BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH : CÔNG NGHỆ THÔNG TIN

Sinh viên : Bùi Đức Thắng

Giảng viên hướng dẫn: ThS. Đỗ Văn Tuyên

HẢI PHÒNG – 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

XÂY DỰNG WEBSITE LẤY Ý KIẾN TRỰC TUYẾN VỀ CÔNG TÁC GIẢNG DẠY CỦA GIẢNG VIÊN HPU

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Bùi Đức Thắng

Giảng viên hướng dẫn: ThS. Đỗ Văn Tuyên

HẢI PHÒNG – 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Bùi Đức Thắng **Mã SV**: 1912111003

Lớp : CT2301M

Ngành : Công nghệ thông tin

Tên đề tài: Xây dựng Website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a) Nội dung

- Tìm hiểu tổng quan về kiến trúc Microservices
- Tìm hiểu và phân tích thiết kế hệ thống theo DDD (Domain Driven Design)
- Xây dựng phần mềm

b) Yêu cầu cần giải quyết

- Hiểu được tổng quan về kiến trúc Microservices và áp dụng vào bài toán thực tế.
- Sử dụng GraphQL để giải quyết các bài toán Realtime
- Xây dựng thành công website thăm dò công tác giảng dạy tại HPU.

2. Các tài liệu, số liệu cần thiết

- Quyết định số 73/QĐ-HĐT về việc ban hành Quy chế đánh giá công tác giảng dạy tại HPU.
- Tài liệu về REST API, JWT (Json Web Token), GraphQL, kiến trúc Microservices, DDD (Domain Driven Design)

3. Địa điểm thực tập tốt nghiệp

- Trường Đại học Quản Lý và Công Nghệ Hải Phòng

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên : Đỗ Văn Tuyên

Học hàm, học vị : Thạc sĩ

Cơ quan công tác : Trung tâm thông tin thư viện - Trường Đại học Quản lý và Công Nghệ Hải Phòng

Nội dung hướng dẫn: Xây dựng Website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU

Đề tài tốt nghiệp được giao ngày 27 tháng 03 năm 2023

Yêu cầu phải hoàn thành xong trước ngày 17 tháng 06 năm 2023

Đã nhận nhiệm vụ ĐTTN	Đã giao nhiệm vụ ĐTTN
Sinh viên	*Giảng viên hướng dẫn*
Bùi Đức Thắng	**ThS. Đỗ Văn Tuyên**

Hải Phòng, ngày ….. tháng….. năm 2023

TRƯỞNG KHOA

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: Đỗ Văn Tuyên.....

Đơn vị công tác: Trung tâm thông tin thư viện - Trường Đại học Quản Lý và Công Nghệ Hải Phòng .....

Họ và tên sinh viên: Bùi Đức Thắng..... Ngành: Công nghệ Thông tin

Nội dung hướng dẫn: Xây dựng Website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU

1.Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp

.....

.....

.....

.....

.....

2.Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đã đề ra trong nhiệm vụ Đ.T. T.N trên các mặt lý luận, thực tiễn, tính toán số liệu…)

.....

.....

.....

.....

.....

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp

Đạt ☐ Không đạt ☐ Điểm:……...

Hải Phòng, ngày .... tháng ..... năm 2023

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên: Vũ Anh Hùng.....

Đơn vị công tác: Khoa Công nghệ thông tin - Trường Đại học Quản Lý và Công Nghệ Hải Phòng .....

Họ và tên sinh viên: Bùi Đức Thắng..... Ngành: Công nghệ thông tin

Đề tài tốt nghiệp: Xây dựng Website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU

1. Phần nhận xét của giảng viên chấm phản biện

.....

.....

.....

.....

.....

.....

2. Những mặt còn hạn chế

.....

.....

.....

.....

.....

.....

3. Ý kiến của giảng viên chấm phản biện

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm:

Hải Phòng, ngày …… tháng ….. năm 2023

Giảng viên chấm phản biện

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành tốt được Đồ án tốt nghiệp, em xin gửi lời cảm ơn chân thành đến các thầy cô trong Khoa Công Nghệ Thông tin của Trường ĐH Quản Lý và Công Nghệ Hải Phòng đã tạo điều kiện tốt nhất cho em để em hoàn thành đề tài đúng như dự kiến. Đặc biệt em xin gửi lời cảm ơn sâu sắc đến Cô Nguyễn Thị Xuân Hương – Lãnh đạo Khoa Công Nghệ Thông Tin và Thầy Đỗ Văn Tuyên – Giảng viên hướng dẫn đồ án đã trực tiếp hướng dẫn và tận tình giúp đỡ em để em có thể hoàn thành tốt đồ án tốt nghiệp của mình.

Em xin chân thành cảm ơn các lãnh đạo của Trường ĐH Quản Lý và Công Nghệ, các Thầy/Cô trong khoa Công Nghệ Thông Tin đã tạo cho em điều kiện tốt nhất từ khi còn ngồi trên ghế nhà trường cho đến khi hoàn thành đồ án tốt nghiệp quan trọng nhất trong cuộc đời sinh viên.

Trong quá trình thực tập, cũng như là trong quá trình làm đồ án tốt nghiệp em không tránh khỏi những sai sót, em rất mong các Thầy, Cô bỏ qua. Đồng thời do trình độ lý luận cũng như trong kinh nghiệm thực tiễn của em còn nhiều hạn chế nên không tránh khỏi những thiếu sót. Vậy nên, em rất mong sự đóng góp ý kiến từ Thầy, Cô để em học thêm được nhiều kinh nghiệm và kiến thức để có thể góp ích cho những công việc sau này.

Em xin chân thành cảm ơn!

Hải Phòng, ngày tháng năm 2023

Sinh viên

(Ký và ghi rõ họ tên)

LỜI CAM ĐOAN

Em xin cam đoan rằng đề tài này được tiến hành một cách minh bạch, công khai. Mọi thứ được dựa trên sự cố gắng cũng như sự nỗ lực của bản thân cùng với sự giúp đỡ của thầy Đỗ Văn Tuyên.

Các số liệu và kết quả nghiên cứu được đưa ra trong đồ án là trung thực và không sao chép hay sử dụng kết quả của bất kỳ đề tài nghiên cứu nào tương tự. Nếu như phát hiện rằng có sự sao chép kết quả nghiên cứu của những đề tài khác, bản thân em xin hoàn toàn chịu trách nhiệm.

Hải Phòng, ngày tháng năm 2023

Sinh viên

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

TỪ VIẾT TẮT	TIẾNG ANH	TIẾNG VIỆT
API	Application Programming Interface	Giao diện lập trình ứng dụng
JWT	Json Web Token	
DDD	Domain Driven Design	
HTTP	Hypertext Transfer Protocol	
HTTPS	Hypertext Transfer Protocol Security	
JSON	JavaScript Object Notation	

DANH MỤC HÌNH ẢNH

LỜI NÓI ĐẦU

Ngày nay, với định hướng phát triển giáo dục phù hợp với nhu cầu xã hội (người học, phụ huynh và người sử dụng lao động) kèm theo đó là yêu cầu ngày càng tăng trách nhiệm giải trình của các cơ sở giáo dục về chất lượng đào tạo thị đánh giá chất lượng hoạt động giảng dạy là một minh chứng cần thiết cho chất lượng đào tạo của các trường đại học.

Chủ trương lấy ý kiến phản hồi từ người học để thay đổi cho phù hợp nhận được sừ đồng tình từ phía các trường, giáo viên và người học. Mục đích của hoạt động này nhằm góp phần thực hiện quy chế dân chủ; xây dựng đội ngủ giáo viên có phẩm chất đạo đức, thái độ nghề nghiệp và trình độ chuyên môn cao, phương pháp và phong cách giảng dạy tiên tiến, hiện đại.

Hoạt động đánh giá công tác giảng dạy phát huy tính tự giác, nêu cao tinh thần trách nhiệm; tích cực, chủ động, sáng tạo trong giảng dạy, tạo động lực phấn đấu hoàn thành tốt nhiệm vụ được giao; khuyến khích việc không ngừng nâng cao chất lượng và hiệu quả công tác đào tạo, góp phần xây dựng Nhà trường ngày càng phát triển; đảm bảo sự công bằng trong phân bổ lương năng suất chất lượng; là cơ sở trong việc bình xét thi đua, khen thưởng; bố trí, sử dụng có hiệu quả nguồn nhân lực, tái ký hợp đồng lao động và thực hiện các chế độ, chính sách khác đối với giảng viên.

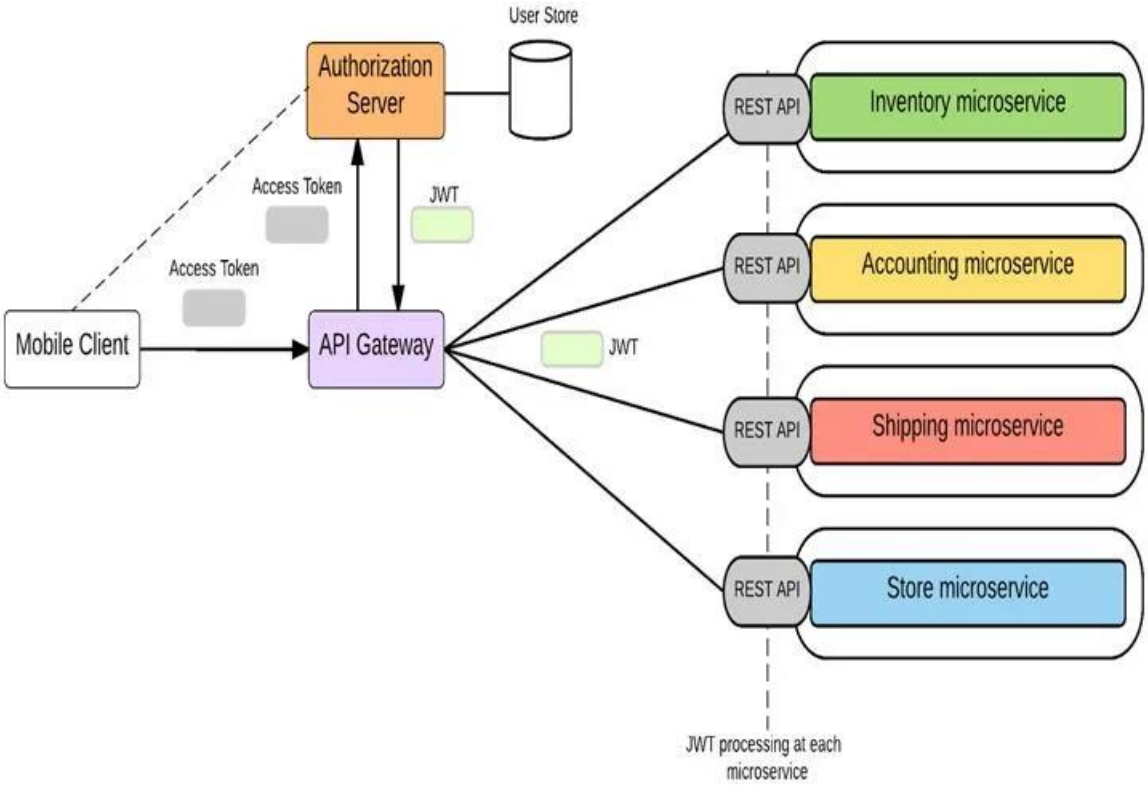
Hoạt động xin ý kiến phản hồi công tác giảng dạy từ sinh viên hiện nay được thực hiện trên hình thức trực tiếp, phát phiếu đến từng sinh viên để xin ý kiến. Với số lượng lớn sinh viên, cũng như giảng viên, môn học, việc này gây tốn rất nhiều thời gian, công sức. Việc in ấn cũng tốn rất nhiều chi phí. Ngoài ra, khi tổng hợp lại kết quả đánh giá còn gây khó khăn trong việc thu lại phiếu, tổng hợp, tính toán khó có tình chính xác cao, tốn nhiều thời gian và sức người.

Vì vậy việc "**Xây dựng Website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU**" là một giải pháp có thể giúp giải quyết phần nào được những vấn đề trên và tăng được hiệu suất của công việc một cách nhanh chóng hơn.

CHƯƠNG 1: TỔNG QUAN VỀ KIẾN TRÚC MICROSERVICES

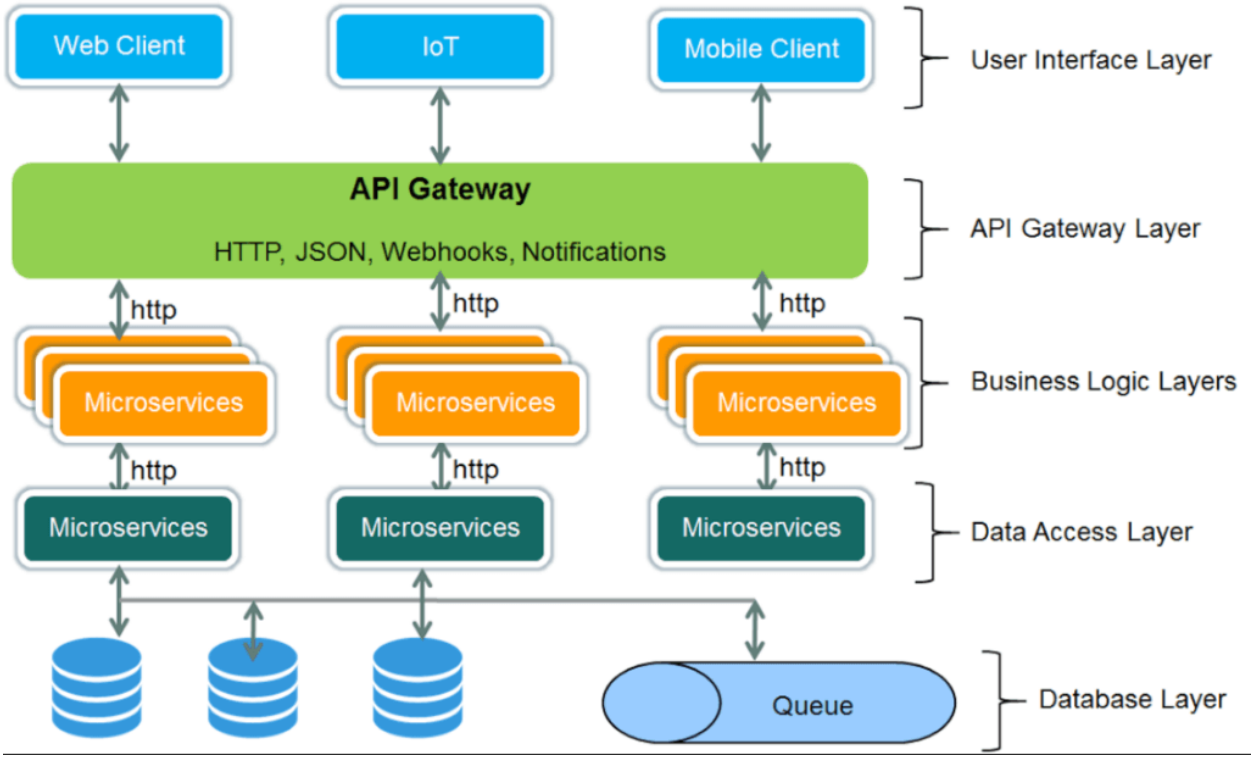
1.1 Microservices là gì?

Microservices [1] là một kiểu kiến trúc phần mềm. Được hiểu một cách ngắn gọn là chúng ta chia một phần mềm lớn thành nhiều phần khác nhau được gọi là service, mỗi service sẽ đảm nhiệm một nhiệm vụ riêng biệt và độc lập với những service khác. Mỗi service sẽ được đặt trên một server riêng để có thể dễ dàng nâng cấp cũng như phát triển ứng dụng.



Hình 1.1.1 Hình ảnh tổng quan về kiến trúc Microservices

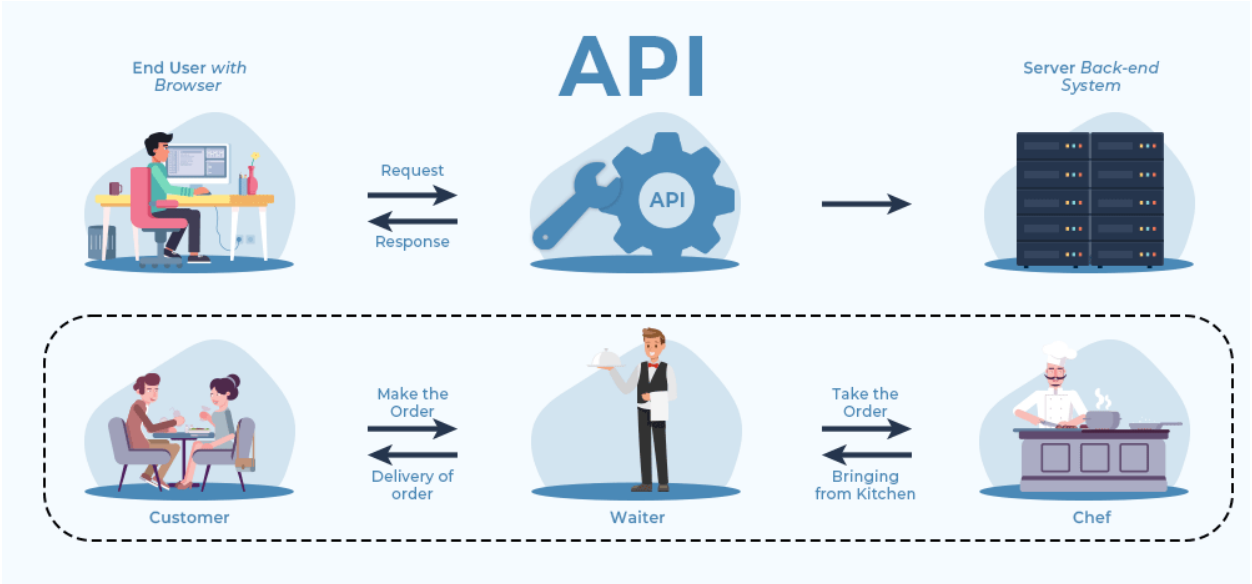
Một framework microservices xây dựng một hệ thống phân tán, có khả năng mở rộng và quy mô lớn, giúp giảm thiểu hiện tượng tắc nghẽn cho cơ sở dữ liệu trung tâm. Đồng thời kiến trúc này còn cải thiện các khả năng kinh doanh cho doanh nghiệp, chẳng hạn như cho phép các ứng dụng phân phối và triển khai liên tục, tiếp cận các hệ thống công nghệ hiện đại.



Hình 1.1.2 Ví dụ về mô hình kiến trúc Microservices

– **Tổng quan về API** [2]

API là các phương thức, giao thức kết nối các ứng dụng khác nhau để chia sẻ tài nguyên và dữ liệu. Nó là viết tắt của **Application Programming Interface** – giao diện lập trình ứng dụng. API có khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng và từ đó có thể trao đổi dữ liệu giữa các ứng dụng với nhau.



Hình 1.1.3 Mô tả API

Theo như hình trên ta có thể hiểu một cách đơn giản thì API đóng vai trò là một người trung gian (bồi bàn) có nhiệm vụ nhận yêu cầu của khách hàng và chuyển về cho đầu bếp, Sau đó bồi bàn sẽ đưa món ăn nhà bếp làm theo yêu cầu của khách hàng.

RESTful API [3] là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP hoặc HTTPS. Chức năng quan trọng nhất của **REST** là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một **RESTful API**.

- GET (SELECT): Trả về dữ liệu hoặc một mảng dữ liệu
- POST (CREATE): Tạo mới một dữ liệu
- PUT (UPDATE): Cập nhật dữ liệu
- DELETE (DELETE): Xoá dữ liệu

Đối với trang web lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên cũng vậy, sẽ sử dụng API để giao tiếp giữa Client và Server. Và để tăng tính bảo mật cho API em có sử dụng thêm JWT (JSON Web Token) để đảm bảo rằng việc giao tiếp giữa Client và Server không bị phá hoại bởi những người khác.

– **Tổng quan về JWT** [4]

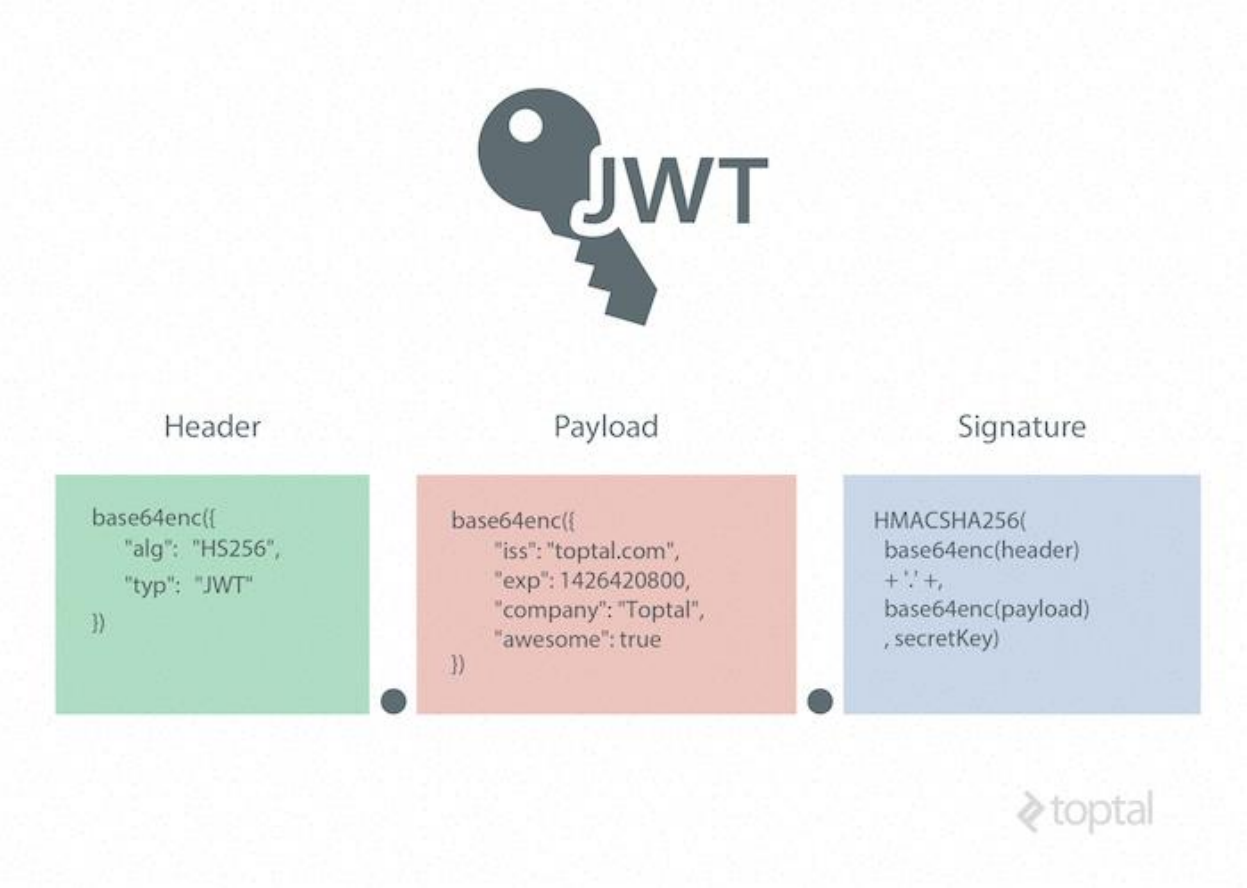
JWT là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server, các thông tin trong chuỗi JWT được định dạng bằng JSON. Trong đó chuỗi TOKEN phải có 3 phần header, payload và signature được ngăn bằng dấu ".". Cả 3 phần phải được mã hoá bằng một mã bí mật để tạo ra một JWT hoàn chỉnh.

JWT dùng để xác định xem người dùng là ai sau khi đã đăng nhập vào hệ thống. Mỗi khi có một yêu cầu lấy dữ liệu qua API thì **JWT** sẽ được tạo ra và gửi đi kèm theo request và khi đó Server sẽ xác thực lại **JWT** đảm bảo rằng người dùng có được phép sử dụng các dữ liệu đã request trước đó hay không.

Header: Phần header sẽ chứa kiểu dữ liệu và thuật toán được sử dụng để mã hoá chuỗi JWT.

Payload: Phần Payload là nơi sẽ chứa những thông tin mà mình muốn gửi đến cho Server, ngoài ra còn có thời gian hết hạn của TOKEN cũng như thời gian TOKEN được tạo ra.

Signature: Phần signature sẽ được tạo ra bằng cách mã hoá phần **header, payload** theo như thuật toán ở phần **header** với một mã khoá bí mật. Kết hợp 3 phần lại ta sẽ có một JWT hoàn chỉnh.



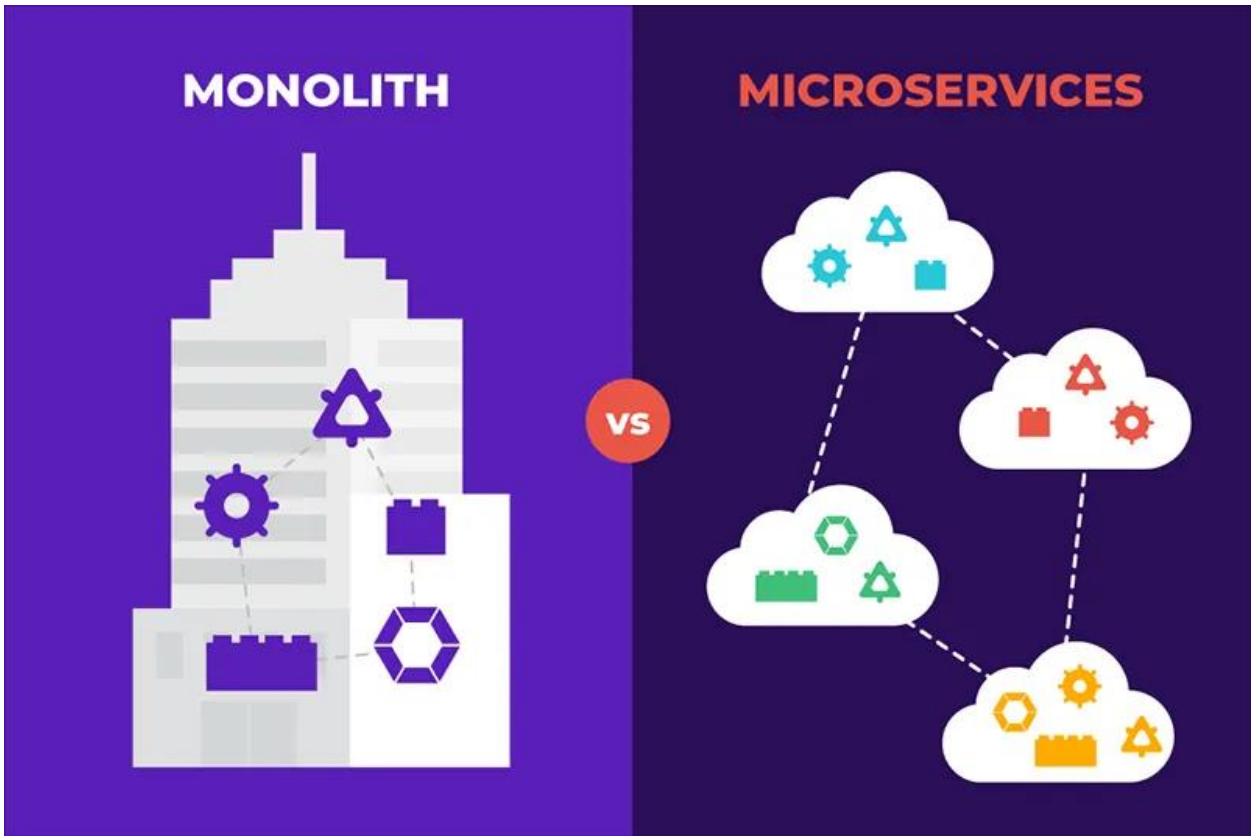
Hình 1.1.4 Cấu trúc của JWT

Như vậy, khoá bí mật sẽ chỉ được lưu ở phía các Server để xác thực các lượt truy cập từ phía client thông qua API. Chúng ta không nên để lộ khoá bí mật và khoá chỉ được biết bởi những người có trách nhiệm về phần mềm được biết để tránh những việc lộ thông tin cũng như giả danh người dùng.

Theo như sơ đồ trên là một thể hiện của mô hình Microservice. Một ứng dụng sẽ được chia thành một bộ các microservice, mỗi microservice thực chất là một service có thể được triển khai và chạy độc lập. Chúng tách biệt về mặt mã nguồn, về hoạt động và dữ liệu. Mỗi microservice có nơi chứa dữ liệu của riêng của nó và chỉ có nó có quyền truy cập vào vùng dữ liệu này. Do các microservice là độc lập, chúng không giao tiếp trực tiếp với nhau mà qua một thành phần trung gian được gọi là API gateway. Có thể thấy vai trò của API gateway rất quan trọng trong mô hình microservice. Nó là điểm đến và đi của mọi yêu cầu hay phản hồi.

1.2 Monolith Application là gì?

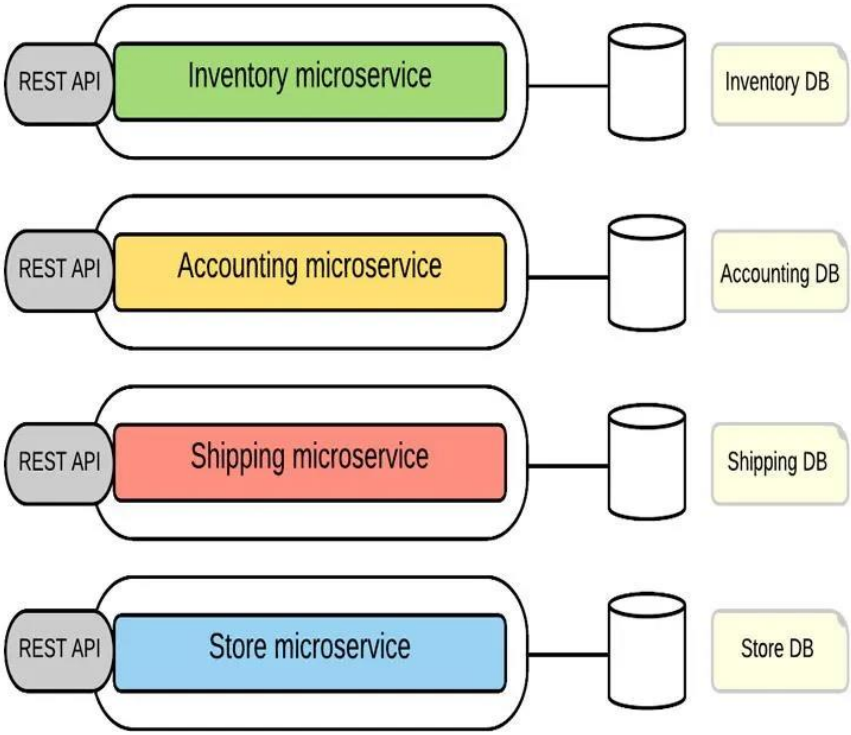
Trái với khái niệm **Microservices**, **Monolith Application** [5] được thiết kế để xử lý nhiều tác vụ liên quan với nhau, thường là những ứng dụng phức tạp và có nhiều tính năng có mối liên hệ chặt chẽ với nhau. Toàn bộ hệ thống có thể sẽ chứa ở một server vì vậy thường sẽ có một khối lượng code rất lớn. Một thay đổi nhỏ trong bất kỳ chức năng nào cũng có thể cần phải biên dịch và thử nghiệm lại toàn bộ nền tảng.



Hình 1.2.1 Mô phỏng Monolith và microservices

1.3 Kiến trúc microservices là gì?

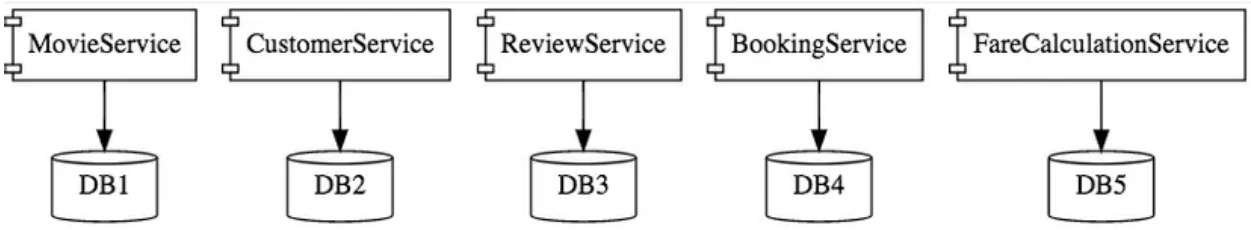
Kiến trúc microservices [6] xem mỗi chức năng của một ứng dụng như một dịch vụ độc lập, có thể thay đổi, cập nhật hay gỡ bỏ mà không ảnh hưởng gì đến những phần còn lại của ứng dụng.



Hình 1.3.1 kiến trúc microservices là gì?

Các ứng dụng truyền thống được xây dựng theo kiến trúc monolithic. Khi đó, việc bổ sung tính năng mới yêu cầu phải cấu hình và cập nhật lại mọi thứ: từ quy trình, giao tiếp cho đến các vấn đề bảo mật trong ứng dụng. Các ứng dụng monolithic truyền thống thường có vòng đời dài, chu kỳ cập nhật không ổn định và các thay đổi thường có hiệu lực lên toàn bộ hệ thống ứng dụng. Việc này sẽ tốn nhiều chi phí và đôi khi có thể gây trì trệ quá trình phát triển ứng dụng trong một doanh nghiệp.

Đây là một trong những nguyên nhân chính dẫn đến sự ra đời của kiến trúc microservices. Trong đó, mọi dịch vụ được xây dựng và phát triển độc lập hoàn toàn với nhau. Khi đó các doanh nghiệp có thể dễ dàng mở rộng dịch vụ của mình dựa trên từng nhu cầu kinh doanh cụ thể. Bên cạnh đó, các dịch vụ cũng có thể được thay đổi và cập nhật nhanh chóng mà không cần ảnh hưởng đến những thành phần khác.



Hình 1.3.2 hình minh hoạ ứng dụng xây dựng theo kiến trúc microservices

1.4 Các đặc trưng của mô hình Microservices

- **Micro-service:** Đặc trưng này được thể hiện ngay từ tên của kiến trúc. Nó là microservice chứ không phải là miniservice hay nanoservice. Trên thực tế không tồn tại mô hình kiến trúc cho miniservice hay nanoservice. Từ microservice được sử dụng để giúp người thiết kế có cách tiếp cận đúng đắn. Một ứng dụng lớn cần được chia nhỏ ra thành nhiều phần, các thành phần đó cần tách biệt về mặt dữ liệu (Database) và phải đủ nhỏ cả về mặt kích cỡ và độ ảnh hưởng của nó trong hệ thống, khi thêm một microservice vào hệ thống cũng nên đảm bảo rằng nó đủ nhỏ để dễ dàng tháo gỡ, xoá bỏ khỏi hệ thống mà không ảnh hưởng nhiều đến các thành phần khác
- **Tính độc lập:** Các microservice hoạt động tách biệt nhau trong hệ thống, do vậy việc build một microservice cũng độc lập với việc build các microservice khác. Thông thường, để tiện cho việc phát triển và duy trì các microservice, người phát triển nên viết các built script khác nhau cho mỗi microservice.

  Do tính tách biệt này mà mỗi microservice đều dễ dàng thay thế và mở rộng. Hơn thế nữa, nó còn giúp việc phát triển các microservice linh động hơn, các microservice có thể được phát triển bởi các team khác nhau, dùng các ngôn ngữ khác nhau và tiến độ phát triển dự án cũng nhanh hơn do không có sự phụ thuộc giữa các team, mỗi team có thể chủ động quản lý phần việc riêng của mình.
- **Giao tiếp qua API:** Các microservice giao tiếp với nhau thông qua API (Application Programming Interface). Điều này giúp giảm sự phụ thuộc

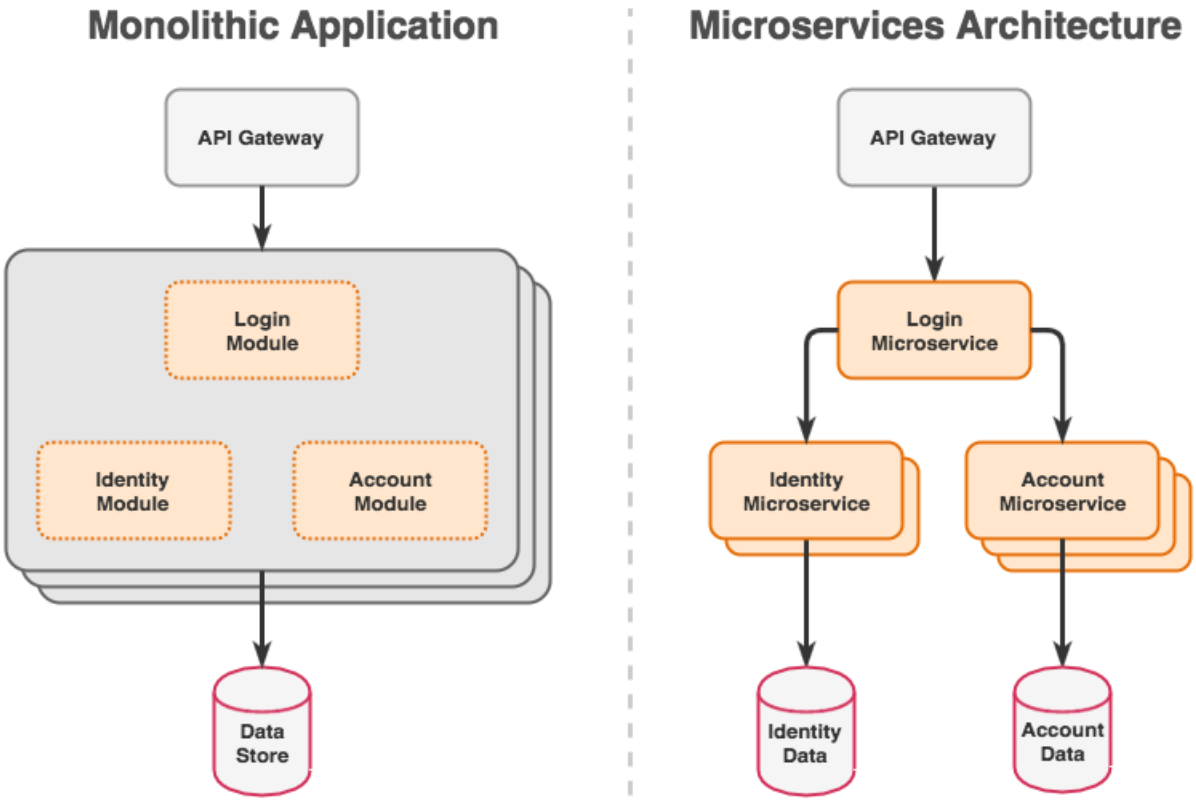
trực tiếp giữa các thành phần và tạo điều kiện cho việc thay đổi và mở rộng dễ dàng.

- **Tính chuyên biệt:** Mỗi microservice là một dịch vụ chuyên biệt, có thể hoạt động độc lập, thông thường mỗi microservice đại diện cho một tính năng mà các công ty/ doanh nghiệp muốn cung cấp tới người dùng, do vậy người thiết kế hệ thống microservice cần hiểu rõ về hoạt động kinh doanh của công ty. Các đầu vào đầu ra và chức năng của mỗi microservice cần được định nghĩa rõ ràng.
- **Phòng chống lỗi:** Kiến trúc microservice sinh ra là để dành cho các hệ thống từ lớn đến vô cùng lớn. Nó áp dụng phương pháp chia để trị, phương pháp này giúp việc áp dụng các công cụ, kỹ thuật cho việc giám sát, phòng chống lỗi phần mềm, lỗi hệ thống hiệu quả.

  Khi một thành phần trong hệ thống bị lỗi, nó có thể được thay thế bằng các thành phần dự phòng một cách dễ dàng, trong quá trình thay thế thành phần bị lỗi, các thành phần khác vẫn hoạt động bình thường, do vậy hoạt động của toàn bộ hệ thống sẽ không hoặc ít bị gián đoạn.

⇨ Tuy kiến trúc microservices mang lại nhiều lợi ích như sự linh hoạt, mở rộng dễ dàng và phát triển song song, nhưng cũng đòi hỏi một quá trình quản lý phức tạp hơn vì số lượng và sự phức tạp của các microservice. Cần đảm bảo quản lý và giám sát chúng một cách hiệu quả để đảm bảo tính sẵn sàng và hiệu suất của hệ thống.

1.5 Các ưu và nhược điểm của Microservices



Hình 1.5.1 Hình ảnh cấu trúc của monolithic và microservices

- **Kiến trúc ứng dụng nguyên khối (monolithic application)**

  Với kiến trúc nguyên khối toàn bộ ứng dụng là một khối lớn, trong khối lớn ấy có chia thành các mô đun nhỏ, mỗi mô đun thực hiện một nhiệm vụ riêng và các mô đun thường gọi nhau qua function call. Việc phát triển và triển khai ứng dụng với kiến trúc này khá đơn giản khi mà các IDE hỗ trợ rất tốt việc kiểm tra và chạy ứng dụng với chỉ một cú click chuột hay một phím tắt. Kiến trúc này cũng đặc biệt phù hợp với các công ty outsource, họ có thể tạo ra một mẫu ứng dụng(template) và khi nhận được dự án với khách hàng mới họ chỉ việc bê mẫu này ra và chỉnh sửa, thêm thắt một vài phần khác biệt mà khách hàng yêu cầu.

  Trong ứng dụng nguyên khối, sự chặt chẽ là vừa là ưu điểm vừa là nhược điểm. Với một khối ứng dụng lớn, người ta phải định nghĩa ra các tiêu chuẩn và các mẫu thiết kế, các mẫu đầu vào, đầu ra để đảm bảo cho code có chất lượng cao và nhất quán. Sự chặt chẽ và nhất quán này đôi khi là rào cản cho

một số lập trình viên khác, họ không quen hoặc họ thích dùng các chuẩn viết code khác hơn, những lập trình viên mới cũng phải mất một thời gian dài ban đầu để làm quen với điều này.

Khi ứng dụng nguyên khối ngày một lớn thì quá trình maintain (duy trì) và sửa lỗi ứng dụng cũng trở thành ác mộng vì thời gian build và khởi động lại ứng dụng rất lớn. Đôi khi chỉ sửa một dòng code mà phải khởi động lại cả một chương trình to mất tới cả nửa tiếng.

Như vậy khi ứng dụng lớn dần lên, đến một lúc nào đó, kiến trúc nguyên khối sẽ không còn thực sự phù hợp nữa. Sự gia đời của kiến trúc Microservice là để khắc phục điều này.

- **Ưu điểm của kiến trúc Microservices** [5]
    - Cho phép lập trình viên linh động hơn trong việc lựa chọn ngôn ngữ, công cụ và nền tảng để phát triển và triển khai các microservice (tuy nhiên trong một hệ thống, việc lựa chọn các ngôn ngữ khác nhau để phát triển các microservice không được khuyến khích).
    - Một microservice có thể được phát triển bởi một team nhỏ. Do vậy việc quản lý sẽ dễ dàng hơn.
    - Mỗi microservice có kích thước nhỏ, giúp cho các lập trình viên dễ tiếp cận, đọc hiểu source code. Do vậy các thành viên mới tham gia team sẽ hòa nhập và đóng góp cho team nhanh hơn.
    - Các microservice khởi động nhanh giúp quá trình phát triển, kiểm thử cũng nhanh hơn.
    - Dễ dàng mở rộng và tích hợp với các dịch vụ của bên thứ ba.
    - Cô lập lỗi tốt hơn, khi một microservice bị lỗi và ngừng hoạt động thì các microservice khác vẫn có thể hoạt động bình thường. Với mô hình nguyên khối, một lỗi nhỏ có thể làm cả hệ thống ngừng hoạt động.
    - Khi cần thay đổi một thành phần, thì chỉ cần sửa đổi, cập nhật và triển khai lại thành phần đó chứ không cần triển khai lại toàn bộ hệ thống.
- **Nhược điểm của kiến trúc Microservices** [5]

- Việc triển khai hệ thống microservice phức tạp hơn nhiều so với việc triển khai hệ thống nguyên khối.
- Các lập trình viên phải tốn nhiều công sức hơn để thực hiện phần giao tiếp giữa các microservice, với kiến trúc nguyên khối có khi họ chỉ cần gọi hàm để thực hiện việc này.
- Các microservice thường (nên) được triển khai bên trong docker container và giao tiếp với nhau qua REST API. Việc này làm hiệu năng của toàn bộ chương trình ứng dụng giảm xuống đáng kể do giới hạn tốc độ truyền tải của các giao thức và tốc độ mạng. Hơn nữa việc giao tiếp giữa các microservice có thể bị lỗi khi các kết nối bị lỗi.
- Cần tính toán kích cỡ của một microservice. Nếu một microservice quá lớn, bản thân nó trở thành một ứng dụng theo kiến trúc nguyên khối. Nếu một microservice quá nhỏ thì độ phức tạp của hệ thống tăng lên rất nhiều, làm cho hệ thống trở lên khó hiểu, lúc này việc quản lý giám sát và triển khai hệ thống sẽ khó khăn hơn.
- Khi ứng dụng ngày càng lớn lên, số lượng microservice ngày càng nhiều, các lập trình viên thường có xu hướng sử dụng sự hỗ trợ từ các công cụ mã nguồn mở, hoặc của bên thứ 3, việc sử dụng, tích hợp các công cụ này làm cho hệ thống khó kiểm soát và có thể bị dính các mã độc làm cho hệ thống kém an toàn.
- Cần nhiều tài nguyên hơn, số lượng microservices sẽ phải tỉ lệ thuận với lượng tài nguyên cần thiết để triển khai hệ thống, đồng thời cũng cần duy trì nhiều database hơn cho ứng dụng.

1.6 Thiết kế phần mềm theo kiến trúc Microservices

- **Mỗi microservices nên có một database riêng biệt**: Việc này đảm bảo chó mỗi microservices có tính đóng gói cao.
- **Giữ source code của microservice ở mức hợp lý**: Như đã đề cập ở phần ưu và nhược điểm của microservice. Kích thước source code của một microservice không nên quá nhỏ hoặc quá lớn. Tuy nhiên cái khó ở đây là

không có một con số định lượng cho kích thước của một microservice, nên thông thường việc quyết định kích thước của một microservice là do kinh nghiệm, cảm tính.

- **Triển khai mỗi microservice bên trong một App (docker container)**: Việc triển khai mỗi microservice trong một docker container đem lại rất nhiều lợi ích cho việc triển khai và mở rộng ứng dụng cũng như việc phân chia tài nguyên phần cứng cho mỗi microservice. Hiện nay có rất nhiều công cụ hỗ trợ cho việc liên tục tích hợp, liên tục triển khai hệ thống microservice. Các công cụ này giúp tăng hiệu quả làm việc cho các lập trình viên, giảm thời gian phôi phối sản phẩm phần mềm, và các công cụ này đòi hỏi mỗi microservice được đóng gói trong một docker image và triển khai trên app.

1.7 Kết luận chương

Chương 1 đã trình bày được về tổng quan cũng như khái niệm về kiến trúc Microservices. Từ đó ta hiểu được mô hình của một kiến trúc microservices. Nắm bắt được ưu cũng như nhược điểm của nó. Giúp cho ta biết được khi nào nên dùng kiến trúc Microservice trong một bài toán thực tế.

CHƯƠNG 2: TỔNG QUAN VỀ DOMAIN DRIVEN DESIGN (DDD)

2.1 DDD là gì?

DDD [7] là một cách tiếp cận để phát triển những phần mềm phức tạp thông qua sự kết nối chặt chẽ giữa việc triển khai ứng dụng với sự phát triển của mô hình kinh doanh. Tiền đề tạo nên DDD là:

- Đặt trọng tâm dự án vào nghiệp vụ chính (core domain) và các logic của nghiệp vụ (domain logic)
- Mô hình hoá là trọng tâm, là nền tảng cho các thiết kế phức tạp.
- Sự cộng tác đầy sáng tạo giữa nhóm dev và các domain expert (chuyên gia về lĩnh vực) tạo nên tiếng nói chung để xác định và giải quyết hiệu quả các vấn đề.

DDD tập trung vào khái niệm domain và bóc tách các bài toán dựa trên các domain đó. Tại sao phải dựa trên domain? Vì đây là cái khách hàng (domain expert) nắm rõ nhất. Chúng ta phát triển ứng dụng theo yêu cầu của khách hàng nên hiển nhiên không ai hiểu các yêu cầu của hệ thống bằng khách hàng. Và khi khách hàng giải thích hệ thống cho chúng ta hiểu, họ sẽ giải thích về các domain của nó. Chính vì thế các domain sẽ làm trọng tâm và công việc của chúng ta là xây dựng nó thành các mô hình để cho tất cả mọi người cùng nắm vấn đề.

Tóm lại, DDD là thiết kế sao cho không chỉ lập trình viên hiểu mà ngay cả khách hàng, những người không biết gì về mặt kỹ thuật cũng có thể nhìn vào nắm được trọng tâm của vấn đề. Trong thực tế, đây là một quá trình đòi hỏi rất nhiều kĩ năng và quá trình tiếp cận xây dựng có hệ thống. Cái khó khăn trong việc triển khai DDD là những lập trình viên như chúng ta hoàn toàn chẳng có tí tẹo gì về cái gọi là domain và phải xây dựng hệ thống dựa trên khái niệm domain nên việc đặt mọi thứ vào đúng vị trí của nó không phải là một điều dễ dàng.’

2.2 Domain là gì?

Domain, là kiến thức về một mảng lĩnh vực nào đó không liên quan đến công nghệ. Khi chúng ta bắt đầu một dự án phần mềm, ta nên tập trung vào domain

và hoạt động trong nó. Mục đích của cả phần mềm là để đề cao một domain cụ thể. Để làm được điều đó, phần mềm cần hài hoà với domain mà nó tạo lên.

Để xây dựng domain, chúng ta cần hiểu domain, trong trường hợp này chúng ta cần theo dõi những chuyên gia trong lĩnh vực này nhưng họ lại không phải là người thiết kế hay chuyên gia phần mềm để mô tả về domain của họ. Để mô hình hoá được domain, chúng ta cần chắt lọc thông tin và tổng quát hoá nó.

2.3 Ubiquitous language

- Sự cần thiết của một ngôn ngữ chung. Theo như phần trên ta thấy rằng việc giao tiếp giữa chuyên gia phần mềm và chuyên gia domain thường sẽ gặp những khó khăn về rào cản giao tiếp cơ bản. Lập trình viên chỉ nghĩ tới lớp, method, thuật toán, pattern và có khuynh hướng diễn tả mọi thứ đời thường bằng những thao tác lập trình. Họ muốn nhìn lớp đối tượng và tạo quan hệ mô hình giữa chúng. Họ nghĩ đến những thứ như kế thừa, đa hình, OOP... Và họ luôn nói theo cách đó. Tuy vậy, chuyên gia domain thường không hiểu những khái niệm đó. Họ không hiểu những khái niệm thuộc về phát triển phần mềm và tất cả họ biết chỉ là chuyên ngành của họ.
- Ngôn ngữ chung giữa domain expert và developer hiển nhiên là vấn đề được đặt lên hàng đầu. Việc sử dụng chung ngôn ngữ giúp tránh mọi nhầm lẫn dẫn đến sai sót trong quá trình xây dựng và phát triển ứng dụng. Đa số các lỗi đều xuất phát từ khách hàng nói một đàng và developer hiểu một nẻo. Đối với nhiều domain, đa số đều có những thuật ngữ riêng và đôi khi nó hoàn toàn xa lạ với dev. Nó gây ra nhiều nhầm lẫn khi thảo luận về ứng dụng. Ngôn ngữ chung giúp giảm bớt những nhầm lẫn như vậy. Việc phản ánh những thuật ngữ, các khái niệm của các domain vào source code hoàn toàn có thể thực hiện được thông qua các đặt tên các package, class, method, properties... Và tất nhiên là ta phải phản ảnh vào mọi tính năng để đảm bảo khách hàng nhìn vào cũng có thể hiểu được chúng là cái gì.

- Một nguyên tắc cốt lõi của thiết kế DDD là sử dụng ngôn ngữ dựa trên mô hình, vì mô hình là xuất phát điểm chung, là nơi ở đó có phần mềm "gặp" domain, việc sử dụng nó là nền tảng cho ngôn ngữ là hợp lý.

2.4 Bounded Context

Với DDD, ý tưởng chính là chia hệ thống phức tạp dựa trên các domain của nó. Tuy nhiên, đôi khi một số domain lại chồng chéo lên nhau và đối với những đối tượng khác nhau thì domain tương ứng cũng khác nhau. Chẳng hạn đơn giản nhất là việc xuất hóa đơn, đối với từng đối tượng thì nghiệp vụ xuất hóa đơn lại có cách xử lý khác nhau. Việc này gây ra những xử lý phức tạp về mặc logic. Nên dẫn đến để xử lý cho trường hợp này, cần phải bóc tách hệ thống thành những hệ thống con phục vụ cho những đối tượng nhất định và các hệ thống con này cũng có các domain tương ứng. Nói cách khác, việc chia hệ thống dựa trên những ngữ cảnh cụ thể, giới hạn từng đối tượng, từng domain (Bounded Context). Và với ý tưởng chia để trị như thế này, DDD trở nên rất phù hợp cho việc áp dụng mircoservice. Việc chia thành các ngữ cảnh cụ thể tương đương với việc tách các xử lý logic và tách biệt về cơ sở dữ liệu.

Một điều chú ý là các ngữ cảnh được chia nhỏ đều dựa trên một domain lớn, có nghĩa là chúng có liên quan đến nhau. Tuy nhiên, chúng cần phải được tách biệt và không phụ thuộc lẫn nhau. Xu hướng thiết kế hiện đại là đảo ngược sự phụ thuộc và tất nhiên DDD cũng thế. Thay vì phụ thuộc lẫn nhau, chúng ta sẽ tạo ra một layer trung gian, ở giữa hai ngữ cảnh và cho chúng phụ thuộc vào layer này. Điều này hoàn toàn hữu ích khi thay đổi hoặc tái cấu trúc các ngữ cảnh, ta chỉ cần sửa lại các layer này và các ngữ cảnh sẽ không bị ảnh hưởng. Các layer này được gọi là Anti-Corruption Layer (layer chống hủy hoại hệ thống).

Tóm lại, Bounded Context là những domain độc lập được chia tách từ các domain có tính phức tạp, việc này rất phù hợp với việc áp dụng kiến trúc Microservices nhưng vẫn cần một trung gian để tránh việc các domain hoạt động không đúng cách.

2.5 Anti - Coruption layer

Anti-Coruption layer là một lớp trung gian giữa các domain nhỏ. Nó được dùng để cô lập 2 domain trước đó hoạt động phụ thuộc vào nhau, làm cho 2 domain hoạt động phụ thuộc vào layer thay vì hoạt động phụ thuộc vào nhau. Bằng cách này, khi chúng ta thay thế một trong những domain con thì ta chỉ cần cập nhật lại lớp layer mà không làm tổn hại gì đến domain con khác.

Việc này đặc biệt hữu ích khi chúng ta cần tích hợp một hệ thống mới với hệ thống cũ đang có. Lớp layer sẽ đảm nhiệm vai trò điều hướng API để thích ứng hệ thống cũ với hệ thống mới.

2.6 Basic element – những thành phần cơ bản

- **Entity**
    - Trong DDD, việc quan trọng cần phải làm là mô hình hóa các domain để cả dev lẫn domain expert đều nắm được. Và để mô hình hóa thì thành phần không thể thiếu là các entity (đối tượng). Tất cả các domain đều phải có đối tượng cụ thể. Entity trong DDD phải được định danh (có ID) và định danh phải bất biến và duy nhất trong toàn bộ hệ thống. Việc phân biệt các thực thể là rất quan trọng. Việc chia hệ thống dựa theo các domain sẽ tạo ra việc đối tượng trong nhiều domain là thực chất là một đối tượng. Việc gắn ID sẽ giúp cho xác định đối tượng có là một hay không trở nên đơn giản hơn. Chúng ta không thể nào phân biệt dựa trên các thuộc tính của đối tượng đó mà cần phải có thuộc tính đặc thù nhất gắn liền với đối tượng.
    - Bên cạnh đó, vì cùng một đối tượng có thể nằm ở nhiều domain khác nhau và các domain này độc lập với nhau nên entity cần thiết phải chứa logic của riêng nó để có thể sử dụng trên nhiều domain và mỗi domain không cần phải quan tâm đến những logic đó. Điều này đảm bảo cho tính nhất quán của hệ thống và giảm bớt những xử lý dư thừa. Thêm một điều cần lưu ý, entity cũng cần phải có life cycle (creation and deletion) trong chính bản thân nó. Vì việc được sử dụng trên nhiều

domain độc lập sẽ không đảm bảo việc tạo hoặc xóa bỏ đối tượng đó đúng cách.

- **Value Object**
  - Không phải bất cứ đối tượng nào cũng bắt buộc phải có định danh. Mà không có định danh nghĩa là nó không phải là entity. Vậy nó là gì? Trong DDD, đối tượng không có định danh được gọi là value object. Những đối tượng này thường là để lưu giá trị và không cần phải phân biệt với nhau. Chỉ đơn giản là lưu giá trị. Ví dụ như một voucher giảm giá, người ta chỉ quan tâm đến giá trị giảm giá được in trên trên voucher chứ chả ai cần biết mã voucher hay cái gì khác. Và những thông tin đó cũng chẳng cần thiết, voucher chỉ cần lưu số tiền giảm giả và thế là đủ. Value object cũng vậy, chỉ cần lưu giá trị là đủ. Value object đóng vai trò làm đối tượng giữ các giá trị của các setting, nó kết hợp với entity để giúp entity có thể phân biệt nhau trong từng domain. Lấy ví dụ voucher giảm giá 1 triệu, nó sẽ không có ý nghĩa nếu không kết hợp với một hóa đơn mua hàng (entity). Hay một địa chỉ nào đó, nếu đứng một mình thì chẳng mang lại nghĩa gì nhưng nếu gắn với một entity cụ thể như người (địa chỉ liên hệ) hay đơn hàng (địa chỉ giao/nhận hàng). Qua ví dụ này, ta có thể nhận ra một tính chất nữa của value object là tính bất biến. Một khi nó được tạo ra thì nó không thể thay đổi trong vòng đời của nó. Voucher 1 triệu thì nó mãi là voucher 1 triệu cho tới khi nó hết hạn áp dụng, không thể đổi thành 2 triệu hay 2 trăm được.
  - Điều cần lưu ý khi sử dụng value object là chỉ đơn thuần là lưu giá trị và không có định danh. Tức là chúng sẽ như nhau khi có giá trị giống nhau, không có sự khác biệt. Chẳng hạn như 2 địa chỉ nhà giống nhau từng con phố ngõ hẻm thì là như nhau. Hay hai voucher giảm giá 1 triệu là như nhau. Vì vậy hãy chắc chắn rằng tất cả các value object đều bình đẳng với nhau vì không có sự khác biệt khi các thuộc tính

đều có cùng giá trị. Đây cũng là điểm phân biệt giữa entity và value object và từ đó có thể đưa ra quyết định chính xác object là entity hay value object.

- **Aggregates**
    - Một Aggregate là một nhóm các entity, nhóm này có thể được xem như là một đơn vị thống nhất. Các entity trong nội bộ aggregate có thể tự do tham chiếu đến nhau tuy nhiên muốn tham chiếu đến đối tượng nằm ở aggregate khác thì thì nó phải thông qua gốc của aggregate (aggregate root). Điều này giúp giảm bớt sự phụ thuộc giữa các entity trong hệ thống. Thay vì chúng phải kết nối lẫn nhau thì bây giờ chúng chỉ cần liên kết thông qua các aggregate root. Giảm đi vô số liên kết tức là giảm đi vô số phụ thuộc. Điều này giúp tăng khả năng linh hoạt của hệ thống, thứ mà đang trở thành yêu cầu hàng đầu trong phát triển ứng dụng ngày nay.
    - Bên cạnh đó, để các entity có thể tham chiếu đến nhau trong aggregate thì nhất thiết phải có logic xử lý nằm ở aggregate. Nên phải chú ý rằng trong một aggregate, phải đảm bảo có đầy đủ các logic liên quan đến tất cả entity chứa trong nó. Từ đó các entity mới có thể giao tiếp với nhau. Và những aggregate khác muốn tác động đến các entity này chỉ cần sử dụng các logic đó mà thôi, không nhất thiết phải tạo thêm logic chỉ đích danh chính entity đó. Tức là chỉ cần giao tiếp với aggregate là có thể giao tiếp với tất cả các entity có trong aggregate đó.
- **Domain Services:** là nơi chứa các logic quá phức tạp trong mà trong phạm vi entity không thể làm được. Service cũng là nơi chứa các logic làm việc trên nhiều aggregate.

2.7 Kết luận chương

Microservice đang là một xu hướng lập trình chủ chốt cho mọi ứng dụng lớn và DDD chính là khởi đầu cho việc triển khai microservice. DDD cung cấp một phương pháp để phát triển ứng dụng phần mềm tập trung vào việc hiểu và mô hình hóa rõ ràng các khái niệm trong lĩnh vực. Nó tạo điều kiện cho việc xây dựng ứng dụng phù hợp với yêu cầu doanh nghiệp và mang lại sự linh hoạt và dễ dàng mở rộng trong quá trình phát triển.

CHƯƠNG 3: ỨNG DỤNG THỰC TẾ

Mô tả bài toán

"Xây dựng Website lấy ý kiến phản hồi của sinh viên về công tác giảng dạy".

Dữ liệu đầu vào:

Lấy dữ liệu **Sinh viên**, **giảng viên** và **các lớp môn học, thời khóa biểu…** trong học kỳ từ hệ thống Quản lý đào tạo Edu của nhà Trường thông qua API để phục vụ lấy ý kiến về công tác giảng dạy. Xác thực dữ liệu về thông tin của giảng viên và sinh viên với **Clerk** để phục vụ cho việc đăng nhập vào hệ thống cho người dùng.

Nghiệp vụ: Thực hiện lấy ý kiến trực tuyến về công tác giảng dạy theo ***Số 73/QĐ-HĐT về Quyết định Ban hành Quy chế đánh giá công tác giảng dạy*.** Lấy ý kiến về công tác giảng dạy của **lớp môn học** trong kỳ.

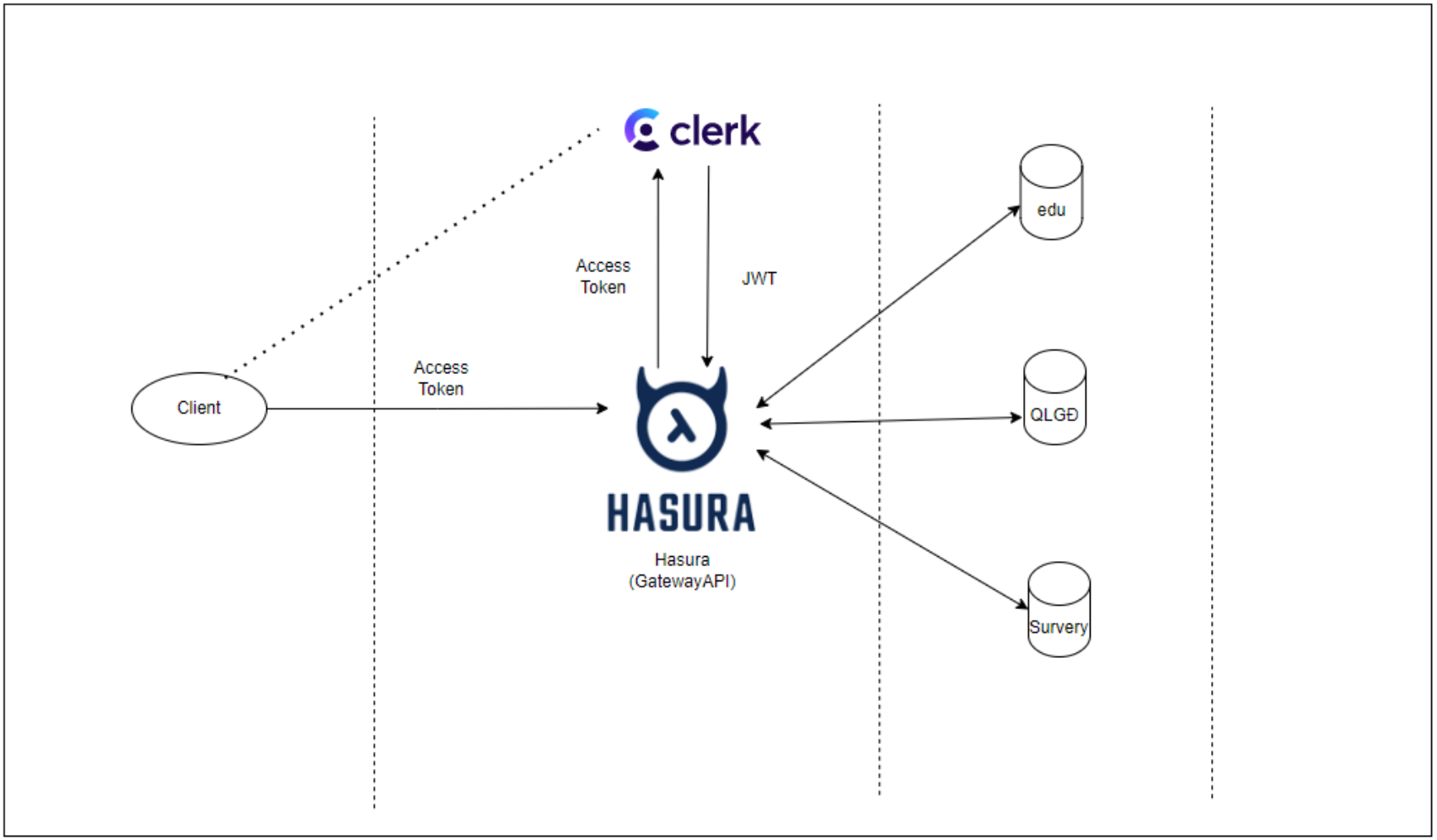
Dữ liệu đầu ra: Xuất ra **Điểm trung bình** và **Xếp loại** về công tác giảng dạy mỗi giảng viên của một lớp môn học.

3.1 Xây dựng theo Microservices

Bởi vì hệ thống có sẵn từ các ngồn dữ liệu thuộc nhà trường đang đi theo hướng Microservices tức mỗi một Module sẽ chia ra làm một phần riêng biệt với nhau để hoạt động một cách độc lập. Vì vậy "Website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU" cũng sẽ đi theo hướng Microservices và việc này sẽ giúp cho việc vận hành hệ thống được tốt hơn và khi một phần lỗi thì các phần khác vẫn có thể hoạt động bình thường không ảnh hưởng đến nhau. Điều này cũng sẽ giúp cho việc trao đổi dữ liệu giữa các hệ thống khác của nhà trường trong tương lai được thuận tiện hơn.

Trong bài đồ án này em đã vận dụng được kiến thức về Microservice để xây dựng lên trang Web. Những dữ liệu về các lớp môn học phục vụ cho việc phản hồi công tác giảng dạy cũng như thông tin về sinh viên tham gia lớp môn học được lấy từ các phần mềm khác nhau như: Edu, Quản lý giảng đường. Vì lấy dữ liệu để phục vụ cho việc phản hồi công tác giảng dạy từ nhiều nguồn khác nhau nên em sẽ sử dụng REST API để lấy dữ liệu về cơ sở dữ liệu của phần mềm.

- **Mô hình hệ thống**

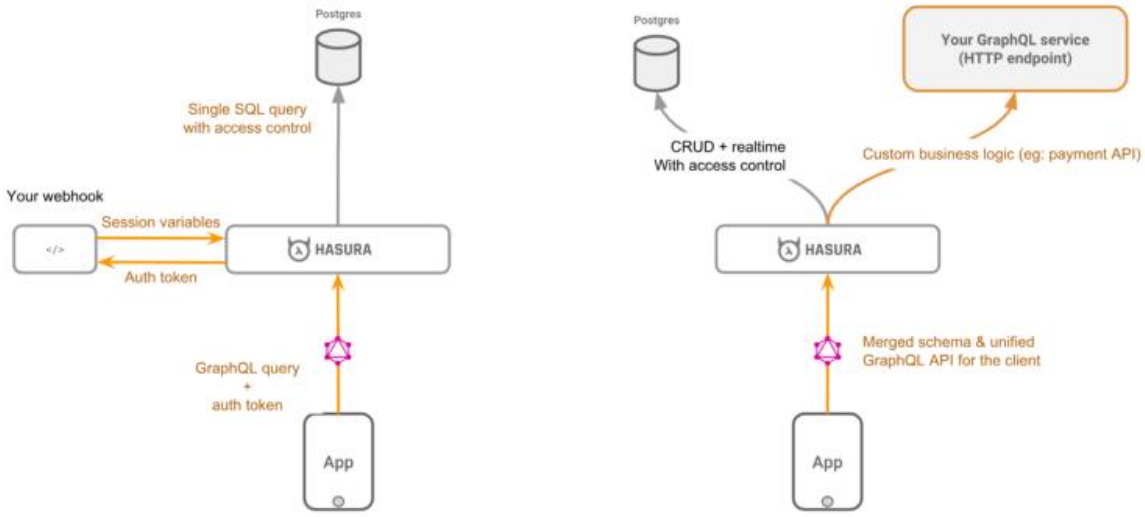


Hình 3.1.1 Mô hình hệ thống

Theo như hình trên, em có sử dụng một bên ứng dụng thứ ba để rút ngắn quá trình thiết lập API cũng như hệ thống đăng nhập đó là Hasura và Clerk.

- **Hasura** là một phần mềm mã nguồn mở phục vụ cho việc xây dựng web APIs với GraphQL. Hasura cung cấp nhiều tính năng hữu ích khác nhau, bao gồm quản lý phiên bản, quyền truy cập dựa trên vai trò, kiểm tra xung đột và đồng bộ dữ liệu thời gian thực. Nó giúp chúng ta xây dựng ứng dụng một cách nhanh chóng và linh hoạt mà không cần viết code quá nhiều và phức tạp ở phía máy chủ. Hasura cũng hỗ trợ **Clerk**, giúp **Clerk** có thể tạo dựng các JWT theo tiêu chuẩn của Hasura để phục vụ cho quá trình xác thực người dùng sau này.

Hình 3.1.2 Hasua



Hình 3.1.3 Tổng quan hasura

Em đã sử dụng hasura để tạo lên được các endpoints API cho hệ thống:

DETAILS	ENDPOINT	METHODS	MODIFY
Lấy thông tin danh sách lớp môn học chờ duyệt trong kỳ	https://relieved-walrus-48.hasura.app/api/rest/approve-subject/:hocky/:namhoc › GraphQL Request	GET	Delete Edit
Lấy danh mục cán bộ trong trường để phân công dự giờ	https://relieved-walrus-48.hasura.app/api/rest/assign-staff › GraphQL Request	GET	Delete Edit
Lấy danh sách môn học phân công của trưởng khoa	https://relieved-walrus-48.hasura.app/api/rest/assign-subject/:khoa/:hocky/:namhoc › GraphQL Request	GET	Delete Edit
Cập nhật thông tin giảng viên dự giờ	https://relieved-walrus-48.hasura.app/api/rest/assigned-teacher › GraphQL Request	POST	Delete Edit
Kiểm tra đã khởi tạo đợt đánh giá cho kỳ	https://relieved-walrus-48.hasura.app/api/rest/check-init/:hocky/:namhoc › GraphQL Request	GET	Delete Edit
Lấy điểm tổng kết cuối cùng của tất cả sinh viên cho môn học	https://relieved-walrus-48.hasura.app/api/rest/final-student › GraphQL Request	POST	Delete Edit
Lấy điểm tổng kết cuối cùng của tất cả giảng viên dự giờ cho môn học	https://relieved-walrus-48.hasura.app/api/rest/final-teacher › GraphQL Request	POST	Delete Edit
Lấy id câu hỏi cho giáo viên theo kỳ đánh giá	https://relieved-walrus-48.hasura.app/api/rest/gv-batch-question/:hocky/:namhoc › GraphQL Request	GET	Delete Edit
Khởi tạo đợt đánh giá	https://relieved-walrus-48.hasura.app/api/rest/init-survey › GraphQL Request	POST	Delete Edit
Lấy những môn được phân bổ dự giờ của giảng viên	https://relieved-walrus-48.hasura.app/api/rest/is-asigned › GraphQL Request	POST	Delete Edit

Hình 3.1. 4 Hình ảnh các EndPoints API

GET https://edu-survey.hasura.app/api/rest/**assign-staff**

Lấy danh mục cán bộ/giảng viên phục vụ cho việc phân công dự giờ

Request

Headers

authorization string required

JWT

Default: Bearer

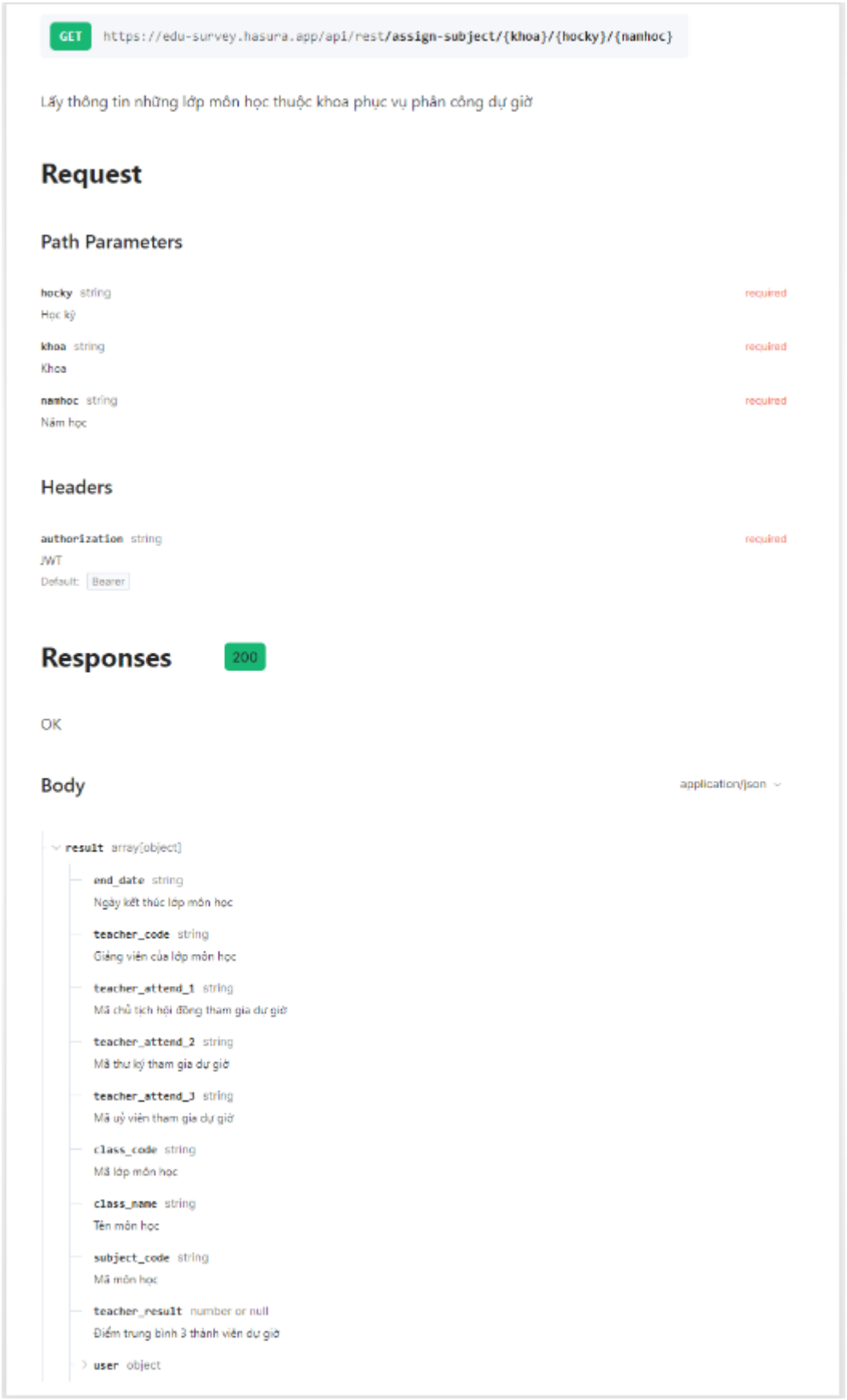
Responses 200

OK

Body application/json

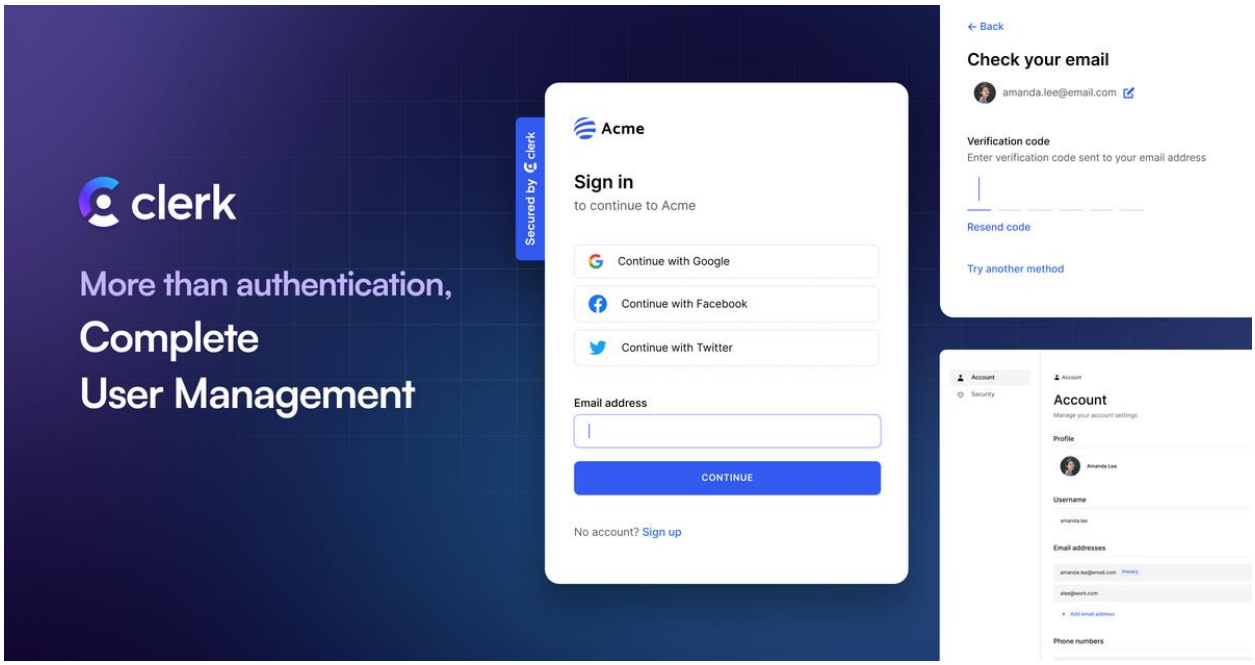
- **result** array[object]
  - **name** string
    Tên cán bộ/giảng viên
  - **code** string
    Mã cán bộ/giảng viên
  - **ten_hoc_vi** string or null
    Tên học vị
  - **khoa_gv** string or null
    Khoa

Hình 3.1. 5 Documents API danh mục cán bộ/giảng viên



Hình 3.1. 6 Document API danh sách lớp môn học thuộc khoa

– **Clerk** là một ứng dụng cho phép chúng ta quản lý và xác thực tài khoản của người dùng khi truy cập vào hệ thống. Clerk cũng chính là nơi chúng ta tạo ra những JWT đi theo API của hasura nhằm xác thực rằng người ấy có quyền gì, được phép làm gì trong hệ thống của chúng ta.



Hình 3.1.7 Clerk

Theo như hình 3.1.1 về mô hình hệ thống của phần mềm, ta có thể thấy **Clerk** sẽ đóng vai trò như một chức năng đăng nhập vào hệ thống và từ đó để có thể phản hồi về công tác giảng dạy thì ta cần phải truy cập thông tin đánh giá qua API và đây cũng chính là lúc **Clerk** sẽ tạo cho người dùng một mã JWT để gửi kèm theo API giúp quá trình đánh giá được ghi nhận vào cơ sở dữ liệu.

3.2 Ứng dụng DDD vào phân tích thiết kế hệ thống

Ở bài báo cáo đồ án này, em xin phép chỉ trình bày quá trình đánh giá của sinh viên trong hệ thống.

– **Một số dữ liệu được trích từ Số 73/QĐ-HĐT về Quyết định Ban hành Quy chế đánh giá công tác giảng dạy** [8]

Phản hồi của sinh viên chiếm **40%**, điểm sinh viên phản hồi của từng học phần($\mathbf{d_1}$) là điểm phản hồi trung bình theo các tiêu chí, có giá trị từ 1 đến 5 điểm tương ứng với 5 mức độ của thang phản hồi. Điểm phản hồi được làm tròn đến 2 chữ số thập phân.

Đánh giá của đồng nghiệp chiếm **40%,** đồng nghiệp đánh giá công tác giảng dạy thông qua việc dự giờ lên lớp trong thời gian giảng dạy theo các tiêu chí quy định tại Phụ lục 02 của quy chế. Điểm đồng nghiệp đánh giá công tác giảng dạy của từng học phần (**d_2**) là điểm đánh giá trung bình theo các tiêu chí và có giá trị từ 0 đến 10 điểm. Điểm đánh giá được làm tròn đến 2 chữ số thập phân.

Đánh giá của các đơn vị quản lý đào tạo chiếm **20%,** việc đánh giá được thực hiện trên cơ sở kết quả theo dõi thực hiện giờ lên lớp, chấp hành các quy định liên quan đến công tác giảng dạy của giảng viên. Điểm đánh giá việc thực hiện quy định về công tác giảng dạy (**d_3**) căn cứ theo tiêu chí quy định tại Phụ lục 03 của quy chế và có giá trị từ 0 đến 10 điểm. Điểm đánh giá được làm tròn đến 2 chữ số thập phân.

Điểm phản hồi/đánh giá chung về công tác giảng dạy của một học phần (**d**) có giá trị từ 0 đến 10 điểm được tính như sau:

$$\mathbf{d = 0{,}8d_1 + 0{,}4d_2 + 0{,}2d_3}$$

Công tác giảng dạy của mỗi giảng viên ở từng học phần được xếp loại theo các mức độ A, B, C, D, E khi đạt điểm đánh giá (**d**) như sau:

Điểm đánh giá	0,00÷3,99	4,00÷5,99	6,00÷7,99	8,00÷8,99	9,00÷10,00
Xếp loại	**E**	**D**	**C**	**B**	**A**

PHỤ LỤC 01. TIÊU CHÍ SINH VIÊN PHẢN HỒI VỀ CÔNG TÁC GIẢNG DẠY

Giảng viên, môn dạy	a. Giảng viên giải thích rõ ràng về nội dung học tập					b. Giảng viên có phản hồi hữu ích đối với việc học tập của em					c. Giảng viên khuyến khích, tạo động lực cho em học tập					d. Giảng viên nỗ lực để thấu hiểu khó khăn của cá nhân em khi học môn này					e. Giảng viên sử dụng nhiều tình huống thực tế để sinh viên thảo luận					f. Giảng viên nỗ lực để tăng sự hấp dẫn của môn học					g. Nói chung, em hài lòng với chất lượng giảng dạy ở môn học này				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Nguyễn Văn A …																																			
Nguyễn Thị B …																																			

Ghi chú: 1. Rất không đồng ý 2. Không đồng ý 3. Lưỡng lự 4. Đồng ý 5. Rất đồng ý

Hình 3.2.1 Tiêu chí sinh viên phản hồi

PHỤ LỤC 02. TIÊU CHÍ ĐỒNG NGHIỆP ĐÁNH GIÁ CÔNG TÁC GIẢNG DẠY

Tiêu chí	Khung đánh giá công tác giảng dạy				
	0÷2,0 điểm	2,0÷4,0 điểm	4,0÷6,0 điểm	6,0÷8,0 điểm	8,0÷10,0 điểm
1. Chuẩn đầu ra (CĐR)	Không rõ các CĐR sinh viên (SV) cần phát triển là gì, hoặc Không có minh chứng là SV đã đạt được CĐR của bài học	Đại đa số SV mới đạt được các CĐR ở mức độ ghi nhớ, làm lại là chính	Có minh chứng cho thấy khoảng 50% số SV đã đạt được các CĐR ở các cấp độ khác nhau của bài học	Có minh chứng cho thấy đại đa số SV đã đạt được các CĐR ở các cấp độ khác nhau của bài học	Có minh chứng cho thấy đại đa số SV đạt được các năng lực theo CĐR ở bậc cao (phân tích, đánh giá, vv..)
2. Nội dung bài giảng	GV còn mắc lỗi, không nắm chắc kiến thức, kỹ năng (KT,KN) hoặc không đảm bảo đủ KT,KN của bài học	GV cơ bản đảm bảo đầy đủ và chính xác về KT,KN cần có trong bài giảng nhưng chưa làm rõ sự kết nối của các KT,KN đó	Có minh chứng cho thấy khoảng 50% SV đã nắm chắc KT,KN và sự kết nối giữa những KT,KN đó; tài liệu học tập đáp ứng các yêu cầu chung của môn học	Có minh chứng cho thấy đại đa số SV đã nắm chắc KT,KN và sự kết nối giữa những KT,KN đó; tài liệu học tập đáp ứng các yêu cầu chung của môn học	Có minh chứng cho thấy các SV có thể ứng dụng tốt KT,KN ở trong và ngoài môn học; tài liệu học tập phù hợp, phong phú, hấp dẫn
3. Khuyến khích và hỗ trợ SV học tập	GV độc giảng gần như toàn bộ tiết dạy hoặc Nhiều SV làm việc riêng, không chú ý vào bài giảng	GV chủ yếu kiểm tra KT, KN và tìm cách hỗ trợ một số nhóm đối tượng SV nhất định nhưng nhiều SV vẫn thụ động trong học tập	Có một vài câu hỏi, cách thức có hiệu quả được GV sử dụng để kiểm tra KT, KN của các nhóm SV khác nhau	Nhiều câu hỏi, nhiều cách thức có hiệu quả được GV sử dụng để kiểm tra KT, KN của các nhóm SV khác nhau; Giờ học vui vẻ, sôi động	SV chủ động, tích cực học tập, có nhiều cơ hội tham gia thực hành, tranh luận, kết luận về các nội dung của bài giảng

Hình 3.2.2 Tiêu chí đồng nghiệp đánh giá

PHỤ LỤC 03. TIÊU CHÍ CÁC ĐƠN VỊ QUẢN LÝ ĐÁNH GIÁ THỰC HIỆN CÁC QUY ĐỊNH VỀ CÔNG TÁC GIẢNG DẠY

Tiêu chí	Khung đánh giá				
	0÷2,0 điểm	2,0÷4,0 điểm	4,0÷6,0 điểm	6,0÷8,0 điểm	8,0÷10,0 điểm
Thực hiện nội quy, quy định, quy chế	- Có trên 3 lần vi phạm giờ lên lớp/học kỳ hoặc/và trên 01 lần bỏ dạy không có lý do chính đáng - Có vi phạm nội quy, quy định, quy chế gây ra hậu quả xấu, làm lãnh đạo Trường phải nhắc nhở hoặc tham gia xử lý vụ việc	- Có 2 đến 3 lần vi phạm giờ lên lớp/học kỳ hoặc có 01 lần bỏ dạy không có lý do chính đáng - Có vi phạm nội quy, quy định, quy chế nhưng chưa để lại hậu quả xấu, cá nhân đã tự khắc phục tốt vụ việc	- Có 01 lần vi phạm giờ lên lớp; - Cơ bản thực hiện đúng nội quy, quy định, quy chế	- Không vi phạm giờ lên lớp; - Thực hiện đúng và đầy đủ nội quy, quy định, quy chế trong thời hạn quy định; không để phát sinh phàn nàn của sinh viên	- Không vi phạm giờ lên lớp; - Thực hiện đúng và đầy đủ nội quy, quy định, quy chế trong thời hạn quy định, được nhiều người đánh giá cao

Hình 3.2.3 Tiêu chí của đơn vị quản lý

KHOA, BỘ MÔN .....

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BIÊN BẢN ĐÁNH GIÁ CÔNG TÁC GIẢNG DẠY

Tên học phần được đánh giá: .....

Giảng viên giảng dạy: .....

Phòng học: .....

Ngày, giờ thực hiện đánh giá: .....

Thành viên Hội đồng đánh giá:

1. .....

2. .....

3. .....

Nhận xét và điểm đánh giá:

Thành viên hội đồng đánh giá	Nhận xét theo từng tiêu chí đánh giá	Điểm đánh giá
Nguyễn Văn A	TC1:	
	TC2:	
	TC3:	
	TC4:	
	TC5:	
Nguyễn Thị B	TC1:	
	TC2:	
	TC3:	
	TC4:	
	TC5:	
Nguyễn Thị C	TC1:	
	TC2:	
	TC3:	
	TC4:	
	TC5:	

Điểm đánh giá trung bình của các thành viên: ..... điểm

Hải Phòng, ngày ... tháng ... năm 2023

THÀNH VIÊN HỘI ĐỒNG ĐÁNH GIÁ

(ký và ghi rõ họ tên)

Hình 3.2.4 Biên bản đánh giá công tác giảng dạy

- **Quy trình thực hiện của từng đối tượng**

Admin	Bước 1: Khởi tạo đợt đánh giá cho kỳ hiện tại.
	Bước 2: Lấy dữ liệu các lớp môn học trong kỳ từ QLGD, EDU để phục vụ quá trình lấy ý kiến.
	Bước 3: Xuất báo cáo tình trạng lấy ý kiến của các lớp môn học trong kỳ ra file CSV.

Quản lý	Bước 1: Tạo bộ câu hỏi cho giảng viên và sinh viên trong đợt.
	Bước 2: Lựa chọn các lớp môn học lấy ý kiến trong kỳ.
	Bước 3: Duyệt cho phép những lớp môn học được lấy ý kiến trong kỳ.

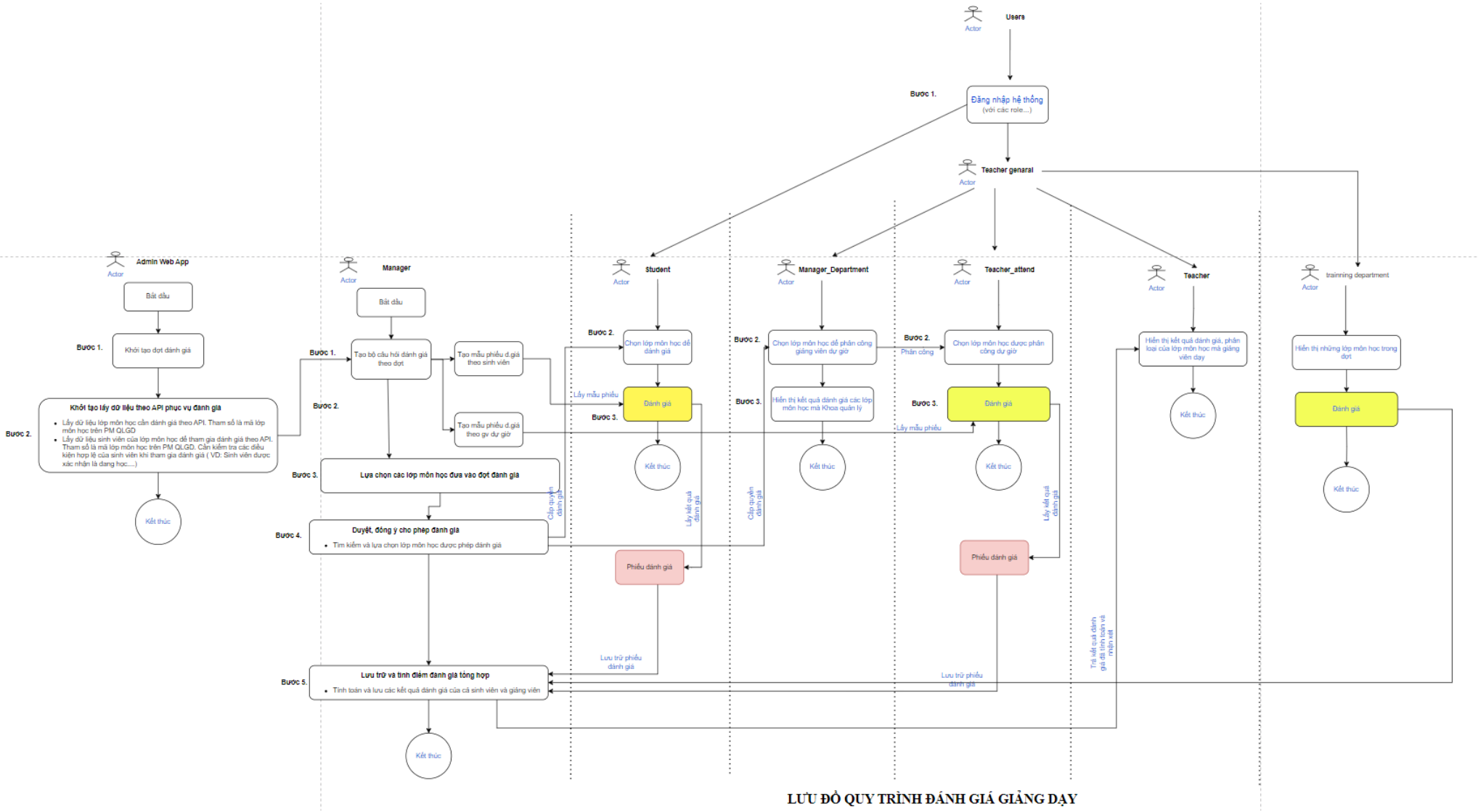
Sinh viên	Bước 1: Lựa chọn những lớp môn học được lấy ý kiến trong kỳ.
	Bước 2: Tiến hành cho ý kiến đánh giá.

Giảng viên	Bước 1: Xem thông tin được đánh giá của bản thân cho các lớp môn học.

Trưởng khoa	Bước 1: Lựa chọn những lớp môn học thuộc khoa để phân công dự giờ.
	Bước 2: Lựa chọn những thành viên hội đồng dự giờ.
	Bước 3: Tiến hành phân công dự giờ.
	Bước 4: Xuất thông tin phân công dự giờ của các lớp môn học thuộc khoa ra file CSV.

Giảng viên dự giờ	Bước 1: Xem những lớp môn học được phân công dự giờ.
	Bước 2: Lựa chọn lớp môn học.
	Bước 3: Tiến hành đánh giá.

Đơn vị quản lý đào tạo	Bước 1: Lựa chọn những lớp môn học được lấy ý kiến trong kỳ.
	Bước 2: Tiến hành cho ý kiến đánh giá.



Hình 3.2.5 Lưu đồ quy trình đánh giá giảng dạy

– **Ứng dụng DDD để thiết kế cơ sở dữ liệu**

- Thiết kế cơ sở dữ liệu bao gồm 4 bước:

Bước 1: Liệt kê, chính xác hoá và chọn lọc thông tin.

Tên chính xác của các chỉ mục đặc trưng	Viết gọn tên đặc trưng	Đánh dấu loại đặc trưng ở mỗi bước		
		(1)	(2)	(3)
Sinh viên	Sinh viên		X	
Giảng viên giảng dạy	Họ tên giảng viên		X	
Thành viên hội đồng đánh giá	Họ tên thành viên hội đồng đánh giá		X	
Đơn vị quản lý đào tạo	Đơn vị quản lý đào tạo		X	
Tên học phần	Tên lớp môn học		X	
Tiêu chí đánh giá	Tiêu chí		X	
Ngày, giờ thực hiện đánh giá	Ngày đánh giá			X
Điểm trung bình đánh giá	Điểm trung bình	X		

Bước 2: Xác định thực thể, thuộc tính và định danh

Thuộc tính tên gọi tìm được	Thực thể tương ứng	Thuộc tính của thực thể	Định danh
Sinh viên	NGƯỜI DÙNG	Mã người dùng	Thêm vào
		Họ tên người dùng	
		Trưởng khoa	Thêm vào
		Khoa	Thêm vào
		Tên học vị	Thêm vào
		Mã học vị	Thêm vào
		Vai trò	
Họ tên giảng viên	GIẢNG VIÊN	Mã giảng viên	Thêm vào
		Họ tên giảng viên	
		Khoa	Thêm vào
Họ tên thành viên hội đồng đánh giá	NGƯỜI DÙNG	Mã người dùng	Thêm vào
		Họ tên người dùng	
		Khoa	Thêm vào
		Tên học vị	Thêm vào
		Mã học vị	Thêm vào
		Vai trò	
Đơn vị quản lý đào tạo	NGƯỜI DÙNG	Mã người dùng	Thêm vào
		Họ tên người dùng	
		Tên học vị	Thêm vào
		Mã học vị	Thêm vào
		Vai trò	
Tiêu chí đánh giá	TIÊU CHÍ	Mã tiêu chí	Thêm vào
		Tên tiêu chí	
		Thang điểm	Thêm vào
Lớp môn học	LỚP MÔN HỌC	Mã lớp môn học	Thêm vào
		Mã môn học	
		Tên lớp môn học	
		Tên môn học	Thêm vào

NHẬN XÉT: SINH VIÊN và THÀNH VIÊN HỘI ĐỒNG và ĐƠN VỊ QUẢN LÝ ĐÀO TẠO đều là thực thể NGƯỜI DÙNG nên sẽ có thuộc tính **Vai trò** để phân biệt. Thuộc tính chung là **Mã người dùng** và **Họ tên người dùng**. Thuộc tính riêng là **Trưởng khoa** và **Khoa** và **Tên học vị** và **Mã học vị**.

Bước 3: Xác định các mối quan hệ và thuộc tính

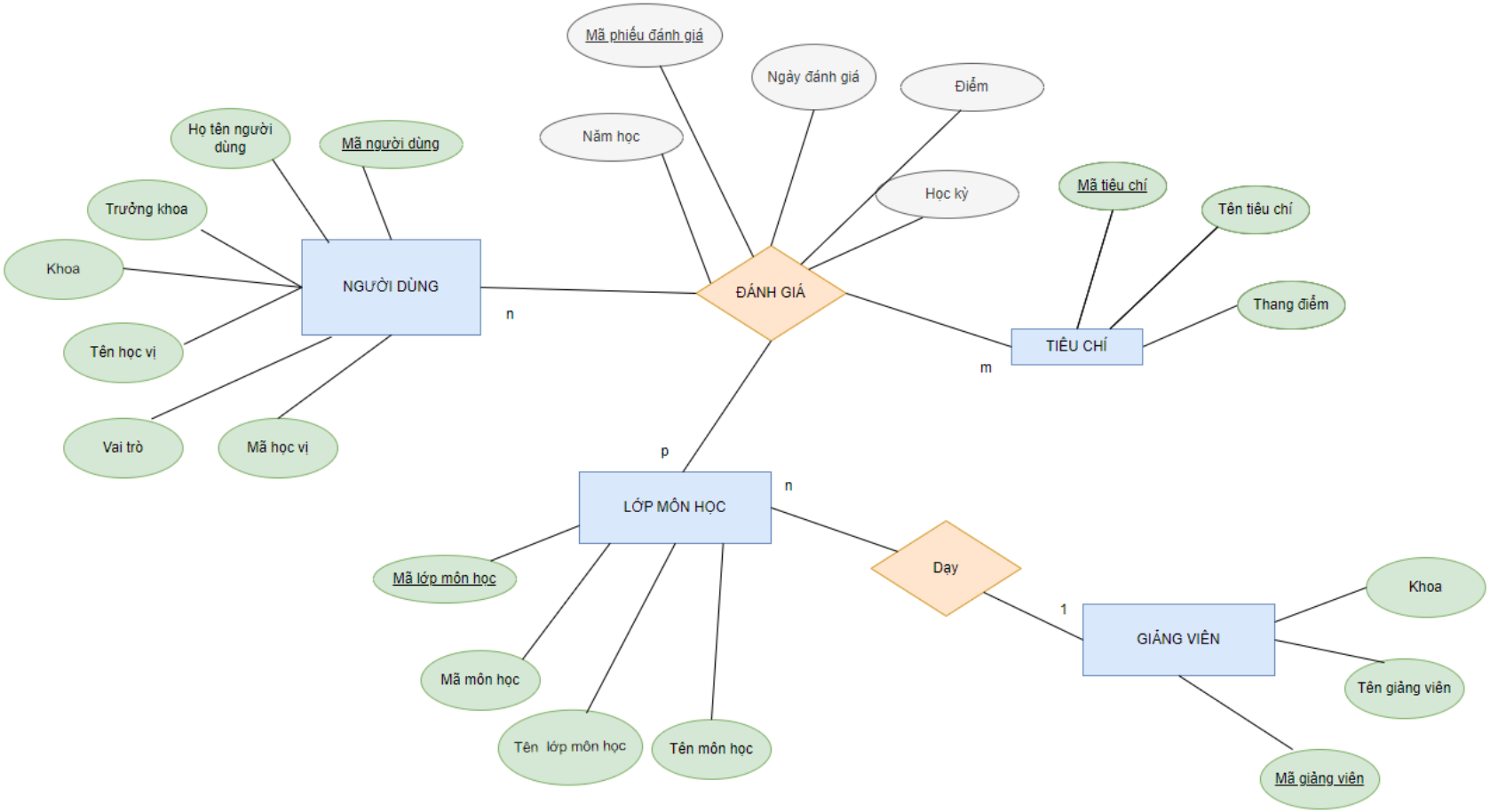
a) Động từ tìm được là: ĐÁNH GIÁ

Câu hỏi cho động từ ĐÁNH GIÁ	Câu trả lời là	
	Thực thể	Thuộc tính
Ai đánh giá?	NGƯỜI DÙNG	
Đánh giá gì?	TIÊU CHÍ	
Đánh giá cho cái gì?	LỚP MÔN HỌC	
Đánh giá lúc nào?		Ngày đánh giá
Sử dụng bằng cách nào?		Mã phiếu đánh giá
Đánh giá cho đợt nào?		Học kỳ
		Năm học
Đánh giá bằng cách nào?		Điểm

b) Xét các mối quan hệ phụ thuộc, sở hữu

Xét từng cặp thực thể		Mối quan hệ	Thuộc tính
GIẢNG VIÊN	LỚP MÔN HỌC	Dạy	

Bước 4: Vẽ biểu đồ mô hình thực thể ER



Hình 3.2.6 Mô hình ER

Chuyển mô hình ER sang mô hình quan hệ:

NGƯỜI DÙNG						
Mã người dùng	Họ tên người dùng	Khoa	Trưởng khoa	Tên học vị	Mã học vị	Vai trò

TIÊU CHÍ		
Mã tiêu chí	Tên tiêu chí	Thang điểm

GIẢNG VIÊN		
Mã giảng viên	Tên giảng viên	Khoa

LỚP MÔN HỌC				
Mã lớp môn học	Mã môn học	Tên lớp môn học	Tên môn học	**Mã giảng viên**

NGƯỜI DÙNG đánh giá **TIÊU CHÍ** cho **LỚP MÔN HỌC**							
Mã phiếu đánh giá	Năm học	Học kỳ	Ngày đánh giá	Điểm	**Mã tiêu chí**	**Mã người dùng**	**Mã lớp môn học**

Vì dữ liệu lớp môn học được lấy hoàn toàn từ API của hệ thống quản lý của trường và một lớp môn học chỉ có một giảng viên nên sẽ gộp bảng **GIẢNG VIÊN** vào bảng **LỚP MÔN HỌC** kèm thuộc tính khoa.

Để tổng kết 3 đầu điểm $\mathbf{d_1, d_2, d_3}$ phục vụ cho việc tổng kết điểm thì em sẽ tạo thêm một bảng để tính điểm trung bình từ các tiêu chí đánh giá của 1 sinh viên hoặc của 1 thành viên hội đồng là bảng **course_respond**.

Tiếp theo sẽ phải tính điểm trung bình của tất cả sinh viên trong lớp môn học và từ 3 thành viên hội đồng đánh giá thì em sẽ thêm 2 cột vào bảng **LỚP MÔN HỌC** đã lấy dữ liệu từ API trước đó là **student_result** (Điểm sinh viên), teacher_result (Điểm thành viên hội đồng). Em thêm 3 cột **teacher_attend_1** (Mã chủ tịch hội đồng), **teacher_attend_2** (Thư ký) và **teacher_attend_3** (Uỷ viên) để truy vết điểm đánh giá của thành viên hội đồng. Đối với đơn vị quản lý đào tạo vì chỉ có một tiêu chí đánh giá nên em sẽ thêm 1 cột **qldt_result** (Điểm đơn vị quản lý đào tạo).

Tất cả việc tổng hợp điểm em sẽ sử dụng **TRIGGER** của postgreSQL để tự động tổng hợp điểm theo công thức và sau khi đã có điểm đánh giá chung thì sẽ tự động tính ra được xếp loại.

Trước khi người dùng có thể đánh giá thì hệ thống phải tạo trước các dòng dữ liệu cho người dùng nhằm tránh việc khi thực hiện **INSERT** sẽ làm lặp các dữ liệu đánh giá. Thay vì đó người dùng chỉ có thể **UPDATE** dòng dữ liệu mà hệ thống đã tạo ra trước đó. Chính vì lý do đó để liên kết dữ liệu giữa các bảng với nhau thì ta cần những Value Object**: hocky, namhoc, class_code, subject_code** để **JOIN** các bảng có liên quan lại với nhau.

- Xác định các entities

  user: Người dùng

  level_point: Thang điểm tiêu chí

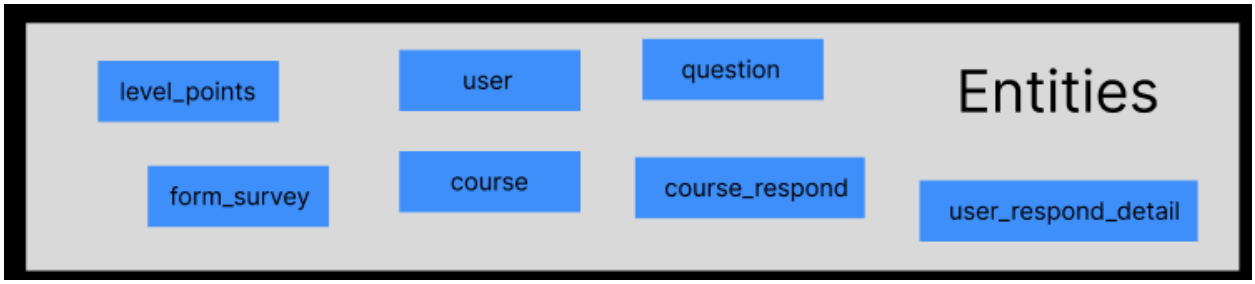
  question: Tiêu chí đánh giá

  form_survey: Danh sách câu hỏi cho từng đợt

  course: Thông tin đánh giá của lớp môn học

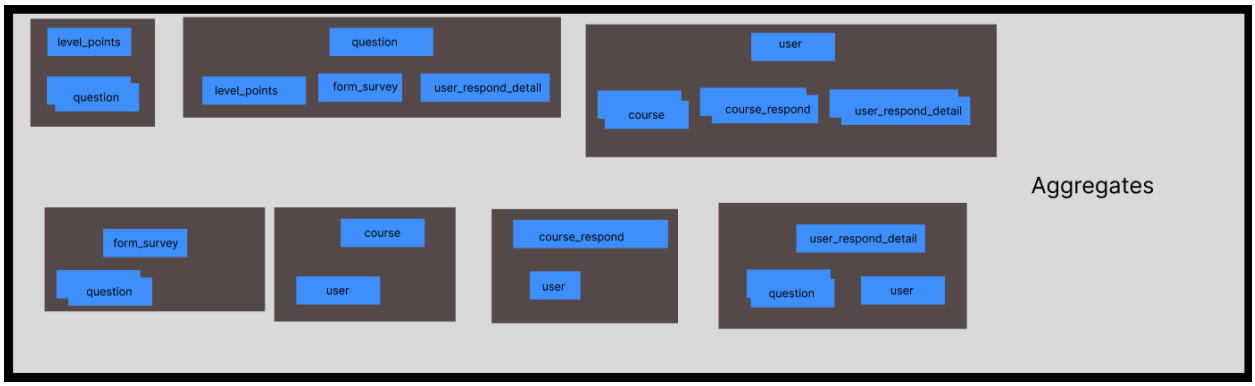
  course_respond: Thông tin đánh giá của người dùng

  user_respond_detail: Chi tiết thông tin đánh giá của người dùng



Hình 3.2.7 Xác định Entities

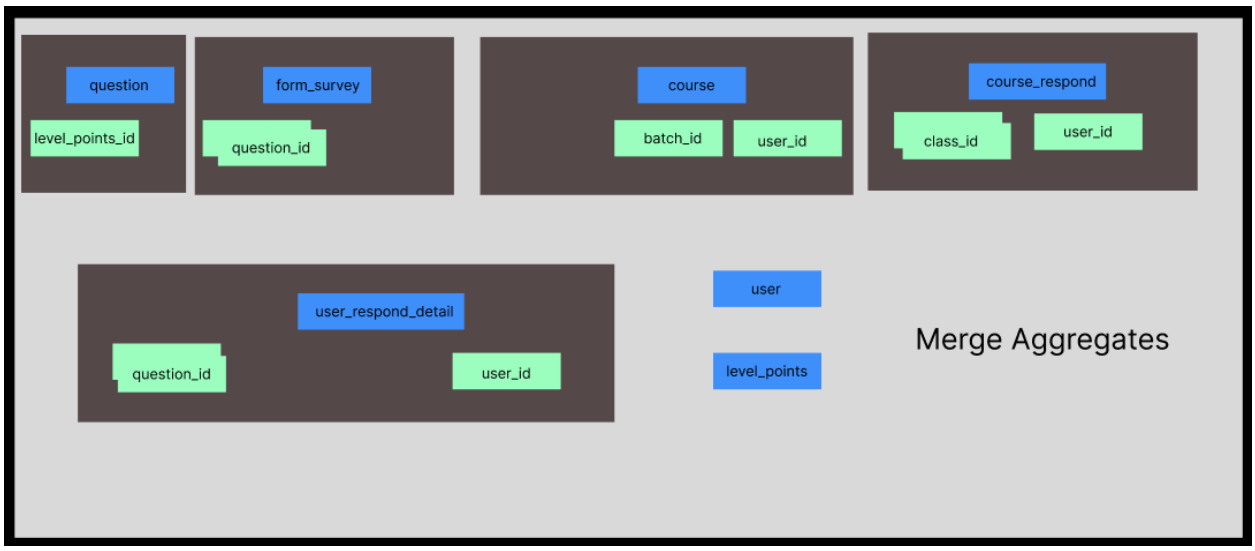
- Xác định Aggregates dựa trên mối liên hệ giữa các entity



Hình 3.2.8 Xác định Aggregates

- Gộp các Aggregate

  Những Aggregate có mối quan hệ một nhiều sẽ được gộp sang bên Aggregate nhiều và xoá liên kết ở Aggregate một.



Hình 3.2.9 Gộp Aggregates

- *Chuyển đổi từ Aggregate sang bảng cơ sở dữ liệu với Value Object*

Chi tiết bảng level_points:

Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
id	bigint	☐	bigint		☑
name	text	☐	text		☐
description	text	☑	text		☐
created_at	timestamp	☑	timestamp(6)	now()	☐
updated_at	timestamp	☑	timestamp(6)		☐

Hình 3.2.10 Bảng level_points

Chi tiết bảng questions:

Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
id	bigint	☐	bigint		☑
level_point_id	bigint	☐	bigint		☐
content	text	☐	text		☐
description	text	☑	text		☐
created_at	timestamp	☑	timestamp(6)	now()	☐
updated_at	timestamp	☑	timestamp(6)		☐

Hình 3.2.11 Bảng questions

Chi tiết bảng users:

9 / 14	Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
▶	id	bigint	☐	bigint		☑
	role_id	bigint	☐	bigint		☐
	name	text	☐	text		☐
	email	text	☑	text		☐
	phone	text	☑	text		☐
	code	text	☐	text		☐
	khoa_gv	text	☑	text		☐
	created_at	timestamp	☑	timestamp(6)	now()	☐
	updated_at	timestamp	☑	timestamp(6)		☐
	is_truong_khoa	boolean	☑	boolean	false	☐
	encoded_position	text	☑	text		☐
	ten_hoc_vi	text	☑	text		☐
	ma_hoc_vi	text	☑	text		☐
	class	text	☑	text		☐

Hình 3.2.12 Bảng users

Chi tiết bảng form_survey:

9 / 6	Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
▶	id	bigint	☐	bigint		☑
	question_id	bigint	☐	bigint		☐
	hocky	integer	☐	integer		☐
	namhoc	text	☐	text		☐
	created_at	timestamp	☑	timestamp(6)		☐
	updated_at	timestamp	☑	timestamp(6)		☐

Hình 3.2.13 Bảng form_survey

Chi tiết bảng course:

9 / 24	Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
▶	id	bigint	☐	bigint		☑
	hocky	integer	☐	integer		☐
	namhoc	text	☐	text		☐
	class_name	text	☑	text		☐
	class_code	text	☐	text		☐
	subject_code	text	☐	text		☐
	ten_khoa	text	☐	text		☐
	teacher_code	text	☐	text		☐
	total_student	integer	☑	integer		☐
	start_date	date	☑	date		☐
	end_date	date	☑	date		☐
	department_id	bigint	☑	bigint		☐
	teacher_attend_1	text	☑	text		☐
	teacher_attend_2	text	☑	text		☐
	teacher_attend_3	text	☑	text		☐
	status	boolean	☑	boolean	false	☐
	batch_created	timestamp	☑	timestamp(6)		☐
	result_evaluate	numeric	☑	numeric		☐
	created_at	timestamp	☑	timestamp(6)	now()	☐
	updated_at	timestamp	☑	timestamp(6)		☐
	student_result	numeric	☑	numeric(38, 2)		☐
	teacher_result	numeric	☑	numeric(38, 2)		☐
	qldt_result	numeric	☑	numeric(38, 2)		☐
	xep_loai	text	☑	text		☐

Hình 3.2.14 Bảng coures

Chi tiết bảng course_respond:

9 / 12	Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
▶	id	bigint	☐	bigint		☑
	respond_result	numeric	☑	numeric		☐
	class_code	text	☐	text		☐
	subject_code	text	☐	text		☐
	user_code	text	☐	text		☐
	hocky	integer	☐	integer		☐
	namhoc	text	☐	text		☐
	comment	text	☑	text		☐
	end_date	date	☑	date		☐
	created_at	timestamp	☑	timestamp(6)	now()	☐
	updated_at	timestamp	☑	timestamp(6)		☐
	status	boolean	☑	boolean	true	☐

Hình 3.2.15 Bảng course_respond

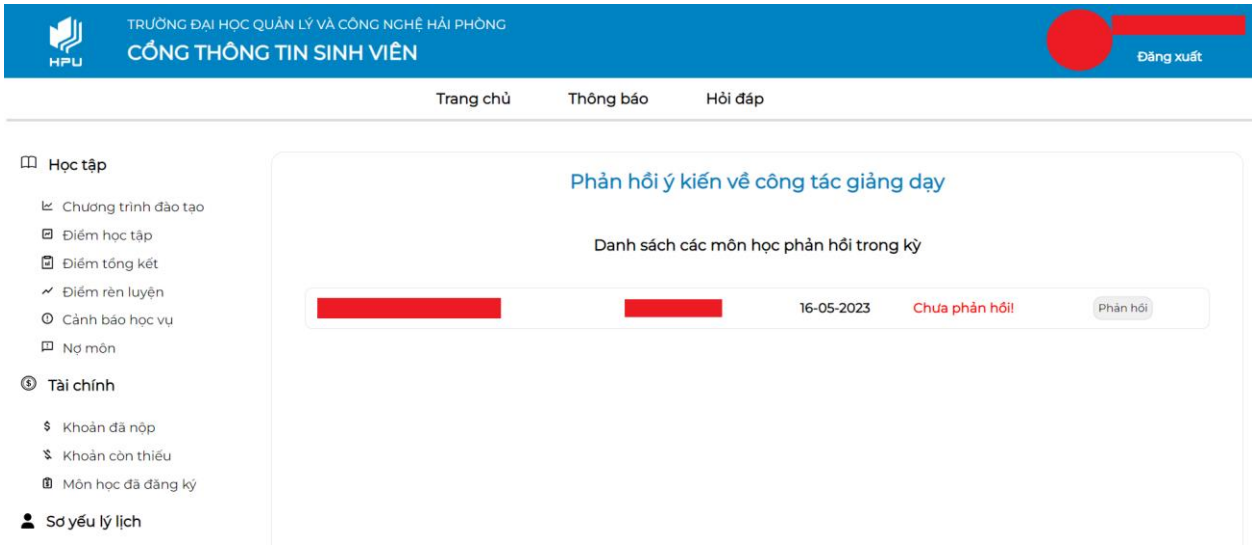
Chi tiết bảng user_respond_detail:

9 / 13	Column Name	Data Type	Null	Data Type(Length)	Default value	Auto
▶	id	bigint	☐	bigint		☑
	question_id	bigint	☐	bigint		☐
	question_point	numeric	☑	numeric		☐
	subject_code	text	☐	text		☐
	class_code	text	☐	text		☐
	user_code	text	☐	text		☐
	status	boolean	☑	boolean	true	☐
	hocky	integer	☐	integer		☐
	namhoc	text	☐	text		☐
	created_at	timestamp	☑	timestamp(6)	now()	☐
	updated_at	timestamp	☑	timestamp(6)		☐
	end_date	date	☑	date		☐
	comment	text	☑	text		☐

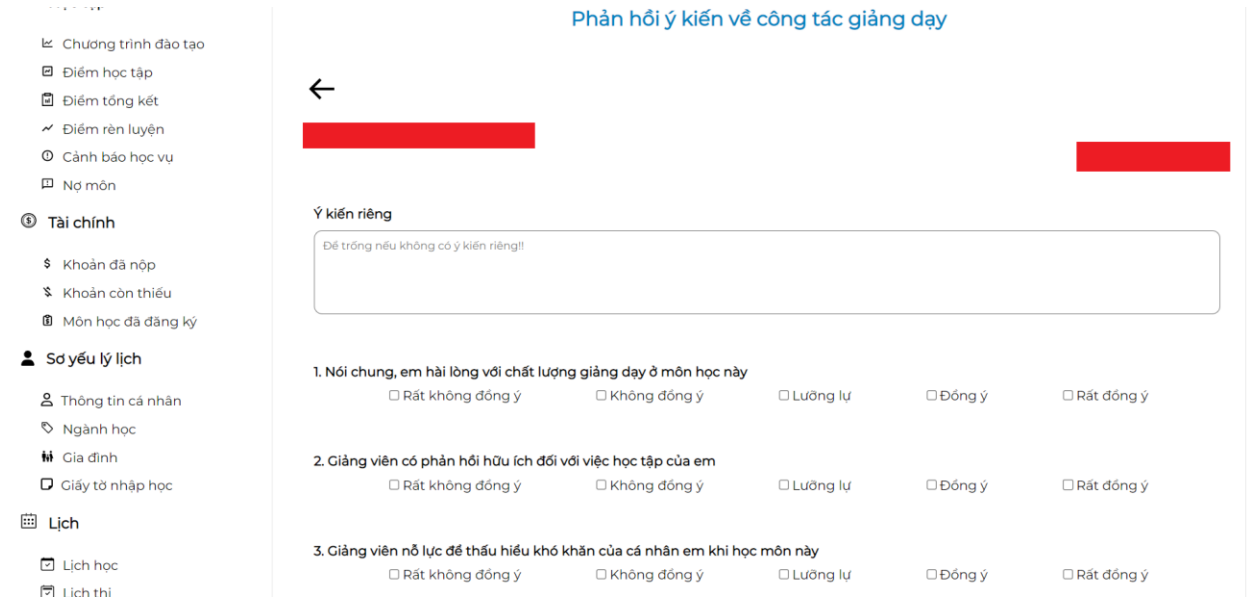
Hình 3.2.16 Bảng user_respond_detail

3.3 Kết quả thực nghiệm

– Một số hình ảnh giao diện

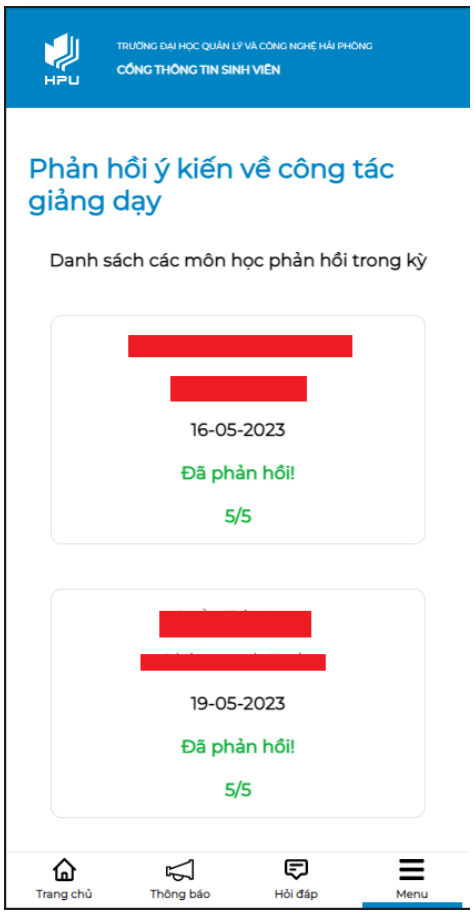


Hình 3.3.1 Giao diện lớp môn học đánh giá của sinh viên

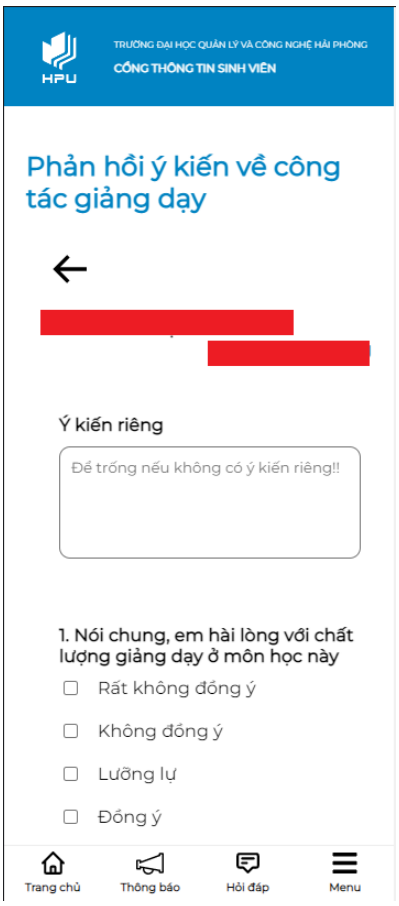


Hình 3.3.2 Giao diện đánh giá chi tiết lớp môn học của sinh viên

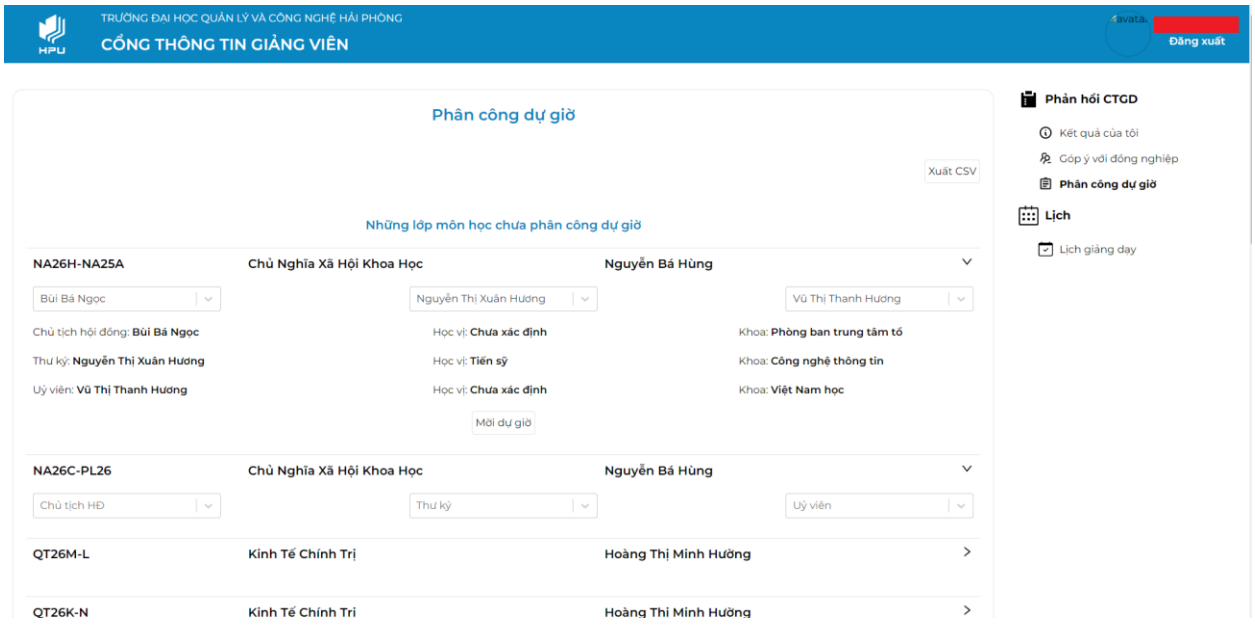
Hình 3.3.3 Giao diện sau khi đánh giá lớp môn học của sinh viên



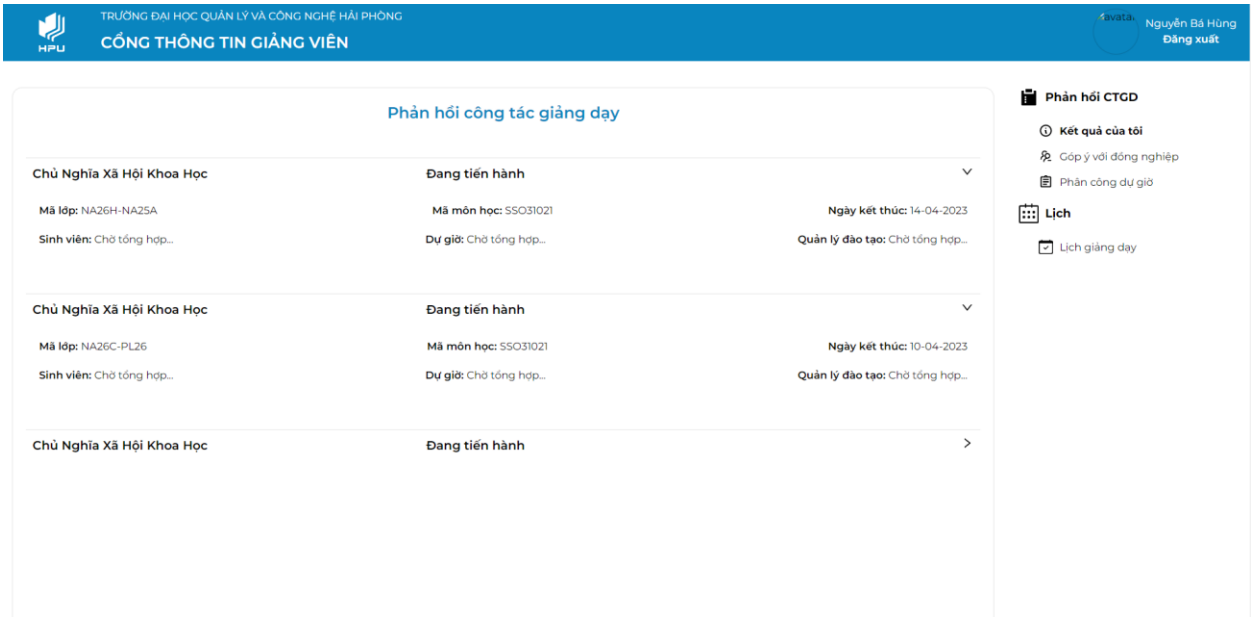
Hình 3.3.4 Giao diện mobile của sinh viên



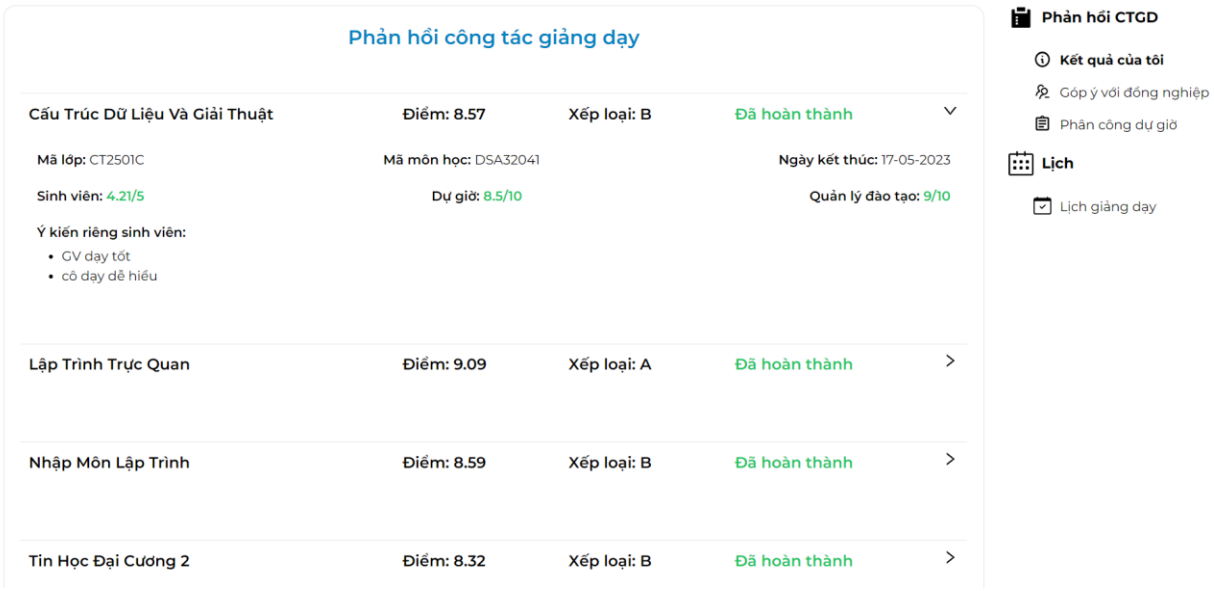
Hình 3.3.5 Giao diện mobile đánh giá của sinh viên



Hình 3.3.6 Giao diện phân công dự giờ của trưởng khoa



Hình 3.3.7 Giao diện các lớp môn học được đánh giá của giảng viên



Hình 3.3.8 Giao diện lớp môn học của giảng viên sau khi hoàn thành

STT	Mã môn học	Mã lớp môn học	Môn học	Giảng viên	Khoa	Điểm trung bình sinh viên	Số lượng sinh v	Tổng số lu	Điểm trung	Số lượng g	Tổng số gi	Điểm quản	Tổng điểm	Xếp loại
1	DSA32041	CT2501C	Cấu Trúc Dữ Liệu Và Giả	Nguyễn Thị Xuân Hương	Công nghệ thông tin	4.21	28	28	8.5	3	3	9	8.57	B
2	VPR33031	CT2401C	Lập Trình Trực Quan	Nguyễn Thị Xuân Hương	Công nghệ thông tin	4.7	11	11	8.83	3	3	9	9.09	A
3	ITP32041	CT2601	Nhập Môn Lập Trình	Nguyễn Thị Xuân Hương	Công nghệ thông tin	4.13	34	34	8.72	3	3	9	8.59	B
4	ICD31022	QT2601L	Tin Học Đại Cương 2	Nguyễn Thị Xuân Hương	Công nghệ thông tin	3.82	23	23	8.67	3	3	9	8.32	B
5	AIN33031	CT2401C	Trí Tuệ Nhân Tạo	Nguyễn Thị Xuân Hương	Công nghệ thông tin	4.75	12	12	8.83	3	3	9	9.13	A
6	OSP33031	CT2501C	Hệ Điều Hành	Phùng Anh Tuấn	Công nghệ thông tin		28	28		0	3	9		
7	NMA33031	CT2401C	Quản Trị Mạng	Phùng Anh Tuấn	Công nghệ thông tin		11	11		0	0	9		
8	ADI33041	CT2501C	Phân Tích Thiết Kế Hệ T	Vũ Anh Hùng	Công nghệ thông tin		28	28		3	3	9		
9	SPM33031	CT2401C	Quản Lý Dự Án Phần Mề	Vũ Anh Hùng	Công nghệ thông tin		11	11		3	3	9		
10	SWP33021	CT2401C	Đồ Án Môn Học Phần Mề	Vũ Anh Hùng	Công nghệ thông tin		12	12		3	3	9		
11	CAR32021	CT2601	Cấu Trúc Máy Tính	Đặng Quang Huy	Công nghệ thông tin		33	33		3	3	9		
12	ICD31022	NA26H-PL26	Tin Học Đại Cương 2	Đặng Quang Huy	Công nghệ thông tin		29	29		3	3	9		
13	ICD31022	QT2601K	Tin Học Đại Cương 2	Đặng Quang Huy	Công nghệ thông tin		21	21		3	3	9		
14	ICD31022	QT2601N	Tin Học Đại Cương 2	Đặng Quang Huy	Công nghệ thông tin		25	25		3	3	9		
15	ICD31022	QT26M-DL26	Tin Học Đại Cương 2	Đặng Quang Huy	Công nghệ thông tin		29	29		3	3	9		
16	IPR33031	CT2401C	Xử Lý Ảnh	Đặng Quang Huy	Công nghệ thông tin		14	14		3	3	9		
17	POS33031	CT2401C	Lập Trình Mã Nguồn Mở	Đỗ Văn Tuyên	Công nghệ thông tin		11	11		0	3	9		
18	PYP32031	CT2601	Lập Trình Python	Đỗ Văn Tuyên	Công nghệ thông tin		33	33		0	0	9		
19	ICD31022	DC26-MT26	Tin Học Đại Cương 2	Đỗ Văn Tuyên	Công nghệ thông tin		25	25		0	0	9		
20	ICD31022	NA2601C	Tin Học Đại Cương 2	Đỗ Văn Tuyên	Công nghệ thông tin		28	28		0	0	9		
21	COM33031	QT24.25M	Truyền Thông Marketing	Đỗ Văn Tuyên	Công nghệ thông tin		28	28		0	0	9		
22	MAT31021	CT2601	Toán Cao Cấp	Hoàng Hải Vân	Cơ sở cơ bản		33	33		0	0	9		
23	MAT31021	DC26-MT26	Toán Cao Cấp	Hoàng Hải Vân	Cơ sở cơ bản		25	25		0	0	9		
24	DOC31021	QT24K,QT26K-N	Văn Bản Học	Nguyễn Thị Hà Anh	Cơ sở cơ bản		60	60		0	0	9		
25	DOC31021	QT24M-N,QT26M	Văn Bản Học	Nguyễn Thị Hà Anh	Cơ sở cơ bản		60	60		0	0	9		

Hình 3.3.9 Ảnh xuất báo cáo tổng kết của quản lý

STT	Mã lớp môn h	Lớp môn học	Phòng h	Thời g	Số ti	Tên giảng viên phụ trác	Chủ tịch hội đồng	Thư ký	Uỷ viên	Điểm trung bình	Ý kiến riêng
1	CT2501C	Cấu Trúc Dữ Liệu Và Giải Thuật				Nguyễn Thị Xuân Hươi	1971016001	1971016003	1971020011	8.5	
2	CT2601	Cấu Trúc Máy Tính				Đặng Quang Huy	1971016004	1971016001	1971020010		
3	CT2401C	Đồ Án Môn Học Phần Mềm				Vũ Anh Hùng	1971016004	1971020010	1971020011		
4	CT2501C	Hệ Điều Hành				Phùng Anh Tuấn	1972000037	1971016004	1971020010		
5	CT2401C	Lập Trình Mã Nguồn Mở				Đỗ Văn Tuyên	1972000037	1971016004	1971020010		
6	CT2601	Lập Trình Python				Đỗ Văn Tuyên					
7	CT2401C	Lập Trình Trực Quan				Nguyễn Thị Xuân Hươi	1971016001	1971016003	1971020011	8.83	
8	CT2601	Nhập Môn Lập Trình				Nguyễn Thị Xuân Hươi	1971016001	1971016003	1971008012	8.72	
9	CT2501C	Phân Tích Thiết Kế Hệ Thống Thông Tin Quản Lý				Vũ Anh Hùng	1971016007	1971016003	1971016004		
10	CT2401C	Quản Lý Dự Án Phần Mềm				Vũ Anh Hùng	1971016004	1971016003	1971016007		
11	CT2401C	Quản Trị Mạng				Phùng Anh Tuấn					
12	NA2601C	Tin Học Đại Cương 2				Đỗ Văn Tuyên					
13	DC26-MT26	Tin Học Đại Cương 2				Đỗ Văn Tuyên					
14	QT26M-DL26	Tin Học Đại Cương 2				Đặng Quang Huy	1971016004	1971016001	1971020011		
15	QT2601K	Tin Học Đại Cương 2				Đặng Quang Huy	1971016004	1971016001	1971011004		
16	NA26H-PL26	Tin Học Đại Cương 2				Đặng Quang Huy	1971016004	1971016001	1971016007		
17	QT2601N	Tin Học Đại Cương 2				Đặng Quang Huy	1971016004	1971016001	1971020010		
18	QT2601L	Tin Học Đại Cương 2				Nguyễn Thị Xuân Hươi	1971016001	1971016003	1971008012	8.67	
19	CT2401C	Trí Tuệ Nhân Tạo				Nguyễn Thị Xuân Hươi	1971016007	1971016001	1971016003	8.83	
20	QT24,25M	Truyền Thông Marketing				Đỗ Văn Tuyên					
21	CT2401C	Xử Lý Ảnh				Đặng Quang Huy	1971016004	1971016001	1971016007		

Hình 3.3.10 Ảnh xuất báo cáo tổng kết của trưởng khoa

3.4 Kết luận chương

Chương 3 đã cho ta thấy được cách để vận dụng lý thuyết vào một ví dụ cụ thể đó là bài toán xây dựng website lấy ý kiến trực tuyến về công tác giảng dạy tại HPU. Qua đó cũng cho ta thấy được cách một phần mềm được tạo ra như thế nào. . Tuy nhiên, việc ứng dụng lý thuyết vào thực tế còn có nhiều sai sót trong việc xây dựng phần mềm và cần phải tiếp tục nghiên cứu chỉnh sửa trong khoảng thời gian tiếp theo.

KẾT LUẬN

Qua bài đồ án tốt nghiệp này, em đã hiểu được thế nào là một hệ thống phát triển theo kiến trúc Microservices và từ đó mở rộng được tính giao tiếp giữa các phần mềm thông qua API, điều mà một hệ thống phát triển theo hướng monolith khó mà làm được do có cấu trúc nguyên khối và sự khác nhau giữa ngôn ngữ. Cùng với đó là Domain Driven Design là một hướng phân tích thiết kế phù hợp với kiến trúc microservices vì nó có thể giúp chúng ta chia hệ thống ra được nhiều phần nhỏ và hoạt động độc lập với nhau.

Từ đó, em cũng đã vận dụng được kiến thức mình tìm hiểu về Microservices và Domain Driven Design vào việc "Xây dựng website lấy ý kiến trực tuyến về công tác giảng dạy của giảng viên HPU".

*Hạn chế: Mặc dù đã rất cố gắng trong việc nghiên cứu và thực hiện đồ án, nhưng do thời gian và cũng là dự án đầu tiên được áp dụng chạy thực tế nên trong quá trình phân tích thiết kế còn gặp phải nhiều những sai sót cũng như việc xây dựng trang Web vẫn còn nhiều vấn đề chưa được giải quyết triệt để. Nên em rất mong nhận được ý kiến đóng góp từ thầy cô. Em xin trân thành cảm ơn các thầy cô.

TÀI LIỆU THAM KHẢO

[1] T. A. Vu, “Microservices là gì?,” 03 11 2018. [Trực tuyến]. Available: https://viblo.asia/p/microservices-la-gi-gAm5yjD8Kdb.

[2] TopDev, “API là gì? Tại sao API được sử dụng nhiều hiện nay?,” 15 06 2019. [Trực tuyến]. Available: https://topdev.vn/blog/api-la-gi/.

[3] TopDev, “RESTful API là gì? Cách thiết kế RESTful API,” 22 04 2019. [Trực tuyến]. Available: https://topdev.vn/blog/restful-api-la-gi/.

[4] TopDev, “JSON Web Token (JWT) là gì ?,” 25 12 2017. [Trực tuyến]. Available: https://topdev.vn/blog/jwt-la-gi/.

[5] N. Hưng, “Microservices là gì? Tổng quan kiến thức về Microservices từ A-Z,” 20 04 2022. [Trực tuyến]. Available: https://vietnix.vn/microservices-la-gi/.

[6] TopDev, “Microservices là gì? Speed up Microservices 1: Tác dụng phụ và một số chiến lược cơ bản,” 22 03 2018. [Trực tuyến]. Available: https://topdev.vn/blog/microservices-la-gi/.

[7] L. V. Q. Trinh, “Khái niệm cơ bản về Domain Driven Design (DDD),” 24 05 2019. [Trực tuyến]. Available: https://viblo.asia/p/khai-niem-co-ban-ve-domain-driven-design-ddd-Do754qL4KM6.

[8] C. t. h. đ. trường, QUYẾT ĐỊNH Ban hành Quy chế đánh giá công tác giảng dạy, Hải Phòng, 2023.