TRƯỜNG ĐẠI HỌC PHENIKA

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO DỰ ÁN

HỌC PHẦN ỨNG DỤNG PHÂN TÁN

ĐỀ TÀI: XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ SÁCH CHO ĐẠI HỌC SỬ DỤNG KỸ THUẬT LẬP TRÌNH PHÂN TÁN JAVA RMI

Nhóm 16 - Ứng dụng phân tán\*-1-2-24(N08)

|  |  |
| --- | --- |
| Thành viên: | Nguyễn Minh Trường Giang – 21011597  Nguyễn Cao Anh – 21012036 |
| Giảng viên hướng dẫn: | TS. Trần Đăng Hoan |

Hà Nội, 2/2025

**BẢNG PHÂN CÔNG CÔNG VIỆC**

|  |  |  |  |
| --- | --- | --- | --- |
| Mã SV | Họ và tên | Nhiệm vụ | Tỉ lệ đóng góp |
| 21011597 | Nguyễn Minh Trường Giang | Làm báo cáo, viết code | 50% |
| 21012036 | Nguyễn Cao Anh | Làm báo cáo, viết code | 50% |

MỤC LỤC

[DANH MỤC HÌNH ẢNH 1](#_Toc190357483)

[DANH MỤC THUẬT NGỮ 2](#_Toc190357484)

[LỜI MỞ ĐẦU 3](#_Toc190357485)

[CHƯƠNG I. TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN 4](#_Toc190357486)

[1.1. Giới thiệu 4](#_Toc190357487)

[1.1.1. Định nghĩa lập trình phân tán 4](#_Toc190357488)

[1.1.2. Đặc điểm của hệ thống phân tán 4](#_Toc190357489)

[1.1.3. Lý do sử dụng hệ thống phân tán 5](#_Toc190357490)

[1.1.4. Ví dụ về hệ thống phân tán 5](#_Toc190357491)

[1.2. Các khái niệm cơ bản 5](#_Toc190357492)

[1.2.1. Kiến trúc Client-Server 5](#_Toc190357493)

[1.2.2. Kiến trúc Peer-to-Peer (P2P) 6](#_Toc190357494)

[1.2.3. Cơ chế giao tiếp giữa các tiến trình (IPC – Inter-Process Communication) 7](#_Toc190357495)

[1.3. Các thách thức trong thiết kế hệ thống phân tán 8](#_Toc190357496)

[1.3.1. Kiểm soát đồng thời (Concurrency Control) 8](#_Toc190357497)

[1.3.2. Khả năng chịu lỗi (Fault Tolerance) 9](#_Toc190357498)

[1.3.3. Tính nhất quán của dữ liệu (Data Consistency) 10](#_Toc190357499)

[1.4. Xây dựng và triển khai hệ thống phân tán 11](#_Toc190357500)

[1.4.1. Cân nhắc trong thiết kế 11](#_Toc190357501)

[1.4.2. Lựa chọn công nghệ phù hợp 12](#_Toc190357502)

[CHƯƠNG II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 14](#_Toc190357503)

[2.1. Xây dựng và triển khai hệ thống phân tán 14](#_Toc190357505)

[2.1.1. Yêu cầu chức năng 14](#_Toc190357508)

[2.1.2. Yêu cầu phi chức năng 16](#_Toc190357509)

[2.1. Mô hình Use-Case 17](#_Toc190357510)

[2.2. Mô hình lớp và đối tượng 22](#_Toc190357511)

[2.3. Biểu đồ tuần tự 23](#_Toc190357512)

[2.4. Thiết kế kiến trúc hệ thống 25](#_Toc190357513)

[CHƯƠNG III. CÀI ĐẶT VÀ TRIỂN KHAI 27](#_Toc190357525)

[3.1. Lựa chọn công nghệ 27](#_Toc190357526)

[3.2. Cấu trúc mã nguồn 28](#_Toc190357527)

[3.3. Triển khai ứng dụng 32](#_Toc190357531)

[KẾT LUẬN 40](#_Toc190357532)

[TÀI LIỆU THAM KHẢO 41](#_Toc190357533)

# DANH MỤC HÌNH ẢNH

[*Hình 1: Mô hình Use-Case 17*](#_Toc190357463)

[*Hình 2: Biểu đồ mô hình lớp và đối tượng 22*](#_Toc190357464)

[*Hình 3: Biểu đồ tuần tự thêm sách 23*](#_Toc190357465)

[*Hình 4: Biểu đồ tuần tự tìm kiếm sách 23*](#_Toc190357466)

[*Hình 5: Biểu đồ tuần tự tìm kiếm sách 24*](#_Toc190357467)

[*Hình 6: Biểu đồ tuần tự xóa sách 24*](#_Toc190357468)

[*Hình 7: Biểu đồ tuần tự xem danh sách của Sách 25*](#_Toc190357469)

[*Hình 8: Kiến trúc hệ thống RMI 25*](#_Toc190357470)

[*Hình 9: Các thư mục mã nguồn 28*](#_Toc190357471)

[*Hình 10: Giao diện chính 37*](#_Toc190357472)

[*Hình 11: Giao diện thêm sách 37*](#_Toc190357473)

[*Hình 12: Giao diện tìm sách 38*](#_Toc190357474)

[*Hình 13: Giao diện cập nhật sách 38*](#_Toc190357475)

[*Hình 14, 15: Giao diện xóa sách 39*](#_Toc190357476)

# DANH MỤC THUẬT NGỮ

|  |  |  |
| --- | --- | --- |
| STT | Thuật ngữ | Định nghĩa |
| 1 | Java RMI (Remote Method Invocation) | Cơ chế cho phép một chương trình Java gọi phương thức từ xa trên một máy khác thông qua mạng. |
| 2 | Client-Server | Mô hình ứng dụng phân tán trong đó máy khách gửi yêu cầu và máy chủ xử lý phản hồi. |
| 3 | RMI Registry | Dịch vụ đăng ký các đối tượng từ xa, giúp máy khách tìm và gọi các phương thức từ xa. |
| 4 | Giao diện từ xa (Remote Interface) | Giao diện Java khai báo các phương thức có thể gọi từ xa trong RMI. |
| 5 | Java Swing | Thư viện Java dùng để xây dựng giao diện đồ họa (GUI) cho ứng dụng máy tính. |
| 6 | Server | Thành phần trong mô hình Client-Server chịu trách nhiệm xử lý yêu cầu và quản lý dữ liệu. |
| 7 | Client | Thành phần trong mô hình Client-Server gửi yêu cầu đến server và nhận kết quả phản hồi. |

# LỜI MỞ ĐẦU

Ngày nay, công nghệ thông tin đóng vai trò quan trọng trong mọi lĩnh vực của đời sống, đặc biệt là trong quản lý và xử lý dữ liệu. Trong bối cảnh đó, các hệ thống phần mềm quản lý ngày càng trở nên cần thiết nhằm nâng cao hiệu suất làm việc, giảm thiểu sai sót và tối ưu hóa quy trình vận hành.

Trong các trường đại học, việc quản lý thư viện và sách đóng vai trò quan trọng trong hoạt động học tập, nghiên cứu của sinh viên và giảng viên. Tuy nhiên, các phương pháp quản lý truyền thống còn nhiều hạn chế như mất nhiều thời gian, khó kiểm soát số lượng sách, và không hỗ trợ tốt trong việc tra cứu, cập nhật thông tin. Để giải quyết vấn đề này, ứng dụng công nghệ vào quản lý sách là một giải pháp tối ưu.

Đề tài ***"Xây dựng chương trình quản lý sách cho đại học sử dụng kỹ thuật lập trình phân tán Java RMI"*** được thực hiện nhằm xây dựng một hệ thống quản lý sách hiện đại, hỗ trợ các chức năng cơ bản như thêm, sửa, xóa, tìm kiếm sách, giúp cho việc quản lý thư viện trở nên dễ dàng và hiệu quả hơn. Trong đồ án này, công nghệ Java RMI (Remote Method Invocation) được sử dụng để hiện thực một hệ thống phân tán, cho phép các máy khách truy cập và tương tác với hệ thống từ xa một cách linh hoạt và an toàn.

Nhóm chúng em xin gửi lời cảm ơn chân thành đến TS. Trần Đăng Hoan, người đã tận tình hướng dẫn, cung cấp những kiến thức quý báu và đóng góp ý kiến quan trọng trong suốt quá trình thực hiện đề tài. Sự hỗ trợ và động viên của cô đã giúp chúng em có thêm động lực để hoàn thành dự án, đồng thời trau dồi thêm kỹ năng về lập trình phân tán, quản lý dữ liệu và phát triển phần mềm.

Chúng em hy vọng rằng dự án này không chỉ là một sản phẩm học thuật mà còn có thể đóng góp thiết thực vào thực tiễn, giúp cải thiện hệ thống quản lý sách và mở ra những hướng nghiên cứu mới trong lĩnh vực công nghệ phần mềm.

# CHƯƠNG I. TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN

## Giới thiệu

### Định nghĩa lập trình phân tán

Hệ thống phân tán (Distributed System) là một tập hợp các máy tính độc lập kết nối với nhau thông qua mạng và phối hợp để thực hiện một nhiệm vụ chung. Các hệ thống này hoạt động như một hệ thống thống nhất, mặc dù các thành phần của nó có thể chạy trên nhiều nền tảng phần cứng và phần mềm khác nhau.

Hệ thống phân tán có thể được sử dụng trong nhiều lĩnh vực khác nhau, từ các hệ thống quản lý dữ liệu lớn, ứng dụng web, đến các mạng xã hội và dịch vụ điện toán đám mây.

### Đặc điểm của hệ thống phân tán

Hệ thống phân tán có một số đặc điểm quan trọng sau:

* **Tính đồng thời (Concurrency)**  
  Nhiều tiến trình có thể thực thi cùng lúc trên các nút khác nhau, giúp tăng hiệu suất hệ thống.
* **Không chia sẻ bộ nhớ (No Shared Memory)**

Các máy tính trong hệ thống phân tán không có vùng nhớ chung, thay vào đó, chúng giao tiếp với nhau thông qua các cơ chế truyền tin như Remote Procedure Call (RPC) hoặc Remote Method Invocation (RMI).

* **Lỗi độc lập (Independent Failures)**

Mỗi nút trong hệ thống có thể gặp lỗi độc lập với các nút khác. Điều này yêu cầu hệ thống phải có cơ chế chịu lỗi để đảm bảo tính liên tục trong hoạt động.

* **Không có đồng hồ toàn cục (Lack of Global Clock)**

Không có một bộ đếm thời gian chung giữa các máy trong hệ thống, điều này gây ra thách thức trong việc đồng bộ hóa dữ liệu.

* **Tính không đồng nhất (Heterogeneity)**

Các thành phần trong hệ thống có thể chạy trên nhiều nền tảng khác nhau, từ hệ điều hành, phần cứng đến các giao thức mạng.

* **Tính trong suốt (Transparency)**

Hệ thống cần che giấu sự phân tán của các thành phần để người dùng có thể sử dụng hệ thống một cách dễ dàng mà không cần quan tâm đến cách các tài nguyên được phân bổ và xử lý phía sau.

### Lý do sử dụng hệ thống phân tán

Hệ thống phân tán được sử dụng rộng rãi vì mang lại nhiều lợi ích:

* **Tăng khả năng mở rộng (Increased Scalability)**  
  Các hệ thống phân tán có thể dễ dàng mở rộng bằng cách thêm các nút mới vào hệ thống mà không ảnh hưởng đến các thành phần hiện có.
* **Cải thiện độ tin cậy (Improved Reliability)**  
  Nếu một phần của hệ thống gặp sự cố, các phần còn lại có thể tiếp tục hoạt động, giúp giảm thiểu thời gian gián đoạn.
* **Chia sẻ tài nguyên hiệu quả (Enhanced Resource Sharing)**  
  Hệ thống cho phép chia sẻ tài nguyên (như dữ liệu, bộ xử lý, dung lượng lưu trữ) giữa nhiều người dùng hoặc ứng dụng khác nhau.
* **Tính linh hoạt (Flexibility)**  
  Hệ thống có thể hoạt động trên nhiều nền tảng phần cứng và phần mềm khác nhau, giúp doanh nghiệp linh hoạt trong việc triển khai và nâng cấp hệ thống.

### Ví dụ về hệ thống phân tán

Ta có 1 số ví dụ về hệ thống phân tán như Các dịch vụ như Amazon Web Services (AWS), Google Cloud Platform (GCP) và Microsoft Azure cung cấp tài nguyên tính toán theo mô hình phân tán.

Hoặc Facebook, Twitter, LinkedIn sử dụng hệ thống phân tán để lưu trữ và xử lý lượng lớn dữ liệu từ hàng triệu người dùng trên toàn cầu.

## Các khái niệm cơ bản

Hệ thống phân tán bao gồm nhiều thành phần hoạt động cùng nhau để thực hiện một nhiệm vụ chung. Để hiểu rõ hơn về cách thức hoạt động và thiết kế của hệ thống phân tán, cần nắm vững các khái niệm cơ bản như kiến trúc Client-Server, kiến trúc Peer-to-Peer, và các cơ chế giao tiếp giữa các tiến trình (IPC).

### Kiến trúc Client-Server

**1. Định nghĩa**

Kiến trúc **Client-Server** là mô hình phổ biến trong hệ thống phân tán, trong đó client (máy khách) gửi yêu cầu đến server (máy chủ), server xử lý yêu cầu và gửi phản hồi lại cho client. Mô hình này giúp phân chia rõ ràng trách nhiệm giữa client và server, đảm bảo tính tổ chức trong việc xử lý dữ liệu.

* **Thành phần chính**
  + **Client (Máy khách)**
    - Gửi yêu cầu đến server để xử lý và nhận phản hồi.
    - Thường là các ứng dụng như trình duyệt web, ứng dụng di động hoặc phần mềm desktop.
  + **Server (Máy chủ)**
    - Xử lý yêu cầu từ client và quản lý dữ liệu tập trung.
    - Chứa các logic nghiệp vụ và cơ sở dữ liệu của hệ thống.
* **Ưu điểm**
  + Dễ triển khai và quản lý do dữ liệu tập trung tại server.
  + Bảo mật cao hơn vì dữ liệu được lưu trữ trên server thay vì trên các thiết bị cá nhân.
* **Nhược điểm**
  + Điểm nghẽn tập trung: Nếu server bị quá tải hoặc gặp sự cố, toàn bộ hệ thống có thể bị gián đoạn.
  + Khả năng mở rộng hạn chế: Khi số lượng client tăng lên, server có thể không đáp ứng đủ yêu cầu.
* **Ứng dụng phổ biến**
  + **Hệ thống quản lý cơ sở dữ liệu (DBMS):** MySQL, PostgreSQL, SQL Server.
  + **Các trang web và dịch vụ trực tuyến:** Google, Facebook, Amazon.
  + **Ứng dụng doanh nghiệp:** Hệ thống quản lý nhân sự, tài chính, bán hàng.

### Kiến trúc Peer-to-Peer (P2P)

Khác với kiến trúc Client-Server, kiến trúc Peer-to-Peer (P2P) không phân biệt rõ ràng giữa client và server. Trong mô hình này, tất cả các nút (nodes) trong hệ thống đều có vai trò ngang hàng và có thể vừa gửi vừa nhận yêu cầu từ các nút khác. Điều này giúp phân tán tải và giảm thiểu điểm nghẽn tập trung.

* **Thành phần chính:**
  + **Peer (Nút mạng)**
    - Là thiết bị tham gia hệ thống, có thể vừa gửi yêu cầu vừa xử lý yêu cầu từ các nút khác.
    - Chia sẻ tài nguyên như dữ liệu, băng thông và sức mạnh tính toán với các nút khác.
* **Ưu điểm**
  + Không có điểm nghẽn tập trung do không có máy chủ trung tâm.
  + Mở rộng dễ dàng bằng cách thêm các nút mới vào hệ thống.
* **Nhược điểm**
  + Khó quản lý do không có máy chủ trung tâm giám sát và kiểm soát dữ liệu.
  + Tốc độ không ổn định vì phụ thuộc vào các nút trong hệ thống.
* **Ứng dụng phổ biến**
  + Một số app có thể chia sẻ tệp tin điển hình là BitTorrent.
  + Tiền mã hóa (Cryptocurrency): Bitcoin, Ethereum.
  + Truyền thông trực tuyến như là Skype, Zoom.

### Cơ chế giao tiếp giữa các tiến trình (IPC – Inter-Process Communication)

Cơ chế giao tiếp giữa các tiến trình (IPC) cho phép các tiến trình trong hệ thống phân tán trao đổi dữ liệu và phối hợp thực hiện nhiệm vụ. Có nhiều cơ chế khác nhau để thực hiện IPC, bao gồm:

* **Message Passing (Truyền tin):**
  + Sử dụng các thông điệp để trao đổi dữ liệu giữa các tiến trình thông qua mạng.
  + **Công nghệ phổ biến:**
    - **Sockets:** Giao tiếp giữa hai tiến trình thông qua địa chỉ IP và cổng.
    - **Remote Procedure Call (RPC):** Cho phép gọi hàm từ xa như gọi hàm cục bộ.
  + **Ưu điểm:** Không cần chia sẻ bộ nhớ, dễ mở rộng cho hệ thống phân tán.
  + **Nhược điểm:** Có độ trễ do phải truyền dữ liệu qua mạng.
* **Shared Memory (Bộ nhớ chia sẻ):**
  + Sử dụng một vùng bộ nhớ chung để các tiến trình truy cập và trao đổi dữ liệu.
  + Ưu điểm: Tốc độ nhanh hơn so với truyền tin.
  + Nhược điểm: Cần cơ chế đồng bộ để tránh xung đột dữ liệu.
* **Remote Method Invocation (RMI):**
  + Cho phép gọi phương thức từ xa giữa các tiến trình đang chạy trên các máy khác nhau trong Java.
  + Ưu điểm: Dễ sử dụng trong hệ sinh thái Java, hỗ trợ giao tiếp đối tượng từ xa.
  + Nhược điểm: Phụ thuộc vào mạng và chỉ hỗ trợ trong môi trường Java.

## Các thách thức trong thiết kế hệ thống phân tán

Thiết kế hệ thống phân tán đối mặt với nhiều thách thức phức tạp do tính chất phân tán của dữ liệu và tiến trình. Các thách thức chính bao gồm kiểm soát đồng thời, khả năng chịu lỗi và tính nhất quán của dữ liệu.

### Kiểm soát đồng thời (Concurrency Control)

**1. Định nghĩa**

Kiểm soát đồng thời là quá trình đảm bảo rằng nhiều tiến trình hoặc người dùng có thể truy cập và thao tác trên cùng một dữ liệu mà không gây ra xung đột hoặc lỗi dữ liệu. Điều này đặc biệt quan trọng trong hệ thống phân tán khi có nhiều máy khách thực hiện các yêu cầu cùng lúc.

**2. Các vấn đề quan trọng trong kiểm soát đồng thời**

*a. Loại trừ tương hỗ (Mutual Exclusion)*

* **Định nghĩa:** Đảm bảo rằng tại một thời điểm, chỉ có một tiến trình được phép truy cập tài nguyên chia sẻ, như một bản ghi trong cơ sở dữ liệu hoặc một tệp tin.
* **Thách thức:** Việc triển khai loại trừ tương hỗ trong hệ thống phân tán khó khăn hơn so với hệ thống tập trung do không có bộ nhớ chung và không có đồng hồ toàn cục.
* **Một số giải pháp:**
  + **Thuật toán Lamport’s Bakery:** Sử dụng đồng hồ logic để xác định thứ tự truy cập tài nguyên.
  + **Thuật toán Ricart-Agrawala:** Sử dụng thông điệp yêu cầu và xác nhận giữa các tiến trình để điều phối quyền truy cập.

*b. Đồng bộ hóa (Synchronization)*

* **Định nghĩa:** Đồng bộ hóa đảm bảo rằng các tiến trình thực hiện các thao tác trên dữ liệu theo đúng thứ tự để duy trì tính nhất quán.
* **Thách thức:**
  + Không có đồng hồ toàn cục khiến cho việc sắp xếp thứ tự sự kiện trở nên khó khăn.
  + Dễ xảy ra tình trạng race condition khi nhiều tiến trình cùng thao tác trên một dữ liệu.
* **Giải pháp:**
  + **Vector Clocks:** Sử dụng mảng đồng hồ logic để theo dõi thứ tự sự kiện trong hệ thống phân tán.
  + **Distributed Locking (Khóa phân tán):** Sử dụng các giao thức như Zookeeper hoặc Chubby để khóa dữ liệu trên toàn bộ hệ thống.

**c. Deadlock**

* **Định nghĩa:** Deadlock xảy ra khi một tập hợp các tiến trình bị khóa và chờ nhau giải phóng tài nguyên, dẫn đến hệ thống bị đình trệ.
* **Thách thức:**
  + Trong hệ thống phân tán, phát hiện và giải quyết deadlock khó khăn hơn do không có bộ nhớ chung và các tiến trình nằm trên nhiều máy khác nhau.
* **Giải pháp:**
  + **Deadlock Prevention (Phòng ngừa deadlock):**
    - **Ordering Resources (Sắp xếp thứ tự tài nguyên):** Yêu cầu tài nguyên theo thứ tự xác định trước để tránh vòng lặp chờ đợi.
    - **Timeout (Giới hạn thời gian chờ):** Hủy bỏ yêu cầu nếu quá thời gian chờ đợi.
  + **Deadlock Detection (Phát hiện deadlock):**
    - Sử dụng **Wait-For Graph (Đồ thị chờ đợi)** để phát hiện chu trình deadlock.
  + **Deadlock Recovery (Khôi phục deadlock):**
    - Hủy bỏ hoặc khởi động lại một số tiến trình để phá vỡ deadlock.

### Khả năng chịu lỗi (Fault Tolerance)

**1. Định nghĩa**

Khả năng chịu lỗi đảm bảo rằng hệ thống có thể tiếp tục hoạt động ngay cả khi một số thành phần gặp sự cố. Điều này đặc biệt quan trọng trong hệ thống phân tán, nơi mà lỗi có thể xảy ra do mất kết nối mạng, lỗi phần cứng hoặc phần mềm.

**2. Các kỹ thuật đảm bảo khả năng chịu lỗi**

*a. Phát hiện và phục hồi lỗi (Fault Detection and Recovery)*

* **Định nghĩa:** Xác định các thành phần bị lỗi và khôi phục chúng để đảm bảo tính liên tục trong hoạt động của hệ thống.
* **Giải pháp:**
  + **Heartbeat (Nhịp đập):** Các nút trong hệ thống gửi tín hiệu định kỳ để thông báo trạng thái của chúng. Nếu không nhận được tín hiệu, hệ thống sẽ xác định nút đó đã bị lỗi.
  + **Replication (Nhân bản):** Sao chép dữ liệu hoặc tiến trình trên nhiều nút để khi một nút gặp lỗi, các nút còn lại có thể tiếp tục xử lý.

*b. Nhân bản dữ liệu (Replication)*

* **Định nghĩa:** Dữ liệu được sao chép trên nhiều máy để đảm bảo hệ thống vẫn hoạt động khi một hoặc nhiều nút bị lỗi.
* **Các mô hình nhân bản dữ liệu:**
  + **Primary-Backup Replication:** Một nút chính (Primary) xử lý yêu cầu, các nút sao lưu (Backup) đồng bộ dữ liệu từ nút chính.
  + **Quorum-based Replication:** Đảm bảo một số lượng nút tối thiểu (quorum) đồng thuận trước khi xác nhận thay đổi dữ liệu.

*c. Checkpointing (Lưu điểm kiểm tra)*

* **Định nghĩa:** Hệ thống lưu trạng thái của tiến trình tại một thời điểm nhất định, cho phép khôi phục lại từ trạng thái đó khi xảy ra lỗi.
* **Ứng dụng:**
  + **Cơ sở dữ liệu (Database):** Checkpointing giúp khôi phục trạng thái trước khi hệ thống bị lỗi.
  + **Tính toán phân tán (Distributed Computing):** Checkpointing lưu lại tiến trình xử lý, giúp tiếp tục công việc khi có lỗi xảy ra.

### Tính nhất quán của dữ liệu (Data Consistency)

**1. Định nghĩa**

Tính nhất quán của dữ liệu đảm bảo rằng tất cả các bản sao của một dữ liệu trong hệ thống phân tán đều có cùng một trạng thái sau mỗi lần cập nhật.

**2. Các mô hình nhất quán phổ biến**

| **Mô hình nhất quán** | **Định nghĩa** | **Ứng dụng** |
| --- | --- | --- |
| **Nhất quán mạnh (Strong Consistency)** | Mọi thay đổi dữ liệu được nhìn thấy ngay lập tức trên tất cả các nút. | Hệ thống tài chính, ngân hàng. |
| **Nhất quán yếu (Weak Consistency)** | Không đảm bảo rằng tất cả các nút có dữ liệu giống nhau ngay lập tức. | Bộ nhớ đệm, mạng xã hội. |
| **Nhất quán cuối cùng (Eventual Consistency)** | Sau một khoảng thời gian, tất cả các bản sao của dữ liệu sẽ đồng bộ với nhau. | Hệ thống DNS, dịch vụ web. |

**3. Các kỹ thuật nhân bản dữ liệu (Data Replication Techniques)**

* **Active Replication:** Mọi bản sao nhận yêu cầu cập nhật cùng lúc, đảm bảo dữ liệu nhất quán ngay lập tức.
* **Passive Replication:** Một nút chính xử lý yêu cầu và cập nhật các nút sao lưu.
* **Quorum-based Replication:** Yêu cầu số lượng nút đồng thuận trước khi xác nhận thay đổi dữ liệu.

## Xây dựng và triển khai hệ thống phân tán

### Cân nhắc trong thiết kế

Khi thiết kế hệ thống phân tán, cần xem xét kỹ các yếu tố quan trọng để đảm bảo hiệu suất, khả năng mở rộng, tính nhất quán và bảo mật. Trước hết, hệ thống phải có hiệu suất cao và khả năng mở rộng tốt bằng cách phân chia tải công việc hợp lý và có thể bổ sung tài nguyên khi cần mà không gây gián đoạn. Đồng thời, cần đảm bảo tính nhất quán và sẵn sàng của dữ liệu thông qua các mô hình nhất quán phù hợp như **Strong Consistency** hoặc **Eventual Consistency**, đồng thời cân nhắc kỹ lưỡng giữa **Consistency, Availability** và **Partition Tolerance** theo **CAP Theorem**.

Ngoài ra, bảo mật cũng là một yếu tố quan trọng. Hệ thống phải áp dụng các biện pháp xác thực, phân quyền chặt chẽ và mã hóa dữ liệu để đảm bảo an toàn thông tin. Cuối cùng, thiết kế hệ thống cần có tính linh hoạt và dễ bảo trì bằng cách áp dụng kiến trúc module hóa hoặc **Microservices**. Điều này giúp hệ thống dễ dàng phát triển thêm tính năng mới và bảo trì trong suốt vòng đời sử dụng. Việc xem xét cẩn thận các yếu tố trên sẽ giúp xây dựng hệ thống phân tán hiệu quả, an toàn và dễ mở rộng.

### Lựa chọn công nghệ phù hợp

Có nhiều ngôn ngữ hỗ trợ lập trình phân tán, Java là một trong số đó với hai thư viện chuẩn là RMI và CORBA. Thực chất RMI là một cơ chế gọi phương thức từ xa đã được tích hợp trong ngôn ngữ Java. Vì Java là một ngôn ngữ lập trình hướng đối tượng, nên phương pháp lập trình trong RMI là phương pháp hướng đối tượng, do đó các thao tác hay các lời gọi phương thức đều liên quan đến đối tượng.

**RMI (Remote Method Invocation)** là một cơ chế gọi phương thức từ xa được tích hợp sẵn trong ngôn ngữ lập trình Java, cho phép một chương trình Java trên một máy có thể gọi và thực thi các phương thức được cài đặt trên một máy khác thông qua mạng. RMI được xây dựng dựa trên mô hình hướng đối tượng, trong đó các lời gọi phương thức đều liên quan đến các đối tượng từ xa.

Trong kiến trúc RMI, mỗi đối tượng từ xa được đăng ký và quản lý bởi RMI Registry. Máy khách (Client) có thể tìm kiếm và gọi các phương thức của đối tượng từ xa thông qua tham chiếu nhận được từ RMI Registry. Cơ chế hoạt động của RMI bao gồm việc truyền tham số và nhận kết quả giữa Client và Server thông qua việc tuần tự hóa đối tượng (Object Serialization), giúp dữ liệu có thể di chuyển giữa các máy một cách an toàn và hiệu quả.

Kiến trúc RMI bao gồm các thành phần chính như:

* Remote Interface (Giao diện từ xa): Xác định các phương thức có thể gọi từ xa, tương tự như các phương thức trong giao diện thông thường, nhưng phải kế thừa từ giao diện java.rmi.Remote.
* Server (Máy chủ): Cài đặt các phương thức được định nghĩa trong Remote Interface và đăng ký các đối tượng từ xa với RMI Registry để máy khách có thể tìm kiếm và sử dụng.
* Client (Máy khách): Tìm kiếm các đối tượng từ xa đã được đăng ký trên máy chủ và gọi các phương thức từ xa như gọi các phương thức cục bộ thông qua Stub – đại diện cục bộ của đối tượng từ xa.

Điểm đặc biệt của RMI là việc cả Client và Server đều sử dụng Java, giúp lập trình viên có thể dễ dàng xây dựng các ứng dụng phân tán mà không cần lo ngại về sự khác biệt giữa các nền tảng hay ngôn ngữ lập trình khác nhau.

**CORBA (Common Object Request Broker Architecture)** là đặc tả của OMG (Object Management Group) dành cho việc đạt được tính tương tác giữa các nút tính toán phân tán. Mục tiêu của CORBA là định nghĩa một cấu trúc mà có thể cho phép những môi trường không đồng nhất có thể liên lạc tại mức đối tượng, không tâm đến người nào đã thiết kế ra hai điểm cuối của ứng dụng phân tán Corba là một ngôn ngữ trung lập được thực thi rộng rãi hơn trên các nền platform khác nhau. Corba sử dụng một giao thức gọi là IIOP (Internet Inter-ORB Protocol) để liên lạc giữa các hệ thống khác nhau.

Sự khác nhau giữa RMI và CORBA:

* RMI là một phần của bộ J2SDK và là các hàm thư viện hỗ trợ các lời gọi phương thức từ xa và trả về giá trị cho các ứng dụng tính toán phân tán Ở đây ngôn ngữ Java được sử dụng ở cả phía gọi và phía bên phương thức được gọi.
* CORBA là một chuẩn công nghiệp cho phép gọi các phương thức từ xa và nhận kết quả trả về Nhưng không giống như RMI, nó có thể được sử dụng khi bên phía gọi và bên phía phương thức được gọi có thể sử dụng các ngôn ngữ lập trình khác nhau, bao gồm cả trường hợp là cả hai bên đều không sử dụng ngôn ngữ Java.
* RMI là một tập các hàm thư viện đơn giản vì cả hai bên đều sử dụng cùng một ngôn ngữ lập trình và kiến trúc máy Điều này sẽ làm cho vấn để triệu gọi phương thức từ xa dễ giải quyết hơn.
* Bộ phát triển J2SDK cũng hỗ trợ cả RMI và CORBA, cho phép một đối tượng Java gọi một đối tượng CORBA sử dụng hai cách tiếp cận khác nhau.

# CHƯƠNG II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG



## Xây dựng và triển khai hệ thống phân tán



### Yêu cầu chức năng

Hệ thống quản lý sách cho đại học sử dụng kỹ thuật lập trình phân tán Java RMI được xây dựng nhằm đáp ứng nhu cầu quản lý sách một cách hiệu quả, dễ sử dụng và đảm bảo tính phân tán, đồng bộ dữ liệu giữa Client và Server. Hệ thống này hỗ trợ các chức năng như thêm mới, chỉnh sửa, xóa, tìm kiếm sách và hiển thị danh sách sách hiện có trong thư viện.

*a. Thêm sách mới (Add Book)*

Hệ thống cho phép người dùng thêm thông tin sách vào cơ sở dữ liệu thông qua các thao tác sau:

* **Nhập thông tin sách bao gồm:**
  + **ISBN:** Mã số sách quốc tế duy nhất.
  + **Tên sách (Title):** Tên đầy đủ của sách.
  + **Tác giả (Author):** Tên tác giả của sách.
  + **Nhà xuất bản (Publisher):** Thông tin về nhà xuất bản.
  + **Năm xuất bản (Year):** Năm xuất bản của sách.
  + **Số lượng (Quantity):** Số lượng bản sách hiện có trong thư viện.
* **Thao tác thêm sách:**
  + Người dùng nhập đầy đủ thông tin sách và nhấn nút **"Thêm Sách"** để lưu thông tin vào cơ sở dữ liệu.
  + Hệ thống sẽ kiểm tra thông tin hợp lệ và lưu vào cơ sở dữ liệu nếu không có lỗi.
  + Nếu ISBN đã tồn tại, hệ thống sẽ thông báo lỗi và yêu cầu người dùng nhập lại.

*b. Tìm kiếm sách*

Hệ thống sẽ cung cấp cho ta những chức năng tìm kiếm sách theo nhiều tiêu chí khác nhau:

* **Tiêu chí tìm kiếm:**
  + **Tìm kiếm theo:** ISBN, Title, Author, Publisher, Year, Quantity.
* **Cách thức hoạt động:**
  + Người dùng nhập từ khóa tìm kiếm vào ô tìm kiếm trên giao diện.
  + Hệ thống sẽ lọc và hiển thị các sách phù hợp với từ khóa tìm kiếm.
  + Kết quả tìm kiếm được hiển thị trực tiếp trên bảng danh sách sách.

*c. Cập nhật thông tin sách*

Người dùng có thể cập nhật thông tin sách đã có trong cơ sở dữ liệu bằng cách:

* **Thao tác chỉnh sửa:**
  + Chọn sách cần chỉnh sửa từ danh sách hiển thị hoặc nhập **ISBN** của sách muốn cập nhật.
  + Thay đổi các thông tin mong muốn như ISBN, Title, Author, Publisher, Year, Quantity.
  + Nhấn nút **"Chỉnh sửa sách"** để lưu các thay đổi vào cơ sở dữ liệu.
* **Kiểm tra thông tin:**
  + Hệ thống kiểm tra tính hợp lệ của dữ liệu trước khi lưu lại thay đổi.
  + Nếu có lỗi, hệ thống sẽ thông báo cho người dùng và yêu cầu chỉnh sửa lại.

*d. Xóa sách*

Hệ thống hỗ trợ xóa thông tin sách thông qua các thao tác sau:

* **Cách thức thực hiện:**
  + Người dùng chọn sách từ danh sách hiển thị.
  + Nhấn nút **"Xóa sách"** để xóa thông tin sách tương ứng khỏi cơ sở dữ liệu.
* **Xác nhận trước khi xóa:**
  + Hệ thống sẽ yêu cầu xác nhận trước khi thực hiện thao tác xóa để tránh nhầm lẫn.
  + Nếu người dùng xác nhận, hệ thống sẽ xóa sách khỏi cơ sở dữ liệu và cập nhật lại danh sách trên giao diện.

*e. Xem danh sách của sách (View book list)*

Hệ thống hiển thị danh sách sách hiện có dưới dạng bảng với các thông tin chi tiết:

* **Thông tin hiển thị:** ISBN, Title, Author, Publisher, Year, Quantity
* **Chức năng hỗ trợ:**
  + Người dùng có thể xem danh sách toàn bộ sách trong thư viện trực tiếp trên giao diện.
  + Bảng hiển thị cho phép người dùng dễ dàng theo dõi và kiểm tra thông tin sách.
  + Danh sách được tự động cập nhật sau khi thêm, xóa hoặc chỉnh sửa sách.

*f. Đồng bộ dữ liệu giữa Client và Server*

Hệ thống sử dụng Java RMI để đảm bảo đồng bộ dữ liệu giữa Client và Server:

* Mọi thao tác thêm, sửa, xóa, và xem dữ liệu đều được thực hiện trên giao diện Client.
* Client gửi yêu cầu đến Server thông qua RMI để xử lý dữ liệu.
* Server thực hiện thao tác trên cơ sở dữ liệu và phản hồi kết quả về cho Client.

### Yêu cầu phi chức năng

*a. Hiệu suất*

* Hệ thống phải phản hồi nhanh, thời gian xử lý mỗi thao tác không được quá lâu.
* Đảm bảo hoạt động ổn định ngay cả khi có nhiều người dùng thao tác đồng thời.

*b. Bảo mật*

* Chỉ người dùng có quyền mới được phép thêm, chỉnh sửa hoặc xóa thông tin sách.
* Kiểm tra dữ liệu đầu vào để tránh lỗi và đảm bảo tính toàn vẹn của dữ liệu.

*c. Giao diện người dùng*

* Giao diện đơn giản, thân thiện, dễ sử dụng với các nút thao tác rõ ràng: **Thêm, Xem, Xóa, Chỉnh sửa, Tìm kiếm.**
* Bảng hiển thị thông tin sách trực quan, dễ tìm kiếm và quản lý dữ liệu.

*d. Khả năng mở rộng*

* Hệ thống có khả năng mở rộng để bổ sung các chức năng mới như:
  + Tìm kiếm nâng cao theo nhiều tiêu chí kết hợp.
  + Thống kê số lượng sách theo tác giả, nhà xuất bản hoặc năm xuất bản.

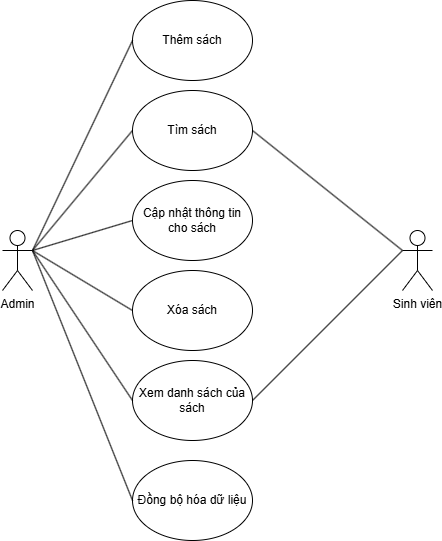
*e. Độ tương thích*

* Ứng dụng phải tương thích với các nền tảng hỗ trợ Java từ phiên bản Java 8 trở lên.

*f. Độ tin cậy*

* Đảm bảo hệ thống hoạt động ổn định, không xảy ra lỗi khi có nhiều thao tác đồng thời.
* Tự động ghi nhật ký (log) các thao tác để hỗ trợ kiểm tra và khắc phục sự cố khi cần thiết.

## Mô hình Use-Case



Hình 1: Mô hình Use-Case

*a. Thêm Sách*

**Mô tả:** Người dùng thêm thông tin sách mới vào cơ sở dữ liệu.

**Tác nhân:** User

**Luồng sự kiện chính:**

1. Người dùng chọn chức năng **“Thêm Sách”** trên giao diện.
2. Hệ thống hiển thị form nhập thông tin sách bao gồm:
   * ISBN (Mã số sách)
   * Tên sách
   * Tác giả
   * Nhà xuất bản
   * Năm xuất bản
   * Số lượng sách
3. Người dùng nhập đầy đủ thông tin và nhấn nút **“Thêm sách”**.
4. Client gửi yêu cầu thêm sách đến Server thông qua **Java RMI**.
5. Server kiểm tra tính hợp lệ của dữ liệu và lưu thông tin sách vào cơ sở dữ liệu.
6. Server phản hồi kết quả cho Client.
7. Client thông báo kết quả cho người dùng:
   * Nếu thành công: Thông báo **“Thêm sách thành công”** và cập nhật danh sách sách.
   * Nếu thất bại: Hiển thị thông báo lỗi (ví dụ: ISBN đã tồn tại).

**Luồng sự kiện phụ:**

* Nếu thông tin nhập không hợp lệ, hệ thống yêu cầu người dùng nhập lại.
* Nếu kết nối với Server thất bại, Client sẽ thông báo lỗi kết nối.

*b. Tìm Sách*

**Mô tả:** Người dùng tìm kiếm sách dựa trên các tiêu chí khác nhau.

**Tác nhân:** User

**Luồng sự kiện chính:**

1. Người dùng nhập từ khóa tìm kiếm trên giao diện (theo ISBN, Title, Author, Publisher, Year, Quantity).
2. Nhấn nút **“Tìm kiếm”**.
3. Client gửi yêu cầu tìm kiếm đến Server qua **Java RMI**.
4. Server thực hiện truy vấn cơ sở dữ liệu để tìm kiếm các sách phù hợp.
5. Server trả về danh sách các sách khớp với từ khóa tìm kiếm.
6. Client hiển thị danh sách kết quả tìm kiếm trên giao diện.

**Luồng sự kiện phụ:**

* Nếu không tìm thấy sách phù hợp, hệ thống thông báo **“Không có kết quả phù hợp”**.
* Nếu từ khóa tìm kiếm không hợp lệ, hệ thống yêu cầu người dùng nhập lại.

*c. Cập nhật thông tin sách*

**Mô tả:** Người dùng chỉnh sửa thông tin sách có trong cơ sở dữ liệu.

**Tác nhân:** User

**Luồng sự kiện chính:**

1. Người dùng chọn sách cần chỉnh sửa từ danh sách hoặc nhập ISBN của sách muốn cập nhật.
2. Hệ thống hiển thị form chỉnh sửa với các thông tin hiện tại của sách.
3. Người dùng chỉnh sửa các thông tin mong muốn (Tên sách, Tác giả, Nhà xuất bản, Năm xuất bản, Số lượng).
4. Nhấn nút **“Cập nhật”** để lưu lại các thay đổi.
5. Client gửi yêu cầu cập nhật đến Server thông qua **Java RMI**.
6. Server kiểm tra tính hợp lệ và cập nhật thông tin sách trong cơ sở dữ liệu.
7. Server phản hồi kết quả cho Client.
8. Client cập nhật danh sách sách trên giao diện và thông báo kết quả cho người dùng.

**Luồng sự kiện phụ:**

* Nếu thông tin cập nhật không hợp lệ, hệ thống yêu cầu người dùng chỉnh sửa lại.
* Nếu ISBN không tồn tại, hệ thống hiển thị thông báo lỗi.

*d. Xóa sách*

**Mô tả:** Người dùng xóa thông tin sách khỏi cơ sở dữ liệu.

**Tác nhân:** User

**Luồng sự kiện chính:**

1. Người dùng chọn sách muốn xóa từ danh sách hiển thị hoặc nhập ISBN của sách cần xóa.
2. Nhấn nút **“Xóa sách”**.
3. Hệ thống yêu cầu xác nhận việc xóa sách.
4. Người dùng xác nhận **“Yes”** hoặc **“No”**.
5. Nếu người dùng xác nhận đồng ý:
   * Client gửi yêu cầu xóa sách đến Server thông qua **Java RMI**.
   * Server kiểm tra sự tồn tại của sách và xóa sách khỏi cơ sở dữ liệu.
   * Server phản hồi kết quả cho Client.
   * Client cập nhật danh sách sách trên giao diện và thông báo kết quả cho người dùng.
6. Nếu người dùng chọn **“Hủy”**, hệ thống dừng thao tác và quay về giao diện chính.

**Luồng sự kiện phụ:**

* Nếu sách không tồn tại, hệ thống hiển thị thông báo lỗi.
* Nếu kết nối Server thất bại, Client sẽ thông báo lỗi kết nối.

*e. Xem danh sách của Sách*

**Mô tả:** Hiển thị danh sách sách hiện có trong thư viện.

**Tác nhân:** User

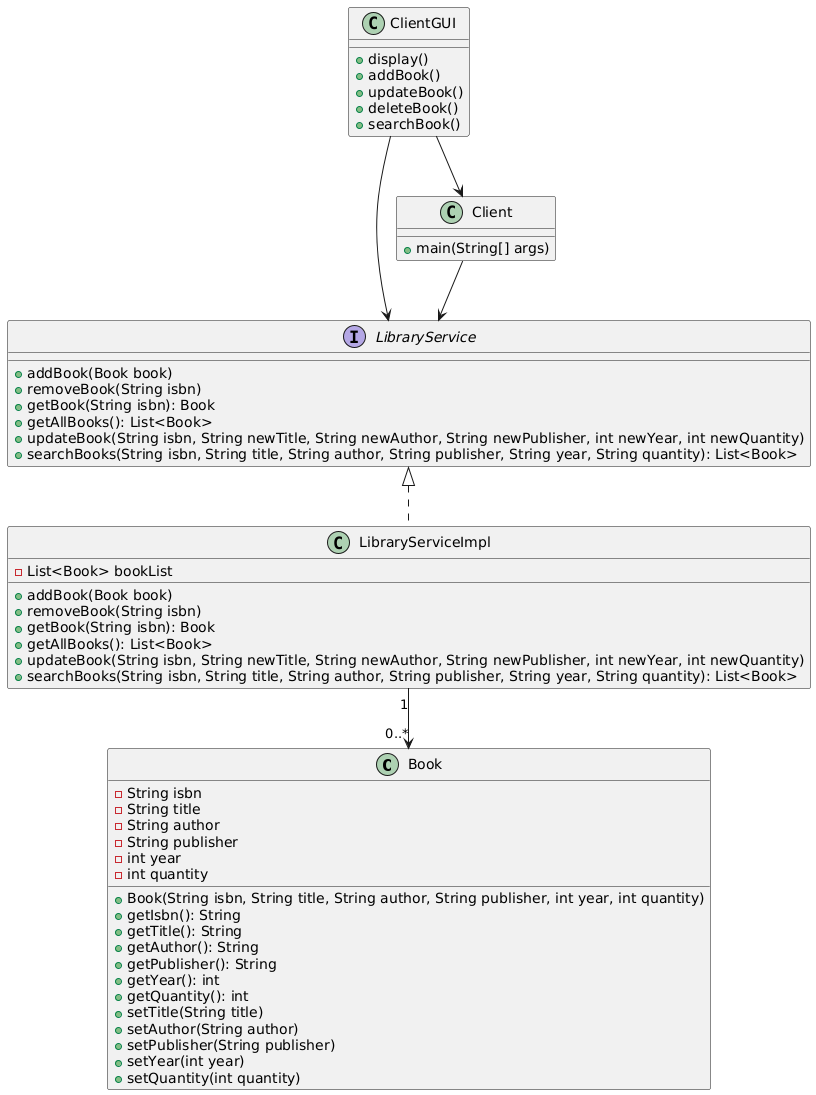
**Luồng sự kiện chính:**

1. Người dùng chọn chức năng **“Xem danh sách Sách”**.
2. Client gửi yêu cầu lấy danh sách sách từ Server thông qua **Java RMI**.
3. Server truy vấn cơ sở dữ liệu và gửi kết quả về cho Client.
4. Client hiển thị danh sách sách trên giao diện, bao gồm ISBN, Title, Author, Publisher, Year, Quantity.
5. Người dùng có thể chọn sách để xem chi tiết, chỉnh sửa hoặc xóa sách.

*f. Đồng bộ dữ liệu*

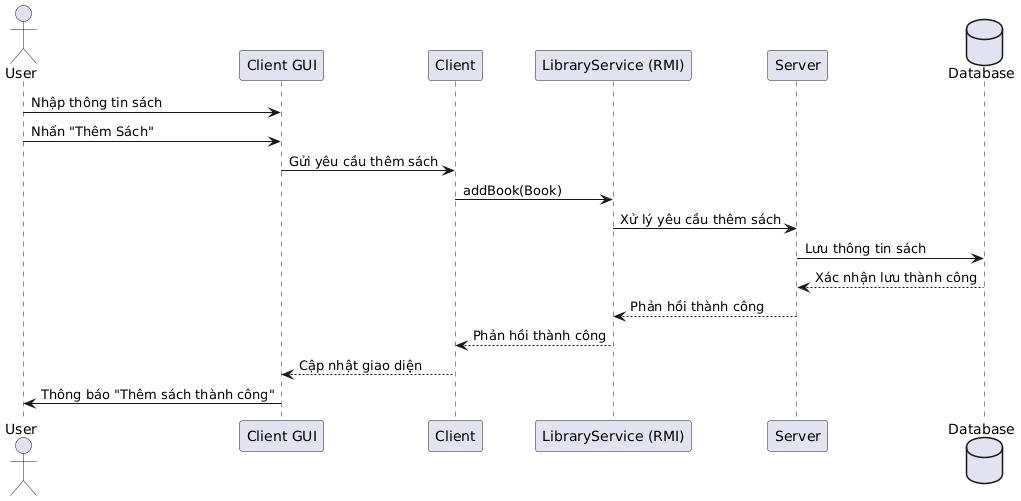
* Admin: Gửi yêu cầu đồng bộ để cập nhật thông tin mới nhất trên hệ thống.
* Hệ thống: Thực hiện đồng bộ hóa dữ liệu giữa các thành phần và xác nhận hoàn tất.

## Mô hình lớp và đối tượng

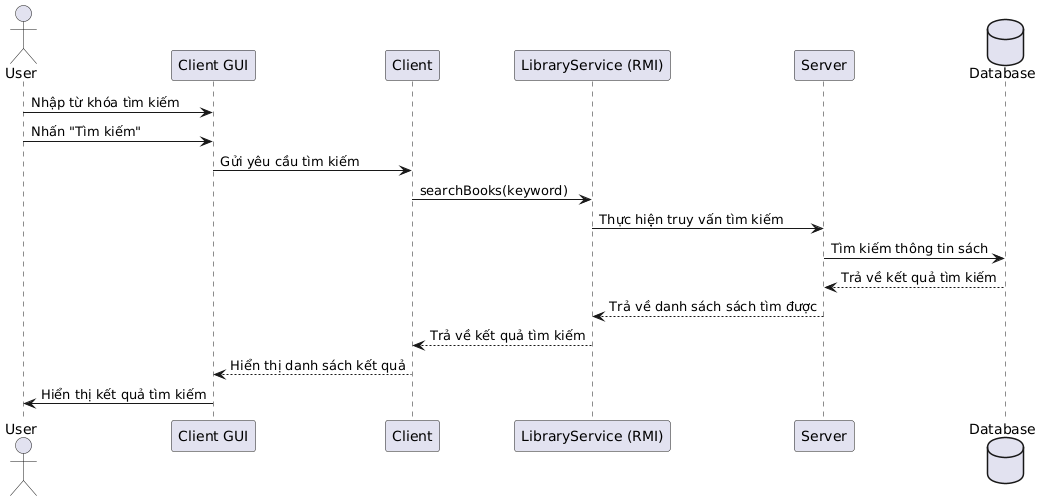


Hình 2: Biểu đồ mô hình lớp và đối tượng

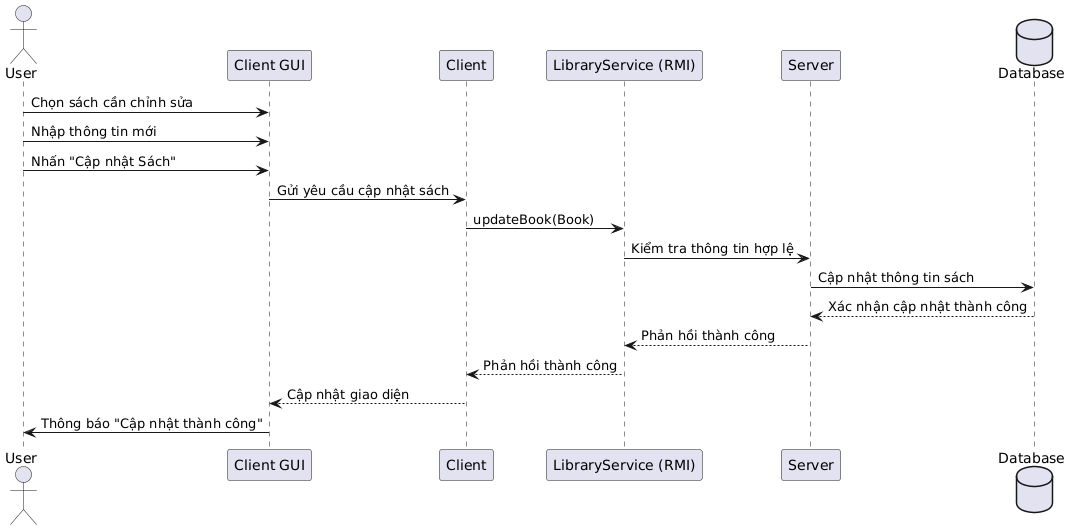
## Biểu đồ tuần tự



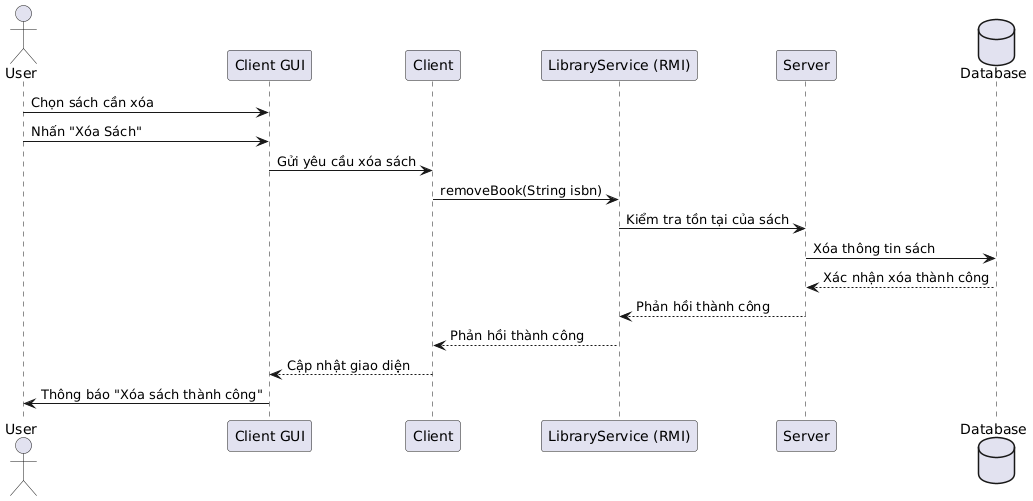
Hình 3: Biểu đồ tuần tự thêm sách



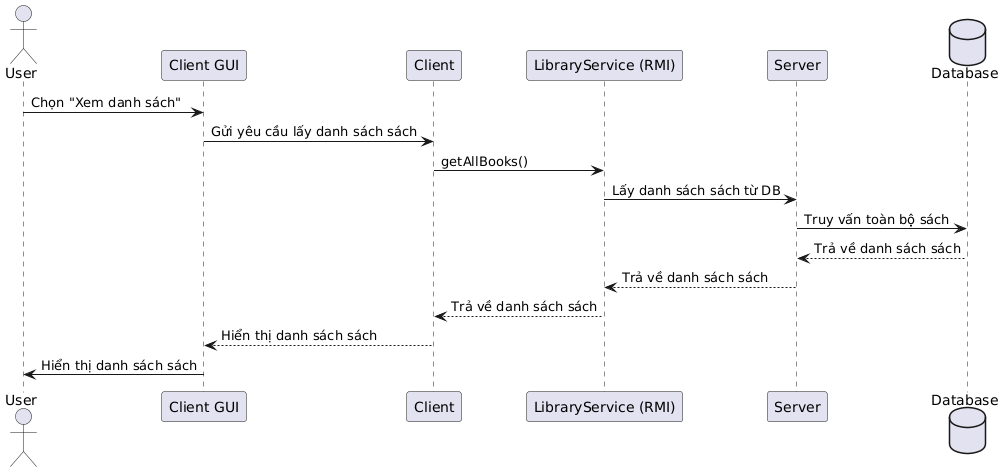
Hình 4: Biểu đồ tuần tự tìm kiếm sách



Hình 5: Biểu đồ tuần tự tìm kiếm sách



Hình 6: Biểu đồ tuần tự xóa sách



Hình 7: Biểu đồ tuần tự xem danh sách của Sách

## Thiết kế kiến trúc hệ thống



A diagram of a server

AI-generated content may be incorrect.

Hình 8: Kiến trúc hệ thống RMI



*a. Tầng Client (Giao diện người dùng)*

* Nhập yêu cầu từ người dùng như CRUD (Create, Read, Update, Delete) dữ liệu.
* Gửi yêu cầu đến tầng Server và nhận phản hồi dữ liệu.

*b. Tầng Server*

* Xử lý logic nghiệp vụ của hệ thống:
* Nhận yêu cầu từ tầng Client và chuyển tiếp đến cơ sở dữ liệu.
* Thực hiện các thao tác như xác thực người dùng, kiểm tra logic và xử lý thông tin.
* Trả về kết quả cho tầng Client.
* Các chức năng chính:
* CRUD: Quản lý thêm, sửa, xóa thông tin trong hệ thống.
* Request handling: Xử lý các yêu cầu từ phía Client.

*c. Tầng Database*

* Lưu trữ thông tin hệ thống bao gồm:
* Thông tin cá nhân, lịch sử công việc, sự kiện liên quan.
* Dữ liệu quản trị từ phía Admin.
* Đảm bảo hiệu năng cao trong truy xuất và bảo mật dữ liệu.

*d. Quy trình hoạt động*

* Người dùng gửi yêu cầu: Thực hiện thao tác qua giao diện (Client).
* Xử lý yêu cầu: Server nhận và xử lý logic nghiệp vụ tương ứng với yêu cầu CRUD.
* Truy vấn dữ liệu: Server gửi yêu cầu truy vấn hoặc cập nhật đến Database.
* Phản hồi kết quả: Dữ liệu từ Database được trả về Server, sau đó hiển thị cho Client.

# CHƯƠNG III. CÀI ĐẶT VÀ TRIỂN KHAI

## Lựa chọn công nghệ

Để phát triển hệ thống quản lý sách dựa trên kiến trúc phân tán sử dụng Java RMI, nhóm đã cân nhắc và lựa chọn các công nghệ phù hợp nhằm đảm bảo tính ổn định, hiệu suất cao và khả năng mở rộng. Mỗi công nghệ được lựa chọn đều có vai trò và ưu điểm riêng, góp phần giúp hệ thống hoạt động hiệu quả. Dưới đây là các công nghệ được sử dụng trong quá trình xây dựng hệ thống:

*a. Ngôn ngữ lập trình*

* Java: Java được chọn làm ngôn ngữ lập trình chính nhờ vào tính năng hướng đối tượng mạnh mẽ và khả năng hỗ trợ đa nền tảng. Với thư viện phong phú và sự tích hợp tốt với các công nghệ mạng, đặc biệt là **Java RMI** **(Remote Method Invocation)**, Java giúp việc triển khai các phương thức từ xa và xử lý giao tiếp giữa Client và Server trở nên đơn giản, đáng tin cậy.

Ngoài ra, Java còn hỗ trợ các thư viện giao diện đồ họa như Swing, giúp xây dựng ứng dụng trực quan và thân thiện với người dùng.

*b. Công cụ phát triển*

* JDK (Java Development Kit): Để tận dụng các tính năng mới nhất của Java RMI, nhóm sử dụng JDK từ phiên bản Java 8 trở lên. JDK cung cấp môi trường biên dịch, thực thi và gỡ lỗi, là công cụ không thể thiếu khi phát triển ứng dụng Java.

*c. Thiết kế giao diện người dùng*

* Java Swing: Giao diện người dùng của ứng dụng Client được thiết kế bằng Java Swing, cho phép xây dựng các thành phần giao diện như nút bấm, hộp thoại, bảng dữ liệu và thanh công cụ. Java Swing hỗ trợ tùy biến giao diện linh hoạt, giúp người dùng tương tác dễ dàng với hệ thống.

*d. Công nghệ phân tán*

* Java RMI (Remote Method Invocation): Java RMI là công nghệ cốt lõi trong kiến trúc phân tán của hệ thống. RMI cho phép các đối tượng Java trên Client gọi các phương thức từ xa được triển khai trên Server, giúp đồng bộ hóa dữ liệu và chia sẻ tài nguyên giữa các thành phần một cách dễ dàng.

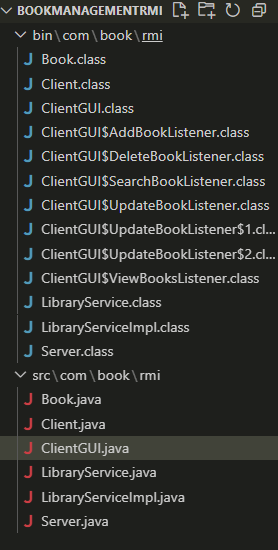
Ưu điểm của Java RMI là khả năng làm việc trực tiếp với các đối tượng Java mà không cần chuyển đổi dữ liệu sang định dạng khác, giúp giảm thiểu chi phí và độ phức tạp trong quá trình xử lý.

*e. Công cụ hỗ trợ phát triển*

* Git: Nhóm sử dụng Git để quản lý mã nguồn. Công cụ này giúp theo dõi sự thay đổi của mã nguồn, hỗ trợ làm việc nhóm hiệu quả và đảm bảo mã nguồn luôn được lưu trữ an toàn.

## Cấu trúc mã nguồn

1. *Thư mục chính của dự án*



Hình 9: Các thư mục mã nguồn

1. *Các thành phần chính trong mã nguồn*

1. Thư mục bin/

Thư mục bin chứa các file đã được biên dịch từ mã nguồn trong thư mục src/. Đây là nơi tập hợp các lớp đã sẵn sàng để thực thi. Các thành phần chính bao gồm:

* **Client.class:**
  + Lớp thực thi chính của client, chịu trách nhiệm khởi tạo kết nối với server thông qua **Java RMI** và giao tiếp với giao diện người dùng.
  + Xử lý yêu cầu từ người dùng như thêm, xóa, tìm kiếm và cập nhật thông tin sách.
* **ClientGUI.class:**
  + Lớp quản lý **giao diện người dùng (Graphical User Interface - GUI)**, cho phép người dùng thao tác với dữ liệu sách.
  + Cung cấp các chức năng như thêm sách, xóa sách, tìm kiếm sách, cập nhật thông tin và xem danh sách sách.
  + Bao gồm các lớp con lắng nghe và xử lý sự kiện từ giao diện:
    - **ClientGUI$AddBookListener.class:** Lớp con xử lý sự kiện thêm sách.
    - **ClientGUI$DeleteBookListener.class:** Lớp con xử lý sự kiện xóa sách.
    - **ClientGUI$SearchBookListener.class:** Lớp con xử lý sự kiện tìm kiếm sách.
    - **ClientGUI$UpdateBookListener.class:** Lớp con xử lý sự kiện cập nhật thông tin sách.
    - **ClientGUI$ViewBooksListener.class:** Lớp con hiển thị danh sách sách từ cơ sở dữ liệu.
* **Server.class:**
  + Lớp thực thi của server, chịu trách nhiệm khởi động dịch vụ **Java RMI**.
  + Đăng ký đối tượng LibraryServiceImpl với **RMI Registry** để cung cấp dịch vụ từ xa cho client.
* **Book.class:**
  + Lớp đại diện cho thực thể sách, chứa các thuộc tính như **ISBN, Title, Author, Publisher, Year, Quantity.**
  + Cung cấp các phương thức getter và setter để thao tác với dữ liệu sách.
* **LibraryService.class:**
  + Giao diện định nghĩa các phương thức quản lý sách như **addBook(), removeBook(), getBook(), getAllBooks(), updateBook(), searchBooks()**.
  + Được triển khai như một interface RMI để client gọi các phương thức từ xa trên server.
* **LibraryServiceImpl.class:**
  + Lớp triển khai LibraryService, thực hiện các thao tác xử lý dữ liệu sách như thêm, xóa, tìm kiếm, cập nhật thông tin sách.
  + Lưu trữ danh sách sách và trả kết quả về cho client thông qua **Java RMI**.

2. Thư mục src/

Thư mục src là nơi chứa mã nguồn chính của ứng dụng, giúp phát triển và quản lý từng thành phần của hệ thống. Các tệp mã nguồn quan trọng nằm trong gói com.book.rmi:

* **Client.java:**
* Mã nguồn khởi động client và thiết lập kết nối tới server thông qua **Java RMI**.
* Khởi tạo giao diện người dùng bằng cách gọi ClientGUI và tương tác với LibraryService để xử lý các yêu cầu từ người dùng.
* **ClientGUI.java:**
* Mã nguồn xây dựng **giao diện người dùng (GUI)** cho ứng dụng bằng **Java Swing**.
* Cung cấp các thành phần giao diện như **JFrame, JTextField, JButton, JTable** để thực hiện các thao tác quản lý sách.
* Tích hợp các sự kiện như **AddBookListener, DeleteBookListener, SearchBookListener, UpdateBookListener, ViewBooksListener** để xử lý các yêu cầu của người dùng.
* **Server.java:**
* Mã nguồn khởi động server và dịch vụ **Java RMI**.
* Đăng ký đối tượng LibraryServiceImpl với **RMI Registry** để cung cấp dịch vụ từ xa cho client.
* Chịu trách nhiệm quản lý kết nối từ nhiều client khác nhau và xử lý yêu cầu truy cập dữ liệu sách.
* **Book.java:**
* Lớp mô tả **thực thể sách (Entity)**, chứa các thuộc tính như:
  + **ISBN:** Mã số sách quốc tế duy nhất.
  + **Title, Author, Publisher, Year, Quantity.**
* Cung cấp các phương thức getter và setter để thao tác với dữ liệu sách.
* **LibraryService.java:**
* Giao diện định nghĩa các phương thức từ xa (Remote Interface) như:
  + **addBook(Book book):** Thêm sách mới.
  + **removeBook(String isbn):** Xóa sách theo ISBN.
  + **getBook(String isbn):** Lấy thông tin sách theo ISBN.
  + **getAllBooks():** Lấy danh sách tất cả sách trong thư viện.
  + **updateBook(String isbn, ...):** Cập nhật thông tin sách.
  + **searchBooks(...):** Tìm kiếm sách theo nhiều tiêu chí khác nhau.
* Giao diện này được triển khai trên Server và gọi từ xa bởi Client thông qua **Java RMI**.
* **LibraryServiceImpl.java:**
* Lớp triển khai LibraryService, xử lý các thao tác dữ liệu như thêm, xóa, tìm kiếm, cập nhật thông tin sách.
* Sử dụng danh sách (List<Book>) để lưu trữ dữ liệu sách và thực hiện các thao tác truy vấn.
* Đảm bảo đồng bộ hóa dữ liệu và trả kết quả chính xác cho Client thông qua **Java RMI**.

3. Các tệp khác

* .gitattributes: File quản lý thuộc tính của Git, đảm bảo mã nguồn được kiểm soát và đồng bộ chính xác khi làm việc nhóm.
* README.md: Tài liệu hướng dẫn chi tiết về cách cài đặt, sử dụng chương trình và mô tả các chức năng chính của ứng dụng.



## Triển khai ứng dụng

**RMI Interface (LibraryService.java):**

package com.book.rmi;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.List;

public interface LibraryService extends Remote {

void addBook(Book book) throws RemoteException;

void removeBook(String isbn) throws RemoteException;

Book getBook(String isbn) throws RemoteException;

List<Book> getAllBooks() throws RemoteException;

void updateBook(String isbn, String newTitle, String newAuthor, String newPublisher, int newYear, int newQuantity) throws RemoteException;

List<Book> searchBooks(String isbn, String title, String author, String publisher, String year, String quantity) throws RemoteException;

}

**Server Implementation (LibraryServiceImpl.java):**

package com.book.rmi;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

import java.util.ArrayList;

import java.util.List;

public class LibraryServiceImpl extends UnicastRemoteObject implements LibraryService {

private List<Book> books;

protected LibraryServiceImpl() throws RemoteException {

books = new ArrayList<>();

}

@Override

public void addBook(Book book) throws RemoteException {

books.add(book);

}

@Override

public void removeBook(String isbn) throws RemoteException {

books.removeIf(book -> book.getIsbn().equals(isbn));

}

@Override

public Book getBook(String isbn) throws RemoteException {

return books.stream()

.filter(book -> book.getIsbn().equals(isbn))

.findFirst()

.orElse(null);

}

@Override

public List<Book> getAllBooks() throws RemoteException {

return books;

}

@Override

public void updateBook(String isbn, String newTitle, String newAuthor, String newPublisher, int newYear, int newQuantity) throws RemoteException {

for (Book book : books) {

if (book.getIsbn().equals(isbn)) {

book.setTitle(newTitle);

book.setAuthor(newAuthor);

book.setPublisher(newPublisher);

book.setYear(newYear);

book.setQuantity(newQuantity);

break;

}

}

}

@Override

public List<Book> searchBooks(String isbn, String title, String author, String publisher, String year, String quantity) throws RemoteException {

List<Book> searchResults = new ArrayList<>();

for (Book book : books) {

boolean match = true;

if (!isbn.isEmpty() && !book.getIsbn().contains(isbn)) match = false;

if (!title.isEmpty() && !book.getTitle().contains(title)) match = false;

if (!author.isEmpty() && !book.getAuthor().contains(author)) match = false;

if (!publisher.isEmpty() && !book.getPublisher().contains(publisher)) match = false;

if (!year.isEmpty() && !String.valueOf(book.getYear()).contains(year)) match = false;

if (!quantity.isEmpty() && !String.valueOf(book.getQuantity()).contains(quantity)) match = false;

if (match) {

searchResults.add(book);

}

}

return searchResults;

}

}

**Client Implementation (Client.java):**

package com.book.rmi;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Client {

public static void main(String[] args) {

try {

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

LibraryService libraryService = (LibraryService) registry.lookup("LibraryService");

libraryService.addBook(new Book("001", "The Witcher", " Andrzej Sapkowski", "A Company", 2007, 10));

libraryService.addBook(new Book("002", "300 bài code thiếu nhi", "Nguyen Van A", "B Company", 2010, 15));

Book book = libraryService.getBook("001");

System.out.println("Book: " + book.getTitle() + ", Author: " + book.getAuthor() +

", Publisher: " + book.getPublisher() + ", Year: " + book.getYear() +

", Quantity: " + book.getQuantity());

libraryService.getAllBooks().forEach(b ->

System.out.println("ISBN: " + b.getIsbn() + ", Title: " + b.getTitle() + ", Author: " + b.getAuthor() +

", Publisher: " + b.getPublisher() + ", Year: " + b.getYear() +

", Quantity: " + b.getQuantity())

);

} catch (Exception e) {

e.printStackTrace();

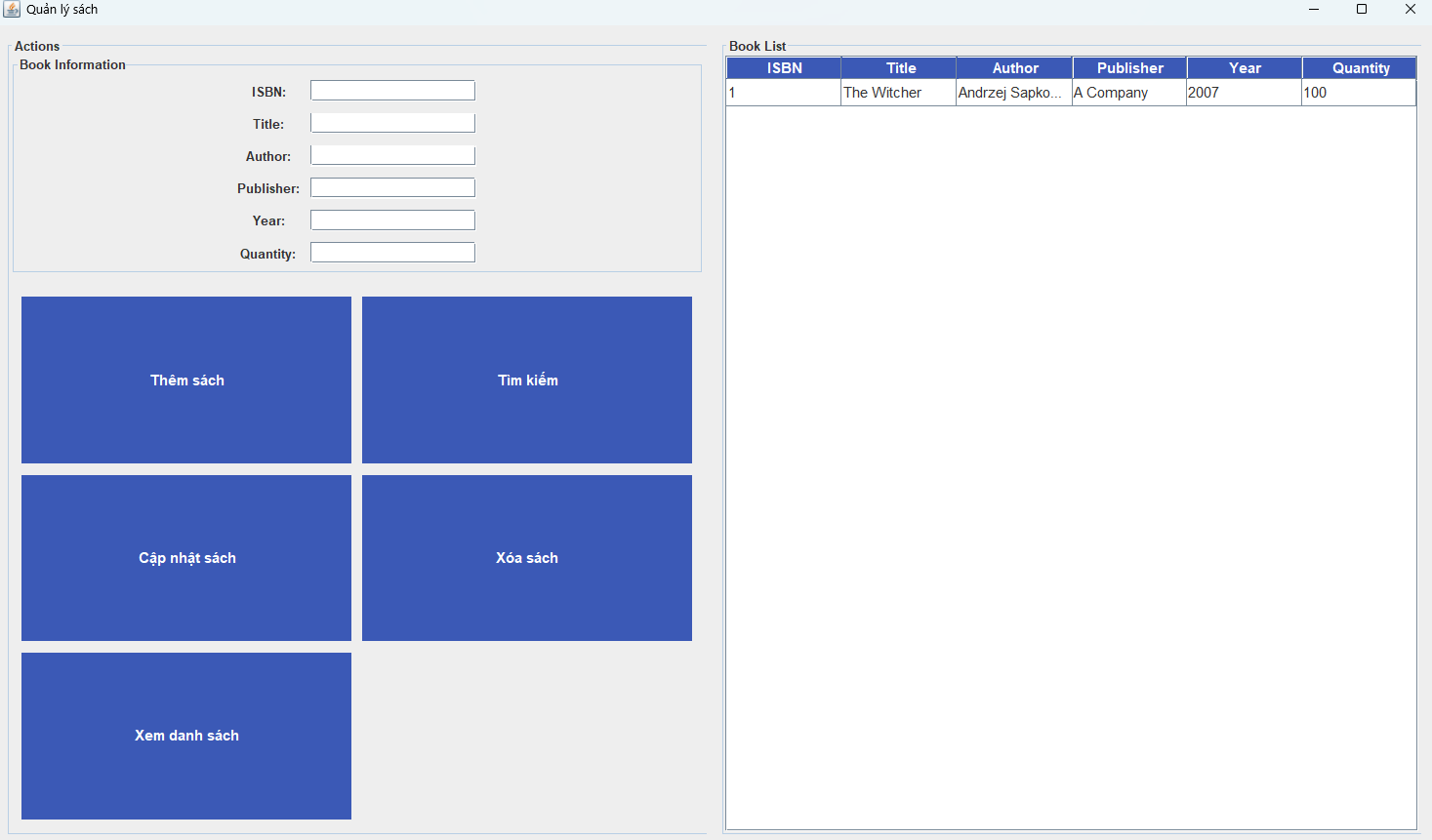
}

}

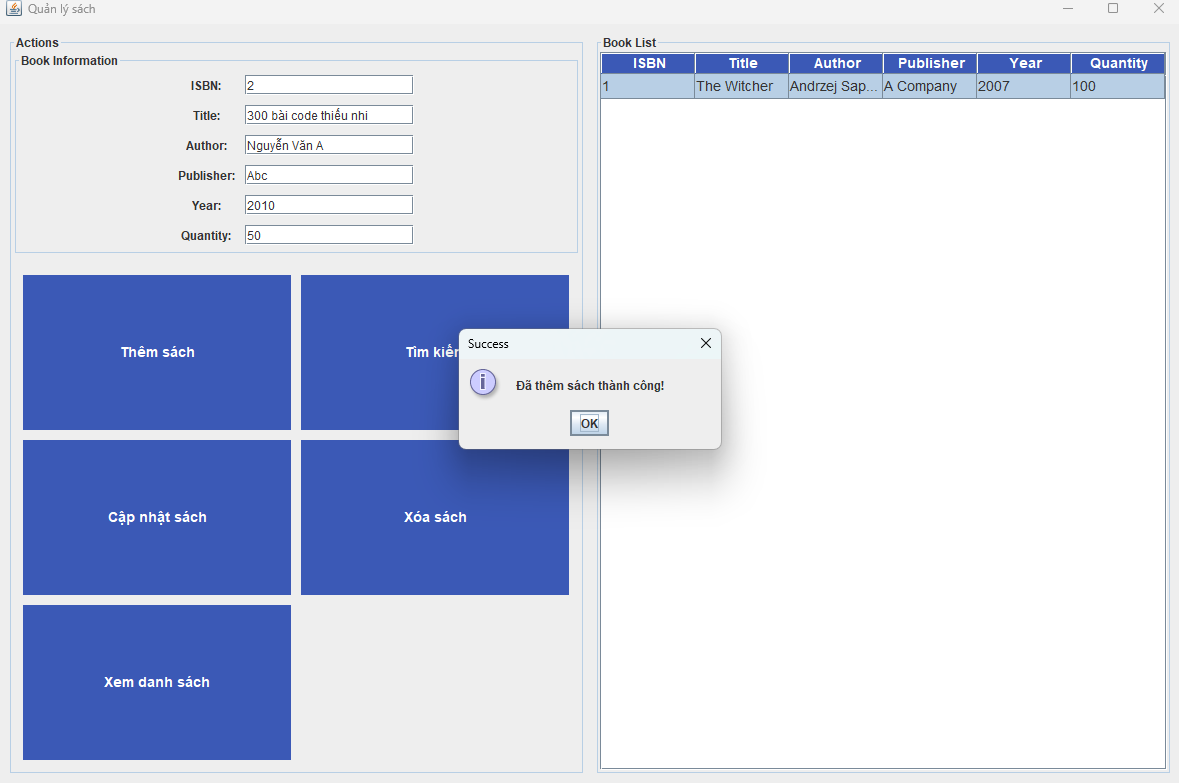
}

* Update code: *javac -d bin src/com/book/rmi/\*.java*
* Chạy server: *java -cp bin com.book.rmi.Server*
* Chạy client: *java -cp bin com.book.rmi.ClientGUI*
* Link source code: https://github.com/Giang1311/BookManagement

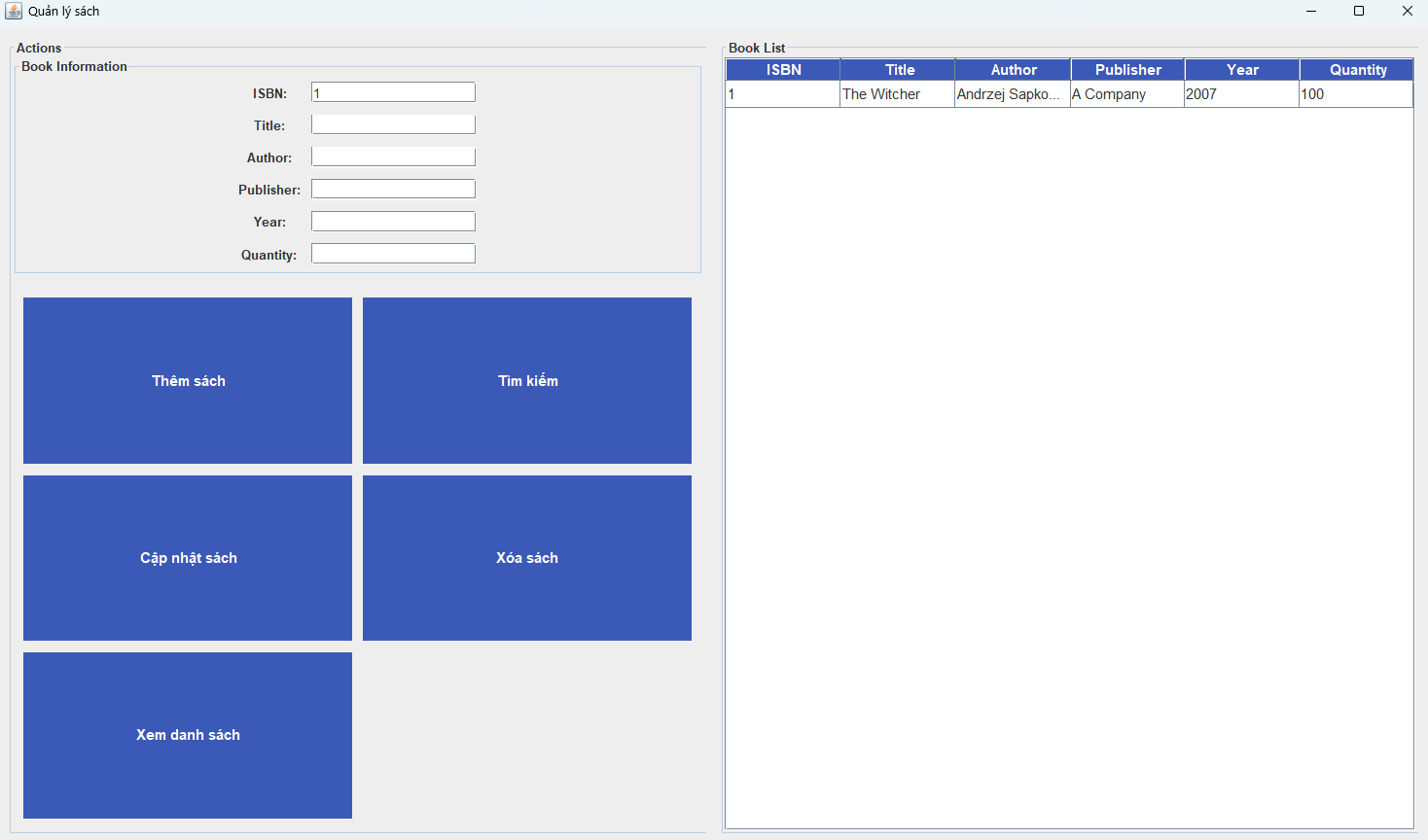
*Giao diện:*



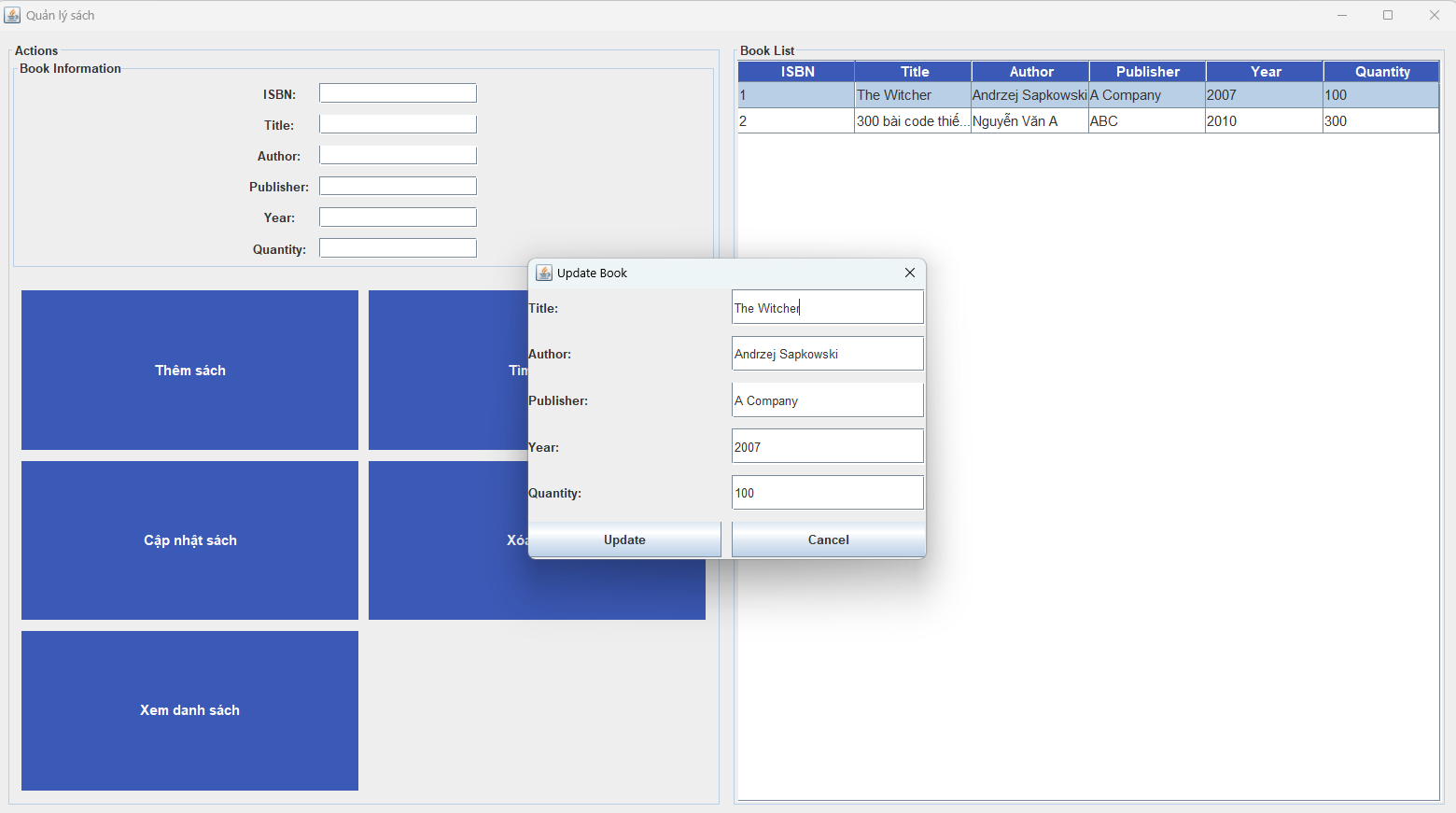
Hình 10: Giao diện chính



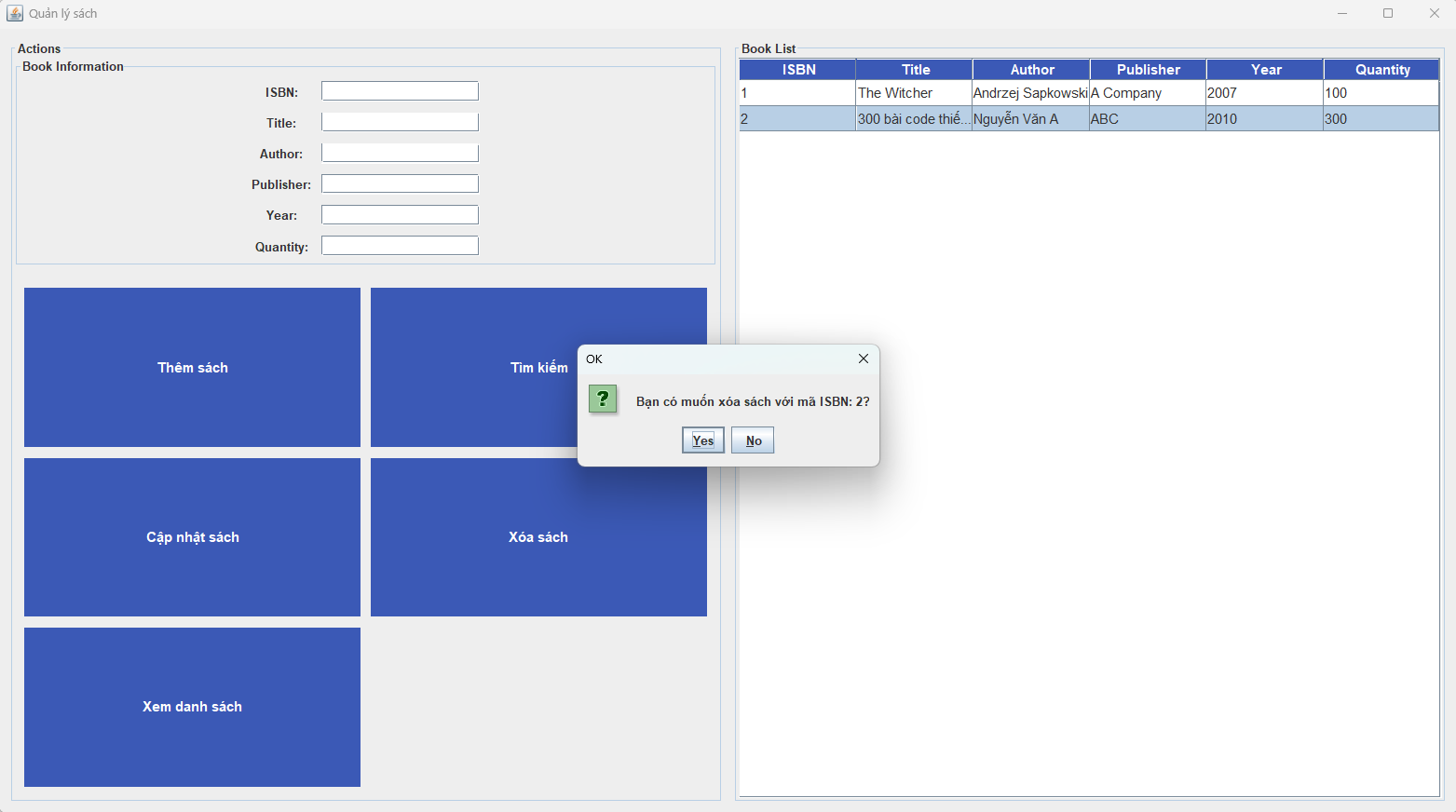
Hình 11: Giao diện thêm sách

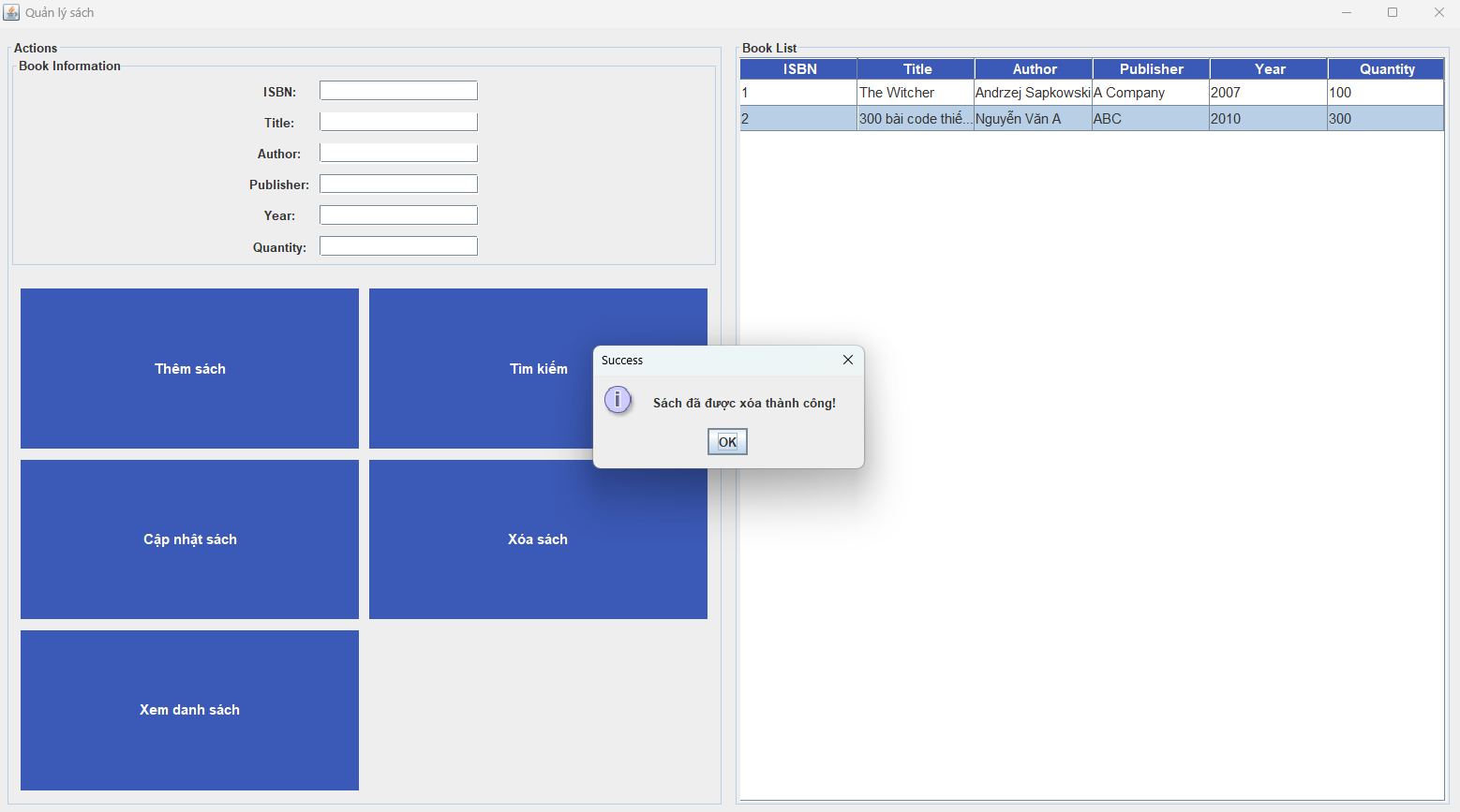


Hình 12: Giao diện tìm sách



Hình 13: Giao diện cập nhật sách





Hình 14, 15: Giao diện xóa sách

# KẾT LUẬN

Sau quá trình nghiên cứu và triển khai, đề tài **“Xây dựng ứng dụng quản lý sách cho đại học sử dụng Java RMI”** đã hoàn thành thành công các chức năng cốt lõi như thêm, xóa, tìm kiếm và cập nhật thông tin sách, giúp quá trình quản lý sách trở nên đơn giản và hiệu quả hơn. Hệ thống cũng đảm bảo kết nối giữa client và server thông qua giao thức RMI, cung cấp khả năng xử lý dữ liệu nhanh chóng, chính xác và an toàn.

Với việc áp dụng Java RMI, nhóm chúng em đã bước đầu hiện thực hóa một hệ thống quản lý tập trung, hỗ trợ người dùng thực hiện các thao tác quản lý thông tin sách từ xa một cách hiệu quả. Kết quả đạt được đã giúp nhóm hiểu sâu hơn về kỹ thuật lập trình phân tán và học được thêm nhiều điều về việc phát triển các kỹ năng quan trọng trong việc xây dựng ứng dụng thực tế.

Tuy nhiên, ứng dụng vẫn còn một số hạn chế như chưa có cơ chế **xác thực và phân quyền người dùng** hoặc **quản lý mượn/trả sách**. Trong tương lai, nhóm dự kiến sẽ mở rộng ứng dụng với các tính năng nâng cao như **xác thực người dùng, phân quyền truy cập, quản lý mượn trả sách, tính phí trễ hạn**, và **tích hợp cơ sở dữ liệu phân tán** để cải thiện hiệu suất và khả năng mở rộng của hệ thống. Đồng thời, nhóm cũng sẽ **tối ưu hóa giao diện người dùng (GUI)** để mang lại trải nghiệm thân thiện và hiện đại hơn. Những cải tiến này sẽ giúp ứng dụng hoàn thiện hơn, đáp ứng tốt hơn nhu cầu quản lý sách trong các thư viện đại học.

Cuối cùng, nhóm chúng em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn TS. Trần Đăng Hoan, người đã tận tình chỉ dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện đề tài. Những đóng góp và ý kiến quý báu của cô đã giúp chúng em hoàn thành tốt dự án này, đồng thời tích lũy thêm nhiều kiến thức và kỹ năng thực tiễn trong lĩnh vực phát triển phần mềm.

# TÀI LIỆU THAM KHẢO

1. Khoa Công nghệ Thông tin. (2025). *Slide bài giảng môn học Ứng dụng phân tán* Đại học Phenika.
2. Maarten van Steen. Andrew S. Tanenbaum. (2017). *Distributed System* (4th ed.).