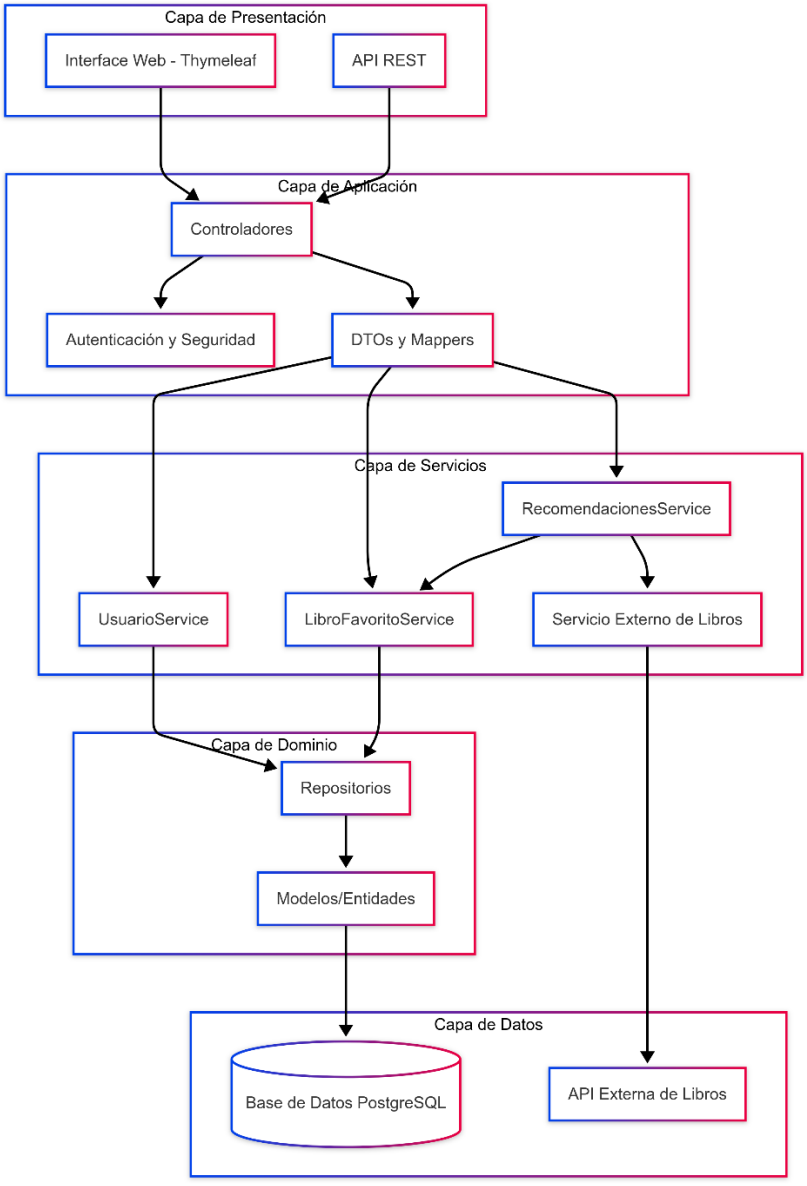
La implementación del patrón Factory Method contribuye de manera significativa a la mejora del diseño del sistema en los siguientes aspectos:

- **Principio Open/Closed (OCP):** El sistema está abierto a la extensión mediante nuevos algoritmos de recomendación, sin necesidad de alterar el código existente.
- **Cohesión mejorada:** Cada clase mantiene una responsabilidad clara y específica, aumentando la claridad del sistema.
- **Mantenibilidad:** La lógica de creación de recomendaciones está encapsulada en clases separadas, facilitando el mantenimiento y la evolución del sistema.

- **Flexibilidad:** El servicio de recomendaciones puede cambiar dinámicamente entre diferentes estrategias de recomendación de acuerdo al contexto o a las preferencias específicas del usuario.

Modelo de Implementación

Arquitectura Técnica del Sistema



Tecnologías y Frameworks Utilizados

Backend

- **Java 17** como lenguaje de programación principal.
- **Spring Boot 3.1.x** para la construcción del backend.
- **Spring Security** para la autenticación y autorización de usuarios.
- **Spring Data JPA** para la persistencia de datos.
- **Hibernate** como implementación de JPA.
- **Lombok** para la reducción de código repetitivo (boilerplate).

Frontend

- **Thymeleaf** como motor de plantillas para la generación dinámica de vistas.
- **Bootstrap 5** para diseño responsivo y componentes visuales.
- **JavaScript/jQuery** para la interactividad del cliente.

Base de Datos

- **PostgreSQL** como sistema de gestión de bases de datos relacional.

API y Comunicación

- **REST API** para la comunicación entre el frontend y el backend.
- **RestTemplate** o **WebClient** como cliente para el consumo de APIs externas de libros.
- **JSON** como formato de intercambio de datos.

Herramientas de Desarrollo

- **Maven** para la gestión de dependencias y el ciclo de vida del proyecto.
- **Git** como sistema de control de versiones.
- **JUnit** y **Mockito** para la implementación de pruebas unitarias y de integración.

Estructura de Componentes y Organización

La aplicación está estructurada en capas, siguiendo una arquitectura limpia que garantiza una separación clara de responsabilidades:

Capa de Presentación

- Controladores MVC para la gestión de vistas Thymeleaf.
- Controladores REST para la exposición de endpoints de API.
- Plantillas Thymeleaf organizadas por funcionalidades específicas.

Capa de Aplicación

- **DTOs (Data Transfer Objects)** para la transferencia eficiente de datos entre capas.
- **Mappers** para la conversión entre entidades y DTOs.
- **Filtros de seguridad** y configuración avanzada de autenticación.

Capa de Servicios

- Implementación de la lógica de negocio central.
- Integración con servicios externos.
- Aplicación de patrones de diseño como Factory Method.

Capa de Dominio

- **Entidades JPA** que representan conceptos del negocio (Usuario, LibroFavorito, Rol).
- **Repositorios** para el acceso y manipulación de datos.
- **Especificaciones** para la creación de consultas complejas.

Capa de Datos

- Configuración de conexiones a la base de datos.
- Migraciones de esquemas mediante herramientas de gestión de cambios.
- Configuración de clientes para el consumo de APIs externas.

Consideraciones sobre Seguridad, Rendimiento y Escalabilidad

Seguridad

Autenticación y Autorización

- Implementación de Spring Security con autenticación basada en sesiones.
- Definición de roles diferenciados (ADMIN, USER) para el control de acceso.
- Cifrado de contraseñas utilizando BCrypt.
- Protección contra ataques comunes como CSRF, XSS y SQL Injection.

Validación de Datos

- Validaciones tanto en el frontend como en el backend.
- Sanitización de entradas de usuario.
- Control estricto de acceso a nivel de servicios y controladores.

Rendimiento

Caché

- Implementación de caché para resultados de consumo de API externa.
- Uso de caché de segundo nivel de Hibernate para entidades de alta demanda.

Optimización de Consultas

- Utilización de consultas específicas y optimizadas en los repositorios.
- Aplicación de paginación para el manejo de grandes volúmenes de resultados.
- Búsquedas eficientemente indexadas.

Escalabilidad

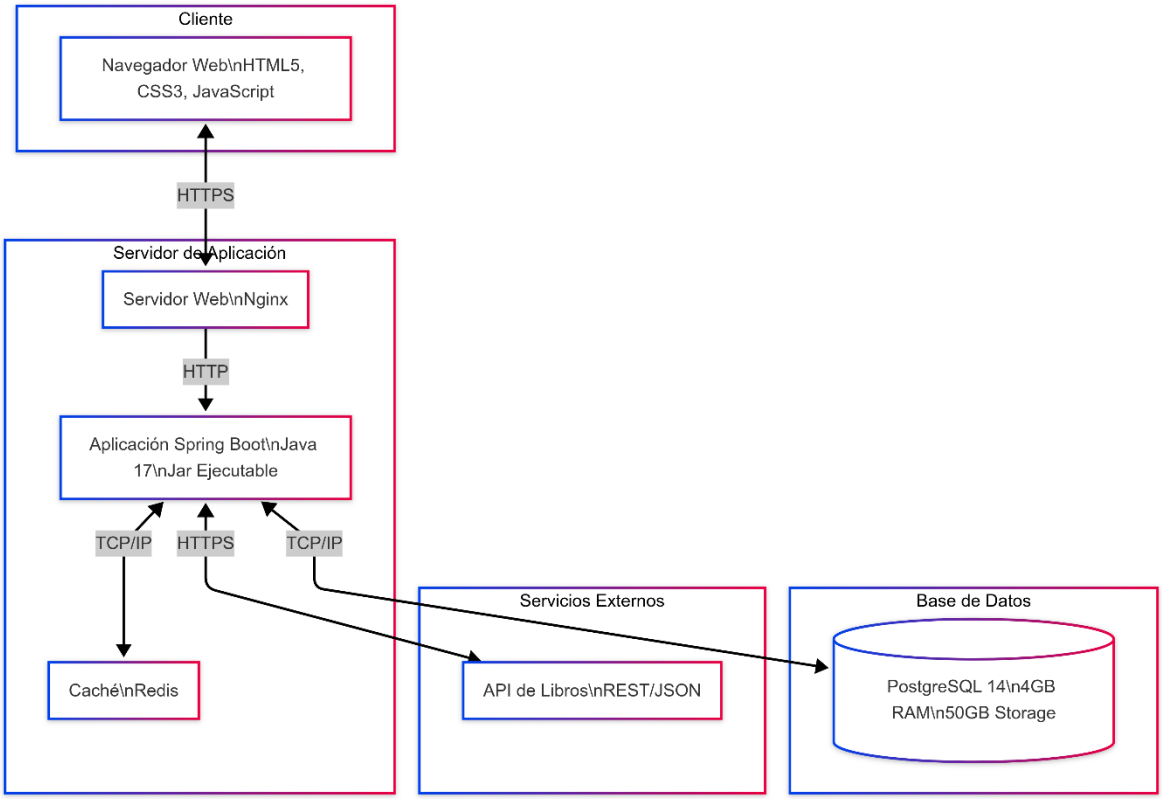
Diseño Modular

- Componentes desacoplados que permiten un escalado horizontal sencillo.
- Servicios diseñados como stateless, favoreciendo la ejecución en múltiples instancias.

Base de Datos

- Índices optimizados para consultas rápidas.
- Uso de conexiones en pool para mejorar la gestión de recursos.
- Planificación para futura migración a clusters de base de datos en caso de necesidad.

Diagrama de Despliegue



Componentes Físicos del Sistema

Cliente

- **Navegador Web**: Cualquier navegador moderno compatible con HTML5, CSS3 y JavaScript.
- **Requisitos mínimos**:
  - Conexión a internet estable.
  - Capacidad para ejecutar JavaScript moderno.

Servidor de Aplicación

Hardware:

- CPU: mínimo 4 núcleos.
- RAM: mínimo 8 GB.
- Almacenamiento: SSD de al menos 50 GB.

Software:

- Sistema Operativo: **Linux Ubuntu 22.04 LTS**.
- **Java 17 JRE** para la ejecución de la aplicación.
- **Nginx** configurado como proxy inverso.
- Aplicación empaquetada como **Spring Boot JAR**.
- **Redis** como sistema de caché (opcional).

Servidor de Base de Datos

Hardware:

- CPU: mínimo 2 núcleos.
- RAM: mínimo 4 GB.
- Almacenamiento: SSD de al menos 50 GB con respaldo configurado.

Software:

- Sistema Operativo: **Linux Ubuntu 22.04 LTS**.
- **PostgreSQL 14** como motor de base de datos.