HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

------oOo------

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

Đề tài:

Phát triển ứng dụng Web quản lý sách nói và sách điện tử

Giảng viên hướng dẫn	: TS. Dương Trần Đức
Sinh viên	: Hoàng Anh Đức
Mã sinh viên	: B19DCCN192
Lớp	: D19CNPM04
Khóa	: 2019 – 2024
Hệ đào tạo	: Đại học chính quy

Hà Nội - 2023

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(CỦA GIẢNG VIÊN PHẢN BIỆN)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:.....(bằng chữ.....)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

Hà Nội, ngày....tháng....năm 2023

Giảng viên phản biện

(Ký và ghi rõ họ tên)

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(CỦA GIẢNG VIÊN HƯỚNG DẪN)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:.....(bằng chữ.....)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

Hà Nội, ngày....tháng....năm 2023

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên em xin gửi lời cảm ơn chân thành nhất đến toàn thể các thầy, các cô trong khoa **Công nghệ thông tin 1** Học viện **Công nghệ Bưu chính Viễn thông**, những người đã dạy dỗ, truyền tải em rất nhiều kiến thức hữu ích, đáng giá trong suốt mấy năm học tập dưới mái trường Học viện, góp phần lớn vào thành quả của em ngày hôm nay.

Tiếp theo, em xin gửi lời cảm ơn chân thành, kính trọng trực tiếp tới thầy **TS. Dương Trần Đức** đã định hướng, chỉ bảo, giúp đỡ em tận tình trong suốt quá trình làm đồ án từ lúc chọn đề tài, quá trình nghiên cứu, thực hiện và hoàn thiện đồ án này.

Ngoài ra, em cũng xin cảm ơn gia đình,, bạn bè, anh chị đồng nghiệp, những người luôn đông viên, giúp đỡ những lúc khó khăn trong suốt quá trình thực hiện đồ án, tạo điều kiện tốt nhất cho em hoàn thành môn học cuối cùng của mình.

Dù đã cố gắng hết sức nhưng rất khó tránh được những sai sót, vì vậy em rất mong nhận được sự thông cảm, góp ý cũng như những nhận xét quý báu của thầy cô để đồ án của em có thể hoàn thiện chỉnh chu hơn.

Với tất cả sự kính trọng, em xin chân thành cảm ơn!

Hà Nội, tháng 12 năm 2023

Sinh viên

Hoàng Anh Đức

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

PHẦN MỞ ĐẦU

I. GIỚI THIỆU

Từ xưa đến nay, sách luôn là nguồn tri thức quý giá của nhân loại. Nó chứa đựng những tinh hoa, kiến thức được đúc kết. Sách được những tác giả soạn ra với mong muốn truyền đạt kiến thức cho đời sau, để những thành tựu trí tuệ sẽ mãi được lưu lại và phát triển. Với người đọc, đọc sách còn là hoạt động giải trí, hành trình khám phá, học hỏi và phát triển bản thân.

Trong thời đại công nghệ ngày càng phát triển, việc tiếp cận sách và tiếp nhận thông tin từ sách ngày càng trở nên phong phú, đa dạng, đó chính là có sự giúp sức của sách nói và sách điện tử. Nó đang trở thành xu hướng không thể phủ nhận. Sự thuận tiện và linh hoạt của việc nghe sách khi đang di chuyển hoặc đọc sách mà không cần mang theo sách giấy, đôi lúc có phần vướng víu, đang góp phần thổi làn gió mới trong việc tiếp nhận kiến thức từ sách trong thế giới hiện đại. Trong nhịp sống thường ngày, có những người muốn nghe sách nói trong khi đang làm việc nhà, hay khi đang nhâm nhi ly cà phê cùng với bữa sáng. Với những cô cậu sinh viên, họ muốn nghe sách nói hoặc đọc sách điện tử khi đang đi xe bus đến trường. Vừa nghe sách nói, vừa có thể nhắm mắt thư giãn sẽ là cảm giác khá dễ chịu sau một ngày làm việc vất vả. Nhu cầu sử dụng các loại hình khác nhau của sách trên các ứng dụng điện tử là không hề nhỏ. Đó là thị trường lớn cho các công ty đầu tư nguồn lực, vừa mang lại giá trị về kinh tế, vừa mang lại giá trị to lớn cho xã hội về mặt tinh thần. Và với một xã hội ngày càng phát triển, nhu cầu đọc sách sẽ ngày càng lớn, càng có kiến thức, con người ta càng tìm tới sách nhiều hơn, bởi vì sách là tri thức của nhân loại, sẽ không thể nào bị ngó lơ và lỗi thời được.

Đã có nhiều ứng dụng cung cấp dịch vụ nghe sách nói và đọc sách điện tử trên điện thoại di động, trên ứng dụng web. Tuy nhiên, phần lớn những ứng dụng về sách sẽ chỉ tập trung vào sách nói hoặc sách điện tử, không nhiều ứng dụng mang đến cho cộng đồng yêu sách cả 2 loại hình trên. Nổi bật hơn cả là Audible, khi mang đến cho người đọc sự tiện dụng với 2 loại sách và cung cấp chức năng có thể đồng bộ, thay đổi linh hoạt từ nghe sách nói sang đọc sách điện tử, và ngược lại.

Với thị trường Việt Nam, Waka và Fonos là 2 ứng dụng được biết đến và chiếm thị phần không nhỏ, góp phần mang đến cho độc giả Việt Nam nói riêng sự tiện dụng trong nhu cầu sử dụng sách trên thiết bị điện tử. Đó đều là những ứng dụng với một thư viện sách nói, sách điện tử đa dạng, bao gồm cả sách tiếng Việt và tiếng nước ngoài, giao diện đẹp mắt, dễ sử dụng và hỗ trợ nhiều tính năng hữu ích như tạo danh sách phát, quản lý thư viện và ghi chú, đánh dấu các mục yêu thích.

Cùng với nhu cầu sử dụng sách nói chung, sử dụng sách điện tử và và sách nói nói riêng, đây vẫn là thị trường tiềm năng cho các công ty lớn và công ty start-up thử sức và dành lấy thị phần. Nhận thấy điều đó, em mong muốn thử sức xây dựng một ứng dụng về sách với sologan "hệ sinh thái cho sách và cộng đồng yêu mến sách". Ứng dụng sẽ mang lại nhiều tính năng tiện ích cho người dùng, từ việc gợi ý sách họ có thể thích, đến khâu đặt mua, nhận sách về tay, và trải nghiệm khi nghe, đọc sách, giúp họ quản lý tủ sách của mình. Ứng dụng mà em hướng tới xây dựng sẽ bao gồm các chức năng như:

- Tìm kiếm sách trong hệ thống, theo những chủ đề hay những loại sách được mọi người quan tâm.
- Gợi ý những đầu sách mà người dùng có thể yêu thích, thông qua lịch sử tìm kiếm, lịch sử đọc, tủ sách của người dụng, hay là dựa vào tâm lý chung của những người cùng tuổi, cùng nghề nghiệp với nhau.
- Hỗ trợ đặt và mua sách, trong đó gồm cả sách giấy, sách nói và sách điện tử, theo nhiều hình thức thanh toán như thanh toán tiền mặt hay qua ngân hàng hoặc các ví điện tử.
- Giúp người dùng đánh giá về một đầu sách đã mua, để giúp mọi người biết được chất lượng từng đầu sách, hữu ích với những người dùng khác khi muốn biết thông tin về một đầu sách mới. Chức năng này cũng giúp các nhà phát triển ứng dụng, các nhà xuất bản hay các tác giả hiểu hơn về người dùng để ngày một nâng cao chất lượng đầu sách được đến với tay bạn đọc.
- Cho phép người dùng đăng tải sách nói với giọng đọc của bản thân lên ứng dụng, chia sẻ nó với mọi người, đáp ứng nhu cầu chia sẻ sách chất lượng, xây dựng một cộng đồng đọc sách thật văn minh.
- Đánh dấu, lưu trữ những câu văn hay trong sách, có một nơi tập trung giúp người dùng có thể xem lại những phần mà họ đánh dấu, giúp họ gợi nhớ những kiến thức hay và có thể chia sẻ tới cộng đồng bạn đọc.
- Người dùng có thể đăng ký nhận thông báo khi có một đầu sách mới được xuất bản từ nhà xuất bản hay từ tác giả mà họ quan tâm.
- Chức năng đồng bộ giữa sách nói và sách điện tử, giúp người dùng có thể chuyển tử nghe sách nói sang đọc sách điện tử tại phần họ đang nghe, hoặc ngược lại, chuyển từ đọc sách điện tử sang nghe sách điện tử tại phần mà họ đang đọc.
- Giúp các đơn vị phân phối sách quản lý các đầu sách, các thông tin về sách một các tiện lợi.

Đồ án "**Phát triển ứng dụng Web quản lý sách nói và sách điện tử**" của em ra đời giúp giải quyết một phần bài toán xây dựng ứng dụng tiện lợi đó. Sản phẩm này sẽ giúp đơn vị phân phối dễ dàng quản lý các đầu sách, các thông tin về sách, thông tin về nhà xuất bản, tác giả và mang tới cho người dùng trải nghiệm sách theo cách linh hoạt và tiện lợi nhất với chức năng đồng bộ sách nói và sách điện tử.

II. MỤC TIÊU ĐỀ TÀI

Xây dựng ứng dụng web giúp người quản lý có thể dễ dàng quản lý thông tin sách, thông tin về nhà xuất bản, về tác giả, và xây dựng chức năng đồng bộ sách nói - sách điện tử, để góp phần mang đến sự thuận tiện cho những thính giả và độc giả yêu mến sách nói chung và sách nói, sách điện tử nói riêng.

III. PHẠM VI ĐỒ ÁN

Đồ án nằm trong hệ thống phát triển ứng dụng về sách nói và sách điện tử, bao gồm 3 thành phần chính: ứng dụng di động quản lý thư viện, hỗ trợ nghe sách nói và đọc sách điện tử; ứng dụng di động về cửa hàng thương mại điện tử cho các loại sách; ứng dụng web quản lý các đơn hàng và các đầu sách. Cụ thể các phần nghiên cứu như sau:

- Em, Hoàng Anh Đức, nghiên cứu về đề tài: Phát triển ứng dụng Web quản lý sách nói và sách điện tử, cùng chức năng đồng bộ dữ liệu giữa sách nói và sách điện tử.
- Bạn Lê Thanh Bình, nghiên cứu về đề tài: Phát triển ứng dụng nghe sách nói và đọc sách điện tử, kết hợp hệ thống khuyến nghị về sách trên thiết bị di động.
- Bạn Đinh Như Cương, nghiên cứu về đề tài: Phát triển ứng dụng thương mại điện tử cho các sản phẩm sách, bao gồm sách giấy, sách nói và sách điện tử.

Phạm vi của đồ án dựa trên những kiến thức đã học trong quá trình học tập cũng như tìm hiểu các thông tin về sách nói và sách điện tử. Tham khảo các tài liệu bên ngoài để đáp ứng tốt nhất mục tiêu đã đưa ra của đồ án.

IV. CẤU TRÚC CỦA ĐỒ ÁN

Nội dung của đồ án được xây dựng thành các chương như sau:

Chương 1. Cơ sở lý thuyết

Chương 2. Phân tích thiết kế hệ thống

Chương 3. Phát triển ứng dụng

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

Hệ thống quản lý sách được tạo nên bởi 4 thành phần chính:

- Front-end: giao diện sử dụng bởi người quản lý

  Công nghệ sử dụng: ReactJS/NextJS.

- Back-end: cung cấp các Api cho ứng dụng

  Công nghệ sử dụng: NestJS.

- Message Service: hệ thống quản lý thông tin sự kiện

  Công nghệ sử dụng: Kafka.

- Matching Service: trích xuất dữ liệu đồng bộ

  Công nghệ sử dụng: ngôn ngữ Python, ASR model.

Ngoài ra, hệ thống sử dụng Postgres làm cơ sở dữ liệu và Docker để ảo hóa một số thành phần hệ thống, giúp hệ thống dễ dàng triển khai và cài đặt ở môi trường mới.

1.1. ReactJS

1.1.1. Khái niệm

Theo "The Road to React: Your journey to master plain yet pragmatic React.js" của Robin Wieruch [1], ReactJS là một thư viện JavaScript mã nguồn mở được xây dựng bởi Facebook để xây dựng các giao diện người dùng tương tác. ReactJS sử dụng mô hình lập trình theo thành phần (component-based programming) để xây dựng các ứng dụng web dễ bảo trì và mở rộng.

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau.

React sử dụng khái niệm DOM ảo (Virtual DOM) để chọn lựa và render những phần tử của node dựa tên sự thay đổi trạng thái khiến cho ta chỉ cần thay đổi ít thành phần nhất có thể để giữ DOM update.

1.1.2. Đặc điểm

Reactjs cực kì hiệu quả: Reactjs tạo ra cho chính nó DOM ảo – nơi mà các component thực sự tồn tại trên đó. Điều này sẽ giúp cải thiện hiệu suất rất nhiều.

Reactjs giúp việc viết các đoạn code JS dễ dàng hơn: Nó dung cú pháp đặc biệt là JSX (Javascript mở rộng) cho phép ta trộn giữa code HTML và Javascript. Ta có thể them vào các đoạn HTML vào trong hàm render mà không cần phải nối chuỗi.

Reactjs là một thư viện component, nó có thể vừa render ở ngoài trình duyệt sử dụng DOM và cũng có thể render bằng các chuỗi HTML mà server trả về.

Làm việc với vấn đề test giao diện: Nó cực kì dễ để viết các test case giao diện vì virtual DOM được cài đặt hoàn toàn bằng JS.

Hiệu năng cao đối với các ứng dụng có dữ liệu thay đổi liên tục, dễ dàng cho bảo trì và sửa lỗi.

1.1.3. Hạn chế

Học tập đường cong: ReactJS có một đường cong học tập dốc, đặc biệt là đối với người mới bắt đầu phát triển web. Ngôn ngữ JSX và mô hình lập trình phản ứng có thể gây nhầm lẫn và khó hiểu đối với một số người.

Hiệu suất: Mặc dù ReactJS nói chung rất nhanh, nhưng nó có thể trở nên chậm chạp khi ứng dụng trở nên lớn hơn và phức tạp hơn. Điều này đặc biệt đúng đối với các ứng dụng có nhiều trạng thái hoặc được kết xuất lại thường xuyên.

Phụ thuộc vào thư viện bên thứ ba: ReactJS phụ thuộc vào một số lượng lớn các thư viện bên thứ ba để hoạt động. Điều này có thể khiến cho việc triển khai và quản lý ứng dụng trở nên phức tạp hơn và làm tăng khả năng xảy ra lỗi.

Thiếu kiểm tra loại: ReactJS là một ngôn ngữ động, vì vậy không có hệ thống kiểm tra loại nào đảm bảo rằng các giá trị có đúng loại hay không cho đến khi chúng được thực thi. Điều này có thể dẫn đến lỗi thời gian chạy khó gỡ lỗi và khó theo dõi.

Các vấn đề về SEO: ReactJS là một khung dựng hình phía trước, nghĩa là nó tạo ra nội dung HTML động trong trình duyệt. Điều này có thể gây khó khăn cho các công cụ tìm kiếm để lập chỉ mục và xếp hạng nội dung của ứng dụng.

Kích thước gói: Các ứng dụng ReactJS có thể trở nên khá lớn, điều này có thể ảnh hưởng đến thời gian tải và hiệu suất của ứng dụng. Để giảm kích thước, các công cụ như webpack và Browserify có thể được sử dụng để đóng gói và thu nhỏ mã ứng dụng.

1.2. NextJS

1.2.1. Khái niệm

NextJS là framework mã nguồn mở được xây dựng trên nền tảng của React.

ReactJS cho phép chúng ta xây dựng các trang web tĩnh có tốc độ siêu nhanh và thân thiện với người dùng, cũng như xây dựng các ứng dụng web React.

NextJS được ra đời vào năm 2016, thuộc sở hữu của Vercel. NextJS bắt đầu trở nên phổ biến vào năm 2018 và tiếp tục tăng trưởng mạnh mẽ trong cộng đồng phát triển web vào những năm sau đó. Sự kết hợp của các tính năng như Server-side Rendering (SSR) với Static Site Generation (SSG) đã giúp NextJS trở thành sự lựa chọn hấp dẫn cho nhiều dự án phát triển ứng dụng web.

1.2.2. Đặc điểm

Kiến trúc tĩnh: Next.js sử dụng kiến trúc tĩnh, có nghĩa là mã của bạn được biên dịch thành các tệp HTML, CSS và JavaScript tĩnh tại thời điểm xây dựng. Điều này làm cho Next.js rất nhanh, vì các tệp tĩnh có thể được tải xuống và hiển thị ngay lập tức mà không cần phải xử lý bất kỳ mã nào trên máy chủ.

Khả năng tạo trang web có khả năng tương tác cao: Next.js cung cấp một số tính năng giúp bạn dễ dàng tạo trang web có khả năng tương tác cao, chẳng hạn như hỗ trợ cho các thành phần tương tác, chuyển đổi trang mượt mà và các hiệu ứng hoạt hình.

Tối ưu hóa SEO: Next.js được thiết kế để tối ưu hóa cho SEO, có nghĩa là các trang web được xây dựng bằng Next.js sẽ có thứ hạng cao hơn trong các công cụ tìm kiếm.

Hỗ trợ TypeScript: Next.js hỗ trợ TypeScript, một siêu tập hợp của JavaScript, giúp bạn viết mã an toàn hơn và bảo trì dễ dàng hơn.

1.2.3. Hạn chế

Khó khăn trong việc tích hợp với một số thư viện bên ngoài: Một số thư viện và plugin có thể cần phải điều chỉnh hoặc tùy chỉnh để hoạt động tốt với Next.js. Ví dụ như để sử dụng Redux trong ứng dụng NextJS, các bạn cần cài thêm thư viện next-redux-wrapper để quản lý state trên cả server và client.

Đòi hỏi chạy trên server NodeJS: Để deploy ứng dụng NextJS, bạn cần có một máy chủ NodeJS, việc này có thể làm tăng chi phí và quá trình triển khai sẽ trở nên phức tạp hơn.

Tốc độ phát triển chậm hơn: Next.js vẫn là một framework tương đối mới và đang trong quá trình phát triển. Do đó, có thể mất nhiều thời gian hơn để xây dựng và triển khai các ứng dụng Next.js so với các framework khác.

Khá phức tạp để thiết lập: Việc thiết lập một ứng dụng Next.js có thể khá phức tạp, đặc biệt là nếu bạn chưa có kinh nghiệm với JavaScript hoặc các framework khác. Bạn sẽ cần phải cài đặt nhiều công cụ và thư viện khác nhau để có thể bắt đầu sử dụng Next.js.

1.3. NestJS

1.3.1. Khái niệm

NestJS là một framework mã nguồn mở để phát triển ứng dụng server-side (backend applications) bằng ngôn ngữ TypeScript hoặc JavaScript. Nó được xây dựng trên cơ sở của Node.js và sử dụng các khái niệm từ TypeScript để tạo ra một môi trường phát triển hiện đại và mạnh mẽ cho việc xây dựng các ứng dụng web và API.

Mục tiêu chính của NestJS là cung cấp một cấu trúc ứng dụng rõ ràng và dễ quản lý, giúp tăng tính bảo trì và sự tổ chức trong mã nguồn. Để đạt được điều này, NestJS triển khai mô hình kiến trúc lõi (core architecture) dựa trên các nguyên tắc của Angular, đặc biệt là sử dụng Dependency Injection (DI) và Modules (Các module).

1.3.2. Đặc điểm

NestS là một framework phát triển backend mã nguồn mở đáng chú ý, được xây dựng dựa trên Node.js và sử dụng ngôn ngữ TypeScript hoặc JavaScript. Đặc điểm nổi bật của NestJS là mô hình kiến trúc lõi giúp tổ chức ứng dụng một cách rõ ràng và dễ quản lý, giúp lập trình viên xây dựng các ứng dụng server-side hiện đại và phức tạp một cách hiệu quả.

Một trong những ưu điểm nổi bật của NestJS là việc hỗ trợ cả TypeScript và JavaScript. Lập trình viên có thể lựa chọn ngôn ngữ phù hợp với nhu cầu và kinh nghiệm của họ. TypeScript là một ngôn ngữ mở rộng của JavaScript, cung cấp kiểu dữ liệu tĩnh và các tính năng nâng cao, giúp mã nguồn dễ đọc và dễ bảo trì hơn.

NestJS triển khai mô hình kiến trúc lõi, trong đó mỗi ứng dụng bao gồm ít nhất một module gốc và có thể có nhiều module con. Mỗi module đại diện cho một phần chức năng cụ thể của ứng dụng, giúp mã nguồn trở nên rõ ràng, tổ chức tốt hơn và dễ quản lý.

Với NestJS, việc sử dụng Dependency Injection (DI) là một cách hiệu quả để giảm sự phụ thuộc cứng giữa các thành phần của ứng dụng. DI giúp bạn tạo ra mã linh hoạt, tái sử dụng và dễ kiểm thử. Các decorator (chú thích) trong NestJS giúp xác định vai trò và mục đích của từng thành phần trong mã nguồn như Controllers, Providers, Middleware, v.v., giúp bạn dễ dàng hiểu và bảo trì mã nguồn.

NestJS hỗ trợ Middleware và Interceptors, giúp bạn thực hiện các thao tác chung trước và sau khi xử lý yêu cầu HTTP. Điều này giúp bạn xử lý các yêu cầu HTTP trước khi chúng đến các route xử lý chính và thay đổi response trước khi nó được gửi về client.

Exception handling (xử lý ngoại lệ) là một phần quan trọng trong NestJS, giúp bạn xử lý các exception xảy ra trong ứng dụng và trả về các thông báo lỗi thích hợp cho client khi có lỗi xảy ra.

1.3.3. Hạn chế

Khúc học cong cao: NestJS có một đường cong học tập dốc hơn một số khuôn khổ JavaScript khác, vì vậy có thể mất nhiều thời gian hơn để bắt đầu. Điều này đặc biệt đúng đối với các nhà phát triển không quen với lập trình hướng đối tượng.

Thiếu một số tính năng: NestJS không có một số tính năng có sẵn trong các khuôn khổ JavaScript khác, chẳng hạn như hỗ trợ các thành phần web tùy chỉnh hoặc khả năng tạo các ứng dụng di động.

Quy trình xây dựng phức tạp: Quy trình xây dựng cho NestJS có thể phức tạp, đặc biệt là đối với các dự án lớn hoặc phức tạp. Điều này có thể khiến việc triển khai và bảo trì ứng dụng trở nên khó khăn hơn.

Khó gỡ lỗi: Quá trình gỡ lỗi cho NestJS có thể khá thách thức, đặc biệt là đối với những người mới bắt đầu. Điều này một phần là do kiến trúc hướng đối tượng của NestJS, có thể gây khó khăn trong việc theo dõi luồng thực thi của ứng dụng.

Thiếu tài liệu và cộng đồng: NestJS vẫn là một khuôn khổ tương đối mới, nên tài liệu và cộng đồng còn khá hạn chế. Điều này có thể khiến việc tìm kiếm trợ giúp hoặc học hỏi từ những người khác trở nên khó khăn hơn.

1.4. Kafka

1.4.1. Khái niệm

Theo "Kafka: The Definitive Guide" của Todd Palino, Gwen Shapira, Neha Narkhede [2], Apache Kafka là một hệ thống nhắn tin phân tán, bền bỉ, có thể mở rộng, được sử dụng để lưu trữ, xử lý và phân tích dữ liệu thời gian thực. Kafka được thiết kế để xử lý khối lượng lớn dữ liệu và cung cấp bảo đảm độ tin cậy cao.

Apache Kafka là một kho dữ liệu phân tán được tối ưu hóa để thu nạp và xử lý dữ liệu truyền phát theo thời gian thực. Dữ liệu truyền phát là dữ liệu được tạo ra liên tục từ hàng nghìn nguồn dữ liệu khác nhau, các nguồn này thường gửi các bản ghi dữ

liệu đồng thời. Nền tảng truyền phát cần phải xử lý luồng dữ liệu liên tục này và xử lý dữ liệu theo trình tự và tăng dần.

Kafka cung cấp ba chức năng chính cho người dùng:

- Xuất bản và đăng ký các luồng bản ghi
- Lưu trữ hiệu quả các luồng bản ghi theo thứ tự tạo bản ghi
- Xử lý các luồng bản ghi trong thời gian thực

Kafka chủ yếu được dùng để xây dựng các quy trình dữ liệu truyền phát trong thời gian thực và các ứng dụng thích ứng với luồng dữ liệu đó. Kafka kết hợp nhắn tin, lưu trữ và xử lý luồng nhằm hỗ trợ hoạt động lưu trữ, phân tích cả dữ liệu lịch sử lẫn dữ liệu trong thời gian thực.

Trong hệ thống này, Kafka được sử dụng với kiến trúc Kafka Kraft, giúp hệ thống quản lý các messages này dễ dàng mở rộng, dễ cài đặt, bảo trì và tăng khả năng phụ hồi khi xảy ra sự cố, so với kiến trúc thông thường của Kafka khi sử dụng Zookeeper.

1.4.2. Đặc điểm

Kafka sử dụng mô hình bản ghi phân vùng, kết hợp các phương pháp tiếp cận hàng đợi nhắn tin và xuất bản đăng ký.

Kafka cung cấp khả năng điều chỉnh quy mô bằng cách cho phép phân vùng được phân phối trên các máy chủ khác nhau.

Dựa trên chính sách, ví dụ: tin nhắn có thể được lưu trữ trong một ngày. Người dùng có thể cấu hình cửa sổ lưu giữ này.

Nhiều người tiêu dùng có thể đăng ký cùng một chủ đề, bởi vì Kafka cho phép cùng một tin nhắn được phát lại trong một khoảng thời gian nhất định.

Các chủ đề được tự động sao chép, nhưng người dùng có thể định cấu hình thủ công để các chủ đề không được sao chép.

Mỗi người tiêu dùng nhận thông tin theo thứ tự vì kiến trúc bản ghi được phân vùng.

Kafka sử dụng giao thức nhị phân qua TCP.

1.4.3. Hạn chế

Độ phức tạp của vận hành: Apache Kafka là một hệ thống phức tạp để vận hành. Nó đòi hỏi một số cấu hình và điều chỉnh cẩn thận để đảm bảo độ tin cậy và hiệu suất cao.

Tăng tải: Apache Kafka có thể gặp khó khăn trong việc xử lý lượng dữ liệu lớn hoặc tăng đột biến. Điều này có thể dẫn đến mất dữ liệu hoặc độ trễ.

Tùy chọn bảo mật hạn chế: Apache Kafka không cung cấp nhiều tùy chọn bảo mật tích hợp sẵn. Điều này có nghĩa là người dùng cần triển khai các biện pháp bảo mật bổ sung để bảo vệ dữ liệu của họ.

Khả năng tương tác hạn chế: Apache Kafka không dễ dàng tương tác với các hệ thống khác, đặc biệt là các hệ thống cũ hơn. Điều này có thể gây khó khăn trong việc tích hợp Kafka với các hệ thống hiện có.

Chi phí cao: Apache Kafka có thể tốn kém để vận hành và duy trì. Điều này là do nó đòi hỏi một số lượng lớn tài nguyên, chẳng hạn như máy chủ, lưu trữ và băng thông.

1.5. Automatic Speech Recognition (ASR)

1.5.1. Khái niệm

Nhận dạng giọng nói, còn được gọi là nhận dạng giọng nói tự động (ASR) hay chuyển giọng nói thành văn bản (STT) là công nghệ cho phép máy tính nhận dạng và chuyển đổi ngôn ngữ nói thành văn bản.

Công nghệ nhận dạng giọng nói sử dụng AI và các mô hình học máy để xác định và phiên âm chính xác các giọng, phương ngữ và mẫu giọng nói khác nhau.

1.5.2. Đặc điểm

Các đặc trưng chính của hệ thống nhận dạng giọng nói là:

Audio preprocessing: Tín hiệu âm thanh thô từ thiết bị đầu vào cần được xử lý trước để cải thiện chất lượng của giọng nói. Mục tiêu chính của tiền xử lý âm thanh là để thu thập được các phần dữ liệu liên quan đến giọng nói có liên quan bằng cách loại bỏ các phần âm thanh không mong muốn và giảm tiếng ồn.

Feature extraction: Chuyển đổi tín hiệu âm thanh đã được xử lý trước đó thành một biểu diễn có nhiều thông tin hơn. Công việc này giúp dữ liệu âm thanh thô trở nên dễ quản lý hơn đối với các mô hình học máy trong hệ thống nhận dạng giọng nói.

Language model weighting: Bước này gán trọng số cho các từ và cụm từ nhất định. Điều này giúp cho các từ và cụm từ đó đó có nhiều khả năng được các hệ thống nhận dạng giọng nói nhận ra khi được truyền vào dữ liệu tương ứng.

Acoustic modeling: Acoustic modeling cho phép bộ nhận dạng giọng nói nắm bắt và phân biệt các đơn vị ngữ âm trong tín hiệu giọng nói. Các mô hình âm thanh được đào tạo trên các tập dữ liệu lớn chứa các mẫu giọng nói từ nhiều nhóm người khác nhau, với các giọng, phong cách nói và đến từ nhiều nơi khác nhau.

Speaker labeling: Speaker labeling cho phép các ứng dụng nhận dạng giọng nói xác định danh tính của nhiều người nói trong bản ghi âm. Mỗi người nói trong dữ liệu đầu vào được gán một nhãn duy nhất, từ đó giúp xác định người nói tại bất kỳ thời điểm nào.

Profanity filtering: Quá trình loại bỏ các từ hoặc cụm từ không phù hợp khỏi dữ liệu âm thanh.

1.5.3. Hạn chế

Trong sách "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition" (2018) [3] viết bởi Daniel Jurafsky and James H. Martin, các tác giả đã đề cập đến một số hạn chế của công nghệ ASR (Speech Recognition) như sau:

Độ chính xác: Hệ thống ASR hiện đại thường có độ chính xác cao, nhưng chúng vẫn không thể đạt được độ chính xác 100%. Độ chính xác phụ thuộc vào nhiều yếu tố, chẳng hạn như chất lượng âm thanh, tốc độ nói, giọng điệu và ngôn ngữ sử dụng.

Hiểu biết về bối cảnh: Hệ thống ASR thường không có khả năng hiểu được bối cảnh của cuộc trò chuyện, điều này có thể dẫn đến hiểu lầm hoặc giải thích không chính xác.

Khả năng xử lý tiếng ồn và tạp âm: Hệ thống ASR có thể gặp khó khăn trong việc xử lý tiếng ồn và tạp âm trong môi trường, chẳng hạn như tiếng ồn đường phố, tiếng người nói chuyện khác hoặc tiếng nhạc.

Nhận diện giọng nói: Hệ thống ASR thường được thiết kế để nhận diện giọng nói của một số người cụ thể, nếu người nói sử dụng giọng nói khác, hệ thống có thể gặp khó khăn trong việc nhận diện.

Sự đa dạng về ngôn ngữ: Hệ thống ASR thường được thiết kế cho một ngôn ngữ cụ thể, nếu người nói sử dụng một ngôn ngữ khác, hệ thống có thể không nhận diện được.

Sự đa dạng về phương ngữ: Hệ thống ASR thường được thiết kế cho một phương ngữ cụ thể, nếu người nói sử dụng một phương ngữ khác, hệ thống có thể gặp khó khăn trong việc nhận diện.

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2.1. Mô tả chung về hệ thống

2.1.1. Mục đích của hệ thống

Hệ thống phục vụ cho quản lý sách bao gồm việc quản lý các thông tin nhà xuất bản, các tác giả, các đầu sách, bao gồm sách giấy, sách nói và sách điện tử do người dùng quản lý, và tăng tính tiện dụng cho độc giả, thính giả với chức năng đồng bộ giữa sách nói và sách điện tử.

2.1.2. Phạm vi của hệ thống

- Chức năng đăng nhập cho admin quản lý.
- Quản lý thông tin nhà xuất bản.
- Quản lý thông tin tác giả.
- Quản lý thông tin các loại sách.
- Đồng bộ giữa đoạn trong sách điện tử với thời gian đoạn đó được phát trong sách nói.

2.1.3. Danh sách các chức năng

STT	Tên chức năng	Mô tả
1	Đăng nhập	Cho phép người quản lý đăng nhập vào hệ thống
2	Quản lý các nhà xuất bản	Cho phép người quản lý xem, thêm, sửa, xóa các nhà xuất bản
3	Quản lý danh sách các tác giả	Cho phép người quản lý xem, thêm, sửa, xóa danh sách các tác giả
4	Quản lý danh sách các đầu sách	Cho phép người quản lý xem, thêm, sửa, xóa danh sách các đầu sách
5	Đồng bộ sách điện tử sang sách nói	Hệ thống trích xuất thông tin, cho biết từng đoạn văn trong sách điện tử ứng với thời gian trong sách nói

Bảng 2.1: Danh sách chức năng của hệ thống

2.1.4. Thông tin các đối tượng

- Nhóm các thông tin liên quan đến con người:
    - Người quản lý: tên đăng nhập, mật khẩu, họ tên, ngày sinh, địa chỉ, email, số điện thoại, ảnh đại diện.
    - Tác giả: họ tên, ngày sinh, mô tả.
- Nhóm các thông tin liên quan đến sơ cở vật chất:
    - Sách: tên, mô tả, số lượng, giá cả, ảnh bìa sách, mã sách ISBN, ngày phát hành, quốc gia, ngôn ngữ, đường dẫn sách điện tử, đường dẫn sách nói.
- Nhóm các thông tin liên quan đến đơn vị, tổ chức:
    - Nhà xuất bản: tên, mô tả, địa chỉ, liên hệ.

2.1.5. Quan hệ giữa các đối tượng, thông tin

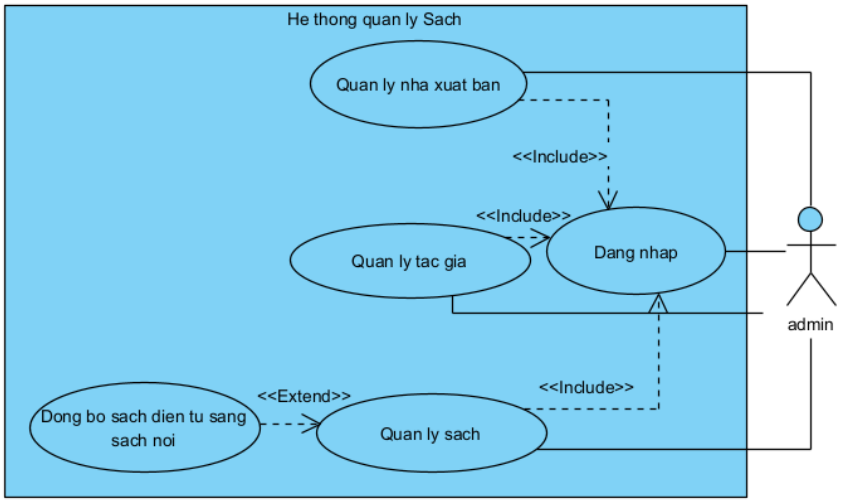
- Một nhà xuất bản có thể phát hành nhiều sách.
- Một đầu sách thuộc một nhà xuất bản.
- Một tác giả có thể viết nhiều sách.
- Một sách có thể được viết bởi nhiều tác giả.
- Một người quản lý có thể thêm nhiều sách vào hệ thống.
- Một đầu sách có một phiên bản sách điện tử.
- Một đầu sách có nhiều chương.
- Mỗi chương sách có một phiên bản sách nói.
- Một nhà xuất bản có thể có nhiều địa chỉ.
- Một nhà xuất bản có thể có nhiều liên hệ.

Quan hệ có thêm trong hệ thống tổng quát:

- Mỗi người dùng có thể có nhiều địa chỉ.
- Mỗi người dùng có thể có nhiều hình thức thanh toán.
- Mỗi người dùng có một giỏ hàng.
- Một giỏ hàng có thể chứa nhiều đầu sách.
- Mỗi người dùng có một thư viện.
- Một thư viện có thể gồm nhiều sách đã mua, đang đọc, đang nghe hoặc yêu thích.
- Mỗi người dùng có thể đặt mua nhiều đơn hàng.
- Mỗi đơn hàng có thể gồm nhiều đầu sách, gồm sách giấy, sách điện tử, sách nói.
- Mỗi đơn hàng có thể có một địa chỉ giao hàng.
- Mỗi đơn hàng có một hình thức thanh toán.

2.2. Biểu đồ use case

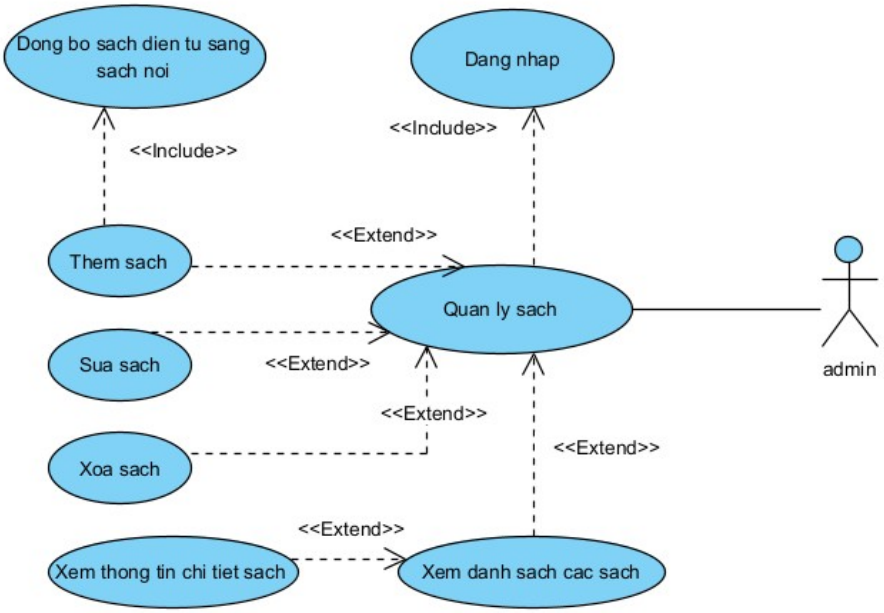
2.2.1. Biểu đồ use case tổng quát



Hình 2.1: Biểu đồ use case tổng quát

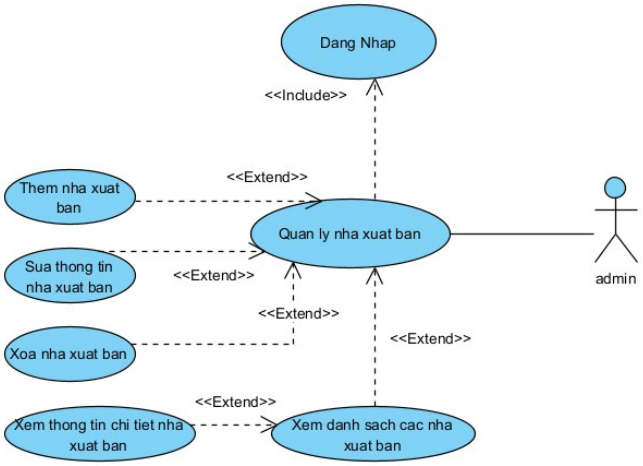
2.2.2. Biểu đồ use case chi tiết

a) Biểu đồ use case quản lý các đầu sách



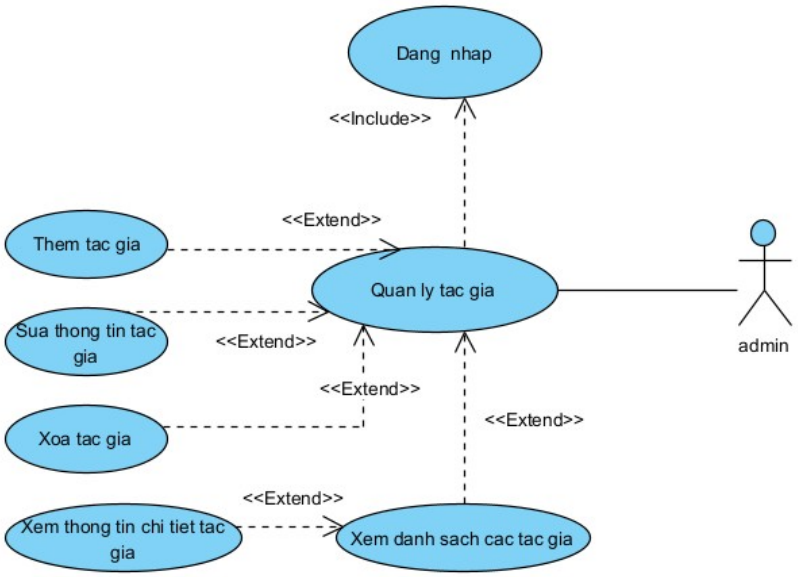
Hình 2.2: Biểu đồ use case chức năng quản lý sách

b) Biểu đồ use case quản lý các nhà xuất bản



Hình 2.3: Biểu đồ use case chức năng quản lý nhà xuất bản

c) Biểu đồ use case quản lý các tác giả



Hình 2.4: Biểu đồ use case chức năng quản lý các tác giả

2.3. Kịch bản chuẩn cho các module

2.3.1. Đăng nhập

Usecase	Người quản lý đăng nhập vào hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã có tài khoản trong hệ thống
Hậu điều kiện	Người quản lý đăng nhập thành công
Kịch bản chính	1. Người quản lý vào trang web quản lý 2. Hệ thống kiểm tra người dùng chưa đăng nhập và điều hướng tới trang đăng nhập 3. Người quản lý nhập tài khoản và mật khẩu được cấp 4. Hệ thống kiểm tra thông tin đăng nhập là đúng và điều hướng sang trang chủ, người quản lý đăng nhập thành công
Ngoại lệ	3.1 Người quản lý nhập sai thông tin mật khẩu 4.1 Hệ thống kiểm tra thông tin đăng nhập là sai và báo lỗi với người dùng

Bảng 2.2: Kịch bản use case đăng nhập

2.3.2. Thêm một sách mới vào hệ thống

Usecase	Thêm sách mới vào hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý tạo một sách mới thành công lên hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các đầu sách 2. Hệ thống hiển thị trang quản lý các đầu sách 3. Người quản lý nhấn nút tạo sách mới 4. Hệ thống hiển thị trang và mẫu tạo một đầu sách mới 5. Người quản lý điền đầy đủ thông tin của đầu sách và ấn vào nút Tạo 6. Hệ thống thực hiện tạo một sách mới, sau đó chuyển

	hướng về trang quản lý đầu sách với danh sách sách chứa đầu sách vừa tạo mới
Ngoại lệ	5.1 Người quản lý điền thiếu thông tin bắt buộc về giá sách và ấn nút Tạo sách 6.1 Hệ thống không thực hiện tạo sách mới và thông báo đỏ ở nơi nhập giá sách

Bảng 2.3: Kịch bản use case thêm sách mới vào hệ thống

2.3.3. Xem chi tiết một sách trong hệ thống

Usecase	Xem thông tin chi tiết một đầu sách trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý xem được thông tin chi tiết của đầu sách
Kịch bản chính	1. Người quản lý nhấn vào nút quản lý các đầu sách 2. Hệ thống hiển thị trang quản lý các đầu sách 3. Người quản lý nhấn nút Xem chi tiết bên cạnh một đầu sách 4. Hệ thống hiển thị trang chứa các thông tin chi tiết của đầu sách
Ngoại lệ	Không có

Bảng 2.4: Kịch bản use case xem thông tin chi tiết một sách trong hệ thống

2.3.4. Chỉnh sửa một sách trong hệ thống

Usecase	Chỉnh sửa một sách trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý chỉnh sửa thành công một đầu sách trong hệ thống

Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các đầu sách 2. Hệ thống hiển thị trang quản lý các đầu sách 3. Người quản lý nhấn nút Chỉnh sửa của một đầu sách 4. Hệ thống hiển thị trang và mẫu chỉnh sửa đầu sách đó với các thông tin đã có 5. Người quản lý điền đầy đủ thông tin mới của đầu sách và ấn vào nút Lưu 6. Hệ thống thực hiện sửa thông tin sách, sau đó chuyển hướng về trang quản lý đầu sách với danh sách sách chứa đầu sách vừa thay đổi
Ngoại lệ	5.1 Người quản lý điền thiếu thông tin bắt buộc về giá sách và ấn nút Lưu 6.1 Hệ thống không thực hiện tạo sách mới và thông báo đỏ ở nơi nhập giá sách

Bảng 2.5: Kịch bản use case sửa thông tin một đầu sách trong hệ thống

2.3.5. Xóa một đầu sách trong hệ thống

Usecase	Xóa một đầu sách trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý xóa thành công đầu sách trong hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các đầu sách 2. Hệ thống hiển thị trang quản lý các đầu sách 3. Người quản lý nhấn nút Xóa của một đầu sách 4. Hệ thống hiển thị thông báo người quản lý có chắc chắn muốn xóa đầu sách đã chọn 5. Người quản lý nhấn đồng ý 6. Hệ thống thực hiện xóa đầu sách, sau đó tải lại trang quản lý đầu sách với danh sách mới không chứa đầu sách vừa được xóa
Ngoại lệ	5.1 Người quản lý nhấn không đồng ý

	6.1 Hệ thống không thực hiện xóa đầu sách và vẫn giữ nguyên ở trang quản lý sách hiện tại

Bảng 2.6: Kịch bản use case xóa một đầu sách trong hệ thống

2.3.6. Thêm một sách nhà xuất bản mới vào hệ thống

Usecase	Thêm nhà xuất bản mới vào hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý thêm một nhà xuất bản mới thành công lên hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các nhà xuất bản 2. Hệ thống hiển thị trang quản lý các nhà xuất bản 3. Người quản lý nhấn nút thêm nhà xuất bản mới 4. Hệ thống hiển thị trang và mẫu tạo một nhà xuất bản mới 5. Người quản lý điền đầy đủ thông tin của nhà xuất bản và ấn vào nút Tạo 6. Hệ thống thực hiện tạo một nhà xuất bản mới, sau đó chuyển hướng về trang quản lý các nhà xuất bản với danh sách sách chứa nhà xuất bản vừa tạo mới
Ngoại lệ	5.1 Người quản lý điền thiếu thông tin bắt buộc về giá sách và ấn nút Tạo sách 6.1 Hệ thống không thực hiện tạo sách mới và thông báo đỏ ở nơi nhập giá sách

Bảng 2.7: Kịch bản use case thêm nhà xuất bản mới vào hệ thống

2.3.7. Xem chi tiết một nhà xuất bản trong hệ thống

Usecase	Xem thông tin chi tiết một nhà xuất bản trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống

Hậu điều kiện	Người quản lý xem được thông tin chi tiết của nhà xuất bản
Kịch bản chính	1. Người quản lý nhấn vào nút quản lý các nhà xuất bản 2. Hệ thống hiển thị trang quản lý các nhà xuất bản 3. Người quản lý nhấn nút Xem chi tiết bên cạnh một nhà xuất bản 4. Hệ thống hiển thị trang chứa các thông tin chi tiết của nhà xuất bản
Ngoại lệ	Không có

Bảng 2.8: Kịch bản use case xem thông tin chi tiết một nhà xuất bản trong hệ thống

2.3.8. Chỉnh sửa một nhà xuất bản trong hệ thống

Usecase	Chỉnh sửa một nhà xuất bản trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý chỉnh sửa thành công một nhà xuất bản trong hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các nhà xuất bản 2. Hệ thống hiển thị trang quản lý các nhà xuất bản 3. Người quản lý nhấn nút Chỉnh sửa của một nhà xuất bản 4. Hệ thống hiển thị trang và mẫu chỉnh sửa nhà xuất bản đó với các thông tin đã có 5. Người quản lý điền đầy đủ thông tin mới của nhà xuất bản và ấn vào nút Lưu 6. Hệ thống thực hiện sửa thông tin nhà xuất bản, sau đó chuyển hướng về trang quản lý nhà xuất bản với danh sách chứa nhà xuất bản vừa thay đổi
Ngoại lệ	5.1 Người quản lý điền thiếu thông tin bắt buộc về tên nhà

	xuất bản và ấn nút Lưu 6.1 Hệ thống không thực hiện sửa thông tin nhà xuất bản và thông báo đỏ ở nơi nhập tên nhà xuất bản

Bảng 2.9: Kịch bản use case sửa thông tin một nhà xuất bản trong hệ thống

2.3.9. Xóa một nhà xuất bản trong hệ thống

Usecase	Xóa một nhà xuất bản trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý xóa thành công nhà xuất bản trong hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các nhà xuất bản 2. Hệ thống hiển thị trang quản lý các nhà xuất bản 3. Người quản lý nhấn nút Xóa của một nhà xuất bản 4. Hệ thống hiển thị thông báo người quản lý có chắc chắn muốn xóa nhà xuất bản đã chọn 5. Người quản lý nhấn đồng ý 6. Hệ thống thực hiện xóa nhà xuất bản, sau đó tải lại trang quản lý các nhà xuất bản với danh sách mới không chứa nhà xuất bản vừa được xóa
Ngoại lệ	5.1 Người quản lý nhấn không đồng ý 6.1 Hệ thống không thực hiện xóa nhà xuất bản và vẫn giữ nguyên ở trang quản lý sách hiện tại

Bảng 2.10: Kịch bản use case xóa một nhà xuất bản trong hệ thống

2.3.10. Thêm một tác giả mới vào hệ thống

Usecase	Thêm tác giả mới vào hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý tạo một tác giả mới thành công lên hệ thống

Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các tác giả 2. Hệ thống hiển thị trang quản lý các tác giả 3. Người quản lý nhấn nút thêm tác giả mới 4. Hệ thống hiển thị trang và mẫu tạo một tác giả mới 5. Người quản lý điền đầy đủ thông tin của tác giả và ấn vào nút Tạo 6. Hệ thống thực hiện tạo một tác giả mới, sau đó chuyển hướng về trang quản lý các tác giả với danh sách chứa tác giả vừa tạo mới
Ngoại lệ	5.1 Người quản lý điền thiếu thông tin bắt buộc về tên tác giả và ấn nút Tạo sách 6.1 Hệ thống không thực hiện tạo tác giả mới và thông báo đỏ ở nơi nhập tên tác giả

Bảng 2.11: Kịch bản use case thêm tác giả mới vào hệ thống

2.3.11. Xem chi tiết một tác giả trong hệ thống

Usecase	Xem thông tin chi tiết một tác giả trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý xem được thông tin chi tiết của tác giả
Kịch bản chính	1. Người quản lý nhấn vào nút quản lý các tác giả 2. Hệ thống hiển thị trang quản lý các tác giả 3. Người quản lý nhấn nút Xem chi tiết bên cạnh một tác giả 4. Hệ thống hiển thị trang chứa các thông tin chi tiết của tác giả
Ngoại lệ	Không có

Bảng 2.12: Kịch bản use case xem thông tin chi tiết một tác giả trong hệ thống

2.3.12. Chỉnh sửa một tác giả trong hệ thống

Usecase	Chỉnh sửa một tác giả trong hệ thống

Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý chỉnh sửa thành công thông tin của một tác giả trong hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các tác giả 2. Hệ thống hiển thị trang quản lý các tác giả 3. Người quản lý nhấn nút Chỉnh sửa của một tác giả 4. Hệ thống hiển thị trang và mẫu chỉnh sửa tác giả đó với các thông tin đã có 5. Người quản lý điền đầy đủ thông tin mới của tác giả và ấn vào nút Lưu 6. Hệ thống thực hiện sửa thông tin tác giả, sau đó chuyển hướng về trang quản lý tác giả với danh sách sách chứa tác giả vừa thay đổi
Ngoại lệ	5.1 Người quản lý điền thiếu thông tin bắt buộc về tên tác giả và ấn nút Lưu 6.1 Hệ thống không thực hiện lưu thông tin mới của tác giả và thông báo đỏ ở nơi nhập tên của tác giả

Bảng 2.13: Kịch bản use case sửa thông tin một tác giả trong hệ thống

2.3.13. Xóa một tác giả trong hệ thống

Usecase	Xóa một tác giả trong hệ thống
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý xóa thành công tác giả trong hệ thống
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các tác giả 2. Hệ thống hiển thị trang quản lý các tác giả 3. Người quản lý nhấn nút Xóa của một tác giả 4. Hệ thống hiển thị thông báo người quản lý có chắc chắn muốn xóa tác giả đã chọn 5. Người quản lý nhấn đồng ý

	6. Hệ thống thực hiện xóa tác giả, sau đó tải lại trang quản lý tác giả với danh sách mới không chứa tác giả vừa được xóa
Ngoại lệ	5.1 Người quản lý nhấn không đồng ý 6.1 Hệ thống không thực hiện xóa tác giả và vẫn giữ nguyên ở trang quản lý các tác giả hiện tại

Bảng 2.14: Kịch bản use case xóa một tác giả trong hệ thống

2.3.14. Đồng bộ sách nói và sách điện tử

Usecase	Đồng bộ sách nói và sách điện tử
Actor	Người quản lý
Tiền điều kiện	Người quản lý đã đăng nhập vào hệ thống
Hậu điều kiện	Người quản lý lấy trích xuất được dữ liệu đồng bộ giữa sách nói và sách điện tử
Kịch bản chính	1. Người quản lý nhấn vào nút xem danh sách các đầu sách 2. Hệ thống hiển thị trang quản lý các đầu sách 3. Người quản lý nhấn nút Tạo một đầu sách mới 4. Hệ thống hiển thị giao diện tạo một sách mới 5. Người quản lý nhấn nhập đủ thông tin của đầu sách, bao gồm tải lên sách nói và sách điện tử, cùng với lựa chọn đồng bộ sách nói và sách điện tử 6. Hệ thống thực hiện tạo một sách mới, sau đó kích hoạt sự kiện trích xuất dữ liệu đồng bộ sách, và trả về kết quả tạo sách thành công. 7. Hệ thống khác trích xuất dữ liệu sách. 8. Khi trích xuất thành công, lưu dữ liệu đồng bộ và đổi trạng thái sách sang trạng thái có thể lấy dữ liệu đồng bộ
Ngoại lệ	5.1 Người quản lý không tải lên sách điện tử 6.1 Hệ thống không thực hiện tạo sách, báo lỗi chưa tải lên sách điện tử

Bảng 2.15: Kịch bản use case đồng bộ sách nói và sách điện tử

2.4. Trích rút lớp thực thể

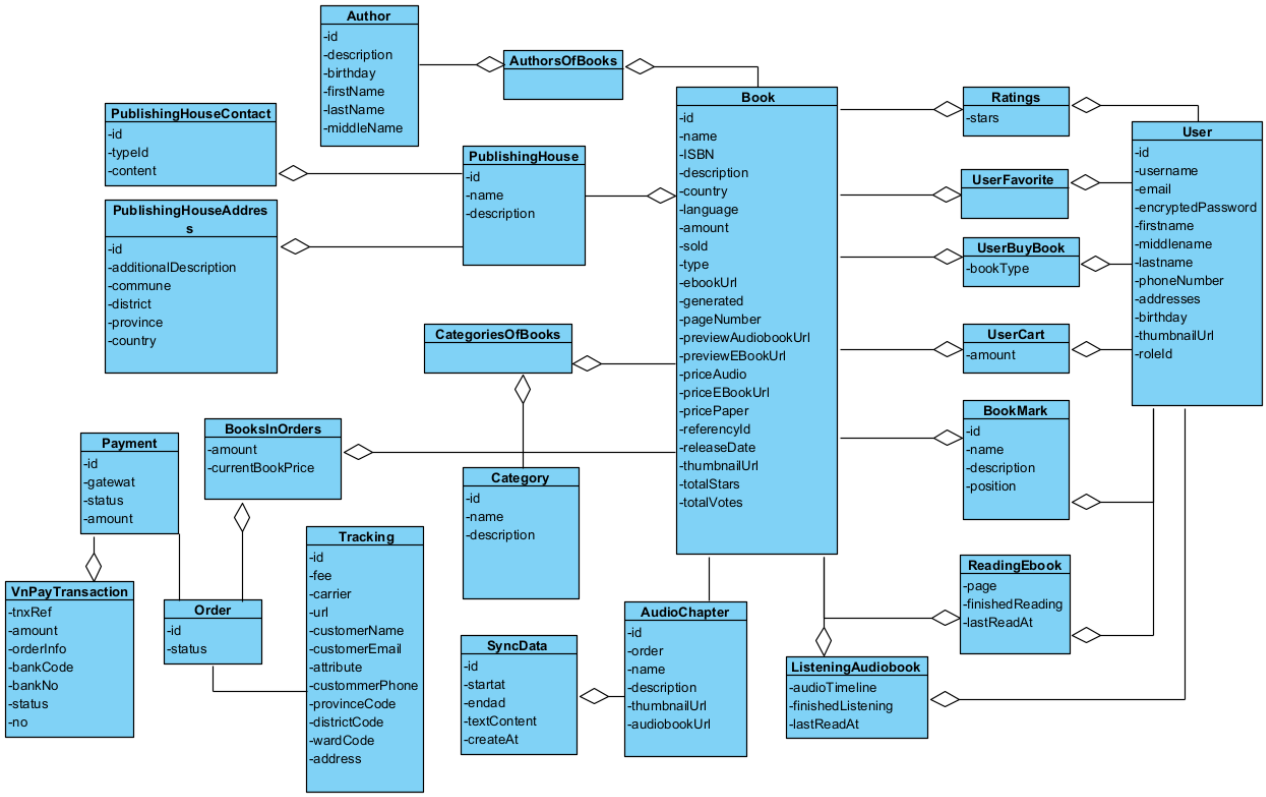
2.4.1. Quan hệ số lượng giữa các thực thể

- Một nhà xuất bản phát hành nhiều đầu sách, một đầu sách chỉ thuộc một nhà xuất bản nên quan hệ giữa nhà xuất bản và đầu sách là 1 – n.
- Một tác giả có thể sáng tác nhiều sách, một cuốn sách có thể được sáng tác bởi nhiều tác giả, nên quan hệ giữa tác giả và sách là n – n. Do đó, đề xuất thêm một lớp kết nối ở giữa tác giả và đầu sách.
- Một người quản lý có thể đăng tải nhiều đầu sách lên hệ thống, một đầu sách chỉ có thể được đăng tải bởi một người quản lý nên quan hệ giữa người quản lý và sách là 1 – n.
- Một đầu sách có nhiều chương sách, một chương sách chỉ thuộc một đầu sách nên quan hệ giữa sách và chương sách là 1 – n.
- Một đầu sách có nhiều thông tin đồng bộ giữa sách nói và sách điện tử, một thông tin đồng bộ giữa sách nói và sách điện tử chỉ chỉ thuộc một đầu sách nên quan hệ giữa sách và thông tin đồng bộ là 1 – n.

Quan hệ có thêm trong hệ thống tổng quát:

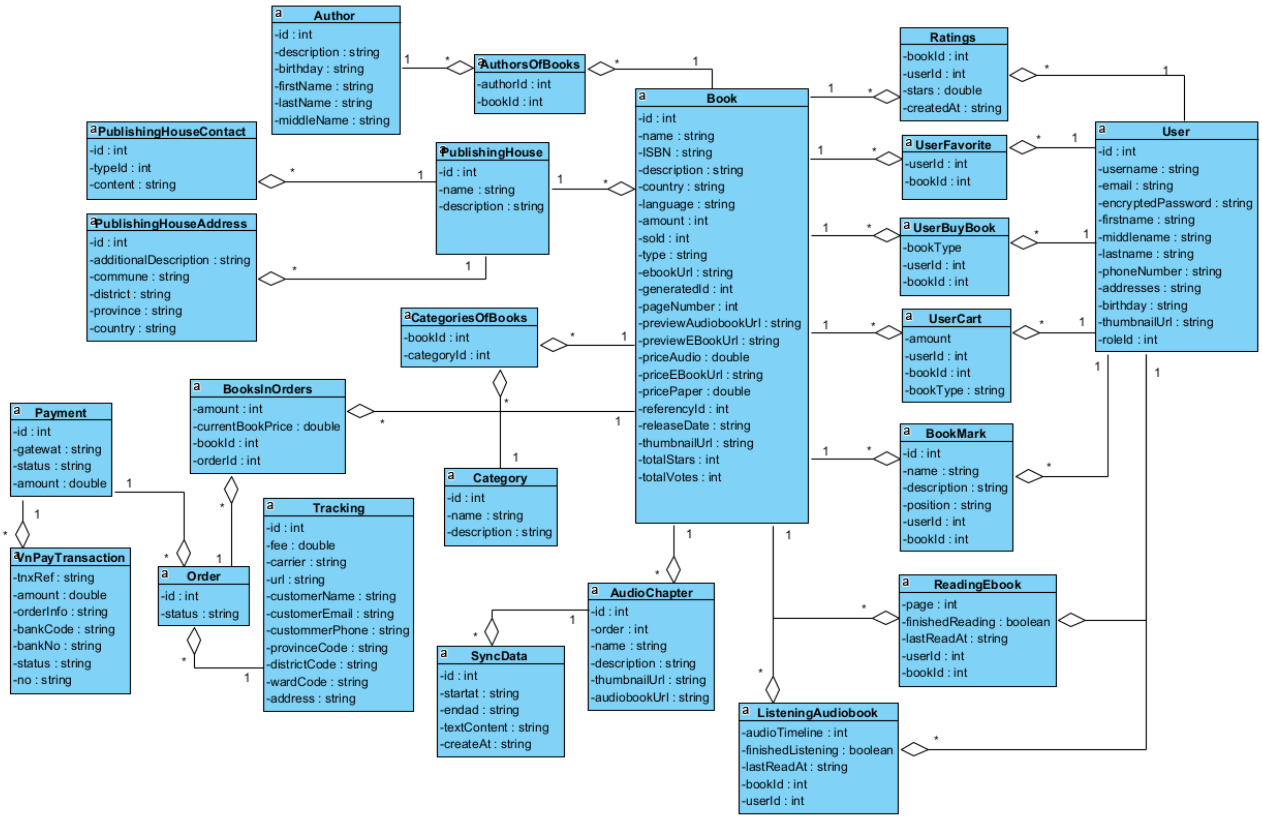
- Mỗi người dùng có thể có nhiều địa chỉ, một địa chỉ chỉ thuộc một người dùng, nên quan hệ giữa người dùng và địa chỉ là 1 – n.
- Mỗi người dùng có thể có nhiều hình thức thanh toán, một hình thức thanh toán chỉ thuộc một người dùng, nên quan hệ giữa người dùng và hình thức thanh toán là 1 – n.
- Mỗi người dùng có một giỏ hàng, một giỏ hành chỉ sở hữu bởi một người dùng, nên quan hệ giữa giỏ hàng và người dùng là 1 – 1.
- Một giỏ hàng có thể chứa nhiều đầu sách, một sách có thể nằm trong nhiều giỏ hàng, nên quan hệ giữa giỏ hàng và sách là n – n, ta đề xuất có thêm một lớp kết nối ở giữa giỏ hàng và đầu sách.
- Mỗi người dùng có thể đặt mua nhiều đơn hàng, một đơn hàng chỉ có thể đặt bởi một người dùng, nên quan hệ giữa người dùng và đơn hàng là 1 – n.
- Mỗi đơn hàng có thể gồm nhiều đầu sách, mỗi đầu sách có thể thuộc nhiều đơn hàng, nên quan hệ giữa đơn hàng và đầu sách là n – n. Do đó, ta thêm một lớp kết nối ở giữa đơn hàng và đầu sách.
- Mỗi đơn hàng có thể có một địa chỉ giao hàng, một địa chỉ giao hàng có thể thuộc nhiều đơn hàng, nên quan hệ giữa đơn hàng và địa chỉ giao hàng là 1 – n.
- Mỗi đơn hàng có một hình thức thanh toán, một hình thức thanh toán có thể thuộc nhiều đơn hàng, nên quan hệ giữa đơn hàng và hình thức thanh toán là 1 – n.

2.4.2. Biểu đồ lớp thực thể pha phân tích



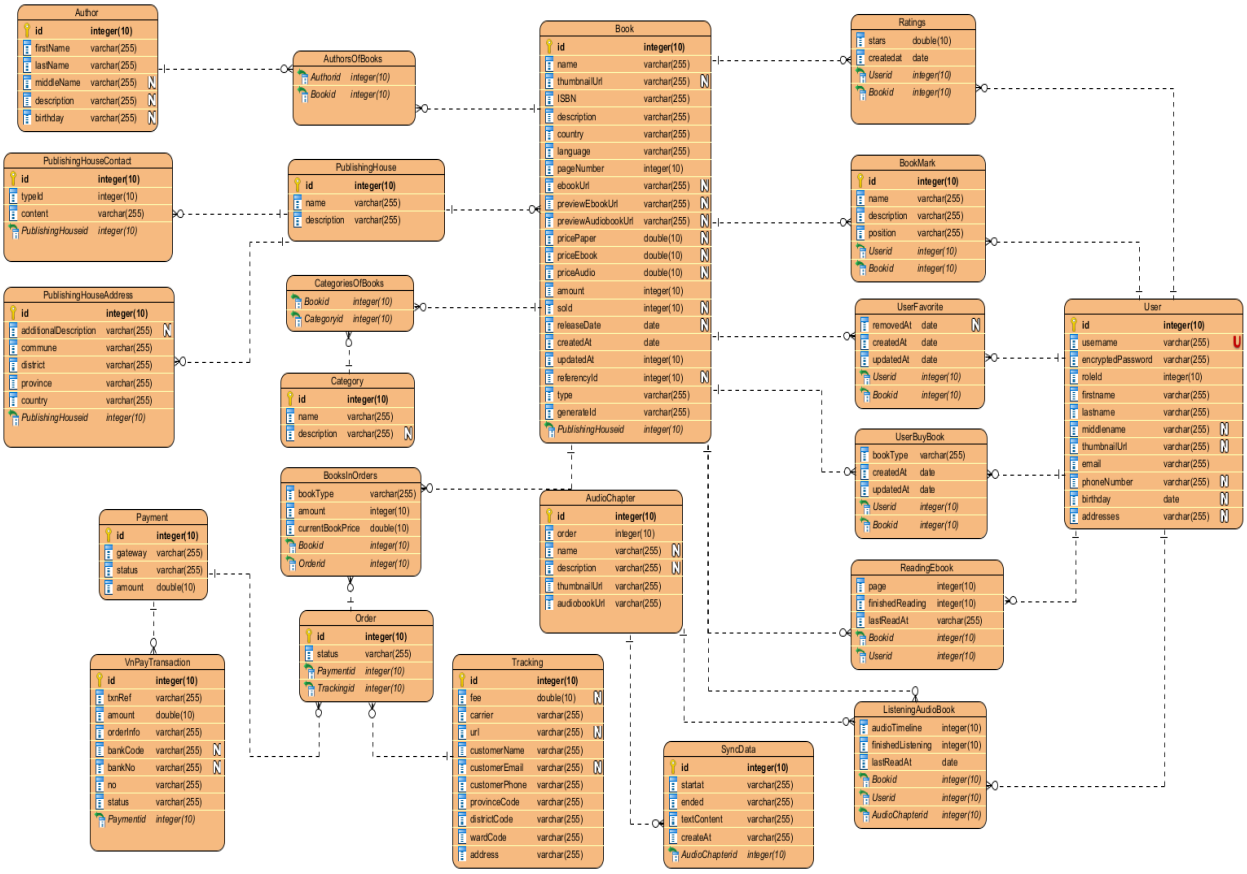
Hình 2.5: Biểu đồ thực thể pha phân tích

2.4.3. Biểu đồ lớp thực thể pha thiết kế



Hình 2.6: Biểu đồ thực thể pha thiết kế

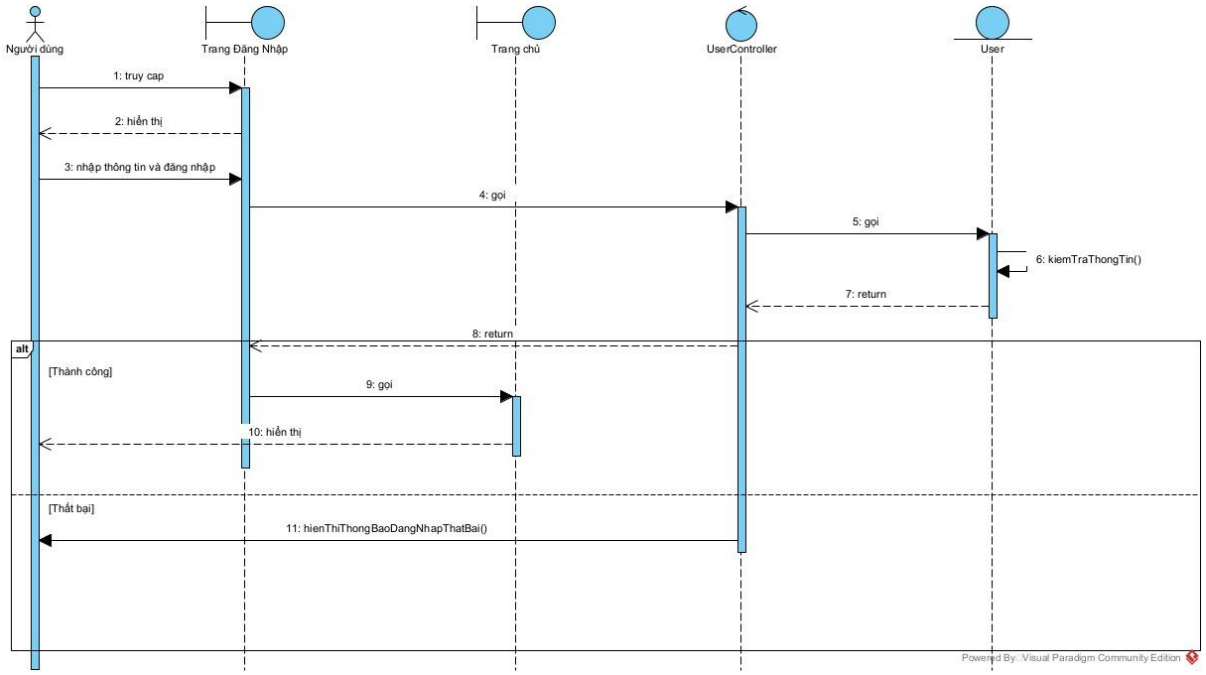
2.4.4. Thiết kế Database



Hình 2.7: Database

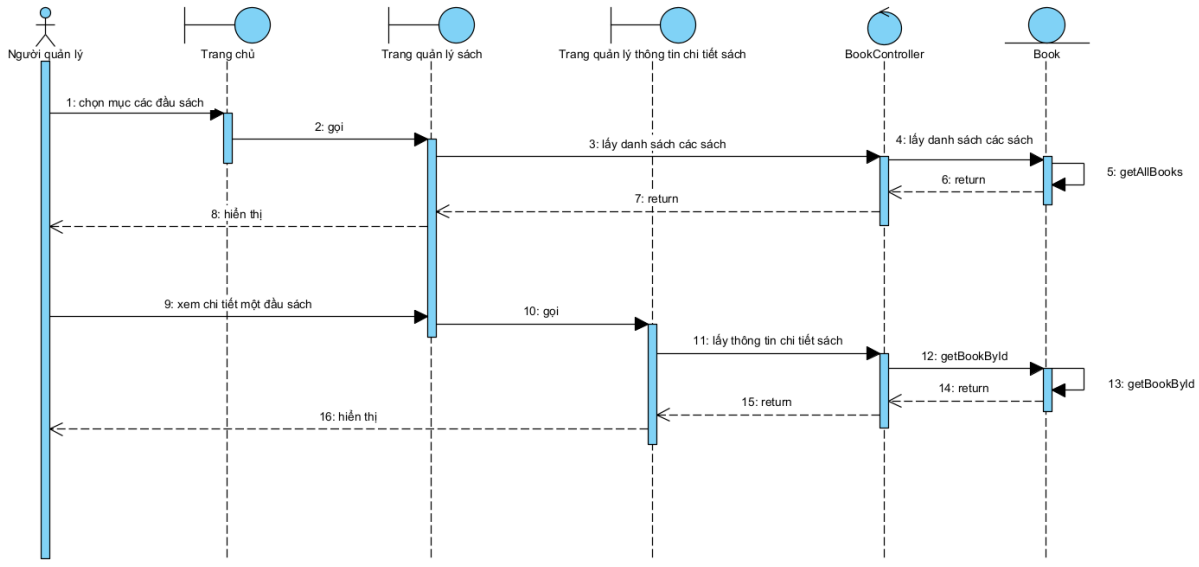
2.4.5. Xây dựng biểu đồ tuần tự

2.4.5.1. Biểu đồ tuần tự chức năng đăng nhập



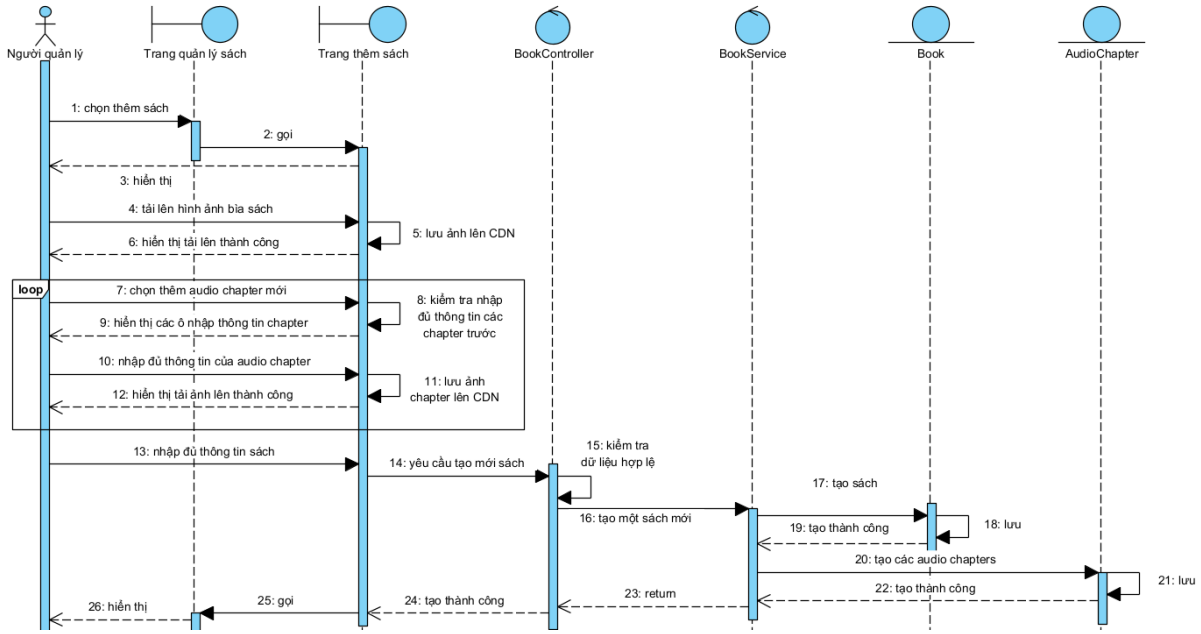
Hình 2.8: Biểu đồ tuần tự chức năng đăng nhập

2.4.5.2. Biểu đồ tuần tự chức năng xem chi tiết thông tin sách



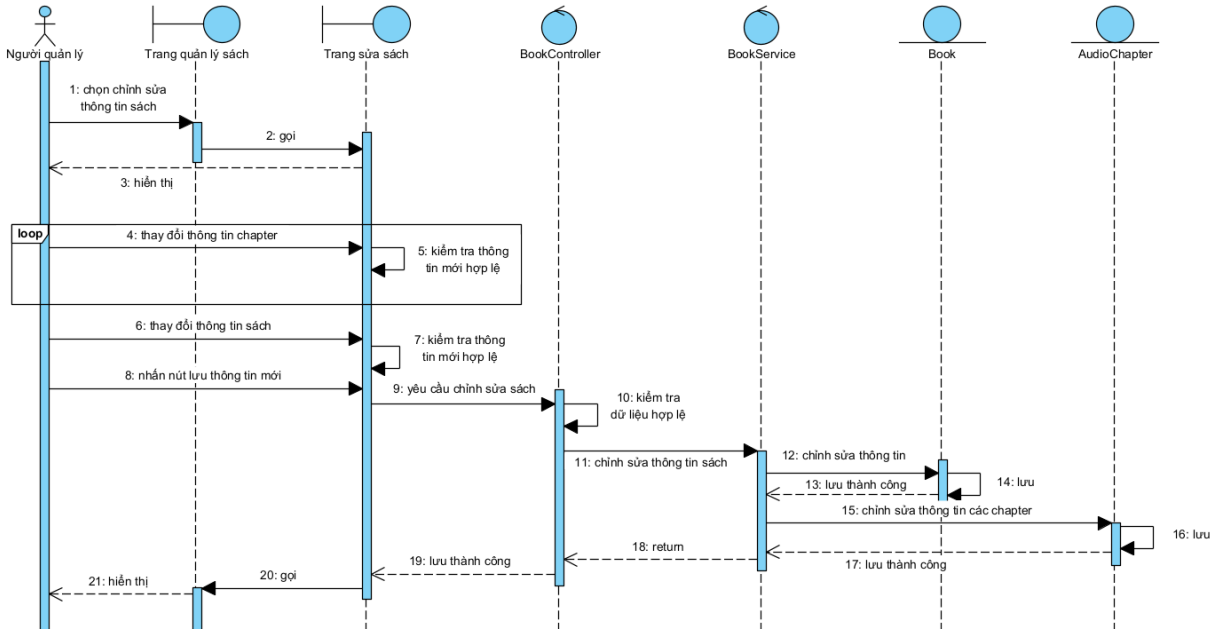
Hình 2.9: Biểu đồ tuần tự chức năng xem chi tiết thông tin sách

2.4.5.3. Biểu đồ tuần tự chức năng thêm mới sách



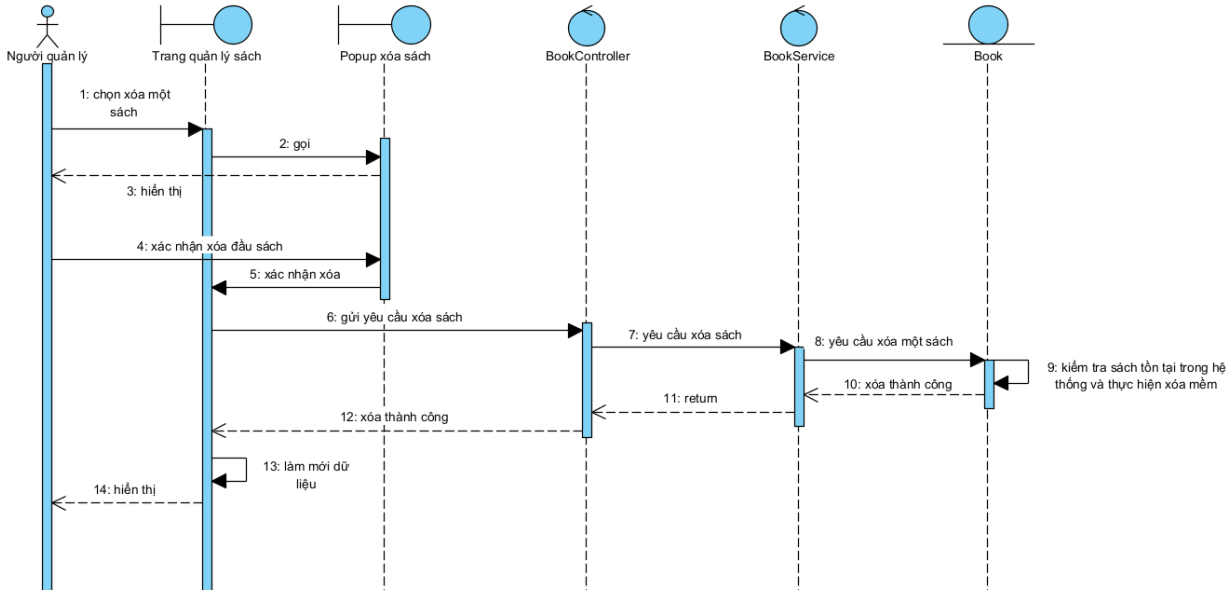
Hình 2.10: Biểu đồ tuần tự chức năng thêm mới sách

2.4.5.4. Biểu đồ tuần tự chức năng sửa thông tin sách



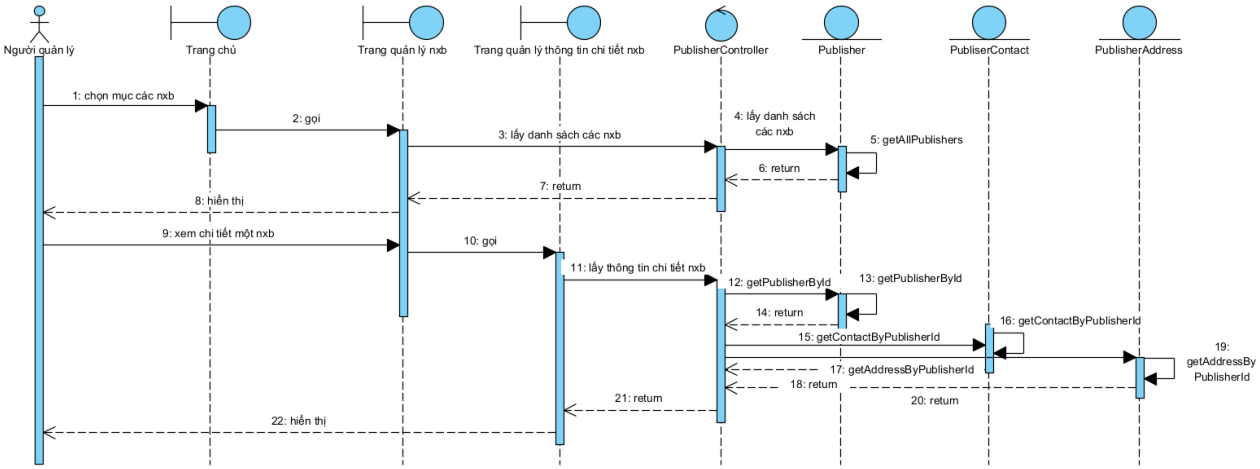
Hình 2.11: Biểu đồ tuần tự chức năng sửa thông tin sách

2.4.5.5. Biểu đồ tuần tự chức năng xóa sách



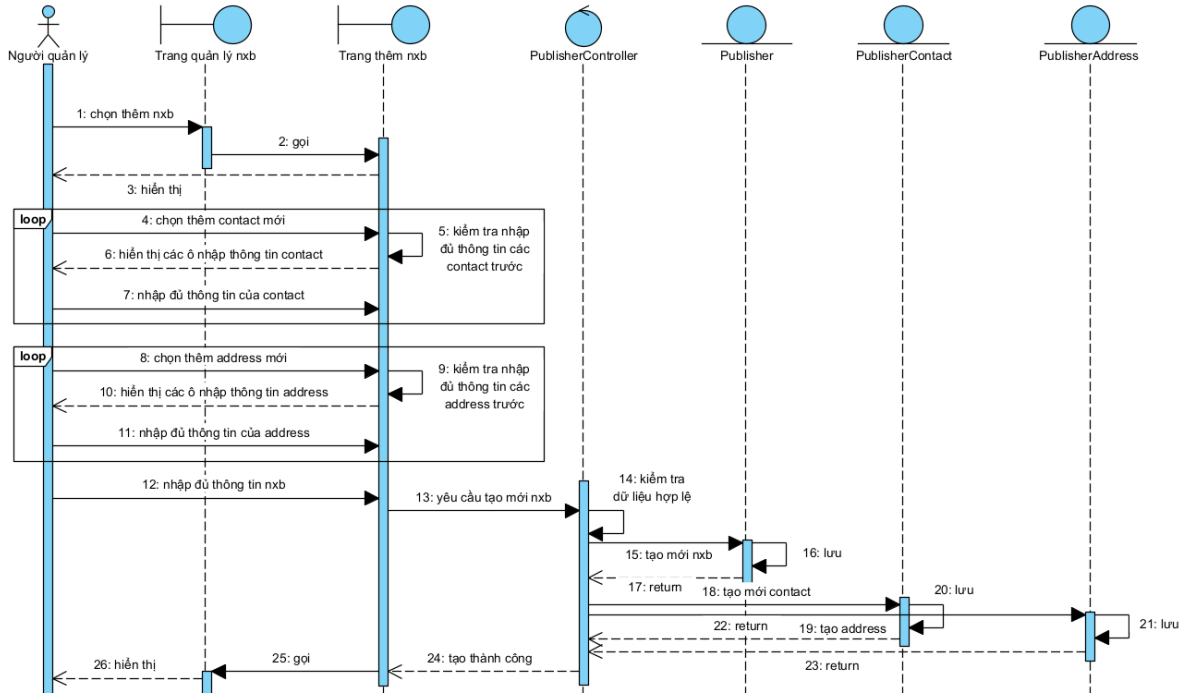
Hình 2.12: Biểu đồ tuần tự chức năng xóa sách

2.4.5.6. Biểu đồ tuần tự chức năng xem chi tiết thông tin nhà xuất bản



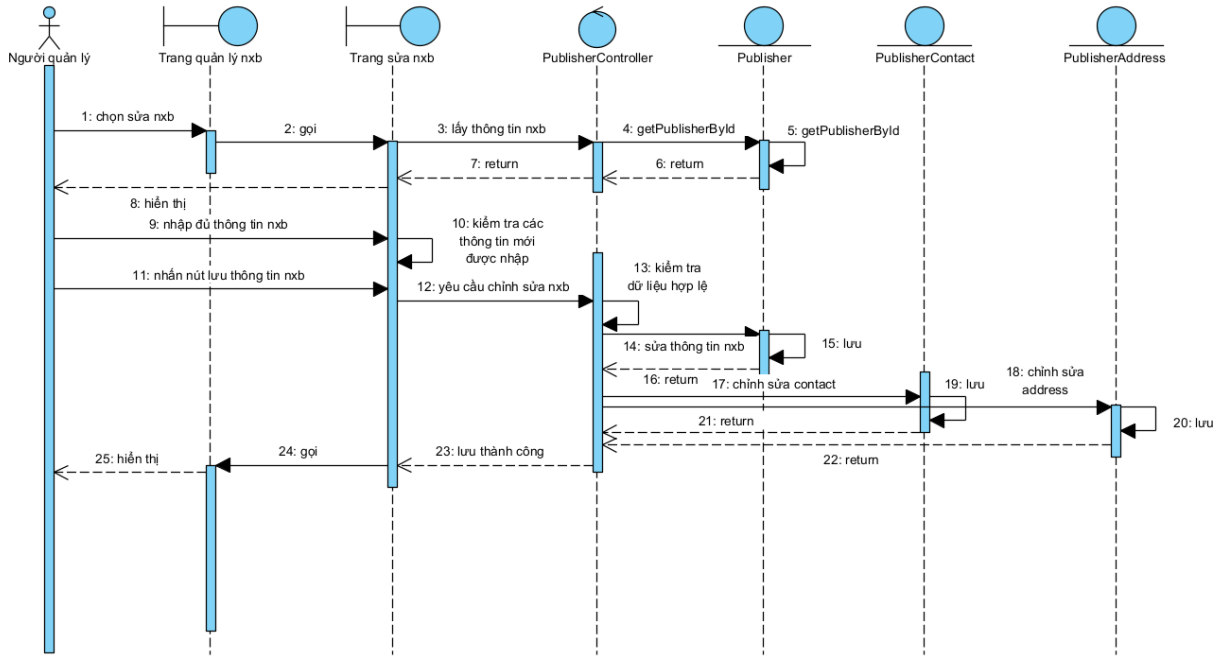
Hình 2.13: Biểu đồ tuần tự chức năng xem chi tiết thông tin nhà xuất bản

2.4.5.7. Biểu đồ tuận tự chức năng thêm mới nhà xuất bản



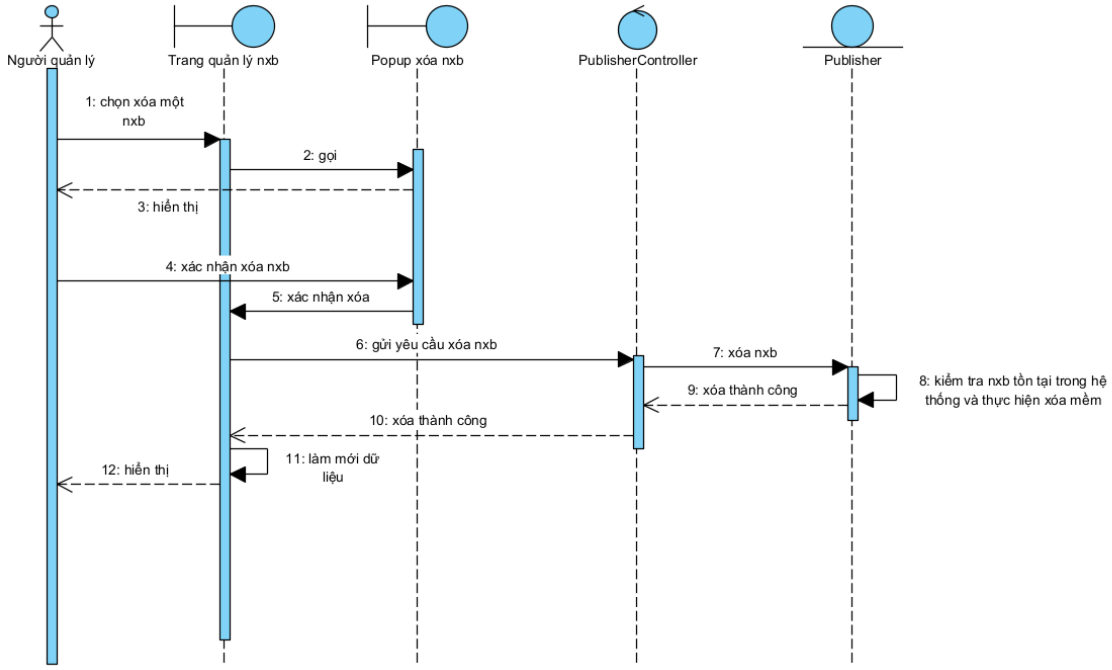
Hình 2.14: Biểu đồ tuần tự chức năng thêm mới nhà xuất bản

2.4.5.8. Biểu đồ tuần tự chức năng chỉnh sửa thông tin nhà xuất bản



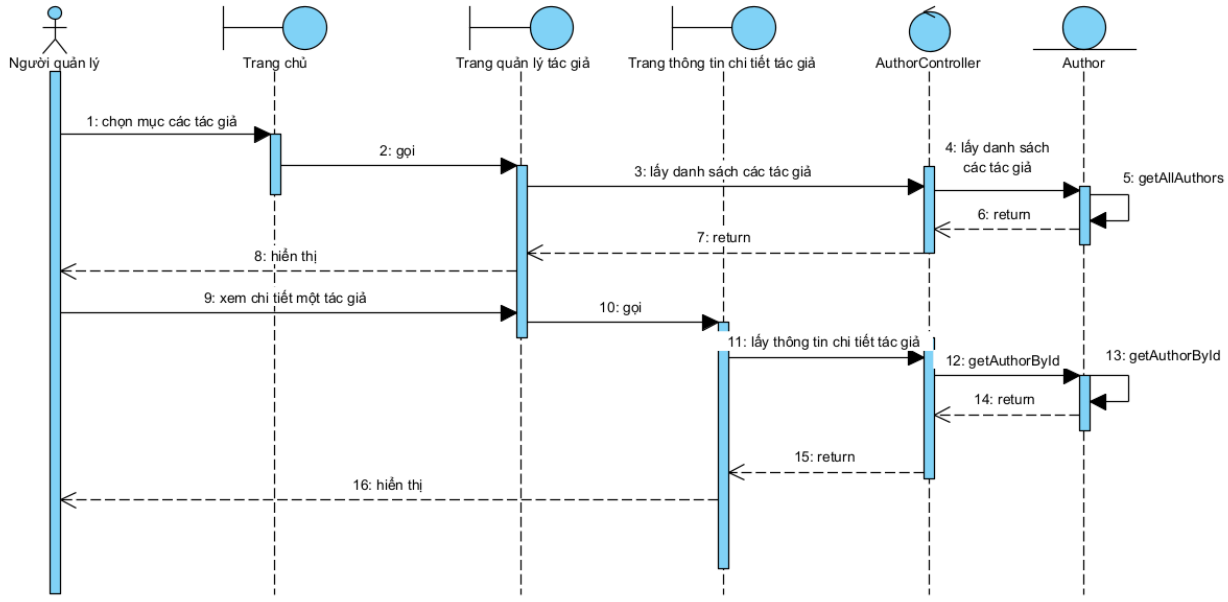
Hình 2.15: Biểu đồ tuần tự chức năng chỉnh sửa thông tin nhà xuất bản

2.4.5.9. Biểu đồ tuần tự chức năng xóa nhà xuất bản



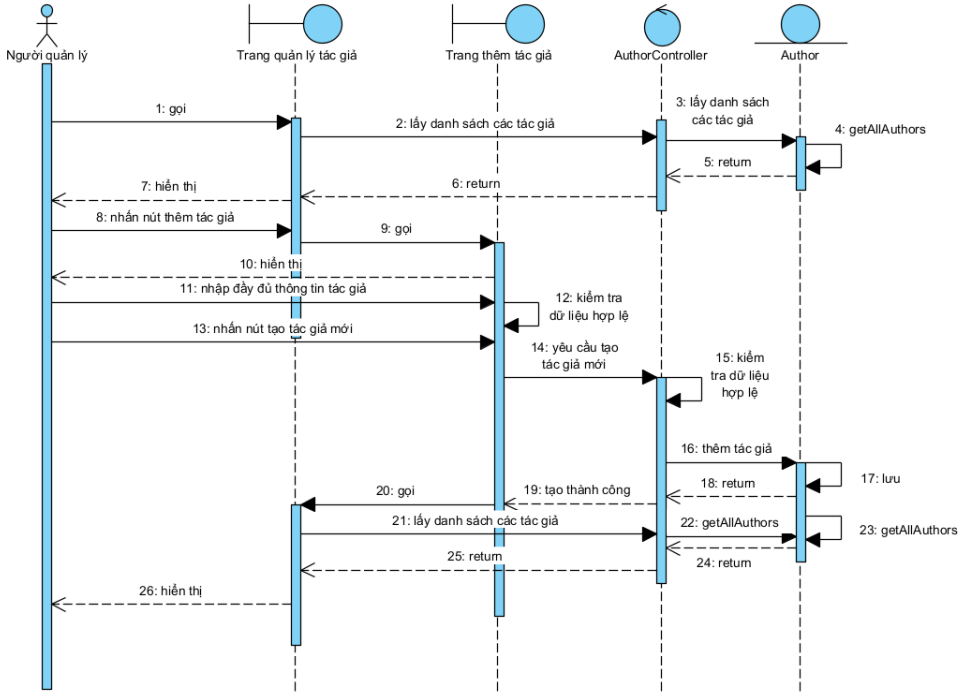
Hình 2.16: Biểu đồ tuần tự chức năng xóa nhà xuất bản

2.4.5.10. Biểu đồ tuần tự chức năng xem thông tin chi tiết tác giả



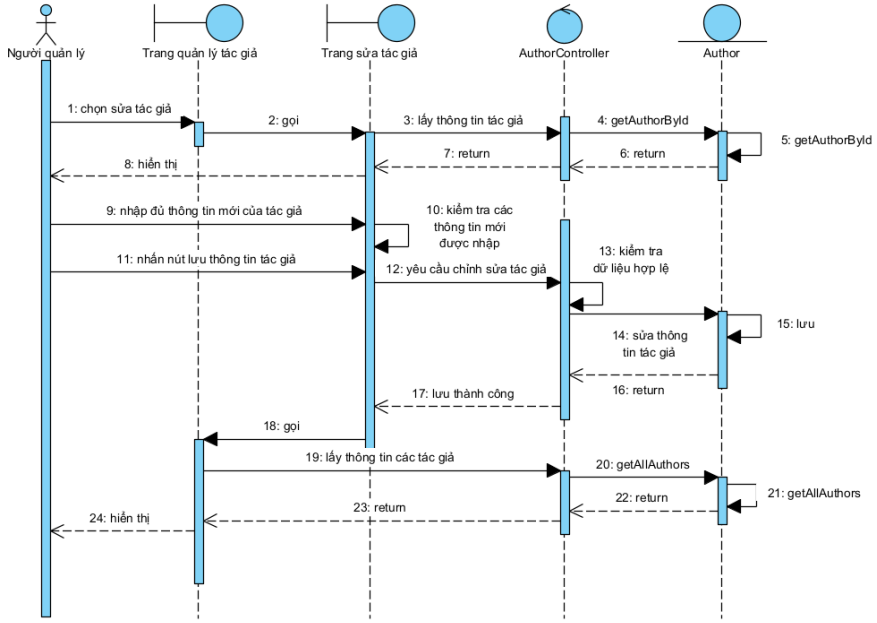
Hình 2.17: Biểu đồ tuần tự chức năng xem thông tin chi tiết tác giả

2.4.5.11. Biểu đồ tuận tự chức năng thêm mới tác giả



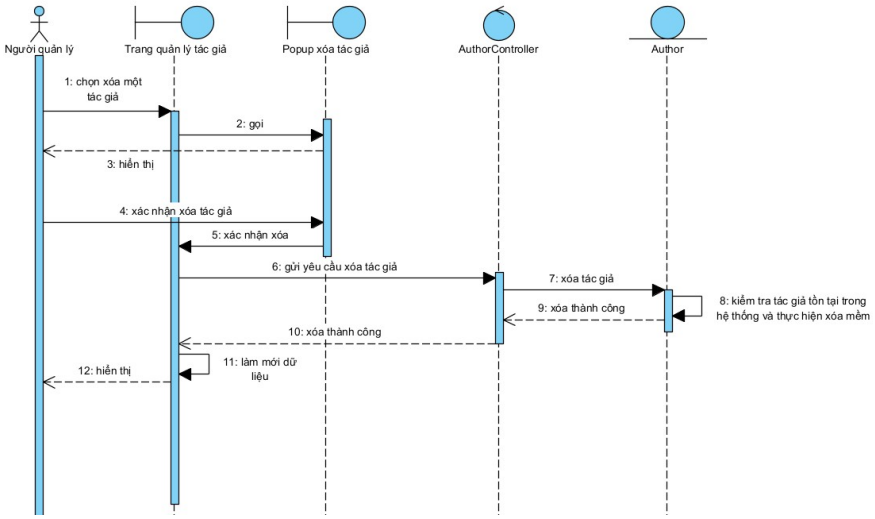
Hình 2.18: Biểu đồ tuần tự chức năng thêm mới tác giả

2.4.5.12. Biểu đồ tuần tự chức năng chỉnh sửa thông tin tác giả



Hình 2.19: Biểu đồ tuần tự chức năng chỉnh sửa thông tin tác giả

2.4.5.13. Biểu đồ tuần tự chức năng xóa tác giả



Hình 2.20: Biểu đồ tuần tự chức năng xóa tác giả

CHƯƠNG 3: PHÁT TRIỂN ỨNG DỤNG

3.1. Chức năng Quản lý các đầu sách

3.1.1. Mô tả chức năng

Chức năng này cho phép người quản lý xem danh sách các đầu sách, xem chi tiết từng đầu sách, thêm một sách mới, sửa hoặc xóa một đầu sách trong hệ thống.

Chức năng tạo sách mới sẽ tạo ra một đối tương sách cùng với đối tượng liên quan là chương sách nói (AudioChapter), và kết nối đối tượng sách với các đối tượng đã có, là nhà xuất bản và tác giả.

```
async create(userId: number, createBookRequest: CreateBookRequest) {
  const bookDto = new CreateBookDto().ParseFromRequest(createBookRequest);
  const book = await this.prisma.book.create({
    data: {
      ...bookDto,
      publishingHouse: {
        connect: {
          id: createBookRequest.publishing_house_id,
        },
      },
      authorsBooks: {
        create: createBookRequest.authorIds.map((ele) => ({ ···
        })),
      },
      audioChapters: {
        create: createBookRequest.chapters.map((ele) => ({ ···
        })),
      },
      userUpload: {
        connect: {      Hoang Anh Duc, last month • create book along with chapters ···
        },
      },
    },
    include: {
      audioChapters: {
        select: { ···
        },
      },
    },
  });

  if (createBookRequest.triggerSync) {
    this.eventEmitter.emit('book.created', new BookCreatedEvent(book));
  }
  return book;
}
```

Hình 3.21: books.service.ts

Sau khi tạo sách thành công, nếu được thông báo cần thực hiện đồng bộ, chương trình sẽ gửi sự kiện "book.created" đến một Listener, một nơi lắng nghe các sự kiện của sách để kích hoạt gửi tin nhắn (message) sang phần xử lý Kafka trong backend.

```
import { Injectable } from '@nestjs/common';
import { OnEvent } from '@nestjs/event-emitter';
import { BookCreatedEvent } from '../events/book-created.event';
import { ProducerService } from 'src/kafka/producer.service';

You, 1 second ago | 2 authors (Hoang Anh Duc and others)
@Injectable()
export class BookCreatedListener {
  constructor(private readonly producerService: ProducerService) {}

  @OnEvent('book.created')
  handleBookCreatedEvent(payload: BookCreatedEvent) {
    this.producerService.produce('book-events', {
      value: JSON.stringify(payload),      Hoang Anh Duc, 3 weeks ago • event-listener
    });
  }
}
```

Hình 3.22: book.listener.ts

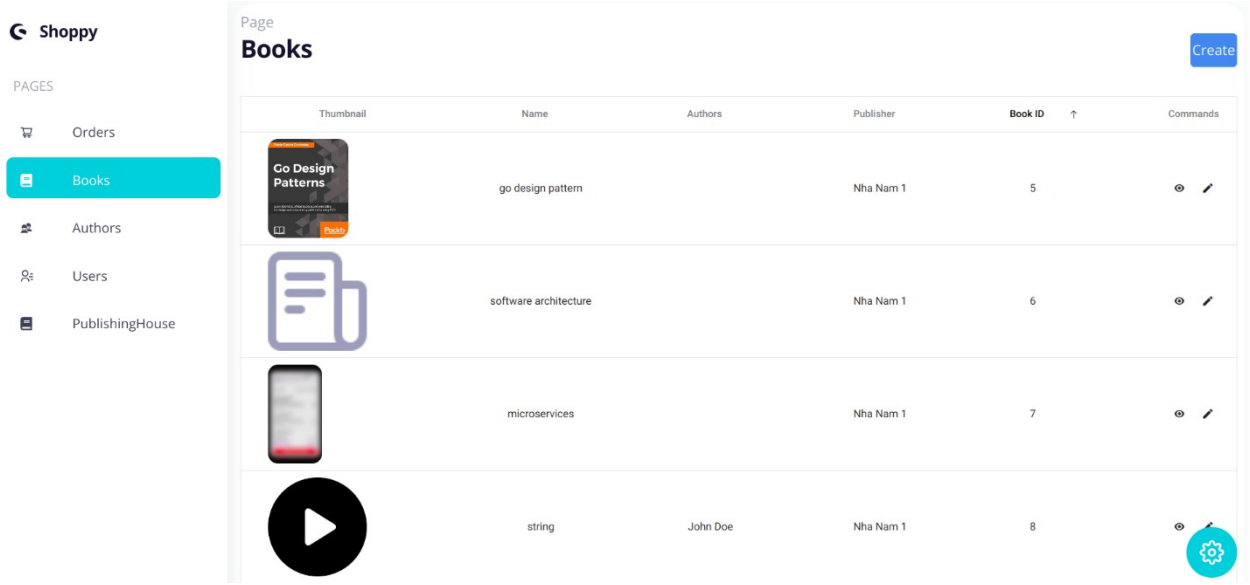
Tiến hành gọi kafkajs.producer để thực hiện gửi tin nhắn (message) đến topic "books-events" của Kafka server.

```
Hoang Anh Duc, 3 weeks ago | 1 author (Hoang Anh Duc)
export class KafkajsProducer implements IProducer {
  private readonly kafka: Kafka;
  private readonly producer: Producer;
  private readonly logger: Logger;

  constructor(private readonly topic: string, broker: string) {
    this.kafka = new Kafka({
      brokers: [broker],
    });
    this.producer = this.kafka.producer();
    this.logger = new Logger(topic);
  }

  async produce(message: Message) {
    await this.producer.send({ topic: this.topic, messages: [message] });
  }

  async connect() {
    try {
      console.log('Connecting ...');

      await this.producer.connect();      Hoang Anh Duc, 3 weeks ago • event-listener and kafka-producer
    } catch (err) {
      this.logger.error('Failed to connect to Kafka.', err);
      await sleep(5000);
      await this.connect();
    }
  }

  async disconnect() {
    await this.producer.disconnect();
  }
}
```

Hình 3.23: kafkajs.producer.ts

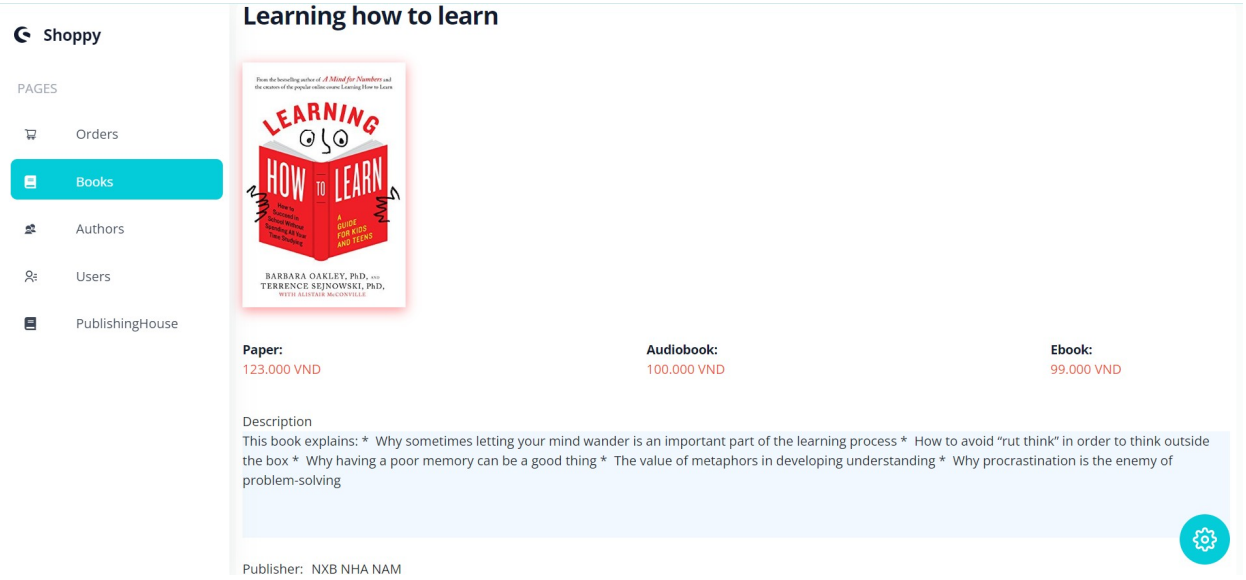
3.1.2. Các giao diện quản lý sách

3.1.2.1. Giao diện trang quản lý các đầu sách

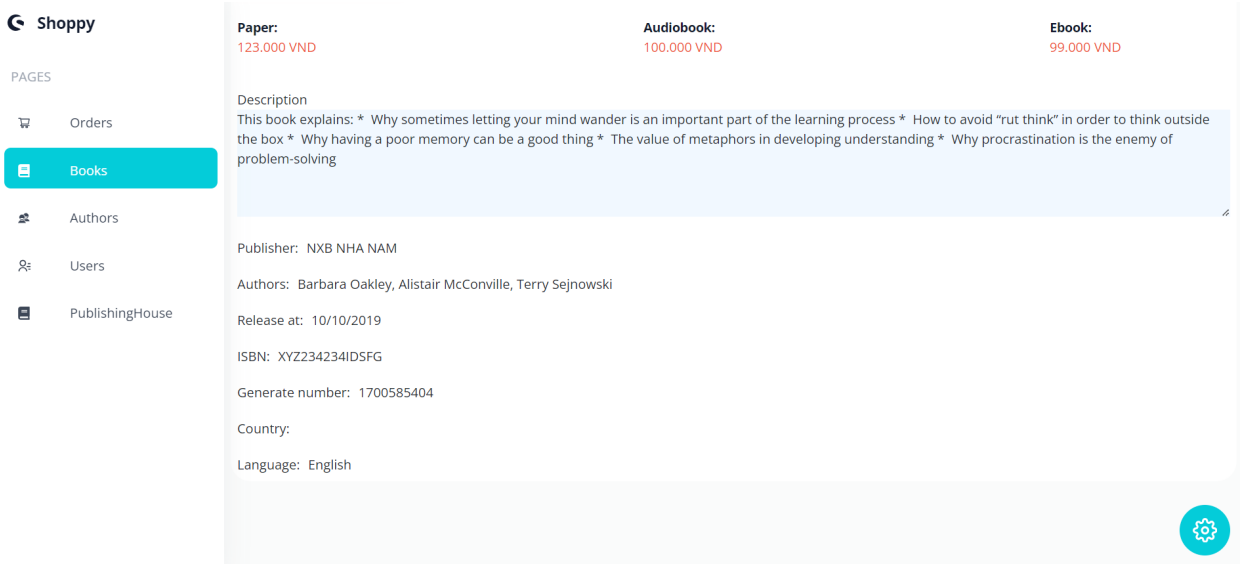


Hình 3.24: Giao diện màn hình quản lý các đầu sách

3.1.2.2. Giao diện trang xem chi tiết đầu sách

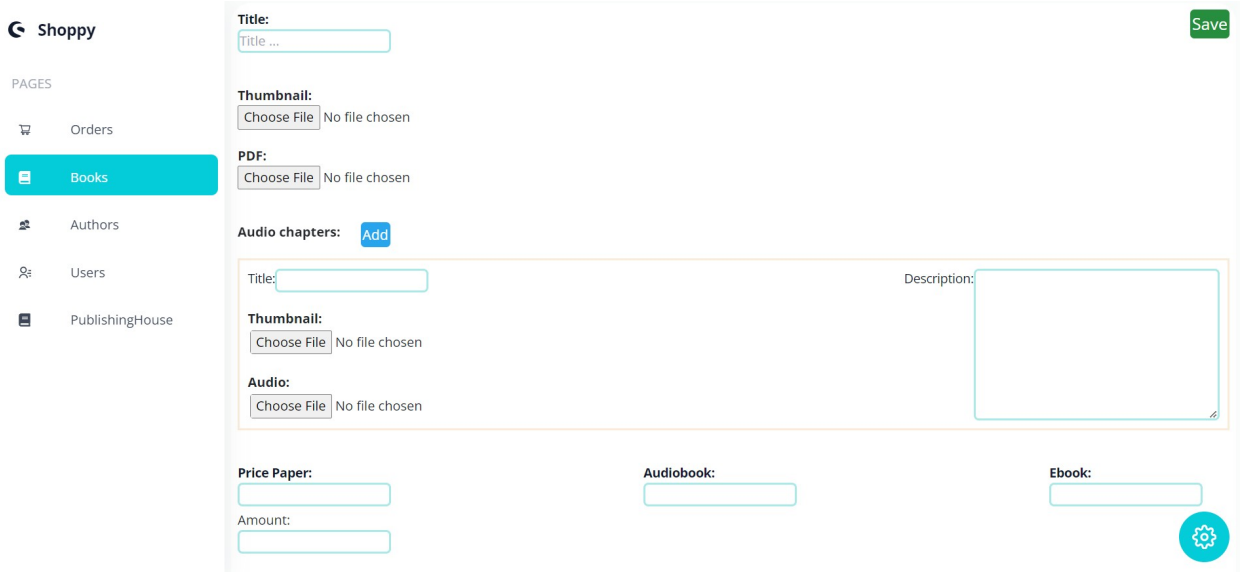


Hình 3.25: Giao diện màn hình xem chi tiết sách



Hình 3.26: Giao diện màn hình xem chi tiết sách (2)

3.1.2.3. Giao diện trang thêm đầu sách



Hình 3.27: Giao diện màn hình thêm sách

3.2. Chức năng Đồng bộ sách nói và sách điện tử

Đồng bộ là một chức năng tiện dụng đối với ứng dụng hỗ trợ nghe sách nói và đọc sách điện tử. Chức năng này đã được biết đến trong ứng dụng Audible với cái tên "Whispersync for Voice", được giới thiệu bởi Amazon, mang lại trải nghiệm với các đầu sách một cách độc đáo và linh hoạt cho người dùng.

Đặc điểm nổi bật của chức năng này có thể kể đến đó là giữ cho dữ liệu của người đọc được đồng bộ giữa phiên bản sách nói và sách điện tử. Khi người đọc chuyển đổi giữa các phiên bản, ứng dụng sẽ tự động đưa tới nơi đã dừng trước đó ở phiên bản trước. Họ không còn cần phải mày mò, tìm kiếm ở phiên bản sách nói hoặc sách điện tử, do đó tiết kiệm được thời gian và tăng trải nghiệm tốt cho người dùng khi sử dụng ứng dụng.

3.2.1. Sơ lược về giải pháp thực hiện

Khi đã có được những đầu vào cần thiết, quá trình đồng bộ, trích xuất dữ liệu giữa hai phiên bản bắt đầu được thực hiện. Quá trình này chia làm 2 phần chính, đó là trích xuất, lấy dữ liệu từ hai phiên bản, và cung cấp dữ liệu đồng bộ mỗi khi ứng dụng cần thông qua Api.

Chú ý: 2 dạng file đầu vào là epub đối với sách điện tử và mp3 đối với sách nói.

3.2.1.1. Giai đoạn trích xuất dữ liệu

Ý tưởng cơ bản để giải quyết bài toán này được mô tả theo 3 bước:

- Dịch file âm thanh: dịch các chương sách nói - chuyển nội dung từ âm thanh thành văn bản.
- Ánh xạ hai bản dịch, xác định vị trí khớp ban đầu: xác định các vị trí có thể là điểm trùng khớp giữa bản dịch từ âm thanh với bản sách điện tử.
- Tìm các vị trí khớp: tìm kiếm vị trí khớp chính xác giữa 2 bản dịch dựa trên vị trí có thể đã được xác định từ bước 2.

3.2.1.2. Cung cấp dữ liệu đồng bộ

Sau khi trích xuất được dữ liệu đồng bộ, back-end sẽ cung cấp thông tin này mỗi khi ứng dụng yêu cầu.

- Đồng bộ từ sách điện tử (epub) sang sách nói (mp3):

Ứng dụng sẽ gửi yêu cầu (request) với tham số cho đầu vào là đoạn văn đầu tiên của trang sách điện tử hiện tại cần đồng bộ. Nếu đoạn văn đầu tiên của trang ngắn hơn 10 từ, sẽ gửi đoạn văn thứ hai nếu có và nếu dài hơn đoạn đầu tiên.

- Đồng bộ từ sách nói (mp3) sang sách điện tử (epub):

Ứng dụng sẽ gửi yêu cầu (request) với tham số đầu vào là khóa chính của chương sách nói hiện tại (chapter_id) và thời gian hiện tại người dùng đang nghe đến (timeline).

3.2.2. Luồng hoạt động

3.2.2.1. Kích hoạt sự kiện xử lý (Trigger)

Sau khi sách được người quản lý tạo mới thành công trên hệ thống, có một Listener trong back-end service sẽ lắng nghe được sự kiện sách được tạo thành công, thực hiện gọi tới KafkaProducer và gửi (public) tin nhắn (message) với chủ đề (topic) sách "book-events" tới Kafka cluster.

Khi nhận thấy topic "book-events" có một message mới, Kafka tiếp tục gửi message này tới những consumers đang đăng ký (subcribe) vào chủ đề đó, chính là matching service.

Matching service nhận được tin nhắn và kích hoạt quá trình trích xuất dữ liệu đồng bộ.

3.2.2.2. Trích xuất dữ liệu

- ***Đọc dữ liệu từ message:***

Tin nhắn được gửi từ back-end service sẽ chứa các thông tin cần thiết, đó là khóa chính của sách (book_id), đường dẫn của sách điện tử (ebook_url) và các đường dẫn của chương sách nói (audiobook_url).

- ***Tải các file sách về matching service:***

Matching service thực hiện tải các file sách nói và sách điện tử về để thực hiện trích xuất từ CDN, ở đây là Amazon S3.

- ***Dịch file sách điện tử và sách nói:***

Thực hiện dịch các file âm thanh mp3 (sách nói) sang văn bản. Các từ được dịch bởi ASR model sẽ có thời gian được phát lên trong audio kèm theo.

Thực hiện lấy toàn bộ nội dung chữ trong file epub (sách điện tử), và cắt bỏ các ký tự đặc biệt, dấu xuống dòng để đưa về dạng giống với văn bản được dịch từ file âm thanh nhất.

- ***Thực hiện matching:***

Thực hiện matching 2 bản dịch với thuật toán được cài đặt. Đầu ra ở bước này, ta sẽ biết được từng đoạn văn của chương trong sách điện tử tương ứng với thời gian nào trong chương sách nói nào.

- ***Lưu dữ liệu trích xuất được vào database:***

Matching service sẽ kết nối trực tiếp đến database và lưu dữ liệu vừa trích xuất vào bảng đã được tạo sẵn.

3.2.3. Giải thích thuật toán để trích xuất dữ liệu đồng bộ

3.2.3.1. *Xác định các tham số*

- ***Đầu vào của thuật toán***

Sau khi thực hiện dịch file âm thanh là các chương sách nói, ta thu được các câu từ cùng với thời gian từ đó được phát lên trong file. Mỗi khoảng thời gian chỉ có 1 từ duy nhất được nói. Do đó, bài toán đồng bộ dữ liệu giữa 2 phiên bản sách, từ việc ánh xạ giữa câu, từ trong file văn bản sách điện tử sang thời gian câu, từ đó được nói trong file âm thanh, trở thành bài toán khớp 2 văn bản gần tương tự, nghĩa là ta sẽ xác định một câu, từ ở trong file văn bản sẽ tương ứng với câu, từ nào được dịch từ file âm thanh.

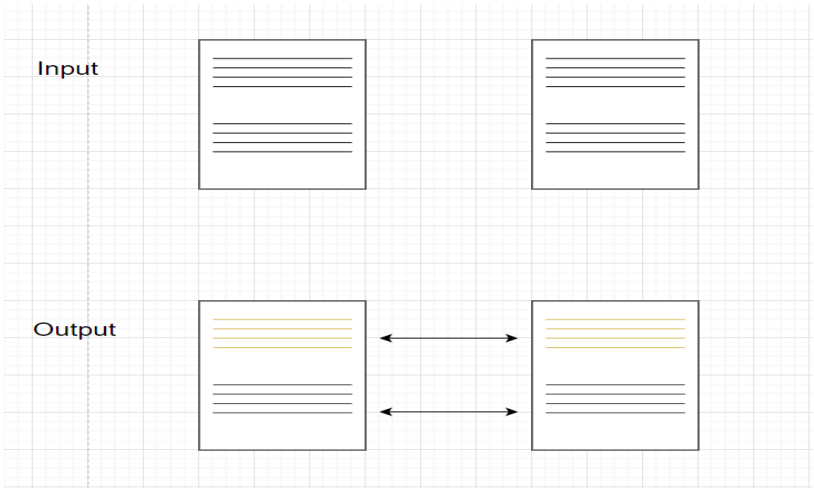
- ***Phần tử lặp trong vòng lặp của thuật toán: đơn vị khớp nhỏ nhất***

Tuy nhiên, vì đặc trưng các ASR model hiện tại chưa đạt được độ chính xác quá cao khi dịch file âm thanh thành các văn bản, nên nếu ta chọn đồng bộ từng từ sẽ là không khả thi. Đồng bộ từng câu cũng sẽ có những trường hợp những câu ngắn dưới 10 từ, độ chính xác khi dịch của ASR model không cao cũng sẽ dễ dẫn đến sai lệch khi khớp 2 đoạn văn bản. Vậy nên ta chọn "đơn vị khớp nhỏ nhất" giữa 2 đoạn văn bản là từng đoạn văn. Như vậy sẽ cải thiện độ chính xác của thuật toán và khiến thời gian trích xuất dữ liệu cũng sẽ nhanh hơn do không phải lặp quá nhiều lần.

Chọn "đơn vị khớp nhỏ nhất" là đoạn văn không chỉ phù hợp bởi độ chính xác của các ASR model mà nó cũng phù hợp trong bối cảnh bài toán đồng bộ giữa sách nói và sách điện tử. Khi người nghe nghe đến một đoạn của chương sách nói, trong trường hợp lý tưởng ta cần biết sách nói đang nhắc đến câu, từ nào ở sách điện tử. Nhưng thực tế không cần như vậy, ta chỉ cần biết đoạn mà sách nói đang phát nằm ở trang nào của sách điện tử. Ngược lại, nếu người đọc đang đọc một trang, một đoạn của sách điện tử, ta sẽ đưa người đọc đến điểm đầu, từ đầu tiên của đoạn, trang đó được phát ở sách nói. Sai số nếu người dùng chọn ở giữa đoạn nhưng bắt đầu phát ở đầu đoạn văn là có thể chấp nhận được. Và trong trường hợp thông thường, trừ khi đoạn văn quá dài, nếu không người đọc sẽ đọc hết đoạn văn để không bị ngắt nhịp trải nghiệm sách. Do đó, lần đồng bộ tiếp theo nếu có sẽ là bắt đầu của một đoạn văn mới.

3.2.3.2. *Ý tưởng thuật toán*

Với bài toán khớp 2 văn bản gần tương tự nhau, ta có đầu vào là 2 văn bản đều đã được chuẩn hóa, bỏ đi các dấu câu, ký tự đặc biệt, chuyển về viết thường và cắt đi các khoảng trống không cần thiết. Đầu ra của thuật toán là sự tương ứng giữa từng đoạn văn trong 2 văn bản.

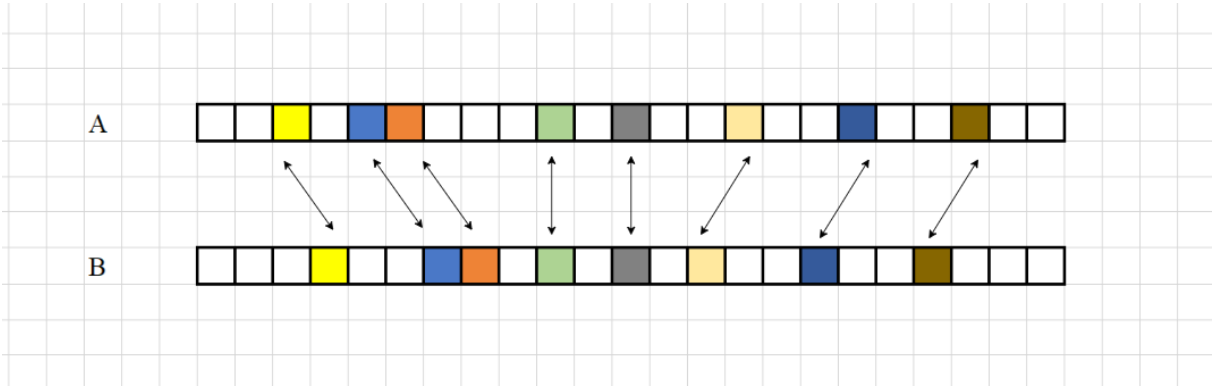


Hình 3.28: Mô tả đầu vào và đầu ra của bài toán

- ***Trường hợp 2 bản dịch khớp từ đầu văn bản***

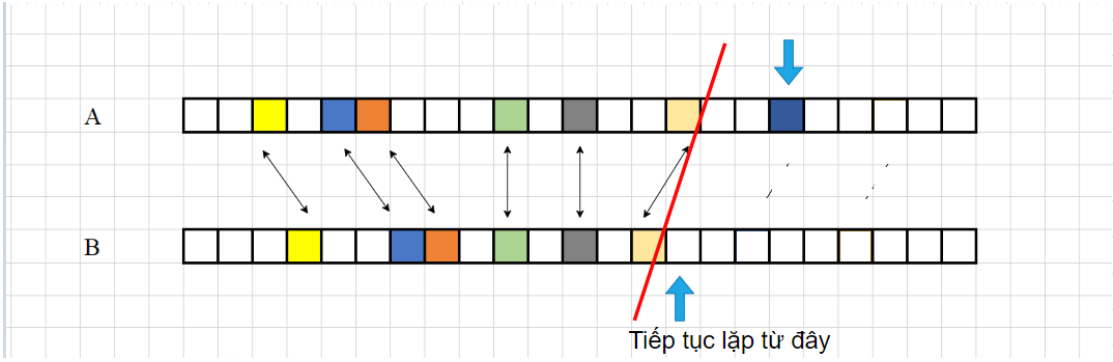
Trong trường hợp tốt, 2 bản dịch, gọi là A và B, sẽ có nội dung khớp với nhau ngay từ mở đầu văn bản, dù ASR model luôn dịch sai một vài từ, có thể xảy ra ở mọi nơi của bản dịch, không làm ảnh hưởng nhiều đến độ chính xác thuật toán. Văn bản dịch dược lấy từ sách điện tử epub sẽ là phiên bản chuẩn. Thuật toán thực thi yêu cầu cần thực hiện so khớp, lặp qua từng đoạn văn của bản sách nói.

Để có cái nhìn đơn giản hơn về bài toán so khớp 2 văn bản, ta tổng quát hóa như sau: coi 2 văn bản là 2 mảng, từng đoạn văn trở thành các phần tử của mảng, phép so sánh 2 đoạn văn đổi từ so khớp gần đúng sang so sánh ngang bằng của 2 phần tử. Bài toán trở thành: cho 2 mảng A, B với A gồm x1 đến xn, B gồm y1 đến ym, tìm các cặp phần tử xi = yj, …, xi+n = yj+n, thứ tự trước sau của mỗi phần tử trong các cặp trên mỗi dãy được bảo đảm.



Hình 3.29: Minh họa các cặp phần tử bằng nhau của bài toán tổng quạt

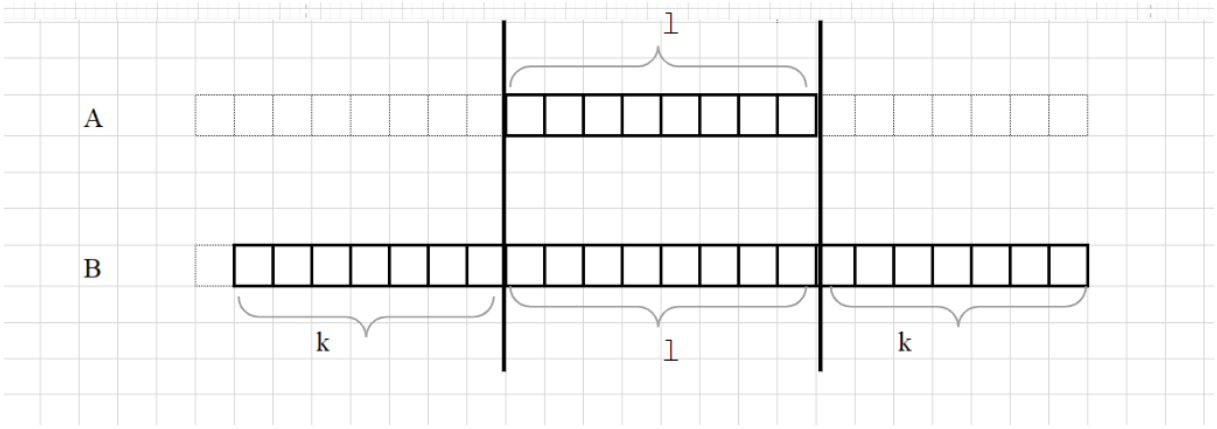
Thoạt nhìn, để thực hiện các cặp phần tử bằng nhau, thuật toán cần 2 vòng lặp lồng nhau, nhưng bởi vì với mỗi phần tử trong vòng lặp của A, ta bắt đầu lặp tiếp từ điểm cuối trong vòng lặp trước của B đến hết, nên độ phức tạp của thuật toán chỉ là O(n+m).



Hình 3.30: Minh họa cách duyệt trong vòng lặp của bài toán tổng quạt

Trở lại với bài toán so khớp 2 văn bản, vòng lặp sẽ có các phần tử lặp là từng đoạn văn, nhưng thay vì so sánh bằng sẽ là so sánh gần đúng đối với đoạn văn.

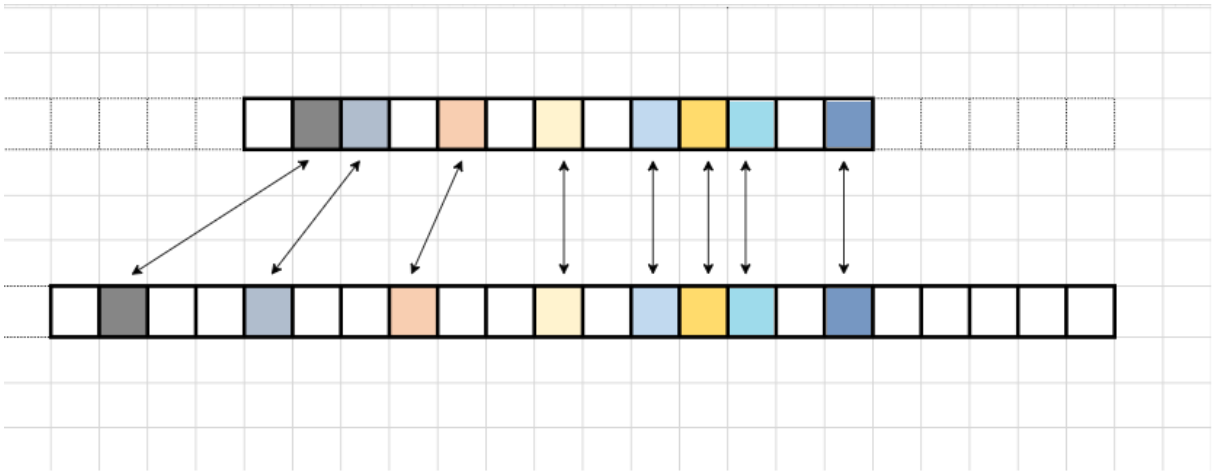
Bởi vì chỉ đoạn văn được lấy từ sách điện tử là văn bản chuẩn, còn đoạn văn dịch từ sách nói sẽ bị dịch sai ở một số từ, cùng với việc người nói có thể thêm bớt 1 vài từ trong khi nói, sai lệch về cả từ ngữ và số lượng từ ở 2 văn bản, nên với mỗi đoạn văn được lặp ở bản dịch sách nói, ta sẽ lấy thêm 1 khoảng k ở 2 đầu để chắc chắn không bỏ sót những phần có thể khớp lại.



Hình 3.31: Minh họa thành phần được duyệt qua trong mỗi vòng lặp

Để xác định phần khớp của 2 đoạn văn trong mỗi vòng lặp, ta áp dụng thuật toán tìm đoạn con chung dài nhất của 2 mảng (longest common subsequence), thuật toán quen thuộc của giải thuật quy hoạch động.

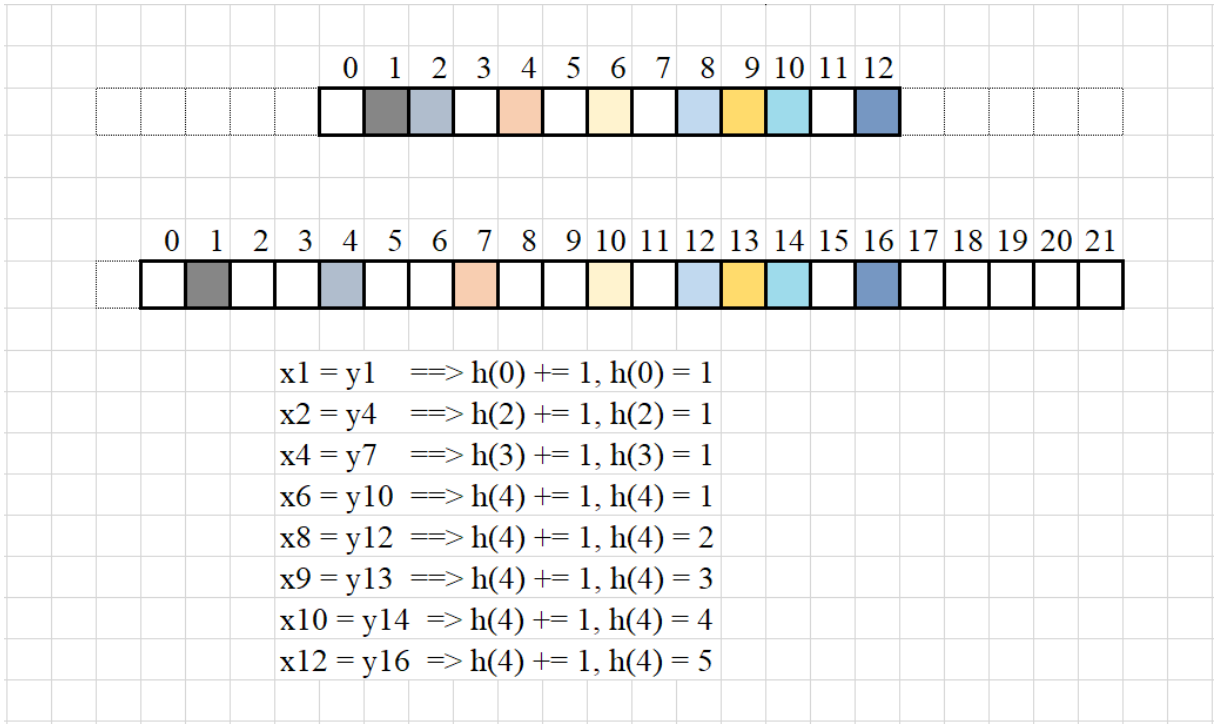
Thông thường đầu ra của thuật toán khi so khớp mỗi 2 đoạn văn, các từ ngữ trùng nhau sẽ nằm thưa thớt như hình thể hiện bên dưới.



Hình 3.32: Minh họa các cặp từ khớp nhau trong 2 đoạn văn

Do hạn chế của đầu ra mặc định của thuật toán tìm đoạn con chung dài nhất và sự thưa thớt của các cặp phần tử trùng nhau sẽ gây khó khăn trong việc xác định đoạn hay từ ngữ khớp, ta cần phát triển thuật toán để phù hợp với yêu cầu hiện tại.

Ở thuật toán mới, ta sẽ tìm từ phù hợp nhất trong B làm từ bắt đầu của đoạn sẽ khớp với đoạn của A. Gọi h(z) là hàm đánh giá sự phù hợp của mỗi từ trong B, h(j) lớn nhất thì từ thứ j trong B là từ phù hợp làm từ đầu tiên của đoạn khớp. Với mỗi cặp từ trùng nhau xi = yj, ta tăng giá trị của h(j – i) lên 1 đơn vị. Ví dụ như hình dưới.



Hình 3.33: Ví dụ minh họa kết quả chạy thuật toán

Vậy ta chọn từ có chỉ số là 4, tức là từ thứ 5 trong đoạn văn của B làm từ bắt đầu của câu sẽ khớp với đoạn của A. Độ phức tạp của thuật toán này vẫn là O(n.m) với n, m là độ dài 2 đoạn của A và B, vì với mỗi từ trong đoạn của A đều sẽ lặp qua các từ trong đoạn của B.

Độ phức tạp tổng quát của bài toán (trong trường hợp 2 bản dịch A, B khớp với nhau ngay từ đầu) là O(n.p.(p+k).(t+z)), với:

- n là số đoạn của bản dịch A.
- p là độ dài trung bình (số từ) của một đoạn văn của A.
- k là khoảng lấy thêm của B.
- t, z là độ dài trung bình của một từ trong A và B.

- ***Trường hợp 2 bản dịch không khớp với nhau ngay từ đầu***

Thông thường, phần đầu của 2 bản dịch thường sẽ không khớp với nhau. Đối với sách nói, người đọc sách sẽ giới thiệu sơ qua về sách, tác giả. Phần này thường không được đọc từ chính cuốn sách. Đối với sách điện tử, phần đầu cuốn sẽ là đề mục, mục lục hoặc lời nói đầu mà người đọc sách nói có thể bỏ qua. Do đó ta cần tìm điểm bắt đầu khớp trong mỗi bản dịch A và B.

Để tìm vị trí bắt đầu khớp, ta cũng sẽ áp dụng thuật toán đã được nêu ra ở phần trước. Tuy nhiên, thay vì lấy 1 khoảng k ngắn ở B như phần trước, ta sẽ lấy khoảng dài k dài hơn, và mỗi khi đoạn trong A không tìm thấy khớp ở trong B, ta tăng k thêm theo cấp số cộng. Thuật toán duyệt vét cạn đến khi tìm thấy vị trí khớp. Độ phức tạp thuật toán trong trường hợp xấu nhất là O(n.(p.r).(t+z)), với:

- n là số đoạn của bản dịch A.
- p là độ dài trung bình 1 đoạn văn của A.
- r là số từ trong bản dịch B.
- t, z là độ dài trung bình của một từ trong A và B.

Độ phức tạp trung bình tổng quát của bài toán trong cả 2 trường hợp vẫn sẽ là O(n.p.(p+k).(t+z)).

3.2.4. Cài đặt thuật toán

- ***Kafka consumer nhận message từ topic và kích hoạt trích xuất dữ liệu đồng bộ:***

```python
from kafka import KafkaConsumer
import json
import os
import execute

# Define server with port
bootstrap_servers = [os.environ['KAFKA_BROKER']]
topicName = os.environ['BOOK_EVENTS_TOPIC']

# Initialize consumer variable
consumer = KafkaConsumer (topicName, bootstrap_servers = bootstrap_servers,
                          value_deserializer=lambda m: json.loads(m.decode('utf-8')))

# Read and print message from consumer
for msg in consumer:
    print(msg)
    execute.do(msg)       You, last week • update
```

Hình 3.34: kafka_consumer.py

- ***Dịch file (lấy bản dịch) sách điện tử và sách nói:***

```python
import re
import tika
from tika import parser

def extract(epub_filepath):
    parsed = parser.from_file(epub_filepath)
    content = parsed["content"]

    output_list = re.split(r'\n+', content)
    output_list = [s.strip() for s in output_list if s.strip()]
    output_list = [re.sub(r'[-\'“”\.,…"@_!#$%^&*()<>?/\|}{~:]+', '', s) for s in output_list]
    output_list = [s.strip().lower() for s in output_list if s.strip()]
    print(output_list)
    return output_list
```

Hình 3.35: extract_text_epub.py

```python
import torch
import math
from transformers import pipeline
from datasets import Dataset, Audio

##############
# load model & audio and run audio through model      You, 2 weeks ago • first commit
model_name = './wav2vec2-base-vietnamese-250h'
device = "cuda:0" if torch.cuda.is_available() else "cpu"


def transcribe(audio_filepath):
    pipe = pipeline(
        "automatic-speech-recognition", model=model_name, device=device
    )
    audio_dataset = Dataset.from_dict({"audio": [audio_filepath]}).cast_column("audio", Audio(sampling_rate=16_000))
    # process
    torch.cuda.empty_cache()
    transcribed_arr = pipe(audio_dataset[0]["audio"], max_new_tokens=256,
                           chunk_length_s=30,batch_size=8,return_timestamps='word',)["chunks"]
    print(transcribed_arr)

    words = [item['text'] for item in transcribed_arr]
    timestamps = [(math.floor(item['timestamp'][0]), math.ceil(item['timestamp'][1])) for item in transcribed_arr]
    print(words)
    print(timestamps)
    return words, timestamps
```

Hình 3.36: transcribe.py

- ***Thực hiện matching:***

```python
def matching_process(text, text_transcribed, time_transcribed):      You, last week • update
    sync_audio_time_text = [(0, 0)] * len(text)
    length_transcribed = len(text_transcribed)
    length_text = count_words_in_array(text)

    forecast_index_start = [0] * len(text)
    num_word_prev = 0
    for i in range(0, len(text)):
        forecast_index_start[i] = math.floor(num_word_prev * length_transcribed / length_text)
        num_word_prev += len(text[i].split())

    for i in range (0, len(text)):
        # get only first 20 words of line in ebook
        line = text[i]
        line_arr = line.split(' ')
        num_word = len(line_arr)
        if num_word > 20:
            num_word = 20
            line_arr = line_arr[:20]

        position = get_start_end_transcribe(text_transcribed, forecast_index_start[i], num_word)
        compare_trans = text_transcribed[position[0]:position[1]]
        index_start_match = find_index_start_word(line_arr, compare_trans, forecast_index_start[i] - position[0]) + position[0]
        sync_audio_time_text[i] = (time_transcribed[index_start_match][0], index_start_match)
        print(i, sync_audio_time_text[i])
    return sync_audio_time_text
```

Hình 3.37: matching.py

```python
def determine_start_matching(text, text_transcribed):
    index_start_check = next((i for i in range(0, len(text)) if len(text[i].split(" ")) >= 20), None)
    matching_text = text[index_start_check:]

    for i in range(0, len(matching_text)):
        # get only first 20 words of line in ebook
        line = matching_text[i]
        line_arr = line.split(' ')
        num_word = len(line_arr)
        if num_word > 20:
            num_word = 20
            line_arr = line_arr[:20]

        compare_trans = text_transcribed[:(200 * math.ceil(i+1/5))]
        index_start_match = find_index_start_word(line_arr, compare_trans)
        if index_start_match != -1:
            return index_start_check + i, index_start_match
    return 0, 0
```

Hình 3.38: matching.py (2)

- ***Hàm đánh giá từ phù hợp và chọn làm vị trí bắt đầu khớp:***

```python
def find_all_word_indices(sentence, word):
    indices = []
    words = sentence
    for i, w in enumerate(words):      You, 2 weeks ago • update
        if w == word:
            indices.append(i)
    return indices

def get_start_end_transcribe(transcribed, index, len_line, search_range_character = 40):···

def find_index_start_word(line_arr, compare_trans, forecast_index_start = -1):
    h_words = [0] * len(compare_trans)
    for i in range(0, len(line_arr)):
        indices = find_all_word_indices(compare_trans, line_arr[i])
        for idx in indices:
            h_words[idx - i] += 1

    index_start_word = forecast_index_start
    max_eval = h_words[forecast_index_start]
    for i in range(0, len(h_words)):
        if h_words[i] > max_eval:
            max_eval = h_words[i]
            index_start_word = i

    if max_eval < len(line_arr) * 4 / 10:
        return forecast_index_start
    return index_start_word
```

Hình 3.39: matching.py (3)

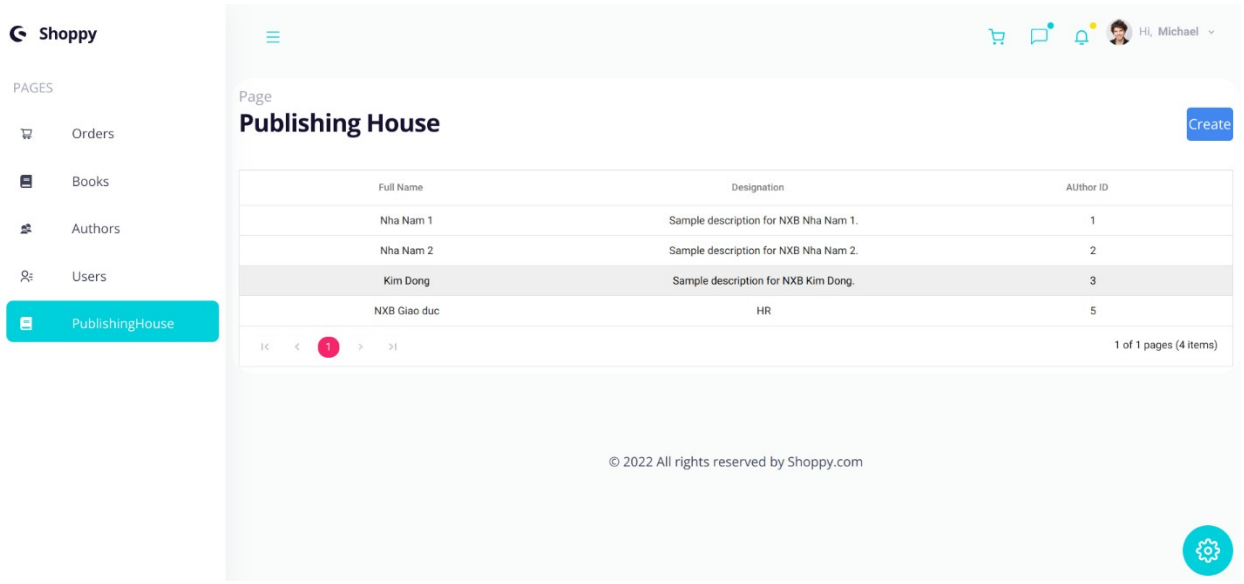
3.3. Chức năng Quản lý các nhà xuất bản

3.3.1. Mô tả chức năng

Chức năng quản lý tác giả có các thành phần xử lý cơ bản tương tự với chức năng quản lý sách.

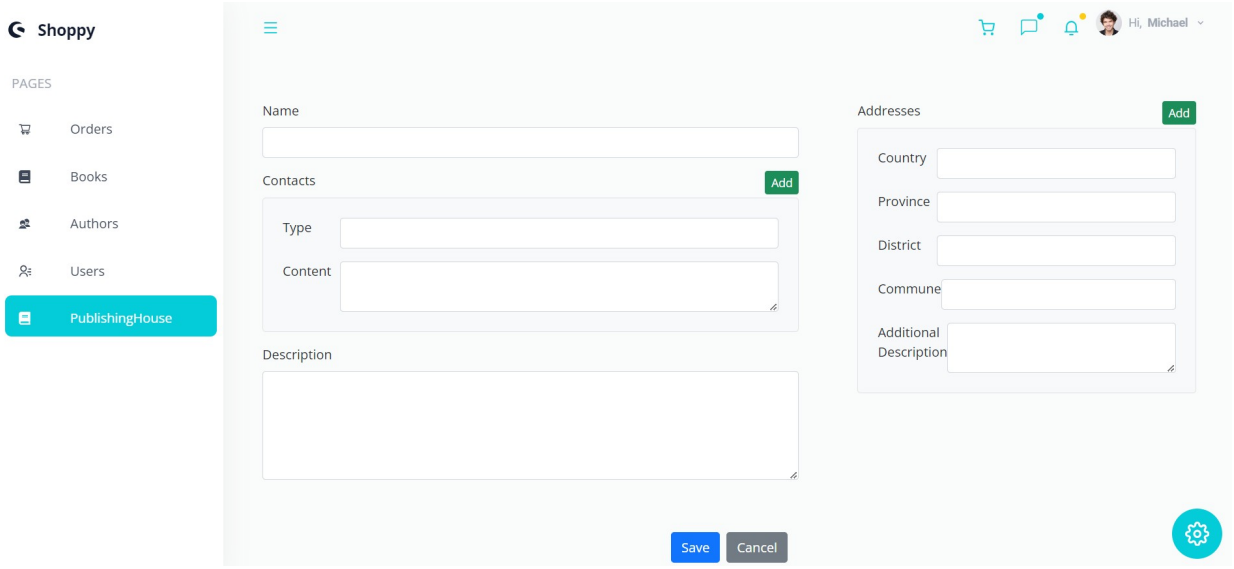
3.3.2. Các giao diện quản lý các nhà xuất bản

3.3.2.1. Giao diện trang quản lý các nhà xuất bản



Hình 3.40: Giao diện màn hình quản lý các nhà xuất bản

3.3.2.2. Giao diện trang thêm nhà xuất bản mới



Hình 3.41: Giao diện màn hình thêm nhà xuất bản

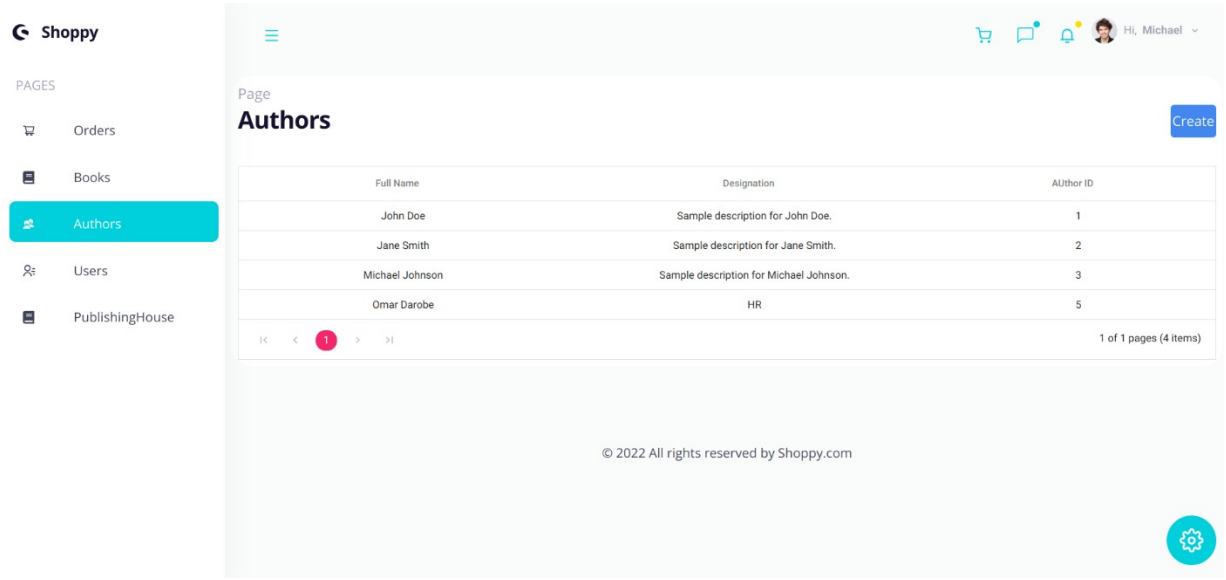
3.4. Chức năng Quản lý các tác giả

3.4.1. Mô tả chức năng

Chức năng quản lý tác giả có các thành phần xử lý cơ bản tương tự với chức năng quản lý sách.

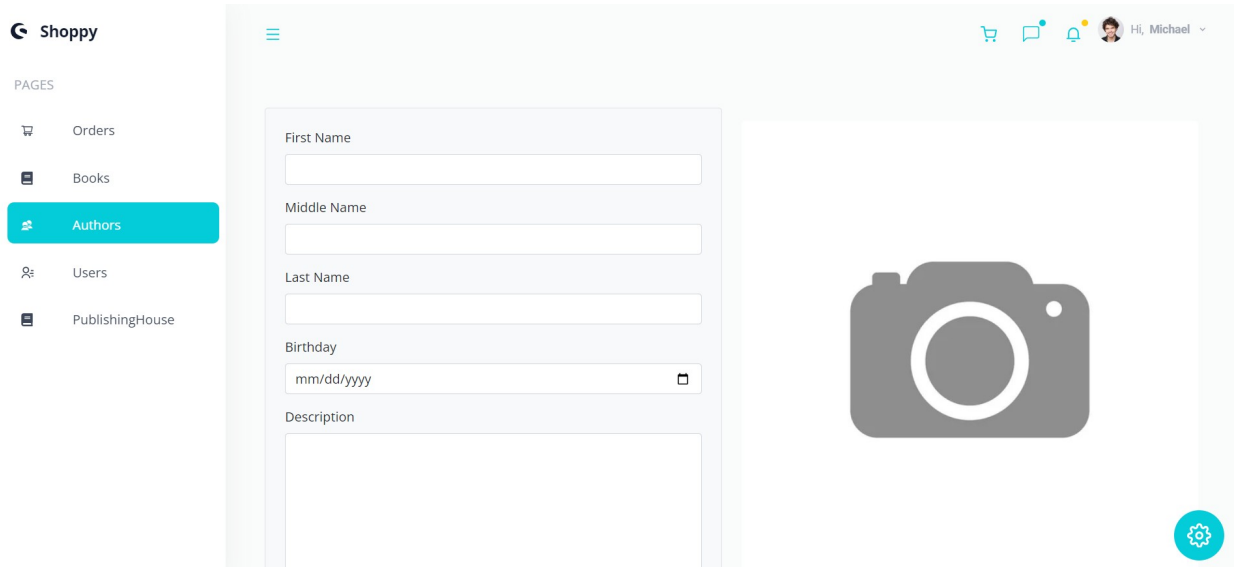
3.4.2. Các giao diện quản lý các tác giả

3.4.2.1. Giao diện trang quản lý các tác giả



Hình 3.42: Giao diện màn hình quản lý các tác giả

3.4.2.2. Giao diện trang thêm tác giả mới



Hình 3.43: Giao diện màn hình thêm tác giả mới

3.5. Cài đặt môi trường

3.5.1. Môi trường phát triển

- Ngôn ngữ lập trình: javascript/typescript, python.
- Framework: ReactJS/NextJS, NestJS.

- Cơ sở dữ liệu: Postgres.
- Giao tiếp nội bộ giữa service: Kafka (Kraft mode).

3.5.2. Phần mềm sử dụng

- Visual Studio Code
- PgAdmin 4

3.5.3. Xây dựng hệ thống cho chức năng đồng bộ

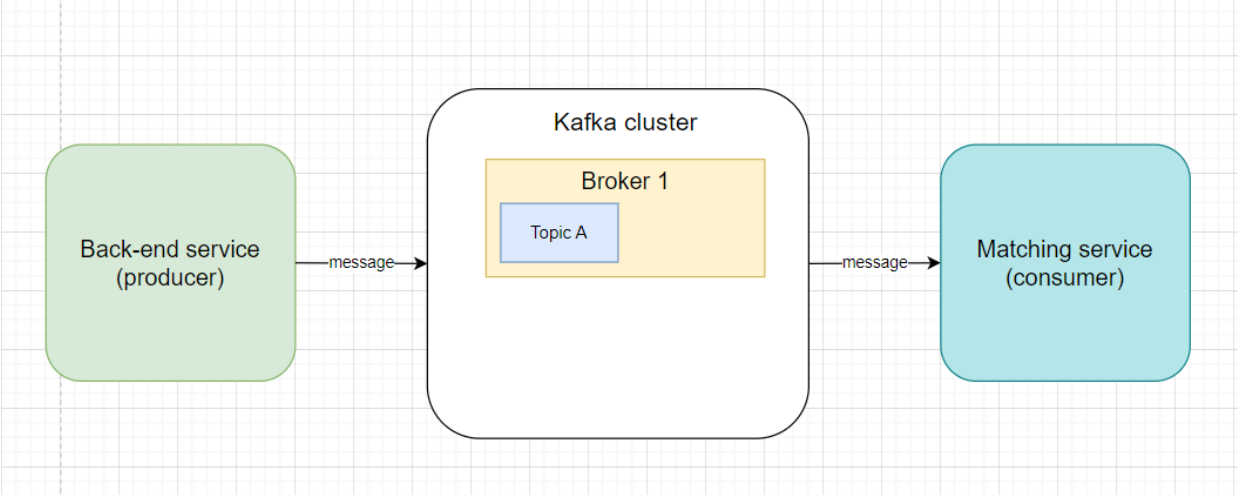
Tham gia vào quá trình đồng bộ có sự tham gia của 3 thành phần chính, đó là back-end service (NestJS), Kafka server (Kraft mode) và matching service (Python). Ngoài ra, database (Postgres) cũng là thành phần không thể thiếu, để lưu trữ dữ liệu đã được trích xuất.

Vì các Api được phục vụ bởi back-end service có chức năng độc lập, không trực tiếp liên quan đến quá trình trích xuất dữ liệu đồng bộ 2 phiên bản sách, do đó 2 thành phần này được tách biệt, tách thành những service riêng.

Ngôn ngữ được sử dụng ở matching service nên là Python, bởi vì đã có rất nhiều thư viện hỗ trợ những tác vụ liên quan đến AI và chạy model nói chung, và với chức năng đồng bộ nói riêng là ASR (Automatic Speech Recognition) model. Matching service sẽ sử dụng một asr model tiếng Việt đã được public để thực hiện transcribe, đó là nguyenvulebinh/wav2vec2-base-vietnamese-250h.

Vì back-end service và matching service đã được tách riêng, nên sẽ cần có một cách thức để giao tiếp nội bộ giữa 2 services này, đó chính là Kafka, là một hệ thống message pub/sub phân tán. Bên public gửi dữ liệu (producer) là back-end service và bên subcribe nhận dữ liệu (consumer) là matching service.

Sơ đồ tổng quan hệ thống:



Hình 3.44: Sơ đồ tổng quan hệ thống

KẾT LUẬN

1. Những vấn đề đã giải quyết trong đồ án

- Hệ thống đã đáp ứng được các chức năng cơ bản của một hệ thống quản lý sách, bao gồm nhà xuất bản và tác giả viết sách.
- Khả năng đồng bộ, thay đổi linh hoạt giữa nghe sách nói và đọc sách điện tử, mang lại sự tiện dụng, tăng sự trải nghiệm tốt của người dùng đối với ứng dụng.
- Xây dựng các thành phần hệ thống độc lập, mỗi hệ thống xử lý những tác vụ riêng, tăng khả năng mở rộng, tìm kiếm lỗi và bảo trì của hệ thống.

2. Những hạn chế tồn tại trong đồ án

- Cần cải thiện giao diện sử dụng màn hình quản lý cho người quản lý tốt hơn.
- Hiện tại chức năng đồng bộ chỉ hỗ trợ với các đầu sách có ngôn ngữ là tiếng Việt.

3. Hướng phát triển tiếp theo

- Tối ưu ASR model, tăng khả năng dịch âm thanh sang văn bản chính xác hơn.
- Tính năng đồng bộ sách nói và sách điện tử sử dụng được cho đa ngôn ngữ.
- Có thể cho người dùng đăng tải sách nói lên hệ thống, chia sẻ bản nói của chính họ.
- Thêm các tính năng về đánh giá sách, phân loại sách.

LỜI KẾT

Do hạn chế về mặt thời gian và trình độ nên chắc chắn đồ án không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của các thầy cô để đồ án được hoàn thiện hơn.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] Robin Wieruch. “The Road to React: Your journey to master plain yet pragmatic React.js”.

[2] Todd Palino, Gwen Shapira, Neha Narkhede. “Kafka: The Definitive Guide” (2017).

[3] Daniel Jurafsky, James H. Martin. “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition” (2018).

[4] "Building a REST API with NestJS and Prisma," [Online]. Available: https://www.prisma.io/blog/nestjs-prisma-rest-api-7D056s1BmOL0.

[5] "Vietnamese end-to-end speech recognition using wav2vec 2.0," [Online]. Available: https://huggingface.co/nguyenvulebinh/wav2vec2-base-vietnamese-250h.

[6] "Automatic speech recognition with a pipeline," [Online]. Available: https://huggingface.co/learn/audio-course/chapter2/asr_pipeline.

[7] "APACHE KAFKA QUICKSTART," [Online]. Available: https://kafka.apache.org/quickstart.

[8] "Công nghệ tự động nhận dạng giọng nói," [Online]. Available: https://vinbigdata.com/cong-nghe-giong-noi/cong-nghe-tu-dong-nhan-dang-giong-noi-tong-hop-cac-thuat-toan-va-giai-phap.html