HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

SÁCH BÀI TẬP

PHÁT TRIỂN PHẦN MỀM HƯỚNG DỊCH VỤ

Biên soạn: TS. Đặng Ngọc Hùng

Hà Nội – 2024

LỜI NÓI ĐẦU

Kiến trúc hướng dịch vụ (Service-Oriented Architecture - SOA) là một giải pháp xây dựng các ứng dụng đáng tin cậy, linh hoạt và dễ mở rộng, đáp ứng yêu cầu triển khai và quản lý liên tục trong môi trường doanh nghiệp hiện đại. Sách bài tập SOA nhằm giúp sinh viên hiểu rõ hơn về SOA qua hệ thống câu hỏi lý thuyết, các mẫu thiết kế phổ biến và các bài tập tình huống, từ đó trang bị kiến thức và kỹ năng thực tiễn cho người học trong quá trình làm việc.

Sách bài tập được tổ chức với cấu trúc:

- **Phần 1: Câu hỏi lý thuyết,** cung cấp nền tảng lý thuyết về SOA qua các câu hỏi trắc nghiệm và tự luận, giúp người học nắm vững các khái niệm cơ bản, phân tích và mô hình hóa dịch vụ, thiết kế hợp đồng và API dịch vụ.
- **Phần 2: Các mẫu thiết kế hướng dịch vụ**, giới thiệu các mẫu thiết kế quan trọng trong SOA, từ tổ chức, thiết kế, giao tiếp dịch vụ đến tối ưu hóa triển khai. Các vấn đề kỹ thuật thường gặp trong kiến trúc hướng dịch vụ được mô tả giúp người học làm quen với các mẫu thiết kế hữu ích trong kiến trúc hướng dịch vụ, từ đó có thể áp dụng linh hoạt trong các dự án thực tế.
- **Phần 3: Phân tích thiết kế hướng dịch vụ với các use case cụ thể**, cung cấp các tình huống thực tiễn như đăng ký môn học, tạo bài thi, xử lý đơn hàng… giúp người học vận dụng lý thuyết và các mẫu thiết kế để phân tích, thiết kế giải pháp hướng dịch vụ cho một usecase cụ thể.
- **Phụ lục: Đáp án và gợi ý tham khảo**, cung cấp các đáp án và gợi ý tham khảo nhằm hỗ trợ người học đối chiếu và tự đánh giá kết quả, củng cố kiến thức đã học qua từng câu hỏi và bài tập.

Hi vọng sách bài tập SOA sẽ là tài liệu hữu ích giúp người học xây dựng nền tảng lý thuyết từ cơ bản đến nâng cao, đồng thời phát triển kỹ năng phân tích, thiết kế và triển khai ứng dụng theo kiến trúc hướng dịch vụ một cách thành thạo.

Chúc các bạn học tập hiệu quả và đạt được nhiều thành công!

Tác giả

MỤC LỤC

PHẦN 1: CÂU HỎI LÝ THUYẾT

1.1. Các khái niệm cơ bản với SOA

1.1.1. Một số khái niệm và thuật ngữ

Dịch vụ (Service): Là một chương trình phần mềm cung cấp các chức năng của nó thông qua một API công khai, được xem như một phần của hợp đồng dịch vụ.

Hợp đồng dịch vụ (Service Contract): Là một bản mô tả về các giao diện (API) mà dịch vụ cung cấp để người dùng có thể truy cập vào các chức năng của dịch vụ.

Web Service dựa trên SOAP (Simple Object Access Protocol): Là các dịch vụ được triển khai dựa trên giao thức SOAP, cho phép trao đổi dữ liệu giữa các ứng dụng thông qua XML.

Dịch vụ RESTful (RESTful Services): Là các dịch vụ tuân theo nguyên tắc REST, thường sử dụng các phương thức GET, POST, PATCH, PUT và DELETE để thực hiện các tác vụ, có tính đơn giản, dễ tích hợp hơn so với SOAP.

API công khai (Published API): Là giao diện lập trình được công khai trong hợp đồng dịch vụ, cho phép người dùng bên ngoài truy cập vào các chức năng mà dịch vụ cung cấp.

Service-Orientation as a Design Paradigm: Hướng dịch vụ là một mô hình thiết kế giúp xây dựng các giải pháp phần mềm phân tán. Nó dựa trên lý thuyết kỹ thuật phần mềm gọi là "phân tách các mối quan tâm" (separation of concerns), nghĩa là một vấn đề lớn sẽ được giải quyết hiệu quả hơn khi nó được phân chia thành các vấn đề/mối quan tâm nhỏ hơn.

Tổ hợp dịch vụ (Service Composition): Là một tập hợp dịch vụ được phối hợp với nhau, nhằm cung cấp chức năng tự động hóa cho một nhiệm vụ hoặc quy trình kinh doanh cụ thể.

Kho dịch vụ (Service Inventory): Là một tập hợp dịch vụ được chuẩn hóa và quản lý độc lập, có tính bổ trợ lẫn nhau, nằm trong một phạm vi xác định đại diện cho toàn bộ doanh nghiệp hoặc một phần quan trọng của doanh nghiệp.

Tự động hóa quy trình kinh doanh bằng tổ hợp dịch vụ (Business Process Automation via Service Compositions): Nhiều quy trình kinh doanh có thể được tự động hóa bằng cách tạo các tổ hợp dịch vụ từ kho dịch vụ chung gồm các dịch vụ độc lập với các ngữ cảnh cụ thể.

Logic giải pháp hướng dịch vụ (Service-Oriented Solution Logic): Được triển khai dưới dạng các dịch vụ và tổ hợp dịch vụ được thiết kế phù hợp với nguyên tắc hướng dịch vụ.

Các nguyên lý thiết kế của hướng dịch vụ (*Service-Orientation Design Principles*):

1. Chuẩn hóa hợp đồng dịch vụ (*Standardized Service Contract*): Các dịch vụ trong cùng một danh mục dịch vụ tuân thủ cùng một tiêu chuẩn thiết kế hợp đồng.
2. Tính linh hoạt (*Service Loose Coupling*): Hợp đồng dịch vụ yêu cầu mức độ kết nối thấp đối với service consumer và bản thân chúng cũng tách rời khỏi môi trường xung quanh.
3. Tính trừu tượng hóa dịch vụ (*Service Abstraction*): Hợp đồng dịch vụ chỉ chứa thông tin cần thiết, che giấu các chi tiết không cần thiết nhằm duy trì kết nối lỏng lẻo và giúp thiết kế tổ hợp dịch vụ.
4. Tính tái sử dụng dịch vụ (*Service Reusability*): Dịch vụ được thiết kế để có thể tái sử dụng, đóng vai trò như tài nguyên doanh nghiệp đa mục đích.
5. Tính tự chủ của dịch vụ (*Service Autonomy*): Dịch vụ có mức độ kiểm soát cao đối với môi trường thực thi, đảm bảo tính ổn định và đáng tin cậy.
6. Phi trạng thái (*Service Statelessness*): Dịch vụ hạn chế quản lý thông tin trạng thái để tối ưu hóa tài nguyên, chỉ giữ trạng thái khi cần thiết
7. Khả năng phát hiện của dịch vụ (*Service Discoverability*): Dịch vụ được bổ sung metadata để dễ dàng nhận diện, phân phối và hiểu rõ khi cần tái sử dụng.
8. Khả năng tổ hợp của dịch vụ (*Service Composability*): Dịch vụ có khả năng tham gia hiệu quả trong các tổ hợp, bất kể độ phức tạp, đáp ứng yêu cầu thiết yếu trong xây dựng các giải pháp hướng dịch vụ phức tạp.

Thế nào là kiến trúc hướng dịch vụ (SOA = Service Orientation + distributed technology), để xây dựng các giải pháp hướng dịch vụ, chúng ta cần một kiến trúc công nghệ phân tán với các đặc điểm cụ thể. Các đặc điểm này phân biệt kiến trúc công nghệ hướng dịch vụ SOA với các kiến trúc khác.

Bốn đặc điểm của SOA:

1. *Hướng doanh nghiệp (Business-Driven):* Kiến trúc hỗ trợ các mục tiêu chiến lược dài hạn của doanh nghiệp, tránh bị lệch hướng khi các yêu cầu thay đổi theo thời gian.
2. *Trung lập với nhà cung cấp (Vendor-Neutral):* Tránh phụ thuộc vào một nền tảng cụ thể, đảm bảo khả năng mở rộng và thích nghi với công nghệ từ nhiều nhà cung cấp.
3. *Tập trung vào toàn doanh nghiệp (Enterprise-Centric):* Dịch vụ được xem là tài nguyên của doanh nghiệp, có khả năng sử dụng ngoài các ranh giới thực hiện cụ thể và tuân thủ các tiêu chuẩn doanh nghiệp.
4. *Tập trung vào tính tổ hợp (Composition-Centric):* Dịch vụ được thiết kế linh hoạt để tham gia vào nhiều tổ hợp khác nhau, sẵn sàng thích ứng trong nhiều cấu trúc giải pháp.

Bốn loại kiến trúc SOA phổ biến:

1. *Kiến trúc dịch vụ (Service Architecture):* Kiến trúc của một dịch vụ đơn lẻ.
2. *Kiến trúc tổ hợp dịch vụ (Service Composition Architecture):* Kiến trúc của một tập hợp dịch vụ được kết hợp thành tổ hợp dịch vụ.
3. *Kiến trúc kho dịch vụ (Service Inventory Architecture):* Kiến trúc hỗ trợ một tập hợp dịch vụ liên quan, được chuẩn hóa và quản lý độc lập.
4. *Kiến trúc doanh nghiệp hướng dịch vụ (Service-Oriented Enterprise Architecture):* Kiến trúc của toàn doanh nghiệp, mở rộng theo mức độ áp dụng hướng dịch vụ.

Mô hình dịch vụ (Service Model): là mô hình phân loại dịch vụ theo kiểu logic, mức độ tái sử dụng và mối liên hệ với logic kinh doanh, giúp tổ chức hệ thống thành các nhóm chức năng rõ ràng. Mô hình dịch vụ giúp quản lý và phân loại dịch vụ dựa trên tính tái sử dụng, phạm vi ứng dụng và khả năng tương tác, góp phần xây dựng các lớp dịch vụ, giúp hệ thống dễ mở rộng và tái sử dụng.

Các loại Service Model phổ biến:

1. *Task Service:* Dịch vụ không có tính chất chung (*non-agnostic*), phục vụ mục tiêu cụ thể của một quy trình kinh doanh.
2. *Microservice:* Một phiên bản nhỏ gọn của Task Service, nhưng với chức năng đơn lẻ và độc lập hơn, phù hợp cho các yêu cầu cụ thể trong quy trình.
3. *Entity Service:* Dịch vụ có tính chất chung (*agnostic*), xử lý logic liên quan đến một hoặc nhiều thực thể kinh doanh, với khả năng tái sử dụng cao trong nhiều quy trình.
4. *Utility Service:* Dịch vụ cung cấp các chức năng công nghệ chung, có tính chất chung (agnostic), không liên quan trực tiếp đến thực thể hoặc quy trình kinh doanh.

Service Layers (Lớp Dịch Vụ) : Dựa trên mô hình dịch vụ, các dịch vụ được tổ chức thành các lớp dựa trên ngữ cảnh logic và mức độ sử dụng lại. Mỗi lớp dịch vụ đại diện cho một nhóm dịch vụ có cùng tính chất, giúp hệ thống trở nên có tổ chức và dễ bảo trì.

Các lớp dịch vụ phổ biến:

1. *Task Service Layer*: Nhóm các dịch vụ thực hiện các tác vụ cụ thể.
2. *Entity Service Layer*: Nhóm các dịch vụ liên quan đến các thực thể.
3. *Utility Service Layer*: Nhóm các dịch vụ cung cấp chức năng tiện ích.
4. *Microservice Layer*: Nhóm các dịch vụ microservices cho phép độc lập triển khai.

Ngữ cảnh Bất khả tri và khả tri (Agnostic vs. Non-Agnostic Context):

1. Logic bất khả tri *(Agnostic logic):* Là logic có thể tái sử dụng trong nhiều quy trình khác nhau, không phụ thuộc vào ngữ cảnh cụ thể.
2. Logic khả tri *(Non-Agnostic logic):* Là logic chỉ được sử dụng cho các quy trình cụ thể, có tính chất riêng biệt.

Các bước phân tích và phân rã mô hình kinh doanh:

1. *Phân rã chức năng (Functional Decomposition):* Chia nhỏ các bước trong mô hình kinh doanh thành các hành động chi tiết để xác định ngữ cảnh và ranh giới dịch vụ.
2. *Đóng gói dịch vụ (Service Encapsulation):* Xác định phần logic phù hợp để đóng gói trong các dịch vụ.
3. *Tách ngữ cảnh bất khả tri (Agnostic Context):* Tách logic không đặc thù thành các ngữ cảnh có khả năng tái sử dụng.
4. *Xác định khả năng bất khả tri (Agnostic Capability):* Sắp xếp logic thành các khả năng đa mục đích trong ngữ cảnh bất khả tri.
5. *Trừu tượng hóa tiện ích (Utility Abstraction):* Xác định ngữ cảnh chức năng chuyên biệt cho các tiện ích chung, hình thành lớp dịch vụ tiện ích.
6. *Trừu tượng hóa thực thể (Entity Abstraction):* Định hình ngữ cảnh dịch vụ liên quan đến thực thể kinh doanh cụ thể, tạo lớp dịch vụ thực thể.
7. *Tách ngữ cảnh khả tri (Non-Agnostic Context):* Tách logic đặc thù cho quy trình kinh doanh, được phân loại là khả tri.
8. *Trừu tượng hóa vi dịch vụ (Microservices Abstraction):* Đưa các tác vụ có yêu cầu phi chức năng đặc thù như hiệu năng vào lớp dịch vụ microservices.
9. *Trừu tượng hóa quy trình và dịch vụ tác vụ (Process Abstraction and Task Services):* Đóng gói logic đặc thù quy trình thành lớp dịch vụ tác vụ, chịu trách nhiệm điều khiển và tự động hóa quy trình.

1.1.2. Câu hỏi trắc nghiệm

Câu 1. Đặc điểm nào sau đây không nằm trong các đặc trưng của mô hình SOA?
A. Phát triển mô hình riêng lẻ
B. Hướng mô hình kinh doanh
C. Lấy doanh nghiệp làm trung tâm
D. Nhà cung cấp trung lập

Câu 2. Đâu là một trong những loại SOA phổ biến?
A. Kiến trúc dịch vụ
B. Kiến trúc kho dịch vụ
C. Kiến trúc doanh nghiệp hướng dịch vụ
D. Tất cả đáp án trên

Câu 3. SOA là viết tắt của từ gì?
A. Service - Orienting Architecture
B. Service - Orientation Architecture
C. Society of Architecture
D. Service - Oriented Architecture

Câu 4. Đâu không phải là đặc điểm của SOA?
A. Business driven
B. Flexibility
C. Neutral Driven
D. Composition Centric

Câu 5. Loại SOA nào được sử dụng phổ biến?
A. Service Composition
B. Service Inventory
C. Service-Oriented Enterprise
D. Tất cả các loại SOA kể trên

Câu 6. Vì sao đặc điểm hướng mô hình kinh doanh của SOA lại giúp tăng giá trị và tuổi thọ của kiến trúc?
A. Giữ đồng bộ liên tục với cách doanh nghiệp phát triển theo thời gian
B. Làm mô hình kinh doanh phát triển nhanh
C. Hỗ trợ quá trình chuẩn hóa kinh doanh
D. Phát triển theo thời gian

Câu 7. Các giai đoạn đầu tiên khi phân tích kho dịch vụ là gì?
A. Define Technology Architecture
B. Define Enterprise Business Models
C. Perform Service-Oriented Analysis
D. Define Service Inventory Blueprint

Câu 8. Các đặc điểm của SOA?
A. Business driven, Neutral Driven, Composition Centric,Enterprise-Centric
B. Business driven, Neutral Driven, Flexibility,Enterprise-Centric
C. Neutral Driven, Composition Centric,Enterprise-Centric
D. Accuracy, Neutral Driven, Flexibility,Enterprise-Centric

Câu 9. Vì sao khi thiết kế kiến trúc dịch vụ cần lấy doanh nghiệp làm trung tâm?
A. Để dùng chung các service
B. Tránh nguy cơ tạo ra các silo mới trong doanh nghiệp
C. Trách các dịch vụ xa rời mục đích kinh doanh
D. Tuổi thọ kiến trúc thấp

Câu 10. Sắp xếp theo thứ tự quy mô lớn dần các loại SOA phổ biến?
A. Kiến trúc dịch vụ, Kiến trúc thành phần dịch vụ, Kiến trúc kho dịch vụ, Kiến trúc doanh nghiệp hướng dịch vụ
B. Kiến trúc dịch vụ, Kiến trúc kho dịch vụ, Kiến trúc thành phần dịch vụ, Kiến trúc doanh nghiệp hướng dịch vụ

C. Kiến trúc kho dịch vụ, Kiến trúc dịch vụ, Kiến trúc thành phần dịch vụ, Kiến trúc doanh nghiệp hướng dịch vụ
D. Kiến trúc thành phần dịch vụ, Kiến trúc doanh nghiệp hướng dịch vụ, Kiến trúc kho dịch vụ, Kiến trúc dịch vụ

Câu 11. Mục đích của quá trình mô hình hóa dịch vụ là gì?
A. Tổ chức các đơn vị logic thành các giải pháp hướng dịch vụ
B. Phân loại các đơn vị logic dựa trên kích thước của chúng
C. Tạo nhãn cho các lớp hướng dịch vụ
D. Xác định bản chất của các đơn vị logic

Câu 12. Chỉ ra các mô hình dịch vụ phổ biến?
A. Task service, Microservice, Management service, Utility service
B. Task service, Microservice, Entity service, Utility service
C. Task service, Microservice, Entity service, Identity service
D. Routing service, Microservice, Entity service, Utility service

Câu 13. Dịch vụ được xem là "sẵn sàng để ghép nối" khi nó có tính chất gì?
A. Tính tương tác
B. Tính tương thích
C. Tính tương phản
D. Tính tương đồng

Câu 14. Mục đích chính của quá trình mô hình hóa dịch vụ trong kiến trúc dịch vụ (SOA) là gì?
A. Xác định quy trình kết hợp các dịch vụ thành một giải pháp hướng dịch vụ
B. Xác định các yêu cầu và chức năng của các dịch vụ trong hệ thống
C. Tổ chức một lượng lớn các đơn vị logic để cuối cùng chúng có thể được tập hợp lại thành các giải pháp hướng dịch vụ
D. Xác định cấu trúc và quan hệ giữa các dịch vụ trong kiến trúc dịch vụ

Câu 15. Định nghĩa microservice trong kiến trúc hướng dịch vụ (SOA)?
A. Một dịch vụ bất khả tri thường có phạm vi chức năng nhỏ bao gồm logic với các yêu cầu xử lý và triển khai cụ thể
B. Một dịch vụ không bất khả tri thường có phạm vi chức năng nhỏ bao gồm logic với các yêu cầu xử lý và triển khai cụ thể
C. Một dịch vụ lớn được chia nhỏ thành các thành phần nhỏ hơn
D. Một dịch vụ được thiết kế để chạy trên nhiều máy chủ khác nhau

Câu 16. Một dịch vụ khi nào được coi là bất khả tri?
A. Chứa logic có khả năng tái sử dụng bởi nhiều quy trình nghiệp vụ
B. Chức logic không thể tái sử dụng nhưng có thể có tiềm năng tái sử dụng trong giải pháp
C. Khi nó có khả năng triển khai độc lập
D. Chức các logic không thể tái sử dụng bởi các quy trình nghiệp vụ

Câu 17. Mục đích của bước phân tách chức năng trong phá vỡ vấn đề nghiệp vụ là gì?
A. Tăng tính linh hoạt và mở rộng của hệ thống phần mềm

B. Giảm độ phức tạp của hệ thống phần mềm
C. Tối ưu hóa hiệu suất của hệ thống phần mềm
D. Phân tách một vấn đề lớn hơn thành các vấn đề nhỏ hơn mà các đơn vị logic giải pháp tương ứng có thể được xây dựng

Câu 18. Microservice có tính tự chủ cao nhằm mục đích gì?
A. Tăng cường tính linh hoạt của hệ thống
B. Giảm thiểu sự phụ thuộc vào các tài nguyên bên ngoài
C. Đảm bảo tính bảo mật của dịch vụ
D. Tạo điều kiện cho việc triển khai dư thừa

Câu 19. Một trong những đặc điểm cơ bản phân biệt SOA/kiến trúc công nghệ hướng dịch vụ với các dạng kiến trúc phân tán khác là gì?
A. Tính tương thích (Compatibility)
B. Tính linh hoạt (Flexibility)
C. Tính tập trung vào thành phần (Component-centric)
D. Tính phân tán (Decentralization)

Câu 20. Loại dịch vụ nào thường tương ứng với logic quy trình kinh doanh mục tiêu, bao gồm logic ghép nối các dịch vụ khác để hoàn thành nhiệm vụ?
A. Task service
B. Microservice
C. Entity service
D. Utility service

Câu 21. Dịch vụ loại nào thường có phạm vi chức năng nhỏ và có yêu cầu xử lý và triển khai cụ thể?
A. Task service
B. Microservice
C. Entity service
D. Utility service

Câu 22. Dịch vụ loại nào được sử dụng để xử lý các thực thể kinh doanh liên quan, như hóa đơn, khách hàng hoặc yêu cầu?
A. Task service
B. Microservice
C. Entity service
D. Utility service

Câu 23. Dịch vụ loại nào chứa các chức năng cấp thấp liên quan đến công nghệ, như thông báo, ghi nhật ký và xử lý bảo mật?
A. Task service
B. Microservice
C. Entity service
D. Utility service

Câu 24. Dịch vụ thực thể là gì?
A. Dịch vụ không thể tái sử dụng với bối cảnh chức năng khả tri liên quan đến một hoặc nhiều thực thể kinh doanh liên quan

B. Dịch vụ có thể tái sử dụng với bối cảnh chức năng khả tri liên quan đến một hoặc nhiều thực thể kinh doanh liên quan
C. Dịch vụ không thể tái sử dụng với bối cảnh chức năng bất khả tri liên quan đến một hoặc nhiều thực thể kinh doanh liên quan
D. Dịch vụ có thể tái sử dụng với bối cảnh chức năng bất khả tri liên quan đến một hoặc nhiều thực thể kinh doanh liên quan

Câu 25. Dịch vụ tiện ích là gì?
A. Dịch vụ có thể tái sử dụng với bối cảnh chức năng bất khả tri, không được bắt nguồn từ các mô hình và mô hình phân tích kinh doanh
B. Dịch vụ có thể tái sử dụng với bối cảnh chức năng khả tri, không được bắt nguồn từ các mô hình và mô hình phân tích kinh doanh
C. Dịch vụ có thể tái sử dụng với bối cảnh chức năng bất khả tri, bắt nguồn từ các mô hình và mô hình phân tích kinh doanh
D. Dịch vụ có thể tái sử dụng với bối cảnh chức năng khả tri, bắt nguồn từ các mô hình và mô hình phân tích kinh doanh

Câu 26. Phân loại các lớp mô hình dịch vụ?
A. Bất khả tri: Entity Service Layer, Microservices Layer ; Khả tri: Task Service Layer, Utility Service Layer
B. Bất khả tri: Entity Service Layer, Utility Service Layer; Khả tri: Task Service Layer, Microservices Layer
C. Bất khả tri: Task Service Layer, Microservices Layer; Khả tri: Entity Service Layer, Utility Service Layer
D. Bất khả tri: Task Service Layer, Entity Service Layer ; Khả tri: Microservices Layer, Utility Service Layer

Câu 27. Điều kiện để một dịch vụ được coi là bất khả tri?
A. Chứa logic có khả năng tái sử dụng bởi nhiều quy trình kinh doanh
B. Chứa logic được sử dụng trong một quy trình kinh doanh duy nhất
C. Các chức năng được phát triển từ các thực thể
D. Các chức năng chỉ gồm logic không liên quan đến thực thể

Câu 28. Điểm khác nhau giữa dịch vụ khả tri và bất khả tri?
A. Dịch vụ bất khả tri bắt nguồn từ các thực thể. Dịch vụ khả tri không bắt nguồn từ các thực thể
B. Dịch vụ khả tri bắt nguồn từ các thực thể. Dịch vụ bất khả tri không bắt nguồn từ các thực thể
C. Dịch vụ bất khả tri có thể được sử dụng bởi nhiều mô hình nghiệp vụ. Dịch vụ khả tri chỉ có thể sử dụng cho một mô hình nghiệp vụ nhất định
D. Dịch vụ khả tri có thể được sử dụng bởi nhiều mô hình nghiệp vụ. Dịch vụ bất khả tri chỉ có thể sử dụng cho một mô hình nghiệp vụ nhất định

1.1.3. Câu hỏi tự luận

Câu 1. Dịch vụ (Service) là gì?

Câu 2. Hợp đồng dịch vụ (Service Contract) có vai trò gì trong SOA?
Câu 3. Web Service dựa trên SOAP khác gì so với dịch vụ RESTful?
Câu 4. API công khai (Published API) là gì trong bối cảnh hợp đồng dịch vụ?
Câu 5. Hướng dịch vụ (Service Orientation) là gì và vai trò của nó trong thiết kế phần mềm?
Câu 6. Tổ hợp dịch vụ (Service Composition) có vai trò gì trong tự động hóa quy trình kinh doanh?
Câu 7. Kho dịch vụ (Service Inventory) là gì và chức năng của nó trong một doanh nghiệp?
Câu 8. Các nguyên lý thiết kế nào giúp tăng tính tái sử dụng và linh hoạt cho dịch vụ?
Câu 9. Đặc điểm nào của SOA giúp doanh nghiệp tránh phụ thuộc vào nền tảng cụ thể của nhà cung cấp?
Câu 10. Bốn đặc điểm chính của kiến trúc hướng dịch vụ (SOA) là gì?
Câu 11. Kiến trúc dịch vụ (Service Architecture) là gì?
Câu 12. Kiến trúc tổ hợp dịch vụ (Service Composition Architecture) được sử dụng để làm gì?
Câu 13. Kiến trúc kho dịch vụ (Service Inventory Architecture) hỗ trợ điều gì trong SOA?
Câu 14. Kiến trúc doanh nghiệp hướng dịch vụ (Service-Oriented Enterprise Architecture) khác biệt thế nào so với kiến trúc dịch vụ đơn lẻ?
Câu 15. Mô hình dịch vụ (Service Model) đóng vai trò gì trong tổ chức hệ thống?
Câu 16. Task Service và Microservice khác nhau ở điểm nào?
Câu 17. Entity Service có đặc điểm gì giúp nó có khả năng tái sử dụng cao?
Câu 18. Service Layers (Lớp Dịch Vụ) là gì và có tác dụng như thế nào trong hệ thống?
Câu 19. Khác biệt giữa ngữ cảnh bất khả tri (Agnostic Context) và ngữ cảnh khả tri (Non-Agnostic Context) là gì?
Câu 20. Phân rã chức năng (Functional Decomposition) được sử dụng trong bước nào của phân tích mô hình kinh doanh?
Câu 21. Tại sao hợp đồng dịch vụ (Service Contract) phải tuân thủ các tiêu chuẩn và cách tiếp cận contract-first trong SOA? Hãy giải thích ý nghĩa của việc này đối với tính nhất quán và tích hợp dịch vụ.
Câu 22. So sánh tính tái sử dụng của Web Service dựa trên SOAP và RESTful Service. Trong trường hợp nào RESTful Service có thể phù hợp hơn so với SOAP và ngược lại?
Câu 23. Việc sử dụng các dịch vụ không trạng thái (Stateless Services) trong SOA ảnh hưởng như thế nào đến hiệu suất và tài nguyên hệ thống? Nêu các lợi ích và hạn chế của phương pháp này.
Câu 24. Giải thích vì sao các nguyên lý của hướng dịch vụ như Trừu tượng hóa dịch vụ (Service Abstraction) và Khả năng phát hiện của dịch vụ (Service Discoverability) là cần thiết cho quá trình tự động hóa quy trình kinh doanh.
Câu 25. Dựa vào các đặc điểm của SOA, hãy phân tích tại sao SOA có thể là lựa chọn tốt cho doanh nghiệp muốn duy trì tính linh hoạt trong môi trường thay đổi liên tục về công nghệ.

Câu 26. Trong kiến trúc SOA, vì sao cần tập trung vào khả năng tổ hợp dịch vụ (Service Composability)? Điều này ảnh hưởng như thế nào đến quy trình phát triển và vận hành ứng dụng?

Câu 27. Đối với các dịch vụ thực thể (Entity Services) trong một kho dịch vụ, tại sao cần phân biệt giữa các dịch vụ có tính độc lập cao (agnostic) và dịch vụ không độc lập? Điều này giúp ích gì cho việc tái sử dụng và tổ chức dịch vụ?

Câu 28. Nếu một dịch vụ có độ tự chủ (Service Autonomy) cao, điều này có thể gây ra khó khăn gì khi tích hợp với các dịch vụ khác? Hãy thảo luận cách giải quyết vấn đề này.

Câu 29. Làm thế nào để các doanh nghiệp có thể áp dụng SOA để giảm thiểu rủi ro khi cần thay đổi hoặc mở rộng hệ thống? Hãy nêu các bước cần thiết trong quy trình này.

Câu 30. Giải thích vai trò của "phân tách các mối quan tâm" (separation of concerns) trong SOA. Làm thế nào mô hình này có thể giúp doanh nghiệp duy trì hệ thống trong dài hạn?

Câu 31. Trong thực tế, làm thế nào việc sử dụng kiến trúc tổ hợp dịch vụ (Service Composition Architecture) có thể giúp doanh nghiệp phản ứng nhanh với thay đổi trong quy trình kinh doanh?

Câu 32. Khi nào một dịch vụ nên được phân loại vào lớp dịch vụ tiện ích (Utility Service Layer) thay vì lớp dịch vụ thực thể (Entity Service Layer)?

Câu 33. Tại sao microservices có tính chất linh hoạt hơn trong triển khai so với các dịch vụ trong các lớp khác?

Câu 34. Hãy phân tích lợi ích của việc tổ chức các dịch vụ có cùng tính chất thành các lớp dịch vụ khác nhau trong một hệ thống phức tạp.

Câu 35. Trong kiến trúc hướng dịch vụ, việc sử dụng dịch vụ bất khả tri (agnostic) mang lại lợi ích gì khi mở rộng và duy trì hệ thống?

Câu 36. Hãy thảo luận những rủi ro có thể gặp phải khi quá phụ thuộc vào các dịch vụ non-agnostic trong tổ hợp dịch vụ (Service Composition).

Câu 37. Tại sao các bước trừu tượng hóa tiện ích (Utility Abstraction) và trừu tượng hóa thực thể (Entity Abstraction) lại quan trọng trong quá trình tổ chức các dịch vụ trong SOA?

Câu 38. Đóng gói logic đặc thù quy trình vào Task Service có thể tạo ra những thuận lợi gì cho việc tự động hóa quy trình kinh doanh?

Câu 39. Phân tích cách trừu tượng hóa vi dịch vụ (Microservices Abstraction) giúp xử lý hiệu quả các yêu cầu phi chức năng như hiệu năng cao hoặc khả năng mở rộng.

Câu 40. Trong SOA, tại sao cần phân tách các khả năng bất khả tri (Agnostic Capability) thành các dịch vụ có thể tái sử dụng và triển khai độc lập?

1.2. Phân tích và mô hình hóa dịch vụ

1.2.1. Tóm tắt lý thuyết

12 bước phân tích và mô hình hóa dịch vụ với Web Service:

1. *Phân tách quy trình nghiệp vụ:* Chia quy trình thành các bước chi tiết để có thể xác định rõ từng hành động và logic công việc.
2. *Loại bỏ các hành động không phù hợp:* Lọc bỏ những bước thủ công hoặc không cần thiết để tập trung vào các bước cần tự động hóa.
3. *Xác định ứng viên dịch vụ thực thể:* Nhóm các bước theo ngữ cảnh logic, phân loại thành bất khả tri (agnostic) và khả tri (non-agnostic) để chọn ra các dịch vụ thực thể tiềm năng.
4. *Xác định logic quy trình cụ thể:* Gán các bước khả tri vào các quy trình như quy tắc nghiệp vụ, logic có điều kiện và xử lý ngoại lệ.
5. *Áp dụng các nguyên tắc hướng dịch vụ:* Áp dụng các nguyên tắc thiết kế như ít phụ thuộc, trừu tượng hóa, và tự chủ để tối ưu hóa các dịch vụ.
6. *Xác định ứng viên tổ hợp dịch vụ:* Xác định các tổ hợp dịch vụ khả thi để thực hiện các nhiệm vụ chung trong quy trình nghiệp vụ.
7. *Phân tích yêu cầu xử lý:* Xem xét các yêu cầu xử lý của từng dịch vụ để xác định xem có cần trừu tượng hóa thành microservices hoặc dịch vụ tiện ích.
8. *Xác định dịch vụ tiện ích:* Chia nhỏ các bước độc lập thành các dịch vụ tiện ích, xử lý các chức năng chung trong hệ thống.
9. *Xác định ứng viên microservices:* Đánh giá logic cần hiệu suất cao và phi chức năng đặc biệt để tạo các microservices.
10. *Áp dụng lại các nguyên tắc hướng dịch vụ:* Tối ưu hóa các dịch vụ tiện ích và microservices theo hướng dịch vụ.
11. *Sửa đổi tổ hợp dịch vụ:* Tích hợp thêm các dịch vụ mới vào tổ hợp dịch vụ, mở rộng khả năng của tổ hợp.
12. *Rà soát khả năng phân nhóm dịch vụ:* Điều chỉnh việc nhóm các dịch vụ để đảm bảo hiệu quả, bổ sung các dịch vụ nếu cần thiết.

14 bước mô hình hóa dịch vụ với REST Service:

1. *Phân tách quy trình nghiệp vụ:* Chia quy trình thành các bước chi tiết để có thể xác định rõ từng hành động và logic công việc..
2. *Loại bỏ các hành động không phù hợp:* Lọc bỏ những bước thủ công hoặc không cần thiết để tập trung vào các bước cần tự động hóa.
3. *Xác định ứng viên dịch vụ thực thể:* Nhóm các bước theo ngữ cảnh logic, phân loại thành bất khả tri (agnostic) và khả tri (non-agnostic) để chọn ra các dịch vụ thực thể tiềm năng.
4. *Xác định logic cụ thể của quy trình:* Tách logic quy trình thành các dịch vụ tác vụ riêng biệt.

5. *Xác định tài nguyên (resource):* Xác định tài nguyên bất khả tri (agnostic) và khả tri (non-agnostic) cho khả năng tái sử dụng.
6. *Liên kết năng lực dịch vụ với tài nguyên(resource) và phương thức (method):* Liên kết các ứng viên năng lực dịch vụ với tài nguyên và phương thức tương ứng.
7. *Áp dụng các nguyên tắc thiết kế hướng dịch vụ:* Tinh chỉnh và đảm bảo các dịch vụ đáp ứng yêu cầu quy trình.
8. *Xác định ứng viên tổ hợp dịch vụ (service composition):* Xây dựng cấu trúc tổ hợp dịch vụ và mối liên hệ giữa chúng.
9. *Phân tích yêu cầu xử lý:* Xác định logic bổ sung cần thiết để thực hiện chức năng dịch vụ.
10. *Xác định ứng viên dịch vụ tiện ích:* Thiết lập lớp dịch vụ tiện ích và liên kết với tài nguyên và phương thức.
11. *Xác định ứng viên microservice:* Đóng gói logic khả tri (non-agnostic) thành các microservice riêng biệt và liên kết với tài nguyên và phương thức.
12. *Áp dụng lại nguyên tắc thiết kế hướng dịch vụ:* Lặp lại các điều chỉnh mô hình hướng dịch vụ cho các dịch vụ tiện ích.
13. *Sửa đổi tổ hợp ứng viên dịch vụ:* Kết hợp các dịch vụ tiện ích và microservice vào cấu trúc tổ hợp dịch vụ.
14. *Sửa đổi định nghĩa tài nguyên và phân nhóm năng lực:* Rà soát và điều chỉnh tài nguyên và nhóm ứng viên năng lực để đảm bảo tính nhất quán.

1.2.2. Câu hỏi trắc nghiệm

Câu 1. Trong quy trình mô hình hóa Web service, đâu không phải là một cân nhắc khi xác định ứng viên microservice tại bước 9?
A. Yêu cầu về tính tự trị cao
B. Yêu cầu về hiệu suất runtime cụ thể
C. Yêu cầu về phiên bản dịch vụ và triển khai cụ thể
D. Yêu cầu về giao diện người dùng cụ thể

Câu 2. Trong quy trình mô hình hóa Web service, việc rà soát lại nhóm ứng viên capability có thể dẫn đến việc bổ sung các ứng viên nào?
A. Service và service capability
B. Service capability và utility
C. Service và microservice
D. Microservice và utility

Câu 3. Trong quy trình mô hình hóa web service, ứng viên dịch vụ nào được xác định đầu tiên?
A. Entity service
B. Task service

C. Microservice
D. Utility Service

Câu 4. Trong quy trình mô hình hóa Web service, việc soát lại ứng viên tổ hợp dịch vụ có thể thêm những layer nào vào ứng viên tổ hợp dịch vụ?
A. Task service và microservice
B. Microservice và entity service
C. Microservice và utility service
D. Entity service và utility service

Câu 5. Trong quy trình mô hình hóa web service, Đâu không phải là mục đích của bước 6 (Xác định ứng viên tổ hợp dịch vụ)?
A. Cung cấp cái nhìn sâu sắc về cách nhóm các bước quy trình
B. Cho thấy mối quan hệ task service và entity service
C. Xác định các thành phần dịch vụ tiềm năng
D. Giảm chi phí phát triển dịch vụ

Câu 6. Trong quy trình mô hình hóa web service, nguyên tắc nào không được xem xét tại Bước 5 (Áp dụng hướng dịch vụ) khi áp dụng hướng dịch vụ?
A. Tách rời dịch vụ (Service Loose Coupling)
B. Trừu tượng dịch vụ (Service Abstraction)
C. Tự chủ dịch vụ (Service Autonomy)
D. Bảo mật dịch vụ (Service Security)

Câu 7. Yếu tố nào sau đây là một trong những cân nhắc điển hình khi quyết định đóng gói một logic trong kinh doanh thành một microservice riêng biệt?
A. Tính tự chủ cao hơn
B. Thiết kế giao diện người dùng
C. Phát triển và bảo trì mã nguồn dễ dàng
D. Tích hợp với các đối tác bên ngoài

Câu 8. Trong kiến trúc hướng dịch vụ, REST (Representational State Transfer) là gì?
A. Một giao thức truyền tải dữ liệu giữa các dịch vụ
B. Một mô hình kiến trúc phục vụ cho việc truyền tải các tài nguyên qua giao thức HTTP
C. Một phương pháp tạo ra các dịch vụ mạng phân tán
D. Một công nghệ dựa trên SOAP để tạo ra các dịch vụ web

Câu 9. Vai trò của Entity Service trong kiến trúc dịch vụ là gì?
A. Điều phối và thực hiện các nhiệm vụ cụ thể trong quy trình kinh doanh
B. Cung cấp các chức năng liên quan đến CRUD (Create, Read, Update, Delete) cho các thực thể
C. Cung cấp các dịch vụ tiện ích chung mà nhiều dịch vụ khác có thể cần
D. Đơn vị triển khai độc lập, thực hiện một chức năng duy nhất và cụ thể trong hệ thống

Câu 10. Sự khác biệt chính giữa RESTful services và SOAP-based web services trong phân tích hướng dịch vụ là gì?
A. Tối ưu hóa hiệu suất của các dịch vụ
B. Xác định ranh giới chính xác cho các dịch vụ

C. Xác định giao diện của các dịch vụ
D. Xác định các thực thể kinh doanh cần dịch vụ hỗ trợ

Câu 11. Mục tiêu chính của việc triển khai hướng dịch vụ là gì?
A. Tăng cường bảo mật
B. Thực hiện việc tách biệt các vấn đề một cách rõ ràng
C. Để giảm chi phí triển khai
D. Tăng tốc độ xử lý

Câu 12. Bước đầu tiên khi mô hình hóa REST Service?
A. Xác định ứng viên dịch vụ thực thể
B. Xác định tài nguyên
C. Xác định logic cụ thể của quy trình
D. Phân rã quy trình nghiệp vụ

Câu 13. Những hành động nào bị coi là cần loại bỏ trong quy trình mô hình hóa REST Service?
A. Các hoạt động có sự tham gia của con người
B. Các hoạt động có sự tham gia của con người và những logic nghiệp vụ cũ không thể đóng gói vào trong dịch vụ
C. Các hoạt động là những logic nghiệp vụ cũ không thể đóng gói vào trong dịch vụ
D. Đáp án khác

Câu 14. Phương thức HTTP nào thường được sử dụng để truy xuất tài nguyên từ dịch vụ RESTful?
A. GET
B. PUT
C. POST
D. DELETE

Câu 15. Logic khả tri là gì?
A. Logic dành riêng cho một nhiệm vụ có mục đích duy nhất
B. Logic dành riêng cho một nhiệm vụ có đa mục đích
C. Logic đủ chung chung, không cụ thể, không liên quan đến một nhiệm vụ gốc cụ thể
D. Logic đủ chung chung, không cụ thể, dành cho nhiệm vụ có mục đích duy nhất

Câu 16. Logic bất khả tri là gì?
A. Logic đủ chung chung, không cụ thể, không liên quan đến một nhiệm vụ gốc cụ thể
B. Logic đủ chung chung, không cụ thể, dành cho nhiệm vụ có mục đích duy nhất
C. Logic dành riêng cho một nhiệm vụ có mục đích duy nhất
D. Logic dành riêng cho một nhiệm vụ có đa mục đích

Câu 17. REST là viết tắt của gì?
A. Representational State Transfer
B. Remote Execution and Service Transfer
C. Resourceful Entity State Transition
D. Real-time Service Transformation

Câu 18. Trong mô hình REST, tài nguyên được đại diện bởi gì?
A. URI (Uniform Resource Identifier)

B. URL (Uniform Resource Locator)
C. HTML (HyperText Markup Language)
D. XML (eXtensible Markup Language)

Câu 19. Trong mô hình REST, đâu là phương pháp chính để tương tác với tài nguyên?
A. Gửi email
B. Gửi yêu cầu URL
C. Gửi yêu cầu DNS
D. Gửi yêu cầu HTTP

Câu 20. Để mô hình hóa quan hệ giữa các tài nguyên trong dịch vụ REST, ta sử dụng phương pháp nào?
A. Liên kết
B. Truy xuất
C. Truy vấn
D. Thừa kế

Câu 21. Bước đầu tiên trong quá trình mô hình hóa dịch vụ REST là gì?
A. Phân tách quy trình nghiệp vụ
B. Xác định ứng viên dịch vụ thực thể
C. Xác định tài nguyên
D. Xác định logic cụ thể của quá trình

Câu 22. Khi nào một logic khả tri (non-agnostic) có thể được đóng gói thành một microservice riêng biệt?
A. Chi phí triển khai và bảo trì
B. Mức độ độc lập về logic và dữ liệu
C. Yêu cầu hiệu suất và khả năng mở rộng
D. Mức độ phức tạp của dịch vụ

Câu 23. Sau khi phân tách quy trình kinh doanh thành các hành động chi tiết, những loại hành động nào không phù hợp cho các bước mô hình hóa dịch vụ REST tiếp theo và nên được loại bỏ?
A. Đảm bảo tính nhất quán của dữ liệu
B. Tăng cường tính bảo mật của hệ thống
C. Cung cấp một lớp trừu tượng cho các thực thể kinh doanh
D. Tối ưu hóa hiệu suất của hệ thống

Câu 24. Trong phân tích hướng dịch vụ, các agnostic logic trong quy trình kinh doanh là những logic như thế nào?
A. Có khả năng tái sử dụng lại cao trong nhiều quy trình nghiệp vụ
B. Chỉ phục vụ một quy trình nghiệp vụ cụ thể và không thể tái sử dụng trong các quy trình khác
C. Liên quan đến việc xử lý dữ liệu của các hệ thống cụ thể mà không phụ thuộc vào ngữ cảnh nghiệp vụ
D. Yêu cầu sự tùy chỉnh đặc thù cho từng quy trình nghiệp vụ

Câu 25. Khi áp dụng mô hình hóa dịch vụ REST cho quy trình kinh doanh “Xử lý yêu cầu bảo hiểm”, tại sao hành động "Tính toán số tiền bồi thường dựa trên quy định của hợp đồng" được phân loại là non- agnostic?
A. Vì nó không phụ thuộc vào bất kỳ ngữ cảnh cụ thể nào
B. Vì nó có thể tái sử dụng trong các quy trình khác mà không cần sửa đổi
C. Vì hành động này tuân theo logic riêng biệt áp dụng cho hoạt động bảo hiểm
D. Vì nó đòi hỏi sự xác minh của thông tin khách hàng

Câu 26. Khi nào một non-agnostic logic có thể được đóng gói thành một microservice riêng biệt?
A. Khi logic hành động yêu cầu tăng cường tính tự chủ
B. Khi logic hành động yêu cầu ít tài nguyên
C. Khi logic hành động không có yêu cầu đặc biệt về hiệu suất
D. Khi logic hành động không cần phiên bản hóa và triển khai riêng biệt

Câu 27. Task Service đóng vai trò gì trong các hệ thống hướng dịch vụ?
A. Quản lý các thực thể dữ liệu trong cơ sở dữ liệu
B. Điều phối và quản lý các quy trình kinh doanh phức tạp
C. Cung cấp các chức năng tiện ích độc lập
D. Xác định các quan hệ giữa các dịch vụ thực thể

Câu 28. Sau khi phân tách quy trình kinh doanh thành các hành động chi tiết, những loại hành động nào không phù hợp cho các bước mô hình hóa dịch vụ REST tiếp theo và nên được loại bỏ?
A. Các hành động liên quan đến truy cập cơ sở dữ liệu
B. Các hành động yêu cầu sử dụng giao diện người dùng đồ họa
C. Các hành động liên quan đến bảo mật và mã hóa thông tin
D. Các hành động cần phải thực hiện bởi con người hoặc các hệ thống cũ không thể đóng gói thành dịch vụ

Câu 29. Khi xác định tài nguyên, lợi ích chính của việc xác định các tài nguyên đa mục đích (agnostic resources) là gì?
A. Để giảm thiểu chi phí triển khai của dịch vụ
B. Để tăng cường tính bảo mật cho hệ thống
C. Để đánh dấu các phần của doanh nghiệp có khả năng được chia sẻ và tái sử dụng nhiều hơn
D. Để giảm thiểu số lượng lỗi trong quy trình nghiệp vụ

Câu 30. Quá trình mô hình hóa dịch vụ có cần phải đảm bảo tính linh hoạt của hợp đồng dịch vụ?
A. Có, để dễ dàng cập nhật và thay đổi trong quá trình phát triển
B. Không, để đảm bảo tính nhất quán và ổn định của dịch vụ
C. Có, để giảm thiểu sự phức tạp của dịch vụ
D. Không, để giảm thiểu rủi ro và lỗi trong quá trình triển khai

Câu 31. Điều gì **KHÔNG** đúng khi nói về việc xác định các tài nguyên trong giai đoạn mô hình hóa dịch vụ ?
A. Các tài nguyên có thể được biểu diễn bằng dấu gạch chéo làm dấu phân cách

B. Các tài nguyên nên được giữ ở dạng đơn giản trong giai đoạn này
C. Các tài nguyên xác định có thể không liên quan đến các thực thể kinh doanh đã biết
D. Các tài nguyên cần phải tuân thủ hoàn toàn các tiêu chuẩn URL ngay từ đầu

Câu 32. Yếu tố nào **KHÔNG** phải là một yếu tố phổ biến cần tính đến khi nhóm các ứng viên năng lực dịch vụ để hình thành các ranh giới chức năng của dịch vụ?
A. Các ứng viên khả năng dịch vụ có liên quan chặt chẽ với nhau
B. Các ứng viên khả năng dịch vụ là hướng về kinh doanh hay hướng về tiện ích
C. Các loại ngữ cảnh dịch vụ chức năng phù hợp với ngữ cảnh kinh doanh tổng thể của kho dịch vụ
D. Các ứng viên khả năng dịch vụ đều có cùng độ phức tạp

Câu 33. Trong quá trình xác định các tương tác dịch vụ, các tương tác được xác định dựa trên điều gì?
A. Yêu cầu về tài nguyên và phương thức
B. Các tình huống thành công và thất bại có thể xảy ra trong luồng công việc của quy trình
C. Khả năng mở rộng và triển khai
D. Yêu cầu về hiệu suất và độ tin cậy

Câu 34. Sự khác biệt chính giữa RESTful services và SOAP-based web services trong phân tích hướng dịch vụ là gì?
A. RESTful services sử dụng các phương thức HTTP tiêu chuẩn và tập trung vào tài nguyên, trong khi SOAP-based web services sử dụng SOAP và tập trung vào các hoạt động
B. RESTful services sử dụng XML để truyền dữ liệu, trong khi SOAP-based web services sử dụng JSON
C. RESTful services đòi hỏi tuân thủ nhiều tiêu chuẩn WS-*, trong khi SOAP-based web services đơn giản hơn và ít tiêu chuẩn hơn
D. RESTful services không sử dụng HTTP/HTTPS, trong khi SOAP-based web services chỉ sử dụng HTTP/HTTPS

1.2.3. Câu hỏi tự luận

Câu 1. Bước "Phân tách quy trình nghiệp vụ" có vai trò gì trong quy trình phân tích và mô hình hóa dịch vụ?
Câu 2. Tại sao cần loại bỏ các hành động không phù hợp khi mô hình hóa dịch vụ với Web Service?
Câu 3. Ứng viên dịch vụ thực thể (Entity Service Candidate) là gì và được xác định như thế nào?
Câu 4. Logic quy trình cụ thể được xử lý thế nào trong các bước mô hình hóa dịch vụ?
Câu 5. Các nguyên tắc hướng dịch vụ cần áp dụng trong mô hình hóa dịch vụ bao gồm những gì?
Câu 6. Tổ hợp dịch vụ (Service Composition) có vai trò gì trong tự động hóa quy trình nghiệp vụ?

Câu 7. Khi nào các dịch vụ tiện ích (Utility Services) được xác định trong quy trình phân tích và mô hình hóa dịch vụ?
Câu 8. Tại sao các microservice cần hiệu suất cao và xử lý đặc biệt lại được tách riêng?
Câu 9. Bước "Sửa đổi tổ hợp dịch vụ" có ý nghĩa gì trong việc đảm bảo tính liên kết của các dịch vụ trong hệ thống?
Câu 10. Bước "Sửa đổi định nghĩa tài nguyên và phân nhóm năng lực" giúp gì trong việc tối ưu hóa các dịch vụ REST?
Câu 11. Tại sao việc phân tách quy trình nghiệp vụ thành các bước chi tiết lại cần thiết trong phân tích và mô hình hóa dịch vụ?
Câu 12. Lợi ích của việc phân loại dịch vụ thành các nhóm bất khả tri (agnostic) và khả tri (non-agnostic) là gì trong xây dựng kiến trúc SOA?
Câu 13. Áp dụng các nguyên tắc hướng dịch vụ như ít phụ thuộc, trừu tượng hóa và tự chủ có ý nghĩa gì đối với tính ổn định và khả năng mở rộng của dịch vụ?
Câu 14. Tại sao việc phân tích yêu cầu xử lý của từng dịch vụ lại quan trọng trong quá trình xác định các ứng viên dịch vụ tiện ích và microservices?
Câu 15. So sánh cách tiếp cận mô hình hóa dịch vụ của Web Service và REST Service trong việc xác định tài nguyên và phương thức.
Câu 16. Bước xác định tài nguyên trong mô hình hóa dịch vụ REST có vai trò gì trong việc đảm bảo tính nhất quán và khả năng tái sử dụng?
Câu 17. Tại sao bước "Áp dụng lại nguyên tắc thiết kế hướng dịch vụ" lại được lặp lại trong quá trình mô hình hóa dịch vụ?
Câu 18. Làm thế nào việc xác định tổ hợp ứng viên dịch vụ (service composition candidates) giúp tối ưu hóa quy trình nghiệp vụ?
Câu 19. Bước trừu tượng hóa tiện ích và trừu tượng hóa thực thể có tác động thế nào đến tính hiệu quả của hệ thống dịch vụ?
Câu 20. Khi nào cần sửa đổi định nghĩa tài nguyên và phân nhóm năng lực trong mô hình hóa dịch vụ REST? Thay đổi này tác động gì đến tính nhất quán và hiệu quả của hệ thống?
Câu 21. Tại sao REST Service lại phù hợp cho các dịch vụ không trạng thái (stateless services) và cách tiếp cận này có ảnh hưởng gì đến hiệu quả xử lý và tài nguyên hệ thống?
Câu 22. Làm thế nào việc phân chia tài nguyên trong REST Service giúp tăng cường khả năng tái sử dụng và quản lý dịch vụ? Hãy nêu các ví dụ để minh họa.
Câu 23. Việc sử dụng các phương thức HTTP chuẩn như GET, POST, PUT, DELETE trong REST Service ảnh hưởng như thế nào đến tính nhất quán và khả năng mở rộng của API?
Câu 24. Trong những trường hợp nào REST Service có thể gặp hạn chế khi triển khai các dịch vụ yêu cầu sự đảm bảo về tính nhất quán chặt chẽ của dữ liệu?
Câu 25. So sánh cách REST Service và Web Service tiếp cận phân tích và mô hình hóa tài nguyên (resource). Điều này ảnh hưởng như thế nào đến cách chúng tích hợp với các hệ thống khác?

Câu 26. Việc tổ chức REST Service thành các tài nguyên cụ thể có thể tác động đến hiệu quả của tổ hợp dịch vụ (service composition) như thế nào?

Câu 27. Làm thế nào để thiết kế REST Service với khả năng phát hiện (discoverability) cao mà không cần phải sử dụng các công cụ như service registry?

Câu 28. Trong REST Service, làm thế nào để đảm bảo các API không bị lỗi thời nhanh chóng khi hệ thống mở rộng và các yêu cầu mới phát sinh?

Câu 29. Tại sao việc xác định và phân loại tài nguyên bất khả tri (agnostic resources) trong REST Service lại quan trọng để tối ưu hóa khả năng tái sử dụng và giảm chi phí bảo trì?

Câu 30. Khi áp dụng các nguyên tắc thiết kế hướng dịch vụ trong REST Service, yếu tố nào đóng vai trò quan trọng trong việc cải thiện khả năng mở rộng và tính linh hoạt của hệ thống?

1.3. Thiết kế hợp đồng và API dịch vụ

1.3.1. Tóm tắt lý thuyết

Các cân nhắc thiết kế mô hình dịch vụ với web service:

1. *Thiết kế mô hình dịch vụ:* Định nghĩa các thông điệp mà dịch vụ xử lý qua cấu trúc WSDL, tổ chức và định kiểu bằng XML schema.
2. *Thiết kế dịch vụ thực thể (Entity Service):* Đại diện cho các đối tượng kinh doanh chính, đảm bảo dễ mở rộng và duy trì theo chu kỳ sống của thực thể.
3. *Thiết kế dịch vụ tiện ích (Utility Service):* Cung cấp chức năng chung, tập trung vào khả năng tái sử dụng và tính độc lập.
4. *Thiết kế microservice:* Dịch vụ nhỏ, tự động triển khai, mở rộng độc lập, cần mô-đun hóa cao.
5. *Thiết kế dịch vụ tác vụ (Task Service):* Thực hiện quy trình kinh doanh cụ thể, linh hoạt trong phối hợp với các dịch vụ khác.

Hướng dẫn thiết kế Web Service:

1. *Tiêu chuẩn hóa tên:* Đảm bảo tính nhất quán của API dịch vụ.
2. *Mức độ chi tiết API:* Thiết kế mức độ chi tiết phù hợp, tránh lặp lại không cần thiết.
3. *Mở rộng dễ dàng:* Thiết kế để dịch vụ có thể mở rộng mà không ảnh hưởng đến chức năng hiện tại.
4. *WSDL mô-đun:* Tăng cường tái sử dụng và dễ bảo trì.
5. *Quản lý namespaces:* Tránh xung đột và duy trì tính rõ ràng.
6. *Sử dụng thuộc tính SOAP Document và Literal:* Đảm bảo thông điệp SOAP tương thích giữa các phiên bản.

Các cân nhắc thiết kế mô hình dịch vụ REST:

1. *Entity service:* Tập trung vào thực thể, thực hiện CRUD, có tính tái sử dụng cao (agnostic), thường dựa trên phương thức cơ bản và phức tạp.
2. *Utility service:* Cung cấp chức năng chung như logging, thông báo, xử lý lỗi, có tính tái sử dụng cao, không phụ thuộc vào phạm vi chức năng cụ thể.
3. *Microservice:* Thường khả tri (non-agnostic) với số lượng người dùng hạn chế, ít yêu cầu tuân thủ nguyên tắc hướng dịch vụ do không cần tái sử dụng rộng.
4. *Task service:* Tập trung vào logic nghiệp vụ cụ thể, ít tái sử dụng (non-agnostic), tên năng lực dịch vụ thường là động từ mô tả bản chất tác vụ..

Hướng dẫn thiết kế REST Service:

1. *Thiết kế hợp đồng nhất quán:* Đảm bảo hợp đồng dịch vụ đáp ứng các yêu cầu đặc thù của kho dịch vụ và có thể chuẩn hóa, tùy chỉnh các yếu tố như phương thức, kiểu dữ liệu, và xử lý lỗi.
2. *Thiết kế và chuẩn hóa phương thức:* Giới hạn số lượng phương thức và thêm khi cần thiết, đảm bảo tương thích với dịch vụ mới khi kho dịch vụ thay đổi.
3. *Thiết kế và chuẩn hóa HTTP Headers:* Sử dụng headers để cung cấp metadata cho xử lý yêu cầu, như *Accept* cho thương lượng nội dung hoặc *Location* cho mã phản hồi 201.
4. *Thiết kế và chuẩn hóa HTTP Response Codes:* Mã phản hồi phân loại theo điều kiện, từ mã 2xx (thành công) đến mã 5xx (lỗi dịch vụ), giúp dịch vụ xác định lỗi và cách khắc phục.
5. *Tùy chỉnh mã phản hồi:* Tạo mã phản hồi tùy chỉnh dựa theo các nhóm mã HTTP, đảm bảo rõ ràng và hữu ích cho người dùng.
6. *Thiết kế kiểu dữ liệu:* Chọn kiểu dữ liệu có sẵn như *application/json* hoặc *text/plain* cho các đại diện đơn giản, cân nhắc tạo mới nếu cần thiết.
7. *Phương thức phức tạp:* Tùy chỉnh phương thức để tăng cường tính dự đoán và chất lượng dịch vụ, đặc biệt trong môi trường doanh nghiệp.
8. *Phương thức phức tạp không trạng thái:* Ví dụ như phương thức *Fetch* có khả năng tự động thử lại và thương lượng nội dung khi lấy dữ liệu.
9. *Phương thức phức tạp có trạng thái:* Như *Trans* hỗ trợ cam kết hai pha, đảm bảo dữ liệu đồng bộ giữa các dịch vụ thông qua các giao dịch chuẩn bị trước khi cam kết hoặc hoàn tác.

1.3.2. Câu hỏi trắc nghiệm

Câu 1. Tại sao việc đặt tên tiêu chuẩn cho dịch vụ và năng lực dịch vụ là quan trọng?

A. Giúp dịch vụ dễ dàng khám phá và sử dụng lại
B. Tăng tốc độ triển khai
C. Dễ dàng bảo trì
D. Giảm thiểu lỗi

Câu 2. Các mức độ chi tiết khác nhau cần được xem xét khi thiết kế dịch vụ là gì?
A. Mức độ chi tiết năng lực, giá trị thông điệp, dữ liệu, kiểu dữ liệu
B. Mức độ chi tiết dịch vụ, năng lực, dữ liệu, kiểu dữ liệu
C. Mức độ chi tiết dịch vụ, năng lực, giá trị thông điệp, dữ liệu
D. Mức độ chi tiết dịch vụ, năng lực, ràng buộc, dữ liệu
Câu 3. Điều gì cần cân nhắc khi sử dụng không gian tên trong tài liệu WSDL?
A. Tăng số lượng không gian tên
B. Giảm số lượng không gian tên
C. Tổ chức không gian tên một cách cẩn thận và xuyên suốt tài liệu WSDL
D. Sử dụng không gian tên riêng
Câu 4. Hợp đồng dịch vụ Web cần phản ánh chính xác điều gì?
A. Tên thao tác
B. Chức năng và ngữ cảnh của dịch vụ
C. Quy trình nghiệp vụ
D. Tên dịch vụ
Câu 5. Microservices thường sử dụng giao thức nào thay vì SOAP để đạt hiệu suất cao?
A. HTTP
B. REST
C. SMTP
D. FTP
Câu 6. Loại dịch vụ nào ít bị ảnh hưởng nhất bởi các lớp dịch vụ khác?
A. Dịch vụ tiện ích
B. Dịch vụ tác vụ
C. Dịch vụ thực thể
D. Dịch vụ vi mô
Câu 7. Kết hợp thuộc tính nào là ưa chuộng trong thiết kế hợp đồng dịch vụ Web?
A. style:document + use:literal
B. style:document + use:encoded
C. style:RPC + use:encoded
D. style:RPC + use:literal
Câu 8. Thuật ngữ "Service Granularity" đề cập đến điều gì?
A. Phạm vi chức năng của một dịch vụ
B. Mức độ tương tác của người dùng
C. Tần suất cập nhật dịch vụ
D. Băng thông mạng được sử dụng
Câu 9. Thuật ngữ "Service Autonomy" liên quan đến điều gì?
A. Độc lập trong việc thực hiện chức năng
B. Khả năng tương tác với các dịch vụ khác
C. Tốc độ phản hồi dịch vụ
D. Khả năng xử lý dữ liệu theo thời gian thực
Câu 10. Namespaces cần được quản lý như thế nào?
A. Tùy ý, không cần quy tắc

B. Cẩn thận để tránh xung đột
C. Sử dụng duy nhất một namespace cho tất cả
D. Chỉ sử dụng khi cần thiết

Câu 11. Thông điệp SOAP được xử lý trong phần nào của thông điệp SOAP?
A. Header
B. Body
C. Footer
D. Envelope

Câu 12. Tại sao quá trình thiết kế Task Service thường yêu cầu ít nỗ lực hơn so với các mô hình dịch vụ khác?
A. Vì chúng sử dụng nhiều công nghệ tiên tiến
B. Vì chúng có nhiều nhân viên làm việc cùng lúc
C. Vì chúng chỉ yêu cầu một thao tác để khởi động một quy trình logic
D. Vì chúng không cần kiểm tra và giám sát

Câu 13. Điều gì có thể dẫn đến nhu cầu cần phải quản lý trạng thái trong các Task Service?
A. Số lượng lớn người dùng
B. Sự thay đổi của các yêu cầu bảo mật
C. Thiếu tài nguyên hệ thống
D. Sự kết hợp của các dịch vụ có phụ thuộc xử lý

Câu 14. Hợp đồng dịch vụ Web được thiết kế để phản ánh chính xác điều gì?
A. Cách triển khai dịch vụ
B. Ngữ cảnh và chức năng của các ứng viên dịch vụ tương ứng của chúng
C. Phần mềm cơ sở hạ tầng
D. Phân tích và thiết kế hệ thống

Câu 15. Nguyên tắc nào tự nhiên là một phần của mô hình thiết kế thực thể?
A. Service Contract và Service Loose Coupling
B. Service Statelessness và Service Discoverability
C. Service Abstraction và Service Reusability
D. Service Reusability và Service Autonomy

Câu 16. Ngôn ngữ chính dùng để định nghĩa cấu trúc thông điệp trong các dịch vụ web dựa trên SOAP là gì?
A. JSON
B. HTML
C. YAML
D. XML

Câu 17. Hợp đồng dịch vụ Web giúp xác định điều gì?
A. Cách dịch vụ giao tiếp với khách hàng
B. Cách dịch vụ lưu trữ dữ liệu
C. Cách dịch vụ xử lý dữ liệu.
D. Cách dịch vụ bảo mật thông tin

Câu 18. Điều nào sau đây là bước đầu tiên trong thiết kế hợp đồng dịch vụ web?
A. Xác định các cấu trúc thông điệp trong khu vực WSDL types

B. Thiết kế logic dịch vụ tương ứng
C. Xác định quy ước đặt tên cho các hoạt động
D. Định nghĩa endpoint dịch vụ

Câu 19. Sử dụng thuộc tính nào trong cấu trúc ràng buộc WSDL để xác định định dạng tải trọng thông điệp SOAP?
A. WSDL và REST
B. Document và Literal
C. Style và Use
D. SOAP và XML

Câu 20. Thiết kế hợp đồng dịch vụ Web thường bắt đầu bằng việc làm gì?
A. Xác định định nghĩa chính thức về các thông báo của dịch vụ được yêu cầu xử lý
B. Xác định loại giao thức truyền thông
C. Xác định cấu trúc dữ liệu tải trọng
D. Xác định cách triển khai dịch vụ qua REST

Câu 21. Phương pháp nào được khuyến nghị để xây dựng microservices từ góc độ công nghệ?
A. Dịch vụ Web dựa trên SOAP
B. Microservices dựa trên REST
C. Môi trường WS-*
D. Công nghệ gRPC

Câu 22. Quy ước đặt tên dịch vụ thực thể (Entity Service) sử dụng cấu trúc đặt tên nào?
A. Động từ + danh từ
B. Tên thực thể
C. Chỉ bao gồm danh từ
D. Tên hoạt động

Câu 23. Quy ước đặt tên hoạt động dịch vụ cho các dịch vụ thực thể (Service operations for entity services) là gì?
A. Chỉ sử dụng danh từ
B. Sử dụng động từ và không lặp lại tên thực thể
C. Sử dụng động từ + danh từ
D. Sử dụng cấu trúc "động từ + danh từ" hoặc chỉ danh từ

Câu 24. Quy ước đặt tên dịch vụ tiện ích (Utility Service) là gì?
A. Chỉ sử dụng danh từ
B. Sử dụng động từ và không lặp lại tên thực thể
C. Sử dụng động từ + danh từ
D. Sử dụng cấu trúc "động từ + danh từ" hoặc chỉ danh từ

Câu 25. Quy ước đặt tên hoạt động dịch vụ tiện ích (Utility service operations) là gì?
A. Chỉ sử dụng danh từ
B. Sử dụng động từ và không lặp lại tên thực thể
C. Sử dụng động từ + danh từ
D. Sử dụng cấu trúc "động từ + danh từ" hoặc chỉ danh từ

Câu 26. Mức độ chi tiết năng lực dịch vụ (Capability Granularity) được đo bằng gì?

A. Mức độ chi tiết của logic xác thực
B. Phạm vi chức năng của khả năng dịch vụ
C. Số lượng dữ liệu được xử lý
D. Tất cả đều đúng

Câu 27. Mức độ chi tiết dữ liệu (Data Granularity) thể hiện cho điều gì?
A. Phạm vi chức năng của dịch vụ
B. Số lượng logic liên quan đến dịch vụ
C. Lượng dữ liệu được xử lý
D. Ngữ cảnh chức năng tổng thể của dịch vụ

Câu 28. Thường thì mức độ chi tiết của dịch vụ được xác định khi nào?
A. Trong giai đoạn phân tích và mô hình hóa
B. Trong giai đoạn thiết kế hợp đồng dịch vụ
C. Trong giai đoạn kiểm tra hiệu năng
D. Trong giai đoạn triển khai dịch vụ

Câu 29. Tại sao khi thiết kế các hoạt động của dịch vụ web hướng tới có thể mở rộng?
A. Để tăng hiệu suất xử lý dịch vụ web
B. Để giảm độ phức tạp của dịch vụ web
C. Để đáp ứng được yêu cầu mở rộng trong tương lai
D. Để tạo sự linh hoạt trong quản lý dịch vụ web

Câu 30. Việc mở rộng một hợp đồng dịch vụ hiện có thường ảnh hưởng đến điều gì?
A. Các hoạt động dịch vụ
B. Phạm vi chức năng dịch vụ
C. Lược đồ XML tương ứng
D. Dữ liệu xử lý của dịch vụ

Câu 31. Khi nào cần cung cấp các lược đồ XML mới cho một hợp đồng dịch vụ?
A. Khi triển khai dịch vụ lần đầu
B. Khi có yêu cầu mở rộng dịch vụ
C. Khi có thay đổi trong quy trình kinh doanh
D. Khi cần thay đổi hợp đồng dịch vụ hiện có

Câu 32. Sử dụng tài liệu WSDL để làm gì?
A. Để xác định các mô-đun cho các loại, hoạt động và liên kết có thể chia sẻ trên WSDL
B. Để tách riêng các mô-đun XML Schema
C. Để tạo ra một kho lưu trữ trung tâm của schema
D. Để xác định các tài liệu WSDL chính tùy chỉnh

Câu 33. Tính năng nào của WSDL cho phép nhập các mô-đun XML Schema?
A. Import statement
B. Export statement
C. Include statement
D. Module statement

Câu 34. Sử dụng Namespaces trong WSDL giúp thực hiện điều gì?
A. Tạo tài liệu WSDL tổng thể
B. Tạo các quy tắc định dạng cho tài liệu WSDL

C. Tạo tên duy nhất cho các phần tử trong WSDL
D. Xác định phạm vi của dịch vụ web

Câu 35. Giá trị của thuộc tính style trong SOAP có thể là gì?
A. Document
B. RPC
C. Cả A và B đều sai
D. Cả A và B đều đúng

Câu 36. Giá trị của thuộc tính use trong SOAP có thể là gì?
A. Document
B. RPC
C. Cả A và B đều sai
D. Cả A và B đều đúng

Câu 37. Thuộc tính use với giá trị "literal" đại diện cho việc sử dụng hệ thống kiểu dữ liệu nào?
A. Hệ thống kiểu dữ liệu riêng của SOAP
B. Hệ thống kiểu dữ liệu XML Schema
C. Hệ thống kiểu dữ liệu WSDL
D. Hệ thống kiểu dữ liệu RPC

Câu 38. Kết hợp style:document + use:literal được ưu tiên trong SOA vì điều gì?
A. Hỗ trợ việc nhúng toàn bộ tài liệu XML trong phần thân SOAP
B. Hỗ trợ việc giao tiếp RPC truyền thống
C. Hỗ trợ mô hình gửi tin theo kiểu tài liệu là chìa khóa để thực hiện các tính năng của nhiều thông số WS-* khác nhau
D. Hỗ trợ khái niệm của các thông số WS-*

Câu 39. Thuộc tính style với giá trị "document" hỗ trợ việc gì?
A. Nhúng toàn bộ tài liệu XML trong phần thân SOAP
B. Biểu diễn dữ liệu loại tham số
C. Hỗ trợ việc giao tiếp RPC truyền thống
D. Áp dụng các kiểu dữ liệu XML Schema

Câu 40. Khi lắp ráp một WSDL từ các mô-đun, điều gì xảy ra với namespaces?
A. Số lượng namespaces giảm xuống
B. Số lượng namespaces tăng lên
C. Các namespaces không còn cần thiết bị loại bỏ
D. Các namespaces được gộp lại để tạo ra một tài liệu WSDL duy nhất

Câu 41. Quy ước thông thường trong WSDL đối với việc sử dụng targetNamespace là gì?
A. Phải chỉ định targetNamespace cho mỗi phần tử trong WSDL
B. TargetNamespace chỉ được sử dụng cho các schema XML
C. TargetNamespace chỉ được sử dụng cho tên WSDL tổng thể
D. TargetNamespace phải giống nhau cho tất cả các phần tử trong WSDL

Câu 42. Điều gì cần được đảm bảo trước khi thực hiện mở rộng một hợp đồng dịch vụ?
A. Tính nhất quán và dự đoán của SOA
B. Tính mở rộng của các khả năng dịch vụ

C. Các tiêu chuẩn quản lý phiên bản đã được thiết lập
D. Tính mở rộng của phạm vi chức năng dịch vụ

Câu 43. Điều gì có thể xảy ra nếu hợp đồng dịch vụ có mức độ chi tiết thô (coarse)?
A. Nó cung cấp nhiều khả năng tái sử dụng
B. Nó áp đặt việc xử lý hoặc trao đổi dữ liệu dư thừa và khả năng tái sử dụng ít
C. Nó cải thiện hiệu năng
D. Nó hạn chế khả năng tương tác

Câu 44. Hướng dẫn nào dưới đây giúp giải quyết vấn đề về mức độ chi tiết của hợp đồng dịch vụ?
A. Hiểu rõ giới hạn hiệu năng của môi trường triển khai mục tiêu
B. Cung cấp định nghĩa WSDL thay thế cho dịch vụ Web
C. Gán các API có mức độ chi tiết thô cho các dịch vụ được chỉ định là điểm kết hợp giải pháp
D. Hỗ trợ phương thức giao tiếp thứ cấp trong kho dịch vụ

Câu 45. Điều gì quan trọng để một SOA đáp ứng yêu cầu hiệu năng trong khi vẫn duy trì tiêu chuẩn?
A. Tính nhất quán và có thể dự đoán
B. Mức độ chi tiết dịch vụ tốt hơn
C. Giảm số lượng dữ liệu được xử lý
D. Sử dụng một giao thức truyền thông duy nhất

Câu 46. Mối quan hệ giữa data granularity và số lượng dữ liệu được xử lý là gì?
A. Càng chi tiết data granularit, thì càng ít dữ liệu được xử lý
B. Càng đơn giản data granularit, thì càng ít dữ liệu được xử lý
C. Data granularity không ảnh hưởng đến số lượng dữ liệu được xử lý
D. Data granularity không liên quan đến lượng dữ liệu

Câu 47. Tại sao việc xác định mức độ chi tiết dịch vụ (Service Granularity) quan trọng trong quá trình thiết kế?
A. Quyết định mức độ chi tiết của logic xác thực
B. Xác định phạm vi chức năng của dịch vụ
C. Ảnh hưởng đến mô hình hóa và thiết kế vật lý của hợp đồng dịch vụ
D. Tất cả đều đúng

Câu 48. Tiêu chuẩn đặt tên dịch vụ web nhằm mục đích gì?
A. Tạo tính nhất quán và dễ hiểu trong hệ thống
B. Tái sử dụng các đặc điểm tiết lộ quy trình kinh doanh
C. Xác định mô hình thực thể ban đầu của tổ chức
D. Khuyến nghị áp dụng các tiêu chuẩn thiết kế cho microservice

Câu 49. Điều gì làm cho việc thiết kế utility services khó khăn?
A. Không có sự liên quan đến quy trình kinh doanh
B. Được thiết kế dựa trên các yêu cầu công nghệ và thực tế của hệ thống
C. Yêu cầu sử dụng API phức tạp
D. Yêu cầu chuyên môn phân tích nghiệp vụ cao

Câu 50. Tại sao xây dựng một microservice dưới dạng dịch vụ Web dựa trên SOAP không phổ biến?
A. Xử lý liên quan đến nhắn tin SOAP và ngăn xếp công nghệ nhiều lớp của dịch vụ Web và môi trường WS-* có thể gây ra độ trễ
B. Microservices đòi hỏi một công nghệ xếp chồng đa tầng phức tạp
C. Microservices dựa trên REST cung cấp hiệu suất tốt hơn
D. Dịch vụ Web dựa trên SOAP đã lỗi thời

Câu 51. Để làm cho một dịch vụ trở nên dễ tìm kiếm cho các dịch vụ khác, chúng ta có thể sử dụng phương pháp nào?
A. Bổ sung thông tin siêu dữ liệu (metadata)
B. Sử dụng hợp đồng dịch vụ chi tiết
C. Tạo lớp dịch vụ trừu tượng
D. Thực hiện kiểm tra hiệu suất dịch vụ

Câu 52. Trong Utility Service Design, nguyên tắc "Tái sử dụng Dịch vụ" nhấn mạnh vào điều gì?
A. Xây dựng microservice như một dịch vụ web dựa trên SOAP
B. Tạo ra API phong phú
C. Phát triển dịch vụ tiện ích linh hoạt và tái sử dụng
D. Thực hiện các chức năng xử lý cấp thấp

Câu 53. Nguyên tắc "Service Reusability" áp dụng như thế nào trong thiết kế dịch vụ?
A. Thiết kế dịch vụ không phụ thuộc vào nghiệp vụ cụ thể
B. Thiết kế dịch vụ phải phụ thuộc vào nghiệp vụ cụ thể
C. Thiết kế dịch vụ phải thay đổi theo từng lần sử dụng
D. Thiết kế dịch vụ phải tối ưu hóa cho từng tình huống cụ thể

Câu 54. Việc tạo các sơ đồ XML riêng cho từng thành phần dịch vụ giúp đạt được điều gì?
A. Tăng tính độc lập và khả năng tái sử dụng của từng thành phần dịch vụ
B. Giảm thời gian triển khai dịch vụ
C. Đảm bảo rằng tất cả các dịch vụ đều tuân theo cùng một chuẩn
D. Tăng hiệu suất xử lý của dịch vụ

Câu 55. Namespaces cần được quản lý cẩn thận để làm gì?
A. Tránh xung đột
B. Tăng tốc độ xử lý
C. Giảm chi phí phát triển
D. Tăng tính bảo mật

Câu 56. Thiết kế dịch vụ tiện ích tập trung vào yếu tố nào?
A. Tính độc lập và tái sử dụng
B. Tính bảo mật cao
C. Tốc độ xử lý nhanh
D. Khả năng tích hợp

Câu 57. Tại sao việc áp dụng nguyên tắc Lược đồ chuẩn là quan trọng trong thiết kế dịch vụ thực thể?
A. Để đảm bảo tính nhất quán và tính tái sử dụng của lược đồ XML

B. Để giảm thiểu việc lặp lại lược đồ
C. Để tăng cường tính bảo mật của dịch vụ
D. Để tối ưu hóa hiệu suất dịch vụ

Câu 58. Chức năng chính của dịch vụ tác vụ trong thiết kế hướng dịch vụ là gì?
A. Xử lý các chức năng xử lý cấp thấp
B. Điều phối một thành phần dịch vụ cơ sở
C. Quản lý kết nối cơ sở dữ liệu
D. Xử lý phân tích dữ liệu theo thời gian thực

Câu 59. RPC Style trong SOAP tập trung vào gì?
A. Trao đổi tài liệu XML hoàn chỉnh
B. Gọi các hàm từ xa với các tham số
C. Gửi và nhận email
D. Quản lý file hệ thống

Câu 60. Làm thế nào để bổ sung dịch vụ thực thể bằng các chi tiết siêu dữ liệu để làm cho dịch vụ dễ phát hiện hơn đối với người khác?
A. Sử dụng phần tử style
B. Sử dụng phần tử documentation
C. Sử dụng phần tử binding
D. Sử dụng phần tử schema

Câu 61. Mô hình nào được áp dụng khi cần quản lý trạng thái trong các dịch vụ nhiệm vụ để hỗ trợ nguyên tắc Stateless Service?
A. Tất cả các phương án còn lại
B. State Repository
C. Partial State Deferral
D. State Messaging

Câu 62. Điều gì cần cân nhắc khi thiết kế dịch vụ để đảm bảo tính nhất quán và chuẩn hóa?
A. Sử dụng kiểu dữ liệu riêng
B. Giảm độ chi tiết của API
C. Tăng số lượng thao tác
D. Áp dụng các nguyên tắc hướng dịch vụ liên quan đến hợp đồng

Câu 63. Service contract REST được thiết lập dựa trên điều gì trong quá trình phân tích hướng dịch vụ?
A. Hiệu suất của hệ thống
B. Ngữ cảnh chức năng
C. Tính bảo mật của dữ liệu
D. Tính khả dụng của dịch vụ

Câu 64. Các phương thức nào thường được sử dụng trong các hợp đồng dịch vụ thực thể REST?
A. POST và PATCH
B. GET, PUT, DELETE
C. CONNECT và OPTIONS

D. TRACE và HEAD

Câu 65. Khi nào Task service sẽ trả về một kết quả tạm thời phản hồi cho người tiêu dùng trong khi các bước xử lý tiếp theo có thể vẫn cần được thực hiện?

A. Khi dịch vụ tác vụ xử lý các tác vụ đồng bộ
B. Khi dịch vụ tác vụ tự động hóa một long-running business process và các bước xử lý tiếp theo vẫn cần phải diễn ra
C. Khi dịch vụ tác vụ không có tham số hóa
D. Khi dịch vụ tác vụ không cần hỗ trợ giao tiếp không đồng bộ

Câu 66. Kỹ thuật nào có thể được sử dụng để đảm bảo rằng phương thức POST trong dịch vụ tác vụ dựa trên REST hoạt động đáng tin cậy?

A. Chỉ sử dụng các phương thức HTTP khác như GET hoặc PUT
B. Bao gồm các tiêu đề bổ sung và xử lý các thông báo phản hồi, hoặc sử dụng một mã định danh yêu cầu duy nhất do người dùng tạo ra trong mã định danh tài nguyên
C. Tạo ra nhiều phiên bản của dịch vụ tác vụ để tăng độ tin cậy
D. Sử dụng SOAP thay vì REST để truyền tải các thông điệp

1.3.3. Câu hỏi tự luận

Câu 1. Mô hình thiết kế dịch vụ với Web Service được thực hiện như thế nào thông qua WSDL và XML schema?

Câu 2. Tại sao dịch vụ thực thể (Entity Service) cần đảm bảo tính dễ mở rộng và bảo trì trong suốt chu kỳ sống của thực thể?

Câu 3. Dịch vụ tiện ích (Utility Service) có vai trò gì trong việc cung cấp các chức năng chung cho hệ thống?

Câu 4. Đặc điểm nào giúp các microservices có thể triển khai và mở rộng độc lập so với các dịch vụ khác?

Câu 5. Mức độ chi tiết API cần được thiết kế như thế nào để đạt được tính hiệu quả và tránh lặp lại không cần thiết trong hệ thống Web Service?

Câu 6. Tại sao việc chuẩn hóa tên trong thiết kế Web Service lại đóng vai trò quan trọng đối với sự nhất quán và dễ hiểu của API dịch vụ?

Câu 7. Phân tích cách tổ chức WSDL theo dạng mô-đun có thể cải thiện khả năng bảo trì và tái sử dụng dịch vụ như thế nào?

Câu 8. Việc thiết kế Entity Service dựa trên chu kỳ sống của các đối tượng kinh doanh chính giúp hệ thống duy trì tính linh hoạt và mở rộng ra sao?

Câu 9. Làm thế nào để việc quản lý namespaces trong Web Service giúp tránh xung đột và đảm bảo tính rõ ràng trong hợp đồng dịch vụ?

Câu 10. Phân tích vai trò của SOAP Document và Literal trong việc duy trì tính tương thích giữa các phiên bản của dịch vụ Web Service.

Câu 11. Khi nào nên sử dụng dịch vụ tiện ích (Utility Service) thay vì các loại dịch vụ khác để hỗ trợ một quy trình kinh doanh? Lý giải về tính hiệu quả của cách lựa chọn này.

Câu 12. Thiết kế microservice khác gì so với thiết kế Entity Service và Utility Service về mặt mô-đun và tính tự động hóa?
Câu 13. Làm thế nào thiết kế mức độ chi tiết API có thể ảnh hưởng đến hiệu suất và khả năng mở rộng của hệ thống Web Service?
Câu 14. Hãy phân tích cách các dịch vụ tác vụ (Task Service) có thể phối hợp linh hoạt với các dịch vụ khác để hoàn thành các quy trình nghiệp vụ phức tạp.
Câu 15. Việc thiết kế các dịch vụ để có thể mở rộng dễ dàng mà không ảnh hưởng đến chức năng hiện tại có ý nghĩa gì trong việc duy trì tính ổn định của hệ thống?
Câu 16. Entity Service trong REST Service là gì và tại sao nó thường được coi là có tính tái sử dụng cao?
Câu 17. Utility Service trong REST Service có vai trò gì và được sử dụng trong những tình huống nào?
Câu 18. Tại sao Microservice trong REST Service thường có số lượng người dùng hạn chế và ít tuân thủ các nguyên tắc hướng dịch vụ?
Câu 19. Task Service trong REST Service có đặc điểm gì nổi bật về logic nghiệp vụ và tính tái sử dụng?
Câu 20. Khi thiết kế hợp đồng REST Service, tại sao cần duy trì sự nhất quán và tùy chỉnh các yếu tố như phương thức và kiểu dữ liệu?
Câu 21. Tại sao việc thiết kế một hợp đồng dịch vụ nhất quán lại quan trọng trong REST Service và nó có ảnh hưởng thế nào đến khả năng mở rộng của hệ thống?
Câu 22. Phân tích cách chuẩn hóa phương thức trong REST Service có thể giúp duy trì tính tương thích của các dịch vụ khi hệ thống mở rộng.
Câu 23. So sánh cách tiếp cận của Entity Service và Utility Service trong REST Service về khả năng tái sử dụng và độc lập. Tại sao Entity Service thường được sử dụng lại nhiều hơn?
Câu 24. Phân tích cách các HTTP Headers như Accept và Location có thể cải thiện khả năng thương lượng nội dung và quản lý phản hồi trong REST Service.
Câu 25. Khi nào nên sử dụng mã phản hồi tùy chỉnh trong REST Service và việc này có lợi ích gì cho người dùng?
Câu 26. Làm thế nào để phương thức phức tạp không trạng thái (stateless complex methods) trong REST Service như Fetch có thể tăng cường tính dự đoán và chất lượng dịch vụ?
Câu 27. Phân tích lợi ích của các phương thức phức tạp có trạng thái trong REST Service, như phương thức Trans, trong việc duy trì tính nhất quán dữ liệu trong các giao dịch.
Câu 28. Tại sao việc chọn đúng kiểu dữ liệu lại quan trọng khi thiết kế REST Service và những vấn đề nào có thể phát sinh nếu kiểu dữ liệu không phù hợp?
Câu 29. Làm thế nào để các Task Services trong REST Service đáp ứng yêu cầu cụ thể của các quy trình kinh doanh mà không ảnh hưởng đến tính linh hoạt và khả năng mở rộng của hệ thống?

Câu 30. Khi nào nên sử dụng REST Service thay vì Web Service cho một hệ thống cần tích hợp nhiều dịch vụ, và các yếu tố nào nên được xem xét để đảm bảo hiệu quả của thiết kế?

PHẦN 2: MẪU THIẾT KẾ HƯỚNG DỊCH VỤ

2.1. Kiến trúc ứng dụng

2.1.1. Hướng dịch vụ

Bộ phận kỹ thuật được giao nhiệm vụ phát triển một ứng dụng quan trọng, phục vụ kinh doanh cho doanh nghiệp. Ứng dụng cần có khả năng phản ứng nhanh, đáng tin cậy và linh hoạt với các thay đổi liên tục và phức tạp từ yêu cầu kinh doanh, đáp ứng tốt các tiêu chí sau:

- *Tần suất triển khai (Deployment frequency)*: Mức độ thường xuyên mà các thay đổi phần mềm được đưa vào môi trường sản phẩm.
- *Thời gian thực hiện thay đổi (Lead time for changes)*: Khoảng thời gian từ lúc commit đến lúc triển khai.
- *Thời gian khôi phục dịch vụ (Time to restore service):* Thời gian khôi phục dịch vụ sau sự cố hoặc gián đoạn.
- *Tỷ lệ thất bại khi thay đổi (Change failure rate):* Tỷ lệ lỗi hoặc yêu cầu khắc phục khi triển khai thay đổi vào môi trường sản phẩm.

Theo đó, bộ phận kỹ thuật sẽ được tổ chức thành các nhóm nhỏ, đa chức năng và ít phụ thuộc. Mỗi nhóm cần thành thạo phát triển và triển khai liên tục, đảm bảo các thay đổi nhỏ, thường xuyên được kiểm thử qua quy trình triển khai tự động trước khi đưa vào sản phẩm. Mỗi nhóm phụ trách một hoặc nhiều miền con (*subdomain***)** – đại diện cho một mô hình triển khai của một phần chức năng nghiệp vụ (khả năng kinh doanh), bao gồm logic nghiệp vụ và các thực thể liên quan (gọi là một tập hợp trong thiết kế hướng miền - DDD) thực hiện các quy tắc nghiệp vụ.

Câu hỏi: Làm thế nào để tổ chức các subdomain thành các thành phần có thể triển khai và thực thi hiệu quả?

2.1.2. Mỗi dịch vụ một cơ sở dữ liệu

Một ứng dụng được yêu cầu phát triển sử dụng mô hình kiến trúc microserivces. Trong đó, hầu hết các dịch vụ đều cần lưu trữ dữ liệu lâu dài trong một loại cơ sở dữ liệu nào đó. Ví dụ, Order Service lưu trữ thông tin về các đơn hàng, và Customer Service lưu trữ thông tin về khách hàng.

Câu hỏi: Tổ chức mô hình cơ sở dữ liệu trong ứng dụng theo kiến trúc microservice này như thế nào nếu mỗi dịch vụ có một cơ sở dữ liệu riêng?

2.1.3. API Gateway

Một ứng dụng bán sách trực tuyến được xây dựng sử dụng mô hình kiến trúc microservice. Yêu cầu phát triển trang chi tiết sản phẩm hỗ trợ nhiều phiên bản giao diện người dùng:

- UI dựa trên HTML5/JavaScript cho trình duyệt trên máy tính và điện thoại di động
- Ứng dụng di động cho Android và iPhone - các ứng dụng này tương tác với máy chủ thông qua các REST APIs.

Ngoài ra, ứng dụng còn cần phải cung cấp chi tiết sản phẩm qua giao diện (*REST API*) cho các dụng của bên thứ ba sử dụng. Giao diện chi tiết sản phẩm được mô tả có thể hiển thị rất nhiều thông tin về sản phẩm sách. Cụ thể gồm:

- Thông tin cơ bản về sách như tiêu đề, tác giả, giá, v.v.
- Lịch sử mua của bạn về cuốn sách này
- Tình trạng còn hàng
- Các tùy chọn mua
- Các sản phẩm khác thường được mua kèm với cuốn sách này
- Các sản phẩm khác được mua bởi những khách hàng đã mua cuốn sách này
- Đánh giá của khách hàng
- Xếp hạng người bán

Vì cửa hàng trực tuyến sử dụng mô hình kiến trúc microservice, dữ liệu hiển thị trên trang chi tiết sản phẩm được phân tán qua nhiều dịch vụ. Giả sử là:

- Product Info Service - thông tin cơ bản về sản phẩm như tiêu đề, tác giả
- Pricing Service - giá sản phẩm
- Order Service - lịch sử mua của sản phẩm
- Inventory Service - tình trạng sản phẩm
- Review Service - đánh giá của khách hàng
- …

Như vậy, có nghĩa là để hiển thị chi tiết sản phẩm cần phải lấy thông tin từ tất cả các dịch vụ này.

Câu hỏi: Làm thế nào để các client của một ứng dụng dựa trên kiến trúc microservice có thể truy cập vào các dịch vụ riêng lẻ này?

2.2. Phân rã một ứng dụng thành các dịch vụ

2.2.1. Phân rã theo năng lực kinh doanh và theo subdomain

Với yêu cầu phát triển một ứng dụng lớn, phức tạp sử dụng kiến trúc microservice. Mục tiêu: 1) Tổ chức kiến trúc ứng dụng là một tập hợp các dịch vụ ít phụ thuộc lẫn nhau (loosely coupled service) và 2) Đẩy nhanh quá trình phát triển phần mềm bằng cách cho

phép chuyển giao/triển khai liên tục (CD). Kiến trúc microservice sử dụng 03 yếu tố chính gồm (quy trình, tổ chức và kiến trúc phần mềm) để đạt được các mục tiêu trên.

- Đơn giản hóa việc testing và cho phép các component được triển khai độc lập
- Tổ chức nhóm kỹ thuật thành một tập hợp các nhóm nhỏ (6-10 thành viên), tự chủ, mỗi nhóm chịu trách nhiệm cho một hoặc nhiều service.

Có thể đạt được các mục tiêu trên thông qua việc phân tách một cách cẩn thận các chức năng của ứng dụng thành các service. Các nguyên tắc đã biết từ thiết kế hướng đối tượng (Object-oriented design - OOD) là rất hữu ích.

Một service phải đủ nhỏ để được phát triển bởi một nhóm nhỏ và dễ dàng kiểm thử. Nguyên tắc về trách nhiệm duy nhất của một lớp (Single Responsibility Principle - SRP) tuyên bố rằng một lớp nên đóng gói một trách nhiệm duy nhất và chỉ nên có một lý do để thay đổi. Áp dụng nguyên tắc SRP vào thiết kế service là một cách tiếp cận hợp lý. Nó giúp chúng ta thiết kế các service có tính gắn kết cao, thực hiện một tập hợp nhỏ các chức năng có mối liên hệ chặt chẽ với nhau.

Ứng dụng cũng phải được phân rã sao cho hầu hết các yêu cầu mới và thay đổi chỉ ảnh hưởng đến một service duy nhất. Điều này là vì các thay đổi ảnh hưởng đến nhiều service yêu cầu sự phối hợp giữa nhiều nhóm, làm chậm quá trình phát triển. Một nguyên tắc hữu ích khác từ OOD là nguyên tắc thành phần (Common Closure Principle - CCP), nguyên tắc này chỉ ra rằng các lớp thay đổi cùng vì một lý do hay nói cách khác là có tính liên hết cao nên được đặt chung trong một package. Ví dụ, có thể có hai lớp thực hiện các khía cạnh khác nhau của cùng một nguyên tắc kinh doanh. Mục tiêu là khi nguyên tắc đó thay đổi, các nhà phát triển chỉ cần thay đổi mã nguồn trong một số ít - lý tưởng nhất là chỉ một package. Cách suy nghĩ này có ý nghĩa khi thiết kế service vì nó giúp đảm bảo rằng mỗi thay đổi chỉ ảnh hưởng đến một số ít các dịch vụ.

Câu hỏi: Làm thế nào để phân rã một ứng dụng thành các dịch vụ?

Trong đó:

- Đảm bảo kiến trúc hoạt động ổn định
- Các service phải có tính gắn kết. Một service nên thực hiện một tập hợp nhỏ các chức năng liên quan chặt chẽ.
- Các service phải tuân theo CCP - những thứ thay đổi cùng nhau nên được đóng gói cùng nhau - để đảm bảo rằng mỗi thay đổi chỉ ảnh hưởng đến một service
- Các service phải đảm bảo tính linh hoạt, ít phụ thuộc - mỗi service có một API đóng gói việc thực hiện của nó. Việc triển khai các thay đổi, có thể được thực hiện mà không ảnh hưởng đến các consumer.
- Mỗi service phải đủ nhỏ để được phát triển, kiểm thử bởi một nhóm nhỏ khoảng 6-10 người. Mỗi nhóm sở hữu một hoặc nhiều service phải tự chủ. Một nhóm

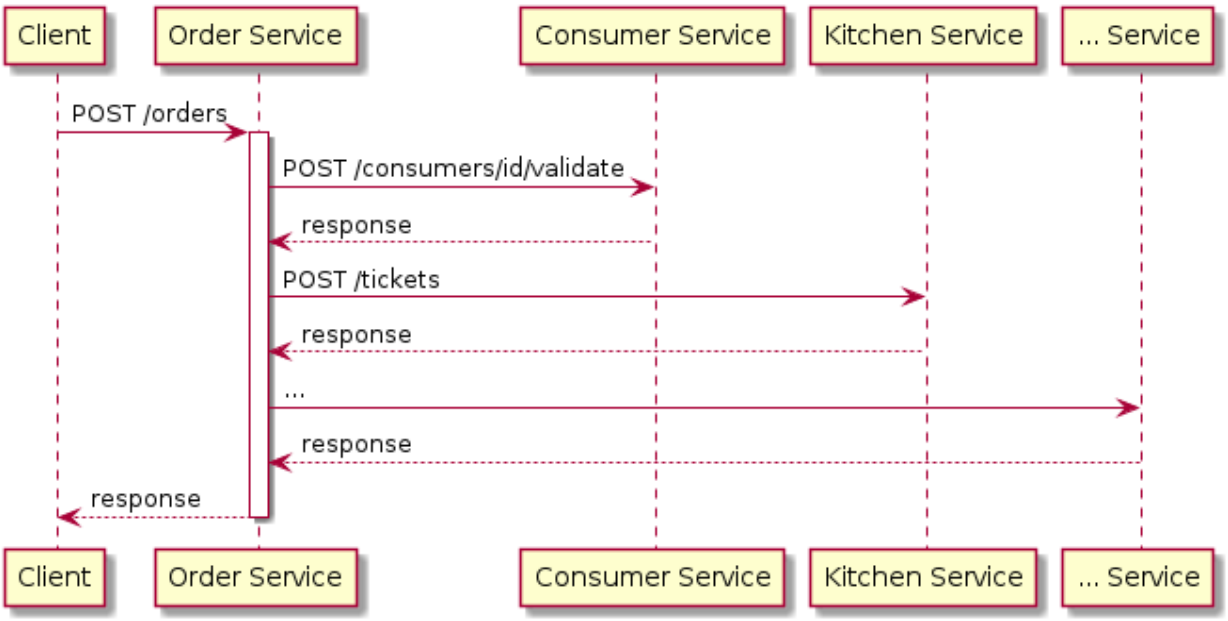
phải có khả năng phát triển và triển khai các service của họ với sự hợp tác tối thiểu với các nhóm khác.

2.2.2. Dịch vụ tự chứa (Self-contained service)

Xem xét một ứng dụng cho phép giao đồ ăn trực tuyến sử dụng kiến trúc hướng dịch vụ. Một client của ứng dụng tạo đơn hàng bằng cách gửi yêu cầu HTTP với phước thức POST /orders và mong đợi phản hồi trong khoảng 600ms. Vì ứng dụng sử dụng kiến trúc microservice, trách nhiệm thực hiện việc tạo đơn hàng được phân tán qua nhiều service. Yêu cầu HTTP POST đầu tiên sẽ được tiếp nhận bởi Order Service rồi sau đó sẽ phải kết nối với các dịch vụ sau:

- Restaurant Service – Chịu trách nhiệm về thực đơn và giá cả của nhà hàng
- Consumer Service - Chịu trách nhiệm về trạng thái của Khách hàng đặt hàng
- Kitchen Service - Chịu trách nhiệm về tạo Ticket chỉ dẫn đầu bếp nấu món gì
- Accounting Service - Chịu trách nhiệm về Xác thực thẻ tín dụng của khách hàng

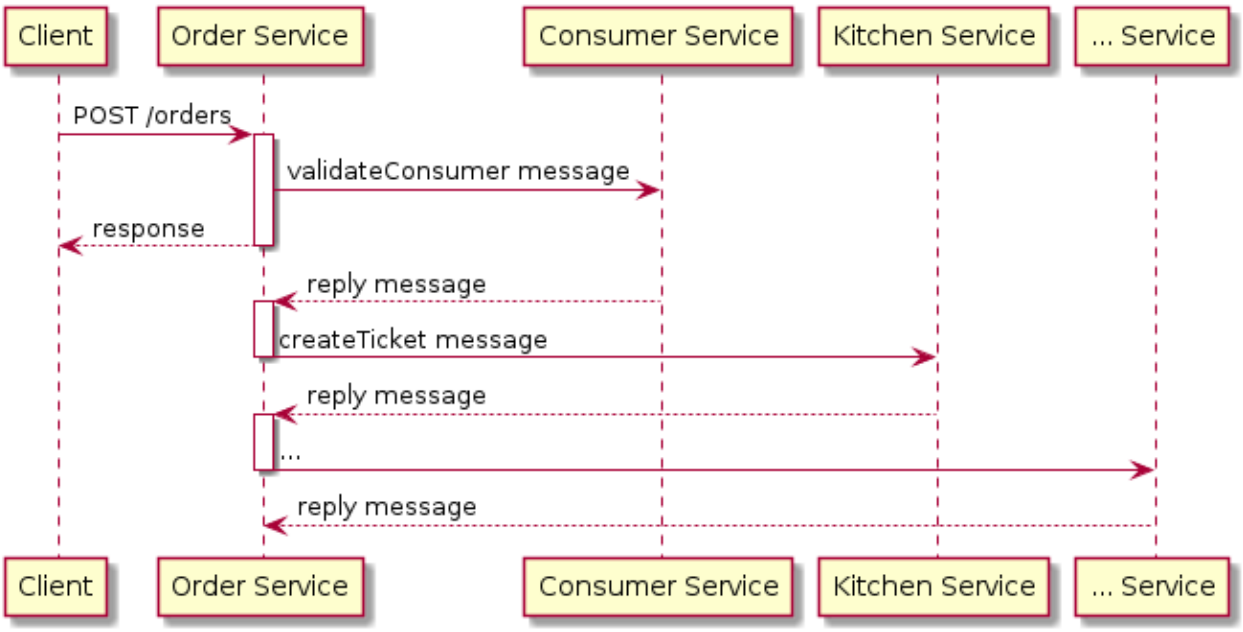
Order Service có thể gọi từng service này bằng cách sử dụng request/response đồng bộ, ví dụ thông qua REST hoặc gRPC. Đặc điểm của các tiếp cận này là Order Service phải đợi phản hồi từ mỗi service trước khi tiếp tục bước tiếp theo.



Hình 1. Minh họa tính năng đặt đồ ăn sử dụng kỹ thuật đồng bộ trong SOA (nguồn: microservice.io)

Nhược điểm chính của việc sử dụng request/response đồng bộ là làm giảm tính sẵn sàng. Nếu bất kỳ service nào trong số các service trong nghiệp vụ không khả dụng, Order Service sẽ không thể tạo đơn hàng và phải trả về lỗi cho khách hàng.

Một cách tiếp cận khác là loại bỏ tất cả giao tiếp đồng bộ giữa Order Service và các service liên quan bằng cách sử dụng mẫu CQRS và SAGA. Order Service tạo một Order ở trạng thái PENDING và gửi lại phản hồi cho POST /order. Sau đó, nó hoàn tất việc tạo đơn hàng bằng cách giao tiếp bất đồng bộ với các service khác.



Hình 2. Minh họa tính năng đặt đồ ăn bất đồng bộ trong SOA (nguồn microservices.io)

Ưu điểm chính của cách tiếp cận này là cải thiện tính sẵn sàng. Order Service luôn có thể phản hồi yêu cầu POST /orders ngay cả khi một trong các service khác không khả dụng. Tuy nhiên, một nhược điểm của việc sử dụng SAGA để hoàn tất việc tạo đơn hàng là phản hồi cho POST không cho khách hàng biết đơn hàng đã được chấp nhận hay chưa. Khách hàng có thể sẽ phải định kỳ gọi GET /orders/{orderId} để biết kết quả.

Câu hỏi: Một dịch vụ nên giao tiếp với các dịch vụ khác thế nào để xử lý một yêu cầu đồng bộ?

2.2.3. Dịch vụ và nhóm phát triển

Một tổ chức phát triển phần mềm gồm nhiều team khác nhau, mỗi team tồn tại lâu dài, nhỏ (khoảng 5-10 thành viên), ít phụ thuộc, tự chủ và đa năng. Luật Conway nói rằng một kiến trúc sẽ phản ánh cấu trúc giao tiếp của tổ chức xây dựng ra nó. Theo đó, một tổ chức gồm các nhóm độc lập như trên cần một kiến trúc ít phụ thuộc tương ứng.

Kiến trúc microservice là một kiến trúc ít phụ thuộc như vậy. Nó cấu trúc ứng dụng thành một tập hợp các dịch vụ liên kết lỏng lẻo, ít phụ thuộc. Các mẫu thiết kế "Phân rã theo subdomain" và "Phân rã theo năng lực kinh doanh" được sử dụng để xác định và tổ

chức các dịch vụ xung quanh mô hình nghiệp vụ. Tuy nhiên, câu hỏi đặt ra là: mối quan hệ giữa dịch vụ và nhóm là gì?

- Một cách tiếp cận là mô hình chia sẻ quyền sở hữu, trong đó nhiều nhóm làm việc trên một dịch vụ nếu cần thiết. Mặt tiêu cực là nó làm tăng sự phối hợp cần thiết giữa các nhóm. Ngoài ra, việc chia sẻ quyền sở hữu code làm tăng nguy cơ chất lượng code kém.
- Một cách tiếp cận tốt hơn, tăng tính tự chủ và liên kết linh hoạt của nhóm, là mô hình sở hữu code/dịch vụ. Nhóm chịu trách nhiệm cho một chức năng/năng lực kinh doanh sở hữu riêng một codebase, bao gồm dạng một hoặc nhiều dịch vụ. Từ đó, nhóm có thể tự do phát triển, kiểm thử, triển khai và mở rộng các dịch vụ của mình. Họ chủ yếu tương tác với các nhóm khác để thống nhất về giao diện API.

Lý tưởng nhất, một nhóm chỉ nên sở hữu một dịch vụ vì điều đó đủ để đảm bảo tính tự chủ và liên kết lỏng lẻo của nhóm, và mỗi dịch vụ bổ sung đều làm tăng độ phức tạp và chi phí. Một nhóm chỉ nên triển khai code của mình dưới dạng nhiều dịch vụ nếu nó giải quyết được vấn đề cụ thể, như giảm đáng kể thời gian triển khai hoặc cải thiện khả năng mở rộng hoặc khả năng chịu lỗi. Vì một nhóm phải nhỏ thì năng lực nhóm có thể hạn chế. Do vậy, để đảm bảo hiệu quả, codebase của họ nên được giới hạn để không vượt quá khả năng của nhóm. Nói cách khác, nó phải "vừa" với khả năng của nhóm. Do đó, thường có một giới hạn trên về kích thước và/hoặc độ phức tạp của một dịch vụ.

Câu hỏi: Mối quan hệ giữa các nhóm và dịch vụ nên được giải quyết/xử lý như thế nào?

2.3. Giao tiếp và triển khai dịch vụ

2.3.1. Tổ hợp API

Các ứng dụng áp dụng mô hình kiến trúc hướng dịch vụ và mỗi dịch vụ là một chương trình độc lập có cơ sở dữ liệu riêng. Điều này dẫn tới, việc thực hiện các truy vấn kết hợp dữ liệu từ nhiều dịch vụ không còn đơn giản như trong kiến trúc nguyên khối.

Câu hỏi: Làm thế nào để thực hiện các truy vấn phức tạp từ nhiều dữ liệu dịch vụ khác nhau trong kiến trúc microservices?

2.3.2. Phân tách trách nhiệm truy vấn (CQRS)

Các ứng dụng áp dụng mô hình kiến trúc hướng dịch vụ và mỗi dịch vụ là một chương trình độc lập có cơ sở dữ liệu riêng. Điều này dẫn tới, việc thực hiện các truy vấn kết hợp dữ liệu từ nhiều dịch vụ không còn đơn giản như trong kiến trúc nguyên khối. Bên cạnh đó, nếu ứng dụng còn triển khai các mô hình thiết kế Event Sourcing, việc truy vấn dữ liệu lại càng khó khăn hơn.

Câu hỏi: Làm thế nào để có thể triển khai được một truy vấn dữ liệu từ nhiều dịch vụ khác nhau trong kiến trúc microservices?

2.3.3. Bản sao phía thực hiện lệnh

Các ứng dụng áp dụng mô hình kiến trúc hướng dịch vụ và mỗi dịch vụ là một chương trình độc lập với cơ sở dữ liệu riêng. Điều này dẫn tới, một dịch vụ khi thực hiện tính năng/lệnh thường cần truy vấn tới các dịch vụ khác. Ví dụ: tính năng createOrder() được triển khai bởi Order Service cần lấy danh sách món ăn của nhà hàng từ Restaurant Service để lấy thông tin giá và xác thực danh mục.

Một lựa chọn xử lý là mỗi lần cần thực hiện Order Service sẽ truy vấn sang Restaurant Service để lấy thông tin. Việc truy vấn có thể theo mô hình đồng bộ hoặc bất đồng bộ sử dụng SAGA. Tuy nhiên, cách tiếp cận này có thể dễ dàng nhìn thấy một vài nhược điểm về việc gia tăng lưu lượng mạng và sự phụ thuộc của Order Service vào Restaurant Service. Một giải pháp thay thế tốt hơn đó là tạo một bản sao danh mục dữ liệu món ăn tích vào Order Service.

Câu hỏi: Làm thế nào để một dịch vụ có thể triển khai một hành động/lệnh lấy dữ liệu từ một dịch vụ khác?

2.3.4. Giao dịch dài hạn (Saga)

Các ứng dụng áp dụng mô hình kiến trúc hướng dịch vụ và mỗi dịch vụ là một chương trình độc lập với cơ sở dữ liệu riêng. Tuy nhiên, một vài giao dịch (transaction) liên quan tới nhiều dịch vụ khác nhau đòi hỏi một cơ chế triển khai đảm bảo các đặc điểm của giao dịch qua nhiều dịch vụ. Ví dụ: Một hệ thống thương mại điện tử theo kiến trúc hướng dịch vụ, phải đảm bảo rằng giá trị đơn hàng (Order Service) không được vượt quá hạn mức tín dụng của khách hàng (Customer Service). Tuy nhiên, hai thông tin này nằm ở hai cơ sở dữ liệu khác nhau, thuộc sở hữu của 02 dịch vụ khác nhau nên không thể cài đặt cơ chế ACID cục bộ dành cho giao dịch.

Câu hỏi: Làm thế nào để triển khai đặc tính giao dịch trên nhiều dịch vụ khác nhau?

2.3.5. Quản lý sự kiện (Domain event)

Một dịch vụ trong kiến trúc microserive thường cần phát đi/thông báo các sự kiện cho các thành phần/dịch vụ khác khi nó cập nhật dữ liệu của mình. Những sự kiện này là rất cần thiết, ví dụ như trong:

- Hệ thống sử dụng CQRS: Để cập nhật dữ liệu bản sao dữ liệu ở phía dịch vụ thực thi lệnh

- Hệ thống sử dụng SAGA: Để phối hợp giữa các giao dịch cục bộ, trực tiếp hoặc thông qua trung gian.

Câu hỏi: Làm thế nào để một dịch vụ phát đi/công bố một sự kiện khi dữ liệu của nó được cập nhật?

2.3.6. Quản lý trạng thái với Event sourcing

Một lệnh dịch vụ (service command) thường cần tạo/cập nhật/xóa một tập hợp dữ liệu trong cơ sở dữ liệu và gửi messages/events đến các message broker. Ví dụ, một service tham gia vào một Saga cần cập nhật các thực thể kinh doanh và gửi messages/events khi thực hiện. Tương tự, một service khi phát đi một sự kiện miền (domain event) phải cập nhật một tập hợp dữ liệu và công bố một sự kiện mỗi khi cập nhật.

Lệnh xử lý dữ liệu này phải thay đổi dữ liệu và gửi thông điệp một cách nguyên tử để tránh các lỗi không nhất quán về dữ liệu và lỗi phần mềm. Tuy nhiên, không khả thi khi sử dụng giao dịch phân tán truyền thống (two-phase commit - 2PC) bao gồm cả cơ sở dữ liệu và message broker. Cơ sở dữ liệu và message broker có thể không hỗ trợ 2PC. Và ngay cả khi chúng hỗ trợ, việc gắn kết dịch vụ với cả cơ sở dữ liệu và message broker thường có thể phát sinh ngoại lệ không mong muốn.

Nhưng nếu không sử dụng 2PC, việc gửi message trong quá trình thực hiện giao dịch không đảm bảo được tính tin cậy. Không có gì đảm bảo rằng giao dịch sẽ được commit. Tương tự, nếu một service gửi một message sau khi commit giao dịch, không có gì đảm bảo rằng service sẽ không gặp sự cố trước khi gửi message.

Ngoài ra, messages phải được gửi đến message broker theo thứ tự mà service đã gửi. Thông thường, chúng phải được phân phối đến từng consumer theo cùng thứ tự, mặc dù điều này nằm ngoài phạm vi của mẫu thiết kế này. Ví dụ, giả sử một aggregate được cập nhật bởi một loạt các giao dịch T1, T2, v.v. Các giao dịch này có thể được thực hiện bởi cùng một phiên bản service hoặc bởi các phiên bản service khác nhau. Mỗi giao dịch phát hành một event tương ứng: T1 -> E1, T2 -> E2, v.v. Vì T1 xảy ra trước T2, event E1 phải được phát hành trước E2.

Câu hỏi: Làm thế nào để cập nhật dữ liệu và gửi message đến message broker một cách nguyên tử (atomically) ?

2.3.7. Shared database

Một ứng dụng được phát triển sử dụng mô hình kiến trúc hướng dịch vụ. Hầu hết các dịch vụ đều cần lưu trữ dữ liệu lâu dài trong một loại cơ sở dữ liệu nào đó. Ví dụ, Order Service lưu trữ thông tin về các đơn hàng, và Customer Service lưu trữ thông tin về khách hàng.

Câu hỏi: Làm thế nào để triển khai cơ sở dữ liệu chia sẻ một cách hiệu quả trong ứng dụng theo kiến trúc hướng dịch vụ?

2.3.8. Xử lý thông điệp trong giao dịch

Một lệnh dịch vụ (service command) thường cần tạo/cập nhật/xóa một tập hợp dữ liệu trong cơ sở dữ liệu và gửi messages/events đến các message broker. Ví dụ, một service tham gia vào mẫu thiết kế SAGA cần cập nhật các thực thể kinh doanh và gửi messages/events khi thực hiện. Tương tự, một service khi phát đi một sự kiện miền (domain event) phải cập nhật một tập hợp dữ liệu có liên quan và public các sự kiện này mỗi khi cập nhật.

Lệnh xử lý dữ liệu này phải thay đổi dữ liệu và gửi thông điệp một cách nguyên tử để tránh các lỗi không nhất quán về dữ liệu và lỗi phần mềm. Tuy nhiên, không khả thi khi sử dụng giao dịch phân tán truyền thống (two-phase commit - 2PC) bao gồm cả cơ sở dữ liệu và message broker. Cơ sở dữ liệu và message broker có thể không hỗ trợ 2PC. Và ngay cả khi chúng hỗ trợ, việc gắn kết dịch vụ với cả cơ sở dữ liệu và message broker thường có thể phát sinh ngoại lệ không mong muốn.

Nhưng nếu không sử dụng 2PC, việc gửi message trong quá trình thực hiện giao dịch không đảm bảo được tính tin cậy. Không có gì đảm bảo rằng giao dịch sẽ được commit. Tương tự, nếu một service gửi một message sau khi commit giao dịch, không có gì đảm bảo rằng service sẽ không gặp sự cố trước khi gửi message.

Ngoài ra, messages phải được gửi đến message broker theo thứ tự mà service đã gửi. Thông thường, chúng phải được phân phối đến từng consumer theo cùng thứ tự, mặc dù điều này nằm ngoài phạm vi của mẫu thiết kế này. Ví dụ, giả sử một aggregate được cập nhật bởi một loạt các giao dịch T1, T2, v.v. Các giao dịch này có thể được thực hiện bởi cùng một phiên bản service hoặc bởi các phiên bản service khác nhau. Mỗi giao dịch phát hành một event tương ứng: T1 -> E1, T2 -> E2, v.v. Vì T1 xảy ra trước T2, event E1 phải được phát hành trước E2.

Câu hỏi: Làm thế nào để cập nhật dữ liệu và gửi message đến message broker một cách nhất quán, nguyên tử (atomically) ?

2.4. Triển khai các dịch vụ của ứng dụng

Một số đặc điểm thường thấy khi quan sát các ứng dụng được phát triển và triển khai theo mô hình hướng dịch vụ:

- Các service có thể được viết bằng nhiều ngôn ngữ, framework, và phiên bản framework khác nhau.

- Mỗi service bao gồm nhiều thể hiện (service instances) để tăng khả năng đáp ứng/xử lý và tính sẵn sàng.
- Mỗi service có khả năng triển khai và mở rộng độc lập. Tương ứng các thể hiện dịch vụ (service instances) được triển khai độc lập với nhau.
- Yêu cầu khả năng xây dựng và triển khai dịch vụ một cách nhanh chóng.
- Yêu cầu về giới hạn tài nguyên (CPU và bộ nhớ) mà một service sử dụng.
- Yêu cầu về quản lý phiên bản dịch vụ.
- Việc triển khai đảm bảo độ tin cậy, hiệu quả về chi phí nhất có thể.

Dưới đây là một số mô hình triển khai dịch vụ phổ biến được xem xét.

2.4.1. Một thể hiện (instance) dịch vụ trên một host

Một ứng dụng được triển khai theo mô hình kiến trúc hướng dịch vụ. Ứng dụng này bao gồm một tập hợp các dịch vụ độc lập. Mỗi dịch vụ được triển khai với nhiều thể hiện (instances) khác nhau để tăng khả năng xử lý và tính sẵn sàng.

Câu hỏi: Các dịch vụ nên được đóng gói và triển khai như thế nào với một thể hiện trên một máy chủ/host?

2.4.2. Nhiều thể hiện (instance) dịch vụ trên một host

Một ứng dụng được triển khai theo mô hình kiến trúc hướng dịch vụ. Ứng dụng này bao gồm một tập hợp các dịch vụ độc lập. Mỗi dịch vụ được triển khai với nhiều thể hiện (instances) khác nhau để tăng khả năng xử lý và tính sẵn sàng.

Câu hỏi: Các service nên được đóng gói và triển khai như thế nào với nhiều thể hiện trên một máy chủ/host?

2.4.3. Một thể hiện (instance) dịch vụ trên một máy ảo

Một ứng dụng được triển khai theo mô hình kiến trúc hướng dịch vụ. Ứng dụng này bao gồm một tập hợp các dịch vụ độc lập. Mỗi dịch vụ được triển khai với nhiều thể hiện (instances) khác nhau để tăng khả năng xử lý và tính sẵn sàng.

Câu hỏi: Các service nên được đóng gói và triển khai như thế nào với một thể hiện trên một máy ảo (VM)?

2.4.4. Một thể hiện (instance) dịch vụ trên một container

Một ứng dụng được triển khai theo mô hình kiến trúc hướng dịch vụ. Ứng dụng này bao gồm một tập hợp các dịch vụ độc lập. Mỗi dịch vụ được triển khai với nhiều thể hiện (instances) khác nhau để tăng khả năng xử lý và tính sẵn sàng.

Câu hỏi: Các service nên được đóng gói và triển khai như thế nào với một thể hiện trên một container?

2.4.5. Triển khai dịch vụ không máy chủ (serverless)

Một ứng dụng được triển khai theo mô hình kiến trúc hướng dịch vụ. Ứng dụng này bao gồm một tập hợp các dịch vụ độc lập. Mỗi dịch vụ được triển khai với nhiều thể hiện (instances) khác nhau để tăng khả năng xử lý và tính sẵn sàng.

Câu hỏi: Các service nên được đóng gói và triển khai như thế nào với môi trường serverless?

2.4.6. Sử dụng các nền tảng triển khai dịch vụ

Một ứng dụng được triển khai theo mô hình kiến trúc hướng dịch vụ. Ứng dụng này bao gồm một tập hợp các dịch vụ độc lập. Mỗi dịch vụ được triển khai với nhiều thể hiện (instances) khác nhau để tăng khả năng xử lý và tính sẵn sàng.

Câu hỏi: Các service nên được đóng gói và triển khai như thế nào thông qua các nền tảng triển khai dịch vụ?

PHẦN III: PHÂN TÍCH THIẾT KẾ HƯỚNG DỊCH VỤ

3.1. Gửi bảng chấm công

Mô tả nghiệp vụ:

Công ty TSL đã thuê một số nhân viên để thực hiện các công việc bảo trì chuyên biệt tại nhiều địa điểm của khách hàng. Nhân viên được yêu cầu điền bảng chấm công hàng tuần, ghi rõ thời gian làm việc tại từng địa điểm. Trước đây, nhân viên kế toán nhập thủ công số giờ làm việc từ lịch hẹn của nhân viên để lập hóa đơn cho khách hàng. Tuy nhiên, do sự không nhất quán giữa số giờ trên bảng chấm công và hóa đơn, công ty quyết định tích hợp hệ thống theo dõi thời gian của bên thứ ba vào hệ thống kế toán. Theo quy trình mới, mọi bảng chấm công đều phải qua xác minh và nếu được xác nhận, hệ thống sẽ tự động tạo hóa đơn cho khách hàng.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống gửi bảng chấm công dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo hệ thống có thể xử lý việc nhập bảng chấm công, xác minh và tự động tạo hóa đơn cho khách hàng.

Mô tả chi tiết các bước nghiệp vụ

1. **Nhân viên nhập thời gian làm việc**: Nhân viên truy cập vào hệ thống và nhập bảng chấm công, ghi rõ số giờ làm việc tại các địa điểm của khách hàng trong tuần.
2. **Gửi bảng chấm công**: Nhân viên gửi bảng chấm công cho hệ thống để bắt đầu quy trình xác minh.
3. **Xác thực thời gian làm việc với hệ thống theo dõi thời gian**: Hệ thống gọi API của hệ thống theo dõi thời gian bên thứ ba để lấy dữ liệu thời gian làm việc thực tế tại các địa điểm của khách hàng.
4. **Đối chiếu dữ liệu thời gian**: Hệ thống đối chiếu số giờ do nhân viên nhập với số giờ thực tế từ hệ thống theo dõi thời gian.
5. **Tính toán giờ hợp lệ**: Nếu số giờ khớp, bảng chấm công sẽ được xác minh là hợp lệ. Nếu không khớp, bảng chấm công bị từ chối.
6. **Thông báo kết quả xác minh**: Hệ thống thông báo cho nhân viên về kết quả xác minh. Nếu bảng chấm công bị từ chối, hệ thống sẽ cung cấp chi tiết về sự không khớp và yêu cầu sửa lại.
7. **Gửi dữ liệu giờ làm việc đã xác minh đến hệ thống kế toán**: Khi bảng chấm công được xác minh thành công, dữ liệu sẽ được gửi đến hệ thống kế toán để lập hóa đơn cho khách hàng.
8. **Lập hóa đơn cho khách hàng**: Hệ thống kế toán tạo hóa đơn dựa trên số giờ làm việc đã được xác minh và gửi đến khách hàng.

9. **Xác nhận từ khách hàng**: Khách hàng xem xét và xác nhận hóa đơn. Nếu khách hàng đồng ý, quá trình hoàn tất. Nếu không, nhân viên có thể phải chỉnh sửa và gửi lại bảng chấm công.
10. **Lưu trữ và báo cáo**: Tất cả các bảng chấm công đã xác minh và hóa đơn được lưu trữ để tạo báo cáo hàng tháng.

3.2. Đăng ký giải thưởng

Mô tả nghiệp vụ:

Một trường đại học muốn triển khai hệ thống cho phép sinh viên đăng ký nhận giải thưởng cho các sự kiện và thành tích cụ thể. Khi sinh viên nộp đơn, hệ thống sẽ xác minh thông tin sự kiện và điều kiện của sinh viên dựa trên các quy định trao giải. Nếu sinh viên đủ điều kiện, giải thưởng sẽ được trao và ghi lại vào bảng điểm cũng như cơ sở dữ liệu giải thưởng của trường. Nếu không, sinh viên sẽ nhận được thông báo từ chối cùng các lý do giải thích.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để thực hiện quy trình đăng ký và trao giải thưởng, từ lúc sinh viên nộp đơn, xác minh sự kiện và điều kiện, đến khi trao giải và ghi nhận thông tin vào hệ thống.

Mô tả chi tiết các bước nghiệp vụ:

1. **Sinh viên khởi tạo đơn xin nhận thưởng**: Sinh viên truy cập hệ thống và nộp đơn xin nhận giải thưởng.
2. **Nhận thông tin chi tiết về sự kiện**: Hệ thống tiếp nhận thông tin sự kiện mà sinh viên đăng ký.
3. **Xác minh chi tiết sự kiện**: Kiểm tra tính hợp lệ và điều kiện của sự kiện liên quan đến giải thưởng.
4. **Kiểm tra điều kiện sự kiện**: Nếu sự kiện không hợp lệ hoặc không đủ điều kiện, hệ thống sẽ kết thúc quy trình.
5. **Nhận thông tin chi tiết về giải thưởng**: Lấy thông tin về loại giải thưởng mà sinh viên đăng ký.
6. **Nhận bảng điểm sinh viên**: Hệ thống lấy thông tin bảng điểm của sinh viên từ cơ sở dữ liệu.
7. **Xác minh điều kiện bảng điểm**: Đối chiếu bảng điểm của sinh viên với các tiêu chí trao giải.
8. **Từ chối nếu không đủ tiêu chuẩn**: Nếu bảng điểm không đủ tiêu chuẩn, hệ thống sẽ từ chối đơn đăng ký.
9. **Xác minh từ chối thủ công**: Quá trình từ chối được xác minh lại thủ công.
10. **Gửi thông báo từ chối**: Gửi thông báo cho sinh viên về lý do từ chối.

11. **Xác minh chấp nhận thủ công**: Nếu đủ tiêu chuẩn, hệ thống xác minh sự chấp nhận giải thưởng thủ công.
12. **Gửi thông báo chấp nhận**: Thông báo chấp nhận được gửi đến sinh viên.
13. **Trao giải thưởng**: Thực hiện trao giải thưởng cho sinh viên.
14. **Ghi lại giải thưởng trong bảng điểm**: Thông tin trao giải được ghi vào bảng điểm của sinh viên.
15. **Lưu giải thưởng vào cơ sở dữ liệu**: Ghi nhận giải thưởng vào cơ sở dữ liệu quản lý giải thưởng của trường.
16. **In bản cứng hồ sơ trao giải thưởng**: Tạo bản cứng hồ sơ trao giải thưởng.
17. **Nộp bản cứng hồ sơ trao giải**: Bản cứng được nộp cho bộ phận quản lý.

3.3. Đặt phòng khách sạn

Mô tả nghiệp vụ:

Hệ thống đặt phòng khách sạn cho phép người dùng tìm kiếm và đặt phòng trực tuyến. Người dùng sẽ cung cấp thông tin cá nhân và lựa chọn phòng tại khách sạn mà họ muốn đặt. Hệ thống cần xác minh xem phòng còn trống hay không, nếu không còn phòng, quy trình sẽ dừng lại. Nếu đặt phòng thành công, hệ thống sẽ gửi thông báo đến người dùng và cập nhật lịch sử đặt phòng.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đặt phòng khách sạn dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo hệ thống có thể xử lý việc nhận thông tin đặt phòng, xác minh tình trạng phòng và gửi thông báo sau khi đặt phòng thành công.

Mô tả chi tiết các bước nghiệp vụ:

1. **Bắt đầu đặt phòng khách sạn**: Người dùng bắt đầu quy trình đặt phòng qua giao diện hệ thống.
2. **Nhận thông tin người dùng đặt phòng**: Hệ thống nhận và lưu trữ thông tin cá nhân của người dùng (họ tên, địa chỉ, số điện thoại...).
3. **Nhận thông tin khách sạn được đặt**: Người dùng lựa chọn khách sạn mà họ muốn đặt phòng.
4. **Nhận thông tin phòng được đặt**: Người dùng chọn phòng cụ thể tại khách sạn.
5. **Xác minh xem còn phòng không**: Hệ thống kiểm tra xem phòng được đặt còn trống hay không.
6. **Nếu không còn phòng, dừng quy trình**: Nếu phòng đã được đặt bởi người khác, quy trình dừng lại và thông báo đến người dùng.
7. **Nếu đặt phòng thành công, gửi thông báo đặt phòng thành công**: Nếu phòng còn trống, hệ thống sẽ hoàn tất việc đặt phòng và gửi thông báo xác nhận đến người dùng.

8. **Cập nhật lại lịch sử đặt phòng**: Lưu thông tin đặt phòng vào lịch sử đặt phòng của người dùng và hệ thống khách sạn.

3.4. Tạo bài thi trắc nghiệm

Mô tả nghiệp vụ:

Hệ thống tạo bài thi trắc nghiệm cho phép giáo viên khởi tạo một quiz với danh sách câu hỏi, cấu hình quiz và danh sách sinh viên tham gia. Giáo viên sẽ import thủ công danh sách câu hỏi, cấu hình bài thi, và danh sách sinh viên. Hệ thống cần xác thực tính hợp lệ của danh sách câu hỏi, cấu hình quiz và danh sách sinh viên. Nếu có bất kỳ lỗi nào về format hoặc dữ liệu không hợp lệ, quy trình sẽ kết thúc và thông báo lỗi cho người dùng.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống tạo bài thi trắc nghiệm dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo hệ thống có thể xử lý việc import danh sách câu hỏi, cấu hình bài thi và danh sách sinh viên, đồng thời xác thực tính hợp lệ của chúng.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo quy trình tạo quiz**: Giáo viên bắt đầu quy trình tạo bài thi trắc nghiệm.
2. **Import thủ công danh sách câu hỏi**: Giáo viên nhập danh sách câu hỏi vào hệ thống.
3. **Xác thực danh sách câu hỏi**: Hệ thống kiểm tra tính hợp lệ của danh sách câu hỏi (format, nội dung...).
4. **Nếu danh sách câu hỏi không hợp lệ**: Quy trình kết thúc và thông báo lỗi.
5. **Import thủ công cấu hình bài quiz**: Giáo viên nhập cấu hình bài thi vào hệ thống.
6. **Xác thực cấu hình của quiz**: Hệ thống kiểm tra tính hợp lệ của cấu hình bài thi.
7. **Nếu cấu hình không hợp lệ**: Nếu cấu hình không hợp lệ, quy trình kết thúc và thông báo lỗi.
8. **Import thủ công danh sách tên hoặc mã sinh viên**: Giáo viên nhập danh sách tên hoặc mã sinh viên tham gia thi.
9. **Xác thực danh sách tên hoặc mã sinh viên**: Kiểm tra tính hợp lệ của danh sách sinh viên.
10. **Nếu danh sách tên hoặc mã sinh viên không hợp lệ**: Quy trình kết thúc và thông báo lỗi.
11. **Nếu tất cả (câu hỏi, cấu hình, danh sách sinh viên) đều hợp lệ**: Xử lý đưa dữ liệu vào hệ thống và sinh mã bài thi quiz
12. **Gửi thông báo thành công:** Gửi mã bài quiz cho sinh viên và kết thúc quy trình.

3.5. Tham gia thi trắc nghiệm

Mô tả nghiệp vụ:

Hệ thống thi trắc nghiệm cho phép sinh viên tham gia làm bài thi trực tuyến. Sinh viên sẽ nhập tên hoặc mã sinh viên để xác thực và truy cập bài thi. Sau khi xác thực, hệ thống sẽ gửi câu hỏi đến sinh viên và nhận câu trả lời. Hệ thống cũng phải đảm bảo thời gian nộp bài hợp lệ, so sánh câu trả lời với đáp án và chấm điểm. Cuối cùng, kết quả sẽ được lưu lại trong hệ thống.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống thi trắc nghiệm dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước xác thực sinh viên, gửi câu hỏi, nhận câu trả lời, chấm điểm và lưu kết quả được thực hiện một cách chặt chẽ và chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo ứng dụng làm bài quiz**: Hệ thống khởi tạo ứng dụng để sinh viên tham gia bài thi trắc nghiệm.
2. **Hiển thị danh sách tên hoặc mã sinh viên chưa tham gia**: Hệ thống hiển thị danh sách sinh viên chưa tham gia để sinh viên chọn.
3. **Chọn thủ công tên hoặc mã sinh viên**: Sinh viên nhập hoặc chọn tên hoặc mã sinh viên của mình.
4. **Xác thực sinh viên**: Hệ thống xác thực sinh viên đang truy cập dựa trên thông tin nhập vào.
5. **Nếu xác thực không thành công, gửi thông báo cho sinh viên**: Nếu không xác thực được, hệ thống sẽ gửi thông báo lỗi cho sinh viên.
6. **Lấy câu hỏi**: Hệ thống lấy câu hỏi của bài thi.
7. **Gửi câu hỏi cho sinh viên**: Hệ thống gửi câu hỏi đến sinh viên tham gia bài thi.
8. **So sánh thời gian nộp câu trả lời với thời gian kết thúc bài thi**: Hệ thống kiểm tra xem thời gian nộp bài của sinh viên có vượt quá thời gian kết thúc hay không.
9. **Nếu thời gian nộp câu trả lời quá thời gian kết thúc bài thi, từ chối nhận câu trả lời và gửi thông báo cho sinh viên**: Nếu quá hạn, hệ thống từ chối nhận câu trả lời và thông báo cho sinh viên.
10. **Lấy đáp án của câu hỏi**: Hệ thống lấy đáp án đúng của câu hỏi.
11. **Chấm điểm**: Hệ thống tính toán điểm cho từng câu hỏi.
12. **Lưu kết quả đáp án của sinh viên**: Hệ thống lưu kết quả cuối cùng của sinh viên sau khi đã chấm điểm.

3.6. Đồng bộ thời khóa biểu

Mô tả nghiệp vụ:

Hệ thống đồng bộ thời khóa biểu cho phép người dùng nhập thông tin tài khoản và đồng bộ thời khóa biểu cá nhân từ hệ thống của trường vào Google Calendar. Sau khi xác minh thông tin tài khoản và quyền truy cập Google Calendar, hệ thống sẽ chuyển đổi thời khóa biểu sang dạng sự kiện trong Google Calendar. Hệ thống cũng cần kiểm tra và loại bỏ các sự kiện trùng lặp trước khi cập nhật thời khóa biểu mới lên Google Calendar.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đồng bộ thời khóa biểu dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ xác minh tài khoản, lấy thời khóa biểu, xác minh quyền truy cập Google Calendar, và đồng bộ hóa lịch trình diễn ra chính xác và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Người dùng nhập thông tin tài khoản và địa chỉ Gmail**: Người dùng nhập thông tin đăng nhập vào hệ thống và cung cấp địa chỉ Gmail.
2. **Xác minh thông tin tài khoản**: Hệ thống xác minh tính chính xác của tài khoản.
3. **Nếu thông tin không chính xác, kết thúc quy trình**: Nếu thông tin tài khoản không hợp lệ, quy trình sẽ dừng lại và thông báo lỗi.
4. **Lấy thời khóa biểu**: Hệ thống truy xuất thời khóa biểu của người dùng từ hệ thống của trường.
5. **Chuyển đổi thời khóa biểu sang sự kiện trong Google Calendar**: Hệ thống chuyển đổi thời khóa biểu thành các sự kiện trong Google Calendar.
6. **Xác minh quyền truy cập Google Calendar của người dùng**: Hệ thống kiểm tra quyền truy cập vào Google Calendar của người dùng.
7. **Nếu không có quyền, kết thúc quy trình**: Nếu không có quyền truy cập, quy trình dừng lại và thông báo lỗi.
8. **Tìm và xóa những thời khóa biểu bị trùng (nếu có) trên Google Calendar**: Hệ thống tìm kiếm và loại bỏ các sự kiện trùng lặp trên Google Calendar.
9. **Cập nhật lên Google Calendar**: Hệ thống cập nhật thời khóa biểu mới lên Google Calendar.
10. **Thông báo cho người dùng qua email**

3.7. Phúc khảo điểm thi

Mô tả nghiệp vụ:

Hệ thống phúc khảo điểm thi cho phép sinh viên gửi đơn yêu cầu phúc khảo, Trung tâm khảo thí (TTKT) tra cứu và kiểm tra bài thi và chuyển sang Bộ môn để phân công Giảng viên thực hiện chấm phúc khảo. Quy trình bao gồm việc lập bảng điểm, xác minh, điều chỉnh nếu cần, và công bố kết quả phúc khảo cho sinh viên trong vòng 15 ngày kể từ khi nhận đơn.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống phúc khảo điểm thi dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo hệ thống có thể xử lý việc kiểm tra tình trạng bài thi, lập bảng điểm phúc khảo, phân công giảng viên chấm phúc khảo, và thông báo kết quả cho sinh viên.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo đơn phúc khảo cho TTKT**: Sinh viên khởi tạo đơn phúc khảo và gửi đến TTKT.
2. **Kiểm tra thời hạn phúc khảo**: Hệ thống kiểm tra xem thời hạn phúc khảo còn hay đã hết.
3. **Nếu còn hạn, khởi tạo thành công**: Đơn phúc khảo được tạo thành công nếu trong thời hạn. Gửi thông báo chấp nhận: Gửi thông báo chấp nhận đơn phúc khảo cho sinh viên.
4. **Nếu hết hạn, khởi tạo thất bại**: Đơn phúc khảo không được tạo nếu đã hết thời hạn. Gửi thông báo từ chối: Gửi thông báo từ chối đơn phúc khảo cho sinh viên.
5. **Nộp tiền mặt cho TTKT**: Sinh viên nộp lệ phí phúc khảo.
6. **Lấy thông tin chi tiết đơn phúc khảo**: Hệ thống lấy thông tin chi tiết của đơn phúc khảo.
7. **Bộ môn phân công giáo viên chấm phúc khảo**: Bộ môn phân công giáo viên thực hiện việc chấm phúc khảo.
8. **Giáo viên nhận bài và chấm thủ công**: Giáo viên đến TTKT nhận bài thi và tiến hành chấm thủ công.
9. **Giáo viên chấm xong, lập bảng điểm chấm phúc khảo**: Giáo viên hoàn thành chấm phúc khảo và lập bảng điểm.
10. **Nếu điểm công bố bằng điểm chấm phúc khảo, giữ nguyên điểm**: Nếu điểm chấm phúc khảo trùng khớp với điểm đã công bố, giữ nguyên điểm.
11. **Nếu chênh lệch lớn hơn 1 điểm, lấy điểm phúc khảo làm chính thức**: Nếu điểm chấm phúc khảo chênh lệch 1 điểm, lấy điểm phúc khảo làm điểm chính thức.
12. **Ghi nhận điểm sau phúc khảo và các thông tin kèm theo**: Hệ thống ghi nhận điểm sau khi phúc khảo và lưu trữ các thông tin liên quan.
13. **Sau 15 ngày, thông báo kết quả phúc khảo**: TTKT thông báo kết quả phúc khảo cho sinh viên sau 15 ngày kể từ khi nhận đơn.

3.8. Đặt hàng

Mô tả nghiệp vụ:

Hệ thống đặt hàng cho phép khách hàng lựa chọn sản phẩm và tiến hành đặt hàng. Hệ thống sẽ nhận thông tin của khách hàng và sản phẩm, kiểm tra yêu cầu đặt hàng và xác minh số lượng sản phẩm trong kho. Nếu số lượng sản phẩm còn đủ, hệ thống sẽ lưu

đơn đặt hàng, cập nhật số lượng hàng tồn kho, và gửi email xác nhận đơn hàng cho khách hàng. Nếu không đủ số lượng sản phẩm, quy trình sẽ dừng lại.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đặt hàng dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ nhận đơn đặt hàng, kiểm tra số lượng sản phẩm, cập nhật kho hàng, đến việc gửi email xác nhận cho khách hàng được thực hiện một cách tự động và chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Bắt đầu nhận đơn đặt hàng**: Hệ thống bắt đầu quy trình đặt hàng khi khách hàng chọn sản phẩm và xác nhận mua hàng.
2. **Kiểm tra người dùng**: Xác minh thông tin tài khoản và quyền hạn của khách hàng. Nếu xác minh thất bại hoặc không đủ quyền hạn, kết thúc quy trình.
3. **Nhận thông tin của khách hàng**: Hệ thống nhận và lưu trữ thông tin cá nhân của khách hàng (tên, địa chỉ, thông tin liên hệ...).
4. **Nhận thông tin của sản phẩm**: Hệ thống nhận thông tin về sản phẩm mà khách hàng muốn đặt (tên sản phẩm, số lượng...).
5. **Kiểm tra yêu cầu từ khách hàng**: Hệ thống kiểm tra tính hợp lệ của yêu cầu đặt hàng (ví dụ: số lượng yêu cầu, thời gian giao hàng...).
6. **Xác minh xem còn đủ số lượng sản phẩm hay không**: Kiểm tra kho hàng để xem số lượng sản phẩm còn đủ để đáp ứng đơn hàng không.
7. **Nếu không đáp ứng thì dừng quy trình**: Nếu số lượng sản phẩm không đủ, quy trình đặt hàng dừng lại và thông báo cho khách hàng.
8. **Lưu bản ghi order vào database**: Nếu đủ số lượng, hệ thống lưu bản ghi đơn hàng vào cơ sở dữ liệu.
9. **Cập nhật lại số lượng sản phẩm trong kho**: Cập nhật số lượng sản phẩm còn lại sau khi trừ đi số lượng đã được đặt.
10. **Xóa sản phẩm khỏi giỏ hàng**: Xóa sản phẩm đã đặt ra khỏi giỏ hàng của khách hàng.
11. **Gửi email thông báo thành công đến khách hàng**: Gửi email xác nhận đơn hàng thành công đến khách hàng.

3.9. Đặt vé xem phim

Mô tả nghiệp vụ:

Hệ thống đặt vé xem phim cho phép khách hàng chọn phim, suất chiếu, ghế ngồi và tiến hành đặt vé trực tuyến. Sau khi khách hàng chọn phim, hệ thống sẽ kiểm tra tình trạng chỗ ngồi, xác minh thông tin vé, và xử lý thanh toán. Nếu đặt vé thành công, hệ thống sẽ gửi thông báo xác nhận vé đã đặt đến email của khách hàng.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đặt vé xem phim dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ khi khách hàng chọn phim, ghế ngồi, kiểm tra tình trạng vé, thanh toán và xác nhận vé được thực hiện chính xác và nhanh chóng.

Mô tả chi tiết các bước nghiệp vụ:

1. **Bắt đầu quy trình đặt vé**: Khách hàng truy cập hệ thống và bắt đầu quy trình đặt vé xem phim.
2. **Chọn phim và suất chiếu**: Khách hàng chọn phim muốn xem, cùng với suất chiếu và rạp phim cụ thể.
3. **Chọn ghế ngồi**: Khách hàng chọn ghế ngồi từ danh sách các ghế còn trống.
4. **Xác minh ghế ngồi còn trống**: Hệ thống kiểm tra tình trạng ghế ngồi đã được chọn xem còn trống hay đã có người đặt.
5. **Nếu không còn ghế ngồi, kết thúc quy trình**: Nếu ghế đã được đặt bởi người khác, hệ thống thông báo và dừng quy trình.
6. **Nhập thông tin khách hàng**: Khách hàng cung cấp các thông tin cá nhân như họ tên, số điện thoại và email.
7. **Xác minh thông tin vé**: Hệ thống xác minh thông tin vé xem phim và các chi tiết liên quan.
8. **Nhập và thực hiện thanh toán**: Khách hàng tiến hành thanh toán trực tuyến thông qua các cổng thanh toán.
9. **Nếu thanh toán thành công, gửi thông báo xác nhận**: Nếu giao dịch thành công, hệ thống gửi thông báo xác nhận vé đã đặt đến email của khách hàng.
10. **Cập nhật tình trạng ghế ngồi**: Hệ thống cập nhật tình trạng ghế đã được đặt vào cơ sở dữ liệu.
11. **Lưu thông tin vé và khách hàng vào cơ sở dữ liệu**: Hệ thống lưu thông tin vé đã đặt và thông tin khách hàng vào cơ sở dữ liệu để quản lý.

3.10. Đăng ký môn học A

Mô tả nghiệp vụ:

Hệ thống đăng ký môn học cho phép giảng viên khởi tạo và quản lý lịch học. Quy trình bao gồm việc xác minh sự kiện đăng ký có hợp lệ hay không, nhận thông tin về môn học, giáo viên, phòng học, và kiểm tra các nhóm lớp tín chỉ đã tồn tại, trùng lặp lịch học. Thực hiện gửi thông báo về việc thêm lịch học thành công hoặc thất bại và thêm vào cơ sở dữ liệu nếu lịch học hợp lệ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đăng ký môn học dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ khởi tạo yêu cầu đăng ký, chọn môn học, kiểm tra lịch học, xác minh dữ liệu và cập nhật cơ sở dữ liệu được thực hiện một cách chính xác và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo yêu cầu đăng ký lịch học**: Hệ thống bắt đầu khởi tạo quy trình đăng ký môn học khi giảng viên khởi tạo yêu cầu.
2. **Nhận thông tin chi tiết về sự kiện**: Hệ thống nhận thông tin chi tiết đăng ký (thời gian, môn học, giáo viên...).
3. **Xác minh sự kiện**: Hệ thống xác minh xem đăng ký có hợp lệ không và có trong khoảng thời gian cho phép không.
4. **Nếu đăng ký không hợp lệ hoặc ngoài thời gian thì kết thúc quá trình**: Nếu không hợp lệ, quy trình dừng lại và gửi thông báo đăng ký thất bại.
5. **Nhận thông tin về giáo viên, môn học, phòng học**: Hệ thống nhận thông tin chi tiết về giáo viên, môn học, và phòng học từ giảng viên.
6. **Nhập môn học, giáo viên cần đăng ký**: Giảng viên nhập môn học từ danh sách hoặc theo mã
7. **Kiểm tra và hiển thị danh sách**: Hệ thống kiểm tra xem lớp tín chỉ với môn học và giáo viên đã tồn tại chưa, nếu có thì hiển thị các nhóm đã tồn tại.
8. **Giảng viên chọn nhóm hoặc tạo nhóm mới**: Giảng viên có thể chọn nhóm lớp tín chỉ hiện có hoặc tạo nhóm mới.
9. **Giảng viên chọn phòng học và lịch học**: Giảng viên chọn phòng học và thời gian học cho lớp.
10. **Xác minh trùng lặp lịch học**: Hệ thống kiểm tra xem lịch học mới có trùng (thời gian, lớp, địa điểm) không.
11. **Nếu trùng gửi thông báo bị trùng**: Nếu lịch học bị trùng, hệ thống gửi thông báo cho giảng viên.
12. **Nếu không trùng gửi thông báo thêm thành công**: Nếu lịch học không trùng, hệ thống gửi thông báo thành công.
13. **Thêm lịch học vào cơ sở dữ liệu**: Hệ thống cập nhật lịch học vào cơ sở dữ liệu.

3.11. Đăng ký môn học B

Mô tả nghiệp vụ:

Hệ thống đăng ký môn học B cho phép sinh viên kiểm tra và đăng ký các môn học trong kỳ học. Quy trình bắt đầu bằng việc kiểm tra danh sách các môn học đã đăng ký, xác minh xem tổng số tín chỉ có đủ điều kiện tối thiểu của kỳ học hay không. Nếu sinh viên đáp ứng đủ điều kiện, hệ thống sẽ cho phép đăng ký môn học. Trong quá trình đăng ký, hệ thống kiểm tra các điều kiện về trùng lịch học, số lượng sinh viên trong lớp học phần, và

tổng số tín chỉ. Cuối cùng, hệ thống sẽ lưu danh sách đăng ký môn học nếu tất cả các điều kiện đều hợp lệ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đăng ký môn học B dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ kiểm tra điều kiện đăng ký, chọn lớp học phần, xác minh lịch học và tín chỉ, đến khi lưu thông tin đăng ký môn học được thực hiện chính xác và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Bắt đầu đăng ký môn học**: Sinh viên bắt đầu quy trình đăng ký môn học cho kỳ học.
2. **Kiểm tra danh sách đăng ký môn học kỳ học của sinh viên**: Hệ thống kiểm tra danh sách các môn học mà sinh viên đã đăng ký trong kỳ học hiện tại.
3. **Kiểm tra điều kiện tín chỉ tối thiểu của kỳ học**: Hệ thống kiểm tra xem danh sách đăng ký môn học của sinh viên có đáp ứng đủ số tín chỉ tối thiểu của kỳ học không.
4. **Nếu không đủ tín chỉ, từ chối đăng ký môn học**: Nếu sinh viên không đáp ứng điều kiện tín chỉ tối thiểu, hệ thống từ chối cho phép đăng ký thêm môn học.
5. **Nếu đủ tín chỉ, cho phép đăng ký môn học**: Nếu đáp ứng đủ điều kiện tín chỉ, sinh viên được phép tiếp tục đăng ký môn học.
6. **Sinh viên chọn lớp học phần**: Sinh viên lựa chọn lớp học phần từ danh sách các môn học đã đăng ký cho kỳ học.
7. **Kiểm tra trùng lịch học**: Hệ thống kiểm tra xem lịch học của lớp học phần mới có trùng với các lớp đã đăng ký hay không.
8. **Kiểm tra số lượng sinh viên trong lớp học phần**: Hệ thống kiểm tra xem số lượng sinh viên trong lớp học phần đã đủ hay chưa.
9. **Nếu số lượng sinh viên trong lớp đã đầy, từ chối đăng ký**: Nếu lớp học phần đã đủ số lượng sinh viên, hệ thống từ chối cho phép đăng ký.
10. **Kiểm tra tổng số tín chỉ đã chọn**: Hệ thống kiểm tra xem tổng số tín chỉ mà sinh viên đã chọn có đạt yêu cầu tối thiểu không.
11. **Gửi thông báo chưa đủ tín chỉ, từ chối lưu**: Nếu tổng số tín chỉ chưa đủ điều kiện, hệ thống gửi thông báo và từ chối lưu đăng ký.
12. **Nếu đủ tín chỉ, lưu danh sách đăng ký môn học**: Nếu tổng số tín chỉ đã đạt yêu cầu, hệ thống lưu danh sách đăng ký môn học của sinh viên.

3.12. Nhập điểm sinh viên

Mô tả nghiệp vụ:

Hệ thống nhập điểm sinh viên cho phép giảng viên nhập điểm cho các môn học mà họ giảng dạy. Quy trình bao gồm việc kiểm tra file điểm nhập vào, xác minh kỳ học, môn học và lớp học phần có thuộc quyền giảng dạy của giảng viên hay không, xác thực định

dạng điểm, và cuối cùng là lưu bản ghi điểm vào cơ sở dữ liệu nếu tất cả các điều kiện đều hợp lệ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống nhập điểm sinh viên dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ việc kiểm tra file điểm, xác minh môn học và lớp học phần, đến việc lưu điểm vào cơ sở dữ liệu diễn ra chính xác và tuân thủ quy định.

Mô tả chi tiết các bước nghiệp vụ:

1. **Bắt đầu nhập điểm**: Giảng viên khởi tạo quy trình nhập điểm cho kỳ học và môn học mà họ đang dạy.
2. **Xác định thời gian nhập điểm**: Hệ thống kiểm tra thời gian nhập điểm có hợp lệ hay không.
3. **Nếu ngoài thời gian quy định, không cho phép nhập**: Nếu quá thời gian cho phép, quy trình sẽ dừng lại và hiển thị thông báo lỗi.
4. **Nhập file điểm**: Giảng viên nhập file điểm vào hệ thống.
5. **Lấy ra kỳ học trong file**: Hệ thống trích xuất thông tin kỳ học từ file điểm.
6. **Xác minh kỳ học**: Hệ thống kiểm tra xem kỳ học trong file có khớp với kỳ học mà giảng viên đang dạy hay không.
7. **Lấy ra môn học trong file**: Hệ thống trích xuất thông tin môn học từ file điểm.
8. **Xác minh môn học**: Hệ thống kiểm tra xem môn học có thuộc quyền giảng dạy của giảng viên không.
9. **Nếu môn học không đúng, từ chối và gửi thông báo**: Nếu môn học không đúng, hệ thống từ chối và gửi thông báo từ chối cho giảng viên.
10. **Lấy ra lớp học phần trong file**: Hệ thống trích xuất thông tin lớp học phần từ file điểm.
11. **Xác minh lớp học phần**: Hệ thống kiểm tra xem lớp học phần có thuộc quyền giảng dạy của giảng viên không.
12. **Nếu lớp học phần không đúng, từ chối và gửi thông báo**: Nếu lớp học phần không đúng, hệ thống từ chối và gửi thông báo từ chối cho giảng viên.
13. **Lấy ra danh sách sinh viên trong file**: Hệ thống trích xuất danh sách sinh viên từ file điểm.
14. **Xác minh định dạng điểm**: Hệ thống kiểm tra xem các giá trị điểm trong file có đúng định dạng hay không.
15. **Nếu định dạng điểm sai, từ chối và gửi thông báo**: Nếu định dạng không đúng, hệ thống từ chối và gửi thông báo lỗi.
16. **Nếu tất cả hợp lệ, thêm bản ghi điểm vào cơ sở dữ liệu**: Hệ thống lưu bản ghi điểm vào cơ sở dữ liệu sau khi xác minh tất cả các điều kiện.

3.13. Thanh toán hóa đơn

Mô tả nghiệp vụ:

Hệ thống thanh toán hóa đơn cho phép người dùng chọn sản phẩm, thêm vào giỏ hàng, và tiến hành thanh toán trực tuyến hoặc COD (Cash on Delivery). Quy trình bao gồm việc xác thực thông tin đơn hàng, thông tin thanh toán, xử lý giao dịch, và cập nhật số lượng sản phẩm. Nếu thanh toán thành công, hệ thống sẽ lưu thông tin đơn hàng và gửi thông báo đến người dùng qua email.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống thanh toán hóa đơn dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ khi thêm sản phẩm vào giỏ hàng, xác thực thông tin, xử lý thanh toán, đến việc lưu thông tin đơn hàng và thông báo cho người dùng được thực hiện chính xác và bảo mật.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo quá trình mua hàng**: Người dùng bắt đầu quy trình mua sắm và thanh toán.
2. **Thêm sản phẩm vào giỏ hàng**: Người dùng chọn sản phẩm và thêm vào giỏ hàng.
3. **Xác thực việc thêm sản phẩm vào giỏ hàng**: Hệ thống xác thực sản phẩm đã được thêm thành công hay chưa.
4. **Chọn đặt hàng**: Người dùng chọn thực hiện đơn hàng.
5. **Nhập thông tin hóa đơn và chọn phương thức thanh toán**: Người dùng nhập thông tin hóa đơn và chọn phương thức thanh toán (mặc định là COD).
6. **Xác thực thông tin thẻ thanh toán**: Nếu người dùng chọn thanh toán bằng thẻ, hệ thống xác thực thông tin thẻ.
7. **Nếu thông tin thẻ không hợp lệ, yêu cầu kiểm tra lại**: Nếu thông tin thẻ không đúng, hệ thống yêu cầu người dùng kiểm tra và nhập lại.
8. **Thực hiện thanh toán**: Hệ thống tiến hành xử lý thanh toán dựa trên thông tin thẻ hoặc lựa chọn COD.
9. **Nếu thanh toán thất bại, yêu cầu kiểm tra lại**: Nếu thanh toán không thành công, hệ thống yêu cầu người dùng kiểm tra lại.
10. **Lưu thông tin đơn hàng**: Nếu thanh toán thành công, hệ thống lưu thông tin đơn hàng vào cơ sở dữ liệu.
11. **Cập nhật số lượng sản phẩm**: Hệ thống cập nhật lại số lượng sản phẩm trong kho sau khi đơn hàng được xác nhận.
12. **Thông báo thành công trên giao diện và gửi email xác nhận**: Hệ thống hiển thị thông báo đặt hàng thành công và gửi email xác nhận cho người dùng.

3.14. Tạo và giao việc

Mô tả nghiệp vụ:

Quy trình tạo và giao công việc mới cho phép quản trị viên trong một nhóm (team) tạo công việc, gán người thực hiện và lưu trữ thông tin công việc vào cơ sở dữ liệu. Người dùng phải có quyền quản trị viên trong team mới có thể thực hiện thao tác này. Sau khi công việc được tạo thành công, hệ thống sẽ gửi thông báo đến những người được giao việc.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống tạo và giao việc dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ xác thực quyền quản trị viên, nhập thông tin công việc, chọn người thực hiện, lưu công việc và gửi thông báo được thực hiện chính xác và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Vào Team muốn giao việc**: Người dùng truy cập vào team mà họ muốn tạo và giao công việc.
2. **Xác thực quyền quản trị viên**: Hệ thống kiểm tra xem người dùng có quyền quản trị viên trong team hay không.
3. **Nếu không phải quản trị viên, kết thúc quy trình**: Nếu người dùng không có quyền quản trị viên, hệ thống sẽ dừng quy trình và thông báo lỗi.
4. **Bắt đầu tạo công việc mới**: Người dùng chọn tùy chọn để bắt đầu tạo công việc mới trong team.
5. **Tạo công việc mới**: Người dùng nhập thông tin công việc mới, bao gồm tiêu đề, mô tả và thời hạn hoàn thành.
6. **Tạo danh sách người thực hiện công việc**: Người dùng bắt đầu tạo danh sách người sẽ thực hiện công việc.
7. **Tìm người để thêm vào danh sách**: Hệ thống hiển thị danh sách thành viên trong team để người dùng lựa chọn.
8. **Chọn người dùng vào danh sách người thực hiện**: Người dùng chọn các thành viên từ danh sách để giao việc.
9. **Nhấn tạo công việc mới**: Người dùng xác nhận việc tạo công việc mới sau khi nhập thông tin và chọn người thực hiện.
10. **Lưu công việc vào cơ sở dữ liệu**: Hệ thống lưu thông tin công việc và danh sách người thực hiện vào cơ sở dữ liệu.
11. **Thông báo cho từng người trong danh sách người thực hiện**: Hệ thống gửi thông báo đến những người được giao công việc qua email hoặc thông báo trong ứng dụng.

3.15. Tạo sự kiện

Mô tả nghiệp vụ:

Hệ thống tạo sự kiện cho phép người dùng upload thông tin sự kiện, lịch trình, và danh sách khách mời từ các file Excel. Hệ thống sẽ kiểm tra xem thời gian của sự kiện có trùng với bất kỳ sự kiện nào đã được tạo trước đó hay không. Nếu thời gian không trùng, hệ thống sẽ gửi thông báo mời tham gia sự kiện đến các khách mời và lưu trữ các thông tin sự kiện, lịch trình, và danh sách khách mời vào cơ sở dữ liệu.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống tạo sự kiện dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ khi upload thông tin sự kiện, kiểm tra trùng lặp thời gian, gửi thông báo đến khách mời, và lưu trữ thông tin vào cơ sở dữ liệu được thực hiện một cách chính xác và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo quy trình tổ chức sự kiện**: Người dùng bắt đầu quy trình tạo sự kiện mới.
2. **Tải file Excel chứa thông tin sự kiện**: Người dùng tải lên file Excel chứa các thông tin chi tiết về sự kiện.
3. **Kiểm tra thời gian sự kiện có bị trùng không**: Hệ thống kiểm tra xem thời gian sự kiện có trùng với bất kỳ sự kiện nào đã được tạo trước đó hay không.
4. **Nếu thời gian bị trùng, gửi thông báo lỗi**: Nếu thời gian sự kiện bị trùng, hệ thống gửi thông báo lỗi về việc trùng thời gian và kết thúc quy trình.
5. **Tải file Excel chứa danh sách khách mời**: Người dùng tải lên file Excel chứa danh sách khách mời tham gia sự kiện.
6. **Kiểm tra định dạng khách mời**: Hệ thống kiểm tra xem danh sách khách mời đã đầy đủ thông tin email hay chưa, nếu chưa thì hiển thị lỗi và kết thúc quy trình
7. **Ghi lại lịch trình vào cơ sở dữ liệu Schedule**: Hệ thống lưu thông tin lịch trình sự kiện vào cơ sở dữ liệu.
8. **Ghi lại sự kiện vào cơ sở dữ liệu**: Hệ thống lưu thông tin sự kiện vào cơ sở dữ liệu.
9. **Ghi lại danh sách khách mời**: Hệ thống lưu danh sách khách mời vào cơ sở dữ liệu.
10. **Gửi email cho khách mời**: Hệ thống gửi email thông báo mời tham gia sự kiện cho danh sách khách mời.
11. **Hiển thị thông báo tạo sự kiện thành công:** Hệ thống hiển thị thông báo tạo sự kiện thành công và kết thúc quy trình

3.16. Điểm danh tự động

Mô tả nghiệp vụ:

Hệ thống điểm danh tự động cho phép người dùng nhập mã phòng Zoom và yêu cầu điểm danh. Hệ thống sẽ lấy thông tin lịch học, kiểm tra sự tồn tại của mã phòng và so sánh thời gian thực tế của sinh viên trong phòng Zoom với thời gian học dự kiến. Nếu thời gian

tham dự đạt trên 80%, sinh viên sẽ được chấp nhận điểm danh, nếu không, sẽ được coi là vắng mặt. Kết quả điểm danh sẽ được lưu vào cơ sở dữ liệu và thông báo được gửi đến sinh viên.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống điểm danh tự động dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi nhận yêu cầu điểm danh, kiểm tra mã phòng, so sánh thời gian tham gia, và lưu trữ kết quả được thực hiện tự động và chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Người dùng nhập mã phòng Zoom yêu cầu**: Người dùng nhập mã phòng Zoom để bắt đầu quy trình điểm danh.
2. **Bắt đầu yêu cầu điểm danh**: Hệ thống nhận yêu cầu điểm danh từ người dùng.
3. **Nhận thông tin lịch học**: Hệ thống lấy thông tin lịch học tương ứng với mã phòng Zoom.
4. **Kiểm tra mã phòng có tồn tại trong lịch học hay không**: Hệ thống kiểm tra xem mã phòng Zoom có tồn tại trong lịch học không.
5. **Nếu không tồn tại, từ chối điểm danh**: Nếu mã phòng không tồn tại trong lịch học, hệ thống từ chối yêu cầu điểm danh.
6. **Gửi thông báo sai thông tin mã phòng**: Hệ thống gửi thông báo về lỗi mã phòng không hợp lệ cho người dùng.
7. **Lấy danh sách lớp**: Hệ thống lấy danh sách lớp học của mã phòng Zoom.
8. **Lấy thông tin các sinh viên của lớp trong lịch học**: Hệ thống truy xuất thông tin sinh viên trong lớp học từ dữ liệu lịch học.
9. **Lấy thông tin lịch sử ra vào thực tế của các sinh viên từ Zoom**: Hệ thống nhận dữ liệu về thời gian ra vào phòng Zoom của sinh viên.
10. **So sánh thời gian thực tế với thời gian trong lịch học**: Hệ thống so sánh thời gian thực tế của mỗi sinh viên trong phòng với thời gian học theo lịch.
11. **Nếu thời gian thực tế < 80% so với lịch học, xác định vắng mặt**: Nếu sinh viên tham gia dưới 80% thời gian học, sinh viên sẽ bị coi là vắng mặt.
12. **Nếu thời gian thực tế >= 80%, chấp nhận điểm danh**: Nếu sinh viên tham gia đủ 80% thời gian trở lên, điểm danh sẽ được chấp nhận.
13. **Lưu thông tin điểm danh vào server điểm danh**: Hệ thống lưu kết quả điểm danh của sinh viên vào cơ sở dữ liệu.
14. **Gửi thông báo điểm danh đến các sinh viên**: Hệ thống gửi thông báo về trạng thái điểm danh (đã điểm danh hoặc vắng mặt) cho từng sinh viên.
15. **Gửi thông báo điểm danh thành công kèm file danh sách**: Hệ thống gửi thông báo điểm danh thành công cho người yêu cầu kèm theo file danh sách kết quả điểm danh.
16. **Kết thúc quá trình**: Kết thúc quy trình điểm danh tự động.

.

3.17. Chấm điểm rèn luyện

Mô tả nghiệp vụ:

Hệ thống chấm điểm rèn luyện cho phép sinh viên nộp minh chứng về các hoạt động rèn luyện để đánh giá dựa trên các tiêu chí cụ thể. Quy trình bao gồm việc kiểm tra thời hạn đánh giá, nộp minh chứng, xác minh tính hợp lệ của bằng chứng theo tiêu chí đã chọn, và gửi thông báo chấp nhận hoặc từ chối kết quả. Hệ thống sẽ lưu thông tin về hoạt động rèn luyện của sinh viên sau khi quá trình đánh giá hoàn tất.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống chấm điểm rèn luyện dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ việc lấy thông tin tiêu chí, nộp minh chứng, xác minh và ghi nhận hoạt động rèn luyện của sinh viên được thực hiện chính xác và đúng quy trình.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo quy trình nộp minh chứng**: Sinh viên khởi tạo quy trình nộp bằng chứng rèn luyện để được đánh giá.
2. **Lấy thông tin thời hạn đánh giá**: Hệ thống lấy thông tin về thời hạn nộp minh chứng cho kỳ đánh giá hiện tại.
3. **Xác minh thời hạn đánh giá**: Hệ thống kiểm tra xem thời hạn nộp minh chứng còn hợp lệ hay đã hết hạn. Nếu hết hạn, kết thúc quy trình: Nếu thời hạn đã hết, hệ thống dừng quy trình và thông báo lỗi.
4. **Lấy thông tin tiêu chí đánh giá**: Hệ thống hiển thị các tiêu chí đánh giá mà sinh viên cần nộp minh chứng.
5. **Chọn tiêu chí đánh giá**: Sinh viên chọn tiêu chí phù hợp để nộp minh chứng.
6. **Gửi minh chứng đánh giá**: Sinh viên gửi minh chứng lên hệ thống để được đánh giá.
7. **Lấy thông tin minh chứng của sinh viên**: Hệ thống lấy thông tin minh chứng mà sinh viên đã nộp.
8. **Xác minh minh chứng dựa trên tiêu chí đánh giá thủ công**: Hệ thống hoặc giảng viên tiến hành xác minh bằng chứng theo tiêu chí đã chọn.
9. **Nếu minh chứng không hợp lệ, khởi tạo từ chối**: Nếu minh chứng không phù hợp, hệ thống sẽ bắt đầu quy trình từ chối đánh giá.
10. **Xác nhận từ chối minh chứng**: Quá trình từ chối được xác nhận thủ công bởi giảng viên hoặc người quản lý.
11. **Gửi thông báo từ chối**: Hệ thống gửi thông báo từ chối kết quả đánh giá cho sinh viên.

12. **Xác minh chấp nhận thủ công**: Nếu minh chứng hợp lệ, hệ thống tiến hành xác minh kết quả chấp nhận.
13. **Gửi thông báo chấp nhận**: Hệ thống gửi thông báo chấp nhận kết quả đánh giá cho sinh viên.
14. **Ghi nhận hoạt động của sinh viên**: Hệ thống lưu thông tin về hoạt động và điểm rèn luyện của sinh viên vào cơ sở dữ liệu.

3.18. So sánh giá sản phẩm

Mô tả nghiệp vụ:

Hệ thống so sánh giá sản phẩm cho phép người dùng tìm kiếm sản phẩm, lấy danh sách các sản phẩm phù hợp từ nhiều nhà cung cấp, và so sánh giá giữa các nhà cung cấp. Sau khi so sánh, người dùng có thể chọn nhà cung cấp đáp ứng nhu cầu của họ và điều hướng đến trang bán hàng của nhà cung cấp đó để hoàn tất giao dịch.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống so sánh giá sản phẩm dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ tìm kiếm sản phẩm, so sánh giá và điều hướng người dùng đến nơi bán sản phẩm được thực hiện một cách chính xác và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Bắt đầu so sánh giá sản phẩm**: Người dùng nhập từ khóa tìm kiếm để truy vấn sản phẩm mà họ quan tâm.
2. **Nếu không có sản phẩm phù hợp, gửi thông báo và kết thúc quy trình**: Nếu không có sản phẩm nào phù hợp với nội dung tìm kiếm, hệ thống gửi thông báo và dừng quy trình.
3. **Lấy danh sách các sản phẩm phù hợp**: Hệ thống tìm kiếm và trả về danh sách các sản phẩm phù hợp với từ khóa tìm kiếm của người dùng.
4. **Chọn sản phẩm được quan tâm**: Người dùng chọn một sản phẩm từ danh sách các sản phẩm phù hợp.
5. **Lấy thông tin chi tiết sản phẩm**: Hệ thống lấy thông tin chi tiết của sản phẩm mà người dùng đã chọn (bao gồm mô tả, thông số kỹ thuật, hình ảnh...).
6. **Lấy danh sách đề xuất**: Hệ thống gợi ý thêm các sản phẩm tương tự hoặc có liên quan cho người dùng.
7. **Lấy danh sách nhà cung cấp**: Hệ thống tìm kiếm và liệt kê các nhà cung cấp có bán sản phẩm mà người dùng quan tâm.
8. **So sánh giá giữa các nhà cung cấp**: Hệ thống so sánh giá của sản phẩm từ các nhà cung cấp khác nhau để đưa ra danh sách so sánh giá.
9. **Chọn nhà cung cấp đáp ứng nhu cầu người dùng**: Người dùng chọn nhà cung cấp phù hợp dựa trên giá cả, chất lượng dịch vụ hoặc các yếu tố khác.

10. **Điều hướng tới nơi bán mà người dùng mong muốn**: Hệ thống điều hướng người dùng đến trang bán hàng của nhà cung cấp mà họ đã chọn để hoàn tất giao dịch.

3.19. Ghi chỉ số điện/nước

Mô tả nghiệp vụ:

Hệ thống cập nhật chỉ số điện/nước cho phép người dùng tải ảnh công tơ nước, sau đó hệ thống sẽ kiểm tra thông tin từ mã QR và sử dụng xử lý hình ảnh để lấy chỉ số nước từ ảnh. Sau khi chỉ số nước được xác minh, nếu chưa được cập nhật, hệ thống sẽ tạo hóa đơn mới và tiến hành thanh toán. Nếu thanh toán thành công, hệ thống sẽ cập nhật chỉ số mới vào cơ sở dữ liệu và kết thúc quy trình.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống cập nhật chỉ số điện/nước dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ khi tải ảnh công tơ, xác minh thông tin QR, kiểm tra và cập nhật chỉ số, đến việc tạo hóa đơn và thanh toán được thực hiện tự động và chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo quy trình**: Người dùng bắt đầu quy trình cập nhật chỉ số điện/nước.
2. **Chụp và gửi thủ công ảnh công tơ nước**: Người dùng tải ảnh công tơ nước lên hệ thống.
3. **Trích rút thông tin từ ảnh công tơ nước**: Hệ thống xử lý để đọc thông tin mã QR và phân tích ảnh công tơ để lấy chỉ số nước.
4. **Nếu QR không hợp lệ, kết thúc quy trình**: Nếu mã QR không hợp lệ, hệ thống dừng quy trình và thông báo lỗi.
5. **Xác minh từ chối theo cách thủ công**: Nếu cần thiết, quy trình từ chối sẽ được xác minh thủ công bởi nhân viên quản lý.
6. **Nhận thông tin người dùng từ QR và lấy số nước từ ảnh công tơ nước**: Hệ thống lấy thông tin người dùng từ mã QR và số nước từ ảnh công tơ.
7. **Kiểm tra thủ công số nước cập nhật và số nước trên công tơ**: Người dùng hoặc hệ thống xác minh thủ công chỉ số nước trên ảnh và chỉ số được cập nhật.
8. **Nếu số nước đã được cập nhật trong tháng, kết thúc quy trình**: Nếu chỉ số nước đã được cập nhật trước đó trong tháng, quy trình sẽ kết thúc.
9. **Tự động tạo hóa đơn mới cho công tơ vừa cập nhật (ở trạng thái chưa thanh toán)**: Hệ thống tạo hóa đơn mới cho công tơ với trạng thái chưa thanh toán.
10. **Tiến hành thanh toán**: Hệ thống gọi cổng thanh toán ngân hàng tích hợp để tiến hành thanh toán hóa đơn.
11. **Nếu tài khoản không đủ tiền, kết thúc quy trình**: Nếu tài khoản người dùng không đủ tiền để thanh toán hóa đơn, hệ thống sẽ dừng quy trình và thông báo lỗi.

12. **Nếu thanh toán thành công**: Cập nhật thông tin vào cơ sở dữ liệu và kết thúc quy trình.

3.20. Xuất dữ liệu điểm danh

Mô tả nghiệp vụ:

Hệ thống điểm danh cho phép giảng viên nhập thông tin của sinh viên tham gia buổi học (bao gồm ID Meet/Zoom và danh sách sinh viên), sau đó so sánh với dữ liệu thực tế lấy từ Zoom/Meet để xác định sinh viên có tham gia hay không. Hệ thống sẽ xuất ra một danh sách đã điểm danh, cho phép giảng viên lưu thông tin, xuất file CSV, và nộp bản cứng.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống điểm danh từ dữ liệu Zoom hoặc Meet (lựa chọn một trong 02 trường hợp) dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ nhập thông tin điểm danh, so sánh với dữ liệu thực tế, đến việc xuất file CSV và lưu trữ được thực hiện chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Giảng viên nhập thông tin điểm danh của sinh viên (bao gồm ID Meet/Zoom và danh sách sinh viên)**: Giảng viên nhập thông tin ID của buổi học và danh sách sinh viên tham gia buổi học.
2. **Nhận thông tin chi tiết về sinh viên (mã sinh viên, tên sinh viên)**: Hệ thống nhận thông tin chi tiết của từng sinh viên (mã sinh viên và tên sinh viên) từ danh sách mà giảng viên đã nhập.
3. **So sánh với dữ liệu lấy được từ Meet/Zoom**: Hệ thống lấy dữ liệu tham gia buổi học từ Zoom/Meet (bao gồm thời gian vào, thời gian ra) và so sánh với danh sách sinh viên.
4. **Xuất ra file danh sách đã điểm danh**: Hệ thống xuất ra danh sách sinh viên đã được điểm danh, bao gồm các thông tin như mã sinh viên, tên sinh viên, trạng thái điểm danh (có tham gia hoặc vắng mặt), thời gian vào, và thời gian ra.
5. **Giáo viên bấm Kết thúc buổi học**: Giảng viên kết thúc buổi học, hệ thống sẽ hoàn thành quá trình điểm danh.
6. **Lưu thông tin vào hệ thống**: Hệ thống lưu thông tin điểm danh của sinh viên vào cơ sở dữ liệu để quản lý và theo dõi.
7. **Giáo viên bấm Export file CSV**: Giảng viên xuất file CSV chứa danh sách sinh viên đã điểm danh.
8. **Nộp bản cứng thông tin điểm danh**: Giảng viên in và nộp bản cứng thông tin điểm danh cho phòng quản lý hoặc bộ phận có liên quan.

3.21. Làm bài thi

Mô tả nghiệp vụ:

Hệ thống làm bài thi cho phép sinh viên nhập mã sinh viên và mã đề thi để bắt đầu quá trình thi. Hệ thống sẽ xác minh thông tin sinh viên, kiểm tra thông tin chi tiết của bài thi và chỉ hiển thị câu hỏi khi tất cả thông tin hợp lệ. Nếu sinh viên không có trong danh sách hoặc bài thi chưa bắt đầu, hệ thống sẽ từ chối truy cập.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống làm bài thi dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo các bước từ khi sinh viên nhập thông tin, xác minh danh tính, kiểm tra bài thi và hiển thị câu hỏi được thực hiện chính xác và bảo mật.

Mô tả chi tiết các bước nghiệp vụ:

1. **Sinh viên nhập mã sinh viên và mã đề thi**: Sinh viên nhập thông tin cá nhân (mã sinh viên) và mã đề thi vào hệ thống.
2. **Sinh viên bấm nút "Bắt đầu làm bài"**: Sinh viên xác nhận bắt đầu quá trình làm bài thi.
3. **Nhận thông tin về sinh viên và bài thi**: Hệ thống nhận thông tin mã sinh viên từ đầu vào của sinh viên.
4. **Xác minh sinh viên và bài thi**: Hệ thống kiểm tra xem sinh viên có hợp lệ, có trong danh sách tham gia làm bài thi không.
5. **Nếu sinh viên không có trong danh sách lớp, kết thúc quá trình**: Nếu sinh viên không hợp lệ hoặc không thuộc danh sách lớp, hệ thống sẽ dừng quy trình và thông báo lỗi.
6. **Nếu thông tin sinh viên hợp lệ, truy xuất thông tin chi tiết sinh viên**: Hệ thống truy xuất và xác minh thông tin chi tiết của sinh viên.
7. **Nhận thông tin chi tiết về sinh viên**: Hệ thống nhận và lưu trữ thông tin chi tiết về sinh viên, bao gồm tên, lớp, và các thông tin liên quan.
8. **Nhận thông tin chi tiết về bài thi**: Hệ thống lấy thông tin chi tiết về bài thi từ cơ sở dữ liệu, bao gồm danh sách câu hỏi, thời gian thi, và trạng thái bài thi.
9. **Xác minh các thông tin chi tiết**: Hệ thống kiểm tra lại các thông tin chi tiết về sinh viên và bài thi.
10. **Nếu thông tin không hợp lệ, kết thúc quá trình**: Nếu thông tin không hợp lệ, ví dụ như ngoài thời gian làm bài, đã làm bài … quy trình sẽ kết thúc.
11. **Nếu thông tin hợp lệ, lấy thông tin câu hỏi và hiển thị**: Hệ thống hiển thị danh sách câu hỏi cho sinh viên bắt đầu làm bài thi.

3.22. Phê duyệt yêu cầu nghỉ phép

Mô tả nghiệp vụ:

Quy trình phê duyệt yêu cầu nghỉ phép cho phép nhân viên nộp đơn xin nghỉ phép qua hệ thống. Quản lý sau đó sẽ xem xét yêu cầu và phê duyệt hoặc từ chối dựa trên tình trạng làm việc của nhân viên và các quy định nghỉ phép của công ty. Nếu đơn xin nghỉ được phê duyệt, hệ thống sẽ gửi thông báo cho nhân viên và cập nhật tình trạng nghỉ phép của họ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống phê duyệt yêu cầu nghỉ phép dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi nhân viên nộp đơn, xác minh thông tin, đến phê duyệt hoặc từ chối, và thông báo kết quả được thực hiện tự động và chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Nhân viên nộp yêu cầu nghỉ phép**: Nhân viên nhập thông tin về ngày nghỉ dự kiến, loại nghỉ phép (ví dụ: nghỉ ốm, nghỉ phép năm), và lý do nộp đơn xin nghỉ.
2. **Nhận thông tin chi tiết về nhân viên**: Hệ thống nhận thông tin về nhân viên, bao gồm mã nhân viên, tên, và phòng ban.
3. **Kiểm tra lịch sử nghỉ phép**: Hệ thống truy xuất lịch sử nghỉ phép của nhân viên để kiểm tra số ngày nghỉ phép đã sử dụng trong năm.
4. **Kiểm tra số ngày nghỉ còn lại**: Hệ thống xác minh xem nhân viên còn bao nhiêu ngày nghỉ phép và liệu yêu cầu có vượt quá số ngày nghỉ hiện có hay không.
5. **Gửi yêu cầu đến quản lý**: Nếu thông tin hợp lệ, hệ thống gửi yêu cầu nghỉ phép đến quản lý trực tiếp của nhân viên để phê duyệt.
6. **Quản lý nhận thông báo yêu cầu phê duyệt**: Quản lý nhận được thông báo về yêu cầu nghỉ phép của nhân viên và truy cập hệ thống để xem chi tiết.
7. **Xem xét yêu cầu**: Quản lý kiểm tra thông tin yêu cầu, bao gồm ngày nghỉ, lý do, và tình trạng công việc hiện tại.
8. **Phê duyệt hoặc từ chối yêu cầu**: Quản lý quyết định phê duyệt hoặc từ chối yêu cầu dựa trên thông tin đã xem xét.
9. **Nếu phê duyệt, hệ thống gửi thông báo chấp nhận**: Nếu quản lý phê duyệt yêu cầu, hệ thống gửi thông báo chấp nhận nghỉ phép cho nhân viên và cập nhật lịch nghỉ.
10. **Nếu từ chối, hệ thống gửi thông báo từ chối**: Nếu quản lý từ chối yêu cầu, hệ thống gửi thông báo từ chối cho nhân viên kèm lý do.
11. **Cập nhật trạng thái nghỉ phép của nhân viên**: Nếu yêu cầu được phê duyệt, hệ thống cập nhật trạng thái nghỉ phép của nhân viên trong hệ thống quản lý nhân sự.
12. **Kết thúc quy trình**: Quy trình kết thúc sau khi hệ thống gửi thông báo và cập nhật thông tin.

3.23. Phê duyệt yêu cầu thanh toán công tác phí

Mô tả nghiệp vụ:

Hệ thống phê duyệt yêu cầu thanh toán công tác phí cho phép nhân viên nộp đơn yêu cầu thanh toán sau khi hoàn thành công tác. Đơn yêu cầu bao gồm thông tin chi tiết về chi phí phát sinh trong chuyến đi công tác (như tiền di chuyển, lưu trú, ăn uống, v.v.). Sau khi nộp đơn, quản lý sẽ xem xét và quyết định phê duyệt hoặc từ chối. Nếu đơn yêu cầu được phê duyệt, hệ thống sẽ chuyển thông tin đến bộ phận tài chính để thanh toán.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống phê duyệt yêu cầu thanh toán công tác phí dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi nhân viên nộp đơn, xác minh thông tin chi phí, đến việc phê duyệt và chuyển thông tin thanh toán cho bộ phận tài chính được thực hiện chính xác và tự động.

Mô tả chi tiết các bước nghiệp vụ:

1. **Nhân viên nộp yêu cầu thanh toán công tác phí**: Nhân viên nhập thông tin chi phí phát sinh trong chuyến công tác, bao gồm tiền vé máy bay, khách sạn, ăn uống, và các chi phí khác, kèm theo hóa đơn hoặc biên lai.
2. **Nhận thông tin chi tiết về chi phí công tác**: Hệ thống nhận thông tin chi tiết về các khoản chi phí được nộp và lưu trữ trong cơ sở dữ liệu.
3. **Kiểm tra tính hợp lệ của hóa đơn/biên lai**: Hệ thống kiểm tra xem các hóa đơn, biên lai đính kèm có hợp lệ và có đáp ứng yêu cầu của công ty hay không.
4. **Gửi yêu cầu đến quản lý để phê duyệt**: Nếu thông tin hợp lệ, hệ thống gửi yêu cầu đến quản lý trực tiếp của nhân viên để phê duyệt thanh toán.
5. **Quản lý nhận thông báo phê duyệt yêu cầu thanh toán công tác phí**: Quản lý nhận được thông báo về yêu cầu thanh toán công tác phí và truy cập hệ thống để xem chi tiết.
6. **Xem xét yêu cầu thanh toán**: Quản lý kiểm tra thông tin chi phí, các hóa đơn, và tình trạng công việc của nhân viên trước khi đưa ra quyết định.
7. **Phê duyệt hoặc từ chối yêu cầu**: Quản lý quyết định phê duyệt hoặc từ chối yêu cầu dựa trên thông tin được cung cấp.
8. **Nếu phê duyệt, hệ thống gửi thông báo chấp nhận**: Nếu quản lý phê duyệt yêu cầu, hệ thống sẽ gửi thông báo chấp nhận thanh toán công tác phí đến nhân viên.
9. **Nếu từ chối, hệ thống gửi thông báo từ chối**: Nếu quản lý từ chối yêu cầu, hệ thống gửi thông báo từ chối đến nhân viên kèm lý do.
10. **Chuyển thông tin đến bộ phận tài chính**: Sau khi phê duyệt, hệ thống chuyển thông tin chi tiết về yêu cầu thanh toán cho bộ phận tài chính để tiến hành thanh toán.

11. **Bộ phận tài chính xử lý thanh toán**: Bộ phận tài chính xử lý thanh toán công tác phí cho nhân viên thông qua tài khoản ngân hàng.
12. **Gửi thông báo hoàn tất thanh toán**: Sau khi thanh toán hoàn tất, hệ thống gửi thông báo đến nhân viên về việc thanh toán đã được thực hiện.
13. **Cập nhật trạng thái thanh toán**: Hệ thống cập nhật trạng thái thanh toán cho yêu cầu trong cơ sở dữ liệu để hoàn tất quy trình.

3.24. Xử lý đơn hàng trực tuyến

Mô tả nghiệp vụ:

Hệ thống xử lý đơn hàng trực tuyến cho phép khách hàng đặt mua sản phẩm từ cửa hàng trực tuyến. Sau khi khách hàng đặt hàng, hệ thống sẽ xử lý đơn hàng bằng cách xác minh tình trạng hàng trong kho, xác nhận đơn hàng, xử lý thanh toán và lên lịch giao hàng. Nếu đơn hàng được thanh toán thành công, sản phẩm sẽ được gửi đến khách hàng.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống xử lý đơn hàng trực tuyến dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi khách hàng đặt hàng, xác nhận tình trạng hàng tồn kho, xử lý thanh toán và giao hàng được thực hiện tự động và hiệu quả.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khách hàng đặt hàng**: Khách hàng duyệt qua các sản phẩm trên trang web, chọn sản phẩm cần mua và thêm vào giỏ hàng.
2. **Xác nhận đơn hàng**: Khách hàng xác nhận đơn hàng và nhập thông tin giao hàng, bao gồm địa chỉ giao hàng và phương thức thanh toán.
3. **Kiểm tra tình trạng hàng tồn kho**: Hệ thống kiểm tra tình trạng hàng tồn kho để đảm bảo sản phẩm còn hàng và có thể giao.
4. **Nếu hàng không có sẵn, thông báo khách hàng**: Nếu sản phẩm đã hết hàng, hệ thống thông báo cho khách hàng và đề xuất các sản phẩm tương tự hoặc hủy đơn hàng.
5. **Nếu hàng có sẵn, tiếp tục xử lý đơn hàng**: Nếu sản phẩm còn hàng, hệ thống tiếp tục xử lý đơn hàng.
6. **Xử lý thanh toán**: Hệ thống chuyển yêu cầu thanh toán đến nhà cung cấp dịch vụ thanh toán, như thẻ tín dụng hoặc ví điện tử.
7. **Nếu thanh toán thất bại, thông báo lỗi**: Nếu quá trình thanh toán gặp lỗi hoặc không thành công, hệ thống sẽ thông báo lỗi cho khách hàng và yêu cầu họ nhập lại thông tin thanh toán hoặc chọn phương thức thanh toán khác.
8. **Nếu thanh toán thành công, xác nhận đơn hàng**: Nếu thanh toán thành công, hệ thống xác nhận đơn hàng và gửi thông báo cho khách hàng.

9. **Lên lịch giao hàng**: Hệ thống lên lịch giao hàng dựa trên địa chỉ giao hàng và thông tin từ đơn vị vận chuyển.
10. **Gửi thông tin đơn hàng đến đơn vị vận chuyển**: Hệ thống gửi thông tin về đơn hàng và địa chỉ giao hàng đến đơn vị vận chuyển.
11. **Theo dõi trạng thái giao hàng**: Hệ thống theo dõi trạng thái giao hàng và cập nhật cho khách hàng qua email hoặc thông báo trên ứng dụng.
12. **Xác nhận đơn hàng đã giao**: Sau khi đơn hàng được giao thành công, hệ thống sẽ cập nhật trạng thái đơn hàng thành "Đã giao".
13. **Gửi thông báo giao hàng thành công**: Hệ thống gửi thông báo đến khách hàng rằng đơn hàng đã được giao thành công và yêu cầu khách hàng đánh giá sản phẩm.
14. **Lưu thông tin đơn hàng vào hệ thống quản lý**: Hệ thống lưu trữ thông tin đơn hàng vào cơ sở dữ liệu để theo dõi và quản lý các đơn hàng đã hoàn thành.

3.25. Xử lý khiếu nại khách hàng

Mô tả nghiệp vụ:

Hệ thống xử lý khiếu nại của khách hàng cho phép khách hàng gửi khiếu nại về sản phẩm hoặc dịch vụ mà họ đã sử dụng. Sau khi khiếu nại được gửi, hệ thống sẽ phân loại khiếu nại, chỉ định cho nhân viên phụ trách và theo dõi quá trình giải quyết. Khi khiếu nại được xử lý, hệ thống sẽ gửi thông báo đến khách hàng về kết quả giải quyết.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống xử lý khiếu nại của khách hàng dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi tiếp nhận khiếu nại, phân công xử lý, theo dõi tiến độ và thông báo kết quả cho khách hàng được thực hiện tự động và minh bạch.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khách hàng gửi khiếu nại**: Khách hàng nhập thông tin khiếu nại về sản phẩm hoặc dịch vụ (bao gồm mô tả vấn đề và bằng chứng, nếu có) qua ứng dụng hoặc website.
2. **Tiếp nhận khiếu nại**: Hệ thống tiếp nhận thông tin khiếu nại từ khách hàng và tạo một mã khiếu nại duy nhất.
3. **Phân loại khiếu nại**: Hệ thống tự động phân loại khiếu nại theo loại (ví dụ: lỗi sản phẩm, dịch vụ không đạt yêu cầu, vấn đề thanh toán).
4. **Chỉ định nhân viên xử lý**: Hệ thống chỉ định một nhân viên phụ trách dựa trên loại khiếu nại và kỹ năng xử lý.
5. **Gửi thông báo tiếp nhận cho khách hàng**: Hệ thống gửi thông báo cho khách hàng rằng khiếu nại đã được tiếp nhận và cung cấp mã khiếu nại để theo dõi.
6. **Xử lý khiếu nại**: Nhân viên phụ trách kiểm tra thông tin khiếu nại, liên hệ với các bộ phận liên quan (nếu cần) để giải quyết vấn đề.

7. **Gửi yêu cầu bổ sung thông tin**: Nếu cần thêm thông tin từ khách hàng, hệ thống gửi yêu cầu bổ sung thông tin qua email hoặc ứng dụng.
8. **Khách hàng cung cấp thông tin bổ sung (nếu cần)**: Khách hàng phản hồi bằng cách gửi thêm thông tin hoặc bằng chứng theo yêu cầu.
9. **Theo dõi tiến độ giải quyết**: Hệ thống theo dõi tiến độ xử lý khiếu nại, đảm bảo việc xử lý được thực hiện trong thời gian quy định.
10. **Hoàn tất xử lý khiếu nại**: Sau khi khiếu nại được giải quyết, nhân viên phụ trách cập nhật kết quả vào hệ thống (ví dụ: hoàn tiền, thay thế sản phẩm, xin lỗi).
11. **Gửi thông báo kết quả cho khách hàng**: Hệ thống gửi thông báo cho khách hàng về kết quả xử lý khiếu nại và bất kỳ hành động nào sẽ được thực hiện (hoàn tiền, thay thế sản phẩm, v.v.).
12. **Lưu thông tin khiếu nại vào hệ thống quản lý**: Hệ thống lưu trữ thông tin khiếu nại và kết quả xử lý để phục vụ cho việc quản lý và cải tiến dịch vụ trong tương lai.
13. **Khảo sát mức độ hài lòng**: Hệ thống gửi khảo sát mức độ hài lòng để khách hàng đánh giá quy trình xử lý khiếu nại.

3.26. Đăng ký thuê bao

Mô tả nghiệp vụ:

Hệ thống đăng ký dịch vụ thuê bao cho phép khách hàng lựa chọn và đăng ký các gói dịch vụ như di động, internet hoặc truyền hình. Sau khi đăng ký, hệ thống sẽ xác minh thông tin khách hàng, kiểm tra khả năng cung cấp dịch vụ tại khu vực và kích hoạt dịch vụ. Nếu dịch vụ được kích hoạt thành công, khách hàng sẽ nhận được thông báo và bắt đầu sử dụng dịch vụ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đăng ký dịch vụ thuê bao dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ lúc khách hàng đăng ký dịch vụ, xác minh thông tin, kiểm tra tính khả dụng và kích hoạt dịch vụ diễn ra một cách tự động và chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khách hàng chọn gói dịch vụ thuê bao**: Khách hàng lựa chọn gói dịch vụ muốn đăng ký (ví dụ: gói cước di động, internet hoặc truyền hình).
2. **Khách hàng nhập thông tin cá nhân**: Khách hàng cung cấp các thông tin cá nhân cần thiết như tên, địa chỉ, số điện thoại và thông tin thanh toán.
3. **Xác minh thông tin khách hàng**: Hệ thống xác minh thông tin cá nhân của khách hàng, bao gồm kiểm tra số CMND/CCCD, địa chỉ, và thông tin thanh toán.
4. **Kiểm tra tính khả dụng của dịch vụ tại khu vực**: Hệ thống kiểm tra xem dịch vụ có thể cung cấp được tại khu vực của khách hàng hay không (ví dụ: kiểm tra tín hiệu di động, hạ tầng internet, hoặc truyền hình cáp).

5. **Nếu dịch vụ không khả dụng, thông báo cho khách hàng**: Nếu dịch vụ không khả dụng tại khu vực của khách hàng, hệ thống gửi thông báo từ chối và đề xuất các lựa chọn khác (nếu có).
6. **Nếu dịch vụ khả dụng, tiếp tục quy trình đăng ký**: Nếu dịch vụ khả dụng tại khu vực, hệ thống tiếp tục quy trình đăng ký.
7. **Xác nhận đơn đăng ký**: Hệ thống gửi thông báo xác nhận đơn đăng ký cho khách hàng và tiến hành kích hoạt dịch vụ.
8. **Kích hoạt dịch vụ**: Hệ thống gửi yêu cầu kích hoạt dịch vụ đến các hệ thống liên quan (ví dụ: hệ thống mạng di động, hệ thống ISP).
9. **Theo dõi quá trình kích hoạt**: Hệ thống theo dõi quá trình kích hoạt dịch vụ để đảm bảo mọi thứ diễn ra suôn sẻ.
10. **Thông báo kết quả kích hoạt**: Sau khi dịch vụ được kích hoạt thành công, hệ thống gửi thông báo cho khách hàng rằng dịch vụ đã sẵn sàng sử dụng.
11. **Cung cấp thông tin sử dụng dịch vụ**: Hệ thống cung cấp cho khách hàng các thông tin cần thiết để sử dụng dịch vụ, bao gồm tài khoản, mật khẩu, và hướng dẫn sử dụng.
12. **Tạo hợp đồng thuê bao**: Hệ thống tạo hợp đồng thuê bao dựa trên gói dịch vụ đã chọn và thông tin cá nhân của khách hàng.
13. **Cập nhật cơ sở dữ liệu khách hàng**: Hệ thống cập nhật thông tin khách hàng và tình trạng đăng ký dịch vụ vào cơ sở dữ liệu để quản lý và theo dõi.
14. **Gửi hóa đơn thanh toán**: Hệ thống gửi hóa đơn thanh toán cho khách hàng qua email hoặc ứng dụng.

3.27. Đăng ký số điện thoại sử dụng dịch vụ

Mô tả nghiệp vụ:

Hệ thống đăng ký số điện thoại cá nhân và xác minh danh tính cho phép người dùng đăng ký số điện thoại để liên kết với tài khoản và sử dụng dịch vụ. Sau khi đăng ký, hệ thống sẽ yêu cầu người dùng xác minh danh tính bằng cách nhập mã OTP (mã xác thực một lần) và tải lên các tài liệu chứng minh danh tính như CMND/CCCD. Quá trình xác minh sẽ đảm bảo rằng số điện thoại được liên kết với một danh tính hợp lệ trước khi kích hoạt dịch vụ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống đăng ký số điện thoại và xác minh danh tính dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo rằng quy trình từ việc đăng ký số điện thoại, nhận mã OTP, xác minh tài liệu danh tính đến việc kích hoạt tài khoản được thực hiện chính xác và bảo mật.

Mô tả chi tiết các bước nghiệp vụ:

1. **Người dùng nhập số điện thoại**: Người dùng nhập số điện thoại cá nhân để bắt đầu quy trình đăng ký.
2. **Gửi mã OTP qua SMS**: Hệ thống gửi mã OTP (One-Time Password) đến số điện thoại mà người dùng đã nhập để xác thực số điện thoại đó.
3. **Người dùng nhập mã OTP**: Người dùng nhập mã OTP nhận được để xác thực số điện thoại.
4. **Xác minh mã OTP**: Hệ thống kiểm tra mã OTP. Nếu mã OTP hợp lệ, quy trình tiếp tục. Nếu không hợp lệ, yêu cầu người dùng nhập lại mã.
5. **Yêu cầu tải lên tài liệu xác minh danh tính**: Sau khi xác minh số điện thoại, hệ thống yêu cầu người dùng tải lên tài liệu chứng minh danh tính như CMND, CCCD, hoặc hộ chiếu.
6. **Xử lý tài liệu danh tính**: Hệ thống phân tích tài liệu chứng minh danh tính sử dụng OCR (nhận dạng ký tự quang học) hoặc các công nghệ tương tự để đọc thông tin từ tài liệu.
7. **Xác minh tính hợp lệ của tài liệu**: Hệ thống kiểm tra tính hợp lệ của tài liệu như ngày hết hạn, tính xác thực và kiểm tra xem thông tin có khớp với thông tin cá nhân người dùng đã cung cấp hay không.
8. **Nếu tài liệu không hợp lệ, thông báo và yêu cầu tải lại**: Nếu tài liệu không hợp lệ hoặc không đúng định dạng, hệ thống sẽ yêu cầu người dùng tải lại tài liệu.
9. **Nếu tài liệu hợp lệ, tiếp tục quy trình xác minh**: Nếu tài liệu hợp lệ, hệ thống tiếp tục quá trình xác minh danh tính.
10. **Kiểm tra chống gian lận (Fraud Detection)**: Hệ thống kiểm tra các yếu tố bảo mật và chống gian lận để đảm bảo rằng số điện thoại và danh tính không bị lạm dụng.
11. **Thông báo kết quả xác minh danh tính**: Sau khi hoàn tất quy trình xác minh, hệ thống gửi thông báo về trạng thái xác minh danh tính của người dùng (đã thành công hoặc bị từ chối).
12. **Kích hoạt tài khoản hoặc dịch vụ**: Nếu xác minh thành công, hệ thống kích hoạt tài khoản hoặc dịch vụ liên quan cho số điện thoại của người dùng.
13. **Gửi thông báo kích hoạt thành công**: Hệ thống gửi thông báo đến người dùng qua SMS hoặc email về việc tài khoản đã được kích hoạt thành công và dịch vụ đã sẵn sàng sử dụng.
14. **Cập nhật cơ sở dữ liệu khách hàng**: Hệ thống lưu trữ thông tin về số điện thoại và danh tính đã được xác minh trong cơ sở dữ liệu khách hàng.
15. **Người dùng bắt đầu sử dụng dịch vụ**: Sau khi số điện thoại và danh tính được xác minh, người dùng có thể bắt đầu sử dụng các dịch vụ liên quan đến số điện thoại đã đăng ký.

3.28. Đặt lịch hẹn trực tuyến

Mô tả nghiệp vụ:

Hệ thống quản lý đặt lịch hẹn trực tuyến cho phép khách hàng đặt lịch với các dịch vụ (ví dụ: tư vấn, khám bệnh, chăm sóc tóc) thông qua ứng dụng hoặc website. Khách hàng có thể chọn thời gian, dịch vụ mong muốn và nhân viên cung cấp dịch vụ. Sau khi đặt lịch hẹn, hệ thống sẽ xác nhận, thông báo và gửi lời nhắc cho khách hàng. Nếu cần, khách hàng cũng có thể thay đổi hoặc hủy lịch hẹn.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống quản lý đặt lịch hẹn trực tuyến dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi khách hàng đặt lịch, xác nhận dịch vụ và thời gian, đến khi gửi lời nhắc và quản lý thay đổi hoặc hủy lịch hẹn được thực hiện tự động và tiện lợi.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khách hàng chọn dịch vụ và thời gian**: Khách hàng truy cập vào hệ thống, chọn loại dịch vụ mà họ muốn sử dụng (ví dụ: khám sức khỏe, tư vấn, chăm sóc sắc đẹp) và thời gian phù hợp.
2. **Chọn nhân viên cung cấp dịch vụ (nếu cần)**: Hệ thống hiển thị danh sách nhân viên hoặc chuyên viên có sẵn để cung cấp dịch vụ. Khách hàng có thể chọn nhân viên mà họ muốn gặp.
3. **Kiểm tra tình trạng lịch hẹn**: Hệ thống kiểm tra tình trạng trống của lịch hẹn dựa trên thời gian và dịch vụ đã chọn. Nếu thời gian hoặc nhân viên không khả dụng, hệ thống đề xuất các thời gian thay thế.
4. **Xác nhận lịch hẹn**: Sau khi chọn thời gian và nhân viên (nếu có), hệ thống xác nhận thông tin lịch hẹn với khách hàng.
5. **Gửi thông báo xác nhận qua email hoặc SMS**: Hệ thống gửi thông báo cho khách hàng qua email hoặc SMS về lịch hẹn đã được xác nhận, kèm theo chi tiết về dịch vụ, thời gian, địa điểm và nhân viên (nếu có).
6. **Cập nhật lịch hẹn vào hệ thống**: Lịch hẹn được lưu trữ trong hệ thống để theo dõi và quản lý.
7. **Gửi lời nhắc trước lịch hẹn**: Hệ thống tự động gửi lời nhắc qua email hoặc SMS cho khách hàng trước lịch hẹn (ví dụ: trước 1 ngày hoặc vài giờ).
8. **Khách hàng có thể thay đổi lịch hẹn (nếu cần)**: Khách hàng có thể truy cập vào hệ thống để thay đổi thời gian hoặc dịch vụ của lịch hẹn nếu có nhu cầu.
9. **Hệ thống kiểm tra và xác nhận thay đổi**: Nếu khách hàng thay đổi lịch hẹn, hệ thống sẽ kiểm tra tình trạng trống và xác nhận thay đổi với khách hàng.
10. **Khách hàng có thể hủy lịch hẹn (nếu cần)**: Nếu khách hàng không thể tham dự, họ có thể hủy lịch hẹn trực tuyến thông qua hệ thống.

11. **Gửi thông báo hủy lịch hẹn cho nhân viên**: Khi lịch hẹn bị hủy, hệ thống gửi thông báo cho nhân viên liên quan về việc hủy hẹn.
12. **Quản lý theo dõi lịch hẹn**: Hệ thống cho phép nhân viên hoặc quản lý theo dõi lịch hẹn của từng ngày, tuần hoặc tháng để tối ưu hóa thời gian làm việc.
13. **Hoàn thành dịch vụ và phản hồi**: Sau khi dịch vụ được hoàn thành, khách hàng có thể được yêu cầu cung cấp phản hồi về dịch vụ thông qua hệ thống.
14. **Gửi hóa đơn hoặc thông tin thanh toán**: Nếu có phí dịch vụ, hệ thống có thể gửi hóa đơn hoặc thông tin thanh toán cho khách hàng qua email hoặc SMS sau khi hoàn thành lịch hẹn.

3.29. Yêu cầu bảo hành sản phẩm

Mô tả nghiệp vụ:

Hệ thống quản lý yêu cầu bảo hành sản phẩm cho phép khách hàng gửi yêu cầu bảo hành trực tuyến khi sản phẩm gặp sự cố hoặc hỏng hóc trong thời gian còn bảo hành. Sau khi tiếp nhận, hệ thống sẽ kiểm tra điều kiện bảo hành của sản phẩm, xác nhận với khách hàng và tiến hành bảo hành hoặc thay thế sản phẩm nếu đủ điều kiện. Quá trình bảo hành được theo dõi chặt chẽ, và khách hàng sẽ được thông báo khi sản phẩm đã được sửa chữa hoặc thay thế thành công.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống quản lý yêu cầu bảo hành sản phẩm dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi khách hàng gửi yêu cầu, kiểm tra điều kiện bảo hành, đến khi hoàn tất bảo hành hoặc thay thế sản phẩm được thực hiện hiệu quả và tự động.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khách hàng gửi yêu cầu bảo hành**: Khách hàng truy cập vào hệ thống, chọn sản phẩm cần bảo hành và nhập thông tin chi tiết về sự cố gặp phải (kèm theo hình ảnh hoặc video nếu có).
2. **Kiểm tra thời hạn bảo hành của sản phẩm**: Hệ thống kiểm tra thời hạn bảo hành của sản phẩm dựa trên số serial hoặc thông tin mua hàng. Nếu sản phẩm vẫn còn trong thời hạn bảo hành, tiếp tục quy trình; nếu không, thông báo cho khách hàng rằng sản phẩm không đủ điều kiện bảo hành.
3. **Kiểm tra điều kiện bảo hành**: Hệ thống kiểm tra xem sản phẩm có đáp ứng các điều kiện bảo hành (ví dụ: không bị hư hỏng do tác động ngoại lực, không vi phạm điều kiện sử dụng).
4. **Nếu sản phẩm không đủ điều kiện bảo hành, từ chối yêu cầu**: Nếu sản phẩm không đáp ứng điều kiện bảo hành, hệ thống thông báo cho khách hàng về việc từ chối bảo hành và cung cấp lý do chi tiết.

5. **Nếu sản phẩm đủ điều kiện bảo hành, xác nhận với khách hàng**: Nếu sản phẩm đáp ứng điều kiện bảo hành, hệ thống gửi thông báo xác nhận với khách hàng và hướng dẫn cách gửi sản phẩm đến trung tâm bảo hành.
6. **Tiếp nhận sản phẩm bảo hành**: Sau khi nhận sản phẩm từ khách hàng, hệ thống cập nhật trạng thái tiếp nhận và gửi thông báo xác nhận cho khách hàng.
7. **Tiến hành kiểm tra và bảo hành**: Kỹ thuật viên tiến hành kiểm tra sản phẩm để xác định lỗi và sửa chữa. Nếu sản phẩm không thể sửa chữa, hệ thống sẽ chuyển sang quy trình thay thế sản phẩm.
8. **Theo dõi quá trình bảo hành**: Hệ thống theo dõi tiến độ bảo hành, bao gồm các bước như kiểm tra lỗi, thay thế linh kiện và kiểm tra chất lượng sau sửa chữa.
9. **Hoàn tất bảo hành**: Sau khi sản phẩm đã được sửa chữa hoặc thay thế, hệ thống cập nhật trạng thái hoàn thành bảo hành.
10. **Gửi thông báo hoàn tất bảo hành cho khách hàng**: Hệ thống gửi thông báo qua email hoặc SMS cho khách hàng về việc sản phẩm đã hoàn tất bảo hành và sẵn sàng trả lại.
11. **Gửi sản phẩm cho khách hàng**: Sản phẩm được gửi trả lại cho khách hàng thông qua các đơn vị vận chuyển hoặc khách hàng có thể đến trực tiếp trung tâm bảo hành để nhận lại.
12. **Cập nhật thông tin bảo hành vào cơ sở dữ liệu**: Hệ thống lưu trữ thông tin về lần bảo hành vào hồ sơ của sản phẩm để quản lý về sau.
13. **Khảo sát mức độ hài lòng của khách hàng**: Sau khi hoàn tất bảo hành, hệ thống gửi yêu cầu khảo sát mức độ hài lòng của khách hàng về dịch vụ bảo hành.
14. **Lưu trữ phản hồi của khách hàng**: Các phản hồi từ khảo sát được lưu trữ để đánh giá và cải thiện quy trình bảo hành trong tương lai.

3.30. Gửi nhận tiền trực tuyến

Mô tả nghiệp vụ:

Hệ thống gửi và nhận tiền trực tuyến cho phép người dùng gửi tiền từ tài khoản của họ đến tài khoản của người nhận qua nền tảng ngân hàng hoặc ví điện tử. Sau khi người dùng nhập thông tin giao dịch, hệ thống sẽ xác minh tài khoản người nhận, xử lý giao dịch và gửi thông báo cho cả hai bên. Nếu giao dịch thành công, tiền sẽ được chuyển ngay vào tài khoản của người nhận và trạng thái giao dịch sẽ được lưu trữ.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống gửi và nhận tiền trực tuyến dựa trên kiến trúc hướng dịch vụ (SOA), đảm bảo quy trình từ khi người dùng nhập thông tin giao dịch, xác minh tài khoản, cho đến khi xử lý giao dịch thành công và gửi thông báo được thực hiện một chính xác.

Mô tả chi tiết các bước nghiệp vụ:

1. **Người gửi nhập thông tin giao dịch**: Người gửi nhập thông tin cần thiết để thực hiện giao dịch chuyển tiền, bao gồm số tài khoản người nhận, số tiền, và mô tả giao dịch (nếu có).
2. **Xác minh tài khoản người nhận**: Hệ thống kiểm tra tính hợp lệ của số tài khoản người nhận và xác nhận rằng tài khoản đó có thể nhận tiền.
3. **Kiểm tra số dư của người gửi**: Hệ thống kiểm tra số dư tài khoản của người gửi để đảm bảo rằng tài khoản có đủ tiền để thực hiện giao dịch.
4. **Xác minh thông tin giao dịch**: Hệ thống xác minh lại thông tin giao dịch như số tiền và tài khoản người nhận.
5. **Nếu không đủ số dư, thông báo lỗi**: Nếu tài khoản của người gửi không đủ số dư để thực hiện giao dịch, hệ thống sẽ thông báo cho người gửi về tình trạng này.
6. **Nếu giao dịch hợp lệ, gửi mã xác nhận (OTP)**: Hệ thống gửi mã OTP (One-Time Password) đến người gửi qua SMS hoặc ứng dụng xác thực để xác minh lại ý định thực hiện giao dịch.
7. **Người gửi nhập mã OTP để xác nhận giao dịch**: Người gửi nhập mã OTP để hoàn tất quá trình xác nhận giao dịch.
8. **Xử lý giao dịch**: Hệ thống thực hiện giao dịch chuyển tiền từ tài khoản của người gửi sang tài khoản của người nhận.
9. **Cập nhật số dư tài khoản**: Hệ thống tự động cập nhật số dư tài khoản của cả người gửi và người nhận sau khi giao dịch được xử lý.
10. **Gửi thông báo thành công cho người gửi**: Hệ thống gửi thông báo cho người gửi về việc giao dịch đã hoàn thành thành công.
11. **Gửi thông báo nhận tiền cho người nhận**: Hệ thống gửi thông báo cho người nhận rằng họ đã nhận được tiền từ người gửi.
12. **Lưu trữ thông tin giao dịch**: Hệ thống lưu trữ chi tiết giao dịch trong cơ sở dữ liệu để phục vụ cho việc tra cứu và kiểm toán sau này.
13. **Xem lịch sử giao dịch**: Người dùng (cả người gửi và người nhận) có thể truy cập vào lịch sử giao dịch của họ trên hệ thống để kiểm tra chi tiết các giao dịch đã thực hiện.

3.31. Xử lý yêu cầu bảo hiểm khách hàng

Mô tả nghiệp vụ:

Hệ thống xử lý yêu cầu bảo hiểm cho phép khách hàng gửi yêu cầu bồi thường bảo hiểm trực tuyến. Khi khách hàng gửi yêu cầu, hệ thống sẽ xác minh thông tin khách hàng và hợp đồng bảo hiểm, sau đó tính toán số tiền bồi thường dựa trên điều khoản của hợp đồng. Nếu yêu cầu hợp lệ, hệ thống sẽ chấp nhận và cập nhật vào cơ sở dữ liệu. Nếu không, khách hàng sẽ nhận được thông báo từ chối cùng lý do chi tiết.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để xử lý yêu cầu bảo hiểm khách hàng, từ khâu khởi tạo yêu cầu, xác minh thông tin, tính toán bồi thường, đến khi hoàn tất quá trình xử lý và lưu trữ thông tin vào hệ thống.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo yêu cầu bảo hiểm**: Khách hàng truy cập hệ thống và tạo yêu cầu bảo hiểm mới, cung cấp thông tin về sự kiện cần bảo hiểm.
2. **Xác minh thông tin khách hàng**: Hệ thống xác minh tính hợp lệ của thông tin khách hàng, bao gồm các chi tiết như CMND/CCCD và địa chỉ liên hệ.
3. **Xác minh thông tin hợp đồng bảo hiểm**: Kiểm tra hợp đồng bảo hiểm của khách hàng để xác nhận rằng yêu cầu nằm trong phạm vi bảo hiểm.
4. **Kiểm tra điều kiện yêu cầu**: Nếu yêu cầu không hợp lệ hoặc không thuộc phạm vi bảo hiểm, hệ thống sẽ từ chối.
5. **Thông báo từ chối**: Gửi thông báo cho khách hàng về việc từ chối yêu cầu bảo hiểm và lý do từ chối.
6. **Thông báo chấp nhận**: Nếu yêu cầu hợp lệ, hệ thống sẽ gửi thông báo chấp nhận cho khách hàng, xác nhận rằng yêu cầu sẽ được xử lý tiếp.
7. **Tính toán số tiền bồi thường**: Hệ thống tự động tính toán số tiền bồi thường dựa trên các điều khoản trong hợp đồng bảo hiểm.
8. **Cập nhật hồ sơ yêu cầu bảo hiểm**: Thông tin yêu cầu được cập nhật vào hồ sơ bảo hiểm của khách hàng.
9. **Lưu trữ yêu cầu vào cơ sở dữ liệu**: Hệ thống lưu trữ chi tiết yêu cầu vào cơ sở dữ liệu để phục vụ cho việc tra cứu và xử lý sau này.
10. **In bản cứng của yêu cầu**: Tạo bản cứng của yêu cầu bảo hiểm để lưu trữ trong hồ sơ vật lý, nếu cần.

3.32. Vay tiền trực tuyến

Mô tả nghiệp vụ:

Một hệ thống quản lý quy trình vay vốn trực tuyến cho phép khách hàng cá nhân đăng ký vay vốn, cung cấp thông tin tài chính và xác minh các yêu cầu vay. Khi khách hàng gửi đơn đăng ký, hệ thống sẽ xác minh thông tin cá nhân, kiểm tra điểm tín dụng và đánh giá rủi ro. Sau đó, hệ thống sẽ đề xuất các khoản vay phù hợp, khách hàng sẽ chọn khoản vay và ký hợp đồng trực tuyến. Khi khoản vay được phê duyệt, số tiền sẽ được giải ngân vào tài khoản của khách hàng và các đợt thanh toán sẽ được lên lịch.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để thực hiện quy trình vay vốn trực tuyến, từ lúc khách hàng đăng ký vay, xác minh thông tin, đánh giá khả năng thanh toán, đến khi phê duyệt khoản vay, giải ngân và quản lý các đợt thanh toán.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo yêu cầu vay vốn**: Khách hàng đăng nhập vào hệ thống và khởi tạo yêu cầu vay vốn mới.
2. **Cung cấp thông tin cá nhân và tài chính**: Khách hàng cung cấp các thông tin cá nhân (CMND/CCCD, địa chỉ, thông tin tài khoản) và minh chứng thu nhập hàng tháng.
3. **Xác minh danh tính và thông tin cá nhân**: Hệ thống xác minh tính hợp lệ của thông tin cá nhân và danh tính khách hàng thông qua kết nối với cơ sở dữ liệu hoặc các dịch vụ bên thứ ba.
4. **Kiểm tra điểm tín dụng**: Hệ thống truy cập vào các dịch vụ điểm tín dụng để kiểm tra lịch sử tín dụng của khách hàng.
5. **Đánh giá khả năng thanh toán**: Dựa trên thông tin tài chính và điểm tín dụng, hệ thống đánh giá khả năng thanh toán của khách hàng và tính toán khoản vay phù hợp.
6. **Đề xuất khoản vay và điều kiện**: Hệ thống đề xuất các gói vay phù hợp với khả năng thanh toán của khách hàng và hiển thị chi tiết các điều khoản (lãi suất, thời hạn, và phương thức thanh toán).
7. **Chọn khoản vay**: Khách hàng chọn gói vay phù hợp với nhu cầu và khả năng tài chính của mình.
8. **Ký hợp đồng điện tử**: Hệ thống tạo hợp đồng điện tử và khách hàng ký xác nhận qua hình thức xác minh (ví dụ: OTP, email …).
9. **Phê duyệt khoản vay**: Bộ phận tín dụng phê duyệt khoản vay dựa trên các tiêu chí nội bộ và chấp nhận các điều khoản hợp đồng.
10. **Giải ngân khoản vay**: Sau khi phê duyệt, hệ thống chuyển số tiền vay vào tài khoản của khách hàng.
11. **Lên lịch các đợt thanh toán**: Hệ thống tự động lên lịch thanh toán dựa trên thời hạn vay và kỳ hạn thanh toán.
12. **Gửi thông báo giải ngân thành công**: Hệ thống gửi thông báo cho khách hàng về hợp đồng vay, việc giải ngân khoản vay và các thông tin chi tiết khác.

3.33. Đồng bộ dữ liệu phân tán

Mô tả nghiệp vụ:

Một hệ thống dữ liệu phân tán cần đồng bộ hóa dữ liệu giữa các cụm dữ liệu (data clusters) để đảm bảo tính nhất quán trong trường hợp có nhiều thao tác ghi xảy ra đồng thời ở các vị trí địa lý khác nhau. Hệ thống này phải xử lý các trường hợp xung đột dữ liệu, cung cấp độ trễ thấp cho người dùng, và đảm bảo không xảy ra mất mát dữ liệu. Đồng thời, hệ thống cần lưu trữ thông tin chi tiết về các phiên bản khác nhau của dữ liệu và hỗ trợ khôi phục khi có sự cố.

Yêu cầu phân tích thiết kế hướng dịch vụ cho nghiệp vụ (Usecase):

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để thực hiện quy trình đồng bộ hóa dữ liệu phân tán, quản lý phiên bản dữ liệu, xử lý xung đột và đảm bảo độ nhất quán cuối (eventual consistency) cho dữ liệu.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo yêu cầu ghi dữ liệu**: Người dùng gửi yêu cầu ghi dữ liệu từ một dịch vụ hoặc cụm dữ liệu gần nhất.
2. **Lưu trữ phiên bản local**: Dữ liệu được lưu tạm thời vào phiên bản local trên cụm dữ liệu gần nhất, cùng với thông tin phiên bản.
3. **Khởi tạo quy trình đồng bộ dữ liệu phân tán**: Hệ thống tự động bắt đầu quy trình đồng bộ để truyền dữ liệu tới các cụm dữ liệu khác.
4. **Truyền dữ liệu tới các cụm dữ liệu khác**: Hệ thống truyền dữ liệu tới các cụm dữ liệu khác, bao gồm thông tin phiên bản và thời gian thực hiện.
5. **Kiểm tra phiên bản dữ liệu**: Tại các cụm dữ liệu khác, hệ thống kiểm tra phiên bản dữ liệu và so sánh với phiên bản hiện có.
6. **Phát hiện xung đột dữ liệu**: Nếu có phiên bản xung đột (các phiên bản cùng thời điểm từ các cụm khác), hệ thống sẽ kích hoạt quy trình xử lý xung đột.
7. **Xử lý xung đột dữ liệu**: Hệ thống sử dụng chiến lược xử lý xung đột, như ưu tiên phiên bản mới nhất hoặc hợp nhất các phiên bản.
8. **Ghi nhận phiên bản dữ liệu**: Phiên bản dữ liệu cuối cùng được lưu vào hệ thống, đồng bộ tại tất cả các cụm dữ liệu liên quan.
9. **Ghi lại lịch sử phiên bản**: Hệ thống lưu trữ thông tin về các phiên bản trước để phục vụ cho việc khôi phục khi cần.
10. **Đảm bảo nhất quán cuối**: Hệ thống kiểm tra và xác nhận rằng dữ liệu trên các cụm đã đạt độ nhất quán (eventual consistency).
11. **Phát hiện lỗi trong quá trình đồng bộ**: Nếu phát hiện lỗi trong quá trình truyền dữ liệu, hệ thống sẽ tự động kích hoạt quy trình khôi phục.
12. **Khởi động quy trình khôi phục dữ liệu**: Hệ thống khôi phục phiên bản ổn định nhất của dữ liệu và gửi thông báo về trạng thái đồng bộ hóa.
13. **Gửi thông báo hoàn tất**: Hệ thống gửi thông báo hoàn tất đồng bộ và cập nhật phiên bản cho các dịch vụ phụ thuộc.

3.34. Cảnh báo giám sát thời gian thực

Mô tả nghiệp vụ:

Một hệ thống giám sát an ninh thời gian thực sử dụng nhiều nguồn dữ liệu, bao gồm video từ camera, cảm biến nhiệt độ, và báo động để phát hiện và cảnh báo về các mối đe dọa tiềm tàng. Hệ thống xử lý dữ liệu từ các nguồn khác nhau này một cách đồng thời trong thời gian thực để đảm bảo các cảnh báo được gửi ngay lập tức đến trung tâm giám sát. Dữ liệu phân tích sẽ được lưu trữ để phục vụ cho việc đánh giá và phân tích sau này.

Yêu cầu:

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để thực hiện quy trình giám sát an ninh thời gian thực, từ lúc thu thập dữ liệu, xử lý thời gian thực, đưa ra cảnh báo, đến lưu trữ dữ liệu để phục vụ cho phân tích và đánh giá sau này.

Mô tả chi tiết các bước nghiệp vụ:

1. **Thu thập dữ liệu từ các nguồn**: Hệ thống nhận dữ liệu video từ camera, dữ liệu nhiệt độ từ cảm biến, và các tín hiệu báo động từ thiết bị giám sát.
2. **Khởi tạo quy trình xử lý luồng dữ liệu**: Dữ liệu từ các nguồn được chuyển vào các luồng xử lý dữ liệu thời gian thực.
3. **Phân tích video thời gian thực**: Hệ thống chạy các thuật toán nhận diện khuôn mặt và phát hiện hành vi bất thường từ dữ liệu video.
4. **Phân tích dữ liệu cảm biến**: Dữ liệu nhiệt độ từ cảm biến được phân tích để phát hiện các thay đổi đột ngột (ví dụ: tăng nhiệt độ bất thường có thể là dấu hiệu của cháy).
5. **Tích hợp và hợp nhất dữ liệu**: Hệ thống kết hợp dữ liệu từ các nguồn khác nhau (video, cảm biến) để tạo ra một bức tranh toàn cảnh và cung cấp thông tin bổ sung cho các phân tích tiếp theo.
6. **Phát hiện mối đe dọa và xung đột tín hiệu**: Khi dữ liệu chỉ ra dấu hiệu bất thường, hệ thống phát hiện mối đe dọa và xử lý xung đột trong các tín hiệu nhận được.
7. **Xử lý các quy tắc giám sát**: Hệ thống áp dụng các quy tắc giám sát được định nghĩa trước (ví dụ: quy tắc về mức nhiệt độ tối đa, khoảng cách người lạ gần tài sản).
8. **Tạo cảnh báo sự cố**: Nếu hệ thống phát hiện mối đe dọa, cảnh báo được tạo và gửi đến trung tâm giám sát.
9. **Xác thực cảnh báo**: Hệ thống xác thực dữ liệu từ các nguồn để tránh trường hợp cảnh báo giả và tăng độ chính xác.
10. **Gửi cảnh báo đến nhân viên giám sát**: Cảnh báo thời gian thực được gửi ngay đến nhân viên giám sát thông qua ứng dụng hoặc SMS.
11. **Lưu trữ dữ liệu phân tích**: Tất cả dữ liệu và cảnh báo được lưu trữ để phục vụ cho các phân tích nâng cao và điều tra sau này.

3.35. Phân tích hành vi người dùng

Mô tả nghiệp vụ:

Hệ thống phân tích hành vi người dùng trực tuyến và cung cấp đề xuất cá nhân hóa trong thời gian thực cho các trang web thương mại điện tử hoặc nền tảng truyền thông trực tuyến. Hệ thống phải thu thập và phân tích các sự kiện từ người dùng (như nhấp chuột, thời gian xem trang, và tương tác) để nhận diện sở thích và xu hướng. Dựa trên các hành vi này, hệ thống sẽ đưa ra các đề xuất nội dung hoặc sản phẩm phù hợp trong thời gian thực, đồng thời lưu trữ dữ liệu phân tích.

Yêu cầu:

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để thực hiện quy trình phân tích hành vi người dùng, xử lý dữ liệu thời gian thực và cung cấp đề xuất cá nhân hóa. Quy trình này cần đảm bảo tính nhanh chóng, độ chính xác cao và hiệu suất ổn định cho các phiên người dùng đồng thời.

Mô tả chi tiết các bước nghiệp vụ:

1. **Thu thập sự kiện từ người dùng**: Hệ thống thu thập các sự kiện từ người dùng như nhấp chuột, thời gian trên trang, và các hành động khác trong thời gian thực.
2. **Khởi tạo quy trình phân tích hành vi**: Khi sự kiện từ người dùng được gửi lên, hệ thống khởi động quy trình phân tích dữ liệu hành vi.
3. **Phân tích các sự kiện trực tuyến**: Dữ liệu sự kiện được xử lý và phân tích ngay lập tức để nhận diện xu hướng và các sản phẩm/nội dung mà người dùng quan tâm.
4. **Tính toán điểm sở thích cá nhân**: Hệ thống cập nhật điểm sở thích cá nhân của người dùng dựa trên các yếu tố như tần suất tương tác, thời gian xem và các yếu tố nhân khẩu học.
5. **Kết nối với mô hình đề xuất**: Dữ liệu hành vi được kết nối với mô hình đề xuất để tạo ra các nội dung hoặc sản phẩm cá nhân hóa.
6. **Đưa ra đề xuất thời gian thực**: Hệ thống hiển thị đề xuất cá nhân hóa cho người dùng dựa trên kết quả từ mô hình đề xuất.
7. **Cập nhật hồ sơ hành vi người dùng**: Hệ thống lưu trữ hồ sơ hành vi của người dùng, cập nhật thông tin về sở thích và xu hướng theo thời gian.
8. **Theo dõi và điều chỉnh mô hình đề xuất**: Dựa trên dữ liệu tích lũy, hệ thống điều chỉnh các tham số mô hình đề xuất để tối ưu hóa độ chính xác và tăng hiệu quả cá nhân hóa.
9. **Gửi thông báo nhắc nhở qua email/SMS**: Khi người dùng rời khỏi trang web, hệ thống gửi nhắc nhở qua email hoặc SMS với các sản phẩm liên quan hoặc ưu đãi cá nhân.
10. **Lưu trữ dữ liệu và phân tích sau phiên truy cập**: Toàn bộ dữ liệu hành vi được lưu trữ và phân tích sau phiên truy cập để phục vụ cho các chiến dịch tiếp thị và nhắm đến đối tượng.
11. **Phân tích và tối ưu hóa chiến dịch tiếp thị nhắm đến từng cá nhân**: Hệ thống sử dụng dữ liệu hành vi để thiết kế các chiến dịch tiếp thị nhắm đến từng khách hàng và tối ưu hóa chi phí quảng cáo.

3.36. Thu thập dữ liệu tự động

Mô tả nghiệp vụ:

Một hệ thống cần thu thập và cập nhật dữ liệu liên tục từ nhiều nguồn trực tuyến, bao gồm trang web, API, và các dịch vụ công khai khác. Quy trình này yêu cầu khả năng

xử lý khối lượng lớn dữ liệu, kiểm tra và lọc thông tin trùng lặp, quản lý và xử lý các thay đổi dữ liệu để đảm bảo dữ liệu luôn mới nhất và đồng bộ. Hệ thống cũng cần hỗ trợ các yêu cầu tối ưu hóa về tần suất thu thập nhằm hạn chế rủi ro bị chặn từ các nguồn dữ liệu và kiểm soát tài nguyên sử dụng.

Yêu cầu:

Phân tích và thiết kế hệ thống theo kiến trúc hướng dịch vụ (SOA) để thực hiện quy trình crawl và cập nhật dữ liệu từ nhiều nguồn khác nhau, bao gồm xử lý, tối ưu hóa tần suất crawl, kiểm tra trùng lặp và xử lý thay đổi dữ liệu. Hệ thống phải đảm bảo tính ổn định, hiệu quả, và tối ưu hóa tài nguyên.

Mô tả chi tiết các bước nghiệp vụ:

1. **Khởi tạo yêu cầu crawl dữ liệu**: Hệ thống thiết lập lịch trình crawl dữ liệu từ các nguồn được chỉ định.
2. **Xác định tần suất crawl cho từng nguồn**: Hệ thống thiết lập tần suất crawl riêng biệt dựa trên mức độ cập nhật của từng nguồn để tránh quá tải và bị chặn.
3. **Thu thập dữ liệu từ các nguồn**: Hệ thống lấy dữ liệu từ các nguồn trực tuyến, bao gồm trang web và API công khai, đồng thời lưu lại dữ liệu thô.
4. **Kiểm tra và lọc dữ liệu trùng lặp**: Sau khi lấy dữ liệu, hệ thống tiến hành kiểm tra và lọc bỏ các bản ghi trùng lặp, chỉ giữ lại dữ liệu mới nhất.
5. **Xử lý dữ liệu thô và chuẩn hóa**: Hệ thống xử lý dữ liệu thô, chuyển đổi sang định dạng chuẩn và loại bỏ các phần không cần thiết.
6. **So sánh và phát hiện thay đổi trong dữ liệu**: Dữ liệu mới được so sánh với dữ liệu cũ để phát hiện các thay đổi. Chỉ những bản ghi đã thay đổi mới được cập nhật vào hệ thống.
7. **Lưu trữ dữ liệu mới và cập nhật**: Dữ liệu đã qua xử lý được lưu trữ trong cơ sở dữ liệu, đồng thời cập nhật các bản ghi cũ với các thay đổi mới.
8. **Tối ưu hóa lịch trình crawl**: Dựa trên mức độ thay đổi của từng nguồn, hệ thống điều chỉnh tần suất crawl để tối ưu hiệu suất.
9. **Xử lý lỗi và thử lại**: Nếu có lỗi khi lấy dữ liệu từ một nguồn nào đó, hệ thống sẽ thử lại sau một khoảng thời gian và ghi nhận lỗi.
10. **Ghi nhật ký hoạt động**: Hệ thống lưu trữ thông tin về các hoạt động crawl, bao gồm thời gian, nguồn dữ liệu, và các thay đổi phát hiện được.
11. **Phân tích hiệu suất hệ thống**: Hệ thống phân tích các thông số như thời gian xử lý, tần suất crawl, và số lượng dữ liệu trùng lặp để tối ưu hóa quy trình.
12. **Cảnh báo khi có thay đổi đột ngột từ nguồn**: Nếu phát hiện thay đổi bất thường (ví dụ: dữ liệu tăng đột biến), hệ thống gửi cảnh báo để xem xét và xử lý kịp thời.
13. **Xuất báo cáo định kỳ**: Hệ thống tạo báo cáo định kỳ về kết quả crawl, tỷ lệ thành công và hiệu suất của quy trình để phục vụ cải tiến.

TÀI LIỆU THAM KHẢO

[1] Thomas Erl, Service-Oriented Architecture: Analysis & Design for Services and Microservices, 2nd Edition, 2016

[2] Chris Richardson, Microservices Pattern: Microservice Architecture Pattern, 2019

[3] Sam Newman, Building Microservices 2nd Edition, 2021

[4] Sam Newman, Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith, 1st Edition, 2019

[5] https://microservices.io/

PHỤ LỤC ĐÁP ÁN VÀ GỢI Ý THAM KHẢO

PHẦN 1: CÂU HỎI LÝ THUYẾT

1.1. Các khái niệm cơ bản với SOA

1	**A**		**7**	**B**		**13**	**B**		**19**	**C**		**25**	**A**
2	**D**		**8**	**A**		**14**	**C**		**20**	**A**		**26**	**C**
3	**D**		**9**	**B**		**15**	**B**		**21**	**B**		**27**	**A**
4	**B**		**10**	**A**		**16**	**A**		**22**	**C**		**28**	**C**
5	**D**		**11**	**A**		**17**	**D**		**23**	**D**			
6	**A**		**12**	**B**		**18**	**B**		**24**	**D**			

1.2. Phân tích và mô hình hóa dịch vụ

1	**D**		**8**	**B**		**15**	**A**		**22**	**C**		**29**	**C**
2	**A**		**9**	**B**		**16**	**A**		**23**	**C**		**30**	**A**
3	**A**		**10**	**C**		**17**	**A**		**24**	**A**		**31**	**D**
4	**C**		**11**	**B**		**18**	**B**		**25**	**C**		**32**	**D**
5	**D**		**12**	**D**		**19**	**D**		**26**	**A**		**33**	**B**
6	**D**		**13**	**B**		**20**	**A**		**27**	**B**		**34**	**A**
7	**A**		**14**	**A**		**21**	**A**		**28**	**D**			

1.3. Thiết kế hợp đồng và API dịch vụ

1	**A**		**15**	**D**		**29**	**C**		**43**	**B**		**57**	**A**
2	**D**		**16**	**D**		**30**	**C**		**44**	**C**		**58**	**B**
3	**C**		**17**	**A**		**31**	**B**		**45**	**A**		**59**	**B**
4	**B**		**18**	**A**		**32**	**A**		**46**	**A**		**60**	**B**
5	**B**		**19**	**C**		**33**	**A**		**47**	**D**		**61**	**A**
6	**C**		**20**	**A**		**34**	**C**		**48**	**A**		**62**	**D**
7	**A**		**21**	**B**		**35**	**D**		**49**	**B**		**63**	**B**
8	**A**		**22**	**C**		**36**	**C**		**50**	**A**		**64**	**B**
9	**A**		**23**	**B**		**37**	**B**		**51**	**A**		**65**	**B**
10	**B**		**24**	**D**		**38**	**C**		**52**	**C**		**66**	**B**
11	**B**		**25**	**C**		**39**	**A**		**53**	**A**			
12	**C**		**26**	**A**		**40**	**B**		**54**	**A**			
13	**D**		**27**	**C**		**41**	**C**		**55**	**A**			
14	**B**		**28**	**A**		**42**	**C**		**56**	**A**			

PHẦN 2: MẪU THIẾT KẾ HƯỚNG DỊCH VỤ

2.1. Kiến trúc ứng dụng

2.1.1. Hướng dịch vụ

Thiết kế một kiến trúc cấu trúc ứng dụng như một tập hợp các thành phần có thể triển khai độc lập, ít phụ thuộc, gọi là services. Mỗi service bao gồm một hoặc nhiều subdomain và thuộc sở hữu của nhóm (hoặc các nhóm) sở hữu các subdomain đó.

Để có thể triển khai độc lập, mỗi service thường có kho mã nguồn (*source code repository*) riêng và luồng quy trình triển khai (*deployment pipeline*) riêng, bao gồm các bước build, kiểm thử và triển khai dịch vụ độc lập.

Một số tính năng của hệ thống sẽ là cục bộ (được triển khai bởi một service duy nhất), trong khi các tính năng khác có thể được xử lý phân tán qua nhiều service. Một hoạt động phân tân nh uvậy có thể được triển khai theo kiểu đồng bộ bằng cách sử dụng các giao thức như HTTP/REST hoặc bất đồng bộ bằng cách sử dụng message broker như Apache Kafka.

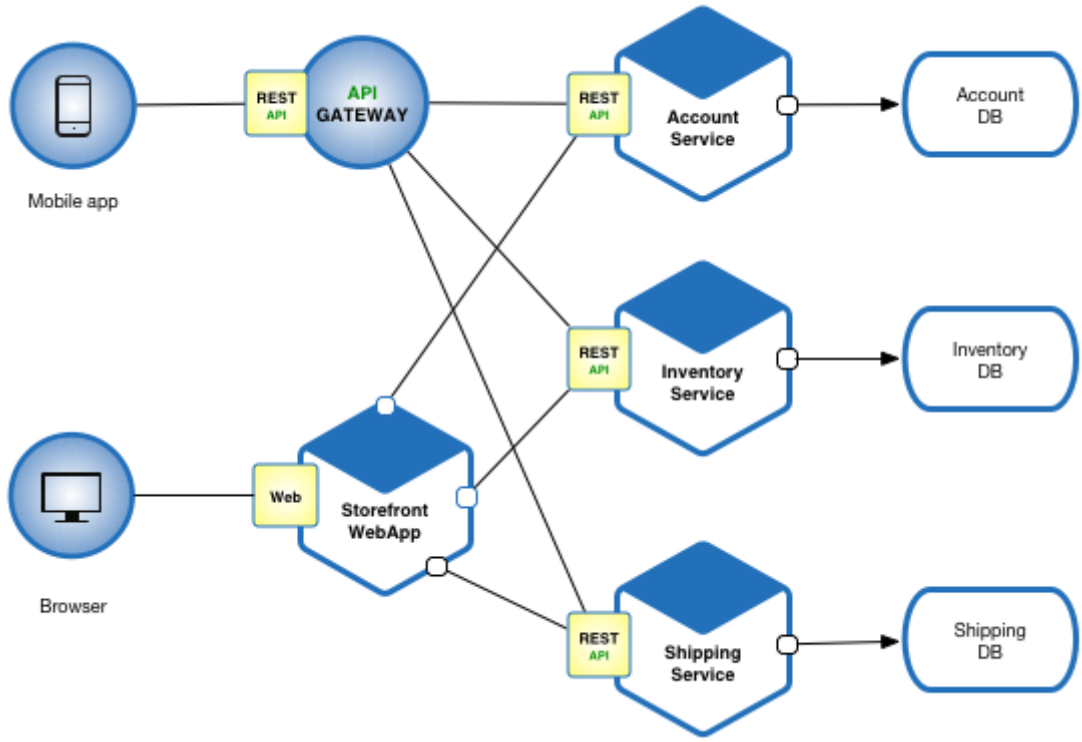
Cách tiếp cận này có một số lợi ích:

- *Simple services* - mỗi *service* bao gồm một số ít *subdomain* - có thể chỉ một - nên dễ hiểu và bảo trì hơn.
- *Team autonomy* - một đội có thể phát triển, kiểm thử và triển khai *service* của mình độc lập với các đội khác.
- *Fast deployment pipeline* - mỗi *service* nhỏ và nhanh chóng để kiểm thử, do đó có thể triển khai độc lập.
- *Support multiple technology stacks* - các *service* khác nhau có thể sử dụng các công nghệ khác nhau và có thể được nâng cấp độc lập.
- *Segregate subdomains by their characteristics* - các *subdomain* có thể được phân tách dựa trên các đặc điểm riêng biệt vào các *service* riêng để cải thiện khả năng mở rộng, tính sẵn sàng, bảo mật, v.v.

Cách tiếp cận này có một số (tiềm ẩn) hạn chế:

- Một số tính năng phân tán có thể phức tạp, khó hiểu và khó khắc phục sự cố.
- Một số tính năng phân tán có thể kém hiệu quả.
- Một số tính năng có thể cần được triển khai bằng cách sử dụng quản lý giao dịch phức tạp, cuối cùng sẽ nhất quán (không theo *ACID*), vì liên kết lỏng lẻo yêu cầu mỗi *service* phải có cơ sở dữ liệu riêng.
- Một số tính năng phân tán có thể liên quan đến sự liên kết chặt chẽ trong thời gian chạy giữa các *service*, làm giảm tính sẵn sàng của chúng.
- Nguy cơ liên kết chặt chẽ trong thời gian thiết kế giữa các *service*, yêu cầu thực hiện các thay đổi đồng bộ tốn nhiều thời gian.

Lấy ví dụ về một ứng dụng thương mại điện tử theo mô hình kiến trúc này. Với yêu cầu xử lý: 1) Nhận đơn hàng từ khách hàng; 2) Kiểm tra hàng trong kho; 3) Kiểm tra hạn mức tín dụng khả dụng và sau đó 4) Vận chuyển đơn hàng. Có thể thấy ứng dụng này bao gồm giao diện người dùng, cùng với một tập hợp các dịch vụ phía backend để xử lý kiểm tra tín dụng, quản lý hàng tồn kho, và vận chuyển đơn hàng.

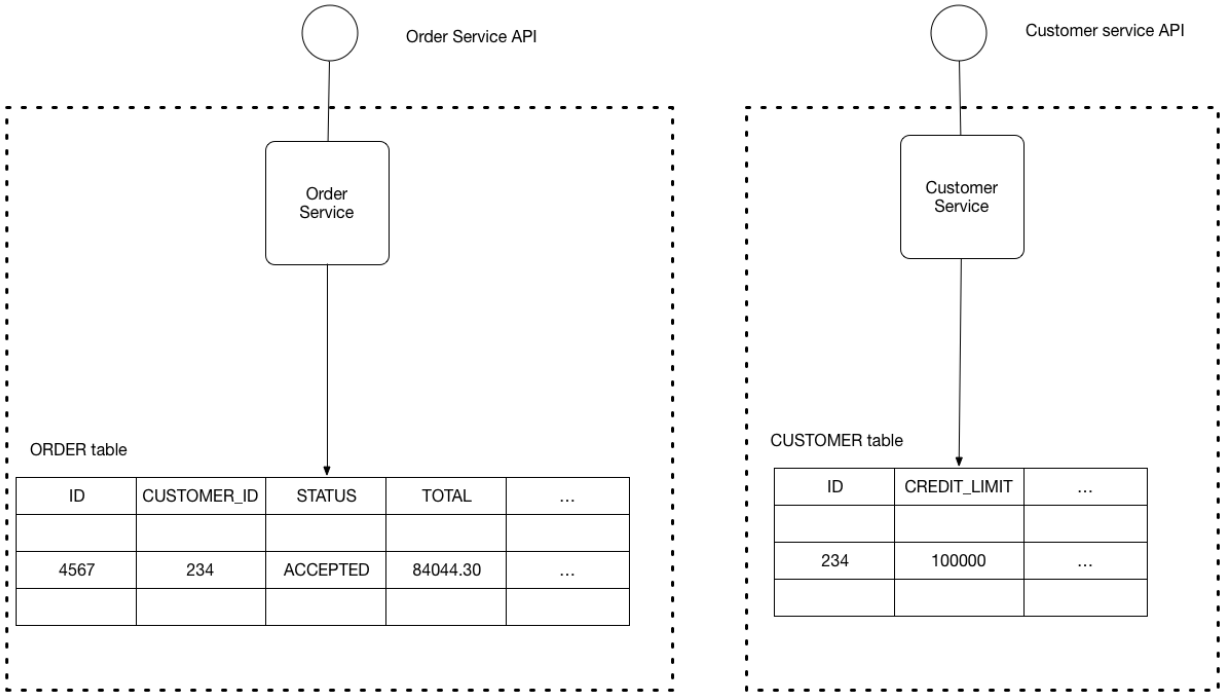


Hình 3. Minh họa mô hình kiến trúc hướng dịch vụ (nguồn microservices.io)

2.1.2. Mỗi dịch vụ một CSDL

Giữ dữ liệu lưu trữ của mỗi microservice ở dạng riêng tư đối với service đó và chỉ có thể truy cập thông qua API của nó. Các giao dịch của một service chỉ liên quan đến cơ sở dữ liệu của nó.

Sơ đồ dưới đây minh họa cấu trúc của mẫu thiết kế này.



Hình 4. Minh họa mẫu thiết kế mỗi dịch vụ một cơ sở dữ liệu riêng (nguồn microservices.com)

Cơ sở dữ liệu của service về mặt hiệu quả là một phần của việc triển khai service đó. Nó không thể được truy cập trực tiếp bởi các service khác.

Có một số cách khác nhau để giữ dữ liệu lưu trữ của service ở dạng riêng tư. Bạn không cần phải cung cấp một máy chủ cơ sở dữ liệu cho mỗi service. Ví dụ, nếu sử dụng cơ sở dữ liệu quan hệ, thì các lựa chọn là:

- *private-tables-per-service* – mỗi service sở hữu một tập hợp các bảng chỉ có thể được truy cập bởi service đó.
- *schema-per-service* – mỗi service có một schema cơ sở dữ liệu riêng tư chỉ dành cho service đó.
- *database-server-per-service* – mỗi service có máy chủ cơ sở dữ liệu riêng của nó.

private-tables-per-service và *schema-per-service* có mức chi phí thấp nhất. Sử dụng *schema-per-service* rất hấp dẫn vì nó làm cho quyền sở hữu trở nên rõ ràng hơn. Một số dịch vụ có nhu cầu xử lý và lưu lượng cao có thể cần có máy chủ cơ sở dữ liệu riêng.

Để duy trì tính mô đun này, bạn nên tạo ra các rào cản để đảm bảo mẫu thiết kế được duy trì. Ví dụ, cấp một ID người dùng cơ sở dữ liệu riêng cho mỗi service và sử dụng cơ chế kiểm soát truy cập cơ sở dữ liệu thông qua lệnh *grants*. Bởi nếu không có một rào cản nào để thực thi tính đóng gói, các nhà phát triển sẽ thường lựa chọn bỏ qua *API* của service để truy cập trực tiếp vào dữ liệu.

2.1.3. API Gateway

Có thể thấy:

- Độ chi tiết của các API do các microservice cung cấp thường khác với những gì một client cần. Các microservice thường cung cấp API có độ chi tiết cao, điều này có nghĩa là các client cần phải tương tác với nhiều service. Ví dụ, như đã mô tả ở trên, một client cần chi tiết sản phẩm phải lấy dữ liệu từ nhiều service.
- Các client khác nhau cần dữ liệu khác nhau. Ví dụ, phiên bản trên trình duyệt máy tính của trang chi tiết sản phẩm thường chi tiết hơn phiên bản trên điện thoại di động.
- Hiệu suất mạng khác nhau giữa các loại client. Ví dụ, mạng di động thường chậm hơn nhiều và có độ trễ cao hơn so với mạng không phải di động. Và tất nhiên, bất kỳ WAN nào cũng chậm hơn so với LAN. Điều này có nghĩa là một native mobile client sử dụng mạng có các đặc tính hiệu suất rất khác biệt so với LAN mà ứng dụng web phía máy chủ sử dụng. Ứng dụng web phía máy chủ có thể thực hiện nhiều yêu cầu đến các backend service mà không ảnh hưởng đến trải nghiệm người dùng, trong khi mobile client chỉ có thể thực hiện một số ít.
- Số lượng phiên bản service và vị trí của chúng (máy chủ + cổng) thay đổi một cách động.
- Phân chia thành các service có thể thay đổi theo thời gian và cần phải được ẩn khỏi client.
- Các service có thể sử dụng một tập hợp các giao thức đa dạng, trong đó một số có thể không thân thiện với web.

Như vậy, triển khai một API gateway làm điểm vào duy nhất cho tất cả các client. API gateway xử lý các yêu cầu theo hai cách. Một số yêu cầu được đơn giản hóa bằng cách chuyển tiếp đến đúng service. Các yêu cầu khác được xử lý bằng cách phân tán (fanning out) đến nhiều service.

Thay vì cung cấp một API duy nhất cho tất cả, API gateway có thể cung cấp một API khác nhau cho từng loại client. Ví dụ, Netflix API gateway chạy mã adapter cụ thể cho từng client, cung cấp cho mỗi client một API phù hợp nhất với các yêu cầu của nó. Một biến thể của mẫu này là Backends for frontends. Mẫu này định nghĩa một API gateway riêng biệt cho mỗi loại client.

Ví dụ, có ba loại client: ứng dụng web, ứng dụng di động và ứng dụng của bên thứ ba, chúng ta sẽ triển khai ba API gateway khác nhau. Mỗi cái cung cấp một API cho client của mình.

API gateway còn có thể dùng để triển khai lớp bảo mật hoặc thành phần proxy. Ưu điểm của API gateway:

- Giải phóng client khỏi việc cần hiểu biết về cách ứng dụng được phân chia thành các dịch vụ như thế nào.
- Giải phóng client khỏi vấn đề xác định vị trí của các phiên bản dịch vụ (service instances).
- Cung cấp API tối ưu cho từng client.
- Giảm số lượng yêu cầu/lượt đi về. Ví dụ, API gateway cho phép các client truy xuất dữ liệu từ nhiều dịch vụ chỉ với một lượt đi về. Số lượng yêu cầu ít hơn cũng có nghĩa là giảm chi phí và cải thiện trải nghiệm người dùng. API gateway rất cần thiết cho các ứng dụng di động.
- Đơn giản hóa client bằng cách chuyển logic gọi nhiều dịch vụ từ client sang API gateway..

API gateway cũng có một số hạn chế:

- Tăng độ phức tạp - API gateway là một thành phần bổ sung cần phải được phát triển, triển khai và quản lý.
- Tăng thời gian phản hồi do có thêm một bước mạng thông qua API gateway - tuy nhiên, đối với hầu hết các ứng dụng, chi phí của một lượt đi về thêm là không đáng kể.

2.2. Phân rã ứng dụng thành dịch vụ

2.2.1. Phân rã theo năng lực kinh doanh và theo subdomain

Phân rã theo năng lực kinh doanh

Định nghĩa các dịch vụ tương ứng với các năng lực trong mô hình kinh doanh (business capabilities). Một năng lực kinh doanh là một khái niệm từ việc mô hình hóa kiến trúc kinh doanh. Đó là điều mà một doanh nghiệp thực hiện để tạo ra giá trị. Một năng lực kinh doanh thường tương ứng với một đối tượng kinh doanh cụ thể.

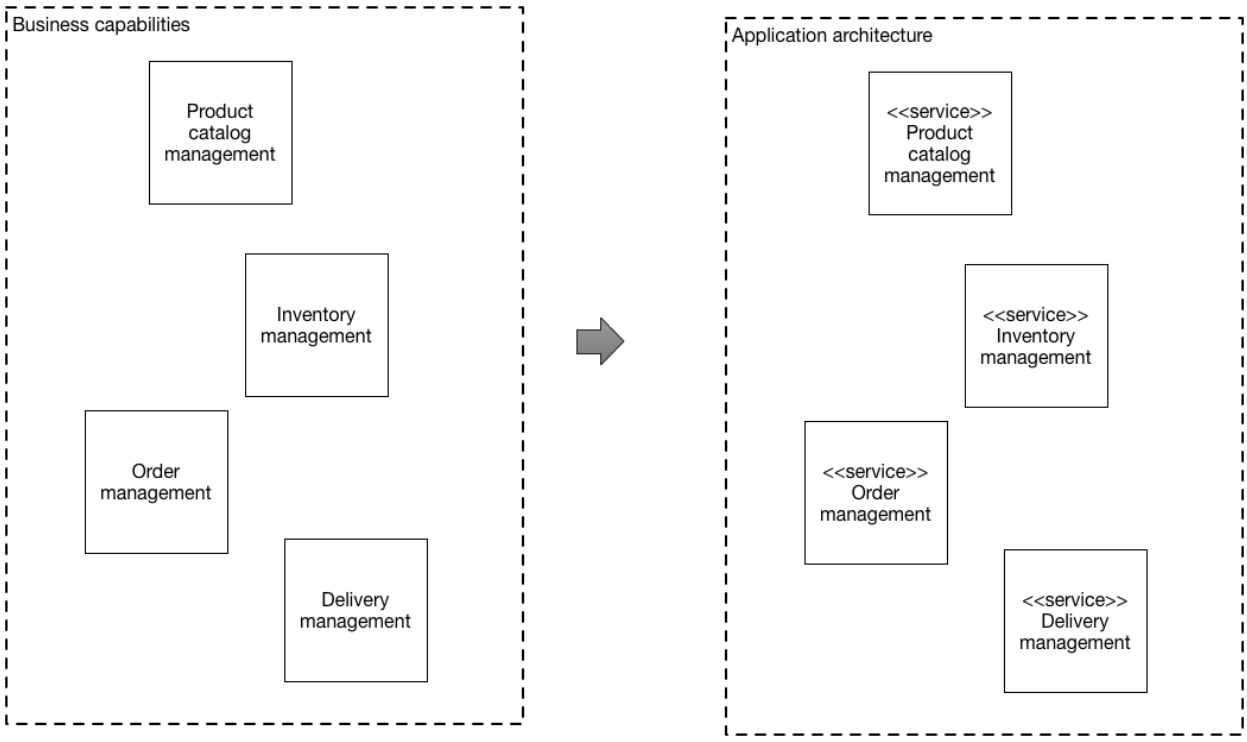
Việc xác định các năng lực kinh doanh và sau đó là các service đòi hỏi sự hiểu biết về doanh nghiệp. Các năng lực kinh doanh của một tổ chức được xác định bằng cách phân tích mục đích, cấu trúc, quy trình kinh doanh và các lĩnh vực chuyên môn của tổ chức. Các giới hạn biên được xác định tốt nhất bằng cách sử dụng một quy trình lặp đi lặp lại. Một số điểm khởi đầu có thể cân nhắc để xác định các năng lực kinh doanh là:

- Cấu trúc tổ chức - các nhóm khác nhau trong một tổ chức có thể tương ứng với các business capabilities hoặc nhóm năng lực kinh doanh.
- Mô hình domain cấp cao - các năng lực kinh doanh thường tương ứng với các domain (DDD)

Ví dụ: Xem xét các năng lực kinh doanh của một hệ thống thương mại điện tử:

- Quản lý danh mục sản phẩm
- Quản lý kho
- Quản lý đơn hàng
- Quản lý giao hàng
- …

Theo đó, kiến trúc microservice tương ứng sẽ có các service tương ứng với mỗi năng lực kinh doanh này.



Hình 5. Minh họa phân rã ứng dụng theo năng lực kinh doanh (nguồn microservices.com)

Quá trình này thể hiện nguyên tắc cơ bản trong thiết kế microservices: mỗi service nên tương ứng với một khả năng kinh doanh cụ thể. Điều này giúp đảm bảo rằng mỗi service có một trách nhiệm rõ ràng và độc lập, phù hợp với Nguyên tắc Trách nhiệm Đơn lẻ (SRP) đã được đề cập trước đó.

Phân rã theo subdomain

Định nghĩa các dịch vụ tương ứng với các domain con trong mô hình thiết kế hướng domain (Domain-Driven Design – DDD). DDD coi mỗi không gian vấn đề cần giải quyết của ứng dụng - hay là mô hình kinh doanh - như là một domain. Một domain bao gồm nhiều subdomain. Mỗi subdomain tương ứng với một phần khác nhau của mô hình kinh doanh. Các subdomain có thể được phân loại như sau:

- Core (Cốt lõi) - yếu tố tạo sự khác biệt chính cho doanh nghiệp và là phần giá trị nhất của ứng dụng.

- Supporting (Hỗ trợ) - liên quan đến hoạt động kinh doanh nhưng không tạo sự khác biệt. Có thể được triển khai nội bộ hoặc thuê ngoài.
- Generic (Chung) - không đặc thù cho doanh nghiệp và lý tưởng nhất là được triển khai bằng phần mềm có sẵn

Trong đó:

1. Miền (Domain): Đây là phạm vi tổng thể của ứng dụng, bao gồm tất cả các chức năng kinh doanh.
2. Subdomain: Là các phân vùng của miền, mỗi subdomain đại diện cho một lĩnh vực cụ thể của hoạt động kinh doanh:
3. Dịch vụ: Mỗi subdomain có một service tương ứng, thực hiện các chức năng cụ thể của subdomain đó.

Làm thế nào để xác định các subdomain?

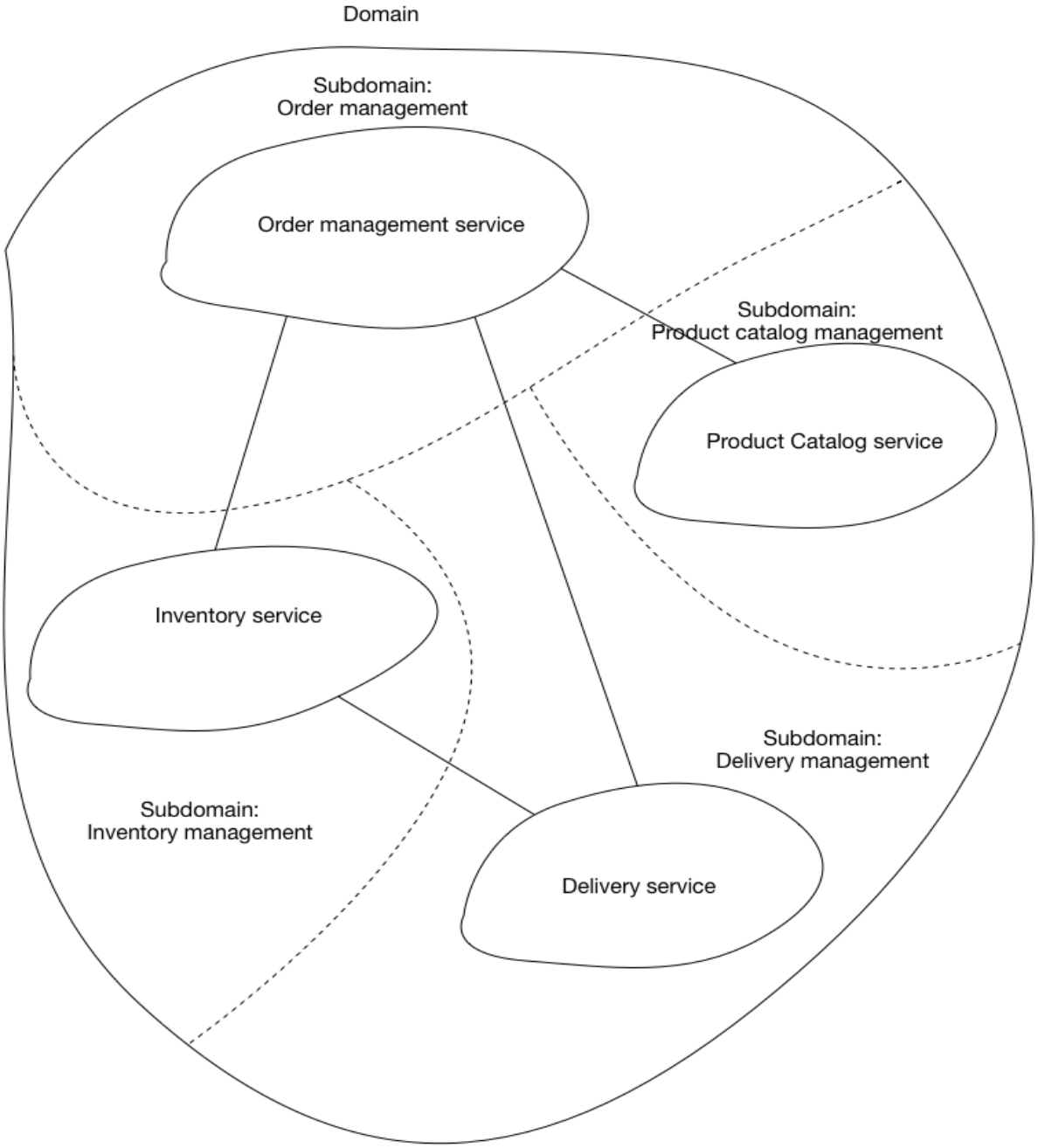
Việc xác định các subdomain và theo đó là các service đòi hỏi sự hiểu biết về lĩnh vực kinh doanh. Giống như các khả năng kinh doanh (ở trên), các subdomain được xác định bằng cách phân tích mô hình, hoạt động kinh doanh, cấu trúc tổ chức và xác định các lĩnh vực chuyên môn khác nhau. Subdomain được xác định tốt nhất thông qua một quá trình lặp đi lặp lại. Những điểm khởi đầu có thể cân nhắc để xác định subdomain là:

- Cấu trúc tổ chức - các nhóm khác nhau trong một tổ chức có thể xem xét tương ứng như là các subdomain
- Mô hình high-level domain - các subdomain thường là một miền đối tượng chính trong tổng thể mô hình.

Ví dụ: Các subdomain của một cửa hàng trực tuyến bao gồm:

- Quản lý danh mục sản phẩm
- Quản lý tồn kho
- Quản lý đơn hàng
- Quản lý giao hàng
- ...

Kiến trúc microservice tương ứng sẽ có các service tương ứng với mỗi subdomain này



Hình 6. Minh họa phân rã ứng dụng theo các miền con (nguồn microservices.com)

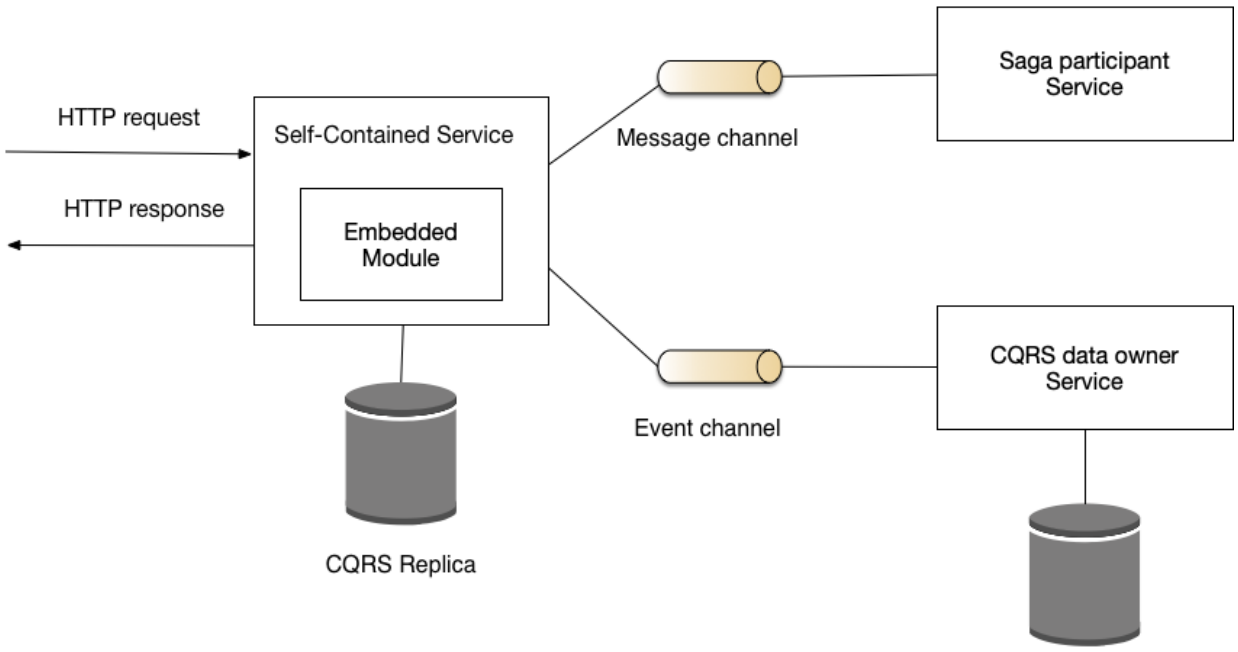
2.2.2. Dịch vụ tự chứa (Self-contained service)

Thiết kế dịch vụ sao cho nó có thể phản hồi các yêu cầu bất đồng bộ mà không cần đợi phản hồi từ bất kỳ một dịch vụ nào khác trong quy trình nghiệp vụ cần xử lý. Các dịch vụ dạng này có thể gọi là dịch vụ tự chứa (self-contained service). Có thể cân nhắc tạo các dịch vụ tự chứa theo hướng:

- Triển khai chức năng cần thiết như một module dịch vụ thay vì một dịch vụ riêng biệt

- Sử dụng các mẫu CQRS và SAGA. Trong đó, mẫu SAGA để duy trì tính nhất quán dữ liệu trong ngữ cảnh bất đồng bộ và mẫu CQRS để duy trì một bản sao của dữ liệu thuộc sở hữu của service khác.

Ví dụ: Order Service trong bài toán đã được mô tả là một ví dụ về dịch vụ tự chứa. Thao tác createOrder(), ví dụ, truy vấn một bản sao CQRS của dữ liệu thuộc sở hữu của Restaurant Service để xác thực và định giá đơn hàng, sau đó khởi tạo một saga để hoàn tất việc tạo đơn hàng.



Hình 7. Minh họa mẫu thiết kế CQRS và SAGA trong dịch vụ tự chứa (nguồn microservices.com)

2.2.3. Dịch vụ và nhóm phát triển (Service per team)

Mỗi dịch vụ thuộc sở hữu của một nhóm, nhóm này có trách nhiệm duy nhất trong việc thực hiện thay đổi. Lý tưởng nhất là mỗi nhóm chỉ phụ trách một dịch vụ. Một số thuật ngữ:

- Sở hữu và Triển khai: Mỗi nhóm sở hữu và triển khai một dịch vụ cụ thể.
- Trách nhiệm: Mỗi nhóm chịu trách nhiệm cho API của dịch vụ
- Thương lượng: Các nhóm thương lượng với nhau về API

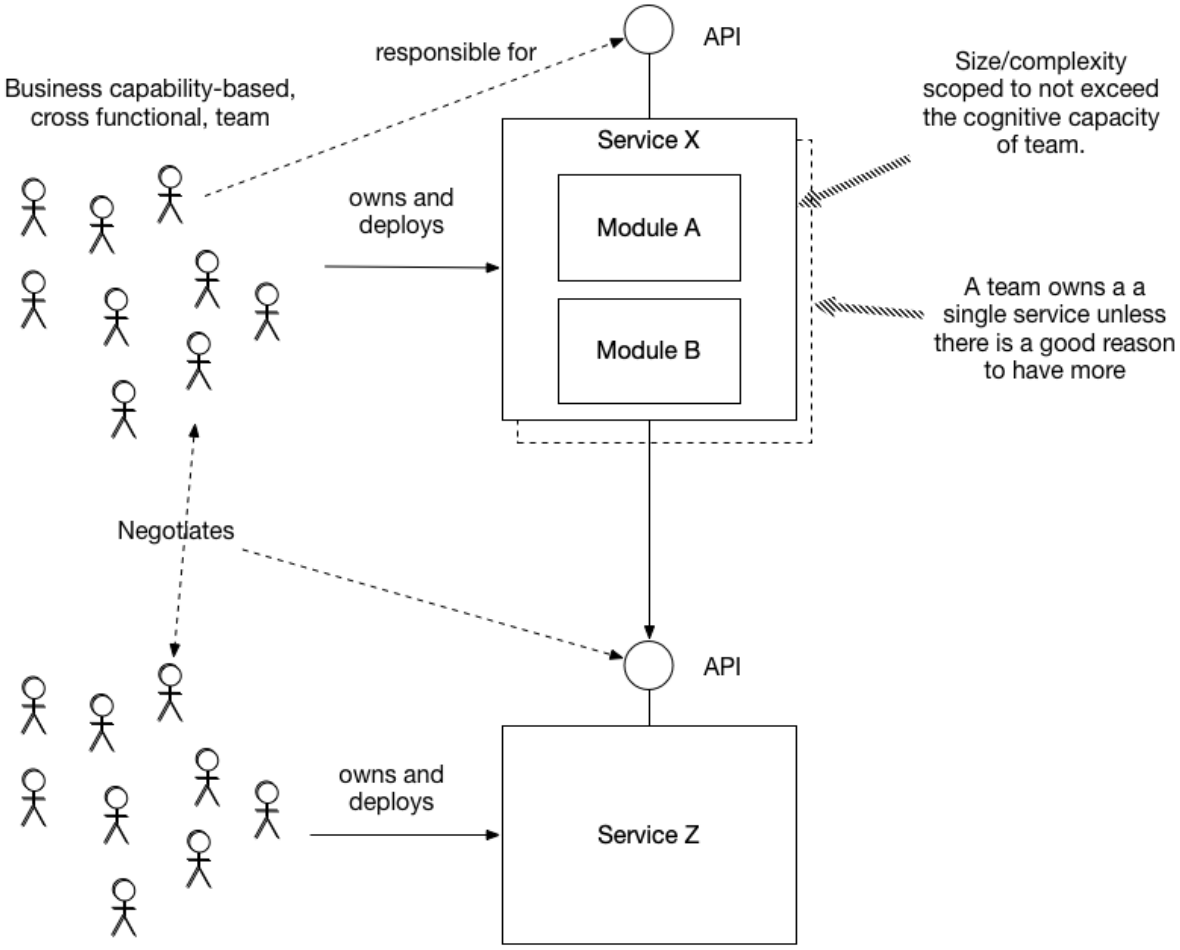
Mỗi nhóm chịu trách nhiệm cho một hoặc nhiều chức năng kinh doanh (ví dụ: năng lực kinh doanh). Một nhóm sở hữu (có trách nhiệm duy nhất thay đổi) một codebase gồm một hoặc nhiều module. Codebase được định kích thước sao cho không vượt quá khả năng xử lý của nhóm. Nhóm triển khai có thể xây dựng codebase của mình dưới dạng một hoặc nhiều dịch vụ.

Mô hình mỗi nhóm một dịch vụ có những ưu điểm sau:

- Cho phép mỗi nhóm tự chủ và làm việc với sự phối hợp tối thiểu với các nhóm khác
- Cho phép các nhóm liên kết lỏng lẻo, ít phụ thuộc
- Cải thiện chất lượng code nhờ quyền sở hữu code lâu dài

Tuy nhiên, nó cũng có thể gặp hạn chế nếu việc triển khai các tính năng kinh doanh trải rộng trên nhiều dịch vụ phức tạp hơn và đòi hỏi nhiều sự hợp tác giữa các nhóm.

Ví dụ minh họa:



Hình 8. Minh họa mối quan hệ giữa dịch vụ và nhóm phát triển (nguồn microservices.com)

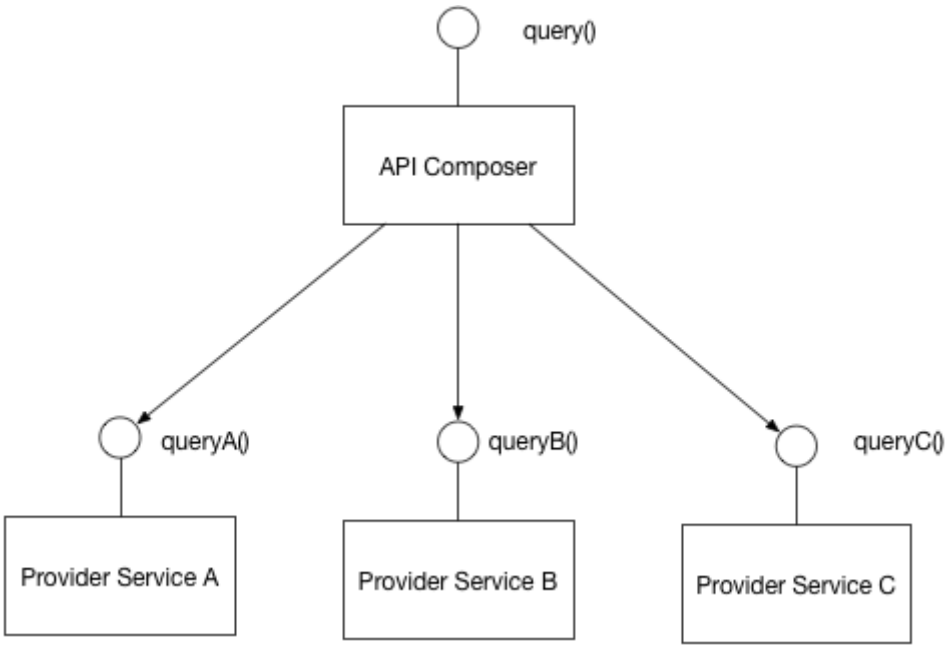
2.3. Giao tiếp và triển khai dịch vụ

2.3.1. API Composition

Triển khai thực thi một truy vấn thông qua việc định nghĩa một thành phần trung gian là *API Composer*. Trình trung gian này sẽ triệu gọi các dịch vụ sở hữu dữ liệu tương ứng và thực hiện việc tổng hợp kết quả trong bộ nhớ.

Đây là cách đơn giản để truy vấn dữ liệu tổng hợp trong kiến trúc microservice. Lưu ý, một số truy vấn có thể hoạt động không hiệu quả, không phù hợp với xử lý tổng hợp trong bộ nhớ.

Ví dụ: Một API Gateway thường có thể thực hiện vai trò như một API Composition.



Hình 9. Minh họa giao diện tổ hợp dịch vụ (nguồn microservices.com)

2.3.2. Phân tách trách nhiệm truy vấn (CQRS)

Định nghĩa một view database, là một bản sao cơ sở dữ liệu read-only được thiết kế để hỗ trợ cho truy vấn. Cơ sở dữ liệu này sẽ được cạp nhật liên tục bằng cách lắng nghe các sự kiện miền (Domain events) được cung cấp bởi các dịch vụ sở hữu dữ liệu. Điều này đảm bảo view database này luôn được cập nhật và sẵn sàng cho các truy vấn.

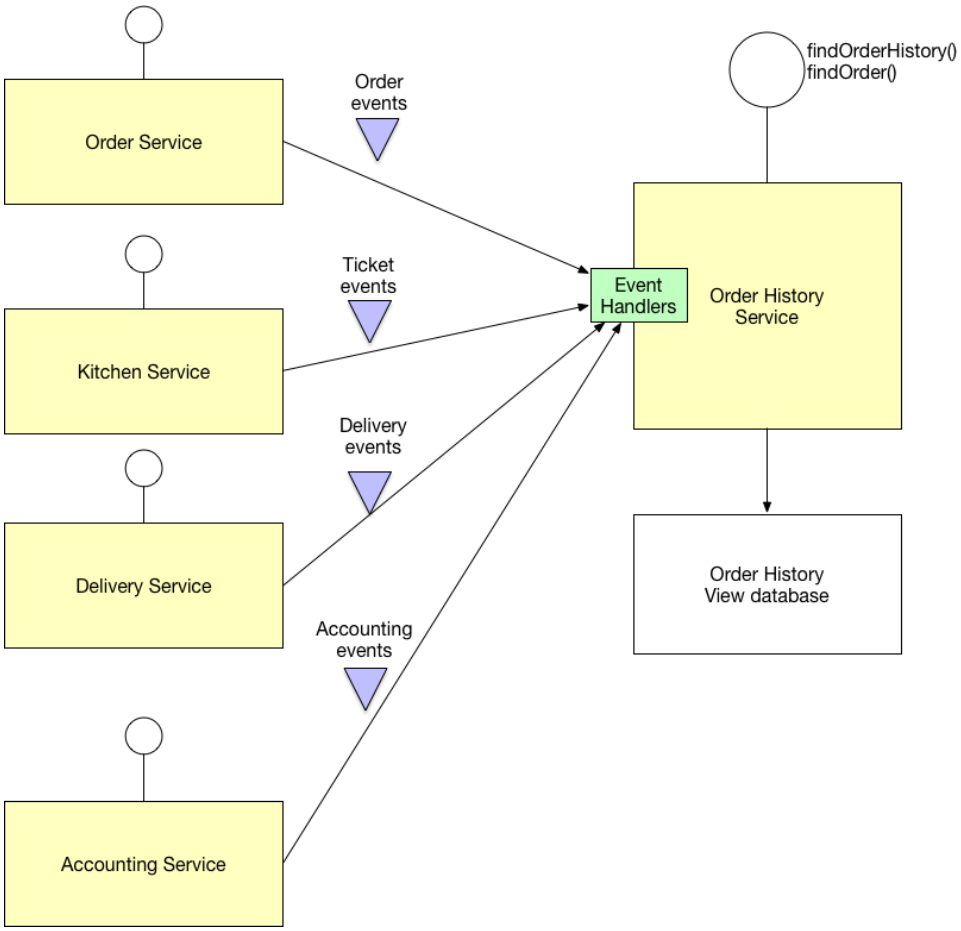
Ưu điểm của các tiếp cận sử dụng CQRS:

- Cho phép tạo ra view tùy chỉnh, tối ưu hóa cho các truy vấn cụ thể
- Cải thiện sự tách biệt trách nhiệm: Mô hình command và query được tách biệt rõ ràng, dễ quản lý hơn
- Cần thiết trong kiến trúc lưu trữ sự kiện

Nhược điểm:

- Tăng độ phức tạp: cần quản lý thêm một cơ sở dữ liệu view và đảm bảo tính nhất quán của nó
- Có thể dẫn tới trùng lặp mã
- Độ trễ sao chép: dữ liệu trong view có thể không nhất quán ngay lập tức với dữ liệu gốc

Ví dụ về dịch vụ Order History triển khai theo mô hình thiết kế CQRS với bản sao cơ sở dữ liệu Order history View riêng biệt.



Hình 10. Minh họa mẫu thiết kế CQRS với dịch vụ Order History (nguồn microservices.com)

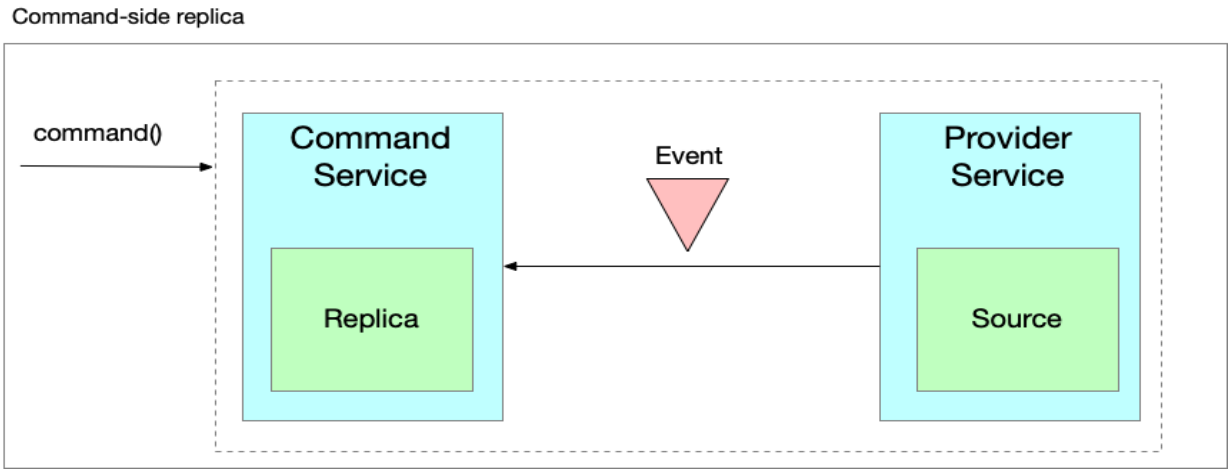
2.3.3. Bản sao phía thực hiện lệnh

Dịch vụ thực hiện lệnh (command service) sẽ tạo ra một bản sao cơ sở dữ liệu chỉ đọc từ dữ liệu của dịch vụ sở hữu dữ liệu (provider service). Command service sẽ duy trì cập nhật bản sao dữ liệu này bằng cách đăng ký nhận các domain events do provider service tạo ra. Giải pháp này gồm các thành phần:

- Command Service: dịch vụ thực hiện các lệnh (ví dụ: createOrder())
- Provider Service : dịch vụ chứa dữ liệu mà Command Service cần (ví dụ: Restaurant Service)

- Replica database: Bản sao cơ sở dữ liệu chỉ đọc của dữ liệu từ Provider Service.

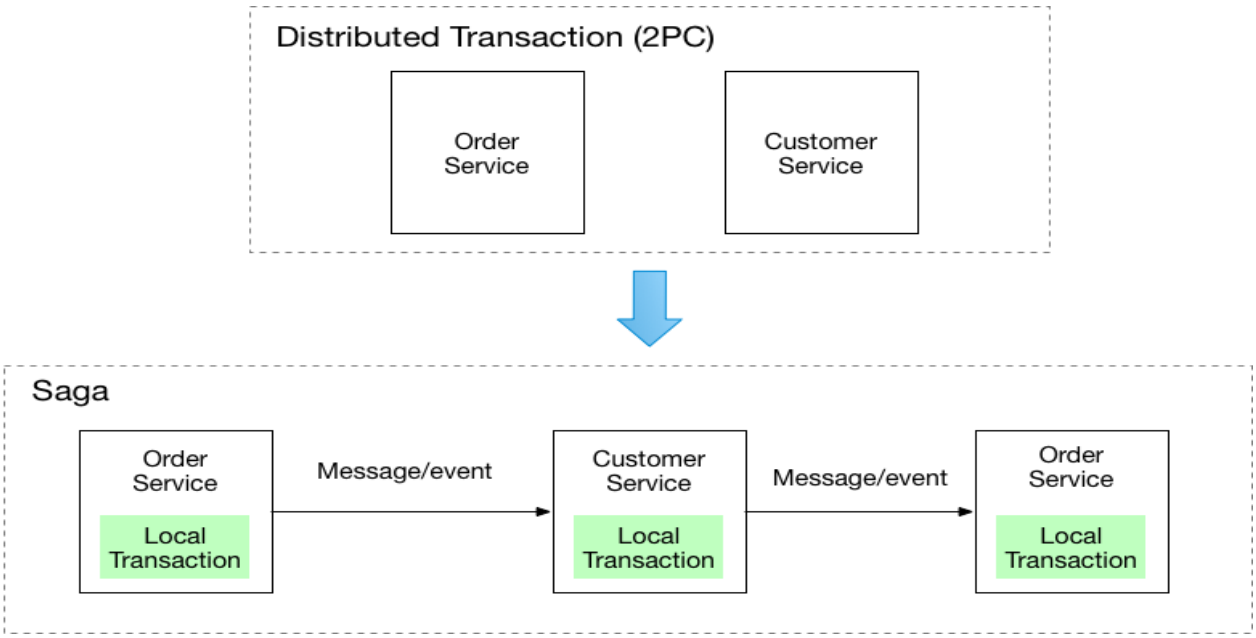
Mô hình thiết kế này có ưu điểm lệnh được thực hiện đơn giản, hiệu quả và giảm thời gian thực thi nhờ tính sẵn có của dữ liệu và không cần tương tác với dịch vụ chứa dữ liệu. Tuy nhiên, nhược điểm là có thể tăng tính phụ thuộc thiết kế giữa các dịch vụ và gây gánh nặng xử lý khi số lượng sự kiện miền tăng.



Hình 11. Minh họa mẫu thiết kế tạo bản sao dữ liệu phía thực hiện (nguồn microservices.com)

2.3.4. Giao dịch dài hạn (Saga)

Triển khai cài đặt mỗi giao dịch kinh doanh liên quan tới nhiều dịch vụ khác nhau như một giao dịch dài hạn (Saga). Một Saga là một chuỗi các giao dịch cục bộ. Mỗi giao dịch cục bộ cập nhật cơ sở dữ liệu riêng và phát hành một thông điệp (message) hoặc một sự kiện (event) để kích hoạt (trigger) giao dịch kế tiếp trong Saga. Nếu có bất kỳ một giao dịch cục bộ nào thất bại vì không đáp ứng được yêu cầu nghiệp vụ thì Saga sẽ thực hiện một chuỗi các giao dịch bù trừ để hủy bỏ các thay đổi mà các giao dịch cục bộ trước đó đã thực hiện.

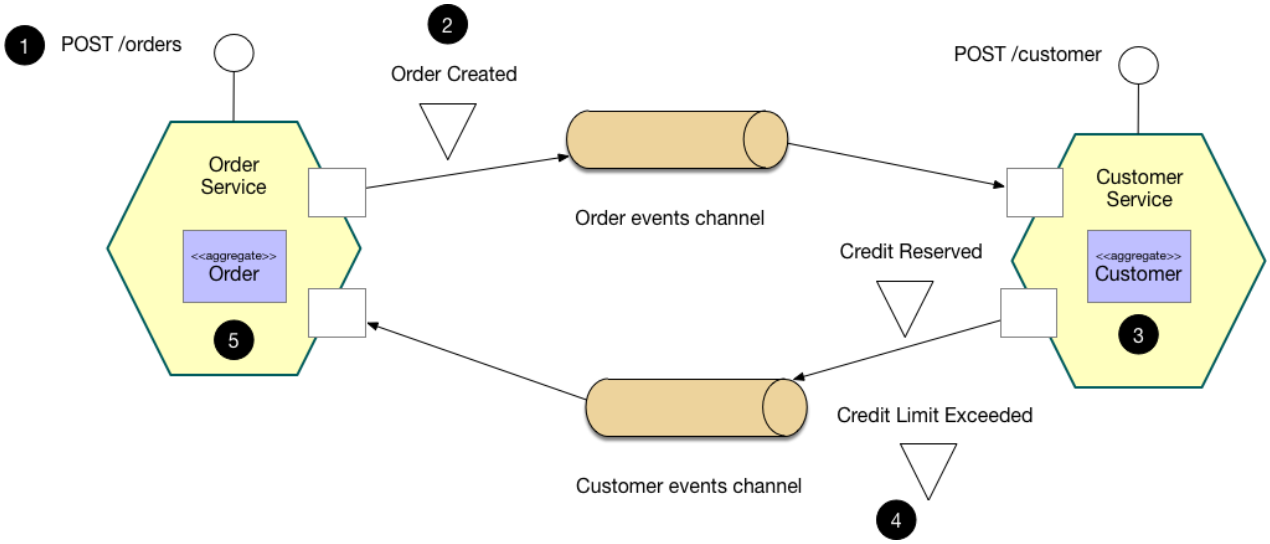


Hình 12. Minh họa mẫu thiết kế giao dịch dài hạn SAGA (nguồn microservices.com)

Có 02 hướng tiếp cận khi triển khai Saga:

- Dựa trên sự kiện (Choreography): Mỗi một giao dịch cục bộ sẽ phát hành một thông điệp hoặc một sự kiện miền để kích hoạt giao dịch cục bộ kế tiếp ở dịch vụ khác.
- Dựa trên thành phần điều phối (Orchestration): Một thành phần điều khối (đối tượng) sẽ chỉ định cho các giao dịch tham gia biết giao dịch cục bộ nào sẽ được thực hiện.

Ví dụ 1: Mô hình Saga dựa trên choreography

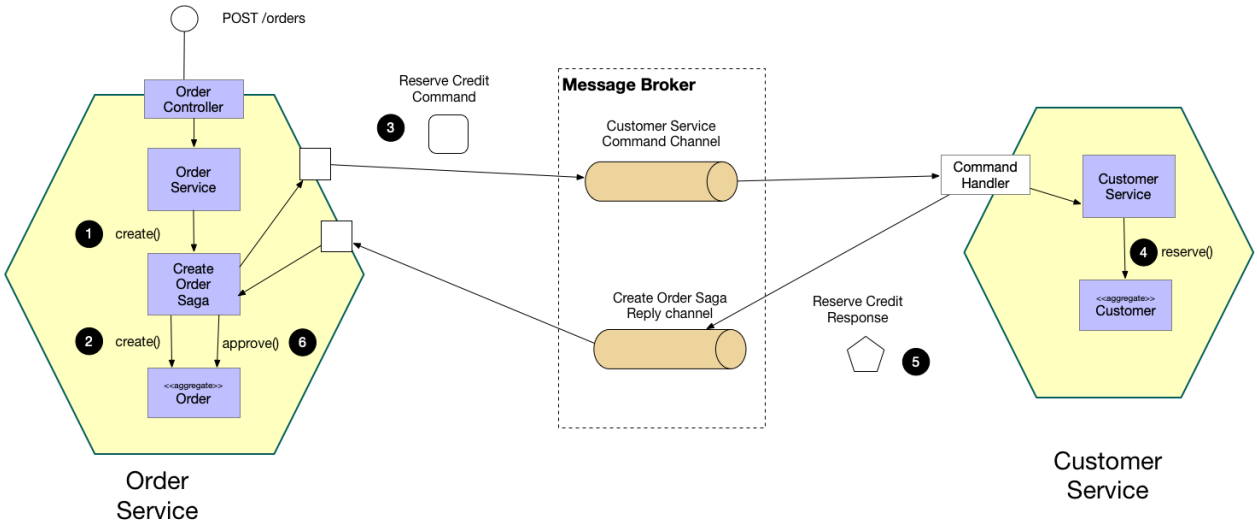


Hình 13. Minh họa mẫu thiết kế Saga dựa trên sự kiện (nguồn microservices.com)

Một ứng dụng thương mại điện tử sử dụng cách tiếp cận này để xử lý giao dịch tạo một hóa đơn sử dụng mô hình Saga dựa trên choreography. Theo đó, các bước xử lý gồm:

1. Dịch vụ Order tiếp nhận yêu cầu xử lý giao dịch tạo hóa đơn qua phương thức POST /orders và tạo một Order với trạng thái PENDING
2. Đẩy/phát đi một sự kiện Order Created
3. Trình xử lý sự kiện của dịch vụ Customer thực hiện kiểm tra tài khoản.
4. Đẩy/phát đi một sự kiện event chỉ ra trạng thái của tài khoản
5. Trình xử lý sự kiện của Order thực hiện việc phê duyệt hoặc từ chối Order

Ví dụ 2: Mô hình Saga dựa trên Orchestration



Hình 14. Minh họa mẫu thiết kế Saga dựa trên thành phần điều phối (nguồn microservices.com)

Một ứng dụng thương mại điện tử sử dụng cách tiếp cận này để xử lý giao dịch tạo một hóa đơn sử dụng mô hình Saga dựa trên Orchestration. Theo đó, các bước xử lý gồm:

1. Dịch vụ Order tiếp nhận yêu cầu xử lý giao dịch tạo hóa đơn qua phương thức POST /orders và gửi một yêu cầu tạo Order Saga
2. Saga orchestrator tạo một Order với trạng thái PENDING
3. Gửi lệnh kiểm tra thông tin tài khoản tới dịch vụ Customer
4. Dịch vụ Customer thực hiện việc kiểm tra tài khoản
5. Dịch vụ Customer gửi lại một thông điệp phản hồi chỉ ra trạng thái của tài khoản
6. Saga orchestrator thực hiện phê duyệt hoặc từ chối Order

2.3.5. Quản lý sự kiện miền (Domain event)

Tổ chức logic nghiệp vụ của một dịch vụ dưới dạng một tập hợp các đối tượng(aggregate) trong thiết kế hướng miền (DDD). Các tập hợp này (có thể coi như một đơn vị) sẽ phát ra các sự kiện miền (domain events) khi chúng được tạo hoặc cập nhật. Sau

đó, dịch vụ sẽ phát hành các sự kiện, để các dịch khác trong hệ thống theo kiến trúc hướng dịch vụ có thể lắng nghe và xử lý chúng. Có thể xem CQRS và SAGA là hai mẫu thiết kế áp dụng quản lý sự kiện theo miền.

2.3.6. Quản lý trạng thái với Event sourcing

Một số ràng buộc:

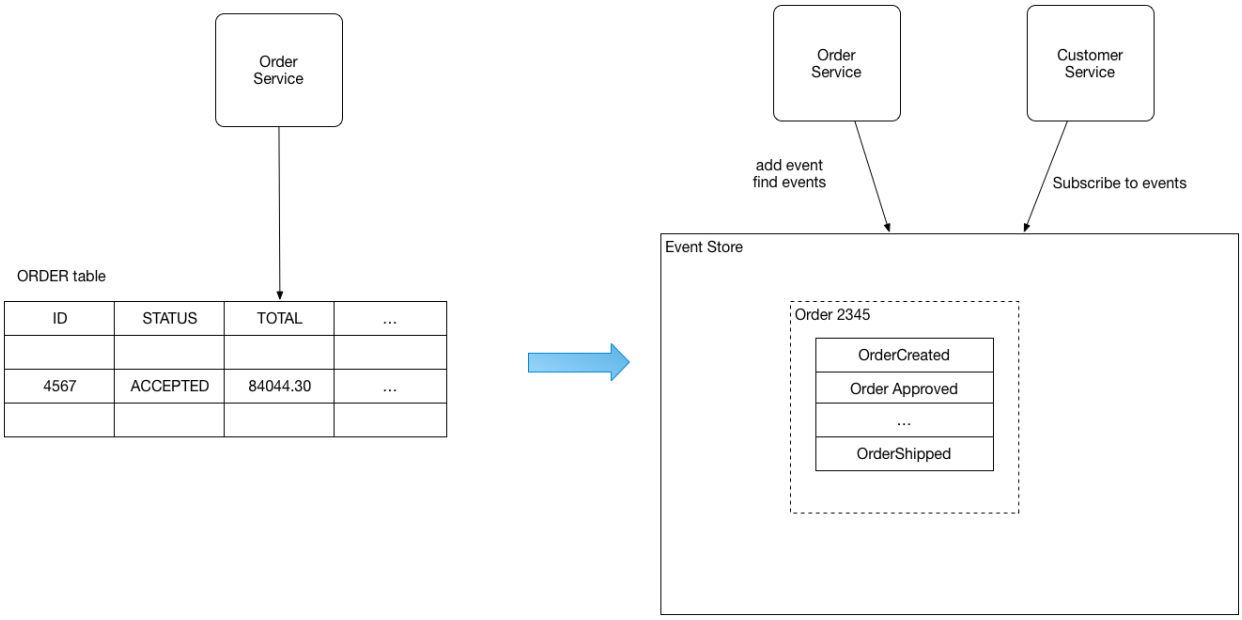
- 2PC không phải là một tùy chọn khả thi. Cơ sở dữ liệu và/hoặc message broker có thể không hỗ trợ 2PC. Ngoài ra, việc gắn kết service với cả cơ sở dữ liệu và message broker thường phát sinh ngoại lệ không mong muốn.
- Nếu giao dịch của cơ sở dữ liệu được commit, thì các message phải được gửi. Ngược lại, nếu giao dịch của cơ sở dữ liệu bị rollback, các message không được phép gửi.
- Messages phải được gửi đến message broker theo thứ tự mà service đã gửi. Thứ tự này phải được bảo toàn khi có nhiều phiên bản service cập nhật cùng một aggregate.

Một giải pháp tốt cho vấn đề này là sử dụng event sourcing. Event sourcing lưu trữ trạng thái của một business entity như Order hoặc Customer dưới dạng một chuỗi các sự kiện thay đổi trạng thái. Mỗi khi trạng thái của một business entity thay đổi, một event mới sẽ được thêm vào danh sách các sự kiện. Vì việc lưu một event là một thao tác đơn lẻ, nên nó vốn dĩ mang tính nguyên tử (atomic). Ứng dụng sẽ tái tạo trạng thái hiện tại của một thực thể bằng cách phát lại các sự kiện.

Các ứng dụng sẽ lưu trữ các event trong một event store, đây là một cơ sở dữ liệu chứa các event. Event store có một API để thêm và truy xuất các event của một thực thể. Ngoài ra, event store cũng hoạt động như một message broker. Nó cung cấp một API cho phép các service đăng ký nhận các sự kiện. Khi một service lưu một event vào event store, nó sẽ được gửi đến tất cả các subscriber quan tâm.

Một số thực thể, như Customer, có thể có số lượng lớn các sự kiện. Để tối ưu hóa việc tải dữ liệu, ứng dụng có thể định kỳ lưu trữ snapshot của trạng thái hiện tại của một thực thể. Để tái tạo trạng thái hiện tại, ứng dụng sẽ tìm snapshot gần đây nhất và các sự kiện xảy ra sau snapshot đó. Kết quả là sẽ có ít sự kiện hơn cần phải phát lại.

Ví dụ: Customers and Orders dưới đây là một ví dụ về một ứng dụng được xây dựng bằng Event Sourcing và CQRS. Sơ đồ dưới đây minh họa cách ứng dụng lưu trữ các orders.



Hình 15. Minh họa ứng dụng sử dụng Event Sourcing và CQRS (nguồn microservices.com)

Thay vì đơn giản lưu trữ trạng thái hiện tại của mỗi order dưới dạng một hàng trong bảng ORDERS, ứng dụng sẽ lưu trữ mỗi Order như một chuỗi các event. CustomerService có thể đăng ký nhận các order events và cập nhật trạng thái của riêng nó.

2.3.7. Shared database

Việc triển khai cơ sở dữ liệu cho các dịch vụ trong kiến trúc hướng dịch vụ phụ thuộc nhiều yếu tố và nên có những cân nhắc cụ thể. Một số yếu tố, gợi ý để triển khai kiến trúc cơ sở dữ liệu trong microservice:

- Các service khác nhau có yêu cầu lưu trữ dữ liệu khác nhau. Với một số service, cơ sở dữ liệu quan hệ (relational database) là lựa chọn tốt nhất. Các service khác có thể cần NoSQL database như MongoDB, phù hợp để lưu trữ dữ liệu phức tạp và không có cấu trúc, hoặc Neo4J, được thiết kế để lưu trữ và truy vấn dữ liệu đồ thị một cách hiệu quả.
- Sử dụng một cơ sở dữ liệu duy nhất được chia sẻ bởi nhiều service. Mỗi service có thể tự do truy cập dữ liệu của các service khác bằng cách sử dụng các giao dịch ACID cục bộ.
- OrderService và CustomerService có thể tự do truy cập vào các bảng của nhau. Ví dụ, OrderService có thể sử dụng giao dịch ACID sau để đảm bảo rằng một đơn hàng mới sẽ không vượt quá hạn mức tín dụng của khách hàng.

Đây gọi là mô hình thiết kế chia sẻ cơ sở dữ liệu (shared database), ưu điểm của mẫu này là:

- Nhà phát triển có thể sử dụng các giao dịch ACID quen thuộc và dễ hiểu để đảm bảo tính nhất quán của dữ liệu.

- Một cơ sở dữ liệu duy nhất đơn giản hơn để vận hành.

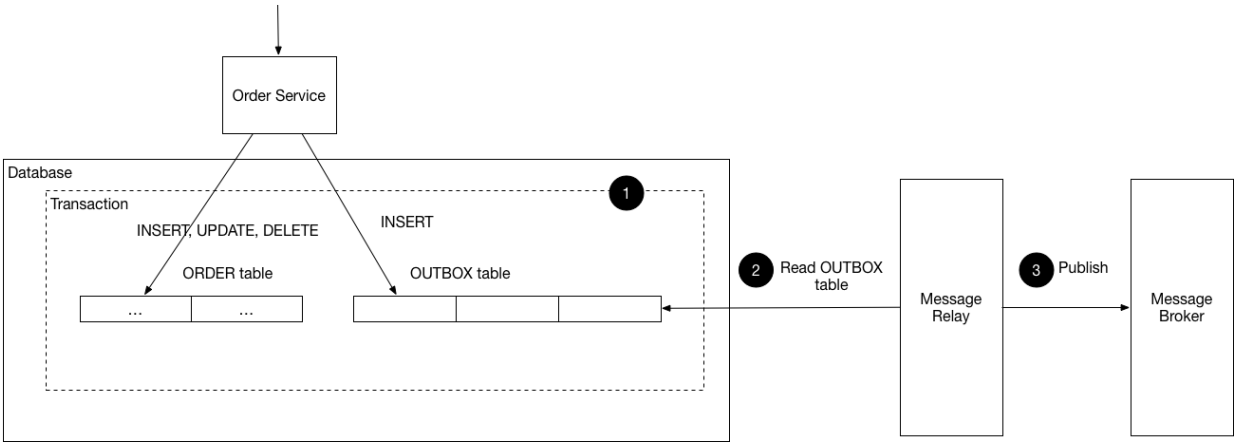
Hạn chế của mẫu thiết kế này:

- Phối hợp trong thời gian phát triển: Nhà phát triển làm việc trên, ví dụ, OrderService, sẽ cần phối hợp với các nhà phát triển của các service khác khi thay đổi cấu trúc cơ sở dữ liệu. Sự phụ thuộc này và việc phối hợp thêm sẽ làm chậm quá trình phát triển.
- Phối hợp trong thời gian chạy: Vì tất cả các service đều truy cập vào cùng một cơ sở dữ liệu, chúng có thể gây cản trở lẫn nhau. Ví dụ, nếu giao dịch dài của CustomerService giữ một khóa trên bảng ORDER, thì OrderService sẽ bị chặn.

2.3.8. Xử lý thông điệp trong giao dịch

Sử dụng mẫu transactional outbox

Giải pháp là service xử lý lệnh trước tiên sẽ lưu trữ message vào cơ sở dữ liệu như một phần của giao dịch đang cập nhật các thực thể nghiệp vụ. Sau đó, một tiến trình riêng biệt sẽ gửi các message này đến message broker.



Hình 16. Minh họa xử lý giao dịch sử dụng mẫu thiết kế transactional outbox (nguồn microservices.com)

Các thành phần tham gia vào mẫu thiết kế này gồm:

- Sender – Dịch vụ (Order) sẽ sinh ra message sau khi thực hiện lệnh với thực thể nghiệp vụ.
- Database – Cơ sở dữ liệu lưu trữ cả thực thể nghiệp vụ lẫn bảng Outbox chứa message tạm thời.
- Message outbox - nếu là cơ sở dữ liệu quan hệ (relational database), đây là một bảng lưu trữ các message cần được gửi. Nếu là NoSQL database, outbox sẽ là một thuộc tính của từng bản ghi trong cơ sở dữ liệu (ví dụ: tài liệu hoặc đối tượng)

- Message relay – Là tiến trình chạy ngầm gửi các message được lưu trữ trong outbox đến message broker

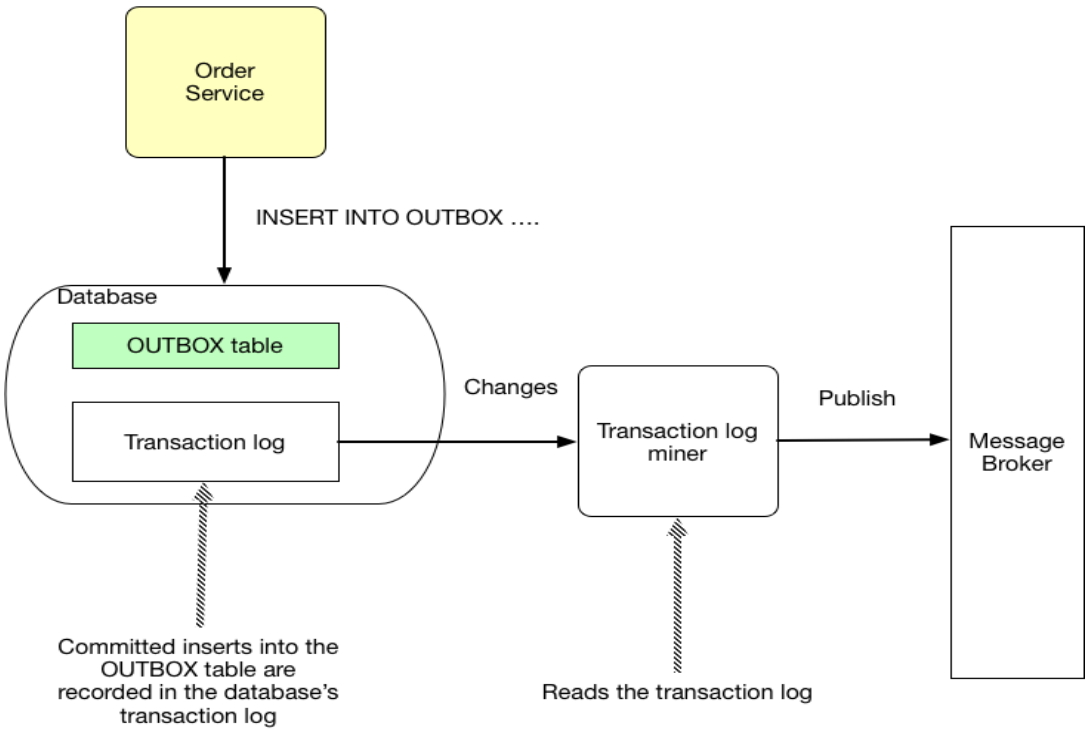
Việc sử dụng mẫu thiết kế này có ưu điểm là:

- Messages được đảm bảo sẽ chỉ được gửi nếu và chỉ khi giao dịch cơ sở dữ liệu được commit.
- Messages được gửi đến message broker theo đúng thứ tự mà ứng dụng đã gửi.

Tuy nhiên, nhược điểm là lỗi có thể dễ phát sinh vì nhà phát triển có thể quên không phát hành message/event sau khi cập nhật cơ sở dữ liệu và tiến trình message relay có thể gửi một message nhiều lần. Ví dụ, nó có thể gặp sự cố sau khi gửi một message nhưng chưa kịp cập nhật rằng nó đã thực hiện điều đó. Khi Message relay khởi động lại, nó sẽ gửi lại message. Do đó, message consumer phải có khả năng xử lý đồng nhất (đảm bảo rằng việc xử lý một message nhiều lần không gây ra kết quả khác biệt), có thể bằng cách theo dõi các ID của message mà nó đã xử lý. May mắn là message consumers thường phải đồng nhất (do message broker có thể gửi message nhiều lần) nên điều này thường không phải là vấn đề lớn.

Sử dụng mẫu Transaction log tailing

Theo dõi (tail) transaction log của cơ sở dữ liệu và phát hành mỗi message/event được chèn vào outbox đến message broker..



Hình 16. Minh họa xử lý giao dịch sử dụng mẫu thiết kế log tailing (nguồn microservices.com)

Việc triển khai cơ chế theo dõi transaction log sẽ phụ thuộc vào từng loại cơ sở dữ liệu:

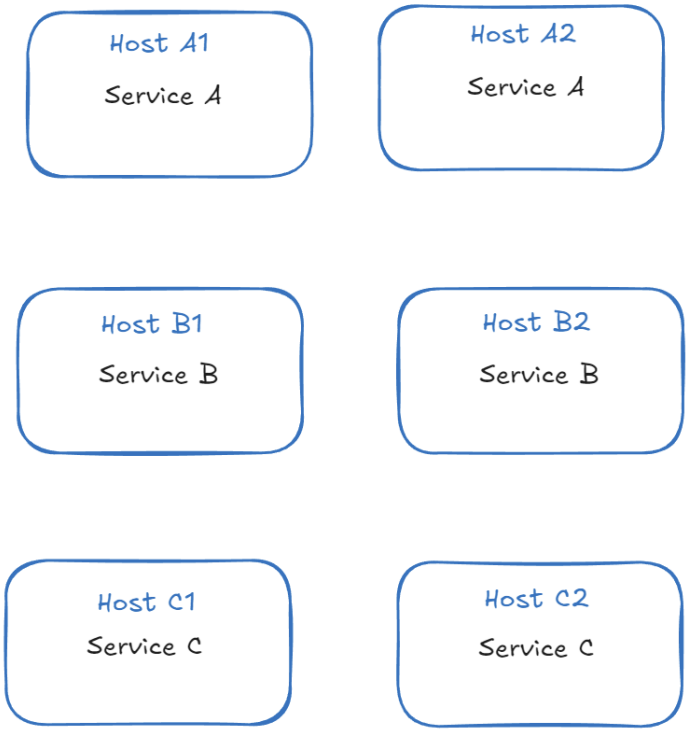
- MySQL binlog
- Postgres WAL
- AWS DynamoDB table streams

Sử dụng mẫu Polling publisher : Phát hành message bằng cách polling bảng outbox trong cơ sở dữ liệu

2.4. Triển khai các dịch vụ của ứng dụng

2.4.1. Một thể hiện dịch vụ trên một host

Mỗi service instance được triển khai trên một host riêng biệt (physical hoặc virtual). Ưu điểm là các thể hiện dịch vụ được triển khai độc lập, không gặp nguy cơ xung đột tài nguyên (CPU, memory) hay phiên bản dịch vụ, giới hạn rõ ràng tài nguyên sử dụng. Nhược điểm là có thể không tận dụng hết tài nguyên của mỗi host khi so sánh với triển khai nhiều dịch vụ trên một host.

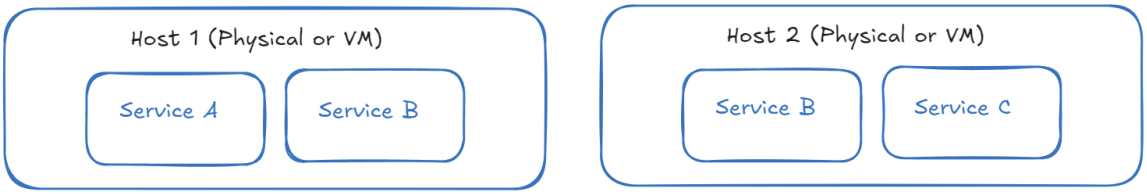


Hình 17. Minh họa triển khai mỗi dịch vụ trên một máy chủ

2.4.2. Nhiều thể hiện dịch vụ trên một host

Chạy nhiều thể hiện của các service khác nhau trên một máy chủ (máy vật lý hoặc máy ảo).

Ưu điểm của cách triển khai này là tối ưu hóa việc sử dụng tài nguyên vì nhiều instance chia sẻ tài nguyên của một host, từ đó giúp giảm chi phí. Nhược điểm là có thể có nguy cơ xung đột tài nguyên, phiên bản giữa các dịch vụ và khó theo dõi, cô lập lỗi khi có vấn đề xảy ra.



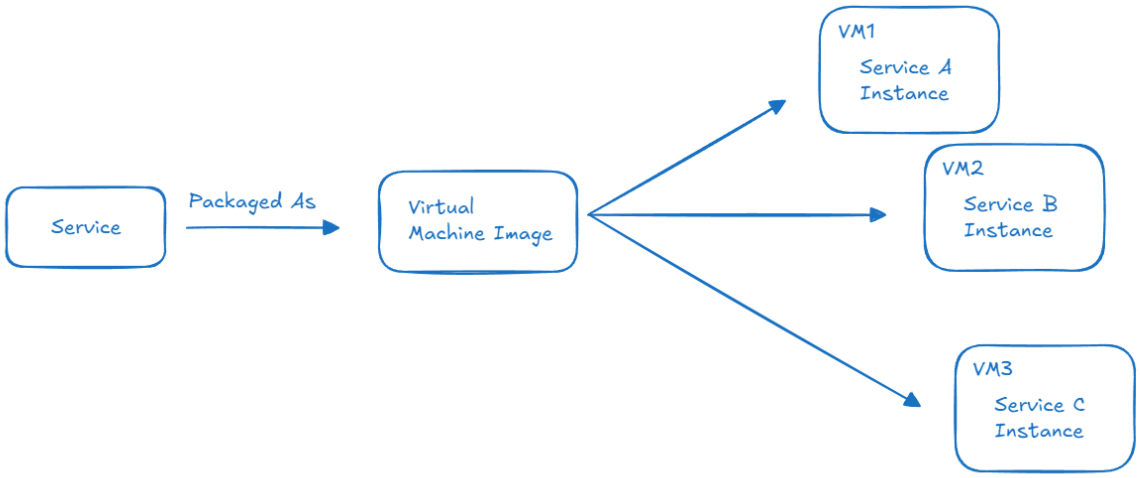
Hình 18. Minh họa triển khai nhiều dịch vụ trên một máy chủ vật lý hoặc máy chủ ảo

2.4.3. Mỗi thể hiện dịch vụ trên một VM

Mỗi service instance được đóng gói dưới dạng một Virtual Machine Image (Ví dụ: Amazon Machine Image - AMI), và mỗi instance của service sẽ được chạy trên một virtual machine riêng biệt. Ưu điểm:

- Dễ dàng mở rộng số lượng service instance bằng cách tạo thêm VM. Ví dụ, với Amazon Autoscaling Groups, số lượng instance có thể tự động tăng lên khi tải cao.
- Mỗi service instance được chạy trên một VM riêng, độc lập hoàn toàn với các instance khác.
- VM có thể đặt giới hạn cụ thể về CPU và memory cho từng service instance.
- Nhất quán trong quản lý, các dịch vụ được quản lý giống nhau, bất kể ngôn ngữ hay framework, vì chúng đều chạy trong môi trường VM.

Nhược điểm của cách triển khai này là việc build một VM image thường chậm và tốn thời gian. Một ví dụ, Netflix triển khai các dịch vụ của họ dưới dạng EC2 AMI và mỗi service instance được triển khai như một EC2 instance trên AWS. Nhờ đó, khi cần tăng tải lên, có thể tăng số lượng service instance thông qua cơ chế autoscaling.



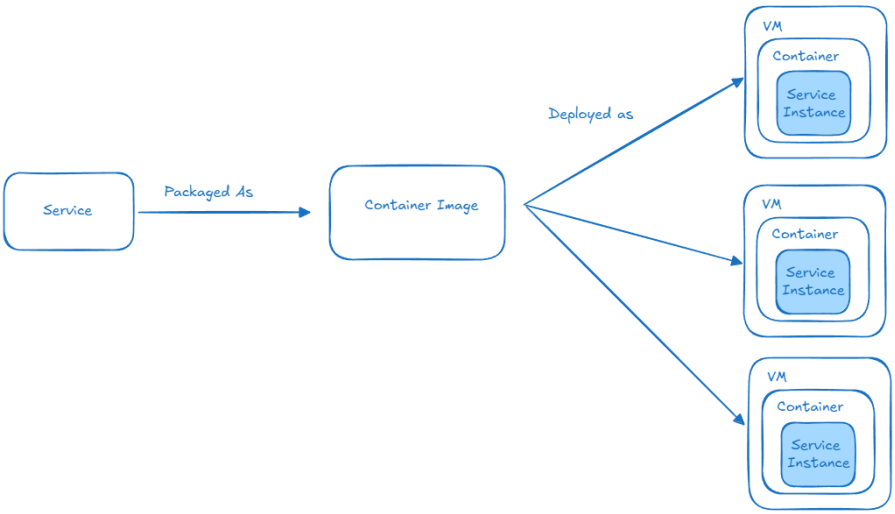
Hình 19. Minh họa triển khai mỗi dịch vụ trên một máy chủ ảo

2.4.4. Một thể hiện dịch vụ trên một container

Docker đang trở thành một cách rất phổ biến để đóng gói và triển khai các dịch vụ. Mỗi service được đóng gói dưới dạng một Docker image và mỗi phiên bản dịch vụ là một Docker container. Có nhiều hệ thống quản lý, giám sát cho Docker bao gồm:

- Kubernetes
- Marathon/Mesos
- Amazon EC2 Container Service

Khi triển khai có thể chạy nhiều container, mỗi container chứa một service instance.



Hình 20. Minh họa triển khai mỗi dịch vụ trên một container

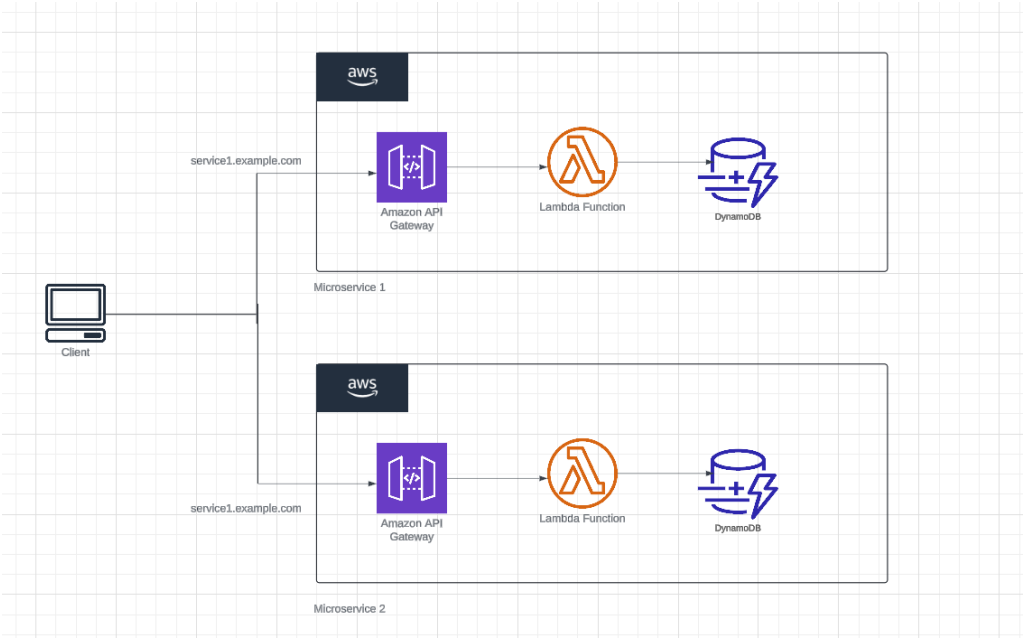
2.4.5. Triển khai không máy chủ

Sử dụng hạ tầng triển khai mà không sử dụng bất kỳ khái niệm nào về máy chủ (nghĩa là tài nguyên được dành riêng hoặc phân bổ trước) - bao gồm cả máy vật lý, máy ảo hoặc containers. Bạn chỉ cần đóng gói, triển khai mã nguồn lên một hạ tầng đám mây kèm theo đặc tả yêu cầu. Hạ tầng này sẽ tự động quản lý việc chạy service, đảm bảo khả năng mở rộng (scalability) và chỉ tính phí dựa trên số lượng yêu cầu và tài nguyên sử dụng (CPU, memory).

Một số môi trường triển khai serverless phổ biến:

- AWS Lambda
- Google Cloud Functions
- Azure Functions

Một ví dụ sử dụng AWS Lambda để triển khai dịch vụ serverless.



Hình 21. Minh họa triển khai dịch vụ với mô hình serverless

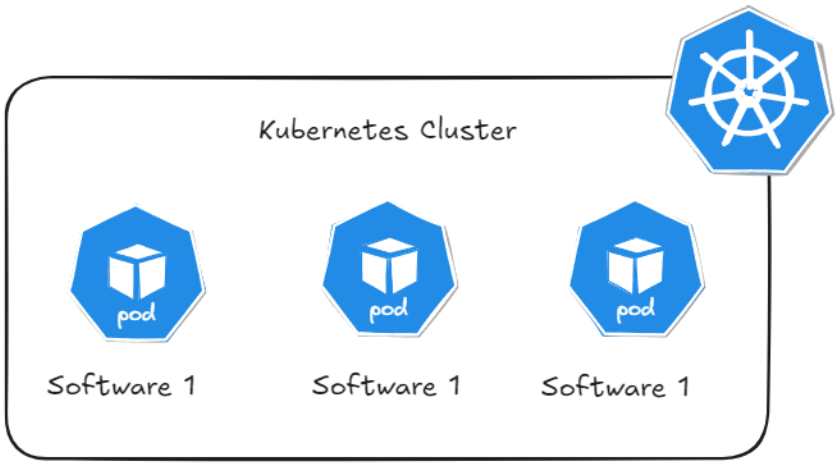
2.4.6. Nền tảng triển khai dịch vụ

Service Deployment Platform là một hạ tầng tự động cho việc triển khai ứng dụng. Nó cung cấp một lớp trừu tượng của dịch vụ, giúp triển khai và quản lý nhiều service instance một cách dễ dàng, đảm bảo tính khả dụng cao (high availability) và phân phối tải

(load balancing). Hạ tầng này tự động quản lý việc khởi chạy, theo dõi, và mở rộng (scaling) các service instance mà không cần can thiệp thủ công.

Một số nền tảng phổ biến:

- Các framework điều phối Docker bao gồm Docker swarm mode và Kubernetes
- Các nền tảng serverless như AWS Lambda
- PaaS bao gồm Cloud Foundry và AWS Elastic Beanstalk



Hình 22. Minh họa triển khai dịch vụ sử dụng một nền tảng triển khai dịch vụ

PHẦN III: PHÂN TÍCH THIẾT KẾ HƯỚNG DỊCH VỤ

3.1. Gửi bảng chấm công

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **TimeSheet Submission Service**: Gửi xác minh số giờ làm việc từ bảng chấm công với hệ thống theo dõi thời gian.

Entity Service:

- **Employee Service**: Quản lý thông tin nhân viên và quyền truy cập.
- **TimeSheet Service**: Quản lý quá trình gửi và theo dõi bảng chấm công.
- **Accounting Service**: Tạo hóa đơn dựa trên giờ làm việc đã xác minh.
- **Customer Service**: Gửi hóa đơn cho khách hàng và xử lý phản hồi.

Micro Service:

- **Time Tracking Service**: Kết nối và lấy dữ liệu thời gian từ hệ thống theo dõi bên thứ ba.

Utility Service:

- **Reporting Service**: Lưu trữ và tạo báo cáo về các bảng chấm công và hóa đơn.

3.2. Đăng ký giải thưởng

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Award Submission Service**: Gửi nộp xác minh chi tiết sự kiện và điều kiện của sinh viên để nhận giải thưởng.

Entity Service:

- **Student Service**: Quản lý thông tin sinh viên và bảng điểm.
- **Event Service**: Quản lý thông tin sự kiện.
- **Award Service**: Quản lý chi tiết về giải thưởng và ghi nhận thông tin trao giải.

Micro Service:

- **Manual Verification Service**: Xác minh từ chối hoặc chấp nhận theo cách thủ công nếu cần xử lý thêm.

Utility Service:

- **Reporting Service**: Ghi lại thông tin trao giải thưởng vào bảng điểm sinh viên và cơ sở dữ liệu giải thưởng.
- **Print Service**: In bản cứng hồ sơ trao giải thưởng.
- **Award Notification Service**: Gửi thông báo từ chối hoặc chấp nhận cho sinh viên sau khi xác minh.

3.3. Đặt phòng khách sạn:

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Booking Service**: Xử lý toàn bộ quá trình đặt phòng, bao gồm nhận thông tin, xác minh phòng trống và cập nhật lịch sử đặt phòng sau khi đặt thành công.

Entity Service:

- **User Service**: Quản lý thông tin cá nhân của người dùng.
- **Hotel Service**: Quản lý thông tin khách sạn và các phòng trong khách sạn.
- **Room Availability Service**: Quản lý trạng thái trống hoặc đã đặt của các phòng khách sạn.

Micro Service:

- **Notification Service**: Gửi thông báo đến người dùng sau khi đặt phòng thành công hoặc không thành công.

Utility Service:

- **History Service**: Lưu trữ và quản lý lịch sử đặt phòng của người dùng.

3.4. Tạo bài thi trắc nghiệm

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Quiz Creation Service**: Xử lý toàn bộ quy trình tạo bài thi trắc nghiệm, từ import danh sách câu hỏi, cấu hình bài thi, danh sách sinh viên, và xác thực các dữ liệu liên quan.

Entity Service:

- **Question Service**: Quản lý danh sách câu hỏi được nhập vào và xử lý xác thực.
- **Quiz Config Service**: Quản lý cấu hình của bài thi trắc nghiệm, bao gồm thời gian, số lượng câu hỏi, điểm số, và các quy định khác.
- **Student Service**: Quản lý danh sách sinh viên tham gia thi, bao gồm việc xác thực tên hoặc mã sinh viên.

Micro Service:

- **Validation Service**: Xác thực tính hợp lệ của các danh sách nhập vào hệ thống (danh sách câu hỏi, cấu hình bài thi, danh sách sinh viên).

Utility Service:

- **Import Service**: Xử lý việc import thủ công các danh sách (danh sách câu hỏi, cấu hình bài thi, danh sách sinh viên) vào hệ thống.

3.5. Tham gia thi trắc nghiệm

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Quiz Participation Service**: Xử lý toàn bộ quy trình tham gia thi, từ xác thực sinh viên, gửi câu hỏi, nhận câu trả lời, so sánh với đáp án, chấm điểm và lưu kết quả.

Entity Service:

- **Student Service**: Quản lý thông tin sinh viên và xác thực danh tính.
- **Question Service**: Quản lý danh sách câu hỏi của bài thi.
- **Answer Service**: Quản lý đáp án đúng và tính điểm cho các câu trả lời của sinh viên.
- **Score Service**: Quản lý tính điểm và lưu kết quả thi của sinh viên.

Micro Service:

- **Notification Service**: Gửi thông báo cho sinh viên trong trường hợp xác thực thất bại hoặc hết hạn nộp bài.

Utility Service:

- **Submission Service**: Xử lý việc nộp bài thi và lưu kết quả vào hệ thống.

3.6. Đồng bộ thời khóa biểu

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Schedule Sync Service**: Xử lý toàn bộ quy trình đồng bộ thời khóa biểu, từ xác minh tài khoản, lấy thời khóa biểu, xác minh quyền truy cập Google Calendar, chuyển đổi và cập nhật lịch trình.

Entity Service:

- **User Service**: Quản lý thông tin tài khoản và xác thực người dùng.
- **Schedule Service**: Quản lý việc truy xuất và xử lý thời khóa biểu từ hệ thống của trường.
- **Google Calendar Service**: Quản lý việc tương tác với Google Calendar, bao gồm kiểm tra quyền truy cập và quản lý sự kiện.

Micro Service:

- **Google Calendar Service**: Quản lý việc tương tác với Google Calendar, bao gồm kiểm tra quyền truy cập và quản lý sự kiện ; Xác định và loại bỏ các sự kiện trùng lặp trên Google Calendar trước khi đồng bộ.

Utility Service:

- **Email Service**: Thông báo cho người dùng tự động thông qua email.

3.7. Phúc khảo điểm thi

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Reassessment Service**: Xử lý toàn bộ quy trình phúc khảo điểm thi, từ khi nhận đơn phúc khảo, kiểm tra tình trạng bài thi, đến lập bảng điểm và công bố kết quả phúc khảo.

Entity Service:

- **Student Service**: Quản lý thông tin sinh viên, kiểm tra thời hạn phúc khảo, và xử lý các yêu cầu phúc khảo.
- **Exam Service**: Quản lý bài thi của sinh viên, bao gồm việc tra cứu và kiểm tra sai sót trong chấm thi.
- **Grade Service**: Quản lý điểm thi, xử lý việc điều chỉnh điểm sau phúc khảo và cập nhật điểm vào cơ sở dữ liệu.

Utility Service:

- **Report Service**: Lập bảng tổng hợp khiếu nại và tạo biên bản phúc khảo để lưu trữ và xử lý.
- **Notification Service**: Gửi thông báo chấp nhận hoặc từ chối đơn phúc khảo, và thông báo kết quả phúc khảo cho sinh viên

3.8. Đặt hàng

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Order Processing Service**: Xử lý toàn bộ quy trình đặt hàng từ khi nhận thông tin đơn hàng, kiểm tra số lượng sản phẩm, cập nhật kho hàng, đến việc gửi email xác nhận đơn hàng cho khách hàng.

Entity Service:

- **Customer Service**: Quản lý thông tin khách hàng và xác minh tài khoản của họ.
- **Product Service**: Quản lý thông tin sản phẩm, bao gồm kiểm tra số lượng hàng tồn kho và cập nhật khi có đơn hàng.
- **Inventory Service**: Quản lý số lượng sản phẩm trong kho và cập nhật sau mỗi đơn đặt hàng.

Micro Service:

- **Email Notification Service**: Gửi email thông báo xác nhận đơn hàng thành công đến khách hàng.

Utility Service:

- **Cart Service**: Xóa sản phẩm khỏi giỏ hàng sau khi đơn đặt hàng được xử lý thành công.

3.9. Đặt vé xem phim

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Ticket Booking Service**: Xử lý toàn bộ quy trình đặt vé từ khi khách hàng chọn phim, ghế ngồi, xác minh thông tin, đến khi thanh toán và gửi thông báo xác nhận.

Entity Service:

- **Movie Service**: Quản lý thông tin về phim, suất chiếu và tình trạng ghế ngồi.
- **Customer Service**: Quản lý thông tin khách hàng, bao gồm việc nhận thông tin cá nhân và xác minh thông tin đặt vé.
- **Payment Service**: Quản lý việc xử lý thanh toán vé xem phim trực tuyến.

Micro Service:

- **Seat Availability Service**: Kiểm tra và xác minh tình trạng ghế ngồi, đảm bảo ghế ngồi đã chọn còn trống trước khi đặt vé.

Utility Service:

- **Notification Service**: Gửi email thông báo xác nhận đặt vé thành công đến khách hàng sau khi thanh toán hoàn tất

3.10. Đăng ký môn học A

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Class Registration Service**: Xử lý toàn bộ quy trình đăng ký môn học, từ khởi tạo yêu cầu đăng ký, xác minh thông tin, chọn môn học, lịch học, phòng học, kiểm tra trùng lặp, đến việc cập nhật lịch học vào cơ sở dữ liệu.

Entity Service:

- **Teacher Service**: Quản lý thông tin giảng viên, bao gồm việc xác minh giảng viên và chọn nhóm lớp tín chỉ.
- **Course Service**: Quản lý thông tin về các môn học và lớp tín chỉ, bao gồm việc kiểm tra tồn tại của môn học và giáo viên.
- **Room Service**: Quản lý thông tin về phòng học và xác nhận phòng học đã được chọn.

Micro Service:

- **Schedule Conflict Resolution Service**: Xử lý việc kiểm tra và xác minh trùng lặp lịch học trước khi lưu vào cơ sở dữ liệu.

Utility Service:

- **Notification Service**: Gửi thông báo cho giảng viên về trạng thái thành công hay thất bại trong việc thêm lịch học.

3.11. Đăng ký môn học B

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Course Registration Service**: Xử lý toàn bộ quy trình đăng ký môn học, từ kiểm tra điều kiện tín chỉ, chọn lớp học phần, kiểm tra trùng lịch học và số lượng sinh viên, đến việc lưu thông tin đăng ký môn học.

Entity Service:

- **Student Service**: Quản lý thông tin sinh viên, bao gồm việc kiểm tra điều kiện tín chỉ và danh sách môn học đã đăng ký.
- **Course Service**: Quản lý thông tin về môn học, lớp học phần và lịch học.
- **Classroom Service**: Quản lý số lượng sinh viên trong lớp học phần và thông tin phòng học.

Micro Service:

- **Schedule Conflict Service**: Kiểm tra và xác minh các trùng lặp về lịch học trước khi lưu đăng ký môn học.

Utility Service:

- **Notification Service**: Gửi thông báo cho sinh viên về kết quả đăng ký môn học, bao gồm việc đạt hoặc không đạt yêu cầu về tín chỉ và lịch học.

3.12. Nhập điểm sinh viên

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Grade Import Service**: Xử lý toàn bộ quy trình nhập điểm, từ việc nhận file điểm, kiểm tra thời gian nhập, xác minh môn học, lớp học phần, và định dạng điểm đến khi lưu thông tin điểm vào cơ sở dữ liệu.

Entity Service:

- **Teacher Service**: Quản lý thông tin giảng viên, xác minh kỳ học, môn học và lớp học phần do giảng viên phụ trách.
- **Student Service**: Quản lý thông tin sinh viên và xác nhận danh sách sinh viên trong file điểm.
- **Course Service**: Quản lý thông tin các môn học và lớp học phần, bao gồm việc kiểm tra quyền giảng dạy của giảng viên.

Micro Service:

- **File Validation Service**: Kiểm tra định dạng file và xác minh tính hợp lệ của dữ liệu điểm từ file nhập.

Utility Service:

- **Notification Service**: Gửi thông báo về việc chấp nhận hoặc từ chối nhập điểm sau khi kiểm tra và xác minh các điều kiện.

3.13. Thanh toán hóa đơn

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Payment Processing Service**: Xử lý toàn bộ quy trình thanh toán, từ việc xác thực thông tin thẻ thanh toán, xử lý giao dịch, đến khi hoàn thành thanh toán hoặc thông báo lỗi.

Entity Service:

- **Order Service**: Quản lý thông tin đơn hàng, bao gồm lưu trữ đơn hàng sau khi thanh toán thành công.
- **Product Service**: Quản lý thông tin sản phẩm và cập nhật số lượng sản phẩm trong kho.
- **Customer Service**: Quản lý thông tin khách hàng và xử lý các thông tin liên quan đến đơn hàng.

- **Inventory Service**: Kiểm tra và cập nhật số lượng hàng hóa trong kho sau khi đơn hàng được xác nhận.
-

Micro Service:

- **Validation Service**: Xác thực thông tin giỏ hàng, hóa đơn, và thẻ thanh toán trước khi xử lý giao dịch.

Utility Service:

- **Notification Service**: Gửi thông báo về tình trạng thanh toán và đơn hàng đến người dùng qua email và giao diện.

3.14. Tạo và giao việc

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Task Creation Service**: Xử lý toàn bộ quy trình tạo và giao công việc, từ việc nhập thông tin, chọn người thực hiện, đến việc lưu thông tin công việc vào cơ sở dữ liệu và gửi thông báo.

Entity Service:

- **Team Service**: Quản lý thông tin về team, bao gồm việc xác thực quyền quản trị viên của người dùng.
- **User Service**: Quản lý thông tin người dùng và hiển thị danh sách thành viên trong team để chọn người thực hiện công việc.
- **Task Service**: Thông tin về công việc được tạo.

Micro Service:

- **Validation Service**: Kiểm tra quyền hạn của người dùng và xác thực thông tin công việc trước khi lưu vào cơ sở dữ liệu.

Utility Service:

- **Notification Service**: Gửi thông báo cho các thành viên được giao công việc sau khi công việc được tạo thành công.

3.15. Tạo sự kiện

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Event Creation Service**: Xử lý toàn bộ quy trình tạo sự kiện, từ việc upload các file Excel, kiểm tra trùng lặp thời gian, gửi thông báo mời tham gia sự kiện đến khách mời, và lưu trữ thông tin sự kiện vào cơ sở dữ liệu.

Entity Service:

- **Event Service**: Quản lý thông tin sự kiện, bao gồm việc kiểm tra thời gian sự kiện và lưu trữ thông tin sự kiện vào cơ sở dữ liệu.
- **Schedule Service**: Quản lý và lưu trữ lịch trình sự kiện.
- **Guest Service**: Quản lý danh sách khách mời và lưu trữ thông tin khách mời vào cơ sở dữ liệu.

Micro Service:

- **File Upload Service**: Xử lý việc tải lên các file Excel chứa thông tin sự kiện, lịch trình và danh sách khách mời.

Utility Service:

- **Notification Service**: Gửi email thông báo đến khách mời tham gia sự kiện sau khi sự kiện được tạo thành công.

3.16. Điểm danh tự động

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Attendance Service**: Xử lý toàn bộ quy trình điểm danh, từ kiểm tra mã phòng, so sánh thời gian tham gia, đến việc lưu trữ và thông báo kết quả điểm danh.

Entity Service:

- **Schedule Service**: Quản lý thông tin lịch học, bao gồm việc lấy thông tin mã phòng và danh sách sinh viên.
- **Student Service**: Quản lý thông tin sinh viên và lấy dữ liệu lịch sử tham gia Zoom của sinh viên.

Micro Service:

- **Zoom Integration Service**: Tích hợp với Zoom để lấy thông tin thời gian ra vào phòng thực tế của sinh viên.
- **Notification Service**: Gửi thông báo kết quả điểm danh đến sinh viên và người dùng yêu cầu điểm danh.

Utility Service:

- **Validation Service**: Xác minh tính hợp lệ của mã phòng Zoom và so sánh thời gian tham gia với lịch học.

3.17. Chấm điểm rèn luyện

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Student Activity Evaluation Service**: Xử lý toàn bộ quy trình nộp bằng chứng, từ khi sinh viên gửi bằng chứng, xác minh tính hợp lệ, đến việc lưu trữ và gửi thông báo kết quả.

Entity Service:

- **Evaluation Service**: Quản lý thông tin về các tiêu chí đánh giá và kiểm tra xem tiêu chí có hợp lệ hay không.
- **Student Service**: Quản lý thông tin sinh viên và các hoạt động rèn luyện đã được ghi nhận.
- **Evaluation Period Service**: Xác minh thời hạn nộp minh chứng trong đợt và thông báo nếu thời hạn đã hết.

Micro Service:

- **Document Validation Service**: Xử lý việc kiểm tra và xác minh các minh chứng nộp từ sinh viên theo tiêu chí đánh giá.

Utility Service:

- **Notification Service**: Gửi thông báo cho sinh viên về kết quả đánh giá (chấp nhận hoặc từ chối).

3.18. So sánh giá sản phẩm

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Product Price Comparison Service**: Xử lý toàn bộ quy trình từ tìm kiếm sản phẩm, so sánh giá giữa các nhà cung cấp, đến điều hướng người dùng đến trang bán hàng của nhà cung cấp phù hợp.

Entity Service:

- **Product Service**: Quản lý thông tin sản phẩm, bao gồm tìm kiếm sản phẩm, lấy thông tin chi tiết và danh sách các sản phẩm liên quan.
- **Supplier Service**: Quản lý danh sách nhà cung cấp, bao gồm thông tin giá cả và các sản phẩm họ cung cấp.
- **Source Service:** Cung cấp danh sách các nguồn dữ liệu để thu thập dựa trên sản phẩm mà người dùng đã chọn.

Micro Service:

- **Crawler Service**: Thực hiện so sánh giá giữa các nhà cung cấp khác nhau và trả về danh sách so sánh giá chi tiết cho người dùng.

3.19. Ghi chỉ số điện/nước

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Change Meter Service**: Xử lý toàn bộ quy trình cập nhật chỉ số nước, từ khi người dùng tải ảnh công tơ, xác minh thông tin QR, tạo hóa đơn, đến việc thanh toán và cập nhật chỉ số.

Entity Service:

- **User Service**: Quản lý thông tin người dùng, bao gồm việc lấy thông tin từ mã QR và quản lý tài khoản thanh toán.
- **Invoice Service**: Quản lý việc tạo hóa đơn và trạng thái thanh toán của hóa đơn.
- **Meter Service**: Quản lý và cập nhật chỉ số của công tơ nước hoặc điện.

Micro Service:

- **Image Process Service** : Sử dụng xử lý ảnh để phân tích ảnh công tơ và lấy chỉ số nước.
- **Banking Service**: Xử lý thanh toán hóa đơn thông qua tài khoản ngân hàng của người dùng.
- **Validation Service**: Xác minh thông tin chỉ số nước, trạng thái hóa đơn, và thông tin tài khoản trước khi tiến hành cập nhật hoặc thanh toán.

Utility Service:

- **QR Service**: Đọc thông tin từ mã QR trên công tơ nước để xác định thông tin người dùng.

3.20. Xuất dữ liệu điểm danh

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Attendance Service**: Xử lý toàn bộ quy trình điểm danh, từ việc nhận thông tin sinh viên, so sánh với dữ liệu từ Zoom/Meet, đến việc xuất file và lưu trữ thông tin.

Entity Service:

- **Student Service**: Quản lý thông tin sinh viên, bao gồm mã sinh viên, tên sinh viên và trạng thái điểm danh.
- **Meeting Service**: Quản lý thông tin của buổi học, bao gồm ID Meet/Zoom và dữ liệu thời gian ra vào của sinh viên.

Micro Service:

- **Zoom/Meet Integration Service**: Tích hợp với Zoom/Meet để lấy dữ liệu thời gian vào/ra của sinh viên và so sánh với danh sách điểm danh.

Utility Service:

- **File Export Service**: Hỗ trợ xuất file CSV chứa danh sách sinh viên đã điểm danh.

3.21. Làm bài thi

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Exam Service**: Xử lý toàn bộ quy trình làm bài thi, từ việc nhận và xác minh thông tin sinh viên, mã đề thi, đến việc lấy và hiển thị danh sách câu hỏi.

Entity Service:

- **Student Service**: Quản lý thông tin sinh viên, bao gồm mã sinh viên, lớp học và trạng thái thi.

- **ExamPaper Service**: Quản lý thông tin về bài thi, bao gồm mã đề thi, trạng thái thi và danh sách câu hỏi.
- **Question Service**: Lấy và hiển thị danh sách câu hỏi của bài thi dựa trên mã đề thi.

Micro Service:

- **Authentication Service**: Xác minh thông tin sinh viên và kiểm tra tính hợp lệ của sinh viên trước khi bắt đầu bài thi.

3.22. Phê duyệt yêu cầu nghỉ phép

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Leave Request Service**: Xử lý toàn bộ quy trình nộp và phê duyệt yêu cầu nghỉ phép, từ khi nhân viên nộp đơn đến khi quản lý phê duyệt hoặc từ chối.

Entity Service:

- **Employee Service**: Quản lý thông tin chi tiết về nhân viên, bao gồm lịch sử nghỉ phép và tình trạng nghỉ phép hiện tại.
- **Manager Service**: Quản lý thông tin của quản lý, bao gồm việc nhận thông báo phê duyệt yêu cầu nghỉ phép.
- **Approval Service**: Hỗ trợ quản lý phê duyệt hoặc từ chối yêu cầu nghỉ phép và cập nhật trạng thái.
- **Leave Service**: Xác minh số ngày nghỉ còn lại của nhân viên và tính hợp lệ của yêu cầu trước khi gửi đến quản lý.

Utility Service:

- **Notification Service**: Gửi thông báo đến nhân viên và quản lý về tình trạng yêu cầu nghỉ phép.

3.23. Phê duyệt yêu cầu thanh toán công tác phí

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Expense Approval Service**: Xử lý toàn bộ quy trình phê duyệt yêu cầu thanh toán công tác phí, từ khi nhân viên nộp đơn đến khi thông tin thanh toán được chuyển đến bộ phận tài chính.

Entity Service:

- **Employee Service**: Quản lý thông tin nhân viên, bao gồm các thông tin về chuyến công tác và chi phí liên quan.
- **Manager Service**: Quản lý thông tin của quản lý và hỗ trợ quy trình phê duyệt yêu cầu thanh toán.
- **Invoice Service**: Xử lý thông tin liên quan đến việc thanh toán của bộ phận tài chính, bao gồm việc thanh toán qua ngân hàng.
- **Payment Service**: Hỗ trợ chuyển thông tin thanh toán đến bộ phận tài chính và xử lý thanh toán thông qua ngân hàng.

Micro Service:

- **Validation Service**: Kiểm tra và xác minh tính hợp lệ của các hóa đơn, biên lai chi phí được nộp kèm theo yêu cầu.

Utility Service:

- **Notification Service**: Gửi thông báo đến nhân viên và quản lý về trạng thái phê duyệt hoặc từ chối yêu cầu thanh toán công tác phí.

3.24. Xử lý đơn hàng trực tuyến

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Order Processing Service**: Xử lý toàn bộ quy trình từ khi khách hàng đặt hàng, kiểm tra hàng tồn kho, xử lý thanh toán, và giao hàng.

Entity Service:

- **Product Service**: Quản lý thông tin về sản phẩm và tình trạng hàng tồn kho.
- **Customer Service**: Quản lý thông tin khách hàng, bao gồm thông tin giao hàng và trạng thái đơn hàng.
- **Order Service**: Quản lý thông tin đơn hàng, bao gồm các sản phẩm trong đơn hàng và trạng thái thanh toán.
- **Shipping Service**: Quản lý việc lên lịch và giao hàng thông qua các đơn vị vận chuyển.
- **Inventory Service**: Kiểm tra và xác minh tình trạng hàng tồn kho trước khi xác nhận đơn hàng.

Micro Service:

- **Payment Service**: Xử lý thanh toán thông qua các cổng thanh toán trực tuyến như thẻ tín dụng, ví điện tử hoặc chuyển khoản ngân hàng.
- **Notification Service**: Gửi thông báo cho khách hàng về trạng thái đơn hàng, bao gồm thông báo xác nhận, giao hàng và phản hồi từ khách hàng.

3.25. Xử lý khiếu nại khách hàng

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Complaint Handling Service**: Xử lý toàn bộ quy trình từ tiếp nhận khiếu nại, phân công nhân viên, theo dõi tiến độ, đến khi hoàn tất xử lý và thông báo kết quả cho khách hàng.

Entity Service:

- **Customer Service**: Quản lý thông tin chi tiết về khách hàng, bao gồm các khiếu nại mà họ đã gửi và trạng thái hiện tại của các khiếu nại.
- **Employee Service**: Quản lý thông tin nhân viên phụ trách, phân công nhân viên phù hợp dựa trên kỹ năng và loại khiếu nại.
- **Complaint Service**: Quản lý thông tin khiếu nại theo loại và mức độ ưu tiên để phân công nhân viên phù hợp.

Micro Service:

- **Progress Tracking Service**: Theo dõi và quản lý tiến độ xử lý khiếu nại, đảm bảo việc giải quyết diễn ra đúng hạn.
- **Survey Service**: Gửi khảo sát cho khách hàng sau khi khiếu nại được giải quyết để thu thập đánh giá về mức độ hài lòng.

Utility Service:

- **Notification Service**: Gửi thông báo qua email hoặc ứng dụng cho khách hàng về trạng thái khiếu nại, yêu cầu bổ sung thông tin, và kết quả giải quyết.
- **Document Management Service**: Quản lý và lưu trữ các tài liệu liên quan đến khiếu nại (như hóa đơn, hình ảnh, hoặc video) do khách hàng cung cấp.

3.26. Đăng ký thuê bao

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Subscription Registration Service**: Xử lý toàn bộ quy trình đăng ký dịch vụ thuê bao, từ lúc khách hàng chọn gói dịch vụ, xác minh thông tin, đến khi kích hoạt và cung cấp dịch vụ cho khách hàng.

Entity Service:

- **Customer Service**: Quản lý thông tin cá nhân và tình trạng đăng ký của khách hàng.
- **Subscription Plan Service**: Quản lý các gói dịch vụ thuê bao, bao gồm các chi tiết về cước phí, lợi ích và điều kiện của từng gói dịch vụ.
- **Contract Service**: Quản lý hợp đồng thuê bao, bao gồm việc tạo hợp đồng và lưu trữ thông tin hợp đồng của khách hàng.

Micro Service:

- **Verification Service**: Xác minh thông tin cá nhân và thanh toán của khách hàng để đảm bảo đăng ký hợp lệ.
- **Activation Service**: Gửi yêu cầu kích hoạt dịch vụ đến hệ thống liên quan và theo dõi quá trình kích hoạt.
- **Billing Service**: Xử lý việc tạo hóa đơn và gửi thông tin thanh toán đến khách hàng.

Utility Service:

- **Notification Service**: Gửi thông báo qua email hoặc ứng dụng cho khách hàng về trạng thái đăng ký dịch vụ, kích hoạt dịch vụ và hóa đơn thanh toán.
- **Service Availability Service**: Kiểm tra tính khả dụng của dịch vụ tại khu vực của khách hàng.

3.27. Đăng ký số điện thoại sử dụng dịch vụ

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Phone Registration Service**: Xử lý toàn bộ quy trình đăng ký số điện thoại và xác minh danh tính, bao gồm việc gửi mã OTP, xử lý tài liệu danh tính và kích hoạt tài khoản.

Entity Service:

- **User Identity Service**: Quản lý thông tin danh tính cá nhân của người dùng, bao gồm các tài liệu xác minh danh tính.
- **Phone Service**: Quản lý thông tin liên quan đến số điện thoại, bao gồm tình trạng đăng ký và xác minh.

- **Account Service**: Quản lý việc tài khoản hoặc dịch vụ liên quan đến số điện thoại đã xác minh thành công.

Micro Service:

- **OTP Service**: Gửi mã OTP qua SMS để xác minh số điện thoại và xử lý việc xác thực mã OTP.
- **Document Verification Service**: Xác minh tài liệu chứng minh danh tính của người dùng bằng cách sử dụng công nghệ OCR và kiểm tra tính hợp lệ.
- **Fraud Detection Service**: Xử lý việc kiểm tra chống gian lận và xác thực xem danh tính và số điện thoại có dấu hiệu bị lạm dụng hay không.

Utility Service:

- **Notification Service**: Gửi thông báo cho người dùng về trạng thái xác minh và kích hoạt tài khoản qua SMS hoặc email.

3.28. Đặt lịch hẹn trực tuyến

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Appointment Booking Service**: Xử lý toàn bộ quy trình đặt lịch hẹn, từ việc chọn dịch vụ, kiểm tra tình trạng lịch hẹn, xác nhận lịch hẹn đến quản lý thay đổi và hủy lịch hẹn.

Entity Service:

- **Customer Service**: Quản lý thông tin khách hàng, bao gồm chi tiết về lịch hẹn, trạng thái và lịch sử các lần đặt lịch.
- **Employee Service**: Quản lý thông tin nhân viên hoặc chuyên viên cung cấp dịch vụ, bao gồm lịch trình làm việc và tình trạng khả dụng.
- **Service Catalog Service**: Quản lý danh mục các dịch vụ có sẵn, bao gồm các chi tiết về loại hình dịch vụ, chi phí và thời gian thực hiện.
- **Scheduling Service**: Quản lý lịch trình của khách hàng và nhân viên, đảm bảo việc kiểm tra và quản lý tình trạng trống của lịch hẹn.

Micro Service:

- **Billing Service**: Gửi hóa đơn hoặc thông tin thanh toán cho khách hàng nếu có chi phí liên quan đến dịch vụ đã đặt.

Utility Service:

- **Feedback Service**: Thu thập phản hồi từ khách hàng sau khi dịch vụ đã được hoàn thành.
- **Notification Service**: Gửi thông báo xác nhận lịch hẹn, lời nhắc và thông tin thay đổi hoặc hủy lịch hẹn qua email hoặc SMS cho khách hàng và nhân viên.

3.29. Yêu cầu bảo hành sản phẩm

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Warranty Processing Service**: Xử lý toàn bộ quy trình từ khi khách hàng gửi yêu cầu bảo hành, kiểm tra điều kiện bảo hành, đến khi sản phẩm được sửa chữa hoặc thay thế.

Entity Service:

- **Product Service**: Quản lý thông tin về sản phẩm, bao gồm số serial, thời hạn bảo hành, và các điều kiện bảo hành.
- **Customer Service**: Quản lý thông tin cá nhân của khách hàng, bao gồm lịch sử mua hàng và yêu cầu bảo hành.
- **Technician Service**: Quản lý thông tin về các kỹ thuật viên phụ trách kiểm tra và sửa chữa sản phẩm.
- **Survey Service**: Thu thập phản hồi từ khách hàng về chất lượng dịch vụ bảo hành.

Micro Service:

- **Condition Verification Service**: Kiểm tra và xác minh các điều kiện bảo hành của sản phẩm dựa trên thông tin và hình ảnh do khách hàng cung cấp.
- **Repair Tracking Service**: Theo dõi tiến độ sửa chữa sản phẩm, cập nhật trạng thái và gửi thông báo về quá trình bảo hành cho khách hàng.

Utility Service:

- **Notification Service**: Gửi thông báo qua email hoặc SMS cho khách hàng về trạng thái yêu cầu bảo hành, quá trình sửa chữa, và hoàn tất bảo hành.
- **Logistics Service**: Quản lý quá trình vận chuyển sản phẩm từ khách hàng đến trung tâm bảo hành và ngược lại.

3.30. Gửi nhận tiền trực tuyến

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Money Transfer Service**: Xử lý toàn bộ quy trình gửi và nhận tiền, từ khi người gửi nhập thông tin giao dịch, xác minh tài khoản, đến khi xử lý giao dịch và thông báo cho các bên liên quan.

Entity Service:

- **Account Service**: Quản lý thông tin tài khoản người dùng, bao gồm số dư tài khoản và trạng thái giao dịch.
- **Transaction Service**: Quản lý lịch sử giao dịch của người dùng, bao gồm các giao dịch gửi tiền và nhận tiền.

Micro Service:

- **OTP Service**: Gửi mã OTP để xác nhận giao dịch và đảm bảo tính bảo mật của giao dịch chuyển tiền.
- **Fraud Detection Service**: Kiểm tra tính an toàn của giao dịch và phát hiện các dấu hiệu gian lận.
- **Audit Logging Service**: Lưu trữ và quản lý các bản ghi giao dịch cho mục đích kiểm toán và an ninh.

Utility Service:

- **Notification Service**: Gửi thông báo cho người gửi và người nhận về trạng thái giao dịch qua email, SMS hoặc thông báo đẩy.

3.31. Xử lý yêu cầu bảo hiểm khách hàng

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Insurance Claim Processing Service**: Xử lý toàn bộ quy trình yêu cầu bảo hiểm từ khi khởi tạo yêu cầu, xác minh thông tin, đến tính toán bồi thường và cập nhật hồ sơ bảo hiểm.

Entity Service:

- **Customer Service**: Quản lý thông tin khách hàng, bao gồm thông tin cá nhân, lịch sử yêu cầu bảo hiểm và các hợp đồng bảo hiểm.
- **Policy Service**: Quản lý các hợp đồng bảo hiểm của khách hàng, bao gồm các điều khoản và điều kiện bồi thường.

Micro Service:

- **Claim Verification Service**: Xác minh tính hợp lệ của yêu cầu bảo hiểm và kiểm tra các điều kiện bảo hiểm.
- **Compensation Calculation Service**: Tính toán số tiền bồi thường dựa trên quy định của hợp đồng bảo hiểm.

Utility Service:

- **Document Generation Service**: Tạo và in bản cứng của yêu cầu bảo hiểm để lưu trữ và phục vụ kiểm tra.
- **Notification Service**: Gửi thông báo chấp nhận hoặc từ chối yêu cầu bảo hiểm cho khách hàng.

3.32. Vay tiền trực tuyến

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Loan Application Service**: Xử lý toàn bộ quy trình đăng ký vay, từ việc khởi tạo yêu cầu, xác minh thông tin cá nhân và tài chính, đến việc phê duyệt và giải ngân khoản vay.

Entity Service:

- **Customer Service**: Quản lý thông tin khách hàng, bao gồm thông tin cá nhân, lịch sử tài chính, và trạng thái tín dụng.
- **Credit Score Service**: Quản lý và truy xuất điểm tín dụng của khách hàng từ các dịch vụ bên ngoài để đánh giá khả năng thanh toán.
- **Loan Contract Service**: Quản lý các hợp đồng vay, bao gồm thông tin điều khoản, lãi suất, và tình trạng hợp đồng.
- **Payment Scheduling Service**: Lên lịch các đợt thanh toán và gửi nhắc nhở cho khách hàng.

Micro Service:

- **Identity Verification Service**: Xác minh danh tính và thông tin cá nhân của khách hàng thông qua các hệ thống bên thứ ba.
- **Risk Assessment Service**: Đánh giá rủi ro dựa trên lịch sử thanh toán và điểm tín dụng của khách hàng.

Utility Service:

- **Notification Service**: Gửi thông báo cho khách hàng qua email hoặc SMS về tình trạng giải ngân, nhắc nhở thanh toán, và hoàn tất khoản vay.
- **Document Storage Service**: Lưu trữ hợp đồng vay, lịch sử thanh toán, và các hồ sơ liên quan để tra cứu và kiểm tra sau này.

3.33. Đồng bộ dữ liệu phân tán

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Data Synchronization Service**: Xử lý toàn bộ quy trình đồng bộ hóa dữ liệu phân tán, từ lúc khởi tạo ghi dữ liệu đến quá trình đồng bộ, xử lý xung đột, và đảm bảo nhất quán cuối.

Entity Service:

- **Version Management Service**: Quản lý thông tin phiên bản của dữ liệu tại từng cụm, bao gồm lịch sử phiên bản và các thông tin thời gian thực hiện.
- **Conflict Resolution Service**: Xử lý các xung đột dữ liệu khi phát hiện các phiên bản khác nhau và chọn phương án đồng bộ tối ưu.
- **Audit Logging Service**: Lưu trữ chi tiết về các phiên bản dữ liệu và lịch sử đồng bộ để phục vụ cho việc kiểm tra, khôi phục, và giám sát chất lượng dữ liệu.

Micro Service:

- **Error Handling Service**: Phát hiện lỗi trong quá trình đồng bộ dữ liệu và tự động khởi động quy trình khôi phục dữ liệu.
- **Latency Monitoring Service**: Theo dõi và ghi nhận độ trễ của quá trình đồng bộ hóa tại các cụm để tối ưu hóa hiệu năng.

Utility Service:

- **Notification Service**: Gửi thông báo đến các dịch vụ phụ thuộc sau khi hoàn tất quy trình đồng bộ hoặc khi có lỗi xảy ra.

3.34. Cảnh báo giám sát thời gian thực

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Real-Time Surveillance Analysis Service**: Xử lý toàn bộ quy trình phân tích thời gian thực, bao gồm thu thập và phân tích dữ liệu từ camera, cảm biến, phát hiện mối đe dọa, và tạo cảnh báo.

Entity Service:

- **Data Collection Service**: Quản lý việc thu thập dữ liệu từ các nguồn khác nhau như camera, cảm biến và thiết bị báo động.
- **Sensor Data Analysis Service**: Xử lý dữ liệu cảm biến để phát hiện các thay đổi bất thường về nhiệt độ và các chỉ số môi trường khác.
- **Rule Management Service**: Quản lý các quy tắc giám sát an ninh được định nghĩa trước và cập nhật khi có các yêu cầu mới.
- **Alert Management Service**: Quản lý các cảnh báo bao gồm thông tin chi tiết, tình trạng cảnh báo, và lịch sử cảnh báo.

Micro Service:

- **Detection Service**: Phát hiện mối đe dọa từ dữ liệu đã hợp nhất, bao gồm các tín hiệu xung đột và các dấu hiệu nguy hiểm.
- **Video Analysis Service**: Phân tích dữ liệu video thời gian thực, bao gồm nhận diện khuôn mặt và phát hiện hành vi bất thường.

Utility Service:

- **Notification Service**: Gửi cảnh báo thời gian thực đến trung tâm giám sát và kích hoạt các hành động phản hồi tự động.
- **Reporting Service**: Tạo báo cáo định kỳ về các sự cố và hiệu quả giám sát để hỗ trợ cải tiến hệ thống.

3.35. Phân tích hành vi người dùng

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Real-Time Recommendation Service**: Xử lý toàn bộ quy trình đề xuất cá nhân hóa thời gian thực, bao gồm thu thập sự kiện người dùng, phân tích hành vi, và hiển thị đề xuất cá nhân hóa.

Entity Service:

- **User Profile Service**: Quản lý hồ sơ người dùng bao gồm lịch sử duyệt web, các sự kiện tương tác và sở thích cá nhân.

- **Behavior Tracking Service**: Ghi nhận và lưu trữ chi tiết các sự kiện hành vi từ người dùng, bao gồm thời gian trên trang, tần suất tương tác và nội dung quan tâm.
- **Campaign Management Service**: Quản lý các chiến dịch tiếp thị nhắm đến đối tượng dựa trên dữ liệu hành vi và sở thích của người dùng.
- **Analytics Service**: Phân tích dữ liệu sau phiên truy cập để tối ưu hóa các chiến dịch tiếp thị và cải thiện mô hình đề xuất.

Micro Service:

- **Event Stream Processing Service**: Xử lý các sự kiện từ người dùng trong thời gian thực để nhận diện xu hướng và hành vi.
- **Personalization Model Service**: Mô hình đề xuất để tạo ra các đề xuất cá nhân hóa cho từng người dùng dựa trên dữ liệu hành vi và sở thích.

Utility Service:

- **Notification Service**: Gửi các thông báo nhắc nhở qua email hoặc SMS với các đề xuất nội dung và ưu đãi cá nhân hóa.
- **Performance Monitoring Service**: Theo dõi hiệu suất và độ trễ của các dịch vụ xử lý sự kiện.

3.36. Thu thập dữ liệu tự động

Gợi ý phân tích thiết kế theo kiến trúc hướng dịch vụ (SOA):

Task Service:

- **Data Crawling Service**: Quản lý toàn bộ quy trình thu thập dữ liệu, từ lập lịch trình, thu thập dữ liệu, đến lưu trữ dữ liệu sau khi xử lý.

Entity Service:

- **Source Management Service**: Quản lý các nguồn dữ liệu, bao gồm URL, API và thông tin tần suất thu thập, và ghi nhận các thay đổi trong cấu trúc nguồn.
- **Data Service**: Quản lý thông tin về các bản ghi dữ liệu.
- **Logging Service**: Lưu trữ nhật ký chi tiết về hoạt động thu thập để hỗ trợ cho việc kiểm tra, truy vết và cải tiến hệ thống.

Micro Service:

- **Duplicate Detection Service**: Kiểm tra và loại bỏ các bản ghi trùng lặp trong quá trình xử lý dữ liệu, đảm bảo chỉ lưu trữ các bản ghi duy nhất.

Utility Service:

- **Notification Service**: Gửi thông báo cho quản trị viên hoặc hệ thống khi phát hiện thay đổi đột ngột hoặc lỗi xảy ra trong quá trình thu thập.
- **Performance Analysis Service**: Phân tích hiệu suất hệ thống, bao gồm các chỉ số như thời gian xử lý, tần suất thu thập và số lượng dữ liệu trùng lặp, để hỗ trợ tối ưu hóa quy trình.
- **Reporting Service**: Tạo báo cáo định kỳ về kết quả thu thập dữ liệu, tỷ lệ thành công và hiệu suất của quy trình để phục vụ cải tiến.