

Pontificia Universidad Javeriana



Jose Guerrero, Samuel Giraldo, Daniel Diaz y Marianne Coy

Jhon Corredor

Sistemas Distribuidos

27 de Febrero 2026

## **1. Descripción General del Sistema**

El proyecto implementa un sistema distribuido de gestión de biblioteca utilizando arquitectura Cliente–Servidor basada en gRPC.

El sistema permite:

- Consultar libros por ISBN
- Prestar libros por ISBN
- Prestar libros por título
- Devolver libros
- Consultar disponibilidad de ejemplares

El servidor central administra la base de datos SQLite y expone servicios remotos mediante gRPC.

Los clientes se conectan remotamente al servidor a través de la red.

## **2. Arquitectura del Sistema**

### **Modelo Cliente – Servidor**

Cliente (Computador A)

↓ gRPC

Servidor (Computador B – Máquina Virtual)

El servidor:

- Ejecuta en Ubuntu 22.04 LTS
- Puerto configurado: 50051
- Gestiona base de datos SQLite
- Controla concurrencia y transacciones

El cliente:

- Se conecta a la IP del servidor
- Consume los servicios gRPC
- Presenta un menú interactivo en consola

## **3. Entorno del Servidor**

- Sistema Operativo: Ubuntu 22.04.5 LTS
- IP: 10.43.99.126
- Usuario: estudiante
- Puerto gRPC: 50051

- Base de datos: SQLite (biblioteca.db)

El servidor se ejecuta con:

- `mvn exec:java -Dexec.args="50051"`
- Salida esperada:
- Servidor gRPC listo en puerto 50051

#### **4. Modelo de datos (BD) y consistencia**

La base de datos en el servidor tiene dos entidades principales:

##### **Tabla libros**

- isbn (PK)
- titulo
- total\_ejemplares
- ejemplares\_disponibles

##### **Invariante clave del sistema**

- $0 \leq \text{ejemplares\_disponibles} \leq \text{total\_ejemplares}$

Este invariante se respeta en:

- Préstamo: decrementa solo si hay disponibles
- Devolución: incrementa sin superar el total

##### **Tabla prestamos**

- id\_prestamo (PK autoincrement)
- isbn (FK lógica hacia libros)
- prestatario
- fecha\_prestamo
- fecha\_devolucion
- devuelto (0/1)
- fecha\_entrega (cuando se devuelve)

##### **Qué representa**

- Historial de transacciones de préstamo.
- Posibilidad de auditoría: quién lo tomó, cuándo, cuándo vencía.

## 5. Lógica de negocio detallada

### Préstamo (servidor)

Tu método prestar() está bien hecho por dos razones importantes:

#### Transacción explícita

```
c.setAutoCommit(false);
```

```
c.commit();
```

Si algo falla, se hace rollback y no quedan estados inconsistentes.

### Control de inventario a prueba de carreras simples

Esta línea es clave:

```
UPDATE libros
```

```
SET ejemplares_disponibles = ejemplares_disponibles - 1
```

```
WHERE isbn=? AND ejemplares_disponibles > 0
```

- Si dos clientes piden el último ejemplar, **solo uno** debería lograr el UPDATE (afecta 1 fila).
- El otro recibe executeUpdate() == 0 y retorna “no disponibles”.

Esto es un control optimista muy válido para el taller.

### Flujo lógico completo

1. Verifica que ISBN exista
2. Verifica disponibles
3. Decrementa disponibles con condición
4. Inserta préstamo

5. Calcula devolución +7 días

6. Commit

### **Resultado esperado**

- Si hay disponibles → ok + fecha devolución
- Si no hay → mensaje de no disponibilidad
- Si no existe → mensaje de inexistente

### **Devolución (servidor)**

Tu método devolver() también está bien:

- Busca un préstamo activo (devuelto=0)
- Marca devuelto=1 y registra fecha\_entrega
- Incrementa disponibles con cap:

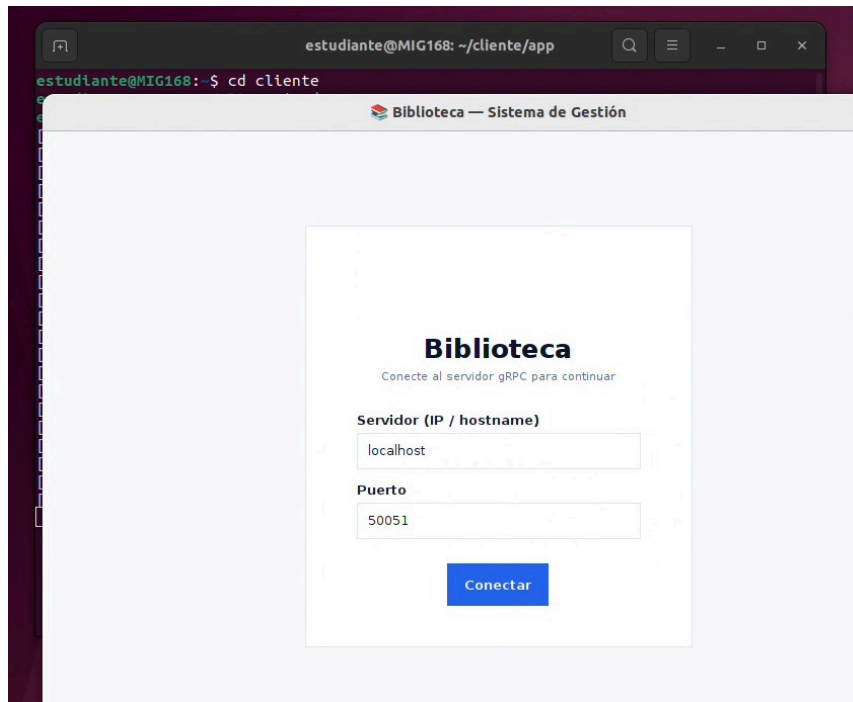
UPDATE libros

SET ejemplares\_disponibles = MIN(total\_ejemplares, ejemplares\_disponibles + 1)

WHERE isbn=?

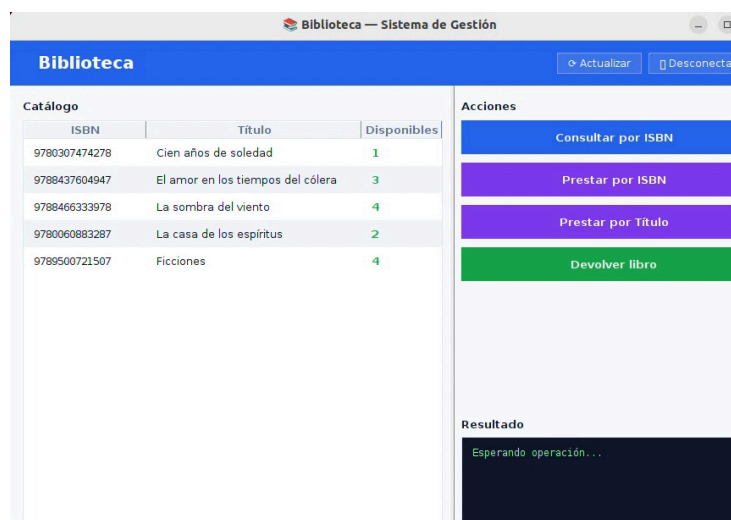
Esto garantiza el invariante de no superar el total.

### **Funcionalidades del Cliente**



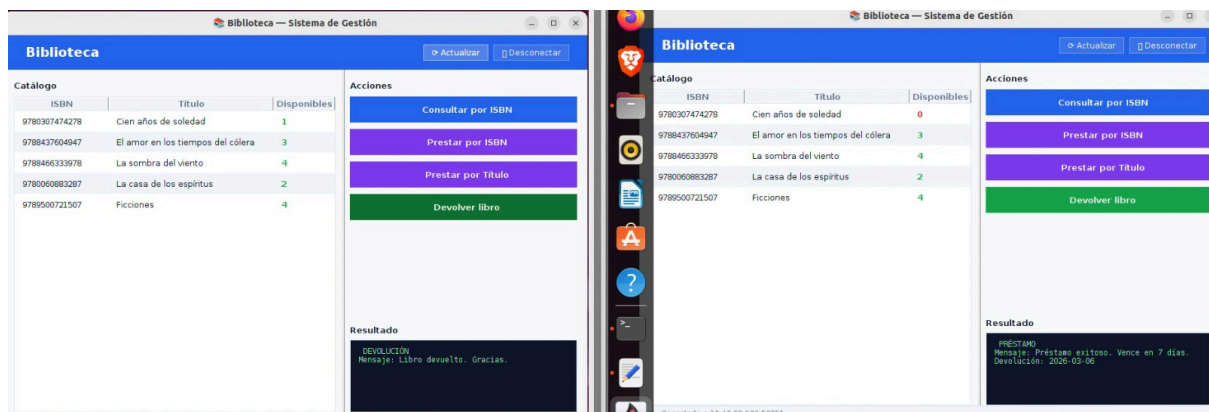
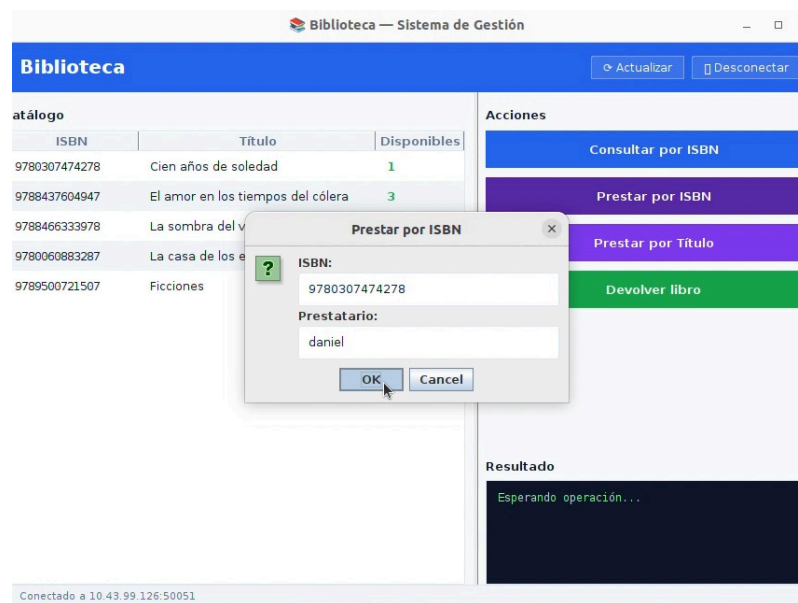
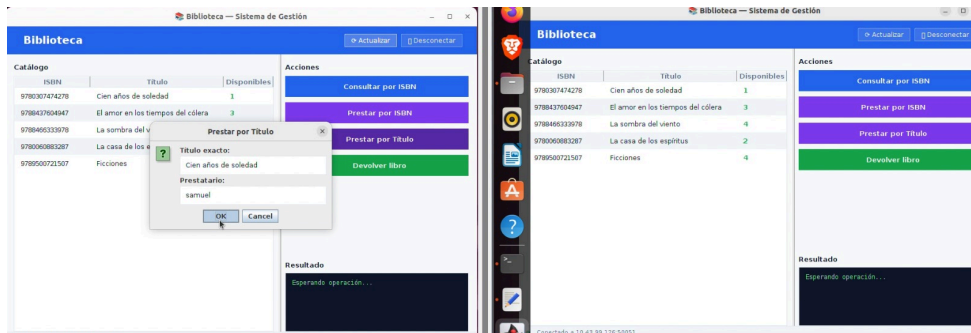
Menú implementado:

1. Listar libros disponibles
2. Consultar libro por ISBN
3. Prestar libro por ISBN
4. Prestar libro por Título
5. Devolver libro



El cliente:

- Se conecta al servidor remoto
- Envía solicitudes gRPC
- Muestra mensajes claros al usuario
- Permite visualizar disponibilidad antes de solicitar préstamo



## **6. Conclusión**

El sistema cumple con los requerimientos del taller:

- Implementa arquitectura distribuida real
- Usa comunicación remota con gRPC
- Gestiona persistencia con base de datos
- Controla transacciones
- Maneja disponibilidad de inventario
- Permite pruebas en múltiples máquinas

El sistema es funcional, consistente y extensible.