

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KỸ THUẬT PHẦN MỀM

ĐỀ TÀI
PHÁT TRIỂN ỨNG DỤNG
HỖ TRỢ QUẢN LÝ LOÀI
SỬ DỤNG KIẾN TRÚC MICROSERVICES

Sinh viên thực hiện:

Họ và tên: **Vương Cẩm Thanh**

Mã số sinh viên: **B1805916**

Lớp: **Kỹ thuật phần mềm A3, K44**

Giảng viên hướng dẫn:

TS. Lâm Hoài Bảo

Học kỳ II, 2021-2022

TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
BỘ MÔN CÔNG NGHỆ PHẦN MỀM



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KỸ THUẬT PHẦN MỀM

ĐỀ TÀI
PHÁT TRIỂN ỨNG DỤNG
HỖ TRỢ QUẢN LÝ LOÀI
SỬ DỤNG KIẾN TRÚC MICROSERVICES

Giảng viên hướng dẫn:

TS. Lâm Hoài Bảo

Sinh viên thực hiện:

Họ và tên: Vương Cẩm Thanh

Mã số sinh viên: B1805916

Khóa: K44

Cần Thơ, 04/2022

TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN
VÀ TRUYỀN THÔNG

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

XÁC NHẬN CHỈNH SỬA LUẬN VĂN THEO YÊU CẦU CỦA HỘI ĐỒNG

Tên luận văn: Phát Triển Ứng Dụng Hỗ Trợ Quản Lý Loài Sử Dụng Kiến Trúc
Microservices

Họ và tên sinh viên: Vương Cẩm Thanh

MSSV: B1805916

Mã Lớp: DI1896A3

Đã báo cáo tại hội đồng ngành: Kỹ thuật phần mềm

Ngày báo cáo: 21/05/2022

Luận văn đã được chỉnh sửa theo góp ý của Hội đồng.

Cần Thơ, ngày 21 tháng 05 năm 2022

Giáo viên hướng dẫn

Lâm Hoài Bảo

LỜI CẢM ƠN

Lời đầu tiên, em xin chân thành cảm ơn thầy Lâm Hoài Bảo đã hướng dẫn và giúp đỡ em rất nhiều để em có thể hoàn thành luận văn. Trong quá trình thực hiện luận văn đã gặp không ít những khó khăn, chậm trễ hơn so với tiến độ đặt ra nhưng nhờ có sự giúp đỡ, hướng dẫn và những lời khuyên, an ủi tận tình từ thầy, em đã có thể quản lý được tiến độ dự án và đảm bảo hoàn thành luận văn đúng thời gian quy định.

Ngoài ra, em cũng xin cảm ơn các Thầy Cô khoa Công nghệ thông tin và truyền thông đã tạo điều kiện cho em học tập và truyền đạt những kiến thức chuyên môn cũng như các kinh nghiệm trong suốt quá trình học tập tại trường để em có đủ khả năng nghiên cứu và thực hiện tốt đề tài này.

Mặc dù đã có nhiều cố gắng thực hiện đề tài một cách hoàn chỉnh nhất, nhưng do hạn chế về mặt thời gian nghiên cứu cũng như kiến thức và kinh nghiệm nên không thể tránh khỏi những thiếu sót nhất định. Em rất mong nhận được sự thông cảm cũng như sự góp ý của quý thầy, cô và các bạn để đề tài của em được hoàn chỉnh hơn.

Xin trân thành cảm ơn.

Cần Thơ, 27 tháng 04 năm 2022

Vương Cẩm Thanh

MỤC LỤC

A. Phần Giới Thiệu	10
1. Đặt vấn đề	10
2. Lịch sử giải quyết vấn đề	10
3. Mục tiêu đề tài.....	12
4. Đối tượng và phạm vi nghiên cứu.....	13
5. Nội dung nghiên cứu	13
6. Những đóng góp chính của đề tài	13
7. Bố cục của quyển luận văn	14
B. Phần Nội Dung.....	15
Chương 1. Mô Tả Bài Toán.....	15
1. Mô tả chi tiết bài toán	15
2. Bối cảnh sản phẩm	15
3. Các chức năng sản phẩm.....	16
4. Đặc điểm của người sử dụng.....	17
5. Môi trường vận hành.....	17
6. Các ràng buộc thực thi và thiết kế.....	18
7. Các giả định và phụ thuộc.....	18
8. Các yêu cầu bên ngoài	19
9. Các quy tắc nghiệp vụ.....	20
10. Các tính năng của hệ thống	21
11. Các yêu cầu phi chức năng.....	35
Chương 2. Kiến Trúc Hệ Thống Microservices	38
1. Kiến thức nền	38
2. Xây dựng mô hình hệ thống.....	43
Chương 3. Identity Service	45

1.	Tổng quan service	45
2.	Microsoft Authentication	47
3.	Các hạn chế của Microsoft Authentication	47
4.	Xác thực bằng token:	48
5.	Bảo mật	49
6.	Các REST API quan trọng	49
7.	Thiết kế dữ liệu	59
8.	Giao tiếp ngoài	60
Chương 4. Organism Service.....		61
1.	Tổng quan service	61
2.	Tìm kiếm loài	63
3.	Các REST API quan trọng	63
4.	Thiết kế dữ liệu	74
5.	Worker service	75
6.	Giao tiếp ngoài	80
Chương 5. Các service hỗ trợ		81
1.	RabbitMQ.....	81
2.	HealthChecks	84
3.	Seq.....	85
Chương 6. Image Search		86
1.	Tổng quan service	86
2.	Thư viện tìm hình ảnh trong Python	86
3.	Tìm loài theo ảnh REST API	86
4.	Lấy hình bên Organism.....	87
Chương 7. Kiểm thử và đánh giá kết quả.....		88
1.	Giới thiệu.....	88

2.	Kế hoạch kiểm thử	89
3.	Các trường hợp kiểm thử	92
C.	Phần kết luận	101
1.	Đánh giá Microservices	101
2.	Kết quả đạt được	102
3.	Triển khai	103
4.	Hướng phát triển	103
D.	Tài liệu tham khảo	104

DANH MỤC HÌNH

Hình 1 Trang web của ứng dụng Netflix	11
Hình 2 Trang web của Amazon	12
Hình 3 Use case của Guest.....	21
Hình 4 Use case của Researcher	22
Hình 5 Use case của Expert	23
Hình 6 Linh vật .NET	38
Hình 7 Logo Blazor	39
Hình 8 Logo Watchdog Health Checks	39
Hình 9 Logo Flask	40
Hình 10 RabbitMQ	41
Hình 11 Docker.....	41
Hình 12 Mô hình hoạt động của Docker.....	42
Hình 13 Kiến trúc tổng quan của hệ thống	43
Hình 14 Identity REST API.....	45
Hình 15 Domain-Driven Design.....	46
Hình 16 JWT.....	48
Hình 17 Sequence Diagram Refresh token.....	53
Hình 18 Sequence Diagram Change password.....	55
Hình 19 Sequence Diagram Create new user	58
Hình 20 Identity Class Diagram	59
Hình 21 Organism REST API.....	61
Hình 22 Kiến trúc N-Layer	62
Hình 23 Sequence Diagram Create new species	67
Hình 24 Sequence Diagram Update species	70
Hình 25 Sequence Diagram Find species by name.....	73
Hình 26 Species Class Diagram.....	74
Hình 27 User Class Diagram	74
Hình 28 Sequence Diagram Create new User event.....	76
Hình 29 Sequence Diagram Update species	79
Hình 30 RabbitMQ UI	81
Hình 31 Unit tests	93
Hình 32 Email Unit Test.....	93

Hình 33 CI/CD trên GitHub.....	102
--------------------------------	-----

DANH MỤC BẢNG

Bảng 1 Bảng danh sách người dùng của hệ thống.....	17
Bảng 2 REST API Refresh token.....	52
Bảng 3 REST API Change password	55
Bảng 4 REST API Create user.....	57
Bảng 5 Bảng người dùng	60
Bảng 6 REST API Tạo loài mới	67
Bảng 7 REST API Cập nhật loài.....	69
Bảng 8 REST API Tìm loài theo tên	73
Bảng 9 Kịch bản kiểm thử	92
Bảng 10 Các test case	100

BẢNG CHÚ GIẢI THUẬT NGỮ

Thuật ngữ, khái niệm/từ viết tắt	Định nghĩa
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
API	Application Programming Interface
URL	Uniform Resource Locator
SPA	Single Page Application
JWT	JSON Web Token

TÓM TẮT

Với sự phát triển nền công nghiệp hiện đại đã làm biến đổi khí hậu, ô nhiễm môi trường,... Cùng với việc săn bắt trái phép, khai thác rừng,... dẫn đến sự sụt giảm nghiêm trọng của các loài động thực vật. Để quản lý tốt hơn các loài vật ở đây cũng như là hội nhập cuộc cách mạng công nghiệp số hóa dữ liệu, ứng dụng hỗ trợ tra cứu quản lý thông tin về các loài vật được phát triển.

Kiến trúc Microservices một kiểu kiến trúc phần mềm phổ biến trên các nền tảng điện toán đám mây. Trong đó các dịch vụ được chia nhỏ để thực hiện một chức năng duy nhất của hệ thống. Việc chia nhỏ các dịch vụ trong kiến trúc Microservices giúp cho hệ thống, dễ mở rộng theo chiều ngang. Với kiến trúc cũ gọi là Monolithic thì tất các chức năng nằm chung một máy chủ và là kiến trúc mở rộng theo chiều sâu vì khi muốn mở rộng phải thêm RAM hoặc CPU. Với Microservices thì khi mở rộng chỉ cần thêm một service khác là được.

Xuất phát từ những ý nghĩa thực tiễn như vậy, em đã thực hiện đề tài luận văn “Phát triển hệ thống quản lý loài dùng kiến trúc Microservices” để tìm hiểu và áp dụng kiến trúc Microservices trong việc xây dựng và phát triển một ứng dụng cụ thể - ứng dụng web “Species Wiki”.

ABSTRACT

In our current life we are facing a lot of problems caused by global warming, which cause pollution, floods and power shortages. On the other hand, illegal hunting and forest exploitation, all these things combined, is leading to a serious decline of many species. For managing species and also digitizing migrations, creating a software will be the solution for the problem.

Microservices architecture is a new growing type of software architecture today, where services are broken down to perform a single function of the system. The breakdown of services in the Microservices architecture makes the system simpler, easier to develop, reduces construction costs, and increases technology adaptability. Microservices architecture is considered as the preeminent solution for the current problem of building and developing service-based systems.

I have come to the conclusion that I will create a Microservices Architecture Application for managing species. "Managing Species Using Microservices Architecture" will introduce Microservices Architecture, its outstanding features, priorities in building applications that use modular and highly reusable – “Species Wiki” a web application.

A. PHẦN GIỚI THIỆU

1. Đặt vấn đề

Dịch vụ đám mây ngày càng phát triển đã giúp các nhà xây dựng và phát triển phần mềm tạo ra các hệ thống có khả năng thích ứng cao với nhiều môi trường khác nhau, tăng khả năng tái sử dụng. Các hệ thống được phát triển nhanh chóng và giảm được sự phức tạp, hạ giá thành khi xây dựng và triển khai.

Tuy nhiên việc không quan tâm đến kích thước của các dịch vụ trong hệ thống đang đặt ra bài toán khó cho các nhà xây dựng và phát triển phần mềm là làm sao giảm chi phí khi xây dựng hệ thống với các dịch vụ, làm sao tránh ảnh hưởng đến cả hệ thống khi muốn thay đổi một số chức năng của hệ thống,...

Phạm vi của dịch vụ càng lớn hệ thống càng trở lên phức tạp, khó phát triển, kiểm thử và bảo trì. Chính những điều này đang làm cho việc xây dựng và phát triển hệ thống phần mềm dựa trên dịch vụ đang vượt khỏi khả năng kiểm soát của các kiểu kiến trúc phần mềm hiện có và cần phải có một kiểu kiến trúc mới để giải quyết vấn đề này.

2. Lịch sử giải quyết vấn đề

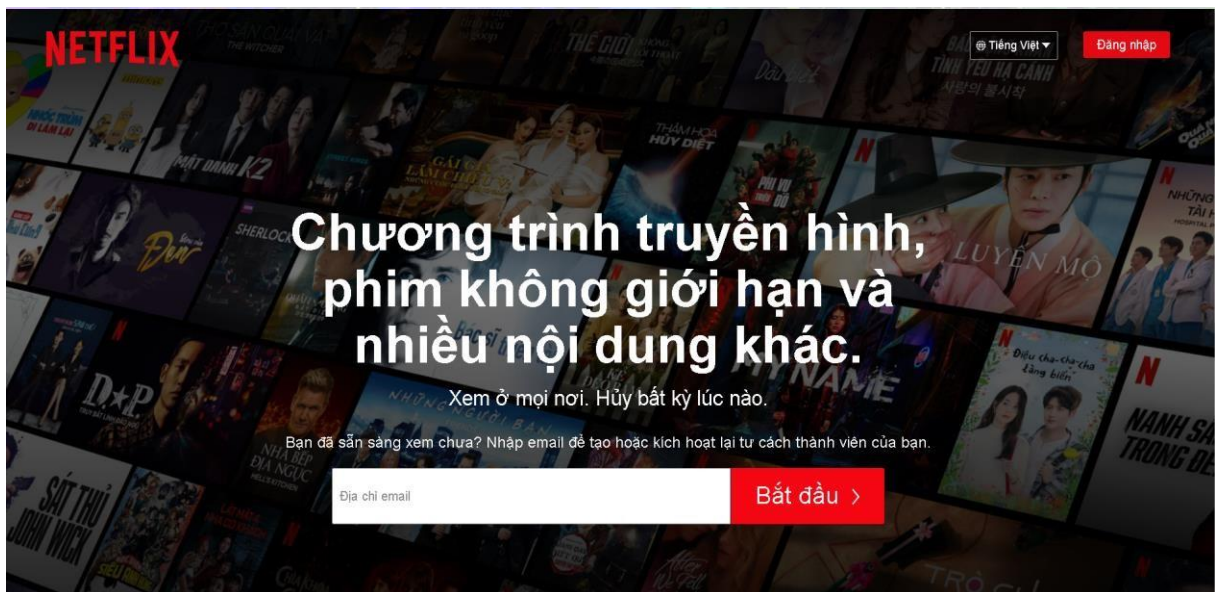
Sự phát triển của mạng internet, đã bắt đầu nhen nhóm về Microservices từ những năm 2004, nhưng chưa quá nổi bật. Nhưng đến những năm gần đây với sự bùng nổ của công nghệ với Smartphone, wifi vv... dẫn đến việc con người ngày càng tiếp cận với các dịch vụ công nghệ nhiều hơn, nhiều business công nghệ xuất hiện nhằm đáp ứng nhu cầu hiện nay. Thay vì trước đây chỉ có những hệ thống nhỏ, với những trang tin tức, thông tin, với lượng truy cập và tương tác không quá lớn, thì bây giờ những dịch vụ mạng xã hội, fintech phát triển đòi hỏi sự phát triển ngày càng nhanh để đáp ứng nhu cầu ngành càng tăng đến người dùng. Do đó, sự bất lợi và nhược điểm của Monolithic bắt đầu xuất hiện.

Kiến trúc Microservices là một kiểu kiến trúc phần mềm mới và đang rất phát triển hiện nay. Trong đó các dịch vụ được chia nhỏ để thực hiện một chức năng duy nhất của hệ thống. Việc chia nhỏ các dịch vụ trong kiến trúc Microservices giúp cho hệ thống đơn giản hơn, dễ phát triển hơn, giảm chi phí xây dựng, tăng khả năng thích ứng công nghệ. Kiến trúc Microservices được coi là lời giải ưu việt cho bài toán xây dựng và phát triển hệ thống dựa trên dịch vụ hiện nay. Nó đã và đang được nghiên cứu và ứng dụng rộng rãi bởi các công ty lớn như Netflix, Ebay,

Amazon, Twitter, Paypal, Gilt, Soundcloud,... Đặc biệt là hiện nay khi các sản phẩm phần mềm đóng gói đang dần được thay thế bởi các phần mềm dịch vụ thì kiến trúc Microservices sẽ là đề tài ngày càng được quan tâm.

Hiện nay, kiến trúc Microservices được coi là giải pháp tối ưu để các công ty coi như là lời giải ưu việt cho bài toán xây dựng và phát triển hệ thống dựa trên dịch vụ như hiện nay các công ty áp dụng kiến trúc này:

- <https://www.netflix.com/vn/>



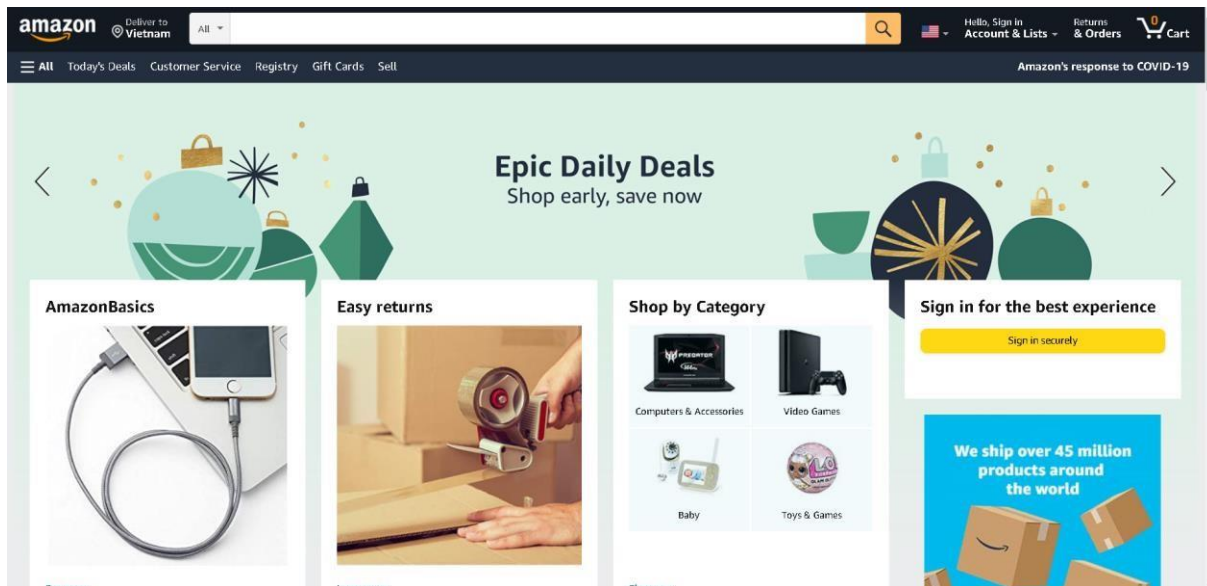
Hình 1 Trang web của ứng dụng Netflix

Netflix là một trong những công ty đi đầu trong việc phát triển Microservices, một công ty dịch vụ xem video trực tuyến của Mỹ, nội dung chủ yếu là phim và các chương trình truyền hình, rất phổ biến ở Mỹ và nhiều nước khác trên thế giới mang đến đa dạng các loại chương trình truyền hình, phim, phim tài liệu đoạt giải thưởng và nhiều nội dung khác trên hàng nghìn thiết bị có kết nối internet.

- <https://www.amazon.com/>

Amazon là công ty hàng đầu thế giới về lĩnh vực bán lẻ trực tuyến. Hơn thế, Amazon đang vươn lên mạnh mẽ với vai trò là nhà cung cấp dịch vụ điện toán đám mây. Xuất phát điểm của

Amazon là trang bán sách. Về sau Amazon mở rộng với hàng loạt hàng hóa tiêu dùng và phương tiện truyền thông kỹ thuật số cũng như các thiết bị điện tử của mình.



Hình 2 Trang web của Amazon

3. Mục tiêu đề tài

Xuất phát từ những ý nghĩa thực tiễn như vậy, em đã thực hiện đề tài luận văn “Phát triển hệ thống quản lý loài sử dụng kiến trúc Microservices” để tìm hiểu và áp dụng kiến trúc Microservices trong việc xây dựng và phát triển một ứng dụng cụ thể - ứng dụng web người dùng cập nhật tra cứu thông tin về các loài sinh vật. Dựa trên việc áp dụng kiến trúc Microservices trong thực tế từ đó đưa ra các phân tích, đánh giá và rút ra các ưu nhược điểm của kiến trúc Microservices.

- Tìm hiểu nền tảng web đã phát triển để lấy thông tin các loài vật.
- Tìm hiểu thêm một số framework và các kỹ thuật khác hiện đang được các công ty phần mềm sử dụng để tích hợp với Microservices.
- Tìm hiểu RabbitMQ, Seq, HealthChecks, Blazor, Webassembly, SignalR, SPA, REST API, ...

Trang web có vài chức năng cụ thể sau:

- Sử dụng hai ngôn ngữ tiếng Anh và tiếng Việt. Tùy chỉnh màu sắc chủ đề.

- Hiển thị thông tin các loài vật hiện có. Tìm kiếm loài theo tên sử dụng machine learning, sắp xếp các loài theo bảng chữ cái.
- Hiển thị loài của một tác giả. Hiển thị thông tin tác giả.
- Cập nhật thông tin các loài vật mới.

4. Đối tượng và phạm vi nghiên cứu

4.1. Đối tượng sử dụng:

- Người dùng bình thường (khách) xem thông tin các loài động thực vật, tìm kiếm nâng cao loài theo tên.
- Nhà nghiên cứu có quyền thêm, sửa thông tin của loài.
- Chuyên gia duyệt bài viết của các nhà nghiên cứu.

4.2. Phạm vi nghiên cứu:

Mô hình kiến trúc Microservices và các công nghệ đi kèm. Mỗi phần em sẽ giới thiệu sơ lược và trình bày những nội dung cơ bản nhất, những điểm mạnh hay lợi ích mà nó mang lại cho các nhà phát triển phần mềm. Sau khi tìm hiểu em sẽ vận dụng kết quả tìm hiểu được vào việc phát triển hệ thống sử dụng kiến trúc Microservices nhằm mục đích minh họa cho phần lý thuyết đã trình bày.

5. Nội dung nghiên cứu

Tìm hiểu và nghiên cứu các tài liệu về mô hình Microservices và các công nghệ có liên quan đến việc phát triển hệ thống như: .Net Framework, Docker, RabbitMq, Entity Framework, SignalR, Blazor WebAssembly, HealthChecks, ... của tác giả trong và ngoài nước, các bài báo, thông tin trên mạng... sau đó chọn lọc và sắp xếp lại theo ý tưởng của mình. Dựa trên kết quả tìm hiểu được để phát triển hệ thống có áp dụng tất cả những nội dung đã nghiên cứu nhằm mục đích minh họa cho phần cơ sở lý thuyết sẽ trình bày trong nội dung luận văn này.

6. Những đóng góp chính của đề tài

- Kết quả nghiên cứu có thể làm tài liệu tham khảo.
- Phần nghiên cứu lý thuyết sẽ cung cấp một cách nhìn tổng quát về quá trình phát triển phần mềm theo kiến trúc Microservices, ưu nhược điểm và các điểm liên quan.
- Phát triển thành công một trang web sử dụng kiến trúc Microservices, đầy đủ các chức năng cơ bản đã đề ra.

- Xây dựng giao diện hài hòa, thân thiện với người dùng.
- Giao diện hệ thống tương thích với nhiều thiết bị khác nhau, chạy tốt trên nhiều trình duyệt phổ biến.
- Tính bảo mật cao.
- Hiệu suất ứng dụng cao, tốc độ load dữ liệu nhanh.
- Dễ mở rộng hệ thống.

7. Bố cục của luận văn

Bố cục luận văn được xây dựng gồm 3 phần và 1 tài liệu tham khảo.

- **Phần giới thiệu:** nêu ra các vấn đề cần giải quyết, lý do tại sao phải thực hiện đề tài này, trong quá khứ đã có các hệ thống, các website nào tương tự được xây dựng để giải quyết các vấn đề đã đặt ra và chưa giải quyết được những vấn đề nào, qua đó xác định được mục tiêu của đề tài, đó cũng chính là những vấn đề trọng tâm mà đề tài đang thực hiện muốn giải quyết cũng như xác định được đối tượng và phạm vi nghiên cứu của đề tài.
- **Phần nội dung:** Phần này được chia ra làm các chương như sau:
 - Chương 1.** Tổng quan hệ thống. Chương này trình bày các yêu cầu cho ứng dụng cũng như các yêu cầu phát triển và yêu cầu nghiên cứu.
 - Chương 2.** Kiến trúc Microservices. Chương này trình bày các kiến thức liên quan đến Microservices, kiến trúc của hệ thống, đồng thời cũng giới thiệu sơ qua về các công nghệ sử dụng khi phát triển hệ thống.
 - Chương 3.** Identity Service. Chương này miêu tả về service dùng để xác thực.
 - Chương 4.** Organism Service. Chương này miêu tả service chính của hệ thống về quản lý loài
 - Chương 5.** Các service khác. Chương này nói về các service: RabbitMQ Service, message queue và nhiệm vụ trong hệ thống. HealthChecks Service, miêu tả về công dụng healthchecks trong Microservice, Species Wiki
 - Chương 6.** Image Search. Chương này đề cập đến việc sử dụng thư viện tìm hình ảnh của Flask api.
 - Chương 7.** Kiểm thử và đánh giá, trong chương này sẽ mô tả mục tiêu kiểm thử, kịch bản kiểm thử và kết quả kiểm thử xem có chạy được và đúng như mong đợi hay không, có phát sinh lỗi ngoài dự đoán hay không nhằm kiểm soát và phát hiện ra các lỗi tiềm ẩn trong hệ thống và khắc phục, sửa chữa.
- **Phần kết luận:** trình bày kết quả đã đạt được sau khi hoàn thành, đưa ra kết quả đạt được, những tiêu chí, đánh giá mức độ hoàn thành và chưa hoàn thành cũng như những mặt hạn chế, những điều chưa làm được của hệ thống.
- **Tài liệu tham khảo:** Ghi chú các tài liệu đã tham khảo.

B. PHẦN NỘI DUNG

Chương 1. Mô Tả Bài Toán

1. Mô tả chi tiết bài toán

Hệ thống có những yêu cầu sau:

- Đối với người dùng bình thường (khách):
 - Xem danh sách loài
 - Tìm kiếm loài
 - Xem danh sách các tác giả
 - Xem danh sách các loài của tác giả
 - Xem chi tiết loài
- Đối với nhà nghiên cứu (Researcher):
 - Xem thông tin cá nhân
 - Cập nhật thông tin
 - Xem danh sách loài của cá nhân
 - Tạo loài mới
 - Sửa thông tin loài
- Đối với chuyên gia (Expert):
 - Xem danh sách các loài cần duyệt
 - Sửa loài cần duyệt
 - Xem thống kê các loài
 - Quản lý các nhà nghiên cứu

2. Bối cảnh sản phẩm

Với sự phát triển nền công nghiệp hiện đại đã làm biến đổi khí hậu, ô nhiễm môi trường,... Cùng với việc săn bắt trái phép, khai thác rừng,... dẫn đến sự sụt giảm nghiêm trọng của các loài động thực vật. Để quản lý tốt hơn các loài vật ở đây cũng như là hội nhập cuộc cách mạng công nghiệp số hóa dữ liệu, ứng dụng hỗ trợ tra cứu quản lý thông tin về các loài vật được phát triển.

Kiến trúc Microservices một kiểu kiến trúc phần mềm phổ biến trên các nền tảng điện toán đám mây. Trong đó các dịch vụ được chia nhỏ để thực hiện một chức năng duy nhất của hệ thống. Việc chia nhỏ các dịch vụ trong kiến trúc Microservices giúp cho hệ thống, dễ mở rộng theo chiều ngang. Với kiến trúc cũ gọi là Monolithic thì tất các chức năng nằm chung một máy chủ

và là kiến trúc mở rộng theo chiều sâu vì khi muốn mở rộng phải thêm RAM hoặc CPU. Với Microservices thì khi mở rộng chỉ cần thêm một service khác là được.

Xuất phát từ những ý nghĩa thực tiễn như vậy, em đã thực hiện đề tài luận văn “Phát triển hệ thống quản lý loài dùng kiến trúc Microservices” để tìm hiểu và áp dụng kiến trúc Microservices trong việc xây dựng và phát triển một ứng dụng cụ thể - ứng dụng web “Species Wiki”.

3. Các chức năng sản phẩm

- Các người dùng bình thường (khách):
 - Xem danh sách loài
 - Tìm kiếm loài
 - Xem danh sách các tác giả
 - Xem danh sách các loài của tác giả
 - Xem chi tiết loài
 - Gợi ý loài tương tự
 - Chuyển đổi ngôn ngữ
 - Chuyển màu nền
 - Đăng nhập
- Đối với nhà nghiên cứu (Researcher):
 - Cập nhật thông tin cá nhân
 - Xem danh sách loài của cá nhân
 - Thêm mới loài
 - Sửa thông tin loài
 - Đăng xuất
- Đối với chuyên gia (Expert):
 - Xem danh sách các loài cần duyệt
 - Sửa loài cần duyệt
 - Xem thống kê các loài
 - Tạo nhà nghiên cứu mới
 - Sửa thông tin nhà nghiên cứu
 - Khóa tài khoản nhà nghiên cứu
 - Reset mật khẩu tài khoản

4. Đặc điểm của người sử dụng

Nhóm người sử dụng	Đặc trưng	Vai trò	Quyền hạn	Mức độ quan trọng
Expert	Là người chịu trách nhiệm quản lý toàn bộ hệ thống	Expert	Expert	Rất quan trọng
Người dùng chưa có tài khoản	Là người dùng có thể tạo tài khoản để sử dụng các chức năng của hệ thống	Người dùng bình thường	Người dùng bình thường	Quan trọng
Researcher	Là người dùng đã đăng ký tài khoản, người thực hiện quản lý các bài viết của hệ thống	Researcher	Researcher	Quan trọng

Bảng 1 Bảng danh sách người dùng của hệ thống

5. Môi trường vận hành

- Phần cứng:
 - Máy tính có kết nối internet.
 - Ram: 12GB
 - HDD: 500GB
 - CPU: Intel Core i5
- Hệ điều hành và phần mềm:

- Hệ điều hành: Windows 11
- Trình duyệt web: Google Chrome, Microsoft Edge
- Phần mềm API: Postman

6. Các ràng buộc thực thi và thiết kế

➤ Thực thi:

- Máy chủ phải được đặt trong môi trường, nhiệt độ đủ tốt để hoạt động.
- Phải có kết nối internet ổn định.
- Thời gian downtime của các service ít nhất là 30s.
- Thời gian phản hồi ngắn.
- Hoạt động 24/7.
- Đáp ứng được tối thiểu 1000 người dùng cùng lúc.
- Website được tải ổn định trên các thiết bị, hạn chế tình trạng quá tải dẫn đến chậm lag gây mất thời gian của người dùng.
- Có độ bảo mật cao, khó bị tấn công và sập bất ngờ
- Đường truyền mạng của hệ thống ổn định, có đường truyền sẵn sàng thay thế

➤ Thiết kế:

- Sử dụng mô hình Client-Server và chuẩn REST API.
- Sử dụng JWT để xác thực không trạng thái (stateless) trên web.
- Ngôn ngữ lập trình sử dụng: C#, Python.
- Giao diện đơn giản, thân thiện với người sử dụng, giảm số bước thao tác khi thực hiện một chức năng.
- Thông điệp giao tiếp giữa client và server bằng cách gửi nhận các gói tin dưới dạng JSON.

7. Các giả định và phụ thuộc

- Tài khoản người dùng có thể bị tấn công.
- Máy chủ có thể bị nhiễm viruses hoặc hacker tấn công làm hỏng dữ liệu.
- Máy chủ có thể bị quá tải do vận hành trong thời gian dài.
- Đường truyền mạng không ổn định làm gián đoạn kết nối.
- Bàn giao sản phẩm không đúng thời hạn

- Tốc độ phản hồi của hệ thống phụ thuộc vào nhiều yếu tố: phản ứng của người dùng, tốc độ mạng, băng thông của nhà cung cấp dịch vụ mạng

8. Các yêu cầu bên ngoài

- Giao diện người dùng:
 - Giao diện nền web: sử dụng Bootstrap, CSS, Material Design.
 - Giao diện thân thiện với người dùng.
 - Màu sắc: Đơn giản, dễ nhìn, không gây rối mắt
 - Thanh header đặt cố định trên cùng của trang để người dùng dễ tương tác
 - Các thành phần của giao diện phải thống nhất
 - Có popup thông báo.
- Giao tiếp phần cứng:
 - Services: giao tiếp qua mạng internet.
 - Client: giao tiếp với người dùng qua màn hình và các thiết bị nhập xuất (bàn phím, chuột, màn hình cảm ứng,...).
- Giao tiếp phần mềm:
 - Hệ điều hành: Windows/Linux(Ubuntu).
 - Các phần mềm khác: Seq, RabbitMQ.
- Giao tiếp truyền thông tin:
 - Trình duyệt web: Google Chrome, Microsoft Edge, FireFox (hỗ trợ tốt nhất cho Google Chrome).
 - Định dạng thông điệp: truyền theo dạng json.
 - Cần internet (wifi, dữ liệu di động,...) để làm cầu nối giữa client và các services
 - Sử dụng giao thức TCP/IP để truyền và nhận dữ liệu giữa service với client

9. Các quy tắc nghiệp vụ

9.1. Thêm email

Mật khẩu hợp lệ	Y	N	*	*
Email đúng định dạng	Y	*	N	*
Email không trùng	Y	*	*	N
Thêm thành công	X			
Báo lỗi		X	X	X

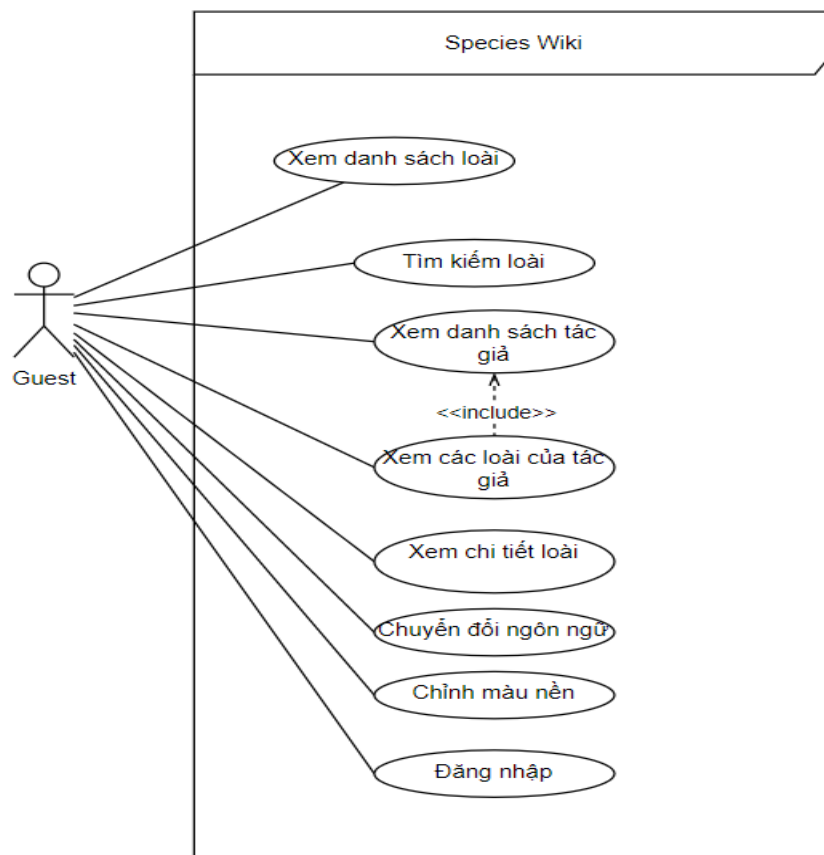
9.2. Đổi mật khẩu

Mật khẩu cũ hợp lệ	Y	N	*	*
Mật khẩu cũ chính xác	Y	*	N	*
Mật khẩu mới hợp lệ	Y	*	*	N
Sửa thành công	X			
Báo lỗi		X	X	X

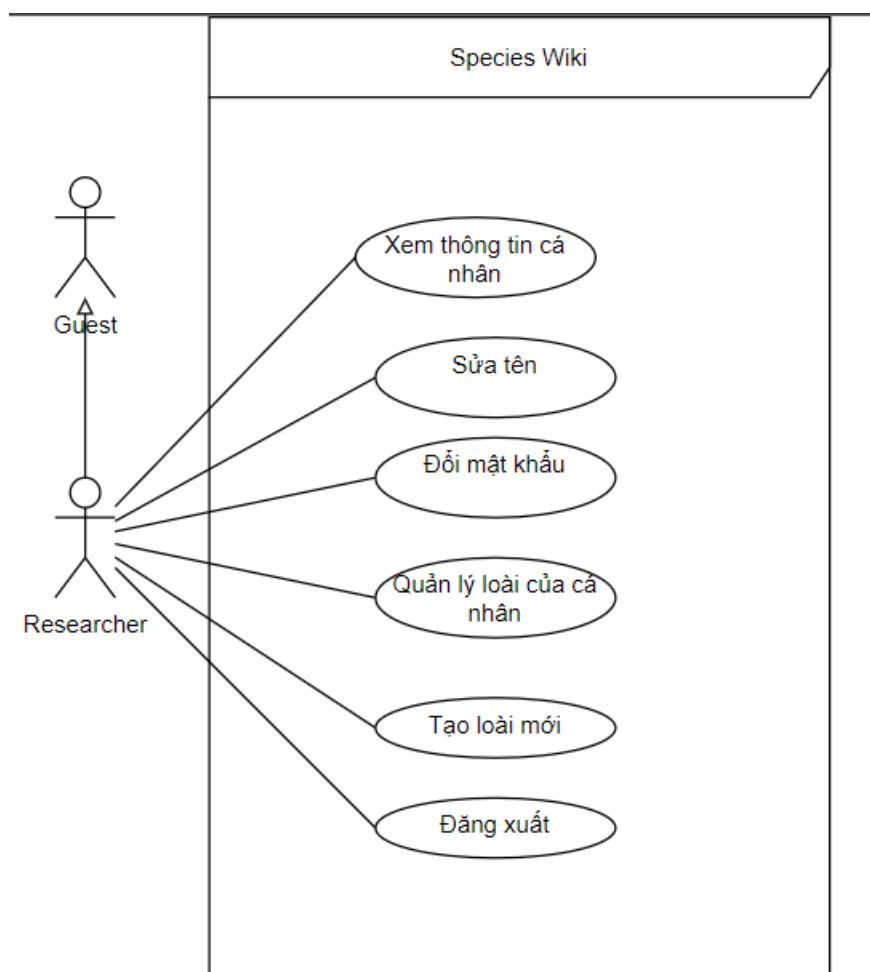
9.3. Đăng nhập

Email hợp lệ	Y	N	*	Y	Y	Y
Mật khẩu hợp lệ	Y	*	N	Y	Y	Y
Đúng email	Y	*	*	N	*	Y
Đúng mật khẩu	Y	*	*	*	N	Y
Email không bị vô hiệu	Y	*	*	*	*	N
Đăng nhập thành công	X					
Báo lỗi không hợp lệ		X	X			
Báo lỗi sai thông tin đăng nhập				X	X	X

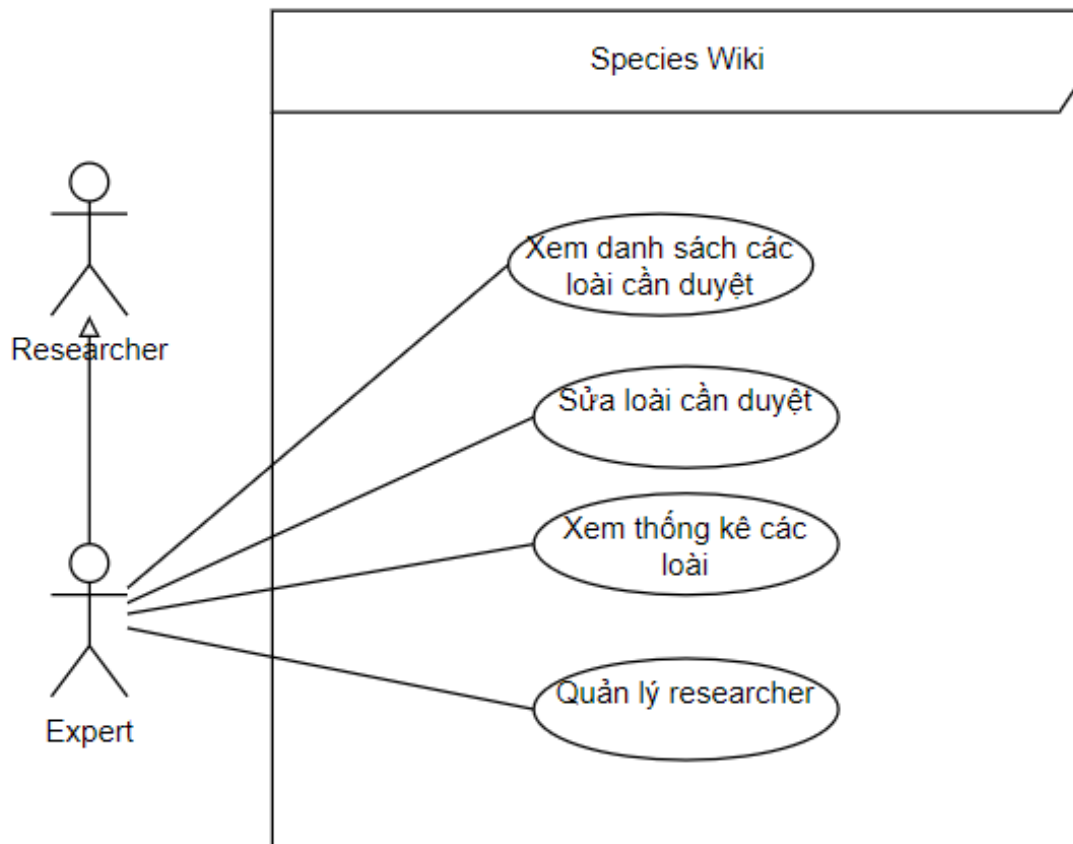
10. Các tính năng của hệ thống



Hình 3 Use case của Guest



Hình 4 Use case của Researcher



Hình 5 Use case của Expert

10.1. Xem danh sách loài

Tên use case: Xem danh sách loài	ID: UC-1
Actor chính: Guest, Researcher, Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn xem danh sách các loài.	

<p>Mô tả tóm tắt:</p> <p>Tác nhân có nhu cầu xem danh sách các loài có trong hệ thống, hệ thống sẽ trả về danh sách loài có trong hệ thống.</p>
<p>Trigger: Khi tác nhân nhấp vào xem danh sách loài.</p> <p>Type: External</p>
<p>Các mối quan hệ:</p> <p>+Association (kết hợp): Guest</p> <p>+Include(bao gồm): Không</p> <p>+Extend(mở rộng): Không</p> <p>+Generalization(tổng quát hóa): Không</p>
<p>Luồng xử lý bình thường của sự kiện:</p> <ol style="list-style-type: none"> 1. Tác nhân nhấp vào xem danh sách loài. 2. Hệ thống hiển thị danh sách các loài lên màn hình. 3. Kết thúc một sự kiện.
<p>Các luồng sự kiện con:</p>
<p>Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):</p>

10.2. Tìm kiếm loài theo tên

Tên use case: Tìm kiếm loài theo tên	ID: UC-2
Actor chính: Guest, Researcher, Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
<p>Các thành phần tham gia và mối quan tâm:</p> <p>Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn tìm loài theo tên.</p>	

<p>Mô tả tóm tắt:</p> <p>Khi tác nhân có nhu cầu tìm loài có trong hệ thống, hệ thống sẽ trả về danh sách loài có trong hệ thống.</p>
<p>Trigger: Khi tác nhân nhấp vào tìm loài theo tên.</p> <p>Type: External</p>
<p>Các mối quan hệ:</p> <p>+Association (kết hợp): Guest</p> <p>+Include(bao gồm): Không</p> <p>+Extend(mở rộng): Không</p> <p>+Generalization(tổng quát hóa): Không</p>
<p>Luồng xử lý bình thường của sự kiện:</p> <ol style="list-style-type: none"> 1. Tác nhân nhập tên loài vào ô tìm kiếm tên loài 2. Tác nhân nhấn vào nút tìm kiếm. 3. Hệ thống hiển thị danh sách các loài lên màn hình. 4. Kết thúc một sự kiện.
<p>Các luồng sự kiện con:</p>
<p>Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):</p> <p>E1: Tác nhân nhập tên chưa chính xác thì hệ thống sẽ tìm tương đối theo tên.</p>

10.3. Tìm kiếm loài theo hình

<p>Tên use case: Tìm kiếm loài theo hình</p>	<p>ID: UC-3</p>
<p>Actor chính: Guest, Researcher, Expert</p>	<p>Mức độ cần thiết: Nên có</p>
	<p>Phân loại: Cao</p>
<p>Các thành phần tham gia và mối quan tâm:</p>	

Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn tìm loài theo hình.
Mô tả tóm tắt: <p>Khi tác nhân có nhu cầu tìm loài có trong hệ thống, hệ thống sẽ trả về danh sách loài có trong hệ thống.</p>
Trigger: Khi tác nhân nhập vào tìm loài theo hình. Type: External
Các mối quan hệ: +Association (kết hợp): Guest +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn hình loài 2. Tác nhân nhấn vào nút tìm kiếm. 3. Hệ thống hiển thị danh sách các loài lên màn hình. 4. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.4. Xem chi tiết loài

Tên use case: Xem chi tiết loài	ID: UC-4
Actor chính: Guest, Researcher, Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm:	

Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn xem thông tin loài.
Mô tả tóm tắt: Khi tác nhân có nhu cầu xem thông tin loài trong hệ thống, hệ thống sẽ trả về thông tin chi tiết của loài trong hệ thống.
Trigger: Khi tác nhân nhập vào xem chi tiết loài. Type: External
Các mối quan hệ: + Association (kết hợp): Guest + Include(bao gồm): Không + Extend(mở rộng): Không + Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn xem chi tiết loài 2. Hệ thống hiển thị danh sách lên màn hình. 3. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.5. Đăng nhập

Tên use case: Đăng nhập	ID: UC-5
Actor chính: Guest	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn đăng nhập.	

Mô tả tóm tắt: Khi tác nhân muốn đăng nhập vào hệ thống để thực các chức năng của hệ thống.
Trigger: Khi tác nhân nhập vào đăng nhập. Type: External
Các mối quan hệ: +Association (kết hợp): Guest +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none">1. Tác nhân chọn đăng nhập2. Hệ thống hiển thị ô nhập tài khoản và mật khẩu.3. Tác nhân nhập vào tài khoản và mật khẩu.4. Tác nhân nhấn nút đăng nhập5. Kết thúc một sự kiện.
Các luồng sự kiện con: S1: Ở bước 3 nếu chiều dài của mật khẩu hoặc tài khoản không hợp lệ sẽ báo lỗi
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.6. Sửa thông tin cá nhân

Tên use case: Sửa thông tin cá nhân	ID: UC-6
Actor chính: Researcher, Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm:	

Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn sửa thông tin cá nhân.
Mô tả tóm tắt: Khi tác nhân muốn sửa thông tin cá nhân của mình trong hệ thống.
Trigger: Khi tác nhân nhấp vào sửa thông tin cá nhân. Type: External
Các mối quan hệ: +Association (kết hợp): Researcher +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn sửa thông tin cá nhân 2. Hệ thống hiển thị thông tin cá nhân 3. Tác nhân sửa thông tin 4. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.7. Tạo loài mới

Tên use case: Tạo loài mới	ID: UC-7
Actor chính: Researcher, Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn tạo loài mới.	

Mô tả tóm tắt: Khi tác nhân muốn tạo loài mới của mình trong hệ thống.
Trigger: Khi tác nhân nhấp vào tạo loài mới. Type: External
Các mối quan hệ: +Association (kết hợp): Researcher +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn tạo loài mới 2. Hệ thống hiển thị ô để nhập thông tin loài 3. Tác nhân nhập vào thông tin loài 4. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.8. Xem các loài cần duyệt

Tên use case: Xem các loài cần duyệt	ID: UC-8
Actor chính: Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn xem các loài cần duyệt.	

Mô tả tóm tắt: Khi tác nhân muốn xem các loài cần duyệt trong hệ thống.
Trigger: Khi tác nhân nhấp vào xem các loài cần duyệt. Type: External
Các mối quan hệ: +Association (kết hợp): Expert +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn xem các loài cần duyệt 2. Hệ thống hiển thị danh sách các loài 3. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.9. Sửa các loài cần duyệt

Tên use case: Sửa các loài cần duyệt	ID: UC-9
Actor chính: Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn sửa các loài cần duyệt.	
Mô tả tóm tắt:	

<p>Khi tác nhân muốn sửa các loài cần duyệt trong hệ thống.</p>
<p>Trigger: Khi tác nhân nhấp vào sửa các loài cần duyệt.</p> <p>Type: External</p>
<p>Các mối quan hệ:</p> <p>+Association (kết hợp): Expert</p> <p>+Include(bao gồm): Không</p> <p>+Extend(mở rộng): Không</p> <p>+Generalization(tổng quát hóa): Không</p>
<p>Luồng xử lý bình thường của sự kiện:</p> <ol style="list-style-type: none"> 1. Tác nhân chọn sửa các loài cần duyệt 2. Hệ thống hiển thị loài 3. Tác nhân sửa loài 4. Kết thúc một sự kiện.
<p>Các luồng sự kiện con:</p>
<p>Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):</p>

10.10. Xem thống kê

Tên use case: Xem thống kê	ID: UC-10
Actor chính: Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
<p>Các thành phần tham gia và mối quan tâm:</p> <p>Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn xem thống kê.</p>	
<p>Mô tả tóm tắt:</p> <p>Khi tác nhân muốn xem thống kê trong hệ thống.</p>	

Trigger: Khi tác nhân nhấp vào xem thống kê. Type: External
Các mối quan hệ: +Association (kết hợp): Expert +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn xem thống kê 2. Hệ thống hiển thị thống kê loài 3. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.11. Xem danh sách Researcher

Tên use case: Xem danh sách Researcher	ID: UC-11
Actor chính: Expert	Mức độ cần thiết: Phải có Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn xem danh sách Researcher.	
Mô tả tóm tắt: Khi tác nhân muốn xem danh sách Researcher trong hệ thống.	

Trigger: Khi tác nhân nhấp vào xem danh sách Researcher. Type: External
Các mối quan hệ: +Association (kết hợp): Expert +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none"> 1. Tác nhân chọn xem danh sách Researcher 2. Hệ thống hiển thị danh sách 3. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

10.12. Tạo Researcher mới

Tên use case: Tạo Researcher mới	ID: UC-12
Actor chính: Expert	Mức độ cần thiết: Phải có
	Phân loại: Cao
Các thành phần tham gia và mối quan tâm: Xem danh sách chi tiết sản phẩm: Khi tác nhân muốn tạo Researcher mới.	
Mô tả tóm tắt: Khi tác nhân muốn tạo Researcher mới trong hệ thống.	
Trigger: Khi tác nhân nhấp vào tạo Researcher mới.	

Type: External
Các mối quan hệ: +Association (kết hợp): Expert +Include(bao gồm): Không +Extend(mở rộng): Không +Generalization(tổng quát hóa): Không
Luồng xử lý bình thường của sự kiện: <ol style="list-style-type: none">1. Tác nhân chọn tạo Researcher mới2. Hệ thống hiển thị form điền thông tin3. Tác nhân nhập vào thông tin4. Kết thúc một sự kiện.
Các luồng sự kiện con:
Luồng luân phiên/đặc biệt (Alternate/Exceptional flows):

11. Các yêu cầu phi chức năng

11.1. Yêu cầu thực thi

- Yêu cầu services:
 - + Thời gian phản hồi ngắn.
 - + Hoạt động 24/7.
 - + Đáp ứng được tối thiểu 1000 người dùng cùng lúc.
- Yêu cầu client:
 - + Có thể vận hành trên các trình duyệt như: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Coccoc, Opera,...
 - + Có mạng.

11.2. Yêu cầu an toàn

- Sao lưu dữ liệu định kỳ.
- Thay đổi mật khẩu quản trị định kỳ.

- Ngăn chặn tối đa các cuộc tấn công:
 - + Phishing.
 - + SQL Injection.
 - + Gian lận thanh toán.
 - + Spam.
 - + DDoS.
 - + Brute-force Attack.
 - + XSS.
- Tuân thủ các quy định của pháp luật, thông tư bộ công thương và nghị định Chính Phủ:
 - + Luật giao dịch điện tử 2005.
 - + Nghị định 52/2013/NĐ-CP.
 - + Thông tư 47/2014/TT-BCT.

11.3. Yêu cầu bảo mật

- Các chức năng được hiển thị dựa trên phân quyền của từng nhóm người sử dụng.
- Mã hóa JWT an toàn.
- Hạn chế tấn công Cross Site Scripting (XSS)
- Sử dụng giao thức HTTPS.
- Người dùng bình thường không thể truy cập trang quản lý.
- Sử dụng chứng chỉ mã hóa SSL.
- Mã hóa mật khẩu SHA-256.
- Không lưu giữ thông tin thẻ của người dùng.
- Sử dụng tường lửa.

11.4. Các đặc điểm chất lượng phần mềm

- Tài liệu của dự án được quản lý có hệ thống, sẵn sàng cung cấp cho khách hàng khi có yêu cầu.
- Hệ thống tiêu hao tối thiểu tài nguyên của người dùng.

- Sử dụng các quy trình mã hóa, khả năng xử lý và chịu lỗi, khả năng phục hồi và quản lý tài nguyên, đảm bảo tính toàn vẹn và thống nhất của dữ liệu.
- Phát triển các thành phần theo hướng dễ dàng chuyển đổi, bảo trì và phát triển trong tương lai.
- Dễ dàng kiểm thử.
- Giao diện đơn giản, trực quan, thân thiện với người dùng.
- Có các chức năng cần thiết nhằm thực hiện tối đa các yêu cầu của người dùng một cách chính xác và nhanh nhất.

Chương 2. Kiến Trúc Hệ Thống Microservices

1. Kiến thức nền

1.1. Định nghĩa về Microservices

Sau đây là một số định nghĩa của kiến trúc Microservices đã được đưa ra.

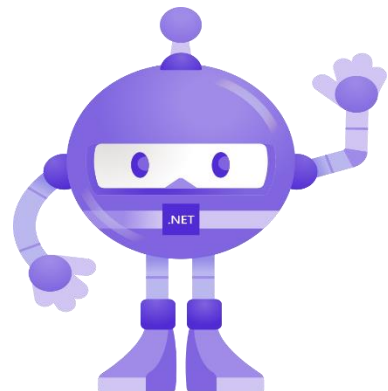
Định nghĩa 1: Đây là cách để phân tách một ứng dụng về mặt chức năng thành một tập hợp các dịch vụ cộng tác, mỗi dịch vụ có một tập hợp các chức năng hẹp, có liên quan, được phát triển và triển khai độc lập, với cơ sở dữ liệu riêng của nó. (Kharbuja, 2016)

Định nghĩa 2: Microservices là các phần chức năng riêng lẻ được phát triển, triển khai và quản lý độc lập bởi một nhóm nhỏ gồm những người thuộc các lĩnh vực khác nhau. (Goetsch, 2017)

Định nghĩa 3: Phong cách kiến trúc Microservices là một cách tiếp cận để phát triển một ứng dụng đơn lẻ như một tập hợp các dịch vụ nhỏ, mỗi dịch vụ chạy trong quy trình riêng và giao tiếp với các cơ chế nhẹ, thường là API tài nguyên HTTP. Các dịch vụ này được xây dựng dựa trên khả năng kinh doanh và có thể triển khai độc lập bằng máy móc triển khai hoàn toàn tự động. Quản lý tập trung các dịch vụ này ở mức tối thiểu, có thể được viết bằng các ngôn ngữ lập trình khác nhau và sử dụng các công nghệ lưu trữ dữ liệu khác nhau. (Lewis & Fowler, 2014)

1.2. .NET 6 LTS

.NET 6 Long-term support là phiên bản chính thức mới nhất của Microsoft. Tiền thân là .NET Core một nền tảng mã nguồn mở miễn phí trên các hệ điều hành Windows, Linux, và macOS. Đây là phiên bản đa nền tảng thừa kế từ .NET Framework. Dự án này chủ yếu được phát triển bởi các nhân viên Microsoft thuộc tổ chức .NET Foundation và được phát hành theo Giấy phép MIT. Đây là một trong các web framework nhanh nhất theo: <https://www.techempower.com/benchmarks/>



Hình 6 Linh vật .NET

.NET hỗ trợ đầy đủ cho C# và F# (và C++/CLI từ phiên bản 3.1; chỉ trên windows) và hỗ trợ Visual Basic .NET (từ phiên bản 15.5 trên .NET Core 5.0.100-preview.4, một số phiên bản trước của Visual Basic .NET cũng được hỗ trợ ở các bản .NET Core cũ hơn).

1.3. Blazor Web Assembly

Blazor là web framework mã nguồn mở và miễn phí cho phép các nhà phát triển tạo các ứng dụng web bằng C# và HTML. Nó đang được phát triển bởi Microsoft. Có 2 phiên bản chính của Blazor là:

Blazor Server: Các code sau khi biên dịch sẽ được chạy ở server và sau đó gửi đến client thông qua SignalR và giao tiếp với các thư viện DOM để điều khiển.

Blazor WebAssembly: Các code sau khi biên dịch thành các file dll sẽ được gửi đến client vào lúc client đến trang web, việc này làm cho việc lần đầu tải trang sẽ lâu hơn. Các thư viện dll thông qua Web Assembly điều khiển các DOM.

Blazor WebAssembly là một SPA (Single Page Application) tương tự như Angular, React hay là Vue. Sử dụng Blazor để tạo một trang web có một số lợi ích nhất định:

- Compile time checking, vì được viết bằng C# nên các lỗi sẽ được báo lúc biên dịch code thay vì lỗi sẽ phát sinh trong lúc chạy code.
- Không cần học thêm quá nhiều ngôn ngữ khác, có thể dùng C# thay thế cho JavaScript trong trang web.
- Thống nhất ngôn ngữ, như Angular, React, Vue sử dụng thường Nodejs làm server thì JavaScript có thể viết back-end lẫn front-end thì tương tự như vậy sử dụng Blazor có thể dùng C# viết code back-end lẫn front-end.



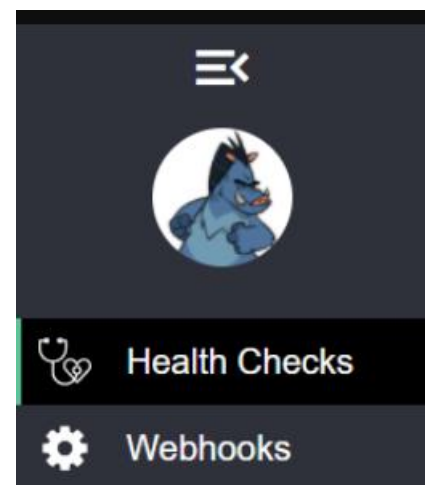
Hình 7 Logo Blazor

1.4. HealthChecks

ASP.NET Core cung cấp Health Checks Middleware và thư viện để báo cáo tình trạng của các thành phần đang hoạt động trong hệ thống.

<https://docs.microsoft.com/en-us/dotnet/architecture/microservices/implement-resilient-applications/monitor-app-health>

Kiểm tra sức khỏe được hiển thị bởi một ứng dụng dưới dạng HTTP endpoints. Các Health check endpoints có thể được định cấu hình cho các tình huống giám sát thời gian thực khác nhau:



Hình 8 Logo Watchdog Health Checks

- Bộ điều phối vùng chứa và bộ cân bằng tải có thể sử dụng các đầu dò sức khỏe để kiểm tra trạng thái của ứng dụng. Ví dụ: bộ điều phối vùng chứa có thể phản hồi khi kiểm tra tình trạng không thành công bằng cách tạm dừng triển khai luân phiên hoặc khởi động lại vùng chứa. Bộ cân bằng tải có thể phản ứng với một ứng dụng không khỏe bằng cách định tuyến lưu lượng truy cập từ service không được khỏe đến một service khác làm nhiệm vụ tương tự service cũ nhưng khỏe hơn.
- Có thể theo dõi việc sử dụng bộ nhớ, đĩa và các tài nguyên máy chủ vật lý khác để đảm bảo tình trạng hoạt động tốt.
- Kiểm tra sức khỏe có thể kiểm tra các kết nối ngoài của ứng dụng, chẳng hạn như cơ sở dữ liệu và hoặc service bên ngoài, để xác nhận tính khả dụng và hoạt động bình thường.

Kiểm tra sức khỏe thường được sử dụng với dịch vụ giám sát bên ngoài hoặc bộ điều phối vùng chứa để kiểm tra trạng thái của ứng dụng. Trước khi thêm kiểm tra sức khỏe vào ứng dụng, hãy quyết định sử dụng hệ thống giám sát nào. Hệ thống giám sát ra lệnh tạo loại kiểm tra sức khỏe nào và cách định cấu hình các điểm cuối của chúng.

1.5. Flask Framework



Hình 9 Logo Flask

Flask là micro web framework được viết bằng Python. Nó được phân loại là microframework vì nó không yêu cầu các công cụ hoặc thư viện cụ thể. Nó không có lớp trừu tượng cơ sở dữ liệu, xác thực biểu mẫu hoặc bất kỳ thành phần nào khác mà các thư viện bên thứ ba đã có từ trước cung cấp các chức năng phổ biến. Tuy nhiên, Flask hỗ trợ các tiện ích mở rộng có thể thêm các tính năng của ứng dụng như thể chúng được triển khai trong chính Flask. Các tiện ích mở rộng tồn tại cho người lập bản đồ quan hệ đối tượng, xác thực biểu mẫu, xử lý tải lên, các công nghệ xác thực người dùng khác nhau. Các ứng dụng sử dụng Flask bao gồm Pinterest và LinkedIn.

1.6. RabbitMQ

RabbitMQ là một chương trình phần mềm trung gian giúp các ứng dụng, hệ thống hay service khác nhau có thể giao tiếp, trao đổi dữ liệu với nhau. Nhiệm vụ của RabbitMQ được hiểu đơn giản là : nơi nhận các message (tin nhắn – thường dưới dạng json) từ các publisher (service gửi tin nhắn) sau đó chuyển đến các queue (hàng đợi) để lưu trữ, rồi các consumer (service xài tin nhắn đó) sẽ lấy các message ra sử dụng.



Hình 10 RabbitMQ

Là một Message Broker mã nguồn mở, dung lượng nhẹ, dễ dàng triển khai trên rất nhiều hệ điều hành lẫn Cloud, vì thế RabbitMQ vô cùng được ưa chuộng và trở nên phổ biến trong thời gian qua.

Không chỉ những doanh nghiệp nhỏ muốn tiết kiệm chi phí sử dụng, ngay cả những doanh nghiệp lớn, họ cũng đang sử dụng RabbitMQ cho công việc.

1.7. Docker

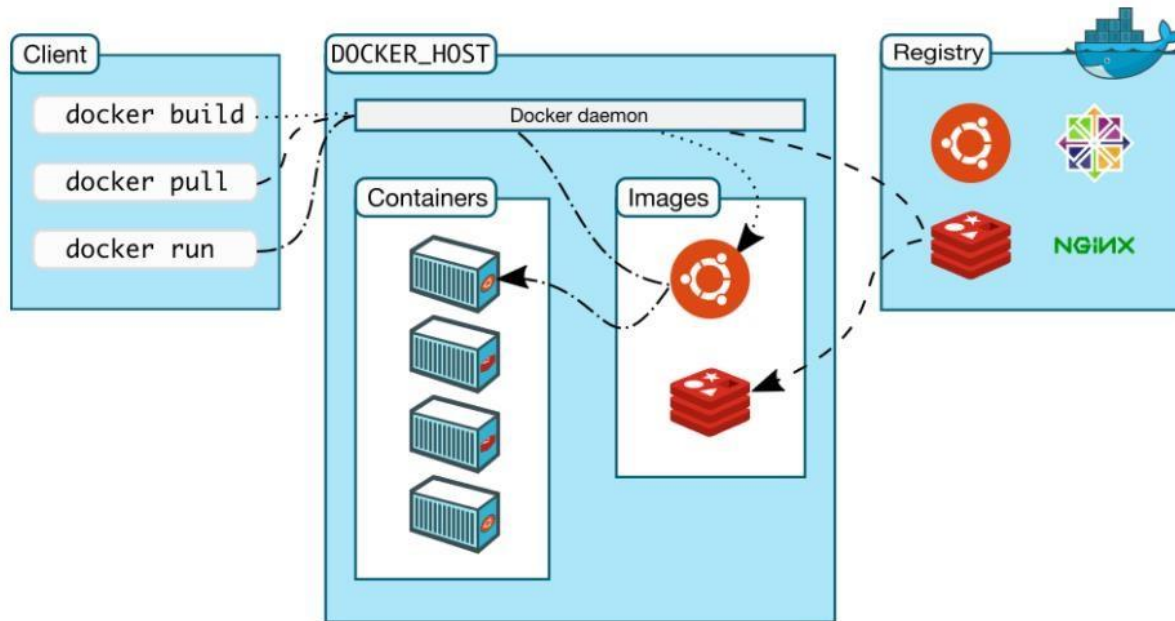


Hình 11 Docker

Docker là một nền tảng phần mềm cho phép bạn dựng, kiểm thử, triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm có thể chạy, trong đó có thư viện, công cụ hệ thống, mã thời gian chạy. Bằng cách sử dụng Docker, em có thể nhanh chóng triển khai và thay đổi quy mô ứng dụng vào bất kỳ môi trường nào và biết chắc rằng mã của em sẽ chạy được.

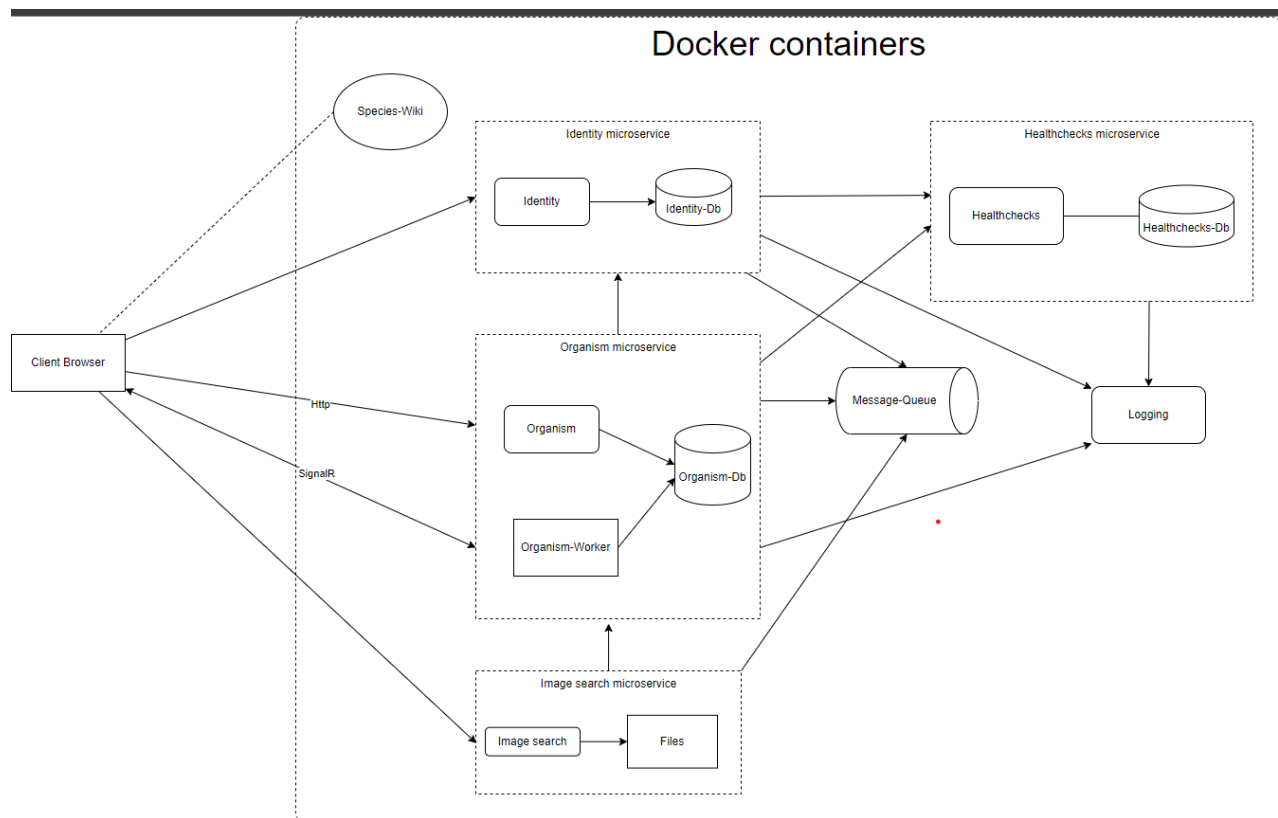
Docker hoạt động bằng cách cung cấp phương thức tiêu chuẩn để chạy mã. Docker là hệ điều hành dành cho container. Cũng tương tự như cách máy ảo ảo hóa (loại bỏ nhu cầu quản lý trực tiếp) phần cứng máy chủ, các container sẽ ảo hóa hệ điều hành của máy chủ. Docker được cài

đặt trên từng máy chủ và cung cấp các lệnh đơn giản mà bạn có thể sử dụng để dựng và khởi động hoặc dùng container.



Hình 12 Mô hình hoạt động của Docker

2. Xây dựng mô hình hệ thống



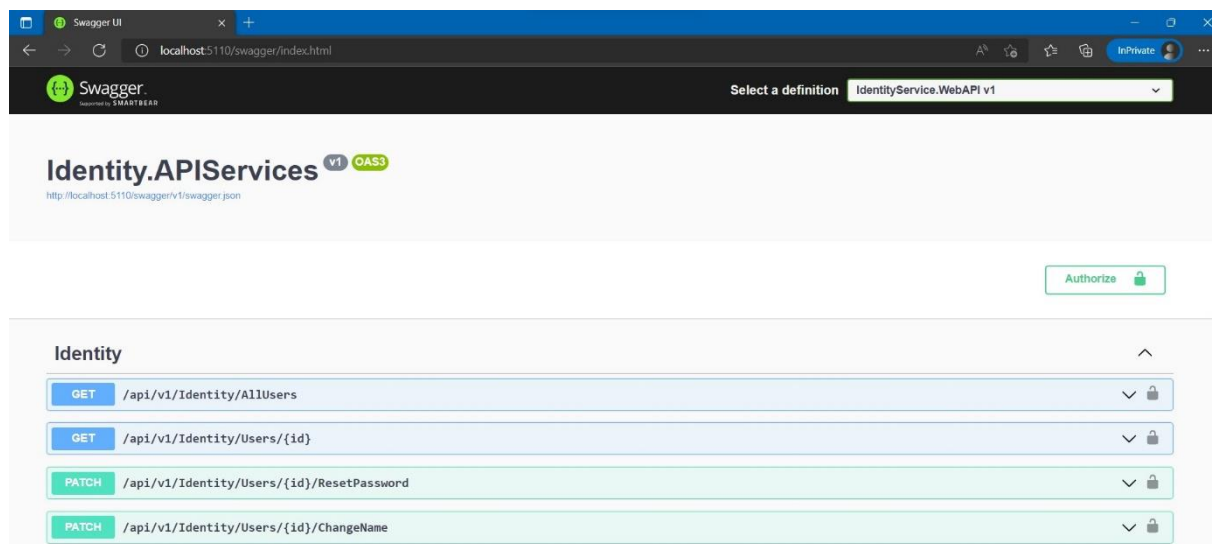
Hình 13 Kiến trúc tổng quan của hệ thống

Tất các services đều chạy dưới dạng một Docker container. Các services bao gồm:

1. Identity: một web service dùng để xác thực người dùng. Thông tin chi tiết sẽ được đề cập ở [chương 3](#).
2. Identity-Db: CSDL quan hệ của Identity.
3. Organism: Web service dùng để quản lý loài. Xem thêm ở [chương 4](#).
4. Organism-Db: CSDL phi quan hệ của Organism.

5. Organism-Worker: một worker service dùng để xử lý dữ liệu không tuần tự có chức năng lắng nghe các tin nhắn từ Message-Queue.
6. Image-Search service: web service dùng để tìm hình ảnh của loài. Thông tin thêm [chương 6](#).
7. Message-Queue: trung tâm trung chuyển các tin nhắn giữa các service với nhau sử dụng RabbitMQ. Thông tin chi tiết được đề cập ở [chương 5](#).
8. HealthChecks: giao diện web với chức năng kiểm tra sức khỏe của các service còn lại trong hệ thống. Thông tin chi tiết được đề cập ở [chương 5](#).
9. HealthChecks-Db: CSDL của HealthChecks.
10. Logging: Trung tâm lưu trữ các log của các service.
11. Species-Wiki: Một nginx service nhiệm vụ chính là gửi các static file của trang web Species-Wiki cho client.

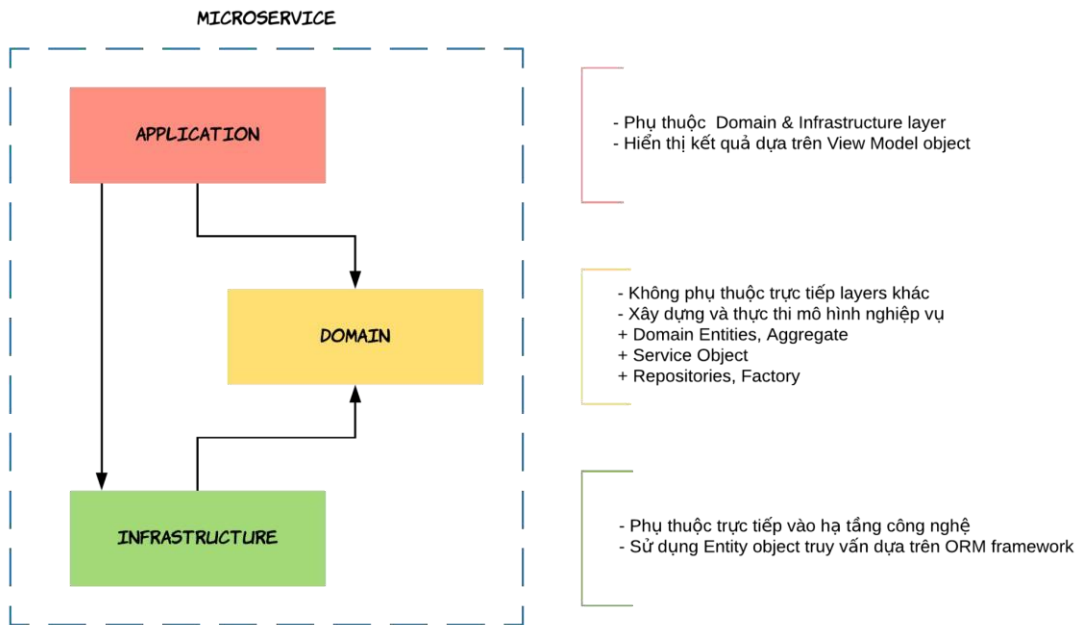
Chương 3. Identity Service



Hình 14 Identity REST API

1. Tổng quan service

Service dùng để xác thực người dùng sử dụng hệ thống sử dụng token. Được xây dựng trên .NET 6 một project WEB API theo mô hình MVC và design pattern DDD (Domain Driven Design), sử dụng Entity Framework để kết nối đến CSDL SQL Server. Kết nối ngoài với các service khác: RabbitMQ, Seq, HealthChecks.



Hình 15 Domain-Driven Design

Hệ thống hoạt động bao gồm các nhóm người dùng sau đây:

1. Researcher (Nhà nghiên cứu) là người dùng đã có tài khoản nhưng quyền hạn bị hạn chế.
2. Expert (Chuyên gia) là nhóm người dùng có toàn quyền trong hệ thống.

Các chức năng mà nhóm người dùng có thể sử dụng:

- Researcher:
 - Đăng nhập
 - Đổi tên
 - Đổi mật khẩu
 - Tạo token mới
 - Lấy thông đăng nhập
- Expert:
 - Lấy tất cả các người dùng
 - Lấy người dùng theo id

- Đổi tên của 1 người dùng theo id
- Reset mật khẩu 1 người dùng theo id
- Tạo tài khoản mới

2. Microsoft Authentication

Các project của C# có thêm phần xác thực người dùng nếu chọn trong lúc khởi tạo project nhưng có một hạn chế về cách xác thực tích hợp này. Một số loại xác thực có sẵn là:

Windows authentication: xác thực người dùng sử dụng email của Microsoft. Một số hạn chế là người dùng phải có email Microsoft và phải sử dụng Windows mới có thể sử dụng cách xác thực này. Trong hệ thống Microservices này sử dụng Docker và hệ điều hành là Linux nên sẽ khó để xác thực người dùng theo cách này.

Microsoft Identity Platform: xác thực người dùng bằng nền tảng của Microsoft, cụ thể ở đây là Azure. Sử dụng cách này thì yêu cầu phải có email Azure và có service đang chạy trên đó để xác thực. Điều này sẽ phải tốn tiền duy trì service trong khi phát triển phần mềm cũng như sẽ không hoạt động khi không kết nối được.

Individual Accounts: cái này thường được sử dụng nhiều hơn, xác thực bằng cách này phải có kết nối đến một sở dữ liệu và từ đó Entity Framework sẽ tạo ra các bảng cần thiết để đăng nhập. Ưu điểm là dữ liệu của mình sẽ được lưu nội bộ và còn có sẵn giao diện để đăng nhập cũng như là chỉnh sửa thông tin các nhân.

3. Các hạn chế của Microsoft Authentication

Xác thực bằng phương pháp Individual Accounts nghe có vẻ rất tiện lợi như có một số lý do em không chọn cái này cho project:

Xác thực người dùng sử dụng cookie: khi đăng nhập vào hệ thống thì người dùng sẽ được cung cấp cookie để duy trì trạng thái đăng nhập mỗi lần gọi server sẽ tự thêm cookie và header. Có một điểm yếu trong cách xác thực này trong hệ thống Microservices, các cookie chỉ hoạt động khi có cùng tên miền liên quan đến cookie đó và khi đổi tên miền thì sẽ không thể nào mà xác thực được nữa, đó là một điểm yếu của cookie vì trong Microservices các service khác nhau sẽ có các tên miền khác nhau và điều này sẽ gây khó khăn trong việc xác thực. Và khi muốn mở rộng trên nền tảng di động sẽ không hoạt động được vì cookie chỉ có ở trên web.

Khó chỉnh sửa các trang web: từ .Net core 2.1 các trang đăng nhập cũng như là chỉnh sửa thông tin sẽ trở thành Razor class UI và được đóng gói trong nuget package

(<https://devblogs.microsoft.com/dotnet/aspnetcore-2-1-identity-ui/>) việc này sẽ khiến các giao diện khó để chỉnh sửa hơn làm giảm việc trải nghiệm của người dùng.

Lưu trữ dữ liệu: Individual Accounts sử dụng Entity Framework để kết nối các cơ sở dữ liệu, Entity Framework là chỉ có thể kết nối đến các cơ sở dữ liệu quan hệ như: SQL Server, SQL LocalDb, SQLite, Và khi chạy Entity Framework sẽ tự tạo các bảng trong sở dữ liệu, với nghiệp vụ hiện tại của hệ thống thì việc tạo nhiều bảng khiến hao bộ nhớ để lưu trữ không cần thiết.

Học sử dụng: việc sử dụng hệ thống có sẵn tất nhiên sẽ phải tốn rất nhiều thời gian tìm tòi các tài liệu về cách sử dụng, sẽ phải hiểu các thư viện cũng như là cấu hình để sử dụng. Vì một số hạn chế của việc xác thực tích hợp của Microsoft nên em đã xây dựng một service riêng dành cho việc xác thực người dùng. Là một web service cung cấp một số các chức năng cơ bản để xác thực người dùng và thay vì sử dụng cookie, em sẽ dùng Json Web Token (JWT).

4. Xác thực bằng token:



Hình 16 JWT

Json Web Token (<https://jwt.io/>): sử dụng jwt sẽ có một số lợi thế nhất định so với cookie. Jwt sẽ có bảo mật cao hơn cookie vì được mã hóa với khóa điều này khiến cho các hacker khi lấy được token sẽ không thể chỉnh sửa token để dùng CSRF (Cross-site request forgery) tấn công người dùng (https://en.wikipedia.org/wiki/Cross-site_request_forgery). Và nếu hacker có được jwt thì jwt cũng có thời hạn ngắn từ 30p đến 1h nên sẽ hacker sẽ không gây ra quá nhiều rắc rối cho hệ thống. Khi đăng nhập hệ thống sẽ trả về jwt token cho người dùng, và khi người dùng sử dụng các service cần đến việc xác thực, ở đây là các REST api endpoints thì với mỗi request đến endpoint trong phần header đều phải chèn vào jwt token. Các RESTful api thường là stateless có

nghĩa là sẽ không có lưu dữ liệu bên api mà là trong CSDL, việc sử dụng jwt các thông tin người dùng đã có sẵn trong jwt nên chỉ cần lấy ra thông tin bên trong jwt.

Refresh token: khác với jwt thì refresh token là một chuỗi mà hóa được tạo ra nhằm mục đích cung cấp cho người dùng một jwt mới, vì theo đề cập bên trên thì jwt có hạn sử dụng ngắn cho nên khi hết hạn người dùng sẽ phải tạo token mới mà việc tạo token mới đòi hỏi người dùng phải đăng nhập vào hệ thống, điều này sẽ làm giảm trải nghiệm của người dùng đi rất nhiều. Vì thế refresh token được tạo ra để cải thiện việc này, thay vì mỗi lúc jwt hết hạn người dùng lại phải đăng nhập lại, các service nào có nhu cầu sử dụng identity service sẽ dùng refresh token để gọi identity service tạo một jwt cho người dùng, từ đó dùng jwt mới để duy trì trạng thái đăng nhập. Refresh token có thời gian sử dụng lâu hơn với jwt token từ vài tuần đến vài tháng.

Lưu trữ các token: jwt token được tạo ra rồi cung cấp cho người dùng và không có lưu lại vào trong CSDL, khi người dùng làm mất thì chỉ cần tạo token mới. Refresh token thì ngược lại vì hạn sử dụng lâu nên sẽ phải lưu lại vào CSDL, refresh token là chuỗi do mình tự đặt ra nên có thể là bất cứ gì ví dụ như là: GUI của C# để tiện trong việc truy vấn, nhưng ở đây em sử dụng luôn jwt token là một refresh token. Khi các service khác gọi identity service để đăng nhập, khi thành công sẽ trả về 2 token là jwt token và refresh token và cả 2 đều là jwt token. Và service khác lưu lại 2 token, species wiki service là một web spa nên các token sẽ được lưu vào local storage và nếu muốn thêm di động thì sẽ lưu vào file như SQLite.

Bảo mật các token: jwt token có hạn sử dụng ngắn để bảo mật, còn đối với refresh token hạn sử dụng dài thì mỗi lúc đăng nhập hệ thống sẽ tạo một refresh token mới và bỏ token cũ nên các hacker sẽ không sử dụng refresh token cũ để tạo jwt token mới.

5. Bảo mật

Để tăng tính bảo mật để phòng trường hợp hacker thâm nhập vào được hệ thống và truy xuất CSDL thì việc mã hóa mật khẩu là một điều cần thiết. Mật khẩu của người dùng sẽ được mã hóa 1 chiều sử dụng thuật toán SHA256 và hash cũng như salt sẽ được lưu vào CSDL.

6. Các REST API quan trọng

6.1. Refresh token

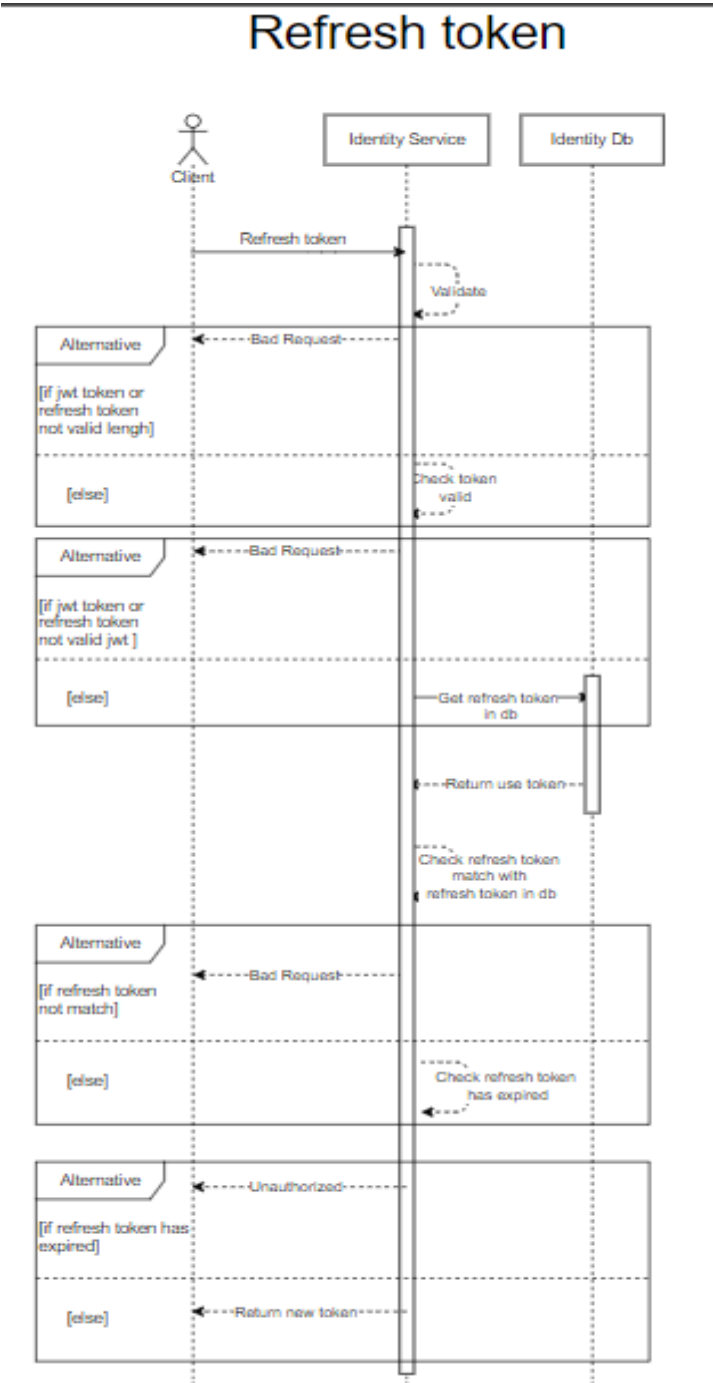
Tên yêu cầu	Tạo một token mới
Mục đích	Chức năng này tạo một token mới từ refresh token
URI	/api/v1/Identity/RefreshToken

Param	Không
Method	POST
Header	<ul style="list-style-type: none"> • accept: */* • Content-Type: application/json
Body	<pre>{ "jwtToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJuYW1laWQiOiJjNWl2ZjZjNi1mYT RilTRlNWEtYTJhMi0wYzc4MDIzMdQ2NzEiLCJuYW1lIjo iQWxleCBXYW5nIiwiaWF0IjoiJHbGV4QGdtYWlsLmNvbSIsIm5iZiI6MTY1MTkwNzEyOSwiZXhwIjoxNjUxOTA4OTI5LCJpYXQiOiJ E2NTE5MDcxMjl9.vRh3RIsytbHMG3T-58AILwuUn6FnGz-PMrmoI57ukL0", "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJVc2VySWQiOiJjNWl2ZjZjNi1mYT RilTRlNWEtYTJhMi0wYzc4MDIzMdQ2NzEiLCJuYmYiOiJ E2NTE5MDcxMjksImV4cCI6MTY1NDQ5OTEyOSwiaWF0IjoxNjUxOTA3MTI5fQ.okfpS3ZigJH3Qi5VLsaXBtY-8kfln-vQurKsEq-1KFQ" }</pre>
Response	<ol style="list-style-type: none"> 400 – Bad Request <pre>[{ "id": "695fc800-deee-4b4d-9ddc-491f249d1ad0", "message": "The field JwtToken must be a string with a minimum length of 50 and a maximum length of 2147483647.", "atTime": "2022-05-07T07:02:33.2011043+00:00" }, { "id": "bd9c5647-d640-44e1-9d68-3cfb708f5603", "message": "The field RefreshToken must be a string with a minimum length of 50 and a maximum length of 2147483647.", "atTime": "2022-05-07T07:02:33.2012639+00:00" }]</pre> 400 – Bad Request (không phải jwt) <pre>[{</pre>

	<pre>"id": "03ad0082-a262-4bad-9a65-2275ac0ef776", "message": "JwtToken:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiJjNWl2ZjZjNi1mYTRILTRlNWEtYTJhMi0wYzc4MDIzMDQ2NzEiLCJuYWl1IjojQWxleCBXYW5nIiwiaWZlhaWwiOiJhbGV4QGdtYWlsLmNvbSIsIm5iZiI6MTY1MTkwNzEyOSwiZ XhwIjojNjUxOTA4OTI5LCJpYXQiOiJlE2NTE5MDcxMjI9.vRh3 RIsytbHMG3T-58AILwuUn6FnGz-PMrmoI57ukLa not valid", "atTime": "2022-05-07T07:08:52.961783+00:00" }]</pre>
3.	<p>400 – Bad Request (Refresh token không tồn tại)</p> <pre>[{ "id": "2f4bbe43-24ee-4a53-80cf-38b66055148d", "message": "RefreshToken:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiJjNWl2ZjZjNi1mYTRILTRlNWEtYTJhMi0wYzc4MDIzMDQ2NzEiLCJuYmYiOiJlE2NDkxMTk1NTIsImV4cCI6MTY1MTE5MzE1MiwiWF0IjojNjUxOTA4OTI5LCJpYXQiOiJlE2NTE5MTE5NTUyYyQwbWJrPvpyub0EX9Ft2ia3s6NKeAXjoKoFtqT99cPwC8 not exist.", "atTime": "2022-05-07T07:35:21.5385318+00:00" }]</pre>
4.	<p>401 – Unauthorized (Refresh token hết hạn)</p> <pre>[{ "id": "3cc70687-f16f-413a-a185-72d02be1bd08", "message": "RefreshToken:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiJjNWl2ZjZjNi1mYTRILTRlNWEtYTJhMi0wYzc4MDIzMDQ2NzEiLCJuYmYiOiJlE2NDkxMTk1NTIsImV4cCI6MTY1MTE5MzE1MiwiWF0IjojNjUxOTA4OTI5LCJpYXQiOiJlE2NTE5MTE5NTUyYyQwbWJrPvpyub0EX9Ft2ia3s6NKeAXjoKoFtqT99cPwC8 has expired.", "atTime": "2022-05-07T07:42:44.3532824+00:00" }]</pre>

	<p>5. 200 – Ok</p> <pre>{ "jwtToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiJjNWl2ZjZjNi1mYTRlLTRlNWEtYTJhMi0wYzcyMDIzMDQ2NzEiLCJuYW1lIjoieWwleCBXYW5nIiwiaWF0IjoiIjBhGV4QGdtYWlsLmNvbSI6Im5iZiI6MTY1MTkwNzEyOSwiZmxwIjoxNjUxOTA4OTI5LCJpYXQiOiE2NTE5MDcxMjI9.vRh3RIsytbHMG3T-58AILwuUn6FnGz-PMrmoI57ukL0" }</pre>
--	---

Bảng 2 REST API Refresh token



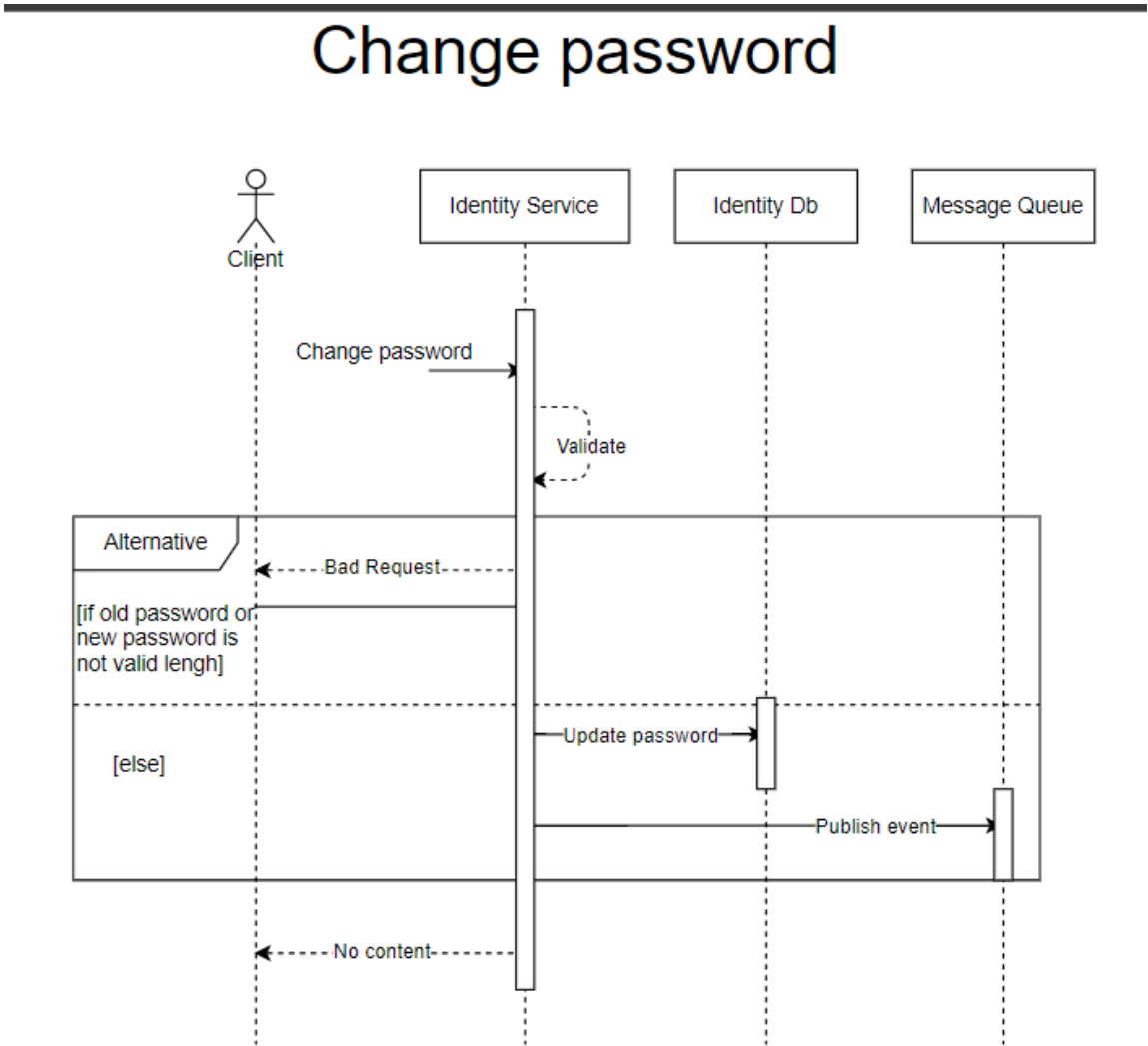
Hình 17 Sequence Diagram Refresh token

6.2. Change password

Tên yêu cầu	Người dùng đổi mật khẩu
Mục đích	Chức năng này cho phép người dùng đổi mật khẩu
URI	/api/v1/Identity/ChangePassword
Param	Không
Method	PATCH
Header	<ul style="list-style-type: none"> • accept: */* • Content-Type: application/json
Body	<pre>{ "oldPassword": "123456", "newPassword": "123456", "confirmNewPassword": "123456" }</pre>
Response	<p>1. 400 – Bad Request</p> <pre>{ "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "00-bfe8a8f5fd64238b2acf5607218d76c4-61114f53c6d6ee52-00", "errors": { "NewPassword": ["The field NewPassword must be a string with a minimum length of 6 and a maximum length of 100."], "OldPassword": ["The field OldPassword must be a string with a minimum length of 6 and a maximum length of 100."], "ConfirmNewPassword": ["The field ConfirmNewPassword must be a string with a minimum length of 6 and a maximum length of 100."] } }</pre>

	2. 204 – No Content
--	---------------------

Bảng 3 REST API Change password



Hình 18 Sequence Diagram Change password

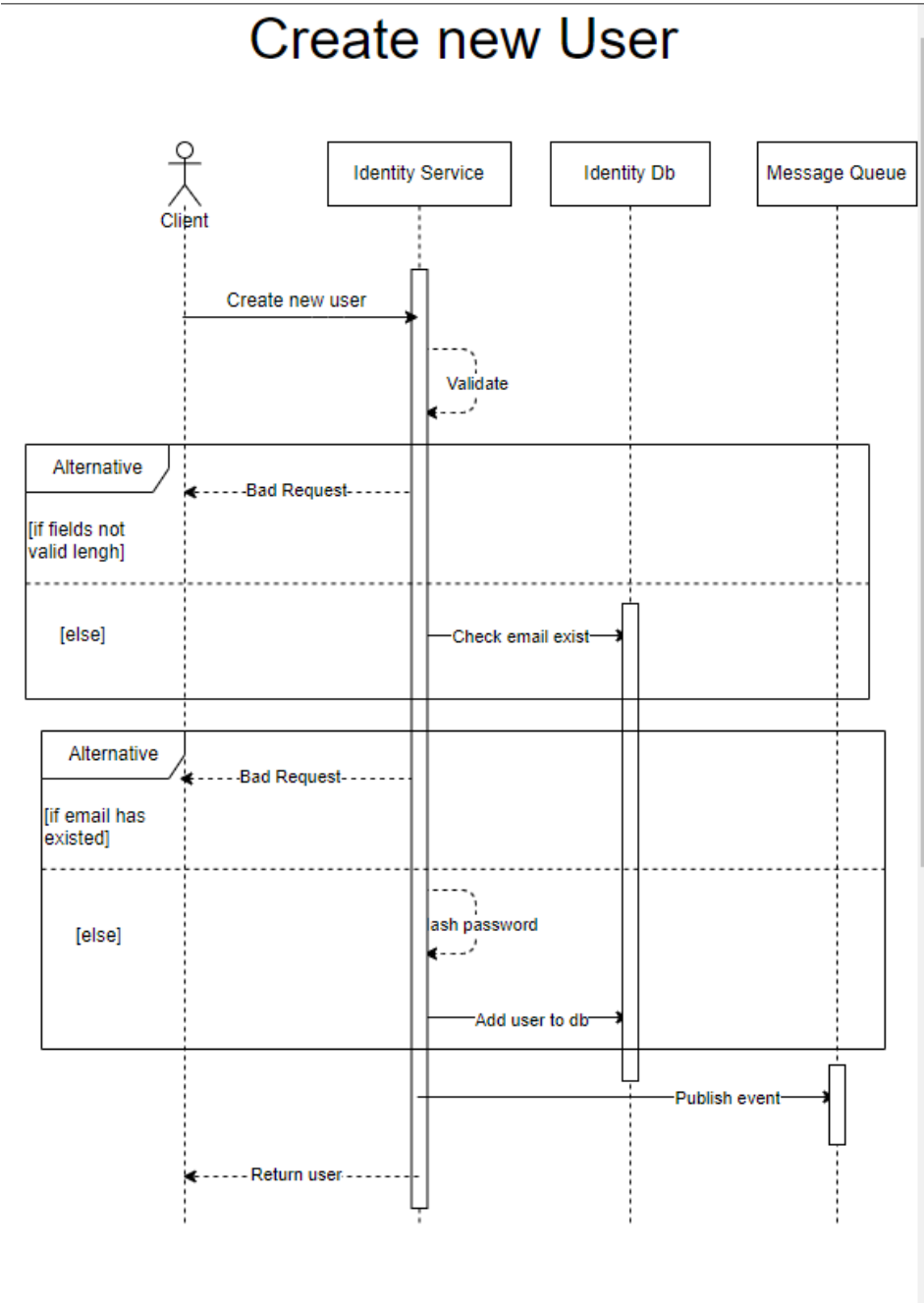
6.3. Create User

Tên yêu cầu	Người dùng tạo tài khoản mới
-------------	------------------------------

Mục đích	Chức năng này cho phép người dùng tạo tài khoản mới
URI	/api/v1/Identity/Register
Param	Không
Method	POST
Header	<ul style="list-style-type: none"> • accept: */* • Content-Type: application/json
Body	<pre>{ "firstName": "string", "lastName": "string", "email": "user@example.com", "password": "string" }</pre>
Response	<p>1. 400 – Bad Request</p> <pre>{ "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "00-efaf09aab63ed91855d90252677c539d-8a81a3ed4e5c6734-00", "errors": { "Email": ["The Email field is not a valid e-mail address."], "LastName": ["The field LastName must be a string with a minimum length of 4 and a maximum length of 50."], "Password": ["The field Password must be a string with a minimum length of 6 and a maximum length of 100."], "FirstName": ["The field FirstName must be a string with a minimum length of 4 and a maximum length of 50."] } }</pre>

	<pre>} 2. 400 – Bad Request [{ "id": "d81633c6-3fde-4e0b-9a7c-0d90e08b1f73", "message": "Email:user@gmail.com has been taken", "atTime": "2022-05-07T08:20:37.6700647+00:00" }] 3. 201 – Created { "id": "c5b6f6c6-fa4e-4e5a-a2a2-0c7802304671", "name": "String String", "email": "user@gmail.com" }</pre>
--	---

Bảng 4 REST API Create user

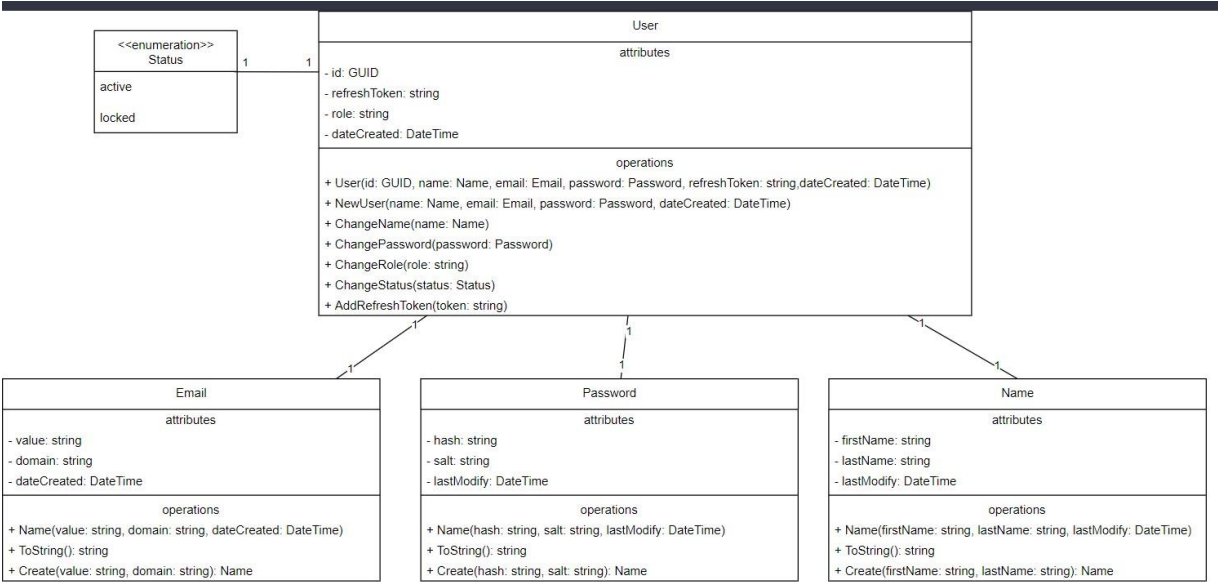


Hình 19 Sequence Diagram Create new user

7. Thiết kế dữ liệu

7.1. Sơ đồ lớp

Dữ liệu được thiết kế theo DDD (Domain Driven Design)



Hình 20 Identity Class Diagram

7.2. Từ điển dữ liệu

STT	Tên thuộc tính	Kiểu	Khóa chính	Duy nhất	Diễn giải
1	Id	UUID	X	X	Mã người dùng
2	Name_FirstName	NVARCHAR(100)			Tên người dùng
3	Name_LastName	NVARCHAR(100)			Họ người dùng
4	Name_LastModify	DATETIME2(7)			Thời gian sửa
5	Email_Value	VARCHAR(MAX)		X	Email của người dùng
6	Email_DateCreated	DATETIME2(7)			Email ngày tạo
7	Password_Hash	VARCHAR(500)			Mật khẩu hash
8	Password_Salt	VARCHAR(500)			Salt cho hash của mật khẩu
9	Password_LastModify	DATETIME2(7)			Thời gian sửa
10	RefreshToken	VARCHAR(500)			
11	Role	VARCHAR(30)			Expert, Researcher

12	Status	VARCHAR(30)			Locked, Active
13	DateCreated	DATETIME2(7)			

Bảng 5 Bảng người dùng

8. Giao tiếp ngoài

Identity Service phụ thuộc vào Identity Mssql vì là cơ sở dữ liệu của service ngoài ra còn có RabbitMq nơi mà identity sẽ gửi thông báo nếu có dữ liệu thay đổi trong identity. Các log của identity cũng sẽ được gửi đến Seq service để lưu trữ.

8.1. Identity Mssql

Identity Mssql là một cơ sở dữ liệu SQL Server, đây sẽ là cơ sở dữ liệu của identity service. Identity sẽ sử dụng Entity Framework Core để truy xuất dữ liệu. Identity service phụ thuộc vào service này và sẽ không thể hoạt động nếu identity-mssql không hoạt động.

8.2. RabbitMq

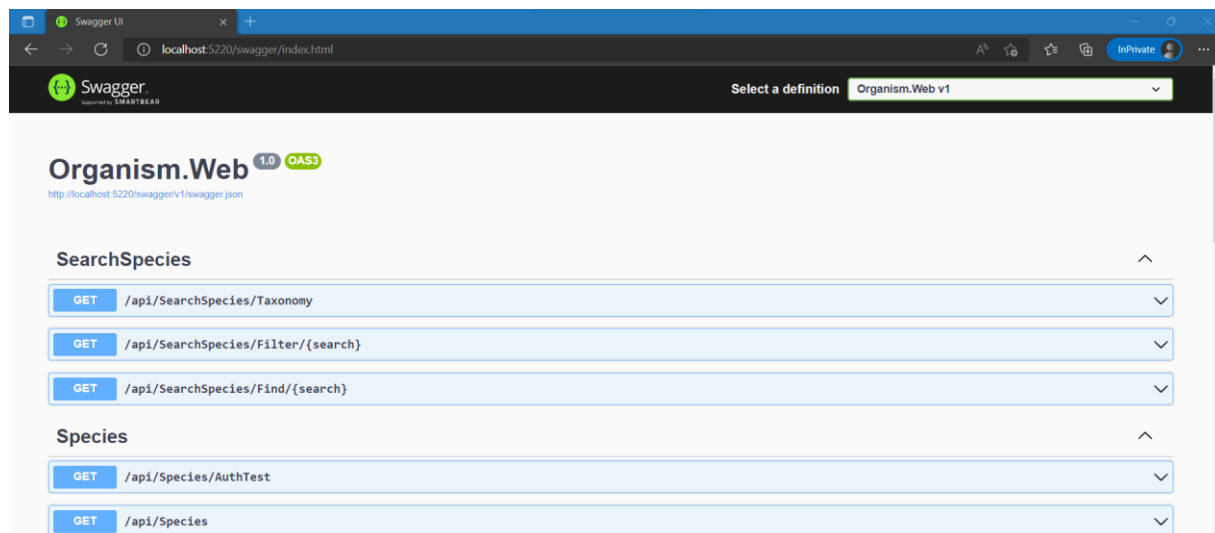
RabbitMq service là nơi mà identity sẽ gửi đi các thông báo nếu các dữ liệu bên trong bị thay đổi. Điều này sẽ giúp bảo vệ tính toàn vẹn của dữ liệu trong các service khác phụ thuộc vào identity.

Vì chỉ dùng để gửi thông báo nên khi RabbitMq không hoạt động sẽ làm ảnh hưởng đến một phần các thao tác chỉnh sửa dữ liệu, các thao tác lấy dữ liệu sẽ không thay đổi.

8.3. Seq

Seq service là nơi identity sẽ gửi các log của mình đến. Identity vẫn sẽ hoạt động bình thường nếu không có Seq service.

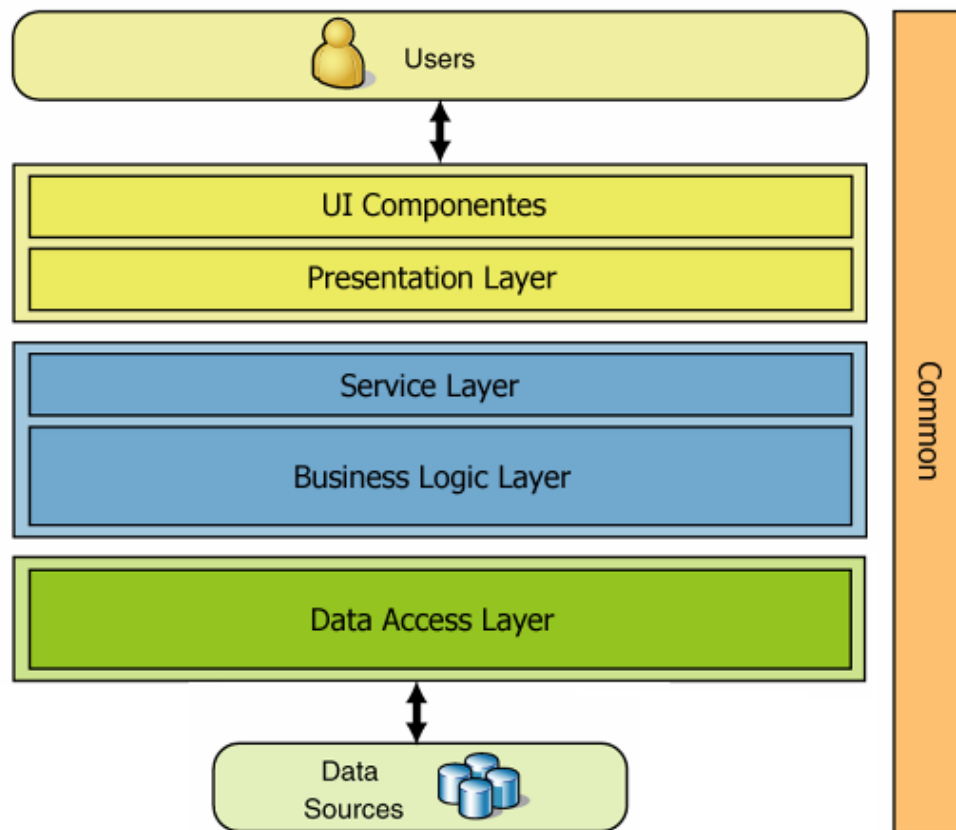
Chương 4. Organism Service



Hình 21 Organism REST API

1. Tổng quan service

Service dùng để quản lý các loài của người dùng sử dụng hệ thống, đăng nhập sử dụng token bên identity. Được xây dựng trên .NET 6 một project WEB API theo mô hình MVC và kiến trúc N-layer, kết nối ngoài với các service khác: RabbitMQ, Seq, HealthChecks.



Hình 22 Kiến trúc N-Layer

Hệ thống hoạt động bao gồm các nhóm người dùng sau đây:

1. Khách - Guest
2. Nhà nghiên cứu - Researcher
3. Chuyên gia - Expert

Các chức năng mà nhóm người dùng có thể sử dụng:

- Guest:
 - Lấy tất cả loài
 - Lấy loài theo id
 - Lấy loài của tác giả
 - Tìm loài theo tên
- Researcher:

- Lấy tất cả loài
- Lấy loài theo id
- Lấy loài của tác giả
- Tìm loài theo tên
- Thêm loài mới
- Sửa thông tin loài
- Expert:
 - Lấy tất cả loài
 - Lấy loài theo id
 - Lấy loài của tác giả
 - Tìm loài theo tên
 - Thêm loài mới
 - Sửa thông tin loài
 - Duyệt loài
 - Lấy thống kê thông tin loài

2. Tìm kiếm loài

Đầu tiên phải tạo index, việc tạo index sẽ bao gồm duyệt qua các trường trong tập dữ liệu, bỏ dấu các từ ví dụ như các dấu được đặt ở trên hoặc dưới các chữ cái, chẳng hạn như é, à, và ç trong tiếng Pháp. Và dựa trên ngôn ngữ được sử dụng, các thuật toán sẽ loại bỏ các từ bổ sung và chỉ giữ lại từ gốc. Trong tiếng anh từ “to eat,” “eating”, and “ate” sẽ chuyển thành “eat”. Và rồi các từ sẽ được chuyển thành viết thường. Index được tạo bằng cách thêm mỗi từ này với một tham chiếu đến tài liệu mà nó có thể được tìm thấy trong đó.

Việc tìm kiếm loài theo tên có áp dụng máy học vào để tăng sự chính xác trong việc tìm kiếm. Em sẽ áp dụng tính năng full-text search bằng cách tạo index. Khi khởi chạy service, hệ thống sẽ tạo index cho việc tìm kiếm theo tên, tìm kiếm theo tên sẽ bao gồm tên chính thức, tên địa phương và tên khoa học. Khi sử dụng tìm kiếm theo tên đầu vào có thể dùng 3 loại tên đều được.

3. Các REST API quan trọng

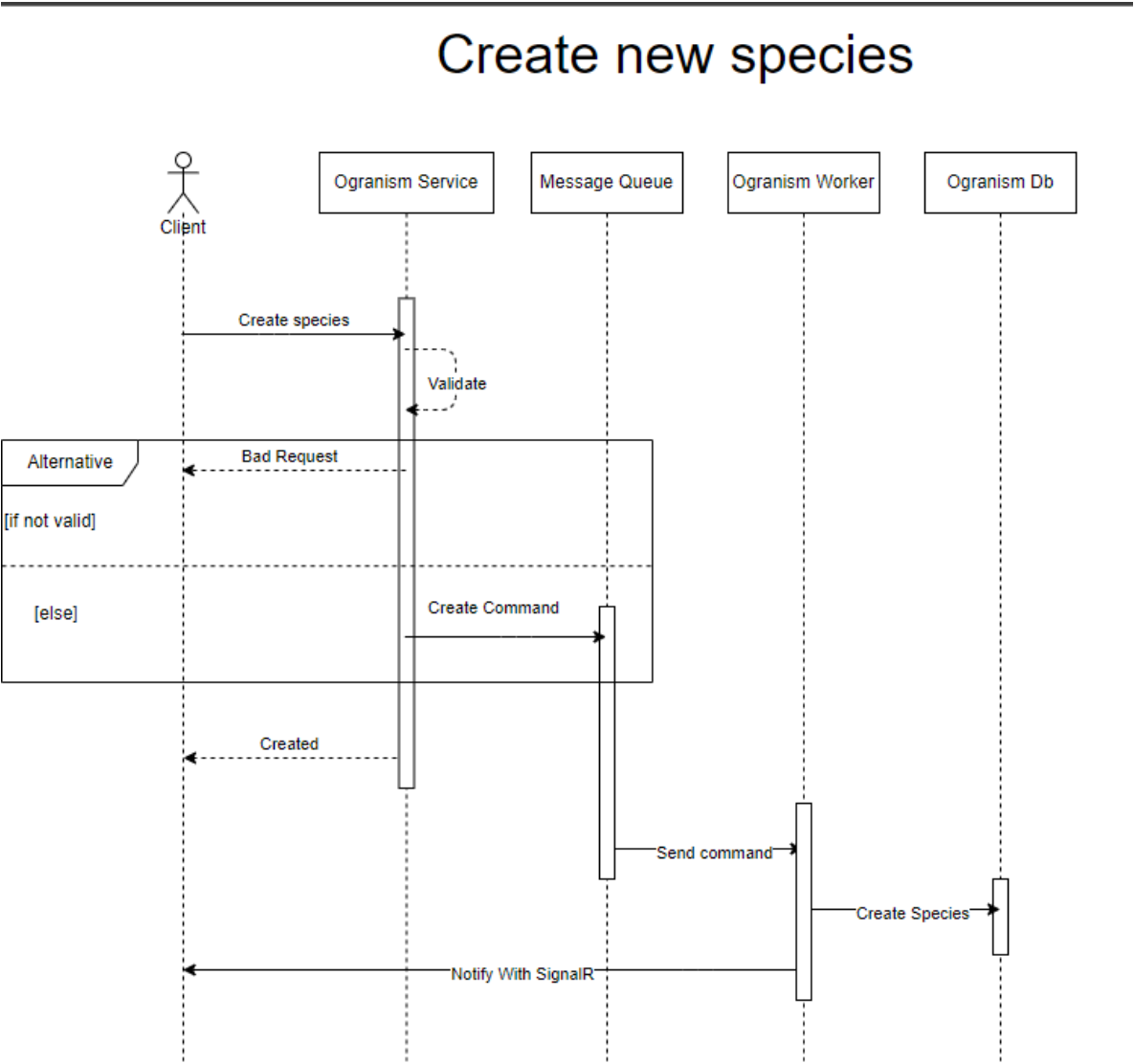
3.1. Tạo loài mới

Tên yêu cầu	Tạo loài mới
Mục đích	Chức năng này cho phép người dùng tạo một loài mới.
URI	/api/Species
Param	Không
Method	POST
Header	<ul style="list-style-type: none"> • accept: */* • Content-Type: application/json
Body	<pre>{ "name": "string", "localName": "string", "images": ["string"], "taxonomy": { "kingdom": "string", "phylum": "string", "class": "string", "order": "string", "family": "string", "genus": "string", "species": "string" }, "description": { "morphology": "stringstringstringstringstring", "ecosystem": "stringstringstringstringstring", "useValue": "string" }, "distribution": [[0]], "moreInfo": { "habitat": "stringstri", "place": "stringstri", </pre>

	<pre>"conservation": { "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" } } }</pre>
Response	<pre>1. 400 - Bad Request { "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "00-3bba7d14ca8f39edd1dc2d69d18c5f61- 77fe7c3a01b5b951-00", "errors": { "Name": ["The field Name is invalid."] } }</pre>

	<pre> 2. 201 – Created { "id": "625579b1d7793c3b2a32496f", "name": "string", "localName": "string", "images": ["string"], "taxonomy": { "kingdom": "string", "phylum": "string", "class": "string", "order": "string", "family": "string", "genus": "string", "species": "string" }, "description": { "morphology": "stringstringstringstringstring", "ecosystem": "stringstringstringstringstring", "useValue": "string" }, "distribution": [[0]], "moreInfo": { "habitat": "stringstri", "place": "stringstri", "conservation": { "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" } } } </pre>
--	--

Bảng 6 REST API Tạo loài mới



Hình 23 Sequence Diagram Create new species

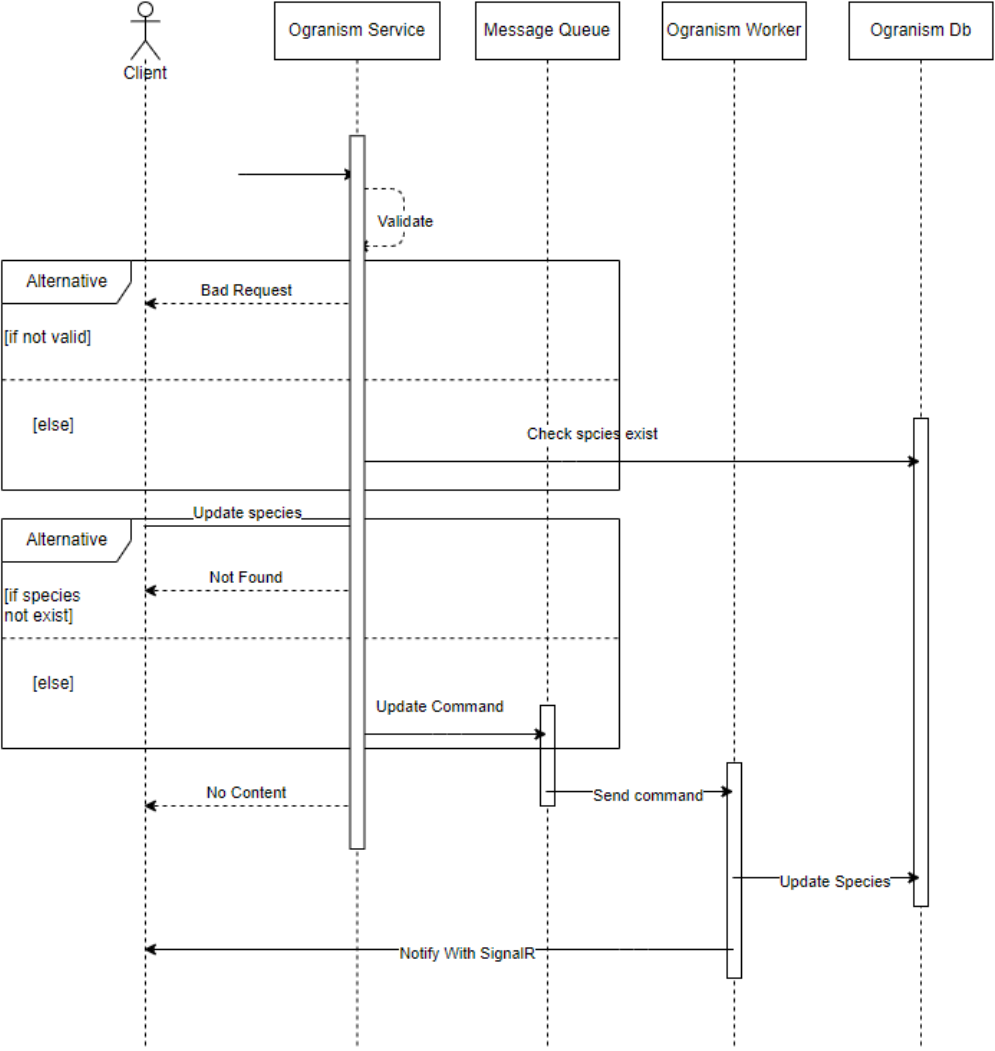
3.2. Cập nhật loài

Tên yêu cầu	Cập nhật loài
Mục đích	Chức năng này cho phép người dùng cập nhật loài.
URI	/api/Species/{id}
Param	Id string
Method	PUT
Header	<ul style="list-style-type: none"> • accept: */* • Content-Type: application/json
Body	<pre>{ "name": "string", "localName": "string", "images": ["string"], "taxonomy": { "kingdom": "string", "phylum": "string", "class": "string", "order": "string", "family": "string", "genus": "string", "species": "string" }, "description": { "morphology": "stringstringstringstringstring", "ecosystem": "stringstringstringstringstring", "useValue": "string" }, "distribution": [[0]], }</pre>

	<pre> "moreInfo": { "habitat": "stringstri", "place": "stringstri", "conservation": { "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" } } </pre>
Response	<p>1. 400 - Bad Request</p> <pre> { "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "00-3bba7d14ca8f39edd1dc2d69d18c5f61-77fe7c3a01b5b951-00", "errors": { "Name": ["The field Name is invalid."] } } </pre> <p>2. 204 – No Content</p>

Bảng 7 REST API Cập nhật loài

Update species



Hình 24 Sequence Diagram Update species

3.3. Tìm loài theo tên

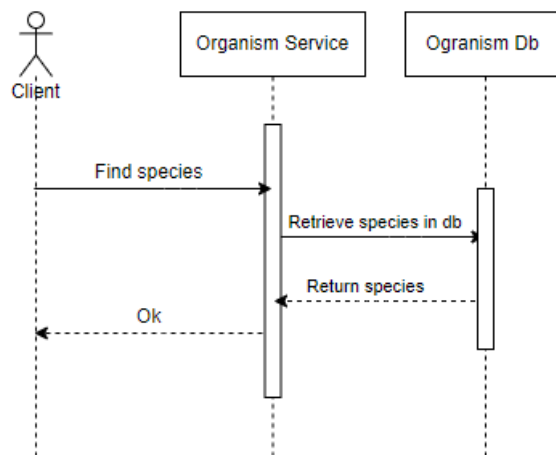
Tên yêu cầu	Tìm loài theo tên
Mục đích	Chức năng này cho phép tìm các loài đã có trong hệ thống theo tên.
URI	/api/ Species/Find/{search}
Param	search string
Method	GET
Header	Không
Body	Không
Response	200 – Ok

	<pre>{ "id": "string", "name": "string", "localName": "string", "taxonomy": { "kingdom": "string", "phylum": "string", "class": "string", "order": "string", "family": "string", "genus": "string", "species": "string" }, "images": ["string"], "description": { "morphology": "stringstringstringstringstring", "ecosystem": "stringstringstringstringstring", "useValue": "string" }, "distribution": [[0]], "moreInfo": { "habitat": "stringstri", "place": "stringstri", "conservation": { "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" } }, "author": { "id": "string",</pre>
--	--

	<pre> "email": "string" }, "timestamp": "2022-05-13T11:41:50.147Z", "status": "string" } </pre>
--	---

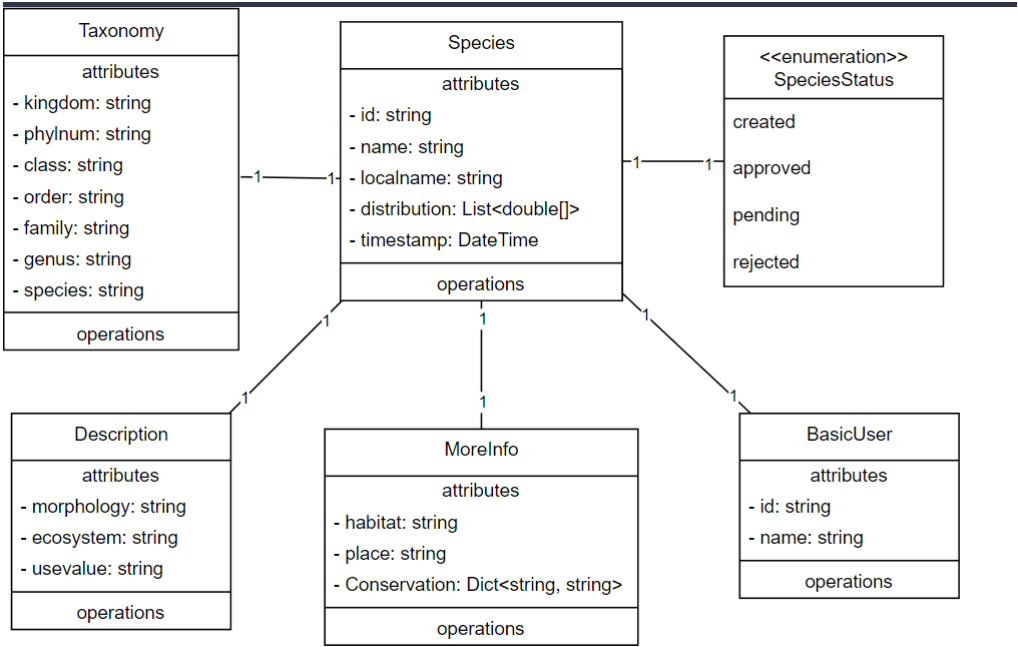
Bảng 8 REST API Tìm loài theo tên

Find species by name



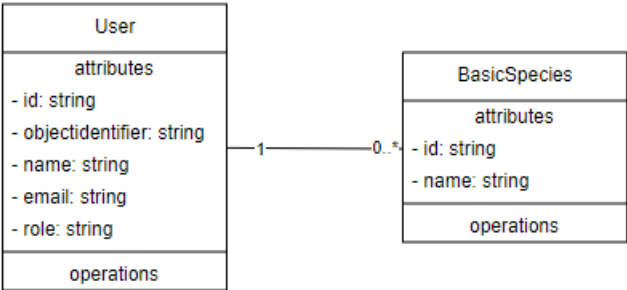
Hình 25 Sequence Diagram Find species by name

4. Thiết kế dữ liệu



Hình 26 Species Class Diagram

Tất cả bảng sẽ được gom lại thành một document. Vì để tối ưu hóa bộ nhớ khi nên lưu thông tin tác giả sẽ sử dụng BasicUser sẽ lưu 2 trường là id và name.



Hình 27 User Class Diagram

Lưu thông tin người dùng bên identity. Khác với identity, thông tin người dùng bên organism chỉ lưu các thông tin cần thiết bao gồm id, objectIdentifier, name, email. Id sẽ là trường id tự tạo và objectIdentifier là sẽ id bên identity, không lưu password vì không có sử dụng đăng nhập.

Lưu các loài của người dùng, một người dùng sẽ có một collection loài. Và collection này sẽ sử dụng BasicSpecies để tối ưu hóa bộ nhớ sẽ chỉ lưu id và name.

5. Worker service

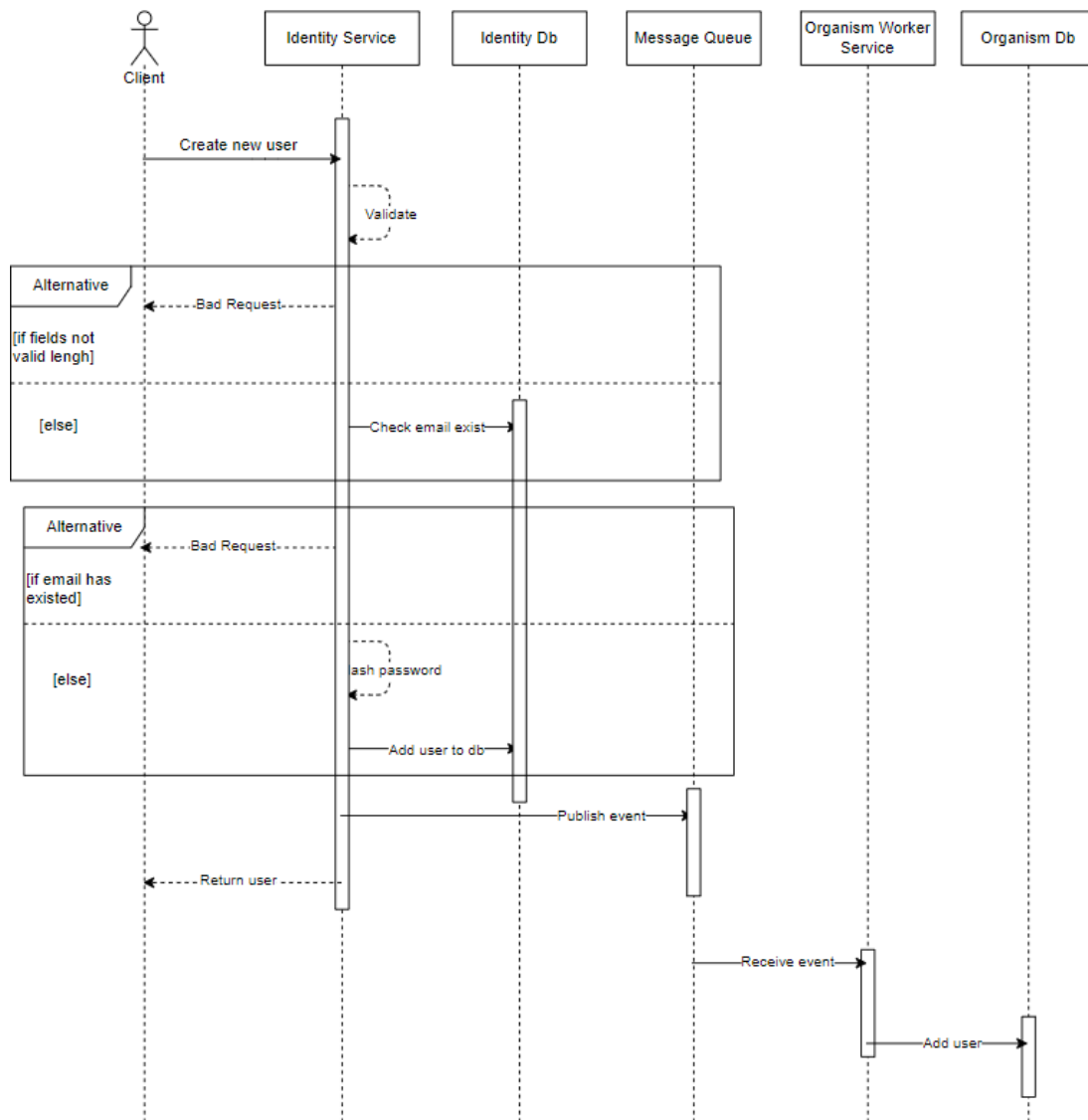
Worker service là loại project mới của ASP .NET Core với chức năng chính là thực hiện các tác vụ chạy background. Trong Organism nhiệm vụ chính của Organism Worker service là lắng nghe các tin nhắn đến từ RabbitMq. Để bảo toàn cho tính toàn vẹn của dữ liệu trong Microservices, worker sẽ lắng nghe mỗi khi dữ liệu bên các service có thay đổi mà làm ảnh hưởng đến Organsim Service.

5.1. Lắng nghe thông báo từ service khác

5.1.1 Trường hợp tốt nhất

Organism service phụ thuộc vào identity service, organism sẽ sử dụng token identity để xác thực người dùng. Khi có một thay đổi dữ liệu bên identity thì identity sẽ gửi tin nhắn đến RabbitMq để thông báo, worker sẽ có nhiệm vụ xử lý thông báo này. Ví dụ về việc tạo người dùng mới bên identity được thể hiện trong sơ đồ tuần tự dưới đây:

Create new User event



Hình 28 Sequence Diagram Create new User event

Sơ đồ tuần tự khác với sơ đồ tuần tự bên identity vì có thêm sự góp mặt của Organism Worker Service và Organism Db. Worker sẽ lấy event từ RabbitMq xuống và thêm người dùng mới vào CSDL Organism Db.

5.1.2 Trường hợp tệ nhất

Trường hợp dữ liệu không thống nhất. Theo sơ đồ bên trên Organism sẽ phải phụ thuộc vào RabbitMq để nhận thông báo khi có người dùng mới được thêm vào identity. Có 2 trường hợp cần phải được cân nhắc đến trong việc này:

1. RabbitMq không hoạt động
2. Identity gặp lỗi khi thông báo

Các trường hợp trên sẽ khiến cho dữ liệu chỉ tồn tại bên CSDL của identity mà không tồn tại trong Organism-Db và lỗi sẽ xảy ra khi người dùng sử dụng token để xác thực đăng nhập mà người dùng không tồn tại trong CSDL.

Cách khắc phục tình trạng này, khi người dùng sử dụng token để đăng nhập, nhưng thông tin của người dùng trong token lại không tồn tại trong Organism-Db i. Thay vì thông báo lỗi vì không tồn tại người dùng trong CSDL, worker service sẽ tạo một lời gọi api đến identity service để kiểm tra xem người dùng có tồn tại bên identity hay không. Khi lấy dữ liệu từ lời gọi api mà có tồn tại người dùng thì worker sẽ tự động thêm người dùng đó vào Organism-Db còn không sẽ thông báo lỗi.

5.2. Xử lý bất đồng bộ

5.2.1 Async trong C#

Trong C# ta có thể tránh performance bottlenecks và nâng cao khả năng đáp ứng tổng thể của ứng dụng bằng cách sử dụng lập trình không đồng bộ. Cách viết ứng dụng không đồng bộ cũ thì rất phức tạp và khiến cho việc gỡ lỗi và bảo trì trở nên khó khăn. C # 5 đã giới thiệu một cách tiếp cận đơn giản, lập trình không đồng bộ, thúc đẩy hỗ trợ không đồng bộ trong .NET Framework 4.5 trở lên, .NET Core và Windows Runtime. Trình biên dịch thực hiện công việc khó khăn mà nhà phát triển đã từng làm và ứng dụng của bạn vẫn giữ cấu trúc logic như lập trình tuần tự. Và kết quả là, bạn nhận được tất cả các lợi thế của lập trình không đồng bộ với một phần nhỏ nỗ lực. Async cải thiện khả năng phản hồi Không đồng bộ là điều cần thiết cho các hoạt động có khả năng bị chặn, chẳng hạn như truy cập web. Quyền truy cập vào tài nguyên web đôi khi bị chậm hoặc bị trì hoãn. Nếu một hoạt động như vậy bị chặn trong một quy trình đồng bộ, thì toàn bộ ứng dụng phải đợi. Trong quy trình không đồng bộ, ứng dụng có thể tiếp tục với công việc khác không phụ thuộc vào tài nguyên web cho đến khi tác vụ chặn tiềm năng kết thúc.

5.2.2 Bất cập của async trong C#

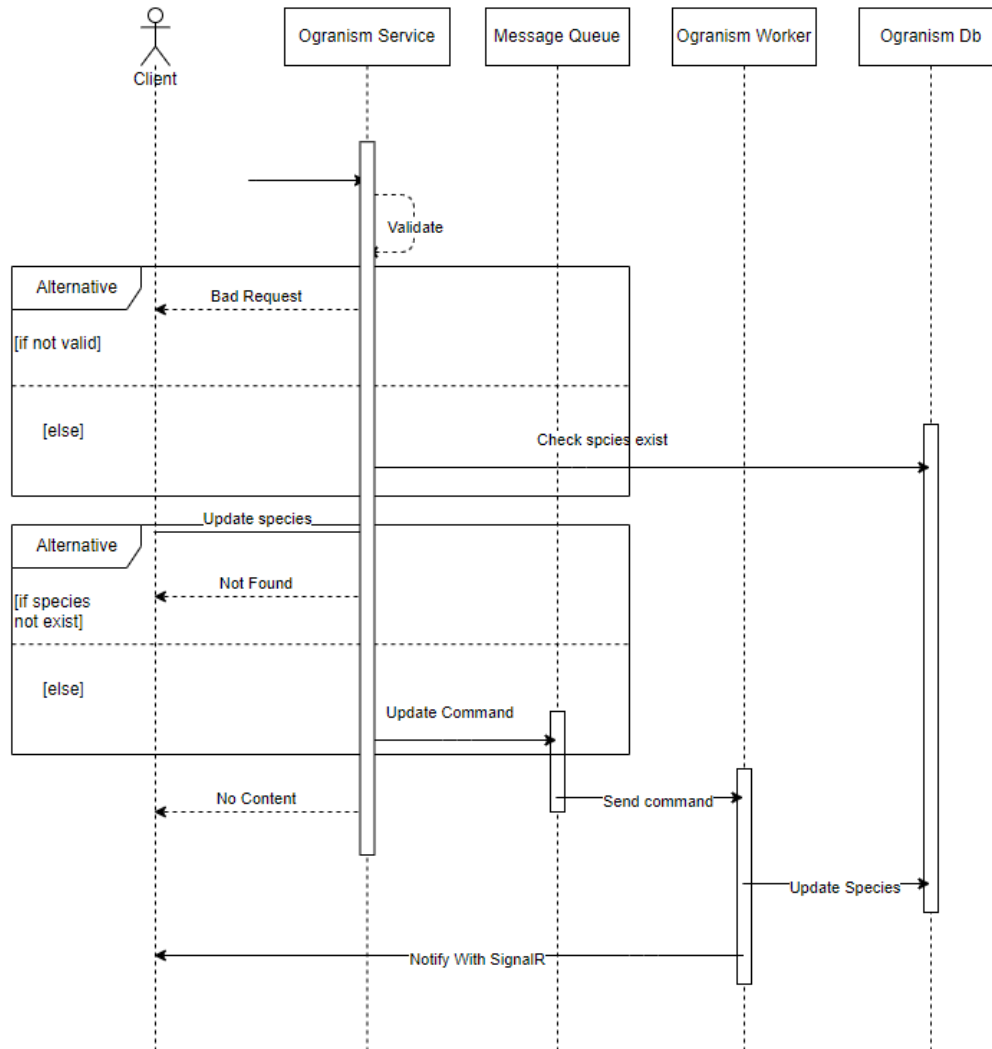
Trong C# có lập trình bất đồng bộ. Tuy nhiên bất đồng bộ này chỉ xảy ra trong nội bộ C#. Client giao tiếp với Organism Service sử dụng các api endpoints và dưới góc nhìn toàn cảnh và giao tiếp các api sử dụng chuẩn http. Trong http hoạt động dưới các cơ chế Request và Response và như thế http lại là tuần tự bởi vì dưới góc nhìn của client sử dụng các api này thì khi gửi Request phải luôn chờ cho Response trở về mới thể tiếp tục và thế http trở thành tuần tự.

5.2.3 Async trong Organism Service

Vậy Organism đảm nhận việc sử dụng bất tuần tự như thế nào?

Nhờ vào sự góp mặt của message queue mà ở đây chính là RabbitMq. RabbitMq ở đây sẽ hoạt động như một hàng đợi. Khi muốn thực thi các thao tác chỉnh sửa thông tin bên Organism, theo cách thông thường trong lúc nhận Request từ client ta sẽ trực tiếp chỉnh sửa thông tin trong Organism-Db và trả về thông báo thành công hay thất bại. Và để làm Request theo cách async, Organism sẽ tạo một tin nhắn sửa thông tin và gửi đến RabbitMq, tiếp theo sẽ trả về 204 http Response code, có ý nghĩa thông báo rằng server đã nhận được và đang xử lý thông tin đó. Ví dụ được miêu tả trong sơ đồ tuần tự dưới đây trong việc cập nhật thông tin loài:

Update species



Hình 29 Sequence Diagram Update species

Cập nhật thông tin loài khi người dùng gọi api thì sẽ trả về 204 khi hoàn tất việc gửi tin nhắn đến RabbitMq. Việc xử lý yêu cầu của người dùng sẽ được thực hiện khi mà worker lấy tin nhắn từ RabbitMq, thông báo thành công hay thất bại thông qua SignalR.

6. Giao tiếp ngoài

Organism Service phụ thuộc vào Organism-Db vì là cơ sở dữ liệu của service ngoài ra còn có RabbitMq nơi mà organism sẽ nhận thông báo nếu có dữ liệu thay đổi trong identity cũng như các công việc chạy async. Các log của identity cũng sẽ được gửi đến Seq service để lưu trữ.

6.1. OrganismDb

OrganismDb là một cơ sở dữ liệu phi quan hệ của organism service. Organism service phụ thuộc vào service này và sẽ không thể hoạt động nếu Organism-Db không hoạt động.

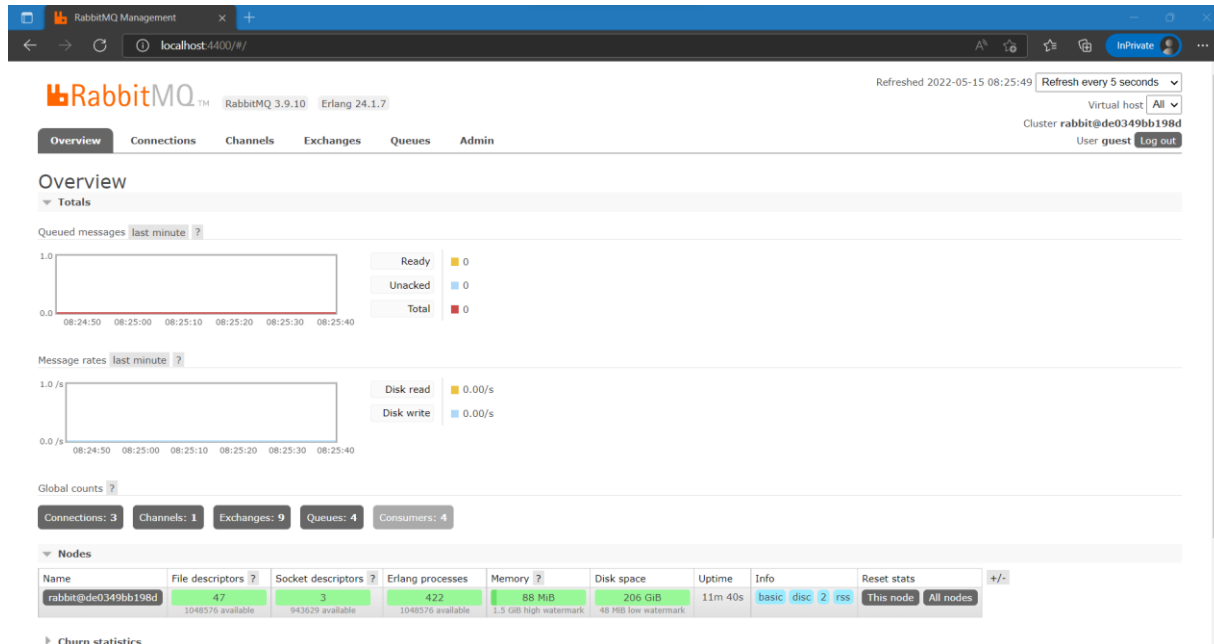
6.2. RabbitMq

RabbitMq service là nơi mà organism sẽ nhận thông báo nếu các dữ liệu bên identity thay đổi. Điều này sẽ giúp bảo vệ tính toàn vẹn của dữ liệu trong Microservices.

Vì chỉ dùng để nhận thông báo nên khi service sẽ không hoạt động nếu RabbitMq không hoạt động.

Chương 5. Các service hỗ trợ

1. RabbitMQ



Hình 30 RabbitMQ UI

RabbitMQ service là một message queue. RabbitMq sẽ đóng vai trò là một service trung gian với nhiệm vụ vận chuyển các tin nhắn giữa các service với nhau.

1.1. Giao tiếp giữa các service

Nếu muốn truy cập dữ liệu của một service khác thì có nhiều cách khác nhau:

1.1.1 Truy cập trực tiếp vào CSDL của service

Ví dụ Organism truy cập trực tiếp vào quan hệ của Identity. Cách làm này là cách tệ nhất và không khuyến khích, việc này không chỉ phá vỡ cấu trúc của Microservices khi định nghĩa rằng mỗi Microservices có CSDL riêng. Việc truy cập CSDL của service khác sẽ làm cho kiến trúc hệ thống từ Microservices mà trở thành Monolithic. Ngoài ra việc này còn làm phá vỡ các ràng buộc bên trong service.

1.1.2 Truy cập thông qua web api

Lấy dữ liệu thông qua web api là cách làm thông dụng nhất lấy dữ liệu từ các service khác và vẫn bảo toàn tính bao đóng của service khác. Tuy nhiên có một số hạn chế là các lời gọi http là lời gọi tuần tự và việc tạo lời gọi http sẽ gây stress cho service.

1.1.3 Truy cập thông qua message queue

Lấy dữ liệu không tuần tự và giảm lời gọi http đến service khác. Sử dụng cách này áp dụng một tính chất giống với Observer Pattern trong design pattern GoF được gọi là Hollywood Principle – Don't Call Us, We'll Call You!

1.2. Event

Trong các tin nhắn được gửi đến RabbitMq, event miêu tả sự việc đã diễn ra trong service. Ví dụ khi người dùng đổi tên bên identity service thì identity sẽ publish một event là NameChanged để thông báo rằng tên đã thay đổi bên identity. Cách đặt tên event theo thì quá khứ trong tiếng anh theo ví dụ đó chính là chữ Changed. Ví dụ mẫu dữ liệu về NameChanged event:

```
{  
  "Id": "891d4dd7-82e7-4818-b78d-62871c0e217d",  
  "Old Name": "Wang thanh",  
  "New Name": "Wang Thanh",  
  "OccurredAt": "2022-05-15T01:28:39.3203084Z",  
  "Version": 1  
}
```

Thông tin event: người dùng có id là "891d4dd7-82e7-4818-b78d-62871c0e217d" đã đổi tên từ "Wang thanh" thành "Wang Thanh". OccurredAt miêu tả thời gian xảy ra còn Version miêu tả phiên bản của tin nhắn vì tin nhắn sẽ có thể thay đổi trong tương lai.

1.3. Command

Khác với event, command sẽ diễn tả hành động sẽ được thực hiện khi subscriber nhận được command. Ví dụ khi người dùng cập nhật thông tin loài thì organism sẽ publish command là UpdateSpeciesName để tạo một hành động rằng hãy cập nhật loài này. Cách đặt tên command

theo thì hiện tại trong tiếng anh theo ví dụ đó chính là chữ Update. Ví dụ mẫu dữ liệu về UpdateSpeciesName command:

```
{  
  "Id": "625579c4d7793c3b2a324983",  
  "Name": "Thần lằn",  
  "OccurredAt": "2022-05-15T01:28:39.3203084Z",  
  "Version": 1  
}
```

Thông tin command: loài có id là "625579c4d7793c3b2a324983" đổi tên thành "Thần lằn". OccurredAt miêu tả thời gian xảy ra còn Version miêu tả phiên bản của tin nhắn vì tin nhắn sẽ có thể thay đổi trong tương lai.

1.4. Sự khác giữa event và command

Command và event cả đều là một dạng tin nhắn được gửi đến RabbitMq, tuy nhiên lại có một số điểm không giống nhau để phân biệt. Trong cách đặt tên, event miêu tả sự việc đã diễn ra nên sẽ là thì quá khứ, command mô tả công việc cần thực hiện nên xài thì hiện tại. Subscriber của command và event cũng khác nhau, ta sẽ có nhiều người nhận event vì đó là thông báo nhưng nên chỉ duy nhất 1 người nhận command, nếu nhiều người nhận command và chỉnh sửa CSDL sẽ gây ra đụng độ.

2. HealthChecks

2.1. Tổng quan service

Trong kiến trúc Microservices sẽ có rất nhiều service để quản lý và việc kiểm soát từng service thông qua console. Khi một service hoạt động không bình thường, ví dụ ở đây chính là một Docker container, thì muốn kiểm tra service hoạt động ra sao em phải vào container và xem console in ra màn hình có lỗi gì hay không. Việc này rất mất thời gian và không kiểm tra được nhiều service cùng lúc. Thư viện .NET đã có giải pháp cho việc này, đó chính là healthchecks. Healthchecks service sẽ đóng vai trò là service có vai trò quan sát các service khác. Sau một giây cố định thì healthchecks service sẽ tạo lời gọi http đến service đã đăng ký, service đó sẽ có 1 endpoint trả về trạng thái.

2.2. Giao tiếp ngoài

Healthchecks Service phụ thuộc vào Healthchecks-Db vì là cơ sở dữ liệu của service ngoài ra còn có. Các log của healthchecks cũng sẽ được gửi đến Seq service để lưu trữ.

2.3. Healthchecks Db

Healthchecks Mssql là một cơ sở dữ liệu SQL Server, đây sẽ là cơ sở dữ liệu của healthchecks service. Healthchecks sẽ sử dụng Entity Framework Core để truy xuất dữ liệu. Healthchecks service phụ thuộc vào service này và sẽ không thể hoạt động nếu healthchecks-db không hoạt động.

3. Seq

Seq là nơi tập trung tất cả các log của các service. Khi một service hoạt động sẽ sinh ra rất nhiều log và với microservices thì việc quản lý các log của nhiều service cùng một lúc sẽ rất khó khăn. Việc log có tầm quan trọng nhất định trong vòng đời của phần mềm, em in ra màn hình khi phần mềm chạy không đúng để kiểm tra lỗi, việc in ra màn hình các dòng thông tin khi phần mềm đang hoạt động. Các service đều có log ra file cũng như log ra màn hình console, tuy nhiên trong Microservices hay Docker container thì nếu một service bị sập thì sẽ được thay thế bởi service mới và tất cả các log sẽ bị mất. Do đó việc gửi log đến một service khác là điều cần thiết. Trong một số trường hợp, dữ liệu log có thể lấy dùng để train các machine learning model.

Chương 6. Image Search

1. Tổng quan service

Image search service là một api service. Xây dựng trên Flask framework, service cung cấp api để tìm hình ảnh tương tự có trong service.

2. Thư viện tìm hình ảnh trong Python

DeepImageSearch là thư viện được sử dụng trong service này để hỗ trợ trong việc tìm kiếm loài giống trong hình: <https://github.com/TechyNilesh/DeepImageSearch>

DeepImageSearch là thư viện bao gồm 2 thư viện khác đó chính là TensorFlow và Annoy:

TensorFlow là một Deep Learning Framework trong Python được tạo bởi Google, DeepImageSearch sẽ load kiến trúc Neural Network gọi là vgg16 và load thêm pre-trained weight để nhận dạng hình ảnh từ ImageNet: <https://www.image-net.org/>.

Sau khi load xong các model sau đó các hình ảnh trong thư mục images của service sẽ được thêm vào model. Model sẽ vectorizes các hình ảnh và chuyển nó sang dữ liệu số và sau đó dữ liệu sẽ được chuyển sang thư viện Annoy.

Annoy là thư viện của Spotify: <https://github.com/spotify/annoy>. Nó rất hiệu quả trong việc clustering và tìm ra các pattern trong các thành phần của dữ liệu, ở đây với tập dữ liệu này sẽ là các dữ liệu số biểu diễn hình ảnh. Thuật toán của Annoy sẽ là duyệt qua tất cả các hình ảnh và tìm ra các pattern, các pattern sẽ biểu thị ra các điểm tương đồng.

Vì DeepImageSearch chỉ hoạt động trên các thư mục nên giải pháp ở đây khi muốn hoạt động qua API sẽ là: client sẽ gửi hình ảnh dưới dạng base64 sau đó Image Search sẽ lưu hình vào một thư mục temp, rồi thực hiện tìm kiếm các hình ảnh tương đồng trong thư viện hình ảnh từ thư mục temp đó.

3. Tìm loài theo ảnh REST API

Tên yêu cầu	Tìm loài theo ảnh
Mục đích	Chức năng này tìm hình tương tự có trong hệ thống.
URI	/search

Param	Không
Method	POST
Header	Không
Body	<pre>{ "size": 10, "img": "data:image/.jpg;base64, iVBORw0KGgoAAAANSUheEUgAAAAUAAAFCAyAAACNbyblA AAAEIEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4O HwAAAABJRU5ErkJggg==" }</pre>
Response	<pre>200 – Ok { "result": ["625579c5d7793c3b2a324984/625579c5d7793c3b2a324984_5.jpg", "625579c1d7793c3b2a324981/625579c1d7793c3b2a324981_5.jpg", "625579c3d7793c3b2a324982/625579c3d7793c3b2a324982_5.jpg", "625579bed7793c3b2a32497d/625579bed7793c3b2a32497d_3.jpg", "625579bed7793c3b2a32497e/625579bed7793c3b2a32497e_1.jpg", "625579bbd7793c3b2a324979/625579bbd7793c3b2a324979_3.jpg", "625579bcd7793c3b2a32497b/625579bcd7793c3b2a32497b_4.jpg", "625579c9d7793c3b2a324989/625579c9d7793c3b2a324989_1.jpg", "625579bbd7793c3b2a324979/625579bbd7793c3b2a324979_4.jpg", "625579bcd7793c3b2a32497b/625579bcd7793c3b2a32497b_3.jpg"] }</pre>

4. Lấy hình bên Organism

Vì DeepImageSearch tìm hình trong thư mục trong service nội bộ nên phải lấy hình bên Organism về để lưu nội bộ. Với kiến trúc Microservices và message queue RabbitMq thì giải pháp trong công việc này sẽ là: Image search service sẽ kết nối đến RabbitMq để nhận thông báo khi mà Organism có bổ sung về hình ảnh các loài vật. Khi Image Search service nhận được thông báo, service sẽ tạo lời gọi http đến Organism để kéo hình mới về. Vì chỉ quan tâm đến hình ảnh bên Organism nên Image Search service sẽ chỉ lắng nghe các tin nhắn về thêm loài và sửa loài.

Chương 7. Kiểm thử và đánh giá kết quả

1. Giới thiệu

1.1. Mục tiêu

Tạo các kiểm thử giúp em theo dõi và đánh giá hành vi của sản phẩm, phần mềm hoàn chỉnh và đã được tích hợp đầy đủ, dựa vào đặc tả và các yêu cầu chức năng đã được xác định trước. Kiểm tra các lỗi hỏng hệ thống có thể gây ra trong lúc hoạt động, giảm thiểu các lỗi có thể xảy ra đảm bảo các luồng dữ liệu được xác thực một cách chặt chẽ, nâng cao chất lượng phần mềm. Phát hiện lỗi và kiểm tra hệ thống có hoạt động đúng theo yêu cầu đã nêu ra trong đặc tả hay chưa. Liệt kê kết quả có được sau kiểm thử. Làm tài liệu cho giai đoạn bảo trì.

1.2. Phạm vi kiểm thử

Quy trình kiểm thử được thực hiện qua những công đoạn như sau:

- Kiểm thử thiết kế: kiểm tra giao diện thiết kế có đúng với đặc tả.
- Kiểm thử chấp nhận: kiểm thử chức năng hệ thống có hoạt động và đáp ứng đặc tả yêu cầu.
- Kiểm thử chức năng: kiểm thử chức năng có xử lý đúng dữ liệu. Kiểm thử cài đặt: tìm và sửa lỗi xảy ra kiểm thử.

2. Kế hoạch kiểm thử

2.1. Các tính năng sẽ được kiểm thử

1. Đăng nhập
2. Hiện danh sách loài
3. Hiện chi tiết loài
4. Tìm kiếm loài
5. Lọc loài
6. Thêm loài
7. Sửa loài
8. Xóa loài
9. Thêm quản trị
10. Sửa quản trị
11. Tìm kiếm quản trị
12. Xóa quản trị
13. Xem thống kê

2.2. Các tính năng không được kiểm thử

1. Đăng xuất
2. Đổi mật khẩu
3. Đổi tên
4. Đổi màu
5. Đổi ngôn ngữ
6. Sắp xếp theo tên
7. Gom nhóm theo tên

2.3. Chiến lược kiểm thử

- Chiến lược: thực hiện Black Box Test.
- Sử dụng kiểm thử tích hợp: Functional test.
- Kiểm thử hệ thống (System test): Kiểm tra môi trường sử dụng phần mềm đòi hỏi.

2.4. Kiểm thử L&P

- Performance profiling là một dạng test hiệu suất trong đó thời gian phản hồi, tỷ lệ giao dịch và các yêu cầu phụ thuộc thời gian khác được đo đạc và đánh giá.

- Mục đích của Performance Profiling là kiểm tra các yêu cầu về hiệu suất có đạt được hay không
- Performance profiling là tiến hành và thực hiện để mô tả sơ lược và điều chỉnh các hành vi hiệu suất của mục tiêu test như một hàm của các điều kiện ví dụ workload hoặc cấu hình phần cứng.

2.5. Kiểm thử hồi quy

- Kiểm thử hồi quy (Regression Testing) là một hoạt động cần thiết để chỉ ra rằng việc thay đổi code không gây ra những ảnh hưởng bất lợi.
- **Mục đích kiểm thử:**
 - o Kiểm thử hồi quy dùng để kiểm tra các phần được sửa chữa trong phần mềm, để đảm bảo rằng những sự thay đổi đó không gây ra lỗi trong những phần khác.
- **Cách thực hiện:**
 - o Tái sử dụng các TC từ những phần kiểm thử trước để kiểm thử các module đã được sửa chữa
 - o Sử dụng công cụ Rational Robot: Tạo một số script kiểm thử về chức năng. Định nghĩa lịch thực hiện tự động cho chúng
 - o 80% các TC được chọn ngẫu nhiên
 - o Xây dựng một chương trình phân tích cơ sở hạ tầng. Em dựng một cơ sở hạ tầng có thể mở rộng được để thực hiện và đánh giá chương trình phân tích. Dựa vào kết quả phân tích em xác định phạm vi cần test hồi quy.
- **Điều kiện hoàn thành:**
 - o Toàn bộ các TC được thực hiện và đạt yêu cầu
 - o Toàn bộ các TC được chọn được thực hiện và đạt yêu cầu

2.6. Cách tiếp cận

Với mỗi tính năng chính hay các nhóm tính năng sẽ được kiểm thử theo thứ tự từ trên xuống dưới và từ trái qua phải để đảm bảo rằng sẽ kiểm thử không bỏ sót chức năng cần kiểm thử.

2.7. Tiêu chí kiểm thử thành công / thất bại

- Tiêu chí kiểm thử thành công: kết quả cuối cùng của chuỗi các thao tác đúng như mong đợi đặt ra ban đầu.

- Tiêu chí kiểm thử thất bại: kết quả kiểm thử không như mong đợi, xuất hiện lỗi sai lệch so với kỳ vọng ban đầu.

2.8. Tiêu chí đình chỉ và yêu cầu bắt đầu lại

- Tiêu chí đình chỉ là dừng việc thực hiện công việc khi một chức năng thông báo lỗi.
- Yêu cầu bắt đầu lại khi chức năng đình chỉ đã được sửa lỗi.

3. Các trường hợp kiểm thử

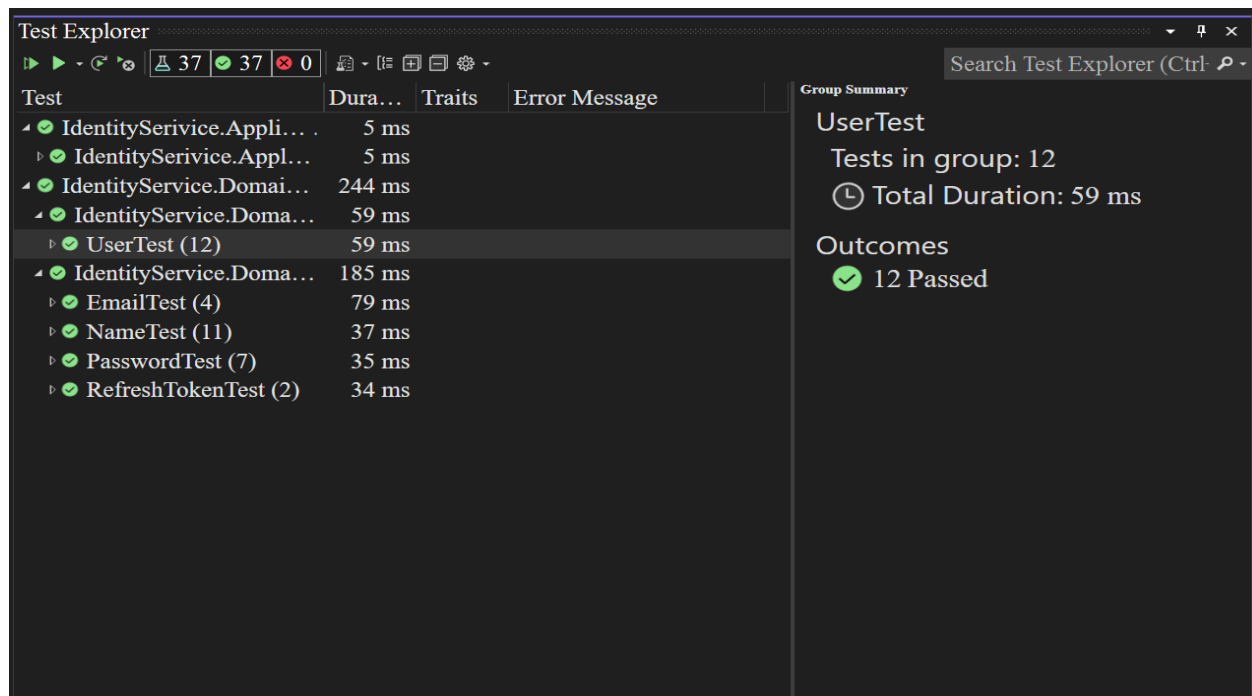
3.1. Kịch bản kiểm thử

Mã	Mô tả kịch bản kiểm thử	Số trường hợp kiểm thử
TS_01	Kiểm tra người dùng có thể đăng nhập	6
TS_02	Kiểm tra hiển thị tìm loài	7
TS_03	Kiểm tra thêm sửa loài	5
TS_04	Kiểm tra quản lý quản trị viên	6

Bảng 9 Kịch bản kiểm thử

3.2. Kiểm thử tự động

Service sẽ có Unit test kiểm thử các hàm. Các unit tests giúp ích rất nhiều trong việc phát triển phần mềm, vì unit test tự động nên mỗi lần chạy test sẽ phải nhanh, và mỗi lần thay đổi code sẽ chạy lại test để kiểm tra xem có gì thay đổi hay không.



Ví dụ dưới đây là một unit test về kiểm tra hợp lệ của email trong hệ thống sử dụng Xunit.

```
{
    ....[Fact]
    ....public void ValueIsEmpty()
    ....{
    ....    Assert.Throws<ArgumentException>(() => new Email(string.Empty));
    ....}

    ....[Fact]
    ....public void ValueIsNotValidEmail()
    ....{
    ....    Assert.Throws<ArgumentException>(() => new Email("test@test"));
    ....}

    ....[Fact]
    ....public void ValueIsValidEmail()
    ....{
    ....    string email = "alex@gmail.com";

    ....    var emailValue = new Email(email);

    ....    Assert.Equal(email, emailValue.Value);
    ....}
}
```

Hình 32 Email Unit Test

3.3. Kiểm thử thủ công

Các kiểm thử dưới đây sẽ được thực hiện bằng tay trên các trình duyệt.

Title	Steps/Actions	Expected Result	Actual Result	Test Data
Đăng nhập với tài khoản và mật khẩu đúng	1. Mở trang đăng nhập quản trị	Đăng nhập thành công, chuyển tới giao diện quản trị	Đăng nhập thành công, chuyển tới giao diện quản trị	Tài khoản và mật khẩu hợp lệ là bắt buộc, được tạo bởi quản trị viên.
	2. Nhập tài khoản hợp lệ			
	3. Nhập mật khẩu hợp lệ			
	4. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
Đăng nhập với mật khẩu sai	1. Mở trang đăng nhập quản trị	Hệ thống thông báo "Thông tin đăng nhập không đúng!"	Hệ thống thông báo "Thông tin đăng nhập không đúng!"	Tài khoản hợp lệ và mật khẩu sai là bắt buộc.
	2. Nhập tài khoản hợp lệ			
	3. Nhập mật khẩu không đúng			
	4. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
Đăng nhập với tài khoản sai	1. Mở trang đăng nhập quản trị	Hệ thống thông báo "Thông tin đăng nhập không đúng!"	Hệ thống thông báo "Thông tin đăng nhập không đúng!"	Tài khoản sai và mật khẩu hợp lệ là bắt buộc.
	2. Nhập tên người dùng tài khoản không tồn tại			
	3. Nhập mật khẩu			
	4. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
Đăng nhập với tài khoản trống	1. Mở trang đăng nhập quản trị	Hệ thống thông báo "Bạn chưa nhập tài khoản!"	Hệ thống thông báo "Bạn chưa nhập tài khoản!"	Tài khoản trống là bắt buộc.
	2. Không nhập tài khoản			
	3. Nhập mật khẩu			
	4. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
	1. Mở trang đăng nhập quản trị	Hệ thống thông báo	Hệ thống thông báo	Mật khẩu trống là bắt buộc.

Đăng nhập với mật khẩu trống	2. Nhập tài khoản hợp lệ	"Bạn chưa nhập mật khẩu!"	"Bạn chưa nhập mật khẩu!"	
	3. Không nhập mật khẩu			
	4. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
Đăng nhập với tài khoản và mật khẩu trống	1. Mở trang đăng nhập quản trị	Hệ thống thông báo	Hệ thống thông báo	Tài khoản và mật khẩu trống là bắt buộc.
	2. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter	"Bạn chưa nhập tài khoản!"	"Bạn chưa nhập tài khoản!"	
Hiển thị danh sách loài	1. Mở trang trang chủ của website	Danh sách loài được hiển thị trong trang chủ	Danh sách loài được hiển thị trong trang chủ	
Hiển thị thông tin loài	1. Mở trang trang chủ của website	Thông tin loài được hiển thị trong trang loài	Thông tin loài được hiển thị trong trang loài	
	2. Chọn một loài bất kì để xem thông tin loài			
Hiển thị chi tiết loài	1. Mở trang trang chủ của website	Thông tin chi tiết loài được hiển thị	Thông tin chi tiết loài được hiển thị	
	2. Chọn một loài bất kì để xem thông tin loài			
	3. Nhấp vào biểu tượng chi tiết loài			
Tìm kiếm loài	1. Mở trang trang chủ của website	Thông tin loài được hiển thị trong trang loài	Thông tin loài được hiển thị trong trang loài	Nhập tên loài tìm kiếm là bắt buộc.
	2. Nhập tên loài cần tìm vào ô tìm kiếm			
	3. Nhấp vào biểu tượng tìm kiếm hoặc nhấn Enter			
Lọc loài	1. Mở trang trang chủ của website	Danh sách loài được hiển thị tương ứng với bộ lọc được chọn	Danh sách loài được hiển thị tương ứng	Chọn bộ lọc loài là bắt buộc.
	2. Chọn bộ lọc loài cần lọc tương ứng			

			với bộ lọc được chọn	
Gợi ý loài	1. Mở trang trang chủ của website	Danh sách loài tương tự sẽ được hiển thị trong phần "Loài gợi ý"	Danh sách loài tương tự sẽ được hiển thị trong phần "Loài gợi ý"	
	2. Chọn loài cần xem			
Thêm loài với đầy đủ thông tin loài	1. Mở trang đăng nhập quản trị	Hệ thống thông báo "Đã lưu thông tin loài"	Hệ thống thông báo "Đã lưu thông tin loài"	Điền tất cả các trường thông tin là bắt buộc.
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
	4. Chọn thẻ "Quản lý loài"			
	5. Chọn thêm loài (+)			
	6. Nhập thông tin loài			
	7. Lưu loài			
Thêm loài với một hoặc nhiều trường thông tin loài bị bỏ trống	1. Mở trang đăng nhập quản trị	Hệ thống thông báo "Vui lòng điền đủ thông tin bắt buộc!"	Hệ thống thông báo "Vui lòng điền đủ thông tin bắt buộc!"	Một hoặc nhiều trường thông tin bị bỏ trống là bắt buộc.
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
	4. Chọn thẻ "Quản lý loài"			
	5. Chọn thêm loài (+)			
	6. Nhập thông tin loài			
	7. Lưu loài			
Chỉnh sửa thông tin loài	1. Mở trang đăng nhập quản trị	Hệ thống thông báo "Đã lưu thông tin loài"	Hệ thống thông báo "Đã lưu thông tin loài"	Chỉnh sửa thông tin loài là bắt buộc.
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			

	<div>4. Chọn thẻ "Quản lý loài"</div> <div>5. Chọn biểu tượng chỉnh sửa loài tương ứng</div> <div>6. Nhập thông tin cần chỉnh sửa</div> <div>7. Lưu loài</div>			
Xóa loài	<div>1. Mở trang đăng nhập quản trị</div> <div>2. Nhập tài khoản và mật khẩu hợp lệ</div> <div>3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter</div> <div>4. Chọn thẻ "Quản lý loài"</div> <div>5. Chọn biểu tượng xóa loài tương ứng</div> <div>6. Xác nhận xóa loài</div>	Hệ thống xóa loài được chọn	Hệ thống xóa loài được chọn	Chọn loài cần xóa là bắt buộc.
Tìm kiếm loài	<div>1. Mở trang đăng nhập quản trị</div> <div>2. Nhập tài khoản và mật khẩu hợp lệ</div> <div>3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter</div> <div>4. Chọn thẻ "Quản lý loài"</div> <div>5. Nhập tên hệ điều hành xuất xứ của loài cần tìm vào ô tìm kiếm</div>	Hiển thị danh sách loài có tên hệ điều hành xuất xứ trùng với thông tin cần tìm	Hiển thị danh sách loài có tên hệ điều hành xuất xứ trùng với thông tin cần tìm	Nhập thông tin vào ô tìm kiếm là bắt buộc
Thêm quản trị viên với đầy đủ thông tin quản trị viên	<div>1. Mở trang đăng nhập quản trị</div> <div>2. Nhập tài khoản và mật khẩu hợp lệ</div>	Hệ thống thông báo "Đã tạo quản trị viên"	Hệ thống thông báo "Đã tạo quản trị viên"	Điền tất cả các trường thông tin là bắt buộc.

	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter 4. Chọn thẻ "Quản lý quản trị viên" 5. Chọn thêm quản trị viên (+) 6. Nhập thông tin quản trị viên 7. Lưu quản trị viên			
Thêm quản trị viên với thông tin bị thiếu	1. Mở trang đăng nhập quản trị 2. Nhập tài khoản và mật khẩu hợp lệ 3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter 4. Chọn thẻ "Quản lý quản trị viên" 5. Chọn thêm quản trị viên (+) 6. Nhập thông tin quản trị viên 7. Lưu quản trị viên	Hệ thống thông báo "Vui lòng điền đầy đủ thông tin quản trị viên"	Hệ thống thông báo "Vui lòng điền đầy đủ thông tin quản trị viên"	Một hoặc nhiều trường thông tin bị bỏ trống là bắt buộc.
Chỉnh sửa thông tin quản trị viên	1. Mở trang đăng nhập quản trị 2. Nhập tài khoản và mật khẩu hợp lệ 3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter 4. Chọn thẻ "Quản lý quản trị viên" 5. Chọn biểu tượng chỉnh sửa quản trị viên tương ứng 6. Nhập thông tin cần chỉnh sửa 7. Lưu quản trị viên	Hệ thống thông báo "Đã lưu thông tin quản trị viên"	Hệ thống thông báo "Đã lưu thông tin quản trị viên"	Chỉnh sửa thông tin quản trị viên là bắt buộc.

Xóa quản trị viên	1. Mở trang đăng nhập quản trị	Hệ thống xóa quản trị viên được chọn	Hệ thống xóa quản trị viên được chọn	Chọn quản trị viên cần xóa là bắt buộc.
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
	4. Chọn thẻ "Quản lý loài"			
	5. Chọn biểu tượng xóa quản trị viên tương ứng			
	6. Xác nhận xóa loài			
Tìm kiếm quản trị viên	1. Mở trang đăng nhập quản trị	Hiển thị danh sách quản trị viên có tên trùng với thông tin cần tìm	Hiển thị danh sách quản trị viên có tên trùng với thông tin cần tìm	Nhập thông tin vào ô tìm kiếm là bắt buộc.
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
	4. Chọn thẻ "Quản lý loài"			
	5. Nhập tên quản trị viên cần tìm vào ô tìm kiếm			
Xem thống kê	1. Mở trang đăng nhập quản trị	Hiển thị các thống kê của website	Hiển thị các thống kê của website	
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			
	4. Chọn thẻ "Thống kê"			
	2. Nhập tài khoản và mật khẩu hợp lệ			
	3. Nhấp chọn nút "Đăng nhập" hoặc nhấn Enter			

	4. Chọn thẻ "Thống kê"			
	5. Chọn biểu tượng xuất báo cáo ra file excel tương ứng			
	6. Chọn địa chỉ lưu file báo cáo			

Bảng 10 Các test case

C. PHẦN KẾT LUẬN

1. Đánh giá Microservices

1.1. Ưu điểm

- Microservices cho phép dễ dàng liên tục chuyển giao và triển khai các ứng dụng lớn và phức tạp hơn.
- Có thể cải thiện khả năng bảo trì: bởi vì các service tương đối nhỏ nên dễ hiểu và dễ thay đổi hơn.
- Có khả năng testing dễ dàng: nhờ các services nhỏ.
- Có thể triển khai tốt hơn: các services thường rất dễ cho việc triển khai độc lập.
- Cho phép các services được phát triển nhanh chóng bởi những team khác nhau. Khi đó, mỗi team đều sẽ được phát triển và thử nghiệm để triển khai cũng như mở rộng được quy mô của dịch vụ của mình một cách độc lập nhất với tất cả các team.
- Nếu như có lỗi xảy ra trong một service thì chỉ có service đó bị ảnh hưởng và các service khác sẽ thực hiện xử lý các yêu cầu cần thiết. Trong khi đó, thì mỗi một thành phần nếu như hoạt động sai của kiến trúc một khối thì nó sẽ làm ảnh hưởng đến toàn bộ hệ thống.
- Lập trình viên có thể thay đổi dễ dàng bằng cách sử dụng công nghệ mới khi triển khai các service. Tương tự như khi có thay đổi lớn thì các service đều có thể thực hiện và bạn dễ dàng thay đổi được công nghệ hơn.

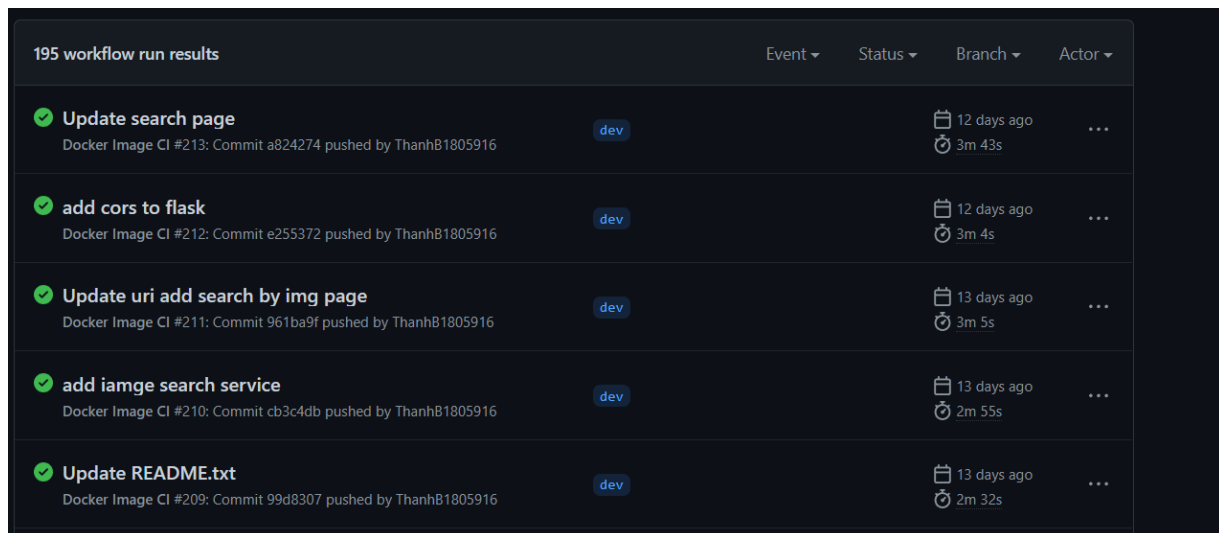
1.2. Nhược điểm

- Bởi vì các nhà phát triển thường xuyên phải đối phó với sự phức tạp khi tạo ra một hệ thống phân tán.
- Cần phải cài đặt việc liên lạc giữa các services nội bộ.
- Handle partial failure rất phức tạp bởi vì luồng xử lý cần phải đi qua nhiều service khác nhau.
- Khi thực hiện các requests trải rộng trên nhiều service khó khăn thì điều này cần đòi hỏi sự phối hợp giữa các team.
- Thường rất khó khăn trong việc đảm bảo toàn vẹn cho CSDL nếu như triển khai theo các cấu trúc cơ sở dữ liệu dạng phân vùng.
- Việc triển khai và quản lý Microservices nếu như làm thủ công theo cách làm với ứng dụng thì sẽ rất phức tạp.
- Lập trình viên cần phải xử lý các sự cố kết nối chậm, lỗi nếu như thông điệp không được gửi hoặc nếu như thông điệp được gửi đến nhiều đích đến vào các thời điểm khác nhau.

2. Kết quả đạt được

2.1. Về lý thuyết và công nghệ

- Khả năng đọc tài liệu, tự tìm hiểu, tự học và giải quyết vấn đề.
- Hiểu biết được các kiến trúc Microservices
- Thiết kế một hệ thống Microservices
- Hiểu biết được quy trình nghiệp vụ trong việc quản lý loài.
- Sử dụng Docker để triển khai phần mềm.
- Nâng cao kỹ năng lập trình và hiểu rõ hơn về .NET framework
 - Thiết kế trang web sử dụng Blazor viết một trang web với giao diện đẹp mắt.
- Sử dụng các công cụ DevOps: Sử dụng Git-GitHub để lưu mã nguồn code, Git-Action chạy CI kiểm tra lỗi mỗi lần gom nhánh lại với nhau.



195 workflow run results		Event ▾	Status ▾	Branch ▾	Actor ▾
✓	Update search page Docker Image CI #213: Commit a824274 pushed by ThanhB1805916	dev	12 days ago 3m 43s	...	
✓	add cors to flask Docker Image CI #212: Commit e255372 pushed by ThanhB1805916	dev	12 days ago 3m 4s	...	
✓	Update uri add search by img page Docker Image CI #211: Commit 961ba9f pushed by ThanhB1805916	dev	13 days ago 3m 5s	...	
✓	add iamge search service Docker Image CI #210: Commit cb3c4db pushed by ThanhB1805916	dev	13 days ago 2m 55s	...	
✓	Update README.txt Docker Image CI #209: Commit 99d8307 pushed by ThanhB1805916	dev	13 days ago 2m 32s	...	

Hình 33 CI/CD trên GitHub

2.2. Hạn chế:

- Chưa có nhiều chức năng nâng cao, chỉ dừng ở chức năng cơ bản
- Giao diện chưa tương thích với màn hình các thiết bị di động
- Còn một số lỗi vẫn chưa được tìm thấy, tỉ lệ bao phủ vẫn chưa tuyệt đối.
- Hệ thống test vẫn chưa trực quan nhất có thể.
- Phần cứng vẫn chưa đáp ứng nhu cầu để chạy test

3. Triển khai

Vì viết trên Docker nên có thể triển khai dễ dàng trên các nền tảng đám mây như Azure, Aws, Google Cloud, ...

Chạy project trên máy cá nhân, yêu cầu phải có Docker.

4. Hướng phát triển

- Tiếp tục hoàn thiện hệ thống.
- Nghiên cứu tích hợp các thiết bị IoT
- Phát triển trên nền tảng di động với MAUI

D. TÀI LIỆU THAM KHẢO

- [1]. .NET Microservice: <https://docs.microsoft.com/en-us/dotnet/architecture/microservices/>
- [2]. Microservices Architecture, <https://Microservices.io/>
- [3]. ADOPTION OF THE MICROSERVICES ARCHITECTURE
by Maryanne Ndungu, https://www.doria.fi/bitstream/handle/10024/169535/ndungu_maryanne.pdf
- [4]. .NET 6 <https://dotnet.microsoft.com/en-us/download/dotnet/6.0>
- [5]. Blazor <https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor>
- [6]. SignalR <https://dotnet.microsoft.com/en-us/apps/aspnet/signalr>
- [7]. .NET healthchecks <https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/health-checks?view=aspnetcore-6.0>
- [8]. RabbitMQ: <https://www.rabbitmq.com/>
- [9]. Seq <https://datalust.co/>
- [10]. Docker: [https://en.wikipedia.org/wiki/Docker_\(software\)](https://en.wikipedia.org/wiki/Docker_(software))
- [11]. Flask [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))
- [12]. Huỳnh Xuân Hiệp, Phan Phương Lan (2011). Giáo trình Nhập môn công nghệ phần mềm, NXB Đại học Cần Thơ.
- [13]. Huỳnh Xuân Hiệp, Phan Phương Lan (2014). Giáo trình Bảo trì phần mềm, NXB Đại học Cần Thơ.
- [14]. Huỳnh Xuân Hiệp, Phan Phương Lan, Võ Huỳnh Trâm (2015). Giáo trình Kiến trúc và Thiết kế phần mềm, NXB Đại học Cần Thơ.
- [15]. Trần Cao Đệ, Đỗ Thanh Nghị (2012). Giáo trình Kiểm thử phần mềm, NXB Đại học Cần Thơ.
- [16]. Trần Cao Đệ, Nguyễn Công Danh (2014). Giáo trình Đảm bảo chất lượng phần mềm, NXB Đại học Cần Thơ.
- [17]. Võ Huỳnh Trâm (2009). Bài giảng Phân tích yêu cầu phần mềm, Khoa Công nghệ thông tin và Truyền thông, Trường Đại học Cần Thơ.