

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

TRẦN KHÁNH LINH – TRẦN THANH TỊNH

**XÂY DỰNG HỆ THỐNG CUNG CẤP
DỊCH VỤ NHẬN DẠNG ÂM THANH
TIẾNG VIỆT**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP. HCM, 2020

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ PHẦN MỀM

TRẦN KHÁNH LINH – 1612339

TRẦN THANH TỊNH – 1612704

**XÂY DỰNG HỆ THỐNG CUNG CẤP DỊCH
VỤ NHẬN DẠNG ÂM THANH TIẾNG VIỆT**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

GIÁO VIÊN HƯỚNG DẪN

TS. NGÔ HUY BIÊN

KHOÁ 2016 – 2020

[illegible]

[Ký tên và ghi rõ họ tên]

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting or typing. There are no margins, text, or other markings on the page.

TpHCM, ngày . . . tháng . . . năm 2020

Giáo viên phản biện

[Ký tên và ghi rõ họ tên]

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Thầy – Tiến sĩ Ngô Huy Biên. Trong suốt thời gian thực hiện khoá luận, thầy là người đã định hướng, thẳng thắn góp ý và tận tình giải đáp các thắc mắc của nhóm về đề tài, cung cấp những kiến thức phù hợp làm nền tảng giúp nhóm hoàn thành khoá luận.

Nhóm chúng em xin gửi lời cảm ơn đến quý Thầy Cô trong khoa Công nghệ Thông tin của trường đại học Khoa Học Tự Nhiên đã giảng dạy, nâng đỡ chúng em trong suốt gần 4 năm học vừa qua. Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em trong quá trình học tập và thực hiện đề tài tốt nghiệp. Đồng thời chúng em cũng không quên gửi những lời cảm ơn chân thành đến những người thân trong gia đình và bạn bè đã giúp đỡ chúng em trong quá trình thực hiện luận văn, đặc biệt là quá trình định hướng kiến trúc tổng quan cho đề tài.

Do trình độ nghiên cứu và thời gian có hạn, chúng em đã cố gắng hết sức nhưng chắc chắn không tránh khỏi những thiếu sót và hạn chế. Rất mong nhận được sự góp ý và chỉ dẫn của quý Thầy Cô.

Cuối cùng, chúng em xin trân trọng cảm ơn và chúc sức khỏe quý Thầy Cô!

TpHCM, ngày ... tháng ... năm 2020

Sinh viên



ĐỀ CƯƠNG KHOÁ LUẬN TỐT NGHIỆP

XÂY DỰNG HỆ THỐNG CUNG CẤP

DỊCH VỤ NHẬN DẠNG ÂM THANH

TIẾNG VIỆT

1 THÔNG TIN CHUNG

Người hướng dẫn:

- TS. Ngô Huy Biên (Khoa Công nghệ Thông tin)

Nhóm sinh viên thực hiện:

1. Trần Khánh Linh (MSSV: 1612339)
2. Trần Thanh Tịnh (MSSV: 1612704)

Loại đề tài: Ứng dụng

Thời gian thực hiện: Từ 21/10/2019 đến 07/2020

2 NỘI DUNG THỰC HIỆN

2.1 Giới thiệu về đề tài

Giao tiếp luôn là một hình thức quan trọng trong việc thấu hiểu và kết nối giữa

hai hoặc nhiều cá thể với nhau. Trong đó, tiếng nói là một phương tiện giao tiếp tự nhiên và được sử dụng phổ biến nhất. Thông qua giọng nói, nó cho phép người nói biểu đạt ý nghĩ, mong muốn, cảm xúc đến người nghe. Tuy nhiên, quá trình giao tiếp không chỉ diễn ra giữa người với người, nó còn diễn ra giữa người với thiết bị, máy móc. Đúng như ý nghĩa của nó, giao tiếp cho phép con người “nói chuyện” được với máy móc, cho phép con người truyền đạt mong muốn, yêu cầu đến máy móc. Dựa vào đó, nhu cầu tự động hoá của con người được đáp ứng. Nói cách khác, nhu cầu tự động hoá của con người ngày càng tăng đồng nghĩa với nhu cầu kết nối, “nói chuyện” với máy móc, thiết bị của con người ngày càng tăng.

Trong những năm gần đây, các nhà phát triển đã dần chuyển đổi cách thức giao tiếp giữa con người và máy tính từ việc gõ phím sang thao tác trên các giao diện ứng dụng và gần đây là giao tiếp qua giọng nói. Nhận thấy tiềm năng trong việc sử dụng nhận dạng âm thanh, hàng loạt các phần mềm ứng dụng đã tích hợp dịch vụ này như các ứng dụng học ngôn ngữ, nhà thông minh, thiết bị thông minh,... Tuy nhiên các ứng dụng vừa và nhỏ đang gặp rất nhiều khó khăn trong việc tiếp cận cách dịch vụ này, do các hệ thống cung cấp dịch vụ đều có chi phí cao, khó sử dụng, chưa hỗ trợ nhiều cho nền tảng tiếng Việt nói chung, ít tài liệu,...

Nhận thấy những khó khăn này đang ảnh hưởng đến sự phát triển của các ứng dụng cho cộng đồng, vì cộng đồng, nhóm chúng em đã xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt phục vụ cho cộng đồng lập trình viên ở Việt Nam nói chung. Tuy nhiên, hệ thống do nhóm chúng em phát triển sẽ phải khắc phục được các yếu điểm ở trên và đạt được một số tiêu chuẩn nhất định sẽ được trình bày ở phần tiếp theo.

2.2 Mục tiêu đề tài

Mục tiêu của đề tài là nghiên cứu, tìm hiểu các công nghệ liên quan đến việc xây dựng một hệ thống lớn, có các yêu cầu nghiệp vụ, kỹ thuật phức tạp. Ví dụ như các yêu cầu về nghiệp vụ liên quan đến quản lý thanh toán, số lượng khách hàng, giao dịch tài khoản, các yêu cầu về kỹ thuật như chịu tải lớn, cho phép một số lượng lớn

truy cập trong một thời gian ngắn. Từ quá trình tìm hiểu qua các nguồn tài nguyên như sách báo, các mã nguồn liên quan đến kỹ thuật để xây dựng một hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt. Cùng với đó là phát triển những chức năng liên quan, cải tiến để đáp các nhu cầu luôn thay đổi của người dùng. Tiếp đến xét tới các yếu tố về giao diện sao cho mới mẻ, độc đáo, chuyên nghiệp, đạt chuẩn để phù hợp với nhiều loại người dùng khác nhau.

2.2.1. Xây dựng dịch vụ web (API) để nhận một tập tin (file) âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản.

- Quản lý người đăng ký, khóa truy cập, số lượng truy cập vào dịch vụ:
 - Cho phép người dùng đăng ký, và thanh toán chi phí sử dụng dịch vụ.
 - Cho phép người quản trị quản lý người đăng ký, khóa truy cập sử dụng dịch vụ.
 - Cung cấp API cung cấp dịch vụ chuyển một tập tin âm thanh tiếng Việt thành nội dung văn bản.
 - Cung cấp các báo cáo về việc sử dụng dịch vụ của người dùng.
- Tạo thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử dụng dịch vụ, với các ngôn ngữ khác nhau:
 - SDK hỗ trợ tối thiểu hai loại ứng dụng là web và mobile (android hoặc iOS), ít nhất có ví dụ cho Java, Javascript (hoặc PHP, Swift) cho mỗi loại web hoặc mobile.
- Thiết kế và hiện thực hóa khả năng mở rộng dịch vụ, khi số lượng truy cập cao:
 - Áp dụng Microservices, CQRS và Event Sourcing.
 - Mô phỏng, thiết kế, thực hiện, báo cáo kết quả các kịch bản kiểm thử khả năng tải mong muốn (load tests), và khả năng chịu tải tối đa (stress tests).

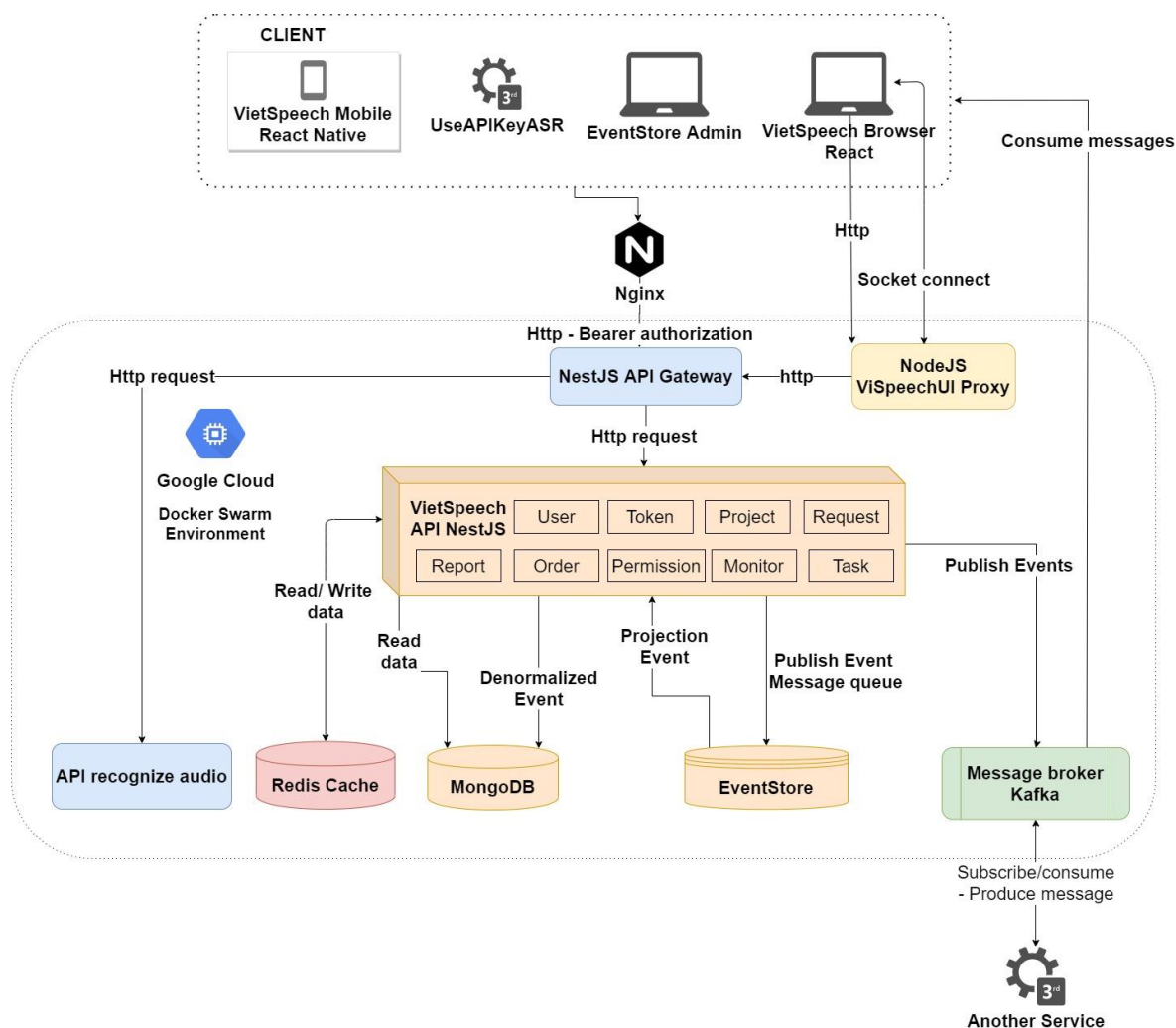
2.2.2. Xây dựng một trang web và một mobile client demo việc sử dụng SDK để tải lên một tập tin âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản (ví dụ: Trò chơi hỗ trợ học tiếng Việt).

2.2.3. Viết 120 trang luận văn.

2.3 Phạm vi đề tài

- Dịch vụ web cho phép tải lên tập tin âm thanh và xuất ra nội dung ở dạng văn bản.
- Ứng dụng mobile được xây dựng cho các thiết bị Android 6.0 cho đến phiên bản mới nhất hiện tại.
- Trang web phục vụ quản trị và người dùng cuối, không hỗ trợ trên các trình duyệt không phổ biến như Internet Explorer và các trình duyệt tương tự.
- Tài liệu SDK và ví dụ mẫu hỗ trợ cho ứng dụng web và mobile (android), phục vụ các ngôn ngữ CSharp, Java, Javascript cho mỗi loại nêu trên.

2.4 Cách tiếp cận dự kiến



Hình 1: Tổng quan kiến trúc hệ thống

Dựa vào yêu cầu, đặc điểm của đề tài, cũng như thời gian, tài nguyên và sự gợi ý của giáo viên hướng dẫn, nhóm đưa ra cách tiếp cận dựa vào các mô hình, kiến trúc, công nghệ được trình bày như hình 1. Từng đặc điểm trong mô hình tổng quan sẽ được làm rõ trong các phần tiếp theo.

2.4.1. Kiến trúc Microservice

Nhận thấy đề tài có nhiều dịch vụ kết nối với nhau như dịch vụ web, trang web quản lý người dùng, trang web cho người dùng, cơ sở dữ liệu, dịch vụ web nhận dạng âm thanh tiếng Việt và các dịch vụ có thể phát sinh khác trong tương lai, đồng thời để thực hiện hoá khả năng mở rộng dịch vụ khi số lượng truy cập cao, nhóm quyết định tổ chức các thành phần hệ thống theo kiến trúc Microservice.

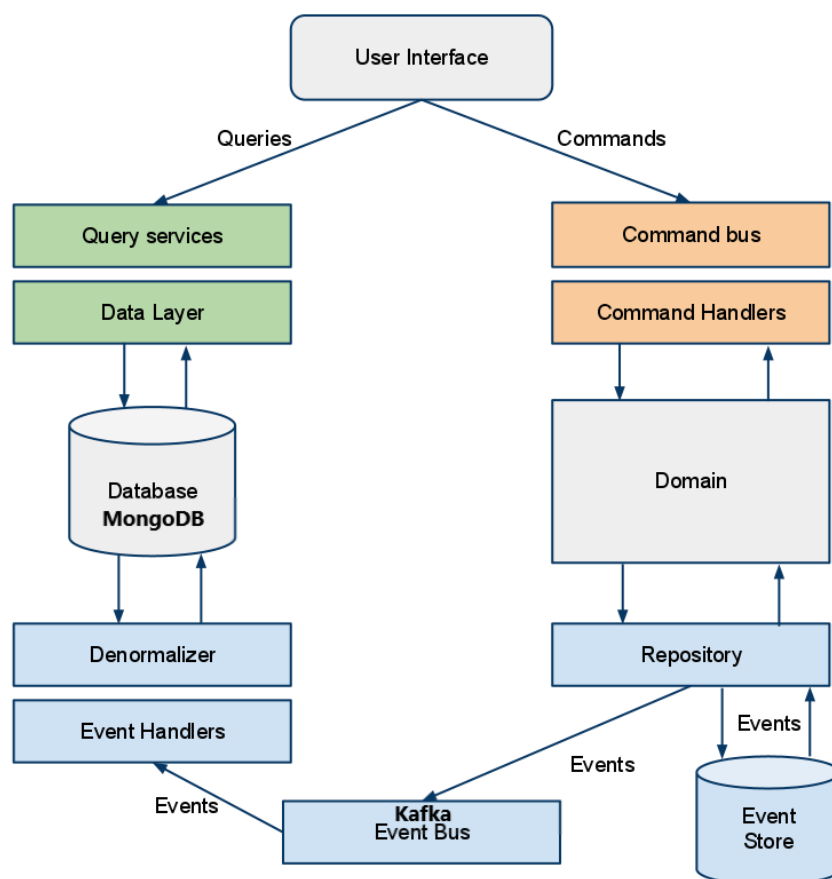
2.4.2. Phương pháp thiết kế từ nghiệp vụ (Domain Driven Design – DDD)

Việc phát triển dựa vào phương pháp DDD sẽ đem lại rất nhiều lợi ích nhờ vào tính chất phát triển độc lập các thành phần trong hệ thống, đơn giản hóa các đối tượng trong nghiệp vụ kinh doanh, thể hiện bằng chính mã nguồn. Đây chính là tiền đề để phát triển hệ thống kết hợp với các mô hình tiếp theo.

2.4.3. Mẫu Command Query Responsibility Segregation (CQRS)

Đối với hệ thống cung cấp dịch vụ, một đặc điểm quan trọng ảnh hưởng tới hiệu năng của hệ thống chính là tốc độ xử lý đọc - ghi của dịch vụ web. Đối với đề tài này, nhu cầu ghi nhận lịch sử, quá trình sử dụng, báo cáo về việc sử dụng dịch vụ chiếm phần lớn so với nhu cầu đọc dữ liệu.

Dựa vào đặc điểm này, mô hình CQRS là giải pháp tối ưu nhằm tách biệt hai luồng đọc và ghi. Trong đó, nhóm chọn cơ sở dữ liệu MongoDB phục vụ cho luồng đọc, Event Store phục vụ cho luồng ghi.



Hình 2: Mô hình mẫu CQRS trong hệ thống

2.4.4. Mẫu Event Sourcing

Lịch sử giao dịch, lịch sử thanh toán, lịch sử sử dụng là những trạng thái, sự kiện mà hệ thống cần lưu giữ và đảm bảo được tính chính xác. Những dữ liệu này là một phần rất quan trọng trong việc xây dựng tính năng quản lý người dùng.

Đồng thời, nhằm đáp ứng cho quá trình mở rộng, nhu cầu tất yếu là tái lập lại tất cả trạng thái, lịch sử cho dịch vụ mới là không thể tránh khỏi. Mẫu Event Sourcing được áp dụng vào đề tài này là một giải pháp thực tế đảm bảo được tính chính xác, tính bền vững, toàn vẹn của dữ liệu [1].

2.5 Kết quả dự kiến của đề tài

- Dịch vụ web để nhận một tập tin âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản, có khả năng mở rộng dịch vụ, khi số lượng truy cập cao.

- Trang web quản lý người đăng ký, khóa truy cập, số lượng truy cập vào dịch vụ.
- Cho phép người dùng đăng ký, và thanh toán chi phí sử dụng dịch vụ.
- Cho phép người quản trị quản lý người đăng ký, khóa truy cập sử dụng dịch vụ.
- Cung cấp dịch vụ web cung cấp dịch vụ chuyển một tập tin âm thanh tiếng Việt thành nội dung văn bản.
- Cung cấp các báo cáo về việc sử dụng dịch vụ của người dùng.
- Thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử dụng dịch vụ, với các ngôn ngữ khác nhau.
- Ứng dụng Android sử dụng dịch vụ web để phục vụ nhu cầu học tiếng Việt của người nước ngoài.
- Trang web mẫu sử dụng dịch vụ web nhận dạng âm thanh tiếng Việt.

2.6 Kế hoạch thực hiện

Thời gian thực hiện	Nội dung thực hiện	Người thực hiện
05/11/2019 – 10/11/2019	<ul style="list-style-type: none"> - Nhận đề tài. - Xây dựng bản kế hoạch sơ bộ cho các công việc cần thực hiện. 	Linh, Tịnh
11/11/2019 – 24/11/2019	<ul style="list-style-type: none"> - Khảo sát và dùng thử các hệ thống cung cấp dịch vụ mẫu có sẵn trên thị trường. - Xây dựng 1 bản mẫu prototype, proof of concept (POC). 	Linh, Tịnh

25/11/2019 – 01/12/2019	<ul style="list-style-type: none"> - Viết chương 1 luận văn. - Xây dựng kiến trúc sơ bộ cho hệ thống. 	Linh, Tịnh
02/12/2019 – 15/02/2020	<ul style="list-style-type: none"> - Tìm hiểu lý thuyết nền tảng về Microservices. - Tìm hiểu công nghệ phù hợp cho mô hình (NestJS, Event store, Kafka). 	Linh, Tịnh
16/12/2019 – 29/12/2019	<ul style="list-style-type: none"> - Tìm hiểu lý thuyết nền tảng về mẫu Command Query Responsibility Segregation (CQRS). - Xây dựng bản mẫu cho mẫu CQRS. 	Linh, Tịnh
30/12/2019 – 05/01/2020	<ul style="list-style-type: none"> - Viết chương 2 luận văn. - Tìm hiểu lý thuyết nền tảng về mô hình Domain Driven Design (DDD). - Xây dựng bản mẫu cho mô hình DDD kết hợp với mẫu CQRS. 	Linh, Tịnh
06/01/2020 – 12/01/2020	<ul style="list-style-type: none"> - Viết chương 2 luận văn. - Tìm hiểu lý thuyết nền tảng về mô hình Event Sourcing (ES). - Xây dựng bản mẫu cho mô hình 	Linh, Tịnh

	ES.	
13/01/2020 – 19/01/2020	<ul style="list-style-type: none"> - Chỉnh sửa chương 2. - Kết hợp mô hình ES với DDD và CQRS dựa trên cơ sở dữ liệu không quan hệ (MongoDB). 	Linh, Tịnh
20/01/2020 – 26/01/2020	<ul style="list-style-type: none"> - Thực hiện chi tiết mã nguồn quản lý khóa, các tác vụ cho dịch vụ. - Tìm hiểu chi tiết các quản lý luồng sự kiện và vận hành chi tiết của hệ thống. 	Linh, Tịnh
27/01/2020 – 02/02/2020	<ul style="list-style-type: none"> - Thiết kế giao diện hệ thống. - Đưa tất cả hệ thống lên máy chủ Google Compute và kiểm tra năng suất hệ thống. - Chỉnh sửa luồng sự kiện, phân quyền và viết tài liệu cài đặt. 	Linh, Tịnh
03/02/2020 – 20/02/2020	<ul style="list-style-type: none"> - Viết chương 3 luận văn. - Thực hiện cài đặt cho quản lý sự kiện trên Event store, đảm bảo lưu vết và tái diễn sự kiện dịch vụ. - Viết mã nguồn giao diện cho khách hàng. 	Linh, Tịnh

21/02/2020 – 04/04/2020	<ul style="list-style-type: none"> - Chỉnh sửa chương 3. - Tiếp tục viết mã nguồn giao diện phía khách hàng. - Kết nối giữa giao diện và dịch vụ, đảm bảo luồng sự kiện và hiệu suất đạt chất lượng. 	Linh, Tịnh
05/04/2020 – 19/04/2020	<ul style="list-style-type: none"> - Viết chương 4 luận văn. - Viết tài liệu SDK và ví dụ sử dụng với nhiều ngôn ngữ. - Viết mã nguồn cho giao diện của người quản trị. 	Linh, Tịnh
20/04/2020 – 05/05/2020	<ul style="list-style-type: none"> - Chỉnh sửa giao diện của người quản lý và kết nối với dịch vụ web (API). - Chỉnh sửa tài liệu SDK. - Viết mã nguồn giao diện cho ứng dụng Android. 	Linh, Tịnh
06/05/2020 – 14/05/2020	<ul style="list-style-type: none"> - Chỉnh sửa chương 4. - Viết mã nguồn kết nối dịch vụ web (API) cho ứng dụng Android. - Viết tài liệu cài đặt cho ứng dụng Android. 	Linh, Tịnh
15/05/2020 – 26/05/2020	<ul style="list-style-type: none"> - Viết chương 5 luận văn. 	Linh, Tịnh

	<ul style="list-style-type: none"> - Chỉnh sửa ứng dụng Android và kiểm tra lại hiệu năng hệ thống với kiểm thử loading testing, stress testing. - Tìm hiểu và nghiên cứu giải pháp mở rộng cho dịch vụ web khi lượt tải tăng cao. 	
27/05/2020 – 15/06/2020	<ul style="list-style-type: none"> - Chỉnh sửa chương 5. - Thực hiện các giải pháp tăng hiệu suất hệ thống và đảm bảo hệ thống luôn hoạt động khi lượt sử dụng tăng cao. 	Linh, Tịnh
16/06/2020 – 05/07/2020	<ul style="list-style-type: none"> - Xây dựng trang demo sử dụng dịch vụ. - Đọc lại và chỉnh sửa cuốn luận văn. 	Linh, Tịnh
06/07/2020 – 20/07/2020	<ul style="list-style-type: none"> - Hoàn chỉnh cuốn luận văn. - Thiết kế slide và chuẩn bị nội dung cho buổi bảo vệ luận văn. - Đóng gói mã nguồn và hoàn thiện tất cả tài liệu cài đặt, SDK. 	Linh, Tịnh
21/06/2020 – 07/08/2020	<ul style="list-style-type: none"> - Hoàn tất cuốn luận văn. - Hoàn tất slide trình bày. - Luyện tập trình bày cho buổi bảo 	Linh, Tịnh

	vệ.	
--	-----	--

Tài liệu

[1] D. B. J. D. G. M. F. S. M. S. G. Young, Exploring CQRS and Event Sourcing. 2012.

XÁC NHẬN
CỦA NGƯỜI HƯỚNG DẪN
(Ký và ghi rõ họ tên)

TP. Hồ Chí Minh, 07/08/2020
NHÓM SINH VIÊN THỰC HIỆN
(Ký và ghi rõ họ tên)

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU	1
1.1 GIỚI THIỆU LUẬN VĂN	1
1.1.1 Lợi ích thực tế	1
1.1.2 Ứng dụng thực tiễn.....	3
1.2 KHẢO SÁT THỊ TRƯỜNG.....	4
1.2.1 Cloud Speech-to-Text do Google phát triển.....	4
1.2.2 FPT.AI Speech To Text do FPT phát triển	5
1.2.3 Viettel AI Speech to Text do Viettel phát triển	6
1.3 LÝ DO LỰA CHỌN ĐỀ TÀI.....	7
1.4 MỤC TIÊU CỦA LUẬN VĂN.....	9
1.5 YÊU CẦU CHỨC NĂNG.....	10
1.6 PHẠM VI ĐỀ TÀI.....	13
CHƯƠNG 2: LÝ THUYẾT NỀN TẢNG	14
2.1 MICROSERVICES	14
2.1.1 Định nghĩa	14
2.1.2 Tính chất.....	24
2.2 PHƯƠNG PHÁP THIẾT KẾ TỪ NGHIỆP VỤ	29
2.2.1 Định nghĩa	29
2.2.2 Đặc điểm và các thành phần.....	30

2.2.3	Duy trì tính toàn vẹn của mô hình	36
2.3	MẪU COMMAND QUERY RESPONSIBILITY SEGREGATION – CQRS	44
2.3.1	Định nghĩa	44
2.3.2	CQRS và DDD.....	46
2.3.3	Commands, events và các thông điệp (messages).....	47
2.3.4	Lợi ích của mẫu thiết kế CQRS	49
2.3.5	Khi nào nên áp dụng mẫu CQRS?	51
2.4	MÔ HÌNH EVENT SOURCING – ES.....	53
2.4.1	Định nghĩa	53
2.4.2	Lợi ích của mô hình Event Sourcing	59
2.4.3	Sự kết hợp giữa Event Sourcing và CQRS	61
2.5	TỔNG KẾT.....	63
CHƯƠNG 3:	GIẢI PHÁP ĐỀ TÀI.....	64
3.1	TỔNG QUAN GIẢI PHÁP VÀ KIẾN TRÚC MÔ HÌNH.....	64
3.1.1	Mô hình API Gateway.....	65
3.1.2	Mô hình Backend For Frontend (BFF)	67
3.1.3	Đảm bảo toàn vẹn dữ liệu.....	69
3.1.4	Cải thiện hiệu năng hệ thống	73
3.2	THIẾT KẾ HỆ THỐNG	75
3.2.1	Thiết kế giao diện hệ thống	75
3.2.2	Thiết kế và giải pháp lưu trữ dữ liệu.....	81

3.2.3	Thiết kế kiến trúc hệ thống.....	82
3.3	GIẢI PHÁP XÂY DỰNG CHỨC NĂNG HỆ THỐNG.....	83
3.3.1	Giải pháp cho chức năng đăng ký/ đăng nhập.....	83
3.3.2	Giải pháp thanh toán chi phí sử dụng dịch vụ âm thanh tiếng Việt	84
3.3.3	Giải pháp cho chức năng mời tham gia dự án.....	85
3.3.4	Giải pháp cung cấp API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt.....	85
3.3.5	Giải pháp cho chức năng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt	86
3.3.6	Giải pháp nâng cấp API key	86
3.3.7	Giải pháp cung cấp các báo cáo sử dụng dịch vụ của người dùng.....	86
3.3.8	Giải pháp xây dựng ứng dụng di động.....	87
3.4	TỔNG KẾT.....	88
CHƯƠNG 4:	CÀI ĐẶT GIẢI PHÁP.....	89
4.1	TYPESCRIPT VÀ NESTJS	89
4.1.1	TypeScript	89
4.1.2	NestJS	90
4.2	REACTJS.....	90
4.2.1	Lập trình khai báo	91
4.2.2	Phát triển dựa trên thành phần (Component-Based).....	91
4.3	CÀI ĐẶT KIẾN TRÚC HỆ THỐNG	91
4.3.1	Cài đặt luồng ghi dữ liệu	91

4.3.2	Cài đặt luồng đọc dữ liệu.....	92
4.4	CÀI ĐẶT CHỨC NĂNG HỆ THỐNG	92
4.4.1	Cài đặt chức năng đăng ký/ đăng nhập	92
4.4.2	Cài đặt chức năng thanh toán chi phí sử dụng dịch vụ nhận dạng âm thanh tiếng Việt.....	94
4.4.3	Cài đặt chức năng mời tham gia dự án.....	96
4.4.4	Cài đặt API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt.....	98
4.4.5	Cài đặt chức năng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt	99
4.4.6	Cài đặt chức năng cấp API key.....	101
4.4.7	Cài đặt chức năng cung cấp các báo cáo sử dụng dịch vụ của người dùng	101
4.5	TỔNG KẾT.....	102
CHƯƠNG 5:	TỔNG KẾT, ĐÁNH GIÁ	104
5.1	KIẾN THỨC THU ĐƯỢC.....	104
5.2	KẾT QUẢ HỆ THỐNG.....	104
5.2.1	Môi trường phát triển	104
5.2.2	Môi trường triển khai	106
5.2.3	Hiệu năng hệ thống	106
5.2.4	Các chức năng đã cài đặt.....	110
5.2.5	So sánh với một số hệ thống khác trên thị trường	111
5.3	KẾT QUẢ ỨNG DỤNG DI ĐỘNG	112

5.3.1	Môi trường phát triển	112
5.3.2	Môi trường triển khai	113
5.3.3	Các chức năng đã cài đặt.....	113
5.4	So sánh các kết quả thu được với mục tiêu ban đầu	113
5.5	ĐỊNH HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU	114
5.5.1	Khả năng mở rộng.....	114
5.5.2	Cải thiện chất lượng nhận dạng âm thanh.....	114
5.5.3	Cải thiện chức năng.....	115
5.6	LỜI KẾT.....	115
CHƯƠNG 6:	PHỤ LỤC	116
6.1	PHỤ LỤC 1: THƯ VIỆN SDK	116
6.2	PHỤ LỤC 2: WEB API.....	117
6.3	PHỤ LỤC 3: GIAO DIỆN CHỨC NĂNG ỨNG DỤNG DI ĐỘNG..	119

DANH MỤC HÌNH

Hình 1.2.1 Giao diện trang chủ của hệ thống dịch vụ Cloud Speech-to-Text API [1]	5
Hình 1.3.1 Mức phí cho dịch vụ của Google Cloud Speech API [4]	8
Hình 1.3.2 Mức phí cho dịch vụ của FPT.AI [5]	8
Hình 2.1.1 Mô hình 4+1 sử dụng bốn góc nhìn và những kịch bản (thể hiện sự hợp tác trong việc xử lý yêu cầu giữa những yếu tố trong mỗi khía cạnh) để mô tả kiến trúc của một ứng dụng [10]	15
Hình 2.1.2 Một dịch vụ có một API đóng gói triển khai nội bộ của nó. API định nghĩa những hoạt động được thực hiện bởi khách hàng của nó [10]	19
Hình 2.1.3 Khối lập phương về khả năng mở rộng định nghĩa ba cách khác nhau để mở rộng ứng dụng: trục X, Y, Z [10]	21
Hình 2.1.4 Mở rộng theo trục X trong khối lập phương về khả năng mở rộng [10]	22
Hình 2.1.5 Mở rộng theo trục Z trong khối lập phương về khả năng mở rộng. Mỗi thực thể chịu trách nhiệm xử lý một tập con dữ liệu người dùng [10]	23
Hình 2.1.6 Mở rộng theo trục Y trong khối lập phương về khả năng mở rộng. Mỗi dịch vụ chịu trách nhiệm cho một chức năng đặc biệt [10]	24
Hình 2.1.7 Luật của Conway khi áp dụng vào cách tổ chức ứng dụng dựa trên tính năng kỹ thuật [14]	26
Hình 2.1.8 Ranh giới giữa các nhóm phát triển củng cố ranh giới giữa các dịch vụ [14]	27
Hình 2.2.1 Trừu tượng hóa nghiệp vụ tạo “mã truy cập” trong hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt thành các đối tượng trong mô hình nghiệp vụ, bao gồm mã truy cập, dự án, người dùng	30

Hình 2.2.2 Biểu đồ và quan hệ của các khuôn mẫu trong mô hình DDD [16]	32
Hình 2.2.3 Mối quan hệ trong Aggregate có gốc là đối tượng người dùng – <i>User</i> cùng các quan hệ với các báo cáo – <i>Report</i> , mã truy cập – <i>Token</i>	36
Hình 2.2.4 Một số kỹ thuật duy trì tính nhất quán của mô hình và mối quan hệ giữa chúng [16]	38
Hình 2.2.5 Ngữ cảnh ánh xạ [16]	42
Hình 2.3.1 Một cách triển khai kiến trúc khi áp dụng mẫu CQRS [17]	45
Hình 2.3.2 Commands và events trong mẫu CQRS [17]	49
Hình 2.4.1 Mô tả cách tiếp cận thứ nhất để lưu trữ tổng số lượng đặt chỗ, sử dụng tầng ORM [17]	54
Hình 2.4.2 Mô tả cách tiếp cận thứ hai để lưu trữ tổng số lượng đặt chỗ, sử dụng ES thay vì tầng ORM và một RDBMS [17]	56
Hình 2.4.3 ES duy trì mỗi aggregate dưới dạng một chuỗi các events, lưu trong bảng Events [10]	58
Hình 2.4.4 Mô tả sự thay đổi trạng thái aggregate Đơn hàng từ S sang S' khi gọi phương thức apply(E). Event E phải chứa dữ liệu cần thiết để thực hiện việc chuyển đổi này [10]	59
Hình 2.4.5 Mô tả sự kết hợp giữa CQRS và ES [17]	62
Hình 3.1.1 Tổng quan kiến trúc mô hình hệ thống cung cấp dịch vụ	64
Hình 3.1.2 Hình ảnh minh họa cho Direct Pattern [20]	65
Hình 3.1.3 Hình ảnh minh họa cho mô hình API Gateway [20]	66
Hình 3.1.4 Minh họa mô hình Backend for frontend [20]	68
Hình 3.1.5 Hình mô tả nguyên tắc ACID [21]	69
Hình 3.1.6 Mô tả phương pháp Eventual consistency [22]	70
Hình 3.1.7 Hình minh họa message queue truyền dữ liệu [23]	71

Hình 3.1.8 Hình minh họa giao tiếp giữa dịch vụ khi không có và có message broker [24]	72
Hình 3.1.9 Minh họa Sagas cho quá trình đặt hàng	75
Hình 3.2.1 Giao diện trang chủ (phần 1)	76
Hình 3.2.2 Giao diện trang chủ (phần 2)	77
Hình 3.2.3 Giao diện trang dùng thử dịch vụ	77
Hình 3.2.4 Giao diện trang kết quả khi sử dụng dịch vụ	78
Hình 3.2.5 Giao diện trang thống kê, phần thống kê chi tiết	79
Hình 3.2.6 Giao diện trang thống kê, phần thống kê tổng	80
Hình 3.2.7 Giao diện trang nâng cấp API key	81
Hình 3.2.8 Mô hình mẫu CQRS kết hợp Event Sourcing áp dụng trong hệ thống [21]	83
Hình 5.2.1 Kết quả Loading test cho luồng ghi trên chức năng tạo mới mã thanh toán	107
Hình 5.2.2 Kết quả Loading test cho luồng đọc dữ liệu truy vấn tất cả người dùng	107
Hình 5.2.3 Kết quả Loading test trên chức năng nhận dạng âm thanh	108
Hình 5.2.4 Kết quả Stress test trên luồng truy vấn tất cả thông tin người dùng	108
Hình 5.2.5 Kết quả Stress test trên chức năng tạo mới API key	109
Hình phụ lục 6.1.1 SDK của JavaScript trên NPM package	116
Hình phụ lục 6.1.2 SDK của CSharp trên NuGet	117
Hình phụ lục 6.2.1 Mô tả cách giao tiếp của các ứng dụng tới Web API	119
Hình phụ lục 6.3.1 Màn hình trang chủ của ứng dụng	120

Hình phụ lục 6.3.2 Màn hình danh sách chủ đề của ứng dụng.....	121
Hình phụ lục 6.3.3 Màn hình danh sách bài học có trong một chủ đề cụ thể	122
Hình phụ lục 6.3.4 Màn hình chi tiết bài học	123
Hình phụ lục 6.3.5 Màn hình câu hỏi trong bài học.....	124
Hình phụ lục 6.3.6 Màn hình sau khi hoàn thành một bài học.....	125
Hình phụ lục 6.3.7 Màn hình lịch sử các bài đã và đang học.....	126

DANH MỤC BẢNG

Bảng 1.5.1 Mô tả chi tiết các chức năng của hệ thống.....	13
Bảng 3.3.1 Tổ chức các thuộc tính để lưu trữ dữ liệu truy cập dịch vụ của người dùng.	82
Bảng 4.4.1 Tổ chức các thuộc tính để lưu trữ dữ liệu lời mời tham gia dự án.....	97
Bảng 4.4.2 Mô tả chi tiết các lỗi khi gọi API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt của hệ thống VietSpeech	98
Bảng 5.2.1 Các thư viện, nền tảng sử dụng trong quá trình xây dựng hệ thống	106
Bảng 5.2.2 Bảng tóm tắt kết quả Loading test cho hệ thống.....	110
Bảng 5.2.3 Bảng tóm tắt kết quả Stress test cho hệ thống	110
Bảng 5.2.4 So sánh chức năng giữa các hệ thống cung cấp dịch vụ	112
Bảng 5.3.1 Các thư viện, nền tảng sử dụng trong quá trình xây dựng ứng dụng di động	113
Bảng 5.4.1 Bảng so sánh mục tiêu ban đầu với kết quả thu được	114

DANH MỤC KÝ TỰ VÀ CÁC TỪ VIẾT TẮT

API	Application programming interface
VietSpeech	Hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt của nhóm
API key	Khoá truy cập sử dụng dịch vụ nhận dạng âm thanh tiếng Việt của hệ thống VietSpeech

TÓM TẮT KHÓA LUẬN

Luận văn là tài liệu chính của khóa luận, nội dung của luận văn sẽ đề cập tới các kiến thức, kỹ thuật liên quan trong quá trình thực hiện khóa luận. Luận văn của nhóm chúng em bao gồm những phần như sau:

Chương 1 – Giới thiệu đề tài: trình bày lý do xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt và những giải pháp sẽ áp dụng để xây dựng. Đồng thời, mô tả những điểm tốt, điểm chưa hoàn thiện của những mô hình, giải pháp này, các mục tiêu đề ra và đóng góp của ứng dụng đối với cộng đồng.

Chương 2 – Lý thuyết nền tảng: Trình bày tóm tắt lý thuyết nền tảng về mô hình Microservices, Domain Driven Design, Command Query Responsibility Segregation, Event Sourcing và trình bày cơ sở lý thuyết của các chức năng trong ứng dụng.

Chương 3 – Thiết kế giải pháp: Nêu ra các giải pháp tổng quát cho kiến trúc, chức năng của hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt cùng với đó chứng minh, lý giải cho việc chọn giải pháp ấy. Ngoài ra, chương 3 còn trình bày một số phương pháp liên quan làm tiền đề để xây dựng.

Chương 4 – Cài đặt giải pháp: Đề cập các công cụ dùng để cài đặt từng giải pháp hoặc thành phần đã nói ở trong chương 3, đề cập cách hoạt động, cách cài đặt chi tiết của một số chức năng. Bên cạnh đó, chương 4 còn đề cập đến một số chức năng và một số khó khăn trong quá trình cài đặt và giải pháp cho chúng.

Chương 5 – Tổng kết, đánh giá: Phần này sẽ liệt kê các kiến thức mà nhóm chúng em đạt được trong quá trình thực hiện luận văn, sản phẩm thu được từ quá trình thực hiện luận văn, những chức năng đã thực hiện được. Trong chương này còn thể hiện một số phương hướng phát triển của hệ thống trong tương lai.

Phụ lục: Trình bày một số khái niệm bổ sung trong lập trình dịch vụ Web mà chương 4 chưa nhắc tới, các thư viện mà hệ thống cung cấp phục vụ người dùng

trong quá trình tích hợp. Ngoài ra, phụ lục còn trình bày về giao diện chi tiết của ứng dụng học nói tiếng Việt thông qua hình ảnh, tích hợp giọng nói.

CHƯƠNG 1: GIỚI THIỆU

1.1 GIỚI THIỆU LUẬN VĂN

Giao tiếp luôn là một hình thức quan trọng trong việc thấu hiểu và kết nối giữa hai hoặc nhiều cá thể với nhau. Thông qua giọng nói, nó cho phép người nói biểu đạt ý nghĩ, mong muốn, cảm xúc đến người nghe. Tuy nhiên, quá trình giao tiếp không chỉ diễn ra giữa người với người, nó còn diễn ra giữa người với thiết bị, máy móc. Đúng như ý nghĩa của nó, giao tiếp cho phép con người “nói chuyện” được với máy móc, cho phép con người truyền đạt mong muốn, yêu cầu đến máy móc. Dựa vào đó, nhu cầu tự động hoá của con người được đáp ứng. Nhu cầu sử dụng các thiết bị công nghệ hiện đại phục vụ cho nhiều mục đích khác nhau trong đời sống của con người ngày càng tăng đồng nghĩa với nhu cầu kết nối, “nói chuyện” với máy móc, thiết bị của con người ngày càng tăng.

Nhận thấy được tiềm năng phát triển trong nhu cầu này, nhiều tổ chức đã xây dựng và phát triển các hệ thống cung cấp dịch vụ nhận dạng âm thanh. Tuy nhiên, những hệ thống này có nhiều điểm bất lợi đối với các lập trình viên và các công ty khởi nghiệp, có thể kể đến như chi phí sử dụng dịch vụ cao, nguồn tài nguyên hạn chế, tài liệu khá ít ỏi, các chức năng còn ít, đơn giản hoặc khó sử dụng, không nhiều hệ thống hỗ trợ nhận dạng âm thanh tiếng Việt.

Để đáp ứng nhu cầu kết nối của con người đồng thời khắc phục các điểm hạn chế đã nêu, nhóm chúng em đã xây dựng một hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt mang tên Viet Speech. Hệ thống cho phép người dùng truy cập sử dụng dịch vụ nhận dạng âm thanh tiếng Việt. Bên cạnh đó, hệ thống hỗ trợ tích hợp vào các nền tảng phần mềm nhờ vào những thư viện khác nhau thông qua các ngôn ngữ lập trình phổ biến, đảm bảo tối ưu về mặt chi phí, kỹ thuật, đa dạng về chức năng, dễ bảo trì và dễ sử dụng.

1.1.1 Lợi ích thực tế

Với thời đại kinh tế phát triển dựa vào lao động có tay nghề cao như hiện nay,

kéo theo đó quỹ thời gian của mỗi người trong ngày sẽ phải tối ưu hóa triệt để nhằm phục vụ cho chất lượng công việc. Cũng vì lẽ đó việc sử dụng các dịch vụ chuyển đổi giọng nói thay thế cho các tác vụ thủ công như ghi chép cho các cuộc họp, ghi chép các cuộc trao đổi điện thoại, soạn các tin nhắn văn bản trên điện thoại,... là rất cần thiết.

Hiện nay, với nhu cầu sử dụng ngày càng nhiều ở thị trường di động, việc phát triển ra một ý tưởng mới cho các nhà phát triển phần mềm ngày càng trở nên khó khăn. Dưới sự ra đời của một hệ thống nhận dạng tiếng nói hợp lý, các nhà phát triển phần mềm không chỉ có thể sáng tạo ra nhiều ý tưởng sản phẩm mới mà còn có thể nâng cấp cải tiến cách thức sử dụng các sản phẩm có sẵn, từ đó ngày càng tạo ra sự tiện dụng và thu hút khách hàng nhiều hơn.

Ví dụ như Google Maps trước đây khi người dùng muốn tìm đường đến một địa điểm nào đó, họ phải nhập địa điểm và thực hiện các thao tác khá phức tạp khác, nhưng với việc tích hợp Google Assistant, Google Maps hiện nay đã cho phép người dùng điều khiển được việc chuyển hướng, chỉ đường bằng giọng nói trong khi họ đang bận rộn với việc lái xe và không thể sử dụng tay để điều khiển ứng dụng.

Ứng dụng cho phép người dùng giao tiếp bằng giọng nói đang thu hút một lượng lớn người dùng trên nhiều độ tuổi, đặc biệt là nhóm người dùng bận rộn với nhiều việc cùng phải làm trong cùng lúc. Đối với những ứng dụng đã có sẵn trên thị trường hoàn toàn có thể sử dụng dịch vụ này để nâng cấp chức năng cho phép người dùng ra lệnh bằng giọng nói, còn đối với những ứng dụng mới, hoàn toàn có thể dựa vào việc khai thác dịch vụ này để phục vụ ý tưởng cho nhà phát triển phần mềm.

Với những công ty công nghệ phần mềm và thiết bị phần cứng, có những sự lựa chọn như sử dụng tiếp dịch vụ nhận dạng tiếng nói hoặc tùy chỉnh lại dịch vụ để phục vụ cho mục đích riêng của họ. Khả năng tích hợp dịch vụ này vào các ứng dụng phần mềm và các thiết bị phần cứng là rất cao, làm cho thị trường nhận dạng giọng nói phong phú, đa dạng và vì thế lợi nhuận đem lại từ thị trường này là rất to

lớn.

1.1.2 Ứng dụng thực tiễn

Dữ liệu âm thanh là một loại dữ liệu ẩn chứa trong đó nhiều thông tin riêng biệt, mà thông qua những thông tin được trích xuất ra từ các loại hình dữ liệu này, ta có thể vận dụng chúng vào trong những mục đích khác nhau. Sau đây sinh viên sẽ đi qua các sản phẩm nổi bật để làm rõ sự đa dạng của loại hình ứng dụng này trên nhiều lĩnh vực.

- Amazon Echo: một thiết bị gia đình thông minh có cơ chế hoạt động giống một chiếc loa cầm tay. Bạn có thể thực hiện nhiều việc với Echo bằng cách ra lệnh giọng nói như: nghe nhạc, thiết lập lịch trình, cập nhật tin tức,... Tuy nhiên mục tiêu của Amazon Echo ra đời là không phải là để cạnh tranh trong lĩnh vực nhận diện giọng nói, mà là để phục vụ cho việc bán hàng qua mạng. Thay vì người dùng phải lên trang web tìm kiếm sản phẩm, thì nay có thể ra lệnh cho Echo thực hiện việc đó thông qua giọng nói.
- Google Assistant: một trợ lý ảo cá nhân được phát triển bởi Google. Đây là một sản phẩm sinh ra cho cuộc cạnh tranh trong lĩnh vực nhận diện giọng nói, người dùng có thể tìm kiếm trên Internet, đặt lịch sự kiện và báo thức, tham gia vào các cuộc trò chuyện hai chiều giữa phần mềm và người dùng. Sự kết hợp giữa Google Assistant với Google Maps, Google Photos, Youtube và một loạt các dịch vụ tiện ích khác tạo nên hệ sinh thái vô cùng mạnh mẽ.
- Cortana: một trợ lý ảo được phát triển bởi Microsoft. Nhưng thay vì bước vào cuộc cạnh tranh trong lĩnh vực nhận diện giọng nói với Google Assistant, hay phục vụ bán hàng như Amazon Echo, thì Microsoft hướng Cortana là một công cụ tăng năng suất trong phần mềm và tập trung mạnh vào các doanh nghiệp. Tương lai của Cortana có thể không phải là một sản phẩm tiêu dùng thú vị mà là xương sống cho các giải pháp trợ lý

ảo mà các doanh nghiệp có thể tùy chỉnh thành trợ lý ảo hoặc chatbox riêng của họ.

- Siri: một ứng dụng trợ lý ảo phát triển bởi Apple cho phép nhận diện giọng nói để làm các công việc mà bình thường phải làm bằng tay ví dụ như: gọi điện, soạn và gửi tin nhắn, mở đèn flash, mở ứng dụng cài trên máy, kiểm tra thời tiết, tạo các ghi chú, tìm đường, gửi email, bật nhạc hay điều chỉnh độ sáng màn hình... Người sử dụng có thể đưa ra câu hỏi và Siri sẽ tìm ra câu trả lời, hoặc người sử dụng có thể ra yêu cầu để Siri thực hiện trong một phạm vi cho phép.

Thông qua các ứng dụng đã trình bày, ta có thể thấy được nguồn dữ liệu âm thanh là một nguồn dữ liệu bao hàm rất nhiều thông tin, và bằng những kỹ thuật khai thác riêng biệt, ta có thể tận dụng nguồn thông tin này để phục vụ cho nhiều lĩnh vực kinh doanh chiến lược khác nhau.

1.2 KHẢO SÁT THỊ TRƯỜNG

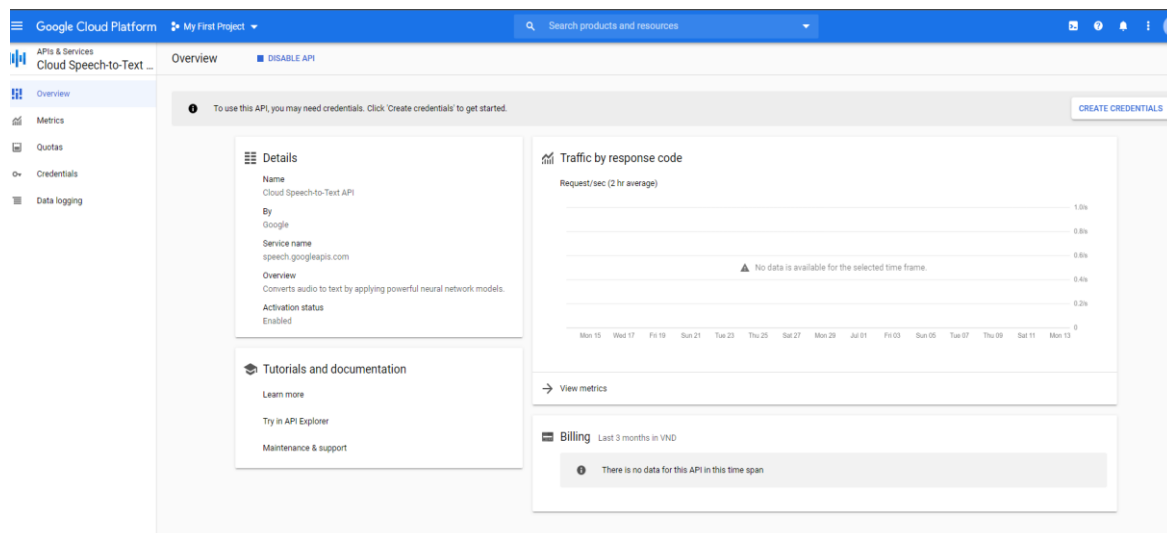
Trên thị trường Việt Nam hiện nay, qua khảo sát và trải nghiệm, nhóm nhận thấy có ba hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt nổi bật, đó là:

- Cloud Speech-to-Text do Google phát triển.
- FPT.AI Speech to Text do FPT phát triển.
- Viettel AI Speech to Text do Viettel phát triển.

1.2.1 Cloud Speech-to-Text do Google phát triển

Hệ thống cung cấp dịch vụ Cloud Speech-to-Text do Google phát triển chỉ cung cấp chức năng cho việc quản lý API key và theo dõi tốc độ truy cập theo thời gian. Mặc dù có nhiều cấu hình cho việc truy cập dịch vụ nhưng điều đó cũng đồng nghĩa với việc đòi hỏi người dùng phải đọc qua rất nhiều tài liệu. Ngoài ra, hệ thống chưa hỗ trợ người dùng chia sẻ khoá truy cập dịch vụ, lưu trữ các lượt truy cập, cũng như kết quả truy cập, hỗ trợ chỉnh sửa và tải xuống văn bản đã dịch khi dùng

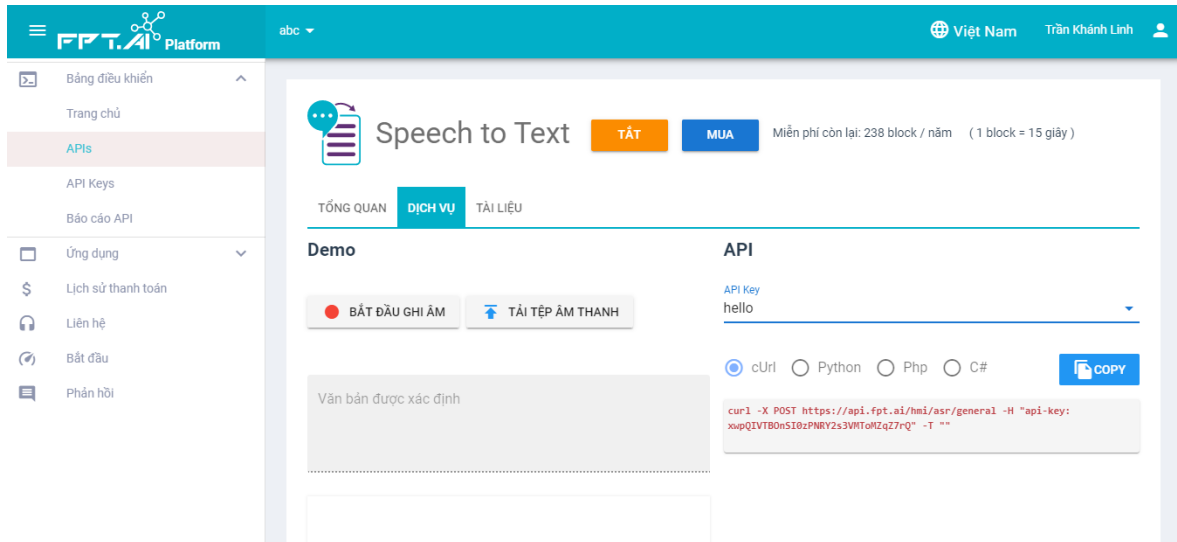
thử dịch vụ, báo cáo số lượng phút sử dụng theo các mốc ngày, tháng, năm,...



Hình 1.2.1 Giao diện trang chủ của hệ thống dịch vụ Cloud Speech-to-Text API [1]

1.2.2 FPT.AI Speech To Text do FPT phát triển

Hệ thống cung cấp dịch vụ FPT.AI Speech To Text của FPT cung cấp một số chức năng khá giống với hệ thống của nhóm, đó là quản lý các API key theo dự án, cung cấp báo cáo về việc sử dụng dịch vụ của người dùng có thể lọc theo nhiều loại thời gian. Nhìn chung, hệ thống có thể đáp ứng các chức năng cơ bản của một hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt. Tuy nhiên, khi dùng thử dịch vụ, hệ thống chưa hỗ trợ chỉnh sửa văn bản và tải xuống kết quả cũng như lưu trữ các lần sử dụng thử dịch vụ. Việc chia sẻ sử dụng API key trong dự án với người dùng khác cũng chưa được hỗ trợ. Bên cạnh đó, màu sắc giao diện và tính tương tác của giao diện với người dùng chưa được tốt.

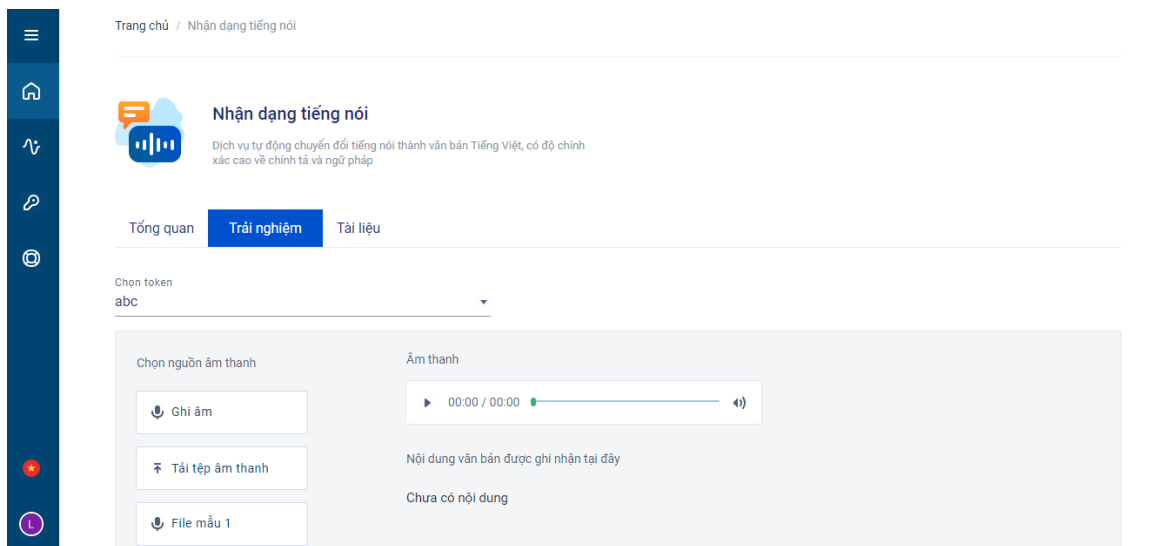


Hình 1.2.2 Giao diện trang dùng thử dịch vụ của hệ thống cung cấp dịch vụ FPT.AI [2]

1.2.3 Viettel AI Speech to Text do Viettel phát triển

Về mặt chức năng, hệ thống cung cấp dịch vụ Viettel AI Speech to Text của Viettel tương tự với hệ thống của FPT.AI ngoại trừ việc hệ thống không quản lý các API key theo dự án. Đặc biệt, độ chính xác trong việc nhận dạng âm thanh tiếng Việt của Viettel AI có phần cao hơn so với FPT.AI.

Về mặt thiết kế, có thể thấy giao diện của hệ thống Viettel AI có bố cục, màu sắc và tính tương tác với người dùng tốt hơn nhiều so với FPT.AI.



Hình 1.2.3 Giao diện trang dùng thử dịch vụ của hệ thống cung cấp dịch vụ Viettel AI [3]

1.3 LÝ DO LỰA CHỌN ĐỀ TÀI

Nhằm mục đích hệ thống hóa các kiến thức của bản thân trong quá trình học tập vào một sản phẩm có ý nghĩa thực tế, có tiềm năng đầu tư cao trong tương lai, nhóm chúng em lựa chọn đề tài “Xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt”.

Như đã trình bày ở trên, ứng dụng sử dụng dịch vụ nhận diện giọng nói ngày càng thu hút các nhà phát triển phần mềm lẫn người tiêu dùng phần mềm. Thị trường Việt Nam đang là thị trường tiềm năng vì đây là lĩnh vực mới. Hiện nay hầu hết các ứng dụng nhận dạng giọng nói tiếng Việt đang sử dụng dịch vụ từ Google Cloud Speech API và một phần nhỏ sử dụng dịch vụ từ FPT.AI.

Có thể thấy, chi phí chi trả cho việc sử dụng các dịch vụ này khá cao. Ví dụ như Google Cloud Speech API, thì mức phí là:

Pricing Table

Feature	Standard models (all models except enhanced video and phone call)		Enhanced models (video, phone call)	
	0-60 Minutes	Over 60 Mins up to 1 Million Mins	0-60 Minutes	Over 60 Mins up to 1 Million Mins
Speech Recognition (without Data Logging - default)	Free	\$0.006 / 15 seconds **	Free	\$0.009 / 15 seconds **
Speech Recognition (with Data Logging opt-in)	Free	\$0.004 / 15 seconds **	Free	\$0.006 / 15 seconds **

Hình 1.3.1 Mức phí cho dịch vụ của Google Cloud Speech API [4]

Chi phí cho dịch vụ được cung cấp bởi FPT.AI là:

Pricing

Plan	Quantity	Price (VND)	Description
Trial	60 minutes	Free of charge	60 minutes/ year No technical support
Business	2,000 minutes	1,400,000	Standard Technical Support
	5,000 minutes	3,100,000	
	10,000 minutes	5,400,000	
Business Premium	Over 10,000 minutes	Please contact for detailed pricing Hotline: 0911886353 Email: support@fpt.ai	

Hình 1.3.2 Mức phí cho dịch vụ của FPT.AI [5]

Xét trong phân khúc thị trường ở Việt Nam, một dịch vụ nhận diện giọng nói tiếng Việt với độ chính xác chưa cao và mức phí khá đắt đỏ đang vô hình làm chậm lại sự phát triển của các phần mềm tự động.

Các nhà đầu tư mới bắt đầu sẽ không có đủ nguồn lực và nguồn vốn để chi trả

cho việc phát triển một ứng dụng có tích hợp dịch vụ nhận diện giọng nói tiếng Việt. Với hai lý do kể trên, nhóm chúng em quyết định chọn đề tài “Xây dựng hệ thống cung cấp dịch vụ nhận dạng tiếng Việt” để tạo ra một dịch vụ có mức phí thấp hơn (hoặc miễn phí) và với độ chính xác chấp nhận được.

1.4 MỤC TIÊU CỦA LUẬN VĂN

Nhóm chúng em xác định rõ các mục tiêu cần đạt được khi thực hiện luận văn như sau:

- ❖ Trình bày lý do xây dựng hệ thống cung cấp dịch vụ web nhận dạng âm thanh tiếng Việt.
- ❖ Trình bày tóm tắt lý thuyết nền tảng về Microservices, Domain Driven Design (DDD), Command Query Responsibility Segregation (CQRS) và Event Sourcing để thiết kế và hiện thực hóa khả năng mở rộng dịch vụ khi số lượng truy cập cao.
- ❖ Xây dựng dịch vụ web (API) để nhận một tập tin (file) âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản.
 - Quản lý người đăng ký, khóa truy cập, số lượng truy cập vào dịch vụ:
 - Cho phép người dùng đăng ký, và thanh toán chi phí sử dụng dịch vụ. Cho phép người quản trị quản lý người đăng ký, khóa truy cập sử dụng dịch vụ.
 - Cung cấp API cung cấp dịch vụ chuyển một tập tin âm thanh tiếng Việt thành nội dung văn bản.
 - Cung cấp các báo cáo về việc sử dụng dịch vụ của người dùng.
 - Tạo thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử dụng dịch vụ, với các ngôn ngữ khác nhau. SDK hỗ trợ tối thiểu hai loại ứng dụng là web và mobile (android hoặc iOS), ít nhất có ví dụ cho Java, Javascript (hoặc PHP, Swift) cho mỗi loại web hoặc mobile.

- Thiết kế và hiện thực hóa khả năng mở rộng dịch vụ, khi số lượng truy cập cao:
 - Áp dụng Microservices, CQRS và Event Sourcing.
 - Mô phỏng, thiết kế, thực hiện, báo cáo kết quả các kịch bản kiểm thử khả năng tải mong muốn (load tests), và khả năng chịu tải tối đa (stress tests).
- ❖ Xây dựng một trang web và một mobile client demo việc sử dụng SDK để tải lên một tập tin âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản (ví dụ: Trò chơi hỗ trợ học tiếng Việt).
- ❖ Viết 120 trang luận văn theo luồng logic trình bày trong tài liệu “Hướng dẫn thực hiện luận văn” mà giáo viên cung cấp, theo đúng chuẩn nhà trường yêu cầu và trích dẫn tài liệu tham khảo một cách chi tiết, đầy đủ.

1.5 YÊU CẦU CHỨC NĂNG

Để hoàn thành tốt luận văn, nhóm chúng em đề xuất các chức năng mà một hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt cần phải đáp ứng như sau.

STT	Tên chức năng	Mô tả chi tiết về chức năng
1	Đăng ký/ đăng nhập	Cho phép người dùng đăng ký tài khoản (các tài khoản có tên đăng nhập và email không được trùng nhau), đăng nhập bằng tài khoản đã được đăng ký với hệ thống hoặc bằng tài khoản Facebook, Google. Mỗi người dùng khi đăng ký đều được cấp một API key miễn phí.
2	Tạo dự án	Người dùng quản lý các API key (có được sau khi đã thanh toán sử dụng dịch vụ) của mình bằng các dự án. Người dùng có thể tạo nhiều dự án bằng cách cung cấp tên (bắt buộc

		và không được trùng) và mô tả (không bắt buộc). Mỗi dự án chứa nhiều API key.
3	Thanh toán chi phí sử dụng dịch vụ	Hệ thống đưa ra nhiều loại API key với các mức giá khác nhau cho người dùng lựa chọn. Người dùng chọn loại API key phù hợp và thực hiện thanh toán chi phí sử dụng dịch vụ để được cấp một API key, phục vụ cho việc sử dụng dịch vụ nhận dạng hệ thống cung cấp. Khi thanh toán, người dùng phải chọn một dự án, đặt tên cho API key và cung cấp thông tin thẻ tín dụng. Sau đó, tiến hành giao dịch. Giao dịch thành công, hệ thống sẽ hiển thị API key cho người dùng.
4	Xem lịch sử giao dịch	Người dùng có thể xem lịch sử những lần thanh toán chi phí sử dụng dịch vụ. Lịch sử bao gồm mã giao dịch, trạng thái (đang xử lý/ thành công/ thất bại), thời gian giao dịch, API key được cấp nếu giao dịch thành công, loại API key được cấp.
5	Mời tham gia dự án	Người dùng có thể mời người dùng khác tham gia vào dự án của mình. Người được mời có quyền sử dụng chung API key với chủ dự án nếu chấp nhận tham gia.
6	Sử dụng thử dịch vụ	Người dùng có thể dùng thử dịch vụ với các API key của mình. Bằng việc cung cấp một tập tin âm thanh tiếng Việt và chọn một API key, hệ thống sẽ thực hiện chuyển đổi tập tin âm thanh này thành nội dung văn bản, hỗ trợ người

		dùng chỉnh sửa văn bản và tải xuống dưới dạng tập tin có định dạng *.docx.
7	Xem lịch sử sử dụng dịch vụ	Người dùng có thể xem lịch sử những lần sử dụng dịch vụ nhận dạng của hệ thống. Lịch sử bao gồm tên dự án, tên API key, tên tập tin âm thanh, kích thước tập tin (phút), trạng thái (đang xử lý/ thành công/ thất bại), thời gian sử dụng.
8	Cung cấp API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt	Cung cấp một API xác thực người dùng bằng API key đã mua, chuyển một tập tin âm thanh tiếng Việt thành nội dung văn bản.
9	Nâng cấp API key	Người dùng có thể gia hạn thời gian sử dụng dịch vụ hợp lệ của API key bằng cách thực hiện nâng cấp API key. Người dùng phải chọn dự án, tên API key cần nâng cấp, loại API key muốn nâng cấp lên và cung cấp thông tin thẻ tín dụng. Sau đó, tiến hành nâng cấp.
10	Báo cáo về việc sử dụng dịch vụ của người dùng	Hệ thống cung cấp các loại báo cáo đa dạng về việc sử dụng dịch vụ của người dùng, bao gồm báo cáo chi tiết theo từng dự án, API key, loại API key; báo cáo tổng quát theo dự án, API key, loại API key. Mỗi loại báo cáo đều có thể lọc theo ngày, theo tuần, theo tháng, theo quý và theo năm.
11	Cung cấp thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử	Hệ thống cung cấp thư viện SDK một cách chi tiết, hỗ trợ lập trình viên từng bước thiết lập sử dụng dịch vụ nhận dạng âm thanh tiếng Việt

	dụng dịch vụ, với các ngôn ngữ khác nhau	mà hệ thống cung cấp với API key được cấp.
--	--	--

Bảng 1.5.1 Mô tả chi tiết các chức năng của hệ thống

1.6 PHẠM VI ĐỀ TÀI

Với đề tài “Xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt”, đối tượng chính mà đề tài hướng tới là các lập trình viên, các công ty, đối tác có mong muốn tích hợp việc sử dụng dịch vụ nhận dạng âm thanh tiếng Việt mà hệ thống cung cấp vào sản phẩm của mình. Điều này đồng nghĩa với việc số lượng truy cập dịch vụ của hệ thống sẽ cao.

Do đó, hệ thống sẽ không chú trọng cải thiện chất lượng nhận dạng âm thanh tiếng Việt của dịch vụ mà sẽ tập trung thiết kế và hiện thực hoá khả năng mở rộng dịch vụ nhận dạng âm thanh tiếng Việt bên cạnh việc cài đặt các chức năng hỗ trợ người dùng liên quan được liệt kê trong bảng 1.5.1. Tuy nhiên, dịch vụ nhận dạng mà hệ thống cung cấp vẫn đạt chất lượng ở mức chấp nhận được.

CHƯƠNG 2: LÝ THUYẾT NỀN TẢNG

2.1 MICROSERVICES

Trong những năm gần đây, thuật ngữ “microservices” hay kiến trúc microservice (microservice architecture) đã trở nên khá quen thuộc và rất được quan tâm trong giới phát triển phần mềm. Chúng ta có thể thấy rất nhiều dự án phần mềm được xây dựng và phát triển dựa trên microservices. Vậy microservices thực sự là gì?

Để trả lời câu hỏi trên, chúng ta sẽ tìm hiểu các khái niệm cốt lõi, các tính chất cũng như một số vấn đề gặp phải khi xây dựng và phát triển phần mềm theo hướng microservices.

2.1.1 Định nghĩa

Sự liên quan giữa thuật ngữ “microservices” và cách đặc biệt để xây dựng và phát triển phần mềm đến từ một buổi họp mặt với một số lượng ít các kỹ sư phần mềm tham dự. Tại đây, các kỹ sư thấy được sự tương đồng trong cách phát triển phần mềm của một tập hợp các công ty và đặt tên cho cách phát triển đó [6].

2.1.1.1 Microservices là một kiểu kiến trúc phần mềm (architectural style)

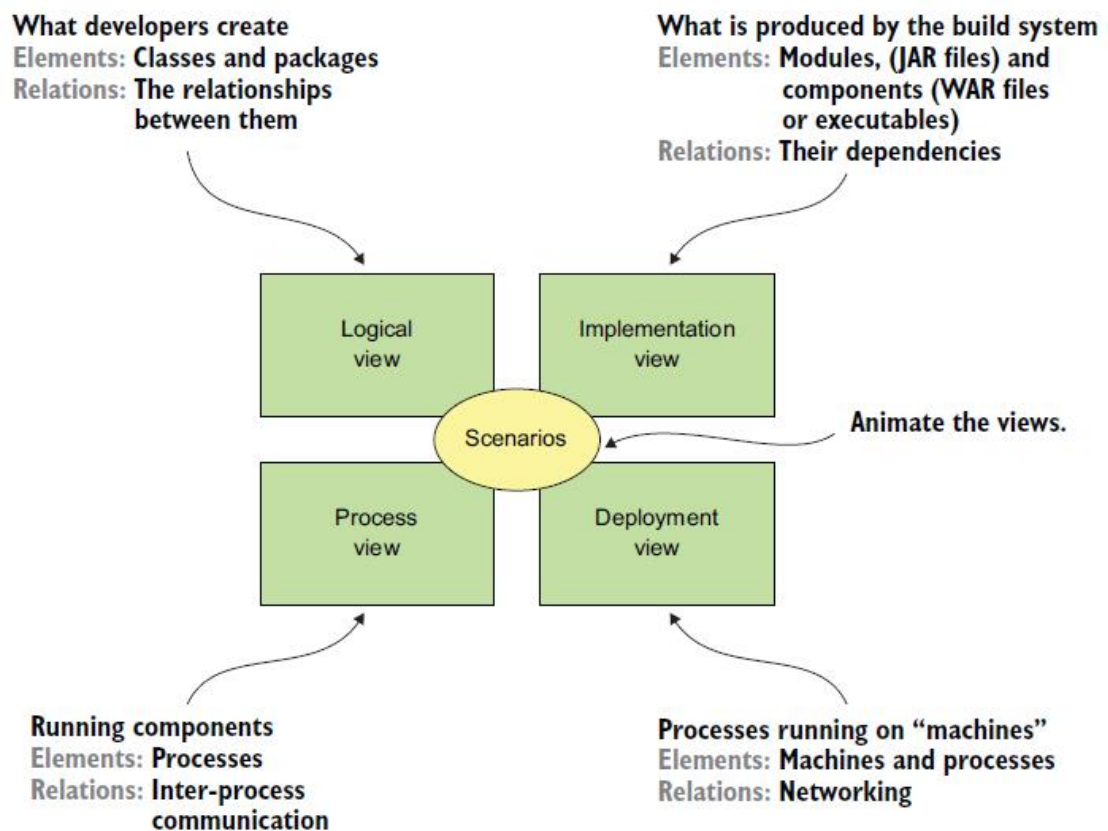
2.1.1.1.1 Kiến trúc phần mềm (software architecture)

Hãy bắt đầu phân định nghĩa bằng cách tiếp cận thuật ngữ kiến trúc phần mềm. Có rất nhiều định nghĩa về kiến trúc phần mềm. Nhưng định nghĩa mà nhóm chúng em muốn giới thiệu đến từ Len Bass – một kỹ sư phần mềm người Mỹ và những đồng nghiệp của ông tại Viện kỹ thuật phần mềm [7]. Họ định nghĩa kiến trúc phần mềm như sau: Kiến trúc phần mềm của một hệ thống sử dụng máy tính (computing system) là một tập hợp các cấu trúc cần thiết để giải quyết những vấn đề của hệ thống, bao gồm các yếu tố phần mềm (software elements), mối quan hệ giữa chúng và những tính chất (properties) của cả hai [8]. Từ định nghĩa, kiến trúc của một ứng dụng là sự phân rã thành nhiều thành phần (các yếu tố phần mềm) và mối quan hệ giữa các thành phần đó. Sự phân rã hệ thống đóng vai trò quan trọng vì hai lý do

sau:

- Nó tạo điều kiện cho việc phân chia công việc và kiến thức. Nghĩa là, nó cho phép nhiều người (hoặc đội ngũ) với kiến thức chuyên môn khác nhau làm việc trên cùng một ứng dụng.
- Nó giải thích cách tương tác của các yếu tố phần mềm.

Cụ thể hơn, hãy tìm hiểu kiến trúc của một ứng dụng từ nhiều góc nhìn thông qua mô hình 4+1 (4+1 view model) của Phillip Krutchen [9]. Mô hình này giải thích bốn góc nhìn khác nhau của một kiến trúc phần mềm. Mỗi góc nhìn chứa một tập hợp yếu tố phần mềm và mối quan hệ giữa chúng.



Hình 2.1.1 Mô hình 4+1 sử dụng bốn góc nhìn và những kịch bản (thể hiện sự hợp tác trong việc xử lý yêu cầu giữa những yếu tố trong mỗi khía cạnh) để mô tả kiến trúc của một ứng dụng [10]

Trong đó:

- Góc nhìn logic (logical view): Những yếu tố phần mềm được tạo bởi lập trình viên. Trong ngôn ngữ hướng đối tượng (object-oriented languages), những yếu tố này là lớp (classes) và gói (packages) – tập các lớp được nhóm lại với nhau theo một qui định nào đó. Mỗi liên hệ giữa chúng là mối quan hệ giữa lớp và gói, bao gồm kế thừa (inheritance), liên kết (associations), phụ thuộc (depends-on).
- Góc nhìn thực hiện (implementation view): kết quả (output) của việc xây dựng hệ thống, bao gồm nhiều mô-đun (module) – gói chức năng (functional package), phản ánh một tính năng kỹ thuật (technical capability) trong ứng dụng và nhiều thành phần (component) – những đơn vị trong hệ thống có thể triển khai (deployable) hoặc thực thi (executable), gồm một hay nhiều mô-đun. Trong ngôn ngữ Java, một mô-đun là một tập tin JAR (Java archive) và một thành phần là một tập tin WAR (web application archive) hoặc một tập tin JAR có thể thực thi. Mỗi liên hệ giữa chúng là mối quan hệ phụ thuộc giữa các mô-đun và kết hợp (composition) giữa mô-đun và thành phần.
- Góc nhìn quá trình (process view): Các thành phần trong thời gian thực thi (runtime). Mỗi thành phần là một tiến trình (process) và mỗi liên hệ giữa những tiến trình này tượng trưng cho sự giao tiếp liên tiến trình (interprocess communication - IPC).
- Góc nhìn triển khai (deployment view): mô tả sự ánh xạ các tiến trình vào máy móc (machines). Những yếu tố phần mềm bao gồm máy móc vật lý (physical) hoặc ảo (virtual) và những tiến trình. Góc nhìn này thể hiện mối quan hệ giữa những tiến trình và máy móc.
- Những kịch bản (scenarios) – tượng trưng cho “+1” trong “4+1”: kịch bản chính là nơi mô tả cách mà những thành phần kiến trúc trong một góc nhìn phối hợp để xử lý một yêu cầu.

Nhìn chung, mô hình 4+1 là một cách mô tả xuất sắc kiến trúc của một ứng dụng. Mỗi góc nhìn mô tả một khía cạnh quan trọng của kiến trúc, những kịch bản diễn giải cách mà những yếu tố của một góc nhìn hợp tác với nhau.

Vậy kiến trúc phần mềm có tầm ảnh hưởng như thế nào đối với ứng dụng? Một ứng dụng có hai loại yêu cầu, một là yêu cầu chức năng (functional requirements) giải thích ứng dụng phải làm được gì. Và kiến trúc thể hiện tầm quan trọng của nó thông qua việc cho phép ứng dụng đáp ứng loại yêu cầu thứ hai: yêu cầu về chất lượng dịch vụ (quality of service requirements) hay còn gọi là yêu cầu phi chức năng (nonfunctional requirements). Yêu cầu phi chức năng xác định chất lượng ứng dụng trong thời gian thực thi (runtime qualities) như khả năng mở rộng (scalability) và độ tin cậy (reliability). Đồng thời, nó còn xác định chất lượng ứng dụng trong thời gian phát triển (development time qualities) như khả năng bảo trì (maintainability), kiểm tra (testability) và khả năng triển khai (deployability). Như vậy, kiến trúc quyết định mức độ ứng dụng đáp ứng những yêu cầu về chất lượng này.

2.1.1.1.2 Kiến trúc phần mềm nguyên khối (monolithic architecture)

Microservices là một kiểu kiến trúc phần mềm và kiến trúc nguyên khối cũng là một kiểu kiến trúc phần mềm. Trước khi kiến trúc microservice xuất hiện, các ứng dụng thường được xây dựng và phát triển dựa theo kiến trúc nguyên khối. Và kể từ khi kiến trúc microservice ra đời, đó cũng là lúc các ứng dụng doanh nghiệp hay ứng dụng có quy mô lớn, phức tạp (enterprise applications) thay đổi kiến trúc, cân nhắc lựa chọn microservices. Lý do cho sự thay đổi này là gì? Microservices là kiểu kiến trúc như thế nào? Trước khi đến với định nghĩa tiếp theo về kiến trúc microservice, cùng tìm hiểu về kiến trúc phần mềm nguyên khối.

Kiến trúc nguyên khối là kiểu kiến trúc mà góc nhìn thực hiện được tổ chức dưới dạng một thành phần đơn lẻ có thể thực thi hay triển khai. Đối với các ứng dụng có quy mô tương đối nhỏ, kiến trúc này là một sự lựa chọn tốt. Tuy nhiên, khi ứng dụng phát triển, quy mô và yêu cầu chức năng dần trở nên phức tạp, sự lựa

chọn này khiến ứng dụng gặp rất nhiều khó khăn về mặt kỹ thuật.

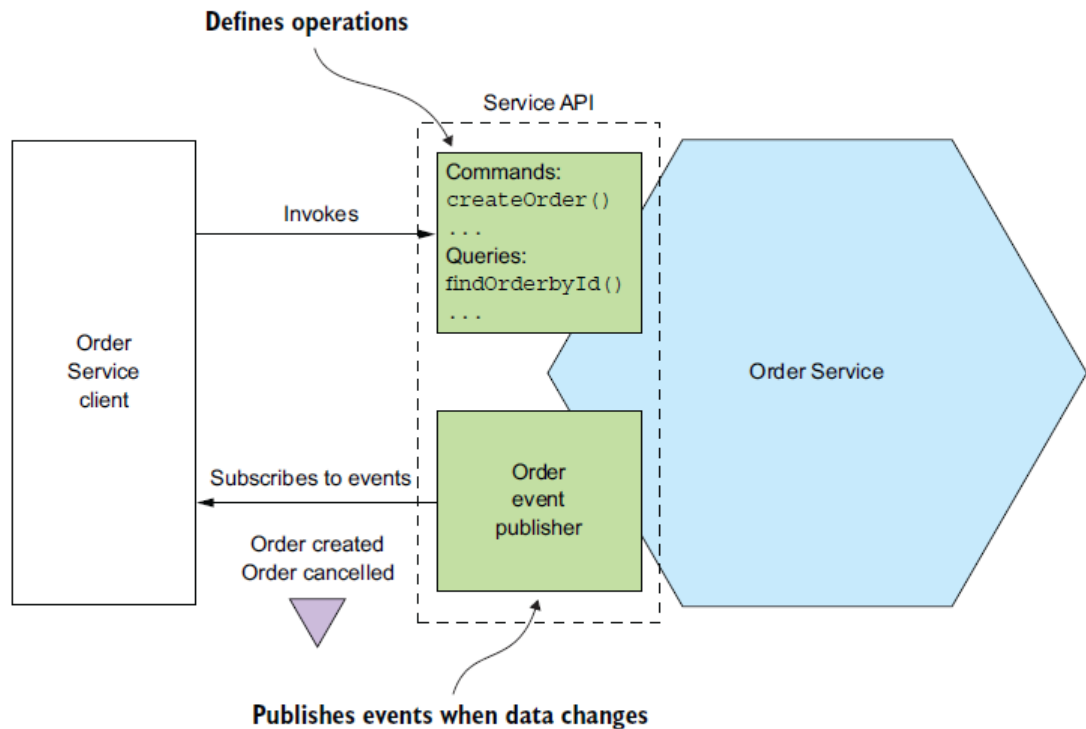
- Sự phức tạp đe dọa các lập trình viên: ứng dụng có quy mô lớn và phức tạp khiến không một lập trình viên nào có thể hoàn toàn hiểu được để tiếp tục làm việc hiệu quả.
- Việc phát triển ứng dụng diễn ra chậm chạp: ứng dụng lớn khiến môi trường tích hợp để phát triển ứng dụng (Integrated Development Environment – IDE) quá tải và hoạt động chậm. Từ đó, việc xây dựng và phát triển ứng dụng tốn rất nhiều thời gian.
- Con đường từ cài đặt đến triển khai là một quá trình gian nan: có rất nhiều lập trình viên tham gia viết mã nguồn vào cùng một nơi và việc kiểm thử do đó chiếm khá nhiều thời gian.
- Khó mở rộng ứng dụng: nhiều mô-đun khác nhau có những yêu cầu về tài nguyên khác nhau như bộ nhớ máy chủ có dung lượng lớn, máy chủ có nhiều CPU (Central Processing Unit),...
- Ứng dụng thiếu sự tin cậy: việc kiểm thử ứng dụng một cách kỹ lưỡng là rất khó vì quy mô của ứng dụng khá lớn. Điều này dẫn đến nhiều lỗi xảy ra trong quá trình sử dụng ứng dụng.
- Bị ràng buộc bởi những công nghệ lỗi thời: việc chuyển đổi sử dụng nền tảng (framework) mới hay ngôn ngữ mới rất mạo hiểm vì phải viết lại toàn bộ ứng dụng nguyên khối.

Để thoát khỏi vấn đề này, ứng dụng nên cân nhắc chuyển sang áp dụng kiến trúc microservice. Kiến trúc microservice có thể giải quyết hầu hết các vấn đề của mô hình phần mềm nguyên khối dựa vào định nghĩa kế tiếp của nó.

2.1.1.2 Microservices phân rã chức năng (functionally decompose) của ứng dụng thành một tập hợp những dịch vụ (service) có thể triển khai độc lập

2.1.1.2.1 Định nghĩa dịch vụ

Dịch vụ là một thành phần phần mềm có thể triển khai độc lập, một ứng dụng phiên bản nhỏ tập trung thực hiện một vài chức năng cụ thể. Ví dụ, hình bên dưới là góc nhìn bên ngoài của một dịch vụ, dịch vụ về đơn hàng (OrderService).



Hình 2.1.2 Một dịch vụ có một API đóng gói triển khai nội bộ của nó. API định nghĩa những hoạt động được thực hiện bởi khách hàng của nó [10]

Trong đó:

- Mỗi dịch vụ có một API (Application Programming Interface) cho phép truy cập vào những hàm chức năng của nó.
- Có hai loại hoạt động (operation): lệnh (command) và truy vấn (query). API bao gồm nhiều lệnh, truy vấn và sự kiện (event). Một lệnh, ví dụ như `createOrder()` dùng để biểu diễn một hành động và cập nhật dữ liệu. Một hoạt động truy vấn, ví dụ như `findOrderById()` dùng để lấy dữ liệu. Một dịch vụ đồng thời công bố sự kiện (publish event), ví dụ như `OrderCreated()` hay `OrderCancelled()`. Khi dữ liệu thay đổi, dịch vụ sẽ công bố sự kiện và

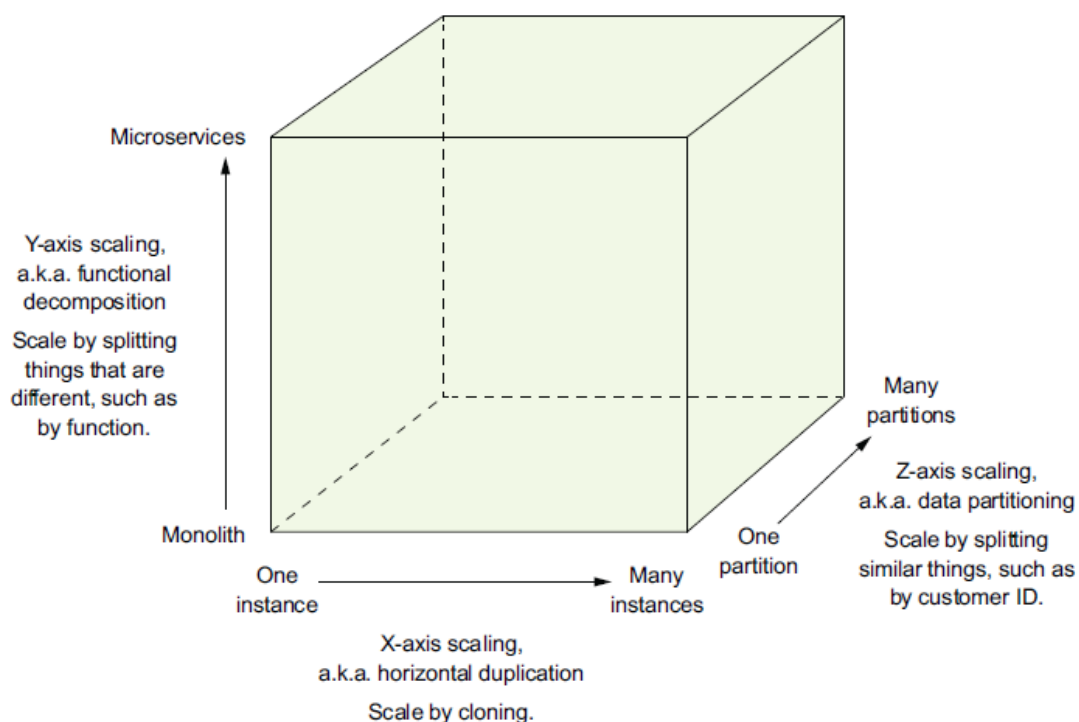
khách hàng (client) có thể theo dõi (subscribe) sự kiện này.

2.1.1.2.2 Dịch vụ trong kiến trúc microservice

Sự gắn kết (cohesion) hay nỗ lực để gom nhóm những dòng mã nguồn liên quan với nhau là một ý tưởng quan trọng khi nói về microservices. Điều này được củng cố bởi định nghĩa về nguyên lý đơn nhiệm (Single Responsibility Principle) của Robert C. Martin [11]: Tập hợp những thứ có thể thay đổi vì cùng một lý do, tách biệt những thứ có thể thay đổi vì những lý do khác nhau. Microservices có cách tiếp cận tương tự đối với các dịch vụ của nó. Một ứng dụng được xây dựng và phát triển theo kiến trúc microservice được phân rã thành nhiều dịch vụ. Dịch vụ là khía cạnh quan trọng nhất trong microservices. Mỗi dịch vụ có kiến trúc riêng và có thể sử dụng các nền tảng công nghệ riêng, thực hiện một tập chức năng chuyên biệt. Đặc biệt, các dịch vụ trong hệ thống ít phụ thuộc vào nhau (loosely coupled). Đây chính là một trong những tính chất quan trọng của kiến trúc này.

2.1.1.2.3 Khối lập phương về khả năng mở rộng (AKF Scale Cube)

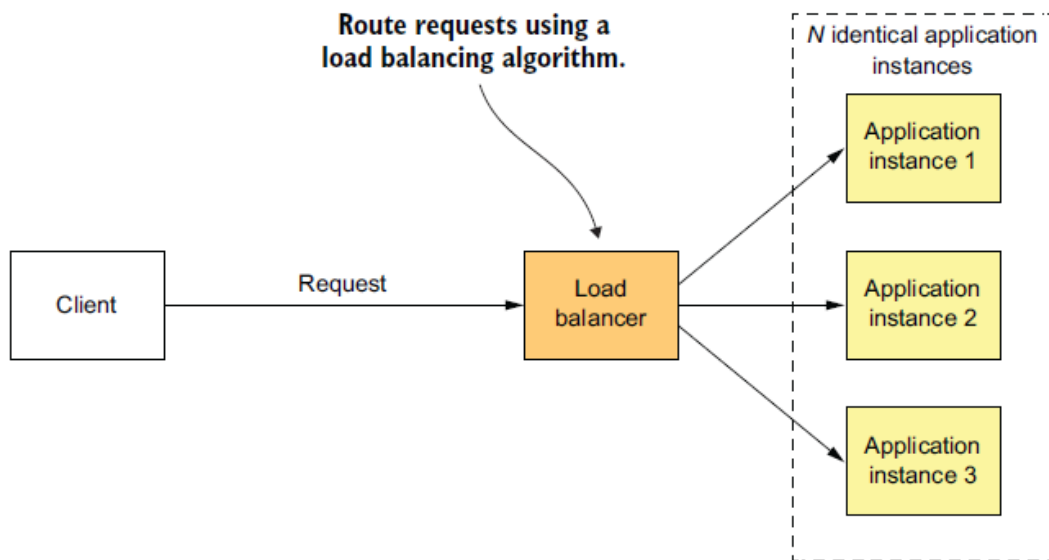
Kiến trúc microservice hoạt động tương đương trục Y trong khối lập phương về khả năng mở rộng. Khối lập phương này được giới thiệu trong quyển sách The Art of Scalability, viết bởi hai tác giả Martin L.Abbott và Michael T.Fisher [12]. Nó là một cách tiếp cận ba chiều để xây dựng ứng dụng có khả năng mở rộng vô hạn.



Hình 2.1.3 Khối lập phương về khả năng mở rộng định nghĩa ba cách khác nhau để mở rộng ứng dụng: trục X, Y, Z [10]

Trong đó:

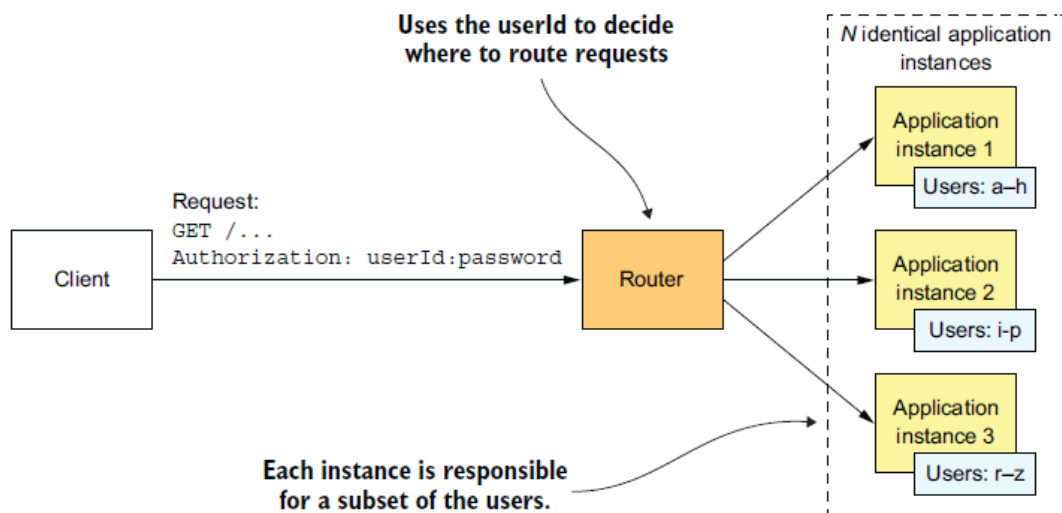
- Mở rộng theo trục X là cách thông thường để mở rộng một ứng dụng có kiến trúc nguyên khối. Bằng cách chạy N thực thể (instance) của một ứng dụng phía sau bộ cân bằng tải (load balancer), bộ cân bằng tải sẽ phân phối các yêu cầu cho N thực thể giống nhau, do đó mỗi thực thể chỉ xử lý 1/N yêu cầu. Đây là một cách tuyệt vời để cải thiện khả năng sẵn sàng phục vụ (availability) của ứng dụng.



Hình 2.1.4 Mở rộng theo trục X trong khối lập phương về khả năng mở rộng [10]

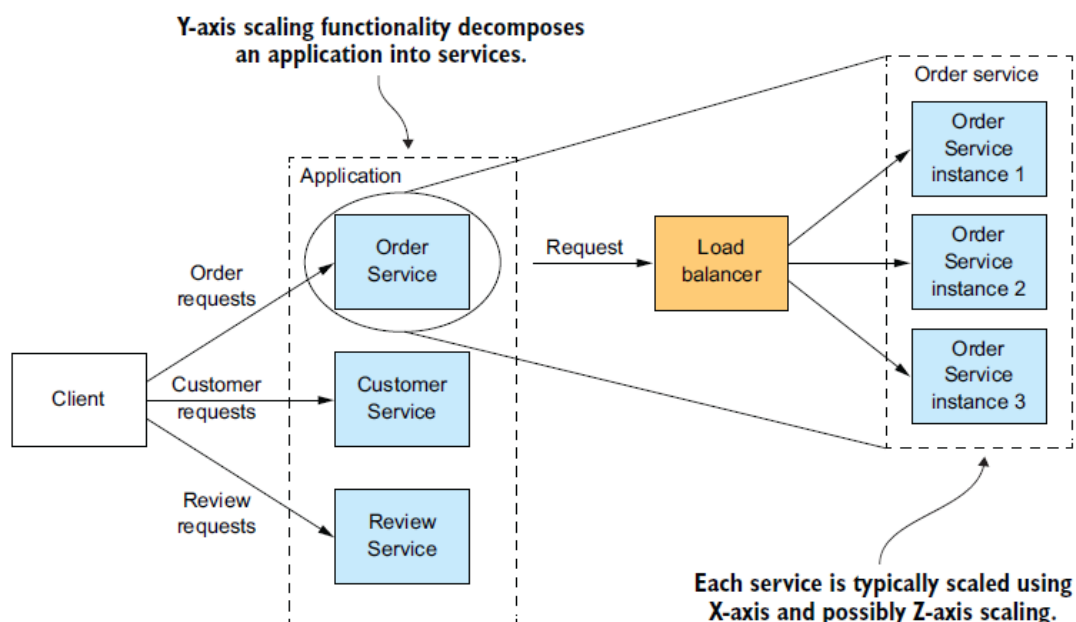
- Mở rộng theo trục Z là một cách cho phép ứng dụng xử lý khối lượng giao dịch và dữ liệu ngày càng tăng. Cũng bằng cách chạy nhiều thực thể của một ứng dụng, nhưng khác với cách mở rộng theo trục X, mỗi thực thể chịu trách nhiệm xử lý cho một phần dữ liệu. Bộ định tuyến (router) nằm phía trước các thực thể sử dụng một thuộc tính của yêu cầu để định tuyến nó đến thực thể thích hợp.

Ví dụ, một ứng dụng có thể định tuyến yêu cầu dựa trên userId như hình bên dưới. Bộ định tuyến sử dụng userId lấy từ Authorization trong phần đầu của HTTP (Hypertext Transfer Protocol) trong mỗi yêu cầu để chọn lựa một trong N thực thể ứng dụng.



Hình 2.1.5 Mở rộng theo trục Z trong khối lập phương về khả năng mở rộng. Mỗi thực thể chịu trách nhiệm xử lý một tập con dữ liệu người dùng [10]

- Mở rộng theo trục Y chính là sự phân rã ứng dụng theo chức năng thành nhiều dịch vụ. Cả hai cách mở rộng theo trục X và Z đều không giải quyết được sự phát triển phức tạp ngày càng tăng của ứng dụng. Mở rộng theo trục Y là cách giải quyết cho vấn đề này. Nó được thể hiện thông qua ví dụ hình 2.1.6. Ứng dụng được phân rã thành các dịch vụ như OrderService, CustomerService, ReviewService. Mỗi dịch vụ độc lập này sẽ được mở rộng theo trục X và có thể cả trục Z. Dựa vào hình, có thể thấy OrderService được mở rộng theo trục X.



Hình 2.1.6 Mở rộng theo trục Y trong khối lập phương về khả năng mở rộng. Mỗi dịch vụ chịu trách nhiệm cho một chức năng đặc biệt [10]

Tóm lại, bản chất của kiến trúc microservice là sự phân rã chức năng của ứng dụng thành nhiều dịch vụ nhỏ.

2.1.2 Tính chất

2.1.2.1 Microservices là kiến trúc lý tưởng cho hệ thống với quy mô lớn

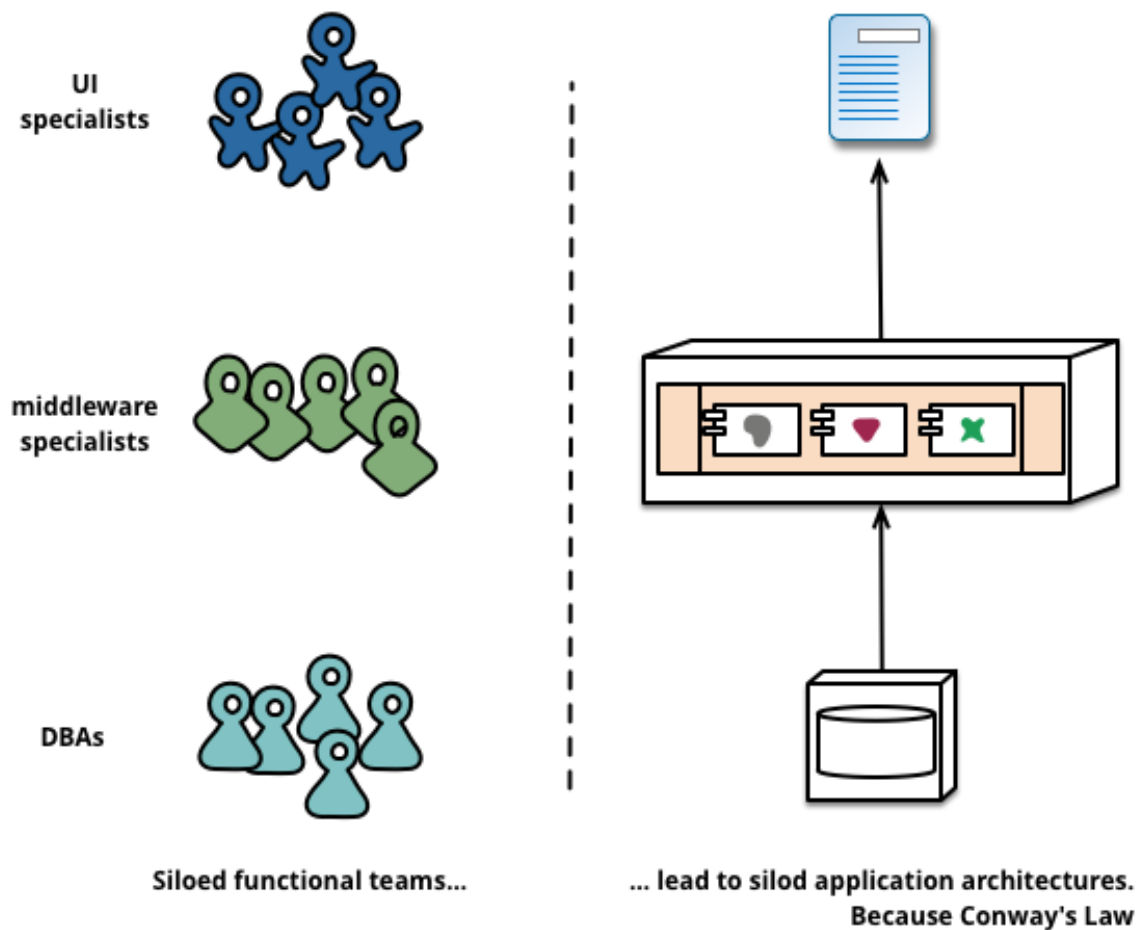
Điều này được xác định dựa trên nguồn gốc ra đời của kiến trúc microservice. Tại phần mở đầu định nghĩa microservices, nhóm chúng em đã đề cập đến buổi họp mặt giữa các kỹ sư phần mềm, trong đó có James Lewis, ông kể lại: những vấn đề mà mọi người đang gặp phải đều liên quan đến việc xây dựng những hệ thống có quy mô lớn. Microservices được thiết kế để giải quyết những vấn đề cho hệ thống có quy mô lớn [6]. Điều đáng quan tâm không phải là định nghĩa về quy mô lớn của hệ thống. Thay vào đó, nó là tình huống mà hệ thống phát triển vượt khỏi ranh giới quy mô ban đầu được xác định, dẫn đến một số vấn đề đặc biệt khi muốn tác động thay đổi hệ thống. Nói cách khác, những vấn đề mới phát sinh do quy mô của hệ thống thay đổi.

2.1.2.2 Kiến trúc microservice như một hình thức mô-đun hoá (modularity)

Khi phát triển những ứng dụng lớn và phức tạp, mô-đun hoá là một việc thiết yếu. Vì sự phức tạp, ứng dụng phải được chia thành các mô-đun, mỗi mô-đun được phát triển và quản lý bởi một nhóm khác nhau. Kiến trúc microservice sử dụng các dịch vụ của nó như một mô-đun. Mỗi dịch vụ có một API – một ranh giới khó để xâm phạm. Nó không cho phép truy cập vào những lớp nội bộ, dẫn đến việc dễ dàng bảo toàn tính mô-đun hoá của ứng dụng.

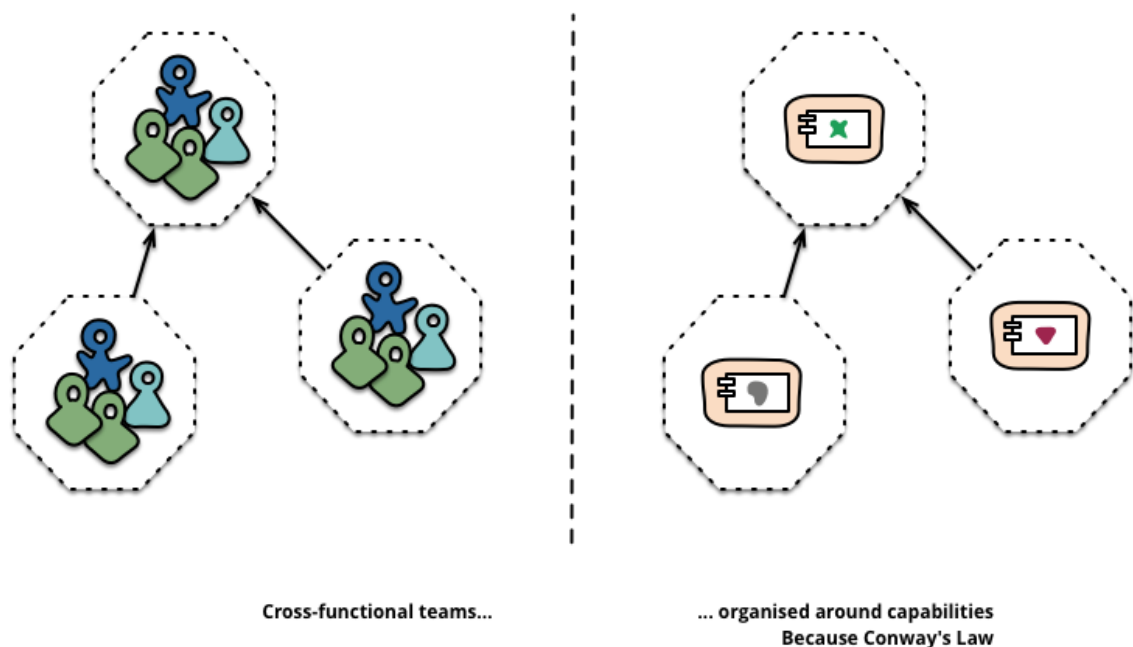
2.1.2.3 Các dịch vụ trong microservices được tổ chức dựa trên khả năng về nghiệp vụ

Khi muốn tổ chức một ứng dụng lớn thành nhiều phần nhỏ hơn, thông thường ứng dụng được chia dựa trên tính năng kỹ thuật, hình thành những nhóm làm về giao diện người dùng, những nhóm xử lý logic phía máy chủ, những nhóm quản lý cơ sở dữ liệu. Khi chia nhóm theo cách này, một vài yêu cầu thay đổi đơn giản sẽ khiến dự án mất khá nhiều thời gian để đáp ứng. Đây là một ví dụ cho luật của Conway: Bất cứ một tổ chức nào thiết kế một hệ thống sẽ tạo ra một thiết kế có cấu trúc rất giống với cấu trúc của hệ thống giao tiếp của tổ chức đó [13].



Hình 2.1.7 Luật của Conway khi áp dụng vào cách tổ chức ứng dụng dựa trên tính năng kỹ thuật [14]

Cách tổ chức ứng dụng theo kiến trúc microservice lại khác. Nó chia ứng dụng thành nhiều dịch vụ dựa trên khả năng về nghiệp vụ. Mỗi dịch vụ này triển khai phần mềm theo nhiều mặt phục vụ cho nghiệp vụ của nó, bao gồm giao diện người dùng, dữ liệu lưu trữ liên tục và các kết nối với dịch vụ bên ngoài nếu có. Vì vậy, các nhóm trở nên đa chức năng, trang bị đầy đủ những kỹ năng cần thiết để phát triển dịch vụ như trải nghiệm người dùng, cơ sở dữ liệu và quản lý dự án.



Hình 2.1.8 Ranh giới giữa các nhóm phát triển củng cố ranh giới giữa các dịch vụ
[14]

Khi tổ chức dịch vụ trong microservices dựa trên khả năng nghiệp vụ, vì khả năng nghiệp vụ của một tổ chức thường ổn định nên kiến trúc của nó cũng sẽ tương đối ổn định. Trong khi từng thành phần trong kiến trúc có thể phát triển khi nghiệp vụ của nó thay đổi cách thực hiện, bản thân kiến trúc vẫn được giữ nguyên.

2.1.2.4 Các dịch vụ trong microservices có thể triển khai độc lập, ít phụ thuộc vào nhau

Một tính chất quan trọng của kiến trúc microservice đó là các dịch vụ hoạt động độc lập, ít phụ thuộc nhau. Tất cả tương tác đến một dịch vụ đều phải thông qua API của nó. Điều này cho phép dịch vụ che giấu chi tiết cài đặt bên trong, cho phép nó thay đổi cách cài đặt mà không ảnh hưởng đến khách hàng của nó. Tính chất này là chìa khoá để cải thiện khả năng bảo trì và kiểm thử ứng dụng trong thời gian phát triển ứng dụng.

Để các dịch vụ ít ràng buộc nhau và chỉ giao tiếp với nhau qua API, mỗi dịch vụ cần có cơ sở dữ liệu riêng. Trong thời gian phát triển dịch vụ, lập trình viên có thể

thay đổi lược đồ cơ sở dữ liệu (database schema) của dịch vụ mà không cần phải làm việc với những lập trình viên phát triển các dịch vụ khác. Trong thời gian thực thi, do các dịch vụ cô lập với nhau, một dịch vụ không thể bị khoá chỉ vì một dịch vụ khác đang giữ khoá cơ sở dữ liệu (database lock).

2.1.2.5 Giao tiếp liên tiến trình là một phần quan trọng

Một ứng dụng xây dựng dựa trên kiến trúc microservice là một hệ thống phân tán (distributed system). Do đó, việc giao tiếp liên tiến trình là một phần quan trọng trong microservices. Giao tiếp liên tiến trình là một phương pháp cho phép các tiến trình riêng biệt trao đổi, giao tiếp với nhau. Nó cho phép các dịch vụ có thể tương tác với nhau và với thế giới bên ngoài.

2.2 PHƯƠNG PHÁP THIẾT KẾ TỪ NGHIỆP VỤ

Trong phần này, nhóm chúng em sẽ trình bày các kiến thức nền tảng về cách tiếp cận đối với các mô hình phức tạp bằng phương pháp thiết kế Domain Driven Design (DDD) qua các ví dụ thực tiễn, từ đó thu được các kiến thức nền tảng về cách xây dựng và phát triển đối với các dự án lớn có yêu cầu nghiệp vụ phức tạp.

2.2.1 Định nghĩa

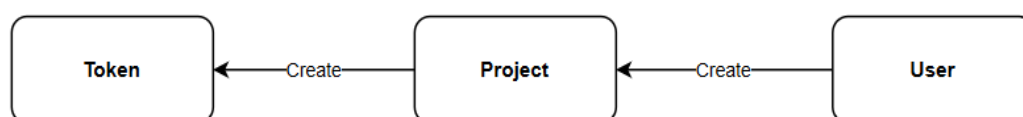
Domain Driven Design là một cách tiếp cận để phát triển những phần mềm có các yêu cầu phức tạp thông qua sự kết nối chặt chẽ giữa việc triển khai ứng dụng với một mô hình tiến hoá các khái niệm cốt lõi trong nghiệp vụ [15].

Mỗi chương trình phần mềm đều liên quan đến một số hoạt động hay mối quan tâm của người dùng. Lĩnh vực hay hoạt động mà người dùng áp dụng chương trình vào chứa “nghiệp vụ” (domain) của phần mềm.

Một số domain liên quan đến thế giới vật chất như domain của chương trình đặt vé máy bay liên quan đến người thật lên chiếc máy bay thật. Một số domain liên quan đến những thứ vô hình như domain của hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt liên quan đến việc mua bán, dùng “mã truy cập” sử dụng dịch vụ. Để tạo ra một phần mềm có giá trị, lập trình viên cần phải hiểu về phần mềm đó, cụ thể là cần phải trang bị kiến thức liên quan đến các hoạt động mà phần mềm được áp dụng vào. Tuy nhiên, lượng kiến thức cần thiết và sự phức tạp của nó có thể là vô cùng lớn. Đây là lúc đội ngũ phát triển có thể sử dụng phương pháp mô hình hoá dữ liệu để đối phó với sự quá tải thông tin này.

Một mô hình là một dạng kiến thức được đơn giản hoá và có cấu trúc. Một mô hình nghiệp vụ (domain model) không phải là một sơ đồ cụ thể nào, ý tưởng mà sơ đồ muốn truyền đạt mới là điều đáng quan tâm. Nó không chỉ là kiến thức trong đầu của một chuyên gia lĩnh vực (domain expert), nó còn là sự trừu tượng có tổ chức chặt chẽ và chọn lọc của những kiến thức đó. Chúng ta sẽ xem xét ví dụ đơn giản về một mô hình nghiệp vụ trong hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng

Việt. Đối với domain liên quan đến việc tạo “mã truy cập” của người dùng, đầu tiên người dùng tạo một “dự án” để chứa các “mã truy cập”. Sau đó, người dùng chọn mua một loại “mã truy cập” cho dự án vừa tạo. Hệ thống sẽ tạo một “mã truy cập” theo yêu cầu. Người dùng có thể dùng “mã truy cập” này để truy cập dịch vụ hoặc chia sẻ quyền sử dụng “mã truy cập” bằng cách mời những người dùng khác tham gia dự án của mình.



Hình 2.2.1 Trừu tượng hóa nghiệp vụ tạo “mã truy cập” trong hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt thành các đối tượng trong mô hình nghiệp vụ, bao gồm mã truy cập, dự án, người dùng

2.2.2 Đặc điểm và các thành phần

2.2.2.1 Ngôn ngữ chung (Ubiquitous Language)

Trong phần trước chúng ta có thể thấy sự cần thiết của việc phát triển mô hình cho domain qua quá trình làm việc giữa khách hàng – những người nắm vững kiến thức nghiệp vụ và đội ngũ phát triển phần mềm. Tuy nhiên, cách làm này ban đầu thường gặp khó khăn về rào cản giao tiếp cơ bản. Lập trình viên chưa có nhiều hiểu biết về nghiệp vụ, trong khi khách hàng có kiến thức hạn chế về lĩnh vực công nghệ. Để vượt qua trở ngại này, chúng ta xây dựng mô hình, chúng ta phải trao đổi, giao tiếp ý tưởng về mô hình, những thành phần liên quan đến mô hình, cách chúng ta liên kết chúng, chúng có liên quan với nhau hay không. Giao tiếp ở mức độ này là tối quan trọng cho sự thành công của dự án. Nếu ai đó nói điều gì đó và người khác không hiểu, hoặc tệ hơn, hiểu sai, thì liệu chúng ta có cơ hội tiếp tục dự án không?

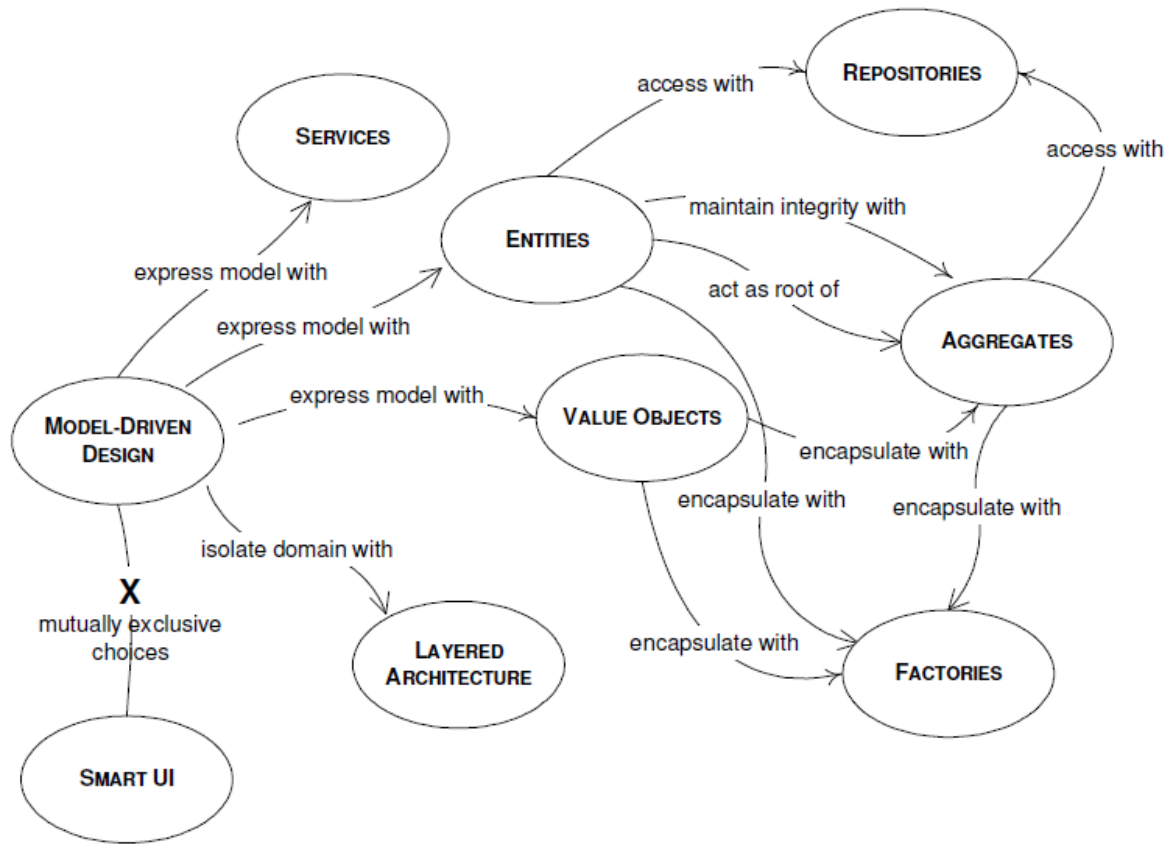
2.2.2.2 Thiết kế hướng mô hình

Các phần trước đã nhấn mạnh tầm quan trọng của cách tiếp cận tới quy trình phát triển phần mềm tập trung vào lĩnh vực nghiệp vụ. Chúng ta đã biết nó có ý nghĩa then chốt để tạo ra một đối tượng có gốc rễ là domain.

Ngôn ngữ chung nên được thực hiện đầy đủ trong suốt quá trình mô hình hóa nhằm thúc đẩy giao tiếp giữa các chuyên gia phần mềm với khách hàng, và khám phá ra các khái niệm chính của domain nên được sử dụng trong đối tượng. Mục đích của quá trình mô hình hóa là nhằm tạo ra một đối tượng tốt. Bước tiếp theo là hiện thực nó trong mã nguồn. Đây là một giai đoạn quan trọng không kém của quá trình phát triển phần mềm.

2.2.2.3 Các khối ghép của mô hình

Phần sau đây của chương này sẽ giới thiệu các khuôn mẫu quan trọng nhất được sử dụng trong DDD. Mục đích của những khuôn mẫu này là để trình bày một số yếu tố chính của mô hình hóa hướng đối tượng và thiết kế phần mềm từ quan điểm của DDD. Sơ đồ sau đây là một biểu đồ của các khuôn mẫu sẽ được trình bày và các mối quan hệ giữa chúng.



Hình 2.2.2 Biểu đồ và quan hệ của các khuôn mẫu trong mô hình DDD [16]

Khi chúng ta tạo ra một ứng dụng phần mềm, một lượng lớn thành phần của ứng dụng không liên quan trực tiếp đến nghiệp vụ, nhưng chúng là một phần của hạ tầng phần mềm hoặc phục vụ chính bản thân ứng dụng. Nó giúp ổn định phần nghiệp vụ của ứng dụng nhỏ so với các phần còn lại, vì một ứng dụng điển hình chứa rất nhiều đoạn mã liên quan đến truy cập cơ sở dữ liệu, tệp hoặc mạng, giao diện người dùng,...

Trong một ứng dụng hướng đối tượng thuần túy, giao diện người dùng, mã truy cập cơ sở dữ liệu, và các đoạn mã hỗ trợ khác thường được viết trực tiếp vào trong các đối tượng nghiệp vụ. Thêm vào đó, đối tượng nghiệp vụ này lại được nhúng vào trong các hành vi của giao diện người dùng và các kịch bản cơ sở dữ liệu. Đôi khi điều này diễn ra bởi vì nó là cách dễ nhất để làm mọi việc trở nên nhanh chóng.

Tuy nhiên, khi các đoạn mã nguồn liên quan đến nghiệp vụ được trộn lẫn giữa

các tầng lại với nhau, nó trở nên vô cùng khó khăn cho việc đọc cũng như suy nghĩ về chúng. Các thay đổi ở giao diện người dùng cũng có thể thực sự thay đổi cả logic nghiệp vụ. Để thay đổi logic nghiệp vụ có thể yêu cầu tới truy vết tỉ mỉ các đoạn mã của giao diện người dùng, cơ sở dữ liệu, hoặc các thành phần khác của chương trình. Mô hình phát triển hướng đối tượng trở nên phi thực tế. Kiểm thử tự động sẽ khó khăn. Với tất cả các công nghệ và mã nguồn liên quan trong từng hoạt động, chương trình cần được giữ rất đơn giản hoặc không thì sẽ biến chúng trở nên không thể hiểu được.

Do đó, hãy phân chia một chương trình phức tạp thành các lớp. Phát triển một thiết kế cho mỗi lớp để chúng trở nên gắn kết và chỉ phụ thuộc vào các tầng bên dưới. Tuân theo khuôn mẫu kiến trúc chuẩn để nối lỏng các liên kết tới các tầng phía trên. Tập trung các đoạn mã liên quan đến các đối tượng nghiệp vụ trong một lớp và cô lập chúng khỏi lớp giao diện người dùng, ứng dụng, và hạ tầng.

Các đối tượng nghiệp vụ, được giải phóng khỏi việc hiển thị, lưu trữ chính chúng, hay quản lý các nhiệm vụ của ứng dụng, vận vận, và có thể tập trung vào việc biểu hiện của domain. Điều này cho phép một đối tượng tiến hóa đủ phong phú và rõ ràng, nhằm nắm bắt kiến thức nghiệp vụ thiết yếu và áp dụng nó vào làm việc.

Một giải pháp kiến trúc chung cho DDD chứa bốn lớp (trên lý thuyết):

- Giao diện người dùng: chịu trách nhiệm trình bày thông tin tới người dùng và thông tin lệnh của người dùng.
- Tầng ứng dụng: phối hợp hoạt động của ứng dụng, không chứa logic nghiệp vụ, không lưu giữ trạng thái của các đối tượng nghiệp vụ nhưng nó có thể giữ trạng thái của một tiến trình của ứng dụng.
- Lớp domain: Tầng này chứa thông tin về các lĩnh vực. Đây là trọng tâm của nghiệp vụ phần mềm. Trạng thái của đối tượng nghiệp vụ được giữ tại đây, quản lý vòng đời đối tượng nghiệp vụ và trạng thái của chúng được ủy quyền cho lớp hạ tầng.

- Lớp hạ tầng: lớp này đóng vai trò như một thư viện hỗ trợ cho tất cả các lớp còn lại. Nó cung cấp thông tin liên lạc giữa các lớp, cài đặt cho đối tượng nghiệp vụ, đồng thời chứa các thư viện hỗ trợ cho tầng giao diện người dùng,...

2.2.2.4 Mô-đun

Với hệ thống lớn và phức tạp, mô hình thường có xu hướng phình càng ngày càng to. Khi mô hình phình tới một điểm ta khó nắm bắt được tổng thể, việc hiểu quan hệ và tương tác giữa các phần khác nhau trở nên khó khăn. Vì lý do đó, việc tổ chức mô hình thành các mô-đun là cần thiết. Mô-đun được dùng như một phương pháp để tổ chức các khái niệm và tác vụ liên quan nhằm giảm độ phức tạp.

Mô-đun được dùng rộng rãi trong hầu hết các dự án. Sẽ dễ dàng hơn để hình dung mô hình lớn nếu ta nhìn vào những mô-đun chứa trong mô hình đó, sau đó là quan hệ giữa các mô-đun. Khi đã hiểu tương tác giữa các mô-đun, ta có thể bắt đầu tìm hiểu phân chi tiết trong từng mô-đun. Đây là cách đơn giản và hiệu quả để quản lý sự phức tạp.

Với hệ thống lớn và phức tạp, mô hình thường có xu hướng phình càng ngày càng to. Khi mô hình phình tới một điểm ta khó nắm bắt được tổng thể, việc hiểu quan hệ và tương tác giữa các phần khác nhau trở nên khó khăn. Vì lý do đó, việc tổ chức mô hình thành các mô-đun là cần thiết. Mô-đun được dùng như một phương pháp để tổ chức các khái niệm và tác vụ liên quan nhằm giảm độ phức tạp. Mô-đun được dùng rộng rãi trong hầu hết các dự án. Sẽ dễ dàng hơn để hình dung mô hình lớn nếu ta nhìn vào những mô-đun chứa trong mô hình đó, sau đó là quan hệ giữa các mô-đun. Khi đã hiểu tương tác giữa các mô-đun, ta có thể bắt đầu tìm hiểu phân chi tiết trong từng mô-đun. Đây là cách đơn giản và hiệu quả để quản lý sự phức tạp.

2.2.2.5 Aggregate

Việc quản lý vòng đời các đối tượng trong domain không hề đơn giản, nếu như làm không đúng sẽ có thể gây ảnh hưởng đến việc mô hình hóa domain. Sau đây

chúng ta sẽ đề cập đến mẫu thiết kế thường dùng để hỗ trợ giải quyết vấn đề này. Aggregate là mẫu thiết kế để định nghĩa việc sở hữu đối tượng và phân cách giữa chúng.

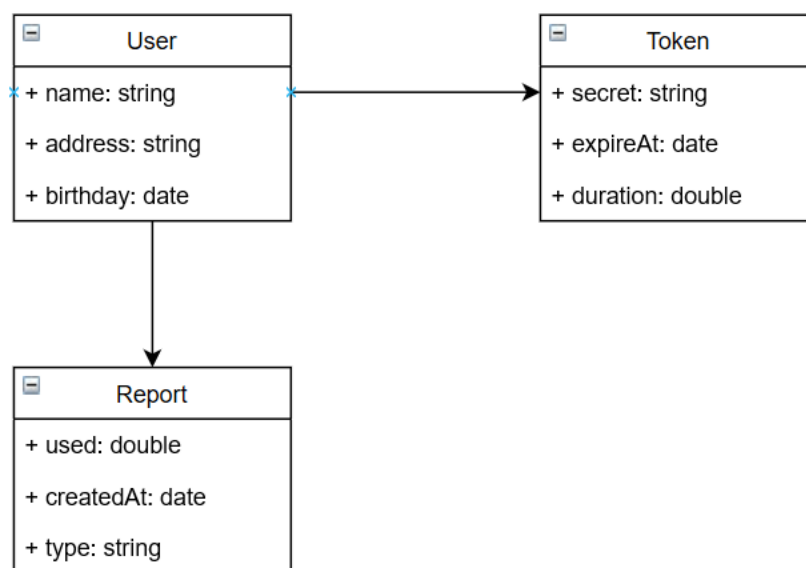
Một mô hình có thể được tạo thành từ nhiều đối tượng trong domain. Cho dù có cẩn thận trong việc thiết kế như thế nào thì chúng ta cũng không thể tránh được việc sẽ có nhiều mối quan hệ chằng chịt giữa các đối tượng, tạo thành một lưới các quan hệ. Có thể có nhiều kiểu quan hệ khác nhau, với mỗi kiểu quan hệ giữa các mô hình cần có một cơ chế phần mềm để thực thi nó.

Các mối quan hệ sẽ được thể hiện trong mã nguồn phần mềm, và trong nhiều trường hợp trong cả cơ sở dữ liệu nữa.

Một Aggregate là một nhóm các đối tượng, nhóm này có thể được xem như là một đơn vị thống nhất đối với các thay đổi dữ liệu. Một Aggregate được phân tách với phần còn lại của hệ thống, ngăn cách giữa các đối tượng nội tại và các đối tượng ở ngoài. Mỗi Aggregate có một "gốc" (root), đó là một thực thể và cũng là đối tượng duy nhất có thể truy cập từ phía ngoài của Aggregate. Gốc của Aggregate có thể chứa những tham chiếu đến các đối tượng khác trong Aggregate, và những đối tượng trong này có thể chứa tham chiếu đến nhau, nhưng các đối tượng ở ngoài chỉ có thể tham chiếu đến gốc. Nếu như trong Aggregate có những thực thể khác thì định danh của chúng là nội tại, chỉ mang ý nghĩa trong aggregate.

Vậy bằng cách nào mà Aggregate có thể đảm bảo được tính toàn vẹn và các ràng buộc của dữ liệu? Vì các đối tượng khác chỉ có thể tham chiếu đến gốc của Aggregate, chúng không thể thay đổi trực tiếp đến các đối tượng nằm bên trong mà chỉ có thể thay đổi thông qua gốc, hoặc là thay đổi gốc Aggregate trực tiếp. Tất cả các thay đổi của Aggregate sẽ thực hiện thông qua gốc của chúng, và chúng ta có thể quản lý được những thay đổi này, so với khi thiết kế cho phép các đối tượng bên ngoài truy cập trực tiếp vào các đối tượng bên trong thì việc đảm bảo các invariant sẽ đơn giản hơn nhiều khi chúng ta phải thêm các logic vào các đối tượng ở ngoài để thực hiện. Nếu như gốc của Aggregate bị xóa và loại bỏ khỏi bộ nhớ thì những

đối tượng khác trong Aggregate cũng sẽ bị xóa, vì không còn đối tượng nào chứa tham chiếu đến chúng.



Hình 2.2.3 Mối quan hệ trong Aggregate có gốc là đối tượng người dùng – *User* cùng các quan hệ với các báo cáo – *Report*, mã truy cập – *Token*

2.2.3 Duy trì tính toàn vẹn của mô hình

Chương này đề cập tới những dự án lớn, đòi hỏi sự tham gia của nhiều nhóm. Chúng ta gặp nhiều loại thách thức khác nhau khi có nhiều nhóm, được quản lý và phối hợp theo cách khác nhau. Những dự án cho doanh nghiệp thường là những dự án lớn, đòi hỏi nhiều loại công nghệ và nguồn lực. Thiết kế của những dự án ấy vẫn thường dựa trên mô hình domain và chúng ta cần đo đạc thích hợp để đảm bảo sự thành công của dự án.

Khi nhiều nhóm làm cùng một dự án, họ viết mã nguồn song song, mỗi nhóm được chỉ định làm một phần của mô hình. Những phần này không độc lập, nhưng ít liên hệ chéo.

Họ đều bắt đầu từ một mô hình lớn và chia nhỏ ra để thực thi. Giả sử rằng một nhóm nào đó đã tạo ra một mô-đun và họ mở cho các nhóm khác dùng. Một lập

trình viên từ nhóm khác dùng mô-đun này và phát hiện ra rằng mô-đun nó thiếu một số chức năng anh ta cần. Anh ta thêm những chức năng đó và chia sẻ mã nguồn để mọi người đều có thể dùng. Điều anh ta không ngờ đến là việc anh ta làm thay đổi cả mô hình, và có thể rằng thay đổi này sẽ làm hỏng tính năng của ứng dụng. Điều này rất dễ xảy ra, không ai có thời gian để hiểu cả mô hình. Mọi người đều có sân chơi riêng của mình và những "sân riêng" này không đủ rõ với người khác.

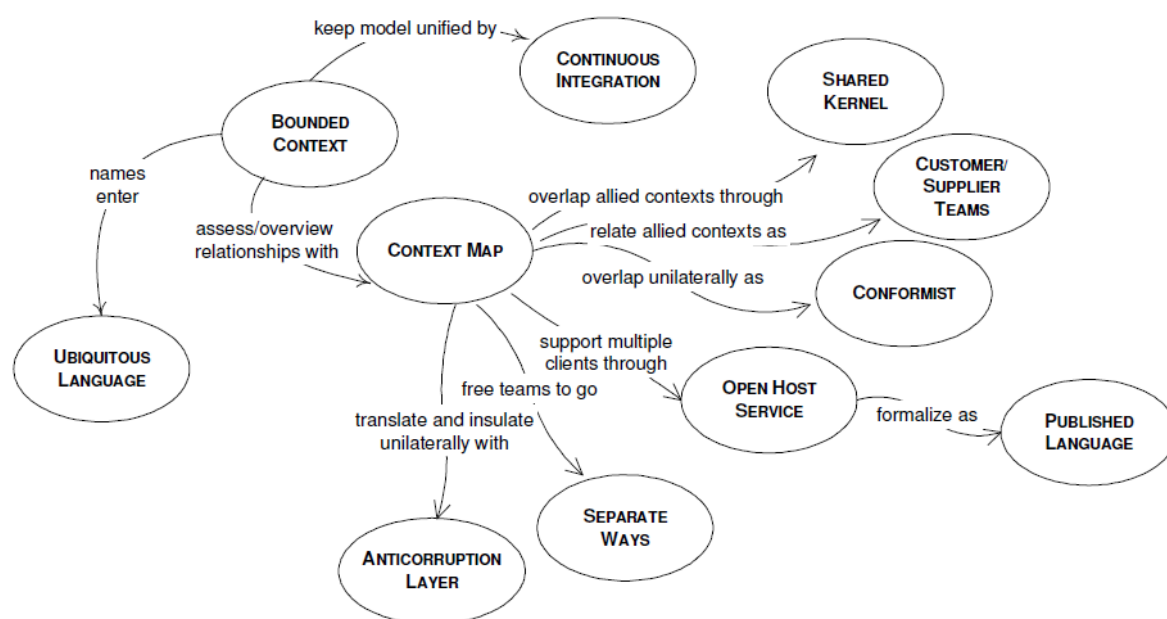
Sẽ dễ dàng hơn nếu ta bắt đầu từ một mô hình tốt và sau đó biến thành các mô hình không nhất quán. Yêu cầu đầu tiên của mô hình là phải nhất quán, với những điều khoản không bất biến và không có mâu thuẫn. Sự nhất quán nội tại của một mô hình được gọi là sự thống nhất. Một dự án doanh nghiệp có thể có một mô hình phủ mọi domain của doanh nghiệp, không có mâu thuẫn và không có điều khoản chồng chéo. Một mô hình doanh nghiệp thống nhất lý tưởng không dễ đạt được và đôi khi không đáng giá để làm thử. Những dự án như thế cần nỗ lực tổng hợp của nhiều nhóm.

Các nhóm này cần độ độc lập cao trong quy trình phát triển vì họ không có thời gian gặp nhau thường xuyên để trao đổi về thiết kế. Việc điều phối những nhóm như thế là một nhiệm vụ khó khăn. Họ có thể thuộc về nhiều phòng ban khác nhau với cách quản lý khác nhau. Khi thiết kế của mô hình tiến hóa từng phần một cách độc lập, chúng ta đối mặt với rủi ro mất tính nhất quán của mô hình. Duy trì tính nhất quán của mô hình bằng việc cố gắng giữ một mô hình thống nhất lớn cho toàn dự án doanh nghiệp sẽ không thể hoạt động được. Giải pháp không dễ vì nó đi ngược với tất cả những gì chúng ta đã tìm hiểu tới nay.

Thay vì giữ một mô hình lớn nhưng sẽ bị phân nhỏ về sau này, chúng ta nên chia một cách có ý thức mô hình thành nhiều mô hình khác. Những mô hình này sẽ tích hợp và tiến hóa độc lập nếu chúng tuân thủ giao ước của chúng. Mỗi mô hình cần có một ranh giới rõ ràng, và quan hệ giữa các mô hình phải được định nghĩa chính xác.

Chúng ta sẽ trình bày một số kỹ thuật dùng để duy trì tính nhất quán của mô

hình. Hình 2.2.4 sau đây mô tả những kỹ thuật và mối quan hệ giữa chúng.



Hình 2.2.4 Một số kỹ thuật duy trì tính nhất quán của mô hình và mối quan hệ giữa chúng [16]

2.2.3.1 Ngữ cảnh giới hạn (bounded context)

Mỗi mô hình đều có một ngữ cảnh tương ứng với nó. Chúng ta làm việc với một mô hình, ngữ cảnh là ẩn. Chúng ta không cần định nghĩa chúng, khi chúng ta tạo một chương trình tương tác với phần mềm khác, ví dụ một ứng dụng cũ, hiển nhiên là ứng dụng mới có mô hình và ngữ cảnh riêng của nó và chúng được chia riêng với mô hình và ngữ cảnh cũ.

Chúng không thể gộp, trộn lẫn và không bị hiểu lầm. Tuy nhiên, khi chúng ta làm việc với một ứng dụng doanh nghiệp lớn, chúng ta cần định nghĩa ngữ cảnh cho từng mô hình ta tạo ra.

Dự án lớn nào cũng có nhiều mô hình. Tuy vậy, mã nguồn dựa trên các mô hình riêng biệt lại được gộp lại và phần mềm trở nên nhiều lỗi, kém tin tưởng và khó hiểu, giao tiếp giữa các thành viên trong nhóm trở nên rối rắm. Thường thì việc quyết định mô hình nào áp dụng cho ngữ cảnh nào là không rõ ràng.

Không có một công thức nào để chia mọi mô hình to thành nhỏ thành nhỏ hơn.

Hãy thử đặt những thành phần có liên quan vào mô hình theo những khái niệm một cách tự nhiên. Một mô hình cần đủ nhỏ để nó phù hợp với một nhóm, sự phối hợp và trao đổi nhóm sẽ thông suốt và hoàn chỉnh, giúp cho lập trình viên làm việc trên cùng một mô hình. Ngữ cảnh của mô hình là tập các điều kiện cần được áp dụng để đảm bảo những điều khoản của mô hình có ý nghĩa cụ thể.

Ý tưởng chính cho việc định nghĩa một mô hình là vẽ ra ranh giới giữa các ngữ cảnh, sau đó cố gắng giữ các mô hình có được sự thống nhất càng cao càng tốt. Việc giữ một mô hình “thuần khiết” khi nó mở rộng ra cả dự án doanh nghiệp là rất khó, nhưng dễ dàng hơn khi chúng ta hạn chế trong một mảng cụ thể. Định nghĩa một cách hiển minh ngữ cảnh trong ngữ cảnh giới hạn. Hãy xác định hiển minh ranh giới giữa các điều khoản của tổ chức nhóm, dùng nó trong một phần cụ thể của mô hình, thể hiện nó một cách vật lý bằng, chẳng hạn như schema của mã nguồn. Duy trì mô hình này tương thích chặt chẽ với các giới hạn nó. Tuy vậy, đừng để mất tập trung hay rối loạn vì những vấn đề bên ngoài.

Một ngữ cảnh giới hạn không phải là một mô-đun. Một ngữ cảnh giới hạn cung cấp khung logic bên trong của mô hình. Mô-đun tổ chức các thành phần của mô hình, do đó ngữ cảnh giới hạn chứa mô-đun. Khi nhiều nhóm làm việc trên cùng một mô hình, chúng ta phải cẩn thận sao cho chúng không dẫm chân lên nhau. Chúng ta phải luôn ý thức tới sự thay đổi có thể ảnh hưởng đến chức năng. Khi dùng nhiều mô hình, mọi người phải làm việc tự do trên mảng của họ. Chúng ta đều biết hạn chế trong mô hình của chúng ta và không bước ra ngoài giới hạn. Chúng ta phải đảm bảo giữ được mô hình tinh khiết, nhất quán và thống nhất. Mỗi mô hình cần hỗ trợ việc tái cấu trúc dễ hơn, không gây ảnh hưởng tới mô hình khác. Thiết kế có thể được làm mịn và chất lọc để đạt được sự tinh khiết tối đa.

Việc có nhiều mô hình có cái giá phải trả. Chúng ta cần định nghĩa ranh giới và quan hệ giữa các mô hình khác nhau. Điều này đòi hỏi nhiều công sức và có thể cần sự "phiên dịch" giữa các mô hình khác nhau. Chúng ta không thể chuyển đổi tương đương bất kỳ giữa các mô hình khác nhau và chúng ta không thể gọi tự do giữa các mô

hình như là không có ranh giới nào. Đây không phải là một công việc quá khó và lợi ích nó đem lại đáng được đầu tư.

Ví dụ chúng ta muốn tạo ra một phần mềm thương mại điện tử để bán phần mềm trên Internet. Phần mềm này cho phép khách hàng đăng ký, và chúng ta thu thập dữ liệu cá nhân, bao gồm số thẻ tín dụng. Dữ liệu được lưu trong cơ sở dữ liệu quan hệ. Khách hàng có thể đăng nhập, duyệt trang để tìm hàng và đặt hàng. Chương trình cần công bố sự kiện khi có đơn đặt hàng vì ai đó cần phải gửi thư có hạng mục đã được yêu cầu. Chúng ta cần xây dựng giao diện báo cáo dùng cho việc tạo báo cáo để có thể theo dõi trạng thái hàng còn, khách hàng muốn mua gì, họ không thích gì... Ban đầu, chúng ta bắt đầu bằng một mô hình phủ cả domain thương mại điện tử. Chúng ta muốn làm thế vì sau chúng, chúng ta cũng sẽ cần làm một chương trình lớn. Tuy nhiên, chúng ta xem xét công việc này cẩn thận và phát hiện ra rằng chương trình thực ra không liên quan đến chức năng báo cáo. Chúng quan tâm đến cái khác, vận hành theo một cách khác và thậm chí, không cần dùng công nghệ khác. Điểm chung duy nhất ở đây là khách hàng và dữ liệu hàng hóa được lưu trong cơ sở dữ liệu mà cả hai đều truy cập vào.

Cách tiếp cận được khuyến nghị ở đây là tạo mô hình riêng cho mỗi domain, một domain cho e-commerce, một cho phần báo cáo. Chúng có thể tiến hóa tự do không cần quan tâm tới domain khác và có thể trở thành những chương trình độc lập. Có trường hợp là chương trình báo cáo cần một số dữ liệu đặc thù mà chương trình e-commerce lưu trong cơ sở dữ liệu dù cả hai phát triển độc lập.

Một hệ thống thông điệp cần để báo cáo cho người quản lý kho về đơn đặt hàng để họ có thể gửi mail về hàng đã được mua. Người gửi mail sẽ dùng chương trình và cung cấp cho họ thông tin chi tiết về hàng đã mua, số lượng, địa chỉ người mua cũng như yêu cầu giao hàng. Việc có mô hình e-shop phủ cả hai domain là không cần thiết. Sẽ đơn giản hơn nhiều cho chương trình e-shop chỉ gửi đối tượng giá trị chứa thông tin mua tới kho bằng phương thức thông điệp phi đồng bộ. Rõ ràng là có hai mô hình có thể phát triển độc lập và chúng ta chỉ cần đảm bảo giao diện giữa

chúng hoạt động tốt.

2.2.3.2 Tích hợp liên tục

Khi một ngữ cảnh giới hạn đã được định nghĩa, chúng ta cần đảm bảo rằng nó luôn mới và hoạt động như kỳ vọng. Khi nhiều người cùng làm việc trên cùng một ngữ cảnh giới hạn, mô hình có khuynh hướng bị phân mảnh. Nhóm càng lớn, vấn đề càng lớn, nhóm nhỏ từ ba đến bốn người cũng có thể gặp vấn đề nghiêm trọng.

Tuy nhiên, nếu ta chia nhỏ hệ thống thành những ngữ cảnh rất nhỏ thì thực ra, ta đánh mất mức độ giá trị về tích hợp và tính tương liên. Ngay cả khi nhóm làm việc cùng trên một ngữ cảnh giới hạn thì vẫn có thể có lỗi. Chúng ta cần giao tiếp trong nội bộ nhóm để đảm bảo rằng mọi người đều hiểu vai trò của từng phần tử trong mô hình. Nếu có người không hiểu quan hệ giữa các đối tượng, họ có thể sửa mã nguồn làm nó mâu thuẫn với mô hình ban đầu. Lỗi dễ xảy ra và chúng ta không thể đảm bảo 100% tập trung vào sự tinh khiết của mô hình.

Một thành viên của nhóm có thể thêm mã nguồn trùng với mã đã có mà không biết, hay có thể lặp lại mã nguồn mà không hề thay đổi mã hiện tại vì sợ rằng có thể làm hỏng chức năng đang có.

Không thể định nghĩa đầy đủ một mô hình ngay từ đầu. Mô hình được tạo ra, tiến hóa liên tục dựa trên thực tế của domain và phản hồi từ quy trình phát triển. Điều này nghĩa là khái niệm mới có thể xuất hiện trong mô hình, phần tử mới được thêm vào mã nguồn.

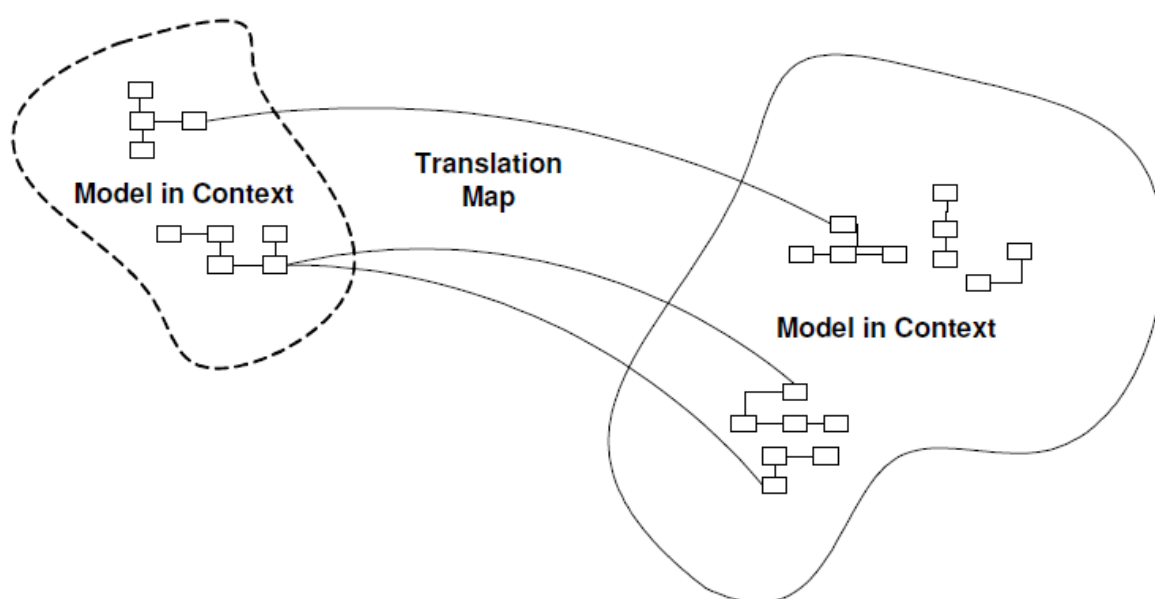
Mọi nhu cầu kiểu này được tích hợp vào một mô hình thống nhất, được thực thi tương ứng trong mã nguồn. Đó là lý do tại sao tích hợp liên tục là quy trình cần thiết trong ngữ cảnh giới hạn. Chúng ta cần quy trình tích hợp để đảm bảo rằng mọi phần tử được thêm vào một cách hài hòa trong toàn bộ phần còn lại của mô hình và được thực thi đúng trong mã nguồn. Chúng ta cần có một thủ tục khi tích hợp mã nguồn, tích hợp mã nguồn càng sớm càng tốt. Với một nhóm nhỏ, nên tích hợp hàng ngày, chúng ta cũng cần xây dựng quy trình. Mã nguồn đã được kết hợp cần được khởi tạo tự động và được kiểm thử. Một yêu cầu cần thiết khác là kiểm thử tự

động. Nếu nhóm có công cụ kiểm thử và tạo ra một bộ kiểm thử thì việc kiểm thử có thể chạy mỗi khi khởi tạo để đưa ra mọi cảnh báo khi có lỗi. Sửa mã nguồn để sửa lỗi đã được báo cáo là dễ vì chúng ta phát hiện sớm. Sau đó chúng ta tiếp tục quy trình với các bước: kết hợp mã nguồn, khởi tạo, kiểm thử tiếp.

Tích hợp liên tục dựa trên khái niệm tích hợp của mô hình, tìm cách thực thi những chỗ được kiểm thử. Bất kỳ sự không nhất quán nào trong mô hình có thể được phát hiện trong quá trình thực thi. Tích hợp liên tục áp dụng vào một ngữ cảnh giới hạn và nó được dùng để đối phó với quan hệ giữa các ngữ cảnh xung quanh.

2.2.3.3 Ngữ cảnh ánh xạ

Một phần mềm doanh nghiệp có nhiều mô hình, mỗi mô hình có ngữ cảnh giới hạn riêng. Bạn nên dùng ngữ cảnh như là cơ sở để tổ chức nhóm. Người trong cùng nhóm sẽ giao tiếp dễ hơn, tích hợp công việc và thực thi công việc dễ hơn. Khi mọi nhóm làm việc với mô hình của riêng họ, mọi người cần hiểu bức tranh tổng thể. Một ánh xạ ngữ cảnh mô tả khái quát các ngữ cảnh giới hạn và quan hệ giữa chúng. Một ánh xạ ngữ cảnh có thể là một giản đồ như hình dưới đây, hoặc là tài liệu viết, mức độ chi tiết của nó tùy trường hợp. Điều quan trọng là mọi người làm việc trên cùng một dự án phải hiểu và cùng chia sẻ bức tranh này.



Hình 2.2.5 Ngữ cảnh ánh xạ [16]

Sẽ là không đủ nếu phải tách riêng mô hình thống nhất. Chúng được tích hợp lại vì chức năng của mỗi mô hình chỉ là một phần của hệ thống. Cuối cùng mỗi mảnh ghép đó gộp lại và cả hệ thống phải chạy đúng. Nếu ngữ cảnh không được định nghĩa rõ ràng, có thể là chúng sẽ đâm chân lên nhau. Nếu quan hệ giữa các ngữ cảnh không được chỉ ra, hệ thống có thể sẽ không chạy khi tích hợp.

Khi việc tích hợp chức năng bị hạn chế, chi phí cho tích hợp liên tục có thể tăng cao. Điều này đặc biệt đúng khi nhóm không đủ kỹ năng hay yếu tài chính của tổ chức để duy trì tích hợp liên tục, hoặc khi cỡ của một nhóm quá lớn và không thể hành động được. Do đó việc phân chia ngữ cảnh giới hạn lúc này mới tách và phân chia thành nhiều nhóm. Nhiều nhóm không gắn kết cùng làm trên một phần mềm có liên kết chặt chẽ có thể chạy độc lập, nhưng kết quả họ làm ra có thể không ăn khớp. Họ sẽ cần nhiều thời gian để dịch lớp và chỉnh lại hơn là dành thời gian cho việc tích hợp liên tục ngay từ ban đầu, tức là trong lúc họ làm việc ấy, họ tốn công vào việc người khác đã làm và đánh vật với lợi ích của ngôn ngữ chung. Do đó, hãy chỉ định một vài tập con của mô hình domain mà các nhóm đồng ý chia sẻ.

Tất nhiên, việc này bao gồm, đi cùng với tập con của mô hình, là tập con của mã nguồn và thiết kế cơ sở dữ liệu đi cùng với phần tương ứng của mô hình. Rõ ràng, việc làm này chia sẻ những cái có trạng thái đặc biệt và không thể được thay đổi nếu không có sự tham vấn của nhóm khác.

Hãy tích hợp một hệ thống chức năng thường xuyên, nhưng ít thường xuyên hơn nhịp của tích hợp liên tục trong nội bộ nhóm. Trong những phiên tích hợp này, hãy chạy kiểm thử cho cả hai nhóm. Mục đích của nhân chung là giảm sự lặp lại, nhưng vẫn giữ riêng hai ngữ cảnh riêng biệt. Phát triển trên một nhân chung đòi hỏi độ chú ý cao hơn. Cả hai nhóm có thể thay đổi mã nguồn của nhân, và họ phải tích hợp các thay đổi. Nếu các nhóm chia riêng bản sao chép của mã nguồn nhân, họ phải tích hợp mã nguồn lại càng sớm càng tốt, ít nhất là một tuần một lần. Một bộ kiểm thử là cần thiết để mọi thay đổi tới nhân cần được báo cho các nhóm khác để họ biết khi có chức năng mới.

2.3 MẪU COMMAND QUERY RESPONSIBILITY SEGREGATION – CQRS

Nhiều ứng dụng doanh nghiệp sử dụng một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System – RDBMS) làm hệ thống lưu trữ dữ liệu và một cơ sở dữ liệu hỗ trợ truy vấn, tìm kiếm dữ liệu như Elasticsearch, Solr. Một số ứng dụng đảm bảo hai cơ sở dữ liệu này được đồng bộ bằng cách ghi dữ liệu vào cả hai cùng một lúc. Một số khác thì định kỳ sao chép dữ liệu từ RDBMS sang cơ sở dữ liệu văn bản (text database). Các ứng dụng với kiến trúc này tận dụng ưu điểm của nhiều cơ sở dữ liệu: các tính chất khi thực hiện các transactions (transaction là một chuỗi các hành động được thực thi một cách tuần tự và độc lập với nhau trong cơ sở dữ liệu) trong RDBMS và khả năng truy vấn trong cơ sở dữ liệu văn bản.

Mẫu Command Query Responsibility Segregation (CQRS) là một khái quát của dạng kiến trúc này. Để hiểu rõ hơn về CQRS, cùng tìm hiểu về định nghĩa, cách hoạt động cũng như tầm ảnh hưởng đối với kiến trúc ứng dụng khi áp dụng nó qua các phần sau.

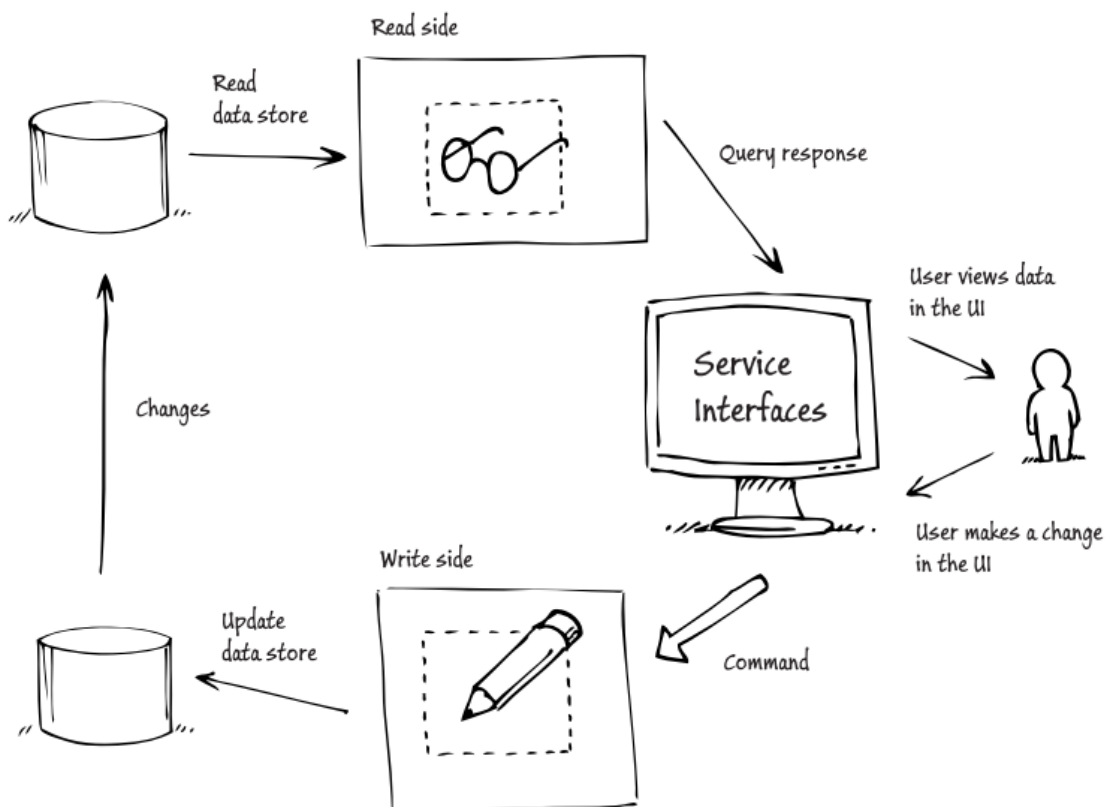
2.3.1 Định nghĩa

Trong quyển sách “Object Oriented Software Construction”, Bertrand Meyer đã giới thiệu một thuật ngữ “Command Query Separation” để mô tả một nguyên tắc rằng các phương thức của đối tượng nên là những lệnh (commands) hoặc những câu truy vấn (queries). Một query có nhiệm vụ trả về dữ liệu và không thay đổi trạng thái của đối tượng. Một command thì ngược lại, nó thay đổi trạng thái của đối tượng nhưng không trả về dữ liệu nào. Nguyên tắc này giúp ta hiểu rõ hơn cái gì có thể và không thể thay đổi trạng thái của các đối tượng trong hệ thống.

CQRS dựa trên nguyên tắc này và mở rộng thêm để định nghĩa nên một mẫu (pattern) đơn giản.

Mẫu CQRS được giới thiệu lần đầu bởi Greg Young và được định nghĩa như

sau: “CQRS đơn giản là việc tạo ra hai đối tượng trong khi trước đó chỉ có một. Sự phân tách này xảy ra dựa trên việc định nghĩa các phương thức là một command hay một query” [17]. Đúng như tên gọi của nó, CQRS giao trách nhiệm sửa đổi và truy vấn dữ liệu hệ thống cho hai đối tượng khác nhau: mô hình đọc và mô hình ghi. Hình 2.3.1 mô tả một ứng dụng điển hình sử dụng mẫu CQRS cho một phần của hệ thống.



Hình 2.3.1 Một cách triển khai kiến trúc khi áp dụng mẫu CQRS [17]

Hình 2.3.1 cho thấy cách chia một phần của hệ thống thành hai phía, phía ghi (write side) và phía đọc (read side) dữ liệu. Những đối tượng bên phía đọc chỉ chứa những queries, còn những đối tượng bên phía ghi chỉ chứa những commands. Có nhiều lý do dẫn đến sự tách biệt này, bao gồm:

- Trong nhiều hệ thống, có một sự mất cân đối khá lớn giữa số lần đọc và số lần ghi dữ liệu. Thời gian một hệ thống xử lý hàng ngàn yêu cầu truy vấn dữ liệu có thể tương ứng với thời gian hệ thống đó xử lý một yêu cầu ghi dữ

liệu. Sự phân tách thành hai phía cho phép tối ưu hoá hai hành động đọc và ghi dữ liệu một cách độc lập.

- Thông thường, các commands chứa logic nghiệp vụ phức tạp để đảm bảo rằng hệ thống ghi dữ liệu chính xác và nhất quán vào kho lưu trữ dữ liệu (data store). Thao tác đọc thường đơn giản hơn rất nhiều so với thao tác ghi. Sự phân tách thành hai phía tạo ra những mô hình đơn giản, dễ bảo trì và linh hoạt hơn.
- Sự tách biệt cũng có thể xảy ra ở cấp lưu trữ dữ liệu. Phía ghi dữ liệu có thể sử dụng một lược đồ cơ sở dữ liệu được chuẩn hoá gần đạt đến dạng chuẩn 3 (third normal form – 3NF) và được tối ưu hoá phục vụ cho việc sửa đổi dữ liệu. Trong khi phía đọc dữ liệu có thể sử dụng một cơ sở dữ liệu không được chuẩn hoá (denormalized) để tối ưu các thao tác truy vấn. Mặc dù hình 2.3.1 sử dụng hai kho lưu trữ dữ liệu, việc áp dụng mẫu CQRS không bắt buộc phải chia tách kho lưu trữ dữ liệu hay phải sử dụng một công nghệ lưu trữ cụ thể nào như cơ sở dữ liệu quan hệ hay phi quan hệ,... Mẫu CQRS chỉ tạo điều kiện cho việc phân tách kho lưu trữ dữ liệu và cho phép lựa chọn tuỳ ý cơ chế lưu trữ dữ liệu.

Điều quan trọng và thú vị về mẫu CQRS nằm ở việc tại sao lại sử dụng nó, khi nào nên sử dụng và sử dụng như thế nào khi xây dựng hệ thống. Những phần tiếp theo sẽ lần lượt giải đáp các câu hỏi này.

2.3.2 CQRS và DDD

Nhiều ý kiến cho rằng mẫu CQRS ra đời để giải quyết một số vấn đề gặp phải khi áp dụng DDD tiếp cận các vấn đề trong thế giới thực. Do vậy, nếu sử dụng DDD, có thể thấy rằng mẫu CQRS rất phù hợp cho một vài ngữ cảnh giới hạn được xác định bên trong hệ thống. Một số chuyên gia cho rằng việc áp dụng DDD là điều kiện tiên quyết để triển khai mẫu CQRS. Tuy nhiên, Greg Young cho rằng việc áp dụng DDD không phải là điều kiện tiên quyết khi triển khai mẫu CQRS, nhưng trong thực tế, chúng thường đi đôi với nhau [17].

Có hai lĩnh vực quan trọng cần được nhắc đến:

- Khái niệm về mô hình: Trong hình 2.3.1 việc triển khai mô hình tồn tại bên phía ghi dữ liệu. Nó gói gọn những logic nghiệp vụ phức tạp diễn ra trong một phần của hệ thống. Phía đọc dữ liệu là một góc nhìn các trạng thái hệ thống. Góc nhìn này đơn giản, chỉ được phép đọc và truy cập thông qua các queries.
- Cách DDD phân chia các hệ thống lớn, phức tạp thành nhiều đơn vị dễ quản lý hơn hay còn gọi là các ngữ cảnh giới hạn. Một ngữ cảnh giới hạn định nghĩa một ngữ cảnh cho một mô hình.

Khi nói rằng nên áp dụng mẫu CQRS cho một phần của hệ thống, nó có nghĩa rằng nên áp dụng mẫu CQRS trong một ngữ cảnh giới hạn định nghĩa bởi DDD. Nên xác định rõ những phần khác nhau trong hệ thống mà có thể thiết kế và triển khai độc lập với nhau và chỉ nên áp dụng mẫu CQRS vào những phần có thể đem lại lợi ích nghiệp vụ rõ ràng khi áp dụng nó.

2.3.3 Commands, events và các thông điệp (messages)

DDD là một phương pháp thiết kế từ nghiệp vụ khuyến khích sử dụng các mô hình và một ngôn ngữ phổ biến để thu hẹp khoảng cách giữa doanh nghiệp và đội ngũ phát triển bằng cách bồi dưỡng sự hiểu biết thông thường về một lĩnh vực. Do đó, DDD được định hướng để phân tích hành vi thay vì chỉ phân tích dữ liệu trong lĩnh vực nghiệp vụ, tập trung mô hình hoá và triển khai cài đặt hành vi trong phần mềm. Để triển khai mô hình nghiệp vụ bằng những dòng mã một cách tự nhiên, cần sử dụng commands và events.

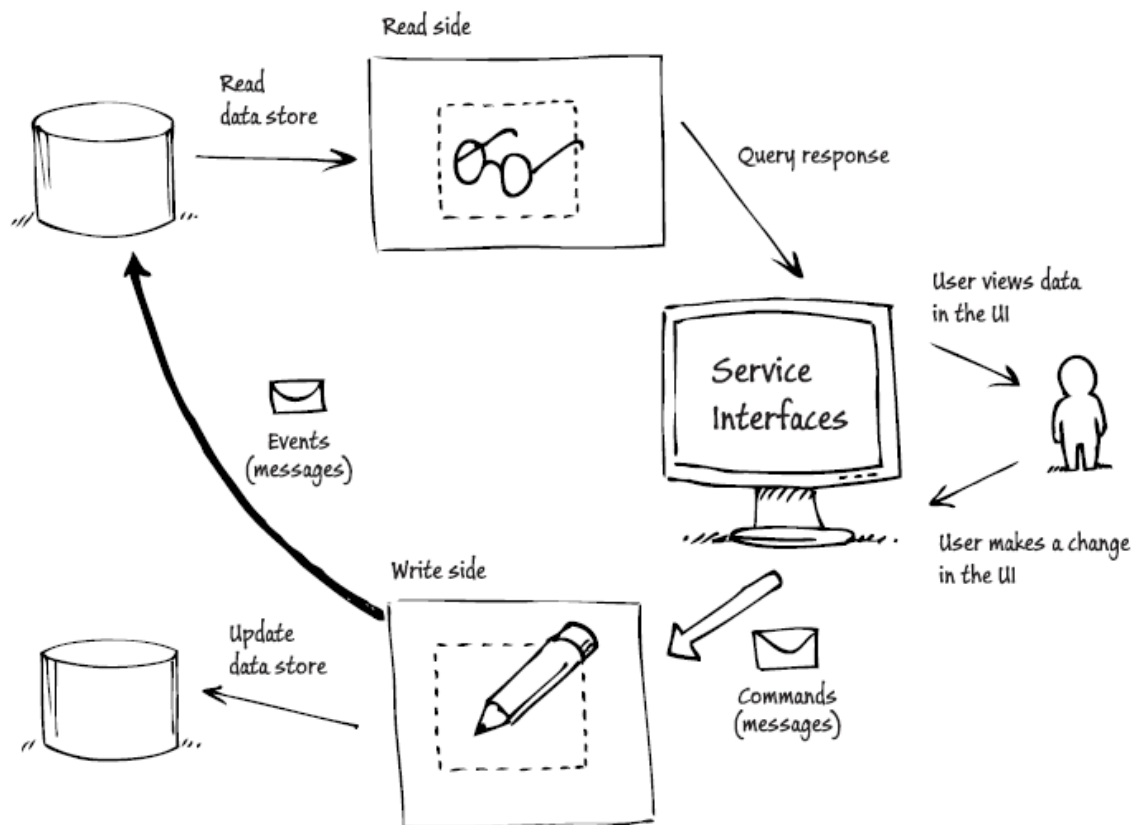
DDD không phải là cách tiếp cận duy nhất sử dụng commands và events để triển khai các nhiệm vụ và hành vi được xác định trong mô hình nghiệp vụ. Tuy nhiên, nhiều người ủng hộ mẫu CQRS cũng bị tác động bởi DDD nên mỗi khi có một cuộc thảo luận về mẫu CQRS, thuật ngữ DDD cũng thường được nhắc đến. Commands là những lệnh yêu cầu hệ thống thực hiện một tác vụ hay một hành động. Ví dụ,

“đặt hai chỗ ngồi tại hội nghị X” hay “phân bổ người phỏng vấn Y vào phòng Z”. Commands thường được xử lý chỉ một lần bởi một bên nhận được nó.

Events là những thông báo mô tả điều đã xảy ra đến các bên quan tâm khác. Ví dụ, “đơn hàng đã được giao đến địa chỉ A” hay “mười chỗ ngồi đã được đặt tại hội nghị X”. Events có thể được xử lý nhiều lần bởi nhiều bên nhận được nó.

Commands và events là những loại thông điệp dùng để trao đổi dữ liệu giữa các đối tượng. Trong DDD, những thông điệp này đại diện cho những hành vi nghiệp vụ. Do đó, chúng giúp hệ thống nắm bắt được hành vi nghiệp vụ ẩn sau các thông điệp.

Một cách triển khai mẫu CQRS có thể sử dụng nhiều kho lưu trữ dữ liệu riêng biệt cho phía đọc và phía ghi dữ liệu. Mỗi kho lưu trữ dữ liệu được tối ưu hoá phục vụ các trường hợp thao tác với dữ liệu mà nó hỗ trợ. Events làm cơ sở cho cơ chế đồng bộ các thay đổi tại phía ghi (kết quả của việc xử lý các commands) với phía đọc. Nếu phía ghi đưa ra một event mỗi khi trạng thái hệ thống có sự thay đổi, phía đọc sẽ phản hồi đối với event đó và cập nhật dữ liệu được dùng để truy vấn.



Hình 2.3.2 Commands và events trong mẫu CQRS [17]

2.3.4 Lợi ích của mẫu thiết kế CQRS

Một trong những lợi ích của việc phân chia lĩnh vực nghiệp vụ thành nhiều ngữ cảnh giới hạn trong DDD là có thể xác định và tập trung vào những phần (ngữ cảnh giới hạn) phức tạp hơn.

Những phần bị ảnh hưởng bởi những quy tắc nghiệp vụ luôn bị thay đổi hay phải triển khai một chức năng quan trọng, đặc trưng trong hệ thống. Nên xem xét áp dụng mẫu CQRS trong một ngữ cảnh giới hạn cụ thể khi và chỉ khi nó đem lại lợi ích nghiệp vụ rõ ràng. Những lợi ích nghiệp vụ phổ biến có thể đạt được khi áp dụng mẫu CQRS là tăng cường khả năng mở rộng, đơn giản hoá khía cạnh phức tạp trong nghiệp vụ, tăng tính linh hoạt trong các giải pháp và thích ứng tốt hơn đối với sự thay đổi các yêu cầu nghiệp vụ.

2.3.4.1 Khả năng mở rộng

Trong nhiều hệ thống lớn, lượng yêu cầu đọc dữ liệu lớn hơn rất nhiều so với lượng yêu cầu ghi, do đó yêu cầu mở rộng sẽ khác nhau ở hai phía. Bằng cách phân tách phía đọc và phía ghi thành hai mô hình riêng biệt trong ngữ cảnh giới hạn, giờ đây có thể mở rộng mỗi phía một cách độc lập.

2.3.4.2 Đơn giản hoá sự phức tạp

Tại nhiều khía cạnh phức tạp trong nghiệp vụ, thiết kế và triển khai các đối tượng chịu trách nhiệm cho cả hai phần đọc và ghi dữ liệu có thể làm tăng độ phức tạp sẵn có. Trong nhiều trường hợp, logic xử lý nghiệp vụ phức tạp chỉ nên được áp dụng khi hệ thống phải xử lý các cập nhật và thao tác với dữ liệu trong cơ sở dữ liệu. Logic đọc dữ liệu thường đơn giản hơn nhiều. Khi logic nghiệp vụ và logic đọc kết hợp với nhau trong cùng một mô hình, việc giải quyết các vấn đề khó như nhiều người dùng, dữ liệu có thể chia sẻ, hiệu suất, tính nhất quán, dữ liệu cũ (stale data) trở nên khó khăn hơn nhiều. Việc tách logic nghiệp vụ và logic đọc dữ liệu thành những mô hình riêng biệt giúp dễ dàng phân chia và giải quyết những vấn đề phức tạp này. Một lợi ích tiềm năng khác của việc đơn giản hoá ngữ cảnh giới hạn bằng cách tách biệt logic đọc và logic nghiệp vụ là nó cho phép kiểm thử một cách dễ dàng hơn.

2.3.4.3 Tính linh hoạt

Tính linh hoạt của một giải pháp áp dụng mẫu CQRS xuất phát từ việc phân tách thành những mô hình phía đọc và phía ghi. Việc thay đổi ở phía đọc giờ đây trở nên dễ dàng hơn nhiều, ví dụ như việc thêm một câu truy vấn mới hỗ trợ hiển thị dữ liệu cho màn hình thống kê trên giao diện người dùng, không cần phải lo lắng nó sẽ ảnh hưởng đến hành vi xử lý bên logic nghiệp vụ. Tại phía ghi, có một mô hình chỉ quan tâm đến phần logic xử lý cốt lõi trong nghiệp vụ đồng nghĩa với việc xử lý một mô hình đơn giản hơn là một mô hình bao gồm cả logic đọc dữ liệu.

Về lâu dài, một mô hình tốt, hữu dụng, mô tả chính xác logic xử lý nghiệp vụ cốt lõi sẽ trở thành một tài sản có giá trị. Nó cho phép bạn trở nên nhanh nhẹn hơn khi đối mặt với một môi trường nghiệp vụ thay đổi và áp lực cạnh tranh đối với tổ

chức của bạn.

Tính linh hoạt và nhanh nhẹn này liên quan đến khái niệm tích hợp liên tục trong DDD. Theo Eric Evans [18], tích hợp liên tục có nghĩa là mọi công việc trong ngữ cảnh đang được đồng nhất và việc này được thực hiện thường xuyên đủ để khi có sự cố xảy ra, chúng được phát hiện và sửa chữa nhanh chóng.

Trong một vài trường hợp, có thể có những đội ngũ phát triển khác nhau làm việc bên phía đọc và phía ghi dữ liệu, mặc dù trong thực tế, điều này có thể phụ thuộc vào ngữ cảnh giới hạn cụ thể có mức độ lớn như thế nào.

2.3.4.4 Tập trung vào nghiệp vụ

Việc áp dụng mẫu CQRS giúp tập trung vào nghiệp vụ và xây dựng giao diện người dùng theo hướng nhiệm vụ. Kết quả của việc tách biệt thành hai phía đọc và ghi dữ liệu là một giải pháp thích ứng tốt đối với sự thay đổi yêu cầu nghiệp vụ. Về lâu dài, giải pháp này giúp tiết kiệm chi phí phát triển và bảo trì.

2.3.5 Khi nào nên áp dụng mẫu CQRS?

Mặc dù đã có những lý do giúp đưa ra quyết định có nên áp dụng mẫu CQRS vào một số ngữ cảnh giới hạn trong hệ thống hay không, vẫn cần những quy tắc giúp xác định ngữ cảnh giới hạn nào nên áp dụng CQRS để đạt được lợi ích nhất định.

Nhìn chung, những ngữ cảnh giới hạn có tính tương tác, phức tạp, gồm những qui tắc nghiệp vụ luôn bị thay đổi và phải triển khai một chức năng quan trọng, đặc trưng trong hệ thống khi áp dụng CQRS sẽ mang lại giá trị cao nhất. Phân tích các yêu cầu nghiệp vụ, xây dựng một mô hình hữu dụng, thể hiện nó bằng các dòng mã và triển khai nó bằng việc áp dụng mẫu CQRS cho một ngữ cảnh giới hạn đều mất nhiều thời gian và chi phí. Nếu không kỳ vọng những lợi ích như khả năng thích ứng và tính linh hoạt trong hệ thống tăng hay chi phí bảo trì giảm thì không đáng để đầu tư thời gian và chi phí như vậy.

2.3.5.1 Những nghiệp vụ có tính tương tác

Theo Udi Dahan, trong một nghiệp vụ mang tính tương tác, một thuộc tính vốn có của nghiệp vụ là có nhiều hoạt động diễn ra song song trên cùng một bộ dữ liệu. Một hệ thống đặt chỗ cho một buổi hoà nhạc là một ví dụ điển hình cho một nghiệp vụ mang tính tương tác, mọi người đều muốn đặt được chỗ ngồi tốt nhất [19]. Cả Udi Dahan và Greg Young đều xác định sự tương tác là một đặc điểm trong một ngữ cảnh giới hạn mà có thể cho thấy các lợi ích một cách rõ ràng nhất khi áp dụng mẫu CQRS.

Mẫu CQRS đặc biệt hữu ích khi sự tương tác bao gồm các quyết định phức tạp về kết quả sẽ như thế nào khi có nhiều hoạt động diễn ra trên cùng tập dữ liệu. Các phần mang tính tương tác như vậy trong hệ thống thường là những ngữ cảnh giới hạn phức tạp nhất, tuy nhiên, đặc điểm này chỉ là một sự gợi ý. Không phải tất cả những nghiệp vụ có tính tương tác đều đạt được lợi ích khi sử dụng mẫu CQRS.

2.3.5.2 Dữ liệu cũ

Trong một môi trường có tính tương tác, nơi mà nhiều người dùng sẽ thao tác đồng thời trên cùng một tập dữ liệu, vấn đề về dữ liệu cũ sẽ xuất hiện. Nếu một người dùng đang xem một phần dữ liệu trong khi có một người dùng khác thay đổi nó, phần dữ liệu mà người dùng đang xem trở thành dữ liệu cũ.

Dù xây dựng theo kiến trúc nào, vấn đề này cũng cần được giải quyết. Mẫu CQRS giúp giải quyết vấn đề này một cách rõ ràng ở cấp kiến trúc. Những thay đổi về dữ liệu xảy ra bên phía ghi, người dùng xem dữ liệu bằng cách truy vấn bên phía đọc. Cơ chế được chọn dùng để đẩy các thay đổi dữ liệu từ phía ghi sang phía đọc cũng là cơ chế dùng để kiểm soát khi nào dữ liệu phía đọc trở nên cũ và thời gian duy trì như vậy là bao lâu.

2.4 MÔ HÌNH EVENT SOURCING – ES

Mô hình Event Sourcing (ES) và mô hình CQRS là những mô hình thường được kết hợp cùng nhau để phát huy tối ưu hiệu năng. Mặc dù không có bất cứ ràng buộc nào về kiến trúc, nhưng những yếu tố bổ sung cho nhau đã làm cho hầu hết các kiến trúc phần mềm có ES đều được xây dựng và phát triển với sự tham gia của CQRS. Phần này sẽ trình bày lý thuyết nền tảng của mô hình ES phục vụ cho kiến trúc hệ thống và mối quan hệ mật thiết với mô hình CQRS.

2.4.1 Định nghĩa

2.4.1.1 Events là gì?

Events là những sự kiện xảy ra trong quá khứ. Ví dụ, ta có các events “chỗ ngồi đã được đặt”, “đơn hàng đã được tạo”, “tiền đã được hoàn trả”,...

Events là dữ liệu bất biến, vì events là những sự kiện đã xảy ra nên không thể thay đổi hay hoàn tác chúng. Tuy nhiên, các sự kiện diễn ra sau đó có thể thay đổi hay phủ nhận các sự kiện trước đó. Ví dụ, ta có event “đơn hàng đã bị hủy” là một event thay đổi kết quả của event trước nó “đơn hàng đã được tạo”.

Events là những tin nhắn một chiều (one-way messages). Chỉ tồn tại một nguồn (publisher) để công bố (publish) các events và có thể có một hoặc nhiều nguồn nhận (subscribers) nhận được các events này.

Đặc biệt, events bao gồm những thông số cung cấp thông tin về events. Ví dụ, event “Chỗ ngồi có mã số J26 đã được đặt bởi Linh” bao gồm 2 thông số “J26” và “Linh” là những thông tin quan trọng xác định chỗ ngồi nào được đặt bởi ai.

2.4.1.2 Event Sourcing là gì?

Mô hình Event Sourcing là phương pháp duy trì trạng thái ứng dụng bằng cách lưu trữ lịch sử để xác định trạng thái hiện tại của ứng dụng. Ví dụ, một hệ thống quản lý hội nghị cần theo dõi số lượng chỗ ngồi đã được đặt cho một hội nghị để có thể kiểm tra số ghế còn trống khi có yêu cầu đặt chỗ. Hệ thống sẽ lưu trữ tổng số lượng đặt chỗ cho một hội nghị theo 2 cách:

- Lưu trữ tổng số lượng đặt chỗ cho một hội nghị và điều chỉnh số lượng này mỗi khi có yêu cầu đặt hoặc hủy chỗ.

Lưu trữ tất cả events đặt chỗ và hủy chỗ cho mỗi hội nghị. Sau đó, tính toán số lượng đặt chỗ hiện tại bằng cách tái diễn (replay) các events liên quan đến hội nghị có nhu cầu kiểm tra tổng số lượng đặt chỗ hiện tại. Cách này chính là việc xây dựng hệ thống theo mô hình ES.

2.4.1.2.1 So sánh giải pháp sử dụng tầng ORM (Object Relational Mapping) và ES

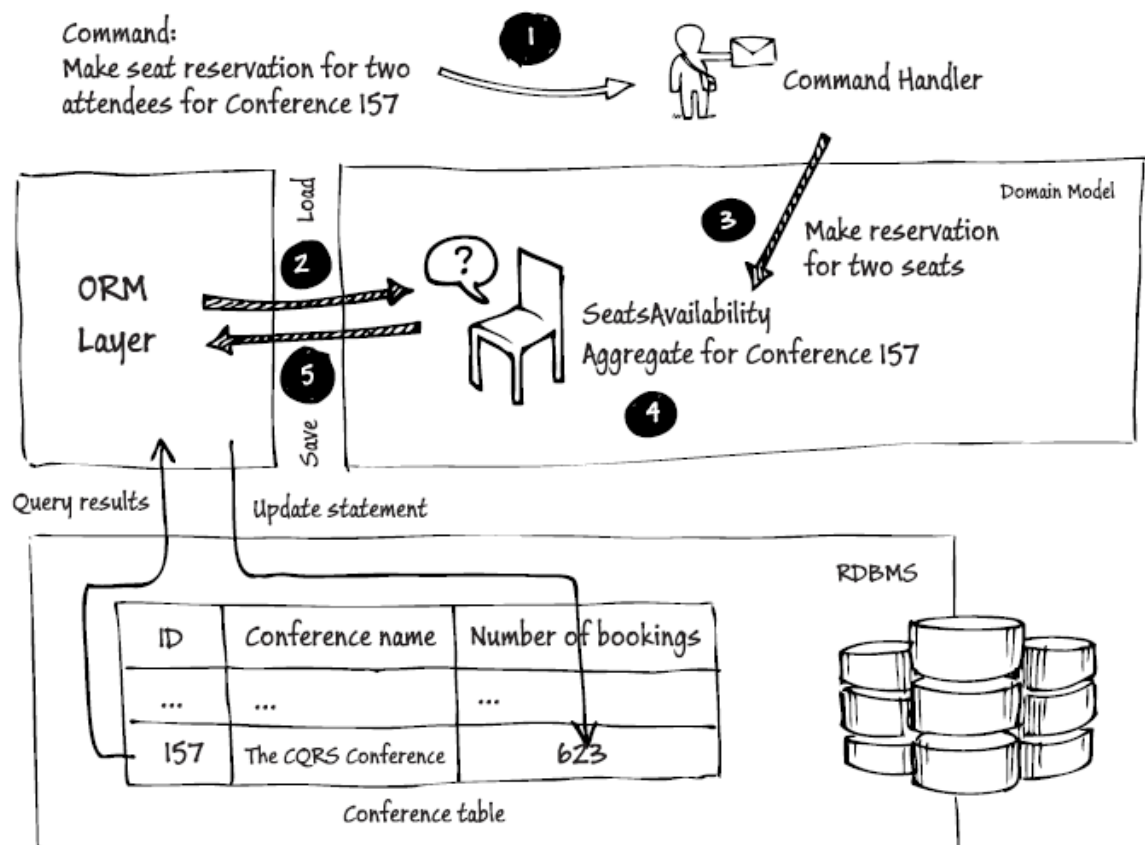


FIGURE 1
Using an object-relational mapping layer

Hình 2.4.1 Mô tả cách tiếp cận thứ nhất để lưu trữ tổng số lượng đặt chỗ, sử dụng tầng ORM [17]

Trong đó, ORM là một kỹ thuật lập trình giúp ánh xạ dữ liệu trong RDBMS sang

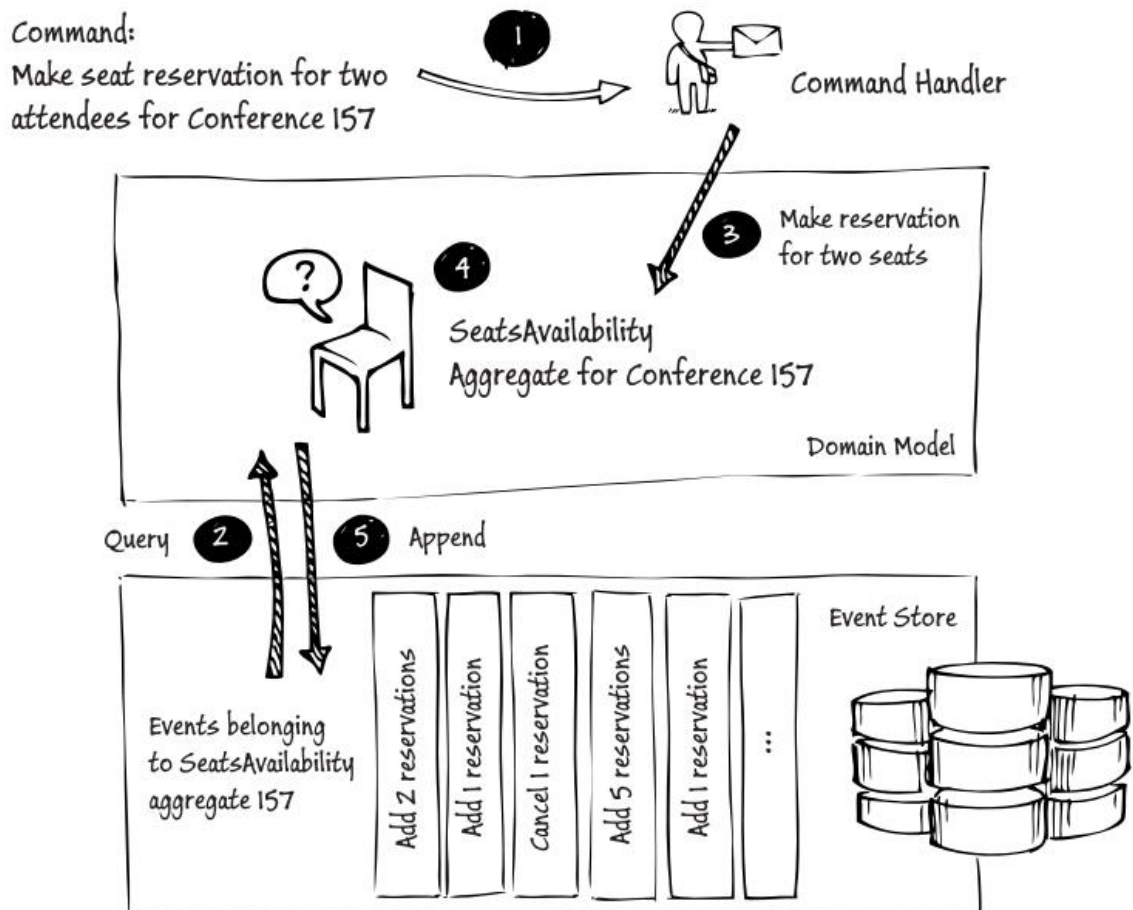
đối tượng được định nghĩa trong các lớp:

- Bước 1: Một sagas hoặc giao diện người dùng (user interface - UI) sẽ đưa ra một command đặt chỗ cho hai người tại một hội nghị có ID là 157. Command này được xử lý bởi command handler của loại aggregate SeatsAvailability.
- Bước 2: Nếu cần thiết, tầng ORM sẽ tạo một thể hiện (instance) của aggregate với dữ liệu truy vấn tại kho lưu trữ dữ liệu. Dữ liệu này bao gồm số lượng đặt chỗ hiện tại cho hội nghị có ID là 157.
- Bước 3: Command handler sẽ gọi những phương thức xử lý trên instance được tạo ở bước 2 để thực hiện đặt chỗ.
- Bước 4: Aggregate SeatsAvailability thực hiện logic xử lý của nó. Trong ví dụ này, nó tính toán lại số lượng đặt chỗ cho hội nghị.
- Bước 5: Tầng ORM cập nhật thông tin vừa được xử lý của instance được tạo ở bước 2 tại kho lưu trữ dữ liệu.

Cách tiếp cận này hiệu quả vì hầu hết các ứng dụng doanh nghiệp đều lưu trữ dữ liệu theo cách này. Tuy nhiên, tồn tại nhiều hạn chế và giới hạn:

- Khó khăn trong việc lưu trữ những đối tượng dữ liệu phức tạp của một ứng dụng được viết theo ngôn ngữ hoặc kiểu lập trình hướng đối tượng vào kho lưu trữ dữ liệu quan hệ (relational store), thuật ngữ gọi là Object-Relational impedance mismatch.
- Thiếu lịch sử thay đổi trạng thái của aggregate. Cách tiếp cận này chỉ lưu lại trạng thái hiện tại của aggregate. Một khi aggregate được cập nhật, không thể tìm thấy trạng thái trước đó của nó. Nếu một hệ thống phải lưu giữ lịch sử của một aggregate, các lập trình viên phải tự triển khai cơ chế này. Việc triển khai lưu trữ lịch sử cho các aggregate sẽ tốn rất nhiều thời gian.

- Công bố events là một cơ chế hữu dụng trong việc đồng bộ dữ liệu và gửi thông báo trong kiến trúc microservice. Tuy nhiên, cách tiếp cận này không hỗ trợ tự động công bố events mỗi khi aggregate được cập nhật. Do đó, lập trình viên phải tự triển khai dẫn đến khả năng gây ra sự bất đồng bộ với logic xử lý.



Hình 2.4.2 Mô tả cách tiếp cận thứ hai để lưu trữ tổng số lượng đặt chỗ, sử dụng ES thay vì tầng ORM và một RDBMS [17]

Trong đó, bước 1, 3, 4 giống với giải pháp sử dụng tầng ORM:

- Bước 1: Một saga hoặc giao diện người dùng (user interface - UI) sẽ đưa ra một command đặt chỗ cho hai người tại một hội nghị có ID là 157. Command này được xử lý bởi command handler của loại aggregate SeatsAvailability.

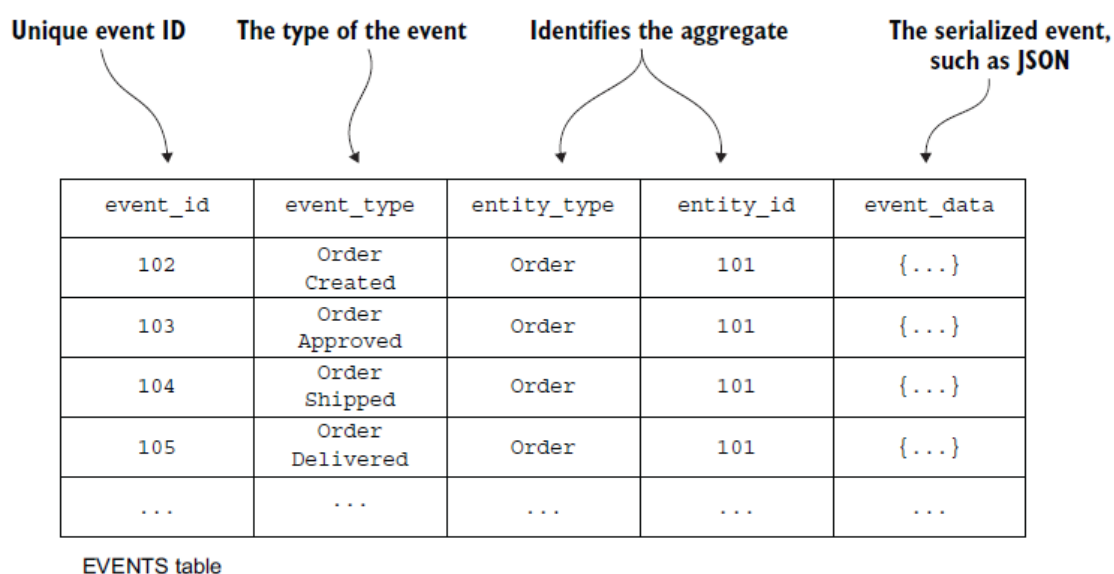
- Bước 2: Một instance được tạo bằng cách truy vấn tất cả những events liên quan đến aggregate SeatsAvailability với ID 157.
- Bước 3: Command handler sẽ gọi những phương thức xử lý trên instance được tạo ở bước 2 để thực hiện đặt chỗ.
- Bước 4: Aggregate SeatsAvailability thực hiện logic xử lý của nó. Trong ví dụ này, nó tính toán lại số lượng đặt chỗ cho hội nghị. Đồng thời, nó tạo một event tương ứng với command ban đầu.
- Bước 5: Hệ thống sẽ thêm event đặt chỗ vào danh sách các events liên quan đến aggregate SeatsAvailability với ID 157 trong event store.

Có thể thấy cách tiếp cận thứ hai đơn giản hơn vì nó không cần dùng đến tầng ORM và nó thay thế một lược đồ quan hệ phức tạp trong kho lưu trữ dữ liệu với một cái đơn giản hơn – event store. Event store là nơi lưu trữ các events và trả về luồng events liên quan đến một instance của aggregate để có thể tái diễn các events và tạo lập trạng thái cho aggregate. Event store chỉ cần hỗ trợ truy vấn events bằng ID của aggregate và thêm vào những events mới. Với cách tiếp cận này, ta có được một lịch sử đầy đủ chứa các yêu cầu đặt hoặc huỷ chỗ của một hội nghị. Do đó, luồng sự kiện (event stream) trở thành “source of truth”, các events được truy xuất từ một nguồn duy nhất là event stream, dẫn đến mọi trạng thái của ứng dụng đều được tái lập từ event stream. Việc khôi phục trạng thái của ứng dụng tại bất kỳ thời điểm nào trở nên dễ dàng nhờ việc tái diễn events.

2.4.1.2.2 *Event Sourcing duy trì trạng thái của aggregate bằng cách sử dụng events*

ES duy trì mỗi aggregate dưới dạng một chuỗi các events trong cơ sở dữ liệu, gọi là event store. Hình 2.4.3 cho thấy một ví dụ về việc duy trì trạng thái của aggregate Đơn hàng. Thay vì lưu trữ mỗi đơn hàng như một hàng (row) trong bảng Đơn hàng, ES lưu trữ mỗi đơn hàng như một hay nhiều hàng trong bảng Events. Mỗi hàng là một events, ví dụ như “Đơn hàng đã được tạo” (Order Created), “Đơn

hàng được chấp nhận” (Order Approved), “Đơn hàng đã được giao” (Order Shipped),...



Hình 2.4.3 ES duy trì mỗi aggregate dưới dạng một chuỗi các events, lưu trong bảng Events [10]

Khi hệ thống tạo hay cập nhật một aggregate, nó thêm events được tạo ra bởi aggregate vào bảng Events. Hệ thống sẽ lấy một aggregate từ event store bằng cách lấy các events liên quan đến nó và tái diễn các events này. Cụ thể, việc lấy một aggregate bao gồm ba bước:

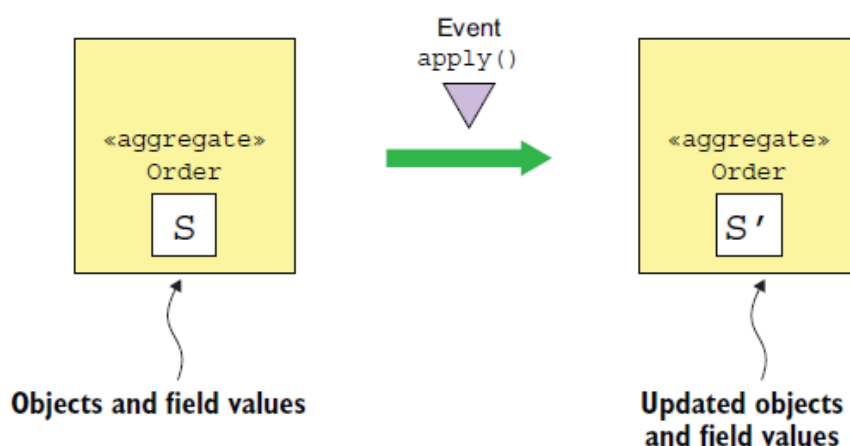
- Bước 1: Lấy các sự kiện liên quan đến aggregate từ event store.
- Bước 2: Tạo một instance cho aggregate sử dụng hàm khởi tạo mặc định.
- Bước 3: Lặp qua các events và gọi phương thức *apply()*.

Cách lấy aggregate này thực chất không khác biệt lắm so với việc tầng ORM lấy một thực thể (entity). Tầng ORM lấy một đối tượng bằng cách thực hiện một hoặc nhiều câu truy vấn SELECT để lấy được trạng thái hiện tại, sau đó khởi tạo đối tượng bằng hàm khởi tạo mặc định của nó. Điều khác biệt ở ES là việc xây dựng lại trạng thái của aggregate được thực hiện bằng các events.

2.4.1.2.3 Events đại diện cho sự thay đổi trạng thái

Mỗi sự thay đổi trạng thái của aggregate, bao gồm sự tạo lập, được đại diện bởi một event. Mỗi khi trạng thái aggregate thay đổi, nó phải đưa ra một event. Ví dụ, một aggregate Đơn hàng phải đưa ra event “Đơn hàng đã được tạo” khi nó được tạo, và event tương ứng khi nó được cập nhật.

Một event phải chứa dữ liệu cần thiết để aggregate thực hiện thay đổi trạng thái. Trạng thái của aggregate bao gồm giá trị của các thuộc tính của các đối tượng tạo nên aggregate. Một sự thay đổi trạng thái có thể chỉ là một sự thay đổi giá trị của một thuộc tính của một đối tượng. Đồng thời, nó cũng có thể bao gồm việc thêm hoặc xóa đối tượng, như việc thêm hay xóa các mặt hàng trong một đơn hàng. Ví dụ trạng thái hiện tại của aggregate là S và trạng thái mới là S' như trong hình 2.4.4. Một event E đại diện cho sự thay đổi trạng thái phải chứa dữ liệu để khi đơn hàng đang ở trạng thái S, gọi phương thức apply(E) thì trạng thái đơn hàng sẽ được cập nhật thành S'.



Hình 2.4.4 Mô tả sự thay đổi trạng thái aggregate Đơn hàng từ S sang S' khi gọi phương thức apply(E). Event E phải chứa dữ liệu cần thiết để thực hiện việc chuyển đổi này [10]

2.4.2 Lợi ích của mô hình Event Sourcing

Event Sourcing chứa một lịch sử đầy đủ các events liên quan đến các

aggregates. Đây là một tính năng quan trọng trong một số lĩnh vực, chẳng hạn như kế toán, ngân hàng, nơi luôn cần đến một lịch sử giao dịch tài chính đầy đủ và các giao dịch không thể bị thay đổi. Khi một giao dịch xảy ra, hệ thống không thể xóa hoặc thay đổi nó, mà chỉ có thể tạo một giao dịch mới để điều chỉnh hoặc hoàn tác nếu cần thiết [17].

- Hiệu suất: Events là dữ liệu bất biến, chỉ có thể dùng thao tác thêm vào (append-only) khi lưu trữ chúng. Đồng thời, events là những đối tượng đơn giản, độc lập. Hai yếu tố này làm tăng hiệu suất và khả năng mở rộng cho hệ thống so với việc sử dụng các mô hình lưu trữ phức tạp.
- Đơn giản hoá: Events là những đối tượng mô tả những gì đã xảy ra trong hệ thống. Bằng cách đơn giản là lưu trữ các events, sẽ tránh được sự phức tạp liên quan đến việc lưu trữ các đối tượng dữ liệu vào cơ sở dữ liệu quan hệ.
- Lưu vết các thay đổi: Event Sourcing lưu trữ một lịch sử đầy đủ những trạng thái của hệ thống. Từ đó, có thể kiểm tra chi tiết những gì đã diễn ra trong hệ thống.
- Tích hợp với các hệ thống khác: Events cho phép tương tác một cách hiệu quả với các hệ thống khác. Event store có thể công bố sự kiện để thông báo cho các hệ thống sự thay đổi trạng thái của hệ thống.
- Trích xuất dữ liệu nghiệp vụ từ lịch sử sự kiện: Việc lưu trữ events cho phép khả năng xác định trạng thái của hệ thống tại thời điểm bất kỳ bằng cách truy vấn những events có liên quan đến đối tượng dữ liệu cho đến thời điểm đó. Điều này giúp trả lời những câu hỏi nghiệp vụ về lịch sử dữ liệu trong hệ thống. Ngoài ra, việc lưu trữ các events sẽ tránh trường hợp loại bỏ những thông tin có giá trị trong tương lai.
- Khắc phục sự cố ứng dụng khi chạy trên môi trường của khách hàng: Có thể dùng event store để khắc phục các sự cố trong hệ thống đang được

khách hàng sử dụng bằng cách sao chép event store này và tái diễn lại các events trong môi trường kiểm thử. Nếu biết được thời điểm sự cố xảy ra, có thể dễ dàng tái diễn các sự kiện xảy ra cho đến thời điểm đó.

- **Sửa lỗi:** Khi phát hiện có lỗi trong quá trình viết mã dẫn đến hệ thống tính toán giá trị sai, thay vì sửa lỗi và điều chỉnh dữ liệu đã được lưu trữ một cách thủ công và rủi ro, có thể sửa lỗi và tái diễn luồng sự kiện.
- **Tính linh động:** Một chuỗi các events có thể được chuyển hoá thành bất kỳ dạng biểu diễn cấu trúc nào, ví dụ như cơ sở dữ liệu SQL (Structured Query Language).

2.4.3 Sự kết hợp giữa Event Sourcing và CQRS

Mẫu CQRS và ES thường được kết hợp với nhau, hỗ trợ lẫn nhau. ES là một mô hình tuyệt vời hỗ trợ việc kết nối giữa bên đọc và bên ghi dữ liệu. Events có thể trở thành nền tảng cho việc đồng bộ hoá trạng thái của hệ thống giữa kho lưu trữ dữ liệu bên ghi và bên đọc. Dữ liệu bên đọc là dữ liệu không được chuẩn hoá để tối ưu hoá quá trình truy vấn dữ liệu, phục vụ cho việc hiển thị dữ liệu cho người dùng. Khi có sự thay đổi dữ liệu bên ghi, có thể sử dụng events lưu trong event store làm dữ liệu để thông báo đến bên đọc. Bên đọc dữ liệu sẽ dùng thông tin lưu trong các events để thực hiện chuẩn hoá dữ liệu cần thiết phục vụ cho quá trình truy vấn.

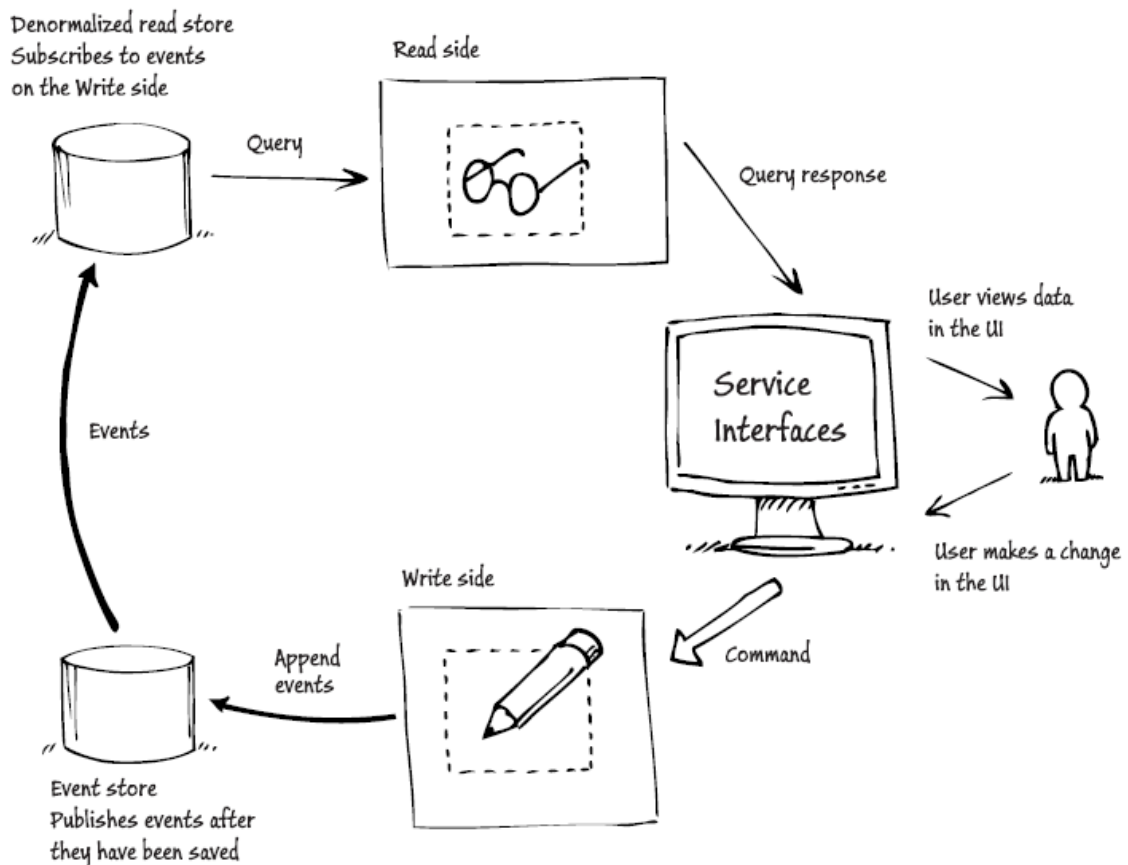


FIGURE 3
CQRS and event sourcing

Hình 2.4.5 Mô tả sự kết hợp giữa CQRS và ES [17]

Trong đó khi bên ghi nhận một yêu cầu thay đổi dữ liệu, nó thực hiện logic xử lý và đưa ra một event tương ứng, đẩy vào event store. Ngay sau đó, nó công bố event này. Bên đọc theo dõi các events bên ghi và xử lý ngay khi có event được truyền đến. Do vậy, dữ liệu giữa bên ghi và bên đọc được đồng bộ trong thời gian thực.

Thông thường, khi triển khai mẫu CQRS, các aggregate sẽ đưa ra events để thông báo thông tin đến các bên quan tâm như các aggregate khác, kho lưu trữ dữ liệu bên đọc,... Khi sử dụng ES, các events này sẽ được lưu vào event store. Mỗi event đại diện cho một sự thay đổi trạng thái của aggregate. Điều này cho phép sử dụng các events đó để lấy được trạng thái của aggregate bằng cách tái diễn chuỗi các events liên quan đến aggregate đó. Do đó, mỗi khi một instance của aggregate đưa ra một event, hai điều sau phải xảy ra: hệ thống phải lưu event vào event store,

hệ thống phải công bố event.

Dữ liệu bên đọc có thể được tái xây dựng tại mọi thời điểm bằng cách tái diễn các events từ event store tại bên ghi. Điều này là cần thiết nếu dữ liệu bên đọc trở nên bất đồng bộ với bên ghi hoặc dữ liệu bên đọc cần được điều chỉnh cấu trúc để hỗ trợ cho một câu truy vấn mới.

2.5 TỔNG KẾT

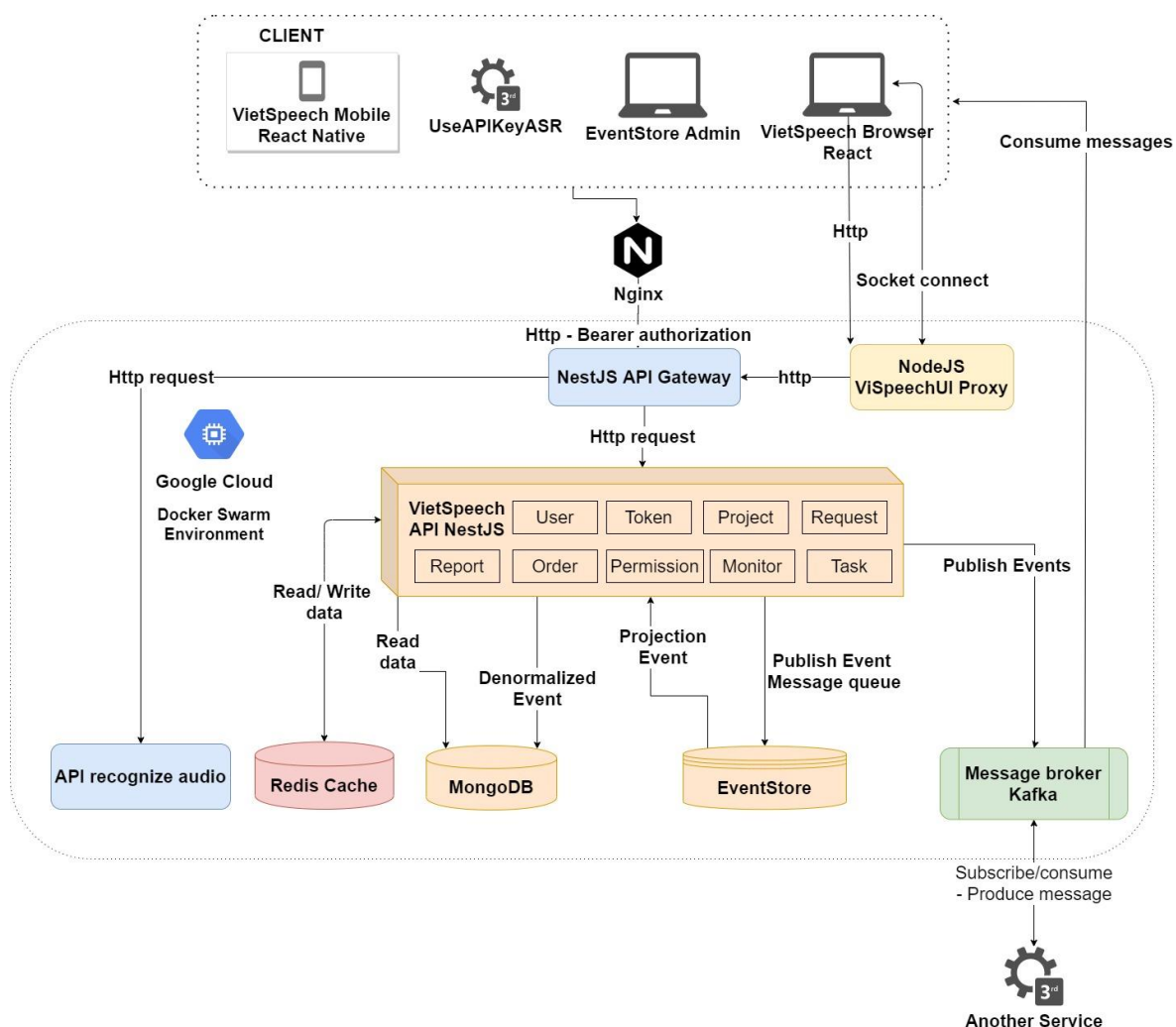
Trên thực tế, Microservices được xây dựng kết hợp giữa các mô hình đã nêu trên để bổ sung cho nhau, khắc phục những điểm yếu của các mô hình còn lại.

Thông qua chương 2, nhóm chúng em đã đề cập những lý thuyết nền tảng liên quan tới các thuật ngữ Microservices, DDD, CQRS và ES. Chương 3 sẽ làm rõ các giải pháp cụ thể cho từng thành phần trong hệ thống nhận dạng âm thanh tiếng Việt, hướng xây dựng máy chủ và cả ứng dụng trên nền tảng di động.

CHƯƠNG 3: GIẢI PHÁP ĐỀ TÀI

3.1 TỔNG QUAN GIẢI PHÁP VÀ KIẾN TRÚC MÔ HÌNH

Nhóm chúng em đề xuất kết hợp các mô hình đã nêu ở chương 2 để xây dựng hệ thống theo kiến trúc được minh họa cụ thể ở hình 3.1.1. Trong đó, nhóm xây dựng giao diện hệ thống trên nền tảng web theo mô hình Backend For Frontend (BFF) được giới thiệu tại mục 3.1.2.



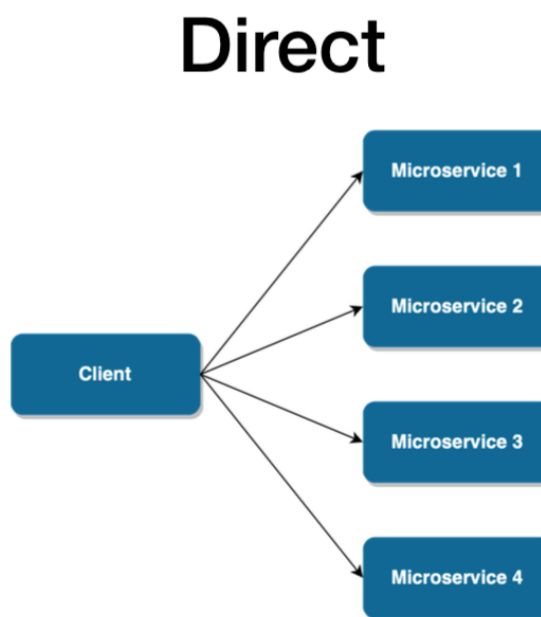
Hình 3.1.1 Tổng quan kiến trúc mô hình hệ thống cung cấp dịch vụ

Sau đây, nhóm sẽ trình bày các giải pháp giải quyết vấn đề cho từng thành phần trong hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt, hướng xây dựng hệ thống và ứng dụng hoá việc sử dụng dịch vụ trong thực tế trên nền tảng di động.

3.1.1 Mô hình API Gateway

3.1.1.1 Giới thiệu giải pháp

Khi nói về phương thức giao tiếp giữa các microservices và ứng dụng phía client, có ba mô hình phổ biến nhất đó là Direct Pattern, API Gateway và Backend For Frontend (BFF). Trước khi đến với API Gateway, nhóm chúng em trình bày một vài ý tưởng của Direct Pattern. Direct Pattern là cách thiết lập cơ bản nhất cho một kiến trúc được xây dựng dựa trên microservices. Cách hoạt động của Direct Pattern được mô tả thông qua hình 3.1.2. Trong đó, mỗi microservice sẽ có một public endpoint (điểm truy cập mở) cho phép ứng dụng phía client hay một microservice khác có thể truy cập để giao tiếp.



Hình 3.1.2 Hình ảnh minh họa cho Direct Pattern [20]

Mô hình này rất dễ cài đặt và phù hợp đối với các ứng dụng tương đối nhỏ. Tuy nhiên, sẽ gặp khá nhiều thách thức khi ứng dụng phát triển về quy mô và trở nên ngày càng phức tạp. Những thách thức đó là:

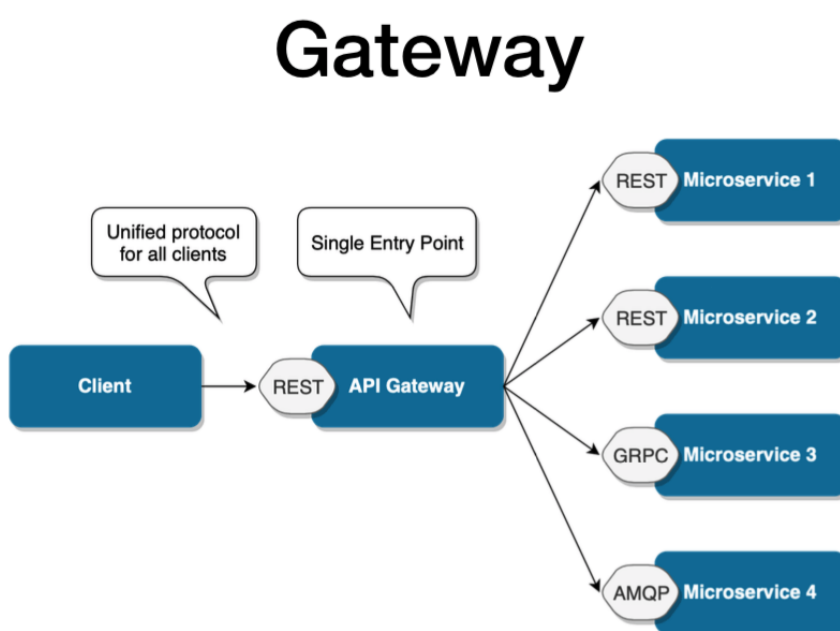
- Vấn đề về hiệu năng: ngay cả một trang truy cập trong ứng dụng cũng có

thể thực hiện nhiều lần gọi đến các microservices khác nhau, gây nên các vấn đề về độ trễ và hiệu suất.

- Khả năng mở rộng: vì ứng dụng phía client “liên kết” trực tiếp đến các microservices nên khi có bất kỳ sự thay đổi nào đối với các microservices đều có khả năng khiến ứng dụng bị lỗi. Điều này khiến việc bảo trì trở nên khó khăn hơn nhiều.
- Vấn đề bảo mật: Đối với phương thức truy cập trực tiếp đến public endpoints của các microservices, việc bảo mật cho hệ thống khá khó khăn.
- Vấn đề về độ phức tạp: việc bảo mật và thực hiện các công việc yêu cầu sự liên kết giữa các microservices cần được cài đặt cho mỗi microservice trong hệ thống. Điều này làm tăng sự phức tạp của hệ thống.

Vì kiến trúc microservice thường được áp dụng cho những ứng dụng phức tạp, API Gateway là một trong những cách tiếp cận phù hợp hơn.

3.1.1.2 Chi tiết giải pháp



Hình 3.1.3 Hình ảnh minh họa cho mô hình API Gateway [20]

Trong hình 3.1.3, API Gateway có thể được xem như điểm đầu vào duy nhất tới hệ thống microservices. API Gateway sẽ nhận các truy cập từ phía client, xác thực và điều hướng chúng đến các API cụ thể trên các microservices. Bằng cách ẩn đi các public endpoints của microservices, API Gateway có thể giải quyết các vấn đề mà Direct Pattern gặp phải. Ngoài ra, API Gateway sẽ đảm nhận các vai trò như bảo mật API, tính toán hiệu suất, phân tích số lượng các truy cập cũng như tình trạng của hệ thống.

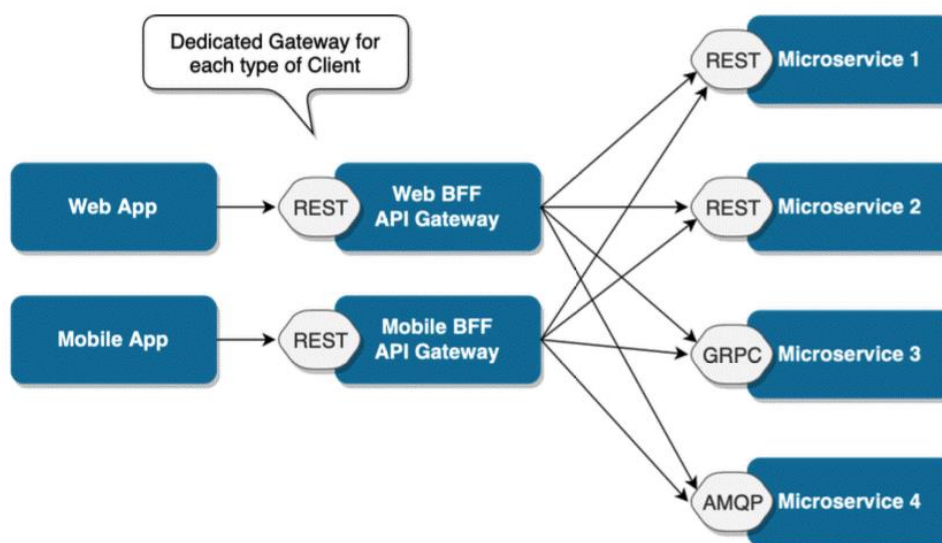
Tuy nhiên, API Gateway vẫn gặp phải vấn đề về khả năng mở rộng khi có nhiều ứng dụng phía client. Vì hệ thống của nhóm chúng em được xây dựng theo hướng có nhiều ứng dụng client, Backend For Frontend sẽ là hướng tiếp cận phù hợp nhất.

3.1.2 Mô hình Backend For Frontend (BFF)

3.1.2.1 Tổng quan về giải pháp

Với tính chất của hệ thống cung cấp dịch vụ, cùng với khả năng mở rộng cao hướng tới mô hình Software as a Service (SAAS – có thể hiểu là một phần mềm tham gia vào mạng lưới các phần mềm khác tương tự như một dịch vụ), có thể tương tác với các phần mềm khác không phân biệt nền tảng như ứng dụng trên nền tảng di động, trình duyệt web, các dịch vụ bên thứ ba, BFF là một mô hình phù hợp để thiết lập giao tiếp giữa hệ thống với các ứng dụng phía client. Hình 3.1.4 mô tả cách hoạt động của BFF.

Backend For Frontend



Hình 3.1.4 Minh họa mô hình Backend for frontend [20]

Trong đó một phần mềm xây dựng theo mô hình này có thể cung cấp dịch vụ cho nhiều loại ứng dụng trên các nền tảng khác nhau cùng lúc mà không bị phụ thuộc vào hạ tầng xây dựng.

3.1.2.2 Chi tiết giải pháp

BFF cơ bản là một biến thể của API Gateway, nó cũng cung cấp một lớp bổ sung giữa microservice và ứng dụng phía client. Nhưng thay vì cung cấp một điểm truy cập duy nhất, nó cung cấp nhiều điểm truy cập, mỗi điểm truy cập phù hợp cho một loại ứng dụng phía client.

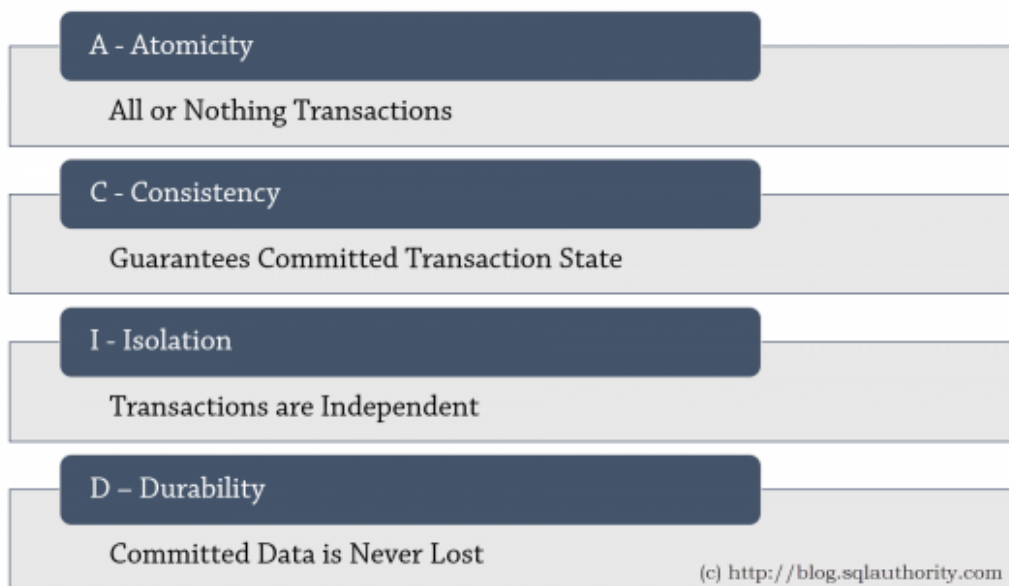
Mô hình này được áp dụng cho các ứng dụng phức tạp. Trong trường hợp dịch vụ web được xây dựng theo kiểu nguyên khối chỉ để phục vụ lưu trữ cho nền tảng web, sau đó một thời gian, hệ thống muốn mở rộng phục vụ cho ứng dụng trên nền tảng di động nhằm mở rộng kinh doanh, nếu không được xây dựng theo mô hình BFF, hệ thống phải chấp nhận xây dựng thêm một dịch vụ web tương tự để phục vụ cho nền tảng di động. Đây là một việc rất tốn kém và bất khả thi đối với bất kỳ

doanh nghiệp nào.

3.1.3 Đảm bảo toàn vẹn dữ liệu

3.1.3.1 Tổng quan về giải pháp

Mô hình microservices được xây dựng kết hợp cùng mẫu CQRS và Event Sourcing sẽ có hai nơi để lưu trữ dữ liệu đọc và ghi. Khi xây dựng mô hình này, điều quan trọng phải đảm bảo là tính đồng nhất dữ liệu giữa hai cơ sở dữ liệu và hạ tầng truyền dữ liệu phải đưa các thay đổi đến các nơi cần thiết. Vì vậy, việc xây dựng hệ thống theo mô hình microservices phải giải quyết vấn đề về đồng bộ dữ liệu mà trong mô hình monolithic đã giải quyết được bằng nguyên tắc ACID trong cơ sở dữ liệu. Nguyên tắc này bao gồm các tiêu chí atomicity, consistency, isolation, durability.



Hình 3.1.5 Hình mô tả nguyên tắc ACID [21]

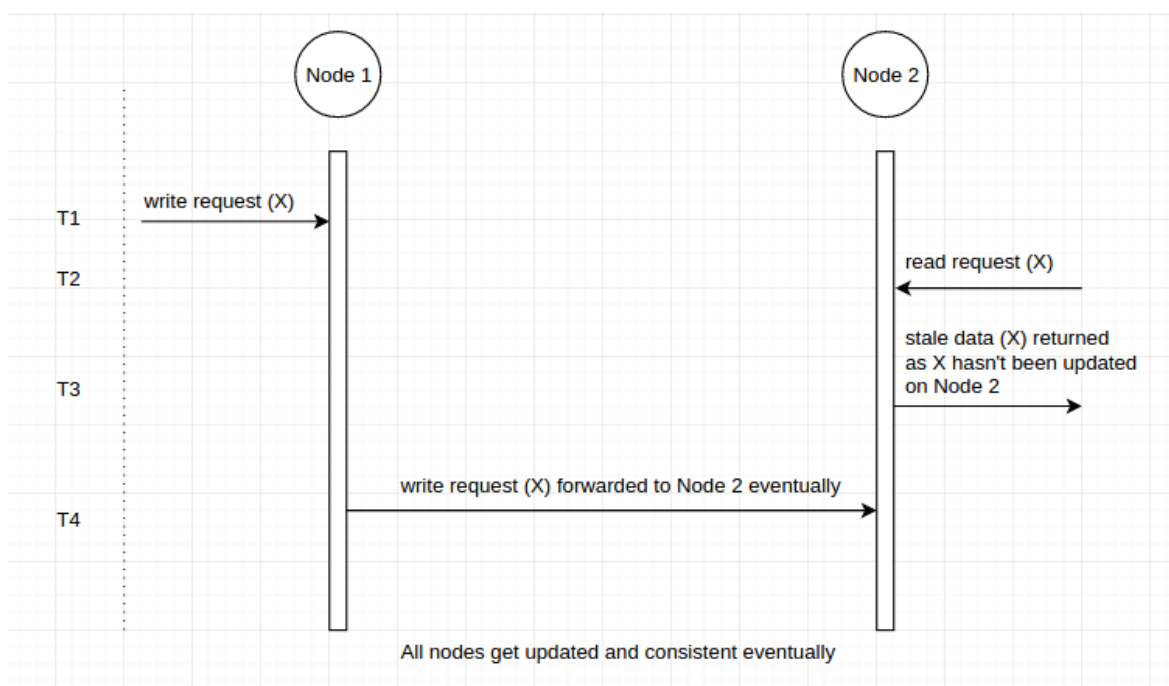
Hệ thống của nhóm chúng em tập trung vào việc quản lý phân phối các dữ liệu và sự kiện (distributed transactions and event sourcing). Nhóm chúng em đề xuất mô hình Eventual consistency làm giải pháp chính cho các vấn đề trên, truyền dữ liệu bằng message queue và message broker, phương pháp versioning system để

đảm bảo tính đúng đắn của dữ liệu trong mô hình và tái hiện lại sự kiện đã xảy ra nhằm khôi phục trạng thái hệ thống (replay events).

3.1.3.2 Chi tiết giải pháp

3.1.3.2.1 *Eventual consistency*

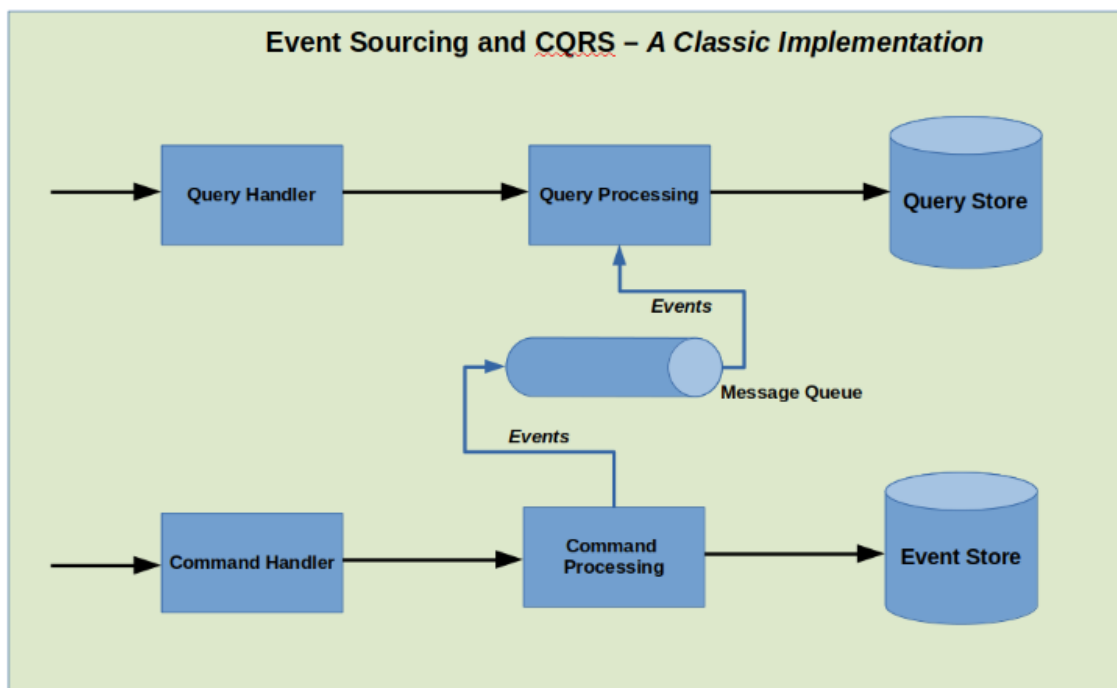
Khi sử dụng nhiều bản sao (replica) cho một cơ sở dữ liệu, khi có một lệnh yêu cầu viết thêm dữ liệu mới đến một bản sao, thì phải đảm bảo được các bản sao còn lại cũng nhận được lệnh này để dữ liệu được đồng bộ.



Hình 3.1.6 Mô tả phương pháp Eventual consistency [22]

Hình 3.1.6 mô tả quá trình đồng bộ dữ liệu giữa các bản sao. Việc đồng bộ này sẽ tốn thời gian (có thể rất nhỏ), nhưng nếu có nhiều bản sao, thời gian này sẽ tăng lên. Trong thời gian đồng bộ dữ liệu, nếu có một yêu cầu đọc dữ liệu đến một bản sao chưa đồng bộ xong thì yêu cầu này sẽ nhận được kết quả (dữ liệu) cũ hơn.

3.1.3.2.2 *Message queue*

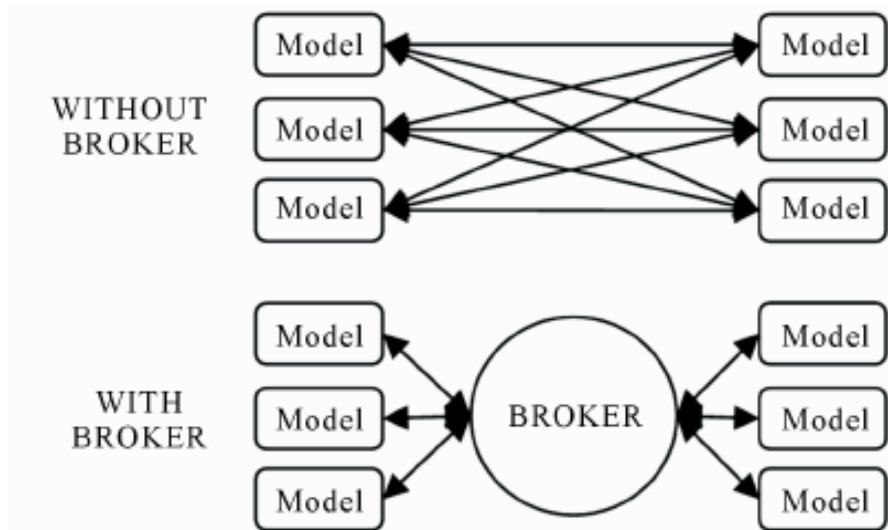


Hình 3.1.7 Hình minh họa message queue truyền dữ liệu [23]

Dựa vào cách sự kiện được lưu trữ, có thể thấy nếu truyền các sự kiện không đúng thứ tự, sự sai lệch dữ liệu sẽ xảy ra. Ví dụ một sự kiện đặt hàng OrderCreated nếu đến trước PaymentCreated sẽ xảy ra tình trạng có khả năng đơn hàng được đặt mà không được thanh toán. Vì vậy, message queue sẽ giữ đúng thứ tự của các sự kiện, đảm bảo dữ liệu chỉ truyền đi một lần duy nhất.

3.1.3.2.3 Message broker

Message broker giải quyết vấn đề giao tiếp giữa các dịch vụ với nhau, đứng ra làm trung gian để giao tiếp với các dịch vụ. Đảm bảo dữ liệu thay đổi được truyền đi đúng thứ tự, toàn vẹn dữ liệu, truyền lại dữ liệu trong thời gian máy chủ không có khả năng tiếp nhận với khả năng truyền hàng trăm ngàn, hàng triệu lượt trên giây.



Hình 3.1.8 Hình minh họa giao tiếp giữa dịch vụ khi không có và có message broker [24]

3.1.3.2.4 Versioning system

Lưu trữ phiên bản cho từng sự kiện giúp kiểm tra việc tạo lập các sự kiện, giúp cho một sự kiện tại một thời điểm chỉ xảy ra một lần duy nhất, kể cả trong thời gian tái diễn sự kiện (replay events). Đồng thời phiên bản của phần mềm cũng được đánh dấu nhằm đảm bảo việc phục vụ các ứng dụng khác không bị ảnh hưởng khi quá trình thay đổi, nâng cấp của hệ thống diễn ra liên tục.

Ví dụ một hệ thống đang ở phiên bản v1, sau đó nâng cấp lên phiên bản v2, hệ thống sẽ lưu trữ những sự kiện của phiên bản cũ và bắt đầu lưu trữ những sự kiện mới. Lúc này, hệ thống phải phân biệt được sự kiện của phiên bản nào để cho biết chính xác trạng thái của hệ thống. Ngoài ra, các ứng dụng sử dụng phiên bản cũ không cần phải thay đổi ngay khi hệ thống được nâng cấp, mà có thể tiếp tục sử dụng phiên bản cũ.

3.1.3.2.5 Replay events

Hệ thống có thể tái diễn lại các sự kiện trong môi trường thực tế (production) nhằm tìm kiếm, phân tích các lỗi để tìm ra nguyên nhân thực sự của các lỗi này. Ngoài ra, cũng có thể phục vụ các mục đích như đồng bộ các trạng thái của hệ

thống trong thời gian không thể tiếp nhận các yêu cầu dịch vụ (down time).

3.1.4 Cải thiện hiệu năng hệ thống

3.1.4.1 Tổng quan về giải pháp

Dựa vào các khảo sát thực tế của các hệ thống khác, cũng như các nhu cầu của khách hàng, hệ thống phải đáp ứng một lượng lớn truy cập cùng lúc, vì vậy phải đáp ứng đạt chất lượng về mặt hiệu năng. Nhóm chúng em xin đề xuất một số giải pháp như sử dụng bộ nhớ đệm (Cache memory), sử dụng cơ sở dữ liệu không quan hệ (NoSQL), thực thi theo hướng sự kiện và hướng thời gian (Event-driven scheduling và Clock-driven scheduling), bất đồng bộ đọc ghi. Các giải pháp này sẽ giúp hệ thống tối ưu hiệu năng nhờ vào công nghệ và giải pháp mới nhất, phù hợp với mô hình và đáp ứng được các yêu cầu nhất định như tính toàn vẹn dữ liệu, độ sẵn sàng cao,... Ngoài ra, khi thực hiện phương pháp bất đồng bộ được trình bày ở phần dưới sẽ sinh ra các vấn đề về chuỗi các sự kiện, sẽ không biết được khi nào các yêu cầu được gửi đến được hoàn tất. Vì vậy, nhóm đề xuất có thêm một phương pháp khác đó là Sagas.

3.1.4.2 Chi tiết giải pháp

3.1.4.2.1 Lưu trữ tạm thời bằng bộ nhớ đệm

Bằng cách sử dụng RAM (Random Access Memory) trên máy chủ hoặc một dịch vụ khác tối ưu để lưu trữ tạm thời ví dụ như Redis (Remote Dictionary Server) để làm bộ nhớ đệm, tốc độ đọc ghi dữ liệu tăng nhanh.

3.1.4.2.2 Sử dụng cơ sở dữ liệu phi quan hệ

Trong mô hình này, việc đưa các logic nghiệp vụ đến mà nguồn dựa theo domain sẽ tăng hiệu suất cực lớn, đồng thời việc ghi dữ liệu trong hệ thống được ưu tiên nhiều hơn. Ngoài ra, trong quá trình vận hành hệ thống sẽ tương tác với các hệ thống khác và nâng cấp theo thời gian, việc sử dụng NoSQL (cơ sở dữ liệu phi quan hệ) sẽ đảm bảo cho quá trình nâng cấp thuận lợi, các thay đổi trên sự kiện không ảnh hưởng nghiêm trọng.

3.1.4.2.3 *Kế hoạch hướng sự kiện và hướng kế hoạch*

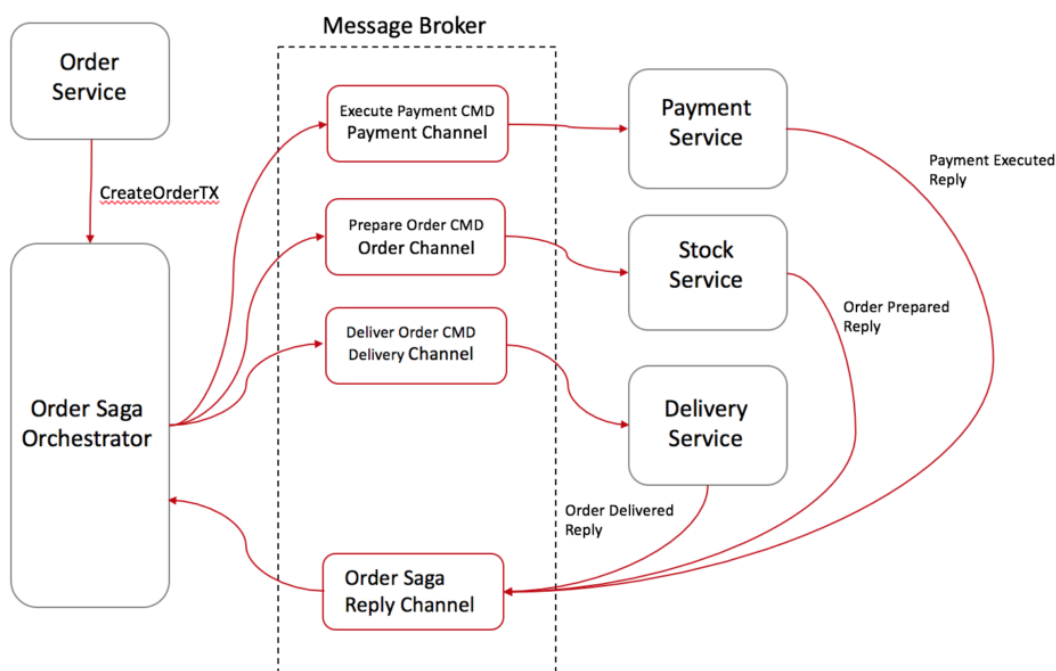
Đối với các nhiệm vụ được thực thi trong hệ thống như báo cáo lịch sử, quá trình sử dụng, các giao dịch trong ngày, tháng năm, hoặc nhiều loại thực thi khác, các kết quả thực thi thay đổi trong thời gian ngắn có khả năng sẽ giống nhau, ngoài cách được ghi trong bộ nhớ tạm theo phương pháp ở trên, các thực thi này sẽ được thực thi theo các thay đổi của sự kiện trong một số trường hợp, ngoài ra sẽ được thực thi chỉ trong một thời gian cố định, có thể thời gian mà luồng truy cập thấp nhất trong ngày. Ví dụ như các báo cáo sử dụng trong tháng sẽ được tính vào 00:00:00 pm vào ngày cuối cùng của tháng, nhờ vậy sẽ giảm được khá nhiều gánh nặng cho hiệu năng của hệ thống.

3.1.4.2.4 *Bất đồng bộ đọc ghi*

Như đã nêu ở mô hình Eventual consistency tại mục 3.1.3.2.1, các máy chủ bản sao (replica) chưa được đồng bộ sẽ trả về các kết quả cũ. Một phần dựa vào mô hình này, nhóm chúng em đề xuất cách đọc ghi dữ liệu bất đồng bộ để tăng tối đa hiệu năng cho hệ thống, dựa vào các Service Bus hiện nay để truyền dữ liệu trở lại sau khi thực hiện xong yêu cầu.

3.1.4.2.5 *Sagas*

Nhiều các sự kiện xảy ra theo một quy trình của nghiệp vụ, nhưng nó cũng có thể được sắp xếp thành một chuỗi các sự kiện theo kịch bản dựng trước trên một hoặc nhiều các dịch vụ. Sagas giữ vai trò là một chuỗi các sự kiện theo kịch bản cho trước, một sự kiện xảy ra sẽ kéo theo các sự kiện khác. Nếu có bất kỳ sự kiện nào gặp lỗi trong quá trình thực thi, sagas sẽ xử lý tương tự để điều chỉnh cho phù hợp.



Hình 3.1.9 Minh họa Sagas cho quá trình đặt hàng

Trong hình 3.1.9, quá trình đặt đơn hàng là một chuỗi các sự kiện xảy ra từ lúc khách hàng yêu cầu đặt hàng. Domain của Order sẽ gửi một Command yêu cầu đặt hàng *CreateOrderTX*, sau đó sagas (Order Saga) sẽ điều phối một chuỗi các sự kiện xảy ra theo trình tự nhất định: thanh toán, xác thực thanh toán, chuẩn bị đơn hàng, vận chuyển đơn hàng.

3.2 THIẾT KẾ HỆ THỐNG

3.2.1 Thiết kế giao diện hệ thống

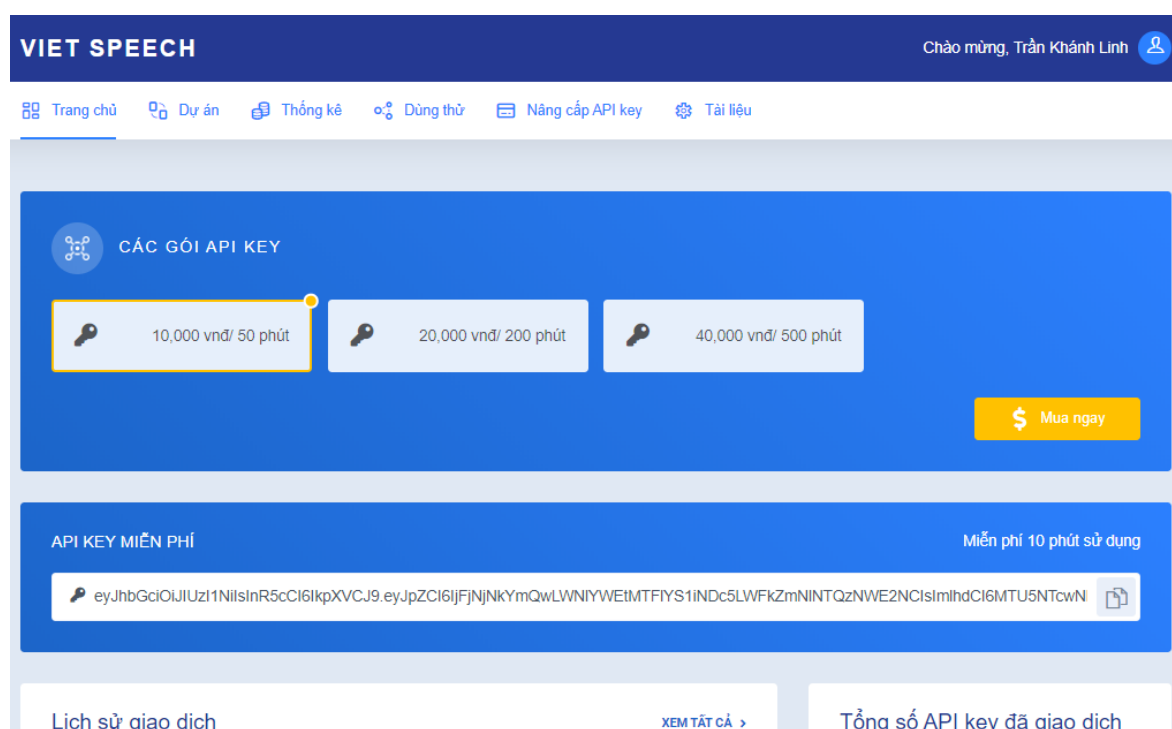
Giao diện hệ thống đóng góp một phần quan trọng quyết định lượng người dùng sử dụng dịch vụ mà hệ thống cung cấp. Vì vậy, nhóm đã tập trung thiết kế giao diện sao cho trải nghiệm của người dùng là tốt nhất. Phần sau đây sẽ liệt kê một số giao diện chính của hệ thống.

3.2.1.1 Giao diện trang chủ của hệ thống

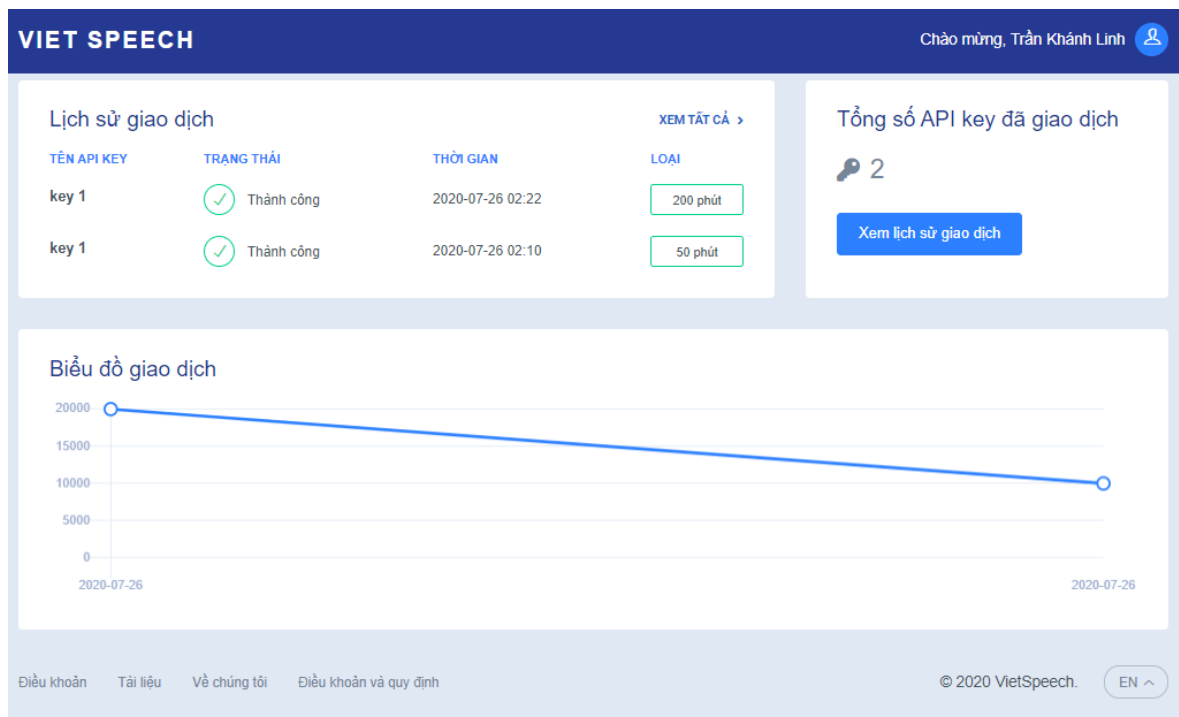
Trang chủ là nơi người dùng được điều hướng đến sau khi đăng nhập thành công. Tại đây, người dùng có thể lấy được API key miễn phí tại mục API key miễn

phí trong hình 3.2.1 hay lấy API key mất phí thông qua việc thực hiện thanh toán chi phí sử dụng dịch vụ dựa trên các gói dịch vụ được liệt kê tại mục Các gói API key trong hình 3.2.1, phục vụ cho mục đích chính là sử dụng dịch vụ nhận dạng âm thanh tiếng Việt hệ thống cung cấp.

Ngoài ra, người dùng có thể có cái nhìn tổng quan về các thông tin như các lần giao dịch gần nhất tại mục Lịch sử giao dịch, biểu đồ thống kê các lần giao dịch tại mục Biểu đồ giao dịch và tổng số API key đã giao dịch tại mục Tổng số API key đã giao dịch (hình 3.2.2).



Hình 3.2.1 Giao diện trang chủ (phần 1)



Hình 3.2.2 Giao diện trang chủ (phần 2)


3.2.1.2 Giao diện trang dùng thử dịch vụ

VIET SPEECH Chào mừng, Trần Khánh Linh

[Trang chủ](#) [Dự án](#) [Thống kê](#) [Dùng thử](#) [Nâng cấp API key](#) [Tài liệu](#)

Dùng thử

* Dự án: * API key:



Nhấn hoặc kéo thả tập tin âm thanh vào khu vực này để tải
Chỉ nhận tập tin âm thanh có định dạng đuôi .wav

[▶ Ghi âm](#)
[|| Dừng ghi âm](#)

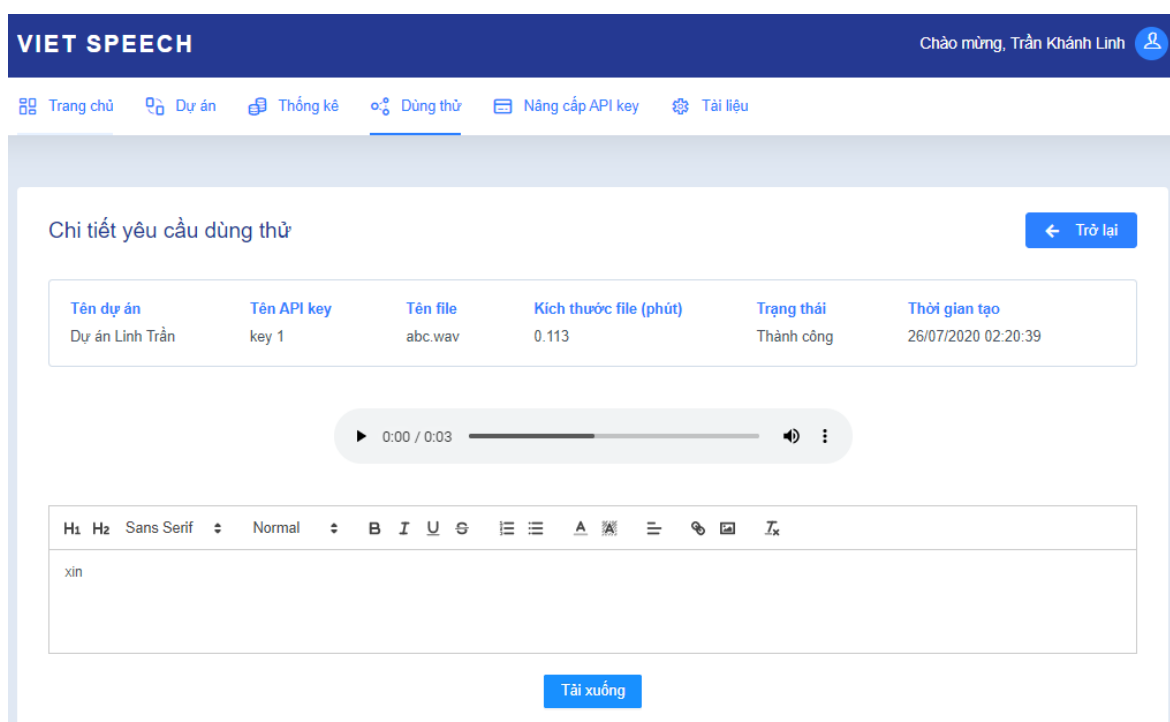
TÊN DỰ ÁN | TÊN API KEY | TÊN FILE | KÍCH THƯỚC FILE (PHÚT) | TRẠNG THÁI | THỜI GIAN TẠO

Hình 3.2.3 Giao diện trang dùng thử dịch vụ

Đây là trang nơi người dùng có thể dùng thử dịch vụ hệ thống cung cấp với các API key. Người dùng chọn một dự án, một dự án chứa nhiều API key, người dùng sẽ chọn một API key trong số đó và cung cấp một tập tin âm thanh bằng hai cách:

- Tải tập tin âm thanh bằng cách nhấn hoặc kéo thả vào khu vực Nhấn hoặc kéo thả tập tin âm thanh vào khu vực này để tải trong hình 3.2.3.
- Ghi âm giọng nói của mình bằng cách ấn vào nút Ghi âm trong hình 3.2.3. Sau khi hoàn tất ghi âm, người dùng ấn vào nút Dừng ghi âm.

Sau khi hoàn tất, hệ thống sẽ xử lý yêu cầu và đưa người dùng đến trang Chi tiết yêu cầu dùng thử như hình 3.2.4.



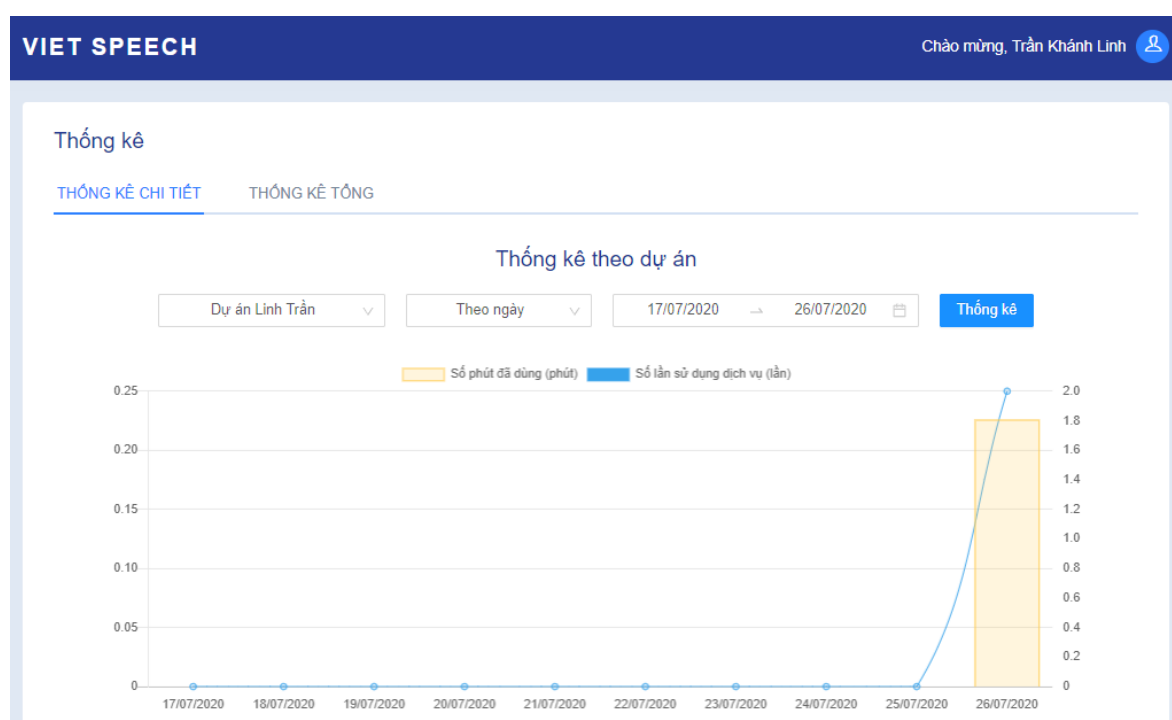
Hình 3.2.4 Giao diện trang kết quả khi sử dụng dịch vụ

Trang Chi tiết yêu cầu dùng thử thể hiện kết quả hệ thống trả về khi người dùng sử dụng dịch vụ, bao gồm thông tin dự án, API key, tên tập tin, kích thước tập tin, trạng thái yêu cầu, thời gian tạo yêu cầu. Kết quả quan trọng nhất là hệ thống cung cấp tập tin âm thanh người dùng đã chọn và văn bản được dịch tương ứng thể hiện trong khung chỉnh sửa nội dung văn bản trong hình 3.2.4. Hệ thống cho phép người

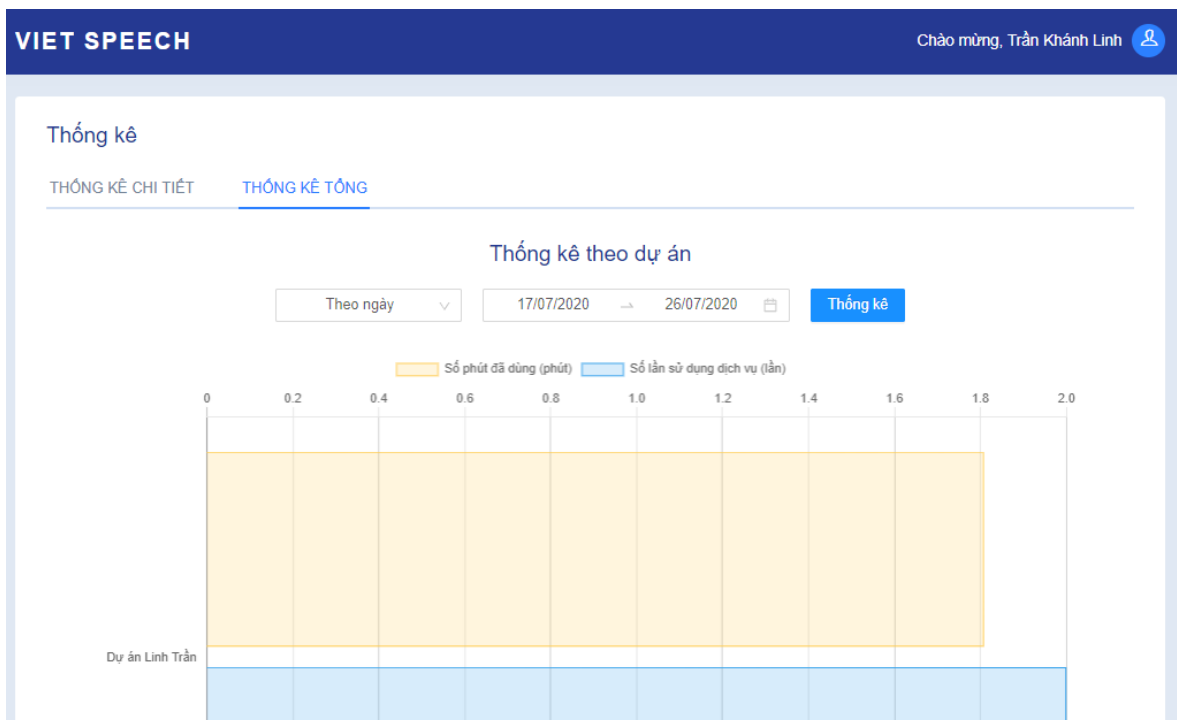
dùng tải xuống tập tin âm thanh và đoạn văn bản được dịch sau khi chỉnh sửa (nếu có).

3.2.1.3 Giao diện trang thống kê

Trang thống kê là nơi người dùng có thể xem lại các báo cáo về việc sử dụng dịch vụ. Có hai loại thống kê, thống kê chi tiết (hình 3.2.5) và thống kê tổng (hình 3.2.6), cung cấp cho người dùng một cách trực quan, đầy đủ các báo cáo về việc sử dụng dịch vụ của mình theo ngày, tuần, quý, tháng, năm. Thống kê chi tiết bao gồm thống kê theo một dự án, theo một API key, theo một gói dịch vụ. Thống kê tổng bao gồm thống kê theo các dự án, theo các API key, theo các gói dịch vụ.



Hình 3.2.5 Giao diện trang thống kê, phần thống kê chi tiết



Hình 3.2.6 Giao diện trang thống kê, phần thống kê tổng

3.2.1.4 Giao diện trang nâng cấp API key

Trang nâng cấp API key cho phép người dùng gia hạn số phút hợp lệ của một API key bất kỳ theo các gói API key mà hệ thống cung cấp. Trang có giao diện như hình 3.2.7. Người dùng sẽ lựa chọn dự án chứa API key muốn nâng cấp, tên API key cần nâng cấp, gói API key và cung cấp thông tin thẻ tín dụng để tiến hành nâng cấp.

VIET SPEECH

Chào mừng, Trần Khánh Linh

Trang chủ

Dự án

Thống kê

Dùng thử

Nâng cấp API key

Tài liệu

Nâng cấp API key

Dự án

Tên API key

Gói API key hiện tại

Dự án Linh Trần

key 1

200 phút

Nâng cấp lên gói

40,000 VNĐ/ 500 phút

Thông tin thẻ

Card number

MM / YY CVC

☐

Tôi đồng ý với điều khoản giao dịch mua bán key của VietSpeech.

Nâng cấp ↑

Hình 3.2.7 Giao diện trang nâng cấp API key

3.2.2 Thiết kế và giải pháp lưu trữ dữ liệu

Đối với hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt, lịch sử truy cập dịch vụ của người dùng là quan trọng nhất. Nhóm thực hiện việc lưu trữ dữ liệu này như sau. Dữ liệu được tổ chức như bảng 3.3.1.

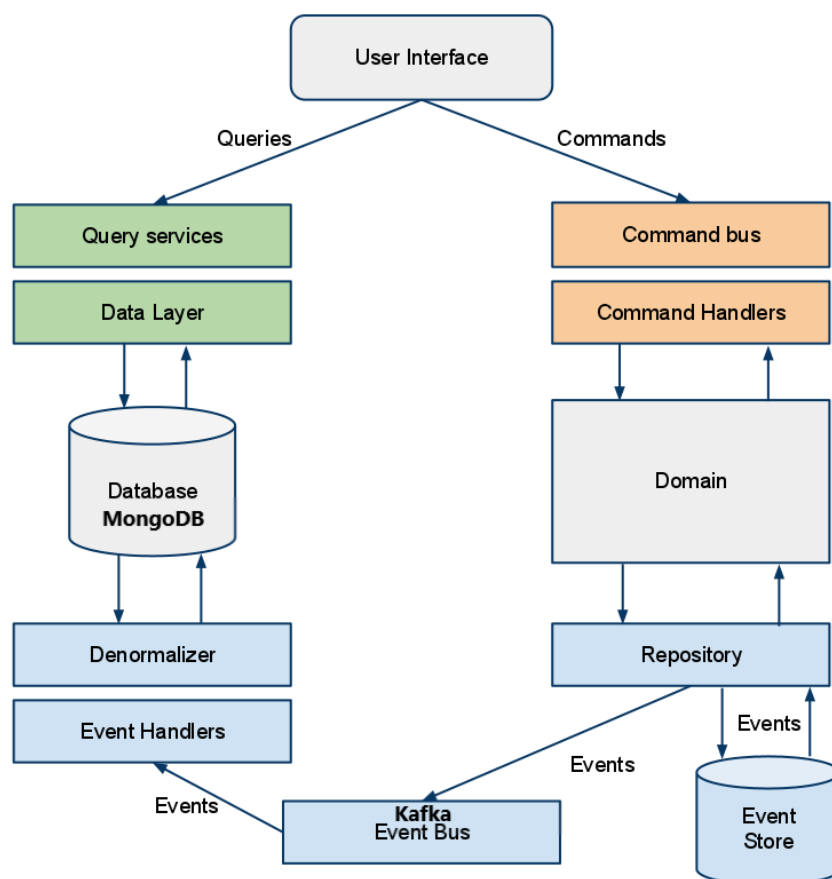
STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	tokenId	string	Định danh của API key dùng để truy cập dịch vụ.
2	projectId	string	Định danh dự án chứa API key dùng để truy cập dịch vụ.
3	userId	string	Định danh người dùng sử dụng dịch vụ.
4	fileName	string	Tên tập tin âm thanh.
5	encoding	string	Quy tắc mã hoá ký tự của tập tin.

6	size	string	Kích thước tập tin tính theo đơn vị byte.
7	duration	number	Kích thước tập tin tính theo đơn vị phút.
8	mimeType	string	Tiêu chuẩn Internet về định dạng cho tập tin âm thanh, đa số có giá trị là audio/wav.
9	status	string	Trạng thái yêu cầu truy cập dịch vụ, có thể bằng một trong ba giá trị: PENDING (đang chờ xử lý), SUCCESS (thành công), FAILED (thất bại).
10	audioFileUrl	string	Đường dẫn truy cập tập tin âm thanh.
11	transcriptFileUrl	string	Đường dẫn truy cập tập tin văn bản đã được dịch.

Bảng 3.2.1 Tổ chức các thuộc tính để lưu trữ dữ liệu truy cập dịch vụ của người dùng.

3.2.3 Thiết kế kiến trúc hệ thống

Trong hệ thống cung cấp dịch vụ, nhu cầu ghi dữ liệu thường lớn hơn rất nhiều so với nhu cầu truy xuất dữ liệu. Do đó, yêu cầu của hệ thống về hiệu suất đọc ghi rất khác biệt, đồng thời một hệ thống cung cấp dịch vụ phải đảm bảo tính đúng đắn của các yêu cầu nghiệp vụ phức tạp. Nhận thấy được sự phù hợp về mặt lợi ích mà mẫu CQRS mang lại được đề cập tại mục 2.3.4 ở chương 2, nhóm đề xuất thiết kế kiến trúc hệ thống áp dụng mẫu CQRS kết hợp Event Sourcing.



Hình 3.2.8 Mô hình mẫu CQRS kết hợp Event Sourcing áp dụng trong hệ thống [21]

Trong hình 3.2.8 nhóm sinh viên áp dụng mô hình CQRS để phân luồng đọc ghi, các event được lưu trữ trong Event Store, và trạng thái hiện tại của hệ thống được lưu trữ trong cơ sở dữ liệu MongoDB. Ngoài ra, event được chuyển từ luồng đọc sang luồng ghi thông qua một đường truyền sự kiện gọi là Event Bus.

3.3 GIẢI PHÁP XÂY DỰNG CHỨC NĂNG HỆ THỐNG

3.3.1 Giải pháp cho chức năng đăng ký/ đăng nhập

3.3.1.1 Chức năng đăng ký

Hệ thống cung cấp một API cho phép người dùng đăng ký tài khoản với các thông tin cơ bản như tên đăng nhập (username), email, tên (firstName), họ (lastName), mật khẩu (password) thông qua phương thức POST. Người dùng khi

đăng ký với hệ thống mặc định có vai trò tên USER. Sau khi đăng nhập và xác thực email, vai trò sẽ được đổi thành MANAGER_USER, cho phép người dùng sử dụng các chức năng của hệ thống. Bên cạnh đó, khi đăng ký thành công, hệ thống sẽ cấp một API key miễn phí cho phép người dùng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt. Ngoài ra, người dùng cũng có thể đăng ký tài khoản bằng cách đăng nhập với tài khoản facebook hoặc google.

3.3.1.2 Chức năng đăng nhập

Hệ thống sử dụng Token-based Authentication để xác thực người dùng và lưu giữ thông tin đăng nhập. Đây là một cơ chế đăng nhập dựa trên việc tạo ra token (một chuỗi ký tự mang thông tin xác định người dùng) ở phía server và lưu tại phía client, cho phép phía client không cần cung cấp thông tin đăng nhập của người dùng mỗi khi gọi đến APIs của hệ thống sau khi đăng nhập, mà chỉ cần cung cấp token này. Loại token được hệ thống chúng em sử dụng là JWT (JSON Web Token) (một tiêu chuẩn mở - RFC 7519 - định nghĩa cách thức truyền tin an toàn giữa các bên dưới dạng đối tượng JSON).

3.3.2 Giải pháp thanh toán chi phí sử dụng dịch vụ âm thanh tiếng Việt

Hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt cung cấp dịch vụ chuyển đổi một tập tin âm thanh tiếng Việt thành nội dung văn bản. Hệ thống hỗ trợ hai hình thức sử dụng dịch vụ, miễn phí và mất phí. Đối với hình thức mất phí, hiện nay, các hệ thống cung cấp dịch vụ tương tự trên thị trường như Google Cloud Speech-to-text hay FPT.AI Speech to Text đều áp dụng mô hình thanh toán trước, sử dụng sau. Có thể thấy mô hình này mang lại lợi ích và sự thoải mái cho cả bên cung cấp dịch vụ và bên sử dụng dịch vụ.

Bên cung cấp dịch vụ không phải xử lý trường hợp người dùng sử dụng dịch vụ nhưng không thanh toán cũng như bên sử dụng dịch vụ không phải khó chịu vì bị nhắc nhở thanh toán chi phí sử dụng dịch vụ mỗi khi đến hạn.

Nhóm chúng em đề xuất áp dụng hình thức thanh toán trước, sử dụng sau này khi cài đặt chức năng thanh toán chi phí sử dụng dịch vụ cho hệ thống. Cụ thể, hệ

thống sẽ hỗ trợ một số gói dịch vụ để lựa chọn khi người dùng có nhu cầu sử dụng hình thức mất phí. Hệ thống sẽ liên kết với dịch vụ bên thứ ba, hỗ trợ thanh toán trực tuyến thông qua thẻ tín dụng. người dùng lựa chọn gói dịch vụ phù hợp và nhập thông tin cần thiết và thực hiện giao dịch. Hệ thống kiểm tra trạng thái giao dịch với dịch vụ bên thứ ba, nếu thành công, hệ thống tạo một API key có số phút tương ứng với gói dịch vụ đã chọn cho người dùng. Hệ thống cho phép người dùng sử dụng API key này để sử dụng dịch vụ cho đến khi số phút bằng 0.

3.3.3 Giải pháp cho chức năng mời tham gia dự án

Dự án giúp người dùng có thể quản lý được các API key theo từng mục đích sử dụng nhất định. Ngoài ra, dự án cho phép người dùng chia sẻ các API key với những người dùng khác thông qua việc gửi email yêu cầu chấp nhận lời mời. Nếu chấp nhận lời mời tham gia dự án, người dùng có thể sử dụng các API key có trong dự án đã tham gia, ngược lại, nếu từ chối lời mời tham gia dự án, người dùng không thể thấy bất kỳ thông tin gì liên quan đến dự án này.

3.3.4 Giải pháp cung cấp API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt

Cung cấp API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt là một chức năng rất quan trọng của hệ thống. Cụ thể hơn, hệ thống có nhiệm vụ cung cấp dịch vụ chuyển đổi một tập tin âm thanh tiếng Việt thành nội dung văn bản. Để thực hiện chức năng này, nhóm đề xuất giải pháp như sau:

API mà hệ thống cung cấp yêu cầu người dùng một API key và một tập tin âm thanh tiếng Việt. Dựa vào API key này, hệ thống xác định được định danh (ID) người dùng, số phút hợp lệ và số phút đã sử dụng của khoá. Nếu hợp lệ, hệ thống gọi API chuyển đổi âm thanh thành văn bản và trả về dữ liệu cho người dùng. Đồng thời, hệ thống tăng số phút đã sử dụng của API key lên một lượng bằng với số phút của tập tin âm thanh và lưu lại yêu cầu sử dụng dịch vụ của người dùng vào kho lưu trữ dữ liệu.

3.3.5 Giải pháp cho chức năng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt

Việc cho phép người dùng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt của hệ thống giúp người dùng có được trải nghiệm cụ thể, qua đó đánh giá và quyết định có tiếp tục sử dụng dịch vụ của hệ thống hay không. Để thực hiện chức năng này, nhóm đề xuất giải pháp như sau:

Người dùng chọn lựa một API key và tải lên một tập tin âm thanh. Hệ thống sử dụng dịch vụ bên thứ ba để lưu trữ tập tin âm thanh này. Sau đó, hệ thống gọi API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt. Nếu API trả về dữ liệu thành công, hệ thống sử dụng dịch vụ bên thứ ba lưu trữ tập tin văn bản đã dịch và hiển thị dữ liệu văn bản cho người dùng.

Việc lưu trữ tập tin âm thanh và tập tin văn bản tại bên thứ ba giúp tiết kiệm bộ nhớ và dung lượng cho hệ thống, giảm độ phức tạp trong việc lưu trữ xuống cơ sở dữ liệu. Đồng thời, việc lưu trữ lại các tập tin giúp hệ thống có thể cung cấp tập tin trong trường hợp người dùng có nhu cầu xem lại hoặc tải về.

3.3.6 Giải pháp nâng cấp API key

Khi muốn gia hạn số phút sử dụng dịch vụ cho API key, người dùng có thể sử dụng chức năng nâng cấp API key của hệ thống. Người dùng sẽ chọn lựa API key cần nâng cấp. Hệ thống sau đó sẽ gợi ý các gói dịch vụ phù hợp và người dùng cần chọn một trong các gói dịch vụ này. Hệ thống sẽ liên kết với dịch vụ bên thứ ba, hỗ trợ thanh toán trực tuyến thông qua thẻ tín dụng. Người dùng nhập thông tin thanh toán cần thiết và thực hiện nâng cấp. Hệ thống kiểm tra trạng thái giao dịch với dịch vụ bên thứ ba, nếu thành công, hệ thống cập nhật loại API key và số phút hợp lệ cho API key theo yêu cầu.

3.3.7 Giải pháp cung cấp các báo cáo sử dụng dịch vụ của người dùng

Giải pháp này thể hiện ở hai quá trình lưu trữ và truy vấn. Thông thường, đối với quá trình lưu trữ, mỗi khi người dùng truy cập dịch vụ của hệ thống, yêu cầu

truy cập này sẽ được hệ thống lưu lại vào kho lưu trữ dữ liệu. Đối với quá trình truy vấn, khi cung cấp báo cáo sử dụng dịch vụ của người dùng theo từng loại, ví dụ như báo cáo theo ngày, tuần, quý, tháng, năm, hệ thống sẽ truy vấn kho lưu trữ dữ liệu những dữ liệu liên quan, tiến hành lọc và tổng hợp dữ liệu phù hợp với từng loại báo cáo.

Đối với hệ thống cung cấp dịch vụ, trong một giây, có thể có hàng trăm, hàng ngàn yêu cầu truy cập, tương ứng với hàng trăm, hàng ngàn dòng trong kho lưu trữ dữ liệu. Việc cung cấp báo cáo theo cách này có thể mất rất nhiều thời gian, làm giảm hiệu suất của hệ thống để lấy được dữ liệu phù hợp. Nhóm chúng em đề xuất giải pháp đối với quá trình lưu trữ dữ liệu như sau. Hệ thống sẽ tự động tổng hợp các yêu cầu truy cập dịch vụ của người dùng trong kho lưu trữ dữ liệu định kỳ theo ngày, tuần, quý, tháng, năm, tương ứng với các loại báo cáo. Nhờ vậy, thời gian truy vấn dữ liệu cho việc cung cấp các báo cáo của hệ thống giảm đáng kể, phản hồi trên giao diện người dùng nhờ đó cũng nhanh hơn.

3.3.8 Giải pháp xây dựng ứng dụng di động

Để thể hiện rõ mục đích của việc xây dựng hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt, nhóm quyết định xây dựng một ứng dụng di động, ứng dụng hoá việc sử dụng dịch vụ mà hệ thống cung cấp, giúp học phát âm tiếng Việt trên nền tảng Android thông qua các câu hỏi.

Ứng dụng di động có chức năng chính là rèn luyện phát âm tiếng Việt của người dùng thông qua việc đưa ra các câu hỏi và yêu cầu người dùng trả lời bằng cách ghi âm tiếng nói. Ứng dụng sẽ sử dụng dịch vụ nhận dạng âm thanh, chuyển đổi âm thanh được ghi âm sang dữ liệu văn bản tương ứng. Sau đó ứng dụng sẽ so sánh văn bản này với kết quả đúng của câu hỏi và hiển thị kết quả đúng hoặc sai cho người dùng. Thông qua kết quả này, người dùng sẽ điều chỉnh phát âm tiếng Việt của mình sao cho phù hợp.

3.4 TỔNG KẾT

Thông qua chương 3, nhóm đã làm rõ một số giải pháp tổng quan và cách thiết kế hệ thống để thực hiện xây dựng và phát triển hệ thống. Đồng thời, nhóm chúng em cũng trình bày sơ lược lý do vì sao chọn lựa các giải pháp này.

Ở chương kế tiếp, nhóm sẽ trình bày cách áp dụng những giải pháp đã nêu trong chương 3 để cài đặt và xây dựng hệ thống. Cụ thể hơn, nhóm sẽ trình bày các thư viện, công cụ phục vụ quá trình cài đặt giải pháp và những cách khắc phục khó khăn cụ thể nếu có cho các giải pháp.

CHƯƠNG 4: CÀI ĐẶT GIẢI PHÁP

4.1 TYPESCRIPT VÀ NESTJS

4.1.1 TypeScript

Mã nguồn xây dựng hệ thống máy chủ phát triển dựa vào TypeScript. TypeScript là một ngôn ngữ mã nguồn mở miễn phí đang được phát triển và bảo trì bởi Microsoft. Nó có thể được xem là phiên bản nâng cao của JavaScript vì được bổ sung các tùy chọn kiểu tĩnh và lớp trên cơ sở lập trình hướng đối tượng (Object Oriented Programming – OOP). TypeScript có thể được sử dụng để phát triển ứng dụng chạy ở phía khách (client-side) và phía chủ (server-side).

4.1.1.1 Bắt đầu và kết thúc với JavaScript

TypeScript bắt đầu từ cùng một cú pháp và ngữ nghĩa được hàng triệu lập trình viên ngôn ngữ JavaScript biết đến. TypeScript biên dịch thành mã JavaScript đơn giản và gọn gàng chạy trên mọi trình duyệt, trong Node.js hay bất kỳ chương trình thực thi mã JavaScript (JavaScript engine) nào hỗ trợ ECMAScript 3 (hoặc mới hơn).

4.1.1.2 Công cụ mạnh cho các ứng dụng lớn

TypeScript cho phép các lập trình viên ngôn ngữ JavaScript sử dụng các công cụ và kỹ thuật phát triển như kiểm tra tĩnh (static checking) hay tái cấu trúc mã khi phát triển ứng dụng JavaScript. Việc TypeScript sử dụng các kỹ thuật mới nhất, kết hợp các thư viện JavaScript phổ biến và phương pháp OOP giúp phát triển các dự án lớn một cách dễ dàng.

4.1.1.3 Phiên bản nâng cao của JavaScript

TypeScript hỗ trợ các tính năng mới nhất và đang phát triển của JavaScript, bao gồm các tính năng từ ECMAScript 2015 và các đề xuất như các hàm bất đồng bộ (async functions) giúp xây dựng các thành phần hoạt động hiệu quả.

4.1.2 NestJS

Nest hay NestJS là một nền tảng để xây dựng các ứng dụng phía chủ bằng Node.js một cách hiệu quả và dễ mở rộng. NestJS sử dụng ngôn ngữ TypeScript (tuy nhiên vẫn cho phép các lập trình viên sử dụng JavaScript thuần túy) kết hợp các tính chất của phương pháp OOP, lập trình hướng hàm (Functional Programming – FP) và lập trình hướng phản hồi hàm (Functional Reactive Programming - FRP).

Về bản chất, Nest sử dụng các nền tảng máy chủ HTTP mạnh như là Express (mặc định) và có thể tùy chọn cấu hình để sử dụng Fastify. Nest cung cấp một mức độ trừu tượng vượt trên các nền tảng Node.js phổ biến này (Express/ Fastify), nhưng cũng hỗ trợ giao diện lập trình ứng dụng (Application Programming Interface – API) trực tiếp cho lập trình viên. Điều này cho phép các lập trình viên có thể tự do sử dụng vô số các mô-đun của bên thứ ba.

Trong những năm gần đây, nhờ có Node.js, JavaScript đã trở thành ngôn ngữ chung cho cả ứng dụng front-end và back-end. Điều này đã tạo ra các dự án tuyệt vời như Angular, React và Vue, giúp cải thiện năng suất của lập trình và cho phép tạo ra các ứng dụng front-end nhanh, có thể kiểm thử và mở rộng. Tuy nhiên, trong khi có rất nhiều thư viện, công cụ tuyệt vời cho Node, không tồn tại thư viện nào giải quyết hiệu quả vấn đề chính là kiến trúc. Nest cung cấp một kiến trúc vượt trội cho ứng dụng, cho phép các lập trình viên và đội ngũ lập trình tạo ra các ứng dụng dễ kiểm thử, có thể mở rộng, kết nối lỏng lẻo và dễ bảo trì. Kiến trúc này được lấy cảm hứng từ Angular.

4.2 REACTJS

Đối với ứng dụng phía khách, nơi cho phép người dùng đăng ký tài khoản và quản lý API key phục vụ cho việc truy cập dịch vụ nhận dạng âm thanh tiếng Việt mà hệ thống cung cấp, nhóm xây dựng bằng React.

React hay ReactJS là một thư viện JavaScript mã nguồn mở để xây dựng giao diện người dùng, được phát triển bởi Facebook và ra mắt năm 2013.

4.2.1 Lập trình khai báo

React giúp tạo giao diện người dùng mang tính tương tác một cách nhanh chóng, thiết kế các góc nhìn đơn giản cho từng trạng thái của ứng dụng. Hơn nữa, khi dữ liệu thay đổi, React sẽ cập nhật và hiển thị đúng các thành phần một cách hiệu quả.

Khi giải quyết một vấn đề, thay vì mô tả rõ những việc cần làm, với React, chỉ cần khai báo những việc muốn làm. Điều này giúp cho những đoạn mã trở nên dễ hiểu và dễ tìm lỗi hơn.

4.2.2 Phát triển dựa trên thành phần (Component-Based)

React xây dựng các thành phần có thể quản lý các trạng thái của riêng chúng, sau đó kết hợp lại để tạo ra giao diện người dùng phức tạp.

Vì logic xử lý của thành phần được viết bằng JavaScript, có thể dễ dàng truyền dữ liệu giữa các thành phần và tránh thao tác với cây DOM (Document Object Model).

4.3 CÀI ĐẶT KIẾN TRÚC HỆ THỐNG

Nhóm chúng em bắt đầu cài đặt kiến trúc cơ bản của hệ thống áp dụng mẫu CQRS kết hợp Event Sourcing bằng cách tham khảo kiến trúc mẫu tại GitHub [21]. Kiến trúc này được xây dựng dựa trên nền tảng NestJS được giới thiệu tại mục 4.1.2, kết hợp áp dụng mẫu CQRS và ES một cách cơ bản nhất. Dựa vào kiến trúc này, nhóm xây dựng và phát triển chức năng hệ thống phù hợp với các yêu cầu của luận văn.

4.3.1 Cài đặt luồng ghi dữ liệu

Luồng ghi dữ liệu trong hệ thống được mô tả như nửa bên phải hình 3.3.7. Các yêu cầu của người dùng gửi đến server sử dụng phương thức HTTP như POST, PUT, DELETE được xem như là các Command. Các Command này được vận chuyển từ tầng services đến Command Handlers (nơi kiểm tra tính hợp lệ và xử lý các Command) thông qua phương thức *execute()* của CommandBus (cung cấp bởi thư viện @nestjs/cqrs). Mỗi Command sẽ có một Command Handler tương ứng xử

lý nó.

Theo mẫu CQRS, mỗi Command sẽ sinh ra một Event. Để tạo và lưu trữ Event tại kho lưu trữ dữ liệu phía ghi Event Store, sau khi kiểm tra tính hợp lệ của Command, hệ thống sử dụng Domain Model. Domain Model kế thừa lớp AggregateRoot (cung cấp bởi thư viện @nestjs/cqrs) sẽ gọi phương thức *apply()* với tham số là một Event tương ứng với Command. Tuy nhiên, phương thức này sẽ không gửi Event đi ngay lập tức vì không tồn tại mối liên hệ giữa Domain Model và lớp EventPublisher (cung cấp bởi thư viện @nestjs/cqrs). Để thiết lập mối liên hệ, hệ thống sử dụng phương thức *mergeObjectContext()* của EventPublisher bên trong Command Handler. Cuối cùng gọi phương thức *commit()* của lớp AggregateRoot để gửi Event đi ngay lập tức.

Sau khi Event được gửi đến Event Store, Event sẽ được lưu lại trong Event Store và tiếp tục được xử lý bởi Event Handler thông qua EventBus (cung cấp bởi thư viện @nestjs/cqrs). Tại Event Handler, Event sẽ được lưu trữ tại kho lưu trữ dữ liệu bên phía đọc là MongoDB dưới dạng dữ liệu có cấu trúc không chuẩn hoá phù hợp. Nếu việc lưu trữ diễn ra thành công, một Event đánh dấu sự thành công được gửi đến Event Store thông qua EventBus và tiếp tục được xử lý tại Event Handler tương ứng (nếu có) và ngược lại.

4.3.2 Cài đặt luồng đọc dữ liệu

Luồng đọc dữ liệu trong hệ thống được mô tả như nửa bên trái hình 3.3.7. Các yêu cầu của người dùng gửi đến server sử dụng phương thức HTTP GET được xem như là các Query. Các Query này được vận chuyển từ tầng services đến Query Handlers (nơi xử lý các Query) thông qua phương thức *execute()* của QueryBus (cung cấp bởi thư viện @nestjs/cqrs). Tại Query Handler, dữ liệu sẽ được truy vấn từ MongoDB và trả về cho người dùng.

4.4 CÀI ĐẶT CHỨC NĂNG HỆ THỐNG

4.4.1 Cài đặt chức năng đăng ký/ đăng nhập

4.4.1.1 Chức năng đăng ký

Hệ thống cung cấp API với endpoint <http://asr.vietspeech.com/v1/users> sử dụng phương thức HTTP POST cho phép người dùng thực hiện đăng ký tài khoản. Tài khoản người dùng khi mới đăng ký mặc định sẽ có vai trò là USER. Command đăng ký tài khoản *CreateUserCommand* được hệ thống xử lý theo luồng ghi dữ liệu đã đề cập tại mục 4.3.1. Theo đó, Event *UserCreatedEvent* được tạo ra và lưu tại Event Store, đồng thời một tài khoản với các thông tin đăng ký và vai trò USER được lưu tại MongoDB.

Sau khi tạo tài khoản thành công, hệ thống sử dụng Saga (cung cấp bởi thư viện @nestjs/cqrs) được đề cập tại mục 3.1.4.2.5 để thực hiện tạo một API key miễn phí cho tài khoản mới tạo. Saga sẽ lắng nghe Event *UserCreatedEventSuccess* (là một event đánh dấu việc lưu trữ tài khoản mới thành công tại MongoDB) và gửi Command *CreateFreeTokenCommand* đến Command Handler tương ứng để tiếp tục quá trình xử lý. Khi Event *FreeTokenCreatedSuccess* được tạo và gửi đến Event Handler, việc đăng ký tài khoản mới được xem là hoàn thành.

4.4.1.2 Chức năng đăng nhập

Hệ thống cung cấp API với endpoint <http://asr.vietspeech.com/login> sử dụng phương thức HTTP POST cho phép người dùng thực hiện đăng nhập. Quá trình xử lý chức năng đăng nhập diễn ra tại AuthModule của hệ thống.

Hệ thống sử dụng Passport.js (thành phần trung gian xác thực yêu cầu cho Node.js) để xác thực yêu cầu đăng nhập bằng tài khoản thường có tên đăng nhập, mật khẩu và tài khoản xã hội (google, facebook) thông qua việc cài đặt bốn thư viện @nestjs/passport, passport-local, passport-google-token, passport-facebook-token. Mỗi cách thức đăng nhập sẽ tương ứng với một lớp, mỗi lớp đều phải kế thừa PassportStrategy (cung cấp bởi @nestjs/passport). Những lớp cách thức đăng nhập này kiểm tra tính hợp lệ trong thông tin đăng nhập của người dùng thông qua phương thức *validate()* với tham số là *username*, *password* đối với tài khoản thường và *accessToken*, *refreshToken*, *profile* đối với tài khoản xã hội. Trong phương thức

validate(), thông tin đăng nhập được kiểm tra bằng cách gọi đến service AuthService của AuthModule.

- Đối với tài khoản thông thường, nếu AuthService trả về dữ liệu là null, việc đăng nhập thất bại và ngược lại.
- Đối với tài khoản xã hội, nếu AuthService trả về dữ liệu là null, đây là lần đầu tiên tài khoản xã hội này đăng nhập vào hệ thống, hệ thống sau đó sẽ gửi đi Command *CreateUserCommand* để thực hiện quá trình tạo tài khoản, cho đến khi Event *FreeTokenCreatedSuccess* được tạo và gửi đến Event Handler, việc đăng nhập mới được xem là thành công. Ngược lại, đăng nhập thành công.

4.4.2 Cài đặt chức năng thanh toán chi phí sử dụng dịch vụ nhận dạng âm thanh tiếng Việt

Chức năng thanh toán chi phí sử dụng dịch vụ nhận dạng âm thanh tiếng Việt giúp người dùng có được API key để sử dụng dịch vụ cung cấp bởi hệ thống. Để thực hiện chức năng này, nhóm sử dụng Stripe (một phần mềm tốt và phù hợp cho các hoạt động kinh doanh trực tuyến) để thực hiện phần thanh toán chi phí qua thẻ tín dụng và JWT để tạo API key cho người dùng.

Tại ứng dụng phía khách xây dựng bằng React, nhóm cài đặt hai gói thư viện hỗ trợ sử dụng Stripe là *@stripe/react-stripe-js* và *@stripe/stripe-js*. Đồng thời, cài đặt gói thư viện stripe tại phía server. Tại trang chủ ứng dụng, người dùng lựa chọn gói dịch vụ phù hợp nhu cầu sử dụng. Sau đó chọn dự án, đặt tên cho API key muốn tạo và nhập thông tin thẻ tín dụng để thực hiện giao dịch. Hệ thống sẽ xử lý quá trình giao dịch này như sau.

- Bước 1: Ứng dụng phía khách gửi yêu cầu tạo mã bí mật cho hệ thống cùng thông tin về giá gói dịch vụ được chọn.
- Bước 2: Hệ thống nhận yêu cầu từ ứng dụng, tạo mã bằng phương thức mà gói thư viện stripe cung cấp *stripe.paymentIntents.create()* với tham số là số

tiền phải trả, đơn vị tiền tệ. Sau đó, hệ thống trả về mã cho ứng dụng.

- Bước 3: Ứng dụng tạo phương thức thanh toán bằng cách gọi hàm *useStripe().createPaymentMethod()* cung cấp bởi thư viện *@stripe/react-stripe-js* với tham số là các thông tin liên quan (số thẻ tín dụng, tên người dùng, email người dùng). Sau đó, xác nhận thanh toán bằng cách gọi hàm *useStripe().confirmCardPayment()* với tham số truyền vào là mã bí mật nhận từ phía server và định danh phương thức thanh toán vừa tạo.
- Bước 4: Sau khi xác nhận thanh toán thành công, Stripe trả về một định danh chứa thông tin thanh toán. Ứng dụng tiến hành gửi về phía server các thông tin sau: loại gói dịch vụ, định danh người dùng, dự án, tên API key và định danh thông tin thanh toán.
- Bước 5: Phía server khi nhận được đầy đủ thông tin, đầu tiên sẽ xác minh người dùng đã thanh toán gói dịch vụ chưa thông qua thông tin thanh toán nhận được. Quá trình này được thực hiện bằng cách gọi hàm *stripe.paymentIntents.retrieve()* với tham số là định danh thông tin thanh toán. Nếu thanh toán thành công, hàm này sẽ trả về một đối tượng chứa thuộc tính *status* với giá trị *succeeded*.
- Bước 6: Sau khi xác nhận thanh toán thành công, hệ thống sẽ kiểm tra tính hợp lệ của các thông tin còn lại như định danh người dùng, dự án,... Cuối cùng, hệ thống tạo một đơn hàng chứa các thông tin cần thiết bằng cách gửi Command *CreateOrderCommand* đến Command Handler tương ứng. Theo luồng ghi dữ liệu đã đề cập tại mục 4.3.1, Event *OrderCreatedEvent* được tạo ra và lưu tại Event Store. Hệ thống sử dụng Saga lắng nghe Event *OrderCreatedSuccessEvent* (là một event đánh dấu việc lưu trữ đơn hàng thành công tại MongoDB) và gửi Command *CreateOrderedTokenCommand* đến Command Handler tương ứng để tạo và lưu trữ một API key sử dụng chuỗi mã hoá JWT chứa thông tin người sử hữu, thông tin đơn hàng vừa tạo và trả về ứng dụng phía khách, hiển thị API key cho người dùng. Khi Event

OrderedTokenCreatedSuccessEvent được tạo và lưu trữ tại Event Store, quá trình giao dịch kết thúc.

Hệ thống quản lý những API key này dựa trên hai thuộc tính: số phút cho phép - minutes và số phút đã dùng - usedMinutes. Minutes có giá trị phụ thuộc vào gói dịch vụ người dùng chọn khi thực hiện thanh toán chi phí. Mỗi khi người dùng dùng API key sử dụng dịch vụ, hệ thống sẽ cộng dồn số phút tương ứng vào thuộc tính usedMinutes của khoá cho đến khi giá trị hai thuộc tính bằng nhau. Lúc này, người dùng không thể sử dụng khoá được nữa vì đã dùng hết số phút cho phép. Tuy nhiên, thay vì giao dịch mua API key mới, hệ thống hỗ trợ chức năng nâng cấp API key.

4.4.3 Cài đặt chức năng mời tham gia dự án

Đối với mỗi dự án của mình, người dùng có thể mời những người dùng khác cùng tham gia. Hệ thống sẽ cung cấp danh sách các tên tài khoản người dùng khác, người dùng chọn một tài khoản và một dự án để thực hiện chức năng này. Hệ thống sau đó sẽ gửi mail đến người dùng được mời kèm theo đường dẫn đến trang phản hồi lời mời. Người dùng được mời sẽ phản hồi lời mời bằng cách chọn chấp nhận hoặc từ chối tại trang phản hồi lời mời.

Để thực hiện gửi mail, ban đầu nhóm chúng em sử dụng thư viện @sendgrid/mail của SendGrid. SendGrid là một trong những dịch vụ gửi email giao dịch phổ biến. Nhóm tạo tài khoản sử dụng gói Miễn phí (có thể gửi 40.000 mail trong tháng đầu sử dụng và 100 mail một ngày cho thời gian sau đó), lấy API key và cấu hình gửi mail dễ dàng. Tuy nhiên, sau một vài tháng đầu sử dụng, tài khoản đăng ký với SendGrid của nhóm bị khoá để xem xét không rõ lý do, nhóm phải gửi mail liên hệ hỗ trợ nhưng mất khá nhiều thời gian (thời gian lâu nhất là gần một tháng) để nhận phản hồi từ SendGrid. Nhận thấy mặt hạn chế này, nhóm quyết định chuyển sang sử dụng Nodemailer. Nodemailer là một mô-đun cho phép các ứng dụng Node.js gửi email một cách dễ dàng. Nhóm chúng em cài đặt thư viện nodemailer. Để sử dụng Gmail gửi email khi hệ thống chạy trên môi trường thực tế, nhóm phải cấu hình sử dụng phương thức xác thực OAuth2. OAuth2 cho phép ứng

dùng lưu trữ và sử dụng mã xác thực để xác thực người dùng thay vì thông tin đăng nhập người dùng như tên tài khoản và mật khẩu.

Nội dung email gửi đến người dùng bao gồm tên tài khoản người mời, tên dự án, đường dẫn đến trang phản hồi lời mời, khoảng thời gian hợp lệ để truy cập đường dẫn. Hệ thống sử dụng chuỗi mã hoá JWT chứa các thông tin như định danh người mời, người được mời, định danh dự án, quyền khi tham gia dự án và thời gian hết hạn của chuỗi JWT đính kèm vào đường dẫn đến trang phản hồi lời mời để phía client có thể lấy các thông tin liên quan đến lời mời.

Để quản lý lời mời và xác thực thao tác với dự án đã tham gia của người dùng, hệ thống xử lý các tác vụ liên quan trong thư mục tên permissions trong mã nguồn hệ thống. Dữ liệu được lưu trữ có các trường thông tin như sau:

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	string	Định danh của lời mời tham gia dự án.
2	permissions	string[]	Quyền được cấp cho người dùng khi tham gia dự án, giá trị mặc định là CSR_USER.
3	assigneeId	string	Định danh người dùng được mời.
4	assignerId	string	Định danh người mời.
5	projectId	string	Định danh dự án.
6	status	string	Trạng thái lời mời. Trạng thái có thể có giá trị bằng PENDING (đang xử lý), ACCEPTED (đã chấp nhận), REJECTED (bị từ chối), INVALID (không còn hợp lệ).

Bảng 4.4.1 Tổ chức các thuộc tính để lưu trữ dữ liệu lời mời tham gia dự án

4.4.4 Cài đặt API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt

Chức năng cung cấp API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt được xem là chức năng cốt lõi của hệ thống. Nó cho phép người dùng sử dụng dịch vụ nhận dạng âm thanh tiếng Việt mà hệ thống cung cấp.

Hệ thống cung cấp API với endpoint <http://asr.vietspeech.com/v1/speech> sử dụng phương thức HTTP POST cho phép người dùng truy cập dịch vụ nhận dạng âm thanh tiếng Việt. Khi nhận được yêu cầu, đầu tiên hệ thống kiểm tra sự hợp lệ của các thông tin. Chi tiết các lỗi trả về nếu thông tin không hợp lệ được mô tả trong bảng 4.4.2.

STT	Mã trạng thái HTTP	Thông điệp đến người dùng
1	400 Bad Request	File is required/ Không tìm thấy tập tin âm thanh.
2	403 Forbidden	Invalid API key/ API key không hợp lệ.
3		Not enough API key's minutes to request ASR service/ Số phút hiện tại của API key không đủ để truy cập dịch vụ nhận dạng âm thanh tiếng Việt.

Bảng 4.4.2 Mô tả chi tiết các lỗi khi gọi API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt của hệ thống VietSpeech

Sau khi kiểm tra thông tin, nếu hợp lệ, hệ thống tạo yêu cầu với trạng thái PENDING (chờ xử lý) lưu vào kho lưu trữ dữ liệu và thực hiện gọi API nhận dạng âm thanh tiếng Việt. Nếu API trả về dữ liệu văn bản thành công, hệ thống trả về dữ liệu cho người dùng đồng thời tiến hành cập nhật trạng thái của yêu cầu vừa tạo bằng cách gửi Command *CallAsrRequestCommand*. Theo luồng ghi dữ liệu đã đề cập tại mục 4.3.1, Event *AsrRequestCalledEvent* được tạo ra và lưu tại Event Store. Hệ thống sử dụng Saga lắng nghe Event *AsrRequestCalledEvent* (trường hợp người dùng gọi API cung cấp dịch vụ không phải tại ứng dụng client VietSpeech), Event *requestTranscriptFileUrlUpdatedSuccess* (trường hợp người dùng gọi API cung cấp dịch vụ khi dùng thử dịch vụ tại ứng dụng client VietSpeech) và gửi Command

UpdateTokenCommand đến Command Handler tương ứng để tiếp tục quá trình xử lý nếu trạng thái của yêu cầu là SUCCESS (thành công). Dữ liệu về các yêu cầu của người dùng được tổ chức và lưu trữ tại MongoDB như mô tả tại bảng 3.3.1.

4.4.5 Cài đặt chức năng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt

Cũng như các hệ thống cung cấp dịch vụ nhận dạng âm thanh khác, hệ thống VietSpeech cho phép người dùng sử dụng thử dịch vụ nhận dạng âm thanh tiếng Việt. Người dùng có thể sử dụng API key miễn phí (được cấp khi đăng ký tài khoản) hoặc các API key thuộc dự án của mình/ dự án đã tham gia để truy cập dịch vụ.

4.4.5.1 Cung cấp tập tin âm thanh

Hệ thống hỗ trợ hai hình thức cung cấp âm thanh giọng nói: tải lên tập tin âm thanh và ghi âm giọng nói.

Đối với hình thức tải lên tập tin âm thanh, nhóm sử dụng Dragger cung cấp bởi thư viện antd. Dragger cho phép các cấu hình cơ bản như chấp nhận loại tập tin nào, cho phép tải lên nhiều tập tin một lúc hay không và hỗ trợ tùy chỉnh hàm xử lý tải lên tập tin.

Đối với hình thức ghi âm giọng nói, nhóm sử dụng thư viện react-mic kết hợp msr. React-Mic cho phép ghi âm giọng nói và hiển thị dưới dạng sóng âm thanh. Sau khi ghi âm, react-mic trả về dữ liệu dưới dạng đối tượng Blob (Binary Large Object). Tuy nhiên, khi gửi đối tượng này về server để xử lý chuyển sang dạng dữ liệu văn bản thì không nhận được kết quả như mong đợi. Do đó, nhóm chúng em đã kết hợp sử dụng MediaStreamRecorder.js. MediaStreamRecorder.js cũng trả về đối tượng Blob sau khi ghi âm và phía server có thể xử lý đối tượng này như mong đợi. Như vậy, nhóm chúng em sử dụng React-Mic để hiển thị trực quan giao diện ghi âm đến người dùng và sử dụng MediaStreamRecorder.js để nhận dữ liệu ghi âm và gửi về server để xử lý.

4.4.5.2 Lưu trữ tập tin âm thanh và tập tin chứa dữ liệu văn bản

Để lưu trữ tập tin âm thanh và tập tin chứa dữ liệu văn bản, nhóm chúng em lựa chọn sử dụng Firebase Storage. Đây là một dịch vụ được xây dựng trên nền tảng Google Cloud Platform, cho phép lưu trữ và quản lý các nội dung như ảnh, video, dữ liệu dạng tập tin.

Đầu tiên, nhóm cài đặt gói thư viện hỗ trợ sử dụng Firebase Storage là firebase tại ứng dụng phía client. Tại trang dùng thử dịch vụ, khi người dùng hoàn tất chọn tập tin âm thanh để tải lên, tập tin sẽ được tải lên Firebase Storage thông qua việc gọi phương thức `firebase.storage().ref(path).put(file)`. Trong đó, `path` là nơi tập tin được lưu trữ trong Firebase Storage (`path` bao gồm cả tên tập tin), `file` là tập tin âm thanh người dùng cung cấp. Sau khi tải lên thành công, đường dẫn đến tập tin đã tải được trả về thông qua phương thức `firebase.storage().ref(path).child(fileName).getDownloadURL().then(fileUrl=>{ })`. Trong đó, `path` là nơi lưu trữ tập tin trong Firebase Storage, `fileName` là tên tập tin, `fileUrl` là đường dẫn đến tập tin đã tải.

Sau khi có được đường dẫn đến tập tin âm thanh đã tải lên Firebase Storage, ứng dụng phía client sẽ gọi API cung cấp dịch vụ nhận dạng âm thanh tiếng Việt kèm API key, tập tin âm thanh và đường dẫn đến tập tin âm thanh. Nếu API trả về dữ liệu văn bản thành công, ứng dụng phía khách tạo tập tin với dữ liệu văn bản nhận được và thực hiện lưu trữ ở Firebase Storage. Quá trình lưu trữ và lấy được đường dẫn đến tập tin đã tải hoàn toàn giống với quá trình đã đề cập đối với tập tin âm thanh. Sau khi có được đường dẫn đến tập tin văn bản đã dịch, ứng dụng gọi API yêu cầu hệ thống cập nhật đường dẫn đến tập tin văn bản cho yêu cầu sử dụng dịch vụ của người dùng.

Sau khi yêu cầu dùng thử dịch vụ được hệ thống xử lý thành công, người dùng sẽ được chuyển đến trang chi tiết sử dụng dịch vụ, tại đây, ứng dụng phía client sẽ sử dụng hai đường dẫn được lưu cho tập tin âm thanh và tập tin văn bản để hiển thị kết quả cho người dùng.

4.4.6 Cài đặt chức năng cấp API key

Người dùng có thể nâng cấp API key khi số phút hợp lệ của API key không đủ để truy cập dịch vụ. Quá trình nâng cấp API key cũng khá giống với quá trình thanh toán chi phí sử dụng dịch vụ, hệ thống cũng sử dụng Stripe để thực hiện thanh toán chi phí qua thẻ tín dụng. Tuy nhiên, nếu thanh toán chi phí sử dụng dịch vụ thành công, hệ thống tạo API key cho người dùng thì nâng cấp API key thành công, hệ thống cập nhật số phút hợp lệ của API key.

Tại trang chủ ứng dụng, người dùng lựa chọn API key cần nâng cấp và gói dịch vụ phù hợp nhu cầu sử dụng. Sau đó nhập thông tin thẻ tín dụng và thực hiện giao dịch nâng cấp API key. Quá trình xử lý giao dịch tương tự với quá trình đã đề cập tại mục 4.4.2.

- Bước 1 – 5: tương tự với bước 1 – 5 tại mục 4.4.2.
- Bước 6: Sau khi xác nhận thanh toán thành công, hệ thống sẽ kiểm tra tính hợp lệ của các thông tin còn lại như định danh người dùng, API key,... Cuối cùng, hệ thống tạo một đơn hàng chứa các thông tin cần thiết bằng cách gửi Command *CreateOrderToUpgradeCommand* đến Command Handler tương ứng. Theo luồng ghi dữ liệu đã đề cập tại mục 4.3.1, Event *OrderToUpgradeCreatedEvent* được tạo ra và lưu tại Event Store. Hệ thống sử dụng Saga lắng nghe Event *OrderToUpgradeCreatedSuccessEvent* (là một event đánh dấu việc lưu trữ đơn hàng thành công tại MongoDB) và gửi Command *UpgradeTokenCommand* đến Command Handler tương ứng để cập nhật số phút hợp lệ của API key theo gói dịch vụ người dùng đã chọn. Khi Event *TokenUpgradedSuccessEvent* được tạo và lưu trữ tại Event Store, quá trình giao dịch kết thúc.

4.4.7 Cài đặt chức năng cung cấp các báo cáo sử dụng dịch vụ của người dùng

Chức năng cung cấp các báo cáo sử dụng dịch vụ của người dùng cho phép

người dùng quản lý việc truy cập dịch vụ của hệ thống, đồng thời giúp ban quản trị hệ thống quản lý được việc truy cập dịch vụ của người dùng. Nhóm chúng em sử dụng kỹ thuật tạo cron job để định kỳ tổng hợp dữ liệu, cho phép hệ thống cung cấp các loại báo cáo một cách nhanh chóng và chính xác. Cron job là các lệnh thực thi định kỳ một hành động nào đó vào một thời điểm nhất định.

Để áp dụng kỹ thuật này, nhóm chúng em cài đặt gói thư viện `@nestjs/schedule` tại phía server. Sau đó, nhóm tiến hành viết các phương thức cài đặt logic xử lý việc lọc và tổng hợp dữ liệu theo từng loại báo cáo (báo cáo theo ngày, tuần, quý, tháng, năm). Cuối cùng, thêm decorator `@Cron(cronTime)` với `cronTime` là một trong các tham số được liệt kê dưới đây. Trong đó, `Cron` và `CronExpression` đều do gói thư viện `@nestjs/schedule` cung cấp.

- `CronExpression.EVERY_DAY_AT_MIDNIGHT`: thực hiện mỗi ngày vào lúc nửa đêm.
- `CronExpression.EVERY_WEEK`: thực hiện mỗi tuần.
- `CronExpression.EVERY_1ST_DAY_OF_MONTH_AT_MIDNIGHT`: thực hiện vào ngày đầu của tháng vào lúc nửa đêm.
- `CronExpression.EVERY_QUARTER`: thực hiện mỗi quý.
- `CronExpression.EVERY_YEAR`: thực hiện mỗi năm.

Bằng cách này, hệ thống định kỳ chạy các phương thức xử lý việc tổng hợp dữ liệu cho báo cáo, cho phép hệ thống cung cấp báo cáo cho người dùng, người quản trị một cách nhanh chóng và hiệu quả.

4.5 TỔNG KẾT

Trong chương 4, nhóm đã trình bày cách thức cài đặt và triển khai cho một số chức năng hệ thống cung cấp. Nội dung chi tiết cho một số phần cài đặt và kiến thức liên quan được nhóm trình bày ở phần phụ lục. Cuối cùng, nhóm sẽ tổng kết quá trình thực hiện luận văn cùng những kiến thức, sản phẩm thu được tại chương

5.

CHƯƠNG 5: TỔNG KẾT, ĐÁNH GIÁ

5.1 KIẾN THỨC THU ĐƯỢC

Trong suốt quá trình thực hiện khoá luận, nhóm đã học tập, lĩnh hội và áp dụng được nhiều kiến thức mới.

- Tiếp cận và áp dụng phương pháp quản lý công việc bằng mô hình Kanban.
- Thử sức ở nhiều vai trò khi xây dựng và phát triển hệ thống cung cấp dịch vụ như lập trình viên, quản lý dự án, kiểm thử dự án, thiết kế giao diện người dùng, khách hàng.
- Tiếp cận và áp dụng kiến trúc microservice, phương pháp DDD, mẫu CQRS kết hợp Event Sourcing vào hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt, tạo nền tảng cho quá trình phát triển trong lĩnh vực công nghệ phần mềm sau này.
- Cải thiện khả năng tự học, tìm kiếm và nghiên cứu tài liệu. Khả năng hệ thống hoá kiến thức cũng được nâng cao.
- Học hỏi kỹ năng quản lý thời gian, sắp xếp các công việc hiệu quả.
- Tích lũy thêm kinh nghiệm làm việc nhóm và cải thiện kỹ năng giao tiếp.
- Nâng cao kỹ năng viết mã nguồn, tìm kiếm và sửa lỗi, tái cấu trúc ứng dụng.

5.2 KẾT QUẢ HỆ THỐNG

5.2.1 Môi trường phát triển

- Hệ điều hành: Windows 10 Pro
- Hệ quản trị cơ sở dữ liệu: MongoDB
- Cơ sở dữ liệu mã nguồn mở: EventStoreDB
- Công cụ xây dựng và phát triển hệ thống: Microsoft Visual Studio Code

1.46.1

- Công cụ kiểm thử API: Postman 7.26.0
- Công cụ quản trị máy chủ từ xa qua giao thức SSH dành cho máy khách: MobaXterm 12.4
- Công cụ quản lý, truy vấn cơ sở dữ liệu MongoDB: MongoDB Compass Community 1.21.2
- Các thư viện/ nền tảng sử dụng

Tên thư viện/ nền tảng	Tóm tắt chức năng
NodeJS	Là một môi trường thực thi JavaScript miễn phí, đa nền tảng, cho phép thực thi các đoạn mã JavaScript phía máy chủ
ExpressJS	Là một phiên bản tối giản của Express, một nền tảng trên ứng dụng web cho NodeJS
Kafka	Là một nền tảng phân tán dữ liệu lớn, phục vụ hàng triệu truy cập trên giây cùng nhiều tính năng khác
Socket.io	Thư viện JavaScript phục vụ cho các ứng dụng thời gian thực, giúp các bên ở những địa điểm khác nhau kết nối với nhau, truyền dữ liệu ngay lập tức thông qua server trung gian.
ReactJS	Thư viện JavaScript mã nguồn mở giúp xây dựng giao diện người dùng.
Ant Design	Tập hợp các thành phần của React, phục vụ cho quá trình thiết kế liên quan đến giao diện.
NestJS	Là một nền tảng để xây dựng các ứng dụng phía

	máy chủ bằng Node.js một cách hiệu quả và dễ mở rộng, đã được tích hợp mô hình CQRS.
MongoDB	Là một cơ sở dữ liệu không quan hệ, đa nền tảng
Event Store	Là một cơ sở dữ liệu lưu trữ sự kiện tích hợp như một message bus.
Maven	Là một công cụ build tự động dùng cho các dự án xây dựng bằng ngôn ngữ Java
NuGet	Là nơi quản lý các gói tính năng dùng cho các dự án viết bằng ngôn ngữ C#.
NPM	Là nơi quản lý các gói tính năng dùng cho các dự án viết bằng JavaScript.

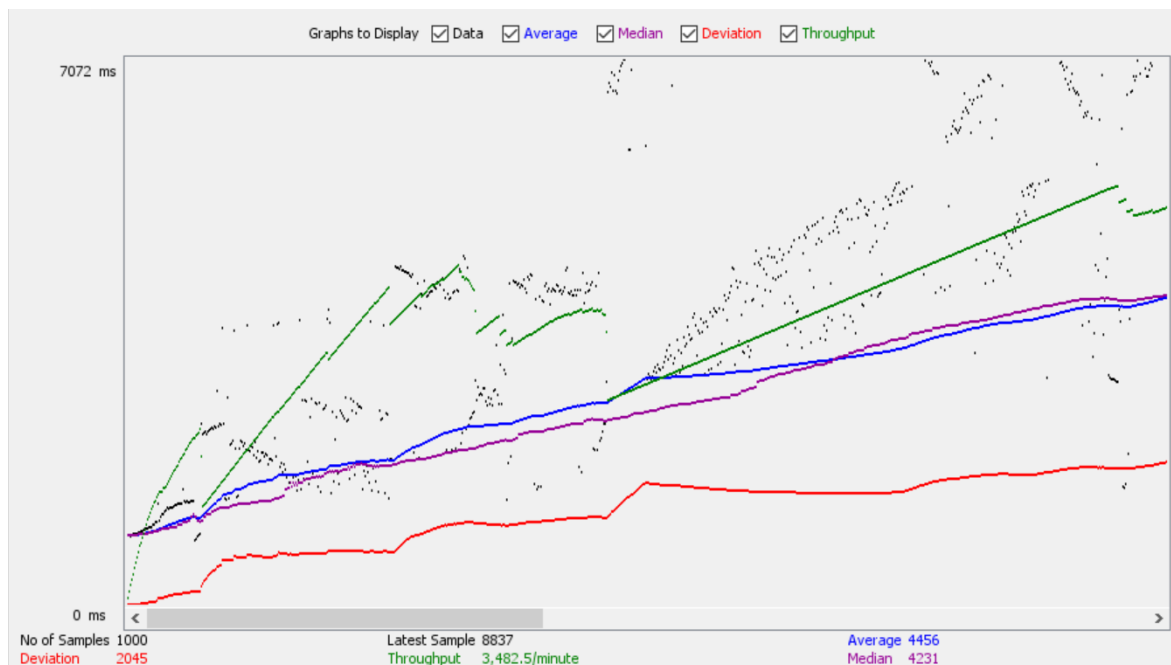
Bảng 5.2.1 Các thư viện, nền tảng sử dụng trong quá trình xây dựng hệ thống

5.2.2 Môi trường triển khai

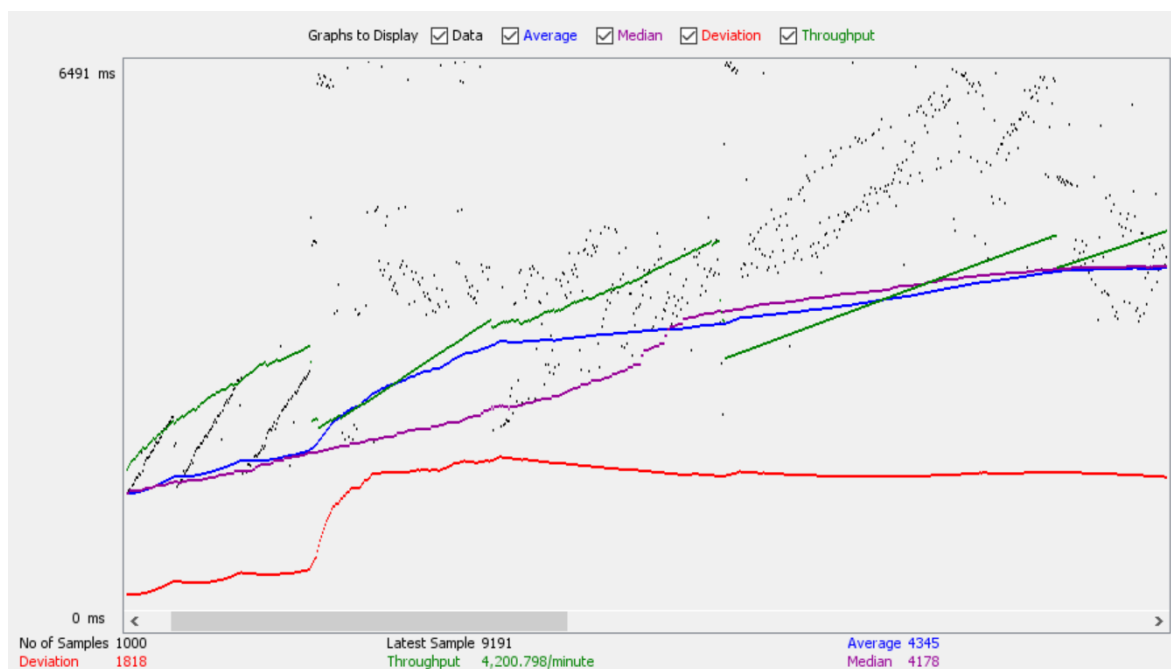
Google Cloud Platfrom – nền tảng đám mây của Google.

5.2.3 Hiệu năng hệ thống

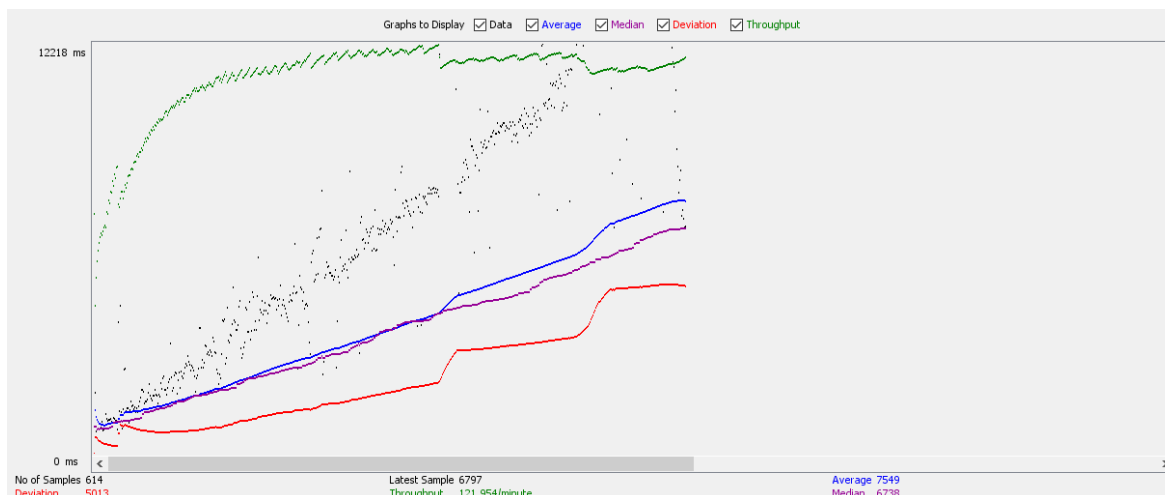
Kết quả Loadting test cho các luồng ứng dụng đọc, ghi với đầu vào là 50 người dùng truy cập tăng dần ổn định trong vòng 60 giây.



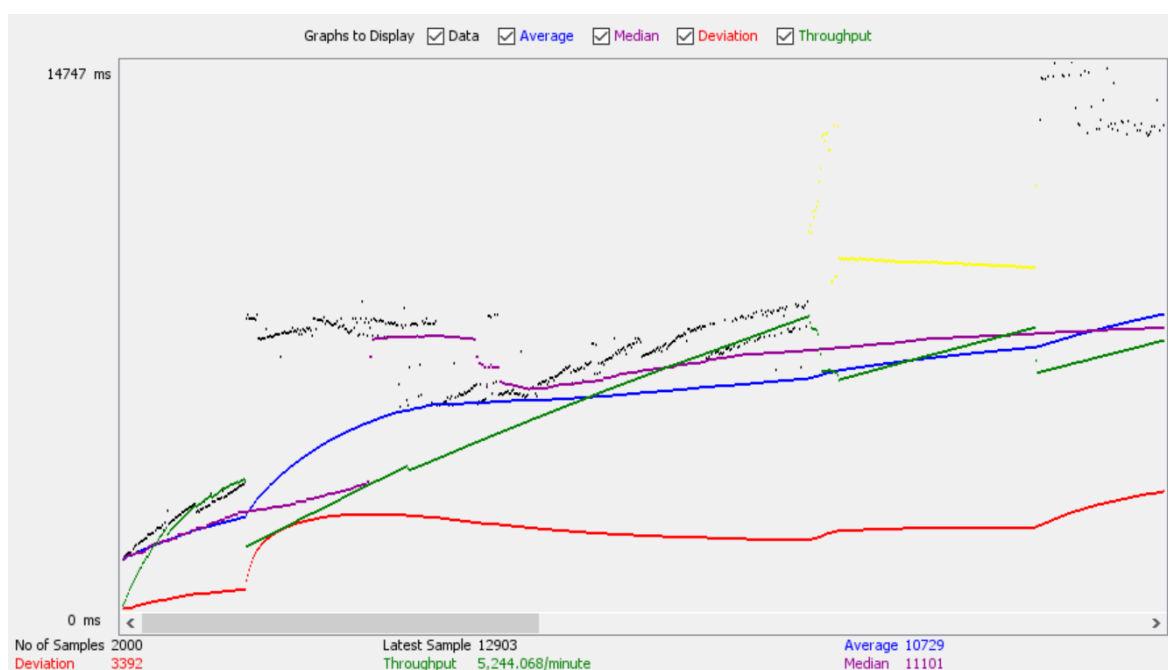
Hình 5.2.1 Kết quả Loading test cho luồng ghi trên chức năng tạo mới mã thành toán



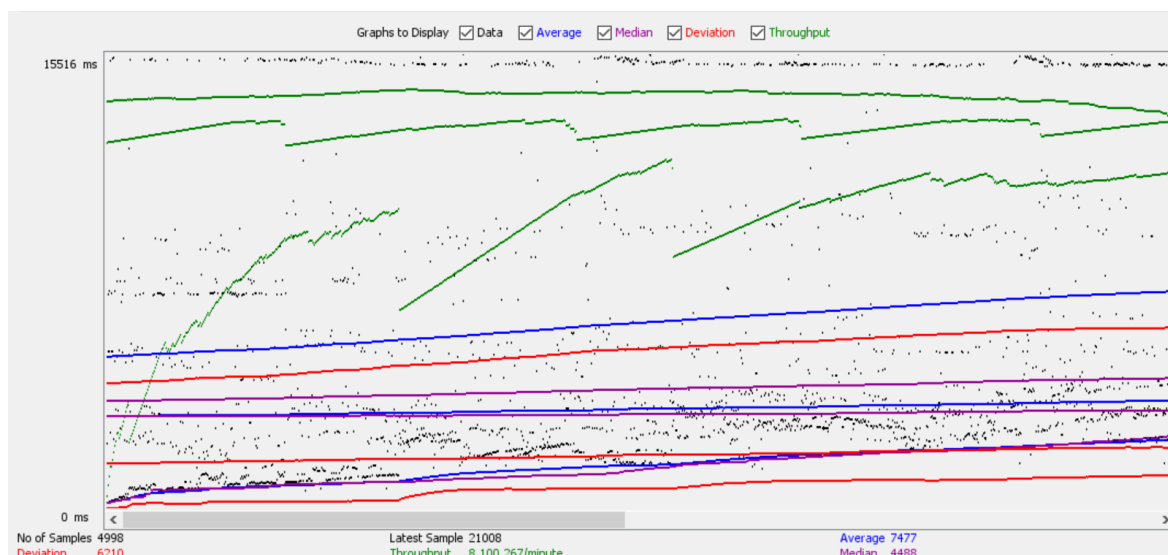
Hình 5.2.2 Kết quả Loading test cho luồng đọc dữ liệu truy vấn tất cả người dùng



Hình 5.2.3 Kết quả Loading test trên chức năng nhận dạng âm thanh



Hình 5.2.4 Kết quả Stress test trên luồng truy vấn tất cả thông tin người dùng



Hình 5.2.5 Kết quả Stress test trên chức năng tạo mới API key

Từ những hình ảnh trên, kết quả sẽ được tóm tắt trong bảng sau, tất cả thông số đã được quy đổi sang đơn vị giây.

Mục tiêu	Số requests	Throughput (requests/ giây)	Tốc độ trung bình (giây)	Kết luận
Ghi dữ liệu (tạo mã)	1000	87	4.4	Dữ liệu, độ lệch chuẩn, tốc độ trả về ổn định theo một đường thẳng
Đọc dữ liệu (truy vấn tất cả tài khoản)	1000	70	4.3	Dữ liệu, độ lệch chuẩn, tốc độ trả về ổn định theo một đường thẳng
Chuyển đổi tập tin	614	2,03	7.549	Dữ liệu, độ lệch, tốc độ trả về tăng cao khi

âm thanh				số lượng truy cập tăng
----------	--	--	--	------------------------

Bảng 5.2.2 Bảng tóm tắt kết quả Loading test cho hệ thống

Mục tiêu	Số requests	Throughput (requests/ giây)	Tốc độ trung bình (giây)	Kết luận
Ghi dữ liệu (tạo mã)	1000	58	4.4	Dữ liệu, độ lệch chuẩn, tốc độ trả về ổn định theo một đường thẳng
Đọc dữ liệu (truy vấn tất cả tài khoản)	1000	70	10.7	Dữ liệu, độ lệch chuẩn, tốc độ trả về ổn định theo một đường thẳng
Chuyển đổi tập tin âm thanh	614	135	4.35	Dữ liệu, độ lệch, tốc độ trả về tăng cao khi số lượng truy cập tăng

Bảng 5.2.3 Bảng tóm tắt kết quả Stress test cho hệ thống

5.2.4 Các chức năng đã cài đặt

- Đăng ký/ đăng nhập
- Tạo dự án, mời tham gia dự án
- Thanh toán chi phí sử dụng dịch vụ của hệ thống
- Xem lịch sử giao dịch, lịch sử sử dụng dịch vụ

- Sử dụng thử dịch vụ của hệ thống
- Nâng cấp API key
- Cho phép người quản trị quản lý người đăng ký, khoá truy cập sử dụng dịch vụ
- Cung cấp API cung cấp dịch vụ chuyển một tập tin âm thanh tiếng Việt thành nội dung văn bản
- Cung cấp các báo cáo về việc sử dụng dịch vụ của người dùng
- Cung cấp thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử dụng dịch vụ, với các ngôn ngữ khác nhau

5.2.5 So sánh với một số hệ thống khác trên thị trường

Tên chức năng	Viettel AI	FPT.AI	Google Cloud	Hệ thống của nhóm (VietSpeech)
Đăng ký/ đăng nhập	✓	✓	✓	✓
Tạo dự án		✓	✓	✓
Thanh toán chi phí sử dụng dịch vụ	✓	✓	✓	✓
Xem lịch sử giao dịch		✓	✓	✓
Mời tham gia dự án				✓
Sử dụng thử dịch vụ (hỗ trợ chỉnh sửa văn bản, lưu trữ và tải xuống)				✓
Xem lịch sử sử dụng dịch vụ	✓	✓	✓	✓

Nâng cấp API key	✓	✓	✓	✓
Cung cấp API cung cấp dịch vụ chuyển một tập tin âm thanh tiếng Việt thành nội dung văn bản	✓	✓	✓	✓
Báo cáo về việc sử dụng dịch vụ của người dùng	✓	✓	✓	✓
Cung cấp thư viện SDK, ví dụ mẫu và tài liệu hỗ trợ lập trình viên sử dụng dịch vụ, với các ngôn ngữ khác nhau	✓	✓	✓	✓

Bảng 5.2.4 So sánh chức năng giữa các hệ thống cung cấp dịch vụ

5.3 KẾT QUẢ ỨNG DỤNG DI ĐỘNG

5.3.1 Môi trường phát triển

- Hệ điều hành: Windows 10 Pro
- Cơ sở dữ liệu: Realm Database
- Công cụ xây dựng và phát triển ứng dụng: Microsoft Visual Studio Code 1.46.1
- Công cụ hỗ trợ phát triển ứng dụng: Android Studio 3.6
- Công cụ kiểm thử: Samsung Galaxy S6 Edge Plus
- Các thư viện/ nền tảng sử dụng

Tên thư viện/ nền tảng	Tóm tắt chức năng
NodeJS	Là một môi trường thực thi JavaScript miễn phí,

	đa nền tảng, cho phép thực thi các đoạn mã JavaScript phía back-end, vượt ra khỏi phạm vi trình duyệt.
React Native	Là một nền tảng do Facebook phát triển cho phép sử dụng React để xây dựng ứng dụng Android và iOS.

Bảng 5.3.1 Các thư viện, nền tảng sử dụng trong quá trình xây dựng ứng dụng di động

5.3.2 Môi trường triển khai

- Thiết bị: Samsung Galaxy S6 Edge Plus
- Hệ điều hành: Android 7.0 Nougat

5.3.3 Các chức năng đã cài đặt

- Cho phép người dùng trả lời câu hỏi của bài học thông qua việc ghi âm giọng nói bằng việc sử dụng microphone của thiết bị.
- Học lại bài học.
- Xem lịch sử quá trình học tập.

5.4 So sánh các kết quả thu được với mục tiêu ban đầu

Mục tiêu ban đầu	Nhận xét mức độ hoàn thành
Trình bày lý do xây dựng hệ thống cung cấp dịch vụ web nhận dạng âm thanh tiếng Việt.	Đã trình bày lý do cơ bản trong chương 1 của luận văn.
Trình bày tóm tắt lý thuyết nền tảng về Microservices, Domain Driven Design (DDD), Command Query Responsibility Segregation (CQRS) và Event Sourcing để thiết kế và hiện thực hóa khả năng mở rộng dịch vụ khi số	Đã trình bày ở chương 2 của luận văn.

lượng truy cập cao.	
Xây dựng dịch vụ web (API) để nhận một tập tin (file) âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản.	Đã xây dựng dịch vụ web (API) ASR VietSpeech.
Xây dựng một trang web và một mobile client demo việc sử dụng SDK để tải lên một tập tin âm thanh tiếng Việt và xuất ra nội dung ở dạng văn bản (ví dụ: Trò chơi hỗ trợ học tiếng Việt).	Đã xây dựng trang web demo và ứng dụng di động ASR VietSpeech học tiếng Việt.
Viết 120 trang luận văn theo đúng nội dung trình bày trong tài liệu “Hướng dẫn thực hiện luận văn” mà giáo viên cung cấp, theo đúng chuẩn nhà trường yêu cầu và trích dẫn tài liệu tham khảo một cách chi tiết, đầy đủ.	Luận văn được viết tương đối đầy đủ và chính xác.

Bảng 5.4.1 Bảng so sánh mục tiêu ban đầu với kết quả thu được

5.5 ĐỊNH HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU

5.5.1 Khả năng mở rộng

Để cải tiến và tiếp tục nâng cấp để phục vụ người dùng, nhóm dự định sẽ thực hiện những ý tưởng sau:

- Mở rộng Event Store bằng các Cluster.
- Mở rộng cơ sở dữ liệu bằng mô hình Master Slave.
- Mở rộng Event Bus bằng nền tảng Kafka Cluster.
- Đảm bảo tính sẵn sàng của hệ thống bằng cách Load Balancing bằng Nginx, tự động replica hệ thống bằng Kubernetes.

5.5.2 Cải thiện chất lượng nhận dạng âm thanh

Nhóm sẽ tiếp tục nghiên cứu sâu hơn về mô hình nhận dạng âm thanh để thực

hiện tăng tính chính xác của mô hình, thêm các tính năng như giảm nhiễu do tiếng ồn, giảm sai lệch cho các ngôn ngữ vùng miền,...

5.5.3 Cải thiện chức năng

Nhóm sẽ cải thiện chức năng cho hệ thống bằng cách xây dựng hướng tới mô hình Software as a Service (SAAS), đảm bảo các giao diện dễ sử dụng, tài liệu đơn giản, hiệu quả,...

5.6 LỜI KẾT

Luận văn hệ thống cung cấp dịch vụ nhận dạng nhận dạng âm thanh tiếng Việt được xây dựng là sản phẩm kết tinh của một quá trình dài nghiên cứu, làm việc và học tập nghiêm túc của nhóm nhóm. Mặc dù còn nhiều hạn chế về chất lượng dịch vụ, giao diện, chức năng của hệ thống, song sản phẩm hệ thống cung cấp dịch vụ nhận dạng âm thanh tiếng Việt đã đem lại cho bản thân nhóm chúng em được nhiều kiến thức cũng như phương hướng để triển khai một dự án trong thực tế. Các hệ thống trên nền tảng microservices nói chung và kết hợp với mô hìnhDDD, CQRS, ES là một xu hướng công nghệ nổi bật trong thời điểm hiện tại. Những kiến thức thu được từ quá trình thực hiện đề tài này sẽ là tiền đề, nền tảng cho nhóm đủ kiến thức để tiếp tục nghiên cứu và phát triển sự nghiệp của bản thân.

CHƯƠNG 6: PHỤ LỤC

6.1 PHỤ LỤC 1: THƯ VIỆN SDK

Các thư viện SDK ở các nền tảng ngôn ngữ khác nhau sẽ giúp lập trình viên khi sử dụng API key của hệ thống có thể dễ dàng cài đặt vào ứng dụng của mình một cách đơn giản nhất dựa vào tài liệu và các mã nguồn mẫu trên từng ngôn ngữ khác nhau, ở đây nhóm nhóm đã xây dựng trên 3 ngôn ngữ phổ biến đó là JavaScript với NPM package, CSharp với NuGet package, Java với Maven package.

The screenshot shows the NPM package page for 'asr-vietspeech'. At the top, it displays the package name 'asr-vietspeech', version '1.0.17', and status 'Public'. Below this, there are navigation links: 'Readme', 'Explore', 'Dependencies' (4), 'Dependents' (0), 'Versions' (17), and 'Settings'. The main heading is 'Speech to text: Node.js Client'. Below the heading, there are release level indicators: 'release level', 'general availability (GA)', and 'npm v4.1.1'. The description reads 'ASR VietSpeech Library for Node.js'. There are three links: 'ASR VietSpeech Library API Reference', 'ASR VietSpeech Library Documentation', and 'https://github.com/trankhanhlinh/vispeech-asr'. A 'Table of contents:' section lists 'Quickstart' with sub-items: 'Before you begin', 'Installing the client library', and 'Using the client library'. On the right side, there is an 'Install' section with the command '> npm i asr-vietspeech'. Below that is a 'Weekly Downloads' graph showing a peak. A table lists 'Version' (1.0.17) and 'License' (ISC). Another table lists 'Unpacked Size' (769 kB) and 'Total Files' (17). At the bottom, it shows 'Issues' (0) and 'Pull Requests' (0).

Version	License
1.0.17	ISC

Unpacked Size	Total Files
769 kB	17

Issues	Pull Requests
0	0

Hình phụ lục 6.1.1 SDK của JavaScript trên NPM package

The screenshot shows the NuGet package page for 'asr-vietspeech' version 1.0.0. The page layout includes a top navigation bar with links like 'Packages', 'Upload', 'Statistics', 'Documentation', 'Downloads', and 'Blog'. Below the navigation bar is a search bar. The main content area is divided into two columns. The left column contains the package icon, name, version, and a 'Package Description' section. Below this is a 'PackageReference' tab with a code snippet: `<PackageReference Include="asr-vietspeech" Version="1.0.0" />`. A note below the code states: 'For projects that support PackageReference, copy this XML node into the project file to reference the package.' Below the code is a 'Dependencies' section showing '.NETStandard 1.4', 'NETStandard.Library (>= 1.6.1)', and 'Newtonsoft.Json (>= 12.0.3)'. The right column contains an 'Info' section with links like 'last updated a month ago', 'Contact owners', 'Report', and 'Download package (15.73 KB)'. Below the 'Info' section is a 'Statistics' section showing '73 total downloads', '73 downloads of current version', and '1 download per day (avg)'.

Hình phụ lục 6.1.2 SDK của CSharp trên NuGet

6.2 PHỤ LỤC 2: WEB API

API là các giao diện lập trình ứng dụng (Application Programming Interface) cho phép lập trình viên giao tiếp với các cài đặt chức năng của một thư viện mà không cần phải biết rõ cài đặt ở mức thấp. Với sự phát triển của Internet, API không còn nhất thiết phải là một thư viện hay một khung làm việc bên trong thiết bị mà chương trình đang được triển khai, các nhà phát triển và thiết kế đưa ra khái niệm Web API, đem các chức năng cài đặt ở một thiết bị khác và cung cấp các yêu cầu dịch vụ đến các API cho các máy khách thông qua Internet.

Web API là một API trên web có thể được truy cập bằng giao thức HTTP, công nghệ mới này loại bỏ hoàn toàn những phức tạp và nhọc nhằn trong việc cài đặt hạ tầng và hướng người lập trình viên đến một môi trường lập trình linh hoạt và dễ dàng hơn.

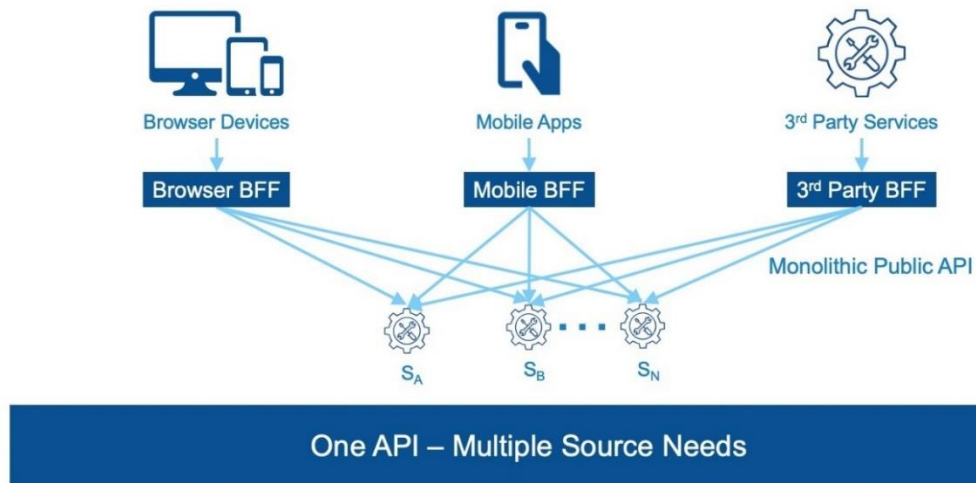
Sự dễ dàng khi cài đặt giao diện lập trình ứng dụng web còn thể hiện ở chỗ người lập trình có thể sử dụng định dạng JSON để trao đổi dữ liệu, như ta đã biết, định dạng XML sẽ rất tốn thời gian cho các bộ dịch trích xuất dữ liệu, thay vào đó, dữ liệu JSON khiến cho việc đọc hiểu cấu trúc dữ liệu và các tiến trình dịch trở nên dễ dàng và nhanh chóng hơn nên dần dần JSON đã dần có xu hướng thay thế cho

XML truyền thống trong việc trao đổi dữ liệu trên Internet. Chúng ta có thể xây dựng API Web bằng các công nghệ khác nhau như Java, .NET, ... Ví dụ API REST của Twitter cung cấp quyền truy cập theo chương trình để đọc và ghi dữ liệu bằng cách ta có thể tích hợp khả năng của Twitter vào ứng dụng của mình.

Web API là một API trên web có thể được truy cập bằng giao thức HTTP, công nghệ mới này loại bỏ hoàn toàn những phức tạp và nhọc nhằn trong việc cài đặt hạ tầng và hướng người lập trình viên đến một môi trường lập trình linh hoạt và dễ dàng hơn. Sự dễ dàng khi cài đặt giao diện lập trình ứng dụng web còn thể hiện ở chỗ người lập trình có thể sử dụng định dạng JSON để trao đổi dữ liệu, như ta đã biết, định dạng XML sẽ rất tốn thời gian cho các bộ dịch trích xuất dữ liệu, thay vào đó, dữ liệu JSON khiến cho việc đọc hiểu cấu trúc dữ liệu và các tiến trình dịch trở nên dễ dàng và nhanh chóng hơn nên dần dần JSON đã dần có xu hướng thay thế cho XML truyền thống trong việc trao đổi dữ liệu trên Internet. Chúng ta có thể xây dựng API Web bằng các công nghệ khác nhau như Java, .NET, ... Ví dụ API REST của Twitter cung cấp quyền truy cập theo chương trình để đọc và ghi dữ liệu bằng cách ta có thể tích hợp khả năng của Twitter vào ứng dụng của mình.

Ngoài khả năng mở rộng, tính linh hoạt và dễ dàng tương thích, việc sử dụng giao diện lập trình ứng dụng Web còn giúp chương trình cài đặt trở nên nhẹ nhàng hơn vì đẩy các xử lý nghiệp vụ phức tạp lên máy chủ, từ đó độ hiệu quả về cả mặt thời gian và hiệu năng được nâng lên đáng kể. Hình phụ lục 6.2.1 mô tả cách giao tiếp của các ứng dụng tới giao diện lập trình ứng dụng Web.

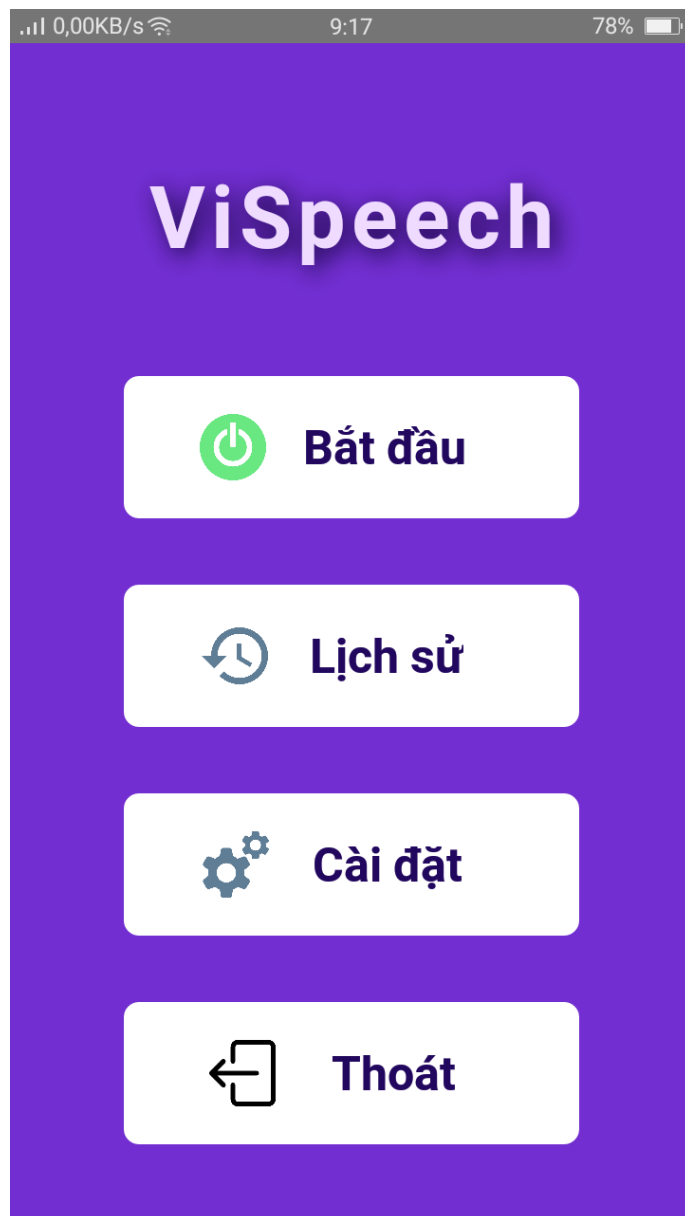
Public API Implementation



Hình phụ lục 6.2.1 Mô tả cách giao tiếp của các ứng dụng tới Web API

6.3 PHỤ LỤC 3: GIAO DIỆN CHỨC NĂNG ỨNG DỤNG DI ĐỘNG

Dưới đây là hình ảnh giao diện các màn hình chức năng cụ thể của ứng dụng di động VietSpeech. Khi người dùng khởi động ứng dụng, trang chủ ứng dụng được hiển thị có giao diện như hình sau.



Hình phụ lục 6.3.1 Màn hình trang chủ của ứng dụng

Để thực hiện chức năng học phát âm tiếng Việt, người dùng cần ấn vào nút Bắt đầu hiển thị trên màn hình trang chủ ứng dụng. Sau khi ấn, màn hình hiển thị danh sách các chủ đề bài học để người dùng lựa chọn.

← Chủ đề



Động vật

0/2 bài



Con người

0/2 bài

Hình phụ lục 6.3.2 Màn hình danh sách chủ đề của ứng dụng

Tại màn hình Chủ đề, người dùng lựa chọn một chủ đề bất kỳ. Màn hình ứng dụng sẽ hiển thị danh sách bài học sẵn có thuộc chủ đề được chọn.

Chủ đề Động vật



Gia súc



Bò sát

Hình phụ lục 6.3.3 Màn hình danh sách bài học có trong một chủ đề cụ thể

Tại màn hình danh sách bài học, người dùng lựa chọn một bài học phù hợp. Sau khi chọn, người dùng được yêu cầu chọn cấp độ (dễ, khó) cho bài học tại màn hình Chi tiết bài học.

Bài học Gia súc



Cấp độ Dễ

0/3 câu hỏi



Cấp độ Khó

0/3 câu hỏi

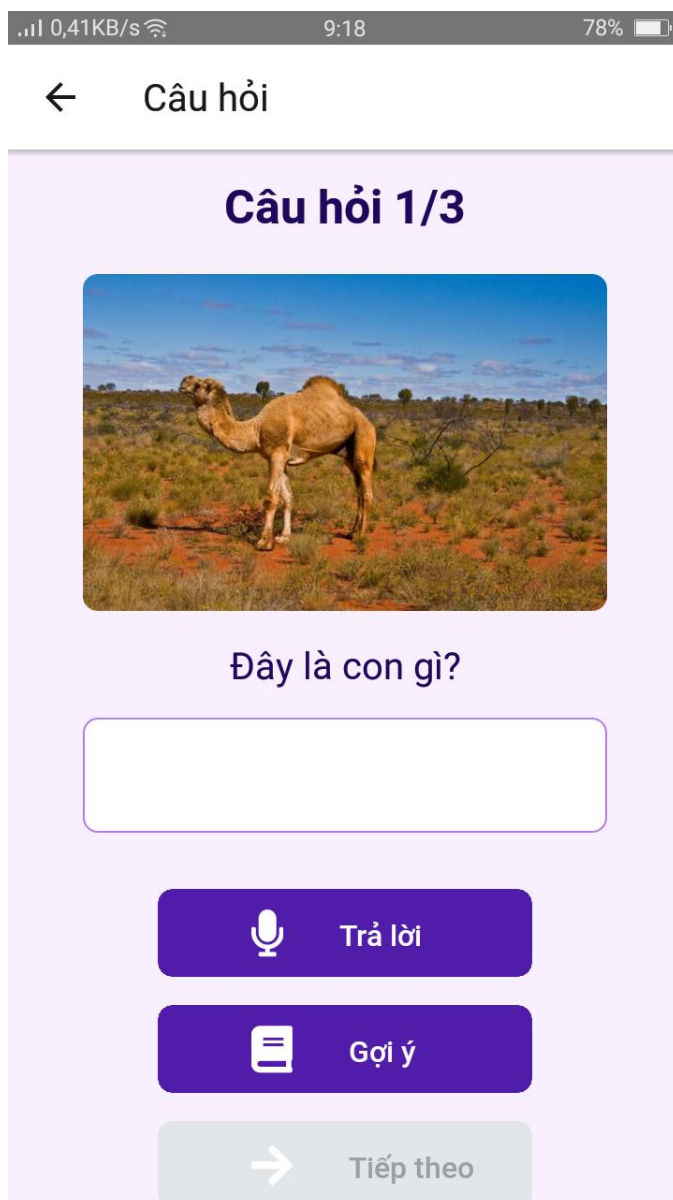
Hình phụ lục 6.3.4 Màn hình chi tiết bài học

Sau khi chọn lựa cấp độ phù hợp, người dùng sẽ bắt đầu bài học của mình tại màn hình câu hỏi. Màn hình Câu hỏi chứa thứ tự câu hỏi, hình ảnh, câu hỏi và ba nút chức năng, bao gồm:

- Nút Trả lời: dùng để thu âm câu trả lời của người dùng. Khi ấn nút, chữ được hiển thị trên nút sẽ chuyển từ “Trả lời” sang “Xong”. Khi người dùng hoàn thành việc ghi âm câu trả lời, cần ấn vào nút Xong.
- Nút Gợi ý: dùng để hiển thị câu trả lời cho câu hỏi hiện tại trong trường hợp

người dùng không biết câu trả lời.

- Nút Tiếp theo: dùng để chuyển sang câu hỏi kế tiếp trong bài học.



Hình phụ lục 6.3.5 Màn hình câu hỏi trong bài học

Sau khi hoàn tất các câu hỏi trong bài học, ứng dụng sẽ hiển thị màn hình Hoàn thành với ba nút chức năng:

- Nút Học tiếp bài mới: quay trở về màn hình chi tiết bài học để học tiếp cấp độ Khó (nếu chưa học).

- Nút Danh sách chủ đề: quay trở về màn hình danh sách chủ đề có trên ứng dụng.
- Nút Danh sách bài học: quay trở về màn hình danh sách bài học trong chủ đề đang học.



Hình phụ lục 6.3.6 Màn hình sau khi hoàn thành một bài học

Để xem lịch sử quá trình học tập, người dùng có thể ấn nút Lịch sử. Màn hình giao diện lịch sử được hiển thị như hình dưới.

← Lịch sử



Chủ đề Con người

- Cơ thể con người
- Cấp độ Khó
- ✓ 2/3 câu hỏi
- 21/06/2020 09:18

Tiếp tục



Chủ đề Con người

- Cơ thể con người
- Cấp độ Khó
- ✓ 1/3 câu hỏi
- 21/06/2020 09:18

Tiếp tục



Chủ đề Động vật

- Gia súc

Hình phụ lục 6.3.7 Màn hình lịch sử các bài đã và đang học

Cuối cùng, để thoát khỏi ứng dụng, người dùng ấn nút Thoát tại màn hình Trang chủ.

TÀI LIỆU THAM KHẢO

- [1] "Cloud Speech-to-Text API," [Online]. Available: <https://console.cloud.google.com/apis/api/speech.googleapis.com>. [Accessed 19 July 2020].
- [2] [Online]. Available: <https://console.fpt.ai/>. [Accessed 19 July 2020].
- [3] "Nhận dạng tiếng nói," [Online]. Available: <https://viettelgroup.ai/dashboard/service/speechtotext>. [Accessed 19 July 2020].
- [4] "Pricing," [Online]. Available: <https://cloud.google.com/speech-to-text/pricing>. [Accessed 19 July 2020].
- [5] "FPT.AI Speech - Speech to Text," [Online]. Available: <https://docs.fpt.ai/docs/en/speech/documentation/speech-to-text>. [Accessed 19 July 2020].
- [6] I. Nadareishvili, R. Mitra, M. McLarty and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*, 1 ed., O'Reilly Media, 2016.
- [7] "Software Engineering Institute," [Online]. Available: <https://www.sei.cmu.edu/>. [Accessed 19 July 2020].
- [8] D. Garlan, F. Bachmann, J. Ivers, R. Little, J. Stafford, L. Bass, P. Clements, P. Merson and R. Nord, *Documenting Software Architectures: Views and Beyond*, 2nd ed., Addison-Wesley Professional, 2010.
- [9] P. Kruchten, "Architectural Blueprints—The '4+1' View Model of Software Architecture," 11 1995. [Online]. Available: <https://www.cs.ubc.ca/>. [Accessed 19 July 2020].

- [10] C. Richardson, *Microservices Patterns: With examples in Java*, 1st ed., Manning Publications, 2018.
- [11] R. C. Martin, "Chapter 76. The Single Responsibility Principle," [Online]. Available: <https://www.oreilly.com/library/view/97-things-every/9780596809515/ch76.html>. [Accessed 19 July 2020].
- [12] M. L. Abbott and M. T. Fisher, *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*, 2nd ed., Addison-Wesley Professional, 2015.
- [13] M. E. Conway, "How Do Committees Invent?," [Online]. Available: http://www.melconway.com/Home/Committees_Paper.html. [Accessed 19 July 2020].
- [14] J. Lewis and M. Fowler, "Microservices," [Online]. Available: <https://martinfowler.com/articles/microservices.html>. [Accessed 19 July 2020].
- [15] "What is Domain-Driven Design," [Online]. Available: https://dddcommunity.org/learning-ddd/what_is_ddd/. [Accessed 19 July 2020].
- [16] A. Avram and F. Marinescu, *Domain-Driven Design Quickly*, Lulu.com, 2007.
- [17] D. Betts, J. Dominguez, G. Melnik, F. Simonazzi, M. Subramanian and G. Young, *Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure*, 1st ed., Microsoft patterns & practices, 2013.
- [18] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*, 1st ed., Addison-Wesley Professional, 2003.
- [19] U. Dahan, "Why you should be using CQRS almost everywhere...", [Online]. Available: <http://udidahan.com/2011/10/02/why-you-should-be-using-cqrs->

- almost-everywhere/. [Accessed 19 July 2020].
- [20] A. Polak and A. Zmenda, "Microservices design patterns for CTOs: API Gateway, Backend for Frontend and more," 19 September 2019. [Online]. Available: <https://tsh.io/blog/design-patterns-in-microservices-api-gateway-bff-and-more/>. [Accessed 25 July 2020].
- [21] P. Dave, "What is ACID Property in Database? – Interview Question of the Week #066," 10 April 2016. [Online]. Available: <https://blog.sqlauthority.com/2016/04/10/acid-properties-database-interview-question-week-066/>. [Accessed 25 July 2020].
- [22] Đ. Q. Huy, "Eventual Consistency và Strong Consistency trong hệ thống Cơ sở dữ liệu phân tán," 12 October 2018. [Online]. Available: <https://techmaster.vn/posts/34879/eventual-consistency-va-strong-consistency-trong-he-thong-co-so-du-lieu-phan-tan>. [Accessed 25 July 2020].
- [23] b. başöz, "Event Sourcing and CQRS with Axon and Spring Boot — Part 1," 29 August 2019. [Online]. Available: <https://medium.com/@berkaybasoz/event-sourcing-and-cQRS-with-axon-and-spring-boot-part-1-6d1c1d4d054e>.
- [24] D. Salas, X. Liang and Y. Liang, June 2012. [Online]. Available: <https://www.scirp.org/journal/paperinformation.aspx?paperid=19659>. [Accessed 19 July 2020].
- [25] Q. Soomro. [Online]. Available: <https://github.com/qas/examples-nodejs-cQRS-es-swagger>. [Accessed 19 July 2020].