HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

□□□□□

BÀI TẬP NHÓM MÔN HỌC

KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM

MIOStore - Assignment 2

Giảng viên hướng dẫn : Trần Đình Quế

Nhóm môn học : 03

Nhóm bài tập : 08

Danh sách thành viên :

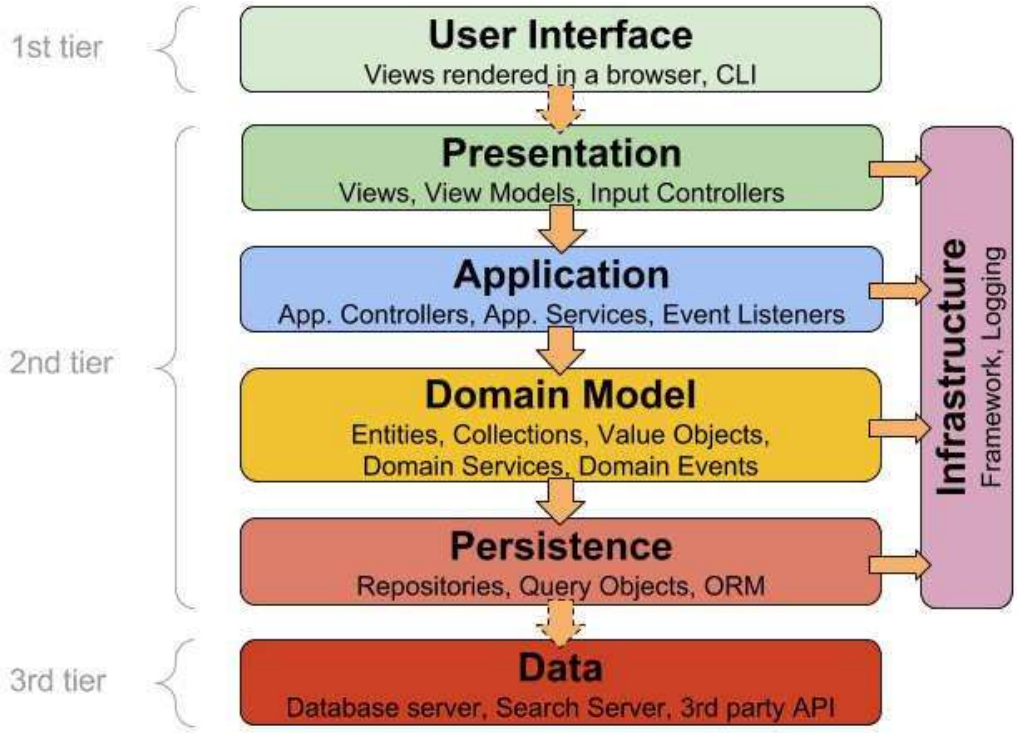
Lê Thị Hoa	B16DCCN151
Phạm Ngọc Hoàng	B16DCCN159
Nguyễn Đình Thắng	**B16DCCN319**
Phan Quang Thành	B16DCCN331
Trần Nhật Minh	B16DCCN519

Câu 1: Tìm hiểu thảo luận về 5 architecture pattern và lựa chọn cho hệ thống:

I. Thảo luận về 5 architecture pattern:

a. The Layered Architectural Pattern (Mô hình kiến trúc nhiều lớp)

- Ý tưởng: Phân chia hệ thống thành các layer, mỗi layer sẽ có một trách nhiệm nhất định và cung cấp dịch vụ cho layer cao hơn
- Thông thường một kiến trúc sẽ gồm có các layer:

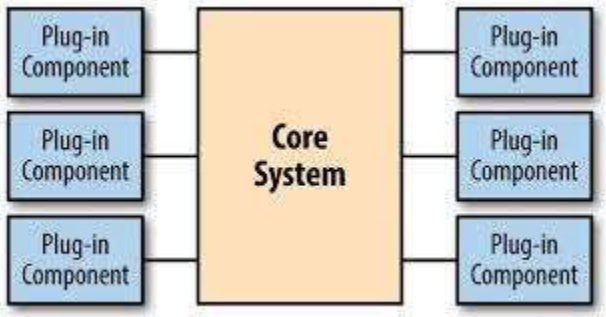


- Lớp UI: chứa các thành phần đảm nhiệm việc gửi phản hồi và hiển thị giao diện phía người dùng
- Lớp ứng dụng: chứa logic code tương ứng với tính năng của ứng dụng
- Lớp miền hoặc nghiệp vụ: tầng này biểu diễn các thực thể hoặc service
- Lớp truy cập cơ sở dữ liệu (Persistence): các đối tượng nằm ở tầng này liên quan tới thao tác với cơ sở dữ liệu, Data Access Object...
- Lớp database: Lớp chứa cơ sở dữ liệu, API bên thứ 3…

- Ưu điểm:
  - Hầu hết các nhà phát triển đều quen với mô hình này
  - Áp dụng mô hình này cũng dễ dàng để viết một ứng dụng có tổ chức tốt và có thể kiểm thử được

- Nhược điểm:
  - Mô hình này có xu hướng dẫn đến các ứng dụng nguyên khối (monolithic applications) khó phân tách sau đó
    Các nhà phát triển thường thấy mình viết rất nhiều code để đi qua các lớp khác nhau, mà không thêm bất kỳ giá trị nào trong các lớp này.
  - Nếu chỉ viết một ứng dụng CRUD đơn giản, mô hình này có thể là quá mức cần thiết.
- Phù hợp cho:
  - Các ứng dụng cho các doanh nghiệp tiêu chuẩn có độ phức tạp hơn là một ứng dụng chỉ CRUD thông thường

2. The Microkernel Architectural Pattern (Mô hình kiến trúc Microkernel)
   - Ý tưởng: Hệ thống cốt lõi sẽ chứa các chức năng cốt lõi để chạy hệ thống, và các thực thể bên ngoài có thể được tích hợp vào. Lõi của ứng dụng sẽ cung cấp các truy cập đầu vào và luồng dùng chung mà không thực sự cần biết các thực thể được tích hợp vào đang làm gì
   - Phù hợp với ứng dụng chứa nhiều điểm tích hợp với các thực thể bên ngoài, mà các thực thể ấy không biết trước được cái nào sẽ được tích hợp vào hệ thống trong tương lai.

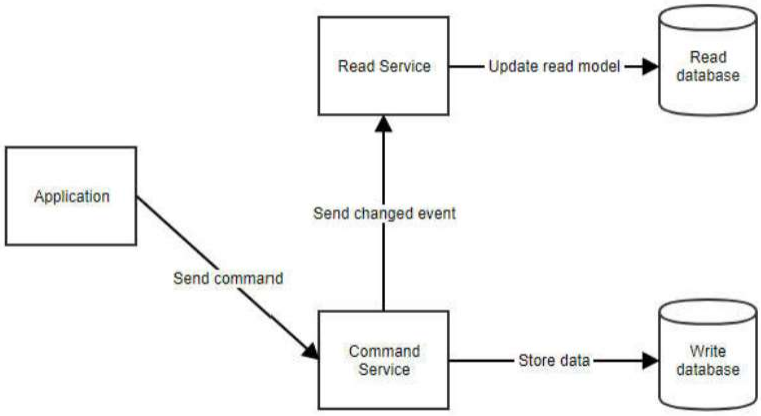


   - Ưu điểm:
     - Mô hình này cung cấp tính linh hoạt và khả năng mở rộng lớn
     - Một số triển khai cho phép việc tích hợp ngay cả khi ứng dụng đang chạy
     - Phần lõi và phần tích hợp vào có thể được phát triển bởi 2 team riêng biệt
   - Nhược điểm:
     - Khó để quyết định phần nào thuộc về lõi phần nào không

- Các API đư ợc xác định trước có thể không phù hợp với việc tích hợp trong tương lại

- Phù hợp cho:
  - Các ứng dụn g lấy dữ liệu từ các nguồn khác nhau, biến đổi dữ liệu và ghi nó ở các điểm đến khác (destinations)
  - Các ứng dụng quy trình làm việc (Workflow applicationss)
  - Các ứng dụng lập kế hoạch công việc và nhiệm vụ

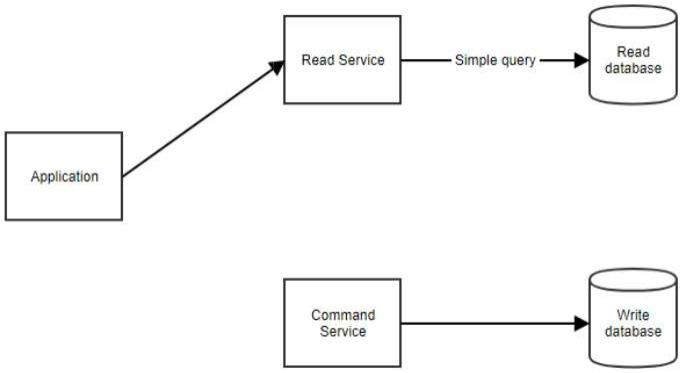
3. The CQRS Architectuural Pattern (Mô hình kiến trúc Commaand and Query Responsibility Segregation)
   - Ý tưởng: CQRS là viết tắt của Command and Query Responsibility Segregation (Phân chia trách nhiệm của lệnh và truy vấn). Lớp ứng dụng của phần mềm sẽ được phân tách thành một ngăn xếp lệnh (command stack) và ngăn xếp truy vấn (query stack). Các lệnh là các hoạt động thaay đổi trạng thái ứng dụng và không trả lại dữ liệu. Truy vấn là các hoạt độngg trả về dữ liệu nhưng không thay đổi trạng thái ứng dụng.
   - Theo thiết kế CQRS cơ bản cho phép tối ưu hóa việc đọc và ghi. Thiết kế của ngăn xếp truy vấn đại diện cho cách đọc dữ liệu hiệu quả nhất. Ngăn xếp lệnh chỉ cần quan tâm về cách ghi dữ liệu.
     Có thể tối ưu hóa kiến trúc với 2 db riêng biệt đọc và ghi
   - Cách mô hình hoạt động:
     - Đối với hoạt động ghi (commands)



Khi người dùng thực hiện một hành động, ứng dụng sẽ gửi lệnh đến dịch vụ lệnh. Dịch vụ lệnh lấy bất kỳ dữ liệu nào nó cần từ cơ sở dữ liệu lệnh, thực

hiện các thao tác cần thiết và lưu trữ trở lại trong cơ sở dữ liệu. Sau đó nó thông báo dịch vụ đọc để mô hình đọc có thể được cập nhật.

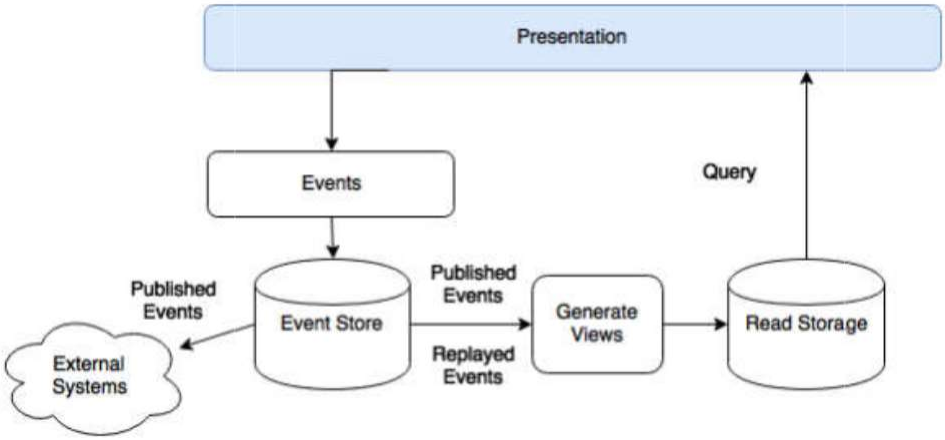
 Đối với mô hình đọc (queries)



Khi ứng dụng cần hiển thị dữ liệu cho người dùng, nó có thể truuy xuất mô hình đọc bằng cách gọi dịch vụ đọc

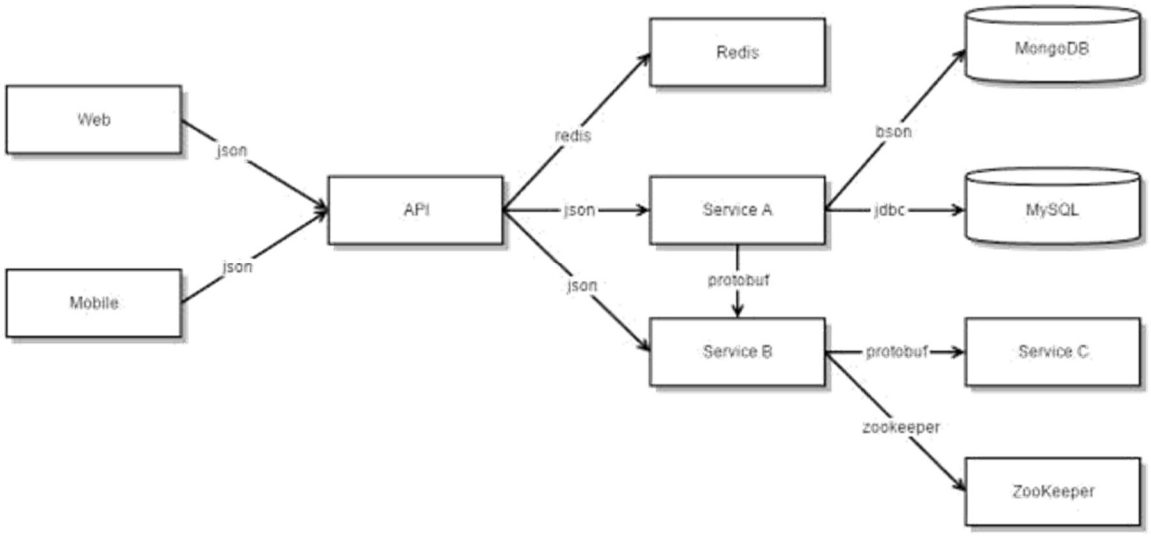
- Ưu điểm:
  - Các mô hình lệnh có thể tập trung vào logic nghiệp vụ và xác nhận trong khi các mô hình đọc có thể được điều chỉnh theo các kịch bản cụ thể.
  - Bạn có thể tránh các truy vấn phức tạp, điều này làm cho việc đọc hiệu quả hơn.
- Nhược điểm:
  - Việc giữu cho lệnh và các mô hình đọc đồng bộ có thể trở nêên phức tạp
- Phù hợp cho:
  - Các ứng dụng thiên về số lượng đọc cao
  - Các ứng dụng cóó độ phức tạp về nghiệp vụ

4. The Event Sourcing Architectural Pattern (Mô hình kiến trúc nguồn cung ứng sự kiện)
   - Ý tưởng: Thay vì lưu trữ trạng thái vào cơ sở dữ liệu thì sẽ lưu các sự kiện đã xảy ra với mô hình. Trong mô hình này, sẽ không loại bỏ các sự kiện đã xảy ra, để khắc phục, chúngg ta sẽ thêm các sự kiện mới
   - Thường kết hợp với mô hình CQRS để cải thiện hiệu suất đặc biệt là khi có rất nhiều sự kiện



- Ưu điểm:
  - Mẫu kiến trúc phần mềm này có thể cung cấp một bản ghi kiểm toán ra khỏi hộp. Mỗi sự kiện đại diện cho một thao tác dữ liệu tại một thời điểm nhất định.
- Nhược điểm:
  - Nó đòi hỏi một số kỷ luật vì bạn không thể sửa dữ liệu sai bằng một chỉnh sửa đơn giản trong cơ sở dữ liệu.
  - Đây không phải là một nhiệm vụ tầm thường để thay đổi cấu trúc của một sự kiện. Ví dụ: nếu bạn thêm một thuộc tính, cơ sở dữ liệu vẫn chứa các sự kiện không có dữ liệu đó. Mã của bạn sẽ cần xử lý dữ liệu bị thiếu này một cách lịch sự.
- Phù hợp cho:
  - Cần công bố sự kiện cho các hệ thống bên ngoài
  - Sẽ được xây dựng với CQRS
  - Có tên miền phức tạp
  - Cần một bản ghi kiểm toán các thay đổi đối với dữ liệu

5. The Microservices Architectural Pattern
   - Ý tưởng : phát triển một ứng dụng bằng các service nhỏ và độc lập chạy trong tiến trình riêng của chúng. Các service này được phát triển và deploy một cách độc lập.



- Ưu điểm:
  - Có thể viết, bảo trì và triển khai các microsevice một cách riêng biệt
  - Cho phép triển khai và phân bố liên tục các ứng dụng phức tạp và lớn
  - Cho phép phân phối và triển khai liên tục các ứng dụng lớn, phức tạp.
    - Cải thiện khả năng bảo trì - mỗi dịch vụ tương đối nhỏ và do đó dễ hiểu và thay đổi hơn
    - Khả năng kiểm tra tốt hơn - các dịch vụ nhỏ hơn và nhanh hơn để kiểm tra
    - Khả năng triển khai tốt hơn - dịch vụ có thể được triển khai độc lập
    - Cho phép tổ chức phát triển với nhiều nhóm mà mỗi nhóm đóphụ trách và chịu trách nhiệm cho một hoặc nhiều dịch vụ. Mỗi nhóm có thể phát triển, thử nghiệm, triển khai và mở rộng quy mô dịch vụ của mình một cách độc lập với tất cả các nhóm khác.
  - Mỗi microservice tương đối nhỏ:
    - Nhà phát triển dễ hiểu hơn
    - IDE nhanh hơn làm cho các nhà phát triển làm việc hiệu quả hơn
    - Ứng dụng khởi động nhanh hơn, giúp các nhà phát triển làm việc hiệu quả hơn và tăng tốc triển khai
  - Nếu các lỗi xảy ra thì sẽ độc lập ở mỗi microservice, không làm ảnh hưởng đến toàn bộ hệ thống
  - Khi triển khai một service mới có thể sử dụng các công nghệ mới mà không bị bó buộc trong các công nghệ

- Nhược điểm:
  - Các nhà phát triển phải đối phó với sự phức tạp bổ sung của việc tạo ra một hệ thống phân tán:
    - o Các nhà phát triển phải thực hiện cơ chế truyền thông liên dịch
    - o Việc thực hiện các yêu cầu trải rộng trên nhiều dịch vụ khó khăn hơn o Kiểm tra sự tương tác giữa các dịch vụ khó khăn hơn
    - o Việc thực hiện các yêu cầu trải rộng trên nhiều dịch vụ đòi hỏi sự phối hợp cẩn thận giữa các nhóm phát triển
    - o Các công cụ / IDE dành cho nhà phát triển được định hướng xây dựng các ứng dụng nguyên khối và không cung cấp hỗ trợ rõ ràng để phát triển các ứng dụng phân tán.
  - Quá trình triển khai và quản lý một hệ thống bao gồm nhiều dịch vụ khác nhau phức tạp
  - +Tăng tiêu thụ bộ nhớ. Kiến trúc microservice thay thế N ứng dụng nguyên khối bằng NxM ứng dụng microservice
- Phù hợp với:
  - Ứng dụng sẽ trở nên phức tạp nếu triển khai theo mô hình nguyên khối
  - Service cung cấp chức năng cho một số ứng dụng khác
  - Các ứng dụng mà có các phần có thể được cần được mở rộng
  - Các ứng dụng theo hướng nghiệp vụ đã xác định rõ được phạm vi nghiệp vụ của ứng dụng

b) Lựa chọn mô hình phù hợp với hệ thống:

- Lựa chọn three-tier (một dạng layer architecture) để sử dụng cho hệ thống MIOStore
- Lý do lựa chọn:
  - Sự linh hoạt:
    - Quản lý dữ liệu độc lập với hỗ trợ lưu trữ vật lý
    - Bảo trì của logic nghiệp vụ dễ dàng hơn
    - Di chuyển sang môi trường đồ họa mới nhanh ơn
    - Nếu có thay đổi nhỏ trong layer business logic, không cần phải cài đặt lại toàn bộ hệ thống trong các PC các nhân của người dùng
  - Tính tái sử dụng:

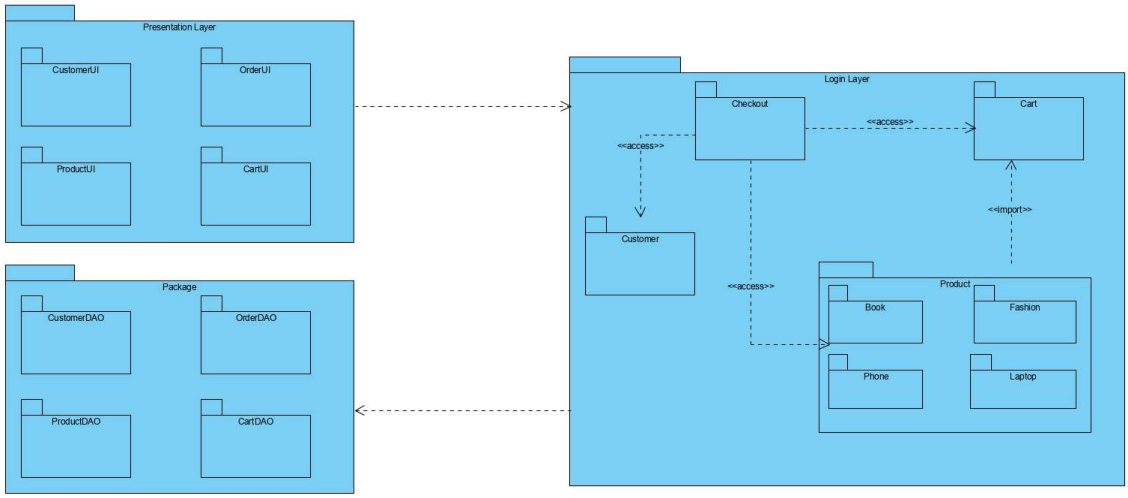
- Ứng dụng được tổ chức và phân chia tốt hơn.
- Sự tái sử dụng của layer business logic là lớn hơn. Khi các thành phần, chức năng đều được cài đặt và kiểm thử, có thể tái sử dụng lại nó cho các project khác

- Bảo mật
  - Tính bảo mật cao vì người dùng không thể truy cập trực tiếp vào data

Câu 2: Package Diagram:

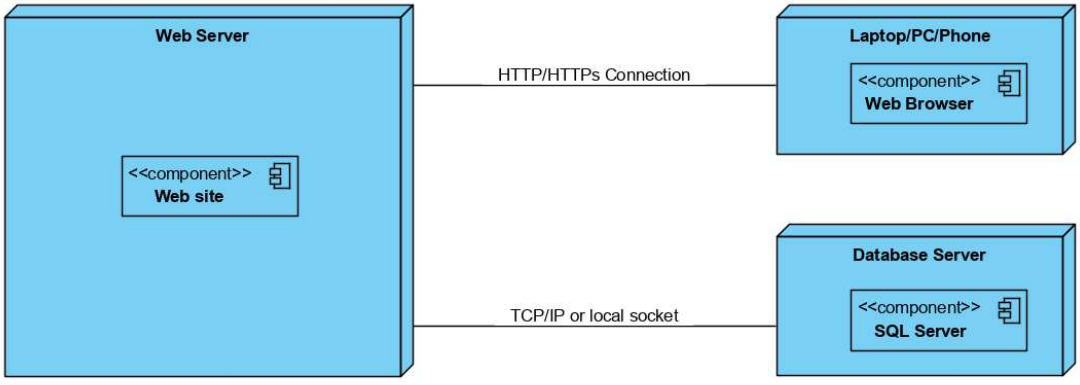
- Mô tả: Hệ thống được xây dựng theo mô hình 3-tier. Các layer được thiết kế như sau:
  - Presentation Layer:
    - CustomerUI package
    - Oder UI package
    - ProductUI package
    - CartUI package
  - Logic Layer
    - Customer package
    - Checkout package
    - Cart Package
    - Product Package
      - Book package
      - Phone package
      - Computer package
      - Fashion package
      - Luggage package
  - Data Layer
    - Customer package
    - Oder package
    - Cart package
    - Product package
  - Các layer được liên kết với nhau bằng quan hệ dependency.
  - Trong Logic Layer:
    - Checkout package có quan hệ access với Cart package và có quan hệ import với Product package.

- Customer packagr có quan hệ import với Checkout package.
- Cart package có quan hệ import với Product package.



Câu 3: Deployment diagram:

- Các Node: Web Server, Laptop, Database Server.
- Chi tiết các Node:
  - Web Server:
    + Các Component: Website .
  - Laptop:
    + Các Component: Web browser.
  - Database Server:
    + Các Componet: Database MySQL
- Deployment Diagram:

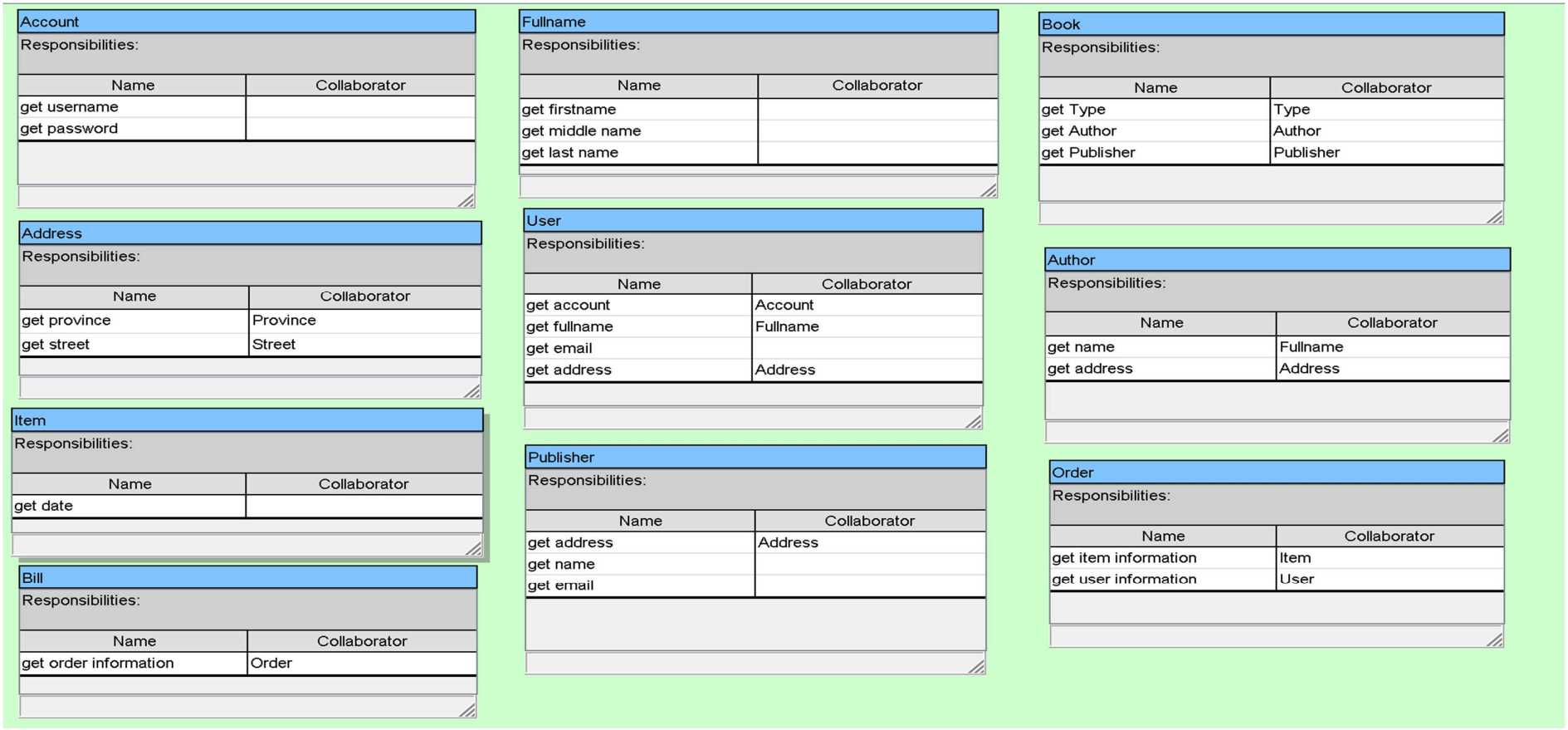


Câu 4:

User story	Acceptance criteria
As a manager I want to see stat by product so I can see number of product sold	• See table result in result Frm • Search for product • See product info in table
As a manager I want to see stat by staff so I can see number of product sold by a staff	• See table result in result Frm • Search for staff in table order in database • See staff information in table
As a manager I want to see stat by product in managerFrm so I can see stat by product in a period	• Click in view stat by product in mainFrm to view • Enter startDate and endate • Search for product in table bill in database
As a manager I want to see stat by staff in managerFrm so I can see stat by staff in a period	• Click in view stat by staff in mainFrm to view • Enter startDate and endate • Search for staff in table bill in database
As a manager I want to see stat by type of product in managerFrm so I can see stat by type product in a period	• Click in view stat by type product in mainFrm to view • Enter startDate and endate • Search for type product in table bill in database
As a manager I want to see stat in a period of time in managerFrm so I can see stat in a period	• Click in view stat by time in mainFrm to view • Enter startDate and endate • Search for time in table bill in database
As a manager I want to see product'information in statFrm so I can distinguish products	• Show product info in table result • Search for product in table productInfo in database • Click in a line to see full information of this product
As a manager I want to see staff'information in specificallyStatFrm so I can see who sell this products	• Show staff'name in table result • Click to view full staff'information
As a manager I want to see date time in specificallyStatFrm so I can see the when the products sold	• Show date time in table result • Click to view full bill'information
As a manager I want to see product'information in	• Show product'id in table result • Click to view full product'information

specificallyStatFrm so I can know products'information	

Câu 5:



Account

Responsibilities:

Name	Collaborator
get username	
get password	

Address

Responsibilities:

Name	Collaborator
get province	Province
get street	Street

Item

Responsibilities:

Name	Collaborator
get date	

Bill

Responsibilities:

Name	Collaborator
get order information	Order

Fullname

Responsibilities:

Name	Collaborator
get firstname	
get middle name	
get last name	

User

Responsibilities:

Name	Collaborator
get account	Account
get fullname	Fullname
get email	
get address	Address

Publisher

Responsibilities:

Name	Collaborator
get address	Address
get name	
get email	

Book

Responsibilities:

Name	Collaborator
get Type	Type
get Author	Author
get Publisher	Publisher

Author

Responsibilities:

Name	Collaborator
get name	Fullname
get address	Address

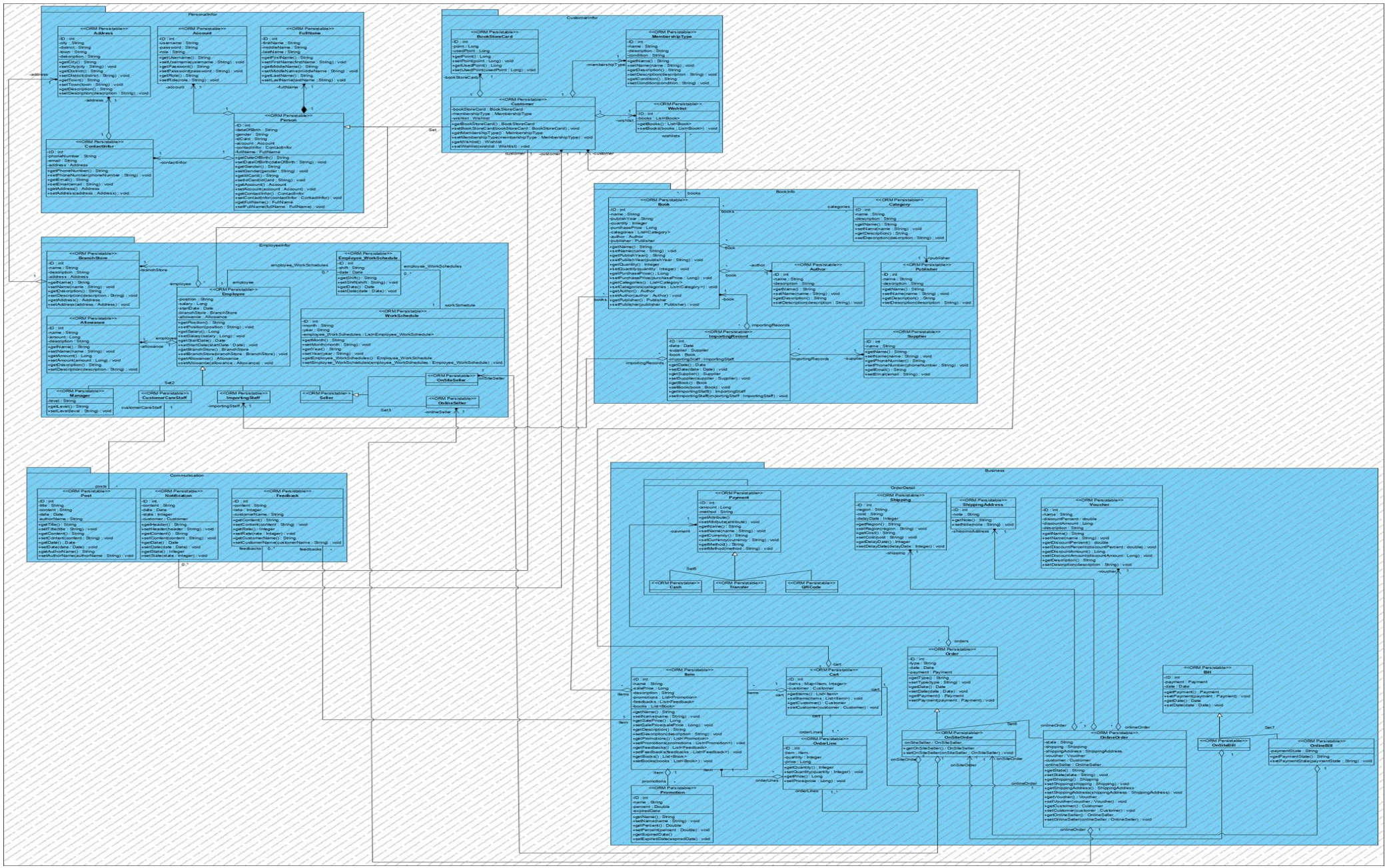
Order

Responsibilities:

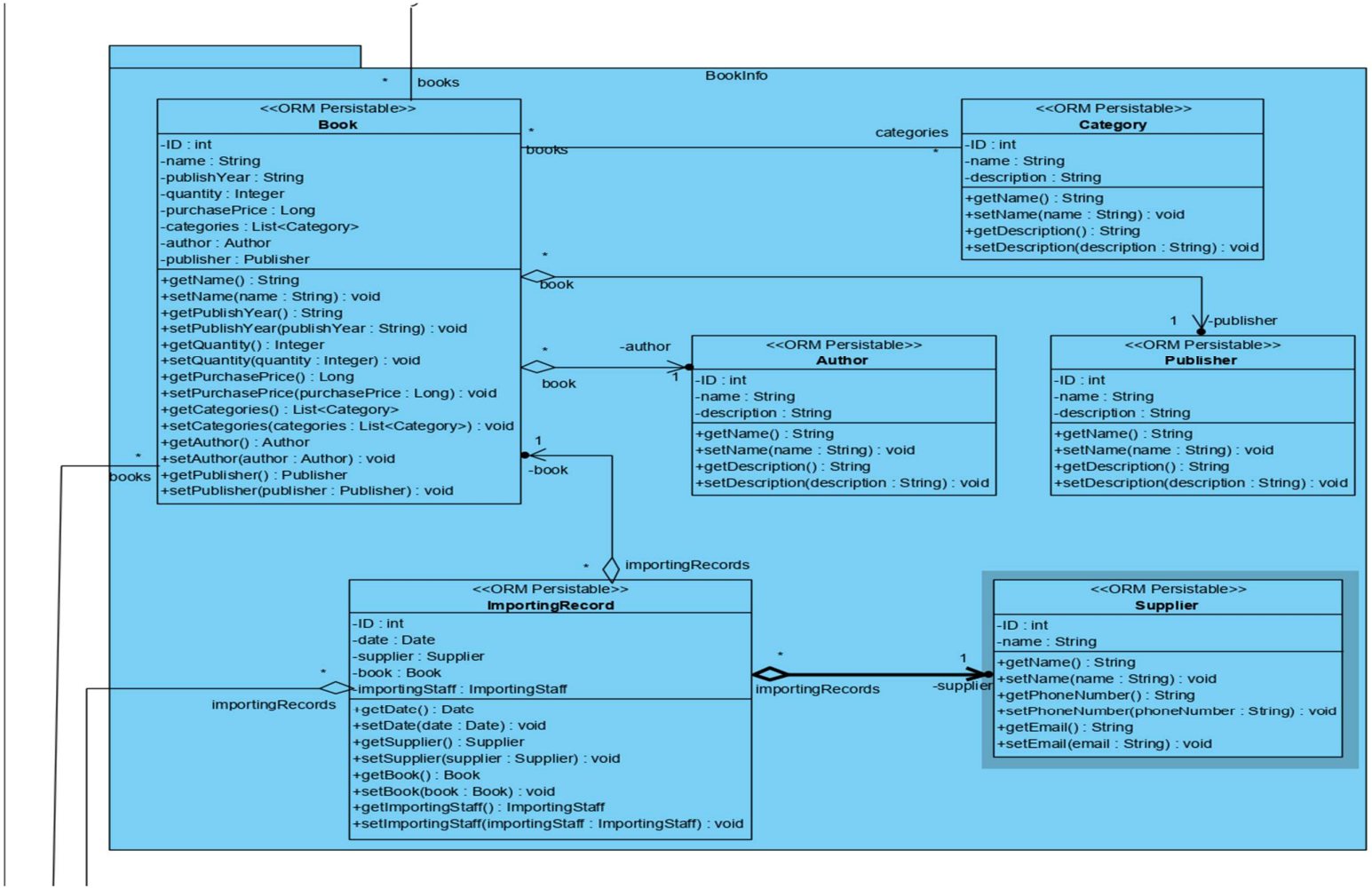
Name	Collaborator
get item information	Item
get user information	User

Câu 6:

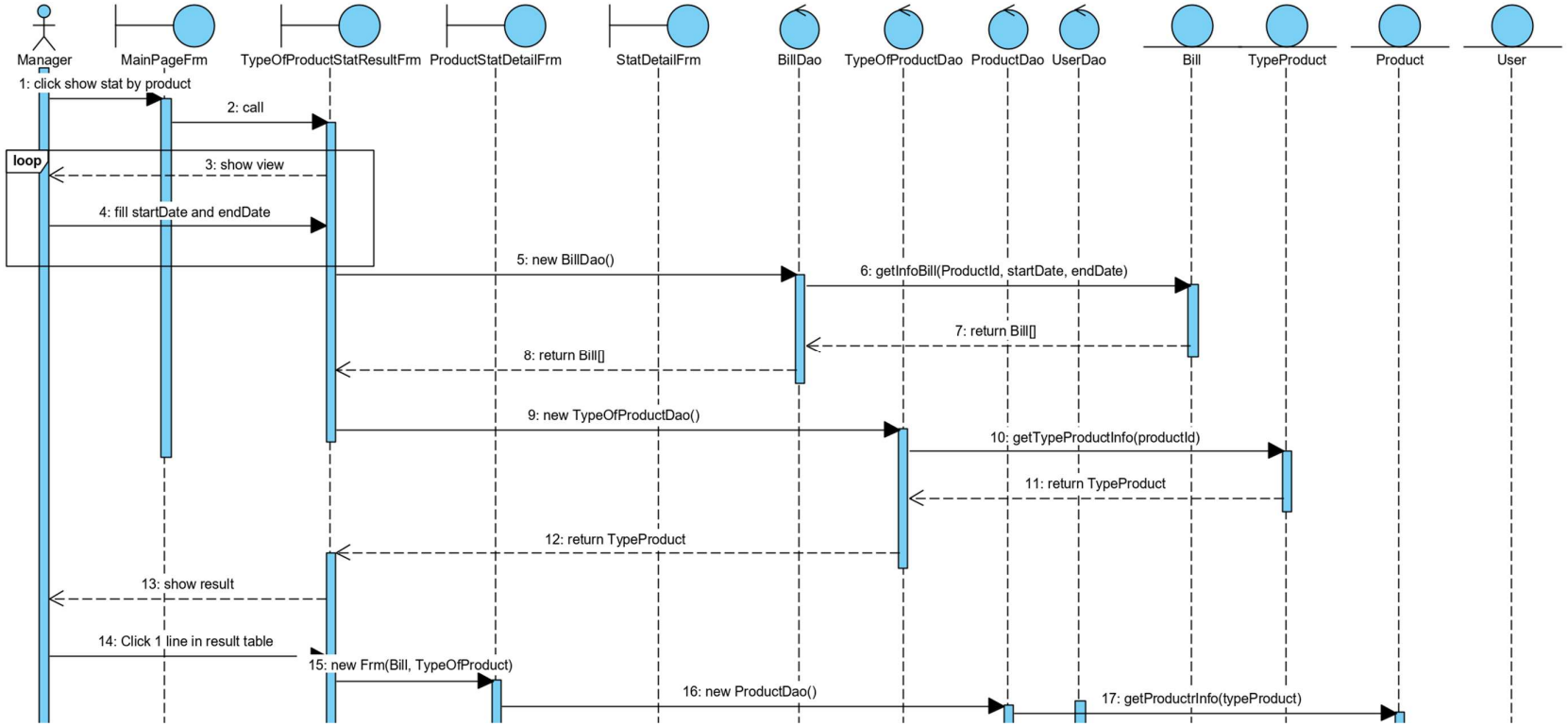
Class diagram thiết kế tổng thể

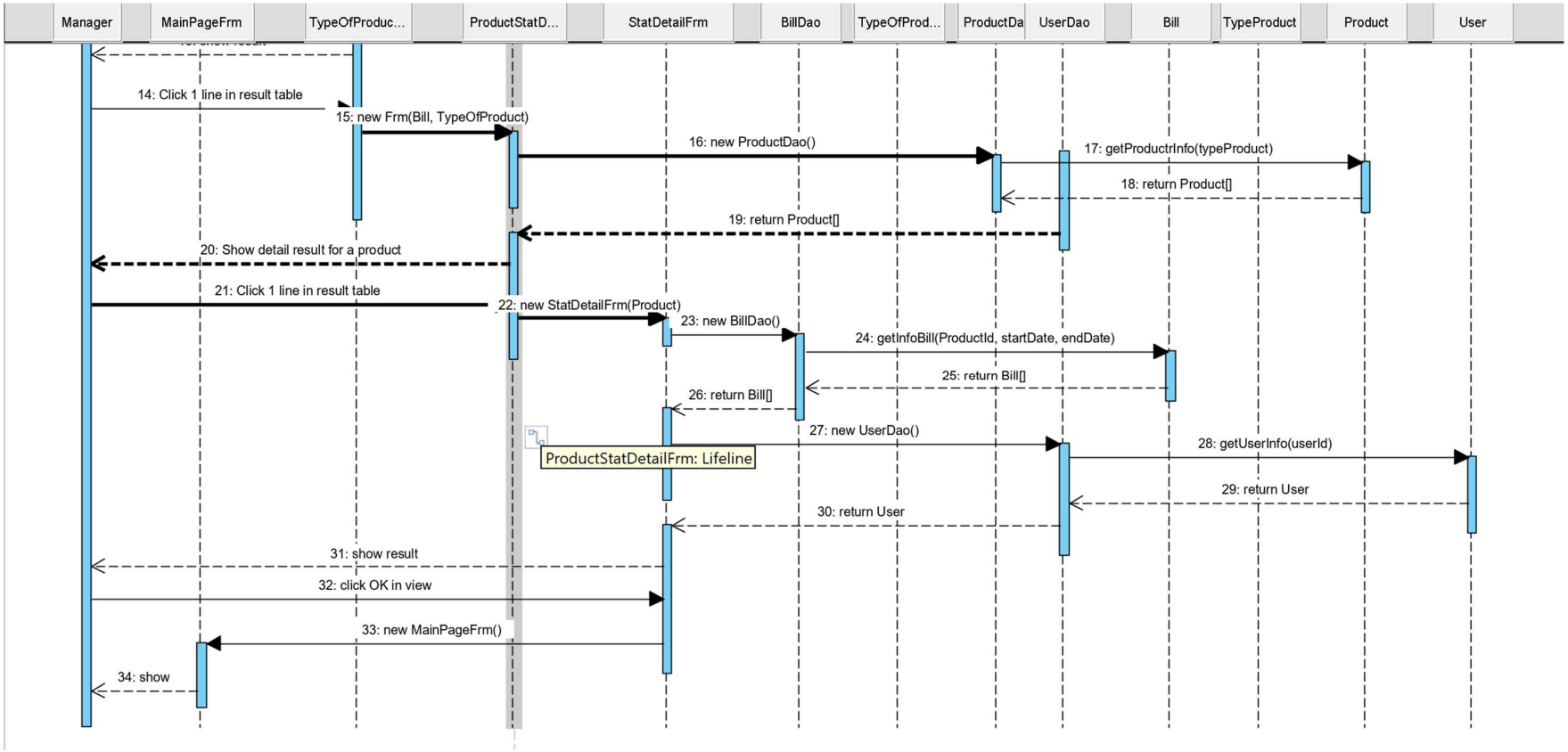

PersonalInfor
Address
Account
FullName
Person
ContactInfor
CustomerInfor
BookStoreCard
MembershipType
Customer
Wishlist
BookInfo
Book
Author
Publisher
Category
ImportingRecord
Supplier
EmployeeInfor
BranchStore
Employee_WorkSchedule
Employee
WorkSchedule
Allowance
Manager
CustomerCareStaff
ImportingStaff
Seller
OnSiteSeller
OnlineSeller
Communication
Post
Notification
Feedback
Business
Payment
Cash
Transfer
QRCode
OrderDetail
Shipping
ShippingAddress
Voucher
Item
Cart
Order
OnSiteOrder
OnlineOrder
OrderLine
Promotion
Bill
OnSiteBill
OnlineBill

Thiết kế chi tiết package:

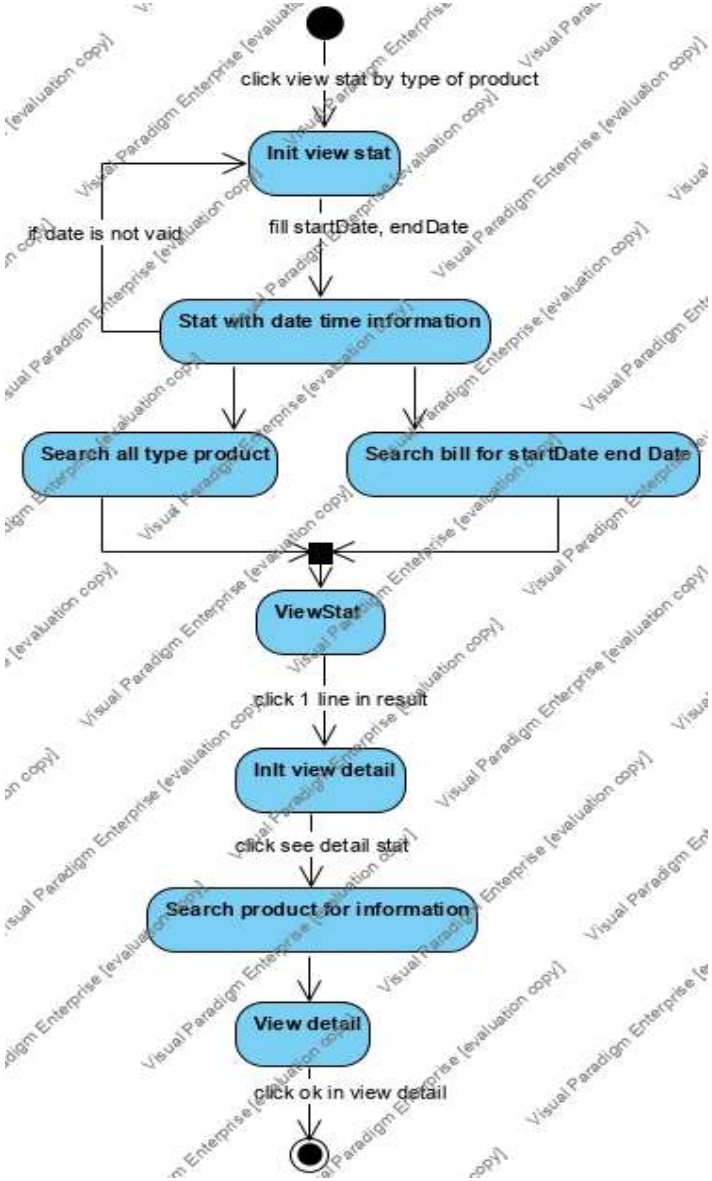


Câu 7: sequence diagram




Manager
MainPageFrm
TypeOfProduc...
ProductStatD...
StatDetailFrm
BillDao
TypeOfProd...
ProductDa
UserDao
Bill
TypeProduct
Product
User
14: Click 1 line in result table
15: new Frm(Bill, TypeOfProduct)
16: new ProductDao()
17: getProductrInfo(typeProduct)
18: return Product[]
19: return Product[]
20: Show detail result for a product
21: Click 1 line in result table
22: new StatDetailFrm(Product)
23: new BillDao()
24: getInfoBill(ProductId, startDate, endDate)
25: return Bill[]
26: return Bill[]
27: new UserDao()
ProductStatDetailFrm: Lifeline
28: getUserInfo(userId)
29: return User
30: return User
31: show result
32: click OK in view
33: new MainPageFrm()
34: show

Câu 8:


click view stat by type of product
Init view stat
fill startDate, endDate
if date is not vaid
Stat with date time information
Search all type product
Search bill for startDate end Date
ViewStat
click 1 line in result
InIt view detail
click see detail stat
Search product for information
View detail
click ok in view detail