Documentación del Proyecto TallerRMI

Arquitectura Cliente–Servidor con Java RMI y SQLite

Autores

Juan Pablo Espinosa Robled
Juan Manuel Lopez Vargas
Juan Santiago Saavedra Holguín
Johan Sebastián Méndez

Programa de Ingeniería de Sistemas

Pontificia Universidad Javeriana

Resumen

El proyecto TallerRMI implementa un sistema distribuido de gestión de biblioteca basado en *Java RMI* para invocación remota y *SQLite* como motor de base de datos embebido. El objetivo es exponer operaciones remotas síncronas que permitan a clientes consultar disponibilidad, realizar préstamos por ISBN o título y registrar devoluciones, manteniendo persistencia y consistencia de datos. Este documento presenta la arquitectura, la descripción exhaustiva de los componentes (archivo por archivo), y el flujo de interacción extremo a extremo entre capas.

Índice

1 Introducción

1.1 Propósito

El propósito de esta documentación es servir como referencia técnica formal para el taller TallerRMI. Se describen los componentes, su rol en la arquitectura, cómo se reflejan en la estructura del repositorio y cómo colaboran para satisfacer los requerimientos funcionales: consulta por ISBN, préstamo por ISBN o título y devolución por ISBN, con cálculo de fecha de devolución a siete días.

1.2 Alcance

Alcance del sistema:

- Exposición de un servicio remoto (RMI) con operaciones síncronas.
- Persistencia local mediante *SQLite*.
- Cliente de consola para interacción con el servicio.
- Despliegue en entornos distribuidos (cliente y servidor en máquinas distintas).

2 Arquitectura general

2.1 Visión de capas

El sistema adopta una arquitectura de tres capas lógicas:

1. **Presentación (CLI)**: `ClientePrincipal.java`.
2. **Negocio/Servicio**: `ServicioBiblioteca.java` (contrato) y `ServicioBibliotecaImpl.java` (implementación).
3. **Persistencia**: `BaseDatos.java` operando sobre `biblioteca.db` (SQLite).

Se utilizan DTOs serializables (`ResultadoConsulta`, `ResultadoPrestamo`, `ResultadoDevolucion`) para transportar datos entre servidor y cliente.

2.2 Diagrama de alto nivel

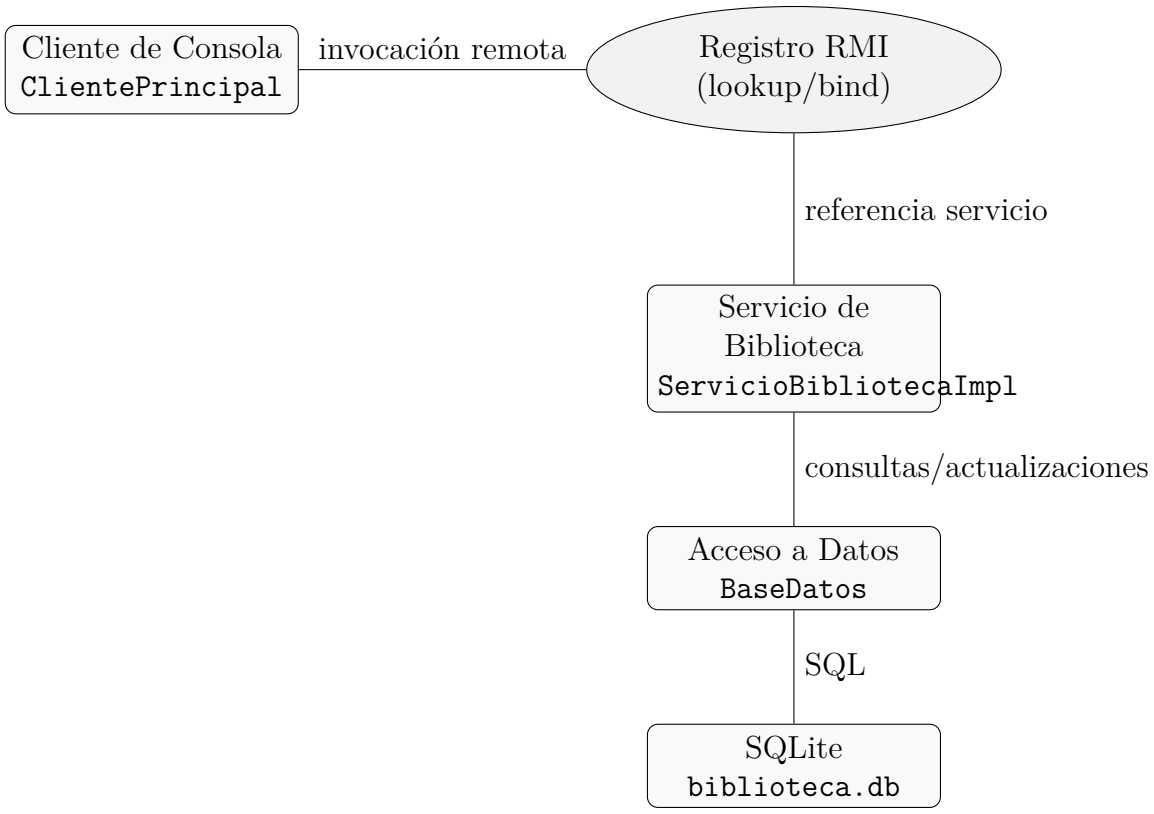

Figura 1: Arquitectura de alto nivel del sistema.

3 Estructura del repositorio

La estructura típica del proyecto es la siguiente (los nombres pueden variar levemente):

```
/ (raz)
          src/
                biblioteca/
                    BaseDatos.java
                    ServicioBiblioteca.java
                    ServicioBibliotecaImpl.java
                    ServidorPrincipal.java
                    ClientePrincipal.java
                    ResultadoConsulta.java
                    ResultadoPrestamo.java
                    ResultadoDevolucion.java
          data/
                biblioteca.db
                seed.sql
          lib/
                sqlite-jdbc-<version>.jar
          out/                  # artefactos compilados (.class)
          compile.sh
          run_server.sh
```

```
run_client.sh
```

4 Descripción exhaustiva por archivo

4.1 `BaseDatos.java` (Persistencia)

Rol: Capa de acceso a datos (DAO), encapsula la interacción con `SQLite`.

Responsabilidades principales:

- Establecer conexiones JDBC hacia `biblioteca.db`.
- Ejecutar consultas y comandos SQL para:
  - Búsqueda por ISBN y por título.
  - Cálculo de disponibilidad (stock vs. préstamos).
  - Registro de operaciones de préstamo y devolución.
- Gestionar transacciones para asegurar atomicidad y consistencia.

Cómo se refleja en la estructura: Ubicado en `src/biblioteca/`. Es consumido por `ServicioBibliotecaImpl`, que solicita operaciones de lectura/escritura. Sus clases compiladas se ubican en `out/biblioteca/`.

Notas de diseño:

- Uso de `PreparedStatement` para parametrización segura.
- Manejo de recursos con `try-with-resources`.
- Posible modelo de tablas: `libro(id, isbn, titulo, total)`, `prestamo(id, isbn, usuario, fecha_prestamo, fecha_devolucion)`.

4.2 `ServicioBiblioteca.java` (Contrato remoto)

Rol: Define la interfaz RMI (contrato) entre cliente y servidor.

Responsabilidades principales:

- Declarar métodos remotos (lanzando `RemoteException`) para:
  - Consulta por ISBN.
  - Préstamo por ISBN y por título.
  - Devolución por ISBN.
- Establecer tipos de retorno mediante DTOs.

Cómo se refleja en la estructura: Ubicada en `src/biblioteca/`, es visible para cliente y servidor, garantizando un contrato compartido y estable.

4.3 `ServicioBibliotecaImpl.java` (Lógica de negocio)

Rol: Implementación de la lógica del servicio remoto.

Responsabilidades principales:

- Validación de entradas (ISBN, título, usuario).
- Coordinación con `BaseDatos` para materializar la operación solicitada.

- Construcción de respuestas mediante `ResultadoConsulta`, `ResultadoPrestamo` y `ResultadoDevolucion`.
- Cálculo de la fecha de devolución para préstamos (7 días posteriores a la fecha actual).

Cómo se refleja en la estructura: Se instancia en `ServidorPrincipal` y se expone vía RMI. Centraliza reglas de negocio y asegura consistencia entre capas.

4.4 `ServidorPrincipal.java` (Arranque del servidor)

Rol: Punto de entrada del servidor, responsable de la publicación del servicio.

Responsabilidades principales:

- Inicializar el Registro RMI (p.ej., puerto 1099).
- Instanciar `ServicioBibliotecaImpl`.
- Registrar el servicio con un nombre lógico (`"BibliotecaService"`).
- Configurar parámetros de ejecución (ruta de `biblioteca.db`, hostname para clientes remotos).

Cómo se refleja en la estructura: Ejecución mediante `run_server.sh`. Publica las dependencias necesarias (clases propias y driver JDBC en `lib/`).

4.5 `ClientePrincipal.java` (Cliente de consola)

Rol: Interfaz de usuario basada en consola, traduce comandos en invocaciones remotas.

Responsabilidades principales:

- Conectarse al Registro RMI (host, puerto).
- Resolver la referencia al servicio e invocar métodos remotos.
- Parsear y validar comandos:
  - `consultar <isbn>`
  - `prestar_isbn <isbn><usuario>`
  - `prestar_titulo <titulo><usuario>`
  - `devolver <isbn><usuario>`
  - `salir`
- Presentar respuestas formateadas de los DTOs al usuario.

Cómo se refleja en la estructura: Ejecución mediante `run_client.sh` con parámetros de host y puerto.

4.6 `ResultadoConsulta.java`, `ResultadoPrestamo.java`, `ResultadoDevolucion.java` (DTOs)

Rol: Objetos de transferencia de datos, serializables, utilizados como respuesta de las operaciones remotas.

Responsabilidades principales:

- Transportar estado, mensaje y datos asociados a cada operación.
- Asegurar compatibilidad entre cliente y servidor con un contrato de datos estable.

Cómo se refleja en la estructura: Ubicados en `src/biblioteca/`, importados por el cliente y el servidor.

4.7 `seed.sql` y `biblioteca.db` (Datos y persistencia)

Rol: `seed.sql` inicializa esquema y datos; `biblioteca.db` almacena el estado persistente.

Responsabilidades principales:

- `seed.sql`: creación de tablas y carga de datos base.
- `biblioteca.db`: repositorio local de información (libros, préstamos, devoluciones).

Cómo se refleja en la estructura: Residen típicamente en `data/`. Es recomendable no versionar `.db` generados.

4.8 `lib/sqlite-jdbc-<version>.jar` (Dependencia externa)

Rol: Driver JDBC para habilitar la conexión de Java con SQLite.

Cómo se refleja en la estructura: Incluido en `lib/` y referenciado en el *classpath* en compilación/ejecución.

4.9 `compile.sh`, `run_server.sh`, `run_client.sh` (Automatización)

Rol: Scripts de automatización de construcción y ejecución.

Responsabilidades principales:

- `compile.sh`: compilar fuentes a `out/`.
- `run_server.sh`: levantar servidor parametrizando ruta de BD y puerto.
- `run_client.sh`: iniciar cliente parametrizando host y puerto.

5 Flujo de funcionamiento entre archivos

5.1 Descripción general del flujo

1. **Arranque del servidor**: `ServidorPrincipal` inicia el Registro RMI, instancia `ServicioBibliotecaImpl` y publica la referencia con nombre lógico.
2. **Conexión del cliente**: `ClientePrincipal` resuelve la referencia del servicio mediante `lookup` y queda listo para invocar operaciones.
3. **Ejecución de operaciones**:
   a) El usuario ingresa un comando en la CLI.
   b) `ClientePrincipal` traduce el comando a una invocación remota de `ServicioBiblioteca`.
   c) La plataforma RMI serializa la solicitud y la entrega a `ServicioBibliotecaImpl`.
   d) `ServicioBibliotecaImpl` valida, consulta/actualiza a través de `BaseDatos`, aplica reglas (p.ej., fecha de devolución a 7 días) y construye el DTO de respuesta.
   e) La respuesta (DTO) viaja de regreso al cliente a través de RMI y `ClientePrincipal` la presenta al usuario.

5.2 Secuencias típicas

5.2.1 Consulta por ISBN

1. Usuario: `consultar <isbn>`.
2. `ClientePrincipal` → `ServicioBiblioteca.consultarPorIsbn(isbn)`.
3. `ServicioBibliotecaImpl` → `BaseDatos` (SELECT por ISBN, cálculo de disponibilidad).
4. Retorno: `ResultadoConsulta` con *existe* y *disponibles*.

5.2.2 Préstamo por ISBN/Título

1. Usuario: `prestar_isbn <isbn><usuario>` o `prestar_titulo <titulo><usuario>`.
2. `ClientePrincipal` → método remoto correspondiente.
3. `ServicioBibliotecaImpl` valida disponibilidad y registra préstamo en `BaseDatos`.
4. Calcula *fecha de devolución* = hoy + 7 días.
5. Retorno: `ResultadoPrestamo` con estado/mensaje y fecha de devolución.

5.2.3 Devolución por ISBN

1. Usuario: `devolver <isbn><usuario>`.
2. `ClientePrincipal` → `ServicioBiblioteca.devolverPorIsbn(isbn, usuario)`.
3. `ServicioBibliotecaImpl` verifica préstamo abierto y actualiza estado en `BaseDatos`.
4. Retorno: `ResultadoDevolucion` con estado y mensaje.

6 Ficha técnica

Atributo	Descripción
Lenguaje	Java 17
Tecnología de comunicación	Java RMI (invocación remota, operaciones síncronas)
Base de datos	SQLite 3 (archivo `biblioteca.db`)
Acceso a datos	JDBC, clase `BaseDatos` (DAO)
Capa de servicio	`ServicioBiblioteca` (contrato), `ServicioBibliotecaImpl` (implementación)
Cliente	`ClientePrincipal` (CLI)
DTOs	`ResultadoConsulta`, `ResultadoPrestamo`, `ResultadoDevolucion`
Operaciones	Consultar por ISBN; Prestar por ISBN/título; Devolver por ISBN
Regla de negocio	Fecha de devolución = 7 días posteriores al préstamo
Automatización	`compile.sh`, `run_server.sh`, `run_client.sh`
Dependencias externas	`lib/sqlite-jdbc-<version>.jar`

Despliegue	Cliente y servidor en máquinas distintas (lookup a Registro RMI)

Apéndice A. Comandos de ejemplo (CLI)

```
# Compilaci n
./compile.sh

# Servidor: (ruta BD, puerto)
./run_server.sh ./data/biblioteca.db 1099

# Cliente: (host, puerto)
./run_client.sh 127.0.0.1 1099

# Comandos en el cliente
consultar <isbn>
prestar_isbn <isbn> <usuario>
prestar_titulo <titulo> <usuario>
devolver <isbn> <usuario>
salir
```