ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

**BÀI TẬP LỚN**

MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

**Đề tài: Hệ thống quản lý thư viện**

Giáo viên hướng dẫn: PGS TS. **Lê Đức Hậu**

Nhóm sinh viên thực hiện: Phạm Trần Nguyên Phú – 20226120

Đào Ngọc Quang – 20226122

Đỗ Quang Bắc Kỳ – 20226111

Nguyễn Minh Tuấn – 20226095

***Hà Nội, tháng 1 năm 2025***

LỜI NÓI ĐẦU

Trong thời đại thông tin phát triển mạnh mẽ như hiện nay, thư viện không chỉ đơn thuần là nơi lưu trữ sách báo mà còn đóng vai trò là một trung tâm tri thức, kết nối người đọc với kho tàng kiến thức vô tận. Để đáp ứng nhu cầu ngày càng cao của người sử dụng, việc xây dựng một hệ thống quản lý thư viện hiện đại, hiệu quả là điều vô cùng cần thiết.

Với mục tiêu ứng dụng những kiến thức đã học được trong môn Lập trình hướng đối tượng vào thực tiễn, nhóm chúng em đã lựa chọn đề tài "Hệ thống quản lý thư viện" để thực hiện bài tập lớn. Qua quá trình nghiên cứu và phát triển, nhóm đã xây dựng một hệ thống quản lý thư viện dựa trên nền tảng lập trình hướng đối tượng, nhằm tự động hóa các quy trình quản lý, nâng cao hiệu quả làm việc của nhân viên thư viện và mang đến trải nghiệm tốt hơn cho người đọc.

Bài báo cáo này trình bày chi tiết quá trình thiết kế, phát triển và đánh giá hệ thống. Phần đầu tiên sẽ giới thiệu về việc khảo sát yêu cầu, đặc tả bài toán. Tiếp theo, bài báo cáo sẽ đi sâu vào quá trình thiết kế chương trình và lựa chọn công nghệ. Phần cuối cùng sẽ đưa ra những kết luận và gợi ý cho các nghiên cứu tiếp theo.

Nhóm xin gửi lời cảm ơn chân thành đến thầy giáo – PGS.TS. Lê Đức Hậu đã hướng dẫn và tạo điều kiện cho nhóm hoàn thành bài tập lớn này. Nhóm cũng xin cảm ơn các bạn trong nhóm đã cùng nhau nỗ lực để hoàn thành dự án.

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| **Họ và tên** | **Email** | **Công việc thực hiện** | **Đánh giá** |
| Phạm Trần Nguyên Phú | phu.ptn226120@sis.hust.edu.vn | Xây dựng backend, thiết kế cơ sở dữ liệu, thiết kế biểu đồ gói, thiết kế biểu đò lớp, làm báo cáo | Hoàn thành |
| Đào Ngọc Quang | quang.dn226122@sis.hust.edu.vn | Thiết kế biểu đồ Use Case, thiết kế cơ sở dữ liệu và làm báo cáo, làm slide | Hoàn thành |
| Đỗ Quang Bắc Kỳ | ky.dqb226111@sis.hust.edu.vn | Xây dựng backend, thiết kế biểu đồ lớp | Hoàn thành |
| Nguyễn Minh Tuấn | tuan.nm226095@sis.hust.edu.vn | Xây dựng frontend, tham gia làm backend | Hoàn thành |

CHƯƠNG 1. KHẢO SÁT, ĐẶC TẢ YÊU CẦU BÀI TOÁN

## Mô tả yêu cầu bài toán

Hệ thống quản lý thư viện nhằm hỗ trợ việc quản lý sách, người dùng (thành viên), và các hoạt động mượn trả sách một cách hiệu quả, nhanh chóng, và chính xác. Hệ thống giúp giảm thiểu công việc thủ công, tối ưu hóa thời gian và nâng cao trải nghiệm của người dùng.

**1.1 Các chức năng chính**

**1.1.1. Quản lý sách**

* Thêm mới sách vào thư viện (bao gồm thông tin: mã sách, tên sách, tác giả, nhà xuất bản, số lượng bản sao, ...).
* Cập nhật thông tin sách.
* Xóa sách khỏi hệ thống.
* Tìm kiếm sách theo nhiều tiêu chí: Tên sách, tác giả, hoặc từ khóa.

**1.1.2. Quản lý thành viên**

* Đăng ký thành viên mới (bao gồm thông tin: mã thành viên, tên, địa chỉ, số điện thoại, email, ...).
* Cập nhật thông tin thành viên.
* Hủy tư cách thành viên.

**1.1.3. Quản lý mượn trả sách**

* Ghi nhận thông tin mượn sách (ngày mượn, ngày phải trả, người mượn, sách được mượn).
* Ghi nhận thông tin trả sách (ngày trả, trạng thái sách khi trả).
* Hỗ trợ gia hạn thời gian mượn (nếu được phép).
* Tính toán phí phạt trả trễ (nếu có).

**1.1.4 Quản lý nhân viên**

* Quản lý tài khoản nhân viên (đăng ký, phân quyền, xóa tài khoản).

**1.1.5 Báo cáo và thống kê**

* Thống kê số lượng sách theo thể loại, tình trạng (còn sẵn/mượn hết).
* Theo dõi danh sách thành viên vi phạm (trả sách trễ, không trả sách).

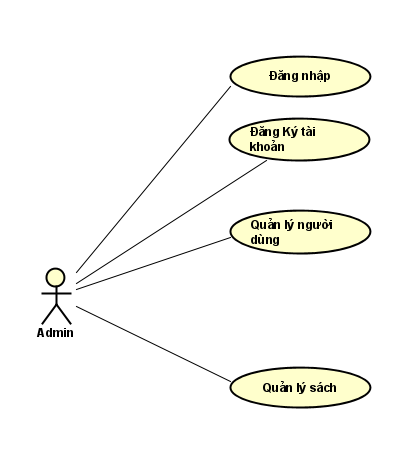
**1.2 Phân quyền người dùng**

* **Đối với Person:**
* Person là một lớp trừu tượng (abstract class) dùng để biểu diễn các đối tượng tham gia trong hệ thống, gồm các đặc điểm và hành vi chung cho tất cả các loại người dùng (**Admin**, **Employee**, và **Customer**).
* Các thông tin chung có thể bao gồm:
  + ID người dùng
  + Tên đầy đủ
  + Địa chỉ email
  + Số điện thoại
* **Đối với Admin:**
* Quản lý tài khoản
* Quản lý sách
* **Đối với Employee (Nhân viên):**
* Tiếp nhận sách
* Tra cứu đơn
* Quản lý đơn mượn
* **Đối với Customer (Khách hàng):**
* Tìm kiếm sách
* Mượn sách

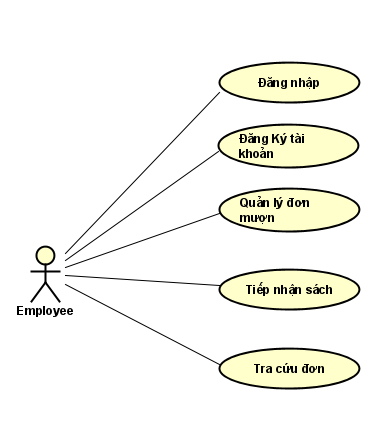
## Biểu đồ use case

### **Biểu đồ use case tổng quan**

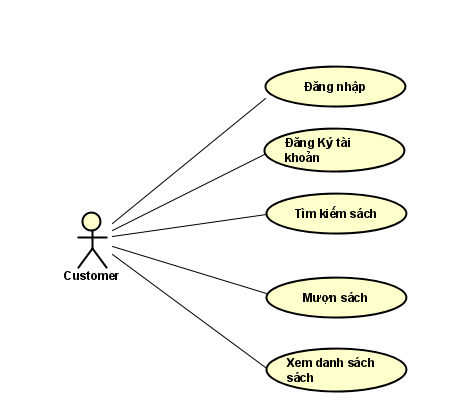
**Đối với Admin:**



**Đối với Employee**

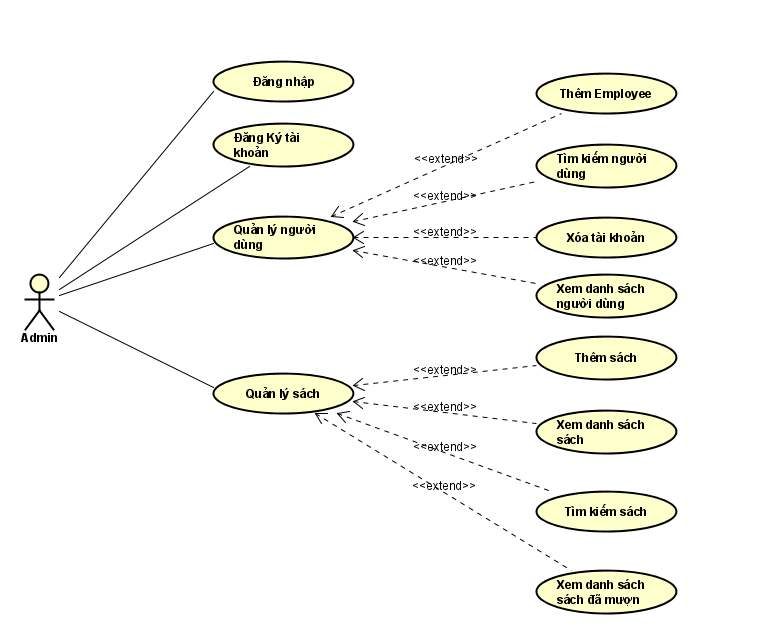


**Đối với Customer**

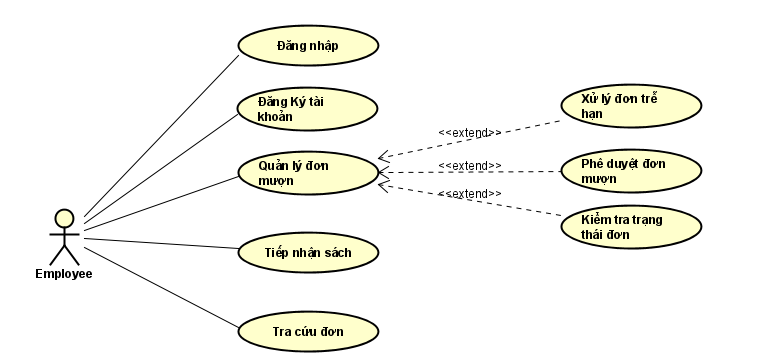


### **Biểu đồ use case phân rã mức 2**

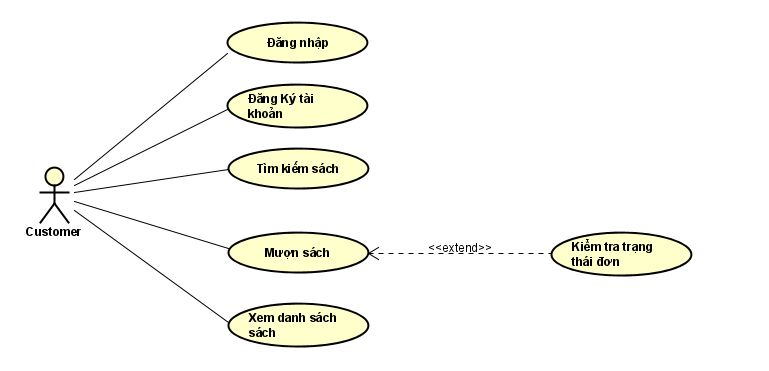
**Đối với Admin:**



**Đối với Employee**



**Đối với Customer**



## **Đặc tả chức năng**

**Danh sách Actor**

|  |  |
| --- | --- |
| **Tên Actor** | **Ý Nghĩa** |
| **Person** | Người dùng chung của hệ thống, bao gồm các thông tin cơ bản và hành động phổ quát như đăng nhập, quản lý thông tin cá nhân. |
| **Admin** | Người quản lý hệ thống, thực hiện các chức năng quản lý tài khoản, quản lý sách và giám sát hoạt động của hệ thống. |
| **Employee** | Nhân viên hỗ trợ, chịu trách nhiệm tiếp nhận sách, tra cứu đơn, và xử lý các tác vụ vận hành thường ngày. |
| **Customer** | Người dùng thư viện, tìm kiếm thông tin sách, mượn sách và theo dõi trạng thái đơn mượn. |

**1.2.3.2. Danh sách Use Case**

|  |  |
| --- | --- |
| **Tên Use Case** | **Ý Nghĩa** |
| **Đăng nhập** | Cho phép tất cả các Actor truy cập hệ thống bằng tài khoản của họ. |
| **Quản lý tài khoản** | Admin thực hiện các thao tác quản lý tài khoản: tạo, đọc, cập nhật, và xóa. |
| **Quản lý sách** | Admin cập nhật, thêm mới, hoặc xóa thông tin sách trong thư viện. |
| **Tiếp nhận sách** | Employee ghi nhận sách mới hoặc cập nhật số lượng sách hiện có. |
| **Tra cứu đơn** | Employee tra cứu thông tin các đơn mượn sách của người dùng. |
| **Tìm kiếm sách** | Customer tìm kiếm sách có trong thư viện theo các tiêu chí như tên sách, tác giả, hoặc thể loại. |
| **Mượn sách** | Customer thực hiện việc chọn sách và gửi yêu cầu mượn thông qua form mượn sách. |
| **Quản lý thông tin cá nhân** | Person (bao gồm tất cả các Actor) cập nhật thông tin cá nhân, như email, mật khẩu, và số điện thoại. |
| **Xem thông báo** | Person xem các thông báo liên quan từ hệ thống, như trạng thái đơn mượn hoặc thông báo hệ thống. |

**1.2.3.3. Đặc tả Use Case**

**1. Use Case: Đăng nhập**

* **Tên Use Case:** Đăng nhập
* **Tên Tác Nhân:** Person

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị giao diện đăng nhập. |
| 2 | Person | Nhập thông tin đăng nhập: tên đăng nhập và mật khẩu. |
| 3 | System | Xác thực thông tin, cho phép truy cập hoặc thông báo lỗi nếu thất bại. |

**2. Use Case: Quản lý tài khoản**

* **Tên Use Case:** Quản lý tài khoản
* **Tên Tác Nhân:** Admin

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị danh sách tài khoản. |
| 2 | Admin | Chọn chức năng: tạo, đọc, cập nhật, hoặc xóa tài khoản. |
| 3 | Admin | Nhập thông tin tài khoản (tên, email, vai trò, mật khẩu, v.v.). |
| 4 | System | Xác nhận thành công hoặc thông báo lỗi. |

**3. Use Case: Quản lý sách**

* **Tên Use Case:** Quản lý sách
* **Tên Tác Nhân:** Admin

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị giao diện quản lý sách. |
| 2 | Admin | Chọn chức năng: thêm sách, xóa sách, hoặc cập nhật thông tin sách. |
| 3 | Admin | Nhập/chỉnh sửa thông tin sách (tên, tác giả, thể loại, số lượng...). |
| 4 | System | Lưu thông tin sách, hiển thị danh sách sách sau khi cập nhật. |

**4. Use Case: Tiếp nhận sách**

* **Tên Use Case:** Tiếp nhận sách
* **Tên Tác Nhân:** Employee

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị danh sách sách đã có trong hệ thống. |
| 2 | Employee | Xác nhận sách mới được tiếp nhận hoặc cập nhật số lượng sách. |
| 3 | System | Cập nhật dữ liệu sách và thông báo thành công. |

**5. Use Case: Tìm kiếm sách**

* **Tên Use Case:** Tìm kiếm sách
* **Tên Tác Nhân:** Customer

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị giao diện tìm kiếm sách. |
| 2 | Customer | Nhập từ khóa tìm kiếm: tên sách, tác giả, thể loại, hoặc mã ISBN. |
| 3 | System | Hiển thị danh sách sách phù hợp tiêu chí tìm kiếm. |

**6. Use Case: Mượn sách**

* **Tên Use Case:** Mượn sách
* **Tên Tác Nhân:** Customer

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị danh sách sách có sẵn để mượn. |
| 2 | Customer | Chọn sách muốn mượn và nhập thông tin đơn mượn. |
| 3 | System | Tiếp nhận và xử lý đơn mượn, thông báo kết quả cho khách. |

**7. Use Case: Quản lý thông tin cá nhân**

* **Tên Use Case:** Quản lý thông tin cá nhân
* **Tên Tác Nhân:** Person

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị giao diện quản lý thông tin cá nhân. |
| 2 | Person | Cập nhật thông tin cá nhân: email, số điện thoại, hoặc mật khẩu. |
| 3 | System | Xác nhận lưu thông tin thành công. |

**8. Use Case: Tra cứu đơn**

* **Tên Use Case:** Tra cứu đơn
* **Tên Tác Nhân:** Employee

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị giao diện danh sách các đơn mượn sách hiện tại. |
| 2 | Employee | Nhập thông tin tra cứu: mã đơn, tên người mượn, hoặc tên sách. |
| 3 | System | Tìm kiếm và hiển thị chi tiết đơn mượn sách theo tiêu chí nhập vào. |
| 4 | Employee | Xem thông tin chi tiết đơn: trạng thái (đang mượn, đã trả, quá hạn). |

**Dòng chảy thay thế:**

* Nếu không tìm thấy đơn mượn khớp với tiêu chí, hệ thống thông báo: **"Không tìm thấy đơn phù hợp."**.

**9. Use Case: Xem thông báo**

* **Tên Use Case:** Xem thông báo
* **Tên Tác Nhân:** Person (áp dụng cho tất cả các Actor: **Admin**, **Employee**, và **Customer**)

|  |  |  |
| --- | --- | --- |
| **STT** | **Thực hiện** | **Hành động** |
| 1 | System | Hiển thị giao diện danh sách thông báo cá nhân. |
| 2 | Person | Chọn một thông báo để xem chi tiết (ví dụ: trạng thái đơn mượn, sách trễ hạn, cập nhật hệ thống). |
| 3 | System | Hiển thị nội dung chi tiết của thông báo. |
| 4 | Person | Đánh dấu thông báo là đã đọc (tùy chọn). |

**Dòng chảy thay thế:**

* Nếu không có thông báo, hệ thống hiển thị: **"Hiện tại không có thông báo mới."**

# CHƯƠNG 2. THIẾT KẾ CHƯƠNG TRÌNH

## Thiết kế kiến trúc

Phần mềm phát triển dựa trên kiến trúc MVC. Mẫu kiến trúc MVC là phương pháp chia nhỏ các thành phần dữ liệu, trình bày và dữ liệu nhập từ người dùng thành những thành phần riêng biệt.

Từ sơ đồ kiến trúc MVC chung, nhóm đã xây dựng và phát triển phần mềm dựa trên khung của sơ đồ kiến trúc này. Cụ thể:

### **Package model**

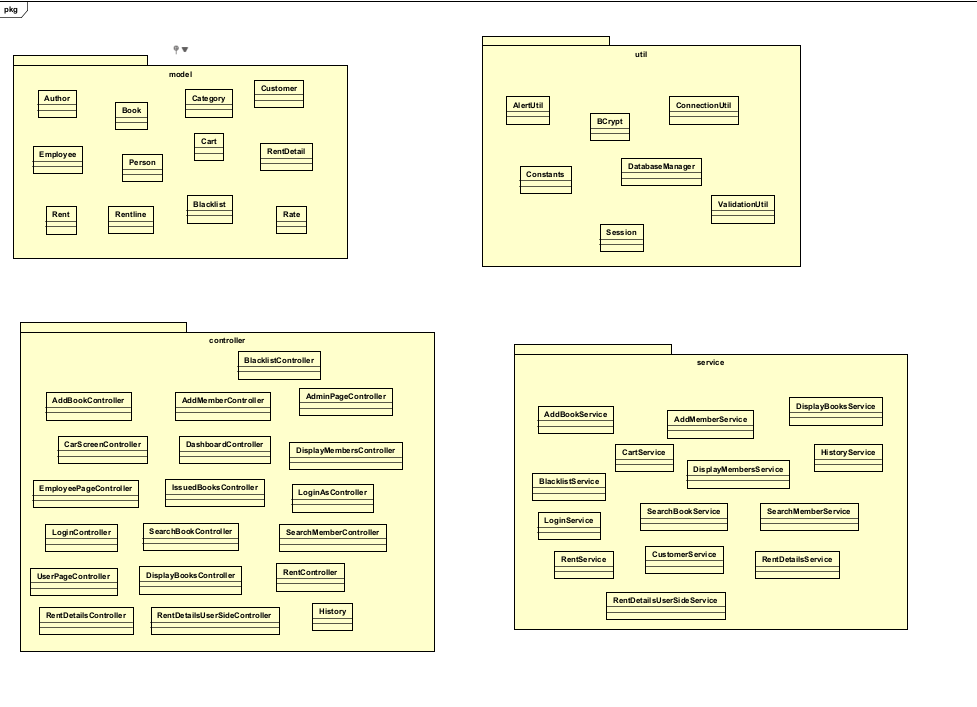
* **Mục đích:**
  + Gói này chứa các lớp (classes) mô hình hóa dữ liệu (entities) của ứng dụng.
  + Đây là nơi đại diện cho các thực thể trong hệ thống.
* **Các lớp chính:**
  + **Author, Book, Category:** Đại diện cho tác giả, sách, và thể loại sách.
  + **Customer, Employee, Person:** Đại diện cho khách hàng, nhân viên, và một thực thể tổng quát Person.
  + **Rent, Rentline:** Dùng để quản lý thông tin thuê mượn sách và các dòng chi tiết trong thuê mượn.
  + **Cart:** Đại diện cho giỏ hàng.
  + **Blacklist:** Đại diện cho danh sách đen các khách hàng vi phạm quy định.
  + **Rate:** Đại diện cho bảng giá thuê sách, có thể chứa các thông tin như mức phí thuê sách theo ngày, phí phạt trả trễ, ...

### **Package util**

* **Mục đích:**
  + Gói này chứa các tiện ích chung (utilities) hỗ trợ cho hệ thống.
* **Các lớp chính:**
  + **AlertUtil:** Quản lý các thông báo cho người dùng.
  + **ConnectionUtil:** Quản lý kết nối với cơ sở dữ liệu.
  + **BCrypt:** Cung cấp công cụ mã hóa mật khẩu.
  + **Constants:** Lưu các hằng số dùng chung trong hệ thống.
  + **DatabaseManager:** Xử lý giao tiếp với cơ sở dữ liệu.
  + **ValidationUtil:** Cung cấp các phương pháp kiểm tra và xác thực dữ liệu.
  + **Session:** Quản lý session của người dùng, lưu trữ thông tin người dùng đã đăng nhập
* **Package controller**
* **Mục đích:**
  + Gói này chứa các bộ điều khiển (controllers) quản lý logic giao tiếp giữa người dùng và hệ thống.
* **Các lớp chính:**
  + Các lớp như AddBookController, AddMemberController, LoginController, và SearchBookController quản lý các chức năng cụ thể như thêm sách, thêm thành viên, đăng nhập, tìm kiếm sách.
  + **AdminPageController, EmployeePageController, UserPageController:** Quản lý các trang dành cho quản trị viên, nhân viên, và người dùng.
  + **DashboardController:** Điều phối giao diện bảng điều khiển.
  + **CarScreenController:** Quản lý một giao diện hoặc chức năng liên quan đến xe (có thể là quản lý vận chuyển sách).
  + **IssuedBooksController:** Xử lý việc hiển thị và quản lý sách đã được mượn.
  + **UserPageController/EmployeePageController**: Điều khiển giao diện người dùng và nhân viên.
  + **History**: Hiển thị lịch sử giao dịch.

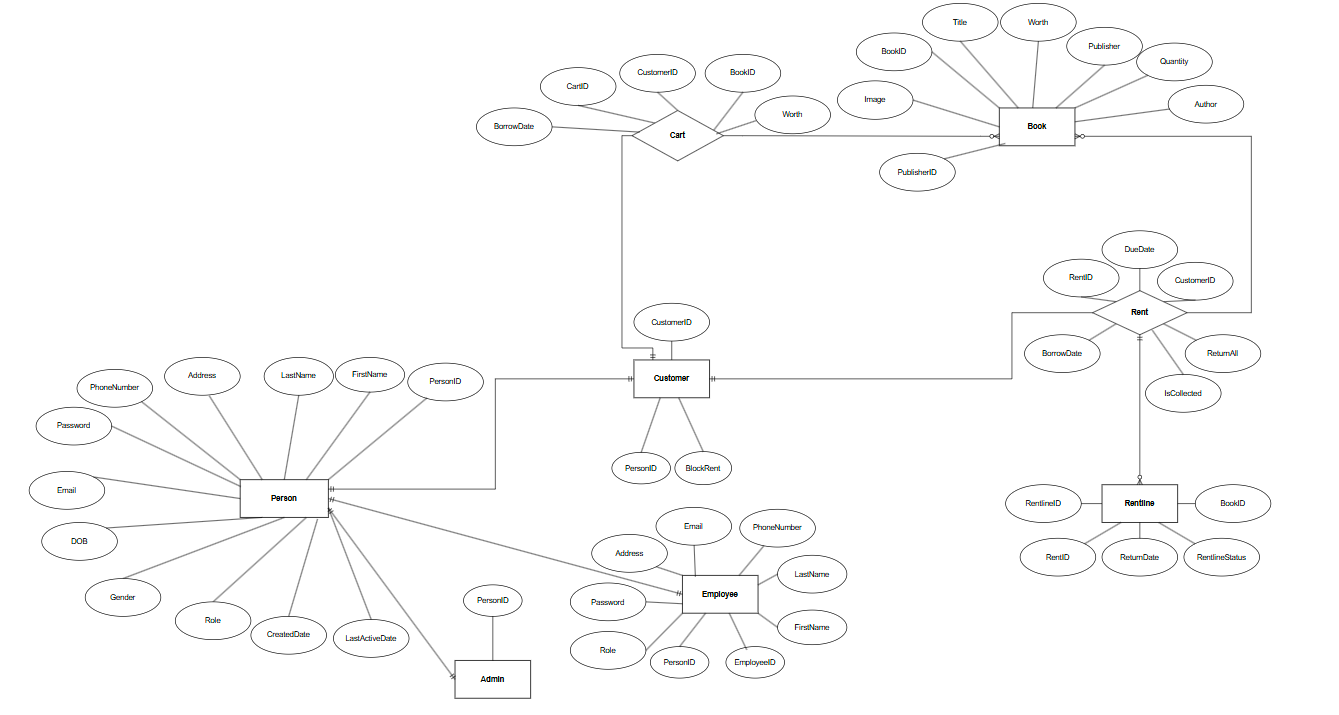
### **Package service**

* **Mục đích:**
  + Gói này chứa các lớp cung cấp các dịch vụ (business services) của hệ thống.
* **Các lớp chính:**
  + **AddBookService, AddMemberService:** Thực hiện logic để thêm sách hoặc thành viên.
  + **DisplayBookService, DisplayMembersService:** Quản lý việc hiển thị sách hoặc danh sách thành viên.
  + **LoginService:** Quản lý xác thực đăng nhập.
  + **SearchBookService, SearchMemberService:** Hỗ trợ tìm kiếm sách và thành viên.
  + **CartService:** Quản lý các chức năng của giỏ hàng.
  + **RentService/RentDetailsService/RentDetailsUserSideService**: Cung cấp dịch vụ cho các giao dịch thuê.
  + **HistoryService**: Cung cấp dữ liệu lịch sử giao dịch.
  + **CustomerService**: Quản lý thông tin khách hàng.
  + **BlacklistService**: Quản lý danh sách đen.



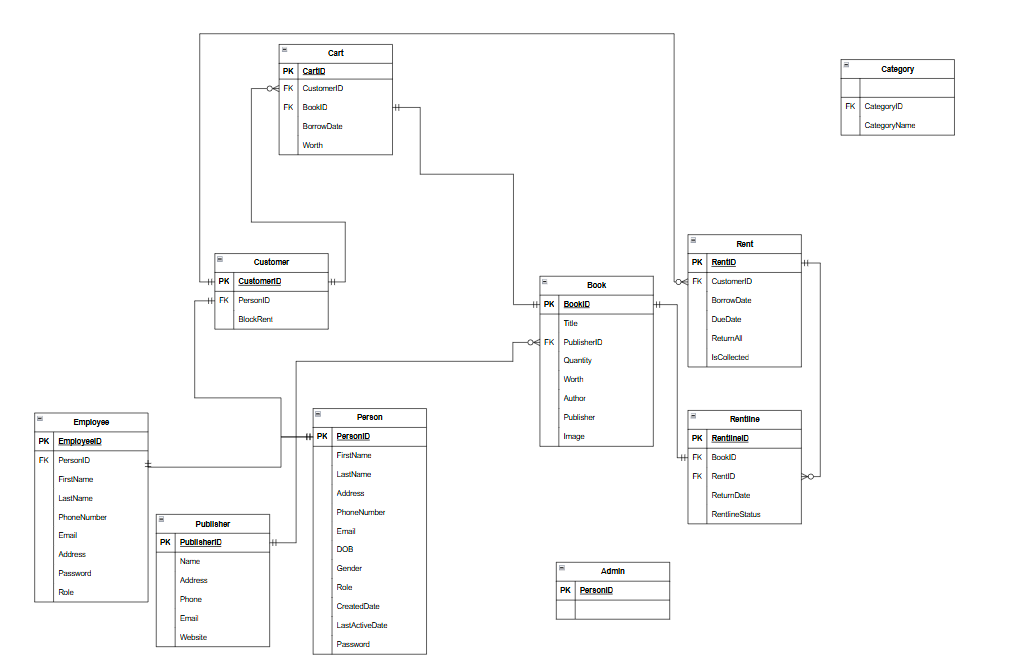
## Thiết kế cơ sở dữ liệu

## Sơ đồ thực thể liên kết



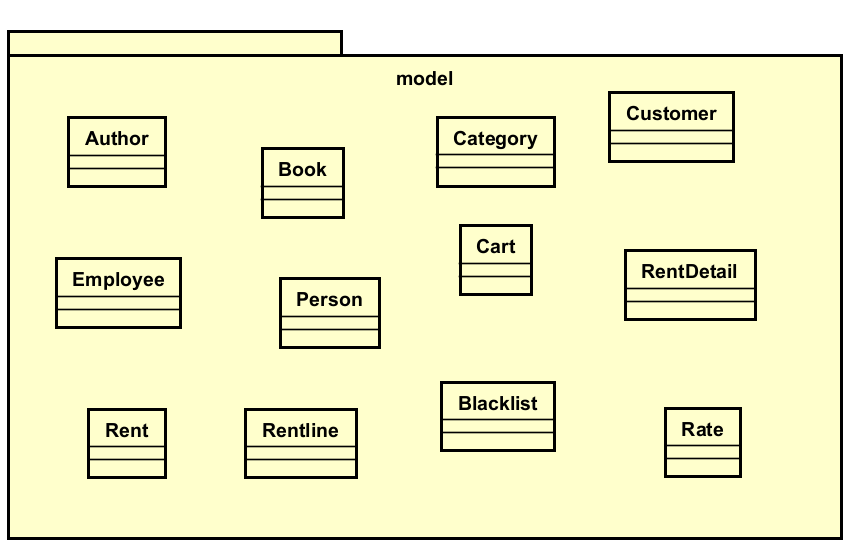
* **Person – Employee/Customer/Admin:**
* Một người có thể là khách hàng, nhân viên, hoặc quản trị viên.
* **Customer – Cart:**
* Một khách hàng có thể thêm nhiều sách vào giỏ hàng.
* **Customer – Rent:**
* Một khách hàng có thể có nhiều giao dịch thuê.
* **Rent – Rentline:**
* Một giao dịch thuê có thể chứa nhiều dòng chi tiết, mỗi dòng liên kết với một cuốn sách.
* **Book – Cart/Rentline:**
* Mỗi cuốn sách có thể nằm trong nhiều giỏ hàng hoặc nhiều dòng giao dịch thuê.

## Mô hình dữ liệu quan hệ

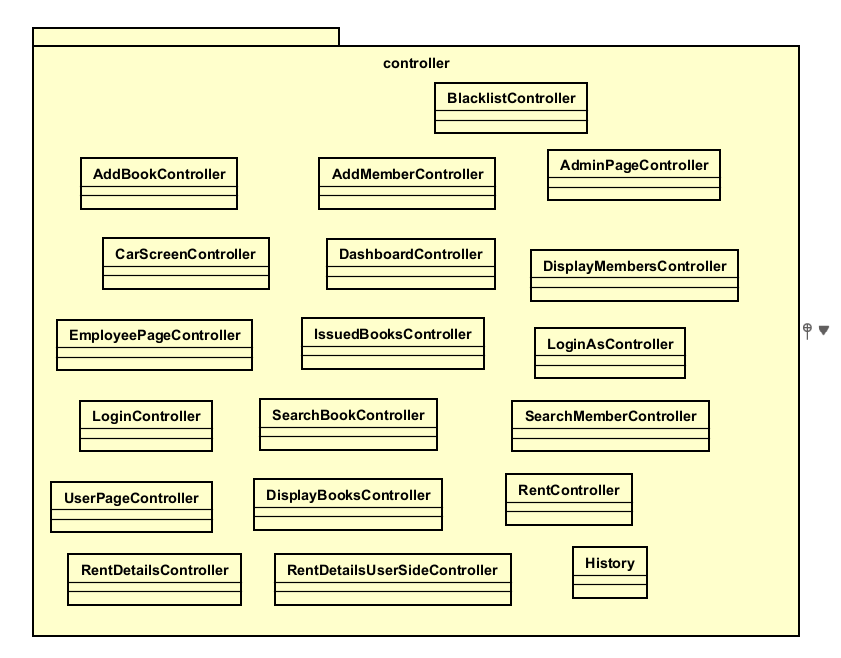


## Thiết kế chi tiết các gói

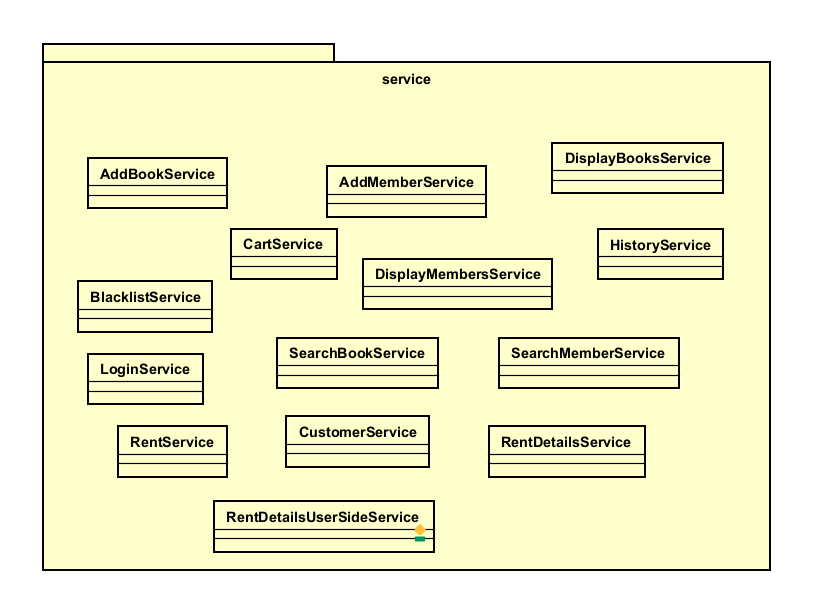
* Biểu đồ gói cho model



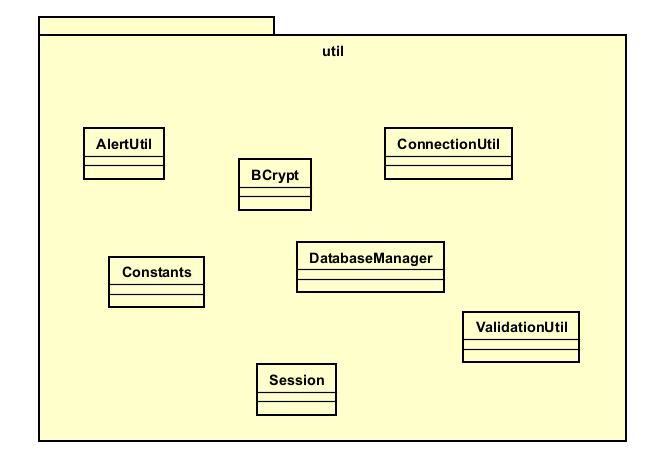
* Biểu đồ gói cho controller



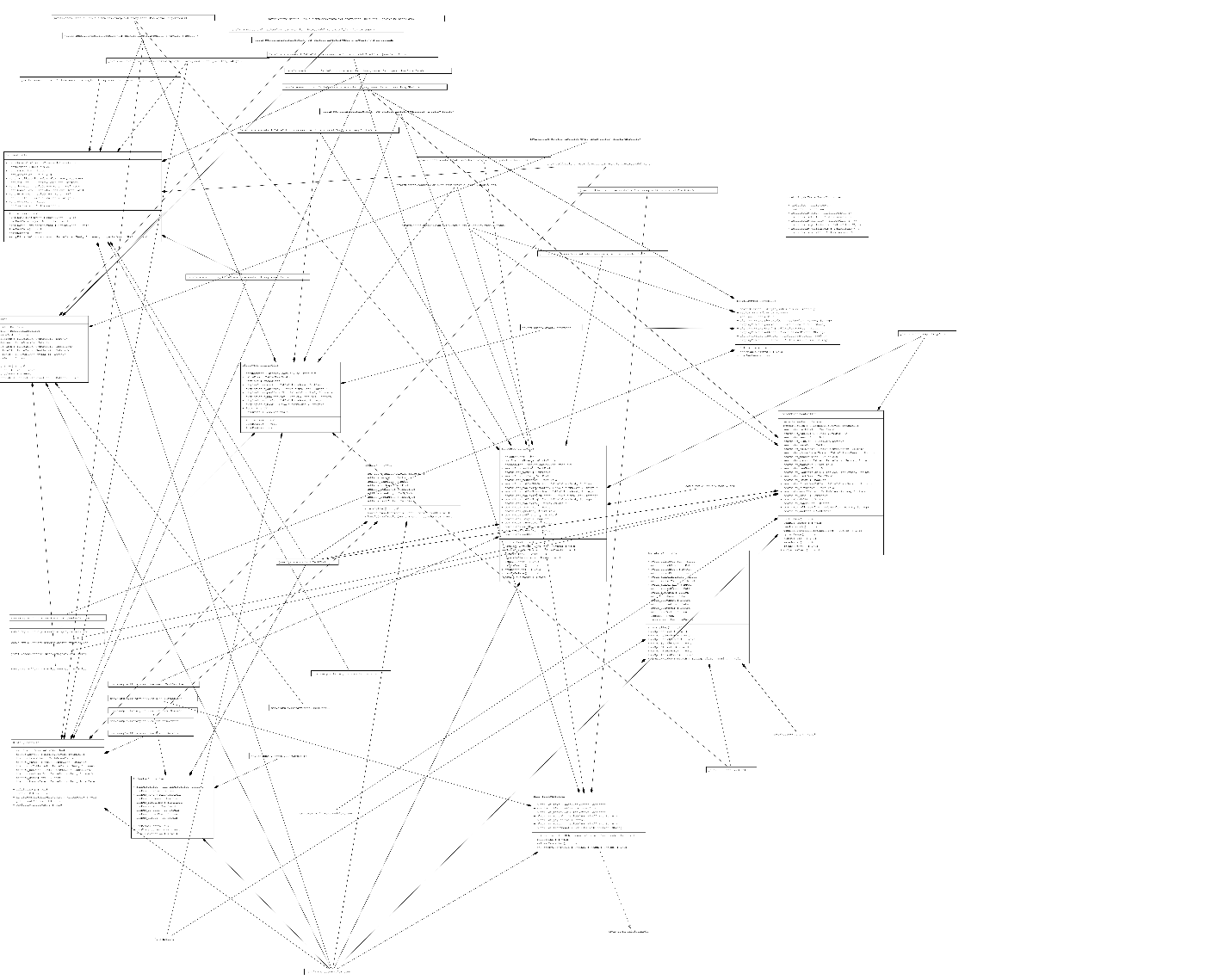
* Biểu đồ gói về service



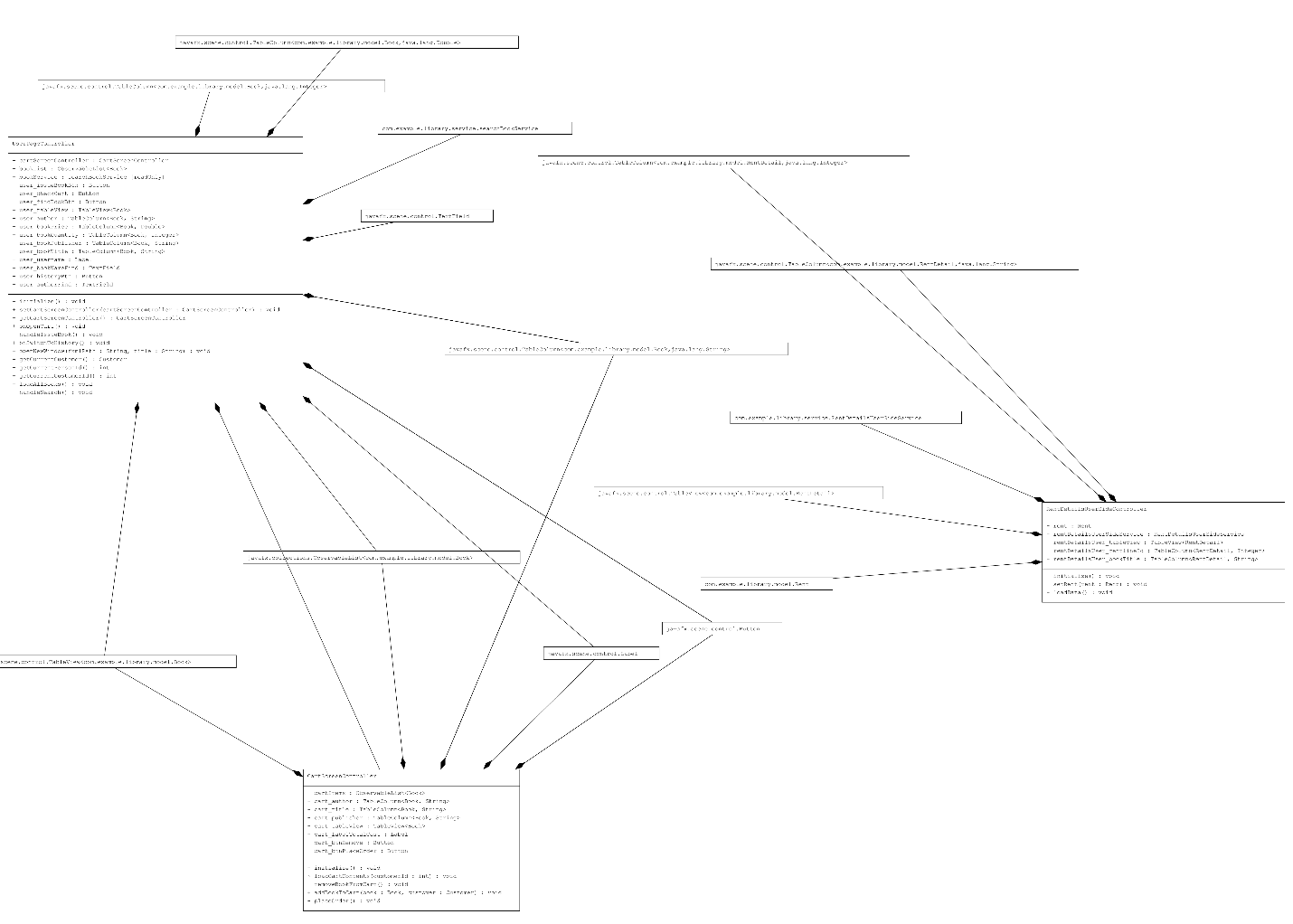
* Biểu đồ gói về util



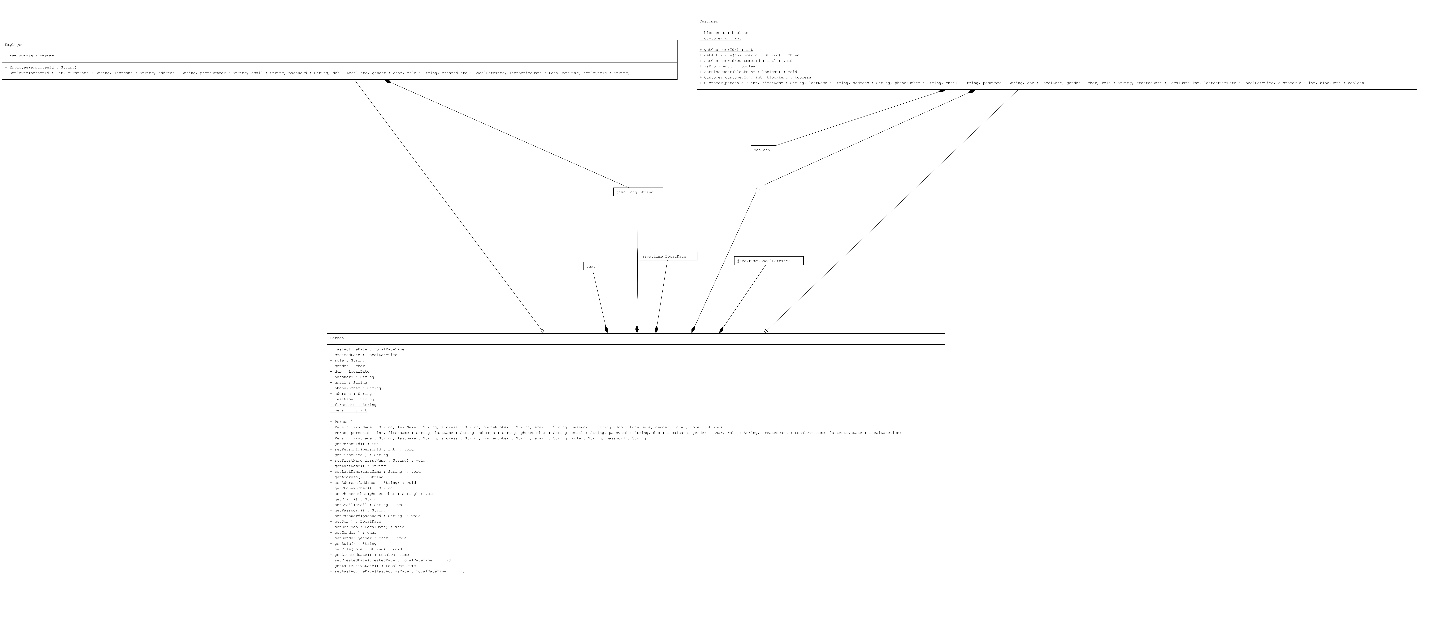
## Thiết kế chi tiết lớp- Biểu đồ lớp thể hiện các chức năng của admin



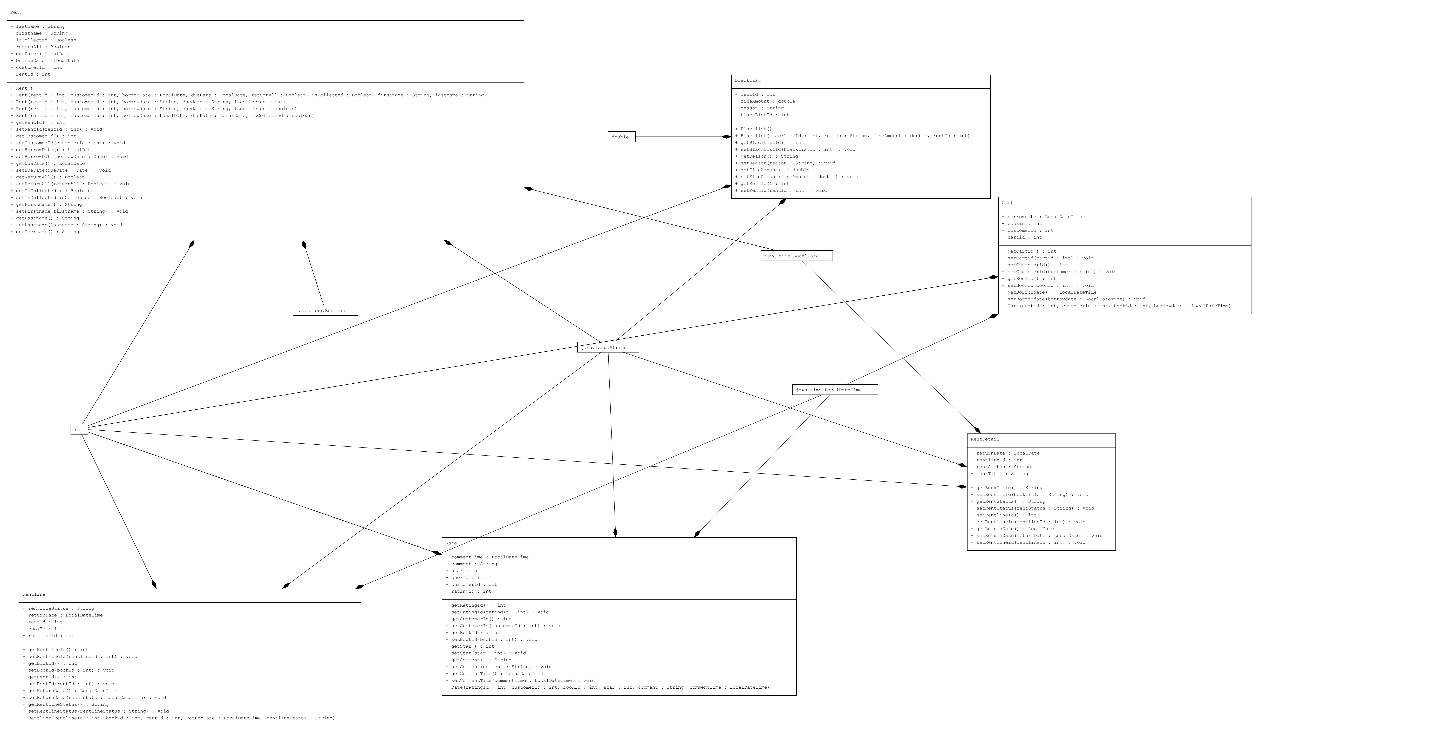
**- Biểu đồ lớp thể hiện các chức năng của user:**



* **Biểu đồ lớp thể hiện mối quan hệ giữa Person, customer và employee:**

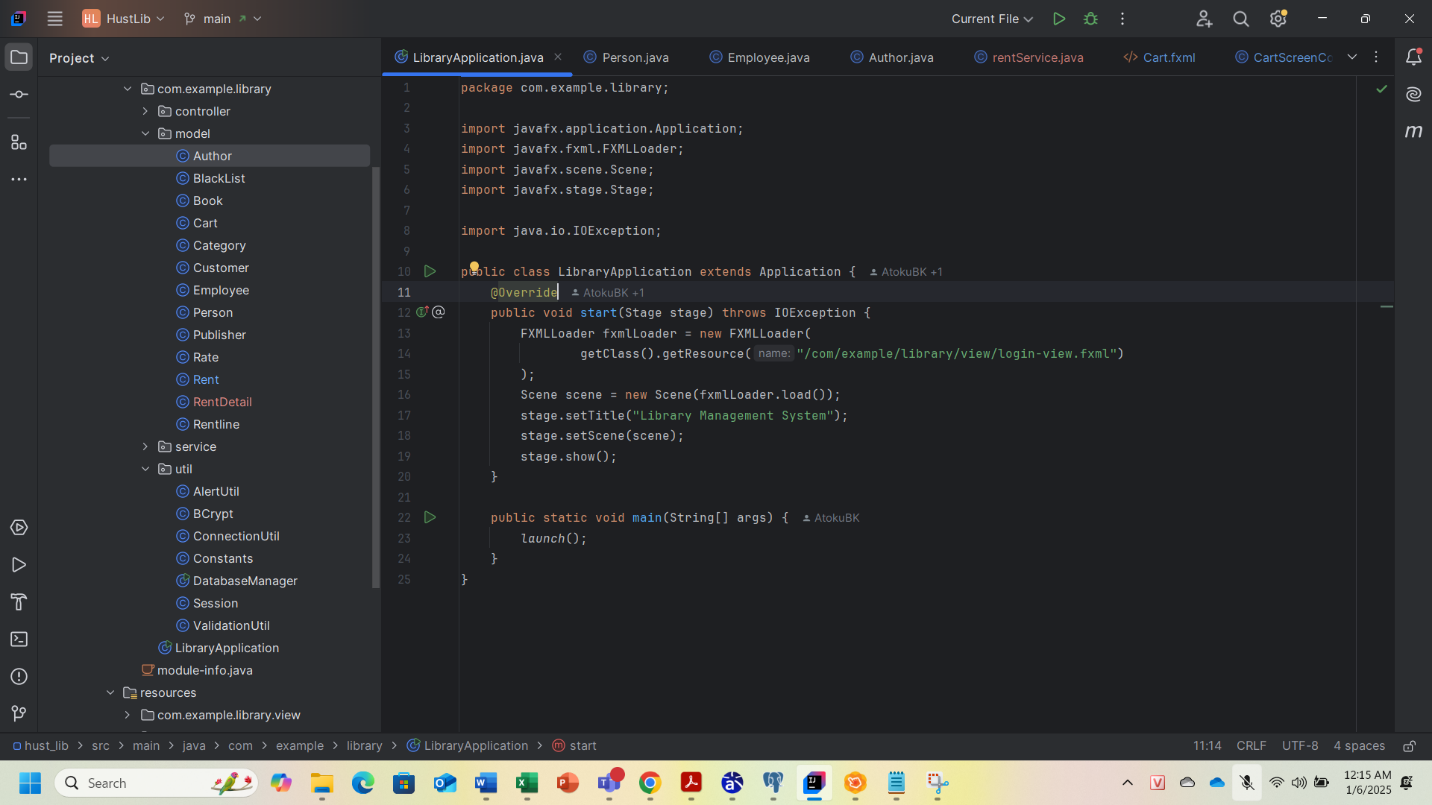


* **Biểu đồ lớp thể hiện mối quan hệ của rent:**



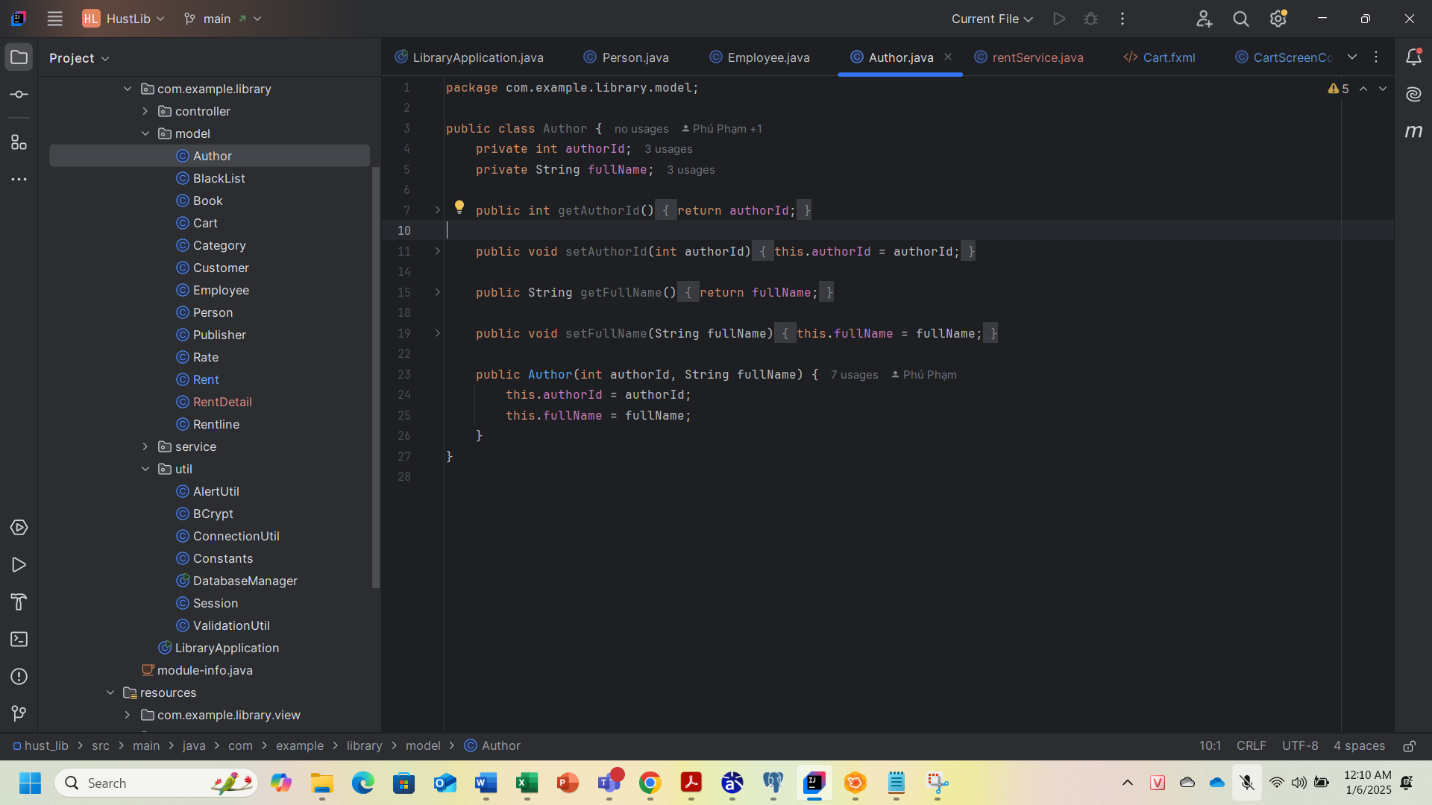
# CHƯƠNG 3. CÁC ĐẶC TÍNH HƯỚNG ĐỐI TƯỢNG

## Tính kế thừa



* Lớp LibraryApplication kế thừa từ lớp Application (thuộc thư viện JavaFX).
* Cụ thể, phương thức start() được **ghi đè** (override) từ lớp cha Application, để định nghĩa cách khởi tạo ứng dụng.

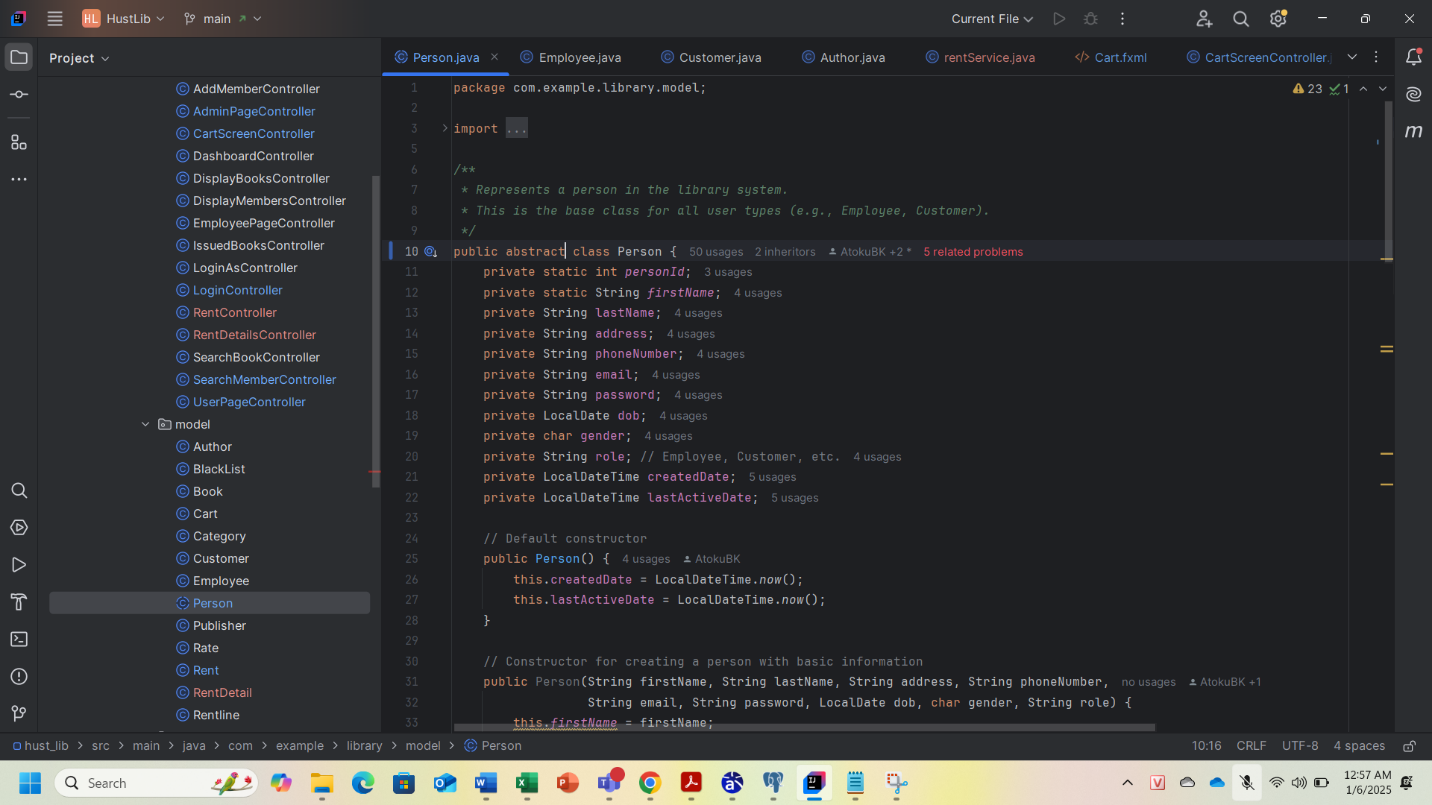
## Tính đóng gói



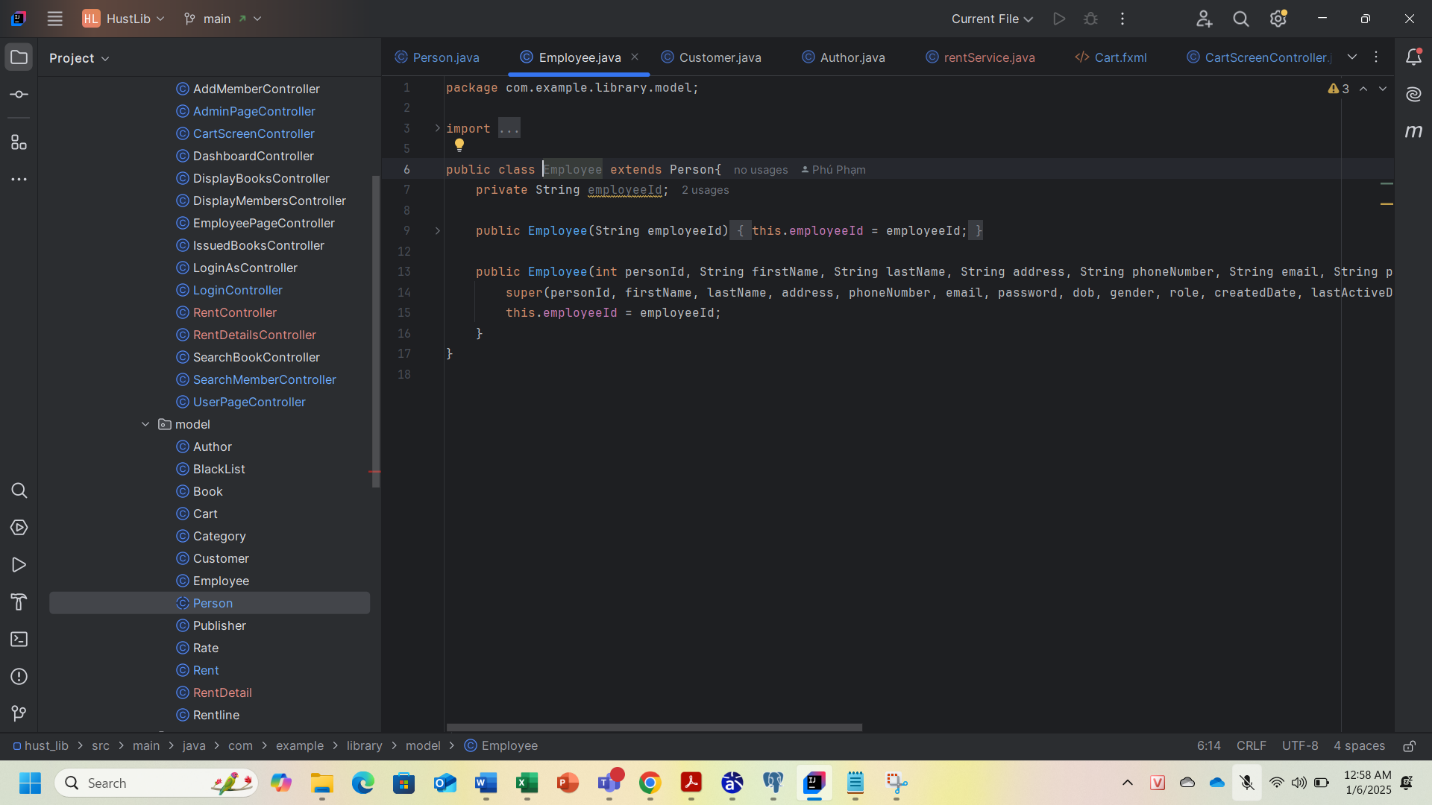
* **Thuộc tính authorId và fullName được khai báo là private:**
* Điều này đảm bảo rằng chúng chỉ có thể được truy cập hoặc sửa đổi thông qua các phương thức của lớp (chính là các getter và setter).
* Nó bảo vệ dữ liệu bên trong lớp, không cho phép thay đổi hoặc truy cập trực tiếp từ bên ngoài.
* **Các phương thức getAuthorId(), setAuthorId(int authorId), getFullName(), và setFullName(String fullName) được sử dụng:**
* Các phương thức này đóng vai trò là **cổng giao tiếp** để truy cập hoặc chỉnh sửa giá trị của authorId và fullName.
* Bạn có thể thêm logic kiểm tra dữ liệu trong các setter để đảm bảo rằng dữ liệu được nhập vào là hợp lệ (ví dụ: không cho phép fullName rỗng).
* **Bảo vệ tính toàn vẹn của dữ liệu:**
* Việc không cho phép truy cập trực tiếp đến các thuộc tính private giúp bảo vệ chúng khỏi bị thay đổi không kiểm soát từ bên ngoài.
* **Hàm khởi tạo (constructor):**
* Hàm khởi tạo Author(int authorId, String fullName) cung cấp một cách an toàn để khởi tạo đối tượng ngay từ đầu.

## Tính trừu tượng

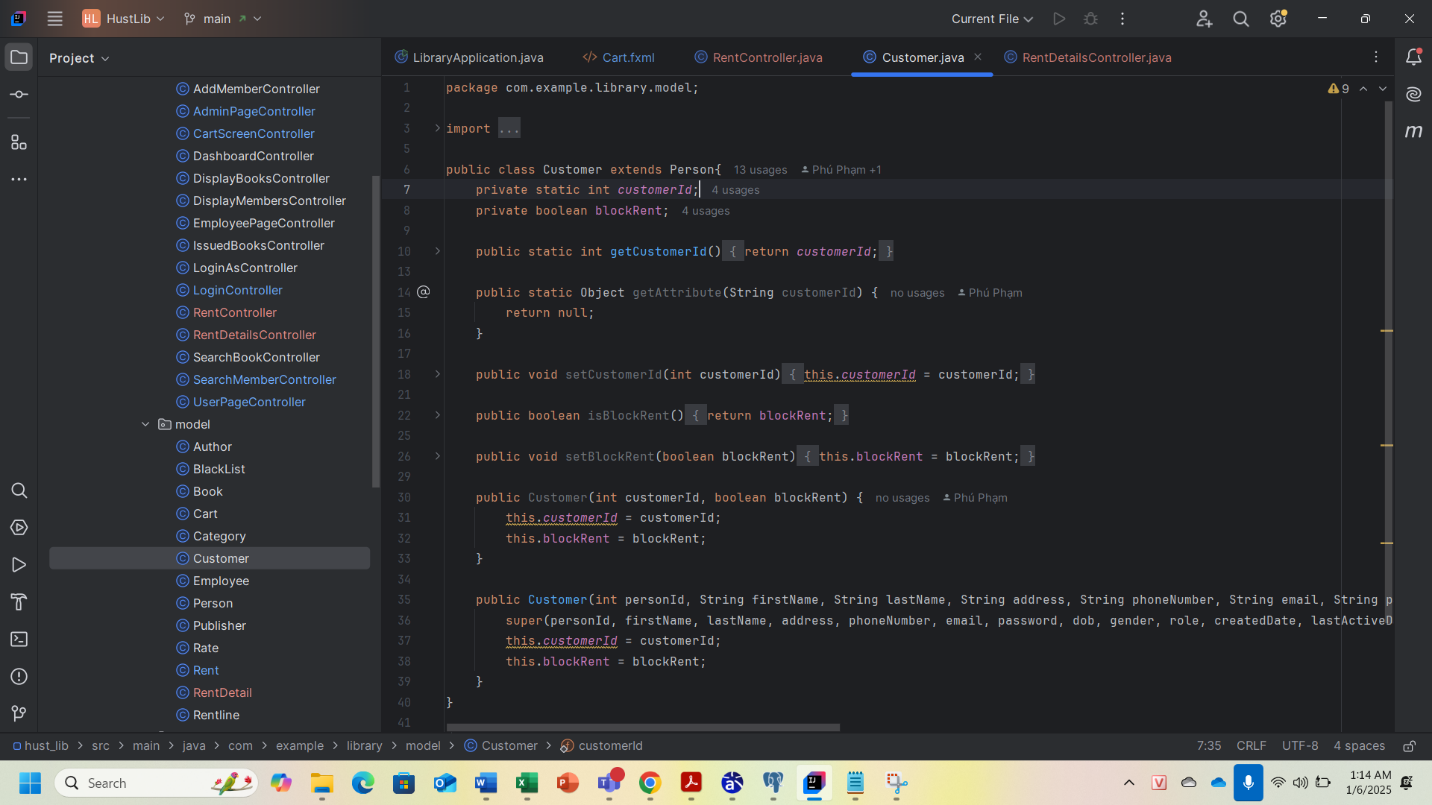
* Person là một lớp trừu tượng (abstract class) dùng để biểu diễn các đối tượng tham gia trong hệ thống, gồm các đặc điểm và hành vi chung cho tất cả các loại người dùng (**Admin**, **Employee**, và **Customer**).
* Lớp cha Person:



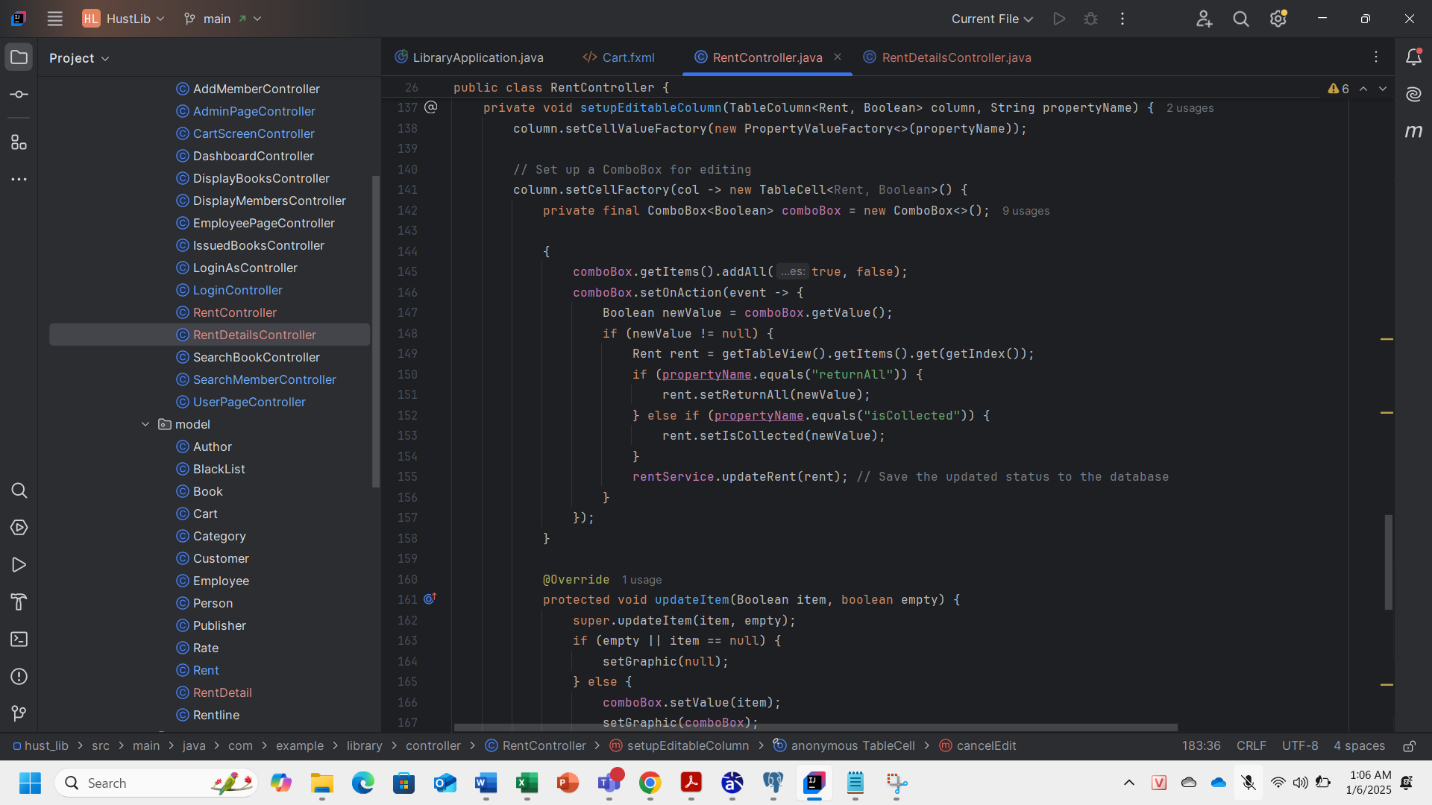
* Lớp con Employee:



* Lớp con Customer



## Tính đa hình

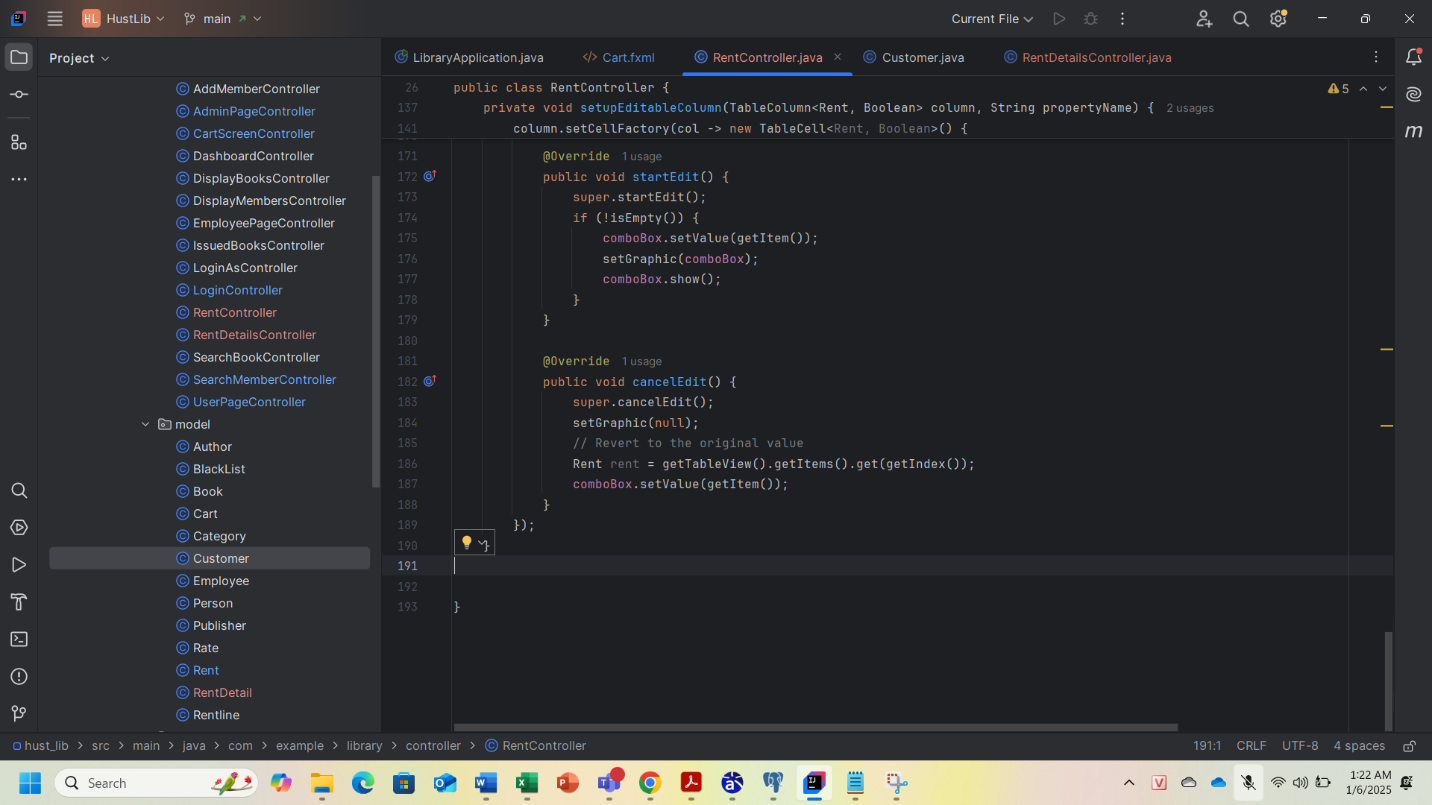


Phương thức setCellFactory của TableColumn nhận một đối tượng Callback, và trong trường hợp này, chúng ta sử dụng một TableCell<Rent, Boolean> để xử lý việc hiển thị và chỉnh sửa giá trị trong cột. TableCell là lớp cơ sở cho các ô trong bảng, và thông qua việc kế thừa từ TableCell, ta có thể định nghĩa cách thức hiển thị và chỉnh sửa các giá trị trong từng ô. Đây chính là nơi đa hình thể hiện, vì:

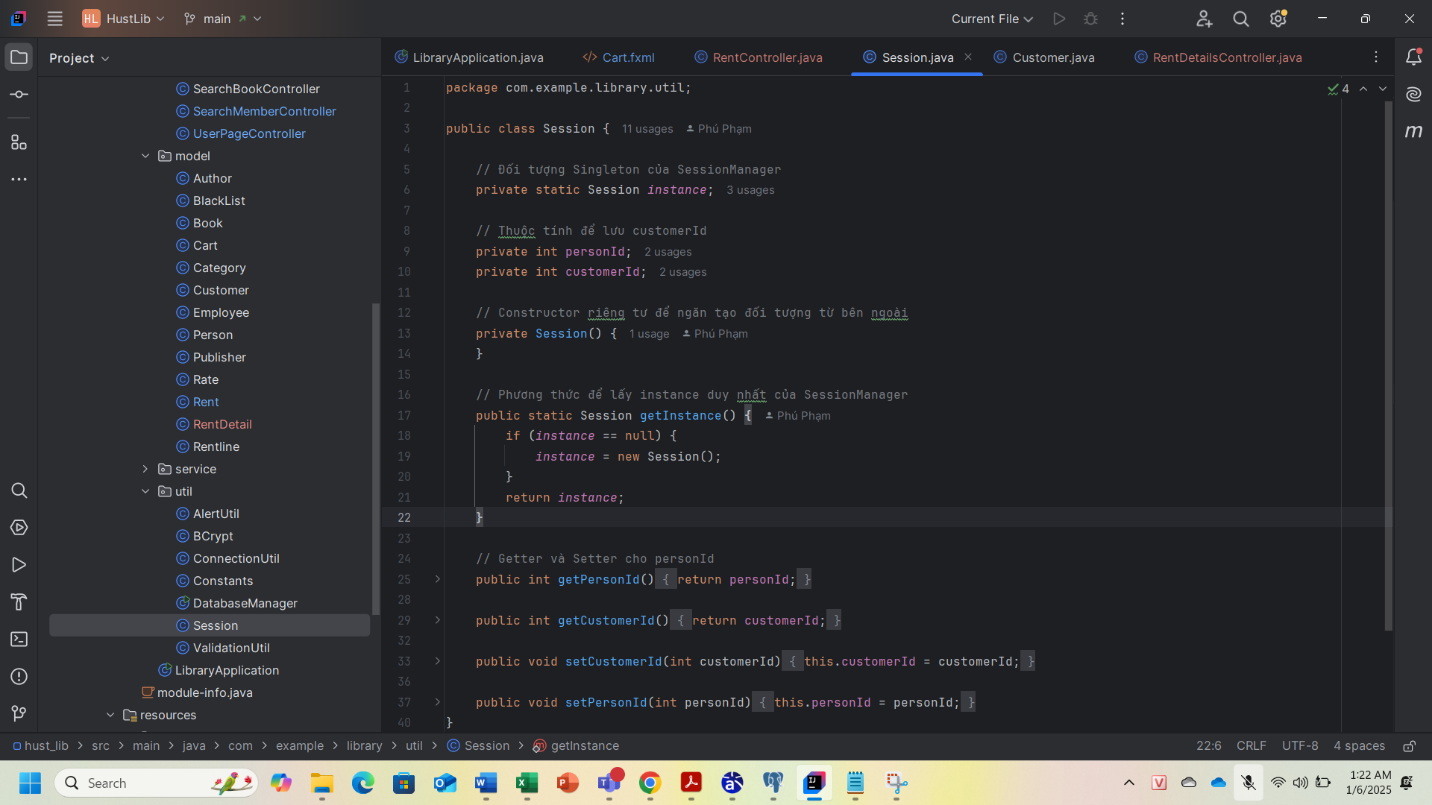
* TableCell có thể được sử dụng cho nhiều kiểu dữ liệu khác nhau (như String, Integer, Boolean...), và ta có thể tùy chỉnh từng loại TableCell cho các kiểu dữ liệu khác nhau.
* Trong đoạn mã này, TableCell<Rent, Boolean> là lớp con của TableCell dành riêng cho đối tượng Rent và kiểu dữ liệu Boolean

Phương thức updateItem trong lớp TableCell được ghi đè lại để xử lý cách hiển thị giá trị trong mỗi ô của bảng:

* updateItem(Boolean item, boolean empty) là phương thức có sẵn trong TableCell, nhưng trong đoạn mã này, nó được ghi đè để cung cấp cách hiển thị giá trị cho ô hiện tại.
* Cách cài đặt này cho phép tính đa hình: lớp con TableCell<Rent, Boolean> có thể thay đổi cách thức hoạt động của phương thức updateItem mà không làm ảnh hưởng đến những TableCell khác trong ứng dụng.
* startEdit() và cancelEdit() là các phương thức của TableCell được ghi đè. Khi người dùng bắt đầu chỉnh sửa ô trong bảng, phương thức startEdit được gọi, và khi người dùng hủy chỉnh sửa, phương thức cancelEdit được gọi.
* Việc ghi đè lại các phương thức này cho phép lớp con điều chỉnh hành vi mặc định của TableCell sao cho phù hợp với yêu cầu chỉnh sửa giá trị của một ô chứa ComboBox<Boolean>. Điều này cũng là một ví dụ của đa hình, vì ta đang thay đổi hành vi của một phương thức đã được định nghĩa trong lớp cơ sở (TableCell).



## Mẫu thiết kế Singleton



**Singleton Pattern** là một mẫu thiết kế trong lập trình, giúp đảm bảo rằng một lớp chỉ có một đối tượng duy nhất trong suốt vòng đời của ứng dụng. Mẫu thiết kế này cung cấp một phương thức để truy cập đối tượng đó, đảm bảo rằng đối tượng chỉ được tạo ra một lần và không thể tạo thêm bất kỳ bản sao nào.

**Mẫu Singleton** đảm bảo rằng chỉ có một đối tượng duy nhất của lớp Session được tạo ra. Điều này rất hữu ích khi cần lưu trữ trạng thái chung (như personId, customerId) mà không muốn tạo nhiều bản sao của đối tượng này trong suốt thời gian chạy của ứng dụng.

Đối tượng Session được truy cập thông qua phương thức getInstance(), và quá trình tạo đối tượng này chỉ xảy ra một lần duy nhất, tránh việc tạo ra nhiều đối tượng không cần thiết.

**CHƯƠNG 4. CÔNG NGHỆ SỬ DỤNG**

## Java

* Java là một ngôn ngữ lập trình hướng đối tượng, được biết đến với tính năng "Write once, run anywhere" (Viết một lần, chạy ở mọi nơi). Điều này có nghĩa là mã Java được biên dịch thành bytecode, có thể chạy trên bất kỳ nền tảng nào có máy ảo Java (JVM) mà không cần biên dịch lại.
* **Ưu điểm:**
  + **Đa nền tảng:** Chạy trên nhiều hệ điều hành khác nhau (Windows, macOS, Linux, ...).
  + **Định hướng đối tượng:** Hỗ trợ lập trình theo mô hình đối tượng, giúp code dễ hiểu và bảo trì hơn.
  + **Thư viện phong phú:** Có một cộng đồng lớn và nhiều thư viện hỗ trợ cho các lĩnh vực khác nhau.
  + **An toàn:** Cơ chế quản lý bộ nhớ tự động giúp giảm thiểu lỗi rò rỉ bộ nhớ.
  + **Mạnh mẽ:** Được sử dụng rộng rãi trong các ứng dụng doanh nghiệp lớn.

## JavaFX

* JavaFX là một thư viện đồ họa của Java, được thiết kế để tạo ra các giao diện người dùng đồ họa (GUI) hiện đại và phong phú. Nó cung cấp một bộ các control, layout, animation, effect,... giúp bạn xây dựng các ứng dụng desktop đẹp mắt và tương tác tốt với người dùng.
* **Ưu điểm:**
  + **Dễ học:** Nếu bạn đã quen với Java thì việc học JavaFX sẽ rất dễ dàng.
  + **Hiệu năng cao:** JavaFX được tối ưu để tạo ra các ứng dụng chạy mượt mà và nhanh chóng.
  + **Hỗ trợ nhiều nền tảng:** Giống như Java, JavaFX cũng chạy được trên nhiều hệ điều hành khác nhau.
  + **Tích hợp với Java:** JavaFX được tích hợp chặt chẽ với Java, cho phép bạn tận dụng tất cả các tính năng của Java.

## PostgreSQL

* PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được biết đến với tính năng mạnh mẽ, ổn định và đáng tin cậy. Nó hỗ trợ nhiều loại kiểu dữ liệu, các tính năng nâng cao như trigger, view, stored procedure, ... tuân thủ chuẩn SQL.
* **Ưu điểm:**
  + **Mã nguồn mở:** Miễn phí và có một cộng đồng lớn hỗ trợ.
  + **Mạnh mẽ:** Hỗ trợ các tính năng phức tạp của một hệ quản trị cơ sở dữ liệu.
  + **Linh hoạt:** Có thể tùy chỉnh để phù hợp với nhiều loại ứng dụng khác nhau.
  + **Ổn định:** Được sử dụng rộng rãi trong các hệ thống sản xuất.

# CHƯƠNG 5. XÂY DỰNG CHƯƠNG TRÌNH MINH HỌA

## Thư viện và công cụ sử dụng

**Danh sách các công nghệ được sử dụng**

|  |  |  |
| --- | --- | --- |
| Mục đích | Công cụ | Địa chỉ URL |
| IDE Lập trình | IntelliJ Idea Community | https://www.jetbrains.com/idea/ |
| Ngôn ngữ lập trình | Java | https://www.java.com/en/ |
| Bộ phần mềm cung cấp môi trường Java | JDK 23 | https://www.oracle.com/java/technologies/downloads/#jdk23-windows |
| Hỗ trợ thiết kế UI | Phần mềm Scene Builder | https://gluonhq.com/products/scene-builder/ |
| Thư viện thiết kế UI | Javafx | https://openjfx.io/ |
| API kết nối CSDL | pgJDBC | https://jdbc.postgresql.org/ |

## Giao diện chương trình

* Giao diện phần đăng nhập:

A screenshot of a computer

Description automatically generated

* Giao diện phần đăng ký:

A screenshot of a login form

Description automatically generated

* Trang lựa chọn quyền đăng nhập:

A screenshot of a computer

Description automatically generated

* Trang Dashboard:

A screenshot of a computer

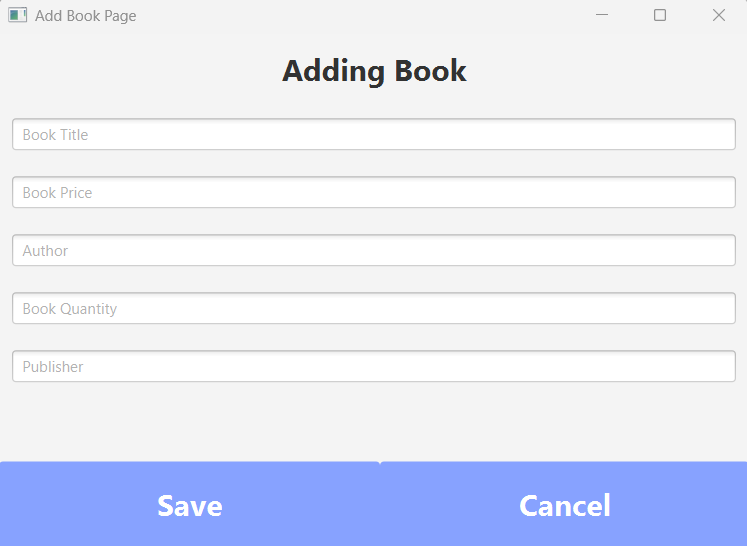
Description automatically generated

* Trang admin:

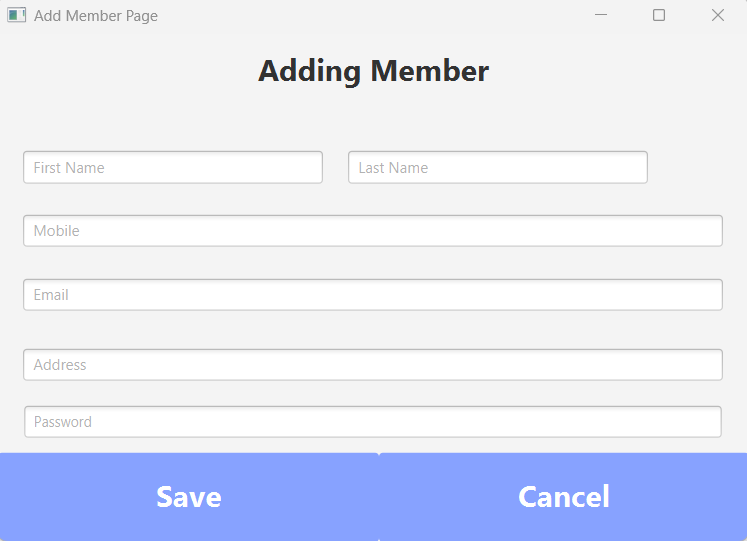
A screenshot of a computer

Description automatically generated

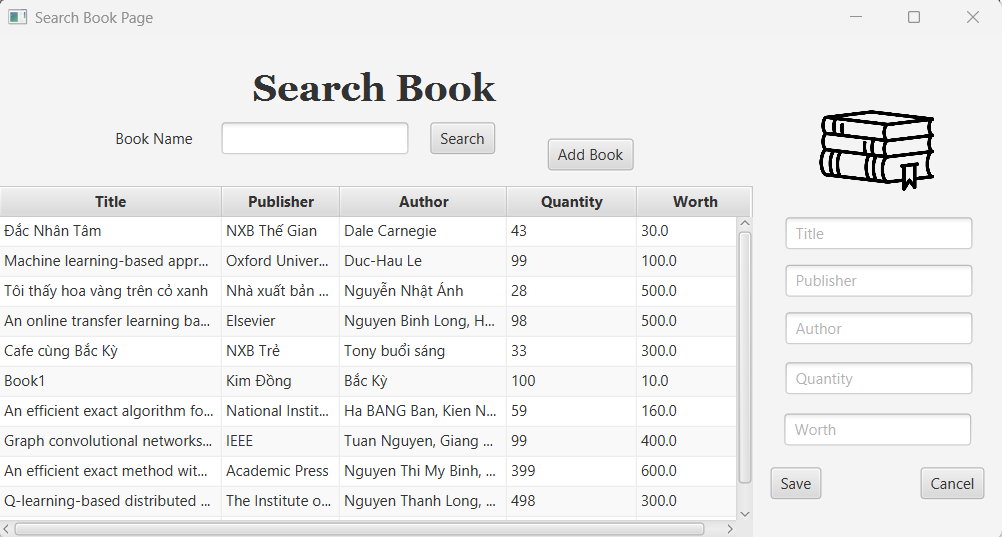
* Trang them sách:



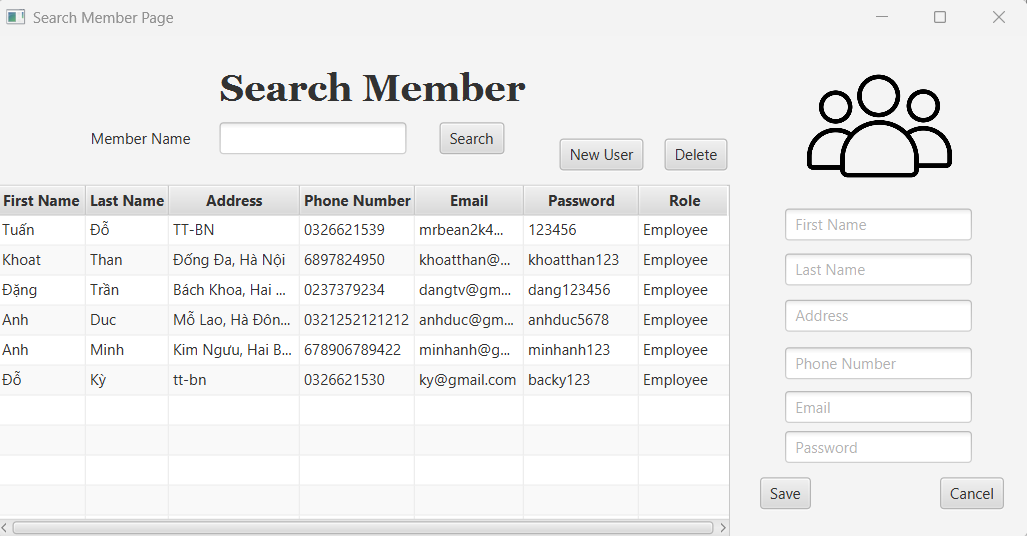
* Trang thêm nhân viên:



* Trang tìm kiếm, xem thông tin, thêm và cập nhật sách:



* Trang xem thông tin, tìm kiếm, thêm, sửa, xóa nhân viên:



* Trang thông tin trạng thái mượn/ trả sách:

A screenshot of a computer

Description automatically generated

* Trang thông tin phí phạt:

A screenshot of a computer

Description automatically generated

* Trang thông tin người dung:

A screenshot of a computer

Description automatically generated

* Trang xem lịch sử sách đã đặt:

A screenshot of a computer

Description automatically generated

* Trang giỏ hàng:

A screenshot of a computer

Description automatically generated

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## Kết luận

Hệ thống quản lý thư viện (LMS - Library Management System) đã và đang đóng vai trò quan trọng trong việc số hóa và hiện đại hóa các thư viện truyền thống. LMS không chỉ đơn thuần là một công cụ quản lý sách, mà còn là một nền tảng kết nối giữa thư viện, người dùng và các nguồn tài liệu khác nhau.

## Ưu điểm

**Tăng hiệu quả quản lý:**

* **Tự động hóa các quy trình:** Các tác vụ như mượn trả sách, tìm kiếm tài liệu, thống kê được tự động hóa, giảm thiểu lỗi sai sót và tiết kiệm thời gian cho nhân viên thư viện.
* **Quản lý tài liệu hiệu quả:** Hệ thống giúp quản lý chi tiết thông tin về sách, tạp chí, tài liệu điện tử, từ đó dễ dàng tìm kiếm, phân loại và cập nhật.
* **Quản lý người dùng:** Theo dõi thông tin về độc giả, lịch sử mượn trả, các khoản phí phát sinh...

**Nâng cao trải nghiệm người dùng:**

* **Tìm kiếm thông tin nhanh chóng:** Người dùng có thể dễ dàng tìm kiếm tài liệu thông qua các từ khóa, chủ đề, tác giả...
* **Truy cập tài liệu trực tuyến:** LMS cho phép người dùng truy cập các tài liệu điện tử, e-book, bài báo trực tuyến từ bất kỳ đâu có kết nối internet.

**Tích hợp các dịch vụ bổ sung:**

* **Đặt chỗ tài liệu:** Người dùng có thể đặt trước các tài liệu đang được mượn.
* **Gia hạn mượn:** Người dùng có thể tự động gia hạn thời hạn mượn sách qua hệ thống.

## Nhược điểm

* **Yêu cầu kết nối internet ổn định:** Để hoạt động hiệu quả, LMS cần có kết nối internet ổn định.
* **Khả năng bảo mật:** Dữ liệu người dùng và tài liệu cần được bảo mật nghiêm ngặt để tránh rò rỉ thông tin.
* **Cần cập nhật thường xuyên:** LMS cần được cập nhật thường xuyên để đảm bảo tương thích với các công nghệ mới và khắc phục các lỗi phát sinh.

## Hướng phát triển

* **Tích hợp trí tuệ nhân tạo:** Áp dụng AI để nâng cao khả năng tìm kiếm, đề xuất tài liệu, tự động hóa các quy trình phức tạp hơn.
* **Phát triển các ứng dụng di động:** Cho phép người dùng truy cập và quản lý tài khoản, mượn trả sách mọi lúc mọi nơi.
* **Tích hợp với các nền tảng mạng xã hội:** Tạo ra một cộng đồng người dùng, chia sẻ thông tin về sách, tạo ra các nhóm thảo luận.
* **Kết nối với các thư viện khác:** Hình thành mạng lưới thư viện liên kết, mở rộng nguồn tài liệu cho người dùng.
* **Ứng dụng công nghệ thực tế ảo và tăng cường:** Tạo ra trải nghiệm đọc sách mới lạ và hấp dẫn.

# TÀI LIỆU THAM KHẢO

[1] Slide bài giảng môn “Lập trình hướng đối tượng” của thầy Lê Đức Hậu

[2] Slide bài giảng môn “Thực hành cơ sở dữ liệu” của cô Nguyễn Thị Oanh