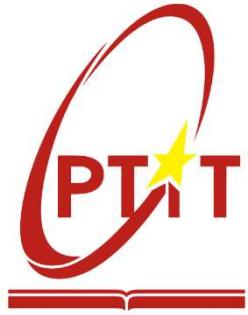


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



BÀI TIỂU LUẬN

HỌC PHẦN

**PHÂN TÍCH VÀ THIẾT KẾ
HỆ THỐNG THÔNG TIN**

Giảng viên hướng dẫn: Trần Đình Quế

Sinh viên thực hiện: Trần Đỗ Minh

Mã sinh viên: B18DCCN411

Nhóm học phần: 08

Hà Nội, 01/2022

LỜI CẢM ƠN

Trước tiên, với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô tại Học viện Công nghệ Bưu chính Viễn thông lời cảm ơn chân thành vì đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho em trong thời gian học tập tại Học viện.

Đặc biệt, trong học kỳ này, Học viện đã tổ chức cho em được tiếp cận với các môn học rất hữu ích đối với sinh viên. Em xin chân thành cảm ơn thầy Trần Đình Quế đã tận tâm hướng dẫn em trong suốt thời gian của môn học *Phân tích và Thiết kế Hệ thống thông tin*. Thầy đã luôn đỗ bên cạnh, tạo điều kiện trong suốt quá trình nghiên cứu, động viên và giúp đỡ để em hoàn thành môn học và thực hiện bài tiểu luận này. Bước đầu đi vào tìm hiểu về lĩnh vực, kiến thức của em còn nhiều hạn chế và thiếu sót. Vì vậy, em rất mong nhận được những ý kiến đóng góp quý báu của thầy để bài tiểu luận của em được hoàn thiện hơn.

Cuối cùng, em xin kính chúc thầy dồi dào sức khỏe, hạnh phúc để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt tri thức cho các thế hệ sau.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Trần Đỗ Minh

MỤC LỤC

PHẦN 1: CÂU HỎI	4
Câu 1: Các kiểu kiến trúc phần mềm.....	4
Câu 2: Các bước phân tích	6
Câu 3: Các bước thiết kế	13
Câu 4: Kiểu kiến trúc 3 lớp & MVC	20
Câu 5: DAO? Tại sao sử dụng? Biểu đồ lớp & code cho DAO. Xem lại phụ lục B1 & Giải thích biểu đồ	22
Câu 6: Thiết kế cơ sở dữ liệu.....	33
Câu 7: Thiết kế giao diện.....	39
PHẦN 2: DỰ ÁN – CỬA HÀNG SÁCH TRỰC TUYẾN	42
Câu 1: Vẽ biểu đồ use case với mức thông tin chi tiết	42
Câu 2: Xây dựng biểu đồ lớp phân tích (Class diagram analysis) với một số phương thức chính	45
Câu 3: Xây dựng biểu đồ lớp cho mô hình dữ liệu (Class diagram model), sinh mô hình dữ liệu và cơ sở dữ liệu trong MySQL	46
Câu 4 và câu 5: Xây dựng biểu đồ lớp thiết kế (Class diagram design), sử dụng mẫu DAO để thiết kế lại các lớp trong hệ thống	48
Câu 6: Xây dựng các biểu đồ gói theo kiến trúc 3 lớp của MVC cho hệ thống	48
Câu 7: Cài đặt hệ thống (sử dụng JSP Servlet)	56
TÀI LIỆU THAM KHẢO.....	59

PHẦN 1: CÂU HỎI

Câu 1: Các kiểu kiến trúc phần mềm

Các kiểu kiến trúc được sử dụng trong khi thiết kế phần mềm như sau:

1. Kiến trúc lấy dữ liệu làm trung tâm (Data-centered architecture)

- Việc lưu trữ dữ liệu (trong tệp hoặc cơ sở dữ liệu) nằm ở vị trí trung tâm của kiến trúc.
- Dữ liệu lưu trữ được truy cập liên tục bởi các thành phần khác như cập nhật, xóa, thêm, sửa đổi từ kho dữ liệu.
- Kiến trúc lấy dữ liệu làm trung tâm giúp đảm bảo tính toàn vẹn.
- Truyền dữ liệu giữa các máy khách bằng cơ chế bảng đen (blackboard).
- Các tiến trình được thực thi độc lập bởi các thành phần của máy khách.

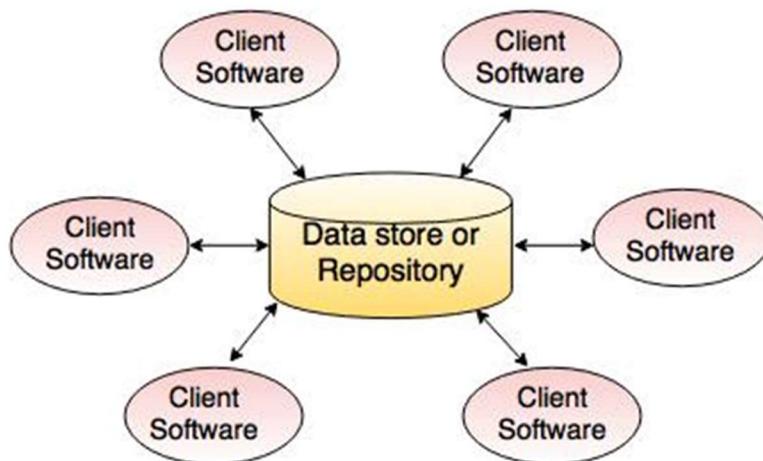


Fig.- Data centered architecture

Data store hay Repository: Kho dữ liệu hoặc kho lưu trữ

2. Kiến trúc luồng dữ liệu (Data-flow architecture)

- Kiến trúc này được sử dụng khi dữ liệu đầu vào được chuyển đổi thành một chuỗi các thành phần thao tác xử lý dữ liệu để tạo ra dữ liệu đầu ra.
- Một đường ống và mẩu bộ lọc là một tập hợp gồm các thành phần gọi là bộ lọc.
- Các bộ lọc được kết nối thông qua các đường ống và truyền dữ liệu từ thành phần này sang thành phần khác.

- Luồng dữ liệu rút gọn thành một đường vận chuyển duy nhất, gọi là tuần tự theo mè.

3. Kiến trúc gọi và trả về (Call and return architectures)

- Kiểu kiến trúc này cho phép ta dễ dàng sửa đổi cấu trúc chương trình.
- Các kiểu kiến trúc con là:
 - *Chương trình chính hoặc kiến trúc chương trình con (Main program or subprogram architecture)*
 - Chương trình được chia thành các phần nhỏ hơn theo thứ bậc.
 - Chương trình chính gọi đến các thành phần của chương trình, theo một hệ thống phân cấp.
 - Trong hệ thống phân cấp này, các thành phần của chương trình lại được phân chia thành các chương trình con
 - *Kiến trúc gọi thủ tục từ xa (Remote procedure call architecture)*
 - Chương trình chính hoặc các thành phần của chương trình con được phân tán trong một mạng nhiều máy tính.
 - Mục đích chính là tăng hiệu năng.

4. Kiến trúc hướng đối tượng (Object-oriented architecture)

- Kiến trúc này là phiên bản mới nhất của kiến trúc gọi và trả về.
- Bao gồm gói dữ liệu và các phương thức.

5. Kiến trúc phân lớp (Layered architecture)

- Kiến trúc xác định nhiều lớp khác nhau, gồm lớp bên ngoài và bên trong.
- Các thành phần của lớp ngoài quản lý các hoạt động giao diện người dùng.
- Các thành phần thực thi giao diện hệ điều hành ở lớp bên trong.
- Các lớp bên trong là lớp ứng dụng, lớp tiện ích và lớp lõi.
- Trong nhiều trường hợp, có thể có nhiều hơn một mẫu kiến trúc phù hợp. Chúng ta có thể thiết kế và đánh giá một kiến trúc thay thế.

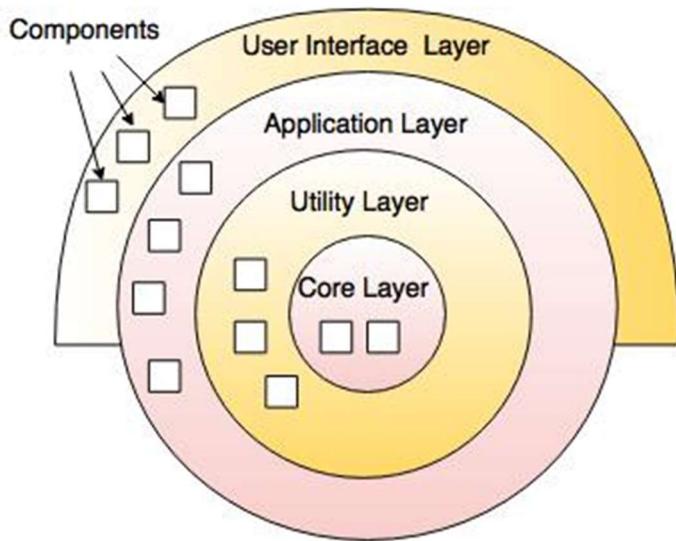


Fig.- Layered Architecture

Câu 2: Các bước phân tích

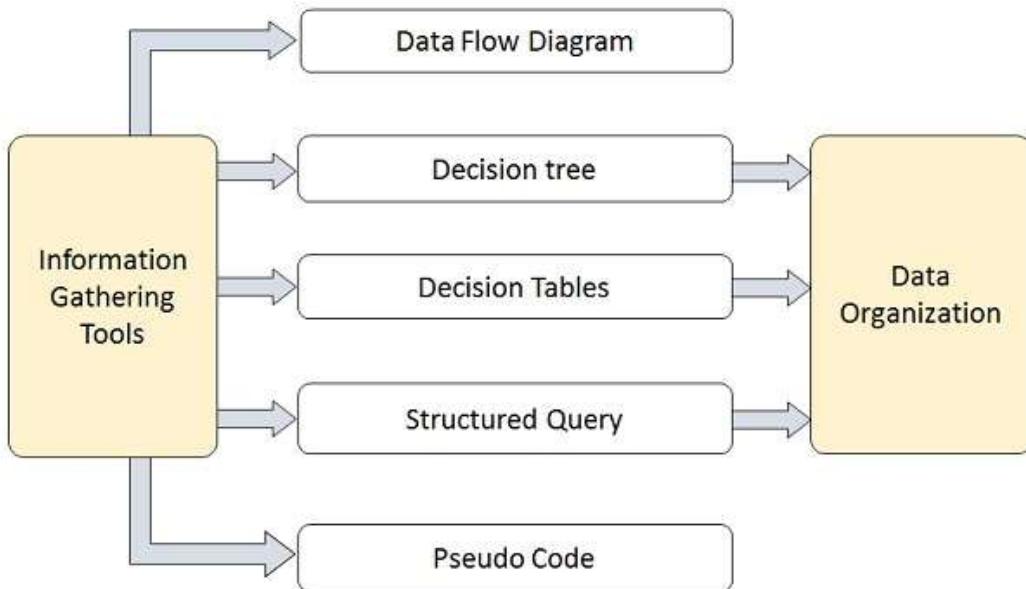
- **Phân tích hệ thống** là một quá trình thu thập và diễn giải các dữ kiện, xác định các vấn đề và phân rã một hệ thống thành các thành phần.
- Đây là một kỹ thuật giải quyết vấn đề nhằm cải thiện hệ thống và đảm bảo rằng tất cả các thành phần của hệ thống hoạt động hiệu quả để hoàn thành mục đích của chúng.
- Phân tích **xác định những gì hệ thống phải làm**.
- Các nhà phân tích sử dụng các công cụ khác nhau để hiểu và mô tả hệ thống thông tin. Một trong các cách để phân tích hệ thống là sử dụng **Phân tích hướng cấu trúc (Structured analysis)**.

1. Phân tích hướng cấu trúc

- Phân tích hướng cấu trúc là một phương pháp phát triển cho phép người phân tích hiểu hệ thống và các hoạt động của nó một cách hợp lý.
- Phương pháp này có các đặc điểm sau:
 - Gồm các hình ảnh (đồ họa) làm rõ cách một ứng dụng thể hiện.
 - Phân chia các tiến trình để đưa ra một bức tranh rõ ràng về luồng hệ thống.
 - Mang tính logic hơn là tính vật lý (các yếu tố trong hệ thống không phụ thuộc vào phần cứng).
 - Là một cách tiếp cận hoạt động từ tổng quan cấp cao đến chi tiết cấp thấp.

2. Công cụ phân tích hướng cấu trúc

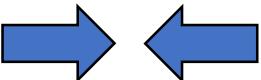
- Trong quá trình Phân tích hướng cấu trúc, nhiều công cụ và kỹ thuật khác nhau được sử dụng để phát triển hệ thống:
 - Biểu đồ luồng dữ liệu (Data Flow Diagrams – DFD)
 - Từ điển dữ liệu (Data Dictionary)
 - Cây quyết định (Decision Trees)
 - Bảng quyết định (Decision Tables)
 - Tiếng Anh có cấu trúc (Structured English)
 - Giả mã (Pseudocode)



3. Biểu đồ luồng dữ liệu (Data Flow Diagram – DFD) hoặc Biểu đồ bong bóng (Bubble Chart)

- Đây là một kỹ thuật được phát triển bởi Larry Constantine để thể hiện các yêu cầu của hệ thống dưới dạng đồ họa.
- Thể hiện luồng dữ liệu giữa các chức năng khác nhau của hệ thống và làm rõ cách hệ thống được cài đặt.
- Đây là giai đoạn ban đầu của pha thiết kế: Phân chia các yêu cầu chức năng tới mức chi tiết cụ thể nhất.
- Việc sử dụng đồ họa khiến phương pháp này trở thành một công cụ giao tiếp tốt giữa người dùng và nhà phân tích, hoặc giữa nhà phân tích và nhà thiết kế hệ thống.
- Cung cấp một cái nhìn tổng quan về dữ liệu mà hệ thống xử lý, những chuyển đổi nào được thực hiện, những dữ liệu nào được lưu trữ, những kết quả nào được tạo ra và chúng lưu chuyển đến đâu.

3.1. Các yếu tố cơ bản của DFD

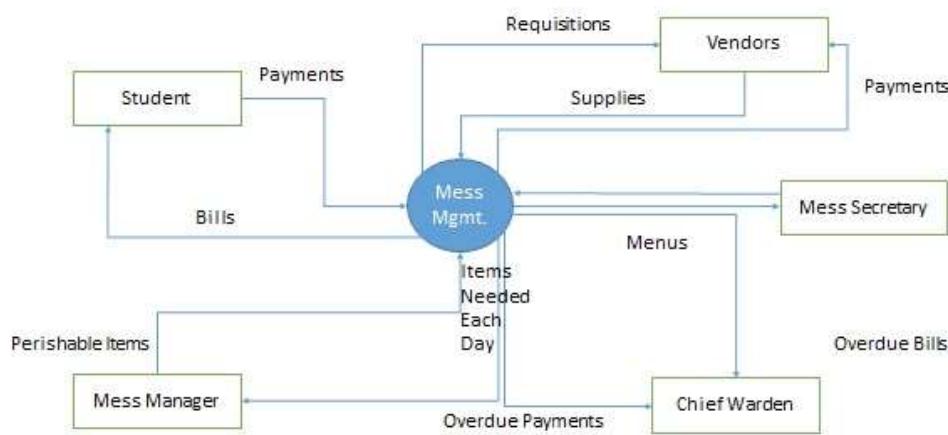
Tên kí tự	Kí tự	Ý nghĩa
Hình vuông		Nguồn hoặc đích đến của dữ liệu
Mũi tên		Luồng dữ liệu
Hình tròn		Quá trình chuyển đổi dữ liệu
Hình chữ nhật mở		Lưu trữ dữ liệu

3.2. Các loại DFD

DFD vật lý	DFD logic
Phụ thuộc vào việc cài đặt, cho biết những chức năng nào được thực hiện.	Độc lập với việc cài đặt, chỉ chú trọng vào luồng dữ liệu giữa các tiến trình.
Cung cấp thông tin chi tiết ở mức thấp về phần cứng, phần mềm, tệp và con người.	Giải thích các sự kiện của hệ thống và dữ liệu mà mỗi sự kiện yêu cầu.
Mô tả cách hệ thống hiện tại vận hành và cách một hệ thống sẽ được cài đặt.	Chỉ ra cách vận hành của nghiệp vụ; không phải cách cài đặt hệ thống.

3.3. Biểu đồ bối cảnh (Context Diagram)

- Biểu đồ bối cảnh giúp **hiểu rõ toàn bộ hệ thống**, thông qua một DFD cho ta cái nhìn tổng quan về một hệ thống.
- Ví dụ, biểu đồ bối cảnh của hệ thống quản lý nhà ăn (mess management) được hiển thị bên dưới.



4. Từ điển dữ liệu (Data Dictionary)

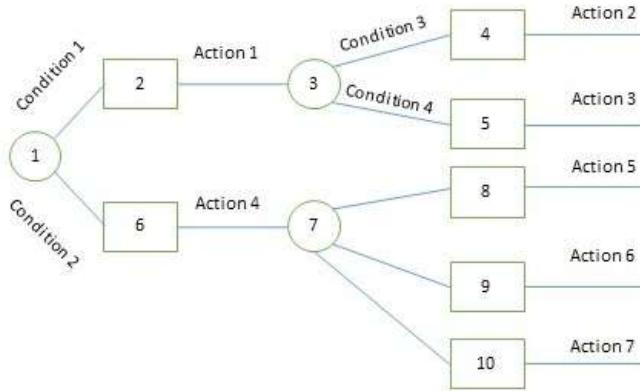
- Từ điển dữ liệu là một kho lưu trữ có cấu trúc chứa các phần tử dữ liệu trong hệ thống.
- Lưu trữ mô tả của tất cả các phần tử dữ liệu DFD, gồm chi tiết và định nghĩa về luồng dữ liệu, kho dữ liệu, dữ liệu được lưu trữ trong kho dữ liệu và các tiến trình.
- Từ điển dữ liệu đóng một vai trò quan trọng trong việc xây dựng cơ sở dữ liệu. Hầu hết các DBMS đều có từ điển dữ liệu như một tính năng tiêu chuẩn.
- Ví dụ:

Sr.No.	Tên dữ liệu	Mô tả	Số kí tự
1	ISBN	Số ISBN	10
2	TITLE	Tên sách	60
3	SUB	Chủ đề sách	80
4	ANAME	Tên tác giả	15

5. Cây quyết định (Decision Trees)

- Cây quyết định là một phương pháp để xác định các mối quan hệ phức tạp bằng cách mô tả các quyết định và tránh các vấn đề trong giao tiếp.
- Cây quyết định là một sơ đồ thể hiện các **hành động và điều kiện khác nhau** trong khuôn khổ cây ngang
- Cây quyết định mô tả mối quan hệ của từng điều kiện và các hành động được phép của chúng. Một nút hình vuông biểu thị một hành động và một nút tròn biểu thị một điều kiện.

- Buộc các nhà phân tích phải xem xét **trình tự của các quyết định** và **xác định quyết định** thực tế cần phải đưa ra.



- Hạn chế chính của cây quyết định là **thiếu thông tin trong định dạng** của nó để mô tả những **tổ hợp điều kiện khác** mà ta có thể dùng để kiểm tra.
- Ví dụ:



6. Bảng quyết định (Decision Tables)

- Bảng quyết định là một phương pháp mô tả mối quan hệ logic phức tạp một cách chính xác và dễ hiểu.
- Nó hữu ích trong các tình huống mà các **hành động kết quả phụ thuộc** vào một hoặc nhiều **tổ hợp các điều kiện độc lập**.
- Nó là một ma trận gồm hàng hoặc cột để xác định một vấn đề và các hành động.

6.1. Các thành phần của Bảng quyết định:

- Stub điều kiện (Condition Stub) - Nằm ở góc phần tư phía trên bên trái, liệt kê tất cả các điều kiện cần kiểm tra.

- Stub hành động (Action Stub) - Nằm ở góc phần tư phía dưới bên trái, phác thảo tất cả các hành động cần thực hiện để đáp ứng điều kiện như vậy.
- Mục điều kiện (Condition Entry) - Nằm ở góc phần tư phía trên bên phải, cung cấp câu trả lời cho các câu hỏi được hỏi trong góc phần tư Stub điều kiện.
- Mục hành động (Action Entry) - Nằm ở góc phần tư phía dưới bên phải, cho biết hành động thích hợp sinh ra từ câu trả lời cho các điều kiện trong góc phần tư mục điều kiện.
- Các mục trong bảng quyết định được đưa ra bởi Quy tắc quyết định (Decision Rules). Quy tắc này xác định mối quan hệ giữa các tổ hợp điều kiện và các quá trình hành động.
- Y cho thấy sự tồn tại của một điều kiện.
- N đại diện cho điều kiện không được thỏa mãn.
- Dấu - thể hiện hành động sẽ được bỏ qua.
- X thể hiện hành động sẽ được thực hiện.

Điều kiện	Quy tắc 1	Quy tắc 2	Quy tắc 3	Quy tắc 4
Advance payment made	Y	N	N	N
Purchase amount >= 10000	-	Y	Y	N
Regular Customer	-	Y	N	-
Hành động				
Give 5% discount	X	X	-	-
Give no discount	-	-	X	X

7. Tiếng Anh có cấu trúc (Structured English)

- Tiếng Anh cấu trúc có nguồn gốc từ **ngôn ngữ lập trình hướng cấu trúc**, giúp mô tả quy trình dễ hiểu và chính xác hơn.
- Dựa trên logic thủ tục sử dụng cách đặt câu và các câu mệnh lệnh được thiết kế để thực hiện thao tác cho hành động.

- Được sử dụng **tốt nhất khi các chuỗi và vòng lặp** trong một chương trình phải được xem xét, và vấn đề cần các chuỗi hành động cùng quyết định.
- Thể hiện tất cả logic về cấu trúc quyết định tuần tự và sự lặp lại.
- Ví dụ:

```

if customer pays advance
    then
        Give 5% Discount
    else
        if purchase amount >=10,000
            then
                if the customer is a regular customer
                    then Give 5% Discount
                else No Discount
            end if
        else No Discount
    end if
end if

```

9. Giả mã (Pseudocode)

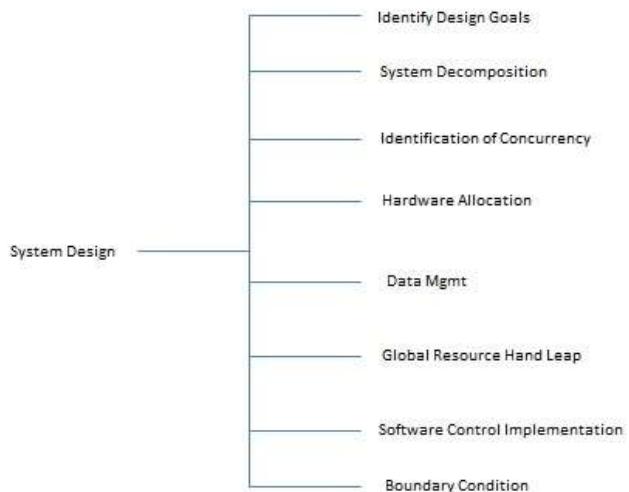
- Mã giả không tuân theo bất kỳ ngôn ngữ lập trình nào và **diễn đạt logic** bằng tiếng Anh đơn giản.
- Có thể **chỉ định logic lập trình vật lý** mà không cần lập trình thực sự trong và sau khi thiết kế vật lý.
- Được sử dụng kết hợp với lập trình hướng cấu trúc.
- Thay thế các sơ đồ khối thể hiện luồng của một chương trình

10. Hướng dẫn lựa chọn công cụ thích hợp

- Sử dụng các nguyên tắc sau để chọn công cụ thích hợp nhất phù hợp với yêu cầu:
 - Sử dụng DFD ở phân tích bậc cao hoặc bậc thấp để có tài liệu hệ thống tốt.
 - Sử dụng từ diễn dữ liệu để đơn giản hóa cấu trúc nhằm đáp ứng yêu cầu dữ liệu của hệ thống.
 - Sử dụng tiếng Anh có cấu trúc nếu có nhiều vòng lặp và hành động phức tạp.
 - Sử dụng các bảng quyết định khi có một số lượng lớn các điều kiện cần kiểm tra và logic phức tạp.
 - Sử dụng cây quyết định khi trình tự các điều kiện là quan trọng và nếu có ít điều kiện cần được kiểm tra.

Câu 3: Các bước thiết kế

- Thiết kế hệ thống là giai đoạn thu hẹp khoảng cách giữa miền của bài toán (problem domain) và hệ thống hiện có theo cách có thể quản lý được.
- Đây là giai đoạn mà tài liệu SRS được chuyển đổi thành một định dạng có thể cài đặt được. Giai đoạn này cũng quyết định cách hệ thống sẽ hoạt động.
- Trong giai đoạn này, hoạt động phát triển hệ thống phức tạp được chia thành nhiều hoạt động phụ nhỏ hơn, phối hợp với nhau để đạt được mục tiêu chính của phát triển hệ thống.



1. Đầu vào cho thiết kế hệ thống

- Thiết kế hệ thống có các yếu tố đầu vào sau:
 - Báo cáo công việc
 - Kế hoạch xác định yêu cầu
 - Phân tích tình hình hiện tại
 - Các yêu cầu hệ thống được đề xuất bao gồm mô hình dữ liệu mức khái niệm, DFD đã sửa đổi và Siêu dữ liệu (Metadata).

2. Đầu ra cho thiết kế hệ thống

- Thiết kế hệ thống cho các kết quả sau:
 - Những thay đổi về cơ sở hạ tầng và tổ chức đối với hệ thống được đề xuất.
 - Một lược đồ dữ liệu, thường là một lược đồ quan hệ.
 - Siêu dữ liệu để xác định bảng / tệp và cột / dữ liệu-mục.
 - Sơ đồ phân cấp chức năng hoặc bản đồ trang web mô tả cấu trúc chương trình bằng đồ họa.
 - Mã thật hoặc mã giả cho mỗi mô-đun trong chương trình.

- Một bản mẫu cho hệ thống được đề xuất.

3. Các loại thiết kế hệ thống

3.1. Thiết kế logic

- Thiết kế logic liên quan đến một **biểu diễn trừu tượng của luồng dữ liệu**, đầu vào và đầu ra của hệ thống.
- Mô tả các đầu vào (nguồn), đầu ra (đích), cơ sở dữ liệu (kho dữ liệu), thủ tục (luồng dữ liệu) tất cả ở định dạng đáp ứng yêu cầu của người dùng.
- Sử dụng biểu đồ luồng dữ liệu (DFD) và mô hình hóa biểu đồ E-R

3.2 Thiết kế vật lý

- Thiết kế vật lý liên quan đến các quá trình đầu vào và đầu ra thực tế của hệ thống.
- Tập trung vào cách dữ liệu được **nhập vào hệ thống, được xác minh, xử lý và hiển thị** dưới dạng đầu ra.
- Tạo ra hệ thống làm việc bằng cách xác định đặc tả thiết kế chỉ định chính xác những gì hệ thống làm.
- Bao gồm các bước sau:
 - Chỉ định phương tiện đầu vào / đầu ra, thiết kế cơ sở dữ liệu và chỉ định các thủ tục sao lưu.
 - Lập kế hoạch cài đặt hệ thống.
 - Lập kế hoạch kiểm tra và cài đặt cũng như chỉ định bất kỳ phần cứng và phần mềm mới nào.
 - Cập nhật chi phí, lợi ích, ngày chuyển đổi và các ràng buộc hệ thống.

3.3 Thiết kế kiến trúc

- Còn gọi là thiết kế bậc cao, tập trung vào việc thiết kế kiến trúc hệ thống.
- Mô tả **cấu trúc và hành vi của hệ thống**.
- Xác định cấu trúc và mối quan hệ giữa các mô-đun khác nhau của quá trình phát triển hệ thống.

3.4. Thiết kế chi tiết

- Tuân theo thiết kế kiến trúc và tập trung vào **phát triển từng mô-đun**.

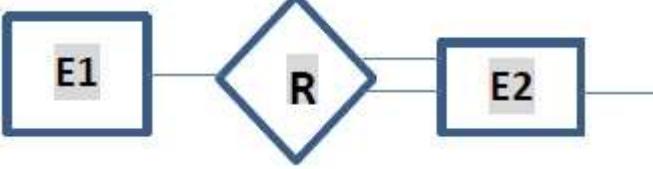
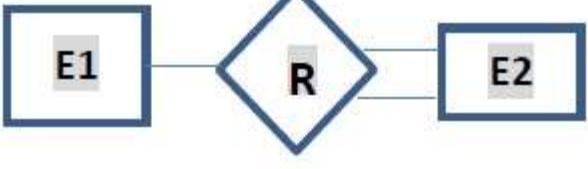
3.5. Mô hình hóa dữ liệu mức khái niệm

- Biểu diễn dữ liệu có tổ chức, gồm tất cả các thực thể và mối quan hệ chính.
- Mục đích chính: nắm bắt được càng nhiều ý nghĩa của dữ liệu càng tốt.

3.6. Mô hình quan hệ-thực thể (Entity Relationship Model)

- Là một kỹ thuật được sử dụng trong thiết kế cơ sở dữ liệu giúp mô tả mối quan hệ giữa các thực thể khác nhau của một tổ chức.
- Gồm các thành phần:
 - Thực thể (Entity) - Chỉ định các đối tượng thực riêng biệt trong một ứng dụng. Ví dụ: nhà cung cấp, mặt hàng, sinh viên, khóa học, giáo viên, v.v.
 - Quan hệ (Relationship) - Mối quan hệ phụ thuộc có ý nghĩa giữa các thực thể. Ví dụ: các mặt hàng cung cấp của nhà cung cấp, giáo viên dạy các khóa học, sau đó việc cung cấp và dạy là mối quan hệ.
 - Thuộc tính (Attributes) - Chỉ định các thuộc tính của các quan hệ. Ví dụ: mã nhà cung cấp, tên sinh viên.
- Các ký hiệu được sử dụng trong mô hình E-R và ý nghĩa tương ứng của chúng:

Kí tự	Ý nghĩa
	Thực thể
	Thực thể yếu
	Quan hệ
	Quan hệ xác định
	Thuộc tính
	Thuộc tính khóa

	Thuộc tính có nhiều giá trị (Ex: Phone)
	Thuộc tính đa hợp (Ex: name)
	Thuộc tính kế thừa (Ex: Từ dob -> age)
	E2 tham gia đầy đủ trong R
	Số lượng tham gia E1:E2 = 1:N

- Ba loại mối quan hệ có thể tồn tại giữa hai tập dữ liệu: một-một, một-nhiều và nhiều-nhiều.

4. Tổ chức tệp

- Mô tả cách các bản ghi được lưu trữ trong một tệp.
- Có bốn phương pháp tổ chức tệp:
 - Theo chuỗi (Serial) - Các bản ghi được lưu trữ **theo thứ tự thời gian** (theo thứ tự khi chúng được nhập hoặc xuất hiện). Ví dụ: ghi chi phí điện thoại, giao dịch ATM, hàng đợi điện thoại.
 - Tuần tự (Sequential) - Bản ghi được lưu **trữ theo thứ tự dựa trên trường khóa** chứa giá trị xác định duy nhất một bản ghi. Ví dụ - Danh bạ điện thoại.
 - Trực tiếp / Tương đối (Direct / Relative) - Mỗi bản ghi được lưu trữ **dựa trên một địa chỉ vật lý trên thiết bị**. Địa chỉ được tính từ giá trị được lưu

trữ trong trường khóa của bản ghi. Quy trình ngẫu nhiên hoặc thuật toán băm thực hiện chuyển đổi.

- Theo chỉ mục (Indexed) - Các bản ghi có thể được **xử lý cả tuần tự và không tuần tự** bằng cách sử dụng các chỉ mục.
- So sánh:

	Serial	Sequential	Direct	Index
Type of Access	Batch	Batch	Online	Batch or Online
Data Organization	Serial	Sequentially by key value	No particular order	Sequentially and by index
Flexibility in handling inquiries	No	No	Yes	Yes
Availability of up to date Data	No	No	Yes	Yes
Speed Retrieval	Slow	Slow	Very Fast	Fast
Activity	High	High	Low	High
Volatility	Low	Low	High	High
Example	ATM Transition Queue	Payroll process script billing operation	Online reservation and banking transaction	Customer ordering and billing

5. Truy cập tệp

- Các phương thức truy cập tệp cho phép các chương trình máy tính đọc hoặc ghi các bản ghi trong một tệp.

5.1. Truy cập tuần tự

- Mọi bản ghi trên tệp được xử lý bắt đầu từ bản ghi đầu tiên cho đến khi **đạt đến Kết thúc Tệp (EOF)**.
- Hiệu quả khi một số lượng lớn các bản ghi trong tệp cần được truy cập vào bất kỳ thời điểm nào.
- Dữ liệu được lưu trữ trên băng (truy cập tuần tự) chỉ có thể được truy cập tuần tự.

5.2. Truy cập trực tiếp (Ngẫu nhiên)

- Các bản ghi được định vị bằng cách biết vị trí hoặc **địa chỉ vật lý của chúng trên thiết bị** hơn là vị trí tương đối của chúng so với các bản ghi khác.
- Dữ liệu được lưu trữ trên thiết bị CD (truy cập trực tiếp) có thể được truy cập tuần tự hoặc ngẫu nhiên.

5.3. Các loại tệp được sử dụng trong hệ thống tổ chức

- Sau đây là các loại tệp được sử dụng trong hệ thống tổ chức:

- Tệp chính (Master file) - Chứa thông tin hiện tại của một hệ thống. Ví dụ: tệp khách hàng, tệp sinh viên, danh bạ điện thoại.
- Tệp bảng (Table file) - Đây là một loại tệp chính thay đổi không thường xuyên và được lưu trữ ở định dạng bảng. Ví dụ, lưu trữ Zipcode.
- Tệp giao dịch (Transaction file) - Chứa thông tin hàng ngày được tạo ra từ các hoạt động kinh doanh. Nó được sử dụng để cập nhật hoặc xử lý tệp chính. Ví dụ: Địa chỉ của nhân viên.
- Tệp tạm thời (Temporary file) - Được tạo và sử dụng bất cứ khi nào hệ thống cần.
- Tệp phản chiếu (Mirror file) - Là bản sao chính xác của các tệp khác. Giúp giảm thiểu rủi ro thời gian chết trong trường hợp bản gốc không sử dụng được. Chúng phải được sửa đổi mỗi khi thay đổi tệp gốc.
- Tệp nhật ký (Log file) - Chứa các bản sao của hồ sơ chính và giao dịch để ghi lại mọi thay đổi được thực hiện đối với tệp chính. Nó tạo điều kiện thuận lợi cho việc kiểm tra và cung cấp cơ chế phục hồi trong trường hợp hệ thống bị lỗi.
- Lưu trữ tệp (Archive file) - Sao lưu tệp có chứa phiên bản lịch sử của các tệp khác.

6. Kiểm soát việc viết tài liệu

- Viết tài liệu là một quá trình ghi lại thông tin cho bất kỳ mục đích tham khảo hoặc hoạt động nào.
- Hỗ trợ người dùng, người quản lý và nhân viên CNTT.
- Tài liệu đã chuẩn bị phải được cập nhật thường xuyên để dễ dàng theo dõi tiến độ của hệ thống.

* Ưu điểm:

- Giảm thời gian ngừng hoạt động của hệ thống, cắt giảm chi phí và tăng tốc các nhiệm vụ bảo trì.
- Cung cấp mô tả rõ ràng về luồng hình thức của hệ thống hiện tại, giúp hiểu loại dữ liệu đầu vào và cách đầu ra có thể được tạo ra.
- Cung cấp cách thức giao tiếp hiệu quả giữa người dùng kỹ thuật và không kỹ thuật về hệ thống.
- Tạo điều kiện thuận lợi cho việc đào tạo người dùng mới để họ có thể dễ dàng hiểu được luồng của hệ thống.
- Giúp người sử dụng giải quyết các vấn đề như khắc phục sự cố, giúp người quản lý đưa ra các quyết định cuối cùng tốt hơn của hệ thống tổ chức.
- Cung cấp khả năng kiểm soát tốt hơn hoạt động bên trong hoặc bên ngoài của hệ thống.

* Có 4 tài liệu chính liên quan đến Thiết kế hệ thống:

- Tài liệu chương trình
- Tài liệu hệ thống
- Tài liệu vận hành
- Tài liệu người dùng

6.1. *Tài liệu chương trình*

- Mô tả các đầu vào, đầu ra và logic xử lý cho tất cả các mô-đun chương trình.
- Tài liệu này hướng dẫn các lập trình viên, những người xây dựng các mô-đun

6.2. *Tài liệu vận hành*

- Tài liệu vận hành chứa tất cả thông tin cần thiết để xử lý và phân phối đầu ra trực tuyến và đầu ra được in.
- Bao gồm các thông tin sau:
 - Chương trình, nhà phân tích hệ thống, người lập trình và nhận dạng hệ thống.
 - Lập lịch thông tin cho đầu ra được in, chẳng hạn như báo cáo, tần suất thực hiện và thời hạn.
 - Tệp đầu vào, nguồn, tệp đầu ra và đích của chúng.
 - E-mail và danh sách phân phối báo cáo.
 - Yêu cầu các biểu mẫu đặc biệt, bao gồm cả biểu mẫu trực tuyến.
 - Các thông báo lỗi và thông tin cho các nhà khai thác và các thủ tục khởi động lại.
 - Hướng dẫn đặc biệt, chẳng hạn như yêu cầu bảo mật.

6.3. *Tài liệu người dùng*

- Gồm các hướng dẫn và thông tin cho người dùng sẽ tương tác với hệ thống.
- Tài liệu hướng dẫn sử dụng có giá trị trong việc đào tạo người dùng và cho mục đích tham khảo.
- Phải rõ ràng, dễ hiểu và dễ tiếp cận cho người dùng ở mọi cấp độ.
- Tài liệu người dùng bao gồm:
 - Tổng quan về hệ thống mô tả rõ ràng tất cả các tính năng, khả năng và hạn chế chính của hệ thống.
 - Mô tả nội dung tài liệu nguồn, chuẩn bị, xử lý và mẫu.
 - Tổng quan về menu và các tùy chọn màn hình nhập dữ liệu, nội dung và hướng dẫn xử lý.
 - Ví dụ về các báo cáo được tạo thường xuyên hoặc có sẵn theo yêu cầu của người dùng, bao gồm cả các mẫu.

- Thông tin về dấu vết kiểm tra và bảo mật.
- Giải trình về trách nhiệm đối với các yêu cầu đầu vào, đầu ra hoặc xử lý cụ thể.
- Thủ tục yêu cầu thay đổi và báo cáo sự cố.
- Ví dụ về các trường hợp ngoại lệ và lỗi.
- Câu hỏi thường gặp (FAQ).
- Giải thích về cách nhận trợ giúp và các thủ tục cập nhật hướng dẫn sử dụng.

6.4. Tài liệu hệ thống

- Tài liệu hệ thống là đặc tả kỹ thuật của hệ thống thông tin (HTTT) và cách đạt được các mục tiêu của HTTT.
- Mô tả từng chương trình trong HTTT và toàn bộ HTTT.
- Mô tả các chức năng của hệ thống, cách chúng được cài đặt, mục đích của từng chương trình trong toàn bộ HTTT, trong đó có đề cập đến thứ tự thực hiện, thông tin được chuyển đến và đi từ các chương trình, cũng như luồng tổng thể của hệ thống.
- Gồm các mục từ điển dữ liệu, biểu đồ luồng dữ liệu, mô hình đối tượng, bối cảnh màn hình, tài liệu nguồn và yêu cầu hệ thống đã khởi tạo dự án.
- Hầu hết các tài liệu hệ thống được chuẩn bị trong giai đoạn phân tích hệ thống và thiết kế hệ thống.
- Trong quá trình cài đặt hệ thống, nhà phân tích phải xem lại tài liệu hệ thống để xác minh rằng tài liệu đó là đầy đủ, chính xác, cập nhật và bao gồm mọi thay đổi được thực hiện trong quá trình cài đặt.

Câu 4: Kiểu kiến trúc 3 lớp & MVC

* Model-View-Controller (MVC) là một kiến trúc phát triển ba lớp nổi tiếng được sử dụng cho việc phát triển ứng dụng web.

- Kiến trúc gồm ba lớp chính: Model, View và Controller.
 - Model (Mô hình): Các lớp model được sử dụng để cài đặt logic của các miền dữ liệu. Các lớp này được sử dụng để truy xuất, chèn hoặc cập nhật dữ liệu vào cơ sở dữ liệu liên kết với ứng dụng.
 - View (Giao diện): Các lớp view được sử dụng để chuẩn bị giao diện của ứng dụng. Người dùng sử dụng giao diện này để tương tác với ứng dụng.
 - Controller (Điều khiển): Các lớp controller được sử dụng để phản hồi các yêu cầu của người dùng. Các lớp điều khiển thực hiện các hành động mà người dùng yêu cầu. Các lớp này làm việc với các lớp mô hình và chọn lớp view phù hợp để hiển thị cho người dùng.

* Mẫu kiến trúc MVC về cơ bản là một kiến trúc ba lớp, phân tách các đặc điểm của ứng dụng:

- Lớp đầu tiên liên quan đến logic đầu vào của người dùng
- Lớp thứ hai liên quan đến logic nghiệp vụ
- Lớp thứ ba cài đặt logic giao diện người dùng.

* Mẫu MVC cho phép phát triển song song, hay **mỗi lớp của ứng dụng độc lập với nhau**. Điều này có nghĩa là ba nhà phát triển có thể làm việc đồng thời trong một ứng dụng duy nhất:

- Một nhà phát triển sẽ làm việc với logic đầu vào của người dùng (logic controller)
- Nhà phát triển khác sẽ làm việc với logic giao diện người dùng (view)
- Nhà phát triển thứ ba sẽ làm việc với logic nghiệp vụ (model)

* Khi có yêu cầu sử dụng kiến trúc MVC, chúng ta cần cân nhắc một số vấn đề sau:

- Ứng dụng có cần giao tiếp không đồng bộ ở phía backend không?
- Ứng dụng có chức năng nào gây ra việc không tải lại đầy đủ cả trang hay không? (Ví dụ: bình luận trong bài viết trên Facebook, hoặc cuộn xuống vô hạn,...)
- Thao tác với dữ liệu chủ yếu nằm ở phía máy khách (trình duyệt) hơn là phía máy chủ không?
- Cùng một kiểu dữ liệu có được chuyển đến một trang duy nhất theo nhiều cách khác nhau hay không? (khi thực hiện điều hướng)
- Ứng dụng có nhiều kết nối nhỏ để chỉnh sửa dữ liệu không? (các button và switch)

* Các ưu điểm của kiến trúc MVC:

- Giúp kiểm soát độ phức tạp của ứng dụng, thông qua việc chia ứng dụng thành 3 thành phần
- Kiến trúc MVC không sử dụng form dựa trên máy chủ, nên đây là kiến trúc lý tưởng khi ta muốn kiểm soát toàn bộ hành vi của ứng dụng
- Hỗ trợ phương pháp phát triển phục vụ kiểm thử
- Kiến trúc MVC sử dụng mẫu controller phía trước (Front controller pattern). Mẫu này xử lý nhiều yêu cầu gửi đến bằng một giao tiếp (controller) duy nhất, đồng thời cung cấp điều khiển tập trung. Do đó, chúng ta chỉ cần cấu hình 1 controller duy nhất trên web server
- Controller phía trước cho phép ta sử dụng nhiều cách định tuyến truyền thông đa dạng để thiết kế ứng dụng web.

* Các công cụ và công nghệ được sử dụng với kiến trúc MVC:

- Công cụ

- Visual Studio
- MySQL Server
- SQL Server
- MySQL Workbench
- NetBeans
- Glassfish Server
- Công nghệ
 - HTML, CSS, JQUERY, AJAX
 - Servlet và JSP sử dụng với Net bean
 - EJB (Enterprise Java bean)
 - JSTL
 - JPA
 - JDBC
 - ASP.NET MVC sử dụng với Visual Studio

Câu 5: DAO? Tại sao sử dụng? Biểu đồ lớp & code cho DAO. Xem lại phụ lục

B1 & Giải thích biểu đồ

1. Định nghĩa DAO và Lý do sử dụng

* *DAO:*

- DAO là viết tắt của Data Access Object, hay Đối tượng truy cập dữ liệu.
- DAO là một đối tượng đóng gói các thao tác truy cập dữ liệu. Trong lập trình hướng đối tượng; đây là một đối tượng cho phép hoặc trừu tượng hóa phần tử truy cập dữ liệu đến / đi vào một hệ thống dữ liệu bên ngoài.
- DAO được sử dụng để trừu tượng hóa các thao tác truy cập cơ sở dữ liệu sang bất kỳ hệ quản trị cơ sở dữ liệu nào khác hệ quản trị cơ sở dữ liệu quan hệ, ví dụ như các hệ quản trị cơ sở dữ liệu dạng NoSQL (MongoDB, Apache Cassandra, Redis, Neo4J,...)

* *Lý do sử dụng DAO:*

- Mẫu thiết kế DAO tách biệt logic persistence (logic tương tác với cơ sở dữ liệu và thao tác dữ liệu) thành 1 lớp riêng
- Dịch vụ (service) không cần quan tâm đến các thao tác truy cập cơ sở dữ liệu mức thấp. Do đó, ta dễ dàng thay đổi cơ chế persistence trong lớp DAO mà không ảnh hưởng đến các dịch vụ

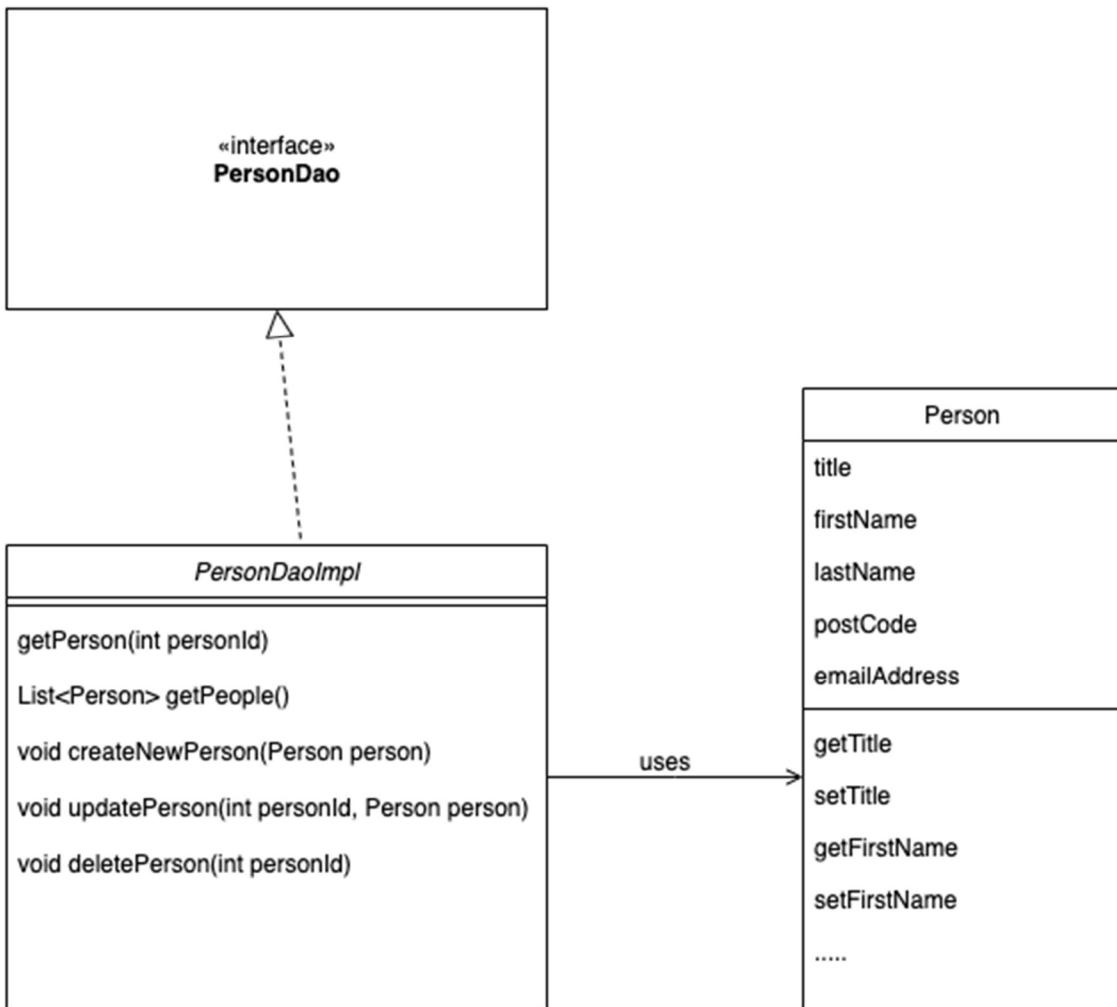
- DAO chú trọng vào việc gắn kết mức thấp giữa các thành phần khác nhau trong ứng dụng. Do đó, lớp View không có phụ thuộc (dependency) vào lớp DAO mà chỉ có lớp Service có phụ thuộc
- Do logic persistence được tách biệt hoàn toàn, ta dễ dàng viết các Kiểm thử đơn vị cho từng thành phần riêng biệt
- DAO nhấn mạnh việc làm việc với giao diện (interface) thay vì với thực thi của giao diện (implementation). Đây là cách lập trình hướng đối tượng xuất sắc.

2. Cấu trúc – Sơ đồ lớp – Code cho DAO

* Các thành phần trong mẫu thiết kế DAO:

- Lớp model: lớp này được vận chuyển từ lớp này sang lớp khác
- Giao diện (interface): cung cấp thiết kế linh hoạt
- Lớp thực thi giao diện (implementation): lớp cài đặt cụ thể của logic persistence

* Sơ đồ lớp:



- Trong biểu đồ lớp ở đây, ta thấy có 3 lớp:
 - Giao diện PersonDao
 - Lớp thực thi giao diện PersonDaoImpl
 - Đối tượng Person đóng gói dữ liệu vào một đối tượng. Đối tượng này được gọi là đối tượng giá trị (Value Object – VO), hoặc đối tượng thực thể (Entity Object – EO), hay đối tượng vận chuyển dữ liệu (Data Transfer Object – DTO)

* *Code cho lớp Person:*

```
1  public class Person {  
2      private String title;  
3      private String firstName;  
4      private String lastName;  
5      private String address1;  
6      private String address2;  
7      private String address3;  
8      private String address4;  
9      private String postCode;  
10     private String emailAddress;  
11  
12     public String getTitle() {  
13         return title;  
14     }  
15  
16     public void setTitle(String title) {  
17         this.title = title;  
18     }  
19  
20     public String getFirstName() {  
21         return firstName;  
22     }  
23  
24     public void setFirstName(String firstName) {  
25         this.firstName = firstName;  
26     }  
27  
28     public String getLastname() {  
29         return lastName;  
30     }  
31  
32     public void setLastName(String lastName) {  
33         this.lastName = lastName;  
34     }  
35  
36     public String getAddress1() {  
37         return address1;  
38     }
```

```
39
40     public void setAddress1(String address1) {
41         this.address1 = address1;
42     }
43
44     public String getAddress2() {
45         return address2;
46     }
47
48     public void setAddress2(String address2) {
49         this.address2 = address2;
50     }
51
52     public String getAddress3() {
53         return address3;
54     }
55
56     public void setAddress3(String address3) {
57         this.address3 = address3;
58     }
59
60     public String getAddress4() {
61         return address4;
62     }
63
64     public void setAddress4(String address4) {
65         this.address4 = address4;
66     }
67
68     public String getPostCode() {
69         return postCode;
70     }
71
72     public void setPostCode(String postCode) {
73         this.postCode = postCode;
74     }
75
76     public String getEmailAddress() {
77         return emailAddress;
78     }
79
```

```
80     public void setEmailAddress(String emailAddress) {  
81         this.emailAddress = emailAddress;  
82     }  
83 }
```

- Lớp Person là một đối tượng Java đơn giản, đóng gói dữ liệu đại diện cho thực thể “Person”. Lớp này gồm các trường tương ứng với các thuộc tính của Person, cùng với các getter/setter

* *Code cho giao diện PersonDao:*

- Giao diện PersonDao xác định danh sách các thao tác mà lớp thực thi DAO sẽ cài đặt
- Giao diện này còn có tên khác là Repository

```
1  public interface PersonDao {  
2      Person getPerson(int personId);  
3      List<Person> getPeople();  
4      void createNewPerson(Person person);  
5      void updatePerson(int personId, Person person);  
6      void deletePerson(int personId);  
7  }
```

* *Code cho lớp thực thi giao diện PersonDaoImpl:*

- Mỗi phương thức trong DAO sẽ chứa code thực thi cho phương thức tương ứng.

```

1  public class PersonDaoImpl implements PersonDao {
2
3      public Person getPerson(int personId) {
4          DBConnection con = null;
5          Person person = new Person();
6          String sql = "select * from person p Where p.person_id = ?";
7
8          try {
9              con = getDBConnection();
10             con.prepareStatement(sql);
11             con.setInt(1, personId);
12             con.executeQuery();
13
14             if (con.next()) {
15                 person.setPersonTitle(con.getInt("Title"));
16                 person.setPersonFirstName(con.getString("First_Name"));
17                 person.setPersonLastName(con.getString("Last_Name"));
18                 person.setAddress1(con.getString("Address_1"));
19                 person.setAddress2(con.getString("Address_2"));
20                 person.setAddress3(con.getString("Address_3"));
21                 person.setAddress4(con.getString("Address_4"));
22                 person.setPostcode(con.getString("Postcode"));
23                 person.setEmail(con.getString("Email_Address"));
24             }
25         } catch (Exception ex) {
26             LOGGER.error(ex);
27             throw new BusinessException("DB Error", ex.getMessage());
28         } finally {
29             try {
30                 con.close();
31             } catch (Exception ex) {
32                 LOGGER.error(ex);
33             }
34         }
35
36         return person;
37     }
38

```

```
39     public List<Person> getPeople() {
40         DBConnection con = null;
41         String sql = "select * from person;";
42         List<Person> people = new ArrayList<>();
43
44         try {
45             con = getDBConnection();
46             con.prepareStatement(sql);
47             con.setInt(1, personId);
48             con.executeQuery();
49
50             while (con.next()) {
51                 Person person = new Person();
52                 person.setPersonTitle(con.getInt("Title"));
53                 person.setPersonFirstName(con.getString("First_Name"));
54                 person.setPersonLastName(con.getString("Last_Name"));
55                 person.setAddress1(con.getString("Address_1"));
56                 person.setAddress2(con.getString("Address_2"));
57                 person.setAddress3(con.getString("Address_3"));
58                 person.setAddress4(con.getString("Address_4"));
59                 person.setPostcode(con.getString("Postcode"));
60                 person.setEmail(con.getString("Email_Address"));
61                 people.add(person);
62             }
63         } catch (Exception ex) {
64             LOGGER.error(ex);
65             throw new BusinessException("DB Error", ex.getMessage());
66         } finally {
67             try {
68                 con.close();
69             } catch (Exception ex) {
70                 LOGGER.error(ex);
71             }
72         }
73         return people;
74     }
75
76     public void createNewPerson(Person person) {
77         DBConnection con = null;
78
79         try {
```

```

80         con = getDBConnection();
81         stmt = con.createStatement();
82         String sql = "insert into person values (" +
83             person.getTitle() + "," +
84             person.getFirstName() + "," +
85             person.getLastName() + "," +
86             person.getAddress1() + "," +
87             person.getAddress2() + "," +
88             person.getAddress3() + "," +
89             person.getAddress4() + "," +
90             person.getPostCode() + "," +
91             person.getEmailAddress() +
92             ");";
93         stmt.executeUpdate(sql);
94
95     } catch (Exception ex) {
96         LOGGER.error(ex);
97         throw new BusinessException("DB Error", ex.getMessage());
98     } finally {
99         try {
100             con.close();
101         } catch (Exception ex) {
102             LOGGER.error(ex);
103         }
104     }
105 }
106 }
107
108 public void updatePerson(int personId, Person person) {
109     DBConnection con = null;
110     try {
111         con = getDBConnection();
112         stmt = con.createStatement();
113         String sql = "update person set title=" + personToUpdate.getTitle() + "," +
114             "First_Name=" + personToUpdate.getFirstName() + "," +
115             "Last_Name=" + personToUpdate.getLastName() + "," +
116             "Address_1=" + personToUpdate.getAddress1() + "," +
117             "Address_2=" + personToUpdate.getAddress2() + "," +
118             "Address_3=" + personToUpdate.getAddress3() + "," +
119             "Address_4=" + personToUpdate.getAddress4() + "," +
120             "PostCode=" + personToUpdate.getPostCode() + ","

```

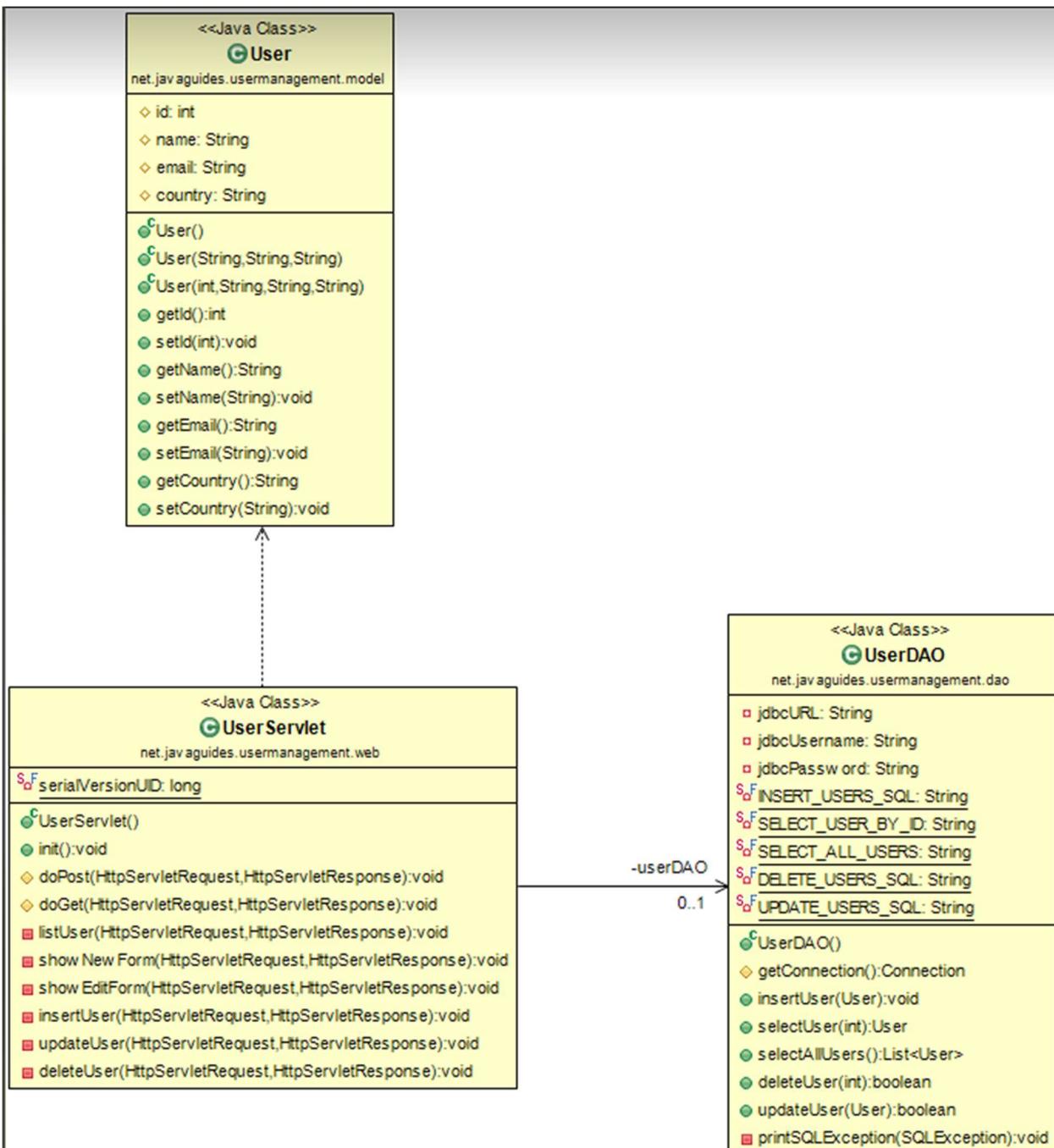
```

121             "Email_Address=" + personToUpdate.getEmailAddress() +
122             " where person.person_id = " + personId;
123         stmt.executeUpdate(sql);
124
125     } catch (Exception ex) {
126         LOGGER.error(ex);
127         throw new BusinessException("DB Error", ex.getMessage());
128     } finally {
129         try {
130             con.close();
131         } catch (Exception ex) {
132             LOGGER.error(ex);
133         }
134     }
135 }
136
137 public void deletePerson(int personId) {
138     DBConnection con = null;
139     String sql = "delete from person p where p.person_id = ?;";
140
141     try {
142         con = getDBConnection();
143         con.prepareStatement(sql);
144         con.setInt(1, personId);
145         con.executeQuery();
146
147     } catch (Exception ex) {
148         LOGGER.error(ex);
149         throw new BusinessException("DB Error", ex.getMessage());
150     } finally {
151         try {
152             con.close();
153         } catch (Exception ex) {
154             LOGGER.error(ex);
155         }
156     }
157 }
158 }
```

- Trong đoạn code trên:
 - Đầu tiên, chúng ta thiết lập kết nối tới cơ sở dữ liệu

- Sau đó, ta dùng code để gửi truy vấn SQL hoặc cập nhật SQL đến cơ sở dữ liệu. Code này sử dụng thư viện JDBC nội bộ của Java để kết nối với cơ sở dữ liệu. JDBC là thư viện nội bộ của Java cho phép kết nối với cơ sở dữ liệu quan hệ.
- Ta sử dụng try-catch để xử lý lỗi

3. Biểu đồ phụ lục B1 và giải thích



- Lớp User là lớp thực thể, hay chính là đối tượng thực thể (Entity Object), chứa dữ liệu được đóng gói của thực thể User. Lớp này được vận chuyển qua lại giữa các lớp khác.
- Lớp UserDao là lớp thực thi logic persistence, hay thực thi các thao tác với cơ sở dữ liệu.
- Lớp UserServlet có quan hệ Dependency với lớp User và quan hệ Association 1 chiều 0-1 với lớp UserDao, điều này đồng nghĩa với việc lớp UserServlet sẽ có:
 - 1 thuộc tính là thẻ hiện của lớp UserDao
 - 1 hoặc nhiều phương thức sử dụng thẻ hiện của lớp User
- Trong mô hình MVC, lớp UserServlet đóng vai trò là 1 controller tiếp nhận request từ máy khách gửi lên, thực hiện 1 số thao tác lấy dữ liệu từ request, sau đó gọi đến thẻ hiện của lớp UserDao để thực hiện xử lý nghiệp vụ. Trong quá trình đó, lớp UserServlet sẽ sử dụng thẻ hiện của lớp User để đóng gói và vận chuyển dữ liệu. Sau khi thực hiện xong các xử lý nghiệp vụ, UserServlet sẽ thực hiện chuyển tiếp request / response đến các lớp View (hay trang JSP). Các lớp View được sinh ra và chuyển cùng với response đến cho máy khách.

Câu 6: Thiết kế cơ sở dữ liệu

1. Quy trình thiết kế cơ sở dữ liệu

* Ưu điểm của cơ sở dữ liệu có cấu trúc tốt:

- Tiết kiệm dung lượng ổ đĩa nhờ loại bỏ dữ liệu dư thừa.
- Duy trì độ chính xác và tính toàn vẹn của dữ liệu.
- Cung cấp truy cập vào dữ liệu theo những cách hữu ích.

* Quy trình thiết kế một cơ sở dữ liệu hiệu quả và hữu ích gồm các giai đoạn sau:

- Phân tích yêu cầu hoặc xác định mục đích của cơ sở dữ liệu.
- Tổ chức dữ liệu thành các bảng
- Xác định khóa chính và phân tích các quan hệ
- Thực hiện chuẩn hóa các bảng

2. Phân tích yêu cầu: Xác định mục đích của CSDL

- Xác định đúng mục đích của CSDL sẽ trang bị thêm thông tin trong suốt quá trình thiết kế.
 - Ví dụ: Khi tạo cơ sở dữ liệu cho thư viện công cộng, cần xem xét các cách mà cả khách hàng quen và thủ thư sẽ cần để truy cập dữ liệu.
- Một số cách để thu thập thông tin trước khi tạo CSDL:

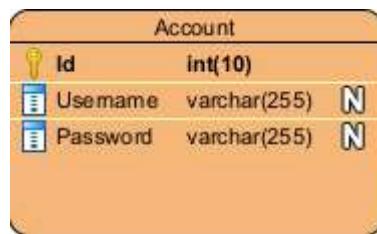
- Phỏng vấn những người sẽ sử dụng CSDL
- Phân tích các form nghiệp vụ như hóa đơn, bảng chấm công, khảo sát..
- Tham khảo các hệ thống dữ liệu hiện có (bao gồm các tệp vật lý và kỹ thuật số)
- Khi đã biết cơ sở dữ liệu sẽ bao gồm những loại dữ liệu nào, dữ liệu đó đến từ đâu và nó sẽ được sử dụng như thế nào, ta đã sẵn sàng để bắt đầu lập kế hoạch cho CSDL thực tế.

3. Cấu trúc cơ sở dữ liệu: Thành phần cơ bản của CSDL

- Bước tiếp theo là trình bày trực quan cơ sở dữ liệu. Do đó, ta cần hiểu chính xác cấu trúc của CSDL quan hệ.
- Ta cần **chuyển đổi danh sách dữ liệu thành bảng** và **tạo bảng cho từng loại thực thể**, chẳng hạn như sản phẩm, khách hàng và đơn đặt hàng.
- Ví dụ:

CustomerId	Name	Age	Address
1	Nguyen An	22	TP Ha Noi
2	Nguyen Duong	22	TP Ho Chi Minh

- Để giữ cho dữ liệu nhất quán từ bản ghi này sang bản ghi tiếp theo, cần **gán kiểu dữ liệu thích hợp** cho mỗi cột. Các kiểu dữ liệu phổ biến bao gồm: CHAR, VARCHAR, TEXT, INT, FLOAT, DOUBLE, BLOB (dữ liệu nhị phân)



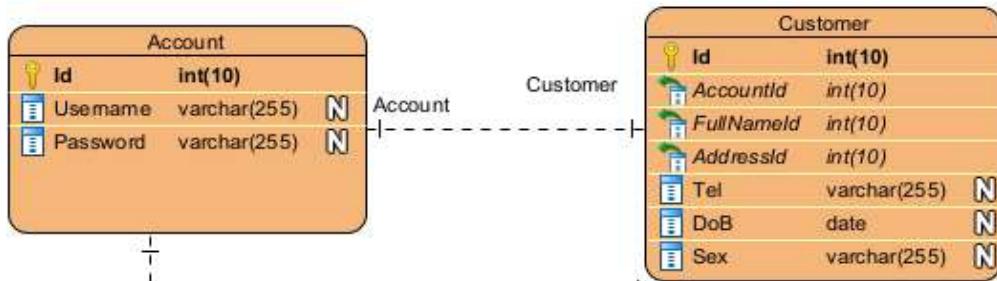
- Sau đó, quyết định thuộc tính nào hoặc các thuộc tính nào sẽ đóng vai trò là **khóa chính** cho mỗi bảng (nếu có). Khóa chính (PK) là số nhận dạng duy nhất cho một thực thể nhất định, có nghĩa là bạn có thể chọn ra một khách hàng chính xác ngay cả khi bạn chỉ biết giá trị đó.
- Ta cũng có thể sử dụng nhiều trường kết hợp làm khóa chính, đây gọi là khóa tổng hợp (Composite Key).

4. Tạo quan hệ giữa các thực thể

- Mỗi thực thể có thể có quan hệ với mọi thực thể khác. Các quan hệ thường rơi vào một trong 3 loại sau đây:

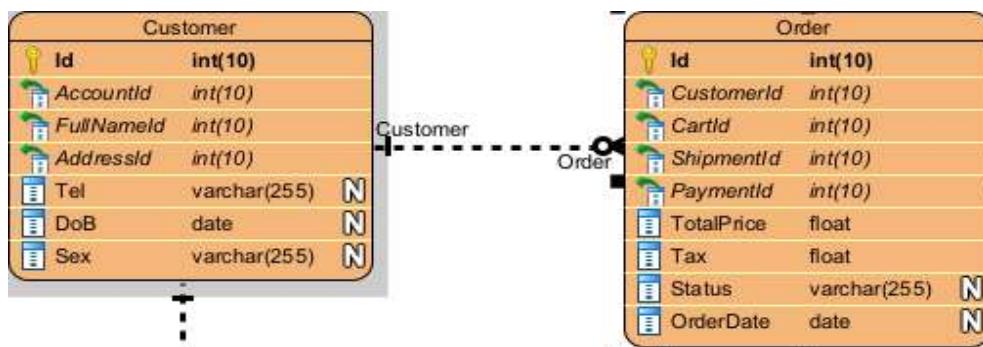
4.1. Quan hệ một-một

- Khi chỉ có một bản sao của Thực thể A cho mỗi bản sao của Thực thể B, chúng được cho là có quan hệ một-một (1:1)
- Mỗi quan hệ 1: 1 thường chỉ ra rằng bạn nên kết hợp dữ liệu của hai bảng thành một bảng duy nhất.



4.2. Quan hệ một-nhiều

- Quan hệ này xảy ra khi một bản ghi trong một bảng được liên kết với nhiều mục dữ liệu trong một bảng khác.
- Ví dụ: một khách hàng có thể đã đặt nhiều đơn đặt hàng hoặc một khách hàng quen có thể mượn nhiều cuốn sách từ thư viện cùng một lúc.
- Để triển khai quan hệ 1:M, chỉ cần thêm khóa chính từ phía "1" của mối quan hệ làm thuộc tính trong bảng kia. Khi một khóa chính được liệt kê trong một bảng khác theo cách này, nó được gọi là **khóa ngoại**. Bảng ở phía "1" của mối quan hệ được coi là **bảng mẹ** đối với bảng con ở phía kia.



4.3. Quan hệ nhiều-nhiều

- Khi nhiều thực thể từ một bảng có thể liên kết với nhiều thực thể trong bảng khác, chúng được cho là có quan hệ nhiều-nhiều (M:N). Điều này có thể xảy ra trong

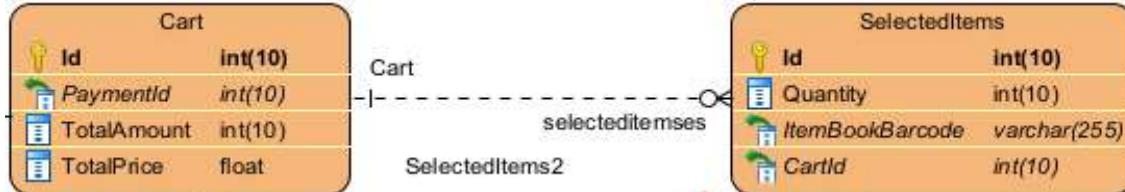
trường hợp Cart và ItemBook, vì một Cart có thể có nhiều ItemBook và một ItemBook có thể thuộc nhiều Cart khác nhau.

- Không thể trực tiếp triển khai loại quan hệ này trong CSDL. Thay vào đó phải chia nó thành hai mối quan hệ một-nhiều.
- Tạo một thực thể mới giữa hai bảng đó. Gọi thực thể mới đó là SelectedItems.



4.4. *Bắt buộc hay không?*

- Một cách khác để phân tích quan hệ là xem xét bên nào của mối quan hệ phải tồn tại để bên kia tồn tại. Bên không bắt buộc có thể được đánh dấu bằng một vòng tròn trên dòng có dấu gạch ngang.
- Ví dụ, một Cart phải tồn tại để có thể có SelectedItems nhưng điều ngược lại là không đúng.
- Hai thực thể có thể phụ thuộc lẫn nhau (một thực thể không thể tồn tại nếu không có thực thể kia).



4.5. *Quan hệ đệ quy*

- Đôi khi, một bảng chỉ trả vào chính nó. Ví dụ: Một bảng nhân viên có thể có thuộc tính "Quản lý" chỉ đến một cá nhân khác trong cùng bảng đó. Đây được gọi là quan hệ đệ quy.

5. Chuẩn hóa CSDL

- Khi đã có thiết kế sơ bộ cho CSDL, ta có thể áp dụng các quy tắc chuẩn hóa để đảm bảo các bảng được cấu trúc chính xác.

5.1. *Dạng chuẩn đầu tiên (1NF)*

- Dạng chuẩn đầu tiên (viết tắt là 1NF) yêu cầu **mỗi ô trong bảng chỉ có thể có một giá trị**, không bao giờ là danh sách các giá trị.

- Có thể tách dữ liệu đó thành các cột bổ sung, nhưng điều đó cũng trái với quy tắc: một bảng có nhóm các thuộc tính lặp lại hoặc có liên quan chặt chẽ sẽ không đáp ứng dạng chuẩn đầu tiên.
- Ví dụ 2 bảng dưới đây không đạt chuẩn 1NF:

ProductID	Color	Price	Products
1	brown, yellow	\$15	Color1
2	red, green	\$13	Color2
3	blue, orange	\$11	Color3 Price

- Thay vào đó, ta sẽ chia dữ liệu thành nhiều bảng hoặc nhiều bản ghi cho đến khi mỗi ô chỉ chứa một giá trị và không có cột thừa. Tại thời điểm đó, dữ liệu được cho là nguyên tử, hoặc được chia nhỏ thành kích thước hữu ích nhỏ nhất.
- Đối với bảng trên, có thể tạo một bảng bổ sung được gọi là "Sales details" sẽ khớp các sản phẩm cụ thể với doanh số bán hàng. Khi đó "Sales" sẽ có mối quan hệ 1:M với "Sales details".

5.2. Dạng chuẩn thứ hai

- Dạng chuẩn thứ hai (2NF) yêu cầu **mỗi thuộc tính phải phụ thuộc hoàn toàn vào toàn bộ khóa chính**. Điều này nghĩa là mỗi thuộc tính nên phụ thuộc trực tiếp vào khóa chính, thay vì gián tiếp thông qua một số thuộc tính khác.
- Ví dụ sau sẽ không đáp ứng được dạng chuẩn thứ hai, vì thuộc tính "Product Name" phụ thuộc vào "Product ID" và không phụ thuộc vào số của đơn đặt hàng "Order number":
 - Order number (khóa chính)
 - Product ID (khóa chính)
 - Product name

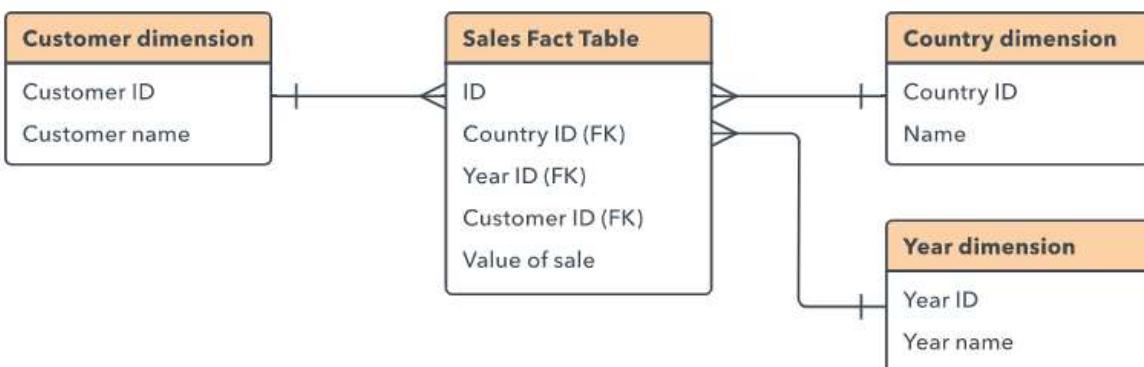
5.3. Dạng chuẩn thứ ba

- Dạng chuẩn thứ ba (3NF) bổ sung vào các quy tắc của dạng chuẩn 1NF và 2NF yêu cầu rằng **mọi cột không phải khóa cần phải độc lập với mọi cột khác**. Nếu việc thay đổi một giá trị trong một cột không phải là khóa làm cho một giá trị khác thay đổi, thì bảng đó không đáp ứng dạng chuẩn thứ ba.
- Ví dụ dưới đây: cột "Tax" trực tiếp phụ thuộc vào tổng giá "Price" của đơn đặt hàng. Như vậy không đạt 3NF.

Order	Price	Tax
14325	\$40.99	\$2.05
14326	\$13.73	\$.69
14327	\$24.15	\$1.21

6. Dữ liệu đa chiều

- Một số người dùng có thể muốn truy cập **nhiều chiều của một loại dữ liệu**, đặc biệt là trong cơ sở dữ liệu OLAP.
- Ví dụ: Họ có thể muốn biết doanh số bán hàng theo khách hàng, tiểu bang và tháng. Trong trường hợp này, nên tạo một bảng chân lý trung tâm mà các bảng khách hàng, tiểu bang và tháng khác có thể tham chiếu đến như sau:



7. Quy tắc toàn vẹn dữ liệu

- Quy tắc toàn vẹn thực thể** nói rằng khóa chính không bao giờ có thể là NULL. Nếu khóa được tạo thành từ nhiều cột, không cột nào trong số chúng có thể là NULL. Nếu không, nó có thể không xác định duy nhất bản ghi.
- Quy tắc toàn vẹn tham chiếu** yêu cầu mỗi khóa ngoại được liệt kê trong một bảng phải được khớp với một khóa chính trong bảng mà nó tham chiếu. Nếu khóa chính thay đổi hoặc bị xóa, những thay đổi đó sẽ cần được thực hiện ở bất kỳ nơi nào khóa đó được tham chiếu trong cơ sở dữ liệu.
- Các **quy tắc toàn vẹn logic nghiệp vụ** đảm bảo rằng dữ liệu phù hợp với các tham số logic nhất định. Ví dụ, một cuộc hẹn sẽ phải ở trong giờ làm việc bình thường.

8. Thêm chỉ mục và view

- Thêm một chỉ mục cho phép người dùng tìm các bản ghi nhanh hơn. Thay vì sắp xếp lại cho từng truy vấn, hệ thống có thể truy cập các bản ghi theo thứ tự được chỉ định bởi chỉ mục.
- Mặc dù các chỉ mục tăng tốc độ truy xuất dữ liệu, nhưng chúng có thể làm chậm việc chèn, cập nhật và xóa, vì chỉ mục phải được xây dựng lại bất cứ khi nào bản ghi được thay đổi.
- Một view chỉ đơn giản là một truy vấn dữ liệu được lưu lại. Chúng có thể kết hợp dữ liệu từ nhiều bảng một cách hữu ích hoặc hiển thị một phần của bảng.

9. Thuộc tính mở rộng

- Khi đã hoàn thành bộ cục cơ bản của CSDL, ta có thể tinh chỉnh CSDL với các thuộc tính mở rộng, chẳng hạn như văn bản hướng dẫn, mặt nạ nhập và quy tắc định dạng áp dụng cho một lược đồ, dạng xem hoặc cột cụ thể
- Các quy tắc này được lưu trữ trong chính CSDL, việc trình bày dữ liệu sẽ nhất quán giữa nhiều chương trình cùng truy cập dữ liệu.

10. SQL và UML

- Ngôn ngữ mô hình hóa hợp nhất (UML) là một cách trực quan khác để diễn đạt các hệ thống phức tạp được tạo ra bằng ngôn ngữ hướng đối tượng.
- Một số khái niệm ở đây được biết đến trong UML bằng các tên khác nhau. Ví dụ, một thực thể được gọi là một lớp trong UML.

11. Hệ quản trị CSDL

- Các hệ quản trị CSDL phổ biến gồm:
 - Oracle DB
 - MySQL
 - Microsoft SQL Server
 - PostgreSQL
 - DB2 của IBM

Câu 7: Thiết kế giao diện

1. UI & Thiết kế UI

- UI (User Interface): Giao diện người dùng là nơi tương tác giữa người dùng và thiết bị điện tử
- Thiết kế giao diện người dùng (UI Design) là quy trình mà các nhà thiết kế sử dụng để xây dựng giao diện (text, nút bấm, icon, màu sắc, khoảng cách, hình ảnh, màn hình ứng dụng...) trong phần mềm hoặc thiết bị máy tính, tập trung vào bối

ngoài hoặc style. Các nhà thiết kế hướng tới việc tạo ra các giao diện mà người dùng cảm thấy dễ sử dụng và hài lòng.

- 4 dạng của giao diện:

- Giao diện dòng lệnh (CLI – Command Line Interface): Người dùng tương tác với máy tính thông qua cửa sổ dòng lệnh
- Giao diện người dùng đồ họa (GUI – Graphical User Interface): Người dùng tương tác với các biểu diễn trực quan trên bảng điều khiển kỹ thuật số.
- Giao diện điều khiển bằng giọng nói (VUI – Voice User Interface): Người dùng tương tác với các giao diện này thông qua giọng nói của họ. Hầu hết các trợ lý thông minh (như Siri trên iPhone và Alexa trên các thiết bị Amazon) đều là VUI.
- Giao diện dựa trên cử chỉ (Gesture-based Interface): Người dùng tham gia vào không gian thiết kế 3D thông qua các chuyển động của cơ thể. Ví dụ như trong các trò chơi thực tế ảo (VR)

2. Một số lưu ý để tạo ra giao diện người dùng tốt nhất

- Người dùng **đánh giá thiết kế nhanh chóng**, cũng như quan tâm **khả năng sử dụng và khả năng thích ứng**.
 - Người dùng không quan tâm đến thiết kế mà quan tâm đến việc hoàn thành nhiệm vụ của họ một cách dễ dàng và với nỗ lực tối thiểu.
 - Do đó, thiết kế cần phải "vô hình" để người dùng không nên tập trung vào nó mà tập trung vào phải hoàn thành nhiệm vụ
 - Vì vậy, cần hiểu bối cảnh và luồng tác vụ của người dùng, để tinh chỉnh giao diện người dùng tốt nhất, trực quan nhất mang lại trải nghiệm liền mạch.
- Giao diện người dùng cũng phải **thú vị** (hoặc ít nhất là thỏa mãn và không gây thất vọng).
 - Khi thiết kế dự đoán được nhu cầu của người dùng, họ có thể tận hưởng những trải nghiệm được cá nhân hóa và phong phú hơn. Người dùng sẽ tiếp tục quay trở lại sử dụng ứng dụng.
 - Có thể sử dụng một số yếu tố trò chơi ở những chỗ thích hợp để làm thiết kế thú vị hơn
- Giao diện người dùng phải **truyền đạt các giá trị thương hiệu** và **củng cố lòng tin của người dùng**.
 - Thiết kế tốt là thiết kế có cảm xúc. Người dùng liên kết cảm xúc tốt với các thương hiệu phù hợp với họ ở mọi cấp độ và giữ cho sự kỳ diệu của những trải nghiệm thú vị, liền mạch luôn tồn tại.

3. Cách tạo giao diện người dùng tuyệt vời

- Các nguyên tắc để tạo giao diện người dùng tuyệt vời:
 - Làm cho các nút và các yếu tố phổ biến khác hoạt động theo cách có thể đoán trước được (bao gồm các phản hồi như chum để thu phóng), để người dùng có thể sử dụng chúng ở mọi nơi một cách vô thíc. Biểu mẫu nên tuân theo chức năng.
 - Duy trì khả năng khám phá cao. Ghi nhãn rõ ràng các biểu tượng và khiến chúng gây chú ý. Ví dụ: ta có thể đỗ bóng cho các nút.
 - Giữ giao diện đơn giản (chỉ với các yếu tố giúp phục vụ mục đích của người dùng) và tạo cảm giác "vô hình".
 - Tôn trọng mắt nhìn và sự chú ý của người dùng về bối cảnh. Tập trung vào phân cấp giao diện và khả năng đọc của người dùng:
 - Sử dụng căn chỉnh thích hợp. Thông thường, ta sẽ chọn căn chỉnh cạnh (trên trung tâm).
 - Thu hút sự chú ý đến các tính năng chính bằng cách sử dụng:
 - Màu sắc, độ sáng và độ tương phản. Tránh dùng quá nhiều màu sắc hoặc nút.
 - Văn bản thông qua các kích thước phông chữ, kiểu in đậm / độ dày của chữ, in nghiêng, viết hoa và khoảng cách giữa các chữ cái. Người dùng nên hiểu nghĩa của văn bản chỉ bằng cách đọc lướt.
 - Giảm thiểu số lượng hành động để thực hiện các tác vụ nhưng tập trung vào một chức năng chính trên mỗi trang. Hướng dẫn người dùng bằng cách chỉ ra các hành động ưu tiên. Đơn giản hóa các nhiệm vụ phức tạp bằng cách thông tin đến người dùng một cách rõ ràng.
 - Đặt điều khiển gần các đối tượng mà người dùng muôn điều khiển. Ví dụ, ta nên đặt một nút để gửi biểu mẫu ở gần biểu mẫu.
 - Thông báo cho người dùng về các phản hồi / hành động của hệ thống bằng phản hồi như rung, thông báo.
 - Sử dụng các mẫu thiết kế giao diện người dùng thích hợp để giúp hướng dẫn người dùng và giảm bớt gánh nặng (ví dụ: điền trước các biểu mẫu). Cẩn thận với việc sử dụng các mẫu tối và khó nhìn thấy
 - Duy trì tính nhất quán của thương hiệu.
 - Luôn cung cấp các bước tiếp theo mà người dùng có thể suy luận một cách tự nhiên, bất kể bối cảnh của họ là gì.

4. Các trường hợp thiết kế giao diện người dùng tệ:

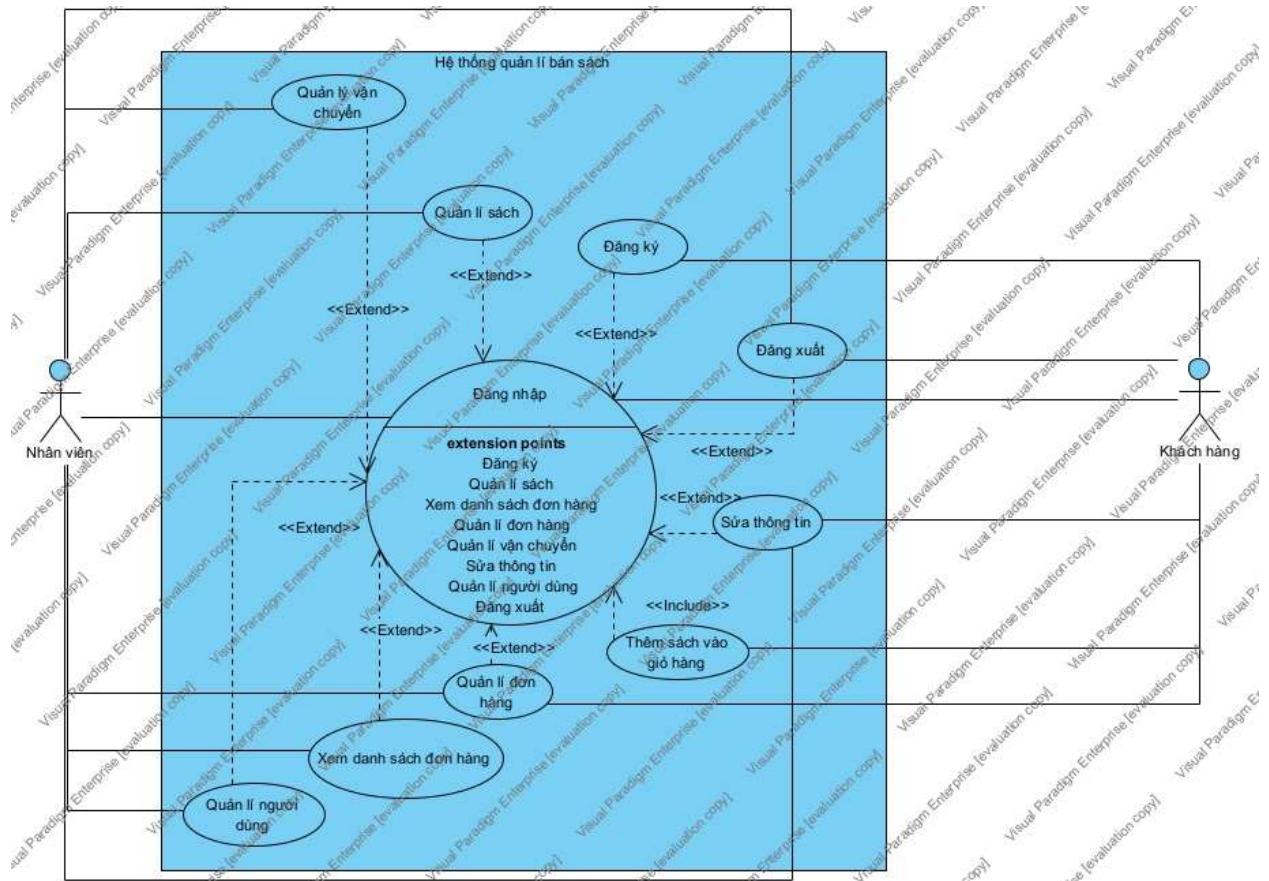
- Không tạo được ấn tượng cho khách hàng
- Gây khó khăn cho quá trình tham khảo
- Không có yếu tố dẫn dắt, thu hút người dùng

- Form điền thông tin rắc rối khiến người dùng cảm thấy khó đăng ký
- Text khó đọc khiến cho người dùng khó tham khảo và tìm thấy thông tin họ cần
- Sử dụng nhiều icon lạ, không đồng bộ gây bối rối cho quá trình sử dụng

PHẦN 2: DỰ ÁN – CỬA HÀNG SÁCH TRỰC TUYẾN

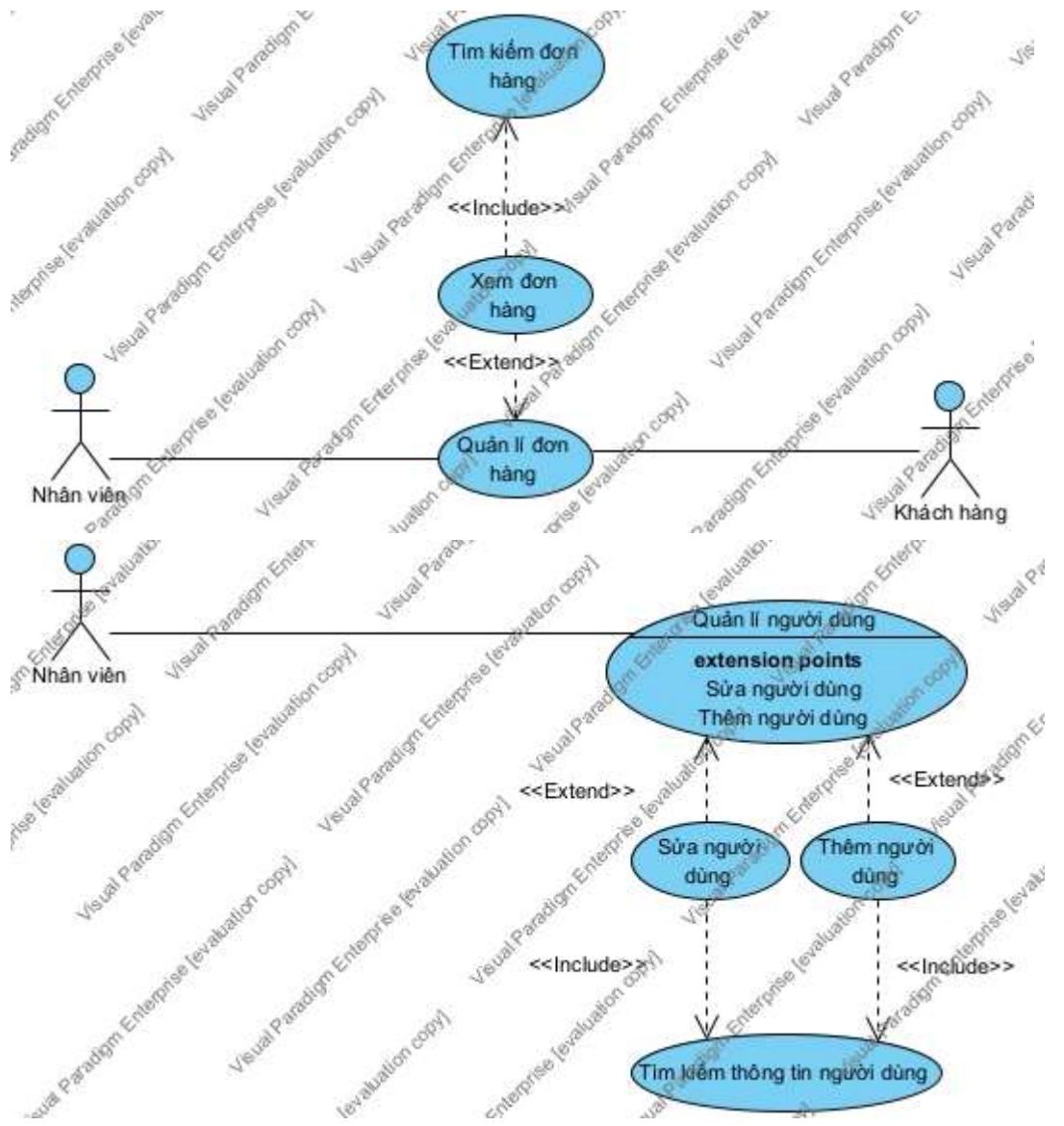
Câu 1: Vẽ biểu đồ use case với mức thông tin chi tiết

- Biểu đồ use case tổng quát:

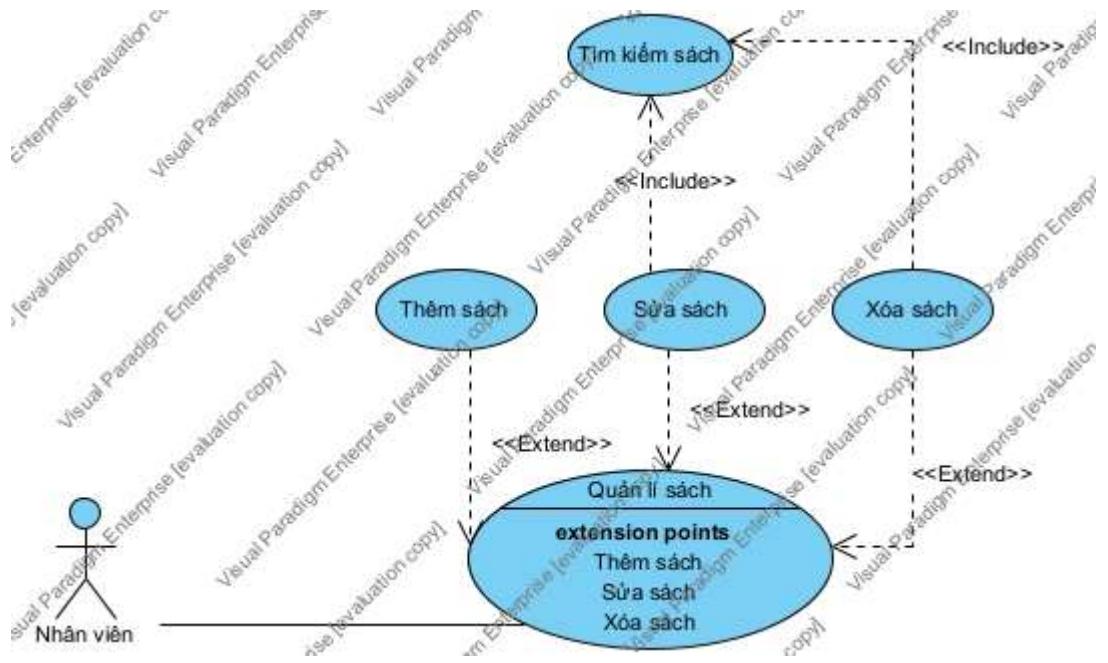


- Biểu đồ các use case phân rã:

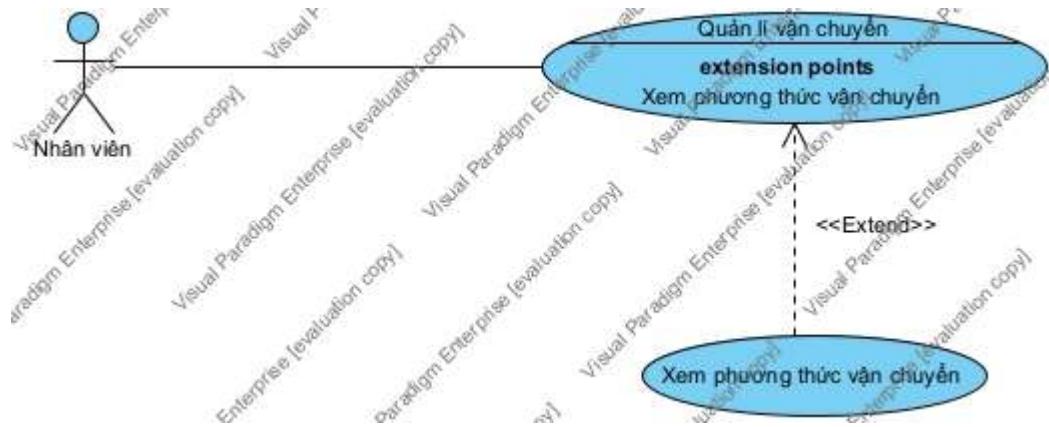
- Quản lý người dùng:



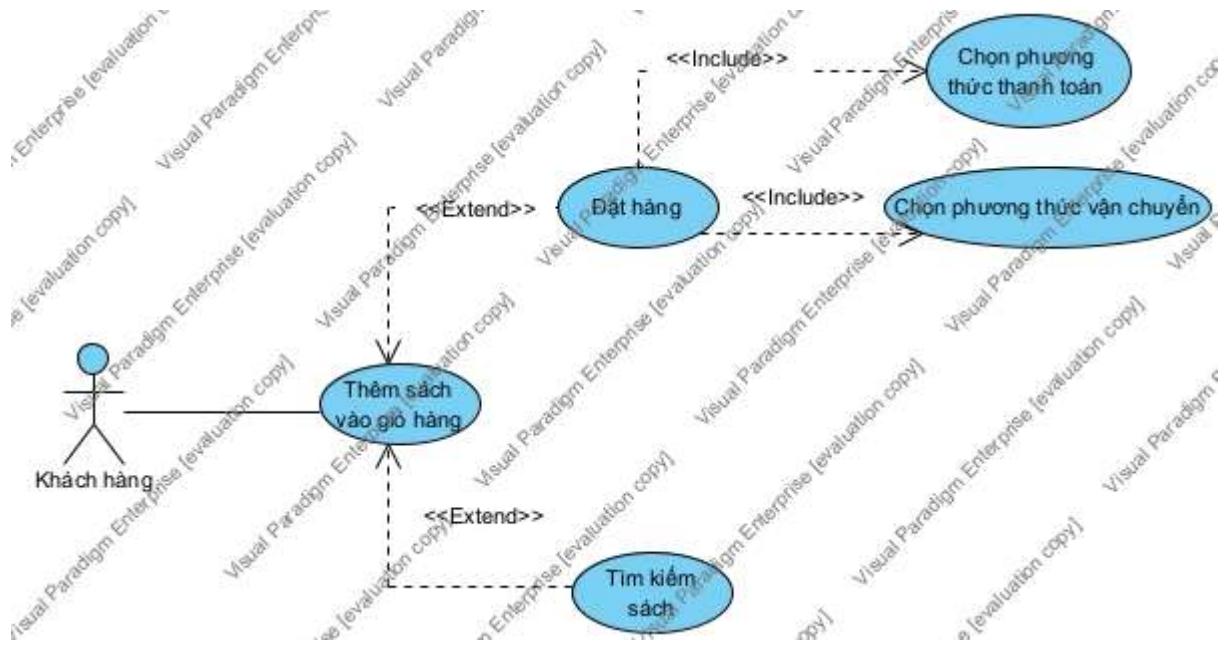
- Quản lý sách:



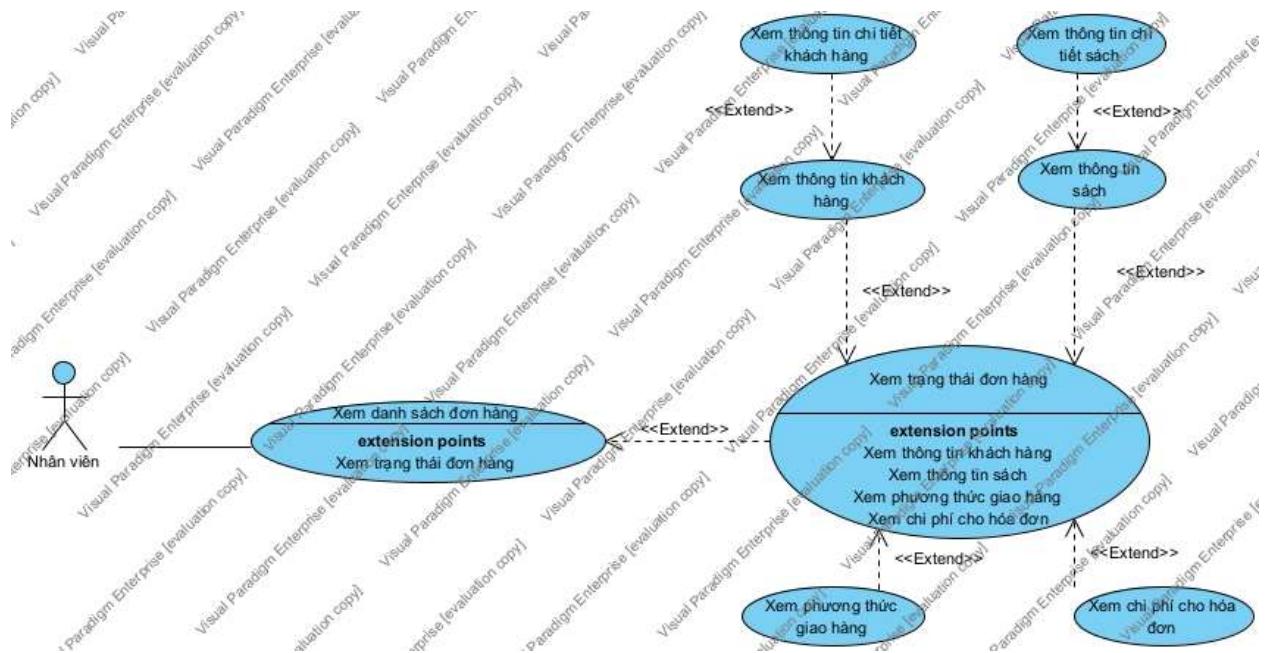
- Quản lý vận chuyển:



- Thêm sách vào giỏ hàng:

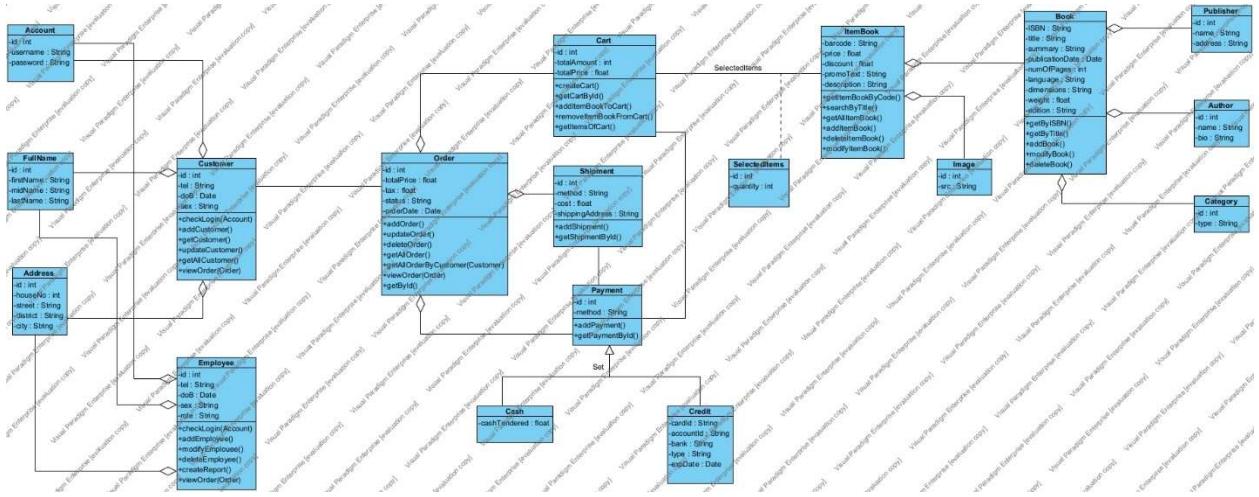


- Xem danh sách đơn hàng:



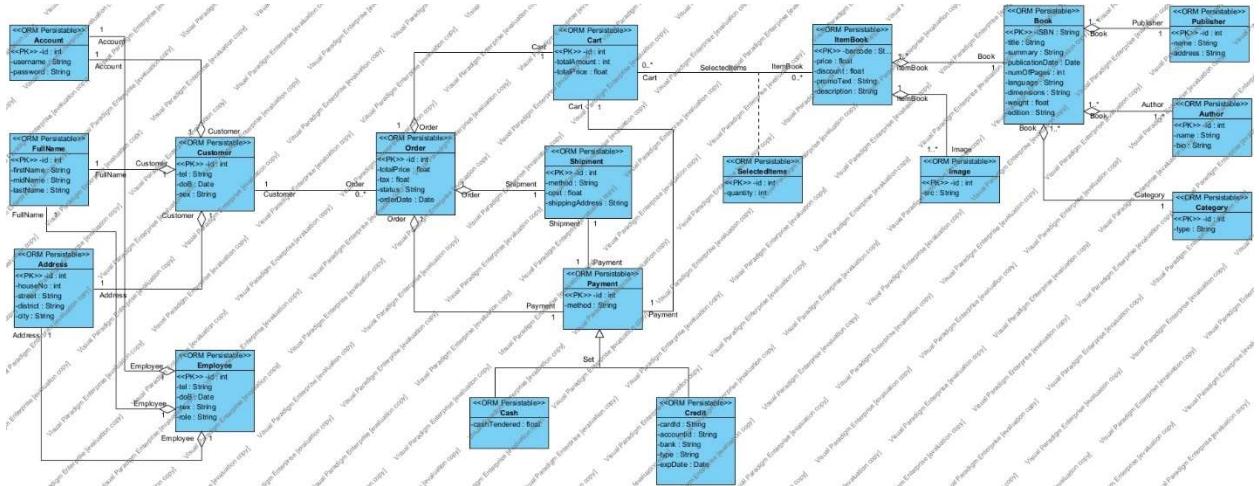
Câu 2: Xây dựng biểu đồ lớp phân tích (Class diagram analysis) với một số phương thức chính

- Biểu đồ lớp phân tích:

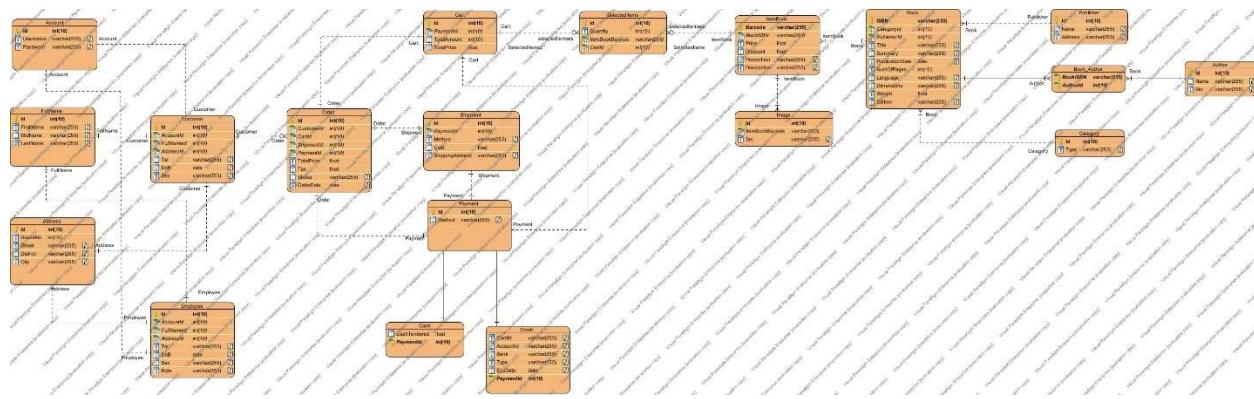


Câu 3: Xây dựng biểu đồ lớp cho mô hình dữ liệu (Class diagram model), sinh mô hình dữ liệu và cơ sở dữ liệu trong MySQL

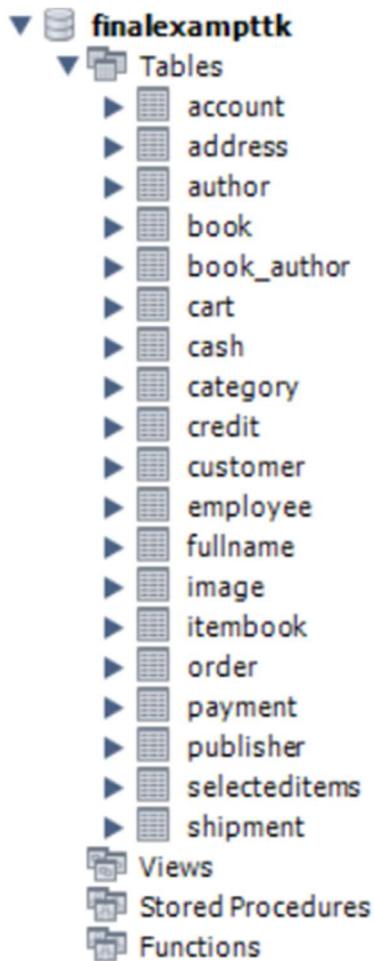
- Biểu đồ lớp cho mô hình dữ liệu:



- Mô hình dữ liệu (Biểu đồ thực thể-quan hệ - ERD):

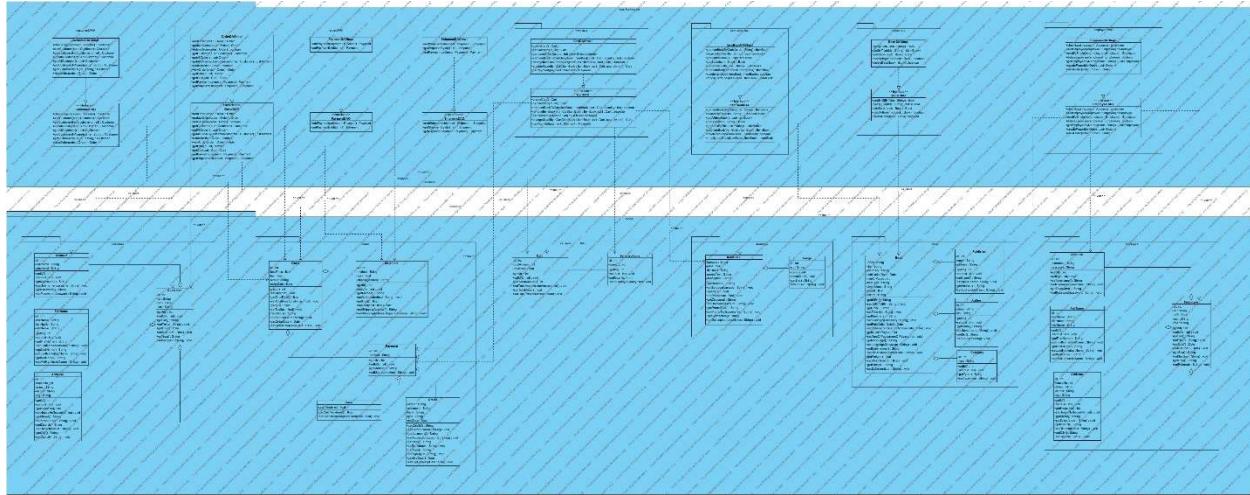


- Cơ sở dữ liệu sinh ra trong MySQL:



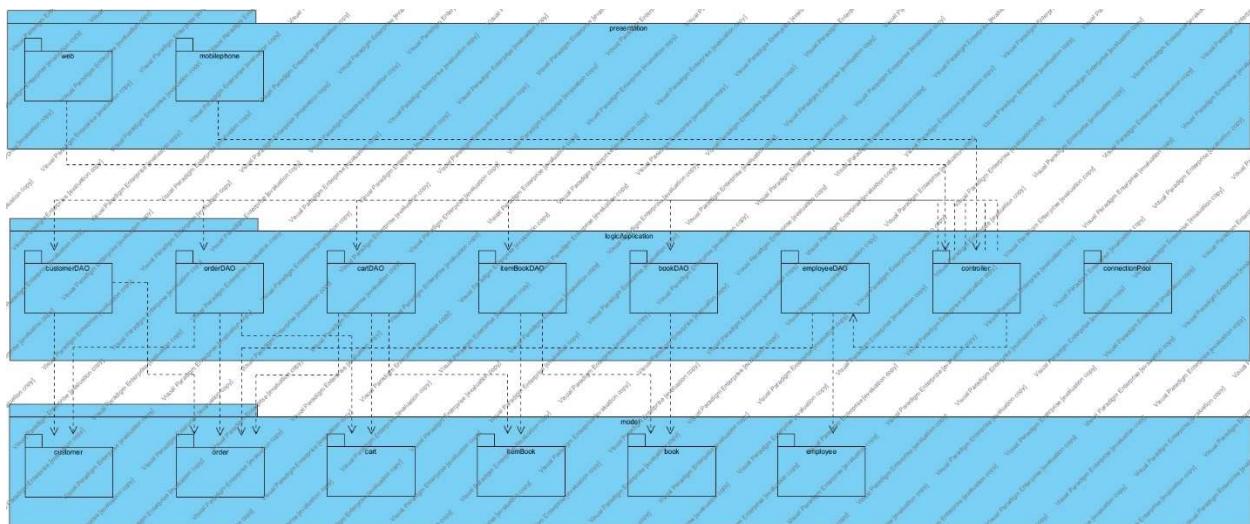
Câu 4 và câu 5: Xây dựng biểu đồ lớp thiết kế (Class diagram design), sử dụng mẫu DAO để thiết kế lại các lớp trong hệ thống

- Biểu đồ lớp thiết kế sử dụng mẫu thiết kế DAO:

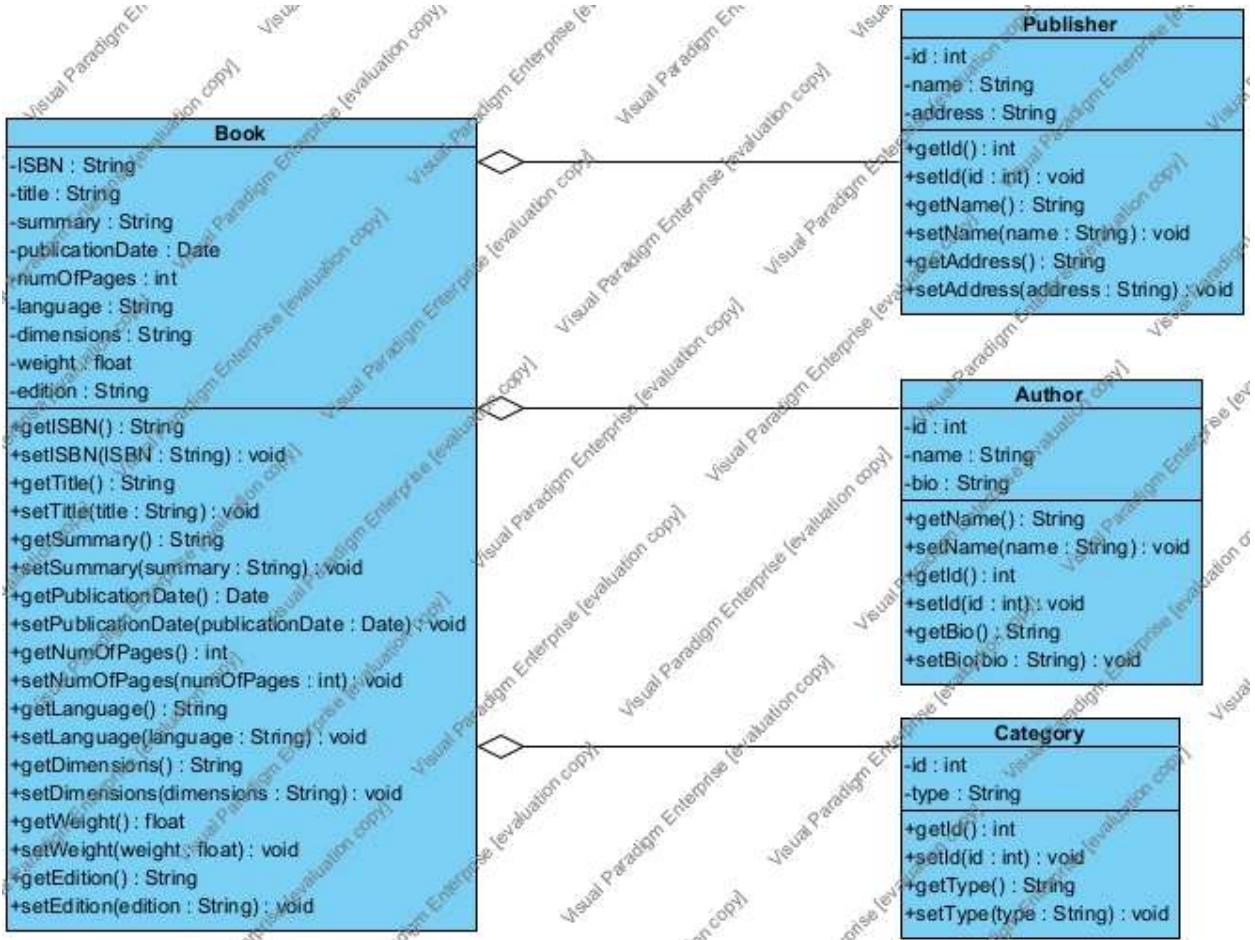


Câu 6: Xây dựng các biểu đồ gói theo kiến trúc 3 lớp của MVC cho hệ thống

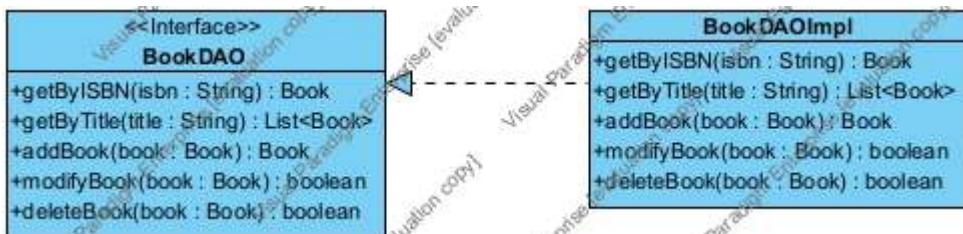
- Biểu đồ gói tổng quát:



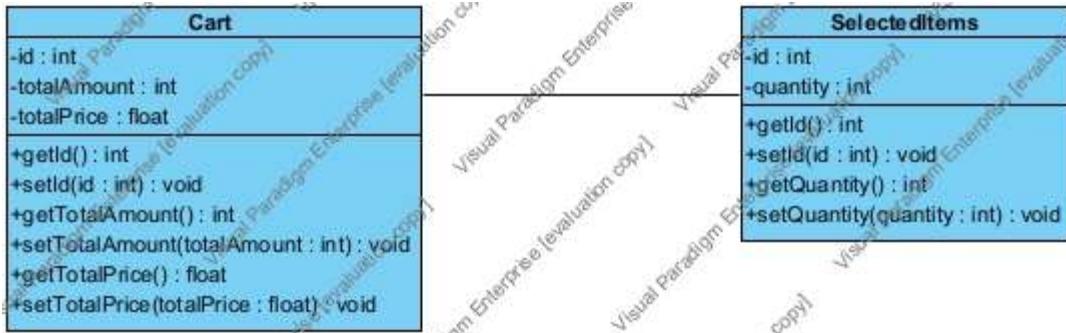
- Biểu đồ lớp của gói model.book:



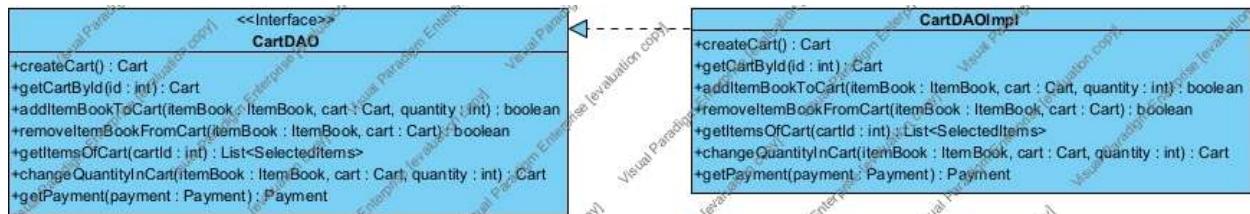
- Biểu đồ lớp của gói logicApplication.bookDAO:



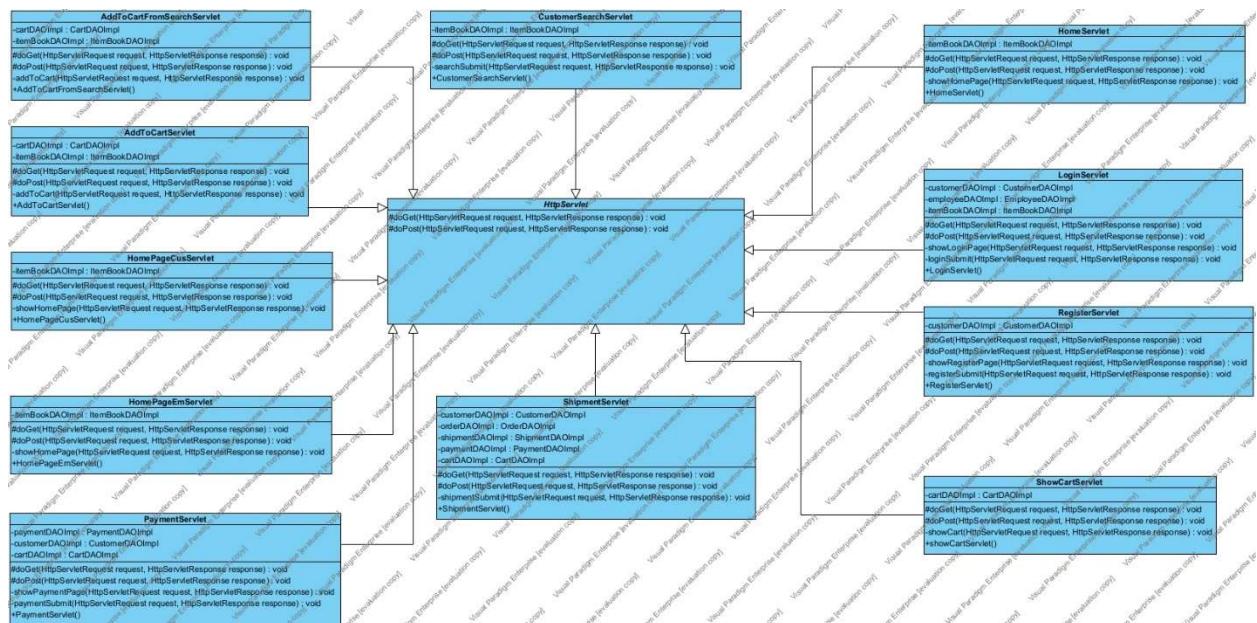
- Biểu đồ lớp của gói model.cart:



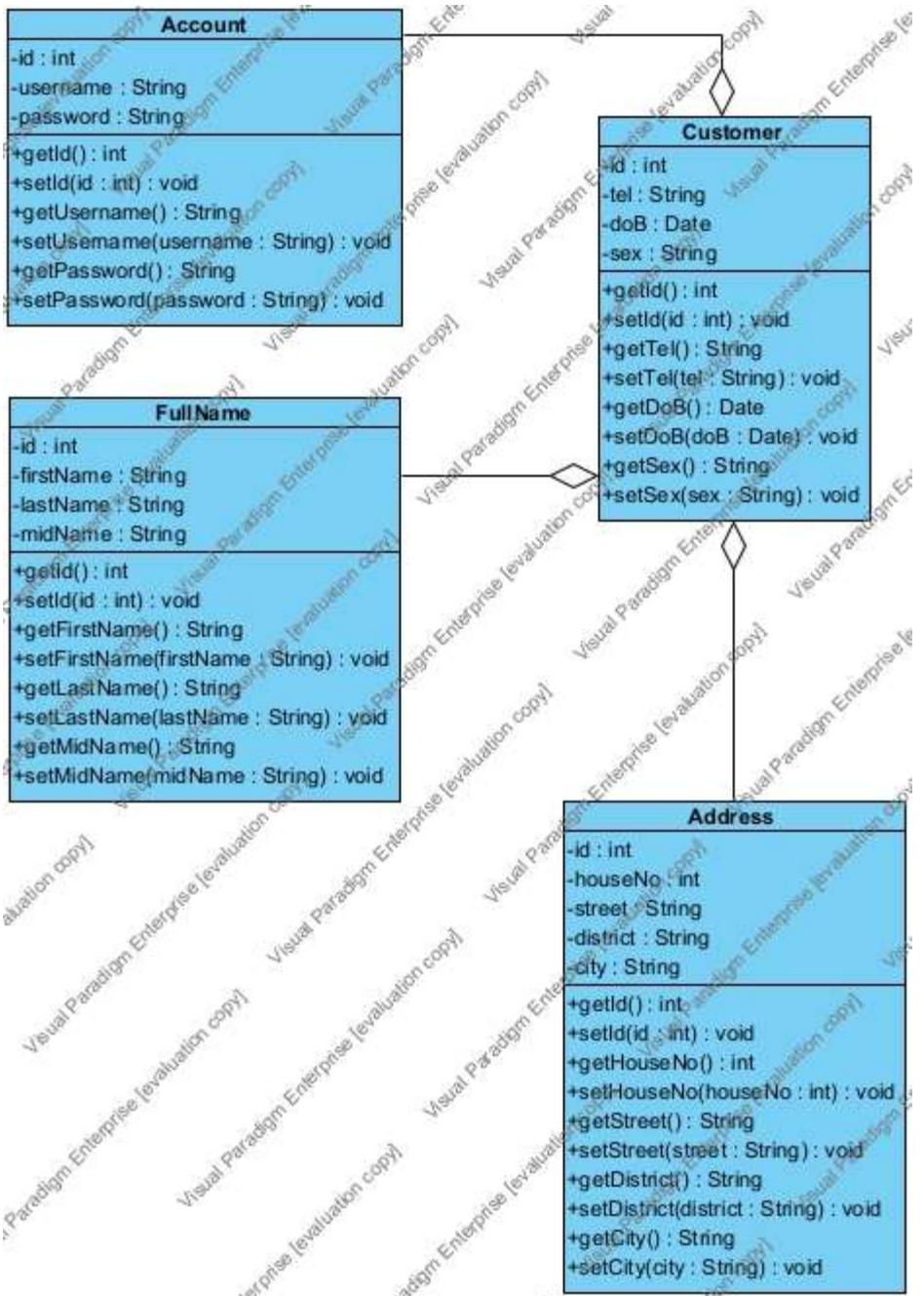
- Biểu đồ lớp của gói logicApplication.cartDAO:



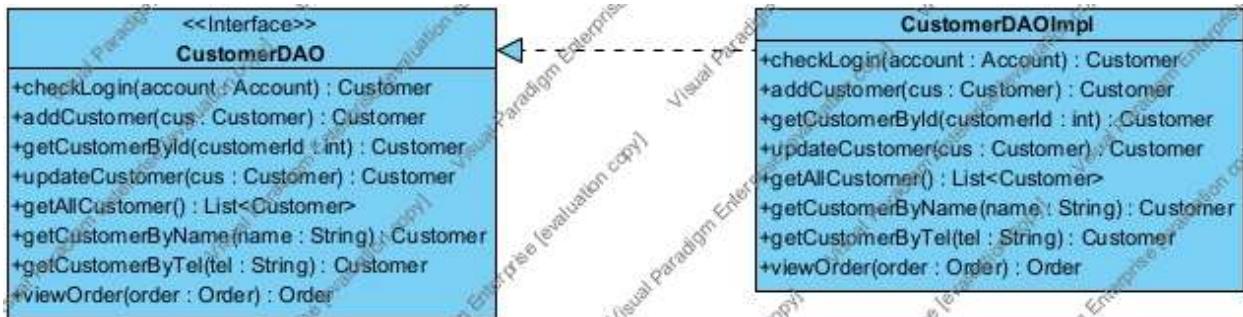
- Biểu đồ lớp của gói logicApplication.controller:



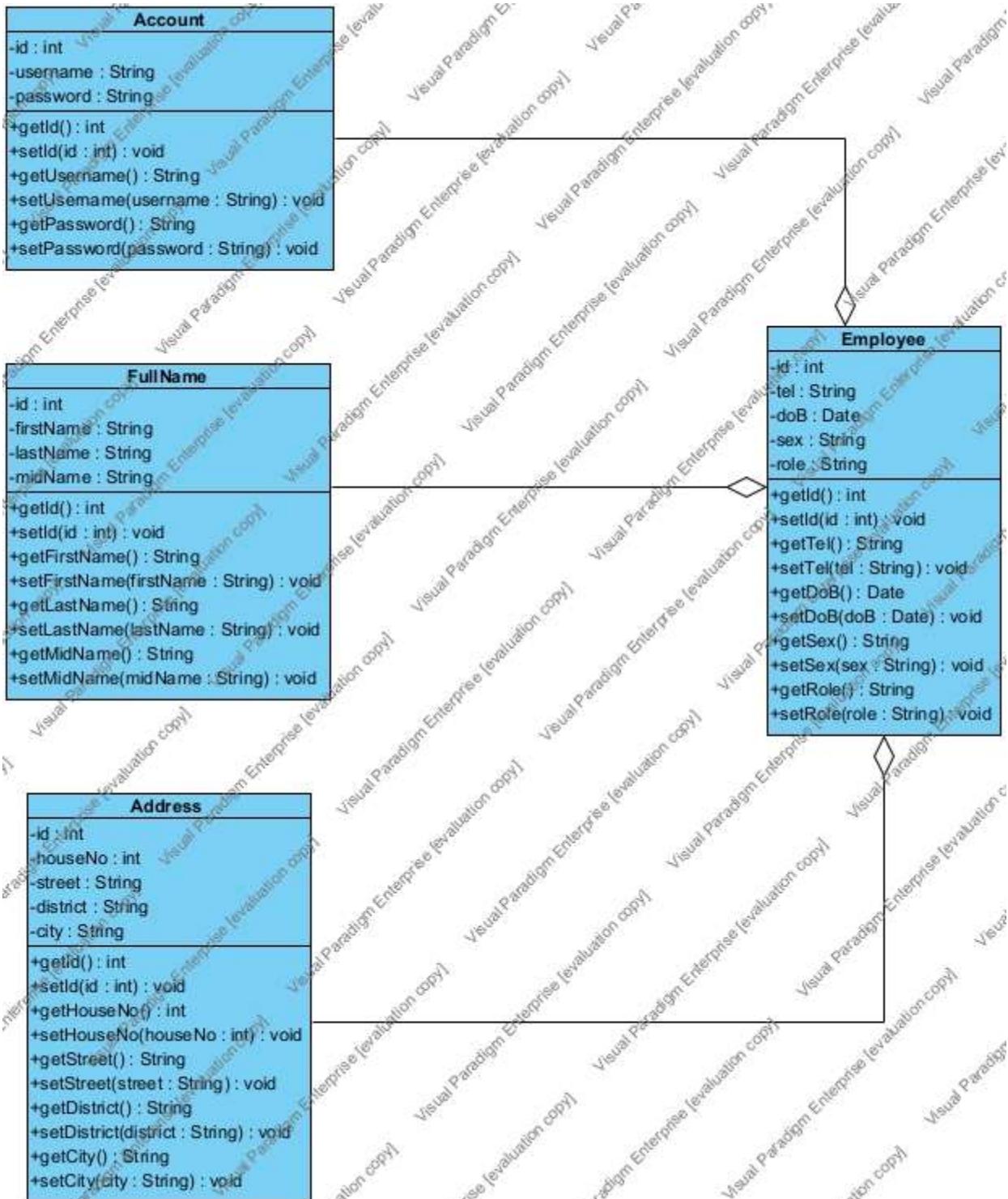
- Biểu đồ lớp của gói model.customer:



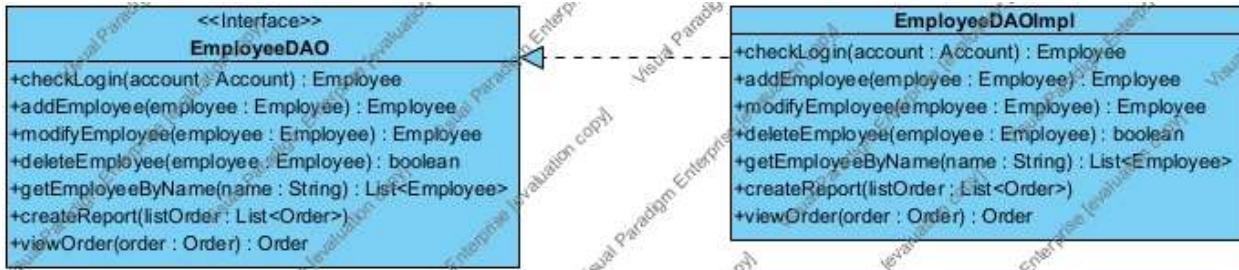
- Biểu đồ lớp của gói logicApplication.customerDAO:



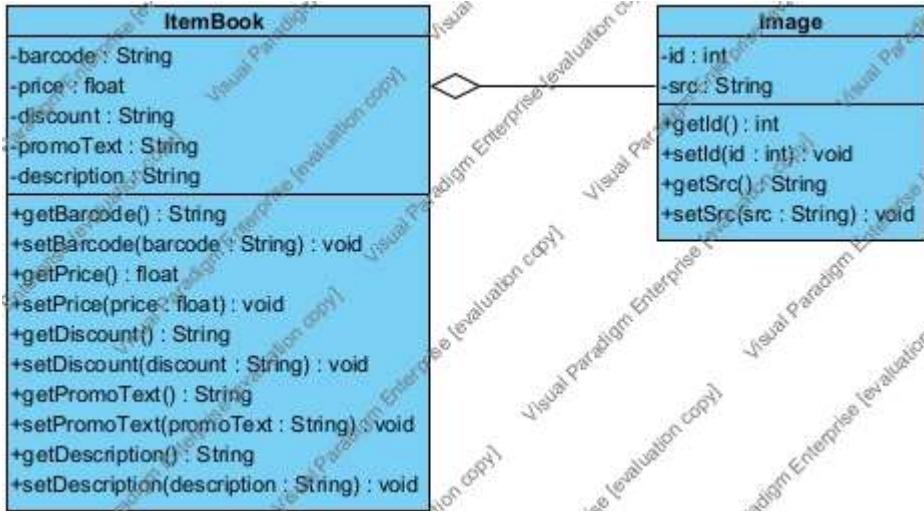
- Biểu đồ lớp của gói model.employee:



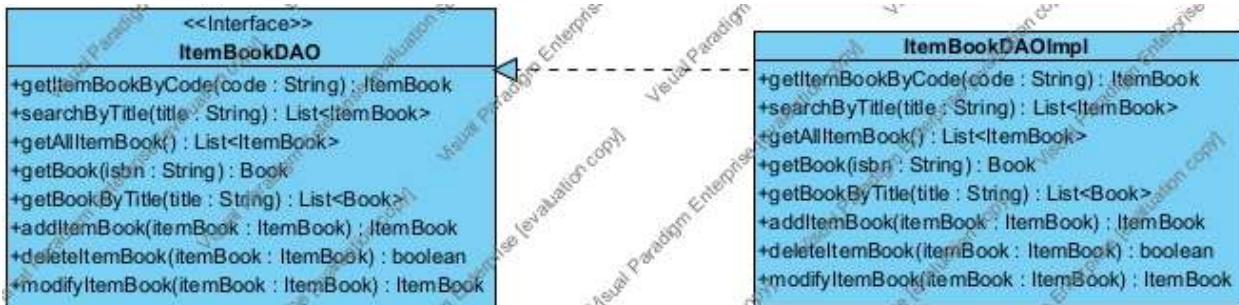
- Biểu đồ lớp của gói logicApplication.employeeDAO:



- Biểu đồ lớp của gói model.itemBook:



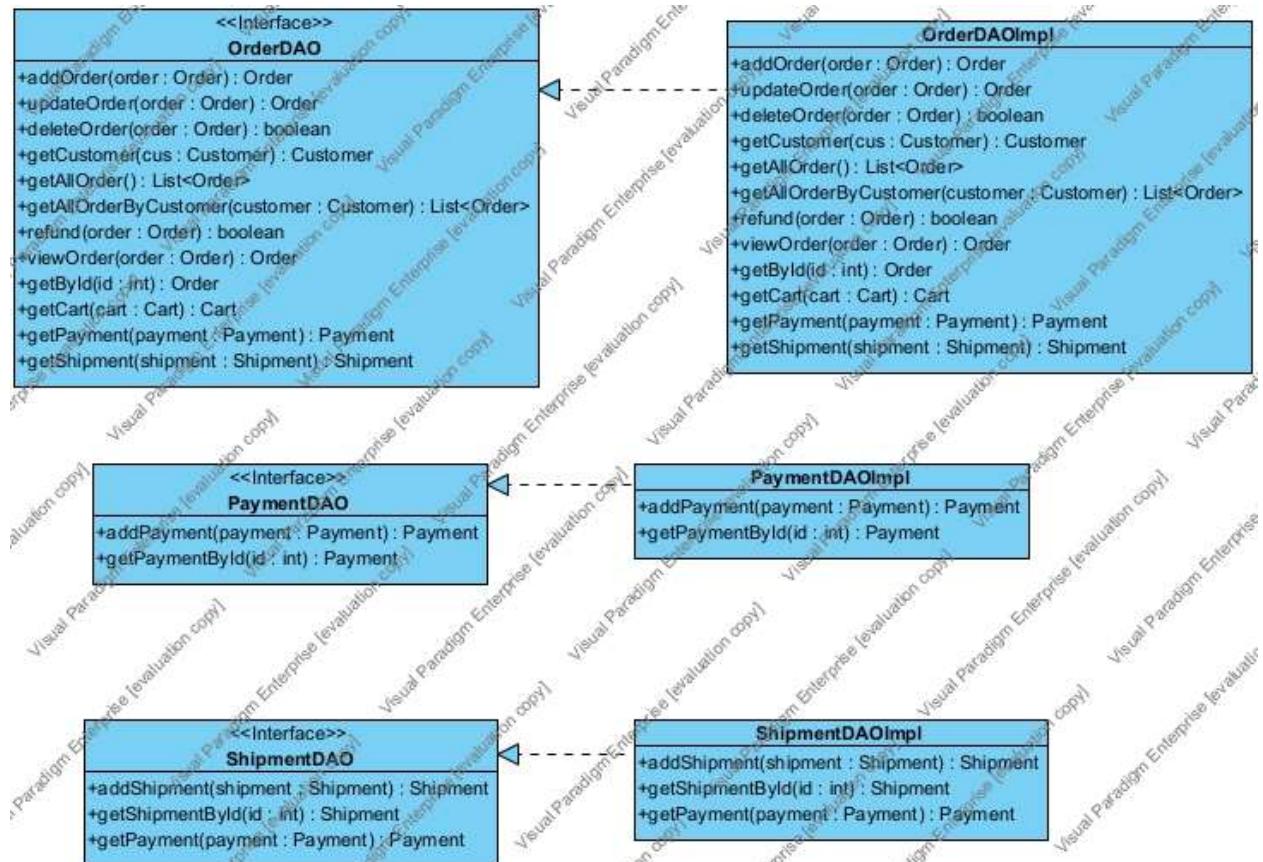
- Biểu đồ lớp của gói logicApplication.itemBookDAO:



- Biểu đồ lớp của gói model.order:

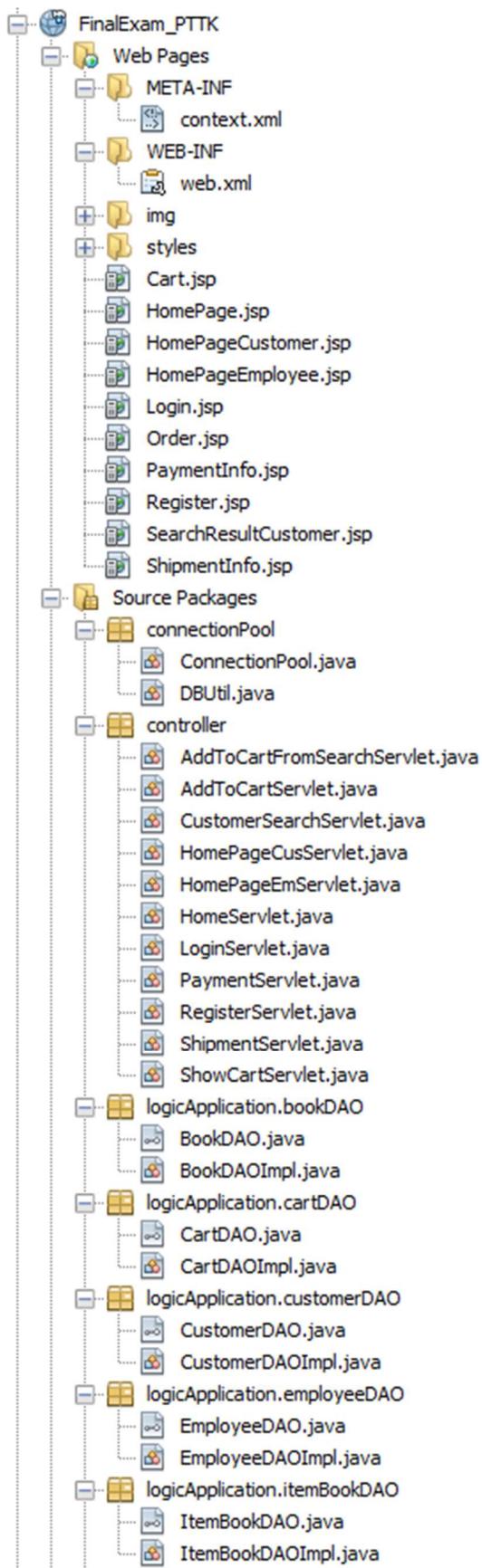


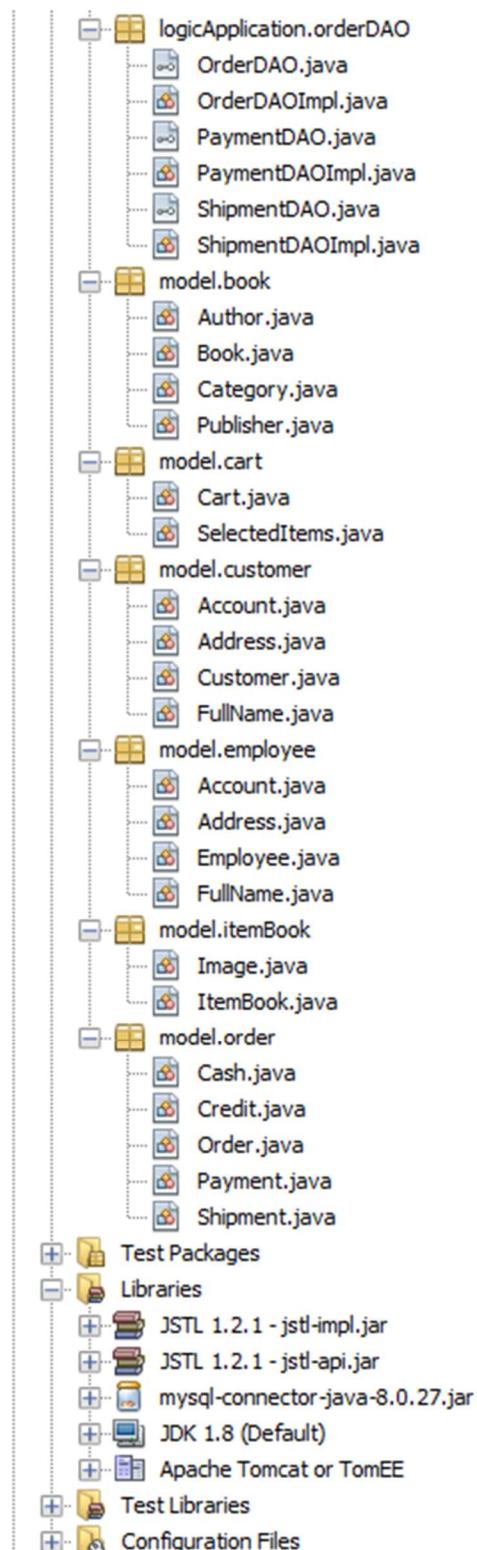
- Biểu đồ lớp của gói logicApplication.orderDAO:



Câu 7: Cài đặt hệ thống (sử dụng JSP Servlet)

- IDE sử dụng: Apache NetBeans 12.0
- Ứng dụng web chạy trên máy chủ Tomcat phiên bản 8.5.73.
- Sử dụng JDK 8, MySQL 8.0.27 cùng các thư viện:
 - MySQL JDBC Connector 8.0.27
 - JSTL 1.2.1
- Cấu trúc thư mục ứng dụng web sử dụng servlet và JSP:





TÀI LIỆU THAM KHẢO

1. Trần Đình Quế, *Giáo trình Phân tích và Thiết kế Hệ thống thông tin*, Học viện Công nghệ Bưu chính Viễn thông, 2018.
2. [Architectural styles for Software Design \(tutorialride.com\)](#)
3. [System Analysis and Design - Quick Guide \(tutorialspoint.com\)](#)
4. [MVC Architecture: A Detailed Insight to the Modern Web Applications Development| Crimsonpublishers.com](#)
5. [DAO Pattern Explained. Introduction | by Colin But | Medium](#)
6. [Database Structure and Design Tutorial | Lucidchart](#)
7. [What is User Interface Design? | Interaction Design Foundation \(IxDF\) \(interaction-design.org\)](#)
8. Phụ lục B1 và B2, File câu hỏi thi kết thúc học phần *Phân tích & Thiết kế Hệ thống thông tin* 2021.